Rectangular Network Reliability Computation Based on Isomorphism Determination

Yufeng Xiao[1,2]
1. Information Engineering School, Southwest University of Science and Technology, Mianyang, China
2. Special Environment Robot Technology Key Laboratory of Sichuan Province, Southwest University of Science and Technology, Mianyang, China
Email: xiaoyf_swit1@163.com

***Abstract*—To decrease repeated computations and improve efficiency, this paper presents a rectangular network reliability computation method based on isomorphism determination. According to the special structure of rectangular network, 2 sorts of subnets are defined for $2\times m$ network, and n sorts of subnets are defined for $n\times m$ network. Furthermore, a hash table is designed to record the OBDD (Ordered Binary Decision Diagram) of defined subnet. When the OBDD of a subnet is constructed, if it is the first time to find this subnet and a defined subnet are isomorphic, its OBDD will be recorded in this hash table. Otherwise, there is no need to construct its OBDD and the recorded OBDD can be accessed in above hash table. Thus, the repeated computations from isomorphic subnets can be reduced, and computation costs are saved. Experiment shows this method can efficiently calculate the reliability of rectangular network with large scale.**

***Index Terms*—network reliability, isomorphic subnet, rectangular network, ordered binary decision diagram**

I. Introduction

As a usual structure, rectangular network has been widely applied in dense deployment of wireless sensor network, backbone network planning and multiprocessor system design [1-4]. Its simple and regular structure is convenient for the network design, and its reliability evaluation will help engineers to efficiently manage network. Network reliability evaluation originated in the 70's of last century, and great achievements have been made in past tens of years [5-7]. In recent researches, Xing made use of ROBDD (reduced ordered binary decision diagrams) to evaluate network with common cause faults, but the method efficiency was not ensured when network became large [6]. Identifying isomorphic subnets with same edge ordering and decomposition level, Hardy's method decreased repeated calculation and enhanced the reliability computation, but its efficiency would deteriorate when the network scale and level number increased [8]; In Kuo's method based on OBDD (ordered binary decision diagrams), a hash table based on bit vector was designed to avoid repeated computations, but the storage overhead increased when network scale expanded [9]. Although some rectangular networks were referred in many released experiments, few discussed the efficient reliability computation of these special networks.

According to the structure feature of rectangular network, this paper proposed an efficient reliability computation method based on isomorphism determination. With the numbering rules, 2 sorts of special subnets are defined for $2\times m$ networks, and n sorts of special subnets are defined for $n\times m$ networks. According to these special subnets, a hash table for defined subnets is added into OBDD algorithm, and the isomorphic subnets are identified to decrease the repeated computation. Experiments show that this method can not only correctly calculate the reliability values, but also quickly evaluate some large rectangular networks.

The rest parts of this paper are organized as follows. Firstly, the rectangular network and notation are introduced in Section II. Then, the principal and algorithms of reliability computation are described in Section III. Next, the experimental results are present in Section IV. Finally, the conclusions are drawn in Section V.

II. Rectangular Network And Notation

A. Rectangular Network

Definition 1. The $n\times m$ **network** is defined as rectangular network which has grid structure, n rows and m nodes per row.

In this paper, the source terminal of $n\times m$ network locates at first row and first column, and the target terminal locates at last row and last column. In Fig. 1, there are three rectangular networks which are called 2×5 network, 3×5 network and 4×5 network.

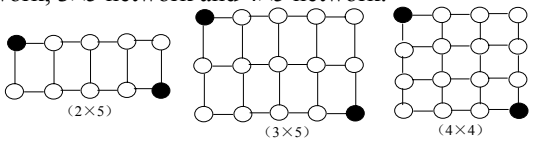


Figure 1. Example rectangular networks

B. Notation

p_{src} source terminal node at first row and first column

Manuscript received January 19, 2014; revised February 10, 2014; accepted February 25, 2014;
Corresponding author: Yufeng Xiao.

doi:10.4304/jcp.9.11.2603-2608

p_{dst} target terminal node at last row and last column

$G(V, E)$ the graph of 2-terminals network topology, V is nodes set, E is edges set

$O_{obdd}(G)$ the OBDD structure of network G

$f_{r\_o}(O_{obdd}(G))$ the reliability calculation function based on the OBDD of network G

G_{*e} the subnet generated from edge e expansion

G_{*a*b} the subnet generated from successive expansions of edge a and b

Supplementary specification:

- The reliability in this paper is defined as connection probability between source terminal and target terminal. If a communication path exists between two terminals, then the network is reliable. Otherwise, the network is unreliable.
- The edges and nodes failures are assumed to be statistical independence.

III. Rectangular Network Reliability Computation Based On Isomorphism Determination

Based on isomorphism determination, the efficiency of OBDD method is enhanced in this paper: firstly, during the OBDD generation process, the numbering rules and n sorts of special subnets are defined to identify isomorphic subnets; secondly, the stored OBDD structure of isomorphic subnet is returned with hash table look-up, and the repeated computations are decreased; lastly, the reliability value is calculated with nodes traversal along OBDD.

A. OBDD Generation

With breadth-first edge search, the OBDD can be constructed with edge expansions, which is described in reference [9]. In Fig. 2(a), the black nodes present source terminal and target terminal of network G (a 2×3 network), and the OBDD is created following successive edge expansions: after expansions of edge e_0 and edge e_1, G is decomposed into G_{*e0} and G_{*e1}; after the deletion of redundant e_4, G_{*e0} is decomposed into G_{*e0*e2} and G_{*e0*e3} with e_0 expansion and e_1 expansion respectively; after e_4 expansion, G_{*e1} is transformed into G_{*e1*e4}; after the deletion of redundant e_6, the source and target of G_{*e0*e2} are merged with e_5 expansion; after the deletion of redundant e_5, the source and target of G_{*e0*e3} are merged with e_6 expansion; after e_6 expansion, the source and target of G_{*e1*e4} are merged; after e_3 expansion, G_{*e1*e4} is transformed into $G_{*e1*e4*e3}$; after successive e_2 expansion and e_5 expansion, the source and target of $G_{*e1*e4*e3}$ are merged. The generated OBDD is illustrated in Fig. 2(b). The mergence means there is a reliable path between source and target terminals.

B Special Subnets Definitions for Isomorphism Determination

For the special structure of rectangular network, if the isomorphic subnets could be identified during network decomposition, the repeated computation would be decreased and the efficiency could be improved. On the basis of 2×m network, the isomorphism determination of n×m network is introduced in this paper. Below is the numbering rule for the nodes of n×m network.

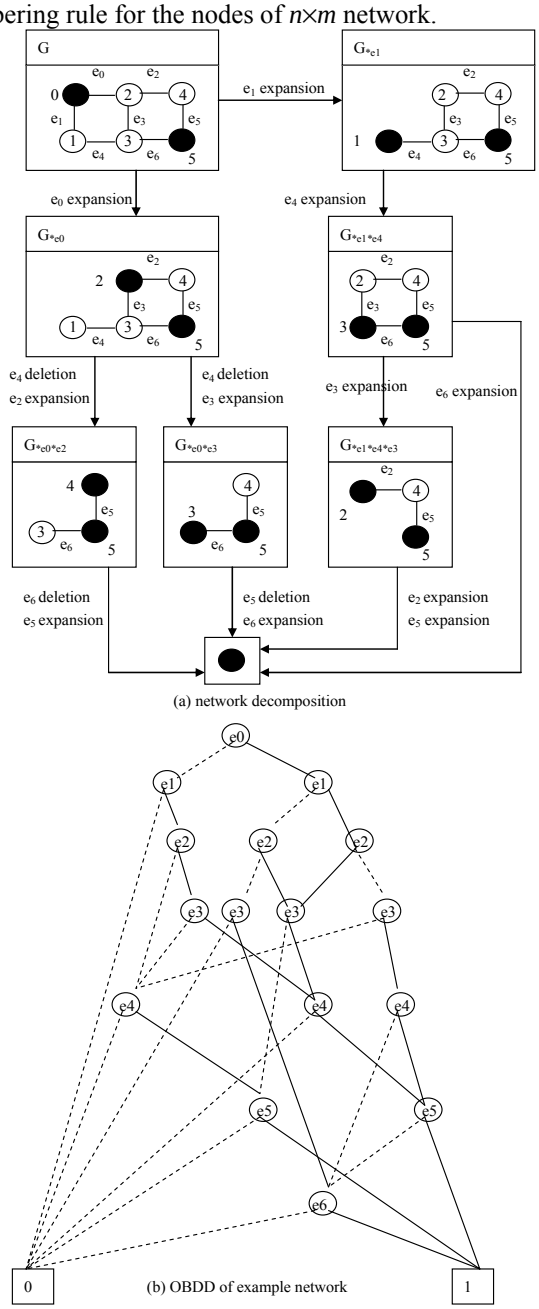


Figure 2. Network decomposition and OBDD construction

Rule 1 For n×m network, if node v locates at i-th row and j-th column, then its number can be calculated as follow:

$$s = n\times(j\text{-}1)+(i\text{-}1) \quad \text{if } 1\le i\le n,\ 1\le j\le m \qquad (1)$$

Where s represents the number of node v.

C. The Special Subnets of 2×m Network

For 2×m network, **Rule 1** can be simplified as follow.

Rule 2 For 2×m network, if node v locates at i-th row and j-th column, then its number can be calculated as follow:

$$s = 2\times(j\text{-}1)+(i\text{-}1) \quad \text{if } i=1 \text{ or } i=2,\ 1\le j\le m \qquad (2)$$

Where s represents the number of node v. Two sorts of rectangular subnets will emerge during 2×m network

decomposition: one sort named (2×*k*)-RN(0), and the other sort named (2×*k*)-RN(1). For these subnets, source locates at *i*-th row and *j*-th column, and *k*=(*m*-*j*+1).

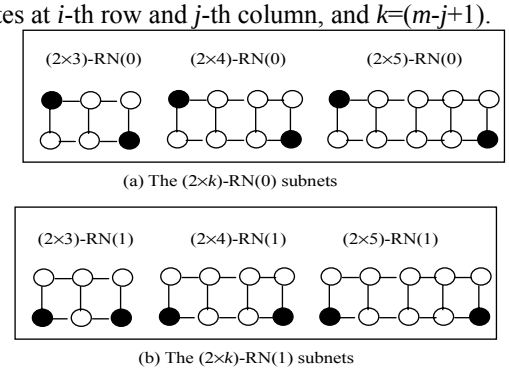


Figure 3. 2×*m* network

Definition 2 For the decomposition of 2×*m* network, let *s* be the source number, and a generated subnet is called **(2×*k*)-RN(0)** if the source satisfies such conditions: the degree of source equals to 2; *s* is even number; *k*=(*m*-*s*/2); two adjacent nodes numbers equals to *s*+1 and *s*+2. For example, subnets in Fig. 3(a) satisfies (2×*k*)-RN(0) features.

Definition 3 For the decomposition of 2×*m* network, let *s* be the source number, and a generated subnet is called **(2×*k*)-RN(1)** if the source satisfies such conditions: the degree of source equals to 2; *s* is odd number; *k*=(*m*-*s*/2); two adjacent nodes number equals to *s*-1 and *s*+2. For example, subnets in Fig. 3(b) satisfies (2×*k*)-RN(1) features.

In Fig. 4, the 2×5 network is discomposed and lots of isomorphic subnets emerge. Without the consideration of isomorphism determination, subnets which are decomposed twice include (2×3)-RN(0), (2×3)-RN(1), (2×2)-RN(0) and (2×2)-RN(1). If same subnets are found being decomposed again, repeated decomposition can be avoided and the computations are saved.

D. The Special Subnets of n×m Network

According to **Rule 1**, *n* sorts of subnets will emerge during *n*×*m* network decomposition: the first one named (*n*×*k*)-RN(0), the second one named (*n*×*k*)-RN(1), and the *p*-th one named (*n*×*k*)-RN(*i*-1). For these subnets, source locates at *i*-th row and *j*-th column, and *k*=(*m*-*j*+1).

Definition 4 For the decomposition of *n*×*m* network, let *s* be the source number, and a generated subnet is called (*n*×*k*)-RN(0) if the source satisfies such conditions: the degree of source equals to 2; *s* mod *n* = 0; *k*=(*m*-*s*/*n*); two adjacent nodes number equals to *s*+1 and *s*+*n*. For example, Fig. 5(a) illustrates a subnet named (3×3)-RN(0).

Definition 5 For the decomposition of *n*×*m* network, let *s* be the source number, and a generated subnet is called (*n*×*k*)-RN(*t*) if the source satisfies such conditions: the degree of source equals to 3; *s* mod *n* = *t*, (1≤*t*≤*n*-2); *k*=(*m*-*s*/*n*); three adjacent nodes number equals to *s*-1, *s*+1 and *s*+*n*. For example, Fig. 5(c) illustrates a subnet named (10×10)-RN(*t*).

Definition 6 For the decomposition of *n*×*m* network, let *s* be the source number, and a generated subnet is called (*n*×*k*)-RN(*n*-1) if the source satisfies such conditions: the degree of source equals to 2; *s* mod *n*=*n*-1; *k*=(*m*-*s*/*n*); two adjacent nodes number equals to *s*-1 and *s*+*n*. For example, Fig. 5(b) illustrates a subnet named (5×5)-RN(4).

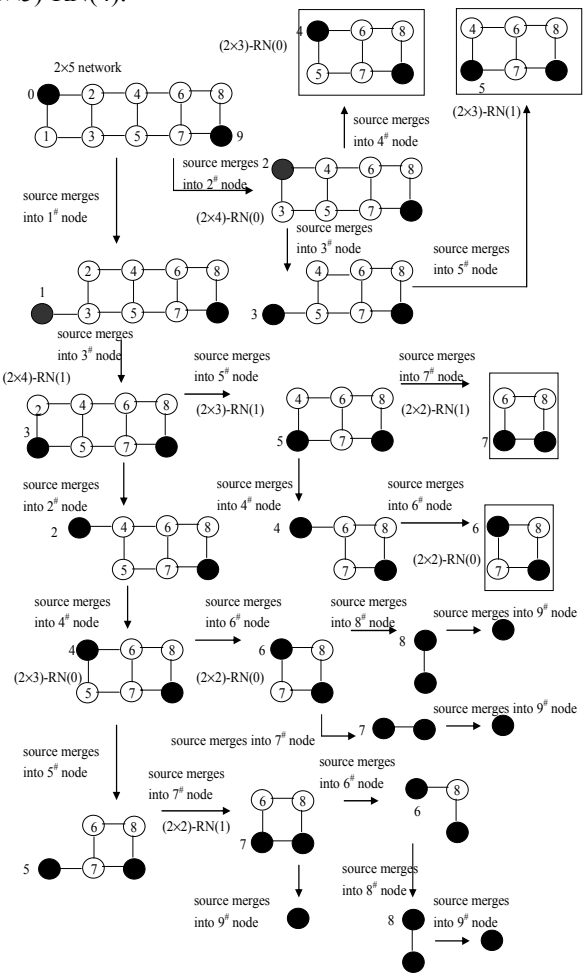


Figure 4. The decomposition of 2×5 network

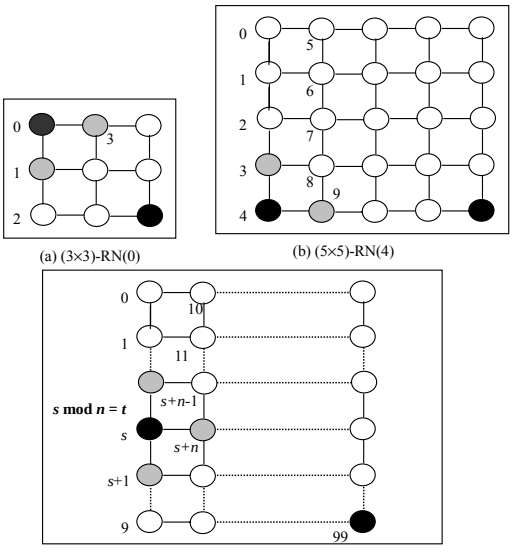


Figure 5. *n*×*m* network

E. Reliability Computation Improvement

Based on defined subnets, the keys to avoid repeated computations are the hash table design and isomorphism determination.

1) Hash table design

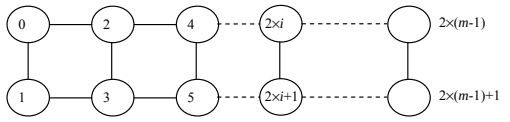
- **The hash table for 2×*m* network**

For the 2×*m* network structure in Fig .6(a), the hash table in Fig. 6(b) is used to record the OBDD of special subnets defined above: the (2×*j*)-RN(0) element records the OBDD of (2×*j*)-RN(0), and the (2×*j*)-RN(1) element records the OBDD of (2×*j*)-RN(1). Here, *j* satisfies 2≤*j*≤*m*-1. Let *num* be the node number, and the hash functions are defined as below:

$$f_{hash\_row}(num) = num / 2, \quad f_{hash\_col}(num) = num \bmod 2 \tag{3}$$

where $f_{hash\_row}()$ returns row number of this table, and $f_{hash\_col}()$ returns column number. Input source number of special subnets, and its OBDD position can be found with these two functions.

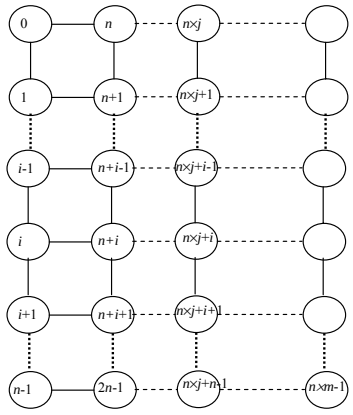
- **The hash table for *n*×*m* network**



(a) 2×*m* network

(2×(*m*-1))-RN(0)	(2×(*m*-2))-RN(0)	……	(2×*j*)-RN(0)	……	(2×2)-RN(0)
(2×(*m*-1))-RN(1)	(2×(*m*-2))-RN(1)	……	(2×*j*)-RN(1)	……	(2×2)-RN(1)

(b) hash table of 2×*m* network



(c)*n*×*m* network

(*n*×(*m*-1))-RN(0)	(*n*×(*m*-2))-RN(0)	……	(*n*×*j*)-RN(0)	……	(*n*×2)-RN(0)
(*n*×(*m*-1))-RN(1)	(*n*×(*m*-2))-RN(1)	……	(*n*×*j*)-RN(1)	……	(*n*×2)-RN(1)
……	……	……	……	……	……
(*n*×(*m*-1))-RN(*i*)	(*n*×(*m*-2))-RN(*i*)	……	(*n*×*j*)-RN(*i*)	……	(*n*×2)-RN(*i*)
……	……	……	……	……	……
(*n*×(*m*-1))-RN(*n*-1)	(*n*×(*m*-2))-RN(*n*-1)	……	(*n*×*j*)-RN(*n*-1)	……	(*n*×2)-RN(*n*-1)

(d) hash table of *n*×*m* network

Figure 6. The hash table of rectangular subnets

For the *n*×*m* network structure in Fig .6(c), the hash table in Fig. 6(d) is used to record the OBDD of special subnets defined above: the (*n*×*j*)-RN(*i*) element records the OBDD of (*n*×*j*)-RN(*i*). Here, *i* satisfies 0≤*i*≤*n*-1, and *j* satisfies 2≤*j*≤*m*-1. Let *num* be the node number, and the hash functions are defined as below:

$$f_{hash\_row}(num) = num / n, \quad f_{hash\_col}(num) = num \bmod n \tag{4}$$

where $f_{hash\_row}()$ returns row number of this table, and $f_{hash\_col}()$ returns column number. Input source number of special subnets, and its OBDD position can be found with these two functions.

2) Isomorphism determination

- **Isomorphism determination for 2×*m* network**

For a subnet *G*' with source terminal *v*, let $f_{degree}()$ return the node degree, let $f_{number}()$ return the node number, and let $k=(m-f_{number}(v)/2)$. The steps to identify the isomorphic subnet are listed below:

Step 1. If ($f_{degree}(v)=2$), continue to next step; otherwise exit.

Step 2. *G*' and (2×*k*)-RN(0) are determined to be isomorphic, if such conditions are satisfied: ($f_{number}(v)$ mod 2=0), and the number of two adjacent nodes are $f_{number}(v)+1$ and $f_{number}(v)+2$. And then, the (2×*k*)-RN(0) position in hash table are calculated: $f_{hash\_row}(f_{number}(v))$ and $f_{hash\_col}(f_{number}(v))$. If it is the first time emergence, the OBDD of (2×*k*)-RN(0) will be recorded into hash table element at above position. Otherwise, the stored OBDD will be returned from hash table. When *G*' and (2×*k*)-RN(0) are not isomorphic, no OBDD is returned and continue to next step.

Step 3. *G*' and (2×*k*)-RN(1) are determined to be isomorphic, if such conditions are satisfied: ($f_{number}(v)$ mod 2=1), and the number of two adjacent nodes are $f_{number}(v)-1$ and $f_{number}(v)+2$. And then, the (2×*k*)-RN(1) position in hash table are calculated: $f_{hash\_row}(f_{number}(v))$ and $f_{hash\_col}(f_{number}(v))$. If it is the first time emergence, the OBDD of (2×*k*)-RN(1) will be recorded into hash table element at above position. Otherwise, the stored OBDD will be returned from hash table. When *G*' and (2×*k*)-RN(1) are not isomorphic, no OBDD is returned and exit.

- **Isomorphism determination for *n*×*m* network**

For a subnet *G*' with source terminal *v*, let $f_{degree}()$ return the degree of node, let $f_{number}()$ return the number of node, and let $k=(m-f_{number}(v)/n)$. The steps to identify the isomorphic subnet are listed below:

Step 1. *G*' and (*n*×*k*)-RN(*t*) are determined to be isomorphic, if such conditions are satisfied: ($f_{degree}(v)=3$), ($f_{number}(v)$ mod *n*=*t*), and the number of three adjacent nodes are $f_{number}(v)-1$, $f_{number}(v)+1$ and $f_{number}(v)+n$. And then, the (*n*×*k*)-RN(*t*) position in hash table are calculated: $f_{hash\_row}(f_{number}(v))$ and $f_{hash\_col}(f_{number}(v))$. If it is the first time emergence, the OBDD of (*n*×*k*)-RN(*t*) will be recorded into hash table element at above position. Otherwise, the stored OBDD will be returned from hash table. When *G*' and (*n*×*k*)-RN(*t*) are not isomorphic, no OBDD is returned and continue to next step.

Step 2. If ($f_{degree}(v)=2$), continue to next step; otherwise exit.

Step 3. G' and ($n\times k$)-RN(0) are determined to be isomorphic, if such conditions are satisfied: ($f_{number}(v)$ mod n=0), and the number of two adjacent nodes are $f_{number}(v)$+1 and $f_{number}(v)+n$. And then, the ($n\times k$)-RN(0) position in hash table are calculated: $f_{hash\_row}(f_{number}(v))$ and $f_{hash\_col}(f_{number}(v))$. If it is the first time emergence, the OBDD of ($n\times k$)-RN(0) will be recorded into hash table element at above position. Otherwise, the stored OBDD will be returned from hash table. When G' and ($n\times k$)-RN(0) are not isomorphic, continue to next step.

```
/* Adjacent: return the adjacent edges */
/* bdd_and, bdd_or: and / or operation for BDD */
/* rn_hash: hash table stores OBDDs of rectangular subnets*/
/* RemoveRedundant: remove the redundant edges */
/* LookupHash: look up rn_hash */
/* InsertHash: inser OBDD into rn_hash */
OBDDConstruct(network G)
{ if (source is merged into target)
      return bddtrue; /*logic true of BDD */
   RemoveRedundant(G);
   if (G exists in the network_hash)
   { bdd bdd_sub_net = LookupHash(G);
    return bdd_sub_net;  /*return OBDD in rn_hash */
   }
   bdd bdd_G=bddfale;
   for(each edge v = Adjacent(source))
   { bdd bdde = BDD(e);
     sub_G = EdgeExpand(G,v)    /*execute edge expansion */
     bdd bdd_sub_net = OBDDConstruct(sub_G);
     bdd_sub_net = bdd_and(bdd_sub_net, bddv);
     bdd_G= bdd_or(bdd_G, bdd_sub_net);
   }
   InsertHash(G,bdd_G);
   return bdd_G;
}
/* high: return right child of OBDD*/
/* low: return left child of OBDD */
double OBDDRel(bdd bdd_graph)
{ p = operation probability of edge e;
   if(bdd_graph is bddtrue)          return 1.0;
   else if(bdd_graph is bddfalse)    return 0.0;
   bdd bdd_high = high(bdd_graph);
   bdd bdd_low = low(bdd_graph);
   reliability = p*OBDDRel(bdd_high)
                 +(1-p)* OBDDRel(bdd_low));
   return reliability;
}
```

Figure 7. The pseudo code of OBDD generation and reliability computation algorithm

Step 4. G' and ($n\times k$)-RN(n-1) are determined to be isomorphic, if such conditions are satisfied: ($f_{number}(v)$ mod n=n-1), and the number of two adjacent nodes are $f_{number}(v)$-1 and $f_{number}(v)+n$. And then, the ($n\times k$)-RN(n-1) position in hash table are calculated: $f_{hash\_row}(f_{number}(v))$ and $f_{hash\_col}(f_{number}(v))$. If it is the first time emergence, the OBDD of ($n\times k$)-RN(n-1) will be recorded into hash table element at above position. Otherwise, the stored OBDD will be returned from hash table. When G' and ($n\times k$)-RN(n-1) are not isomorphic, no OBDD is returned and exit.

3) Enhance the OBDD generation algorithm with isomorphism determination

In Fig. 7, OBDDConstruct() describes the enhanced OBDD generation algorithm with isomorphism determination: rn_hash is the hash table desrcibed above; LookupHash() determine if subnets are isomorphic with special subnets confined by definition 4, definition 5 and definition 6; InsertHash() inserts the OBDDs of defined subnets into hash table.

4) Reliability computation with recursive method

After constructing the OBDD with enhanced algorithm, the network reliability of network G can be computed with below recursive formula:

$$f_{r\_o}(O_{obdd}(G)) = \begin{cases} 1.0, & \text{if } O_{obdd}(G) = bddtrue \\ 0.0, & \text{if } O_{obdd}(G) = bddfalse \\ \Pr(x_k = 1)\times f_{r\_o}(O_{obdd}(G)|_{x_k=1}) & \text{otherwise} \\ +\Pr(x_k = 0)\times f_{r\_o}(O_{obdd}(G)|_{x_k=0}), 0 \le k < n \end{cases} \quad (5)$$

Where, $O_{obdd}(G)|_{x_k=1}$ is the OBDD when x_k=1 (edge e_k is reliable); $O_{obdd}(G)|_{x_k=1}$ is the OBDD when x_k=0 (edge e_k is unreliable); Pr(x_k=1) is the probability when e_k is reliable, and Pr(x_k=0) is the probability when e_k is unreliable; *bddtrue* and *bddfalse* represent the logic true and false of OBDD; the initial value of k is 0. In Fig. 7, OBDDRel() describes the algorithm of this formula. In this function, the reliability is calculated with OBDD traversal, and the OBDD represented by variable bdd_graph is generated by OBDDConstruct().

IV. EXPERIMENT

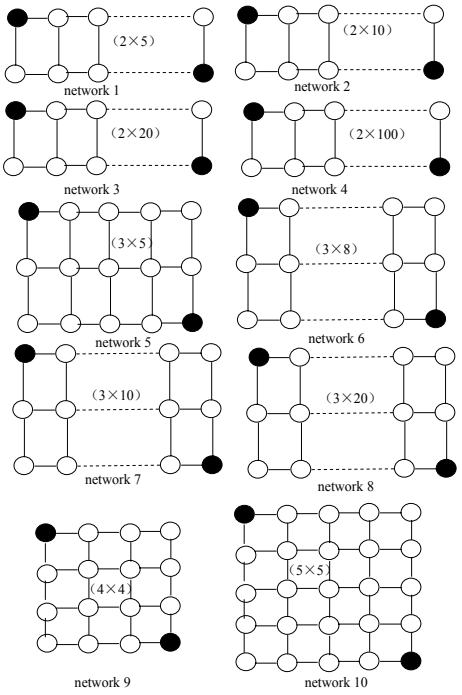


Figure 8. Benchmark networks

The BDD library of Buddy version is used to design the reliability computation method, and the implementation program is called "NetRel_RN" [10]. According to reference [7], a BDD-based algorithm is also implemented with this library, and the experiment

program is called “Bdd”. To evaluate the efficiency of method proposed in this paper, experiment is arranged to compare “NetRel_RN” with “Bdd”. The experiment computer had a CPU processor with 2.4GHz and the memory size is 2GB. The operating system was a Linux system with kernel version 2.6. The networks as shown in Fig. 8 were used and the reliability results were listed in Table I. All the networks had appeared in references [7-9] and they had become the well known benchmark networks to examine the correctness and efficiency of reliability algorithm. In each network, the black solid nodes represent source and target, and all edges had the same functioning probability of 0.9.

In Table II, the “network” column means the benchmark network, the “reliability” column means the network reliability, the “Bdd” column means the running time of method from reference [7], and the “NetRel_RN” column means the running time of method proposed in this paper. In this table, the value of running time is the average from 10 executions of each program, and the running time exceeds one hour is expressed with "----". From the results, it can be found that NetRel_RN’s time cost is less than Bdd’s, especially for network 4 and network 8. This comparison shows that the method proposed in this paper can efficiently evaluate the reliability of large network.

TABLE I.
RELIABILITY AND RUNNING TIME

network	reliability	Bdd(s)	NetRel_RN(s)
1	0.91499	0.2731	0.0062
2	0.88307	1.4802	0.0325
3	0.78448	106.72	12.130
4	0.30432	----	76.040
5	0.97135	3.0712	0.0506
6	0.96723	18.443	0.3403
7	0.96447	302.42	5.3430
8	0.94652	----	92.673
9	0.97505	6.7322	0.7288
10	0.97556	407.87	1.2521

V. CONCLUSIONS

For the $n \times m$ network, isomorphic subnets will emerge when the network is decomposed. If these isomorphic subnets can be identified, repeated computation of same subnets will be decreased. In this paper, n sorts of special subnets for $n \times m$ network are defined, and the isomorphism determination method is proposed. During network decomposition, if a subnet is isomorphic with defined subnet, the stored OBDD of this subnet in hash table will returned and the repeated OBDD construction will be avoided. The experiment shows that the method based on isomorphism determination can efficiently evaluate the reliability of large rectangular network.

ACKNOWLEDGMENT

This work is supported by the Nuclear Energy Development Program of China under Grant No.[2011]1137, Science Foundation of South Western University of Science and Technology under Grant No. 13zx7146, Scientific Research Fund of Education Department of Sichuan Province under Grant No. 14ZA0091, Science and Technology Support Program of Sichuan Province of China under Grant No.2013GZX0152, and Open Fund of Robot Technology Used for Special Environment Key Laboratory of Sichuan Province of China under Grant No.13zxtk03.

REFERENCES

[1] Z. J. Yao, H. Bai, W. D. Yi, “Reliability analysis on grid topology in wireless sensor network,” *Journal of Computer Research and Development*, vol. 47(Suppl.), pp. 55-59, 2010.

[2] H. H. Li, J. Fan, L. Chen, “Application of grid method in deployment of wireless sensor networks,” *Transducer and Microsystem Technologies*, vol. 31(3), pp. 150-152, March 2012

[3] W. Wang, L. Qiao, G. W. Yang, “Interconnection analysis of extended 2-D grid networks-on-chip,” *Journal of Tsinghua University(Science and Technology)*, vol. 50(4), pp. 533-538, April 2010

[4] S. Y. Fan, G. Y. Xiong, “Research on architecture and key technologies of the grid communication network,” *Journal of Xidian University(Science and Technology)*, vol. 36(6), pp. 991-995, December 2009

[5] Y. N. Jiang, R. Y. Li, N. Huang, “Survey on network reliability evaluation methods,” *Computer Science*, vol. 39(5), pp. 9-13, May 2012

[6] A. Rodionov, D. Migov, O. Rodionova, “Improvements in the efficiency of cumulative Updating of all-terminal network reliability,” *IEEE Transactions on Reliability*, vol. 61(2), pp. 460-465, June 2012

[7] L. D. Xing, “An efficient binary decision diagram based approach for network reliability and sensitivity analysis,” *IEEE Transactions on Systems, Man, and Cybernetics--Part A: Systems and Humans*, vol. 38(6), pp. 105-115, January 2008.

[8] G. Hardy, C. Lucet, N. Limnios, “K-terminal network reliability measures with binary decision diagrams,” *IEEE Transactions on Reliability*, vol. 56(3), pp. 506-515, September 2007.

[9] S. Y. Kuo, F. M. Yeh, H. Y. Lin, “Efficient and exact reliability evaluation for networks with imperfect vertices,” *IEEE Transactions on Reliability*, vol. 56(2), pp. 198-211, June 2007.

[10] H. Andersen, *An introduction to binary decision diagrams*, New York: Elsevier, 1997, pp. 5-50.

Yufeng Xiao received his Ph.D. degree from Beijing University of Posts and Telecommunications. He is now a associate professor of Information Engineering School in Southwest University of Science and Technology. His research interests include network reliability, computer networks and communication technology.