©2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

2009 International Conference on Advanced Information Networking and Applications Workshops

Using Multi-valued Decision Diagrams to Solve the Expected Hop Count Problem

Johannes U. Herrmann, Sieteng Soh, Geoff West
Department of Computing
Curtin University of Technology
Bentley, Western Australia
e-mail: jherrmann@gmail.com

Suresh Rai
Department of Electrical and Computer Engineering
Louisiana State University
Baton Rouge, LA, USA

***Abstract*—The Expected Hop Count (EHC) of a computer communication network has so far been computed for network models that consider only device or link failure, but not both. We introduce an Augmented Ordered Multi-valued Decision Diagram (OMDD-A) to obtain the EHC of a network in which both devices and links may fail. The OMDD-A approach can compute the EHC of a 2×100 grid network with 2^{99} paths, which is unsolvable using existing techniques. We show that OMDD-A generates significantly fewer nodes than the corresponding ordered binary decision diagram, leading to large reductions in processing time.**

Keywords- expected hop count, imperfect nodes, imperfect links, multi-value decision diagram, network reliability

I. INTRODUCTION

Reliability and performability are important for the design and maintenance of computer communication networks, road transport, power distribution and many other networks. Both links and nodes of the network are subject to failure, which impacts on performance. Network reliability (REL) has been studied extensively [1]-[5] but only considers the network connectivity. The performance of a network is often measured in terms of how long it takes messages to travel. The Expected Hop Count (EHC) [6]-[9] computes the number of vertices that a message is expected to pass through on the shortest active path from source to target.

AboElFotoh [6] has shown that computing EHC for the general network, where vertices may fail but edges are perfect, is #P-hard. A breadth-first-search combined with the factoring theorem was proposed to calculate EHC for networks with a single source and target; however the solution does not scale well with large networks. Soh, *et al*. [7] proposed a more efficient sum-of-disjoint products (SDP) technique that generates all network minpaths, sorts them in increasing cardinality, and then applies an SDP technique [1]. It is shown [7] that the SDP technique is significantly faster than the factoring approach. Neither of these approaches [6],[7] is feasible for computing EHC for networks with an extremely large number of paths, such as the 2×100 grid network with 2^{99} paths. Brooks, *et al*. [8] use random graph models to approximate EHC in mobile WSN for EHC with a single source and target, but assume fallible edges and perfect vertices. Recently, AboElfotoh, *et al*. [9] extended the factoring approach [6] to solve EHC for multiple sources.

Currently no work considers the more general case where both links and devices can fail. Since EHC is a #P-Hard problem [6], existing solutions are exponential in the number of vertices or edges of the network graph. Ball, *et al*. [3] proposed transforming the graph model G(V,E) into one that considers only edge failures, G'(V',E'). Since |E'| = |V|+|E|, the complexity of computing EHC from G' is exponential in the sum of the edges and vertices of the original graph.

The Ordered Binary Decision Diagram (OBDD) [10] has been considered one of the most efficient methods for representing Boolean functions [11]. OBDDs have been used to solve REL [2]-[5] but do not store sufficient information to solve EHC. Decision Diagrams (DDs) with more than two output values were first suggested in [12] for the simulation of circuits. This concept was extended in [13], which presented a formalization of the Multi-variable Decision Diagram (MDD) and analyzed its properties. Both [14] and [15] produced formal and efficient implementations of the MDD. Research has shown that the MDD uses less space than the equivalent BDD over a wide variety of benchmarks [15],[16]. MDDs have been applied to a number of areas, including circuit design and verification [14], Petri nets [17] and fault tolerant systems [18]. However, to the best of our knowledge, MDDs have not yet been applied to REL or other performability measures such as EHC.

In this paper, we propose an Augmented Ordered MDD (OMDD-A) to compute the EHC of a network when both edges and vertices are susceptible to failure. Our simulations in Section IV show that its performance is far superior to that of the binary equivalent; the Augmented OBDD [19].

The layout of the paper is as follows. Section II discusses terminology and reviews DDs and their use in REL. Section III present the OMDD-A to compute EHC. We give results in section IV and conclusions and future work in Section V.

II. BACKGROUND

A. Network Model and Terminology

We model a computer communication network (CN) using a graph G=(V,E), where each vertex in V represents a communication device and every edge in E represents a communication link between the devices. A vertex v_j or edge e_j is said to be UP (DOWN) if it is functioning (failed). Let p_j (φ_j) be the operational probability of vertex v_j (edge e_j) and assume all failures are statistically independent.

Let n=|V|, and let the vertices (v_0, v_1, …, v_{n-1}) of V be ordered in increasing distance from the source vertex, v_0,

978-0-7695-3639-2/09 $25.00 © 2009 IEEE
DOI 10.1109/WAINA.2009.124

IEEE computer society

Authorized licensed use limited to: CURTIN UNIVERSITY OF TECHNOLOGY. Downloaded on February 26,2010 at 01:33:16 EST from IEEE Xplore. Restrictions apply.

with the target vertex v_t always labeled as v_{n-1}. When two or more vertices have the same distance from v_0, they are ordered arbitrarily. Let (v_i,v_j) ($\{v_i,v_j\}$) denote a directed (undirected) edge between vertices v_i and v_j, with $i>j$ for each $\{v_i,v_j\}$. Fig. 1 shows an example network that illustrates such an ordering.

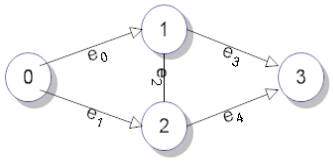

Figure 1. Sample Network

A *path* P_i is a sequence of UP vertices $(v_a, v_b, \ldots v_k)$ in V such that there exist UP edges $e_{ab} = (v_a,v_b)$ or $\{v_a,v_b\}$, $e_{bc} = (v_b,v_c)$ or $\{v_b,v_c\}$, *etc*. A *reaching path* P_i to v_x is a path from v_0 to v_x where each vertex in P_i is traversed only once. A *minpath* is a reaching path to v_t. A *diagram path* to node N_i is a path in a DD starting at the root and leading to N_i. In this paper, we use *node* and *link* to refer to the elements of a DD, and *vertex* and *edge* for those of the CN.

A network state $\Omega=(V_U,E_U)$ of network $G=(V,E)$ is a partition of G such that all vertices in $V_U \subseteq V$ and edges in $E_U \subseteq E$ are UP and all other vertices and edges are DOWN. The probability of a state $\Omega=(V_U,E_U)$ is computed as:

$$\Pr(\Omega)=\prod_{v_i\in V_U} p_i \prod_{v_i\notin V_U}(1-p_i)\prod_{e_i\in E_U}\varphi_i \prod_{e_i\notin E_U}(1-\varphi_i),$$

since all failures are assumed to be statistically independent. A state is a *success state* if it contains at least one minpath. There are $2^{|V|+|E|}$ network states in G, but REL and EHC are computed only from the set of all success states, Ω_S.

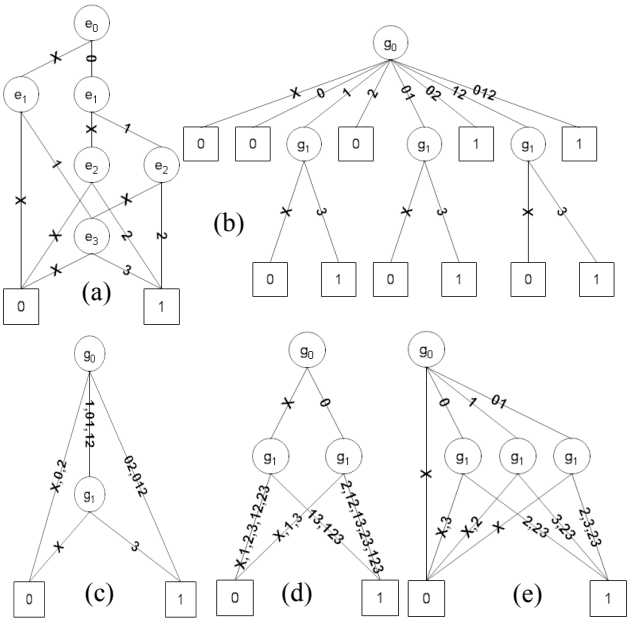

Figure 2: An OBDD (a) and OMDDs (b)-(e) representing $f=e_0e_2+e_1e_3$

In addition to the success state information, computing the EHC requires the *length* of each success state, $\Omega \in \Omega_S$, denoted as $1 \le L(\Omega) \le n-1$. Thus, $L(\Omega)$ is the length (the number of hops or hop count) of the shortest minpath contained in Ω. We assume that the routing protocol in the network always finds the shortest available minpath [9]. When this minpath is unavailable the router finds the next possible shortest minpath. Formally, the EHC is given by:

$$\mathrm{EHC}=\frac{\sum_{\Omega\in\Omega_S}(L(\Omega)\times\Pr(\Omega))}{\sum_{\Omega\in\Omega_S}\Pr(\Omega)} \quad (1)$$

B. Multi-valued Decision Diagrams

Figures 2(a) and 2(b) to 2(e) show the BDD and MDDs, respectively of a Boolean function $f=e_0e_2 + e_1e_3$. Each *non-terminal* node (a circle in Fig. 2) in a DD represents the evaluation of one or more Boolean variables, with one sub-tree representing each possible combination of values of these variables. In an Ordered DD (ODD), the variable order is fixed for all branches/paths of the diagram. Following a path from the root node, variables are decided in a given order until a value is returned. This value is stored in the *terminal* node of that path (a square in Fig. 2).

Links in Fig. 2 are labeled with the subscript of the variables that are UP, or are labeled with an 'X' in the case when no variables are UP. When one link represents multiple combinations of variables, these combinations are separated by commas. For example, in Fig. 2 (c) the label on the leftmost edge leaving the root node is "X,0,2". This label represents three cases; all variables being DOWN ('X'), variable e_0 being UP and e_1 and e_2 being DOWN ('0'), and e_2 being UP and e_0 and e_1 being DOWN ('2') respectively.

Each node in an Ordered BDD (OBDD) such as in Fig. 2(a) represents one Boolean variable, and thus has two children. By comparison, a node in an Ordered MDD (OMDD) may represent a group of several variables, and hence it can have more than two children. For example, the root node of the OMDD in Fig. 2(b) is evaluating a group of three variables (e_0,e_1 and e_2) and thus it has 8 children.

An OMDD has a fixed variable grouping at each level of the diagram, although the number of variables in each grouping does not have to be identical. Nagayama and Sasao [16] showed that, over a wide variety of benchmarks, that such an OMDD uses less space than the equivalent OBDD. Note that each level of an OMDD represents the evaluation of one particular variable group.

The efficiency of a DD implementation is measured by its number of nodes [17] and its depth [16]. A good variable ordering can effectively reduce the number of nodes and depth of an OMDD [16]. A better variable ordering may result in more isomorphism between nodes in the diagram.

Two non-terminal (terminal) nodes are isomorphic if they have equivalent sub-trees (they produce identical outputs). Merging isomorphic nodes reduces the size of the ODD; this operation effectively prunes one of the sub-trees. For example, we obtain the smaller ODD in Fig. 2(c) by merging the three isomorphic nodes (labeled g_1) and the terminal

Authorized licensed use limited to: CURTIN UNIVERSITY OF TECHNOLOGY. Downloaded on February 26,2010 at 01:33:16 EST from IEEE Xplore. Restrictions apply.

nodes in Fig. 2(b). Finding the optimal variable ordering has been shown an NP-Complete problem [20].

The MDD is a natural extension of the BDD, in that it has *d* terminal nodes labeled 0 to *d*-1, and similar labels are possible for the outputs of each MDD node. Each non-terminal node has a fixed number (*d*) of outputs, although some implementations remove redundant outputs. A MDD that has an equal number of outputs for every non-terminal node (before considering isomorphism) is referred to as *homogeneous*; otherwise it is a *heterogeneous* MDD.

Fig. 2 (e) shows the homogeneous OMDD with variable grouping $g_0=\{e_0,e_1\}$ and $g_1=\{e_2,e_3\}$. The OMDD has four nodes, which is more efficient than the equivalent six node OBDD in Fig. 2(a). Fig. 2(c) shows a heterogeneous OMDD with groups $g_0=\{e_0,e_1,e_2\}$ and $g_1=\{e_3\}$, and Fig. 2(d) shows another with $g_0=\{e_0\}$ and $g_1=\{e_1,e_2,e_3\}$; these OMDDs have less nodes than the homogeneous OMDD in 2(e). In this example the OMDD with $g_0=\{e_0,e_1,e_2\}$ and $g_1=\{e_3\}$ gives the optimal result. Unfortunately, finding the optimal grouping is even more difficult than the NP-complete problem of finding the optimal variable ordering [20].

C. Decision Diagrams and Network Reliability

The Boolean function $f=e_0e_2+e_1e_3$ can represent the pathset $\{e_0e_2,\ e_1e_3\}$ of a network if only edges are considered. The OBDD in Fig. 2(a) is used to compute REL for this network with perfect vertices. In the application of the DD technique to REL [2] the probability that the network is connected is given by tracing paths upwards from the success terminal nodes and multiplying by the probability of the variable(s) being UP or DOWN as appropriate. Since each traversed path represents a disjoint event the probability of each such path is summed to give REL.

As an example, consider $f = e_0e_2 + e_1e_3$ as the reliability function of a network (for paths e_0e_2 and e_1e_3), with each edge having a probability of 0.9 of being UP. Thus, the REL of the network can be obtained from Fig. 2(a) by following each diagram path from the root to the success node (marked as 1), and multiplying by 0.9 for each positive edge and 0.1=(1-0.9) for each negative edge (marked with an 'X'). The right-hand diagram path is $e_0e_1e_2$ which has a probability of 0.729. The other diagram paths are $e_0\overline{e_1}e_2$, $e_0e_1\overline{e_2}e_3$ and $\overline{e_0}e_1e_3$. Note that the nodes of the DD do not contain any information other than what is inherent through their position in the diagram. Therefore the existing OBDD approaches [2],[4],[5] cannot be used to compute EHC, since this requires path length information (See Section II A).

OBDD have been efficiently used for computing REL [2],[5], and fault covering and tolerance [18]. Kuo, *et al*. [2] have proposed a recursive EED-ISO algorithm to compute REL for a network with perfect vertices and failed edges. The use of node isomorphism in OBDD makes EED-ISO able to compute REL for a 2×100 grid network with 2^{99} minpaths. Yeh, *et al*. [5] use the OBDD for calculating REL for a one-to-many network, where one vertex must be connected with *k*-1 other vertices of the network. *k*-1 different REL are calculated, which are then combined to give the *k*-terminal reliability. Although these approaches [2],[5] are efficient for computing REL, they are not useful for computing EHC. The method in [2], for example, only generates OBDD nodes to take advantage of isomorphism through hash table lookups, but never explicitly links them into a diagram. The approach can compute REL by traversing the OBDD nodes, but cannot calculate the EHC, whose computation requires path length information.

III. Augmented Decision Diagrams

A. Augmenting Decision Diagram Nodes

For computing EHC, each OMDD node requires more information than just its position in the diagram. In particular each node, N_i, must store the state(s) of the CN that it represents, given the decisions that have been made in the diagram path(s) that lead to that node. We call the heterogeneous OMDD that comprises such nodes an Augmented OMDD (OMDD-A).

State information includes a set VI_i of vertex components, $M_x=\{(v_a, L^x_a), (v_b, L^x_b), \ldots, (v_k, L^x_k), P^x\}$. Each pair $(v_j\ L^x_j) \in M_x$ denotes an undecided vertex v_j known to be reachable from v_0 along with the length L^x_j of the shortest reaching path known. The probability P^x is the probability of being in the network state represented by M_x. Section III E describes how P^x is computed. For each VI_i we define a vertex set, $VS_i=\{v_a, v_b, \ldots v_k\}$. Each VI_i also contains a set CI_i of conditional paths of the form (v_x, v_y, L) where L is the length of the shortest path between undecided vertices v_x and v_y. When the first vertex, v_x, of a conditional path is reachable (*i.e.* $v_x \in VS_i$) and decided UP, (v_x, v_y, L) is moved to VI_i as a minpath to v_y by appending it to the minpath to v_x.

The augmenting information in each OMDD-A node is computed from its parent nodes (discussed in Section III E). Each separate component represents a different network state and the probability of being in this state. Because the OMDD-A stores state-based information, we can construct each level using only the nodes of the previous level. It also allows the tracking of information such as path length needed to compute EHC. Note that the number of children does not affect the size of a node since links are not explicitly stored. However, as the number of components in a node increases so does the size of the node. This size increase is generally manageable since less than two levels of nodes are kept in memory at any one time.

B. Variable Order and Grouping

The depth of an OMDD can be reduced by using variable partitioning [16]. For the OMDD-A algorithm, we group each vertex v_j with its adjacent edges. Note that both directions of undirected edges are considered at one time, hence any edge will only be in one variable grouping. Thus a node that decides a group consisting of v_j with *d* adjacent ungrouped edges has 2^{d+1} children. An undirected edge is grouped with the first endpoint in the ordering; the use of conditions ensures that it only has to be considered once.

Note that those 2^d children for which v_j is DOWN represent identical network states to the child for which all edges are DOWN and v_j is UP. To reduce the number of

Authorized licensed use limited to: CURTIN UNIVERSITY OF TECHNOLOGY. Downloaded on February 26,2010 at 01:33:16 EST from IEEE Xplore. Restrictions apply.

nodes of the OMDD-A, we only generate one *negative* child, which represents the network state of the 2^d+1 identical children. Hence an OMDD-A node deciding the grouping of v_j and d adjacent edges has only 2^d children. The link connecting a parent node to its negative child is referred to as the *negative link* and is marked with an 'X'. For example, consider the network in Fig. 1. The OMDD-A partition for this network is g_0={v_0,e_0,e_1}, g_1={v_1,e_2,e_3} and g_2={v_2,e_4}. The negative child for g_0 represents the case when either v_0 is DOWN, or e_0 and e_1 are both DOWN.

C. OMDD-A Node Types

We consider two types of OMDD-A nodes: *terminal*, and *non-terminal*. Our approach processes each non-terminal node in a breadth-first fashion and completes when there are no more such nodes. A terminal node can be either a *success node* (whose value is a hop count of 1 or more) or a *failure node* (with a value of 0). The REL and EHC are computed from the reaching path probabilities contained in all success nodes (discussed in Section III E).

```
TestNode (Ni):
// Let lb be the least path length in Mx to a vertex vb∈ VIi
// Let lt be the least path length in Mx to target vt
1. if VSi = {} then return 1.        // a failure node
2. else if vt∉ VSi then
3.          return 3.                // a non-terminal node
4. for each Mx∈ VIi do                          // for each component
5.          if (lb < lt-1) then
6.                    return 3.  // a non-terminal node
7. return 2. //a success node
```

Figure 3. The TestNode function

A failure node has no sub-trees containing a success node. It is favorable to detect failure nodes as early as possible, since REL are computed only from success nodes. However the processing cost of testing for failure must be kept to a minimum. N_i is a failure node if VS_i={}; if the node has no information on any undecided vertices then no new vertices (including the target) will be reached. However, a failure N_i may have a non-empty VS_i, and detecting such nodes is computationally expensive. Our TestNode function in Fig. 3 returns a value 1, 2, or 3 if N_i is a failure, success, or non-terminal node, respectively. Note that a node that contains multiple components is not a success node if any of the components do not meet the criteria. The OMDD-A implementation removes successful components from the node as soon as they are detected, and stores their information to avoid unnecessary overhead.

D. Node Isomorphism

We consider non-terminal nodes N_i and N_j at the same level of an OMDD-A *isomorphic* iff VS_i=VS_j and CI_i=CI_j. We check isomorphic nodes only from their equal VS and CI to tradeoff between the number of isomorphic nodes and the processing time complexity per node.

Two isomorphic nodes N_i and N_j can be merged into one node that keeps the VS and CI of merged nodes; without loss of generality let the resulting node be N_i for $i<j$. We say components M_x={(v_a, L^x_a), (v_b, L^x_b), …, (v_k, L^x_k), P^x } and M_y = {(v_a, L^y_a), (v_b, L^y_b), …, (v_k, L^y_k), P^y } are *equal* (and write M_x=M_y) iff L^x_α=L^y_α for α =a to k. Fig. 4 shows the Merge function.

```
Merge( Ni, Nj):
// Let VSi=VSj={va, vb, … , vk)
1. for each My← {(va, L^y_a), (vb, L^y_b), …, (vk, S^y_k), P^y}∈ VIj do
2.       for each Mx ←{(va, L^x_a), (vb, L^x_b), …, (vk, L^x_k), P^x} ∈ VIi do
3.          if Mx = My then
4.                Mx ← {(va, L^x_a), (vb, L^x_b), …, (vk, L^x_k,), P^x+P^y }.
5.                break.
6.          if no Mx was equal to My then //i.e., Step 3 was always False
7.                add My to VIi
8.  Replace old Ni on QN with updated Ni
```

Figure 4. The Merge function

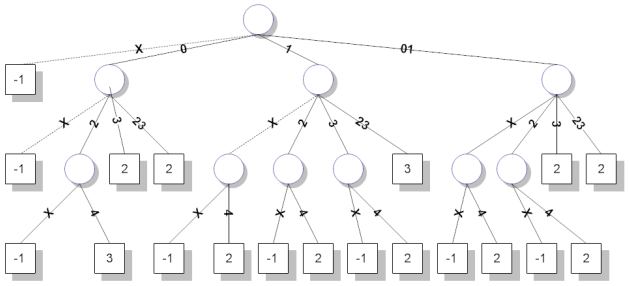

Figure 5: Unmerged OMDD-A

Consider the OMDD-A in Fig. 5, which computes the EHC of the network in Fig. 1; the diagram has not been reduced using isomorphism. Links between a node and its children have been labeled with the edges that are UP (*e.g.* 01 indicates that both e_0 and e_1 are UP, in addition to v_0 being UP). Negative links are labeled with an X. Terminal nodes are marked with the EHC, or 0 for a failure node. The shaded nodes deciding variable group g_2 are isomorphic, and thus are merged before being further processed to create sub-trees; the resulting OMDD-A is shown in Fig. 6.

While the top two levels of the diagram in Fig. 6 are unchanged, the third level now has only three non-terminal nodes instead of seven. Although each g_1 node in Fig. 6 could have a maximum of four sub-trees, two have less than this, and several of the sub-trees consist solely of a terminal node. This represents a reduction from a theoretical maximum of 53 (1+4+16+32) nodes to 19 nodes in this diagram, most of which are terminal nodes.

Terminal nodes are not processed to create sub-trees. Failure nodes are simply discarded, while success nodes have their probability contribution stored before being discarded. For this reason the OMDD-A algorithm does not merge terminal nodes, and it is more appropriate to compare diagrams by counting only non-terminal nodes. The OMDD-A in Fig. 6 has 7 non-terminal nodes.

An OMDD-A has |V| levels, compared to |V|+|E| levels for OBDD-A. The advantage of the OMDD-A having less depth is offset by the fact that more nodes are generated at each level. However, since augmented nodes are only tested for isomorphism with other nodes on the same level, the amount of isomorphism in the diagram increases. This leads to a far superior performance compared to deciding vertices and edges one at a time. For comparison, an OBDD-A equivalent to the OMDD-A in Fig. 6 would contain 31 non-terminal nodes in 9 levels. Hence, for this example OMDD-A uses only 23% ($^7/_{31}$) of the nodes of the OBDD-A.

Authorized licensed use limited to: CURTIN UNIVERSITY OF TECHNOLOGY. Downloaded on February 26,2010 at 01:33:16 EST from IEEE Xplore. Restrictions apply.

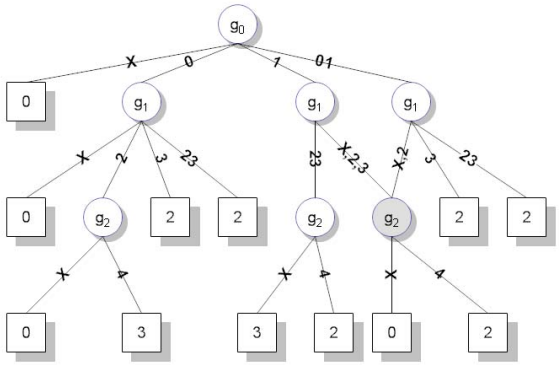

Figure 6: OMDD-A with Non-Terminal Nodes Merged

E. Creating an OMDD-A for Computing EHC

The function in Fig. 7 constructs an OMDD-A to compute the EHC and REL. The function first creates a root node N_0, and places it in a queue Q_C, which is used to hold all nodes at the level currently being decided. The other queue, Q_N, is used to hold the child nodes created at the next lower level, and is initialized to null. The decision variable, DV, and all length probabilities, Pr(L), are initialized to 0.

Lines 3 to 21 are a loop that first checks if Q_C is empty; Q_C being empty indicates that we have finished a level of the diagram. If Q_C and Q_N are both empty the algorithm terminates. If Q_C is empty but Q_N is non-empty, we increment DV and move the contents of Q_N to Q_C to start generating the next level of the diagram.

```
1. Create root node N0
2. QC←{N0}, QN←{ }, DV←0, and Pr(L) ← 0 (for all 0≤L≤|E|).
3. if QC = { } then
4.         if QN = { }then
5.                 calculate REL and EHC from Pr(L)
6.         else
7.                 QC ← QN, QN ← { } and DV ← DV + 1.
8. remove the first node Ni from QC.
9. for each combination of unmarked edges (vDV,vx), (vx,vDV), or
     {vDV,vx}:
10.        create child N based on UP edges
11.        if N is non-terminal then
12.                for each Nq∈QN do
13.                        if N is isomorphic to Nq then
14.                                call Merge(Nq, N).
15.                                break.
16.                if no Nq was isomorphic to N then
17.                        add N to QN.
18.        else if N is a success node then
19.                store results in Pr(L).
20. mark all edges used in step 9 above as decided
21. goto 3.
```
Figure 7. The OMDD-A function

The next step (line 8) removes the first N_i from Q_C. We generate the $D = 2^d$ children of N_i, and label them N_{d*i+1} to N_{d*i+D} where d is the number of undecided edges adjacent to vertex v_{DV}, (*i.e.*, all edges of the form (v_{DV},v_x), (v_x,v_{DV}), or $\{v_{DV},v_x\}$ where $x > DV$). The first of these child nodes, N_{d*i+1}, is the negative child that represents the case where the v_{DV} is DOWN or v_{DV} is UP and all edges grouped with it are DOWN. All other nodes represent v_{DV} being UP and some combination of the edges grouped with it being UP. Each N_j contains information on the state of the CN, in particular the length of the shortest reaching path to each vertex in VS_j. All non-terminal child nodes which are isomorphic with the nodes in Q_C are merged while others are added to the end of Q_C (lines 11 to 17).

Failed terminal nodes are discarded. Success nodes have their information added to the relevant length probabilities, Pr(L), before being discarded. For example if a component in the success node has a path of length 3 to the target vertex with a probability of 0.081, then Pr(3) is increased by 0.081. The sum of these length probabilities produces REL, and can be used to calculate EHC using Equation (1).

IV. Simulation Results and Discussions

OMDD-A has been implemented in C++ and tested on a Pentium computer (2 Xeon 3.2GHz processors, 1MB cache, 2GB RAM) for evaluating the REL and EHC of a variety of networks. Each reported CPU time is averaged over five runs for each simulation. Execution was halted after 5 hours (CPU time). Further, terminal nodes are excluded when stating the number of nodes since non-terminal nodes incur more of a processing cost and since the number of (merged) terminal nodes is fixed for the EHC of a given network.

Other than OBDD-A, we know of no other method that calculates EHC for networks with both node and edge failure. Thus, explicit comparison is made only between the OBDD-A and OMDD-A methods.

A. Computing the EHC using OMDD-A

Both the OMDD-A and OBDD-A were first applied to the the 19 benchmark networks from [1]. While OMDD-A was able to compute the EHC of the networks, OBDD-A failed to compute the metrics for networks 13, 17 and 19 in 5 hours of CPU time. We observed that the efficiency of both implementations is strongly affected by the maximum boundary set size of the network [4].

For those networks solvable by both, the OMDD-A requires less nodes than the OBDD-A. This result is consistent with that in [15],[16] for OMDD and OBDD. For these networks, the OMDD-A requires at most 22.6% of the nodes of the OBDD-A. The most telling difference was for the network from Fig. 7 of [1] (|V|=7, |E|=21), with the OBDD-A generating 292,504 nodes (out of a possible 2^{28}) and the OMDD-A only 102 nodes. Further, OMDD-A reduces the height of OBDD-A (28 levels) to only 7.

The comparison of execution time yields a similar result. As an example, for the network from Fig. 18 from [1] (with 13 vertices and 22 edges) OMDD-A took just 0.5% of that of OBDD-A (0.7s to 159.6s).

In order to compare our results with Soh *et. al.*[7] we generated random networks using BRITE and applied the OMDD-A algorithm. Our algorithm took longer to complete as compared to [7], but the orders of magnitude were the same. In addition it must be noted that our algorithm considers both node and edge failure, which means the problem solved by OMDD-A is in the order of 150 variables instead of 50 for SDP. Further OMDD-A can compute the EHC of some networks (*e.g.*, 2×100 grid) that cannot be solved using the SDP approach.

Authorized licensed use limited to: CURTIN UNIVERSITY OF TECHNOLOGY. Downloaded on February 26,2010 at 01:33:16 EST from IEEE Xplore. Restrictions apply.

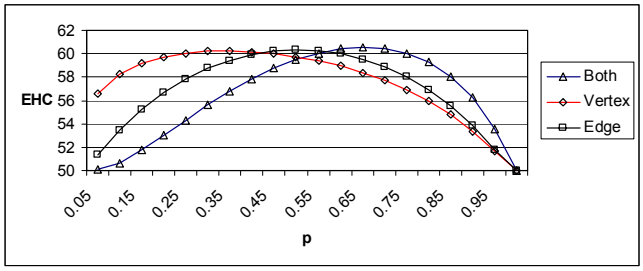

Figure 8: EHC with Different Component Reliability Scenarios

Fig. 8 shows the effect of component reliability on EHC. We calculated the EHC of the 2×50 grid network, and considered three cases: (i) only edge failure, (ii) only vertex failure, (iii) both failures. The curve for EHC is similar for all cases. Note, the result for case (iii) is skewed the most to the right, and that for case (ii) is skewed the most to the left (line with diamond). This shows that vertex failure has the least impact on path length, since for larger p it has the lowest EHC.

B. *Effect of Boundary Set on Performance*

The implementation was also tested on other networks with comparable results. For the 2×100 grid network from [2] the OMDD-A generates 18.8% (593 compared to 3155) of the nodes of the OBDD-A. Similar results were produced for other 2×w grids tested. On networks with a larger maximum boundary set (B_{max}) [4], such as the 4×4 grid, the OMDD-A was able to compute the answer (generating 3098 nodes in 2.87 CPU seconds) while the OBDD-A failed to complete within a reasonable amount of time.

TABLE I. IMPACT OF B_{MAX} ON NODES GENERATED BY OMDD-A

Grid	B_{max}	Nodes	Grid	B_{max}	Nodes
2×10	2	53	2×12	2	65
3×7	3	879	3×8	3	1678
4×5	4	26824	3×6	4	447666

To demonstrate the effect of B_{max} we tested our algorithm on a number of grid networks with a similar number of vertices but varying B_{max}. Table I shows that the number of OMDD-A nodes generated increases exponentially as B_{max} increases. Note that the number of nodes is also influenced by other factors, such as the ordering of the vertices and edges in the network.

V. CONCLUSION

We have shown that OMDD-A is more time and space efficient than OBDD-A for computing the REL and EHC when network edges and vertices can fail. In our simulations, OMDD-A generates from under 1% to around 25% of the nodes of the OBDD-A. Since the complexity of OMDD-A is not directly related to the number of paths, our technique is suitable for networks (*e.g.*, grid) with extremely large pathsets, not solvable by the existing techniques [6]-[9].

We are investigating more effective variable orderings and groupings for the OMDD-A. We will investigate other methods of storing network information in augmented nodes, including boundary sets [4] and the connectivity matrix. These approaches may reduce processing time and memory use, but it is not clear if they can be modified to efficiently record the path lengths of visited vertices.

REFERENCES

[1] S. Soh and S. Rai, "CAREL: Computer Aided Reliability Evaluation for Distributed Computing Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 2, Mar. 1991, pp. 199-213.

[2] S.-Y. Kuo, S.-K. Lu, and F.-M. Yeh, "Determining terminal-pair reliability based on edge expansion diagrams using OBDD," *IEEE Trans. Reliability,* vol. 48, Dec. 1999, pp. 234-246.

[3] M. O. Ball, C. J. Colbourn, and J. S. Provan, "Network Reliability," in *Network Models*. vol. 7, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, Eds.: Elsevier, 1985, pp. 673-762.

[4] G. Hardy, C. Lucet, and N. Limnios, "Computing all-terminal reliability of stochastic networks with Binary Decision Diagrams," in 11th International Symposium on Applied Stochastic Models, 2005.

[5] F.-M. Yeh, S.-K. Lu, and S.-Y. Kuo, "OBDD-based evaluation of k-terminal network reliability," *IEEE Trans. Reliability,* vol. 51, Dec. 2002, pp. 443-451.

[6] H. M. F. AboElFotoh, "Algorithms for Computing Message Delay for Wireless Networks," *Networks*, vol. 27, Dec. 1997, pp. 117-124.

[7] S. Soh, W. Lau, and S. Rai, "On Computing the Reliability and Expected Hop Count of Wireless Communication Networks," *Int. Journal Performability Engineering* (IJPE), vol. 3, no. 2, April 2007, pp. 267-279.

[8] R.R. Brooks, B. Pillai, S. Racunas, and S. Rai, "Mobile network analysis using probabilistic connectivity matrices," *IEEE Trans. Syst. Man and Cybernetics*, Part C, vol. 37, no. 4, Nov. 2007, pp. 694-702.

[9] H.M.F. AboElFotoh, S.S. Iyengar, and K. Chakrabarty, "Computing reliability and message delay for cooperative wireless distributed sensor networks subject to random failures," *IEEE Trans. Reliability*, vol. 54, no. 1, Mar. 2005, pp. 145-155.

[10] R.E. Bryant, "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams," *ACM Computing Surveys*, vol. 24, no. 3, Sep. 1992, pp. 293-318.

[11] F. M. Yeh and C. S. Lin, "Building BDDs with ordering-reshuffle strategy," *Electronics Letters,* vol. 29, Aug.1993, pp. 1540-1541.

[12] E. Cerny and J. Gecsei, "Simulation of MOS Circuits by Decision Diagrams," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 4, pp. 685-693, Oct. 1985.

[13] A. Srinivasan, T. Ham, S. Malik, and R. K. Brayton, "Algorithms for discrete function manipulation," in IEEE Intl Conf. Computer-Aided Design, Santa Clara, CA, 1990, pp. 92-95.

[14] P. C. McGeer, K. L. McMillan, A. Saldanha, A. L. Sangiovanni-Vincentelli, and P. Scaglia, "Fast discrete function evaluation using decision diagrams," in 1995 IEEE/ACM Intl Conf. Computer-Aided Design, San Jose, CA, 1995, pp. 402-407.

[15] D. M. Miller and R. Drechsler, "Implementing a multiple-valued decision diagram package," in 28th IEEE Intl Symp. Multiple-Valued Logic, Fukuoka, 1998, pp. 52-57.

[16] S. Nagayama and T. Sasao, "On the optimization of heterogeneous MDDs," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 24, pp. 1645-1659, Nov. 2005.

[17] A. S. Miner and G. Ciardo, "Efficient reachability set generation and storage using decision diagrams," in 20th Intl Conf. Application and Theory of Petri Nets, 1999, pp. 6-25.

[18] R. Gulati and J. B. Dugan, "A modular approach for analyzing static and dynamic fault trees," *Proc. Annu. Reliability and Maintainability Symp.*, Philadelphia, PA, 1997, pp. 57-63.

[19] J. Herrmann, S. Soh, and G. West, "An OBDD Approach for Computing Expected Hop Count of Communication Networks," in PEECS Perth, Australia: Curtin University of Technology, 2007.

[20] B. Bollig and I. Wegener, "Improving the variable ordering of OBDDs is NP-complete," *IEEE Trans. Computers,* vol. 45, Sep. 1996, pp. 993-1002.

Authorized licensed use limited to: CURTIN UNIVERSITY OF TECHNOLOGY. Downloaded on February 26,2010 at 01:33:16 EST from IEEE Xplore. Restrictions apply.