Comparison of Binary and Multi-Variate Hybrid Decision Diagram Algorithms for *K*-Terminal Reliability

Johannes U. Herrmann and Sieteng Soh
Department of Computing
Curtin University of Technology
GPO Box U1987, Perth 6845, West Australia

jherrmann@ieee.org

Abstract

The Ordered Binary Decision Diagram (OBDD) has been efficiently used to compute the communication network (CN) reliability (REL). The boundary set method (BS) is used to improve the efficiency of the OBDD approach, while the augmented OBDD diagrams (OBDD-A) that stores CN information in its nodes has been proposed to solve other CN performance metrics in addition to REL. The hybrid OBDD (OBDD-H) combines the BS and OBDD-A features to further improve the performance of the OBDD method. However, both BS and OBDD-H address only CN with communication link (edge) failure. In this paper, we generalize OBDD-H for networks with edge and/or device (vertex) failures. We also present a hybrid ordered multi-variate decision diagram (OMDD-H) to compute the performance metrics of CN with vertex and link failures. This paper examines the time and space complexities of OBDD-H, and shows that OMDD-H can compute the REL in CN with fallible vertices and edges in the same order of complexities as BS or OBDD-H computing the same network with only vertex failure.

Keywords: boundary set, decision diagram, network reliability, time and space complexity.

Acronyms

ALL-REL – All terminal reliability – As reliability (*q.v.*) but with the requirement that all devices in the network be connected.

BS – Boundary Set – A way to store information on the state of a network; in this work BS refers to the method of using a boundary set and an OBDD *q.v.* to compute network reliability.

CN – Communication Network – A wired or wireless network where communication occurs between source and target devices.

DD – Decision Diagram – A structure used to compute network reliability.

K-REL - *K* terminal reliability – As reliability (*q.v.*) but with the requirement that a set *K* of devices in the network be mutually connected.

OBDD – Ordered Binary Decision Diagram - A DD whose binary variables are decided in a fixed order.

OBDD-A – Augmented OBDD – An OBDD with information on network paths stored in each node.

OBDD-H – Hybrid OBDD – An OBDD with information on boundary set connectivity stored in each node; a hybrid of the OBDD-A and BS methods.

OMDD – Ordered Multi-variate DD - As OBDD but without the requirement that all variables be binary. There are also multi-variate versions of the OBDD-A and OBDD-H.

REL – Reliability – The probability that the source and target vertices are connected by active devices and communication links.

WSN – Wireless Sensor Network – A communication network in which the devices are sensor devices which communicate with each other in a wireless manner and co-operate to route transmissions to target devices.

1 Introduction

As the use of both wired and wireless communication networks (CNs) increases, their reliability is of importance for both their design and analysis. When components of a network fail, parts of the network may become unavailable and/or performance can degrade. The reliability (REL) of networks has been studied extensively (Hardy *et al.*, 2005, Hardy *et al.*, 2007, Herrmann and Soh, 2009, Herrmann *et al.*, 2009, Herrmann, 2010, Herrmann *et al.*, 2007, Yeh *et al.*, 2002, Carlier and Lucet, 1996) and methods utilizing ordered binary decision diagrams (OBDD) have been shown to be superior to other methods in general (Hardy *et al.*, 2007).

The boundary set OBDD method (BS) introduced by Hardy, *et al*. (2005) uses edge contraction to construct an OBDD which can be used to compute all terminal reliability (ALL-REL) (Hardy *et al.*, 2005) and *K*-terminal reliability (*K*-REL) (Hardy *et al.*, 2007) for the network. BS was shown (Hardy *et al.*, 2005) (Hardy *et al.*, 2007) to be more efficient than other methods available at the time, such as the EED_BFS method proposed by Yeh, *et al*. (2002).

A disadvantage of BS is that it first generates the entire OBDD and then uses this to generate REL, which means that all OBDD nodes must be stored in memory. For large networks this can require that millions of nodes must be stored (Hardy *et al.*, 2007). In addition, the partition numbers used in BS can quickly become larger than native data types in programming languages (*e.g.*, C) can store, requiring the use of computationally expensive libraries that permit the use of numbers of arbitrary size.

Further, BS can be used only for computing the REL of undirected networks whose communication links can fail but whose devices (vertices) are not susceptible to failure.

A network model with fallible communication links but perfect devices is not always appropriate. Wireless networks may suffer both from device failure and the interruption of communication signals (Akyildiz *et al.*, 2002); indeed this holds true for wired networks as well. For example, a mobile telephone communication can fail if the phone battery runs out (device failure) or the phone is taken into a tunnel (link failure).

The augmented ordered decision diagram (OBDD-A) was proposed by Herrmann, *et al*. (2007) to solve *K*-REL and a number of other metrics (e.g., the expected hop count – EHC). The OBDD-A stores additional information (*e.g.*, the state probability) in diagram nodes, allowing nodes to be discarded after use and the metrics to be computed without traversing the diagram a second time. Information was stored by recording which devices were connected to the source device(s) and tracking any existing paths that had not yet been utilized.

Both the BS and OBDD-A methods decrease greatly in performance when computing REL for networks with both link and device failure as compared to device (or link) failure only. The augmented ordered multi-variate decision diagram (OMDD-A) was introduced (Herrmann *et al.*, 2009) as an extension to OBDD-A. It was shown that computing REL and EHC using OMDD-A is comparable in runtime performance to the same computation for a network with only device failure using OBDD-A on networks with only vertex failures.

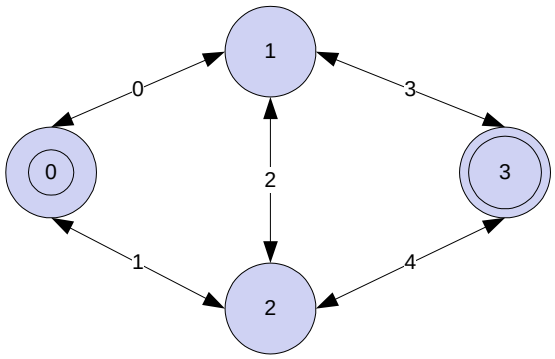

Figure 1: Sample Network

While the augmented diagram methods are able to solve more general networks (*e.g.*, directed networks or those with edge and/or vertex failure) its execution time was far inferior to that of the BS method.

The OBDD-A and BS methods were combined (Herrmann and Soh, 2009) into the hybrid ordered binary decision diagram (OBDD-H) which stored the partition numbers of BS partitioning in diagram nodes. This method was shown empirically to have comparable execution time performance to BS while requiring far less memory (Herrmann and Soh, 2009). However the OBDD-H is also restricted to undirected networks and does not account for vertex failure. In addition, early versions of OBDD-H stored partition numbers, requiring the same libraries as BS for large numbers. This was later modified (Herrmann, 2010) to store partition information directly instead of using partition numbers. This modification decreases running times while maintaining the low space usage of the earlier version.

Our contributions in this paper are twofold. First, we generalize the OBDD-H method to solve networks with vertex and/or edge failures. Then, we propose a multi-variate hybrid decision diagram (OMDD-H) to improve the performances of both OBDD-H and OMDD-A. Our OMDD-A can be used in networks with vertex and link failures.

The rest of this paper is organized as follows. Section 2 presents the necessary network models and terminology, and reviews the BS and OBDD-H methods. The OBDD-H is modified for vertex and edge failure in Section 3 and its performance analysed. We introduce the OMDD-H in Section 4 and summarize the work in Section 5.

2 Background

2.1 Network Model

An undirected network η is modelled using an undirected graph G = (V,E) whose vertices, V, represent the communication devices of η and whose edges, E, represent the communication links of η. If devices are fallible, each vertex $v_i \in V$ has probability $P(v_i)$ of being available. Similarly if communication links are fallible, each edge $e_i \in E$ has probability $P(e_i)$ of being active. If these probabilities are not known they can be estimated as described in (Colbourn, 1987). Each edge between vertices v_x and v_y is written as (v_x, v_y) or simply (x, y) where $x<y$[1].

A network whose devices do not fail is modelled by a graph with perfect vertices (*i.e.*, $Pr(v)=1.0 \ \forall v \in V$). Similarly a network whose communication links do not fail is modelled by a graph with perfect edges (*i.e.*, $Pr(e)=1.0 \ \forall e \in E$). For this work, when a device can fail we assume it does so with a probability of 0.1 (*i.e.* it has a probability of 0.9 of being active).[2]

Depending on the connectivity model in use, each network has one or more source vertices as well as one or more target vertices. For the case where we require a set *K* of vertices to be connected we choose one of these vertices as the source and the others as targets.

For ordered decision diagrams, networks must be ordered. We use a breadth-first ordering for the graph representing each network which orders vertices in increasing distance from the source(s) (except that at least one of the vertices in *K* must be the last vertex in the ordering) and edges in increasing order of lower and then higher endpoints. As an example, consider the network given in Figure 1, with the source vertex v_0 and target v_3. The vertices have been labelled using the ordering given above. Note that switching the labelling of vertices v_1 and v_2 would have been equally correct since both are the same distance from the source vertex. Given the vertex

[1] Strictly speaking this should be $x \leq y$ to allow for self loops. However such loops do not affect the reliability of the network and can hence be ignored.

[2] The algorithm functions with other component probabilities; however this is the standard assumption in the literature.

ordering there is only one correct ordering of edges, which is shown in their labelling.

A network state $\Omega=(V_\Omega,E_\Omega)$ of network $G=(V,E)$ is a partition of G such that all vertices in $V_\Omega \subseteq V$ and edges in $E_\Omega \subseteq E$ are available and all other vertices in V and edges in E are unavailable. The probability of state Ω is computed as:

$$\Pr(\Omega) = \sum_{e_i \in E_\Omega} \Pr(e_i) \sum_{e_i \notin E_\Omega} (1 - \Pr(e_i)) \sum_{v_i \in V_\Omega} \Pr(v_i) \sum_{v_i \notin V_\Omega} (1 - \Pr(v_i))$$

For *K*-terminal reliability (*K*-REL), Ω is a successful state if $K \subseteq V_\Omega$ and each vertex in *K* is connected to all vertices in *K* by a path of vertices in V_Ω and edges in E_Ω. REL can be calculated by summing the probabilities of the success states of the network.

Since each edge and vertex can be in one of two states (available or unavailable), there are $2^{|E|+|V|}$ states for network G, and therefore *K*-REL cannot be solved for large networks through state enumeration.

2.2 Boundary Set Algorithm

Boundary sets were introduced for network reliability by Carlier and Lucet (1996). This method was combined with an OBDD into the BS method by Hardy *et al.* to efficiently solve ALL-REL (2005) and K-REL (2007).

BS first builds an OBDD and then traverses this diagram to compute REL. Each level of the OBDD represents the evaluation of edge e_k of the network. Each node N_i has two children; a positive child representing e_k being available and a negative child representing e_k being unavailable. We refer to the variable (in this case e_k) being decided on level *k* of the diagram as the *decision variable* of that level.

BS utilizes the boundary set (F_k) of each level *k* of the diagram; that is the set of vertices that are of importance to the algorithm at each stage. F_k (deciding edge e_k) is defined as:

$$F_k = \{v_x : v_x \text{ is an endpoint of edges } e_y \text{ and } e_z \text{ with } y \leq k \text{ and } z \geq k\}$$

Each node of the OBDD represents a state of the network, and is encoded by a partitioning of the boundary set. Each partition represents a subset of F_k that is connected; in other words each vertex in the partition is connected to each other vertex by some path.

When computing *K*-REL, we want all vertices in *K* to end up in one partition, showing that they are connected. A node that has such a partition is a success node and a node in which such a partition can no longer exist is a failure node. The connection to *K* is tracked by marking the partition that is connected to the source vertex with an asterisk. If a vertex in *K* is disconnected from the marked partition the node is failed.

The BS algorithm is shown in Figure 2. The algorithm presented here is the one presented by Herrmann and Soh (2009). The algorithm proposed by Hardy *et al.* (2007) does not allow for the failure of positive child nodes (lines 11-12).

The creation of child nodes uses *edge contraction* and *edge deletion*. For a positive child, edge e_k is contracted, merging its endpoints. For a negative child, e_k is deleted, possibly leaving the endpoints disconnected. Vertices that are in F_k but not in F_{k+1} are removed from both child nodes. If such a vertex removal results in an empty partition, this is also removed; when such a partition is removed the child node is *failed* because one or more of the *K* vertices are disconnected from others in *K* (lines 11-12 and 18-19).

```
1. Create root node N2
2. F0={v0}, and REL ← 0.
3. for k = 1 to |E| do
4.     compute Fk+1.
5.     for each Ni on level k do
6.         translate partition number i into parti.
7.         Create negative partition part0.
8.         Create positive partition part1.
9.         if part1 is successful then
10.            positive child of Ni is 1.
11.        else if part1 is failed then
12.            positive child of Ni is 0.
13.        else
14.            translate part1 to number j.
15.            if Nj is not in hash table then
16.                create Nj and insert into hash table.
17.            positive child of Ni is Nj.
18.        if part0 is failed then
19.            negative child of Ni is 0.
20.        else
21.            translate part0 to number j.
22.            if Nj is not in hash table then
23.                create Nj and insert into hash table.
24.            negative child of Ni is Nj.
```

Figure 2: BS Algorithm

This partitioning is itself encoded into a partition number, which is stored and can be transformed back into the partitioning when needed. The process makes use of Stirling numbers of the second kind (A_{ij}). Details of this process can be found in (Hardy *et al.*, 2005) and (Hardy *et al.*, 2007) for ALL-REL[3] and in (Herrmann and Soh, 2009) for *K*-REL.

One of the strengths of the ordered decision diagram is that isomorphic nodes (those that have identical sub-trees) are merged; reducing the number of nodes that need to be stored and processed. This can be seen in Figure 3 where any node with multiple arrows entering is a merged node; each arrow represents one merged node.

The use of boundary sets makes detecting isomorphic nodes simple. Any two nodes on the same level that have an identical partitioning of the boundary set (and hence identical partition number) are isomorphic.

The OBDD created by BS working on the sample network is shown in Figure 3 with the assumption that $K=\{0,3\}$. Solid arrows indicate the positive child and dashed arrows the negative. The shaded non-terminal nodes represent states which are neither succeeded nor

[3] Hardy *et al.* (2007) do give an algorithm to compute the partition numbers for *K*-REL but this is erroneously identical to the ALL-REL version and does not consider marked partitions.

failed. Each such state includes the partitioning that it represents along with the associated partition number (in parentheses) and the probability of being in this state. Note that the partition itself is not stored by BS and the partition number is stored in a separate hash table instead of in the OBDD. Finally the probabilities are not stored in the node but also stored in the hash table when they are computed. The boundary set for each level is shown on the left and the edge being decided is shown on the right.

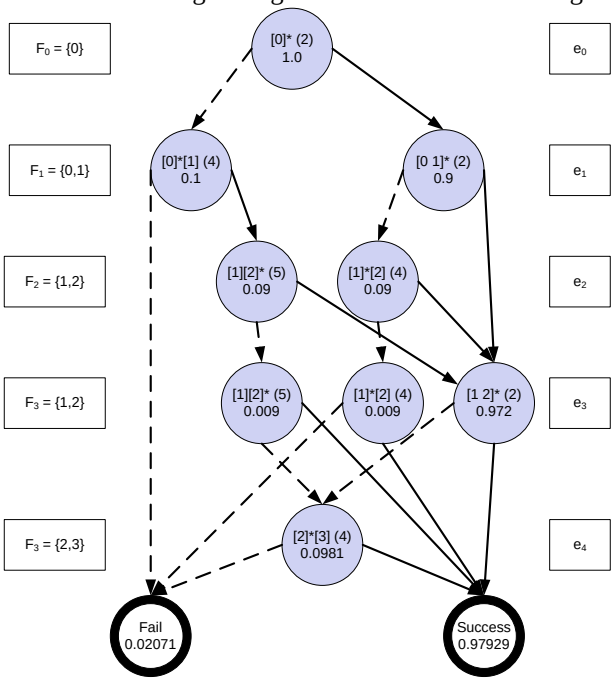

Figure 3: OBDD for Sample Network

The performance of BS was shown to be closely related to the size of the boundary sets; in particular the size of the maximum boundary set (F_{max}). The time and space complexity of the BS method were both shown to be bounded by

$$O\left(|E| \times (F_{max})^3 \times B_{F_{max}}\right)$$

where $B_{F_{max}}$ is the Bell number defined by

$$B_{|F_{max}|} = \sum_{j=1}^{|F_{max}|} A_{|F_{max},j|}$$

2.3 The OBDD-He Algorithm

The OBDD-H was introduced by Herrmann and Soh (2009) for undirected graphs for which edges can fail but vertices are perfect. These conditions are identical to those for BS because the algorithms are closely related. Let us refer to this version of OBDD-H the OBDD-He. Note, in Section 3.1, we present OBDD-Hv for undirected graphs with failed vertices and perfect edges, and in Section 3.2 we describe OBDD-Hve that generalizes OBDD-He and OBDD-Hv for use in networks with failed vertices and/or edges.

Like BS, the OBDD-He uses partitions of F_k to encode the network state. While the OBDD-He (Herrmann and Soh, 2009) initially translated these into partition numbers it was shown empirically (Herrmann, 2010) that it is more efficient to store the partitions directly as structures. In either case, the information is stored directly in each OBDD-He node, either as partition numbers or structures. The OBDD-He for the sample network is shown in Figure 5 and the algorithm is shown in Figure 4.

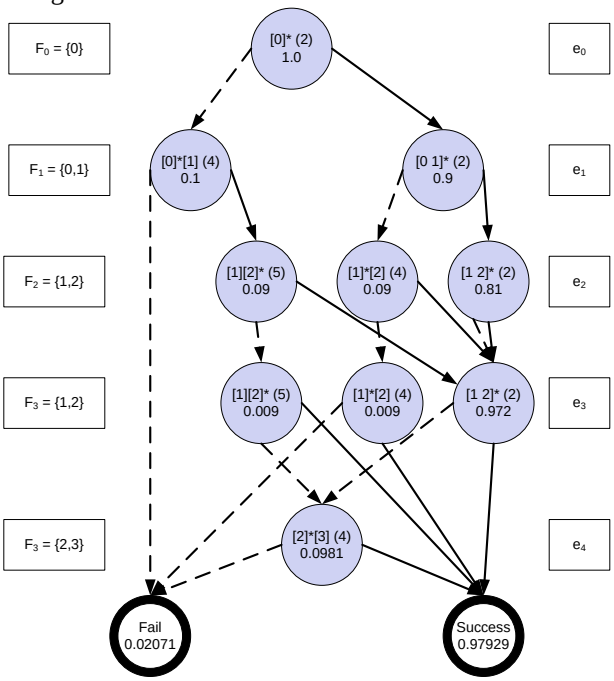

Figure 5: OBDD-He for Sample Network

Also stored in each OBDD-He node is the probability of being in the state represented by that node (and hence the partitioning of F_k stored in that node). BS instead stores this probability and the partition number in a separate hash. The root node of the OBDD-He has probability 1.0 and each child node's probability is computed from that of its parent by multiplying by $P(e_k)$ is available (for the positive child) or unavailable (for the negative).

Most nodes in the OBDD-He are exactly equivalent to

```
1. Create root node N0
2. QC←{N0}, QN←{ }, k←0, and REL ← 0.
3. while (QC != { } or QN != { })
4.     if QC = { } then
5.         QC ← QN, QN ← { } and k ← k + 1.
6.     Fk ← Fk+1, Compute Fk+1
7.     for each Ni on QC.
8.         Create negative child
9.         Create positive child for ek
10.        for each child Nj
11.            if Nj is non-terminal then
12.                for each Nq∈QN do
13.                    if Nj is isomorphic to Nq then
14.                        merge Nj into Nq
15.                        break.
16.                if no Nq was isomorphic to Nj then
17.                    add Nj to QN.
18.            else if Nj is a success node then
19.                REL ← REL + Pr(Nj).
20.    delete Ni
```

Figure 4: OBDD-H for Edge Failure

the comparable node in BS since they have the same partition information. Children of each node are

equivalent since these are created through the same process. When two nodes in the OBDD-He are isomorphic and are merged, the equivalent nodes in BS are also isomorphic and merged.

The difference between the algorithms is that the OBDD-He nodes are not actually linked into a diagram; they are instead stored in queues representing levels *k* and *k*+1 of the diagram. Nodes on level *k* are processed and non-terminal child nodes are either merged with an isomorphic node on level *k*+1 or added to that queue if no isomorphic node exists. Once the child nodes are created the parent node is discarded.

This difference means that the OBDD can be optimized after being created; in particular if both children of a node are isomorphic (and hence merge) the parent node is redundant. This process does not occur for the OBDD-He, as can be seen by the extra node on level 2 of the diagram. It should be noted that this missing node is still created by OBDD and processed, and thus affects time complexity. Because it is subsequently merged it does not affect the space complexity, however.

2.4 The Complexity of OBDD-He

The OBDD-He uses the same methods as BS for generating and merging diagram nodes. For this reason, the structure and number of nodes of both the OBDD generated with BS and the OBDD-He are identical.[4]

Since the time complexity of both algorithms is directly related to the number of diagram nodes generated, the OBDD-He has the same order of time complexity as BS. However the space complexity is not identical.

The OBDD-He discards nodes once they have been processed. This means that nodes from no more than two levels are stored in memory at any one time. This indicates that the space complexity for OBDD-He is bounded by

$$O\big((F_{max})^3 \times B_{F_{max}}\big).$$

When a series of networks increase in size but retain a constant inter-connectivity, it was found (Herrmann and Soh, 2009) that the space complexity had a constant upper bound. The constant inter-connectivity of the networks means that F_{max} is bounded by some constant Γ. Hence the bound for the space complexity becomes $O(\Gamma^3 \times B_\Gamma)$, which is constant and thus verifies the experimental results.

3 The Generalized OBDD-H

3.1 OBDD-H for Vertex Failure (OBDD-Hv)

If vertices fail instead of edges, each level of the decision diagram represents the evaluation of a vertex, which could lead to a number of edges being contracted. For the worst case, each level will represent only one edge being contracted but in general there will be more than one. The resulting nodes will be equivalent to having performed the contraction of each of the edges associated with the vertex, in turn.

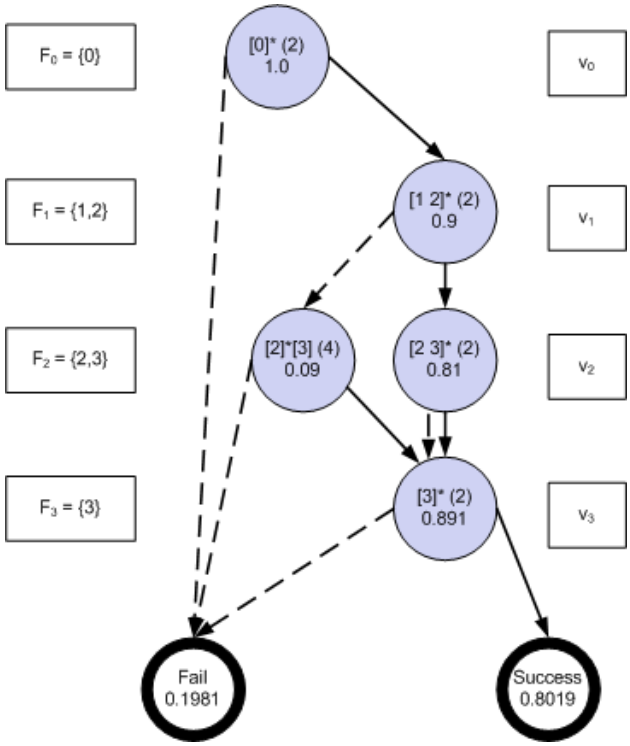

Figure 6: OBDD-Hv of Sample Network

The OBDD-Hv for the sample network is shown in Figure 6, with F_k and the decision variable for each level *k* shown on the left and right of the diagram respectively. Note that this is smaller than the diagram created by OBDD-He (Figure 5) and BS (Figure 3). Indeed the diagram for OBDD-Hv is a subgraph of the larger diagrams.

The process for constructing the OBDD-Hv is similar to that for the OBDD-He except that we effectively perform multiple edge contractions or deletions at every level; one for each edge adjacent to the vertex being decided. Hence the child nodes for OBDD-Hv are equivalent[5] to the nodes in BS where either all adjacent vertices are available (the positive child) or unavailable (the negative child). Hence each level of the OBDD-Hv has the same complexity constraints as for an OBDD-He. However the number of levels of the OBDD-Hv is equal to the number of vertices instead of the number of edges. We deduce that the time complexity of the OBDD-Hv is:

$$O\big((F_{max})^3 \times B_{F_{max}}\big).$$

The space complexity of OBDD-Hv is identical to OBDD-He since there are still at most two levels of nodes stored in memory at any one time.

[4] Given the constraints that certain nodes are removed from the BS OBDD after being processed. These nodes do not affect the worst-case space complexity since the worst case assumes that there are a maximal number of nodes at each level.

[5] The partitions are equivalent for both nodes, but the probabilities of being in the state represented by the nodes are obviously not equal.

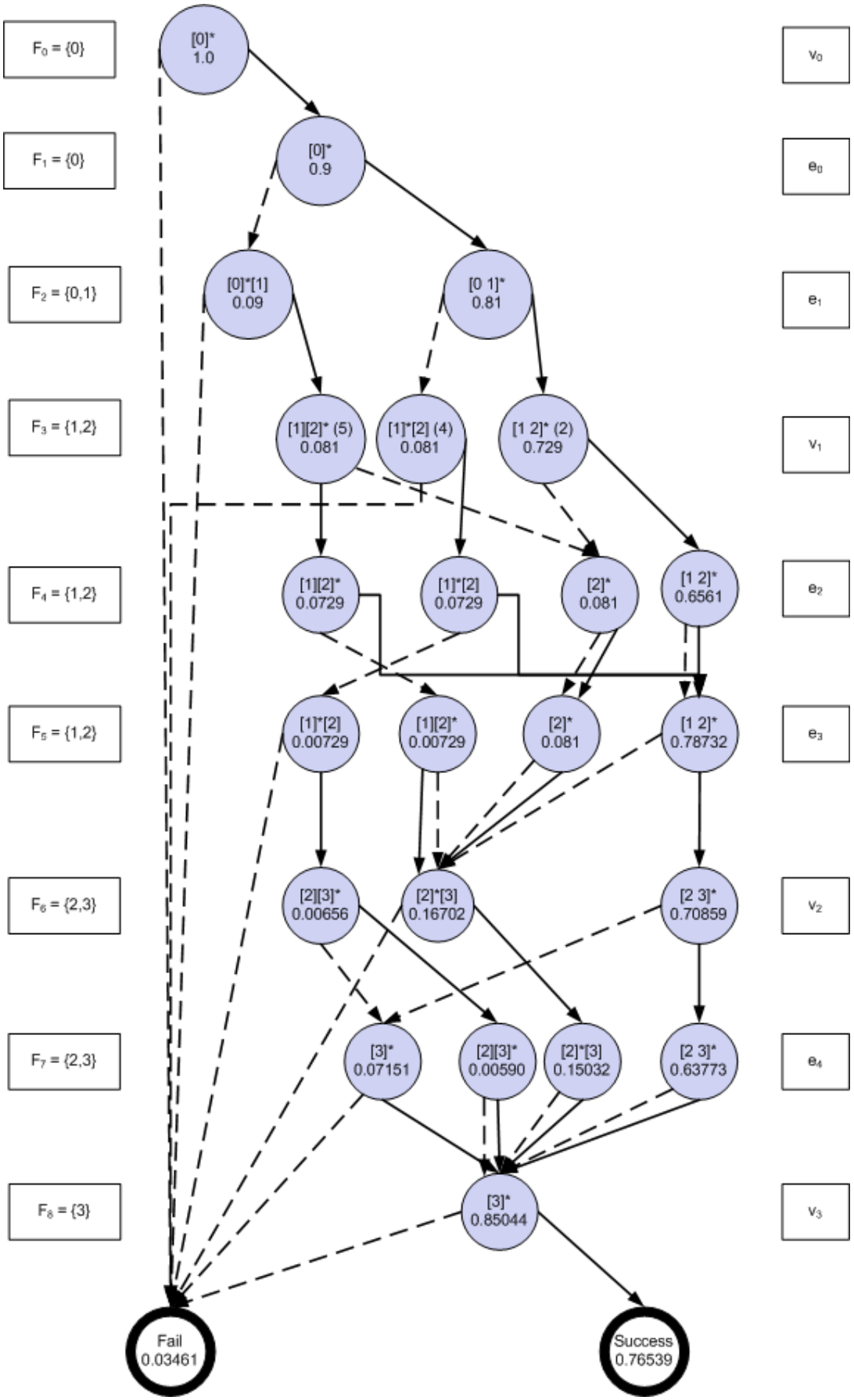

Figure 7: OBDD-Hve for Sample Network

3.2 OBDD-H for Vertex and Edge Failure

When both vertices and edges are fallible each must be decided in turn. We decide each vertex in turn, and decide each edge immediately after the lower endpoint. So for the sample network shown in Figure 1 the variable ordering is v_0, e_0, e_1, v_1, e_2, e_3, v_2, e_4, v_3. The resulting OBDD-Hve is shown in Figure 7. Again, F_k for each level is shown on the left of the diagram and the decision variable is shown on the right. Note that the probabilities shown are rounded to five decimal places where needed.

Note that for the negative child when deciding a vertex we remove the unavailable vertex from its partition. While this means that the result is, strictly speaking, not a complete partitioning of F_k anymore it saves unnecessary computation in the following levels. This means that partition numbers are no longer appropriate, but since OBDD-H does not use them this is not an issue.

Note that the last level of both OBDD-Hv and OBDD-Hve will be a single node containing the partition of the last vertex in the ordering. In some applications it is assumed that the target vertex is always available, in which case the ordering is modified to ensure that the target is last and this level can be omitted. We do not make this assumption in this work, and hence retain the final level.

The depth of the OBDD-Hve means that the number of non-terminal nodes are greater than for both the OBDD-He and OBDD-Hv combined.

The time and space complexities for OBDD-Hve can be similarly deduced to be

$$O\left((|V| + |E|) \times (F_{max})^3 \times B_{F_{max}}\right)$$

and

$$O\left((F_{max})^3 \times B_{F_{max}}\right),$$

respectively, when each vertex and edge is decided in turn. It can be seen that the time required to compute OBBD-Hve is considerably larger than either OBDD-Hv and OBDD-He. The amount of memory required is not greatly affected, however.

4 The Multivariate Hybrid Decision Diagram Algorithm

4.1 The OMDD-A

The augmented ordered multi-variate decision diagram (OMDD-A) was introduced (Herrmann *et al.*, 2009) for solving both REL and the Expected Hop Count problems for networks with both device and link failure. The OMDD-A groups each vertex with any adjacent edges that have not yet been grouped with another vertex. Each level of the diagram decides one grouping. The grouping of variables in a MDD has been shown to affect its performance (Nagayama and Sasao, 2005)

The diagram is further optimized by automatically creating a merged node for the case when all edges in the group are unavailable together with all cases where the vertex is unavailable. For this reason a grouping of m edges with a vertex results in 2^m children for each node before merging, instead of 2^{m+1}. This number is further reduced by merging isomorphic nodes.

As with the OBDD-A, the OMDD-A tracks paths in the graph that have not yet been reached. This number of paths increases rapidly for large networks, degrading the performance. The OBDD-H does not suffer from this issue, since it uses the boundary set system which is independent of the number of paths of the network and depends only of F_{max} and B_{Fmax}.

```
1. Create root node N0
2. QC←{N0}, QN←{ }, k←0, and REL ← 0.
3. while (QC != { } or QN != { })
4.     if QC = { } then
5.         QC ← QN, QN ← { } and k ← k + 1.
6.     Fk ← Fk+1, Compute Fk+1
7.     for each Ni on QC.
8.         Create negative child
9.         for each combination of edges (vk,vx):
10.            create positive child for these edges.
11.        for each child Nj
12.            if Nj is non-terminal then
13.                for each Nq∈QN do
14.                    if Nj is isomorphic to Nq then
15.                            merge Nj into Nq
16.                            break.
17.                if no Nq was isomorphic to Nj then
18.                    add Nj to QN.
19.            else if Nj is a success node then
20.                REL ← REL + Pr(Nj).
21.    delete Ni
```

Figure 8: OMDD-H Algorithm

4.2 The OMDD-H

The hybrid ordered multi-variate decision diagram (OMDD-H) groups variables in the same way as the OMDD-A but uses partitions of F_k to track connectivity information. Like the OBDD-H it is restricted to undirected networks.

For this work we only consider OMDD-H for networks with both vertex and edge failure and don't consider the task of grouping vertices or edges together for other failure conditions (such as fallible vertices and perfect edges).

The OMDD-H for the sample network is shown in Figure 9 with the variable groupings shown on the right and probabilities rounded to five decimal places where required. It has 8 non-terminal nodes compared to the 4 nodes of the OBDD-Hv, the 10 nodes of the OBDD-He and the 24 nodes of the OBDD-Hve. Each link between nodes is labelled with a comma-separated list of the combination of edges that are available, with an X representing the negative child where either no edges are available or the vertex is unavailable. For example the label 3,23 means that this link is followed if edge e_3 is available and all other edges (in this case only e_2) are unavailable, and also if both e_2 and e_3 are available. The vertex in each grouping is available for each link not marked X.

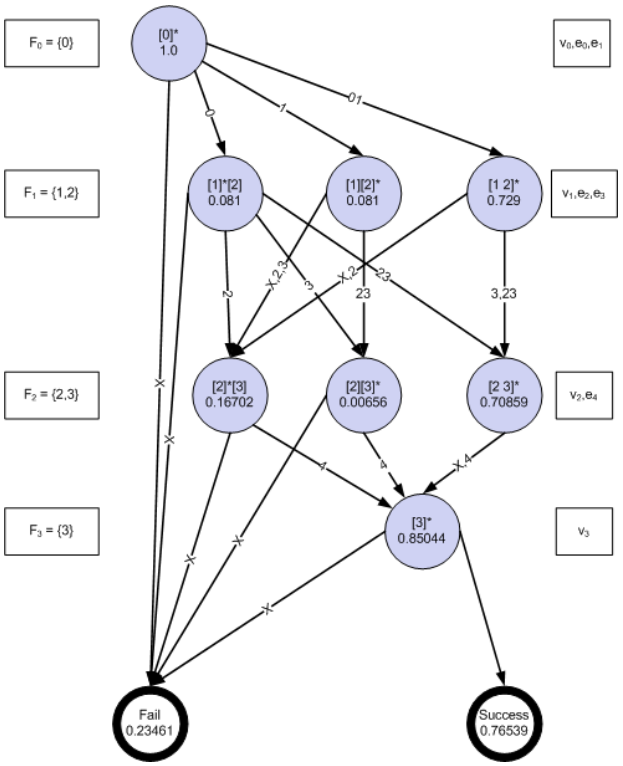

Figure 9: OMDD-H for Sample Network

It can be seen that the nodes of the OMDD-H are equivalent to nodes of the OBDD-Hve on levels that decide vertices. The intermediate nodes on levels of the OBDD-Hve that decide edges are subsumed by the processing of multiple edges on each level of the OMDD-H. Both methods give identical results, but the OMDD-H has fewer nodes.

4.3 The OMDD-H Algorithm

The OMDD-H algorithm given in Figure 8 is closely related to the OBDD-Hv algorithm (Herrmann and Soh, 2009). However instead of one variable being decided per level, an entire grouping of variables is decided.

First the negative child (representing the vertex or all edges in the grouping being unavailable) is created (line 8) by deleting every edge in the grouping. Then the algorithm loops through each combination of available edges and creates the corresponding positive node by contracting every available edge and deleting every unavailable edge (lines 9-10). Note that the combination in which all edges are unavailable is part of the negative child and hence is not considered for the positive loop.

Each non-terminal child created, whether positive or negative, is then compared to the nodes in Q_N. If an isomorphic node is found both are merged. If not the child node is added to Q_N[6] (lines 12-18).

Terminal nodes are never added to the queue. Failure nodes are ignored while success nodes have their probabilities added to REL. When the algorithm terminates, the variable REL contains the appropriate network reliability.

[6] In implementations of OMDD-H it is best to keep Q_N sorted in order to reduce the number of comparisons made. This does not affect the worst-case complexity but does reduce the average processing time.

4.4 The Complexity of OMDD-H

Since the OMDD-H uses partitions of boundary sets, each level is subject to the same bounds as the OBDD-H. Like the OBDD-H, only two levels of nodes are kept in memory at any one time. Hence the time complexity of the OMDD-H is

$$O\left(|V| \times (F_{max})^3 \times B_{F_{max}}\right)$$

and the space complexity is

$$O\left((F_{max})^3 \times B_{F_{max}}\right).$$

It can be seen that these complexities are identical to those of OBDD-Hv.

While this seems to coincides with the experimental results finding that the performance of the OMDD-A is comparable to that of the OBDD-A (Herrmann *et al.*, 2009) it should be noted that the nodes of the OMDD-A may be larger than those of the OBDD-A. By contrast, each OMDD-H node is identical to the corresponding OBDD-H node except that it can have multiple children. Since nodes are never explicitly linked the number of children has no effect on the memory requirements for a node. Hence the sizes of the OMDD-H and OBDD-H nodes are identical.

While the space complexities of OBDD-Hv and OMDD-H are closely related, the time complexities are somewhat misleading. Although the time complexities show that both diagrams process a comparable number of nodes, many OMDD-H nodes processed requires that multiple positive children are created. Many of these will be found to be isomorphic, but the process of creating them and checking for isomorphism must be carried out first. Hence the processing overhead of an OMDD-H node will be greater than that of an OBDD-H node, even if the number of nodes after isomorphism are comparable.

It should be noted that the time complexity for OMDD-H is slightly misleading since it does not take into account that slightly more processing is needed for each positive child compared to OBDD-He, OBDD-Hv and OBDD-Hve.

5 Conclusion

We have described a generalized hybrid ordered binary decision diagram (OBDD-H) method that can be used on networks with edge and/or vertex failures. OBDD-H is shown analytically more efficient than the BS method which is extremely efficient at computing REL for undirected networks with only edge failures. We also proposed the Hybrid Ordered Multi-variate Decision Diagram (OMDD-H) which combines the best features of both the augmented ordered multi-variate decision diagram (OMDD-A) and the boundary set method (BS). The resulting OMDD-H has been shown to have better time complexity than the OBDD-H for networks with fallible devices and links and to have comparable space efficiency. It is thus an extremely appropriate tool for analysing these types of networks.

All of the methods using partitioning of the boundary set (*i.e.*, BS, OBDD-H, OMDD-H) require that the networks be undirected. The less efficient augmented diagrams (*i.e.*, OBDD-A, OMDD-A) have the advantage in that directed networks can be analysed. It would be useful to extend the hybrid diagrams to allow analysis of

directed networks as well, if this can be done without sacrificing performance.

Finally, the OBDD-A and OMDD-A have been shown to be capable of computing metrics other than REL. Experiments will be undertaken to test whether the OBDD-H and OMDD-H can be extended for this purpose.

6 References

Akyildiz, W. S., Su, W., Sankarasubramaniam, Y. & Cayirci, R. (2002). A Survey on Sensor Networks. *IEEE Communications Magazine,* **August:** 102-114.

Carlier, J. & Lucet, C. (1996). A Decomposition Algorithm for Network Reliability Evaluation. *Discrete Applied Mathematics,* **65:** 141-156.

Colbourn, C. J. 1987. *The Combinatorics of Network Reliability,* New York, Oxford University Press.

Hardy, G., Lucet, C. & Limnios, N. (2005). Computing all-terminal reliability of stochastic networks with Binary Decision Diagrams. *In:* 11th International Symposium on Applied Stochastic Models, 2005.

Hardy, G., Lucet, C. & Limnios, N. (2007). K-Terminal Network Reliability Measures With Binary Decision Diagrams. *IEEE Trans. Reliability,* **56:** 506 - 515.

Herrmann, J. (2010). Improving Reliability Calculation with Augmented Binary Decision Diagrams. *In:* AINA, Apr. 2010 Perth, Australia. IEEE.

Herrmann, J., Soh, S. & West, G. (2007). An OBDD Approach for Computing Expected Hop Count of Communication Networks. *In:* PEECS 2007, 2007 Perth, Australia. Curtin University of Technology.

Herrmann, J. U. & Soh, S. (2009). A Space Efficient Algorithm for Network Reliability. *In:* 15th Asia-Pacific Conf. Communications (APCC2009), Oct. 2009.

Herrmann, J. U., Soh, S., West, G. & Rai, S. (2009). Using Multi-valued Decision Diagrams to Solve the Expected Hop Count Problem. *In:* IEEE 23rd Int. Conf. Advanced Information Networking and Applications Workshops, 2009 Bradford, UK. 419-424.

Nagayama, S. & Sasao, T. (2005). On the optimization of heterogeneous MDDs. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* **24:** 1645-1659.

Yeh, F.-M., Lu, S.-K. & Kuo, S.-Y. (2002). OBDD-Based Evaluation of k-Terminal Network Reliability. *IEEE Trans. Reliability,* **51:** 443-451.