International Journal of Performability Engineering, Vol. 8, No. 2, March, 2012, pp. 131-140.

Printed in India

Computing Performability for Wireless Sensor Networks

JOHANNES U. HERRMANN[1,*], SIETENG SOH[1], SURESH RAI[2], and MATJAŽ ŠKORJANC[3]

[1]*Curtin University of Technology, Perth, Australia*
[2]*Louisiana State University, Baton Rouge, L.A., USA*
[3]*University of Maribor, Maribor, Slovenia*

(Received on December 6, 2010, revised on March 25 and May 5, 2011)

Abstract: The performability of a wireless sensor network (WSN) can be measured using a range of metrics, including reliability (REL) and expected hop count (EHC). EHC assumes each link has a delay value of 1 and devices have no delay or *vice versa*, which is not necessarily appropriate for WSNs. This paper generalizes the EHC metric into an expected message delay (EMD) that permits arbitrary delay values for both links and devices. Further, it proposes a method based on Augmented Ordered Multivariate Decision Diagram (OMDD-A) that can be used to compute REL, EHC and EMD for WSN with both device and link failures. Simulation results on various networks show the benefits of the OMDD-A approach.

Keywords: *Expected hop count, expected message delay, multivariate decision diagram, network reliability, sensor network*

1. Introduction

Wireless sensor networks (WSNs) have recently been proposed for a range of applications including military, environmental, and security [13]. The advantages of using a WSN include rapid deployment, co-operation between network nodes and individual nodes being relatively inexpensive [2]. However, wireless sensor devices have limited power and are prone to failure, which means that devices must be deployed densely to allow the network to recover from the loss of one or more nodes. The directed diffusion paradigm [3] includes an event acquisition mechanism that is robust to node and communication failures. Ideally communications should pass through as few nodes as possible in order to conserve power and maximize system responsiveness. Thus the performability of WSNs must be addressed.

AboElFotoh, *et al.* [4] employed the factoring theorem to compute EHC for networks with fallible vertices but perfect edges, and showed that the problem is #P-Hard. An improved method utilizing a Boolean technique was presented by Soh, *et al.* [5]. Recently, Herrmann *et al.* [6] introduced an Augmented Ordered Binary Decision Diagram (OBDD-A) to compute REL and EHC for networks with this failure model. In contrast, Brooks, *et al.* [7] assume networks with fallible edges but perfect vertices, and used random graph models to approximate EHC for mobile WSN. Li, *et al.* [8] extended this failure model for multiple edge states, and proposed a recursive method to compute EHC (called message delay in [8]) for WSN.

One may use the method in [9] to extend the solutions in [4, 5, 7, 8] for networks with both fallible edges and nodes. However, the complexity of these algorithms increases significantly in this case. For example, for a network represented by graph G=(V,E) the method in [5] must compute the metric from $2^{(|V|+|E|)}$ potential paths instead of $2^{|V|}$ and the complexity of the algorithm by Li, *et al.* [8] increases from $2^{|E|}$ to $2^{|V|+|E|}$. Therefore, the

* Corresponding author's email: jherrmann@gmail.com

extension further limits the feasibility of algorithms [5, 8] to compute EHC for larger networks.

This paper generalizes the performability metric EHC to Expected Message Delay (EMD) to permit arbitrary delay for both vertices and edges. EMD is thus a better measure of network goodness than the EHC, which assumes a delay of one for every edge (vertex) and zero for every vertex (edge). Note that the EMD defined in Section 2.3 is different from the message delay addressed in [4, 8]; this message delay is identical to EHC, which is defined in [6].

Table 1: Definition of Key Terms

Term	Definition
Ω	A network state. The special case Ω_g represents a success state in which communication requirements between the source and target vertices of a network have been met.
$D(\Omega_g)$	The delay of X. If X is a component, $D(X)$ is the delay incurred when a message traverses this component. When X is a success state Ω_g, $D(\Omega_g)$ is the least amount of delay experienced between a message leaving a source and arriving at the target given the available minpaths in Ω_g.
e_j	An edge of graph $G = (V,E) - e_j=(v_j,v_t) \in E$.
minpath	A loop-free path of active network devices and communication links (or the vertices and edges representing them) between the source and the target.
Pr (X)	The probability of X occurring. If X is a network state, then $Pr(X)$ is the probability that the network is in that state. If X is a network component (or the graph representation of such a component) then $Pr(X)$ is the probability that the component is available.
reaching path	A loop-free path of active network devices and communication links (or the vertices and edges representing them) between the source and some other device (and vertex).
v_k	A vertex of graph $G = (V,E) - v_k \in V$.
width	The width of a network is defined as $\max_{(v_f,v_t)\in E} \lvert v_t - v_f \rvert$.

Herrmann *et al* [10] proposed an Augmented Ordered Multivariate Decision Diagram (OMDD-A) to compute EHC for networks with fallible vertices and edges. The use of a multivariate decision diagram allows the computation of both vertex and edge failure with a complexity relating to |V|, the number of devices of the network instead of the number of communication links |E|, or |V|+|E|. This paper extends the OMDD-A to compute the EMD.

The layout of this work is as follows: Section 0 introduces the concepts and background for this work. We detail our proposed solution – the OMDD-A for solving EMD in Section 3. We apply the implementation of the OMDD-A algorithm to a number of networks in Section 4 and discuss the results. Finally, Section 5 concludes the paper. For clarity, we list typical notations and their definitions in Table 1.

2. Background

2.1 Network Model and Terminology

We model a WSN using a graph G(V,E), where each vertex in V represents a computer, router or sensor, and every edge in E denotes a wireless communication medium between

the vertices. Communication occurs by a source device (*e.g.*, a sensor) sending a message towards a target device (*e.g.*, a monitoring station). A vertex v_k (or edge e_j) is said to be *active* with probability $Pr(v_k)$ (or $Pr(e_j)$) if it is known to be functioning and *inactive* (or failed) with probability $1\text{-}Pr(v_k)$ (or $1\text{-}Pr(e_j)$) if it is not functioning. We assume that component failures are statistically independent. Due to space restrictions, this paper considers only networks with a single source and a single target.

The network model considers delay values for both edges and vertices. The delay of an edge e_j or vertex v_k (written as $D(e_j)$ and $D(v_k)$, respectively) are small positive integers ($1 \leq D(e_j), D(v_k) \leq 10$). Further, the network vertices are ordered in increasing distance from the source, and edges are ordered in increasing order of their least-valued endpoint [6].

Estimation of component delay is not within the scope of this work. In general, queuing models can be utilized and one may use average delay estimates, obtained over short or long periods. Further details can be found in [11].

2.2 Reliability and Expected Hop Count

Let Ω represent a state of a network $G(V,E)$ when all vertices in $V_\Omega \subseteq V$ and edges in $E_\Omega \subseteq E$ are active and all other vertices and edges are inactive. REL is computed from the set of all success states $\Omega_g \subseteq \Omega$. A state is successful if a path of active vertices from V_Ω and edges from E_Ω connects the source and target. Such a path is called a *minpath*.

In addition to the success state information, computing EHC requires the length of each Ω_g denoted as $1 \leq L(\Omega_g) \leq n\text{-}1$. $L(\Omega_g)$ is the number of edges of the shortest minpath; that is the one with the least edges traversed. EHC is computed as:

$$EHC = \frac{\sum \left(L(\Omega_g) \times Pr(\Omega_g) \right)}{\sum Pr(\Omega_g)} \tag{1}$$

Note that $Pr(\Omega_g) = \prod_{v_i \in V_g} Pr(v_i) \prod_{v_i \notin V_g} (1 - Pr(v_i)) \prod_{e_i \in E_g} Pr(e_i) \prod_{e_i \notin E_g} (1 - Pr(e_i))$, and that the denominator in Eq. (1) is REL. The REL and EHC definitions for fallible edges and vertices are slightly different to those in [6], which are for WSN with fallible vertices only.

2.3 Expected Message Delay (EMD)

EMD and EHC consider both vertices and edges to have delay values; but these values are arbitrary for EMD, while for EHC they are fixed to one for edges and zero for vertices. It is unrealistic to assume that an arbitrary network has equal delays for all edges, and that processing (or buffering) delays in communication devices are negligible.

The EMD of a network is calculated in exactly the same manner as the EHC except for using $D(\Omega_g)$, in place of $L(\Omega_g)$. Define $D(\Omega_g)$, the delay of the network success state Ω_g, as the delay of the shortest minpath; that is the sum of the delays of all edges and vertices in this minpath. Formula (1) is modified for EMD as follows:

$$EMD = \frac{\sum \left(D(\Omega_g) \times Pr(\Omega_g) \right)}{\sum Pr(\Omega_g)} \tag{2}$$

As with EHC, it can be seen that the denominator of (2) is the sum of all success states, and, hence, gives REL. EMD is a generalization of EHC, and the time complexity is at least #P-Hard.

Although (1) and (2) are similar, the use of EHC instead of EMD requires the following assumptions.

(a) The processing time for receiving and passing on the message is equal in all devices of the WSN [4]. This assumption fails when devices have different degree (number of possible other devices broadcasting to them) since this influences the load on the device processor. In a well structured network such as a grid this may be largely true, but in randomly deployed WSN it is generally not the case. In such a case even the average delay may be significantly different.
(b) It is assumed that communication has no delay, or at best that communication between network devices has a constant delay for all devices. This assumption does not allow for different communication media; for example two devices that can communicate either with radio or infra-red signals. When such devices are within line-of-sight they communicate using infra-red but, if this is interrupted, they can fall back to the radio communication mode. In the graph model, this would be represented by two edges between the same vertices, with both edges having different delays and failure probabilities.
(c) Finally, EHC also assumes that communications from different nodes never interfere with each other, which is unrealistic in most routing schemes.

Note that the EMD model assumes each individual delay of a vertex or edge as constant. This is reasonable since computing the *expected* message delay should make use of the *average* delay of each component – and it is this average that we consider to be constant.

2.4 Ordered Multivariate Decision Diagrams

The Ordered Binary Decision Diagram (OBDD) assumes that the variables being decided are binary and are considered one at a time. An Ordered Multivariate Decision Diagram (OMDD) considers multiple variables at a time and thus has fewer nodes as compared to the OBDD [12]. Further, if the variables are binary (such as network devices that are either fully functional or entirely failed) several can be grouped together to form a non-binary variable. See [10] for a detailed comparison between the OBDD and OMDD.

Preliminary work [10] introduced an augmented OMDD (called OMDD-A) to compute REL and EHC for WSN with fallible edges and vertices. However, it does not solve EMD and related issues discussed in Section 3.

3. OMDD-A for Computing REL and EMD

3.1 Introduction

The Augmented OMDD contains network state information stored in each diagram node. The details of the information vary with the application; for REL only the network state and state probability are stored, while for EHC and EMD one requires information on path length stored as well. There are several advantages of storing this information. First, because network state probabilities are stored in each node, the diagram only needs to be traversed once. This means each node can be discarded after it has been processed to create its child nodes. Second, like the OBDD-A [6], this additional network state information allows the computation of other metrics, such as EHC [10].

3.2 The Mathematical Model of the OMDD-A

Let $\Psi(N,G)$ denote an OMDD-A for the graph G(V,E), where N is the set of OMDD-A nodes $\{N_0, N_1, \ldots, N_{2^{|E|}-1}\}$ with N_0 being the root node. $\Psi(N,G)$ is divided into $n=|V|$ levels. Each level j of $\Psi(N,G)$ represents a decision on the state of v_j and all edges grouped with it; we say that a node on this level *decides* variable v_j and call it the *decision variable* (DV) for N_i.

The OMDD-A is an OMDD in which each of the nodes $N_i \in N$ contains a pair $[VI_i, CI_i]$ representing information used to calculate REL and EMD. The *vertex information*, VI_i, is a set of components $\{M^0, M^1, \ldots, M^{|VI|-1}\}$ storing path delay information. Each component $M^x=(\{(v_1, D^x_1), (v_2, D^x_2), \ldots, (v_k, D^x_k)\}, P^x)$ in VI_i contains a probability P^x and a set of ordered pairs of the form (v_a, D^x_a), where D^x_a is the delay of the lowest delay reaching path from v_a to the target vertex v_0. If the set of pairs in M^x contains (v_j, D^x_j) then we write $(v_j, D^x_j) \in M^x$. VI_i has the property that if one component of VI_i contains a pair with vertex v_j then all components of VI_i contain a pair with this vertex.

Given a node N_i, let $VS_i = \{v_a: (v_a, L^x_a) \in M^x \text{ and } M^x \in VI_i\}$ be the set of undecided vertices that have known reaching paths from v_0. As an example, for $N_2=[VI_2=\{(\{(v_1,1), (v_2,1)\}, 0.9)\}, CI_1=\{\ \}]$, we have $VS_2=\{v_1, v_2\}$. Note that decided vertices need not be stored since the position of the node in $\Psi(N,G)$ implicitly encodes all decisions made at higher levels.

Definition: Components M^x and M^y are *equal* ($M^x = M^y$) if for every pair $(v_a, D^x_a) \in M^x$ there exists $(v_a, D^y_a) \in M^y$ such that $D^x_a = D^y_a$.

The *condition information*, CI_i, is a set of conditions $\{C_0, C_1, \ldots C_{|CI|-1}\}$ of the form $C_x=(v_a, v_b, D_x)$ where D_x is the delay of the lowest delay path of active vertices and edges from vertex v_a to vertex v_b. Each condition represents a path through the network that can be taken if its endpoint is reached.

Let $Pr(D)$ denote the probability that a message takes a path with delay D. OMDD-A finds successful components; thus $Pr(D)$ can be calculated. When the generation of OMDD-A nodes is complete all $Pr(D)$ are used to calculate REL and EMD.

3.3 Variable Order and Grouping

The depth of an OMDD can be reduced by using variable partitioning [16]. For the OMDD-A, we group each vertex v_k with its adjacent edges. Both directions of undirected edges are considered at one time; hence, any edge will only be in one variable grouping. 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. Because each edge fails independently of the other edges and of v_k, each node that decides a group consisting of v_k with d adjacent ungrouped edges has 2^d positive (v_k functioning) and 2^d negative (v_k failed).

Each of the 2^d negative child nodes represent identical network states and hence all child nodes will be merged after being created. The OMDD-A generates a single negative child node that represents the result of the merging process and hence generates 2^d positive child nodes and one negative child node, for a total of 2^d+1 child nodes for every parent node.

3.4 OMDD-A Node Type

An OMDD-A node is *terminal* (*non-terminal*) if it does not (does) have children. We process each non-terminal node in a breadth-first fashion to better take advantage of node isomorphism. When a terminal node represents only states that meet the requirements for

the problem, it is a *success* node. The REL and EMD are computed from the reaching path probabilities contained in all success nodes. When the requirements cannot be met from the current state it is a *failure* node, which has no sub-trees containing a success node. Detecting failure nodes earlier avoids generating redundant information. When $VS_i=\varnothing$, node N_i must be a failure node; however, a failure node N_i may have a non-empty VS_i, and detecting such nodes is computationally expensive. Hence the OMDD-A algorithm detects N_i failed only if $VS_i=\varnothing$.

A node N_i for which $VS_i \cap S \neq \varnothing$ (*i.e.* at least one minpath has been found) is not necessarily successful since the EMD calculation requires the shortest path to the target. Because one component of a node might be successful while another is not, individual components are tested for success. A success component representing a state of delay D is removed from the node and its probability is added to $\Pr(D)$.

A component is detected as successful only if it has at least one minpath and if the shortest of these minpaths has delay D, no other reaching paths to a non-target vertex can have delay less than D-1. If a reaching path exists with lower delay it could conceivably reach a target vertex with less delay than the current quickest minpath. The TestNode function in [10] can be used to determine if a node is failed, successful, or non-terminal.

3.5 Node Isomorphism

Definition: Nodes N_i and N_j at the same level in $\Psi(N,G)$ are *isomorphic* if $VS_i=VS_j$ and $CI_i=CI_j$. We write $N_i = N_j$.

Isomorphic nodes have equivalent sub-trees. 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, if $i<j$. When two isomorphic nodes N_i and N_j are merged, the sets of components, VI_i and VI_j, are combined as follows. Every component M^x that is in only one node is present in the merged node with probability unchanged. If $M^x \in VI_i$ and $M^y \in VI_j$ are identical, then the merged node has a component that is identical to M^x but has probability P^x+P^y. Note that since $VS_i=VS_j$ we are guaranteed that for every pair $(v_a, D^x_a) \in M^x$ there exists a pair $(v_a, D^y_a) \in M^y$; for component equality it remains only to compare D^x_a and D^y_a for every $v_a \in VS_i$.

Note that node isomorphism is not affected by the components; hence two nodes that are isomorphic when computing REL may be expected to be isomorphic when computing EHC or EMD. However since the conditions for EHC and EMD have length and delay respectively, some nodes found to be isomorphic for REL will not be found to be isomorphic for EHC and EMD. Hence it is expected that the number of nodes generated for REL is less than the number of nodes generated for EHC and EMD.

3.6 The OMDD-A Algorithm

The OMDD-A algorithm for EMD, shown in Figure 1, is an extension of that given in [10] for computing EHC. A root node is created, representing information on the source vertex with a path of delay 0. This node is added to Q_C, the queue of nodes on level k of the diagram and decision variable k is initialized to 0. Level k of the diagram decides vertex v_k and a subset $E_k \subseteq E$ of edges. Each edge in E_k has the form (v_k, v_x) or $\{v_k, v_x\}$ (*i.e.* it is a directed or undirected edge adjacent to v_k) with $x>k$. Any such edge with $x<k$ will have been already grouped with v_x and an edge with $x=k$ is a self-loop and is ignored since it does not affect either the reliability of delay of the network.

```
1. Create root node N0
2. QC←{N0}, QN←{ }, k←0, and Pr(D) ← 0 (for all D).
3. if QC = { } then
4.         if QN = { }then
5.                 calculate REL and EMD from Pr(D)
6.         else
7.                 QC ← QN, QN ← { } and k ← k + 1.
8. remove the first node Ni from QC.
9. for each combination of unmarked edges (vk,vx), (vx,vk), or {vk,vx}:
10.        create child N based on active and inactive edges
11.        if N is non-terminal then
12.                for each Nq∈ QN do
13.                        if N is isomorphic to Nq then
14.                                merge Nq and 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(D).
20. Modify N for the case where vk is inactive.
21. Repeat steps 11-19 above for this N.
22. goto 3.
```

Figure 1: The OMDD-A function for EMD

Each iteration of the algorithm first checks whether Q_C is empty. If so, k is incremented and the contents of Q_N, the queue for level k+1, are moved onto Q_C. If Q_N is also empty the algorithm terminates. Otherwise the first node is removed from Q_C and processed to give a number of child nodes, which are added to Q_N.

When processing a node N_i, deciding vertex v_k and edges $E_k \subseteq E$ the *positive children*, those with at least one active variable, are created first. The first of these represents the case when v_k is active and all edges in E_k are inactive. The next $2^{|E_k|}-1$ child nodes represent the case when v_k is active and some combination of the edges in E_k are active. The last child node – the *negative child* – represents the case when v_k is inactive as discussed in Section 3.3.

For each positive node, N_j, the state information in the child is updated based on those components that are active. If an active edge $e_l=(v_f,v_t)$ leads to a vertex not yet in VS_i then it is added to VS_j and a condition $(v_f,v_t,D(e_l))$ is added to CI_j. All components of N_i are copied to N_j and the path information in them updated to take advantage of the active edge(s). This step is identical to that in [10] except that paths are updated by adding the delays of the vertex and edge instead of simply increasing their hop count by one.

Once all updates are completed for N_j, all information regarding v_k is removed and no longer needs to be stored. The only exception is if v_k is a target vertex; information on targets must be kept. For the negative child, the only change is the deletion of information on v_k since there are no new paths available. Since the parent node is no longer needed, the negative child is created directly from the parent to save processing time.

When each child node is complete, it is tested to see whether it is a terminal node. A negative child is discarded while a positive child has its component information stored and is then discarded. Non-terminal nodes are compared against all nodes on Q_N; if an isomorphic node exists both are merged and otherwise the child is added to the end of Q_N.

When Q_C and Q_N are empty (indicating that the main loop has completed), the probabilities of each delay, Pr(D), are used to compute REL and EMD as discussed in Section 2.3. Note that if $D(e) = 1\ \forall e \in E$ and $D(v_y)=0\ \forall v \in V$ then the EMD computed is

identical to the EHC for the network, since the EHC is a special case of the more general EMD metric.

4. Simulation Results and Discussions

We have implemented the OMDD-A algorithm in C (compiled using Microsoft Visual Studio 2008) and executed it on a PC (i7 920 2.67GHz processors, 8MB L3 cache, 4GB RAM) on Windows 7. Although our approach can compute REL and EMD of WSN with any vertex and edge success probabilities, we set $Pr(v_k)=Pr(e_k)=0.9$ in these simulations, following accepted practice in the literature. All delays are allocated randomly as numbers between 1 and 8 inclusive using a Perl script. For each simulation, the run time in CPU seconds is averaged over five runs; the performance was similar enough on each run for this to give a suitable indication.

The network files were generated by a Perl script and sorted according to the breadth-first method described in [10] using another Perl script. The majority of the networks chosen are grid networks, which are commonly used to test OBDD algorithms [13-16]. The other networks are networks 18 and 19 from [17], and fully connected networks (K) which are the worst-case for decision diagram methods. The networks are shown in families of similar networks so that the change in performance is clear.

The results of the tests are shown in Table 2. The 'V' and 'E' columns show the number of vertices and edges respectively in the graph of the network tested and the 'W' column shows the width of the network. The number of diagram nodes generated and the time required for the OMDD-A is shown for each metric – REL, EHC and EMD. The processing time (in seconds) and number of nodes are shown for each metric and rounded to two decimal places.

Table 2: OMDD-A Comparison of REL, EHC and EMD

Net	V	E	W	REL		EHC		EMD	
				Time	Nodes	Time	Nodes	Time	Nodes
Path 18	13	22	4	0	278	0.03	1,485	0.16	2,953
Path 19	20	30	6	0.09	4,769	24.30	46,906	40.67	64,471
Grid 2×20	40	58	2	0	134	0.33	476	0.61	476
Grid 2×40	80	118	2	0	274	0.05	1,756	39.31	1,756
Grid 2×60	120	178	2	0	414	0.14	3,836	371.10	3,836
Grid 2×80	160	238	2	0	554	0.38	6,716	1,574.14	6,716
Grid 3×8	24	37	3	0	281	0.06	1,793	1.65	5,456
Grid 3×10	30	47	3	0	369	0.14	3,526	45.08	18,084
Grid 3×12	36	57	3	0	457	0.42	6,111	660.46	43,199
K6	6	15	6	0	222	0	360	0.02	774
K7	7	21	7	0.62	1,206	0.26	2,742	7.66	11,293
K8	8	28	8	1.84	8,207	24.91	27,636	12,289.37	233,255

For comparable sized networks (in terms of the number of nodes and edges), like in other OBDD approaches [13-16], the OMDD-A approach is more efficient when dealing with networks with lower width, W, such as the 2×20 grid network, and is less efficient for network with larger W such as the fully connected networks [18]. For REL, the performance of OMDD-A is not closely related to the number of possible minpaths in the network since only information on connectedness is tracked. However both EHC and

EMD have information on path length/delay, and hence the number of minpaths has a greater impact.

While it may be expected that the number of nodes generated would be the same for REL and the other two metrics, it can be seen that REL has far fewer nodes. As discussed in Section 3.5 this is expected since REL conditions have no lengths and some nodes for EHC and EMD will be found to be non-isomorphic since the condition length differs. Despite this, the performance of REL is much better than that of EHC, which in turn is better than EMD. Both EHC and EMD have components, but EHC has more paths of a similar length, resulting in more merging amongst components. This results in more processing required for EMD which is clearly shown in the processing time.

The EMD may, however, give a far more accurate measurement of network delay than the EHC. For example the EHC of the 2×20 grid network is 21.81 while the EMD for the same network (with random delays) is 119.66. The EHC of a network is only affected by the number of hops, while one or several paths with low delay components can greatly affect the EMD of the network.

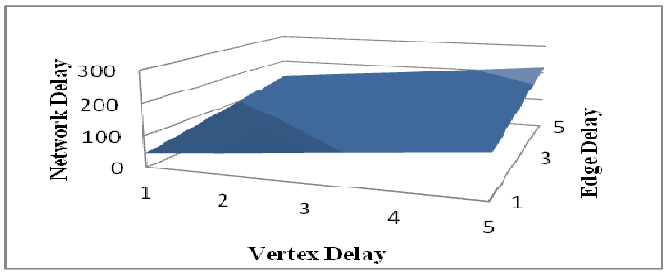

Figure 2: EMD for 2x20 Grid with Varying Delays

Figure 2 shows the EMD of the 2×20 grid network with the delay of each edge and vertex varied from 1 to 5 inclusive. As shown, even a small change in component delay has a cumulative impact on EMD. Note that the relationship between network delay and component delay is a linear one.

5. Conclusions

This paper proposes a technique based on OMDD-A to solve the EMD metric for WSN. EMD provides a more realistic assessment of delay-based network goodness than the EHC metric that had been previously used. It is shown that computing EHC and EMD generate the same number of diagram nodes but that EMD requires more run-time due to a reduced amount of component merging. The OMDD-A algorithm is able to compute EMD for low-width networks of moderate size. Like other decision diagram methods, the OMDD-A approach suffers from decreased performance when network width increases.

Like the OBDD-A, the OMDD-A is applicable for a wide range of networks, *e.g.*, computer communication, road transport and power transmission. Although it has not been addressed in this paper due to space restrictions, the augmented algorithm allows multiple sources and target vertices, and allows the grouping of these vertices. The algorithm allows directed edges as well as parallel edges between two vertices.

References

[1] Soh, S., S. Rai, and R. R. Brooks. *Performability Issues in Wireless Communication Network.* In: K.B. Misra, (Editor). The Handbook on Performability Engineering. London, UK: Springer Verlag; 2008, 1047-67.

[2] Akyildiz, W.S., W. Su, Y. Sankarasubramaniam, and R. Cayirci. *Wireless Sensor Networks: A Survey*. Computer Networks. 2002 March;38:393-422.
[3] Intanagonwiwat, C., R. Govindan, D. Estrin, J. Heidemann, and F. Silva. *Directed Diffusion for Wireless Sensor Networking*. IEEE/ACM Transactions on Networking. 2003 February;11(1):2-16.
[4] AboElFotoh, H.M.F., S. S. Iyengar, and K. Chakrabarty. C*omputing Reliability and Message Delay for Cooperative Wireless Distributed Sensor Networks Subject to Random Failures.* IEEE Trans Reliability. 2005;54(1):145-55.
[5] Soh, S., W. Lau, S. Rai, and R. R. Brooks. *On Computing Reliability and Expected Hop Count of Wireless Communication Networks*. Int'l J Performability Engineering. 2007 April;3(2):267-79.
[6] Herrmann, J. U., S. Soh, S. Rai, and G. West. *On Augmented OBDD and Performability for Sensor Networks*. International Journal of Performability Engineering. 2010 July; 6(4),331-42.
[7] Brooks, R. R., B. Pillai, S. Racunas, and S. Rai. *Mobile Network Analysis Using Probabilistic Connectivity Matrices.* IEEE Trans Systems, Man, and Cybernetics —PART C: Applications and Reviews. 2007; 37(4):1-9.
[8] Li, Y-K., Y. Bai, J. Zhang, Y. Cui, (Editors). *Reliability Computation for Multipath Transmission in Wireless Sensor Networks.* Int'l Joint Conf Computational Sciences and Optimization; 2009; Sanya, Hainan, 694-7.
[9] Aggarwal, K. K., J. Gupta, and K. B. Misra. *A Simple Method for Reliability Evaluation of a Communication System,* IEEE Transactions on Communications, May 1975; 23(5): 563-566.
[10] Herrmann, J. U., S. Soh, G. West, and S. Rai. *Using Multi-valued Decision Diagrams to Solve the Expected Hop Count Problem.* IEEE 23rd Int Conf Advanced Information Networking and Applications Workshops; 2009; Bradford, UK.
[11] Walrand, J., and S. Parekh. *Communication Networks: A Concise Introduction*, Morgan & Claypool, 2010
[12] Nagayama, S., and T. Sasao. *On the optimization of heterogeneous MDDs*. IEEE Trans Computer-Aided Design of Integrated Circuits and Systems. 2005 Nov.;24(11):1645-59.
[13] Hardy, G., C. Lucet, and N. Limnios, (Editors). *Computing all-terminal reliability of stochastic networks with Binary Decision Diagrams.* 11th International Symposium on Applied Stochastic Models; 2005,1469-74.
[14] Hardy, G., C. Lucet, and N. Limnios. *K-Terminal Network Reliability Measures With Binary Decision Diagrams.* IEEE Trans. on Reliability. 2007 Sept.;56(3):506-15.
[15] Kuo, S-Y., S-K. Lu, and F-M. Yeh. *Determining Terminal-Pair Reliability Based on Edge Expansion Diagrams using OBDD*. IEEE Trans. on Reliability. 1999;48(3):234-46.
[16] Yeh, F-M., H-Y. Lin, and S-Y. Kuo, (Editors). *Analyzing network reliability with imperfect nodes using OBDD*. Pacific Rim Int'l Symp Dependable Computing; 2002,89-96.
[17] Soh, S., and S. Rai. *CAREL: Computer Aided Reliability Evaluation for Distributed Computing Networks.* IEEE Trans. on Reliability. 1991;2(2):199-213.
[18] Herrmann, J. U., and S. Soh. *Comparison of Binary and Multi-Variate Hybrid Decision Diagram Algorithms for K-Terminal Reliability.* In M. Reynolds, (Ed.). 34th Australasian Computer Science Conf. (ACSC2011), 2011 Perth, Australia. Australian Computer Society, Inc.; 2011.

For biographies of **Johannes U. Herrmann, and Sieteng Soh,** please refer to Vol. 6, No.4, July 2010, page 342 of International Journal of Performability Engineering.

For biography of **Suresh Rai**, please see page 115 of this issue.

Matjaž Škorjanc is a final year student of Faculty of Electrical Engineering and Computer Science in Maribor. He is about to embark on his diploma, which will be on the parallelization of the OBDD-A or OMDD-A algorithms. His interests are programming and parallel computing.