J. of Mult.-Valued Logic & Soft Computing, Vol. 00, pp. 1–17
Reprints available directly from the publisher
Photocopying permitted by license only

Minimization of Quantum Multiple-valued Decision Diagrams Using Data Structure Metrics

DAVID Y. FEINSTEIN[1], MITCHELL A. THORNTON[1] AND D. MICHAEL MILLER[2]

[1]*Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX, USA*
E-mail: {dfeinste, mitch}@engr.smu.edu
[2]*Department of Computer Science, University of Victoria, Victoria, BC, Canada*
E-mail: mmiller@cs.uvic.ca

Received: July 1, 2008.

This paper describes new metrics for size minimization of the data structure referred to as quantum multiple-valued decision diagrams (QMDD). QMDD are used to represent the matrices describing reversible and quantum gates and circuits. We explore metrics related to the frequency of edges with non-zero weight for the entire QMDD data structure and their histograms with respect to each variable. We observe some unique regularity particular to the methodology of the QMDD. We develop new heuristics for QMDD dynamic variable ordering (DVO) that are guided by the proposed metrics. An exhaustive sifting procedure was implemented for benchmark circuits with up to ten variables to obtain the optimal minimization, demonstrating the effectiveness of the proposed minimization techniques based on data structure metrics.

Keywords: Quantum multiple-valued decision diagrams, quantum logic, reversible logic, sifting, data structure metrics, decision diagram minimization.

1 INTRODUCTION

The *quantum multiple-valued decision diagram* (QMDD) was proposed for the efficient specification and simulation of reversible and quantum circuits [1, 2]. The QMDD data structure has been successfully used for reversible and quantum circuit simulations, equivalence checking, and other applications [3]. In this paper we develop and investigate the use of data structure metrics for the analysis and minimization of QMDD.

Unlike the *binary decision diagram* (BDD) which allows only two possible transition edges from each vertex, and which is used in the QuIDDPro

package [4], a multiple-valued QMDD has r^2 transition edges from each vertex (where r is the radix). Since the transformation matrices representing quantum circuits are often sparse, many QMDD edges point directly to the terminal node with zero weight. This creates an interesting and complex inter-connectivity among the QMDD vertices that allows QMDD to represent relatively complex circuits with a small number of vertices.

The data structure metrics presented here are used to better understand the inter-connectivity characteristics of QMDD. The metrics include the ratio of non-zero weight edges to the number of vertices for the entire QMDD, as well as a histogram (or distribution) of the ratio of the number of non-zero weight edges leaving each variable to the number of vertices measured for each variable. We introduce a set of heuristics based on the values of these metrics to guide *dynamic variable ordering* (DVO) that results in an improvement in minimization for many benchmark circuits.

In our previous work, we have developed a dynamic variable ordering (DVO) sifting algorithm that utilizes an efficient adjacent variable interchange operation [5]. Our sifting algorithm, adapted for QMDD from Rudell's binary ROBDD sifting approach [6], achieved significant size reductions on some benchmarks of reversible/quantum circuits of n-variables while considering only $O(n^2)$ of the $n!$ possible variable orderings.

In this work we offer additional heuristic algorithms for QMDD minimization. The new heuristics achieve improved minimization for some benchmarks over our previous sifting algorithm. Since such DVO minimization efforts consume processing time, it is beneficial to be able to predict in advance whether a certain QMDD structure can be minimized [7]. We describe the heuristics, based on our new data structure metrics, that provide effective prediction in this regard for most of the benchmarks we tested.

The paper is organized as follows. In Section 2 we briefly discuss reversible logic, the QMDD structure, and our previous work on QMDD minimization. The proposed QMDD data structure metrics are described in Section 3. In Section 4 we discuss the application of the proposed new metrics in QMDD minimization heuristics, and in Section 5 we discuss our preliminary experimental results. Conclusions and suggestions for further research appear in Section 6.

2 PRELIMINARIES

2.1 Reversible logic and quantum circuits

In this section, we briefly introduce the basic concepts of reversible and quantum circuits. More extensive background is available in [3].

Definition 1. A gate/circuit is logically *reversible* if it maps each input pattern to a unique output pattern. For classical reversible logic, the mapping is a

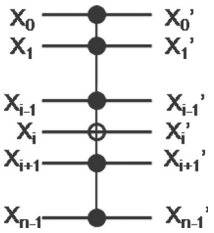

FIGURE 1
n-variable Toffoli Gate.

permutation matrix. For quantum circuits, the gate/circuit operation can be described by a unitary transformation matrix.

Bennet [8] showed that reversible gates can theoretically result in binary circuits that are completely free from energy loss. The concept of reversibility has been extended to MVL circuits [9]. Binary quantum logic gates and circuits are inherently both logically and physically reversible [3]. Non-binary quantum logic circuits are logically reversible.

A variety of basic reversible gates have been proposed in the past few decades. The general n-variable Toffoli gate is shown in Fig. 1. It has $n-1$ control lines (designated by the filled circles) and one target line (designated by the open circle). For each gate, the value on the target line is negated if, and only if, all the $n-1$ control lines are set at '1'. The n-variable Toffoli gate is denoted as TOF($x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n-1}; x_i$), where the target line x_i is separated by a semicolon from the control lines.

Definition 2. An n-variable reversible gate cascade is composed of adjacent reversible gates that operate on the same n variables represented by horizontal lines across the circuit. Each gate may be connected to one or more of the lines and must be extended via the tensor product to affect all n lines.

Cascading of two reversible gates is equivalent to the multiplication of the two permutation matrices representing the corresponding gates. We demonstrate the computation of the transformation matrix on the cascade of four gates shown in Fig. 2.

Each gate is part of a stage of the cascade which must include the gate and all the unconnected lines running through it. For example, gate $G3$, is a reversible NOT gate with a 2×2 transformation matrix. However, in view of the four variables that the circuit depends upon, it must modeled by a 16×16 transformation matrix. We thus extend the reversible NOT operation on variable x_2 into a 16×16 transformation matrix using the tensor product with the 2×2 identity matrix $\mathbf{I}_2$

$$G4 = \mathbf{I}_2 \otimes \mathbf{I}_2 \otimes \mathbf{NOT} \otimes \mathbf{I}_2 \tag{1}$$

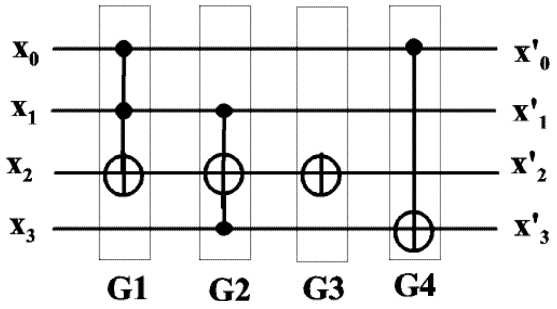

FIGURE 2
A Cascade of 4 Gates with 4 Variables.

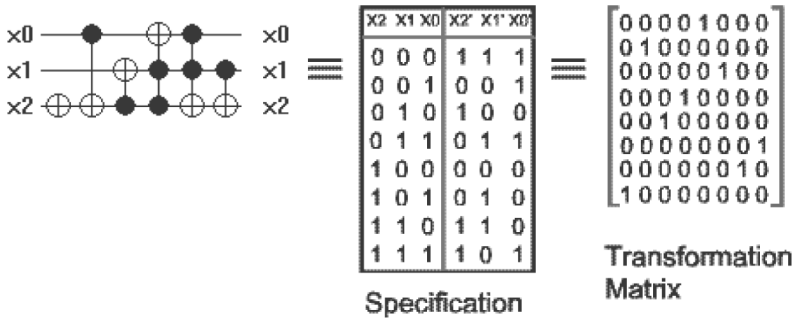

FIGURE 3
3-variable benchmark C3_17.nct.

The overall transformation matrix **C** of this cascade is obtained by multiplying the transformation matrices of the 4 gates in reverse order [10, 11]. Thus,

$$\mathbf{C} = G4 \times G3 \times G2 \times G1 \tag{2}$$

Figure 3 shows a 3-line and 6-gate binary reversible circuit "c3_17.nct" from D. Maslov's benchmarks [12]. The figure also shows the specification and the transformation matrix for the benchmark.

In general, an r-valued reversible circuit with n lines (that is, a circuit with n inputs and n outputs) requires a transformation matrix of dimension $[r^n \times r^n]$, where r is the radix. The exponential growth of the dimensions of the transformation matrix poses a major challenge in the design and simulation of reversible logic and quantum circuits.

2.2 Quantum multiple-valued decision diagrams

The QMDD structure was proposed to simulate and specify reversible and quantum logic circuits in a compact form [1]. A matrix of dimension $r^n \times r^n$

can be partitioned as:

$$M = \begin{bmatrix} M_0 & M_1 & \cdots & M_{r-1} \\ M_r & M_{r+1} & \cdots & M_{2r-1} \\ \vdots & \vdots & \ddots & \vdots \\ M_{r^2-r} & M_{r^2-r+1} & \cdots & M_{r^2-1} \end{bmatrix} \tag{3}$$

where each M_i element is a submatrix of dimension $r^{n-1} \times r^{n-1}$. A QMDD applies this partitioning analogous to how a reduced ordered binary decision diagram (*ROBDD*) [13] recursively applies Shannon decompositions. In a similar manner to ROBDDs, a QMDD adheres to a fixed variable ordering and common substructures (submatrices) are shared. A QMDD has a single terminal vertex with value 1, and each edge in the QMDD, including the edge pointing to the start vertex, has an associated complex-valued weight. Also similar to ROBDDs, the QMDD representation is very useful due to its property of canonicity.

A formal definition for QMDD is provided in Definition 3.

Definition 3. A *quantum multiple-valued decision diagram* (QMDD) is a directed acyclic graph characterized by the following properties:

- There is a single *terminal* vertex with associated value 1. The terminal vertex has no outgoing edges.
- There are some number of *non-terminal* vertices each labeled by an r^2-valued selection variable. Each non-terminal vertex has r^2 outgoing edges designated $e_0, e_1, \ldots, e_{r2-1}$.
- One vertex is designated the *start* vertex and has a single incoming edge that has no source vertex.
- Every edge in the QMDD, including the one leading to the start vertex, is annotated with a complex-valued multiplicative weight value. All with weight of 0 point to the terminal vertex.
- The selection variables are *ordered* (assume with no loss of generality the ordering $x_0 \prec x_1 \prec \cdots \prec x_{n-1}$) and the QMDD satisfies the following two rules:
  - Each selection variable appears at most once on all possible paths from the start vertex to the terminal vertex.
  - An edge from a non-terminal vertex labeled x_i points to a non-terminal vertex labeled x_j, $j < i$ or to the terminal vertex. Hence the vertex with label x_0 is closest to the terminal vertex and x_{n-1} labels the start vertex.

- No non-terminal vertex is *redundant*. *Lack of redundancy means that* no non-terminal vertices exist in a QMDD such that all r^2 outgoing edges have identical weight values and all point to the same vertex.
- Each non-terminal vertex is normalized such that the maximum magnitude of the complex wiegt value is less than or equal to unity.
- All non-terminal vertices are *unique*. Uniqueness means that no two non-terminal vertices labeled by the same variable x_i can have the same set of outgoing edges (destinations and weights).

Theorem 1. *An $r^n \times r^n$ complex-valued matrix M representing a reversible or quantum circuit has a unique (up to variable reordering or relabeling) QMDD representation.*

Proof. A proof by induction based on the iterative construction of a QMDD and the normalization of edge weights that is performed during that construction is detailed in [2]. □

In order to preserve canonicity, QMDDs undergo a normalization procedure whereby the largest valued magnitude of the complex weight value is unity. Normalization is performed by observing the largest valued magnitude edge weight and dividing all edge weights by this value. Because complex numbers can exist with identical magnitude values, a tie is broken by choosing the complex value with a polar angle closest to zero when two values with identical magnitudes occur. This normalization rule correctly normalizes all circuits described by permutation transfer matrices and many complex-valued matrices representing quantum circuits also.

There are some quantum circuits for which the normalization rule does not preserve adjacent variable interchange. As an example, the five-qubit quatum Fourier transform circuit in Fig. 4 does not allow for local adjacent variable swapping. For cases such as these the variable sifting procedure must be adapted to a more general case. In Fig. 4, the gate labeled "H" is a Hadamard gate, the gates labeled "Pn" are phase gates, and those indicated by "X" are generalized swap gates.

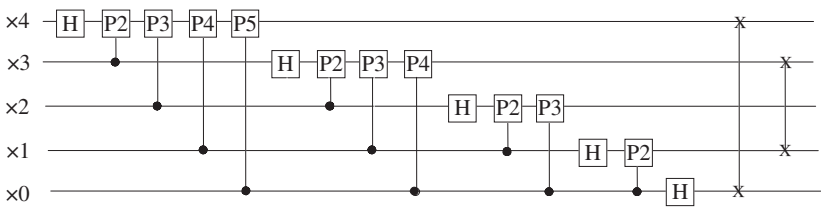

FIGURE 4
5-line Quantum Fourier Transform Circuit.

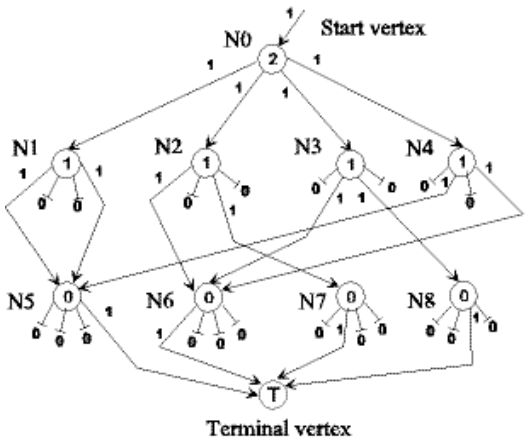

FIGURE 5
The QMDD for benchmark c3_17.nct.

We show in Fig. 5 the QMDD structure of the binary benchmark "c3_17.nct" given in Fig. 3. The full details on the construction and properties of the QMDD are described in [1, 2]. Nine non-terminal vertices are required in addition to the terminal vertex. To make the figure more readable, segmented edges marked with 0 are used to denote edges with zero weight that point to the terminal node.

We note that for binary ($r = 2$) reversible/quantum gates or circuits, the required matrices can be represented using BDDs as is implemented in the QuIDDPro tool [4] which employs the very efficient CUDD package [14]. The QMDD approach is quite different even in the binary case since each nonterminal vertex has four outgoing edges rather than two. This makes each vertex more complex but fewer (about half) vertices are required. In addition, the diagrams typically have half the depth of a BDD-based representation. QMDD are also applicable to multiple-valued situations.

One can verify that this QMDD represents the transformation matrix of Fig. 1, by traversing each path from the start vertex to the terminal vertex and multiplying the weights of the edges. While binary reversible circuits like "c3_17.nct" have only 1 and 0 weights, quantum circuits have complex numbers as edge weights.

2.3 Related work on QMDD variable sifting

In our previous work we demonstrated that local adjacent variable interchange can be efficiently implemented within the QMDD package [5]. We then showed that a minimizing algorithm similar to Rudell's binary ROBDD sifting algorithm [6] is well-suited to reduce the number of nodes in a QMDD representation. Our sifting algorithm employs a greedy selection rule that iteratively picks the variable with the largest number of associated vertices

for sifting. In the event of a tie, the variable closest to the terminal node is selected.

Our QMDD sifting algorithm produced minimization levels that varied significantly from example to example. On some benchmarks (e.g. "rd84d1" and "cycle17_3r") we achieved spectacular size reductions (82% and 92.64% respectively). A low improvement on other benchmarks can be the result of having started from what is already a good ordering, or due to the possibility that the function's QMDD representation is insensitive to variable ordering. Since the sifting heuristic visits only a small fraction ($n^2/n!$) of all possible variable orderings, we are motivated in this paper to explore additional minimization heuristics influenced by the data structure metrics of the QMDD.

3 QMDD DATA STRUCTURE METRICS

The unique table used by the QMDD software continuously maintains and tracks a value called Active[i] that is the number of active vertices for each variable $i, 0 \leq i < n$, where n is the total number of variables. This is an important QMDD data structure metric that we have used extensively in our previous work. We define a new QMDD data structure metric as follows:

Definition 4. Let α be the ratio of the total number of edges with non-zero weight to the total number of vertices in the entire QMDD. Similarly, let $\alpha[i]$ be the ratio of the number of edges with non-zero weight that emerge from all the vertices of variable i to the number of vertices of the same variable. It follows that,

$$\alpha = \frac{\sum_0^{n-1} Active[i] \times \alpha[i]}{\sum_0^{n-1} Active[i]} \tag{4}$$

Since each nonterminal QMDD vertex has r^2 edges, $1 \leq \alpha \leq r^2$ and $1 \leq \alpha[i] \leq r^2$, where r is the QMDD radix. For example, computing $\alpha[1]$ for the QMDD of Fig. 2, we have a total of four vertices and eight non-zero weight edges, thus $\alpha[1] = 8/4 = 2.00$. We observe that while vertex $N1$ has two edges with non-zero weight, they both connect $N1$ to $N5$, essentially connecting the vertex to only one *unique* vertex. We therefore distinguish between the number of edges with non-zero weight and the number of unique *connections* that reach different vertices as follows:

Definition 5. Let β be the ratio of the total number of *unique connections with non-zero weight* to the total number of vertices in the entire QMDD. Similarly, let $\beta[i]$ be the ratio of the number of unique connections with non-zero weight edges that emerge from all the vertices of variable i to the number of vertices of the same variable. As above, $1 \leq \beta \leq r^2$ and $1 \leq \beta[i] \leq r^2$.

It follows that

$$\beta = \frac{\sum_0^{n-1} Active[i] \times \beta[i]}{\sum_0^{n-1} Active[i]} \tag{5}$$

For variable 1 in Fig. 2, we have a total of four vertices and seven unique connections with non-zero weight, thus $\beta[1] = 7/4 = 1.75$. The relation between α and β is given by the following lemma.

Lemma 1. *Let α, β, $\alpha[i]$, and $\beta[i]$ be the data structure metrics of a QMDD with n variables as defined above. Hence, (i) $1 \leq \beta[i] \leq \alpha[i] \leq r^2$ for all $i, 0 \leq i < n$, and (ii) $1 \leq \beta \leq \alpha \leq r^2$.*

Proof. Each vertex of variable i must have at least one edge with non-zero weight, since a QMDD vertex with all edges having zero weights is redundant. This edge creates a unique connection to a nonterminal vertex, thus $1 \leq \beta[i]$ must be true. We observe that the number of unique connections for the vertices of variable i cannot exceed the number of edges with non-zero weight by Definition 3. Thus $\beta[i] \leq \alpha[i]$ must be true.

Let Active$[i]$ denote the number of active vertices for variable i, and E the total number of the QMDD's edges with non-zero weight, and C the total number of unique connections. Then

$$E = \sum_0^{n-1} Active[i] \times \alpha[i]$$

and

$$C = \sum_0^{n-1} Active[i] \times \beta[i]$$

Since we have proven that $\beta[i] \leq \alpha[i]$, it follows that $C \leq E$ and

$$\beta = \frac{C}{\sum_0^{n-1} Active[i]} \leq \frac{E}{\sum_0^{n-1} Active[i]} = \alpha.$$

Clearly, $\alpha[i] \leq r^2$, with the equality occurring when all edges have non-zero weights. The proof for (ii) follows similar arguments. □

Listing the metrics for QMDD metrics in tabular form provides a histogram-like display of the structure of the QMDD which we will, for convenience, refer to as a histogram. Table 1 shows the histogram for the QMDD in Fig. 2.

The bottom line of the histogram provides the overall α, β and number of vertices for the entire QMDD. Note that the sum of Active$[i]$ does not include the terminal node. The histogram lists the variables according to the *current variable order* of the QMDD. In Table 1, variable 2 labels the start node and variable 0 is the closest variable to the terminal node. As a result, variable $n - 1$, the start node, always has Active$[n - 1] = 1$.

Variable	Active[i]	$\alpha[i]$	$\beta[i]$
2	1	4.00	4.00
1	4	2.00	1.75
0	4	1.00	1.00
Overall	9	1.78	1.67

TABLE 1
Histogram for "3_17.nct"

Var	Act[i]	$\alpha[i]$	$\beta[i]$
11	1	4.00	4.00
10	4	4.00	4.00
9	16	4.00	4.00
8	64	4.00	4.00
7	256	3.91	3.91
6	990	2.84	2.84
5	2258	1.37	1.37
4	1174	1.17	1.17
3	304	1.16	1.16
2	76	1.16	1.16
1	19	1.21	1.21
0	4	1.00	1.00
Overall	5167	1.76	1.76

TABLE 2
Histograms for "hwb12"

BDD researchers have observed that the numbers of vertices for each variable (arranged by the variable order) in general exhibit a pear-shaped pattern [15]. We have observed that while most QMDD exhibit similar pear shaped histograms, many histograms exhibit quite a flat pattern, where most variables have roughly the same Active[i] *values*. This allows us to classify QMDD histograms as "FLAT" or "PEAR", as demonstrated in the two histograms of Table 2. The left histogram of "hwb12" exhibits the typical PEAR like pattern commonly observed in classical BDDs. The right histogram of benchmark "cycle10_2*" is of type FLAT as variable 7 to 2 have similar numbers of vertices. We later show that different minimization heuristics apply based on this classification.

The $\alpha[i]$ and $\beta[i]$ histograms provide additional insight that is explored in the following Section. It should be noted that the QMDD unique table provides efficient means to compute the Active[i] as well as the $\alpha[i]$ and $\beta[i]$ histograms without much processing overhead.

var	act[i]	$\alpha[i]$	$\beta[i]$
11	1	2.00	2.00
10	2	4.00	3.00
9	6	1.67	1.67
8	10	2.00	1.60
7	16	1.44	1.19
6	16	1.44	1.19
5	16	2.00	1.56
4	18	1.50	1.17
3	18	2.00	1.00
2	15	1.20	1.00
1	9	1.33	1.00
0	3	1.33	1.00
Overall	131	1.65	1.24

TABLE 3
Histograms for "cycle10_2*"

4 IMPROVING QMDD MINIMIZATION WITH THE DATA STRUCTURE METRICS

The inability of our sifting algorithm (as well as any other similar heuristic algorithm) to achieve consistent positive results with all the benchmarks, is of course, due to the fact that it examines only $O(n^2)$ ordering possibilities out of $n!$ possible orderings. To achieve improved minimization, common binary BDD packages offer a rich choice of reordering heuristics. These heuristics can be grouped into sifting algorithms (that include also group sifting and symmetric sifting), random algorithms, window algorithms, and other special algorithms (including simulated annealing and genetic evolutionary algorithms) [7].

While all of these heuristics can be extended for QMDD, we concentrate here on heuristics primarily based on sifting and random approaches. Our experimentation with the group sifting approach that exploits the affinity among variables achieved limited success. One reason is that benchmarks with sufficiently large numbers of variables are currently not available. Furthermore, it seems that since reversible circuits are maximally connected [3, 16], they tend to display a strong group behavior as a whole, making attempts to identify subgroups more difficult or redundant.

Our random algorithm performs a set of random variable position changes without regard to structure size between repeated applications of the sifting algorithm. This process breaks apart the initial grouping of variables allowing the sifting algorithm to find a better ordering. A set of heuristics inferred

Algorithm 1: QMDD Metrics Based Minimization

(i) Apply the QMDDsift for initial reordering.
(ii) Use heuristics to predict if a better variable ordering is likely. If prediction is negative – stop the procedure.
(iii) Perform a random set (or a specially designed fixed set) of variable rearrangements *without regard* to structure size.
(iv) Apply the QMDDsift for smoothing.
(v) Repeat steps (ii) to (iv). Stop the process if this step is repeated more than k times.

from the data structure metrics is then used to determine when the process should stop.

We present a sample set of the data structure metrics heuristics that have been investigated. Like all heuristics, they obtain different levels of success as described in the following section, hence the k limit in step (iv).

Sample Heuristics:
Heuristic 1: For PEAR type benchmarks, if the sifting algorithm provides moderate or no improvement and if the pattern of the histogram of $\alpha[i]$ and $\beta[i]$histogram is not changed by the sifting, then no significant improvement is likely to be achieved.

Heuristic 2: For FLAT type benchmarks, no significant improvement is likely if the first half of the variables exhibit $\alpha[i] \approx 2.00$.

Heuristic 3: Appearance of $\alpha[i]$ or $\beta[i]$ equal to 2 in the first two variables (of the variable ordering) of a PEAR type benchmark suggests benefit for further reduction attempts, as the QMDD may become FLAT.

Heuristic 4: Benchmarks with $\alpha[i]$ that is not monotonically decreasing (along variable order from the start vertex) resist minimization.

Heuristic 5: If reordering of a FLAT Benchmark changes its histogram shape to PEAR, the total number of vertices will usually increase.

5 EXPERIMENTAL RESULTS

We summarize our results for a set of benchmark circuits in Tables 4 and 5. The PEAR type circuit benchmarks are shown in Table 4 and the FLAT benchmarks appear in Table 5. The second column indicates the histogram type (PEAR or FLAT) as described in Section 3. Columns 6 and 7 depict the minimization results and minimization time obtained with the general sifting algorithm

Name	Hist type	vars	gates	Initial QMDD vertices	Sifting reduction	Sifting time (S)	New algorithm reduction	Number of iterations	Time (S)	Improvement	Exhaustive reduction	Exhaustive time (S)
5mod5	Pear	6	17	28	42.86%	0.032	46.43%	10	0.43	8.33%	46.43%	0.452
mod5adders	Pear	6	21	38	0.00%	0.031	0.00%	5	0.41	0.00%	2.63%	0.453
ham7	Pear	7	23	130	3.08%	0.047	24.62%	10	0.65	699.35%	24.62%	3.791
hwb7	Pear	7	289	179	13.41%	0.047	13.41%	5	0.26	0.00%	13.41%	3.947
rd53d1	Pear	7	12	26	19.32%	0.047	19.32%	5	0.30	0.00%	19.23%	3.401
hwb8	Pear	8	614	343	18.37%	0.11	18.37%	5	0.75	0.00%	18.37%	37.815
hwb9	Pear	9	1541	683	23.87%	0.125	23.87%	5	0.65	0.00%	23.87%	429.109
hwb10	Pear	10	3631	1331	27.87%	0.203	29.30%	5	0.70	5.13%	29.30%	5840.068
6symd2	Pear	10	20	247	50.20%	0.094	56.68%	10	1.20	12.91%	56.68%	3853.705
hwb11	Pear	11	9314	2639	34.44%	0.421	34.44%	5	2.10	0.00%	–	–
hwb12	Pear	12	18393	5167	38.36%	0.795	39.40%	5	4.40	2.71%	–	–
9symd2	Pear	12	28	229	19.65%	0.125	48.19%	10	1.35	145.24%	–	–
ham15	Pear	15	132	4521	49.56%	0.920	72.62%	10	9.80	46.53%	–	–
ham15m	Pear	15	132	1774	19.17%	0.561	36.30%	5	3.40	89.36%	–	–
ham15r	Pear	15	132	4522	60.59%	0.733	73.33%	10	8.30	21.03%	–	–
rd84d1	Pear	15	28	3588	92.64%	0.250	92.64%	5	1.40	0.00%	–	–
rd73d2	Pear	15	28	3588	14.00%	0.093	40.76%	10	1.05	191.14%	–	–

TABLE 4
QMDD Minimization Results for PEAR Type Binary Benchmarks

Name	Hist type	vars	gates	Initial QMDD vertices	Previous sifting reduction	Sifting time (S)	New algorithm reduction	Number of iterations	Time (S)	Improvement
cycle10_2	Flat	12	19	67	40.30%	0.093	62.69%	10	1.20	55.56%
0410184	Flat	14	46	39	15.38%	0.125	15.38%	5	0.75	0.00%
0410184.qc	Flat	14	74	39	15.38%	0.156	15.38%	5	0.80	0.00%
cycle17_3	Flat	20	48	236	57.20%	0.359	82.20%	10	3.90	43.71%
cycle17_3r	Flat	20	48	236	82.20%	0.359	82.20%	5	1.75	0.00%
0406142c2	Flat	35	116	150	9.33%	0.733	9.33%	5	4.20	0.00%

TABLE 5
QMDD Minimization Results for FLAT Type Binary Benchmarks

described in [5]. Columns 8–10 shows the results obtained using metric-based minimization. Column 8 shows the overall minimization result, Column 9 shows the number of iterations guided by the metrics and Column 10 shows the approximate process time (the routine are not yet optimized for accurate timing measurements). We set the maximum iteration number k of Algorithm 1 to 10, and we use early exit threshold value of 5 iterations if the heuristics indicate that further minimization is unlikely. Column 11 indicates the improvement (if any) of the new algorithm over the general sifting algorithm.

For purposes of evaluating the heuristics, we exhaustively compute all $n!$ permutations of the variable ordering for benchmarks with up to 10 variables to choose an optimal variable ordering. The overall minimization appears in column 12 and the time in column 13. Since all our FLAT type benchmark circuits have more than 10 inputs, exhaustive results do not appear in Table 5.

We observe significant improvements with the benchmarks "9symd2", "cycle10_2", "ham7", "ham15m" and "rd73d2". We recorded the QMDD histogram during the entire minimization process. An interesting observation for most benchmarks is that the initial QMDD buildup process produces a monotonically decreasing $\alpha[i]$ histogram. On the other hand, the $\beta[i]$ histogram may exhibit fluctuations, although the values are generally decreasing in view of Lemma 1. The only observed violation of the monotonic decrease of $\alpha[i]$ appears with benchmark "mod5adders". Since this benchmark seems to resist minimization, we use this finding in Heuristics 4.

The "hidden weighted bit" benchmarks ("hwb7", "hwb8", "hwb9", "hwb10", "hwb11", and "hwb12") worked very well with Heuristic 1. In its reversed application, it showed particularly good improvements for "rd73d2", "ham15r", "ham15".

Heuristic 2 yields acceptable results but we note that there are few FLAT type benchmark circuits available to allow for more extensive verification. A similar situation occurs with Heuristic 3.

We also note that setting the parameter k to 10 iterations and the exit point to 5 iterations essentially fixes the time of the proposed algorithm to about 10 or 5 times the sifting algorithm runtime. Reported times in these results are approximate as our routines are not yet optimized. Further improvements of the heuristics accuracies can allow early exit in Algorithm without any iterations (or fewer than 5 iterations), thus reducing the time overhead of the proposed approach.

To further demonstrate Heuristic 5, we subjected the benchmark "cycle10_2" to several random variable reorderings, and captured histograms denoted as random0, random1 and random2. We compared the Active[i] values of these histograms with the minimized histograms obtained by our sifting algorithm and the new algorithm described in this paper. We show our results in Fig. 6, where we also include the original ordering histogram. Each series in the graph illustrates the Active[i] (number of vertices) for each variable. The variable number is assigned from the start node (1) to the last variable (12) as traversed along the QMDD variable order.

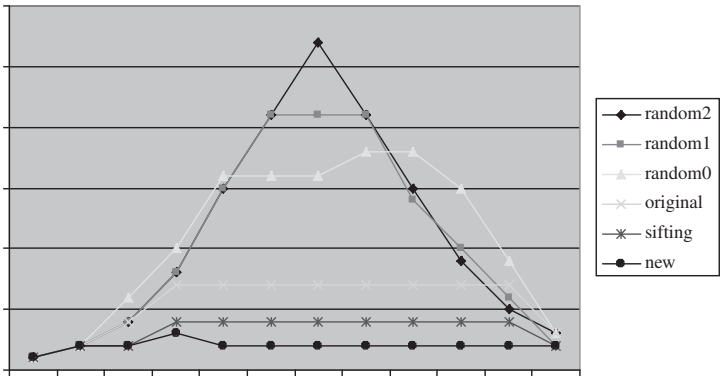

FIGURE 6
Shape changes from FLAT to PEAR for benchmark Cycle10_2.nct

The total number of vertices for random2, random1, and random0 are 132, 126 and 131 respectively. From Table 3, the total number for the original variable order is 67, and for the minimized sifting and new algorithm it is 40 and 25 respectively. We can see that the shape changes from FLAT to PEAR for random2 and random1. Clearly, these variable orders cause the QMDD to use a much higher number of vertices than in our minimized variable orders. Therefore, a minimized flat benchmark must remain flat during further minimizations.

6 CONCLUSIONS

This paper proposes new data structure metrics that provide insight on the inter-connectivity within a QMDD. Heuristics based on these metrics have been developed and used to guide an iterative minimization algorithm. Our preliminary minimization results show some significant improvement versus our sifting algorithm, although no improvement was achieved for some benchmarks. For benchmark circuits of 10 variables or less, our results match or came very close to the optimal results.

The potential of the data structure metrics based heuristics had been demonstrated in this paper. We recognize that the heuristics that perform well with our current benchmarks may err with future benchmarks.

ACKNOWLEDGEMENTS

A preliminary version of this paper was presented at the 2008 IEEE International Symposium on Multiple-Valued Logic [17].

REFERENCES

[1] D. M. Miller and M. A. Thornton. QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits. In *Proceedings of the IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, on CD, May 17–20, 2006.

[2] D. M. Miller, M. A. Thornton and D. Goodman. A Decision Diagram Package for Reversible and Quantum Circuits. In *Proceedings of the IEEE World Congress on Computational Intelligence*, on CD, July 2006.

[3] D. Y. Feinstein, M. A. Thornton and D. M. Miller. Partially Redundant Logic Detection Using Symbolic Equivalence Checking in Reversible and Irreversible Logic Circuits. In *Proceedings of the IEEE/ACM Design, Automation and Test in Europe (DATE)*, March 10–14, 2008, pp. 1378–1381.

[4] G. F. Viamontes, I. L. Markov and J. P. Hayes, QuIDDPro: High-Performance Quantum Circuit Simulation. http://vlsicad.eecs.umich.edu/Quantum/qp/, Oct. 20, 2006.

[5] D. M. Miller, D. Y. Feinstein, and M. A. Thornton. QMDD Minimization using Sifting for Variable Reordering. In *Journal of Multiple-valued Logic and Soft Computing*. 2007. pp. 537–552.

[6] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Proceedings of the International Conference on Computer-Aided Design*. Santa Clara, CA, November 1993, pp. 42–47.

[7] S. N. Yanushkevitch, D. M. Miller, V. P. Shmerko and R. S. Stankovic. *Decision Diagram Techniques for Micro- and Nanoelectronic Design*. CRC Taylor and Francis, 2006.

[8] C.H. Bennett. Logical Reversibility of Computation. *IBM, J. Res. Dev.* **17**(6) (1973), 525–532.

[9] D. M. Miller, G. Dueck and D. Maslov. A Synthesis Method for MVL Reversible Logic. In *Proceedings of the 2004 Int. Symposium on Multiple-Valued Logic*. Toronto, Canada, May 2004, pp. 74–80.

[10] D. C. Marinescu and G. M. Marinescu. *Approaching Quantum Computing*. Pearson Prentice Hall, 2005.

[11] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[12] D. Maslov. Reversible logic synthesis benchmarks page. http://www.cs.uvic.ca/∼dmaslov/, November 15, 2005.

[13] R. E. Bryant. *Graph-Based Algorithms for Boolean Function Manipulation*. *IEEE Transactions on Computers* **C-35**(8) (1986), 677–691.

[14] F. Somenzi. *The CUDD Package*, University of Colorado at Boulder, 1995. Version 2.4.0 available at: http://vlsi.colorado.edu/∼fabio/.

[15] S. Panda and F. Somenzi. Who are the variables in your neighborhood. In *Proceedings of the ICCAD' 95*, December 1995, pp. 1–4.

[16] V. D. Agrawal. An information theoretic approach to digital fault testing. *IEEE Trans. Computers* **30** (1981), 582–587.

[17] D. Y. Feinstein, M. A. Thornton and D. M. Miller. On the Data Structure Metrics of Quantum Multiple-Valued Decision Diagrams. In *Proceedings of the IEEE International Symposium on Multiple Valued Logic (ISMVL)*, May 22–23, 2008, pp. 138–143.