Acta Cybernetica **23** (2017) 503–512.

Storing the Quantum Fourier Operator in the QuIDD Data Structure*

Katalin Friedl[a] and László Kabódi[a]

Abstract

Quantum algorithms can be simulated using classical computers, but the typical time complexity of the simulation is exponential. There are some data structures which can speed up this simulation to make it possible to test these algorithms on classical computers using more than a few qubits. One of them is QuIDD by Viamontes et al., which is an extension of the Algebraic Decision Diagram.

In this paper, we examine the matrix of Fourier operator and its QuIDD representation. To utilize the structure of the operator we propose two orderings (reversed column variables and even-odd order), both resulting in smaller data structure than the standard one. After that, we propose a new method of storing the Fourier operator, using a weighted decision diagram that further reduces its size. It should be the topic of subsequent research whether the basic operations can be performed efficiently on this weighted structure.

Keywords: quantum algorithms, quantum Fourier operator, QuIDD

1 Introduction

Simulating quantum algorithms on a classical computer is a hard problem because of an exponential slow down. A quantum operator on n qubits can be represented on a classical computer by a $2^n \times 2^n$ matrix, so operations have exponential time and space complexity. In some cases, this can be decreased by having a good data structure. The Quantum Information Decision Diagram proposed by Viamontes, Markov, and Hayes [8] can store certain quantum operators in polynomial space, and compute some operations in polynomial time. They also have a quantum circuit simulator called QuIDDPro that employs their data structure [7]. Viamontes et al. [8] illustrated the use of the data structure on Grover's search [5]. In [4], using the different groupings of the operators, we gave an improved bound on the overall complexity of the simulation.

*This study is partially supported by the National Research, Development and Innovation Office (NKFIH), by the OTKA-108947 grant.

[a]Budapest University of Technology and Economics, Department of Computer Science and Information Theory E-mail: {friedl, kabodil}@cs.bme.hu

DOI: 10.14232/actacyb.23.2.2017.5

In this study, we focus on one of the most important quantum operators, namely the quantum Fourier operator. Using the structure of the operator, we present two natural reorderings of the columns that make the QuIDD representation of the operator smaller. Also, we propose a modification of the data structure to further reduce the number of nodes.

Although the motivation of this study is the efficient simulation of quantum algorithms, the problem itself is not limited to this area. The size of Ordered Binary Decision Diagrams (OBDDs) is an intensively researched topic. It has been proven that finding the optimal ordering for a Boolean function is NP-hard [3], but heuristics are useful, because in many areas (e.g. verification, model checking and computer aided design [9]) the size of the OBDD determines the performance of the applications. In some applications the width of the diagram is an important factor, but in this study we only concentrate on the number of nodes. It is known that the difference between the sizes for two orderings may be exponential, as in the case of the most significant bit of binary addition [2]. Our goal here is to analyze the efficiency of some orderings based on the structure of the operator.

Section 2 provides a description of the QuIDD data structure. After, Section 3 compares the number of nodes in the standard and modified orderings. Then Section 4 presents an idea of how to modify the data structure by weights to use significantly less nodes to store the operator. Lastly, in Section 5 we draw some pertinent conclusions and make a suggestion for future study.

2 The QuIDD data structure

The QuIDD data structure was specifically developed for quantum simulations [8]. For completeness, here we describe this data structure in details, based on [8, 4].

2.1 Binary Decision Diagram

For the representation of a Boolean function, the Binary Decision Diagram (BDD) is a popular tool [6]. It is a rooted directed acyclic graph, where every non-leaf node has exactly two child nodes. Each node is labelled with a variable of the function and the edges represent the 0 and 1 value of that variable. The value of the function is obtained by traversing the tree from its root following the edge representing the value of the variable. The value of a leaf is the value of the function. In the Ordered BDD the variables follow each other in a preset order. One of the problems of this representation is that it needs 2^n leaves and $2^n - 1$ internal nodes to store a function with n variables. The Reduced Order BDD (ROBDD) introduces reduction rules to achieve the following properties:

1. There are no nodes v and v' such that the subgraphs rooted at v and v' are isomorphic.

2. There is no internal node with both its edges pointing to the same node.

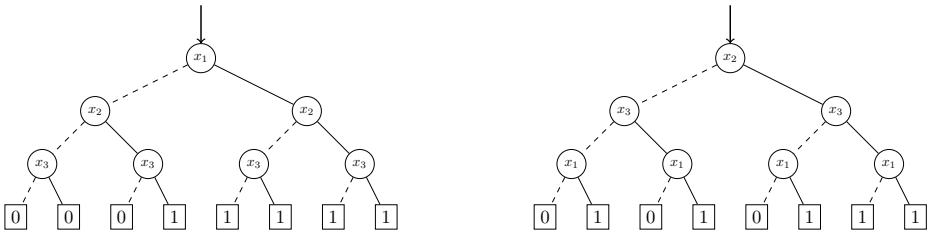

Figure 1: The BDD of function f using orderings x_1, x_2, x_3 and x_2, x_3, x_1. The solid lines represents the 1 edges, the dashed lines the 0 edges.

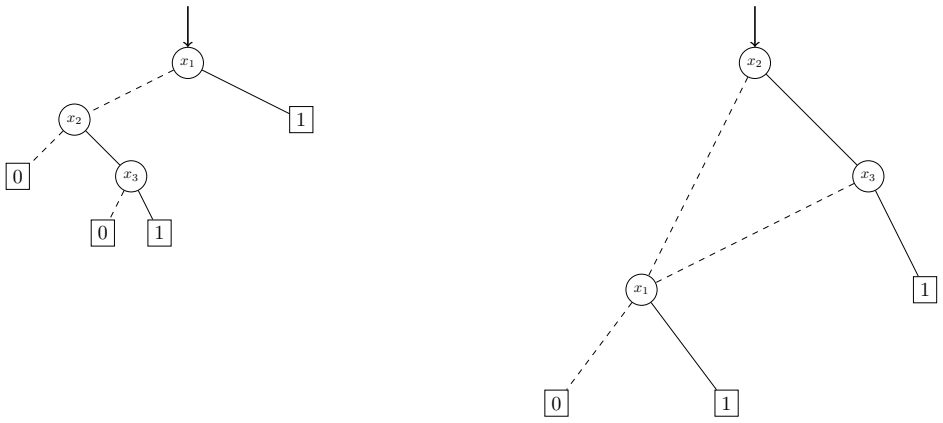

Figure 2: The ROBDD of the same function using the same orderings.

To illustrate the difference between the two structures, figures 1 and 2 show the BDD and ROBDD representations of the function

$$\begin{aligned} f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee \neg x_3) \wedge \\ \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge \\ \wedge (x_1 \vee x_2 \vee x_3) \end{aligned}$$

with two variable orderings. As can be seen, the ROBDD can use significantly fewer nodes than the BDD and its size depends on the ordering.

Although a typical ROBDD is no longer a tree, the words root, leaf are still used for the corresponding nodes.

2.2 Quantum Information Decision Diagram

The QuIDD is a variant of the Algebraic Decision Diagram (ADD) [1], which is based on ROBDD. It is designed to store a matrix with complex elements. The leaves in the QuIDD are pointers to an array, which stores the values of nodes. These values may be real-valued or complex-valued.

When a matrix is stored, the variables of QuIDD correspond to the bits of the binary representation of rows and columns of the matrix. The variable R_i represents the i-th bit of the row numbers, and C_j represents the j-th bit of the column numbers. The numbering starts with 0, the 0th bit being the most significant one. The ordering of the variables is: rows and columns interleaved. This ordering is helpful if the matrices have some block structure, as usually happens when they are constructed from smaller matrices by tensor products.

The number of leaves is the same as the number of different values in the matrix. Let the *size* of a QuIDD be the number of its non-leaf nodes.

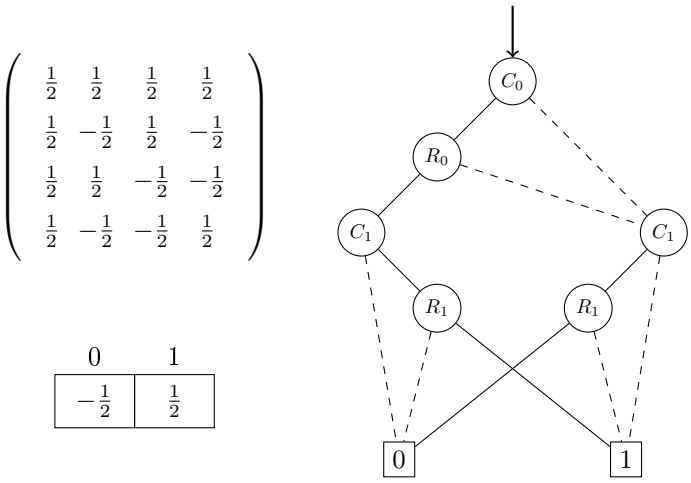

Figure 3: The Hadamard matrix acting on two qubits, and its QuIDD representation.

Example. On the diagram of figure 3, for the third (11) element of the second (10) row, one has to choose the 1 edge (solid line) at C_0, at R_0, and also at C_1, while the last step is the 0 edge at R_1. The 0 at the leaf leads to the value $-\frac{1}{2}$.

The 0th column of the matrix is constant so we can follow the dashed edge from C_0 to C_1 and to the leaf, where the 1 means that the value of the element is $\frac{1}{2}$. Notice that on this path there is no need for R_i nodes, because the value does not depend on the row.

In the following examples, for simplicity, a leaf contains the corresponding element of the matrix instead of a pointer to that element.

This data structure not only stores matrices, but computations can also be performed in this form. The basic operations of two matrices can be calculated in polynomial time and the result has a polynomial size. Namely, if there are two matrices with sizes a and b then their sum has size $O(ab)$, their product $O((ab)^2)$, and their tensor product $O(ab)$.

3 Storing the quantum Fourier operator using the QuIDD data structure

First, we examine the quantum Fourier operator using the standard QuIDD ordering. Unfortunately, in this way the recursive structure of the Fourier operator is not used. Then two other orderings are considered resulting in smaller sizes.

Proposition 1. *Without the reduction rules, the size of the Fourier operator is* $2^{2n} - 1$ *and it has* 2^n *leaves.*

3.1 Standard ordering

Recall that the standard ordering is $C_0, R_0, C_1, R_1, \ldots, C_{n-1}, R_{n-1}$.

Notice that every node in the structure corresponds to some contiguous submatrix formed by some neighboring rows and some neighboring columns of the matrix.

Proposition 2. *A node* C_i *describes a submatrix of size* $2^{n-i} \times 2^{n-i}$*. This submatrix can be obtained by fixing the first* i *bits of the row number and column number. A submatrix of node* R_i *is similar, but there the first* i *bits of the row number and the first* $i-1$ *bits of the column number are fixed. This submatrix has size* $2^{n-i+1} \times 2^{n-i}$*.*

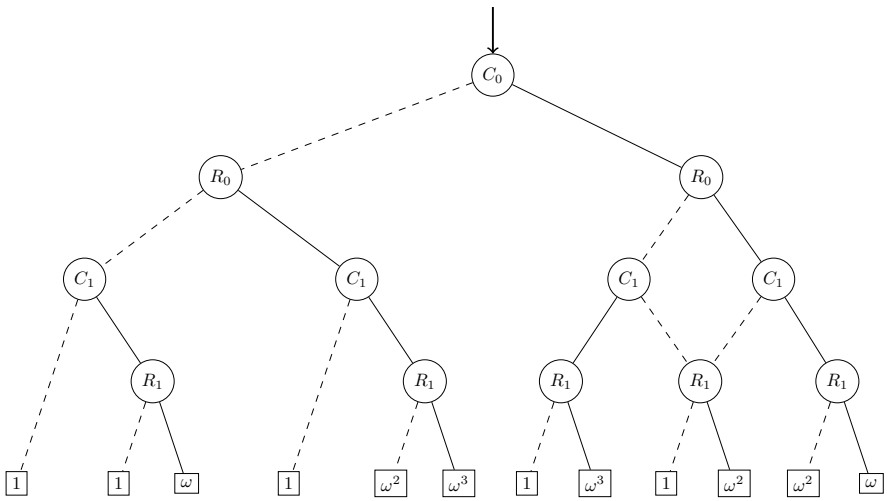

Figure 4: The Fourier operator of 2 qubits using the standard ordering.

Theorem 1. *Using the standard ordering, the size of the Fourier operator is* $\frac{5}{6} \cdot 2^{2n} - \frac{4}{3}$.

Proof. Let ω be a primitive 2^nth root of unity. In the Fourier transform the elements in the ith column are $\frac{1}{2^n} \cdot \omega^{k \cdot i}$, where k is the row number. When i is odd, all the different 2^n powers of ω appear. Since in neighboring rows the difference between the exponents in the same column is i, there are no two identical contiguous submatrices of size at least 2×2. This means that in QuIDD, identical nodes can appear only at the bottom. At the bottom, a node R_{n-1} corresponds to a 2×1 contiguous submatrix. From the above reasoning, these are different submatrices when the column index i is odd. Similarly, they are different when they come from different columns.

Let $i = 2^r \cdot a$ where $r \geq 1$ and a is odd. In this case the column is periodical with length of 2^{n-r}. Because an R_{n-1} node stores two elements, one column uses 2^{n-r-1} nodes.

For a fixed r there are 2^{n-r-1} possible a's. So at the bottom of the tree, the even columns use $\sum_{r=1}^{n-1} 2^{n-r-1} \cdot 2^{n-r-1} = \sum_{s=0}^{n-2} 4^s = \frac{4^{n-1}-1}{3} = \frac{1}{3}(\cdot 2^{2n-2} - 1)$ nodes.

The odd columns are all different, hence they use 2^{2n-2} nodes.

Since all 2×2 submatrices are different, the tree is complete up to the last level, this upper part having $2^{2n-1} - 1$ nodes. Adding it all up, we get $2^{2n-1} - 1 + \frac{1}{3}2^{2n-2} - \frac{1}{3} + 2^{2n-2} = \frac{5}{6}2^{2n} - \frac{4}{3}$. □

3.2 Different orderings

The study of Viamontes et al. [8] uses the standard ordering, but in general, the data structure works when all the matrices use the same ordering, which is not necessarily the standard one.

In the even numbered columns of the Fourier transform, the first 2^{n-1} rows are the same as the remaining ones. Moreover, they are the same as the Fourier operator of $n-1$ qubits. The following orderings are based on this observation.

Let us reverse the ordering of the column variables. In this ordering, we first check the last bit of the column, then the first bit of the row, and so on. Using the previous notation the ordering of the variables is $C_{n-1}, R_0, C_{n-2}, R_1, \ldots, C_0, R_{n-1}$. Figure 5 shows the structure for $n = 2$.

Theorem 2. *Using the reverse ordering of the column variables, the size of the quantum Fourier operator is* $\frac{2}{3} \cdot 2^{2n} - \frac{2}{3}$.

Proof. Let F_k denote the size of the Fourier operator of k qubits in the data structure. The subtree reached by the 0 edge from the root corresponds to the Fourier operator of $n-1$ qubits, and the subtree reached by the 1 edge from the root is a complete binary tree. Using this the recursion $F_n = 1 + F_{n-1} + 2^{2n-1} - 1$ is obtained. Unfolding this yields $F_n = 2^{2n-1} + 2^{2n-3} + \ldots + 2 = \frac{2^{2n+1}-2}{3}$; that is, $\frac{2}{3} \cdot 2^{2n} - \frac{2}{3}$.

□

Notice that this is approximately $\frac{4}{5}$ of the size of the standard ordering.

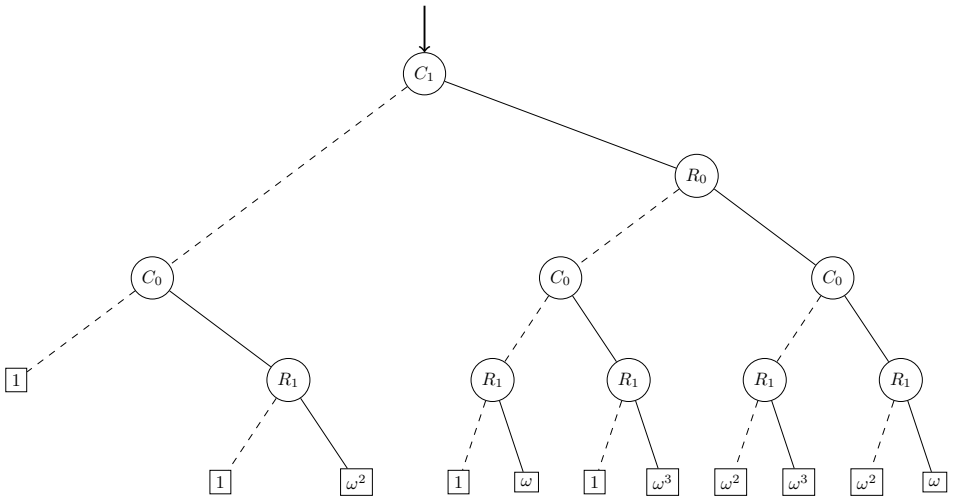

Figure 5: The Fourier operator of 2 qubits, with the column ordering reversed.

Without modifying the software of QuIDDPro, this reordering can be realized if instead of the Fourier operator F one uses the modified $F' = P \cdot F$ matrix, where P is the permutation matrix which reverses the binary form of the number of the columns. This is the same as the reversed QuIDD representation of the Fourier operator. However, there is a problem with this, namely the size of the permutation matrix P is large with this ordering.

There is another approach where we can find a good permutation matrix of small size. We know that the even numbered columns can be used to reduce the size of the QuIDD representation, so we need to collect them in the first half of the matrix. One solution to this problem is a permutation matrix P', which changes the order of columns to $0, 2, 4, \ldots, 2^n - 2, 1, 3, \ldots, 2^n - 1$. This way, we do not get the Fourier operator of $n - 1$ qubits as a submatrix, but we still can exploit the fact that the 0 side of the first variable in the QuIDD representation is one level smaller, for symmetry reasons. Notice that for the Fourier operator of 2 qubits, this representation and the previous one are the same.

Intuitively, this means that we only have to store 3 of the $2^{n-1} \times 2^{n-1}$ submatrices, so the number of nodes will be around $\frac{3}{4} \cdot 2^{2n}$.

Theorem 3. *The size of the Fourier operator using this even-odd ordering is* $\frac{17}{24} \cdot 2^{2n} - \frac{4}{3}$

Proof. The two subtrees of the root behaves differently. The 1 side is a complete binary tree with $2n-1$ levels. Because it contains the odd columns of the operator, it cannot be compressed, and it uses $2^{2n-1} - 1$ nodes.

The 0 side contains the even columns. Here, only the $2^{n-1} \times 2^{n-1}$ submatrix has to be stored, because the upper and lower halves of the $2^n \times 2^{n-1}$ submatrix are the same. By Theorem 1, this subtree uses $\frac{5}{6}2^{2n-2} - \frac{4}{3}$ nodes.

Adding up the subtrees plus the root node, we get the desired $\frac{17}{24} \cdot 2^{2n} - \frac{4}{3}$. □

This representation is somewhat larger than the previous one, but it is still smaller than the size of the standard ordering, and the permutation matrix P' that is used to transform F can be stored efficiently.

4 Modifying the QuIDD data structure

Another way of exploiting the inner structure of the Fourier operator is to change the data structure slightly. Let us allow weights on the edges. Let the value of an element be the product of the weights on the path to the leaf and the element in the leaf. Using this model, if two subgraphs differ only by a constant multiplier, it is enough if we store only one of them.

This method can be combined, for example, with the reversed column ordering.

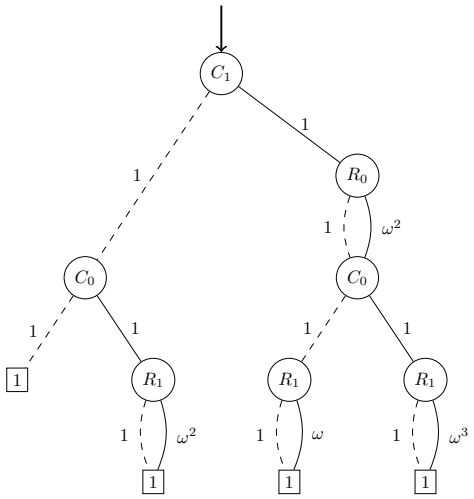

Figure 6: The Fourier operator of 2 qubits, column ordering reversed, using the weighted diagram.

Proposition 3. *The size of the Fourier operator using the weighted diagram and the reversed column ordering is* $3 \cdot 2^n - n$.

Proof. A recursion to calculate the number of the nodes is $F_n = F_{n-1} + 1 + 2^n - 1 + 2^{n-1} - 1$, because the 0 side of the root is F_{n-1}, and the 1 side consists of two complete binary trees – one for the rows, and one for the columns. This means that $F_n = 3 \cdot 2^n - n$. □

In this representation, it is sufficient to have only one leaf because the different values can be stored in the edge leading to the leaf.

Unfortunately, the operations described in [8] do not work on these diagrams. It would be interesting to investigate whether one could modify these operations so that they work on the weighted model.

5 Conclusions

While finding the optimal ordering of an ordered decision diagram is NP-hard, one can try to use the inherent structure of the matrix to improve the standard representation. The Fourier operator has an inner structure that can be seen after reordering its columns, and this reordering in the QuIDD data structure reduced the size by about 20%. A different, but more readily applicable reordering gave a reduction of about 15%.

An even bigger decrease in the size of the diagram came from allowing weighted edges. This change reduces the size of the data structure from $\Theta(2^{2n})$ to $\Theta(2^n)$ in the case of an Fourier operator of n qubits. The standard operations of [8] do not work on this diagram, so future research is necessary to decide if this weighted version can be directly used in simulations of quantum algorithms.

References

[1] Bahar, R Iris, Frohm, Erica A, Gaona, Charles M, Hachtel, Gary D, Macii, Enrico, Pardo, Abelardo, and Somenzi, Fabio. Algebric Decision Diagrams and their applications. *Formal methods in system design*, 10(2-3):171–206, 1997.

[2] Bollig, Beate. On the minimization of (complete) Ordered Binary Decision Diagrams. *Theory of Computing Systems*, pages 1–28, 2014.

[3] Bollig, Beate and Wegener, Ingo. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers*, 45(9):993–1002, 1996.

[4] Friedl, Katalin and Kabódi, László. An idea to improve QuIDD based quantum simulations. *Periodica Polytechnica. Electrical Engineering and Computer Science*, 59(2):48, 2015.

[5] Grover, Lov K. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM Symposium on Theory of Computing*, pages 212–219. ACM, 1996.

[6] Jukna, Stasys. *Boolean Function Complexity: Advances and Frontiers*, volume 27. Springer Science & Business Media, 2012.

[7] Viamontes, George F., Markov, Igor L., and Hayes, John P. QuIDDPro: High-performance quantum circuit simulation. http://vlsicad.eecs.umich.edu/Quantum/qp/.

[8] Viamontes, George F, Markov, Igor L, and Hayes, John P. Improving gate-level simulation of quantum circuits. *Quantum Information Processing*, 2(5):347–380, 2003.

[9] Wegener, Ingo. *Branching Programs and Binary Decision Diagrams: Theory and Applications*. SIAM, 2000.