A Swarm based Binary Decision Diagram (BDD) Reordering Optimizer for Reversible Circuit Synthesis

Conference Paper · February 2020

DOI: 10.1109/DTIS48698.2020.9081221

3 authors:

Amjad Hawash
An-Najah National University
33 PUBLICATIONS **68** CITATIONS
SEE PROFILE

Baker Khaldoun Abdalhaq
An-Najah National University
26 PUBLICATIONS **127** CITATIONS
SEE PROFILE

Ahmed Awad
An-Najah National University
31 PUBLICATIONS **63** CITATIONS
SEE PROFILE

A Swarm based Binary Decision Diagram (BDD) Reordering Optimizer for Reversible Circuit Synthesis

Baker Abdalhaq, Ahmed Awad, and Amjad Hawash

Department of Information & Computer Science, An-Najah National University, Nablus, Palestine.

{baker,ahmedawad,amjad}@najah.edu

***Abstract*—Reversible logic has been considered as a promising direction in the field of low power design and quantum computing. However, synthesizing reversible circuits is totally different from that of irreversible circuits. Therefore, several algorithms have been devoted to optimize the synthesis process of reversible circuits in terms of the Quantum Cost (QC). Binary Decision Diagram (BDD) based synthesis has been found as one of the dominant algorithms in reversible logic synthesis, due to its scalability in synthesizing large-scale circuits within a reasonable synthesis time that is bounded by the BDD size. However, the QC of the synthesized circuit is strongly correlated with the used BDD size in the synthesis process, wherein, each BDD node is substituted by a cascade or reversible gates. As a result, exploiting effective BDD reordering algorithms combined with BDD reduction rules turns out into less QC. In this paper, we propose a BDD reordering methodology wherein recently published swarm based optimization algorithms are integrated and compared in terms of the resultant BDD size. Experimental results show that Moth-Flame Optimization (MFO) algorithm outperforms other algorithms on the public benchmarks.**

***Index Terms*—Reversible Logic, BDD, Swarm.**

I. Introduction

. With the ongoing miniaturization of integrated circuits to keep pace with high performance computing requirements, power dissipation forms a major barrier in the development process due to its excessive resultant heat which results in serious circuit malfunction. According to Laundauer's principle, each bit loss leads to energy dissipation. However, as reversible circuits preserve every bit value from being lost, they significantly reduce energy consumption [1].

Reversible logic imposes a bijective mapping between input and output vectors. Thus, it is possible to obtain an input vector from its corresponding output. This property is highly demanded in several fields such as: digital signal processing, communication, computer graphics, DNA computations, optical computing, low power CMOS design and cryptography. Furthermore, because each quantum circuit is inherently reversible, It is possible to easily apply the advances in reversible computations in the field of quantum computing [2].

The synthesis process of reversible circuits significantly differs from the synthesis of their irreversible counterparts. First of all, the number of inputs should equal to the number of outputs in a reversible circuit with bijective mapping between each input vector and output vector. In addition, feedback and fan-out are prohibited in the synthesis process. Therefore, to synthesize a Boolean function, only reversible gates (such as NOT, CNOT, and Toffoli gates) are cascaded together, to form a reversible circuit wherein the required input function is embedded. However, such embedding might require adding extra input and output lines to achieve the required bijective mapping between inputs and outputs. Additional input lines are known as constants, while additional output lines are known as garbage. Fig. 1 illustrates a cascade of reversible gates in which a full adder function is embedded with one constant input and two garbage outputs.

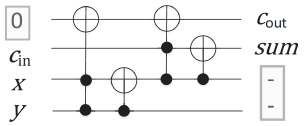

Fig. 1. A full adder embedded in a reversible circuit with one constant input and two garbage outputs.

A reversible circuit has to be designed with low cost in terms of reversible gate count, input/output line number, and with low cost to transform that circuit into a quantum circuit. The latter metric is known as the Quantum Cost (QC) which forms the key metric to be tackled in the synthesis process. Notice that, the QC is directly affected by the gate count and the control lines number as well [3].

Several algorithms have been proposed in the literature to synthesize reversible circuits from input function specifications. Both exact and heuristic based approaches have been developed to construct a network of reversible gates realizing required input function. However, some algorithms are not satisfactory due to their limited scalability in synthesizing complex circuits with large number of inputs, such as Transformation Based Synthesis (TBS), AIGs , and Lookup Table (LUT) based algorithms [4][5]. Other algorithms mainly aim to implement a circuit that realizes the input function without considering the quantum cost. Some other algorithms generate circuits with redundant lines which significantly increases its quantum and transistor level costs. Exact algorithms slowly

978-1-7281-5426-8/20/$31.00 ©2020 IEEE

converge for complex circuits, in addition to their memory demands that rise rapidly with circuit size increase [6].

Binary Decision Diagrams (BDDs) have been proposed as a compact data structure to represent Boolean functions following Shanon decomposition. BDD-based synthesis for reversible circuits has been well exploited for synthesizing large scale input functions. With their compact representation for Boolean functions, the usage of BDDs in the synthesis process makes the size of the resulting circuit to be bounded by the BDD size [7]. In BDD-based synthesis, the BDD for the input function is obtained. Thereafter, reduction rules and reordering algorithms are applied on this BDD to reduce its size. At the last stage, each BDD node is substituted by a cascade of reversible gates following a predefined mapping table. Thus, reordering a BDD with preserving its fidelity in representing the input Boolean function forms a crucial stage in the synthesis process since reducing the BDD size is expected to result in smaller reversible circuits, and thus, less QC [8].

Finding the optimal order of variables in a BDD that results in the least size is NP complete problem [9]. Several algorithms have been proposed to solve this problem such as: SAT solver based [10], greedy (namely sifting algorithm) [11], dynamic programming [12], and meta-heuristic based algorithms such as Genetic Algorithm (GA) and Simulated Annealing (SA) based reordering nodes [13]. Swarm based optimization algorithms have shown great evidence in solving complex problems. However, recent swarm algorithms have not been exploited in BDD reordering problem.

In this paper, we propose a swarm-based optimization methodology for BDD reordering algorithm, wherein, recent swarm optimization algorithms are integrated and evaluated in terms of the resultant BDD size. Our contributions are summarized as follows:

- We construct a BDD reordering optimization engine to minimize the BDD size, and thus, the QC for the synthesized reversible circuit.
- We integrate recent swarm algorithms in the proposed optimizer including: Particle Swarm Optimization (PSO), Cuckoo Search (CS), Firefly Algorithm (FFA), Grey Wolf Optimization (GWO), Moth Flame Optimization (MFO), Bat Algorithm (BAT), Salp Swarm Algorithm (SSA), and Whale Optimization Algorithm (WOA).
- We evaluate the swarm algorithms in terms of the size of the resultant BDD size.

The rest of this paper is organized as follows: Related work is presented in Section II. Section III presents problem formulation and evaluation metrics. Reversible circuit synthesis flow and the proposed optimization methodology are presented in Section IV. Experimental results are presented in Section V and Section VI concludes this paper.

II. Previous Work

Several algorithms have been proposed in the literature to optimize reversible circuit synthesis including: Transformation Based Synthesis (TBS) [4], exact synthesis, and Boolean Satisfiability Solver (SAT) based synthesis [10]. However, such algorithms slowly converge for complex circuits. Heuristic algorithms have been well exploited to find near-optimal synthesis of reversible circuits. For example, Genetic Algorithm and particle swarm optimization have been well exploited in [14, 15] to synthesize a reversible function. However, such algorithms might slowly converge on small-scale circuits and need to be integrated with local search algorithms to produce low cost circuits.

Synthesizing reversible circuits after converting a binary function into a BDD instead of building the circuit depending on other representations has opened the door to exploit BDD reduction rules to reduce the cost of its corresponding reversible circuit [1][16]. In this context, the algorithm in [8] proposes a modified version of transformation based synthesis, wherein, BDD paths are used to synthesize the circuit instead of using truth tables. In the paper of [17], the work proposes a synthesis approach that can cope with Boolean functions containing more than a hundred of variables. Authors here present a technique to derive reversible circuits for a function given by a BDD. The size of the resulting circuit is bounded by the BDD size. This allows to transfer theoretical results known from BDDs to reversible circuits.

The order of variables in the BDD impacts its size, and thus, the cost of the synthesized reversible circuit. Sifting and windowing algorithms have been proposed as a greedy algorithm to reorder the nodes of an existing BDD to reduce its size [11]. Other BDD reordering algorithm exploit dynamic programming [12] and heuristic-based reordering algorithms, such as Genetic Algorithm (GA) and Simulated Annealing (SA) algorithms. Heuristic algorithms have shown great evidence in solving BDD reordering problem [18][13].

Optimization algorithms based on swarm intelligence have some distinct advantages over traditional methods. The analysis has focused on the way of achieving exploration and exploitation and the basic components of evolutionary operators such as crossover, mutation, and selection of the fittest [19]. However, such algorithms have not been well exploited in BDD reordering problem.

Our work is related to applying recent PSO based algorithms to introduce a solution for the BDD variables reordering, and thus, for the reversible circuit synthesis, with low Quantum Cost (QC).

III. Preliminaries

The following provides a brief description for the basic terminology behind a reversible circuit synthesis in addition to a formulation for its cost metrics.

A. Reversible Logic Terminology

An $n \times n$ reversible gate g realizes a reversible function $f : B^n \rightarrow B^n$ such that $B \in \{0, 1\}$, where the mapping applied by f is bijective for every n-bit input vector into n-bit output vector. A reversible circuit $G = (g_0, g_1, \cdots, g_{m-1})$ is a cascade of reversible gates that realizes a reversible function.

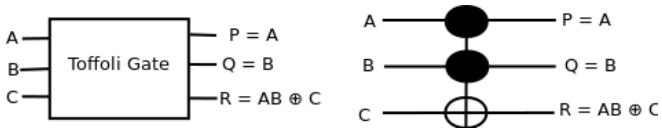

Fig. 2. Toffoli gate: block and schematic representations where A and B represent the control lines and C represents the target line and $\oplus$ represents the XOR operation.

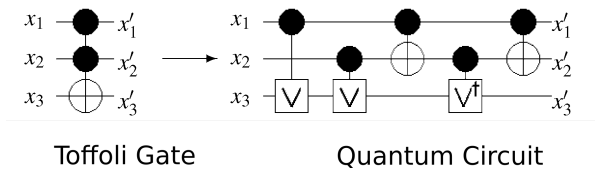

Fig. 3. The quantum circuit for a 3×3 Toffloli gate. The QC equals to 5.

The development of reversible gates and circuits started after Toffoli proposed reversible logic gates in 1977 [20]. In a reversible logic gate there is always a unique input associated with a unique output and vice versa. Several reversible gates have been proposed over the last decades including: NOT, Feynman, Fredkin, Peres, and Clifford gates [21].

A Toffoli gate consists of a target line and a number of control lines (as illustrated in Figure.2). If all signals on all control lines equal to 1, the target line signal will be inverted. Otherwise, it remains the same [1].

B. Reversible Logic Cost Metrics

The Quantum Cost (QC) of a reversible gate is defined as the number of elementary quantum operations needed to realize that reversible gate. For a Toffoli gate g with C positive control lines, the QC is formulated in eq.(1) [22]. Fig.3 illustrates the quantum circuit realization for a 3×3 Toffloli gate. Notice that the QC equals to 5. The QC for a reversible circuit is the summation for the QC for each reversible gate it consists of [1]

$$QC(g) = 2^{C+1} - 3 \tag{1}$$

C. Binary Decision Diagram (BDD) Terminology

A Binary Decision Diagram (BDD) is a compact data structure to represent a Boolean function with less memory requirements than a truth table. BDD is a directed acyclic graph $G(V, E)$ where a node $v \in V$ is either a terminal or non-terminal node. A terminal node has a value of 0 or 1 and has no outgoing edges. A non-terminal node is labeled by one of the input variables of the Boolean function, wherein, Shannon decomposition is applied following eq. (2), where x_i represents one of the input variables, $\bar{x_i}$ represents the complement of x_i, f represents the Boolean function to be represented by the BDD, and f_{x_i} represents the function f when $x_i = 1$.

$$f = \bar{x_i} f_{\bar{x_i}} + x_i f_{x_i} \tag{2}$$

As an example, Fig.4 illustrates the BDD for the Boolean function $f = x_1 \oplus x_2 \cdot x_3$, where $\oplus$ denotes the XOR operation.

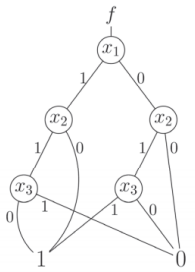

Fig. 4. BDD representation of the boolean function $f = x_1 \oplus x_2 \cdot x_3$ [1].

The size of a BDD is the number of the non-terminal nodes it contains. For example, the size of the BDD shown in Fig.4 equals 5. Some reduction rules can be applied on the BDD to reduce its size without affecting its reliability in representing the required Boolean function. Such rules include: shared nodes, complemented edges, and nodes reordering.

The order in which the variables of the Boolean function are placed in the BDD affects the overall BDD size. For example, consider the function $f = x_1 \cdot x_2 + x_3 \cdot x_4 + \cdots + x_{n-1} \cdot x_n$, Fig.5-a illustrates a BDD representing function f with a size complexity of $O(n)$. On the other hand, Figure.5-b illustrates another BDD for the same function f with another order, which turns out into a size complexity of $O(2^n)$ [23].

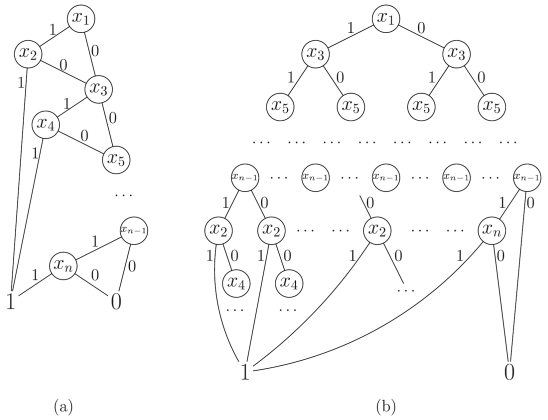

Fig. 5. BDD representation of the boolean function $f = x_1 \cdot x_2 + x_3 \cdot x_4 + \cdots + x_{n-1}.x_n$ with two different node orders: (a) BDD order with size complexity of $O(n)$. (b) BDD order with size complexity of $O(2^n)$ [23]

D. Reversible Circuit Synthesis Problem Formulation

Given a Boolean function f, the objective is to find a reversible circuit G that realizes the function f with low QC and within an acceptable synthesis time. This problem is formulated in eq.(3) where the objective is to find the circuit G that synthesizes the input Boolean function $f(X)$ where $X = x_1, x_2, \cdots, x_n$, such that the quantum cost of the

synthesized circuit $QC(G)$ is minimized and within synthesis time τ that is less than a predefined upper bound τ_{U}.

$$\begin{aligned} &\underset{G}{\text{minimize}}\ QC(G) \\ &\textbf{subject to} \\ &G \text{ realizes } f(X) \\ &\tau \leq \tau_{\mathrm{U}} \end{aligned} \tag{3}$$

In BDD-based synthesis, each BDD node is substituted into a cascade of Toffoli gates to realize the required function. Thus, it is expected that the QC of the synthesized circuit increases with the BDD size based on this one-to-one mapping between each BDD node and a cascade of Toffoli gates.

To show the relation between BDD size and QC, we have recorded the QC versus the BDD size for several benchmarks. Fig.6 illustrates the obtained plot which shows that the QC can be linearly correlated with the BDD size with a correlation coefficient of 0.985. Therefore, BDD size is a suitable evaluation function for reversible circuit synthesis optimization problem.

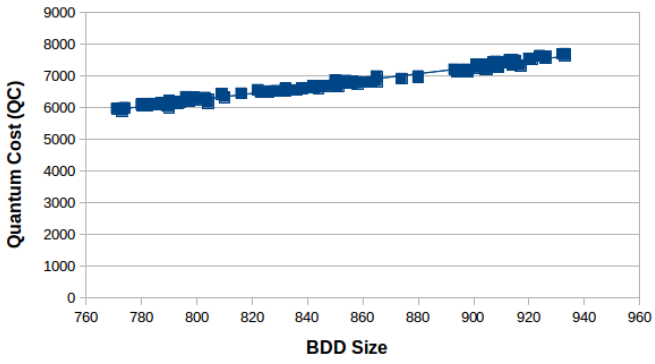

Fig. 6. BDD size versus QC with roughly linear relation for different benchmarks. All results were obtained using Revkit simulator [24].

Hence, the problem of reversible circuit synthesis is reformulated as follows: Given an input Boolean function $f(X)$, the objective is to find the order of the input variables $\pi(X)$ in its BDD that results in the least BDD size as a result of applying BDD reduction rules. This problem is formulated in eq.(4) where π represents the order (permutation) of input variables in the BDD and $\psi(\pi)$ represents the BDD size corresponding with that order after applying reduction rules. Notice that the BDD size should not exceed a maximum size of $\alpha|X|$ where $|X|$ represents the number of input variables and α is a predefined constant. For example, the BDD shown in Fig.5-a is represented by $\pi(X) = (x_1, x_2, x_3, \ldots, x_n)$ where $\psi(\pi) = n$.

$$\begin{aligned} &\underset{\pi(X)}{\text{minimize}}\ \psi(\pi(X)) \\ &\textbf{subject to} \\ &\psi(\pi(X)) \leq \alpha * |X| \\ &\tau \leq \tau_{\mathrm{U}} \end{aligned} \tag{4}$$

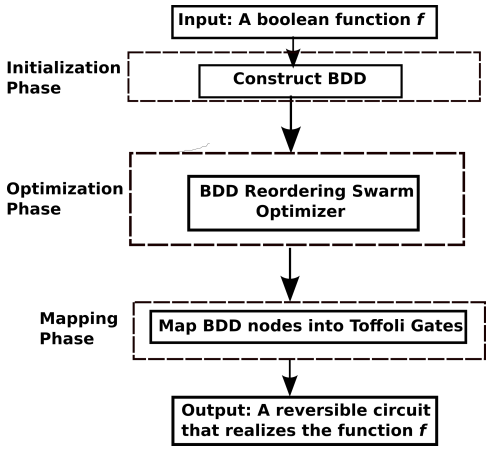

Fig. 7. BDD-based Reversible Circuit Synthesis Algorithm.

IV. Proposed Swarm-based BDD Reordering Algorithm

The algorithm published in [17] has proposed a mapping of each BDD node into a cascade of Toffoli gates to synthesize a given Boolean function. This mapping depends on the location of each node and its neighboring nodes in the graph.

A. Reversible Logic Synthesis Algorithm

Fig.7 illustrates the general flowchart of the synthesis algorithm for reversible circuits. It consists of the following phases:

1) Initialization phase: For the input Boolean function, initial BDD is constructed following Shanon decomposition.
2) Optimization phase: This phase is the core of our work. The swarm-based optimization iteratively reorders the BDD followed by applying reduction rules. This part will be described subsequently.
3) Mapping phase: This phase applies the algorithm published in [17] on the BDD generated from the optimization phase. Each BDD node is substituted by a cascade of Toffoli gates depending on its successor nodes. The mapping table published in [17] is exploited here.

B. Proposed Swarm-based Optimization Engine

Fig.8 illustrates our proposed BDD size optimization engine. It takes an input BDD which undergoes the following phases:

1) Generation phase: It generates a vector V of real numbers within the interval $[\alpha, \beta]$. This vector is initially generated randomly.
2) Discretization phase: It makes mapping between the generated vector V of real numbers and the positions of those numbers in their ascending order to generate the order π of the BDD variables. That is, $f : V \rightarrow \pi$ where f represents the mapping function.
3) Reduction phase: BDD reduction rules are applied here. This includes shared nodes and complemented edges.
4) Evaluation phase: The resultant BDD after reduction is evaluated in terms of its size. This size represents the

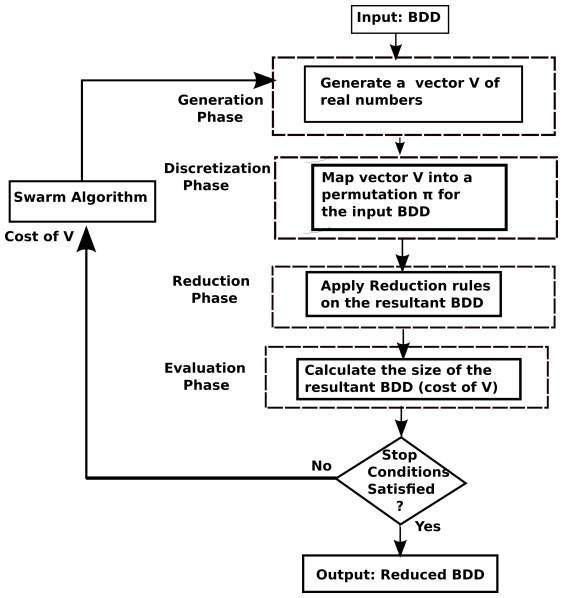

Fig. 8. BDD Size Swarm-based Optimization Engine.

cost of the vector V. If the stop conditions are satisfied, the reduced BDD is outputted. Otherwise, the cost of the vector drives the used swarm optimization algorithm to generate the next vector of real numbers.

C. Recent Swarm Optimization Algorithms

One interesting variation on swarm algorithms is Cuckoo Search (CS). This algorithm mimics the breeding behavior of Cuckoo bird. This algorithm randomly chooses nests to use and worst nests as evaluated by objective function are removed from the population. The algorithm uses levys walk to explore the search space alongside with local walks to explore potential solutions [25].

Firefly Algorithm (FFA) is similar to particle swarm algorithm. It assumes that fireflies are attracted to the other flies that are brighter. The brightness is proportional to the objective function. The distance between fireflies also affects the brightness. The brightness of one individual as seen by others drops exponentially as a function of distance [26].

Grey Wolf Optimization (GWO) is a promising swarm based algorithm. Grey wolfs exhibit complicated social and hunting behaviors. The algorithm simulates this behavior by choosing the best three individuals to act as leaders and the rest as followers (known as omegas). Therefore, omegas move towards the center point of the best three solutions. The exploitation part of the algorithm is generated by divergence of leaders to simulate the search for prays [27].

Moth-Flame Optimization (MFO) is inspired by the deadly spiral movement of moths around flames. The best solution found so far is considered as the flame whereas all other solutions are considered as moths. To update the location of moths the algorithm uses spiral movement [28]. Other interesting swarm algorithms include

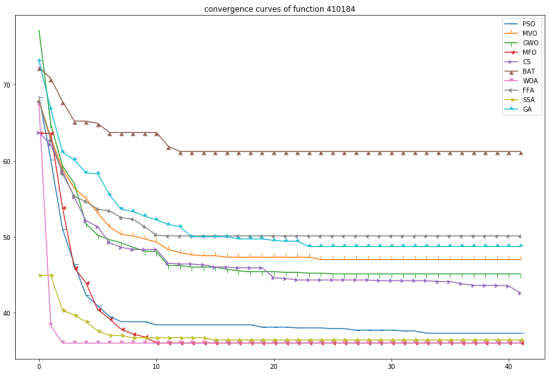

Fig. 9. Convergence Curves for function 0410184 where the X-axis represents iteration# and Y-axis represents the BDD size.

Bat Algorithm (BAT), Salp Swarm Algorithm (SSA), Whale optimization algorithm (WOA), and Multiverse Optimization (MVO) [29] [30] [31] [32].

V. Experimental Tests

To evaluate our proposed algorithm, we compared several swarm algorithms using five benchmarks of different sizes and complexities. The experiment was performed using python implementation of optimization techniques. We also used python binding to CUDD library that is written in C. CUDD was used to build BDDs. We repeated the same experiments ten times. The following pairs of values represent the average results of the ten repetitions at the end of iterations (Urf3, 10), (Hwb6, 6), (41084, 14), (Ham15, 15) and (Apex4, 19) where each pair represents a function name and its corresponding number of variables. The maximum number of iterations is 3 times the number of variables of the synthesized function. Population size (swarm size) was fixed to 20 for all functions.

Table I
Algorithms performance in terms of average BDD size

Algorithm	urf3	hwb6	410184	ham15	apex4	sum
MFO	178.1	26	36	23	47.7	310.8
SSA	178.3	26	36.4	23	47.7	311.4
WOA	179.1	26.2	36	23	49.1	313.4
PSO	178.6	26.3	38.4	23	48.3	314.6
CS	178.3	26	45.9	23	48.5	321.7
GWO	178.3	26.3	45.5	23	48.9	322
MVO	179.3	26.2	47.3	23	48.4	324.2
FFA	178.8	26.3	50.1	23	50.3	328.5
GA	179.6	26.2	49.7	23	50.8	329.3
BAT	181.1	26.4	61.2	23	51.8	343.5
CUDD sifting	200	28	36	23	52	339
no reordering	21158	571	5250	17283	12527	56789

As shown in Table I and Fig.9, it is obvious that MFO, SSA, WOA and PSO exhibited similar convergence speed. The differences between the four algorithms are minor. They all converge to almost the same solution, although MFO is the fastest and gives the best solution. MFO and WOA include explicit spiral movement while MFO, SSA and WOA include

adaptive parameters that are inspired by SA. The parameters control exploration and exploitation throughout the course of search process. The achieved reduction by using MFO over the classical CUDD sifting algorithm and is about 8.3%.

VI. Conclusion and Future Works

In this work we presented swarm based framework to optimize BDD size in order to minimize quantum cost for reversible circuits. We have compared ten algorithms to be used in the framework. The algorithms are PSO, GA, CS, MFO, SSA, WOA, GWA, FFA, BAT, MVO. Experimental results have shown that MFO, SSA, WOA and PSO are potential algorithms to solve this particular problem. MFO showed the best performance and fastest convergence behavior.

Although smaller BDD size often results into less quantum cost, this might not be always the case. Therefore, one of the possible future works is to include the mapping phase of the synthesis algorithm into the optimization per se with setting minimizing the number of control lines as the objective function.

Acknowledgement

This work was supported by An-Najah National University under Grant ANNU-1920-Sc006

References

[1] R. Wille and R. Drechsler, *Towards a Design Flow for Reversible Logic*. Springer, 2010.

[2] T. N. Sasamal, A. K. Singh, and A. Mohan, "Reversible logic circuit synthesis and optimization using adaptive genetic algorithm," *Procedia Computer Science*, vol. 70, pp. 407 – 413, 2015.

[3] P. Kerntopf, "A new heuristic algorithm for reversible logic synthesis," in *Proceedings. 41st Design Automation Conference, 2004.*, July 2004, pp. 834–837.

[4] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, June 2003, pp. 318–323.

[5] L. Biswal, R. Das, C. Bandyopadhyay, A. Chattopadhyay, and H. Rahaman, "A template-based technique for efficient clifford+t-based quantum circuit implementation," *Microelectronics Journal*, vol. 81, pp. 58 – 68, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026269218300430

[6] R. Ray, A. Deka, and K. Datta, "Exact synthesis of reversible logic circuits using model checking," *CoRR*, vol. abs/1702.07470, 2017. [Online]. Available: http://arxiv.org/abs/1702.07470

[7] A. Chattopadhyay, A. Littarru, L. Amar, P. Gaillardon, and G. D. Micheli, "Reversible logic synthesis via biconditional binary decision diagrams," in *2015 IEEE International Symposium on Multiple-Valued Logic*, May 2015, pp. 2–7.

[8] K. Podlaski, "Reversible circuit synthesis using binary decision diagrams," in *2016 MIXDES - 23rd International Conference Mixed Design of Integrated Circuits and Systems*, June 2016, pp. 235–238.

[9] B. Bollig and I. Wegener, "Improving the variable ordering of obdds is np-complete," *IEEE Transactions on Computers*, vol. 45, no. 9, pp. 993–1002, Sep. 1996.

[10] D. Grosse, X. Chen, G. W. Dueck, and R. Drechsler, "Exact sat-based toffoli network synthesis," in *Proceedings of the 17th ACM Great Lakes Symposium on VLSI*, ser. GLSVLSI '07. New York, NY, USA: ACM, 2007, pp. 96–101. [Online]. Available: http://doi.acm.org/10.1145/1228784.1228812

[11] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," in *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, Nov 1993, pp. 42–47.

[12] S. J. Friedman and K. J. Supowit, "Finding the optimal variable ordering for binary decision diagrams," in *Proceedings of the 24th ACM/IEEE Design Automation Conference*, ser. DAC '87. New York, NY, USA: ACM, 1987, pp. 348–356. [Online]. Available: http://doi.acm.org/10.1145/37888.37941

[13] W. Lenders and C. Baier, "Genetic algorithms for the variable ordering problem of binary decision diagrams," in *Foundations of Genetic Algorithms*, A. H. Wright, M. D. Vose, K. A. De Jong, and L. M. Schmitt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–20.

[14] P. Manna, D. K. Kole, H. Rahaman, D. K. Das, and B. B. Bhattacharya, "Reversible logic circuit synthesis using genetic algorithm and particle swarm optimization," in *2012 International Symposium on Electronic System Design (ISED)*, Dec 2012, pp. 246–250.

[15] K. Podlaski, "Reversible circuit synthesis with particle swarm optimization using crossover operator," in *2015 22nd International Conference Mixed Design of Integrated Circuits Systems (MIXDES)*, June 2015, pp. 375–379.

[16] R. Wille and R. Drechsler, "Effect of bdd optimization on synthesis of reversible and quantum logic," *Electronic Notes in Theoretical Computer Science*, vol. 253, no. 6, pp. 57 – 70, 2010, proceedings of the Workshop on Reversible Computation (RC 2009). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066110000198

[17] ——, "Bdd-based synthesis of reversible logic for large functions," in *Proceedings of the 46th Annual Design Automation Conference*, ser. DAC '09, 2009.

[18] B. Bollig, M. Lbbing, and I. Wegener, "Simulated annealing to improve variable orderings for obdds," in *IN INT'L WORKSHOP ON LOGIC SYNTH*, 1995, pp. 5–5.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[20] M. Sarvaghad-Moghaddam and A. A. Orouji, "A new design and simulation of reversible gates in quantum-dot cellular automata technology," *CoRR*, vol. abs/1803.11017, 2018. [Online]. Available: http://arxiv.org/abs/1803.11017

[21] A. Al-Rabadi, *Reversible Logic Synthesis: From Fundamentals to Quantum Computing*. Springer Berlin Heidelberg, 2011, 01 2004.

[22] S. M. Saeed, X. Cui, R. Wille, A. Zulehner, K. Wu, R. Drechsler, and R. Karri, "Towards reverse engineering reversible logic," *CoRR*, vol. abs/1704.08397, 2017. [Online]. Available: http://arxiv.org/abs/1704.08397

[23] R. Drechsler, N. Drechsler, and W. Günther, "Fast exact minimization of bdds," in *Proceedings of the 35th Annual Design Automation Conference*, ser. DAC '98. New York, NY, USA: ACM, 1998, pp. 200–205. [Online]. Available: http://doi.acm.org/10.1145/277044.277099

[24] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "Revkit: An open source toolkit for the design of reversible circuits," in *Reversible Computation*, A. De Vos and R. Wille, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 64–76.

[25] X. Yang and Suash Deb, "Cuckoo search via lvy flights," in *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, Dec 2009, pp. 210–214.

[26] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *IJBIC*, vol. 2, pp. 78–84, 2010.

[27] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[28] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-based systems*, vol. 89, pp. 228–249, 2015.

[29] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, 2010, pp. 65–74.

[30] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Advances in Engineering Software*, vol. 114, pp. 163 – 191, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997816307736

[31] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, p. 5167, 2016.

[32] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," 1988.