GraphRSim: A Joint Device-Algorithm Reliability Analysis for ReRAM-based Graph Processing

Chin-Fu Nien*, Yi-Jou Hsiao†, Hsiang-Yun Cheng*, Cheng-Yu Wen‡, Ya-Cheng Ko‡ and Che-Ching Lin‡
*Academia Sinica, Taiwan †National Chiao Tung University, Taiwan ‡National Taiwan University, Taiwan
Email: *{watchmannien, hycheng}@citi.sinica.edu.tw, †irenegoldson.iie06g@nctu.edu.tw,
‡{b05901048, b05902056, b05901039}@ntu.edu.tw

***Abstract*—Graph processing has attracted a lot of interests in recent years as it plays a key role to analyze huge datasets. ReRAM-based accelerators provide a promising solution to accelerate graph processing. However, the intrinsic stochastic behavior of ReRAM devices makes its computation results unreliable. In this paper, we build a simulation platform to analyze the impact of non-ideal ReRAM devices on the error rates of various graph algorithms. We show that the characteristic of the targeted graph algorithm and the type of ReRAM computations employed greatly affect the error rates. Using representative graph algorithms as case studies, we demonstrate that our simulation platform can guide chip designers to select better design options and develop new techniques to improve reliability.**

I. Introduction

With the explosive growth of data in recent years, graph processing has received intensive interests as it is effective in analyzing the relationship among real-world entities. It has been widely applied in various domains, including social networks, recommendation systems, and machine learning. However, due to its low compute-memory ratio and random data access pattern, conventional computing systems with separated computation unit and data storage are inefficient when processing large-scale graphs [1]. Recent studies [1]–[3] demonstrate that metal-oxide resistive random access memory (ReRAM) [4] provides a promising processing-in-memory solution to accelerate matrix operations in graph algorithms, as it can help to reduce data transfer and enable highly-parallel computations. For example, Song et al. [1] introduces a ReRAM-based graph processing accelerator which can achieve 16x speedup and 34x energy savings compared to CPU.

While ReRAM-based accelerators show great potential to enable energy-efficient graph processing, the intrinsic stochastic behavior of ReRAM devices is likely to make its computation unreliable. In a ReRAM crossbar array, as per-cell current variation accumulates on the bitlines, it is difficult for the analog-to-digital converters (ADCs) to output correct results. Recent work shows that such non-ideal device properties could degrade the inference accuracy of a neural network model from 89% to 59% on a ReRAM-based deep learning accelerator [5]. Nevertheless, to the best of our knowledge, no prior work has analyzed the impact of unreliable ReRAM on the computing accuracy of graph processing.

In this paper, we show that not all of the graph algorithms are resilient to ReRAM-induced errors and the influence on computing accuracy is highly algorithm-dependent. For example, PageRank (PR) is sensitive to device-level errors incurred at high order bits (HOBs) while the error is more serious if a zero output is wrongly sensed as a non-zero value for Weakly Connected Components (CC). Thus, we propose a simulation framework, GraphRSim, for ReRAM-based graph processing accelerators to analyze the impact of stochastic device variation on the computing accuracy of various graph algorithms. Based on our analysis, we use PR and CC as case studies to demonstrate two techniques that can be utilized to improve reliability. The main contributions of this work can be summarized as follows:

- We propose the first reliability simulation framework, GraphRSim, for ReRAM-based graph processing accelerators. It can be utilized to explore the design space for reliability optimization.
- We are the first work that jointly study the impact of device and algorithm features on the reliability of ReRAM-based graph accelerators. Our evaluation shows that many graph algorithms are not resilient to ReRAM-induced errors and the algorithm-dependent accuracy degradation should be carefully examined.
- Based on our analysis, we propose two reliability optimization techniques and show that adapting the reference value in ADCs according to the characteristic of the target algorithm enables PR to converge and can significantly reduce the error rate of CC.

II. Preliminary

A. Graph Processing

Graphs are widely used to represent relationships among entities in various application domains. A graph $G = (V, E)$ is composed of $|V|$ vertices and $|E|$ edges. If each edge in a graph is directed from a source vertex to a destination vertex, as shown in Fig. 1(a), the graph is a *directed graph*. Otherwise, the graph with bi-directional edges is an *undirected graph*, as shown in Fig. 1(b). A graph can be naturally represented as an adjacency matrix, where the matrix elements indicate whether pairs of vertices are connected through an edge (or indicate the weights of the edges). It can also be represented via a compressed form, such as the compressed sparse row (CSR) or coordinate list (COO) [1], to save storage resources.

Graph algorithms traverse vertices and edges to explore application-specific graph properties. Vertex-centric model is widely used to implement various parallel graph algorithms. In

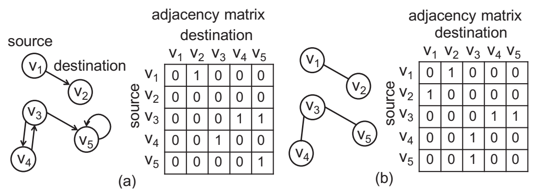

Fig. 1. Examples of (a) directed graph and (b) undirected graph.

general, a vertex-centric graph algorithm can be divided into three phases: Process, Reduce, and Apply [1]. For example, the well-known PageRank (PR) algorithm, which ranks webpages based on its relative importance (PR score), can be formulated as iterating through the following equation:

$$PR_{t+1}(v) = \frac{1-r}{|V|} + \sum_{u \in in\_neigh(v)} r \cdot \frac{PR_t(u)}{out\_deg(u)} \tag{1}$$

where $PR_{t+1}(v)$ is the PR score of vertex v at time $t+1$, r is the damping factor, $in\_neigh(v)$ is the set of input neighbors of vertex v, and $out\_deg(u)$ is the outdegree of vertex u. The calculation of each term in the summation is the *process* operation. The processed values are then *reduced* into one property value through the summation and addition. This property value is *applied* back to the PR score of vertex v for the next iteration.

B. ReRAM Basics

ReRAM [4] is a promising candidate to replace DRAM in future memory architecture, as it can provide high density, fast read access, and low leakage power. Fig. 2(a) shows the structure of a ReRAM cell, where an oxide layer is sandwiched between two metal layers of electrodes (top electrode and bottom electrode). When a highly positive voltage is applied across a ReRAM cell, the conductive filament made out of oxygen vacancies is formed and the cell is *SET* as the low resistance state (LRS). Conversely, when a highly negative voltage is applied to the cell, the conductive filament is ruptured and the cell is *RESET* as the high resistance state (HRS). To read the data from a cell, a small voltage is applied such that the state of the cell would not be changed.

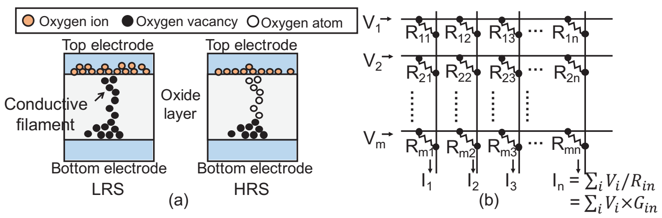

Fig. 2. ReRAM (a) cell structure and (b) crossbar array.

ReRAM crossbar arrays can be used to efficiently perform matrix-vector multiplications. As shown in Fig. 2(b), each ReRAM cell with conductance $G_{ij} = 1/R_{ij}$ is located at the cross point of a wordline (WL) and a bitline (BL). When input voltages (V_i) are applied to wordlines, the accumulated current on bitline j is the sum-of-products of the input voltage and cell conductance: $I_j = \sum_i V_i/R_{ij} = \sum_i V_i \cdot G_{ij}$. Ideally, with the crossbar structure, matrix-vector multiplication can be done in one constant time step. Such processing-in-memory capability provides a promising solution to accelerate graph algorithms, as many graph algorithms can be transformed into performing matrix operations with the adjacency matrix [1].

C. ReRAM-based Graph Processing Accelerators

Many ReRAM-based accelerators have been proposed to enable energy-efficient graph processing, and Fig. 3 shows the generic accelerator architecture assumed in a few works [1], [3]. Memory ReRAM stores graph data in the original compressed format (e.g., COO). The controller (CTRL) converts the compressed graph data to the adjacency matrix and maps it onto the crossbar arrays in graph engines (GEs) to perform matrix-vector multiplications. Digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) are used to convert input and output data respectively between analog and digital domains. As DACs usually have limited resolution and each ReRAM cell can only store limited number of bits, inputs and matrix values are split into multiple bits and the shift-and-add circuit (S/A) is used to assemble results. Simple arithmetic logic units (sALUs) are included in each GE to carry out arithmetic operations for graph algorithms if needed.

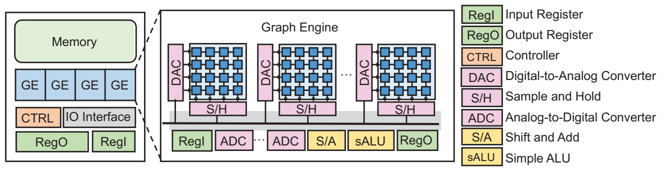

Fig. 3. ReRAM-based graph processing accelerator architecture.

Fig. 4 presents an example of performing PageRank on a ReRAM-based accelerator. The adjacency matrix is preprocessed and mapped to a crossbar in a GE. If an edge (u, v) exists, the $r/out\_deg(u)$ value in Equation (1) is mapped to the ReRAM cell at the u-th row and v-th column. Otherwise, the ReRAM cell stores zero. The PR scores of each vertex ($PR_t(u)$) is converted into input voltage signals. At the first iteration, PR scores ($PR_{t=0}$) are initialized to $1/|V| = 0.2$, as shown in Fig. 4(a). After performing the matrix-vector multiplication, a bias value ($(1-r)/|V| = 0.04$) is added to the outputs of the crossbar array via the sALU to derive new PR scores ($PR_{t=1}$), which will become the inputs of the crossbar array in the next iteration, as shown in Fig. 4(b). This process continues until $\sum_{u \in V} |PR_t(u) - PR_{t-1}(u)| < \epsilon$, where ϵ is typically 10^{-4}, or until a maximum iteration T is reached.

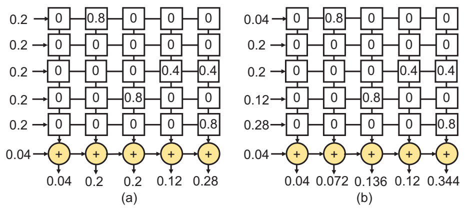

Fig. 4. An example of mapping PageRank on ReRAM crossbar at the (a) 1st and (b) 2nd iteration for the graph shown in Fig. 1(a) with $r = 0.8$.

III. Reliability of ReRAM-based Graph Processing: A Joint Device-Algorithm Impact

The intrinsic stochastic variation of ReRAM devices may make ReRAM-based accelerators unreliable. In this section, we first introduce the stochastic resistance variation in ReRAM and examine its impact on the sensing error rates of output currents. We then discuss its algorithm-dependent impact on the computing accuracy of graph processing.

A. Stochastic Variation in ReRAM

Resistance variation is an intrinsic property of ReRAM devices associated with the stochastic nature of the generation and rupture of oxygen vacancies. The resistance variation of LRS comes from the fluctuation of the number or the size of conductive filaments generated during *SET*, while the resistance variation of HRS is attributed to the varying length of the tunneling gap (i.e., the distance between the tip of the filament and the top electrode) during *RESET*. Recent studies [6]–[8] show that the resistance distribution of ReRAM follows the lognormal distribution and the resistance probability density function (PDF) can be expressed as follows:

$$f_R(r) = \frac{e^{-(ln(r)-\mu)^2/(2\sigma^2)}}{r\sigma\sqrt{2\pi}} \tag{2}$$

where μ and σ are the mean and standard deviation of the random variable $ln(r)$, respectively. Intuitively, μ_{HRS} is larger than μ_{LRS}, and $R\text{-}ratio$ is equal to $e^{\mu_{HRS}}/e^{\mu_{LRS}}$. Since the oxygen ions tend to move back from the top electrode to stuff the vacancies in the oxide layer during *RESET* and this process is more random than the generation of conductive filament, σ_{HRS} is usually larger than σ_{LRS} [4].

B. Impact of ReRAM Stochastic Variation on Sensing Errors

The stochastic variation of ReRAM resistance is likely to degrade the sensing accuracy of output currents. As shown in Fig. 5(a), the accumulated output current on a bitline can be represented as a statistical distribution. To determine the sum-of-products result, ADC is used to compare the accumulated current on the bitline with the reference values (Ref_i). For example, if the accumulated current lies between Ref_0 and Ref_1, it is sensed as “1”. Neighboring states (sum-of-products values) in the accumulated current distribution may overlap and the overlapped region is error-prone. For instance, the accumulated current distribution of state “2” and state “3” in Fig. 5(a) are overlapped. As such, ADC may wrongly output “3” when the actual sum-of-products is “2”, if the sampled accumulated current is larger than Ref_2.

The impact of device variability on the sum-of-products sensing error rates is mainly determined by $R\text{-}ratio$, σ, and the number of activated WLs [5]. ReRAM devices with different oxide and electrode materials have variant $R\text{-}ratio$ and σ [4], [6], [9], [10], and these two parameters greatly affect the sensing error rates. As shown in Fig. 5(b), smaller $R\text{-}ratio$ squeezes the margin between different states, and the overlapped region between neighboring states becomes larger. Similarly, larger σ expands the current distribution of each state and thus enlarges the overlapped region between neighboring states, as shown in Fig. 5(c). In addition, the accumulation of more cell resistance variance increases the variance of the accumulated current. Thus, the sensing error rate increases as the number of concurrently activated WLs increases. As such, in practical designs, only limited number of wordlines (WL_{max}) are activated concurrently to ensure correctness [5].

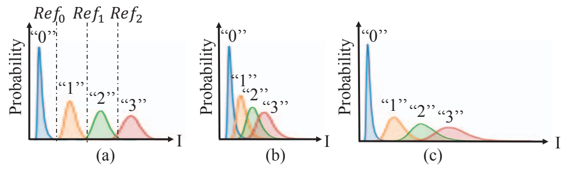

Fig. 5. Example PDFs of the accumulated current distribution on a BL when three WLs are activated: (a) baseline $R\text{-}ratio$ and σ [10]; (b) smaller $R\text{-}ratio$ with fixed current variation; (c) larger σ with fixed $R\text{-}ratio$.

C. Impact of Unreliable ReRAM on Graph Processing

As described in Section III-B, ReRAM-based computation, i.e., the sum-of-products output at each bitline, is unreliable due to the intrinsic stochastic variation of ReRAM devices. The impact of unreliable ReRAM-based computations on the computing accuracy of graph processing is highly algorithm-dependent. We classify graph algorithms into four categories based on their processing characteristics, as shown in Fig. 6.

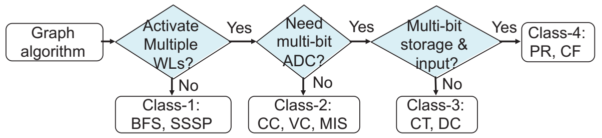

Fig. 6. Classification of graph algorithms according to their processing characteristics in ReRAM-based accelerators.

1) Class-1 - Activate only one single WL at a time: This class of algorithms, such as Breath-First Search (BFS) and Single-Source Shortest Path (SSSP), perform parallel add at the *Process* phase of vertex-centric programming via activating a single WL and adding the output of each BL with each vertex's original property value [2]. Since only one single WL is activated at a time, this class of algorithms are resilient to ReRAM-induced computing errors as long as the ReRAM cell with sufficiently large $R\text{-}ratio$ and small σ is utilized.

2) Class-2 - Activate multiple WLs but only need single-bit ADCs: This class of algorithms, such as Weakly Connected Components (CC), Approximate Vertex Cover (VC), and Maximal Independent Set (MIS), activate multiple WLs to enable parallel computations but only need to use single-bit ADCs to distinguish between zero and non-zero outputs. For example, CC divides the vertex set into subsets where vertices in the same subset are connected together, and Fig. 7(a) demonstrates its ReRAM-based computation. At the iteration shown in Fig. 7(a-1), the 3^{rd} WL is activated and the BLs with non-zero outputs indicate that the corresponding vertices are connected to v_3. The vertex property array is updated via sALU to specify the associated subset of each vertex, and the WLs correspond to the newly updated vertices (v_4 and v_5) are activated in the next iteration to continue to find the connected components, as shown in Fig. 7(a-2). Since we are only interested in whether the output of a bitline is zero or non-zero, rather than its actual value, single-bit ADC is sufficient. The computing accuracy of CC mainly depends on whether the overlapped region between state “0” and other states in the accumulated current distribution is large, and whether the reference value in ADC (Ref_0 in Fig. 5) can accurately separate zero and non-zero outputs. If a zero output is wrongly sensed as non-zero, disconnected vertices would be wrongly grouped into the same subset and the error is hard to be recovered. Conversely, if a non-zero output is wrongly sensed as zero, there are still chances to recover the error during later iterations. For example, if the output of the 4^{th} bitline in Fig. 7(a-1) is wrongly sensed as zero, during later iterations, it can be recovered by activating the 4^{th} WL to find that v_4 is actually connected to v_3.

3) Class-3 - Activate multiple WLs, need multi-bit ADCs, but process with single-bit inputs and adjacency matrix: Different from *Class-2* algorithms, this class of algorithms, including Conductance (CT) and Degree Centrality (DC), are sensitive to actual sum-of-products output values. For example, CT is an algorithm that helps to measure the quality of clustering, i.e., whether a vertex subset S and its complement set $\bar{S}$ are sufficiently isolated. The conductance of a cut $(S, \bar{S})$ can be formulated as $\sum_{i\in S, j\in \bar{S}} a_{ij}/min\left(a(S), a(\bar{S})\right)$, where a_{ij} are the entries of the adjacency matrix and $a(S) = \sum_{i\in S, j\in V} a_{ij}$. Fig. 7(b) shows an example that calculates the conductance of a cut with $S = \{v_4, v_5\}$ in two steps. As the correctness of the sum-of-products outputs is important for this class of algorithms, all the impact factors mentioned in Section III-B affect the computing accuracy.

4) Class-4 - Activate multiple WLs, need multi-bit ADCs, and process with multi-bit inputs and weighted/preprocessed adjacency matrix: For this class of algorithms, such as PageRank (PR) and Collaborative Filtering (CF), due to the limited DAC resolution and cell capacity, the multi-bit inputs are decomposed and fed into WLs using multiple clock cycles and each multi-bit entry of the adjacency matrix is also decomposed and mapped onto multiple BLs. As such, shift-and-add circuit is required to aggregate the outputs, and sum-of-products errors in high order bits (HOBs) contribute more to the reliability degradation of the final result.

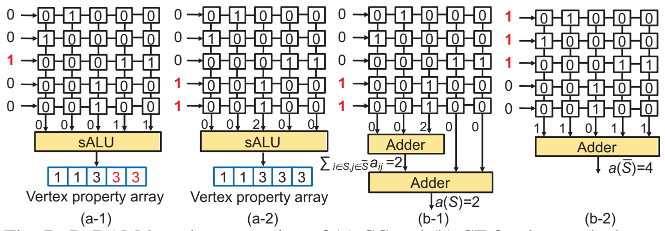

Fig. 7. ReRAM-based computation of (a) CC and (b) CT for the graph shown in Fig. 1(b). The adjacency matrix is mapped to the crossbar. For the example of CC, suppose that v_1 and v_2 are already found to be in the same subset.

IV. Simulation Framework

GraphRSim is a simulation platform that enables the reliability analysis for ReRAM-based graph processing accelerators. It provides a flexible interface for users to analyze the impact of unreliable ReRAM devices on the computing accuracy of various graph algorithms. Fig. 8 shows the simulation flow of GraphRSim. We use the *NVM Error Analytical Module* in DL-RSIM [5] to model the sensing error rates of the accumulated current per bitline and incorporate it with our *Graph Accuracy Simulation Module*, which takes the user-specified graph algorithm, customized hardware configurations, the error rates simulated by DL-RSIM, and the graph dataset as inputs to estimate the computing accuracy.

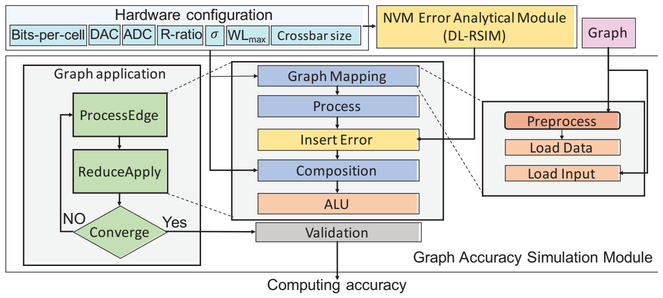

Fig. 8. Simulation framework of GraphRSim.

To enable the simulation of ReRAM-based graph processing, users can use two APIs, *ProcessEdge* and *ReduceApply*, provided by GraphRSim to implement different graph algorithms. The *ProcessEdge* API takes four parameters, including graph dataset, preprocessing function, crossbar input vector, and processing type, as inputs and perform matrix-vector multiplication (MUL) or parallel add (ADD) accordingly. The *ReduceApply* API takes the ALU operation and two input vectors (or one input vector and one constant value) as input parameters, and perform computations based on the specified ALU operation. Fig. 9 shows an example that uses these two APIs to implement PageRank. Other algorithms described in Section III-C can also be implemented in a similar way.

Algorithm 1: PageRank in GraphRSim

```
for $iter \leftarrow 0\, to\, maxIter$ do
    SumE = $ProcessEdge(Graph, r * E.exist/V.outdeg, V.prop, MUL)$;
    V.prop = $ReduceApply(+, SumE, (1-r)/|V|)$;
end
```

Fig. 9. Implementation of PageRank for GraphRSim.

For the simulation of computing accuracy, *Graph Accuracy Simulation Module* performs five steps to transform *ProcessEdge* and *ReduceApply* into ReRAM-based computations, as shown in Fig. 8. First, at the *Graph Mapping* step, the compressed graph dataset is converted to adjacency matrix and is preprocessed if needed. According to the bits-per-cell and crossbar size, each element of the adjacency matrix is decomposed into multiple bits and mapped to the ReRAM cells. Each element in the input vector is also decomposed into multiple bits based on the resolution of DAC. Second, at the *Process* step, MUL or ADD is performed and the computation is partitioned based on WL_{max}. Third, at the *Insert Error* step, the outputs of the crossbar arrays are modified according to the sum-of-products error rates estimated by DL-RSIM. Shift-and-add is then performed to resemble the outputs at the *Composition* step. Finally, the ALU operation specified in *ReduceApply* is performed. After the graph algorithm converges or a maximum iteration T is reached, the final output is compared with ideal results to validate the computing accuracy. Although it is hard to completely validate the correctness of GraphRSim, we make sure the error rates estimated by DL-RSIM [5], which has been validated using SPICE, are appropriately incorporated in the computation results.

V. Evaluation and Results

A. Evaluation Setup

We use four representative graph algorithms, including CC, CT, PR, and CF, as examples to show that GraphRSim can help chip designers to explore the design space for reliability improvement. Note that *Class-1* algorithms are not evaluated as they are less sensitive to ReRAM-induced errors. Table I shows the graph datasets used in the following evaluations, including three undirected graphs and three directed graphs. We use the same setting of $R\text{-}ratio$ ($R\text{-}ratio = 25$) and σ as [10], $WL_{max} = 8$, DAC resolution = 1 bit, and single-level cell as our baseline hardware configuration.

Different metrics are used to measure the computing accuracy for different graph algorithms due to their distinct characteristics. For CC, the output is the group index of each vertex, and thus the error rate is defined as the percentage of vertices with wrong group index. For CT, the conductance error is defined as the difference between the ideal and the simulated conductance value. For PR, the final $PR_{t=T}$ scores, where T is set to 100, is used to re-calculate Equation (1) on CPU, and the validation error is thus defined as the $L1$ difference (mean-absolute error) between the final score derived by the simulated ReRAM-based graph accelerator and the re-calculated score. For CF, a well-trained model with RMSE=0.914 is used and the RMSE between the simulated value and the label is used to measure the reliability.

TABLE I
Selected graph datasets

Dataset	$\|V\|$	$\|E\|$	Dataset	$\|V\|$	$\|E\|$
Facebook [11]	4039	88234	Wiki-Vote [11]	7115	103689
arXiv [11]	5242	14484	P2P [11]	8846	31839
Email [11]	36692	18381	Ratings [12]	193601	100797

B. Results and Discussion

1) Impact of R-ratio and σ: Fig. 10 shows that the error rates of all the evaluated graph algorithms decreases as $R\text{-}ratio$ increases, since increasing $R\text{-}ratio$ reduces the overlapped region between neighboring states of the accumulated current distribution[1]. To achieve reliable computation, a ReRAM device with $R\text{-}ratio = 30$ is enough for CC and CT, while a larger $R\text{-}ratio$ is required for PR and CF. For CC ($Class$-2 algorithm), the error rate dramatically increases (nearly 100%) when $R\text{-}ratio < 30$. It is because whenever a zero output is wrongly sensed as a non-zero value, the error continuously propagates through later iterations and finally all vertices are misclassified into one single group. For CT, the conductance error dramatically increases when $R\text{-}ratio < 30$ but then saturates to a constant value, as both the numerator and denominator of the conductance value grows rapidly when the current sensing errors occur. For PR and CF ($Class$-4 algorithms), errors at HOBs seriously degrade accuracy, especially when $R\text{-}ratio < 30$, since the validation error of PR and the RMSE of CF overflow when a zero in HOBs is wrongly sensed as one. The impact of σ is also shown in Fig. 10. Although reducing σ from 1x to 0.3x of the baseline σ (σ_b) only slightly affects the computing accuracy of CC and CT, it does greatly improve the reliability for PR and CF, as reducing σ can help to decrease HOB errors.

[1]Due to space limitation, when evaluating the impact of $R\text{-}ratio$, σ, and WL_{max}, we only show the results of one graph dataset ("Wiki-Vote" for CC, "Facebook" for CT and PR, and "Ratings" for CF) for each algorithm. The trend is similar for other datasets.

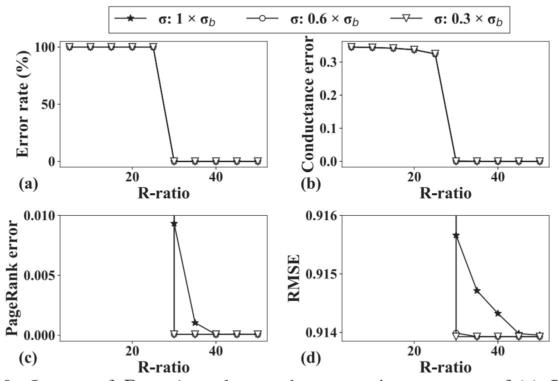

Fig. 10. Impact of $R\text{-}ratio$ and σ on the computing accuracy of (a) CC (b) CT (c) PR and (d) CF.

2) Impact of WL_{max}: Although larger WL_{max} enables a higher degree of computation parallelism, Fig. 11 shows that increasing WL_{max} degrades the computing accuracy. For the baseline ReRAM device ($R\text{-}ratio = 25$ [10]), at most four WLs can be activated concurrently to achieve satisfactory computing accuracy for CC and CT. PR and CF ($Class$-4 algorithms) even need a much smaller WL_{max}, such as one or two, to avoid serious accuracy degradation. It is because PR and CF are sensitive to HOB errors, and activating a higher number of WLs further exacerbates the problem. If chip designers would like to speedup the computation by activating more WLs, a ReRAM device with larger $R\text{-}ratio$, such as the device with $R\text{-}ratio = 35$ or 40 [9], should be chosen.

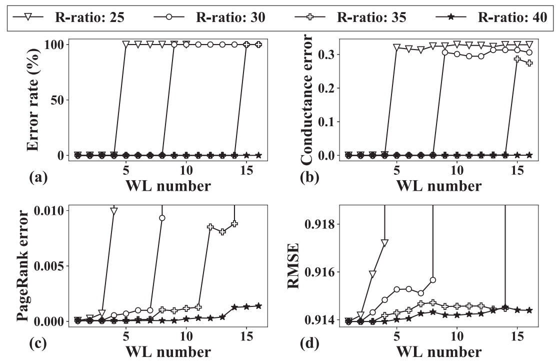

Fig. 11. Impact of WL_{max} on the computing accuracy of (a) CC (b) CT (c) PR (d) CF.

3) Case study - improving the reliability for PR: Based on our analysis using GraphRSim, we propose and evaluate a technique that dynamically tunes the reference value in ADC to reduce the validation error of PR. PR has a special property, that is, its sum of PR scores ($PR_{sum} = \sum_{u \in V} PR(u)$) should be a constant ($Ideal_{sum} = 1 - r \cdot \sum_{u \in V} \{out\_deg(u) == 0\}/|V|$) if the graph is an undirected graph. For directed graphs, this property also holds in the first iteration. For example, in Fig. 4(a), its PR_{sum} is equal to $0.04 + 0.2 + 0.2 + 0.12 + 0.28 = 1 - 0.8 \cdot 1/5 = 0.84$. Thus, for each iteration, if $PR_{sum} > 2 \cdot Ideal_{sum}$, we dynamically shift Ref_0 towards right by 500nA to prevent overflows. If PR_{sum} is just slightly higher than $Ideal_{sum}$ ($2 \cdot Ideal_{sum} > PR_{sum} > Ideal_{sum}$), the overflow situation has been mitigated but there are still some small values being wrongly sensed as larger values, so we shift Ref_1 towards right by 500nA to further reduce errors. Results in Table II shows that shifting only Ref_0 (Dyn_ref0) can eliminate overflows. Shifting both Ref_0 and Ref_1 right (Dyn_ref0+ref1) can further improve the computing accuracy. For arXiv and P2P, when $WL_{max} = 4$, Dyn_ref0+ref1 can even change the result from higher than $12 \cdot 10^{-4}$ error to converge (error$< 10^{-4}$).

TABLE II
VALIDATION ERROR OF PAGERANK WITH VARIOUS REFERENCE SCHEMES

Dataset	WL_{max}	Dynamic Reference Schemes		
		No_dyn	*Dyn_ref0*	*Dyn_ref0+ref1*
Facebook	**4**	99.2	99.2	18.7
	8	Overflow	145	119
arXiv	**4**	43.7	43.7	0.7
	8	Overflow	84	95.2
Email	**4**	146	146	41
	8	Overflow	247	218
Wiki-Vote	**4**	28	26.6	7.7
	8	Overflow	51	44.1
P2P	**4**	12.8	12.8	0.3
	8	Overflow	28.3	33.2

Note: the unit for the validation errors in this table is 10^{-4}.

4) Case study - improving the reliability for CC: Based on the processing characteristics of CC and the analysis shown in Fig. 11, we propose and evaluate a dynamic reference selection scheme to improve the reliability. CC is a $Class$-2 algorithm that is only interested in whether the output value of each bitline is zero or none-zero. As explained in Section III-C, ReRAM-induced errors cannot be recovered if a zero is wrongly sensed as a non-zero value. Since CC activates various number of WLs at each iteration, as shown in Fig. 7, a dynamic Ref_0 selection scheme can help to avoid such an unrecoverable error. From Fig. 11, we find that, if the baseline ReRAM device with $R\text{-}ratio = 25$ is applied, CC can achieve 0% error rate if less than five WLs are activated concurrently. Thus, we design a dynamic reference selection scheme that uses the original Ref_0 as the reference value during the iterations with less than five activated WLs, while the original Ref_1 is used to distinguish zero and non-zero values when more than five WLs are activated at the same time. Results in Table III show that the proposed method can significantly reduce the error rates, especially when $WL_{max} = 16$, since using Ref_1 as the reference value is more effective if there are more output values that are greater than two. Facebook dataset is not evaluated here as it has only one connected component.

TABLE III
ERROR RATE OF CC WITH DIFFERENT REFERENCE SCHEMES

Dataset	Using dynamic scheme?	NO	YES	
	WL_{max}	$>= 5$	*8*	*16*
arXiv		99.99%	0%	0%
Email		8.17%	0%	0%
Wiki-Vote		99.98%	0.048%	0%
P2P		0.045%	0.045%	0%

VI. RELATED WORK

GraphR is the first work targeting graph processing on ReRAM [1]. Some other ReRAM-based graph processing accelerators have also been proposed, such as GraphSAR [2] and RAGra [3]. GraphSAR studies the sparsity-aware design while RAGra targets on designing better mapping schemes for 3D ReRAM based accelerators. The reliability issue is not considered in these prior works. Several simulation platforms [5], [7], [13] are proposed to analyze the performance and reliability of ReRAM-based deep learning accelerators, and some error correction techniques are proposed [14]. However, reliability analysis for graph processing is not covered in these prior studies. Although the device-level ReRAM model is similar, graph algorithms have different computation flows and are not as robust as deep learning. To the best of our knowledge, we are the first work that analyzes the impact of ReRAM-induced errors on graph processing accuracy.

VII. CONCLUSION

ReRAM-based processing-in-memory accelerators provide a promising solution to improve the energy efficiency of graph processing. However, the intrinsic stochastic variation in ReRAM devices makes its computation unreliable. We propose a simulation framework, GraphRSim, to jointly analyze the impact of non-ideal device and the feature of the target algorithm on the computing accuracy of graph processing. It is expected that GraphRSim can help chip designers to select better design options and develop reliability optimization techniques.

ACKNOWLEDGEMENT

This work was supported in part by research grants from the Ministry of Science and Technology of Taiwan (MOST107-2218-E-001-004-MY3).

REFERENCES

[1] L. Song *et al.*, "GraphR: Accelerating graph processing using ReRAM," in *HPCA*, 2018.
[2] G. Dai *et al.*, "GraphSAR: A sparsity-aware processing-in-memory architecture for large-scale graph processing on ReRAMs," in *ASP-DAC*, 2019.
[3] Y. Huang *et al.*, "RAGra: Leveraging monolithic 3D ReRAM for massively-parallel graph processing," in *DATE*, 2019.
[4] H.-S. P. Wong *et al.*, "Metal-oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, June 2012.
[5] M. Lin *et al.*, "DL-RSIM: A simulation framework to enable reliable ReRAM-based accelerators for deep learning," in *ICCAD*, 2018.
[6] M. Suri *et al.*, "Neuromorphic hybrid RRAM-CMOS RBM architecture," in *NVMTS*, 2015.
[7] M. K. F. Lee *et al.*, "A system-level simulator for RRAM-based neuromorphic computing chips," *TACO*, vol. 15, no. 4, pp. 64:1–64:24, Jan. 2019.
[8] K. C. Hsu *et al.*, "A study of array resistance distribution and a novel operation algorithm for WOx ReRAM memory," in *SSDM*, 2015.
[9] F. Mangasa *et al.*, "Status and prospects of ZnO-based resistive switching memory devices," *Nanoscale Res. Lett.*, vol. 11, p. 368, Aug. 2016.
[10] X. Chen and J. Feng, "Improvement in R_{off}/R_{on} ratio and reset current via combining compliance current with multilayer structure in tantalum oxide-based RRAM," *Appl. Phys. A*, vol. 120, no. 1, pp. 67–73, Jul. 2015.
[11] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.
[12] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and context," *TiiS*, vol. 5, no. 4, pp. 19:1–19:19, Dec. 2015.
[13] P. Chen *et al.*, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *TCAD*, vol. 37, no. 12, pp. 3067–3080, Dec 2018.
[14] B. Feinberg *et al.*, "Making memristive neural network accelerators reliable," in *HPCA*, 2018.