Bias by Design: Diversity Quantification to Mitigate Structural Bias Effects in AIG Logic Optimization

Isabella Venancia Gardner,*†, Marcel Walter‡, Yukio Miyasaka§, Robert Wille‡¶, and Michael Cochez†

*Universiteit van Amsterdam, Netherlands
Email: bella.gardner@student.uva.nl
†Vrije Universiteit Amsterdam, Netherlands
Email: m.cochez@vu.nl
‡Chair for Design Automation, Technical University of Munich, Germany
Email: {marcel.walter, robert.wille}@tum.de
§University of California, Berkeley, USA
Email: yukio_miyasaka@berkeley.edu
¶Software Competence Center Hagenberg GmbH (SCCH), Austria

***Abstract*—And-Inverter Graphs (AIGs) are a fundamental data structure in logic optimization, widely used in modern electronic design automation. A persistent challenge in AIG optimization is *structural bias*, where the initial graph structure strongly influences optimization quality by restricting the search space, often resulting in subpar outcomes. Existing methods address this issue by running multiple optimization workflows in parallel, relying on a trial-and-error approach that lacks a systematic way to measure *structural diversity* or assess effectiveness, making them computationally expensive and inefficient. This paper introduces a novel framework for systematically evaluating and reducing structural bias by measuring structural diversity, defined as the degree of dissimilarity between AIG graphs. Several traditional graph similarity measures and newly proposed AIG-specific metrics, including the *Rewrite*, *Refactor*, and *Resub Scores*, are explored. Results reveal limitations in traditional graph similarity metrics and highlight the effectiveness of the proposed AIG-specific measures in quantifying structural dissimilarity. Notably, the *RRR Score* shows a strong correlation (Pearson correlation coefficient, $r = 0.79$) with post-optimization structural differences, demonstrating the reliability of the metric in capturing meaningful variations between AIG structures. This work addresses the challenge of quantifying structural bias and offers a methodology that can potentially improve optimization outcomes, with future extensions applicable to other logic graph types.**

I. Introduction & Motivation

With the increasing complexity of digital circuits and rising manufacturing costs in advanced technology nodes, minimizing chip area and improving performance have become critical challenges. *And-Inverter Graphs* (AIGs) are a state-of-the-art data structure used in technology-independent logic optimization, supporting a wide range of synthesis and optimization tasks that are fundamental in modern electronic design automation [1]–[4].

Many optimization algorithms for AIGs have been developed, focusing on reducing circuit area, delay, and power consumption [1]–[6]. However, these methods are often limited by *structural bias*, a well-recognized phenomenon in the community [7]. Structural bias refers to the sensitivity of optimization algorithms to the initial structure of an AIG, which restricts the search space and hinders the discovery of globally optimal solutions. Consequently, even advanced optimization flows can become trapped in local minima, reducing their overall effectiveness. This challenge has been demonstrated in recent *IWLS Programming Contests* (2022–2024), where many entries struggled to overcome this bias.

Despite structural bias being widely recognized, systematic quantification or mitigation strategies have seen limited progress. Current strategies often attempt to bypass the issue by running multiple synthesis and optimization processes in parallel, hoping that one will outperform the others [8]. While this trial-and-error approach introduces some *structural diversity*, it lacks a systematic method for measuring the degree of diversity or evaluating its effectiveness in addressing structural bias before the optimization process is complete. As a result, this approach remains both computationally expensive and resource-inefficient.

In the field of logic optimization, methods focus on enhancing structural diversity, which is defined as the degree of structural dissimilarity between AIGs. Structurally dissimilar AIGs are known to follow distinct optimization paths, as each structure presents unique opportunities and constraints within the optimization search space [9]. By promoting greater structural diversity, these methods increase the likelihood of exploring alternative optimization paths, potentially leading to more optimal results. A more effective approach would involve analyzing the initial AIG structure to predict optimization outcomes, enabling more informed decisions, reducing computation time, and allowing for early termination of unpromising runs. This research systematically investigates structural bias and presents several key contributions:

1) the development of a framework for quantitatively assessing AIG structural dissimilarity, as outlined in Section III-A,
2) the introduction of fast metrics based on AIG optimization, such as *Rewrite*, *Refactor*, and *Resub Scores*, to measure structural diversity, detailed in Section IV-B,
3) the application of this framework (Section V) using traditional graph theory metrics (Section IV-A), AIG attributes (Section IV-B), and the proposed optimization-based scores (Section IV-B), and
4) a statistical analysis revealing a strong correlation ($r = 0.79$) between structural bias and the *RRR Score*, presented in Section VI.

978-3-9826741-0-0/DATE25/© 2025 EDAA

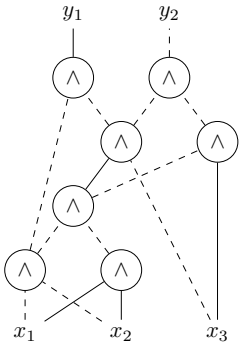

Figure 1: AIG of a full adder with carry $y_1 = \langle x_1 x_2 x_3 \rangle$ and sum $y_2 = x_1 \oplus x_2 \oplus x_3$.

II. And-Inverter Graphs (AIGs)

In modern logic synthesis, efficiently representing and manipulating Boolean functions is a key challenge as circuit complexity grows. Traditional methods like truth tables, SOP, and BDDs become impractical for large designs due to their exponential size growth. *And-Inverter Graphs* (AIGs) offer a more scalable, compact solution, making them essential in contemporary logic design workflows.

A. Data Structure

An AIG is a directed acyclic graph comprised of primary input nodes, which correspond to the inputs of a logic circuit, and AND nodes, each representing a two-input AND gate. A key feature of AIGs is the use of inversion tags on edges, which represent logical negations (NOT operations) without adding extra nodes. The primary output pointers represent the final logic expressions implemented by the AIG. The hierarchical structure of AIGs allows for the decomposition of complex Boolean functions.

Figure 1 displays a visual representation of an AIG that implements the full adder function in 7 AND nodes where dashed lines define inverted edges. Here, $y_1 = \langle x_1 x_2 x_3 \rangle$, i. e., the majority-of-three of all primary inputs, is the carry, and $y_2 = x_1 \oplus x_2 \oplus x_3$, i. e., the three-input XOR of all inputs, is the sum output. It should be noted that primary outputs are generally not regarded as nodes.

B. Synthesis & Optimization

The AIG's minimalist structure and focus on a restricted set of logic primitives, enable various efficient optimizations and powerful iterative transformations that preserve functional equivalence. This allows for the application of aggressive optimizations that reduce circuit area, improve delay, and minimize power consumption.

AIG *Rewriting* [1], [2] is a widely used technique that reduces node count by replacing subgraphs (cuts) with smaller pre-computed equivalents through local greedy transformations. Using fast cut enumeration, NPN equivalence classes, and efficient truth table manipulations, AIG rewriting balances area and delay optimization, making it a standard in synthesis flows.

Refactoring [2], [3], a variant of rewriting, takes a broader approach by targeting larger AIG subgraphs for restructuring. Instead of focusing on local, incremental changes, refactoring collapses and reorganizes more extensive portions of the graph, leading to potentially greater reductions in both size and depth.

It is particularly effective when used alongside rewriting in iterative optimization flows, as it enhances the overall scope of potential improvements in area and delay.

Resubstitution [3], [4] re-expresses the logic of an AIG node using other nodes, called divisors, which are outside the node's transitive fan-out, ensuring cycle-free reuse. The key is to find valid divisors that replace a target node's logic while preserving the circuit's functional equivalence. An extension, *Simulation-Guided Resubstitution* [10], enhances efficiency by using simulation patterns to filter non-promising candidates, reducing computational cost and enabling scalability for larger, more complex circuits.

C. Structural Bias

It is well recognized in the community that most optimization algorithms are prone to a phenomenon known as *structural bias* [7], which presents a challenge to achieve high-quality logic optimization. Algorithms, such as rewriting, refactoring, and resubstitution, operate locally within the graph structure, and consequentially, the quality of the final circuit can vary greatly depending on the initial AIG configuration. Thus, structural bias, an inherent sensitivity, arises because the local structure influences the set of feasible transformations and optimizations that can be applied, which can trap the optimizer in suboptimal results.

In this work, *structural diversity* is defined as the degree of structural dissimilarity between graphs, expanding the optimization search space and preventing bias by ensuring different optimization paths are explored. Although not widely recognized as a formal term, structural diversity plays a central role in mitigating bias by increasing the chances of avoiding local minima.

Consequently, optimization flows often require manual fine-tuning to individual circuits or employ randomization techniques to explore a broader portion of the optimization space and mitigate the effects of structural bias [8]. Another promising approach involves *choices* [7], i. e., resynthesizing parts of the AIG using multiple different strategies, selecting the best result at each step to avoid being locked into a particular structural form. Despite these efforts, the issue of structural bias in AIG optimization remains largely unresolved, requiring further exploration and more sophisticated methodologies to expand the search space and improve the robustness of AIG optimization flows.

III. The Proposed Framework

Structural bias in logic synthesis optimization, as discussed in Section II-C, remains a challenge. This framework addresses this gap by developing metrics that compare AIG dissimilarity and strongly correlate with optimization outcomes, such as node reduction. By aligning graph similarity metrics with optimization performance, this provides a systematic, reliable alternative to trial-and-error methods.

A. Experimental Framework

The proposed framework consists of four main steps, designed to systematically explore structural diversity in AIGs and assess the relationship between a pre-optimization graph similarity measure and post-optimization graph structure. The steps are as follows:

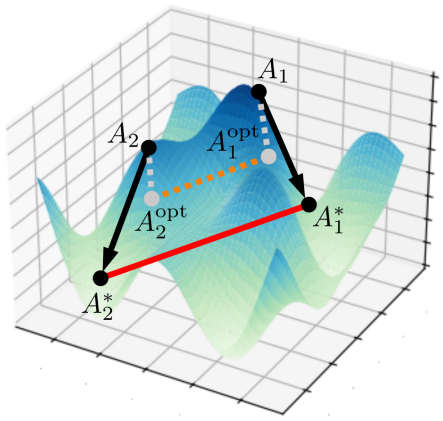

Figure 2: Conceptual illustration of an optimization search space. A_1 and A_2 represent different starting points for AIG optimization, with A_1^* and A_2^* as their optimized endpoints after the full optimization process. The red line between A_1^* and A_2^* shows the Relative Optimizability Difference, representing the final difference in node count reduction. A_1^{opt} and A_2^{opt} represent the positions after one optimization step (resubstitution, rewrite, or refactor). Black lines show optimization trajectories, and the dotted orange line approximates the Relative Optimizability Difference.

1) Generating multiple starting points from function specifications: AIGs are synthesized from a given function (e. g., a truth table) using different synthesis algorithms to create functionally equivalent and structurally diverse graphs.

2) Applying similarity measures: Graph similarity measures are applied to quantify pairwise structural dissimilarity between the generated AIGs. The choice of measure depends on the structural aspects relevant to optimization.

3) Optimizing the generated AIGs: Each AIG undergoes optimization for node reduction using a high-effort optimization flow chosen by the user. The optimization algorithm should be consistent.

4) Calculating the correlation between similarity scores and structural diversity benchmark: Finally, the framework computes the correlation between the similarity scores (obtained in step 2) and the *Relative Optimizability Difference* introduced in Section III-B. This step evaluates the effectiveness of the chosen similarity metric with stronger correlations indicating a better representation for AIG structural dissimilarity.

B. Relative Optimizability Difference

The *Relative Optimizability Difference* quantifies how structurally different two AIGs are based on their optimization outcomes. Specifically, it measures the normalized distance between two AIGs after optimization in the optimization search space. Though not usable pre-optimization, this measure serves as a benchmark for assessing structural dissimilarity post-optimization. A strong correlation between a similarity metric and this measure suggests that the metric effectively captures pre-optimization structural dissimilarity.

Figure 2 illustrates that structurally dissimilar AIGs follow distinct optimization paths, as each structure presents unique opportunities and constraints [9]. In this figure, two functionally equivalent AIGs, A_1 and A_2, synthesized using different algorithms, start from different positions in the optimization search space. After optimization, they reach new states, A_1^* and A_2^*. The black dotted lines trace their optimization paths, where lower points correspond to fewer nodes. The red line between A_1^* and A_2^* represents the Relative Optimizability Difference, reflecting the difference in final node reductions i. e. the distance between endpoints in the optimization search space. A greater distance indicates more distinct optimization outcomes and greater structural dissimilarity.

Since node minimization is the primary goal, the difference in final gate counts between two optimized AIGs serves as a useful proxy for their structural dissimilarity. Mathematically, the Relative Optimizability Difference is defined as:

$$\text{ROD}(A_1, A_2) \coloneqq \frac{|\text{G}(A_1^*) - \text{G}(A_2^*)|}{\max(\text{G}(A_1^*), \text{G}(A_2^*))} \tag{1}$$

Where A_1^* and A_2^* are the optimized versions of A_1 and A_2 after the optimization process, and $\text{G}(A^*)$ represents the total number of gates (nodes) in the optimized AIG A^*.

IV. AIG Dissimilarity Measures

Accurately measuring structural differences between AIGs is crucial for understanding how various optimization techniques affect their overall structure. This section introduces and evaluates a range of graph similarity measures, both traditional and AIG-specific, that can be used to quantify structural dissimilarity.

A. Integrated Traditional Graph Similarity Measures

Graph similarity measures are commonly used to quantify structural properties. This work adapts four traditional metrics from [11] to AIGs to assess their potential in calculating AIG structural diversity. The selected metrics—feature-based, network alignment, kernel-based, and spectral analysis—offer diverse perspectives on structural diversity. [1]

1) Graph Metric Considerations for AIGs: Adapting graph metrics to AIGs involves several challenges. AIGs represent Boolean functions as directed graphs with inverted edges (NOT gates), so metrics must account for edge directionality, disconnected components, and edge weights. Node correspondence is also affected by optimizations that alter internal nodes, while the varying sizes and sparsity of AIGs require additional consideration. To apply metrics designed for undirected graphs, AIGs are converted by removing directionality and merging normal and inverted edges. While this preserves topology, it abstracts functional logic.

2) Vertex-Edge Overlap (VEO): Vertex-Edge Overlap (VEO) [12] measures graph similarity by comparing shared vertices and edges. It is simple, computationally efficient, and well-suited for large, sparse graphs like AIGs. To facilitate comparison, AIGs are converted to undirected edge lists, removing inversion and directionality. Consistent node numbering ensures valid comparisons. VEO computes similarity by taking the ratio of the shared vertices and edges between two graphs to the total number of vertices and edges. It returns a value of 1 for identical graphs and 0 for completely dissimilar ones. With a linear time complexity, VEO provides a fast comparison, although it abstracts AIG-specific details by treating normal and inverted edges equally.

[1] Graph metrics adapted from `https://github.com/peterewills/NetComp`.

3) NetSimile: NetSimile [13] compares graphs by analyzing aggregated structural features rather than relying on direct node-to-node comparisons, making it suitable for AIGs modified by optimization. Key features like node degree, clustering coefficient, and egonet properties are extracted for each node and aggregated using statistical measures such as median, mean, and skewness. The aggregated feature vectors for two AIGs are compared using the Canberra distance, which evaluates the difference between the two sets of aggregated statistics. The exact algorithm can be found in [13]. NetSimile is robust to changes in node structure and has a time complexity of $\mathcal{O}(n \log n)$. A lower Canberra distance reflects greater similarity between the graphs, while a higher value indicates more structural differences.

4) Weisfeiler-Lehman (WL) Kernel Similarity: The *Weisfeiler-Lehman* (WL) *kernel* [14] is a graph similarity method that iteratively relabels nodes to capture subgraph patterns. For its application to AIGs, the graphs are first converted to undirected versions. After conversion, the WL kernel relabels nodes based on their local subgraph structure, enabling an effective comparison between two graphs. The similarity between the graphs is calculated by comparing these relabeled nodes, with the exact formula provided in [14]. With linear time complexity in the number of edges, the WL kernel efficiently handles large AIGs while capturing structural changes resulting from synthesis and optimization. The similarity score ranges from 0 (indicating no matching substructures) to higher values, reflecting stronger structural alignment.

5) Adjacency Spectral Distance: The *Adjacency Spectral Distance* (ASD) [11] measures graph similarity by comparing the eigenvalues of their adjacency matrices. First, AIGs are converted to undirected edge lists, from which adjacency matrices are generated. The eigenvalue spectra of the matrices for graphs G and G' are then compared using the Euclidean distance between their respective eigenvalues $\lambda_{A,i}$ and $\lambda_{A',i}$. The ASD score ranges from 0 for identical graphs to higher values for more dissimilar graphs. The method has a time complexity of $\mathcal{O}(nk^2)$, where n is the number of nodes and k is the number of eigenvalues.

6) Other Methodologies: In addition to the methods in [11], neural network-based approaches like *Graph Neural Networks* [15] embed graphs into vector spaces for similarity comparisons. While effective, these methods are computationally expensive and require extensive training, making them beyond the scope of this work. This paper also excludes methods such as *DeltaCon* [16], *Resistance Distance* [17], and *Graph Edit Distance* [18] due to their computational complexity and dependence on node correspondence.

B. Proposed AIG-Specific Graph Similarity Measures

This section introduces AIG-specific similarity measures to better capture unique structural characteristics. These include attribute-based metrics as well as optimization-based scores derived from rewriting, refactoring, and resubstitution (cf. Section II-B).

1) AIG Attribute-based Similarity Measures: To directly capture key AIG structural features, attribute-based metrics like *Relative Gate Count* (RGC) and *Relative Level Count* (RLC) measure differences in the number of logic nodes (gates) and levels, respectively, between two AIGs. These are computed as normalized differences, exemplified by the RGC:

$$\mathrm{RGC}(A_1, A_2) \coloneqq \frac{|G(A_1) - G(A_2)|}{G(A_1) + G(A_2)} \tag{2}$$

RGC and RLC offer a highly computational efficient way to assess structural differences because they rely solely on global AIG attributes—gate count and level depth—that are typically maintained throughout the AIG's construction. This eliminates the need for complex graph transformations. While less detailed, these scores provide a practical gauge of structural diversity in AIGs.

2) Resub, Rewrite, and Refactor Scores: Attribute-based similarity scores, while highly efficient, may not fully capture the deeper structural features that influence AIG optimization. To address this, algorithms like rewriting, refactoring, and resubstitution (discussed in Section II-B) are used to evaluate AIG optimization paths.

This work introduces the *Rewrite*, *Refactor*, and *Resub Scores*, which quantify the relative optimization success between two functionally equivalent AIGs after a single round of optimization. These scores are based on the difference in gate count reduction achieved by each algorithm, and are calculated as follows:

$$\mathrm{RS}(A_1, A_2) \coloneqq \left| \frac{G(A_1) - G(A_1^{\mathrm{opt}})}{G(A_1)} - \frac{G(A_2) - G(A_2^{\mathrm{opt}})}{G(A_2)} \right| \tag{3}$$

where R refers to one of the three algorithms (rewriting, refactoring, or resubstitution), and A^{opt} is the optimized AIG after applying R to A. The function $G(A)$ gives the gate count of A.

These scores are significant because rewriting, refactoring, and resubstitution are key components of modern optimization flows, and their results provide insights into the long-term optimization potential of AIGs. By comparing gate count reductions, these scores reveal structural differences that attribute-based metrics might miss.

Figure 2 illustrates the process. A_1 and A_2 are starting points, while A_1^{opt} and A_2^{opt} are their positions after one optimization step. Black lines show optimization paths, the orange dotted line approximates the Relative Optimizability Difference, and the red line shows the final difference between A_1^* and A_2^* after full optimization.

The key assumption is that different levels of gate count reduction between AIGs reflect deeper structural differences. While full optimization flows are computationally expensive, these algorithms offer an efficient way to approximate structural differences without exhaustive synthesis.

3) RRR Score: The *RRR Score* offers a comprehensive measure of AIG structural diversity by combining the effects of rewriting, refactoring, and resubstitution. It compares two AIGs based on their optimizability through these three techniques, where r_1, r_2, r_3 represent the reductions achieved from rewriting, refactoring, and resubstitution, respectively. The RRR Score is calculated as the Euclidean distance between the reduction vectors, providing a quantitative assessment of the structural differences between the two AIGs:

$$\mathrm{RRR}(A_1, A_2) \coloneqq \sqrt{\sum_{i=1}^{3} \left(r_i(A_1) - r_i(A_2)\right)^2} \tag{4}$$

Table I: Similarity scores with Pearson correlation and confidence intervals for traditional graph similarity measures.

Similarity Measure	r	CI
Vertex-Edge Overlap	-0.36	$[-0.40, -0.32]$
NetSimile	0.33	$[0.29, 0.37]$
Weisfeiler-Lehman Kernel	-0.27	$[-0.31, -0.22]$
Adjacency Spectral Distance	0.21	$[0.17, 0.25]$

This score quantifies the overall difference in how the two AIGs respond to optimization. A lower score indicates similar optimization behavior, while a higher score suggests significant structural differences.

The RRR Score provides a comprehensive measure of AIG structural diversity by leveraging multiple optimization techniques. Although it is less efficient than calculating individual optimization scores, it is still far more efficient than a full optimization flow, offering a good balance between accuracy and computational cost.

V. Framework Application

This section outlines the experimental setup used to evaluate traditional graph similarity measures (Section IV-A) and AIG-specific methods (Section IV-B). The goal is to test the framework and demonstrate how the results in Section VI were obtained. The full dataset, scripts, and processes are publicly accessible on GitHub.[2]

The benchmark set consisted of 100 function specifications from the IWLS Programming Contest 2024, selected for diversity in function types (e. g., random functions, cryptographic algorithms, sorting networks, arithmetic operations, and neural network components).[3] Due to scalability constraints, 87 of them were synthesized into AIGs using seven distinct scripts from *ABC* [5] and *Espresso* [6], including but not limited to methods like Binary Decision Diagrams (BDDs), SOP fast extract, Disjoint-support Decomposition (DSD), and LUT bi-decomposition to generate structurally diverse AIGs from the function specification. Pairwise similarity measures were then applied to quantify structural diversity between these functionally equivalent AIGs.

After synthesis, each AIG was separately optimized using one of the following high-effort scripts from ABC: `orchestrate` [19], `dc2` [2], and `&deepsyn` [20]. The Relative Optimizability Difference was calculated for each AIG pair, providing a benchmark for structural changes induced by optimization.

Finally, Pearson correlation coefficients were used to assess the correlation between similarity scores and the Relative Optimizability Difference, with $95\,\%$ confidence intervals calculated using Fisher transformations. Scatter plots with trendlines were generated to visualize the relationship between similarity metrics and optimization outcomes, providing insights into the effectiveness of each metric in capturing structural diversity and optimization potential. The full data set and all plots can be found in the supplementary GitHub repository.

VI. Experimental Results

This section presents the experimental results obtained from the application of the framework proposed in Section III-A, where multiple graph similarity measures were evaluated for their correlation with AIG structural dissimilarity. Due to space limitations, only an amalgamation of key results can be shown here. The scripts, the complete raw results, and all generated plots will be provided online upon publication.

Table II: Graph dissimilarity measure with Pearson correlation and confidence intervals for proposed AIG-specific metrics evaluated against high-effort optimization scripts.

	`orchestrate`		`dc2`		`&deepsyn -T 10`	
Measure	r	CI	r	CI	r	CI
RGC	0.42	$[0.38, 0.45]$	0.47	$[0.43, 0.50]$	0.75	$[0.73, 0.77]$
RLC	0.41	$[0.37, 0.45]$	0.43	$[0.39, 0.47]$	0.50	$[0.47, 0.54]$
Rewrite Score	0.43	$[0.39, 0.47]$	0.52	$[0.49, 0.56]$	0.31	$[0.26, 0.35]$
Refactor Score	0.48	$[0.45, 0.52]$	0.54	$[0.50, 0.57]$	0.34	$[0.30, 0.38]$
Resub Score	0.72	$[0.70, 0.75]$	0.52	$[0.48, 0.55]$	0.38	$[0.34, 0.42]$
RRR Score	0.79	$[0.78, 0.81]$	0.68	$[0.66, 0.71]$	0.45	$[0.42, 0.49]$

A. Integrated Traditional Graph Similarity Measures

The performance of traditional graph similarity measures for AIGs varied, as shown in Table I. Both positive and negative correlations were observed because VEO and WL Kernel reflect greater similarity with higher values, while NetSimile and ASD do the opposite. Since the focus is on correlation strength rather than direction, both types of correlations are reported.

ASD demonstrated the weakest correlation ($r = 0.21$), likely due to its dependence on eigenvalue spectra, which may not be well-suited for sparse graphs like AIGs. NetSimile showed a slightly higher correlation ($r = 0.33$), but its use of aggregated statistics may miss important functional differences. VEO ($r = -0.36$) and WL Kernel ($r = -0.27$) also performed poorly, reflecting limitations in capturing the structural nuances of AIGs.

Overall, these measures produced weak-to-moderate correlations, underscoring the limitations of traditional metrics in capturing AIG-specific characteristics. Although the results discussed here are based on the `orchestrate` flow, similar poor performance was observed with other optimization strategies, showing no consistent improvement in correlation.

B. Proposed AIG-Specific Graph Similarity Measures

This section details the experimental results of applying the proposed framework to the AIG-specific graph similarity measures discussed in Section IV-B. Table II summarizes the Pearson correlations and confidence intervals for these metrics across various high-effort optimization scripts, showcasing their effectiveness in capturing structural diversity.

1) Proposed AIG Attribute Similarity Scores: The RGC and RLC metrics simplify AIG analysis by focusing on gate count and logic levels, rather than abstract structural features. With AIGs that underwent `orchestrate` optimization, these metrics showed moderate correlations with the Relative Optimizability Difference ($r = 0.42$ and $r = 0.41$). While useful for capturing broader trends, they may miss finer details, especially in localized transformations typical of `orchestrate`.

However, these metrics performed better with `&deepsyn`, which involves broader changes like LUT mapping and resynthesis. RGC, in particular, had a much stronger correlation ($r = 0.75$), suggesting that `&deepsyn`'s more comprehensive transformations align well with node count.

[2] https://github.com/bellavg/aig-similarity

[3] https://www.iwls.org/iwls2024/

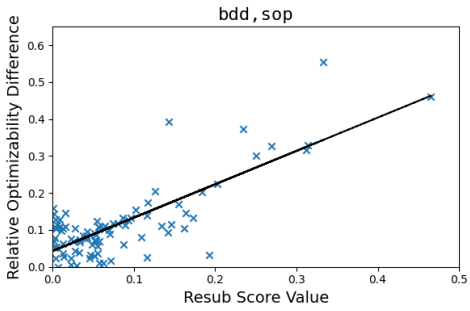

Figure 3: The correlation between Resub Score and Relative Optimizability Difference for two example AIG datasets with a correlation of $r = 0.79$.

Compared to `orchestrate`'s incremental optimizations, `&deepsyn` allows RGC to reflect deeper structural changes. This highlights that while these metrics offer useful insights, their effectiveness increases when paired with broader optimization strategies like `&deepsyn`, enhancing both computational efficiency and correlation with outcomes.

2) Rewrite, Refactor, and Resub Scores: From the individual algorithm metrics, the Resub Score showed the strongest correlation with the Relative Optimizability Difference ($r = 0.72$, CI: $[0.70, 0.75]$), demonstrating its effectiveness in capturing structural changes post-optimization, especially in `orchestrate`, which heavily favors resubstitution. This high correlation, illustrated in Figure 3, reflects deeper functional transformations within the AIG and highlights the Resub Score's strength in resubstitution-driven optimizations. However, its performance drops to $r = 0.52$ with `dc2` and $r = 0.38$ with `&deepsyn`, where resubstitution plays a lesser role.

The Rewrite and Refactor Scores showed moderate correlations across different optimization scripts, with $r = 0.43$ and $r = 0.48$ for `orchestrate`, and slightly better performance with `dc2`. Their lower sensitivity to finer structural details, along with higher runtime overheads, limits their efficiency compared to Resub.

3) RRR Score: To overcome the limitations of individual synthesis script scores, the RRR Score, which combines the effects of resubstitution, refactoring, and rewriting, was introduced. This combined score leveraged the strengths of all three techniques, achieving the highest correlation with the Relative Optimizability Difference ($r = 0.79$). The RRR Score outperformed each individual score and remained robust across intensive optimization workflows, such as `dc2`, maintaining a high correlation of $r = 0.68$. However, its performance was more moderate with `&deepsyn`, achieving a correlation of $r = 0.45$—higher than the individual scores but still lower than the RGC. This result demonstrates that the combined approach provides a more comprehensive view of structural diversity and significantly improves the accuracy of predicting optimization outcomes, though its effectiveness varies depending on the optimization strategy used.

VII. Discussion & Future Work

The results underscore the importance of using optimization-specific metrics for AIG structural analysis, particularly in overcoming structural bias. Metrics like the RRR Score proved highly effective in optimization flows that rely on a combination of resubstitution, rewriting, and refactoring, such as `orchestrate` and `dc2`. The RRR Score delivered the highest correlations ($r = 0.79$) across these flows, demonstrating its versatility in capturing diverse structural transformations and offering reliable insights into optimization potential.

In contrast, the RGC metric performed best in the `&deepsyn` optimization flow, where broader structural changes, like LUT mapping and node reduction, play a key role. This highlights the importance of selecting metrics that align with the specific characteristics of the optimization technique being used, as structural bias is closely tied to the type of optimization applied. The differences in metric performance underscore that structural bias is not a one-size-fits-all challenge; instead, it is dependent on the optimization method, and thus, the selection of metrics must be tailored accordingly.

Looking forward, the flexibility of the proposed framework allows for its seamless application to more complex networks than evaluated here as well as to other graph types, such as XAGs or MIGs, expanding its potential beyond AIGs. The evaluation in this work was conducted on a set of relatively small networks to enable the application of a wide range of synthesis methodologies, thereby providing diverse and representative starting points for the analysis. This choice ensured a robust understanding of structural metrics across various optimization techniques. However, the findings and principles established in this study are equally applicable to real-world scenarios involving large-scale networks. The same metrics can capture structural characteristics and optimization potential in industrial-scale AIGs, where managing structural bias is even more critical due to the increased complexity of synthesis flows and the greater impact of optimization decisions.

While the current work covers three primary optimization scripts—`orchestrate`, `dc2`, and `&deepsyn`—testing additional optimization techniques could provide further insights into the relationship between structural bias and optimization outcomes.

VIII. Conclusion

In this work, the challenge of structural bias in AIG optimization was addressed by developing a framework for quantitatively assessing AIG structural dissimilarity. Metrics such as the Rewrite, Refactor, and Resub Scores, along with traditional graph theory measures, were introduced to evaluate how structural diversity impacts optimization outcomes. The RRR Score proved to be the most effective, achieving a strong correlation ($r = 0.79$) with optimization flows like `orchestrate`. Meanwhile, the RGC metric showed superior performance with the `&deepsyn` optimization flow, highlighting the need for careful metric selection based on the specific characteristics of the optimization technique.

The findings indicate that structural bias is closely linked to the type of optimization applied, and thus, metric selection must be tailored to the method in use. By providing a systematic approach to measure structural diversity, this framework enhances the precision of optimization flows and offers practical tools for improving logic design outcomes. Despite the promising results, further refinement and validation of the framework across a broader set of AIG optimization scenarios could provide additional insights. Additionally, the flexibility of the proposed framework allows for potential application to other graph types, such as XAGs or MIGs. Future work could explore additional optimization techniques to further investigate the relationship between structural bias and optimization outcomes.

REFERENCES

[1] P. Bjesse and A. Borälv, "DAG-Aware Circuit Compression for Formal Verification," in *ICCAD*, 2004, pp. 42–49.

[2] A. Mishchenko, S. Chatterjee, and R. Brayton, "DAG-Aware AIG Rewriting: A Fresh Look at Combinational Logic Synthesis," in *DAC*, 2006, pp. 532–535.

[3] A. Mishchenko and R. Brayton, "Scalable Logic Synthesis using a Simple Circuit Structure," in *IWLS*, vol. 6, 2006, pp. 15–22.

[4] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, 1st ed. McGraw-Hill Higher Education, 1994.

[5] R. Brayton and A. Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool," in *Computer Aided Verification*. Springer, 2010, pp. 24–40.

[6] R. K. Brayton, *Logic Minimization Algorithms for VLSI Synthesis*. Springer, 1984.

[7] S. Chatterjee, A. Mishchenko, R. K. Brayton, X. Wang, and T. Kam, "Reducing Structural Bias in Technology Mapping," *TCAD*, vol. 25, no. 12, pp. 2894–2903, 2006.

[8] Y. Miyasaka, "Transduction Method for AIG Minimization," in *ASP-DAC*, 2024, pp. 398–403.

[9] W. L. Neto, Y. Li, P.-E. Gaillardon, and C. Yu, "FlowTune: End-to-End Automatic Logic Optimization Exploration via Domain-Specific Multiarmed Bandit," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 6, p. 1912–1925, Jun. 2023. [Online]. Available: http://dx.doi.org/10.1109/TCAD.2022.3213611

[10] S.-Y. Lee, H. Riener, A. Mishchenko, R. K. Brayton, and G. De Micheli, "A Simulation-Guided Paradigm for Logic Synthesis and Verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 8, pp. 2573–2586, 2021.

[11] P. Wills and F. G. Meyer, "Metrics for Graph Comparison: A Practitioner's Guide," *Plos one*, vol. 15, no. 2, p. e0228728, 2020.

[12] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina, "Web graph similarity for anomaly detection," *Journal of Internet Services and Applications*, vol. 1, pp. 19–30, 2010.

[13] M. Berlingerio, D. Koutra, T. Eliassi-Rad, and C. Faloutsos, "NetSimile: A Scalable Approach to Size-Independent Network Similarity," *arXiv preprint arXiv:1209.2684*, 2012.

[14] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-Lehman Graph Kernels," *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[15] G. Ma, N. K. Ahmed, T. L. Willke, and P. S. Yu, "Deep Graph Similarity Learning: A Survey," *Data Mining and Knowledge Discovery*, vol. 35, pp. 688–725, 2021.

[16] D. Koutra, J. T. Vogelstein, and C. Faloutsos, "DELTACON: A Principled Massive-Graph Similarity Function," 2013. [Online]. Available: https://arxiv.org/abs/1304.4657

[17] D. Babić, D. J. Klein, I. Lukovits, S. Nikolić, and N. Trinajstić, "Resistance-distance matrix: A computational algorithm and its application," *International Journal of Quantum Chemistry*, vol. 90, no. 1, pp. 166–176, 2002.

[18] A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 3, pp. 353–362, 1983.

[19] Y. Li, M. Liu, M. Ren, A. Mishchenko, and C. Yu, "DAG-aware Synthesis Orchestration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[20] S.-Y. Lee, H. Riener, and G. D. Micheli, "Customizable On-the-Fly Design Space Exploration for Logic Optimization of Emerging Technologies," in *International Workshop on Logic Synthesis (IWLS)*, Lausanne, Switzerland, 2023.