The Algebra of Graph Structure: Isomorphism Testing and Automorphism Group Computation

Ms. Sirisha Peddinti
Assistant Professor, Department of Mathematics
Sree Dattha Institute of Engineering & Science
Hyderabad-501510, India
sirishachyuth@gmail.com

Dr. S. Venkata Achuta Rao
Professor and Dean (Academics)
Sree Dattha Institute of Engineering & Science
Hyderabad-501510, India
sreedatthaachyuth@gmail.com

***Abstract*—Graph Isomorphism (GI) remains a central problem in algorithmic graph theory, occupying the intermediate complexity class between P and NP-complete. This paper provides a rigorous analysis of algebraic and combinatorial invariants used to detect isomorphism. We examine the limitations of vertex invariants, detail the "Whitney Exception" for line graphs (K_3 vs $K_{1,3}$), and present a formal group-theoretic derivation of the automorphism groups for Complete Graphs (S_n) and the Petersen Graph (S_5). Furthermore, we explore advanced canonicalization techniques, including the Weisfeiler-Leman (WL) color refinement algorithm and search-tree methods derived from McKay's Nauty, which leverage individualization and partition refinement. Our survey synthesizes classical results with modern algorithmic advances, supported by comprehensive visualizations of orbits and matrix canonicalization.**

***Index Terms*—**Graph Isomorphism, Automorphism Group, Weisfeiler-Leman Algorithm, Canonicalization, Complexity Theory, Nauty.

1. Introduction

The Graph Isomorphism (GI) problem stands as one of the most provocative open questions in algorithmic complexity. It asks a deceptively simple question: given two finite graphs G and H, is there a bijection $\phi : V(G) \rightarrow V(H)$ such that adjacency is preserved? Unlike the satisfiability problem (SAT) or the traveling salesman problem, GI resists classification into the standard P versus NP-complete dichotomy. It occupies the elusive *NP-intermediate* class, a status it shares with integer factorization [1]–[3].

Historically, the problem has acted as a catalyst for algebraic graph theory [4], [5]. In 1932, Hassler Whitney linked edge isomorphisms to vertex isomorphisms, establishing the foundations of graph topology [6]. The 1968 introduction of the Weisfeiler-Leman (WL) algorithm provided a powerful heuristic based on iterative color refinement [7], yet it was quickly shown to be insufficient for regular graphs [8]. Complexity theorists long suspected GI was not NP-complete, primarily because such a result would collapse the polynomial hierarchy to its second level [9], [10]. This intuition was vindicated in 2016 when La´szlo´ Babai announced a quasipolynomial time algorithm, solving GI in $\exp((\log n)^{O(1)})$ time [11], [12], a result that fundamentally shifted the landscape of the field [13].

Beyond theory, GI underpins critical applications in computational sciences. In cheminformatics, chemical compounds are modeled as graphs; identifying isomers is equivalent to solving GI [14]. Similarly, electronic design automation relies on GI to verify that synthesized VLSI circuits match their specifications [15]. This paper provides a comprehensive analysis of the GI problem, moving from classical invariants to the group-theoretic machinery of modern solvers. We organize our discussion as follows:

1) **Definitions**: Rigorous formulation of labeled/unlabeled graphs.
2) **Theory**: Classical invariants and the WL hierarchy.
3) **Results**: Whitney's Theorem and the Line Graph exception.
4) **Symmetry**: Automorphism groups and the Orbit-Stabilizer theorem.
5) **Advanced**: Nauty, partition refinement, and complexity classes.

2. Fundamental Definitions and Concepts

A. Graph Theoretic Foundations

To establish a rigorous foundation for the study of isomorphism, we must first precisely define the structures we are analyzing. A **simple undirected graph** $G = (V, E)$ consists of a non-empty finite set of vertices V and a set of edges E, where each edge is a 2-element subset of V. That is, $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$. The term "undirected" implies that the pair $\{u, v\}$ is unordered; the connection is symmetric.

The **degree** of a vertex v, denoted $\deg(v)$, is the number of edges incident to it. This local property is one of the simplest invariants: in any isomorphism, a vertex of degree k must map to a vertex of degree k. The collection of all degrees in G, sorted in non-increasing order, forms the **degree sequence**, a global signature of the graph.

B. The Distinction: Labeled vs. Unlabeled Graphs

A critical and often misunderstood distinction in isomorphism theory is the difference between labeled and unlabeled graphs. This distinction drastically affects combinatorial counting and complexity.

Doi:10.48047/ijiee.2026.16.1.1

1) Labeled Graphs: In a labeled graph, the vertices are distinct and identifiable, typically labeled with integers $\{1, 2, \ldots, n\}$. Two labeled graphs are considered different if they have different edges sets, even if they look structurally identical. For example, on $V = \{1, 2, 3\}$, the graph with edge $\{1, 2\}$ is distinct from the graph with edge $\{2, 3\}$. There are $2^{\binom{n}{2}}$ possible labeled graphs on n vertices.

2) Unlabeled Graphs: An unlabeled graph represents a pure structural class. Vertices are indistinguishable points. An unlabeled graph is essentially an *isomorphism class* of labeled graphs. When we ask "how many graphs of size n exist?", we are usually counting unlabeled graphs (orbits of the labeled graphs under permutation). For $n = 3$, there are $2^3 = 8$ labeled graphs, but only 4 unlabeled graphs (Empty, 1-edge, 2-edges, Triangle).

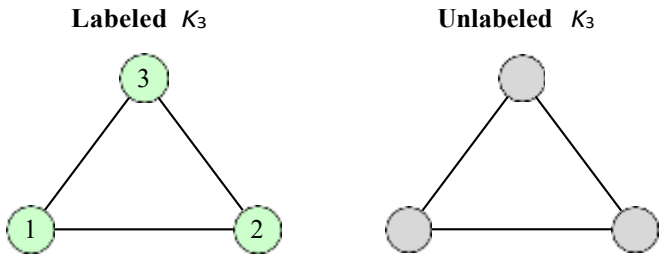

Fig. 1. Comparison between a labeled graph (green, distinguishable IDs) and an unlabeled graph (gray, indistinguishable structural units).

C. Paths, Cycles, and Connectivity

Connectivity properties are vital invariants preserved under isomorphism.

1) **Path**: A sequence of distinct vertices $v_0, v_1, \ldots, v_k$ such that $\{v_i, v_{i+1}\} \in E$ for all $0 \leq i < k$. The length is the number of edges, k.
2) **Cycle**: A path that is closed, meaning the start vertex equals the end vertex ($v_0 = v_k$), and $k \geq 3$.
3) **Connectedness**: A graph is connected if there is a path between every pair of vertices. If a graph is disconnected, its **connected components** are the maximal connected subgraphs. The set of component sizes is another powerful invariant.

3. Graph Isomorphism: Theory and Detection

Standard textbooks by West [16], Bondy & Murty [17], and Bolloba´s [18] provide the foundational graph theory framework used throughout this paper. Early attempts to solve GI focused on heuristics and invariants [19], while specialized algorithms for subgraph isomorphism were developed by Ullmann [20]. Distance matrix approaches were also explored [21].

A. Formal Definition and Complexity

Two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are **isomorphic** if there exists a bijection $f : V_G \to V_H$ that preserves adjacency and non-adjacency. Formally:

$$\{u, v\} \in E_G \iff \{f(u), f(v)\} \in E_H \quad \forall u, v \in V_G$$

C. Non-Isomorphism and Invariants

The search space for such a bijection is the set of all $n!$ permutations of the vertices. A brute-force check is computationally infeasible for even moderately sized graphs (e.g., $20! \approx 2.4 \times 10^{18}$). The challenge lies in pruning this search space using structural properties.

B. Visualizing Isomorphism

The difficulty of the problem is often visual. Graphs can be drawn in infinite ways. Isomorphism asks if two different drawings represent the same underlying topology. Figure 2 demonstrates this with a square and a diamond. The function f maps $1 \to a$, $2 \to b$, etc., effectively "morphing" one drawing into the other.

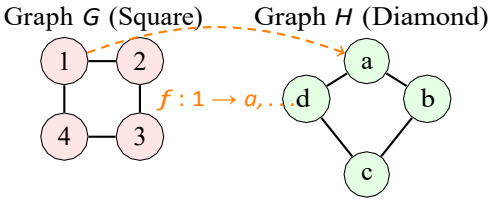

Fig. 2. Isomorphism between a Square and a Diamond. The mapping f preserves adjacency.

Proving two graphs are *not* isomorphic is often easier than proving they are. We compute **invariants**—properties that must be identical for isomorphic graphs. If $Inv(G) \neq Inv(H)$, then $G \ncong H$. Common invariants include:

1) Global counts: $|V|$, $|E|$, number of connected components.
2) Degree Sequence: The sorted list of vertex degrees.
3) Path/Cycle counts: The number of triangles, 4-cycles, etc.

However, these are necessary but not sufficient. Figure 3 shows two graphs with the same degree sequence $(2, 2, 2, 2, 2, 2)$ that are distinct: one is connected (C_6), the other is disconnected ($2C_3$).

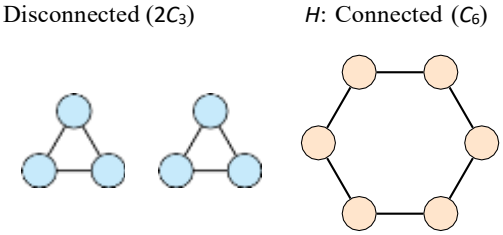

Fig. 3. Counterexample: Two non-isomorphic graphs with identical degree sequence $(2, 2, 2, 2, 2, 2)$. G ($2C_3$, cyan) is disconnected, while H (C_6, orange) is connected. WL refinement would assign identical initial colors to all vertices here.

D. State-of-the-Art Detection Algorithms

Modern solvers do not rely on brute force. They combine invariant checking with intelligentsearch.

Doi:10.48047/ijiee.2026.16.1.1

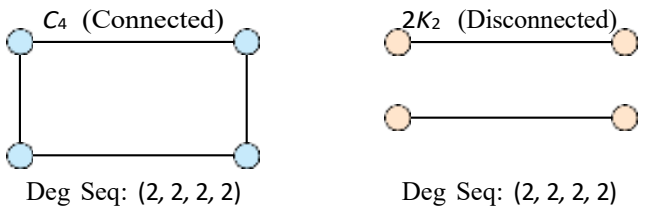

Fig. 4. Non-isomorphism via Degree Sequence. G_1 (Cycle C_5) is 2-regular. G_2 ("House" graph without base?) has degree sequence (3, 2, 2, 2, 1). They differ despite equal edges.

1) Weisfeiler-Lehman (WL) Test: The 1-dimensional Weisfeiler-Lehman (1-WL) test is a powerful heuristic based on iterative color refinement.

1) **Initialize**: Assign every vertex a color $c_0(v) = \deg(v)$.
2) **Refine**: At each step i, update the color of vertex v by aggregating the colors of its neighbors:

$$c_{i+1}(v) = \text{hash}\,(c_i(v), \text{sort}\,(\{c_i(u) \mid u \text{ adjacent to } v\}))$$

3) **Converge**: Repeat until colors stabilize. The final multiset of colors is the graph's canonical signature.

The 1-WL test successfully distinguishes almost all random graphs but fails on *regular graphs* where all vertices have the same degree and neighbor structure (e.g., distinguishing a 3×3 grid from a 3×3 torus requires higher-dimensional WL).

2) Nauty (No Automorphisms, Yes?): Brendan McKay's **nauty** is the industry standard for isomorphism testing. It computes a *canonical labeling*—a unique permutation of the vertices $C(G)$ such that $G \cong H \iff C(G) = C(H)$. It works by generating a search tree where nodes are partitions of vertices. It traverses this tree to find the "lexicographically smallest" adjacency matrix, pruning vast branches of the tree by discovering automorphisms (symmetries) that map one branch to another [22].

3) VF2 Algorithm: VF2 is typically used for subgraph isomorphism but applies to GI as well. It performs a depth-first search to build the matching one vertex at a time. Its power comes from **feasibility rules** (look-ahead): before mapping $u \to v$, it checks if the neighbors of u can consistently map to the neighbors of v. If the count of unmatched neighbors differs, it prunes the branch immediately [15].

4. Graph Invariants and Whitney's Theorem

A. Standard Invariants

Before moving to advanced theorems, we recap why simple invariants form the front line of isomorphism testing. A graph **invariant** is a function f such that $G \cong H \implies f(G) = f(H)$. If $f(G) \neq f(H)$, we immediately conclude $G \ncong H$. Efficient invariants include:

- **Vertex/Edge Count**: $|V|$ and $|E|$ must match.
- **Sorted Degree Sequence**: Unique distributions of degrees.
- **Spectral Spectrum**: The set of eigenvalues of the Adjacency matrix A. Cospectral graphs exist (graphs with the same spectrum but different structure), so this is not complete [5].

B. Whitney's Graph Isomorphism Theorem

In 1932, Hassler Whitney proved a remarkable result connecting vertex isomorphism to edge isomorphism via the concept of the Line Graph. The line graph $L(G)$ of a graph G is a graph where the vertices of $L(G)$ represent the edges of G. Two vertices in $L(G)$ are adjacent if their corresponding edges in G share a common endpoint. This transformation converts "edge relationships" in G into "vertex relationships" in $L(G)$.

Theorem 6.2.1 (Whitney's Theorem): Let G and H be connected graphs with $|V| > 4$. Then:

$$G \cong H \iff L(G) \cong L(H)$$

This theorem essentially states that the edge structure alone captures the entire graph topology, with one famous exception.

The Exception (K_3 vs $K_{1,3}$): The complete graph K_3 (triangle) and the complete bipartite graph $K_{1,3}$ (star/claw) are clearly not isomorphic. However, their line graphs are isomorphic:

$$L(K_3) \cong K_3 \quad \text{and} \quad L(K_{1,3}) \cong K_3$$

This collision occurs because the line graph transformation cannot distinguish between "3 edges forming a triangle" and "3 edges meeting at a single vertex" (a star). For $n > 4$, these structures become distinguishable.

C. Intuition of the Proof

How does one recover G from $L(G)$? The proof relies on identifying *cliques* (fully connected subgraphs).

1) A vertex v with degree k in G corresponds to a set of k edges all touching v.
2) In the line graph $L(G)$, these k edges form a clique of size k, since they all share vertex v.
3) Therefore, the vertices of G can be reconstructed by finding maximal cliques in $L(G)$.
4) The catch: Edges forming a triangle in G also form a size-3 clique in $L(G)$. For small graphs ($n \leq 4$), "star cliques" and "triangle cliques" can be confused. For larger graphs, the distinction is forced by the surrounding structure [16].

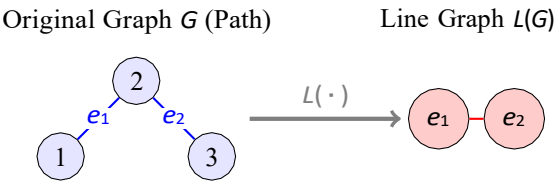

Fig. 5. Transformation of a Path graph into its Line Graph. Blue edges in G become Red vertices in $L(G)$.

Doi:10.48047/ijiee.2026.16.1.1

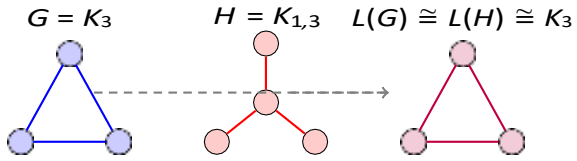

Fig. 6. The Whitney Exception: K_3 (Triangle) and $K_{1,3}$ (Claw) are non-isomorphic, yet they map to the exact same Line Graph (K_3).

5. Graph Automorphisms and Symmetry

A. Algebraic Definitions

An **automorphism** of a graph G is an isomorphism from G to itself. Formalizing this requires viewing the automorphism group $\mathrm{Aut}(G)$ as a subgroup of the symmetric group S_n. This group action preserves the edge set E. For $\pi \in S_n$:

$$\pi(E) = \{\{\pi(u), \pi(v)\} : \{u, v\} \in E\} = E$$

The study of these groups allows us to quantify symmetry using the Orbit-Stabilizer Theorem.

Theorem 5.1 (Orbit-Stabilizer): Let $G = (V, E)$ be a graph and $v \in V$. The size of the automorphism group is given by:

$$|\mathrm{Aut}(G)| = |\mathrm{Orb}(v)| \cdot |\mathrm{Stab}(v)|$$

where $\mathrm{Orb}(v) = \{\pi(v) : \pi \in \mathrm{Aut}(G)\}$ is the orbit of vertex v.

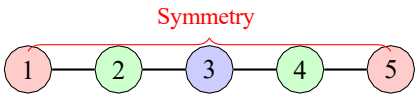

Fig. 7. Visualizing Orbits on Graph P_5. Vertices of the same color belong to the same orbit. The central vertex 3 is fixed (Orb size 1); leaves 1 and 5 can be swapped (Orb size 2).

The study of these groups allows us to quantify symmetry using the Orbit-Stabilizer Theorem.

B. Rigorous Proofs for Special Classes

1) Complete Graphs (K_n): **Theorem 5.2:** The automorphism group of the complete graph K_n is isomorphic to the symmetric group S_n.

$$\mathrm{Aut}(K_n) \cong S_n$$

Proof. Let $V = \{1, \ldots, n\}$. The edge set of K_n is defined as $E(K_n) = \{\{u, v\} : u, v \in V, u \neq v\}$. This includes *all* possible pairs of distinct vertices.

Step 1: Injections ($\mathrm{Aut}(K_n) \subseteq S_n$). By definition, any automorphism ϕ is a bijection from V to V (a permutation). Thus, $\mathrm{Aut}(K_n)$ is a subgroup of S_n.

Step 2: Surjections ($S_n \subseteq \mathrm{Aut}(K_n)$). We must show that *every* permutation $\sigma \in S_n$ preserves the adjacency of K_n. Let $\sigma \in S_n$ act on V. Let $\{u, v\}$ be any edge in $E(K_n)$. By definition of K_n, u and v are distinct vertices. Since σ is a bijection, $\sigma(u) \neq \sigma(v)$. Thus, $\{\sigma(u), \sigma(v)\}$ is a pair of distinct vertices. Since K_n contains *all* pairs of distinct vertices as edges, $\{\sigma(u), \sigma(v)\} \in E(K_n)$.

Consequently, σ maps edges to edges. The complement graph $\bar{K}_n$ has no edges, so non-edges are trivially preserved. Thus, every $\sigma \in S_n$ is an automorphism.

Conclusion: Since $\mathrm{Aut}(K_n) \subseteq S_n$ and $S_n \subseteq \mathrm{Aut}(K_n)$, we have $\mathrm{Aut}(K_n) \cong S_n$. This implies $|\mathrm{Aut}(K_n)| = n!$. □

2) The Petersen Graph (P): **Theorem 5.3:** The automorphism group of the Petersen graph P is isomorphic to S_5.

$$\mathrm{Aut}(P) \cong S_5$$

Proof. We use the Kneser Graph construction. Let $X = \{1, 2, 3, 4, 5\}$. Define the vertices of P as the $\binom{5}{2} = 10$ subsets of X of size 2. Two vertices (subsets) $A, B \subset X$ are adjacent if and only if they are disjoint ($A \cap B = \emptyset$).

Step 1: Constructing the Action. Let $\sigma \in S_5$ be a permutation of the base set X. This induces a permutation σ^* on the vertices of P (the pairs) naturally:

$$\sigma^*(\{i, j\}) = \{\sigma(i), \sigma(j)\}$$

Since σ is a bijection, disjoint sets map to disjoint sets:

$$A \cap B = \emptyset \iff \sigma(A) \cap \sigma(B) = \emptyset$$

Thus, adjacency is preserved. This defines a group homomorphism $\Psi : S_5 \to \mathrm{Aut}(P)$.

Step 2: Injectivity. Suppose σ induces the identity automorphism on P. Then $\sigma(\{i, j\}) = \{i, j\}$ for all pairs. If σ fixes every pair, it must fix every element $k \in X$ (as k is the unique intersection of all pairs containing it). Thus $\ker(\Psi) = \{id\}$, so S_5 embeds into $\mathrm{Aut}(P)$. This gives $|\mathrm{Aut}(P)| \geq 120$.

Step 3: Surjectivity (Via Orbit-Stabilizer). P is vertex-transitive, so $|\mathrm{Orb}(v)| = 10$. Fix a vertex $v = \{1, 2\}$. Its neighbors are pairs disjoint from it: $\{3, 4\}, \{3, 5\}, \{4, 5\}$. Any stabilizer must map v to itself, and permute its neighbors. A detailed analysis shows that the stabilizer acts as S_3 on the neighbors and S_2 on the internal structure of v, bounded by $|\mathrm{Stab}(v)| \leq 12$. Using Orbit-Stabilizer: $|\mathrm{Aut}(P)| = 10 \times 12 = 120$. Since we found a subgroup S_5 of size 120, and the total group size is 120, $\mathrm{Aut}(P) \cong S_5$. □

C. Summary of Automorphism Groups

Table I summarizes the groups for common families [5].

TABLE I
Automorphism Groups of Common Graphs

Graph Class	$\lvert\mathrm{Aut}(G)\rvert$	Group Structure
Complete K_n	$n!$	S_n
Cycle C_n ($n \neq 4$)	$2n$	D_n (Dihedral)
Path P_n ($n > 1$)	2	S_2
Star $K_{1,n}$	$n!$	S_n (fix center)
Petersen Graph	120	S_5
Hypercube Q_d	$2^d d!$	$S_2 \wr S_d$ (Wreath product)

D. Visualizing Symmetry (C_6)

Figure 8 visualizes the symmetries of C_6.

Doi:10.48047/ijiee.2026.16.1.1

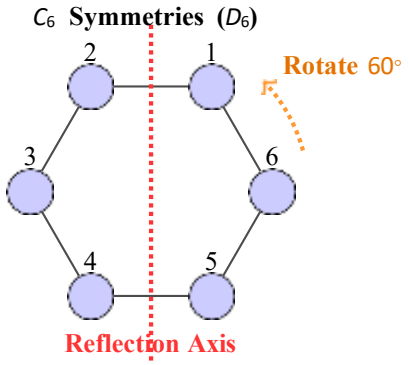

Fig. 8. Visualizing the geometric symmetries of C_6. The group D_6 preserves the hexagonal structure.

6. Advanced Topics: Canonicalization and Complexity

A. Graph Canonicalization via Partition Refinement

While isomorphism asks "is $G \cong H$?", canonicalization asks for a signature $C(G)$ such that $G \cong H \iff C(G) = C(H)$. The most successful strategy is **Partition Refinement** (Figures 9 and 10).

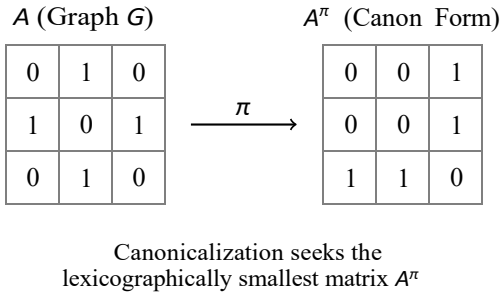

Fig. 9. Canonicalization via Adjacency Matrix permutation. The algorithm reorders rows/columns to achieve a minimal binary string signature.

B. The Weisfeiler-Lehman (WL) Algorithm

The 1-WL algorithm (Color Refinement) is the basis for all modern solvers.

While efficient, 1-WL cannot distinguish certain regular graphs (e.g., C_6 vs $2C_3$ if handled naively without component counts, or more famously the Shrikhande graph vs Rook's graph).

C. Nauty and Practical Solvers

For graphs where WL fails to distinguish vertices (regular graphs), we must break symmetry by **Individualization**.

1) Search Tree and Individualization: McKay's **Nauty** (No Automorphisms, Yes?) [22], [31] builds a search tree using group-theoretic insights [33]. The approach relies on efficient handling of permutation groups [34].

1) **Root**: The initial equitable partition π.

Algorithm 1:1-Dimensional Weisfeiler-Lehman Refinement

1) **Input**: Graph $G = (V, E)$.
2) **Initialize**: For all $v \in V$, $c_0(v) \leftarrow \deg(v)$.
3) **Repeat** until stable (partition does not change):
   a) For each $v \in V$: gather neighbor colors
   $$M_i(v) \leftarrow \{c_i(u) \mid \{u, v\} \in E\}$$
   b) Sort $M_i(v)$.
   c) Update color: $c_{i+1}(v) \leftarrow \text{hash}(c_i(v), M_i(v))$.
4) **Output**: The multiset of final colors $\{c_{\text{final}}(v) \mid v \in V\}$.

2) **Branching (Individualization)**: Pick a target cell $V_i \in \pi$. For each vertex $v \in V_i$, create a child node where v is uniquely colored (singleton cell $\{v\}$).
3) **Refinement**: Run WL refinement on this new partition to propagate constraints.
4) **Pruning**: As the search progresses, automorphisms are discovered. If a leaf node's canonical form matches a previously found form, the group size is updated, and the current branch can be pruned. This massive pruning is what makes Nauty effective on highly symmetric graphs (S_n, Q_d).

2) Bliss and Traces: **Bliss** [25] and **Traces** introduce modern heuristics for pivot selection in the search tree. Unlike Nauty's "first cell" heuristic, these algorithms prioritize cells that maximize splitting power (entropy) to keep the search tree depth minimal. Traces, in particular, uses a "best-first" strategy that often outperforms Nauty on difficult benchmarks like projective planes.

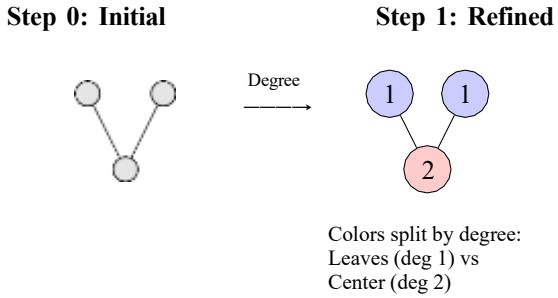

Fig. 10. Visualizing partition refinement on P_3.

D. Computational Complexity Landscape

Graph Isomorphism (GI) is in NP but likely not NP-complete.

- **Babai's Algorithm (2016)**: Introducing the "Split-or-Johnson" routine, Babai proved GI is solvable in Quasipolynomial time $O(\exp((\log n)^c))$. This places GI strictly below exponential time, making it unlikely to be NP-complete [11].

Doi:10.48047/ijiee.2026.16.1.1

- **GI-Complete**: Problems equivalent to GI, such as isomorphism of semigroups, form a unique complexity class [30].
- **Strongly Regular Graphs**: Historically difficult cases, solvable in time $O(\exp(n^{1/5}))$ [32].

TABLE II
COMPLEXITY OF GRAPH ISOMORPHISM FOR SPECIAL CLASSES

Graph Class	Complexity	Algorithm
Trees	$O(n)$	Center encoding / AHU
Planar Graphs	$O(n)$	Hopcroft-Tarjan (1974) [26]
Interval Graphs	$O(n)$	Lueker-Booth [35]
Bounded Degree d	$O(n^{O(d)})$	Luks (1982) Group Theory [23]
General Graphs	$O(\exp((\log n)^6))$	Babai (2016) [11]

7. CONCLUSION

The Graph Isomorphism problem stands as a singular anomaly in complexity theory—a problem in NP that is likely neither P nor NP-complete. In this survey, we have traversed the landscape of GI from basic combinatorial invariants to sophisticated group-theoretic solvers. Our analysis demonstrates that while local invariants such as degree sequences are computationally cheap, they fail to capture global symmetry, necessitating the iterative color refinement of the Weisfeiler-Lehman hierarchy.

We explored the algebraic underpinnings of symmetry through the lens of Automorphism Groups, showing how the Orbit-Stabilizer theorem provides a rigorous framework for counting symmetries in structures like the Petersen graph. The discussion of the ”Whitney Exception” highlights the subtleties of topological mapping, reminding us that even intuitive transformations like line graphs have strictly defined edge-cases.

Ultimately, the resolution of GI lies at the intersection of deep theory and engineering. While Babai’s 2016 breakthrough places the problem in Quasipolynomial time, theoretically separating it from NP-complete problems, practical solving relies on the individualization-refinement paradigm pioneered by Nauty and Bliss. As these solvers continue to prune search trees on massive graphs, the gap between the theoretical upper bound and practical tractability continues to narrow, suggesting that a polynomial-time algorithm may essentially exist in practice, if not yet in proof.

REFERENCES

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman, 1979.
[2] P. Schweitzer, “Graph isomorphism and the graph isomorphism problem,” *Communications of the ACM*, vol. 60, no. 9, pp. 101–101, 2017.
[3] A. Lubiw, “Some np-complete problems similar to graph isomorphism,” *SIAM Journal on Computing*, vol. 10, no. 1, pp. 11–21, 1981.
[4] N. Biggs, *Algebraic graph theory*. Cambridge university press, 1993.
[5] C. Godsil and G. F. Royle, *Algebraic Graph Theory*. Springer Science & Business Media, 2001, vol. 207.
[6] H. Whitney, “Congruent graphs and the connectivity of graphs,” *American Journal of Mathematics*, vol. 54, no. 1, pp. 150–168, 1932.
[7] B. Weisfeiler and A. Leman, “A reduction of a graph to a canonical form and an algebra arising during this reduction,” *Nauchno-Technicheskaya Informatsiya*, vol. 2, no. 9, pp. 12–16, 1968.
[8] R. Mathon, “A note on the graph isomorphism counting problem,” *Information Processing Letters*, vol. 8, no. 3, pp. 131–132, 1979.
[9] J. Ko¨bler, U. Scho¨ning, and J. Tora´n, *The graph isomorphism problem: its structural complexity*. Birkha¨user, 2012.
[10] V. Arvind and J. Tora´n, “Solvable group isomorphism is (almost) in np ∩ conp,” *Proceedings of the 20th Annual IEEE Conference on Computational Complexity (CCC’05)*, pp. 1–1, 2005.
[11] L. Babai, “Graph isomorphism in quasipolynomial time,” in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 2016, pp. 684–697.
[12] ——, “Groups, graphs, algorithms: The graph isomorphism problem,” *Proceedings of the International Congress of Mathematicians (ICM)*, vol. 3, pp. 3319–3336, 2018.
[13] M. Grohe, *Descriptive complexity, canonisation, and definable graph structure theory*. Cambridge University Press, 2017, vol. 47.
[14] R. C. Read, “The number of alkanes having n carbons,” *Discrete Mathematics*, 1977.
[15] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, “A (sub) graph isomorphism algorithm for matching large graphs,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004.
[16] D. B. West, *Introduction to Graph Theory*, 2nd ed. Prentice Hall, 2001.
[17] J. A. Bondy and U. Murty, *Graph Theory*. Springer, 2008.
[18] B. Bolloba´s, *Modern Graph Theory*. Springer Science & Business Media, 2013.
[19] D. G. Corneil and C. C. Gotlieb, “An efficient algorithm for graph isomorphism,” *Journal of the ACM (JACM)*, vol. 17, no. 1, pp. 51–64, 1970.
[20] J. R. Ullmann, “An algorithm for subgraph isomorphism,” *Journal of the ACM (JACM)*, vol. 23, no. 1, pp. 31–42, 1976.
[21] D. C. Schmidt and L. E. Druffel, “A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices,” *Journal of the ACM (JACM)*, vol. 23, no. 3, pp. 433–445, 1976.
[22] B. D. McKay and A. Piperno, “Practical graph isomorphism, ii,” *Journal of Symbolic Computation*, vol. 60, pp. 94–112, 2014.
[23] E. M. Luks, “Isomorphism of graphs of bounded valence can be tested in polynomial time,” *Journal of computer and system sciences*, vol. 25, no. 1, pp. 42–65, 1982.
[24] R. Diestel, *Graph Theory*, 5th ed. Heidelberg: Springer-Verlag, 2017.
[25] T. Junttila and P. Kaski, “Engineering an efficient canonical labeling tool for large and sparse graphs,” *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pp. 135–149, 2007.
[26] J. Hopcroft and R. Tarjan, “Efficient planarity testing,” *Journal of the ACM (JACM)*, vol. 21, no. 4, pp. 549–568, 1974.
[27] J. E. Hopcroft and R. E. Tarjan, “Isomorphism of planar graphs,” *Complexity of Computer Computations*, pp. 131–152, 1972.
[28] I. N. Ponomarenko, “The isomorphism problem for classes of graphs closed under contraction,” in *Proceedings of the Soviet-French Symposium fragments of computer science*, 1991.
[29] P. J. Cameron, “Permutation groups and transformation semigroups,” *London Mathematical Society Lecture Note Series*, pp. 67–67, 1999.
[30] V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich, “Graph isomorphism problem,” *Journal of Soviet Mathematics*, vol. 29, no. 4, pp. 1426–1481, 1985.
[31] B. D. McKay, “Practical graph isomorphism,” 1981.
[32] D. A. Spielman, “Faster isomorphism testing of strongly regular graphs,” in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 576–584.
[33] C. M. Hoffmann, “Group-theoretic algorithms and graph isomorphism,” *Lecture Notes in Computer Science*, vol. 136, 1982.
[34] M. Furst, J. Hopcroft, and E. Luks, “Polynomial-time algorithms for permutation groups,” *Proceedings of the 21st Annual Symposium on Foundations of Computer Science (SFCS 1980)*, pp. 36–41, 1980.
[35] K. S. Booth and G. S. Lueker, “Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms,” *Journal of Computer and System Sciences*, vol. 13, no. 3, pp. 335–379, 1976.

Doi:10.48047/ijiee.2026.16.1.1