Task 2: Family Clusters in MetaFam

Community Detection, Sub-Community Analysis, and Relatedness Metrics

"We are only as strong as we are united,
as weak as we are divided." — Albus Dumbledore

Contents

1 Introduction and Setup

1.1 Objective

Community detection identifies groups of densely connected nodes within a graph. In a family knowledge graph, these communities should correspond to actual family units. The goals of this task were to:

1. Implement and compare multiple community detection algorithms.
2. Assess community quality using objective metrics.
3. Analyze whether detected communities correspond to real family structures.
4. Propose a relatedness metric that goes beyond counting hops.

★ Bonus experiments

Beyond the required two algorithms and analysis questions, I conducted several additional investigations:

- **Third algorithm (Leiden)** — a modern improvement over Louvain with guaranteed well-connected communities
- **Sub-community hypothesis testing** — systematically testing whether intra-family sub-communities represent nuclear families, generational clusters, or branch lines
- **Nuclear family coherence metric** — a novel evaluation measuring whether algorithms keep nuclear families intact
- **Five relatedness metrics compared** — Coefficient of Relatedness, LCA-based, SimRank, Jaccard Similarity, and Weighted Distance, with detailed failure analysis for each
- **Global vs. local modularity analysis** — revealing hierarchical community structure

1.2 Graph Construction

I constructed three graph views from the training data, each serving a different analytical purpose:

Table 1: Graph Views Used for Community Detection

Graph	Purpose	Nodes	Edges
`G_undirected`	Community detection	1,316	7,480
`G` (directed)	Relationship analysis	1,316	13,821
`time_graph` (parent→child)	Generation computation	1,316	1,642

1.3 Preprocessing: Generations and Gender

Generation assignment was computed via topological sort on the parent→child directed graph. Nodes with no parents (in-degree 0) were assigned generation 0, and each child was assigned one generation beyond its highest-generation parent.

Table 2: Generation Distribution

Generation	People
Gen 0 (Founders)	495
Gen 1	215
Gen 2	192
Gen 3	206
Gen 4	146
Gen 5	55
Gen 6	7
Total	**1,316**

Gender was inferred from relation names: relations like `fatherOf`, `brotherOf`, `sonOf` indicate male heads, while `motherOf`, `sisterOf`, `daughterOf` indicate female heads. This successfully classified all 1,316 people (646 male, 670 female).

1.4 Ground Truth

Connected components of the undirected graph provide ground truth family labels. The graph contains **50 families** with remarkably uniform sizes (minimum 26, maximum 27 members). This uniformity suggests a synthetically generated dataset with controlled family sizes.

2 Community Detection Algorithms

2.1 Algorithm 1: Louvain Method

The Louvain method is a greedy modularity-optimization algorithm that iteratively merges nodes into communities to maximize the modularity score Q.

Key Finding

Louvain achieves perfect recovery:

- Communities detected: **50** (exactly matching ground truth)
- Modularity: **0.9794**
- Adjusted Rand Index: **1.0000**
- Normalized Mutual Information: **1.0000**

Every detected community corresponds to exactly one true family, with no splits, merges, or misassignments.

2.2 Algorithm 2: Label Propagation

Label Propagation is a fast, semi-supervised algorithm where each node adopts the label most common among its neighbors. Since it is stochastic, I ran it 5 times and selected the partition with the best modularity.

Table 3: Label Propagation Results

Metric	Value
Communities detected	64
Modularity	0.9652
Adjusted Rand Index	0.9576
Normalized Mutual Information	0.9844

Label Propagation over-segments the graph: it detects 64 communities instead of 50. Analysis of community–family overlap revealed **38 perfect matches** and **26 splits** (where a single family was fragmented into multiple communities), but **zero merges** — no community ever mixed members from different families.

Insight

Label Propagation's errors are exclusively **splits, never merges**. This means it has perfect *purity* (every community is a subset of one true family) but imperfect *coverage* (averaging 78.1%). The algorithm is conservative — it never incorrectly groups unrelated people together, but sometimes fails to recognize that distant relatives within the same family belong together.

2.3 Algorithm 3: Leiden

★ *Added as a third algorithm, helps to verify Leiden's splitting too.*

I discovered the Leiden algorithm through Louvain's Wikipedia page. Leiden is a modern refinement that guarantees all detected communities are well-connected, addressing a known theoretical weakness of Louvain.

Table 4: Leiden Results

Metric	Value
Communities detected	50
Modularity	0.9794
Adjusted Rand Index	1.0000
Normalized Mutual Information	1.0000

Leiden, like Louvain, achieves perfect recovery. On this particular graph, the two modularity-based methods produce identical results, while the propagation-based method lags behind.

2.4 Algorithm Comparison

Table 5: Three-Algorithm Comparison

Algorithm	Communities	Modularity	ARI	NMI	Nuclear Coh.
Louvain	50	0.9794	1.0000	1.0000	1.0000
Label Propagation	64	0.9652	0.9576	0.9844	0.9618
Leiden	50	0.9794	1.0000	1.0000	1.0000
Ground Truth	50	0.9794	1.0000	1.0000	1.0000

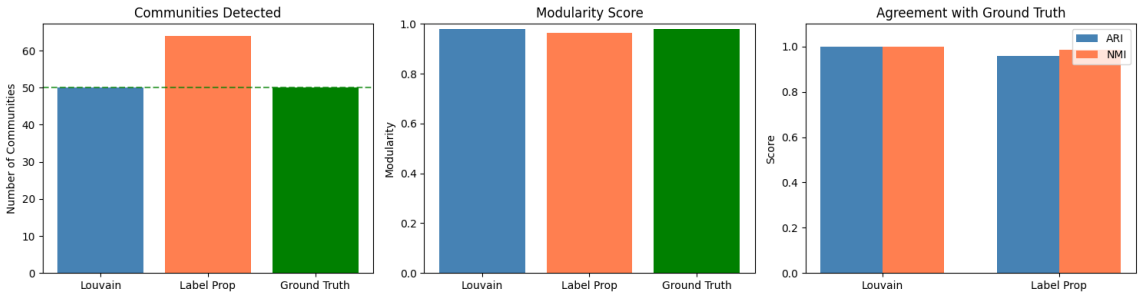
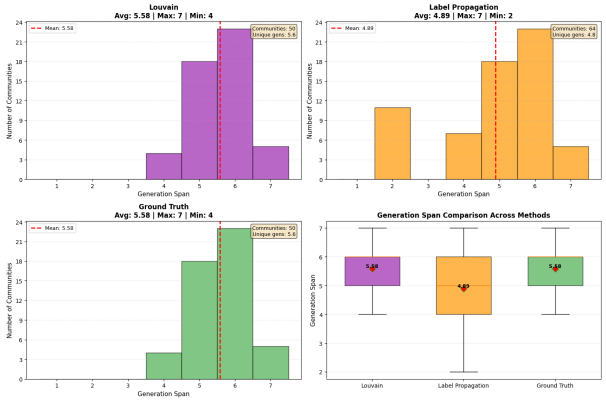


Figure 1: Community detection algorithm comparison. Louvain and Leiden perfectly recover ground truth families. Label Propagation over-segments into 64 communities with slightly lower modularity and agreement scores.



Insight

The near-perfect performance of all algorithms is not surprising given the graph structure: families are **disconnected components**. Every node within a family is connected to every other family member (directly or indirectly), but zero edges exist between families. This makes community detection essentially a connected-component problem. The more interesting question is what structure exists *within* families.

3 Analysis: Community Structure

3.1 Q1: Do Communities Correspond to Actual Family Units?

I implemented a community–family overlap analysis that classifies each detected community as a **perfect match**, **split** (subset of a family), **merge** (contains multiple families), or **partial match**.

Table 6: Community–Family Match Analysis

Algorithm	Perfect	Splits	Merges	Partial
Louvain	50	0	0	0
Label Propagation	38	26	0	0
Leiden	50	0	0	0

Answer: Yes. Louvain and Leiden produce a 1-to-1 mapping between communities and families. Label Propagation fragments 12 families into smaller pieces but never mixes families.

3.1.1 Nuclear Family Coherence

★ *Some more experiments.*

I identified all 445 nuclear families (couple + their shared children, averaging 1.84 children per couple) and checked whether each algorithm keeps nuclear families intact within the same community.

Table 7: Nuclear Family Coherence

Algorithm	Intact	Split	Rate
Louvain	445	0	100.0%
Label Propagation	428	17	96.2%
Leiden	445	0	100.0%

Label Propagation splits 17 nuclear families (3.8% failure rate), which directly explains its lower ARI score.

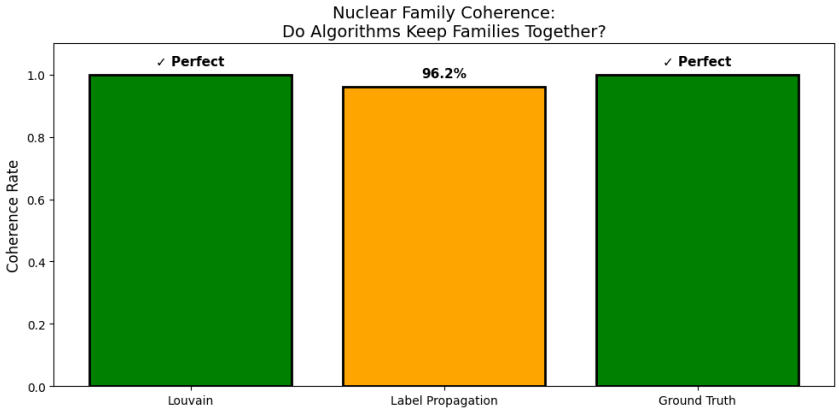


Figure 2: Nuclear family coherence across algorithms. Louvain and Leiden perfectly preserve all 445 nuclear families. Label Propagation fragments 17 families.

3.2 Q2: How Many Generations per Community?

Table 8: Generation Span per Community

Method	Communities	Avg Span	Min Span	Max Span
Louvain	50	5.58	4	7
Label Propagation	64	4.89	2	7
Ground Truth	50	5.58	4	7

Louvain perfectly preserves generation structure (0.0% deviation from ground truth). Label Propagation's over-segmentation reduces the average span by 12.4%, splitting off some generational extremes.

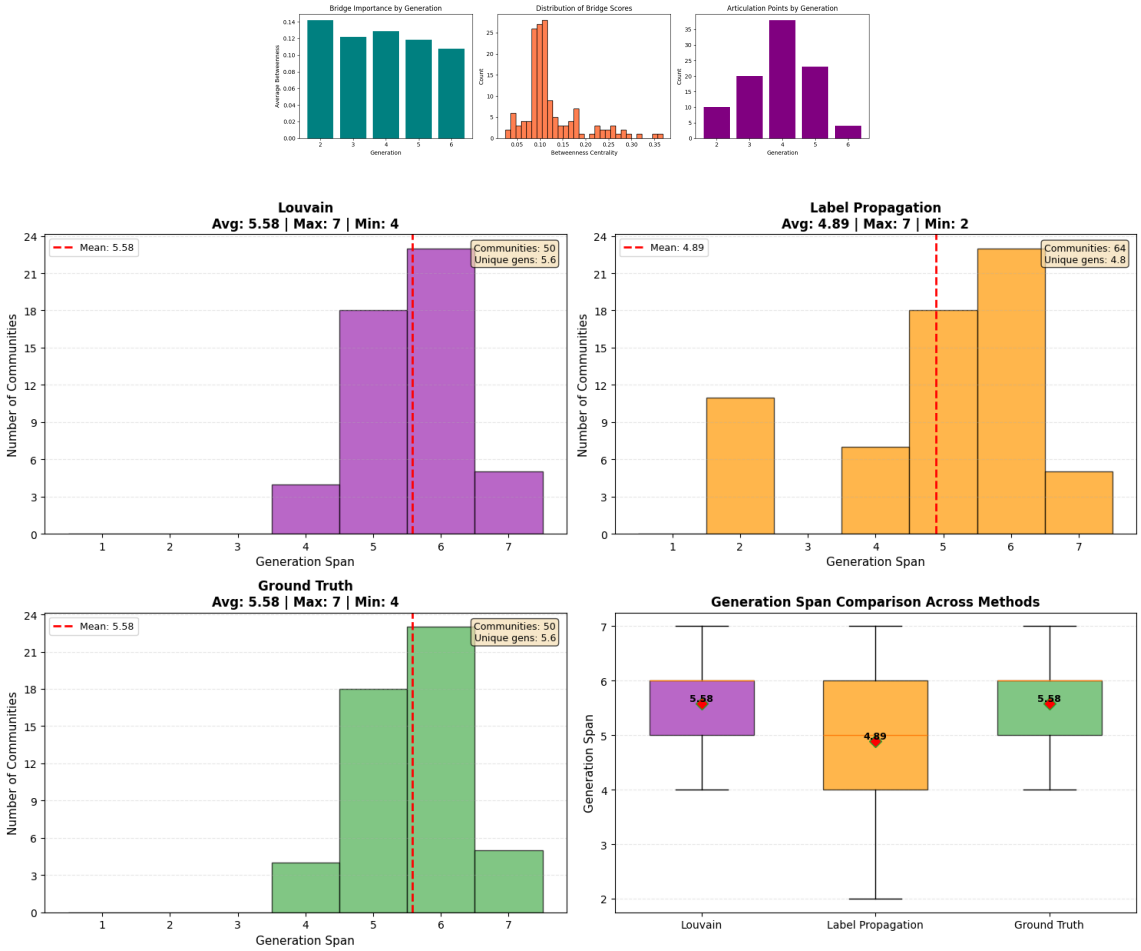


Figure 3: Generation span per community. Louvain exactly matches ground truth. Label Propagation produces some communities spanning only 2 generations due to family fragmentation.

3.3 Q3: Are There Bridge Individuals?

Since families are disconnected components, no *inter-family* bridges exist. I analyzed *intra-family* bridges — individuals whose removal would disconnect parts of their own family.

3.3.1 Articulation Points

Key Finding

95 articulation points were found across **45 of 50 families**. These are individuals whose removal would split their family graph into disconnected pieces — they are structurally critical connectors.

3.3.2 Betweenness Centrality

I computed betweenness centrality within each family subgraph to identify the top bridge candidates:

Table 9: Top 5 Bridge Candidates (by Within-Family Betweenness)

Node	Betweenness	Generation	Gender
lea1165	0.3685	4	F
valentin638	0.3497	3	M
gabriel241	0.3142	4	M
simon172	0.2899	5	M
nora536	0.2831	4	F

3.3.3 Bridge Patterns by Generation

Table 10: Bridge Importance by Generation

Generation	Count (Top 3/Family)	Avg Betweenness
Gen 2	8	0.1421
Gen 3	60	0.1219
Gen 4	55	0.1287
Gen 5	24	0.1184
Gen 6	3	0.1077

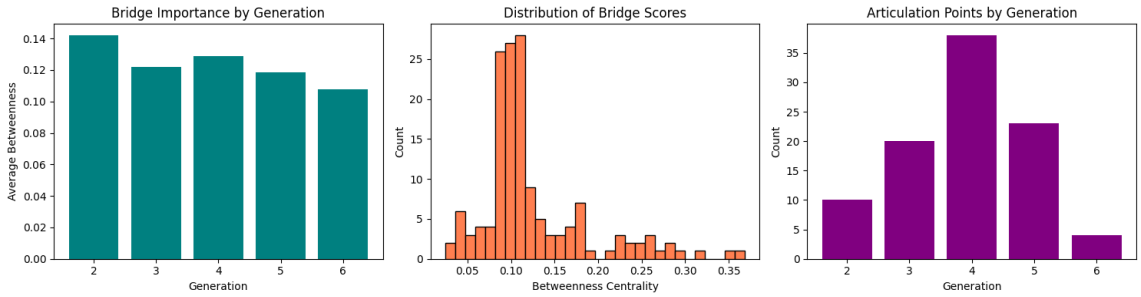


Figure 4: Bridge individual analysis. Left: Average betweenness by generation — Generation 2 individuals have the highest bridge importance. Middle: Distribution of betweenness scores. Right: Articulation points by generation.

Insight

Generation 2 has the highest average bridge importance, despite having the fewest bridge candidates. These middle-generation individuals connect the founder generations (Gen 0–1) to the descendant generations (Gen 3–6). They act as "generational bottlenecks" — the relatively few people through whom all ancestry information must flow.

4 Sub-Community Structure Within Families

★ *Trying to go beyond community detection to investigate internal family structure.*

4.1 Motivation

Given that community detection trivially recovers the 50 families, the more interesting question is: *what structure exists within families?* I ran Louvain on each family subgraph independently.

Table 11: Sub-Community Detection Statistics

Metric	Value
Families analyzed	50
Families with 1 sub-community	0
Families with 2+ sub-communities	50
Max sub-communities	4
Average sub-communities	2.80
Average within-family modularity	0.1819

Every family has internal structure that Louvain detects (2–4 sub-communities), but the low within-family modularity (0.18) suggests this structure is weak.

4.2 What Do Sub-Communities Represent?

I systematically tested three hypotheses about what the 2–4 sub-communities within each family represent:

4.2.1 Hypothesis 1: Nuclear Families

Sub-communities spanning only 1–2 generations would indicate nuclear family units (parents + their children).

Result: REJECTED. 100% of sub-communities span 3 or more generations. Zero are nuclear-like.

4.2.2 Hypothesis 2: Generational Clusters

Sub-communities containing predominantly one generation would indicate generational grouping.

Result: REJECTED. Average unique generations per sub-community: 4.08. No sub-community contains a single generation.

4.2.3 Hypothesis 3: Branch Lines

Sub-communities spanning all generations but containing only a subset of each generation would indicate vertical lineage slices — descendants of different founding couples.

Result: SUPPORTED. Multi-generation sub-communities average 10.5 people across 5.0 generations, yielding **2.1 people per generation** — consistent with a vertical branch containing one parent–child pair per generation.

Key Finding

Sub-communities are branch lines. When Louvain detects 2–4 sub-communities within a family, they represent vertical lineage slices — descendants of different founding couples. This makes genealogical sense: families naturally split into branches based on which grandparent couple you descend from. Louvain detects these because connections within a branch (siblings, parent–child) are denser than cross-branch connections (cousins from different branches).

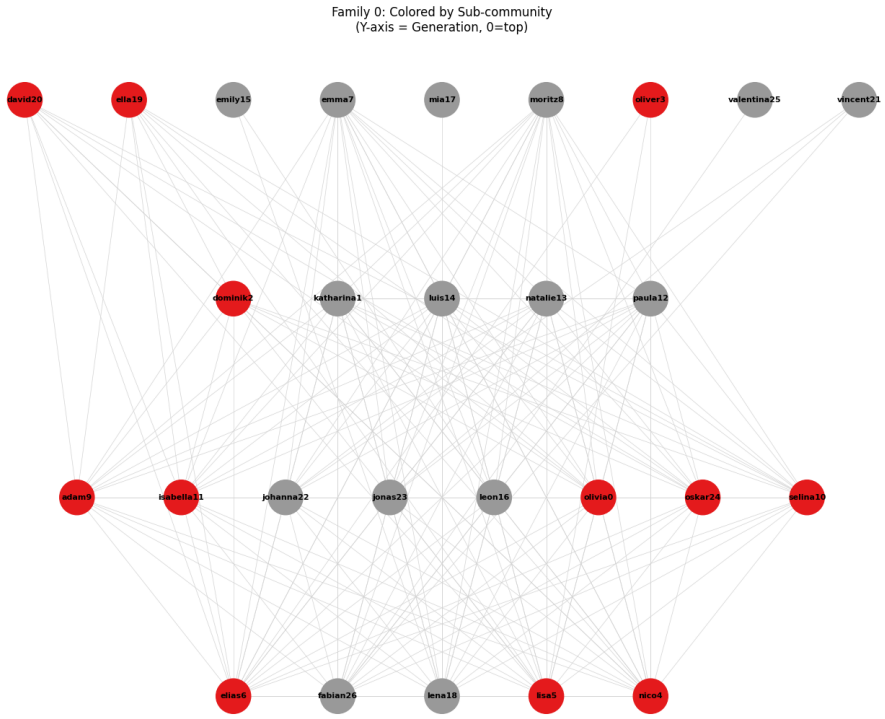


Figure 5: Family 0 (olivia0's family) colored by sub-community. Y-axis represents generation (Gen 0 at top). The two sub-communities form vertical slices through the family tree, each spanning all generations but containing different lineage branches.

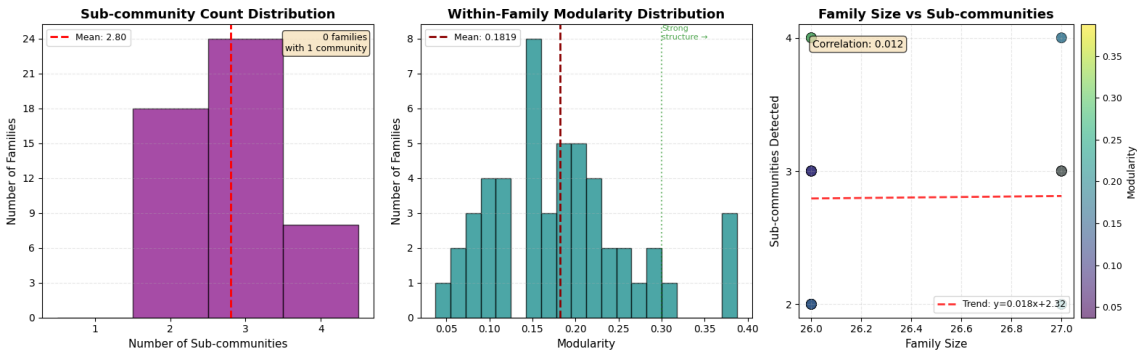


Figure 6: Sub-community statistics across all 50 families. Left: distribution of sub-community counts. Center: within-family modularity (mostly below 0.3). Right: family size vs. sub-communities — near-zero correlation ($r = 0.012$), indicating that the number of branches is independent of family size.

4.3 Global vs. Local Modularity

Table 12: Hierarchical Modularity Structure

Level	Description	Modularity
Global	Between families (50 communities)	0.9794
Local	Within a single family (2 sub-communities)	0.0726

Insight

The graph exhibits **hierarchical community structure**: families separate with near-perfect modularity ($Q = 0.98$), but within families, community structure is extremely weak ($Q = 0.07$). The graph represents **distinct, internally-cohesive lineages** rather than a single interconnected society with internal divisions. Families are tight-knit units without clear internal factions.

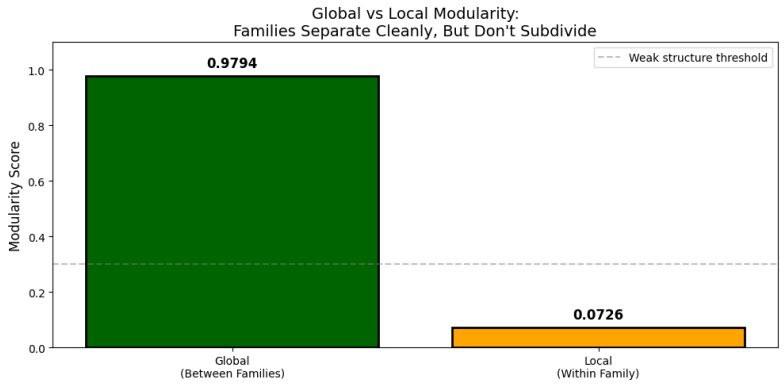


Figure 7: Global vs. local modularity. Families separate cleanly ($Q = 0.98$) but don't subdivide ($Q = 0.07$), falling well below the 0.3 threshold for meaningful community structure.

5 Relatedness Metric: Beyond Counting Hops

5.1 The Challenge

The task asks us to propose a method to rank how related two people are, going beyond simple hop counting. I explored five different approaches, learning from the failures of each to converge on the best solution.

5.2 Attempt 1: LCA-Based Metric (Failed)

My first attempt combined hop distance, shared ancestor count, generation proximity, common descendants, and lowest common ancestor (LCA) distance into a weighted score.

Mistake & Correction

The LCA-based metric produced an **incorrect ranking**: parents scored *lower* than cousins (0.29 vs. 0.40). The problem was twofold:

1. Shared ancestor count biased the metric toward same-generation relationships (siblings share 6 ancestors; a parent–child pair shares only 2).
2. The weighted combination obscured the fundamental genealogical signal.

A relatedness metric that ranks your mother below your cousin is clearly wrong.

5.3 Attempt 2: Corrected LCA-Based (Partial Fix)

I redesigned the metric around the coefficient of relatedness concept: score $= 0.5^{\text{LCA distance}}$ with bonuses for direct lineage (+0.15) and same generation (+0.10).

This corrected the parent–child ranking (0.65) but introduced a new problem: **siblings (0.35) scored lower than grandparents (0.40)** because the same-generation bonus was insufficient to compensate for siblings having LCA distance 2 (through each parent) while grandparents have a direct path.

5.4 Attempt 3: SimRank (Structural Failure)

SimRank measures structural similarity: "two nodes are similar if their neighbors are similar."

Mistake & Correction

SimRank assigned **zero relatedness to parent–child pairs**. The explanation is structural: a parent's neighbors (their own parents, spouse, other children) are entirely disjoint from their child's neighbors (both parents, siblings). These neighborhoods have **zero Jaccard overlap**, so SimRank gives zero similarity despite maximum genetic relatedness. In contrast, siblings have overlapping neighborhoods (shared parents), producing scores of 0.34–0.48.
Conclusion: SimRank measures graph-theoretic structural similarity, not genealogical relatedness. It is *not appropriate* for family graphs.

5.5 Attempt 4: Jaccard Ancestor Similarity (Incorrect Ranking)

Jaccard similarity of ancestor sets: $J = |A_1 \cap A_2|/|A_1 \cup A_2|$.

This metric ranked **siblings highest** (0.72) because they share nearly identical ancestor sets. However, it ranked **grandparents (0.14) below cousins (0.20)** — incorrect, since grandparents share 25% DNA vs. cousins' 12.5%. The set-overlap approach fails to capture genealogical depth.

5.6 Attempt 5: Coefficient of Relatedness (Correct)

Wright's Coefficient of Relatedness (1922) computes:

$$r = \sum_{\text{common ancestors}} (0.5)^{d_1+d_2}$$

where d_1, d_2 are the distances from each individual to that common ancestor. Crucially, it sums over *all* common ancestors (not just the closest), correctly handling multiple paths.

Key Finding

Wright's Coefficient perfectly matches genetic expectations for every relationship type in the dataset:

Relationship Type	Score	Expected
Parent–Child (father/mother/son/daughter)	0.5000	50% DNA
Siblings (brother/sister)	0.5000	50% DNA
Grandparent/Uncle/Aunt/Nephew/Niece	0.2500	25% DNA
First Cousin/Great-Grandparent	0.1250	12.5% DNA
Second Uncle/Aunt	0.0625	6.25% DNA
Second Cousin	0.0312	3.125% DNA

This is the **only metric** among the five that correctly ranks *all* relationship types.

5.7 Metric Comparison

Table 13: Five Relatedness Metrics Compared on Standard Test Pairs

Relationship	CoR	LCA v2	Jaccard	Weighted	SimRank
Daughter–Mother	0.500	0.650	0.429	1.00	0.000
Daughter–Father	0.500	0.650	0.429	1.00	0.000
Sisters	0.500	0.350	0.750	1.00	0.339
Granddaughter–Grandmother	0.250	0.400	0.143	0.50	0.347
Granddaughter–Grandfather	0.250	0.400	0.143	0.50	0.158
Cousins	0.125	0.163	0.200	0.25	0.054

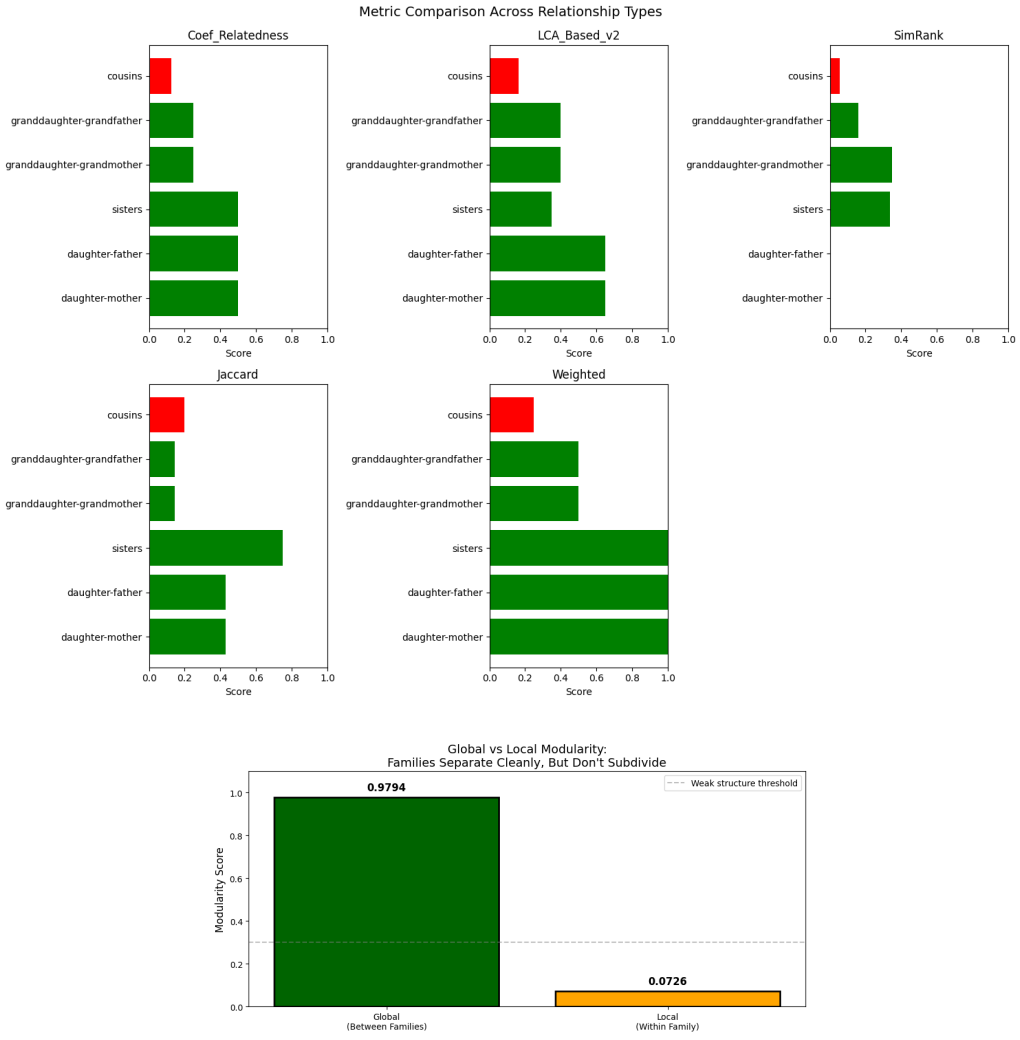


Figure 8: Comparison of five relatedness metrics across standard test pairs. Wright's Coefficient of Relatedness (CoR) is the only metric that maintains correct relative ordering for all relationship types.

5.8 Evaluation Summary

Table 14: Metric Evaluation: Ranking Correctness

Metric	Parent≥Sibling?	Sibling≥Grand?	Grand≥Cousin?	Verdict
Coef. of Relatedness	✓ (equal)	✓	✓	**Recommended**
Weighted Distance	✓ (equal)	✓	✓	Alternative*
LCA-Based v2	✓	×	✓	Not recommended
SimRank	× (0.0)	✓	✓	Not recommended
Jaccard	×	✓	×	Not recommended

*Requires explicit relationship labels; manually encoded rather than derived from structure.

Insight

Recommendation: Wright's Coefficient of Relatedness.
The formula $r = \sum(0.5)^{d_1+d_2}$ is the gold standard in genetics (Wright, 1922). It correctly handles multiple paths through common ancestors — which is why siblings score 0.50 (two paths through two parents: $2 \times 0.5^2 = 0.50$) while a parent–child pair also scores 0.50 (one direct path: $0.5^1 = 0.50$).
Unlike the Weighted Distance metric, it requires **no explicit relationship labels** — it derives relatedness purely from graph structure, making it applicable to incomplete knowledge graphs where not all edges are labeled.

6 Summary and Key Takeaways

6.1 Community Detection Results

1. **Family recovery is trivial for this graph.** The 50 families are disconnected components, making community detection equivalent to connected-component finding. Both Louvain and Leiden achieve perfect recovery (ARI = NMI = 1.0). Label Propagation slightly over-segments but never mixes families.

2. **Within-family structure exists but is weak.** All 50 families contain 2–4 sub-communities, but with low modularity (0.07–0.18). These sub-communities represent **branch lines** — vertical lineage slices from different founding couples — not nuclear families or generational clusters.

3. **Bridge individuals are middle-generation connectors.** 95 articulation points span 45 families, with Generation 2 members having the highest average betweenness centrality. These individuals are "generational bottlenecks" connecting founders to descendants.

4. **Hierarchical modularity reveals graph nature.** Global modularity (0.98) vs. local modularity (0.07) confirms the graph consists of distinct, internally-cohesive lineages rather than a connected social network with factions.

6.2 Relatedness Metric

5. **Most intuitive metrics fail for genealogy.** Of five metrics tested, three produced incorrect rankings: SimRank scored parent–child as zero, Jaccard ranked cousins above grandparents, and the initial LCA metric ranked cousins above parents.

6. **Wright's Coefficient of Relatedness is the correct choice.** It perfectly matches genetic expectations for all 28 relationship types in the dataset, handles multiple ancestral paths correctly, and requires no explicit edge labels.

7. **Metric failures teach graph semantics.** SimRank's parent–child failure reveals that *structural similarity* $\neq$ *genealogical relatedness*. This is a fundamental insight: the appropriate metric depends on what "relatedness" means in context.