Uncovering and Verifying Optimal Community Structure: A MaxSAT Approach

(Thanks to: Carlos **A**nsótequi[1], Vaidyanathan P. **R**.[2], Stefan **S**zeider[2],)
Hai **X**ia[2,*]

[1] Logic and Optimization Group, University of Lleida, Spain
[2] Algorithms and Complexity Group, TU Wien, Austria

WG2@EuroProofNet 2025

TU WIEN

Vienna Center for Logic and Algorithms

Contents

Motivation

Background

- Understanding community structure is crucial in network science.
- Modularity is a key metric to evaluate clustering quality.

Motivation

Background

- Understanding community structure is crucial in network science.
- Modularity is a key metric to evaluate clustering quality.

Computability VS. Optimality

- Exact modularity computation is NP-hard.
- Heuristic methods often fail to find optimal solutions, according to recent surveys.

Really?

- (e.g., ICCS'23, JCS'24, Aref et al.; NeuroCom'24, Li et al.)

Motivation

Answering Questions

1. Do heuristic methods really fail to get optimal modularity?
2. How can we verify/certify the optimality efficiently?

Our Solution Landscape

1. Modularity optimization methods are surveyed.
2. Efficient exact modularity algorithm is proposed for the verification.
3. Proof logging is utilized for certifying optimality further.

Contents

Modularity

- Measures the strength of community structure in a graph.
- Originally formulated by Newman and Girvan (2004).

Modularity

- Measures the strength of community structure in a graph.
- Originally formulated by Newman and Girvan (2004).
- Given a partition P of the graph:

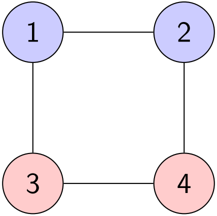
$$Q(P) = \sum_{C \in P} \left[\frac{e(C)}{m} - \left(\frac{\sum_{v \in C} d(v)}{2m} \right)^2 \right]$$

- Where:
  - $e(C)$: Number of edges inside community C
  - $d(v)$: Degree of vertex v
  - m: Total number of edges in the graph
- Optimal clustering maximizes $Q(P)$, with $Q \in [-0.5, 1.0]$

Newman, M. E. J., & Girvan, M. (2004). Finding and evaluating community structure in networks. (citations: $>$ 19000)

Example: Modularity Calculation

- Simple undirected graph with 4 nodes and 4 edges:



- Communities: $C_1 = \{1, 2\}$ (blue), $C_2 = \{3, 4\}$ (red)
- Total number of edges: $m = 4$
- Internal edges: $e(C_1) = 1$, $e(C_2) = 1$
- Degrees: $d(1) = d(3) = 2$, $d(2) = d(4) = 2$
- Modularity:

$$Q = 2 \cdot \left[\frac{1}{4} - \left(\frac{4}{8} \right)^2 \right] = 0.0$$

MaxSAT Overview

- Boolean optimization problem originated from SAT.
- Clauses: Hard (must be satisfied) and Soft (weighted).
- Goal: Satisfy all hard clauses and maximize soft clause weights.

Example: MaxSAT Problem

- **Variables:** x_1, x_2, x_3
- **Clauses:**
  - Hard: $(x_1 \vee x_2)$, $(\neg x_2 \vee x_3)$
  - Soft (with weights):
    - (x_1) [weight 3]
    - $(\neg x_3)$ [weight 2]
- **Goal:** Satisfy all hard clauses and maximize total weight of satisfied soft clauses.

Example: MaxSAT Problem

- **Variables:** x_1, x_2, x_3
- **Clauses:**
  - Hard: $(x_1 \vee x_2)$, $(\neg x_2 \vee x_3)$
  - Soft (with weights):
    - (x_1) [weight 3]
    - $(\neg x_3)$ [weight 2]
- **Goal:** Satisfy all hard clauses and maximize total weight of satisfied soft clauses.

One satisfying assignment: $x_1 = 1, x_2 = 0, x_3 = 1$

Soft clauses satisfied: $(x_1) \rightarrow$ weight 3
Total score: 3 (since $(\neg x_3)$ is false)

Contents

MaxSAT Encoding of Modularity

- Reformulate modularity as a weighted MaxSAT problem.
- Boolean variables x_{uv}: True iff nodes u and v are in the same cluster.
- Hard clauses: Enforce equivalence relation (transitivity):

$$x_{uv} \wedge x_{vw} \rightarrow x_{uw}$$

MaxSAT Encoding of Modularity

- Reformulate modularity as a weighted MaxSAT problem.
- Boolean variables x_{uv}: True iff nodes u and v are in the same cluster.
- Hard clauses: Enforce equivalence relation (transitivity):

$$x_{uv} \wedge x_{vw} \rightarrow x_{uw}$$

- Soft clauses encode modularity gain:

$$w_{uv} \cdot x_{uv} \quad \text{for each node pair } (u, v)$$

  where w_{uv} is the gain when u,v are in the same cluster.
- Objective: Maximize total weight of satisfied soft clauses.

Sparse MaxSAT Encoding

- Reduces the number of transitivity clauses.
- Uses separators $K_G(u, v)$. (a single u, v cut set)
- Same optimal solutions as full encoding.

Theoretical Guarantees

- Theorem 1: MaxSAT weight corresponds to modularity.
- Theorem 2: MaxSAT solution yields optimal clustering.
- Theorem 3: Sparse encoding is equivalent.

Contents

Heuristic Methods

Heuristic methods surveyed before

- Existing heuristics often fall short.
- Aref et al. (2023): Most heuristics fail on larger instances.
- Combo had best success: 90.4%.
- Average across 8 methods: 43.9%.

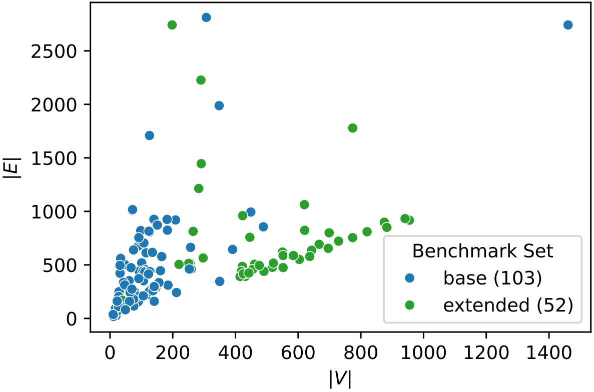
Heuristic Methods

Vienna Clustering Algorithm: The 'Killer'

- Memetic clustering framework (SEA18, Biedermann).
- Ensemble recombination + local search.
- Multi-level and randomized approach.
- Very fast: < 25s per instance.

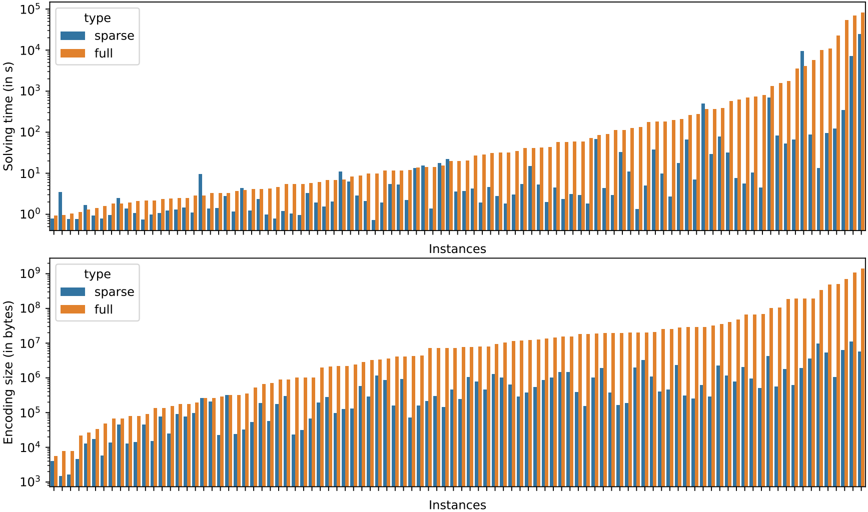
Experimental Setup

- 155 networks: 103 from prior studies, 52 new.
- MaxSAT with MaxHS solver, 48h timeout.



Scatter plot of nodes and edges in benchmark networks

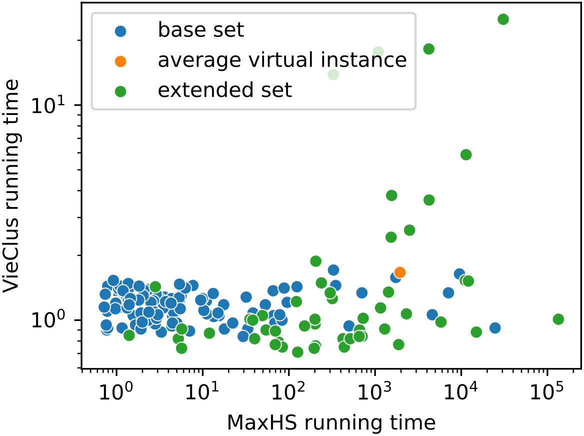
MaxSAT Full vs Sparse Encoding



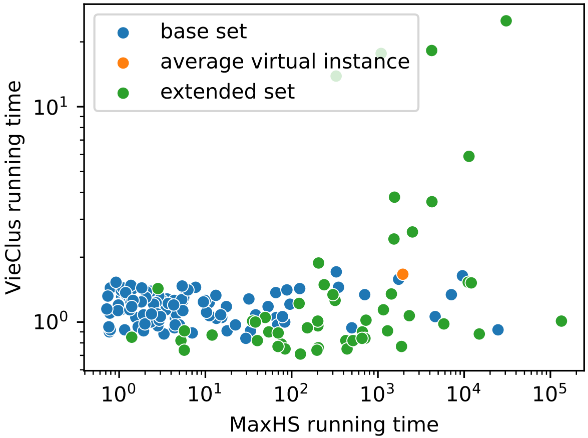
Size and runtime comparison: Full vs sparse encoding

Note: The units are log-scale

MaxSAT (Sparse) vs VieClus



MaxSAT (Sparse) vs VieClus



Note:

- 18 networks have multiple optima.
- MaxSAT can enumerate all optima.
- VieClus can sample only one optima via randomness.

MaxSAT Certifying Pipeline

MaxSAT Solver (Pacose) + Checker (VeriPB)

- The MaxSAT solver outputs proof logs.
- The proof logs are certified again by VeriPB (Certifier).

Initial Attempt

- Logs are too large to store, let alone verify fully.
- Only a limited number of instances (< 10) can be verified.

Engineering Modification

- We modify Pacose to output compressed proof logs.
- E.g., 65 GB $\rightarrow$ 12 GB

MaxSAT with Proof Logging

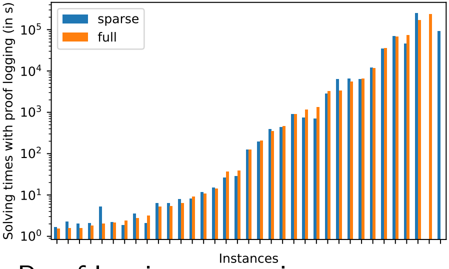
MaxSAT Solver: Pacose

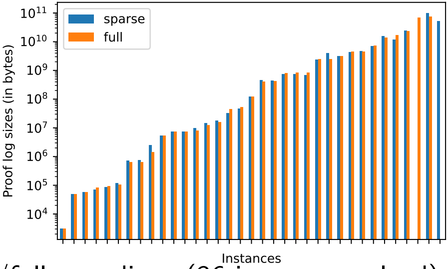
- The MaxSAT solver outputs proof logs.
- The proof logs are certified again by VeriPB (Certifier).

MaxSAT with Proof Logging

MaxSAT Solver: Pacose

- The MaxSAT solver outputs proof logs.
- The proof logs are certified again by VeriPB (Certifier).





Proof logging comparison on sparse/full encodings (36 instances solved)

Contents

Confidence vs Efficiency

1. **VieClus**: Fast, no optimality guarantees. (The real 'SOTA' heuristic method, 'always' getting the optimal solution)
2. **MaxSAT with the Full Encoding**: Guaranteed optima, moderate cost.
3. **MaxSAT with the Sparse Encoding**: Guaranteed optima, moderate cost with faster solving.
4. **MaxSAT with the Proof Logging**: Full verification, high cost.

Contribution Review

Answering Questions

1. Do heuristic methods really fail to get optimal modularity?
2. How can we verify/certify the optimality efficiently?

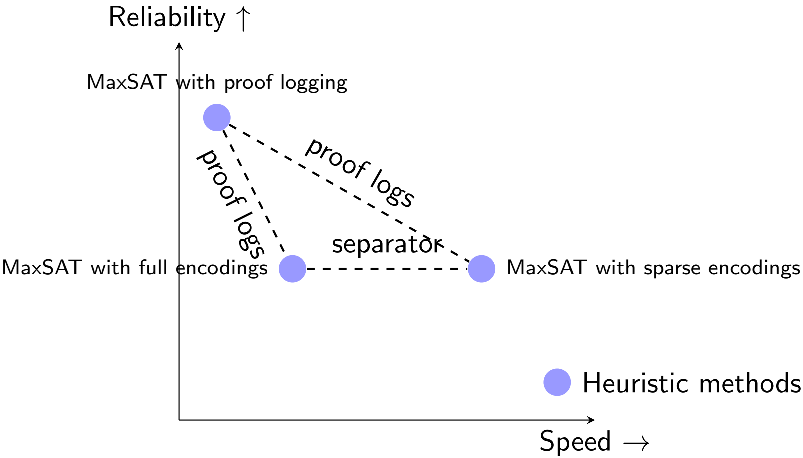
Our Answers

1. No! Previous surveys overlooked the strongest method: VieClus.
2. MaxSAT solving can solve the modularity problem with the optimality guarantee. (Sparse encodings turbocharge the solving.)
3. MaxSAT with the proof logging even can verify the solving further with the highest confidence. (Sparse encodings incur no benefits.)

Future Work

- Investigate why sparse encodings show no benefits under proof logging.
- Scaling proof logging to larger graphs.
- Integrating tuning in MaxSAT solvers.

Q&A



Our Solution Landscape