Uncovering and Verifying Optimal Community Structure in Complex Networks: A MaxSAT Approach

Carlos Ansótegui[1], Vaidyanathan Peruvemba Ramaswamy[2], Stefan Szeider[2], and Hai Xia*[2]

[1] Logic and Optimization Group, University of Lleida, Lleida, Spain
carlos.ansotegui@udl.cat
[2] Algorithms and Complexity Group, TU Wien, Vienna, Austria
{vaidyanathan,sz,hxia}@ac.tuwien.ac.at

Abstract. Network modularity is central to understanding phenomena in diverse domains, from biology and social science to engineering and computational physics. However, computing the optimal modularity—an NP-hard measure quantifying community strength—has remained computationally intractable at large scales. Most approaches resort to heuristics without formal optimality guarantees.
This paper contributes to the computational science of complex systems by introducing a novel MaxSAT-based framework that can compute optimal network modularity values for larger networks than previously possible. Leveraging this new capability, we extensively evaluate heuristic solutions and, for the first time, include the state-of-the-art memetic graph clustering heuristic VieClus. Remarkably, VieClus identifies optimal modularity values for all tested networks, ranging from 103 previously studied instances to 52 new, larger ones, and does so in seconds. This result contrasts with earlier conclusions that heuristics frequently fail to find the optimal modularity.
By combining a powerful MaxSAT encoding, which supports proof logging for verification, with a fast and effective heuristic, we demonstrate that even intricate network structures can be tackled efficiently. This synergy brings us closer to making complex network analysis and community detection tractable, robust, and verifiable—a goal firmly aligned with the core mission of computational science.

Keywords: Complex Networks · Modularity Optimization · Maximum Satisfiability · Graph Clustering Heuristics · Computational Verification

1 Introduction

In network science, the concept of modularity has emerged as a cornerstone for understanding the intricate structure of complex networks. This foundational measure was introduced by Newman and Girvan [25], building on earlier work

* Corresponding author

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

by Freeman [15] (for a more in-depth discussion, we refer to Newman's book [26]). Modularity provides a quantitative lens through which the subtle yet significant patterns of interaction within networks can be discerned, revealing insights pivotal for theoretical exploration and practical application. At its core, modularity measures the strength of division within a network, distinguishing the dense interconnections within communities from the sparser connections between them. This metric has helped to significantly enlarge our understanding of network topology, offering a robust framework to unravel the community structure inherent in various types of networks arising in physics, sociology, biology, telecommunications, and other areas. In propositional satisfiability (SAT), modularity has been used to link the performance of CDCL-based SAT solvers with the structure in industrial instances [1]. MaxSAT is an optimization version of SAT, finding an assignment that maximizes the number of satisfied clauses, and the clauses can be weighted, with a natural extensibility to solving network modularity optimization.

Computing modularity is NP-hard [10], even for trees [12]. As a result, most research focuses on developing heuristics [7–9, 11, 22, 30, 31, 34], while exact ILP-based methods have been proposed but do not scale to large networks [10, 12, 32]. Recent studies by Aref et al. [4, 5] found that near-optimal partitions from heuristics are often disproportionately dissimilar to optimal partitions.

Contributions We challenge these findings through two main contributions. Firstly, we develop an efficient MaxSAT-based approach for computing optimal modularity values, scaling to larger networks than previous exact methods. Our approach provides mathematical guarantees of optimality through its MaxSAT foundation, and additionally enables independent verification of these guarantees through recent advances in proof logging techniques implemented in the Pacose solver. This is in contrast to heuristic algorithms [21, 24] that cannot provide optimality guarantees. Secondly, we analyze the memetic network clustering heuristic VieClus [6], overlooked by previous surveys [4, 5, 21, 24]. We surprisingly found that VieClus finds optimal modularity values for all test networks within seconds—both the 103 networks from their study and 52 additional larger networks.

Our findings have significant implications for computational science: applications in physics, biology, social sciences, and other domains can use VieClus for rapid optimal clustering. When additional verification is needed, our MaxSAT approach provides guaranteed optimality at the cost of longer computation times. For the highest level of trust—crucial for validating scientific results—the proof logging implementation generates independently verifiable certificates for optimality. This portfolio of methods, with increasing levels of verification at corresponding computational cost, makes complex network analysis tractable and trustworthy. Our MaxSAT approach also provides the means for analyzing whether optimal clusterings are unique.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

2 Preliminaries

2.1 Graphs and Networks

We use the terms *graph* and *network* synonymously. All graphs considered are finite, undirected, and simple (i.e., without parallel edges or self-loops). We denote the set of vertices of a graph G by $V(G)$ and the set of edges by $E(G)$. Further, we denote an edge between vertices u and v by uv or, equivalently, vu. We denote the degree of a vertex u by $d(u)$. For subsets $C, C' \subseteq V(G)$, $\mathrm{e}(C, C')$ denotes the number of edges uv of G with $u \in C$ and $v \in C'$; we use $\mathrm{e}(C)$ as a shorthand for $\mathrm{e}(C, C)$.

2.2 Modularity

A *clustering* $\mathcal{P}$ of a graph G is a partition of $V(G)$ into nonempty, mutually disjoint sets, called *clusters*. We will write $u \equiv_{\mathcal{P}} v$ if the vertices u and v belong to the same cluster of $\mathcal{P}$. Following Newman and Girvan [25], the *modularity* $Q(\mathcal{P})$ of a clustering of a graph G with m edges is given by

$$Q(\mathcal{P}) := \sum_{C \in \mathcal{P}} \left[\frac{\mathrm{e}(C)}{m} - \left(\frac{\mathrm{e}(C) + \sum_{C' \in \mathcal{P}} \mathrm{e}(C, C')}{2m} \right)^2 \right]. \tag{1}$$

A clustering $\mathcal{P}$ is *optimal* if $Q(\mathcal{P})$ is maximal. The value $Q(\mathcal{P})$ for an optimal clustering $\mathcal{P}$ of G is called the *modularity* of G. Finding an optimal clustering is an NP-hard optimization problem [10].

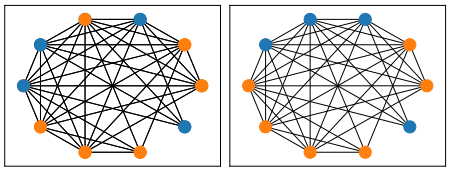
2.3 MaxSAT

A (weighted partial) MaxSAT instance is a propositional formula Φ in conjunctive normal form (CNF), whose clauses are partitioned into hard and soft clauses. Each soft clause has a non-negative integer *weight*. A *solution* τ of a MaxSAT instance Φ is a total truth assignment to Φ that satisfies all the hard clauses. The weight $w(\tau)$ of a solution τ is the sum of weights of all the soft clauses τ satisfies. A solution of maximal weight is *optimal*.

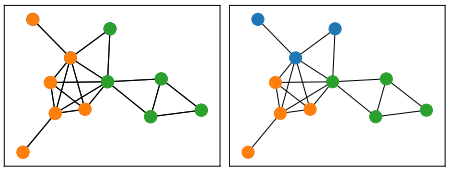
2.4 Graphs with multiple optimal clusterings

It has been empirically observed that most real-world graphs have a unique optimal clustering [5]. However, some graphs do have multiple optimal clusterings. We distinguish between optimal clusterings that are *essentially the same* and those that are *essentially different*. If they are essentially the same, one optimal clustering can be transformed into the other by a symmetry of the graph To define these terms, we need some additional graph-theoretic notions. An *isomorphism* from a graph G to a graph G' is a bijective mapping $\varphi : V(G) \to V(H)$ such that for all pairs $u, v \in V(G)$ we have $uv \in E(G)$ if and only if $\varphi(u)\varphi(v) \in E(H)$. G and H are *isomorphic* if there exists an isomorphism from

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3



(a) Essentially same optimal clusterings for the `dom` network, with modularity value 0.017. In this case, the vertices which switch clusters happen to be so-called twin vertices.



(b) Essentially different optimal clusterings for the `ambassador` network, with modularity value 0.231. In this special case, even the number of clusters is different.

Fig. 1: Two examples of networks with multiple optimal solutions. Nodes which have the same color belong to the same cluster.

G to H. An *automorphism* of a graph G is an isomorphism from G to itself. An automorphism is *non-trivial* if it is not the identity mapping.

We say that two clusterings $\mathcal{P}$ and $\mathcal{P}'$ of a graph G are *essentially the same* if there exists an automorphism φ of G such that $\mathcal{P} = \{\{\varphi(v) \mid v \in C\} \mid C \in \mathcal{P}'\}$; otherwise $\mathcal{P}$ and $\mathcal{P}'$ are *essentially different*. Clearly, if $|\mathcal{P}| \neq |\mathcal{P}'|$ then $\mathcal{P}$ and $\mathcal{P}'$ must be essentially different; however, $|\mathcal{P}| = |\mathcal{P}'|$ does not imply that they are essentially the same. We also note that even if a graph has a non-trivial automorphism, it might have a unique optimal clustering since the automorphism might map vertices to vertices of the same cluster.

We can check whether two clusterings $\mathcal{P} = \{C_1, \ldots, C_k\}$ and $\mathcal{P}' = \{C'_1, \ldots, C'_k\}$ of a graph G are essentially different by using a standard graph isomorphism check. From G, we obtain a graph $G(\mathcal{P})$ by adding k new vertices $v_1, \ldots, v_k$ and all the edges uv_i for $u \in C_i$, $1 \leq i \leq k$. If $\mathcal{P}$ and $\mathcal{P}'$ are essentially the same, then $G(\mathcal{P})$ and $G(\mathcal{P}')$ must be isomorphic. Conversely, if we have found an isomorphism φ from $G(\mathcal{P})$ to $G(\mathcal{P}')$ which induces bijection between $V(G(\mathcal{P})) \setminus V(G)$ and $V(G(\mathcal{P}')) \setminus V(G)$, then $\mathcal{P}$ and $\mathcal{P}'$ are essentially the same.

3 MaxSAT Encoding of Modularity

We now reformulate the definition of modularity as given in (1) so that it can easily be turned into a MaxSAT encoding. Again, let G be the input graph and $m = |E(G)|$. We define the *gain* of a pair $u, v \in V(G)$ as

$$g(u,v) = \begin{cases} 2m - d(u)d(v) & \text{if } uv \in E(G); \\ \quad - d(u)d(v) & \text{otherwise.} \end{cases}$$

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

For a clustering $\mathcal{P}$ of G and a pair $u, v \in V(G)$, we define the *offset gain of* u, v *relative to* $\mathcal{P}$ as

$$g^{+}_{\mathcal{P}}(u,v) = \begin{cases} g(u,v) & \text{if } g(u,v) \geq 0 \text{ and } u \equiv_{\mathcal{P}} v; \\ |g(u,v)| & \text{if } g(u,v) < 0 \text{ and } u \not\equiv_{\mathcal{P}} v; \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

and

$$Q^{+}(\mathcal{P}) = \sum_{u<v\in V(G)} g^{+}_{\mathcal{P}}(u,v). \tag{3}$$

Theorem 1. *For every graph G, there are integers r and s such that for every clustering $\mathcal{P}$ of G, we have $Q(\mathcal{P}) = rQ^{+}(\mathcal{P}) + s$. Consequently, $\mathcal{P}$ maximizes $Q^{+}(\mathcal{P})$ if and only if $\mathcal{P}$ is optimal.*

Proof. Let Ω be the sum of $|w(u,v)|$ over all pairs $u < v \in V(G)$ with $w(u,v) < 0$ and $A_{u,v} \in \{0,1\}$ such that $A_{u,v} = 1$ if and only if $uv \in E(G)$. Then, we can rewrite (3) as

$$\begin{aligned} Q^{+}(\mathcal{P}) &= \Omega + \sum_{C\in\mathcal{P}} \sum_{u<v\in C} g(u,v) \\ &= \Omega + \sum_{C\in\mathcal{P}} \sum_{u<v\in C} \big(2mA_{u,v} - d(u)d(v)\big) \\ &= \Omega + \sum_{C\in\mathcal{P}} \left(2m\cdot \mathrm{e}(C) - \sum_{u<v\in C} d(u)d(v)\right). \end{aligned}$$

We have

$$\begin{aligned} \sum_{u<v\in C} d(u)d(v) &= \frac{1}{2}\left(\sum_{u,v\in C} d(u)d(v) - \sum_{u\in C} d(u)^2\right) \\ &= \frac{1}{2}\left(\sum_{u\in C} d(u) \sum_{v\in C} d(v) - \sum_{u\in C} d(u)^2\right) \\ &= \frac{1}{2}\left(\sum_{u\in C} d(u)\right)^2 - \frac{1}{2}\sum_{u\in C} d(u)^2. \end{aligned}$$

Setting $\Omega' = \Omega + \frac{1}{2}\sum_{u\in V(G)} d(u)^2$ and observing that $\mathrm{e}(C) + \sum_{C'\in\mathcal{P}} \mathrm{e}(C,C') = \sum_{v\in C} d(v)$, we can further rewrite (3) as

$$\begin{aligned} Q^{+}(\mathcal{P}) &= \Omega' + \sum_{C\in\mathcal{P}} \left[2m\cdot \mathrm{e}(C) - \frac{1}{2}\left(\sum_{u\in C} d(u)\right)^2\right] \\ &= \Omega' + 2m^2 \sum_{C\in\mathcal{P}} \left[\frac{\mathrm{e}(C)}{m} - \left(\frac{\sum_{u\in C} d(u)}{2m}\right)^2\right] \\ &= \Omega' + 2m^2 Q(\mathcal{P}). \end{aligned}$$

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

Hence, the claim of the theorem holds for $r = 1/2m^2$ and $s = \Omega'$. □

This theorem gives rise to a MaxSAT encoding. First, we introduce variables and hard clauses that represent a clustering $\mathcal{P}$ of G. Then, we introduce weighted soft unit clauses to maximize modularity.

Let G be the given graph and $V(G)$ linearly ordered by $<$. For any pair $u, v \in V(G)$ with $u < v$, we introduce a variable $c_{u,v}$, indicating whether u and v are in the same cluster (i.e., $u \equiv_{\mathcal{P}} v$); $c_{v,u}$ denotes the same variable as $c_{u,v}$.

To achieve this, we only need to add clauses that enforce the transitivity of $\equiv_{\mathcal{P}}$, i.e., $u \equiv_{\mathcal{P}} v$ and $v \equiv_{\mathcal{P}} w$ implies $u \equiv_{\mathcal{P}} w$. We thus add the following clause for all triples of distinct vertices $u, v, w \in V(G)$

$$\neg c_{u,v} \vee \neg c_{v,w} \vee c_{u,w}. \tag{4}$$

We call u, v and v, w the *premise pairs*, u, w the *forced pair*, and u and w the *end vertices* of the above transitivity clause.

For each pair $u, v \in V(G)$ with $u < v$ and $g(u, v) > 0$ we introduce a soft clause $c_{u,v}$ with weight $g(u, v)$; for each pair $u, v \in V(G)$ with $u < v$ and $g(u, v) < 0$ we introduce a soft clause $\neg c_{u,v}$ with weight $|g(u, v)|$; we do not need to consider pairs with $g(u, v) = 0$.

This concludes the definition of the MaxSAT instance $\Phi(G)$ representing the modularity of G.

Theorem 2. *A solution τ to $\Phi(G)$ is optimal if and only if τ corresponds to an optimal clustering $\mathcal{C}$ of G.*

Proof. By construction of $\Phi(G)$, the weight of an optimal solution τ to $\Phi(G)$ equals $Q^+(G)$, and by Theorem 1, the clustering corresponding to τ is optimal. □

A drawback of the MaxSAT encoding defined above is that it introduces a cubic number of clauses for enforcing transitivity. Adapting ideas from Din and Thai [12] to our MaxSAT setting, we can omit a large fraction of the transitivity clauses without affecting optimal solutions, thus scaling the MaxSAT encoding to larger graphs. For a graph G and vertices $u, v \in V(G)$ with $u \neq v$, let $K_G(u, v)$ be the smallest set of vertices such that u and v belong to different connected components of the graph $G_{u,v}$ obtained by (i) removing all vertices that belong to $K_G(u, v)$ from G and (ii) removing the edge uv in case $uv \in E(G)$. We will later explain how to efficiently compute the sets $K_G(u, v)$ for all pairs $u, v \in V(G)$.

The *Sparse MaxSAT encoding* $\Phi^*(G)$ for modularity is now obtained from $\Phi(G)$ by limiting the transitivity clauses to a subset; namely, we add the clause (4) only if $v \in K_G(u, w)$.

Theorem 3. *$\Phi(G)$ and $\Phi^*(G)$ have the same optimal solutions.*

Proof. We observe that any optimal solution of $\Phi(G)$ is a solution of $\Phi^*(G)$ of the same weight. We will show that also the converse holds, i.e., any optimal solution of $\Phi^*(G)$ is a solution of $\Phi(G)$ of the same weight, which, together with

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

the previous statement, establishes the theorem. Let τ be an optimal solution of $\Phi^*(G)$ and let G_τ be the graph with $V(G_\tau) = V(G) =: V$ and $E(G_\tau) = \{\, uv \mid \tau(c_{u,v}) = 1 \,\}$. We say that a vertex $v \in V$ is *τ-reachable* from u if G_τ contains a path between u and v, and we say v is *$\tau + G$-reachable* from u if G_τ and G contain the same path between u and v; we call such a path a *$\tau + G$-path*.

Claim 1: For any $u, v \in V$, if v is τ-reachable from u then v is also $\tau + G$-reachable from u. To show the claim, suppose to the contrary that there are $u, v \in V$ such that v is τ-reachable from u but not $\tau+G$-reachable. Let $X \subseteq V$ be the set of all vertices that are $\tau + G$-reachable from u, and let $Y \subseteq V$ be the set of all vertices that are τ-reachable from u. Clearly $X \subseteq Y$, and by assumption $v \in Y \setminus X$. Since v is τ-reachable from u, there must be at least one edge $xy \in E(G_\tau)$ with $x \in X$ and $y \in Y \setminus X$. We obtain a new assignment τ' from τ by setting $\tau'(c_{x,y}) = 0$ for all $x \in X$ and $y \in Y \setminus X$. Note also that no transitivity clause forces such $c_{x,y}$ to true since for any such transitivity clause where x, y is the forced pair, at least one of its premise pairs x', y' would have $x' \in X$ and $y' \in Y \setminus X$ and $c_{x',y'}$ set to true. However, since $xy \notin E(G)$, $g(x, y) < 0$, and so the encoding contains the soft clause $\neg c_{x,y}$ of weight $|g(x, y)|$; hence switching the truth value of $c_{x,y}$ from false to true increases the overall weight of the assignment, i.e., $w(\tau') > w(\tau)$, contradicting our assumption that τ is optimal. Hence, the claim is shown to be true.

Claim 2: Any $v \in V$ that is τ-reachable from some $u \in V$ is a neighbor of u in G_τ. Consider pairs $u, v \in V$ such that v is τ-reachable from u. Let $d(u, v)$ denote the length of a shortest $\tau + G$-path between u and v (such a path exists by Claim 1). We show Claim 2 by induction on $d(u, v)$. If $d(u, v) = 1$, then u, v are adjacent in G_τ, and we are done. Now assume Claim 2 holds for all pairs of distance $d - 1 \geq 0$ and consider a τ-connected pair u, v with $d(u, v) = d$. Consider a shortest $\tau + G$ path P of length d between u and v. P runs through a vertex $x \in K_G(u, v)$. We have $d(u, x), d(x, v) \leq d(u, v) - 1$, hence $ux, xv \in E(G_\tau)$ by the induction hypothesis, i.e., $\tau(c_{u,x}) = \tau(c_{x,v}) = 1$. By construction, $\Phi^*(G)$ contains the transitivity clause $\neg c_{u,x} \vee \neg c_{x,v} \vee c_{u,v}$, consequently $\tau(c_{u,v}) = 1$, and so $uv \in E(G_\tau)$ as required. Hence Claim 2 holds.

By Claim 2, G_τ is a disjoint collection of cliques. Consider any transitivity clause $\neg c_{u,v} \vee \neg c_{v,w} \vee c_{u,w}$ of $\Phi(G)$, with end vertices u, w, and v not necessarily from $K_G(u, w)$. If u, w belong to the same clique of G_τ, then $\tau(c_{u,w}) = 1$ and the clause is satisfied. If u, w belong to different cliques of G_τ, v cannot be in the same clique with u and in the same clique with w, hence $\tau(c_{u,v}) = 0$ or $\tau(c_{v,w}) = 0$, and the clause is satisfied. We conclude that τ is indeed a solution of $\Phi(G)$. This concludes the proof of the theorem. □

4 Accuracy of heuristically computed modularity

As mentioned above, it is NP-hard to determine the modularity of a network exactly [10], even for tree-structured networks [12]. Since exact methods for computing modularity do not scale to large networks, scientists rely on heuristic methods to compute modularity and associated clusterings.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

Aref et al. [4, 5] presented a systematic analysis of heuristic methods for modularity computation. In particular, they considered Clauset-Newman-Moore (CNM) [11], Louvain [7], Reichardt-Bornholdt with the configuration model as the null model (LN) [20], Combo [30], Belief [34], Paris [8], Leiden [31], EdMot-Louvain [22], recurrent graph neural network (GNN) [29], and Bayan [2]. They tested all these approaches on 104 networks (54 real-world and 50 synthetic) and computed the exact modularity via an ILP encoding. Their key findings were quite negative: Combo had the highest success rate among the eight heuristics, returning an optimal partition for 90.4% of the networks; the average success rate of all 8 heuristics combined was only 43.9%. On average, GNNs and Bayan achieve optimality for 68.7% and 82.3% of networks, respectively. Aref et al. [4, 5] used several partition similarity metrics (AMI, RMI, ECS) to quantify differences between partitions and found that suboptimal clusterings tend to be disproportionately dissimilar to the optimal clustering. In summary, their results suggest significant limitations in commonly used heuristics for modularity.

However, Aref et al. did not consider the *Vienna graph clustering* framework (VieClus) by Biedermann et al. [6], an omission that turns out to be significant. VieClus provides a general *memetic algorithm* [23] for various graph clustering problems, including modularity computation. A key innovation in VieClus is using recombine operators that leverage ensemble clusterings and multi-level techniques. These operators create an overlay clustering from two input clusterings, determining whether pairs of vertices should be in the same cluster based on their groupings in the input clusterings. This recombination is then enhanced with local search algorithms to improve the clustering quality further. The algorithm also employs a multi-level approach to find even better clusterings. VieClus emphasizes randomized tie-breaking throughout the process to diversify the search and improve solutions. Additionally, the algorithm incorporates a scalable communication protocol, enabling it to compute high-quality solutions efficiently. This combination of techniques results in a powerful and versatile clustering tool that can reproduce or improve upon previous benchmark results for various graph clustering instances.

In the sequel, we extend the study by Aref et al. by comparing the modularity computed by VieClus with the optimal modularity. Our powerful MaxSAT approach for computing exact modularity allows us to extend the benchmark set with significantly larger networks.

5 Checking optimality of VieClus

5.1 Benchmark Networks

Our benchmark set includes 103 networks[3] that were considered in the previous work [5]. 50 of these networks were synthetic LFR and ABCD graphs generated

[3] Previous work [5] used a benchmark set consisting of 104 networks. However, MaxHS cannot find an optimal solution for the network `physician_trust` within 48 hours, so we exclude it from our experimental analysis.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

by Aref et al. [3], while the remaining 53 graphs are real-world benchmarks drawn from the well-known Netzschleuder repository [28]. Aref et al. selected these networks to be (optimally) solved within a reasonable time by their sparse IP formulation and all the 8 heuristics considered. We denote this as the *base* benchmark set. Further, we extend this set by adding another 52 networks from the Netzschleuder repository [28], and solve the networks optimally with our sparse MaxSAT formulation. We denote these 52 new networks as the *extended* benchmark set. The networks in the extended benchmark set contain up to 1461 vertices and 2812 edges. We show the number of nodes and edges of the networks as a scatter plot in 2, where the blue and orange dots represent the networks from the base and extended benchmark sets, respectively.

The newly added networks in the extended set generally have a larger number of nodes and edges compared to the previously used networks. We also provide detailed information on each network in the supplementary material[4].

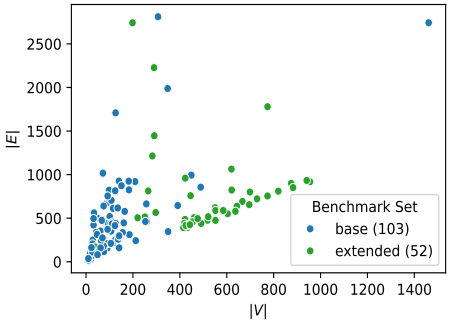


Fig. 2: Scatter plot of the number of nodes ($|V|$) and edges ($|E|$) of the 155 benchmark networks

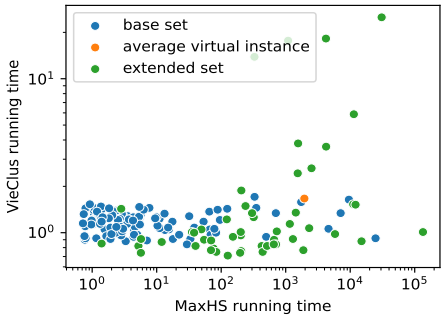


Fig. 3: The running time of MaxHS on sparse encoding vs VieClus

5.2 Experimental Setup

We perform all our experiments on a compute cluster with nodes equipped with two AMD 7403 processors (24 cores at 2.8 GHz) and 32 GB of RAM per core. We use a timeout of 10 hours while running VieClus and a timeout of 48 hours while running the MaxSAT solver. We did preliminary testing with different MaxSAT exact solvers participating in the MaxSAT Evaluation 2022[5], including EvalMaxSAT, MaxCDCL, and MaxHS. Then, we identified MaxHS as the most promising solver for our use case.

We generate the encodings using Python 3.10 along with the NetworkX 3.2 graph library [17] and the PySAT 0.1 library [18]. Recall that to construct the

[4] `https://figshare.com/s/caec9f3c34f5c31488c2`
[5] `https://maxsat-evaluations.github.io/2022/descriptions.html`

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

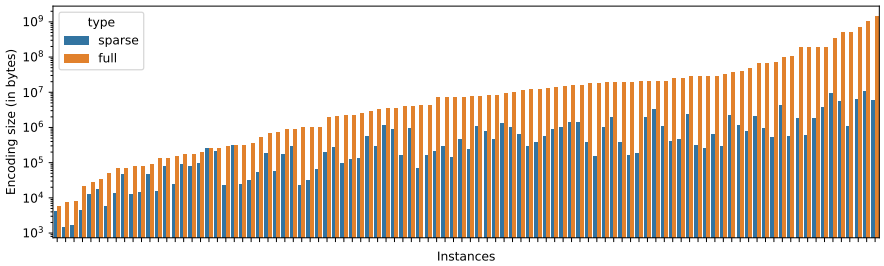


Fig. 4: Comparing the sizes of the sparse vs full encodings

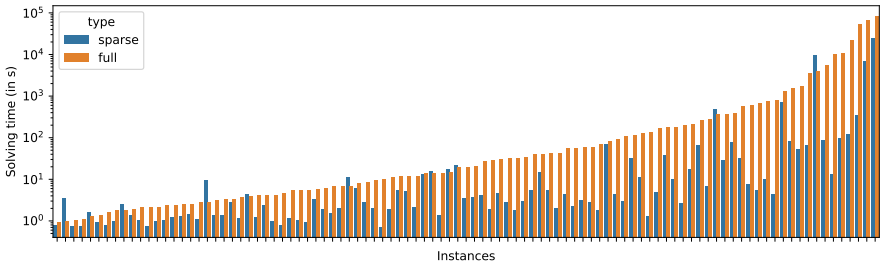


Fig. 5: Comparing the solving times of the sparse vs full encodings

sparse encodings, we need a *separator set* for each pair of vertices. We precompute these separator sets for all networks by adapting the basic functions from the NetworkX library (which compute a single $s-t$ cut using flow-based methods [13, 14, 19]) to run in a parallel setting. The benchmark networks and the scripts for generating the MaxSAT encodings for those networks are available in the supplementary material.

5.3 Results

Sparse vs. Full Encodings We use MaxHS to find the optimal modularity of the 103 networks of the base benchmark set with both the full and the sparse encodings. We then compare the file sizes of the two encodings and their respective solving times. In Figs. 4 and 5, it should be noted that the axes are in logarithmic scale. In Fig. 4, we sort the networks according to the file size of their full encodings. We can see that on the larger networks, the sparse encodings result in a more than 10-fold reduction compared to the corresponding full encodings.

Further, we also compare the solving efficiency of MaxSAT with both the full and the sparse encodings on the base benchmark set. From Fig. 5, sparse encodings can speed up the solving time on many large networks by a factor of 5 compared to the full encoding. For several networks, the full encoding is quicker than the sparse encoding; however, most of these networks are too small

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

to take advantage of the sparse encoding. More specifically, all such networks are solved within 20 seconds by MaxHS on both encodings. On average, the MaxSAT sparse encoding can reduce the solving time by 57.59% among the 103 networks. Considering the space and time savings, our first argument is justified: The sparse MaxSAT encoding provides a higher solving efficiency for computing optimal modularity, especially on larger networks.

VieClus vs. MaxHS with Sparse Encoding We compute the optimal modularity values for all 155 networks in the base benchmark set and the extended benchmark set using MaxHS to solve the sparse MaxSAT encodings. We then compare the modularity values obtained by VieClus with the optimal values obtained by MaxHS. Amazingly, we find that they match on all networks, i.e., VieClus obtains the optimal modularity for all the networks. Furthermore, regarding time consumption, VieClus requires a lot less time than MaxHS (roughly an order of magnitude faster).

Fig. 3 compares the time consumption between MaxHS with the sparse encoding (y-axis) and VieClus (x-axis), where each dot represents a network. The blue and green dots represent the networks from the base and extended sets, respectively. We emphasize that this is for reference only, not a direct comparison between the two algorithms: MaxHS finds an optimal clustering and verifies the optimality, whereas VieClus finds a clustering that happens to be optimal. However, the plot indicates which instances are harder and which are easier for the two approaches.

Regarding the average time consumption, as the orange virtual dot (average solving time of the full and sparse encodings) shown in Fig. 3, the MaxHS takes over 2000 seconds on average to get the optimal modularity values, but VieClus achieves the same modularity values within only 2 seconds, although without verifying their optimality.

In general, on all 155 networks from the base set and the extended set (which were selected such that MaxHS can find an optimal solution within 48 hours), VieClus always manages to match the optimal modularity value (and it does so in under 25 seconds), even though it is only a heuristic algorithm. Our conclusion significantly differs from recent claims [4, 5] that heuristic methods perform poorly and rarely return an optimal partition.

5.4 Analysis of Networks with Multiple Optimal Solutions

A vast majority of the considered networks, 137 out of 155, have a unique optimal clustering; therefore, the optimal clusterings found by VieClus and MaxHS are necessarily the same. As for the remaining 18 networks, we used the MaxSAT encoding to enumerate *all* optimal clusterings. To this end, when MaxHS finds an optimal clustering, the corresponding satisfying assignment is negated and added to the encoding, thereby forbidding the same solution from being discovered again. Now, if MaxHS finds a solution to this new encoding with the same modularity value as before, we will have found another clustering with the optimal modularity. The moment the modularity value drops, we know for certain

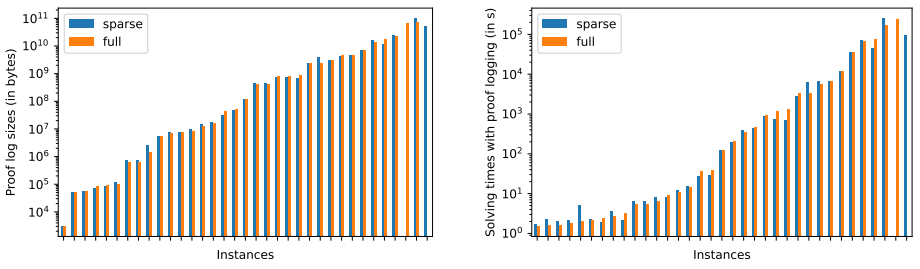
ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

that all clusterings with the optimal modularity value have been enumerated. Furthermore, by analyzing the isomorphisms between the optimal clusterings of these networks, we found that for only 11 of them, the optimal clusterings are essentially different (details are available in the supplementary material).

Due to its randomized nature, VieClus can find different optimal clusterings when run with different random seeds. We verify this by running VieClus repeatedly on some of the networks that have multiple optimal clusterings. However, in contrast to the MaxSAT approach, we cannot block previously found optimal clusterings for VieClus. Hence, it is up to chance that repeated runs will eventually find all optimal clusterings. In practice, we notice that VieClus has a tendency to gravitate towards the same solution. For example, in several instances, despite trying more than 1000 random seeds, VieClus always returns the same optimal clustering. However, there were also a handful of instances where repeated runs of VieClus were eventually able to discover all optimal clusterings.

6 Verifying Optimality Through Proof Logging

State-of-the-art MaxSAT solvers are complex algorithms with lots of moving parts. If they claim a solution is optimal, we cannot always blindly trust this. This limitation can be significant for computational science applications where the high trustworthiness of results is crucial. We address this gap using *proof logging* in MaxSAT solving, which provides machine-verifiable certificates of optimality. Specifically, we use Pacose [27], a state-of-the-art MaxSAT solver that outputs VeriPB [16] proofs—formal certificates that can be independently verified. These proofs record every step in the solving process, from CNF encoding verification to optimality confirmation, allowing third parties to validate results without trusting the solver implementation. Since the proof logs output by Pacose by default can get large, we modify Pacose to directly output compressed proof logs. For instance, compressing a 65 GB proof log can reduce its size to 12 GB. We include the modified version of Pacose along with some generated proof logs in the supplementary material.



(a) Proof log sizes (compressed) (b) Solving times with proof logging

Fig. 6: Comparing sparse vs full encoding when solving with proof logging

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

We run Pacose with proof logging enabled on both sparse and full encodings with a 72-hour timeout. Of the 155 benchmark networks, we attempted proof logging on 91 networks. Out of these, 36 networks were successfully solved with verifiable proofs. Of these 36 networks, the network mu_0_01_LFR_0 was solved only by the full encoding, and the network 7th_graders was solved only by the sparse encoding. The largest number of nodes and edges in a network that was solved with proof logging are 141 and 1017, respectively. Figs. 6a and 6b compare the proof log sizes and solving times, respectively, of the sparse and full encoding when proof logging is enabled.

Surprisingly, while sparse encodings are consistently faster for regular MaxSAT solving, this advantage disappears with proof logging. For approximately half the instances, full encodings resulted in faster solving times. The proof log sizes were also comparable between sparse and full encodings.

This illustrates the three levels of confidence and effort balance that one can achieve with the different methods we consider. On one end, heuristic methods like VieClus can quickly find optimal solutions but do not provide any guarantees. On the other end, MaxSAT solving with proof logging offers the highest level of rigor but requires significant computational overhead. Finally, MaxSAT solving without proof logging is a good middle ground. It provides optimality guarantees with a reasonable degree of confidence while maintaining scalability to a decent extent.

7 Conclusion and Future Work

In our work, we introduce a novel MaxSAT-based approach for computing optimal modularity and demonstrate that VieClus consistently finds optimal clusterings orders of magnitude faster than exact methods, which challenges previous findings about heuristic unreliability in modularity computations. Our findings provide a variety of options for the modularity analysis:

1. VieClus offers lightning-fast optimal solutions, although without formal guarantees.
2. Our MaxSAT encoding provides optimality guarantees at increased computational cost.
3. For requiring the highest trustworthiness, the proof logging implementation generates independently verifiable certificates of optimality.

The high-efficiency VieClus suggests that network analysis based on optimal modularity is more tractable than previously thought. Simultaneously, our advances in MaxSAT solving with proof logging provide a rigorous foundation when verification is crucial. Future work should explore extending proof logging capabilities to larger networks while maintaining reasonable computational overhead, like integrating the tuning techniques into the (Max)SAT solving [33].

Acknowledgements

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3


This work was funded by the European Union's Horizon 2020 research and innovation program under the Maria Skłodowska-Curie grant agreement No. 101034440, project PID2022-138506NB-C21 (Ministerio de Ciencia e Innovación), and project 10.55776/P36420 (Austrian Research Funds). The full MaxSAT encoding is based on an encoding developed in Rupert Ettrich's Bachelor's Thesis, TU Wien, 2020. The authors also thank Jakob Nordström and Andy Oertel for assisting with VeriPB proof logging.

References

1. Ansótegui, C., Levy, J.: On the modularity of industrial SAT instances. In: CCIA 2011. FAIA vol. 232, pp. 11–20. IOS Press (2011).
2. Aref, S., Chheda, H., Mostajabdaveh, M.: The Bayan algorithm: Detecting communities in networks through exact and approximate optimization of modularity. CoRR **abs/2209.04562** (2022). https://doi.org/10.48550/ARXIV.2209.04562
3. Aref, S., Mostajabdaveh, M.: Dataset of synthetic modular graphs from LFR and ABCD benchmark models for community detection. Figshare (2023). https://doi.org/10.6084/m9.figshare.24257293.v1
4. Aref, S., Mostajabdaveh, M.: Analyzing modularity maximization in approximation, heuristic, and graph neural network algorithms for community detection. JCS **78**, 102283 (2024). https://doi.org/10.1016/J.JOCS.2024.102283
5. Aref, S., Mostajabdaveh, M., Chheda, H.: Heuristic modularity maximization algorithms for community detection rarely return an optimal partition or anything similar. In: ICCS 2023. LNCS vol. 14076, pp. 612–626. Springer (2023).
6. Biedermann, S., Henzinger, M., Schulz, C., Schuster, B.: Memetic graph clustering. In: SEA 2018. LIPIcs, vol. 103, pp. 3:1–3:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPICS.SEA.2018.3
7. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. JSTAT **2008**(10), P10008 (2008). https://doi.org/10.1088/1742-5468/2008/10/P10008
8. Bonald, T., Charpentier, B., Galland, A., Hollocou, A.: Hierarchical graph clustering using node pair sampling. In: MLG (2018).
9. Bouguessa, M., Missaoui, R., Talbi, M.: A novel approach for detecting community structure in networks. In: ICTAI 2014. pp. 469–477 (2014). https://doi.org/10.1109/ICTAI.2014.77
10. Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., Wagner, D.: On modularity clustering. IEEE TKDE **20**(2), 172–188 (2008). https://doi.org/10.1109/TKDE.2007.190689
11. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E **70**, 066111 (2004). https://doi.org/10.1103/PhysRevE.70.066111
12. Dinh, T.N., Thai, M.T.: Toward optimal community detection: From trees to general weighted networks. Internet Math. **11**(3), 181–200 (2015). https://doi.org/10.1080/15427951.2014.950875
13. Esfahanian, A.H.: Connectivity algorithms. Topics in structural graph theory pp. 268–281 (2013)
14. Even, S.: Graph Algorithms. WH Freeman & Co. (1979)
15. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**(1), 35–41 (1977). https://doi.org/10.2307/3033543

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3

16. Gocht, S., McCreesh, C., Nordström, J.: Veripb: The easy way to make your combinatorial search algorithm trustworthy. In: CPTAI workshop at CP (2020).
17. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using NetworkX. In: SciPy 2008. pp. 11–15. (2008)
18. Ignatiev, A., Morgado, A., Marques-Silva, J.: PySAT: A Python toolkit for prototyping with SAT oracles. In: SAT 2018. pp. 428–437 (2018). https://doi.org/10.1007/978-3-319-94144-8_26
19. Kammer, F., Täubig, H.: Connectivity, pp. 143–177. Springer (2005). https://doi.org/10.1007/978-3-540-31955-9_7
20. Leicht, E.A., Newman, M.E.J.: Community structure in directed networks. Phys. Rev. Lett **100**, 118703 (2008). https://doi.org/10.1103/PhysRevLett.100.118703
21. Li, J., Lai, S., Shuai, Z., Tan, Y., Jia, Y., Yu, M., Song, Z., Peng, X., Xu, Z., Ni, Y., Qiu, H., Yang, J., Liu, Y., Lu, Y.: A comprehensive review of community detection in graphs. Neurocomputing **600**, 128169 (2024). https://doi.org/https://doi.org/10.1016/j.neucom.2024.128169
22. Li, P., Huang, L., Wang, C., Lai, J.: Edmot: An edge enhancement approach for motif-aware community detection. In: KDD 2019. pp. 479–487. ACM (2019). https://doi.org/10.1145/3292500.3330882
23. Moscato, P., Cotta, C.: A gentle introduction to memetic algorithms. In: Handbook of Metaheuristics, ISOR vol. 57, pp. 105–144. Kluwer / Springer (2003). https://doi.org/10.1007/0-306-48056-5_5
24. Nascimento, M.C., de Carvalho, A.C.: Spectral methods for graph clustering – A survey. EJOR **211**(2), 221–231 (2011). https://doi.org/https://doi.org/10.1016/j.ejor.2010.08.012
25. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69**, 026113 (2004). https://doi.org/10.1103/PhysRevE.69.026113
26. Newman, M.: Networks. Oxford University Press, 2nd edn. (2018)
27. Paxian, T., Reimer, S., Becker, B.: Pacose: An iterative SAT-based MaxSAT solver. MaxSAT Evaluation **2018**, 20 (2018)
28. Peixoto, T.P.: The Netzschleuder network catalogue and repository (2023). https://doi.org/10.5281/zenodo.7839981
29. Sobolevsky, S., Belyi, A.: Graph neural network inspired algorithm for unsupervised network community detection. Appl. Netw. Sci. **7**(1), 63 (2022). https://doi.org/10.1007/S41109-022-00500-Z
30. Sobolevsky, S., Campari, R., Belyi, A., Ratti, C.: General optimization technique for high-quality community detection in complex networks. Physical Review E **90**, 012811 (2014). https://doi.org/10.1103/PhysRevE.90.012811
31. Traag, V., Waltman, L., van Eck, N.J.: From Louvain to Leiden: guaranteeing well-connected communities. Nature Scientific Reports **9**(5233) (2019). https://doi.org/10.1038/s41598-019-41695-z
32. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. JMLR **11**(95), 2837–2854 (2010).
33. Xia, H., Szeider, S.: SAT-Based tree decomposition with iterative cascading policy selection. In: AAAI **38**(8), 8191–8199 (2024). https://doi.org/10.1609/aaai.v38i8.28659
34. Zhang, P., Moore, C.: Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. Proc. Natl. Acad. Sci. U.S.A **111**(51), 18144–18149 (2014)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_3