



Parameter-light algorithms for improving hypermodularity-based community detection

Fabian Brandt-Tumescheit¹ · Henning Meyerhenke²

Received: 31 May 2026 / Revised: 24 July 2026 / Accepted: 25 July 2026
© The Author(s) 2026

Abstract

Community detection is a central problem in network science, traditionally performed on graphs, often via modularity maximization. Yet, many real-world systems inherently involve higher-order interactions between more than two entities, naturally modeled as hypergraphs. In such contexts, pairwise projections obscure higher-order structure and may lead to misleading communities. Recent works thus addressed higher-order community detection, with hypermodularity as one promising candidate. The state-of-the-art algorithm H-LOUVAIN [Kaminski et al., J. ComNetw 2024] mixes greedy optimization techniques for both modularity and hypermodularity in a multilevel scheme. While being effective with default parameters, the algorithm lacks guidance on how to combine the two objectives and thus needs parameter tuning. This parameter tuning step uses probabilistic strategies such as Bayesian Optimization Technique (BOT), resulting in a very significant overhead in running time. In this paper, we propose three algorithmic variants of H-LOUVAIN to recover communities of high accuracy without the need for time-consuming searches of mixing parameters. Our variants consist of new ideas for creating starting solutions for greedy multilevel algorithms and additional post-processing steps inspired by the community detection algorithm LEIDEN [Traag et al., SciRep 2019] for graphs. In extensive experiments with over 50 real-world and randomly generated graphs, we show that across nearly all test cases, a member of our algorithm suite matches or surpasses H-LOUVAIN w. r. t. Asymmetric RMI; hence, they provide a more faithful community representation than the state of the art.

Keywords Community detection · hypergraphs · multilevel heuristics · hypermodularity

1 Introduction

Complex systems are often modeled as networks that capture only pairwise interactions. However, numerous important phenomena in relevant applications involve *group interactions*: (i) social systems such as epidemic-spreading models assume simultaneous contact among several individuals (Wang et al. 2024), (ii) in biology, group interactions are used to identify gene sets that jointly respond to

neural stimuli (Feng et al. 2021), and (iii) in neuroscience, analyses using group interactions improve the detection of autism spectrum disorder (Yang et al. 2023) and migraine patterns (Shen et al. 2024). Unlike graphs, hypergraphs can connect arbitrary subsets of nodes, thus providing a *faithful representation* of higher-order interactions. Moreover, transforming hypergraphs to graphs, e. g., using the clique expansion, yields not only inherent information loss, it also leads to potential memory overflows for large hyperedges (Fig. 1).

Community detection (CD) is a central research area in network mining (Brahim et al. 2024). When applied to hypergraphs (see the example in Figure 1), CD can reveal more detailed communities than graph-based methods, as shown by Benati et al. (2025) via modularity extensions to hypergraphs. Existing hypergraph CD approaches fall into two broad families, where one family expects a fixed number of communities as an input parameter (Ke et al. 2019; Chodrow et al. 2023, 2021; Ruggeri et al. 2023; Chodrow

✉ Fabian Brandt-Tumescheit
brandtfa@hu-berlin.de

Henning Meyerhenke
meyerhenke@kit.edu

¹ Department of Computer Science, Humboldt-Universität zu Berlin, Unter den Linden 6, 10117 Berlin, Germany

² Faculty of Informatics, Karlsruhe Institute of Technology (KIT), KaiserstraSe 12, 76131 Karlsruhe, Germany

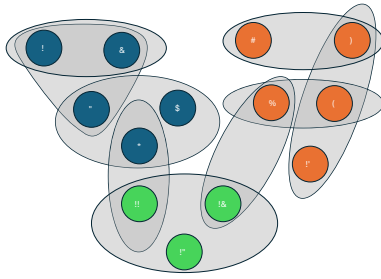


Fig. 1 Example hypergraph with community membership shown as vertex color. Hyperedges are shown by areas shaded in grey

et al. 2021), while the other infers this information automatically. The latter family includes two papers: one uses clique expansion with a weighting scheme to reduce the hypergraph to a graph (Kumar et al. 2020, the other one a bipartite graph representation (Benati et al. 2025)). The algorithm H-LOUVAIN (Kamiński et al. 2024), in turn, operates directly on the hypergraph. It uses a *mixed-modularity* technique that combines hypergraph modularity and graph modularity on the same hypergraph in a single objective function. This approach exploits the fact that (hyper)modularity on a hypergraph equals (graph) modularity on its clique expansion, while it respects the original hypergraph structure. However, this approach raises two major open issues: (i) the required number of steps for computing the mixing parameters for the combined objective is unclear and may depend on the optimization process and (ii) the parameter tuning relies on a Bayesian optimization technique (BOT), which is very compute-intensive. These limitations motivate an improved framework for hypergraph CD by being less parameter-dependent and of at least comparable solution quality.

Contributions In this paper, we first propose two new H-LOUVAIN variants that eliminate the need for intricate and compute-intensive parameter tuning. Their respective central ideas are different heuristic starting solutions: F-LOUVAIN (F for fast) is based on one round of graph-based modularity maximization, whereas P-LOUVAIN (P for partitioning-based) uses k -way hypergraph partitioning. When partitioning, the number of communities k can be guessed easily, making complex assumptions and time-consuming optimization in H-LOUVAIN obsolete. We also propose H-LEIDEN, a generalization of the popular graph-based Leiden algorithm to hypergraphs, equipped with a hypergraph-cut metric. H-LEIDEN aims for well-connected communities and, due to an expanded search space, solutions with higher quality. Our extensive experiments on a larger corpus (> 50 real-world and synthetic hypergraphs) than previous studies use hypermodularity values as well as comparisons to ground

truth based on the ARMI measure (Jerdee et al. 2025) to assess the solution quality. The results show that our algorithm suite with the three algorithms matches or exceeds H-LOUVAIN regarding ARMI, hence produces more faithful community representations.

Outline The remaining paper is structured as follows: In Sect. 2, we first define the problem formally and settle notation; we then discuss related work, with a particular emphasis on the H-LOUVAIN algorithm. Section 3 explains our proposed algorithms F-LOUVAIN, P-LOUVAIN, and H-LEIDEN. Our experiments are described, evaluated, and discussed in Sect. 4. Finally, we draw conclusions from our findings in Sect. 5.

2 Preliminaries

2.1 Problem definition and notation

Let $H = (V, \mathcal{E})$ be a hypergraph with a node set V and a hyperedge set $\mathcal{E}$, where each hyperedge $E \in \mathcal{E}$ is a subset of V . The hypergraph has two weight functions, $\omega_e : \mathcal{E} \rightarrow \mathbb{N}^+$ and $\omega_n : \mathcal{E} \times V \rightarrow \mathbb{N}$. ω_e assigns each hyperedge a positive weight, representing its importance. For ω_n , the assigned weight is specific to a node-hyperedge pair, with $\omega_n(E, v) = 0$ if $v \notin E$. In the following, we refer to these latter weights as node-edge weights. The weighted degree of a node is defined as $\deg(v) := \sum_{E \in \mathcal{E}} \omega_n(E, v)$. Summing over $\deg(v)$ for all nodes in a set S forms the volume $\text{vol}(S) := \sum_{v \in S} \deg(v)$.

A community C in a hypergraph contains a subset of nodes. The set of all communities $\mathcal{C} := \{C_1, \dots, C_k\}$ forms a clustering of H . $\mathcal{C}$ consists of k subsets C_i with $i \leq k$, where each C_i consists of nodes of V and $\bigcup C_i = V$. In this work, we restrict ourselves to the case where any two communities C_i and C_j are pairwise disjoint. The goal of CD, when regarded as an optimization process, is to find a clustering $\mathcal{C}_{opt}$ that optimizes a specified objective function. One objective function for hypergraphs, motivated by the popularity of its counterpart for graphs, is hypermodularity q_H :

$$q_H(\mathcal{C}) = \sum_{d \geq 2} \sum_{c=|d/2+1|}^d \eta_{c,d} q_H^{c,d}(\mathcal{C}). \quad (1)$$

Hypermodularity optimizes for associative communities, meaning that nodes in a community should share more connections with each other than with the rest of the hypergraph. The parameters c and d denote the volume of a community

inside a hyperedge E with $\text{vol}(E) = d$. Based on these two, the hyperparameter $\eta_{c,d} \in [0, 1]$ controls how fractions of c and d influence hypermodularity.

$q_H^{c,d}(\mathcal{C})$ is the core scoring function with various components: For a clustering $\mathcal{C}$ of a hypergraph H , $q_H^{c,d}(\mathcal{C})$ rates how many hyperedges of size d have exactly c members in each community of $\mathcal{C}$ compared to what we would expect from a null model based on random hypergraphs. Let $\mathcal{E}_d := \{E \in \mathcal{E} : \text{vol}(E) = d\}$ be a set of hyperedges with volume d and $e_H^{c,d}(C)$ be the number of such hyperedges with exactly c members in community C . $q_H^{c,d}(\mathcal{C})$ is then defined as:

$$q_H^{c,d}(\mathcal{C}) = \frac{1}{\mathcal{E}} \sum_{C \in \mathcal{C}} \left(e_H^{c,d}(C) - |\mathcal{E}_d| \cdot \mathbb{P} \left(\text{Bin} \left(d, \frac{\text{vol}(C)}{\text{vol}(V)} \right) = c \right) \right). \quad (2)$$

In the subtrahend, $\text{Bin}(d, p)$ is a binomial random variable that represents the null model: $\mathbb{P}(\text{Bin}(d, p) = c) = \binom{d}{c} p^c (1-p)^{d-c}$. We refer to the complete subtrahend also as the *degree tax* $t_d(C)$. Optimizing Eq. (1) mainly depends on the choice of $\eta_{c,d}$. One choice is to set $\eta_{c,d} := (c/d)^\tau$ with $\tau \in [0, \infty)$ (Kamiński et al. 2022). By varying τ , the scores from $q_H^{c,d}(\mathcal{C})$ are scaled (and nullified if $\eta_{c,d} = 0$). Proposed values are $\tau := 0$ (majority modularity), $\tau := 1$ (linear modularity), $\tau := 2$ (quadratic modularity) and $\tau := \infty$ (strict modularity). The latter is a special case, since $\eta_{c,d} = 1$ only if $c = d$ and 0 for all other combinations of c and d . For strict modularity, the inner sum in Eq. (1) collapses and only hyperedges where all nodes are part of one community C are summed to obtain the final score q_H . If the hypergraph has only dyadic interactions, the hypermodularity parametrized with τ reduces to graph modularity. That is why the strong $\mathcal{NP}$ -hard-hardness of modularity optimization for graphs (Brandes et al. 2007) carries over directly to the optimization of Eq. (1) for hypergraphs.

As a rule of thumb, high scores for q_H represent a clustering with communities of (relatively) dense, interconnected nodes and a low relative volume $\frac{\text{vol}(C)}{\text{vol}(V)}$. Note that Eq. (1) is restricted to summands where the majority of one hyperedge is part of one community. More variants of hypermodularity and other optimization techniques for CD on hypergraphs are discussed in Sect. 2.2.

2.2 Related work

Finding communities in networks is one of the most studied topics in the field of network analysis. The various methods range from (i) the optimization of objective functions over

(ii) geometric methods based on distances in an m -dimensional embedding space to (iii) deep-learning methods – to name only the most popular categories. Our focus in this paper and thus also in this section is on the first category. We refer the interested reader to Brahim et al. (2024); Rossetti et al. (2019); Su et al. (2022) for a broader overview on CD in graphs and hypergraphs.

One family of CD algorithms uses the number of expected communities c as an input parameter for the underlying model. This includes all Stochastic Block Model (SBM) variants, e. g., DCSBM (Karrer and Newman 2011). Due to receiving c as input, algorithms from this family generally reproduce known ground-truth clustering fairly accurately Peixoto (2014); Ruggeri et al. (2023); Zhang and Peixoto (2020).

In many applications, however, networks lack a known ground truth, creating the need for inferring this property. As a result, there has been a lot of research interest in CD algorithms that infer c , with agglomerative clustering as a major subcategory. One well-studied CD algorithm from this family is the LOUVAIN algorithm (Blondel et al. 2008), which optimizes for modularity (Newman 2006) via greedy local switches of community assignments. In a recursive alternation between greedy refinement and aggregation, LOUVAIN builds a hierarchy of successively smaller graphs. While this approach has the advantage to infer the number of communities automatically, it has the drawback of creating undesirably sparse or even disconnected communities. Over time, several variations and improvements have been proposed (Bonald et al. 2018), with LEIDEN (Traag et al. 2011) being a primary example. LEIDEN maintains a certain level of connectedness by enforcing a minimum level of internal edges in a community.

Concerning hypergraphs, many works for agglomerative algorithms focus on mapping the hypergraph to a reweighted dyadic graph representation. Kumar et al. (2020) define an iterative hyperedge reweighting scheme, in which each execution of default LOUVAIN is applied to a normalized clique expansion of the original hypergraph. In addition to the loss in structural information due to the conversion, the algorithm is limited to smaller instances due to the recurrent creation of clique expansions and their quadratic space consumption. For a cut-based algorithm, Contreras-Aso et al. (2023) convert the hypergraph to a so-called *derivative graph* by a similarity measure that compares the neighborhoods of two different nodes. Since structurally identical nodes (w. r. t. the similarity measure) are merged, the graph derivative loses information about the inclusion hierarchy of hyperedges completely, though.

Recent works on agglomerative CD algorithms focus on working directly on the hypergraph. To this end, Kamiński et al. (2019) proposed a generalization of graph modularity based on the Chung-Lu model. In their starting point of this line of work, hyperedges only contribute to the objective function if they are community-homogeneous, i. e., part of only one community (see strict hypermodularity in Sect. 2.1). Chodrow et al. (2021) use a maximum likelihood method to derive an algebraic objective function called All-or-Nothing Maximum Likelihood Louvain Method. Despite a different methodology, it corresponds to (strict) hypermodularity. By optimizing regarding community-homogeneous hyperedges, finer substructures in communities are missed, though; hence, follow-up works (Kamiński et al. 2022) favor more heterogeneous hyperedge contributions, as long as at least $\lfloor \text{vol}(E)/2 \rfloor$ nodes are part of one community. Benati et al. (2025), for example, introduce the γ -modularity function for hypergraphs, with $\gamma \in [0, 1]$ controlling the proportion of hyperedge homogeneity needed for a modularity contribution. They show for a small set of generated instances ($|V| \leq 100, |\mathcal{E}| \leq 200$) and $\gamma = 0.5$ that γ -modularity outperforms hypermodularity in accuracy due to the detection of finer community structures. It is doubtful, however, that these results can be generalized to larger hypergraphs: γ -modularity uses a hypergeometric distribution as null model instead of a binomial distribution for hypermodularity. As a result, the binomial coefficients scale with the size of the communities, making γ -modularity only suitable for small hypergraphs.

Working directly on the hypergraph with hypermodularity thus seems most promising for instances of non-modest size, especially after Kamiński et al. (2024) recently addressed a structural problem of the so far proposed LOUVAIN-like approaches for hypergraphs, leading to the formulation of H-LOUVAIN. Since our proposed algorithms reuse concepts of H-LOUVAIN and we use it as baseline in our experiments, we explain its approach and limitations in more detail in Sect. 2.3.

So far, to the best of our knowledge, publications on agglomerative CD-algorithms for hypergraphs share one

additional shortcoming: the lack of a diverse set of networks for comparison. In general, only a small set of hypergraphs is used and many works focus only on quite small networks ($|V| \leq 1k$) – often generated with the synthetic h-ABCD model (Kamiński et al. (2021, 2023), also see Sect. 4.1). This leaves a lot of structural complexity untouched (Battiston et al. 2020; Dai and Gao 2023; Lee et al. 2024) and might lead to wrong conclusions concerning the algorithms' accuracy. For our experiments in Sect. 4, we evaluate several similar algorithms with a wider set of both generated and real-world hypergraphs. We also propose to use an additional random model for instance generation, because algorithms may perform differently on h-ABCD hypergraphs than on other generated hypergraphs and, more importantly, than on real-world ones.

2.3 The H-LOUVAIN algorithm

As a multilevel algorithm, H-LOUVAIN (Kamiński et al. 2024) operates (recursively or iteratively) in rounds, where each round consists of a local optimization/refinement phase (GREEDYMOVE) and an aggregation phase (AGGREGATE). GREEDYMOVE starts with a singleton clustering (every node forms its own community). Then, each node v is visited in arbitrary order; when doing so, v can change its community membership to one of its neighbor communities that yield the largest (strictly positive) gain in terms of hypermodularity. The process of visiting nodes continues until there are no more local gains possible. In the subsequent AGGREGATE phase, one contracts communities into so-called supernodes. Moreover, for each community C , each hyperedge E , and each node $v \in E$, one sums the node-edge weights $\omega_n(E, v)$ and assigns the resulting value as new node-edge weight to the supernode that represents C in the coarser graph. Each hyperedge is then either an intra- or inter-edge, depending on whether it contains aggregated nodes from one or more communities. An example for a round of H-LOUVAIN is shown in Fig. 2. This alternation stops when GREEDYMOVE finds no improvement at all. Then, the current solution is prolonged to the input hypergraph.

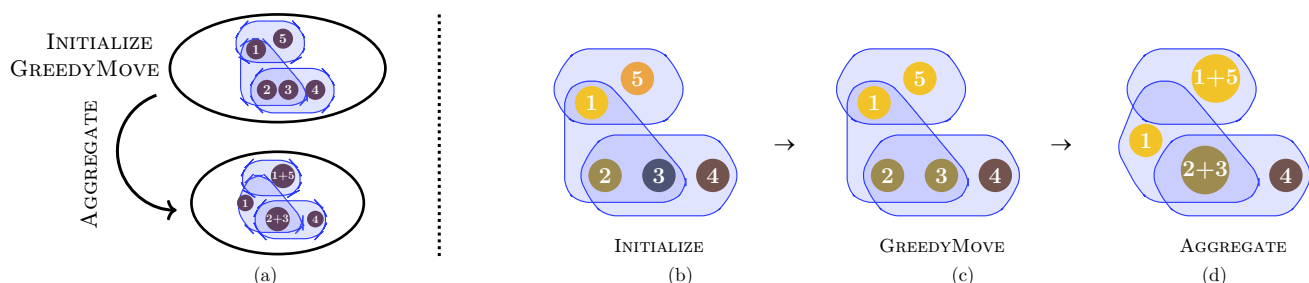


Fig. 2 (a) Illustration of one H-LOUVAIN round. (b): After INITIALIZE – each node is assigned its own community (singleton clustering). (c): After GREEDYMOVE – community assignments have changed due to

greedy local switches. (d): After AGGREGATE – nodes are combined to supernodes in their respective hyperedges. The aggregation preserves the distribution of communities in the hyperedges

Mixed-Modularity Optimization The design of H-LOUVAIN is based on the LOUVAIN algorithm for graphs (Blondel et al. 2008), but some algorithmic steps are adjusted to the more complex topology of hypergraphs. One adaptation happens in the AGGREGATE phase, where node-edge weights are only necessary on hypergraphs to conserve mathematical properties of hypermodularity between rounds. Another crucial change concerns the starting solution of the GREEDY-MOVE phase: For dyadic graphs, the algorithm is able to make progress (unless the singleton clustering is already a (locally) optimal solution). However, for hypergraphs, Eq. (2) and $e_H^{c,d}(C)$ are only evaluated for $c > d/2$. Gains for q_H can therefore only occur if the majority of the nodes from a hyperedge E share the same community. That is why the algorithm can get stuck after the initialization if the volume of all hyperedges is at least four:

Observation 1 (Kamiński et al. 2024) *Given a hypergraph H , a singleton clustering $\mathcal{C}_S$, and hypermodularity score $q_H(\mathcal{C}_S)$ for this clustering. If $\forall v \in V, E \in \mathcal{E} : w_n(E, v) = 1$ and $\forall E \in \mathcal{E} : \text{vol}(E) \geq 4$, GREEDYMOVE does not change $q_H(\mathcal{C}_S)$. As a result, H-LOUVAIN returns the hypermodularity score of a singleton clustering.*

To mitigate this problem, H-LOUVAIN proposes a mixed-modularity term

$$q = \alpha \cdot q_G(\mathcal{C}) + (1 - \alpha) \cdot q_H(\mathcal{C}), \quad (3)$$

using both q_H and graph modularity q_G directly on the hypergraph topology. To this end, q_G assumes pairwise interactions between the nodes of a hyperedge (clique expansion). By using the mixed-modularity term from Eq. (3), optimization is done for both the hypergraph and its clique expansion. While being a natural choice, clique expansion is not bijective (Wang and Kleinberg 2024); therefore, optimization could be done w. r. t. a misleading network topology. In addition, clique expansion assumes an underlying dyadic interaction within a hyperedge, which

is not necessarily the case for real-world networks (Battiston et al. 2020).

Parameter Tuning By default, H-LOUVAIN initializes the mixing parameter with $\alpha := 1$, performing greedy modularity q_G in the first round only. After that, α may be decreased for the next multilevel round (= hierarchy level) – depending on various conditions. This decrease is not a linear function in general and its progress hard to predict. In particular, it is not guaranteed that $\alpha = 0$ is reached in the end. H-LOUVAIN supports different design parameters tied to α , controlling when a change should take place. Overall, the mixing parameter α introduces another rather complex design parameter of the algorithm, with no guidance in the model itself. Kamiński et al. (2024) solve this by a Bayesian optimization technique of the parameter. This leads, however, to a significant increase in running time due to the optimization process. For example, a hypergraph with 193 nodes and 434 hyperedges takes nearly an hour in our tests. Hence, this tuning process limits the size of the hypergraphs that can be processed by H-LOUVAIN in reasonable time. One could see the fact that the instances in Kamiński et al. (2024) have only around 1k nodes/hyperedges as further evidence for our observation.

3 Proposed algorithms

To lift the described shortcomings of mixed-modularity optimization and parameter tuning, we propose new starting solutions and optimization strategies for H-LOUVAIN-like methods. Our approach alleviates the need for time-consuming parameter tuning. The first two proposed strategies are called F-LOUVAIN (Sec. 3.1) and P-LOUVAIN (Sec. 3.2). In addition, we propose to exploit a wider search space in a post-processing step, inspired by the Leiden algorithm for graphs (Traag et al. 2019). We call this variant H-LEIDEN (Sec. 3.3). It should be noted that both P-LOUVAIN and H-LEIDEN need some parameter tuning, but it is rather simple.

```

1: function LOUVAINLIKE( $H$ )
2:   repeat
3:      $\mathcal{C} \leftarrow \text{INITIALIZE}(H)$  ▷ Phase I: Compute and assign starting solution
4:      $\mathcal{C} \leftarrow \text{GREEDYMOVE}(H, \mathcal{C}, \alpha)$  ▷ Phase II: Move nodes to adj. communities
5:      $\text{done} \leftarrow |\mathcal{C}| = |V(H)|$ 
6:     if not done then
7:        $\mathcal{C} \leftarrow \text{REFINE}(\mathcal{C})$  ▷ Phase III: Refinement after local opt.
8:        $H \leftarrow \text{AGGREGATE}(H, \mathcal{C})$  ▷ Phase IV: Coarsen hypergraph
9:     end if
10:   until done
11: return  $\text{flat}^*(\mathcal{C})$  ▷ Solution is prolonged to the original input hypergraph
12: end function

```

Algorithm 1 General multilevel scheme for LOUVAIN-like algorithms for hypergraphs

Generic Framework Our proposed variants introduce INITIALIZE and REFINES, two new phases in the multilevel algorithm, in addition to GREEDYMOVE and AGGREGATE. In the following, we refer to these phases as Phases I-IV. The resulting generic framework of LOUVAIN-like algorithms is shown in Pseudocode 1. Like H-LOUVAIN, which only consists of Phases II+IV, variants can skip phases; this may also depend on the multilevel round. Note that GREEDYMOVE (Phase II) refers to greedy local community changes as described in Sect. 2.3. It uses a variable α for mixed-modularity optimization, adapted in each round. To simplify the matter, we only use two special cases with $\alpha := 1$ (q_G only) and $\alpha := 0$ (q_H only), in the following referred to as HYMODMOVE and MODMOVE, respectively. By the design of H-LOUVAIN, α is chosen to be 1 in the first round; hence, pure graph modularity optimization is done on the uncompressed hypergraph. Assuming the optimal solution is not a singleton clustering, this process is guaranteed to make progress towards maximizing q_G . As a result, the clustering after the first round provides more potential for refinement in subsequent rounds. After the main loop terminates, $flat^*(\mathcal{C})$ prolongs the solution from the coarsest level to the finest

(input) hypergraph. The necessary data structures for this (describing the relation between nodes on a fine level and supernodes on the neighboring coarse level) are retained for this purpose but not shown in the pseudocode.

3.1 The F-LOUVAIN algorithm

For dyadic graphs, modularity maximization is known to compress the original graph structure already very effectively in the first round. While so far it is unclear whether this property can be transferred to q_H and hypergraphs, we start by assuming it does – at least to a certain extent:

Assumption 1 By setting $\alpha := 1$ and thus applying q_G on the hypergraph in the first round, at least some hyperedges are compressed significantly in Phase IV. This avoids in subsequent rounds the situation described in Observation 1 where no progress is made in Phase II.

This assumption leads us to a simple variant of H-LOUVAIN called F-LOUVAIN, shown as Algorithm 2.

```

1: function FASTLOUVAIN( $H$ )
2:    $\mathcal{C}_G \leftarrow \text{INITIALIZE}(H)$  ▷ Round I: Phase I + Phase IV
3:    $H \leftarrow \text{AGGREGATE}(H, \mathcal{C}_G)$ 
4:   repeat ▷ Subsequent rounds: Phase II + Phase IV
5:      $\mathcal{C}_S \leftarrow \text{SINGLETONCLUSTERING}(V_H)$ 
6:      $\mathcal{C} \leftarrow \text{HYMODMOVE}(H, \mathcal{C}_S)$  ▷ Phase II: Local change based on  $q_H$ 
7:     done  $\leftarrow |\mathcal{C}| = |V(H)|$ 
8:     if not done then
9:        $H \leftarrow \text{AGGREGATE}(H, \mathcal{C})$ 
10:    end if
11:  until not done
12:  return  $flat^*(\mathcal{C})$  ▷ Solution is prolonged to the original input hypergraph
13: end function
14:
15: function INITIALIZE( $H$ )
16:    $\mathcal{C}_S \leftarrow \text{SINGLETONCLUSTERING}(V_H)$ 
17:    $\mathcal{C}_G \leftarrow \text{MODMOVE}(H, \mathcal{C}_S)$  ▷ Local change based on  $q_G$ 
18:  return  $\mathcal{C}_G$ 
19: end function

```

Algorithm 2 Multilevel scheme for F-LOUVAIN algorithm.

Its first round in the multilevel scheme consists only of Phase I (line 2) and Phase IV (line 3). In all consecutive rounds, F-LOUVAIN performs only local optimization (w. r. t. hypermodularity) and aggregation (Phase II and Phase IV). This is equivalent to setting $\alpha := 1$ in the first round and $\alpha := 0$ for all others in H-LOUVAIN. The rationale behind this simple heuristic is to remove parameter dependency completely, while still offering appropriate quality. By optimizing w. r. t. q_G only in the first round (and not thereafter), running time can be saved, while the potentially mediocre intermediate solution can usually be corrected in subsequent rounds.

3.2 The plouvain algorithm

While F-LOUVAIN can give good results for rather simple networks, its simplicity may yield unsatisfactory starting solutions for more complex topologies. Therefore, instead of relying on this simple heuristic as starting solution, the second proposed H-LOUVAIN variant uses an approach that is potentially closer to the unknown optimal clustering $\mathcal{C}_{opt}$. An interesting candidate for such a starting solution is a hypergraph k -partition $\mathcal{P}_k := \{P_1, \dots, P_k\}$. Formally, it is a clustering with $|P_i| = |P_j|$ for every pair of communities as an additional constraint. To this end, numerous heuristic algorithms have been proposed (Schlag 2020), some of which use clustering as their starting solution. Many compute a partition with a small imbalance factor δ . Reversing the order of clustering before partitioning, we use a (k, δ) -partition as the starting solution for clustering instead. The algorithmic design is based on the following assumption:

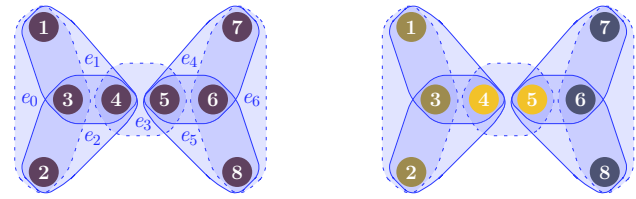


Fig. 3 **Left:** Hypergraph consisting of two dense connected four-node substructures: $V_1 = \{1, 2, 3, 4\}$ and $V_2 = \{5, 6, 7, 8\}$. V_1 and V_2 are connected by a dyadic edge e_3 . e_1, e_2 and e_4, e_5 are intra-edges (solid line) and have a weight $w_{intra} = 1.5$. The other edges are inter-edges (dashed line) and have weight $w_{inter} = 1.0$. All nodes have a node-edge weight of 1. **Right:** Optimal non-overlapping node clustering $\mathcal{C}_{opt} = \{\{1, 2, 3\}, \{4, 5\}, \{6, 7, 8\}\}$ based on hypermodularity. The coloring of the nodes corresponds to their assigned community. By doing greedy hypermodularity maximization, H-LOUVAIN computes the non-optimal community $\mathcal{C}_{lou} = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\}$, misled by $w_{intra} > w_{inter}$

Assumption 2 Let a hypergraph $H = (V, \mathcal{E})$ with $n = |V|$ and a clustering $\mathcal{C}_{hl}$ obtained from H-LOUVAIN be given. If $|\mathcal{C}_{hl}| < n$ and $k \ll n$, then using a k -partition instead of a singleton clustering as the starting solution in each round enables progress in Phase II for all possible values of α . This is particularly true for hypermodularity maximization with $\alpha := 0$.

In other words, a k -partition with large enough communities is likely to create hyperedges with heterogeneous community assignments. Greedy changes in this assignment are likely to create positive changes in $q_H(\mathcal{C})$. We refer to the overall strategy sketched above as P-LOUVAIN in the following. Its pseudocode is shown in Algorithm 3.

```

1: function P-LOUVAIN( $H, k, \delta$ )
2:   repeat
3:      $\mathcal{C} \leftarrow \text{SINGLETONCLUSTERING}(V_H)$  ▷ Create singleton clustering
4:     if  $|\mathcal{C}| > k$  then
5:        $\mathcal{C} \leftarrow \text{PARTITIONING}(\mathcal{C}, k, \delta)$  ▷ Create Partition
6:     end if ▷ Phase I: Use either singleton clustering or partition
7:      $\mathcal{C} \leftarrow \text{HYMODMOVE}(H, \mathcal{C})$  ▷ Phase II: Local change based on  $q_H$ 
8:      $\text{done} \leftarrow |\mathcal{C}| = |V(H)|$ 
9:     if not done then
10:       $H \leftarrow \text{AGGREGATE}(H, \mathcal{C})$  ▷ Phase IV: Coarsen hypergraph
11:    end if
12:  until done
13:  return  $\text{flat}^*(\mathcal{C})$  ▷ Solution is prolonged to the original input hypergraph
14: end function

```

Algorithm 3 Multilevel scheme for P-LOUVAIN algorithm.

P-LOUVAIN applies a starting solution at the beginning of each round (lines 3-7) and has no Phase III. Depending on two inequalities, the starting solution is either a standard singleton clustering $\mathcal{C}$ (line 3) or a clustering $\mathcal{C}_P$ obtained by k -way partitioning (line 7). Using $\mathcal{C}_P$ only if $|\mathcal{C}| > k$ is supposed mitigate effects from overfitting due to too small values of k . During Phase II, local community changes are done based on hypermodularity only, removing the need of setting a particular value for α . This is based on the underlying assumption that a partition already enables at least one positive change for q_H . Finally, Phase IV is the aggregation known from H-LOUVAIN.

While P-LOUVAIN introduces additional parameters (δ, k) for the hypermodularity optimization, our experiments indicate that a simple heuristic suffices to enable good guesses independent of the input hypergraph (cf. Sect. 4.2). Also, the overhead of computing a partition in each round of the multilevel algorithm is mitigated by the fact that a good starting solution reduces the number of steps in Phase II.

3.3 The H-LEIDEN algorithm

Let $\Delta q_v(C_i \rightarrow C_j)$ denote the change in any our modularity variants whenever a node v switches from community C_i to C_j . Originally, GREEDYMOVE in LOUVAIN-like algorithms only allows moves to neighboring communities of v which are positive and maximize the change in modularity ($\Delta q_v^{max} > 0$). Independently from the value of α , H-LOUVAIN is no exception in this respect. Hence, community changes during GREEDYMOVE are also based on maximal gains. As shown by Traag et al. (2019), for graphs it is often not possible to find the optimal solution for q_G using Δq_v^{max} gains – even for very small networks. The same holds for mixed-modularity q from Eq. (3) and hence all previously discussed variants of H-LOUVAIN, as the example of the simple weighted hypergraph in Fig. 3 shows. The optimal solution concerning q_H consists here of three communities, whereas mixed-modularity optimization q in H-LOUVAIN detects four, misled by higher edge weights for edges e_1, e_2, e_4 , and e_5 . This suboptimal result is obtained even with BOT in H-LOUVAIN.

Inspired by the potential advantage of performing strictly positive (but not necessarily locally optimal) moves known from dyadic graphs, we adopt this strategy here as well. Even if the benefit may not be equally high due to higher-order interactions, this approach still provides access to a wider search space, leading to the assumption:

Assumption 3 Let a hypergraph $H = (V, \mathcal{E})$ and any $v \in V$ be given. By allowing strictly positive changes $\Delta q_v(C_i \rightarrow C_j)$ to neighboring communities $C_j \in N_C(v)$, a LOUVAIN-like

algorithm is (often) able to obtain a higher hypermodularity value than with using maximum greedy gains Δq_v^{max} only.

Motivated by 3 and 1, we propose an additional refinement step for communities already found (Phase III). The design is inspired by the LEIDEN algorithm (Traag et al. 2019) for dyadic graphs, which introduces a post-processing step (see Sect. 2.2). This step potentially splits communities formed during Phase II into subcommunities based on a threshold metric cut_G (see Eq. (4) below) and strictly positive moves. By doing so, LEIDEN also increases the resolution of the inferred communities, i. e., finer substructures can be detected. Since we generalize both the concept of strictly positive moves and the threshold metric to hypergraphs, we assume the same resolution-enhancing property for hypergraphs:

Assumption 4 Let a clustering $\mathcal{C}_P$ computed by P-LOUVAIN and a community C_i obtained by any iteration of Phase II be given. If C_i is not a singleton clustering, it is possible to find finer sub-communities using an appropriate refinement procedure (Phase III). Furthermore, the clustering $\mathcal{C}_{SR}$ obtained by this strategy has a comparable hypermodularity score to $\mathcal{C}_P$, i. e., $q_H(\mathcal{C}_{SR}) \approx q_H(\mathcal{C}_P)$.

Following Assumptions 3 and 4, we define our third proposed variant H-LEIDEN as an extension of P-LOUVAIN. H-LEIDEN applies all four phases I-IV in each multilevel round. While the other phases remain unchanged, Phase III contains the refinement procedure. For this, each formed community is split into subsets $C_i \rightarrow \mathcal{C}_i$ such that $\mathcal{C}_i$ is a set of sets of nodes from C_i . Analogous to the refinement step in LEIDEN for graphs, we allow only subsets S of $\mathcal{C}_i$ to be merged when they exceed a connectivity threshold cut_H (Eq. (5) below) w. r. t. the rest of the community $\mathcal{C}_i$.

On graphs, Traag et al. (2011) define the cut-threshold cut_G as the number of edges between any non-empty subset S for a community C . This threshold is then compared to the expected number of edges as given by the sum of node degrees:

$$cut_G(S, C) = \{(s, v) \mid s \in S, v \in C, (s, v) \in E\} \geq \gamma \cdot vol(S) \cdot (vol(C) - vol(S)). \quad (4)$$

By splitting a community based on this threshold, only “well-connected” communities remain intact, with γ controlling the strictness of the metric. For hypergraphs, there is no straightforward conversion of cut_G due to the edge counting rules in hypermodularity (see Eq. (1)) and the combinatorial explosion of the expected number of hyperedges. Instead, we define cut_H as the positive impact on q_H if a non-empty set S is combined with a community C . By construction, it holds that $S \subset C$, therefore cut_H counts the

edge contribution of C and compares it with the (reduced) contribution if S is removed from C :

$$cut_H(S, C) = \sum_{d \geq 2} \sum_{c=[d/2+1]}^d (e_H^{c,d}(C) - e_H^{c,d}(C \setminus S)). \quad (5)$$

The corresponding threshold for community splitting in hypergraphs involves the change in degree tax t_d , in the following referred to as $T_d(C, S)$:

$$cut_H(S, C) \geq \sum_{d \geq 2} \sum_{c=[d/2+1]}^d \gamma \cdot (t_d(C) - t_d(C \setminus S)) =: T_d(C, S). \quad (6)$$

Note that the threshold definitions for both graphs and hypergraphs share the omission of self-loops. The reason is the focus on the “connectedness” of community subsets with the rest of the community.

The pseudocode, including the proposed addition of Phase III, is shown in Algorithm 4. Function HYPERLEIDEN contains the main loop, which iterates through Phases I-IV (lines 3-11) until there are no more changes in the clustering. Since H-LEIDEN is an extension of P-LOUVAIN, Phases I, II and IV are used as before; the only change is the addition of Phase III (REFINE). The initial splitting is done based on the starting solution $\mathcal{C}_I$ of the current hierarchy level (line 19), i. e., either a k -way partition or singleton clustering, depending on the number of nodes on the coarsening level. That is why $\mathcal{C}_I$ is fixed throughout GREEDYMOVE and handed to the REFINE function as parameter (line 10). Refinement is then done only for those subsets that fulfill the threshold in Eq. (6) (line 20). For all eligible subsets R , a merge with another subset S is chosen randomly; the probability is proportionate to Δq_H of the merge if it leads to a positive gain: $\Delta q_H(C_R, C_S \rightarrow C_{R \cup S}) > 0$ (line 23).

```

1: function HYPERLEIDEN( $H, k, \delta$ )
2:   repeat
3:      $\mathcal{C}, \mathcal{C}_I \leftarrow \text{SINGLETONCLUSTERING}(V_H)$  ▷ Create singleton clustering
4:     if  $|\mathcal{C}| > k$  then
5:        $\mathcal{C}, \mathcal{C}_I \leftarrow \text{PARTITIONING}(\mathcal{C}, k, \delta)$  ▷ Create partition
6:     end if ▷ Phase I: Use either singleton clustering or partition
7:      $\mathcal{C} \leftarrow \text{HYMODMOVE}(H, \mathcal{C})$  ▷ Phase II: Local change based on  $q_H$ 
8:      $\text{done} \leftarrow |\mathcal{C}| = |V(H)|$ 
9:     if not done then
10:       $\mathcal{C} \leftarrow \text{REFINE}(\mathcal{C}, \mathcal{C}_I)$  ▷ Phase III: Refine communities
11:       $H \leftarrow \text{AGGREGATE}(H, \mathcal{C})$  ▷ Phase IV: Coarsen hypergraph
12:    end if
13:  until done
14:  return  $\text{flat}^*(\mathcal{C})$  ▷ Solution is prolonged to the original input hypergraph
15: end function
16:
17: function REFINE( $\mathcal{C}, \mathcal{C}_I$ )
18:   for  $C \in \mathcal{C}$  do
19:      $\mathcal{C}_R \leftarrow \{T \mid \forall C_I \in \mathcal{C}_I : T = (C \cap C_I)\}$ 
20:      $\mathcal{R} := \{S \mid S \in \mathcal{C}_R, cut_H(S, C) \geq T_d(C, S)\}$ 
21:     for  $R \in \mathcal{R}$  do
22:        $\mathcal{T} \leftarrow \{S \mid S \in \mathcal{R}, cut_H(S, C) \geq D_t(C, S)\} \setminus R$ 
23:        $S_M \leftarrow \text{Sample} \left( S \in \mathcal{T}, \begin{cases} \exp(\frac{1}{\theta} \Delta q_H(C_R, C_S \rightarrow C_{R \cup S})) & \text{if } \Delta q_H \geq 0 \\ 0 & \text{else} \end{cases} \right)$ 
24:        $R \leftarrow R \cup S_M, \mathcal{R} \leftarrow \mathcal{R} \setminus S_M$ 
25:     end for
26:      $C \leftarrow C_R$ 
27:   end for
28: end function

```

Algorithm 4 Multilevel scheme for H-LEIDEN algorithm

Table 1 Datasets used for comparison of clustering quality (ARMI, q_H)

Network	nodes	edges	$\deg(v)$	$ \bar{e} $	$\max(e)$	2-edge	3-edge	3σ	$ g $	type
h-abcd (8x)	1000	1576-3374	9-29	3-6	9	0.08–0.56	0.09–0.21	0	7-10	cl
h-abcd (8x)	10000	7769-29660	9-26	3-11	19	0.02–0.54	0.02–0.2	0	7-11	cl
h-abcd (8x)	50000	38723-146830	11-19	3-11	19	0.02–0.54	0.02–0.2	0	8-12	cl
bakery-basket	81	1436	26	3	10	0.25	0.38	0.57	81	recommendation
bar-reviews	1234	1188	16	6	73	0.19	0.16	0	15	contact
bar-reviews-swapped	15	222	10	1	6	0.34	0.05	0.45	15	swapped
bills-house	1494	54933	16	11	399	0.14	0.08	0.02	4	agenda
bills-senate	294	21721	10	6	99	0.15	0.14	0.12	9	agenda
committees-house	1290	336	10	26	82	0.0	0.01	0	2	politics
committees-senate	282	302	15	16	31	0.0	0.0	0	2	politics
cooking	6714	39674	20	10	65	0.0	0.01	0	20	geography
cooking-swapped	20	3818	15	5	20	0.15	0.07	0.76	20	swapped
dawn	2109	86968	10	4	22	0.24	0.29	0.44	10	medicine
dawn-swapped	196	1277	16	5	119	0.11	0.02	0.31	196	swapped
enron-email	4423	5734	10	4	25	0.42	0.16	0.05	2	communication
eurobarometer	13	1032	16	4	4	0.08	0.28	1	13	category
gene-disease	9262	3128	11	6	1074	0.28	0.13	0	25	biology
high-school	327	7818	10	2	5	0.7	0.27	0.86	9	contact
hospital-lyon	75	1824	16	2	5	0.61	0.36	0.95	4	contact
hyperlap (4x)	1000	1568-3283	10-27	3-6	9	0.08–0.55	0.09–0.21	0	2-10	sbm
hyperlap (4x)	10000	7769-28978	8-17	3-11	19	0.02–0.53	0.02–0.2	0	2-10	sbm
hyperlap (4x)	50000	38719-144925	10-29	3-11	19	0.02–0.53	0.02–0.2	0	2-10	sbm
music-reviews	1106	686	18	10	83	0.11	0.08	0	7	recommendation
music-reviews-swapped	7	158	18	1	4	0.37	0.01	0.5	7	swapped
primary-school	242	12704	10	2	5	0.61	0.36	0.89	11	contact
restaurant-reviews	565	594	10	5	43	0.24	0.16	0	9	contact
restaurant-reviews-swapped	9	164	18	2	5	0.39	0.05	0.75	9	swapped
spanish-barometer	61	2199	10	3	3	0.01	0.99	0	61	category
united-nations	193	434	10	45	186	0.01	0.02	0	193	politics
walmart	88836	65898	8	5	25	0.19	0.15	0	50	recommendation
walmart-swapped	60	25292	13	5	41	0.08	0.09	0.93	60	swapped
workspace1	92	788	15	2	4	0.94	0.06	0.77	5	contact
zoo-animals	101	41	26	27	93	0.02	0	0	101	biology

3.4 Time complexity

It is quite challenging to derive tight worst-case bounds for the algorithmic framework in general, also on graphs. The reason is that the number of iterations of the repeat-loop in Algorithm 1 and of GREEDYMOVE are very difficult to analyze in the general case. What we can state, though, is that each iteration in GREEDYMOVE takes

$$\mathcal{O}\left(|E_{\max}|^2 \cdot \sum_{E \in \mathcal{E}} |E|^2\right) \quad (7)$$

time (where E , $E_{\max}$, and $\mathcal{E}$ relate to the hyperedges of the current hierarchy level): in each iteration, all nodes visit all their neighbors for potential community swaps. For

graphs, this loop and search process is linear in the number of edges. For hypergraphs, however, we get a quadratic term for each hyperedge since each vertex pair is involved, hence $\mathcal{O}(\sum_{E \in \mathcal{E}} |E|^2)$. For each node and candidate community, the hypermodularity change has to be computed (see Eq. (1)), which takes $\mathcal{O}(|E_{\max}|^2)$ time (Kamiński et al. 2024). P-LOUVAIN and H-LEIDEN add an additional call to a partitioner. State-of-the-art partitioners take $\mathcal{O}(|V|)$ time in practice (Gottesbüren et al. 2020). H-LEIDEN also adds a refinement phase, which is similar to GREEDYMOVE, but confined to movements in a single community. Hence, GREEDYMOVE is the bottleneck operation on each hierarchy level; it dominates the overall running time (assuming that the hypergraphs shrink from level to level sufficiently) with Eq. (7).

4 Experiments

The implementation of the proposed algorithms is done in Python, using the original code of H-LOUVAIN as a foundation. Code, reproducibility instructions, datasets, and results are available on Github¹. All experiments for the comparison (quality, running time) are executed on a machine with a dual-socket Intel Xeon 6126 processor with 12 cores each (24 in total) and 192 GB RAM. Since all algorithms are sequential, quality and label grouping benchmarks are conducted with one core per process. Running time experiments are carried out with exclusive machine usage to avoid disturbances from other processes. To ensure reproducibility, the command-line tool SIMEXPAL (Angriman et al. 2019) is used to automate the experiments.

Each experiment is divided into runs. A run is a combination of an algorithm and a certain set of parameters. Each run is repeated 5 times for running time measurements and 50 times for quality measurements, the arithmetic mean values are reported. While in the following the most relevant aspects of the experiments are discussed in detail, an overview of all numerical results is shown in Table 2 in the appendix.

4.1 Instances

The datasets include a wide variety of real-world and randomly generated hypergraphs; an overview is shown in Table 1. All networks are sanitized such that multiple occurrences of the same node in a hyperedge are resolved by aggregating them into one node with summed weight. The same is done for multiple occurrences of identical hyperedges. We do not restrict the maximum size of hyperedges.

Real-World Datasets One subset of our real-world datasets is selected as is from the literature, such as networks used in Amburg et al. (2020); Benson et al. (2018); Benati et al. (2025) and hypergraphs from repositories such as XGI (Landry et al. 2023). They are chosen so that the resulting clusterings produce meaningful node groups—such as study classes and faculty members—in contact networks of schools (high schools and primary schools). This enables using both q_H -scores and ARMI-scores in our comparison, enabling a more informed evaluation than using one measure only. In addition to $\deg(v)$ and the mean edge volume $|e|$, we also report the percentage of dyadic edges, of edges with $|e| = 2, 3$ and of 3-simplices (3σ) for each network. Combinations of these properties are known to be

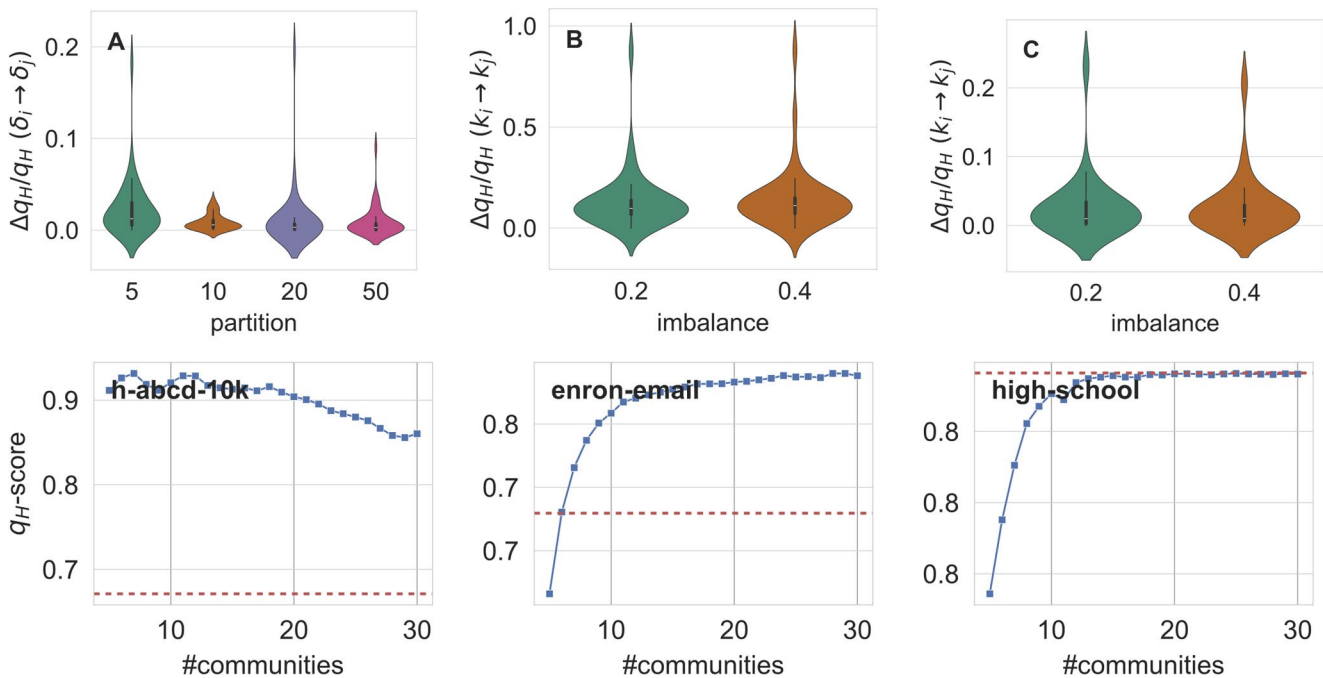


Fig. 4 Upper row: (A) Relative change in hypermodularity $\Delta q_H/q_H$ when switching $\delta : 0.2 \rightarrow 0.4$ for partition sizes $k \in \{5, 10, 20, 50\}$. (B) Relative change in hypermodularity $\Delta q_H/q_H$ between all $k \in \{5, 10, 20, 50\}$ when using different imbalance factors δ . (C) Same relative change as in (B), but k is restricted to $k \in \{10, 20\}$. Changes

are much less pronounced, indicating k should neither be too high nor too low. Lower row: Dependence of $q_H(c)$ on k for P-LOUVAIN for three example instances taken from the sets: *h-abcd* and *real*. For all networks $\delta = 0.2$ is used. A red horizontal line shows the q_H -score of the clustering obtained by H-LOUVAIN

¹ Github link: <https://github.com/KIT-EAE/parameter-light-hyperleiden>.

discriminative features for finding hypergraph classes (Batistoni et al. 2020; Benson et al. 2018).

In addition to the already mentioned hypergraphs, we create transformed versions of the *review* datasets (*bar*, *music*, *restaurant*, *cooking*, *dawn*, and *walmart*), which were originally intended for benchmarking overlapping CD algorithms (Amburg et al. 2020). For non-overlapping CD, we adapt the networks such that each node is assigned a community based on the most common edge label. In case the number of edge labels and the size of the original network facilitate plausible results, we additionally create a dual hypergraph (nodes and hyperedges are exchanged), marked with the suffix *swapped* in the overview table. By swapping nodes and hyperedges, edge labels from the *review* set become node labels. Analogous to (Benati et al. 2025), node labels can be used as ground truth after the switch. Using asymmetric scoring functions, we can then rate whether a computed clustering is a coarser representation of the given node labels.

Randomly Generated Datasets To create randomly generated networks, we use h-ABCD (Kamiński et al. 2021, 2023), a generalization of the LFR generator (Lancichinetti et al. 2008) to hypergraphs. The model uses different power-law exponents for node degree $\mathcal{D}_V$, edge volume $\mathcal{D}_E$, and community size distribution $\mathcal{D}_C$ in addition to a mixing parameter x and contribution parameter τ_c for hyperedges, which controls how communities are distributed between hyperedges. While τ_c only influences the generated ground truths, x is also connected to the hyperedge structure by varying the average node degree of the generated network. In general, higher values for x lead to disassociative community structures for the ground truths. Given that we treat $\mathcal{D}_V$ and $\mathcal{D}_C$ as constants, $\mathcal{D}_E \in \{-2, 2\}$, $x \in \{0, 0.5\}$, $\tau_c \in \{0, \infty\}$, there are eight combinations for each number of nodes n . For $n \in \{1k, 10k, 50k\}$, we create one hypergraph and ground truth clustering for each combination, hence 24 hypergraphs in total.

We also create a second set of randomly generated networks, now using the so-called Hyperlap model (Lee et al. 2021). Hyperlap uses an extension of the generative Chung-Lu model, creating hyperedges inside hierarchical node partitions, where higher hierarchical levels contain

finer partitions. Hyperedges are generated inside one randomly chosen hierarchical level. Hence, the probabilities for hyperedges to contain the same nodes vary by level. Lee et al. show that the local structure of real-world hypergraphs can be reproduced more accurately, compared to using the Chung-Lu model without hierarchies (like *h-abcd*). Hyperlap takes $\mathcal{D}_V$ and $\mathcal{D}_E$ as input and preserves them for the generated hypergraph. For consistency, we use $\mathcal{D}_V$ and $\mathcal{D}_E$ from our generated *h-abcd* instances. Hyperlap itself does not generate ground truths. Therefore, we use HyMMSBM (Ruggeri et al. 2023) to infer ground truths using different values for $k \in \{2, 4, 6, 8, 10\}$, assuming for each k either dissociative or associative community structures. For the systematic experiments, the clustering where H-LOUVAIN yields the highest ARMI score is used as ground truth for comparison.

4.2 P-LOUVAIN: Heuristic for choosing (k, δ)

To arrive at a well-working heuristic for choosing (k, δ) for P-LOUVAIN, we evaluate both parameters separately at first. To this end, we perform a parameter study with all networks and $\delta \in \{0.2, 0.4\}$, $k \in \{5, 10, 20, 50\}$. We report the relative change $\Delta q_H/q_H$ when one parameter is kept constant in Fig. 4 (upper row).

When we fix k and measure the relative change when varying δ (see Fig. 4A), the most obvious effect on clustering quality occurs when δ is changed from 0.2 to 0.4 for $k = 5$. Higher values of k are more robust against changes in δ . Due to the construction of P-LOUVAIN, this result is expected, since GREEDYMOVE does not allow nodes to divide communities further (= finer) than the starting solution. If k is chosen too small, q_H becomes smaller since the subtrahend of Eq. (2) becomes more dominant due to high values of $vol(\mathcal{C})$. When assessing the relative variance between min and max values of k (see Fig. 4B), there is no measurable difference for both values of δ . Overall, a change in k has a much higher impact than using different imbalance factors δ . When excluding $k = 5$ and $k = 50$ from the parameter study, the relative change in hypermodularity is much less pronounced (see Fig. 4C), suggesting that k should neither be chosen too high nor too low. Furthermore, when comparing

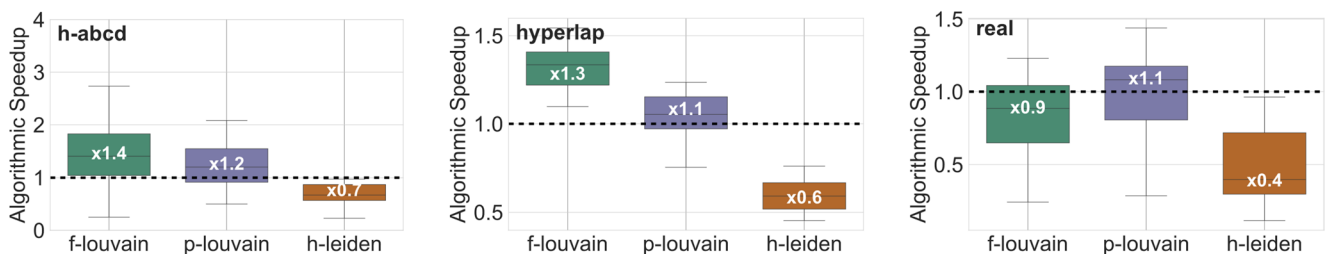


Fig. 5 Algorithmic speedup (higher is better) of our new algorithms compared to H-LOUVAIN. The results are divided into *h-abcd*, *hyperlap* and *real* instances. For all datasets, the parameters are chosen such that $\tau = 0$ and $(k, \delta) = (50, 0.2)$

the absolute values of q_H for all networks, one of the values for $k \in \{10, 20\}$ relates to the highest q_H -score for 30 out of 40 networks. For 10 out of 40 networks, however, either $k = 5$ or $k = 50$ is the best choice. Evaluating for $k \in [5, 30]$, we find plateaus with high q_H -scores as shown for three example networks in Fig. 4 (lower row). When k is low, q_H -scores start decreasing. Inferred from these results, we propose the following heuristic for choosing (k, δ) for P-LOUVAIN: Fix δ and use a binary search-like approach for finding the best k value: start with a high value for $k = k_s < |V|$ and repeat P-LOUVAIN with $k := \lfloor k/f \rfloor$ (where $f \in \mathbb{N}$ is a small number) until the q_H -score decreases. This leads to $\mathcal{O}(\log k)$ many runs.

Large values for k should be avoided due to hypergraph properties like those in Observation 1. Additional experiments using both generated and real-world instances indicate that choosing $k > 200$ for P-LOUVAIN leads to lower or at best comparable hypermodularity scores. As a result, the search normally does not exceed more than a few runs of P-LOUVAIN for one network. While this heuristic does not find the best k -value in general, a value that leads to a q_H -score from the high plateau is likely. Like BOT for H-LOUVAIN, finding k for P-LOUVAIN has only to be done once for each network, but the latter is much faster. In the following, when evaluating the solution quality for P-LOUVAIN, we use $k \in \{5, 10, 20, 50, 200\}$ for each dataset and select k with the highest score.

4.3 Algorithmic running time comparison

As part of the evaluation, we compare the running time of our algorithms to that of H-LOUVAIN (using default values for α in the latter). Since the basic framework for all algorithms is the same, we measure the time it takes for the core algorithm to complete one run, in the following referred to as algorithmic running time. Hence, algorithmic running time does not include the time for preparing the hypergraph in memory and initializing necessary helper data structures.

For completeness, we also report the slowdown of algorithmic running time imposed by the Bayesian Optimization Technique (BOT) used in H-LOUVAIN for a few selected instances. To this end, we use one network from *h-abcd* and *hyperlap*-set each as well as the two real-world networks (*united-nations*, *workspace1*). The network size ranges from $n = 92$ to $n = 1k$ nodes. As expected due to the increasing complexity of choosing α with larger size, the slowdown increases from 25× for the smallest instance to 80× for $n = 1k$. For instances with more nodes or hyperedges, the running time is in the order of hours; that is why we refrain from measuring BOT for other networks. For consistency across all networks, we report the algorithmic speedup of the variants relative to H-LOUVAIN without BOT. When running H-LOUVAIN without BOT, a strategy for α has to be chosen. Since the model has no guidance except the values provided by Kamiński et al. (2024), we use them as suggested there.

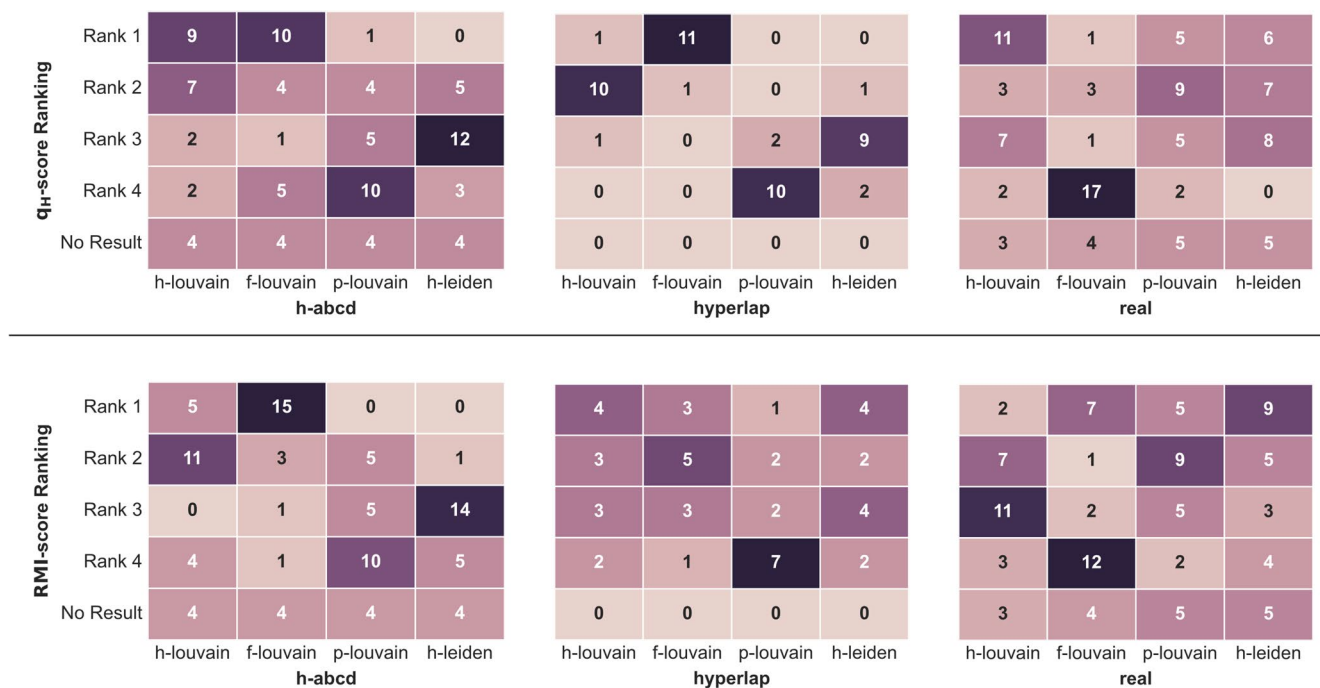


Fig. 6 Number of datasets where each algorithm achieved each rank (higher is better). Rankings are based on the highest q_H score (upper row), respective ARMI score (lower row) achieved by each algorithm. Columns: *h-abcd*, *hyperlap*, and *real-world* instance groups.

τ is chosen for all algorithms independently for each instance such that it represents the highest q_H -score for H-LOUVAIN. (k, δ) is chosen such that $q_H(\mathcal{C})$ of P-LOUVAIN/H-LEIDEN is maximized based on $k \in \{5, 10, 20, 50, 200\}$

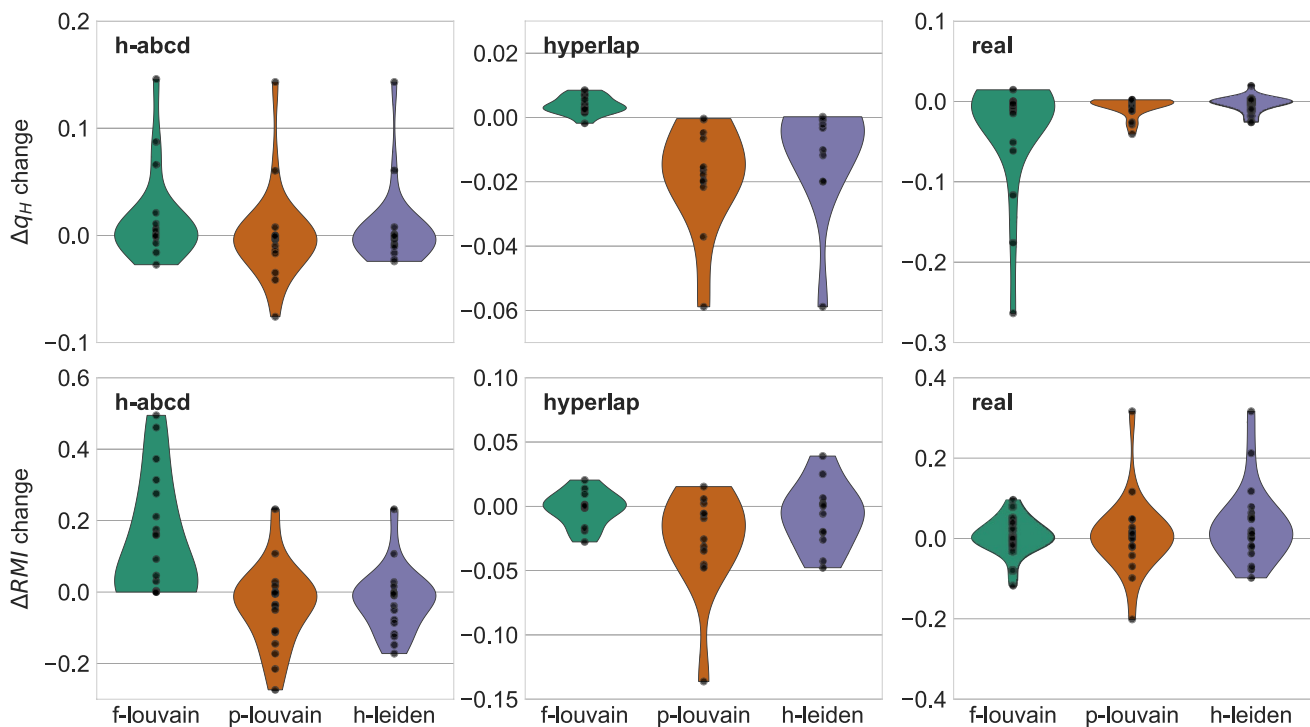


Fig. 7 Absolute change of the hypermodularity score Δq_H (upper row) respective the change in ARMI score ΔARMI (lower row) for F-LOUVAIN, P-LOUVAIN and H-LEIDEN compared to H-LOUVAIN for each dataset. The results are divided into *h-abcd*, *hyperlap* and *real-world*

Hence, a set of values $M_\alpha = \{0, 0.8, 0.92, 0.992, \dots\}$ is generated such that most of the values are near 1. The sequence index is incremented whenever the number of communities is halved, arriving at $\alpha = 1.0$ only for sufficiently large networks. The results for the algorithmic speedup are shown in Fig. 4, divided again into three sets based on the instance groups due to heterogeneous results (Fig. 5).

The only consistent speedup is achieved by P-LOUVAIN (1.2x). F-LOUVAIN achieves a speedup (1.4x) for generated networks only, while it shows a slowdown for real-world hypergraphs. For h-abcd networks, the speedup shows a higher variance for P-LOUVAIN. A closer inspection shows that the speedup increases both with network size n and mixing factor x , showing the largest improvements for $n = 50k, x = 1.0$ with 10x faster algorithmic running time. This is further evidence in favor of P-LOUVAIN. As expected, H-LEIDEN on average doubles the running time due to the additional refinement phase (Phase III); for a few networks (*commitees-house*, *zoo-animals*) the slowdown reaches 10x.

4.4 Solution quality

Methodology To evaluate the solution quality of the proposed algorithms in comparison to H-LOUVAIN, we use two different scores: hypermodularity q_H and (Asymmetric)

instances, whereas datasets without a q_H -score for H-LOUVAIN are left out. For all remaining datasets, τ is chosen based on the best q_H -score for H-LOUVAIN. (k, δ) is chosen, such that $q_H(\mathcal{C})$ of P-LOUVAIN/H-LEIDEN is maximized based on $k \in \{5, 10, 20, 50, 200\}$

Reduced Mutual Index (ARMI) (Jerdee et al. 2025; Cantwell and Young 2020).

Hypermodularity q_H scores a single network and ranges from -0.5 to 1 . Higher scores reflect a stronger community structure with communities of (roughly) similar sizes, see Eq. (1). All LOUVAIN-like algorithms are steered by q_H and compute a clustering that is locally (but typically not globally) optimal w. r. t. q_H . Since globally optimal scores are not feasible to compute, we compare the q_H values of all algorithms against each other (higher is better).

The Reduced Mutual Index (RMI) in the symmetric version is a progression from the well-known AMI score (Vinh et al. 2009); it removes a bias and provides a more accurate estimate. The asymmetric version of RMI is specifically designed to compare communities (or, even more generally, sets of sets) with a single ground truth (Felippe et al. 2024). ARMI scores range from $-\infty$ to 1 , where higher is better. In contrast to AMI, ARMI produces high scores not only if two communities are structurally similar, but also if the generated community is a coarser/finer representation of the given ground truth. ARMI scores for random community assignments are near 0. Even lower scores are obtained by candidate solutions which are close to a singleton clustering.

Due to its very high running time overhead, we do not use BOT in the following when evaluating the solution quality of H-LOUVAIN. The values for α are chosen as discussed in

Table 2 Overview of results (running time, quality) for all networks and Louvain-like algorithms. Results shown are mean values of multiple runs: 5 runs for running time measurements and 50 for quality measurements. Generated instances are listed grouped by the number of nodes. Instances marked with ‘-’ represent a failed execution due to exceeding the time limit of 20 h (for all 5 or 50 runs, respectively)

Network	Running Time (in seconds)				qH score				ARMI score			
	H-LOU	F-LOU	P-LOU	H-LEI	H-LOU	F-LOU	P-LOU	H-LEI	H-LOU	F-LOU	P-LOU	H-LEI
h-abc (8x, n=1k)	8.2	5.8	8.1	13.9	0.63	0.63	0.61	0.62	0.78	0.92	0.66	0.70
h-abc (8x, n=10k)	410.3	233.8	328.3	565.6	0.63	0.65	0.63	0.63	0.68	0.85	0.67	0.68
h-abc (8x, n=50k)	8441.8	10656.5	3332.5	6657.0	0.76	0.79	0.80	0.80	0.86	0.95	0.92	0.92
bakery-basket	2.5	2.8	2.2	6.1	0.06	0.06	0.07	0.07	0.68	0.66	0.71	0.73
bar-reviews	178.0	229.2	191.9	510.7	0.79	0.68	0.79	0.79	0.70	0.75	0.69	0.69
bills-senate	1977.1	4284.1	1773.0	2389.0	0.37	0.37	0.34	0.35	0.10	0.13	0.05	0.06
committees-house	172.3	231.3	206.2	995.4	0.59	0.58	0.58	0.61	0.00	0.00	0.00	0.00
committees-senate	19.5	29.9	7.0	12.0	0.48	0.48	-	-	0.00	0.00	-	-
cooking	2438.2	4536.7	3950.6	6131.9	0.43	0.43	0.43	0.43	0.19	0.21	0.17	0.17
cooking-swapped	3359.0	8454.0	3022.7	4220.2	0.36	0.35	0.36	0.36	0.72	0.77	0.74	0.78
dawn	764.7	1785.0	921.9	3076.9	0.33	0.33	0.33	0.33	0.01	0.01	0.01	0.03
dawn-swapped	-	-	11784.2	-	-	-	-	-	-	-	-	-
enron-email	71.4	70.4	112.3	215.7	0.73	0.72	0.73	0.73	-0.05	-0.03	-0.00	-0.00
eurobarometer	0.3	0.3	0.2	1.3	0.0	0.0	0.0	0.0	0.31	0.35	0.24	0.24
gene-disease	5617.5	23205.5	14211.2	-	0.69	-	-	-	0.01	-	-	-
high-school	2.2	1.8	1.6	2.3	0.82	0.82	0.82	0.82	0.89	0.89	0.89	0.89
hospital-lyon	2.0	2.1	1.8	3.3	0.38	0.37	0.38	0.38	0.14	0.13	0.15	0.15
hyperlap (4x, n=1k)	7.2	5.5	7.8	12.4	0.66	0.66	0.64	0.66	0.62	0.63	0.58	0.62
hyperlap (4x, n=10k)	202.9	141.1	201.4	402.0	0.75	0.75	0.74	0.74	0.56	0.56	0.54	0.55
hyperlap (4x, n=50k)	3411.3	2696.4	2755.4	5070.2	0.78	0.78	0.75	0.76	0.45	0.44	0.43	0.43
music-reviews	227.7	254.8	216.0	719.0	0.72	0.55	0.71	0.71	0.79	0.87	0.59	0.71
music-reviews-swapped	1.6	1.7	1.5	1.9	0.58	0.58	0.58	0.58	0.90	0.90	0.95	0.95
primary-school	17.8	15.9	15.7	48.3	0.68	0.68	0.68	0.68	0.69	0.69	0.70	0.77
restaurant-reviews	26.5	36.2	36.1	97.6	0.64	0.59	0.60	0.62	0.56	0.65	0.46	0.46
restaurant-reviews-swapped	2.2	1.8	1.6	2.3	0.68	0.68	0.68	0.68	0.94	0.94	0.94	0.94
spanish-barometer	1.5	1.4	1.1	3.6	0.09	0.08	0.09	0.09	0.81	0.78	0.82	0.82
united-nations	123.4	190.0	180.0	445.5	0.47	0.21	0.45	0.45	0.69	0.57	0.80	0.80
walmart	19856.1	-	5611.3	6778.3	-	-	-	-	-	-	-	-
walmart-swapped	4870.8	-	4079.5	13396.7	0.16	0.10	0.16	0.16	0.70	0.62	0.72	0.91
workspace1	0.3	0.3	0.3	0.5	0.39	0.39	0.39	0.39	0.76	0.75	0.76	0.76
zoo-animals	3.2	3.1	11.3	28.0	0.51	0.52	0.50	0.50	0.68	0.67	1.00	1.00

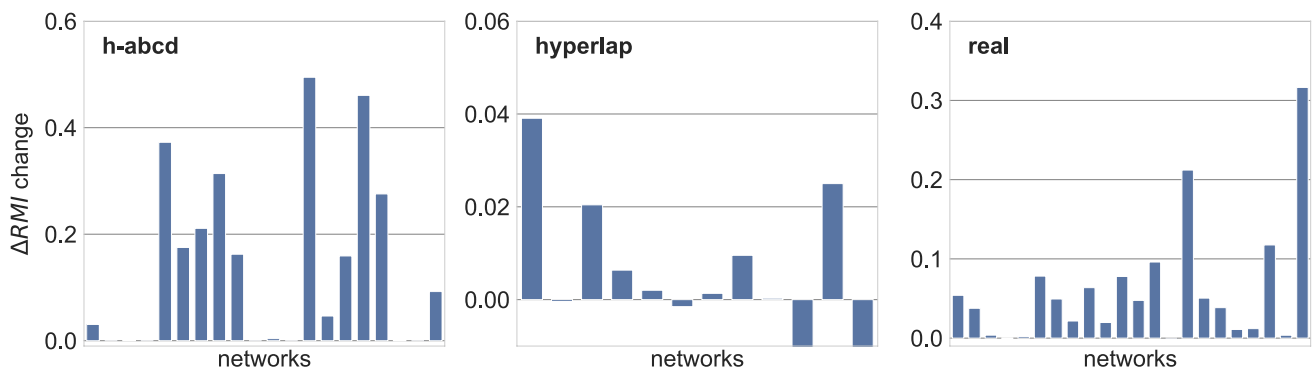


Fig. 8 Absolute change of ARMI score ΔARMI of $\max(\text{F-LOUVAIN}, \text{P-LOUVAIN}, \text{H-LEIDEN})$ compared to H-LOUVAIN for each dataset. The results are divided into *h-abcd*, *hyperlap* and *real-world* instances, whereas datasets without an ARMI-score for H-LOUVAIN are omitted.

the previous section. Based on the insights from Sec. 4.2, the parameter tuple (k, δ) for both P-LOUVAIN and H-LEIDEN is chosen such that $k \in \{5, 10, 20, 50, 200\}$ and $\delta := 0.2$. Recall that F-LOUVAIN does not require any parameters.

Ranking Results To show the strengths and weaknesses of each algorithm, we split the evaluation further by looking at the qualitative and quantitative results. For the qualitative evaluation, we rank each algorithm (first to fourth) w. r. t. q_H and ARMI, respectively, and count the number of times a rank is achieved. Timeouts are covered as an additional data point. In this way, we can track how well an algorithm performs overall and how many times it does not converge to a result. The rankings are shown in Figure 6 (upper row for q_H , lower row for ARMI).

Results based on Aggregate Scores For the quantitative comparison, we track the deviation in hypermodularity Δq_H score and ARMI score for F- and P-LOUVAIN, H-LEIDEN, with H-LOUVAIN as baseline value. Networks for which H-LOUVAIN does not produce a result are neglected (to be overly fair). The results are shown in Figure 7 (upper row for q_H , lower row for ARMI). Concerning q_H alone, H-LOUVAIN performs fairly stable overall, even without BOT. For the majority of cases for *h-abcd* and *real-world* networks, it comes out on top. On closer inspection, for all such cases, either F-LOUVAIN or H-LEIDEN yield q_H scores that are quite close to those of H-LOUVAIN. With a few exceptions for F-LOUVAIN on *real-world* networks, most instances show a comparable performance of all algorithms concerning the q_H -scores, often within the range $|\Delta q_H| < 0.05$. F-LOUVAIN performs generally better (both in ranking and score-wise) for generated hypergraphs, while P-LOUVAIN and H-LEIDEN perform better for *real-world* instances. H-LEIDEN and P-LOUVAIN perform similarly overall, with H-LEIDEN showing slightly higher q_H scores, albeit these differences are

often very minor. The more prominent effect of H-LEIDEN is the reduced variance of the results for *real-world* networks compared to P-LOUVAIN.

Both properties (higher scores and reduced variance) for H-LEIDEN are confirmed when also considering ARMI rankings/scores. For *real-world* instances, the deviations in the ARMI scores of H-LEIDEN compared to H-LOUVAIN are considerable, reaching up to ± 0.3 depending on the instance. To a lesser extent, this is also true for *hyperlap* instances, showing both changes in the range ± 0.05 . The only *consistently positive* deviations compared to H-LOUVAIN can be observed for F-LOUVAIN and the ARMI score for *h-abcd* networks; how strong the deviation is depends largely on the instance and thus on the set of parameters used to generate the network.

Fine-grained Results The results above suggest that further analysis is needed to identify clear advantages of the proposed algorithms over H-LOUVAIN. Since many results for ARMI and especially q_H are very close to each other, and using the fact that both scores are normalized, we say in the following that an algorithm has a higher/lower score than its competitors if it is separated by at least ± 0.05 in absolute score. Using this comparator, H-LOUVAIN has only higher q_H scores for smaller generated networks with $n = 1k$ (both *h-abcd* and *hyperlap*). In all other cases, the results are closer to each other, showing not enough separation.

When inspecting the instances from all sets one-by-one (see Figure 8), either F-LOUVAIN or H-LEIDEN/P-LOUVAIN (not both) shows equal or higher ARMI-scores than H-LOUVAIN. This indicates that we can group the proposed algorithms into two subsets. Indeed, we can identify subsets of networks for which F-LOUVAIN shows a higher ARMI score than H-LEIDEN and vice versa, one always creating comparable or higher scores than H-LOUVAIN. One example subset is the set

of *h-abcd* networks with the generator parameter x set to 0.5. For these networks, F-LOUVAIN creates higher ARMI scores than H-LOUVAIN, H-LEIDEN lower ones. The results indicate that (at least for the *h-abcd* model) F-LOUVAIN is suited to detect communities with higher disassociative community structure. Another set is given by the *real* networks, for which the average hyperedge volume is ≥ 5 , while at the same time having a low percentage of hyperedges with $\text{vol}(E) \in \{2, 3\}$ (like *music-reviews*). Here the same trend of higher ARMI scores for F-LOUVAIN and lower scores for H-LEIDEN can be observed. On the other hand, the higher ARMI scores for H-LEIDEN in the *real* set are all observed for networks where the ground truth is either very coarse or when each node is its own community. One example is *enron-mail*, where the ground truth only consists of two groups, either internal staff members or external partners being part of an email conversation. One can easily imagine further substructures in both groups; when detected, they create high ARMI scores. On the other end of the spectrum is *bakery-basket*, where the community assignment refers to items sold on a shopping trip. High ARMI scores for these cases indicate that H-LEIDEN and P-LOUVAIN can detect such finer/coarser communities more faithfully.

4.5 Summarizing Discussion

Based on the quality results and also considering the running time evaluation, we recommend the following usage of the considered algorithms in real-world network analysis workflows (for which the instance group *real-world* provides the best fit):

- When aiming for a **high ARMI score**, hence a faithful community representation, execute all three of our algorithms and take the best result. The reason is that at least one of the three is often better than H-LOUVAIN, but it is unclear (so far) a priori which one. When the hypergraph has certain structural features (in particular a low percentage of hyperedges with volume 2 or 3 and a mean edge volume $|e| \geq 5$), then our results indicate a strong performance of F-LOUVAIN, so that one could forego to run all three algorithms.
- Use H-LOUVAIN if a **stable and good score for q_H** is of interest.

While there can be reasons to aim for high q_H scores in real-world analysis workflows, we believe that high ARMI are preferable in most cases. Finally, it should be noted that the generated instances show different qualitative and quantitative results for both q_H and ARMI compared to the results obtained with the *real-world* instances. Network analysts are probably more interested in real-world behavior, though.

5 Conclusions

We proposed three new variants of the H-LOUVAIN algorithm for hypergraph community detection that reduce the need for expensive parameter tuning while preserving the direct optimization of hypergraph-based community structure. F-LOUVAIN uses one graph-modularity maximization round as a fast starting solution, whereas P-LOUVAIN initializes the multilevel process by hypergraph partitioning with an empirically robust search over the number of communities. In addition, H-LEIDEN extends the Leiden refinement idea to hypergraphs using a hypergraph-cut criterion, thereby enlarging the local search space. Our experiments on more than 50 generated and real-world hypergraphs show that these variants are competitive with H-LOUVAIN and often preferable when the goal is a faithful community representation measured by ARMI. Overall, the results indicate that simple, structure-aware initialization and refinement strategies can replace complex mixed-modularity parameter searches in many practical settings.

There are several directions for interesting future work: (i) a detailed investigation of particular hypernetworks along the lines of (Benati et al. 2025) with our algorithms, (ii) a software-driven automatic selection of one of our algorithms, (iii) a better scalability of hypergraph community detection algorithms, for example by using parallelism, and (iv) a dynamic algorithm for time-varying hypergraphs which computes a new solution that is consistent with previous time steps and does so faster than recomputing a clustering from scratch.

Author contributions FBT and HM designed the research project; FBT designed the algorithms and implemented them; HM provided guidance on algorithm design; FBT and HM designed the experiments; FBT implemented and executed the experiments; FBT wrote the original draft manuscript; FBT and HM reviewed and edited the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. We acknowledge support by the Open Access Publication Fund of Humboldt-Universität zu Berlin. Open Access funding enabled and organized by Projekt DEAL.

Data availability Code, reproducibility instructions, datasets, and results are available on Github: <https://github.com/KIT-EAE/parameter-light-hyperleiden>.

Declarations

Conflict of interest The authors have no conflict of interest or conflicts to declare that are relevant to the content of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the

source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Amburg I, Veldt N, Benson AR (2020) Clustering in graphs and hypergraphs with categorical edge labels. *Proceedings of the Web Conference*
- Angriman E, Grinten A, Looz M, Meyerhenke H, Nöllenburg M, Predari M, Tzovas C (2019) Guidelines for experimental algorithmics: a case study in network analysis. *Algorithms* 12(7):127
- Benson AR, Abebe R, Schaub MT, Jadbabaie A, Kleinberg J (2018) Simplicial closure and higher-order link prediction. *Proc. Natl. Acad. Sci. USA*
- Bonald T, Charpentier B, Galland A, Holloco A (2018) Hierarchical graph clustering using node pair sampling. *arXiv:1806.01664 arXiv preprint*
- Battiston F, Cencetti G, Iacopini I, Latora V, Lucas M, Patania A, Young J-G, Petri G (2020) Networks beyond pairwise interactions: Structure and dynamics. *Phys Rep* 874:1–92
- Brandes U, Delling D, Gaertler M, Gorke R, Hoefer M, Nikoloski Z, Wagner D (2007) On modularity clustering. *IEEE Trans Knowl Data Eng* 20(2):172–188
- Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. *J Stat Mech: Theory Exp* (10):10008
- Brahim L, Mouad M, Chihab-Eddine C, Ali I (2024) A survey on community detection: Applications, algorithms, and challenges. *J Theor Appl Inf Tech* 102(12):
- Benati S, Puerto J, Temprano F (2025) Modularity for hypergraph clustering: Methodologies and applications. Available at SSRN 5009706:
- Contreras-Aso G, Criado R, Salas GV, Yang J (2023) Detecting communities in higher-order networks by using their derivative graphs. *Chaos Solitons & Fractals* 177:114200
- Chodrow P, Eikmeier N, Haddock J (2023) Nonbacktracking spectral clustering of nonuniform hypergraphs. *SIAM J Math Data Sci* 5(2):251–279
- Chodrow PS, Veldt N, Benson AR (2021) Generative hypergraph clustering: From blockmodels to modularity. *Sci Adv* 7(28):1303
- Cantwell GT, Young J-G (2020) Improved mutual information measure for clustering, classification, and community detection. *Phys Rev E* 101(4):042304
- Dai Q, Gao Y (2023) Mathematical foundations of hypergraph. *Hypergraph Computation*. Springer, Singapore, pp 19–40
- Felippe H, Battiston F, Kirkley A (2024) Network mutual information measures for graph similarity. *Commun Phys* 7(1):335
- Feng S, Heath E, Jefferson B, Joslyn C, Kvinge H, Mitchell HD, Praggastis B, Eisfeld AJ, Sims AC, Thackray LB (2021) Hypergraph models of biological networks to identify genes critical to pathogenic viral response. *BMC Bioinf* 22(1):287
- Gottesbüren L, Hamann M, Schlag S, Wagner D (2020) Advanced Flow-Based Multilevel Hypergraph Partitioning. In: Faro S, Cantone D (eds) 18th International Symposium on Experimental Algorithms (SEA 2020), vol 160. *Leibniz International Proceedings in Informatics (LIPIcs)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 11–11115
- Jerde M, Kirkley A, Newman M (2025) Normalized mutual information is a biased measure for classification and community detection. *Nat Commun*
- Kamiński B, Misiorek P, Prałat P, Thériège F (2024) Modularity based community detection in hypergraphs. *Journal of Complex Networks* 12(5):041
- Karrer B, Newman MEJ (2011) Stochastic blockmodels and community structure in networks. *Phys Rev E* 83(1):016107. <https://doi.org/10.1103/PhysRevE.83.016107>
- Kamiński B, Olczak T, Pankratz B, Prałat P, Thériège F (2022) Properties and performance of the abcde random graph model with community structure. *Big Data Research* 30:100348
- Kamiński B, Poulin V, Prałat P, Szufel P, Thériège F (2019) Clustering via hypergraph modularity *PloS one* 14(11):0224307
- Kamiński B, Prałat P, Thériège F (2021) Artificial benchmark for community detection (abcd)—fast random graph model with community structure. *Network Science* 9(2):153–178
- Kamiński B, Prałat P, Thériège F (2023) Hypergraph artificial benchmark for community detection (h-abcd). *Journal of Complex Networks* 11(4):028
- Ke ZT, Shi F, Xia D (2019) Community detection for hypergraph networks via regularized tensor power iteration. *arXiv:1909.06503 arXiv preprint*
- Kumar T, Vaidyanathan S, Ananthapadmanabhan H, Parthasarathy S, Ravindran B (2020) Hypergraph clustering by iteratively reweighted modularity maximization. *Applied Network Science* 5(1):52
- Lee G, Bu F, Eliassi-Rad T, Shin K (2024) A survey on hypergraph mining: Patterns, tools, and generators. *arXiv:2401.08878 arXiv preprint*
- Lee G, Choe M, Shin K (2021) How do hyperedges overlap in real-world hypergraphs?-patterns, measures, and generators. *Proceedings of the Web Conference 2021*. pp 3396–3407
- Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 78(4), 046110
- Landry NW, Lucas M, Iacopini I, Petri G, Schwarze A, Patania A, Torres L (2023) XGI: A Python package for higher-order interaction networks. *Journal of Open Source Software* 8(85):5162
- Newman MEJ (2006) Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* 103(23), 8577–8582 <https://www.pnas.org/doi/pdf/10.1073/pnas.0601602103>
- Peixoto TP (2014) Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Phys Rev E* 89(1):012804
- Ruggeri N, Contisciani M, Battiston F, De Bacco C (2023) Community detection in large hypergraphs. *Sci Adv* 9(28):9159
- Rossetti G, Milli L, Cazabet R (2019) Cdlb: a python library to extract, compare and evaluate communities from complex networks. *Appl Netw Sci* 4(1):1–26
- Schlag S Karlsruhe Institut für Technologie (KIT) (2020) (46.12.02; LK 01 PhD thesis High-quality hypergraph partitioning)
- Su X, Xue S, Liu F, Wu J, Yang J, Zhou C, Hu W, Paris C, Nepal S, Jin D (2022) A comprehensive survey on community detection with deep learning. *IEEE Transactions on Neural Networks and Learning Systems*
- Shen G, Zeng W, Yang J (2024) Research on migraine classification model based on hypergraph neural network. *J Supercomput* 80(17):25403–25423
- Traag VA, Van Dooren P, Nesterov Y (2011) Narrow scope for resolution-limit-free community detection. *Phys Rev E* 84:016114
- Traag VA, Waltman L, Van Eck NJ (2019) From Louvain to Leiden: guaranteeing well-connected communities. *Sci Rep* 9(1):5233
- Vinh NX, Epps J, Bailey J (2009) Information theoretic measures for clusterings comparison: is a correction for chance necessary?

- Proceedings of the 26th Annual International Conference on Machine Learning. pp 1073–1080
- Wang Y, Kleinberg J (2024) From graphs to hypergraphs: Hypergraph projection and its remediation. [arXiv:2401.08519](https://arxiv.org/abs/2401.08519) arXiv preprint
- Wang W, Nie Y, Li W, Lin T, Shang M-S, Su S, Tang Y, Zhang Y-C, Sun G-Q (2024) Epidemic spreading on higher-order networks. *Physics Rep.* **1056**
- Yang J, Wang F, Li Z, Yang Z, Dong X, Han Q (2023) Constructing high-order functional networks based on hypergraph for diagnosis of autism spectrum disorders. *Front Neurosci* 17:1257982
- Zhang L, Peixoto TP (2020) Statistical inference of assortative community structures. *Physical Review Research* 2(4):043271
- Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.