PASCO (PArallel Structured COarsening): an overlay to speed up graph clustering algorithms

Etienne Lasalle, Rémi Vaudaine, Titouan Vayer, Pierre Borgnat, Rémi Gribonval, Paulo Gonçalves, Màrton Karsai

PASCO (PArallel Structured COarsening): an overlay to speed up graph clustering algorithms

Lasalle Etienne[1*], Vaudaine Rémi[1], Vayer Titouan[1], Borgnat Pierre[2], Gonçalves Paulo[1], Gribonval Rémi[1], Karsai Márton[3,4]

[1]Inria, ENS de Lyon, CNRS, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France.
[2]CNRS, ENS de Lyon, LPENSL, UMR5672, F-69342, Lyon cedex 07, France.
[3]Department of Network and Data Science, Central European University, 1100 Vienna, Austria.
[4]National Laboratory for Health Security, HUN-REN Alfréd Rényi Institute of Mathematics, 1053 Budapest, Hungary.

*Corresponding author(s). E-mail(s): etienne.lasalle@ens-lyon.fr;
Contributing authors: remi.vaudaine@ens-lyon.fr; titouan.vayer@inria.fr; pierre.borgnat@ens-lyon.fr; paulo.goncalves@inria.fr; remi.gribonval@inria.fr; karsaim@ceu.edu;

Abstract

Clustering the nodes of a graph is a cornerstone of graph analysis and has been extensively studied. However, some popular methods are not suitable for very large graphs: *e.g.*, spectral clustering requires the computation of the spectral decomposition of the Laplacian matrix, which is not applicable for large graphs with a large number of communities. This work introduces PASCO, an overlay that accelerates clustering algorithms. Our method consists of three steps: 1- We compute several independent small graphs representing the input graph by applying an efficient and structure-preserving coarsening algorithm. 2- A clustering algorithm is run in parallel onto each small graph and provides several partitions of the initial graph. 3- These partitions are aligned and combined with an optimal transport method to output the final partition. The PASCO framework is based on two key contributions: a novel global algorithm structure designed to enable parallelization and a fast, empirically validated graph coarsening algorithm that preserves structural properties. We demonstrate the strong performance of

PASCO in terms of computational efficiency, structural preservation, and output partition quality, evaluated on both synthetic and real-world graph datasets.

Keywords: Graph Analysis, Community Detection, Large-Scale Networks, Graph Coarsening, Optimal Transport

1 Introduction

Graphs are a fundamental tool to model modern data sets as they become increasingly complex. Graphs allow one to represent complex systems of interacting entities, and applications are found in almost all domains of science. A pillar of graph analysis is the problem of *community detection* where one wants to partition the nodes of a graph so that nodes with similar connectivity patterns are clustered [1]. This problem arises in various domains, such as social sciences and genomics [2]. This task has already been extensively studied both theoretically and practically. However, these algorithms are often unsuited for large-scale community detection problems where the number of nodes N and communities k can become prohibitive.

Several avenues have been explored to solve these scaling issues. Most follow this general scheme: first, reduce the size of the input graph, then cluster the reduced graph, and finally export the partition of the reduced data to the original data. There are two dominant ways to reduce input data size: sampling or coarsening.

The present article proposes a new coarsening-based algorithmic overlay to reduce the overall clustering procedure computation time. We focus on undirected networks and develop a versatile framework *that can be used with any chosen clustering method*. The method consists of three main parts and two novel contributions are proposed. First, the *coarsening* phase computes several simpler and smaller representations of the input graph. We derive a new fast and empirically structure-preserving algorithm based on random edge contractions. The algorithm is executed multiple times in parallel to generate several simplified representations of the input graph. Then, in the *clustering* part, any user-specified clustering algorithm adapted to weighted undirected graphs can be run in parallel on these simple graphs. Finally, after lifting the partitions of the coarsened graphs to partitions of the input graph, we process to the *fusion* part. Using an optimal-transport-based method, we combine these partitions to produce a better and final partition of the input graph.

1.1 Contributions

- We propose PASCO, a new three-step coarsening-based framework to speed up graph clustering algorithms. Innovation comes from the structure of the algorithm that computes many differently coarsened graphs before clustering them independently (see Figure 1). It is a flexible design and serves as a computational overlay that can be applied to any clustering algorithm.

- We design a new fast and efficient random coarsening algorithm as the foundation of the above method. Our approach is opposed to classical coarsening-based clustering approaches that rely on convoluted, and often costly, coarsening mechanisms.
- We extensively evaluate PASCO and its components. The coarsening and fusion steps are first analyzed to confirm the preservation of the structure and the increase in partition quality. Then, the entire PASCO pipeline is tested on synthetic and real graph data. The results show speedups for computationally heavy clustering methods, while maintaining or even improving quality on complex real-world networks.

1.2 Related Works

Clustering the nodes of a graph has attracted a lot of attention: spectral methods [3], information-theoretic approaches [4], model-based approaches [5], and the popular maximization of modularity [6]. We refer the reader to [1] for reviews on community detection methods. However, all of these methods do not always scale well. Hence, various works have been proposed to speed up clustering computations. Some of these approaches are detailed now.

General fast approaches to clustering: Substantial work has been devoted to accelerating spectral clustering, where the efforts essentially focus on faster solving of the spectral decomposition, *e.g.*, using the Nyström method [7] or the power method [8]. In [9], the authors tackle the high computational cost of spectral clustering by approximating the spectral embedding using an efficient graph filtering of random signals and accelerating the k-means part using a sub-sampling strategy. Another way to accelerate clustering is to reduce the number of edges in the graph before computing the clustering. To do so, several sparsification techniques have been proposed, either by sampling and removing random edges [10] or using effective resistance [11]. The review [12] provides an overview of acceleration techniques in the case of spectral clustering. Other fast approaches construct a bipartite graph between the initial set of nodes and a new and smaller set of nodes and recover the community structure of the input graph from this bipartite graph [13, 14].

Coarsening approaches: Most coarsening approaches [15–17] rely on an iterative multilevel edge-contraction-based coarsening algorithm. That is, several coarsened graphs of decreasing sizes are computed iteratively. At each coarsening level, several edges are selected and collapsed to put their end vertices into the same hypernode. Then, some clustering algorithm is run *on the smallest coarsened graph* before *lifting* the result iteratively back to the next larger set of nodes. At each level, existing approaches exploit mainly one coarsening process. Therefore, at each lifting step, the partition is refined by evaluating the gain (w.r.t. a certain cost) to obtain a satisfying final partition. There is typically a trade-off between the degree of graph simplification used to accelerate clustering and the resources required to recover an accurate partition. Our new coarsening algorithm is designed to prioritize efficiency in this trade-off: the quality of the partition will be ensured by its insertion into our three-step framework and, in particular, the fusion of clusters obtained from *multiple* coarsened graphs.

Clustering ensemble: Clustering ensemble combines multiple results of clustering *the same graph* to form a more robust consensus, improving stability and reliability by aggregating diverse partitions from different off-the-shelf algorithms or parameter settings. PASCO can be framed within the clustering ensemble framework, as we obtain several partitions of the initial graph (by random coarsening, clustering, and lifting) and combine them to output a final partition. Although both approaches involve merging multiple partitions, the philosophy is different from the usual clustering ensemble techniques: we first aim to *accelerate* clustering and not especially enhance the final clustering quality in terms of stability and robustness. Overall, clustering ensemble methods can be divided into two main categories [18]. The first one is based on consensus functions where the output clustering is the one optimizing a notion of agreement of the given partitions [19], while the second constructs a co-association matrix that characterizes the similarity between the data items based on the partitions [20].

1.3 Outline of the paper and notations

The general framework of PASCO is introduced in Section 2. Its key phases are then further explained. The coarsening is detailed in Section 3 while alignment and fusion are presented in Section 4. The experimental results are shown in Section 5.

For any integer $n \geq 1$, we denote by $\mathbf{1}_n$ the vector of $\mathbb{R}^n$ with all entries equal to 1. The set of integers ranging from 1 to n is denoted by $[\![n]\!]$. We will use exponents $G^{(\ell)}$, $1 \leq \ell \leq c$ to denote sequences of c coarsened graphs, while the indices r in $G_r, 1 \leq r \leq R$ denote the output of R independent instances of the randomized coarsening algorithm.

2 The PASCO approach for clustering

Our approach aims to speed up clustering computations by applying a given clustering algorithm to several reduced versions of the initial graph and then combining the results to output the final clustering.

Given some initial graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with vertex set $\mathbb{V}$ and edge set $\mathbb{E}$, the *random* coarsening algorithm is run R times to obtain the coarsened graphs $G_1, \cdots, G_R$. A clustering algorithm is then applied to each of these graphs. The resulting partitions of the nodes of $G_1, \cdots, G_R$ are lifted up to partitions of the nodes of $\mathbb{G}$ and then combined to retrieve as much information as possible and output a final clustering. See Figure 1 for a schematic illustration of our approach. Below, we provide an overview of each part of the pipeline (coarsening, clustering, alignment, and fusion). The reader can refer to the next sections for more details.

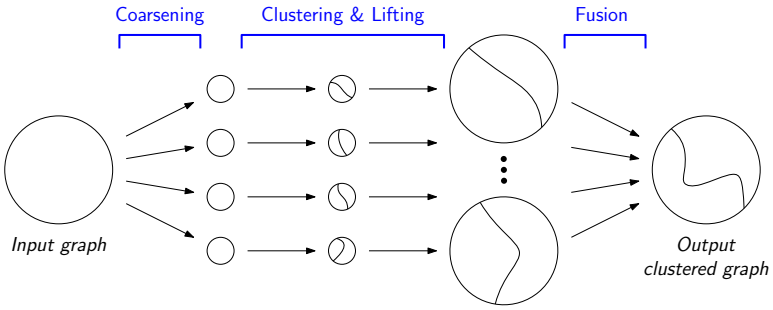


Fig. 1: PASCO pipeline. Coarsening: Apply a coarsening algorithm to compute several small graphs. Clustering & Lifting: Apply (in parallel) an off-the-shelf clustering algorithm on each of the coarsened graphs and lift each partition to a partition of the input graph. Fusion: Combine the partitions to output the final partition.

Coarsening: We propose a new randomized coarsening algorithm that takes into account the structure of the initial graph. This algorithm adopts a multilevel approach where we create the sequence of incrementally coarsened graphs $\mathbb{G} = G^{(0)}, \ldots, G^{(l)}, \ldots, G^{(c)} = G$, starting from the initial graph $\mathbb{G}$ of size N. Each graph $G^{(\ell+1)}$ is obtained by coarsening $G^{(\ell)}$ to reduce the number of nodes from $n^{(\ell)}$ to $n^{(\ell+1)}$, such that $n^{(\ell)} > n^{(\ell+1)}$. The number of coarsening steps c is the one required to reach the small target size n. Each iterative coarsening (from $G^{(\ell)}$ to $G^{(\ell+1)}$) is based on an edge-contraction approach. Our strategy is to sample edges (according to a given rule) and contract them by putting the two end-vertices into the same "hypernode", as shown in Figure 2. We repeat this procedure until the target size (n) of the coarsened graph is reached or no edge is available (according to our sampling rule). Remark that coarsening generates small weighted graphs with self-loops. The challenge of this approach is to find a relevant sampling rule so that the coarsening algorithm is both fast and as information-preserving as possible. In Section 3, we provide all the details for this coarsening step, including details on the sampling rule, its positioning with respect to the state of the art, and the properties of the coarsening.

Clustering: The clustering phase consists in finding a partition of the hypernodes for each of the R coarsened graphs $G_1, \cdots, G_R$ (see Figure 1). Interestingly, we can operate independently on each graph and compute in parallel these partitions to accelerate computation. Our pipeline is designed so that any clustering algorithm can be used as long as it handles undirected weighted graphs. However, we only focus on algorithms that generate non-overlapping partitions. We point out that our fusion part will be able to cope with partitions with different numbers of clusters. Therefore, PASCO can be used with clustering algorithms that automatically choose the number of communities. At this stage, we obtain a partition of the hypernodes for each coarsened graph G_r. These partitions are then *lifted* to partitions of the nodes of the initial graph: each initial node inherits the class of the hypernode to which it belongs.

Alignment and Fusion: The final step of PASCO is to combine the various partitions lifted to the original graph into a single output partition. This is inspired by methods of ensemble clustering. To do so, we propose to achieve consensus among

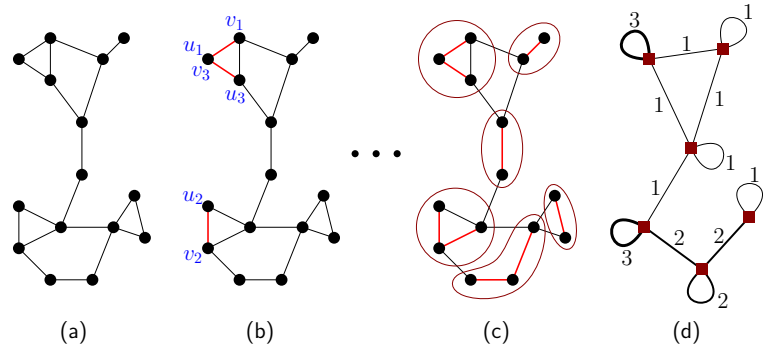


Fig. 2: Illustration of one coarsening iteration (Algorithm 2). (a) The original graph; (b) The first three pairs of sampled nodes (in blue) and their corresponding edges (in red): first, u_i is sampled uniformly at random among the unvisited nodes, then v_i is sampled among the neighbors of u_i (no restriction on v_i). (c) The sampled edges at the end of the sampling phase; each set of vertices connected by sampled edges is circled and yields a hypernode. (d) The coarsened graph. Squares represent hypernodes and edge weights are given and represented by edge thickness (note the presence of self-loops).

multiple partitions by leveraging optimal transport (OT) [21]. We first briefly define the partition matrices that are used to encode partitions.

Definition 1 (Partition matrix)**.** *A matrix $P \in \{0,1\}^{N\times k}$ is a* partition matrix *if it is column-stochastic $P\mathbf{1}_k = \mathbf{1}_N$. The fact that $P_{ij} = 1$ indicates that node i is attributed to cluster j. We denote by $\mathcal{P}_{N,k}$ the set of partition matrices.*

Let $P_1, \cdots, P_R$ be the partition matrices representing the R partitions of the initial graph such that $P_r \in \mathcal{P}_{N,k_r}$ with k_r being the number of clusters in the r-th partition. There are several challenges that need to be tackled in order to obtain a consensus partition from $(P_r)_{r\in[\![R]\!]}$. First, these partitions may not have the same number of clusters as some clustering algorithms infer the number of clusters. Second, the partitions may not be consistent with each other, and even if they are, it is necessary to identify the unknown correspondences between their clusters[1]. To overcome these challenges, a core idea from the literature is to find a reference partition $\overline{P} \in \mathcal{P}_{N,\overline{k}}$ which is the "closest" to all the partitions $P_1, \cdots, P_R$. Optimal Transport (OT) provides tools to align probability distributions according to a "least-effort" principle. It can be used to measure a notion of similarity between partitions. Given a fixed prescribed number of clusters $\overline{k}$, we solve the OT barycenter problem

$$\min_{\overline{P}\in\mathcal{P}_{N,\overline{k}}} \frac{1}{R}\sum_{i=r}^{R} \mathrm{W}_2^2(\mu_{P_r}, \mu_{\overline{P}})\,, \tag{1}$$

where $\mu_{\overline{P}}, \mu_{P_r}$ are discrete probability distributions associated to $\overline{P}, P_r$, W_2^2 is the squared Wasserstein distance and $\mathcal{P}_{N,\overline{k}}$ is the set of partition matrices (Definition 1). In

[1] For example, permuting the columns yields different representations of the same partition.

practice, to solve this barycenter problem, the algorithm starts from an initial reference $\overline{P}$, and then alternates between realigning the partitions to this reference (*alignment step*), and updating this reference (*fusion step*) until convergence. We provide all the details about this alignment and fusion step in Section 4.

3 Coarsening in PASCO: contributions

This section details the implementation of a coarsening-based clustering method, reviews classical coarsening approaches, highlights PASCO's design for enhanced speed, and conjectures a phase transition in stochastic block model parameters when PASCO yields good performance.

3.1 General principles of coarsening methods

Let us present the general principles that are shared by classical coarsening methods. Coarsening is encoded through *coarsening tables*, which are arrays indicating to which hypernode each node is associated, as formalized in this definition.

Definition 2 (Coarsening Table)**.** *For a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with vertex set $\mathbb{V} = \{u_1, \dots, u_N\}$, coarsened into a graph $G = (V, E)$ with $V = \{u'_1, \dots, u'_n\}$, the coarsening table is the vector $h \in [\![n]\!]^N$, such that node $u_i \in V$ is associated to hypernode $u'_{h_i} \in V$. We can also encode this table h into a* coarsening matrix *$H \in \{0,1\}^{N \times n}$, where $H_{i,j} = 1$ if and only if node u_i is associated to hypernode u'_j.*

We recall that coarsening is usually done by constructing a sequence of incrementally coarsened graphs $\mathbb{G} = G^{(0)}, \dots, G^{(c)} = G$ starting from a graph size N down to the target size n. The target size is defined by $n = \lfloor N/\rho \rfloor$, where ρ is called the *compression factor* and is a hyper-parameter of the coarsening method. This main scheme is detailed in Algorithm 1, where h is the coarsening table from the initial graph $\mathbb{G}$ to the current most coarsened graph $G^{(\ell)}$. When one coarsening step is performed (step 8), the coarsening table $h^{(\ell)}$, from $G^{(\ell)}$ to $G^{(\ell+1)}$, is obtained. Hypernodes are then relabeled so that $h^{(\ell)}$ takes consecutive integer values starting at 1. Then, the next coarsened graph $G^{(\ell+1)}$, or rather its adjacency/weight matrix $A^{(\ell+1)}$, can be computed using the adjacency matrix $A^{(\ell)}$ of graph $G^{(\ell)}$ and the coarsening matrix $H^{(\ell)}$ encoding $h^{(\ell)}$ according to $A^{(\ell+1)} = {H^{(\ell)}}^\top A^{(\ell)} H^{(\ell)}$. Then h is updated coordinate-wise using by $h_i \leftarrow h^{(\ell)}_{h_i}, \quad \forall i \in [\![N]\!]$.

The diversity in graph coarsening methods arises from various sampling strategies for selecting collapsing edges. In [15], edges are contracted by randomly selecting an unvisited node and an unvisited neighbor. The heavy-edge heuristic introduced in [22] prioritizes edges with the heaviest weights, aiming to group similar nodes into the same hypernode. This approach has been extended with tailored weights to optimize specific objectives, such as Graclus [16] for cut optimization and [17] for preserving spectral properties.

After coarsening the initial graph $G^{(0)}$ into $G^{(c)}$, a clustering algorithm is run to obtain a partition $P^{(c)}$ of the nodes of $G^{(c)}$. This clustering information is then transferred from the coarsened graph to the initial graph using a so-called *lifting* step. A simple way to lift a partition $P^{(\ell)}$ of $G^{(\ell)}$ to a partition of $G^{(\ell-1)}$ is to state that each node in $G^{(\ell-1)}$ inherits the cluster of the hypernode of $G^{(\ell)}$ to which they belong.

Mathematically, this translates to the matrix product $P^{(\ell-1)} = H^{(\ell-1)}P^{(\ell)}$, where $H^{(\ell-1)}$ is the coarsening matrix from $G^{(\ell-1)}$ to $G^{(\ell)}$. However, for classical coarsening approaches, this simple lifting method does not provide good quality clustering as the coarsening loses too much information and extra refining steps are necessary.

Overall, existing approaches often focus on complex, computationally intensive coarsening steps. Additionally, by exploiting only one coarsening process, they are bound to make use of computationally costly refinement steps in the lifting procedure to recover a satisfying partition. In the next section, we will see how PASCO differentiates itself from these existing works by resorting to simpler (and thus faster) coarsening and lifting steps. Partitions of good quality will be recovered, not by complexifying the procedure, but by using several coarsening processes in parallel and combining the resulting partitions (see Section 4).

3.2 Coarsening in PASCO

The coarsening approach in PASCO is similar to some existing methods in the sense that it is an iterative and multi-level edge-contraction-based coarsening method. Starting from the initial graph $\mathbb{G}$ of size N, we aim to coarsen it to a smaller graph of target size $n = \lfloor N/\rho \rfloor$, ($\rho$ being the *compressive factor*) following Algorithm 1. The innovation for PASCO comes from the way each iterative coarsening (from $G^{(\ell)}$ to $G^{(\ell+1)}$) is performed (step 8 in Algorithm 1). More precisely, as it is an edge-contraction-based approach, we introduce a new simple but efficient edge sampling mechanism, detailed in Algorithm 2. In this approach, we propose to sample uniformly at random an unvisited vertex u of $G^{(\ell)}$, and sample one of its neighbors v (potentially already visited) uniformly at random (steps 5 and 6 of Algorithm 2). The edge (u, v) is used for contraction; that is, u and v are sent to the same hypernode (step 7), and then both vertices u and v are set as visited (step 8).

Computational efficiency: First, the algorithm aims at minimizing the number of intermediate coarsening steps c by creating hypernodes that contain as many nodes as possible at each step. In [15], the authors proposed to sample u and v from the set of unvisited nodes, restricting the hypernodes to contain at most two nodes. As a consequence, the coarsening step quickly runs out of available edges to collapse and a new intermediate coarsened graph must be computed. To avoid this issue and create bigger hypernodes, we relax the restriction about unvisited nodes: we only require that u is unvisited and we *put no restriction on* v. Moreover, the sampling of collapsing edges by first taking a node uniformly at random is very efficient, as it can be done in $\mathcal{O}(1)$. In contrast, strategies to sample node u according to some non-uniform probability (*e.g.*, a probability proportional to the node degree) are more costly, as they require to compute the cumulative sum of probabilities which is in $\mathcal{O}(n^{(\ell)})$, if $n^{(\ell)}$ is the number of nodes. Finally, we remove the refining steps when lifting the partition back to the input graph, as this will be taken care of in the next step with alignment and fusion of the different partitions obtained. In Appendix C, we detail other simple edge sampling rules that we investigated here but were unsatisfactory for the present work. However, it provides insight into the choices that led to our method.

3.3 Structure preserving properties of the coarsening

This section examines the properties and limitations of PASCO's coarsening on random graphs with community structures. To preserve community information, hypernodes must primarily consist of nodes from the same community, which requires collapsing intra-community edges. We analyze the conditions under which PASCO coarsening favors such edges, focusing on graphs generated by the Symmetric Stochastic Block Model (SSBM) defined below.

Definition 3 (Symmetric Stochastic Block Model)**.** *The SSBM is a random graph model with N nodes divided into k equal-sized communities. Each edge is present with probability p_{in} if inside a community or p_{out} if between communities, independently of all other edges. As in [9], we parametrize the model by N, k, the expected degree $d_N = d \log N$ and the intra-to-inter-community probability ratio*[2] *$\alpha = p_{out}/p_{in}$. We refer to this model by* $\mathrm{SSBM}(N, k, d, \alpha)$.

Consider an input graph drawn from an SSBM with k communities, an edge probability inside communities of p_{in}, and an edge probability between communities of p_{out}. In PASCO, edges to collapse are obtained by first drawing some node u and taking a random neighbor. In expectation, u has np_{in}/k neighbors from its community and $n(k-1)p_{out}/k$ neighbors from other communities. So when $p_{in} > (k-1)p_{out}$, v is more likely to be from the same community as u. More generally, under this condition, we expect the coarsening procedure of PASCO to collapse more inside-community edges than between-community edges. Therefore, we conjecture that PASCO conserves the community structure of a graph drawn from a $\mathrm{SSBM}(N, k, d, \alpha)$ as long as $\alpha = p_{out}/p_{in} < 1/(k-1)$. While, at this stage, this reasoning is a conjecture and is not supported by rigorous arguments, experiments are providing empirical evidence that this phase transition correlates with PASCO's performance as described in our experiments in Section 5.1 and Section 5.3.

4 Alignment and fusion

The last step of PASCO is to align and combine the partitions obtained from various coarsened graphs. We advocate the use of an OT-based approach to align the partitions, as initially proposed in [23]. The key idea is to define a notion of distance between partitions by using the Wasserstein distance W_2. Precisely, one can represent a partition matrices $P \in \mathcal{P}_{N,k}$ as a discrete probability distribution. Writing $P = (p_1, \cdots, p_k)$ where p_j is the j-th column of P, the j-th cluster can be represented by the vector $p_j \in \{0,1\}^N$. The discrete probability distribution in $\mathbb{R}^N$ associated to P is then given by $\mu_P = \frac{1}{k}\sum_{j=1}^{k} \delta_{p_j}$ where δ is the Dirac mass.

For two partitions $\overline{P} \in \mathcal{P}_{N,\overline{k}}, P \in \mathcal{P}_{N,k}$, we can compare them by comparing their associated probability measures $\mu_{\overline{P}}, \mu_P$ through the Wasserstein distance

$$\mathrm{W}_2^2(\mu_P, \mu_{\overline{P}}) = \min_{Q \in \mathcal{Q}_{\mathrm{ot}}(k,\overline{k})} \sum_{i,j=1}^{k,\overline{k}} \|p_i - \overline{p}_j\|_2^2 Q_{i,j}\,. \tag{2}$$

[2] The use of α asa parameter will later be relevant for our experiments as we then keep the density fixed and vary the difficulty level α to recevover the blocks.

In Equation (2) the set $\mathcal{Q}_{\text{ot}}(k, \overline{k}) \subseteq \mathbb{R}_+^{k \times \overline{k}}$ denotes the collection of all *coupling matrices*, *i.e.* matrices $Q \in \mathbb{R}_+^{k \times \overline{k}}$ that satisfy the marginal constraints: $Q\mathbf{1}_{\overline{k}} = \frac{1}{k}\mathbf{1}_k$ and $Q^\top \mathbf{1}_k = \frac{1}{\overline{k}}\mathbf{1}_{\overline{k}}$. Intuitively, the element $Q_{i,j} \in [0, 1]$ represents the amount of probability mass shifted from the i-th cluster of P to the j-th cluster of $\overline{P}$. This coupling can be used to align the clusters of the two partitions: in the special case where $k = \overline{k}$, an optimal solution is given by the permutation that best realigns the clusters [21]. We point out that solving (2) is done through a linear program that can be computed with standard solvers [24] with a worst-case complexity $\mathcal{O}(K^3(\log K)^2)$ where $K = \max\{k, \overline{k}\}$. We rely on this distance to achieve a consensus among the different partitions by solving an OT barycenter problem: we fix a number of desired clusters $\overline{k}$ and look for the partition matrix $\overline{P} \in \mathcal{P}_{N,\overline{k}}$ that minimizes (1).

As described in [25], this barycenter problem can be tackled by alternating between solving R problems of OT and updating the reference $\overline{P}$. As described in Lemma 4, the reference update can be obtained in closed-form by a simple majority vote as follows.

$$\forall i \in [\![N]\!], [\overline{P}]_{ij} \leftarrow \begin{cases} 1 & j \in \underset{p \in [\![\overline{k}]\!]}{\text{argmax}} \left[\sum_{r=1}^{R} P_r Q_r\right]_{ip} \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where $Q_1, \cdots, Q_R$ are the optimal coupling matrices obtained in the previous step when solving the individual OT problems between the previous reference partition and the $P_1, \cdots, P_R$. The algorithm for alignment and fusion is sketched in Algorithm 3. This OT-based alignment + fusion algorithm requires a choice for the target number of clusters $\overline{k}$ and an initial reference $\overline{P}$. We use the following heuristic in practice: If all the k_r are equal, we choose $\overline{k} = k_1$ and initialize the reference partition with $\overline{P} = P_1$, otherwise we choose $\overline{k}$ as the k_r closest to the median number of clusters across the partitions in the dataset median$(k_1, \cdots, k_R)$, and we initialize $\overline{P}$ as the corresponding partition.

Complexity analysis: Let $K = \max\{\overline{k}, k_1, \cdots, k_R\}$ then the algorithm runs in $\mathcal{O}(n_{\text{iter}} R(NK^2 + K^3 \log(K)^2))$ where n_{iter} is the number of iterations required for $\overline{P}$ to converge. Overall, the algorithm scales linearly in N and has roughly a cubic complexity *w.r.t.* the number of clusters, which is often small compared to the number of nodes. In practice, we also observe that n_{iter} is small (on the order of 10).

Other alignment+ fusion methods: As an alternative to the described OT approach, we also investigated the so-called *linear-regression-based* and the *many-to-one* methods [26], as well as a slightly different variant of OT based on quadratic-penalized OT [27]. These methods also solve a barycenter problem but for other notions of distance between partitions. In practice, we find that the standard OT-based method performs better for our application (see Section 5 and especially Figures 4 and E3). The presentation of these alternative methods is deferred to Section B.

5 Experiments

We conducted experiments to evaluate PASCO. Section 5.1 shows that the coarsening step preserves well graph spectral properties. Section 5.2 examines the alignment and

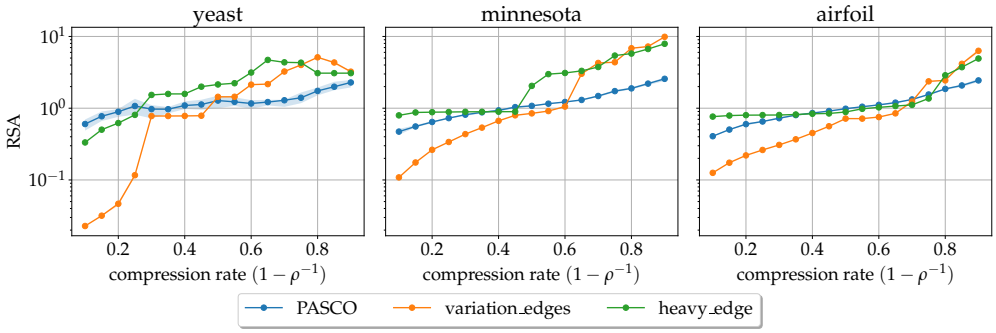


Fig. 3: We represent the RSA (the smaller, the better) of various coarsening schemes (including PASCO) as a function of the compression rate (the higher, the coarser the obtained graph). Shaded areas represent 0.2 upper- and lower-quantiles.

fusion phases. Finally, Section 5.3 and Section 5.4 evaluate PASCO on SSBM and real graphs. See Section E.1 for details about computing resources. The code for PASCO and the experiments is available at https://github.com/elasalle/PASCO.

5.1 Coarsening: conservation of graph spectral properties

We evaluate the coarsening phase using Loukas's framework [17], which analyzes coarsening techniques based on the spectral properties of the graph Laplacian. Loukas introduced the *Restricted Spectral Approximation* (RSA) to quantify how well the projection matrix $\Pi = HH^+$ approximates the identity on $\mathcal{U}_k$, the subspace spanned by the k eigenvectors of L associated to the *smallest* eigenvalues. RSA is defined as the smallest ε such that $\|x - \Pi x\|_L \leq \varepsilon \|x\|_L$ for all $x \in \mathcal{U}_k$. Here, L is the combinatorial Laplacian, H is the binary coarsening matrix, and H^+ is its pseudo-inverse.

Experimental setting: In these experiments, we compare our method to previously existing methods in terms of RSA and computational time. Figure 3 shows the results for our proposed method (PASCO) as well as previously existing methods, namely, the local variation method based on edges proposed by Loukas [17] (`variation_edges`) and the heavy edge collapsing method (`heavy_edge`) [22, 28]). We display the `RSA` values with respect to the *compression ratio* $1 - \rho^{-1}$. Recall that ρ is the *compression factor* defining how much the initial graph is coarsened ($n = \lfloor N/\rho \rfloor$). Assuming that N/ρ is an integer, the *compression ratio* corresponds to the ratio $(N-n)/N = 1-\rho^{-1}$. Figure 3 corresponds to real graphs that were used in the experiments of [17] (see Table E2 for some characteristics of these graphs). For graphs drawn under the SSBM, results are deferred to Figure E1. Also, in Figure E2, we provide the computational times of each coarsening method.

Results: While `variation_edges` and to some extent `heavy_edge` generally yield the best RSA at *small compression rates*, we emphasize that PASCO was not specifically tailored to preserve such spectral properties unlike `variation_edges` which is designed to optimize the RSA. Moreover, we are rather interested in *high compression rates*, as significant clustering computation time gains are to be expected. In this high compression regime, despite the fact that `variation_edges` was tailored to optimize

the RSA, PASCO is still competitive, especially on real graphs where it outperforms both `variation_edges` and `heavy_edge`. Moreover, PASCO proves to be much faster than both other methods: by more than a factor 10 for `heavy_edge` and around a factor 100 for `variation_edges` (see Figure E2). These experiments demonstrate that, although the coarsening step of PASCO is primarily designed for computational efficiency, it also preserves the structures of the initial graph as effectively as, or sometimes even better than, traditional coarsening methods. They also demonstrate that PASCO is much faster than the other coarsening algorithms tested, as expected.

5.2 Effectiveness of the alignment/fusion phase

To demonstrate the alignment+fusion procedure, we now consider a synthetic dataset generated from a two-dimensional Gaussian Mixture Model (GMM) with three clusters. The clusters consist of 500, 400 and 200 points, respectively. Each cluster is sampled from isotropic Gaussian distributions with a standard deviation of 0.25 and centers located at $(0, 1)$, $(1, 0)$, and $(0, 0)$. The resulting dataset is shown in the left panel of Figure 4.

We generate 15 different partitions of the dataset with the goal of recovering the true partition corresponding to the original GMM clusters. The partitions are constructed as follows: first, we randomly select a number of clusters k, drawn uniformly between 3 and 10. Then, we designate k centroids: the first three are randomly selected from each of the true GMM clusters (so that we have at least one starting centroid in each true cluster), while the remaining centroids are uniformly sampled from the remaining points. Each point in the dataset is assigned to the nearest centroid, thereby forming a partition. Forcing each true cluster to be initially represented by at least one centroid ensures that the resulting partition is related to the true partition. See two examples of these generated partitions in Figure 4.

The effectiveness of the proposed alignment and fusion method (Algorithm 3) is evaluated by comparing several alignment techniques: `lin-reg`, `many-to-one`, and the proposed `ot` to recover the true partition. For each case, we consider a randomly initialized reference with $\overline{k} = 3$, such that each point is assigned to a cluster chosen uniformly at random. As a baseline, we compare with a K-means clustering with $K = 3$. We emphasize that the K-means algorithm benefits from the spatial coordinates of the data points, whereas the alignment+fusion methods operate solely on the different partitions $P_1, \cdots, P_R$, and ignore the positions. The experiment is repeated five times, and the average Adjusted Mutual Information (AMI) [29] (see Section E.4 for the definition) between the inferred and true partitions is plotted as a function of the number of partitions R on the right panel of Figure 4. The results indicate that the OT methods achieve performance comparable to K-means (high AMI, with small variance) when $R \geq 10$, while `lin-reg` and `many-to-one` have high variance and struggle to retrieve the true partition for any value of R. This can be explained by the fact that the partitions are quite unbalanced and thus more suited for the coupling constraints. Finally, a partition recovered using the `ot` method is shown in the fourth panel of Figure 4, illustrating that it is nearly a perfect permutation of the true partition.

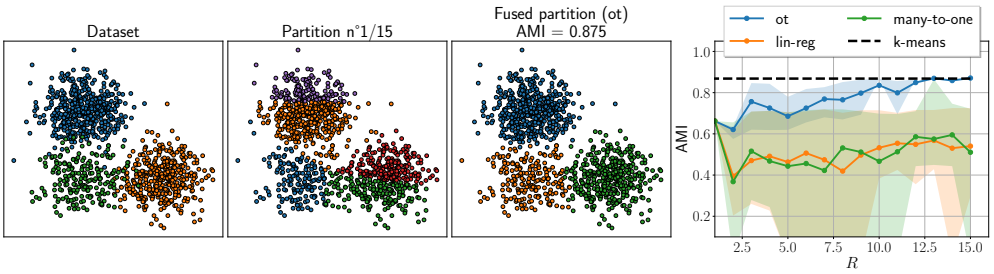


Fig. 4: Experience with a toy dataset drawn from a $2D$ GMM (left). Colors indicate clusters. One partition (out of 15) is depicted, as well as the recovered partition using the alignment+fusion procedure based on `ot`. The right panel presents the average AMI between the true partition and the fused partitions obtained by `ot`, `many-to-one`, and `lin-reg`. Shaded areas represent 0.2 upper- and lower-quantiles over the 100 runs of the alignment+fusion algorithms.

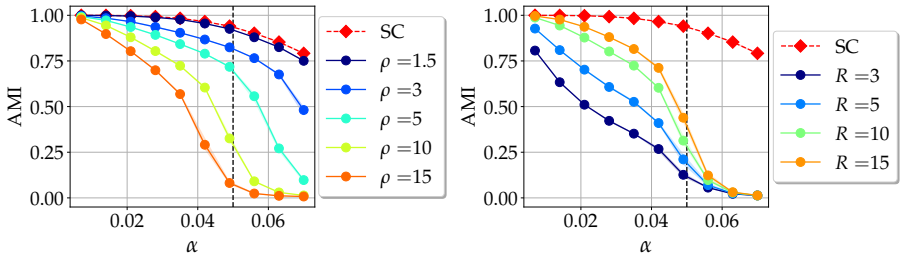
5.3 Synthetic graph experiment and parameter analysis.

We explore the performance of PASCO (coupled with spectral clustering to cluster the coarsened graphs) on an initial graph generated using the symmetric stochastic block model.

Quality of the output partition: To study the influence of the hyperparameters of PASCO, we conduct an experiment on synthetic graph data from the Symmetric Stochastic Block Model SSBM(N, k, d, α), see Definition 3. We take $N = 10^4$ and $k = 20$ and set p_{in} and p_{out} such that the average degree is $d_N = d\log(N)$ with $d = 3/2$. We vary the fraction $\alpha = p_{out}/p_{in}$ from 0 (excluded) to $\alpha_{\text{sup}} = \frac{4}{3}\frac{1}{k-1}$. This range includes both the phase transition threshold of PASCO, of value $1/(k-1)$ (as conjectured in Section 3.3) and the threshold of exact recovery for spectral clustering, *i.e.*, $\alpha_c = (d - \sqrt{d})/(d + (k-1)\sqrt{d})$.

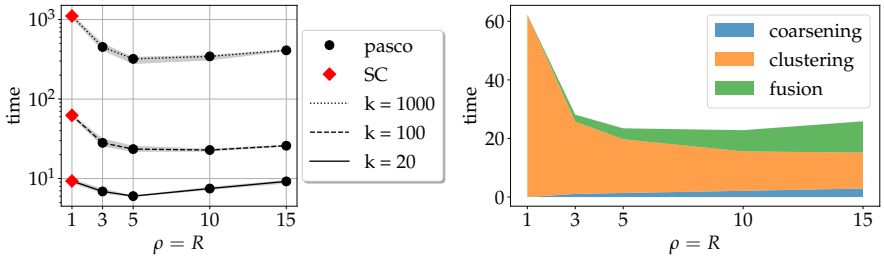
For each set of SSBM parameters, we draw 10 graphs, ensuring their connectedness with rejection sampling. For each graph, we compute the performance of PASCO with spectral clustering measured by the AMI. We study the influence of each parameter: the compression factor ρ (such that $n = \lfloor N/\rho \rfloor$), the number of coarsened graphs R, and the method used for alignment `align_method`. Each parameter varies as follows, while the others are kept constant to a default value (written here in bold): $\rho \in \{1, 3, 5, \mathbf{10}, 15, 20\}$, $R \in \{1, 3, 5, \mathbf{10}, 15, 20\}$, `align_method` $\in$ {`lin_reg`, `many_to_one`, `ot`}. Performance is averaged over the 10 realizations and displayed in Figure 5.

The compression factor ρ determines how small the coarsened graphs are. Hence, the larger ρ, the harder it is to retrieve the communities of the input graphs (see Figure 5a). Moreover, it is even more difficult to recover the communities when they are not much denser than the rest of the graph (large α). However, better performance is achieved with more coarsened graphs (bigger R); see Figure 5b. The rise in AMI with R confirms that the alignment and fusion process is effectively able to combine the noisy information contained in each clustering. We also observe a change in the behavior of



(a) Influence of ρ ($R = 10$). (b) Influence of R ($\rho = 10$).

Fig. 5: Parameters influence on PASCO. The AMI score is average over the 10 runs and shaded areas represent 0.2 upper- and lower-quantiles. Dashed lines correspond to the PASCO threshold.



(a) Computational time. (b) Time details ($k = 100$).

Fig. 6: Influence of ρ and R on PASCO computational time, letting $R = \rho$. The number of communities varies in $\{20, 100, 1000\}$. Timings are averaged over 10 runs and shaded areas represent 0.2 upper- and lower-quantiles.

PASCO after the conjectured threshold ($\alpha > 1/(k-1)$, vertical dashed line). In the Appendix, Figure E3 indicates that all alignment methods perform similarly, excepted when close to PASCO's conjectured phase transition as then `ot` outperforms the other methods.

Study of the computational time: In this experiment, we study the gains in computational time due to PASCO (coupled with spectral clustering (SC) to cluster the coarsened graphs), compared to plain SC. As highlighted in the previous experiment, larger compression factors ρ should be compensated by larger numbers of coarsened graphs R to ensure good partition quality. In the following, we arbitrarily decide to fix the number of coarsened graphs equal to the compression factor, *i.e.*, $R = \rho$ and study the impact of ρ on computational time. The results are presented in Figures 6. The SSBM parameters are set to $N = 10^5$, $d = 3/2$, and $\alpha = 1/(2(k-1))$ and k varies in $\{20, 100, 1000\}$. For each set of SSBM parameters, we draw 10 graphs, ensuring their connectedness thanks to rejection sampling. For each graph, we compute the computational time of PASCO for values of ρ in $\{3, 5, 10, 15\}$.

Figure 6a shows that using $\rho = R > 1$ accelerates the overall computation compared to plain SC, which corresponds to $\rho = R = 1$. For this experiment, an empirical optimum is found around $\rho = R = 5$. The overall speedup is greater when the number of communities k is larger. Indeed, in this case, the spectral decomposition in SC is really computationally demanding, and performing it on smaller coarsened graphs leads to a significant improvement in the computational time. Moreover, even for large ρ, the coarsening and fusion part still amount to only a small proportion of the computational effort, as illustrated in Figure 6b. Figure E4 attests that the significant speedup of Figure 6a does not come at the cost of poor quality of the output partitions.

5.4 Real graphs experiment

To further validate the ability of PASCO to improve the performance of clustering algorithms, we conduct experiments on real datasets. We consider three large graphs (`arxiv`, `mag`, and `products`) come from `Open Graph Benchmark` [30]. The experiments are run on the largest connected component of each graph. Their characteristics are in Table E2. Some datasets have features associated with the nodes, but they are ignored in these experiments as the study is limited to non-attributed graph clustering. Only the structure of the graphs is used to compute clustering.

Clustering Algorithms: We below list the clustering algorithms that we use within the PASCO pipeline, and we provide details on the graph characteristics they try to optimize. The *Spectral Clustering* algorithm (SC) exploits eigenvectors of the Laplacian matrix. It minimizes a notion of *generalized normalized cut* (gnCut) [31]. A modified version of SC was proposed by [9] to accelerate and reduce memory print and is called *Compressive Spectral Clustering* (CSC). The classical *Louvain* method [6] and its modern variant *Leiden* [32] are both modularity maximization methods. We recall that modularity measures the partition quality by comparing inside/outside community densities [33]. Finally, we also include clustering algorithms that use the Minimization of Description Length (MDL) to either maximize the likelihood of the stochastic block model [5], or the compression of the graph into clusters as per the *infomap* method [4]. These algorithms rely on the *Description Length* (DL) [34] that quantifies the amount of information required to describe the parameters of an SBM adjusted to the observed graph. In Section E.4, we recall the definition of the scores mentioned above: gnCut, modularity, DL, as well as the AMI.

Experimental setting: We evaluate PASCO's impact on computational time and partition quality by comparing clustering methods with and without PASCO. Graphs are clustered using SC, CSC, Louvain, Leiden, MDL, and Infomap, then re-clustered by applying PASCO combined with the same clustering algorithms. Except for CSC, which we re-implemented in Python due to it being only available in Matlab, standard implementations were used. The compression factor ρ is fixed at 10, and coarsening repetitions R vary in $\{1, 3, 5, 10, 15\}$. Partition quality is assessed using AMI for agreement with ground truth and intrinsic scores like modularity, gnCut, and DL. For these scores, the relative difference $(s_{\text{est}} - s_{\text{true}})/s_{\text{true}}$ is reported, where s_{est} and s_{true} are the scores of the, respectively, estimated and ground-truth partitions. We also compute the total computational time for each method and report the ratio between

PASCO running time and the one of the standalone clustering method. See the paragraph below for some precision on how computational time is measured. The results are in Figure 7. To guide the reader, we extract first a simpler example in Figure E5. For completeness, as we present relative measures for both computational time and clustering quality, we provide the numerical results in the tables in Section E.5.

Measuring Computational time: To measure computational time, we record the duration from start to finish. Since PASCO relies on parallelization, we run each clustering method on a single core, whether used alone or with PASCO. This ensures that PASCO's parallelization does not conflict with that of the clustering methods, allowing us to evaluate PASCO's speedup effect. Consequently, clustering methods optimized with parallelization will have diminished performance, so comparisons between algorithms based on computational time are not meaningful. However, we can effectively analyze PASCO's impact on individual clustering methods. Extending PASCO to support parallelized clustering across multiple cores or machines is beyond the scope of this article.

Results: The experiments show that PASCO improves runtime or clustering quality for most clustering methods and graphs under study, sometimes achieving both. With SC, CSC, or Infomap, PASCO significantly reduces runtime in most cases, often by a factor of 10 or more. For Leiden, runtime improvements are smaller and mostly seen with fewer coarsening repetitions ($R < 5$). With MDL, the effect is lighter and more data-dependent. No runtime improvement is observed with Louvain, as its fast clustering is offset by the additional time for PASCO's coarsening and fusion steps.

PASCO achieves a similar quality in clustering as the methods it uses. This is a notable result, given that it was initially designed for reducing computational costs and this property on the preservation of partition quality is only supported by heuristics. For AMI, which compares partitions to ground truth, PASCO improves results for SC and CSC, while other methods show no significant change. Modularity and Description Length scores tend to be more "regularized" again, with values closer to ground truth scores. The gnCut criterion is generally preserved, except for one case (MDL on `mag`). This preservation may be linked to the ability of the coarsening step to preserve spectral properties (Section 5.1), although further theoretical study would be needed to better understand the reason behind that.

6 Conclusion

We introduce PASCO, a novel approach to accelerate graph clustering algorithms through a coarsening-based strategy. PASCO provides a full pipeline for that, which can be used for a variety of clustering methods. This innovative PASCO approach is built around three main steps: reducing the graph via a fast and empirically structure-preserving random coarsening process, running clustering algorithms in parallel on the coarsened graphs, and combining the resulting partitions using an optimal transport-based fusion technique. The experimental results reported in the present article demonstrate the efficiency of PASCO in significantly reducing computational time while maintaining, or even enhancing, the quality of the resulting partitions, as shown on both synthetic and real-world graph datasets. As PASCO is a modular

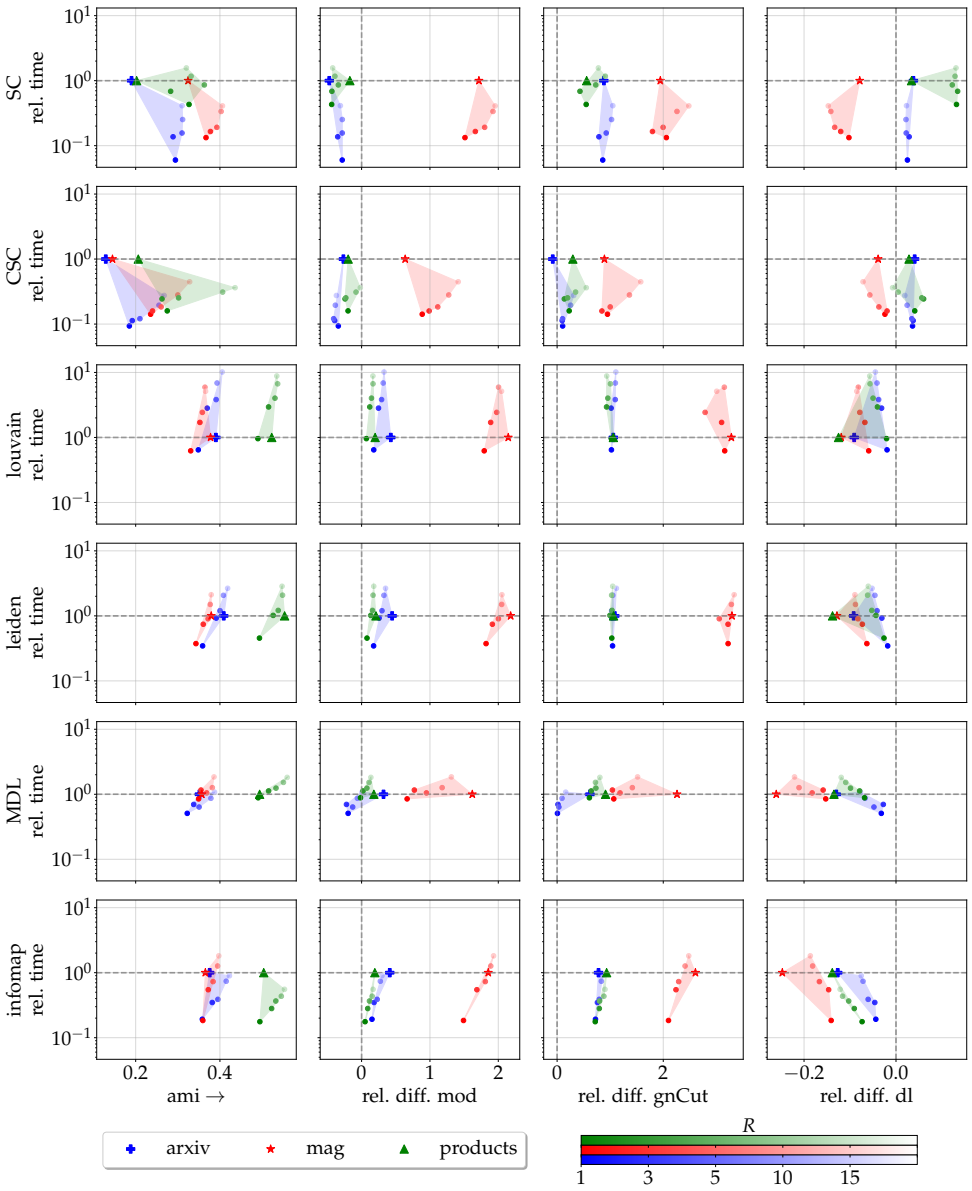


Fig. 7: Results of the real graphs experiment. Rows correspond to clustering methods, while columns correspond to clustering quality measures. On y-axes, we represent the computational time relative to the one of the standalone method. On the x-axis we display either the AMI with the true partition or the relative quality score difference. This experiment is performed with different graphs (colors) and for different numbers of repetitions of the coarsening R (transparency). The reduction factor is $\rho = 10$.

framework, it can be seamlessly integrated with various clustering algorithms, making it a versatile tool to tackle large-scale graph clustering challenges.

Acknowledgements

This work has been supported by the DATAREDUX project, ANR19-CE46-0008, and the ANR DARLING project ANR-19-CE48-0002. This project was supported in part by the AllegroAssai ANR project ANR-19-CHIA0009.

References

[1] Fortunato, S., Hric, D.: Community detection in networks: A user guide. Physics Reports **659** (2016). Community detection in networks: A user guide

[2] Karataş, A., Şahin, S.: Application areas of community detection: A review. In: 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT) (2018)

[3] Von Luxburg, U.: A tutorial on spectral clustering. Statistics and computing **17** (2007)

[4] Rosvall, M., Bergstrom, C.T.: Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. PLOS ONE **6** (2011)

[5] Peixoto, T.P.: Efficient monte carlo and greedy heuristic for the inference of stochastic block models. Physical Review E **89**(1) (2014)

[6] Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10) (2008)

[7] Pourkamali-Anaraki, F.: Scalable spectral clustering with nyström approximation: Practical and theoretical aspects. IEEE Open Journal of Signal Processing **1** (2020)

[8] Boutsidis, C., Kambadur, P., Gittens, A.: Spectral clustering via the power method-provably. In: ICML (2015)

[9] Tremblay, N., Puy, G., Gribonval, R., Vandergheynst, P.: Compressive spectral clustering. ICML (2016)

[10] Spielman, D.A., Teng, S.-H.: Spectral sparsification of graphs. SIAM Journal on Computing **40**(4) (2011)

[11] Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing

(2008)

[12] Tremblay, N., Loukas, A.: Approximating spectral clustering via sampling: a review. Sampling Techniques for Supervised or Unsupervised Tasks (2020)

[13] Yan, D., Huang, L., Jordan, M.I.: Fast approximate spectral clustering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2009)

[14] Li, Y., Nie, F., Huang, H., Huang, J.: Large-scale multi-view spectral clustering via bipartite graph. In: AAAI (2015)

[15] Hendrickson, B., Leland, R.W., et al.: A multi-level algorithm for partitioning graphs. SC **95**(28) (1995)

[16] Dhillon, I.S., Guan, Y., Kulis, B.: Weighted graph cuts without eigenvectors a multilevel approach. IEEE transactions on pattern analysis and machine intelligence **29**(11) (2007)

[17] Loukas, A.: Graph reduction with spectral and cut guarantees. Journal of Machine Learning Research **20**(116) (2019)

[18] Ghosh, J., Acharya, A.: Cluster ensembles: Theory and applications. Data Clustering (2018)

[19] Wu, J., Liu, H., Xiong, H., Cao, J., Chen, J.: K-means-based consensus clustering: A unified view. IEEE transactions on knowledge and data engineering **27**(1) (2014)

[20] Fred, A.: Finding consistent clusters in data partitions. In: International Workshop on Multiple Classifier Systems (2001)

[21] Peyré, G., Cuturi, M., et al.: Computational optimal transport: With applications to data science. Foundations and Trends® in Machine Learning **11**(5-6) (2019)

[22] Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on scientific Computing **20**(1) (1998)

[23] Li, J., Seo, B., Lin, L.: Optimal transport, mean partition, and uncertainty assessment in cluster analysis. Statistical Analysis and Data Mining: The ASA Data Science Journal **12**(5) (2019)

[24] Flamary, R., Courty, N., Gramfort, A., Alaya, M.Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., et al.: Pot: Python optimal transport. JMLR **22**(78) (2021)

[25] Cuturi, M., Doucet, A.: Fast computation of wasserstein barycenters. In: ICML (2014)

[26] Ayad, H.G., Kamel, M.S.: On voting-based consensus of cluster ensembles. Pattern Recognition **43**(5) (2010)

[27] Blondel, M., Seguy, V., Rolet, A.: Smooth and sparse optimal transport. In: AISTATS (2018)

[28] Loukas, A., Vandergheynst, P.: Spectrally approximating large graphs with smaller graphs. In: ICML (2018)

[29] Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: ICML (2009)

[30] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687 (2020)

[31] Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2004)

[32] Traag, V.A., Waltman, L., Van Eck, N.J.: From louvain to leiden: guaranteeing well-connected communities. Scientific reports **9**(1) (2019)

[33] Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical review E **69** (2004)

[34] Rissanen, J.: Information and complexity in statistical modeling (2007)

[35] Liu, D.C., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Mathematical programming **45**(1) (1989)

[36] Newman, M.E.J.: Networks : an introduction (2010)

[37] Quemener, E., Corvellec, M.: Sidus—the solution for extreme deduplication of an operating system. Linux Journal **2013**(235), 3 (2013)

[38] Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. AAAI (2015)

Appendix A Pseudo-codes

Algorithm 1 Coarsening algorithm: global

Input: Adjacency matrix $\mathbb{A}$ of graph $\mathbb{G}$ of size N.
Compression facor ρ.
Target graph size $n = \lfloor N/\rho \rfloor$.
$A^{(0)} \leftarrow \mathbb{A}$
$h \leftarrow (1, \dots, |\mathbb{A}|)$
$\ell \leftarrow 0$
while $A^{(\ell)}$ has more than n nodes **do**
    Coarsen $A^{(\ell)}$ to obtain $h^{(\ell)}$ and $A^{(\ell+1)}$ with Algorithm 2
    Update h given $h^{(\ell)}$
    $\ell \leftarrow \ell + 1$
end while
return $A^{(\ell)}$, h

Algorithm 2 Coarsening algorithm: one level

Input: Current adjacency matrix $A^{(\ell)}$, target graph size n.
$V_a \leftarrow \{1, \dots, n^{(\ell)}\}$ // Initialize the set of available nodes.
$h^(\ell) \leftarrow (1, \dots, n^{(\ell)})$ // Initialize the coarsening table.
while $V_a \neq \emptyset$ **do** // while there are available nodes
    Choose u a node in V_a uniformly at random
    Choose v a neighbor of u uniformly at random
    $h_u^{(\ell)} \leftarrow h_v^{(\ell)}$ // Put u and v into the same hypernode
    $V_a = V_a \backslash \{u, v\}$ // Remove u and v from the available nodes
    If target size is reached **do break**
end while
Relabel $h^{(\ell)}$ and compute $A^{(\ell+1)}$
return $A^{(\ell+1)}$, $h^{(\ell)}$.

Algorithm 3 Alignment & Fusion algorithm

Partitions $P_1, \cdots, P_R$, number of clusters $\overline{k}$ and initial reference $\overline{P}$.
while $\overline{P}$ has not converged **do**
  for $r \in [\![R]\!]$ **do** // Alignment step
    Find Q_r so that $P_r Q_r$ is aligned on $\overline{P}$ (*i.e.*, solve the OT problem (2) in Section 4)
  end for
  Update $\overline{P}$ using majority vote on $(P_r Q_r)_{r \in [\![R]\!]}$, as in (3) in Section 4. // Fusion step
end while
return $\overline{P} \in \{0,1\}^{N \times \overline{k}}$

Appendix B Finding consensus between partitions

In this section we describe the different approaches that we investigated for aligning different partitions into a reference partition. We recall that the problem reformulates as finding the partition $\overline{P}$ that best agrees with all (P_r) with $r \in [\![R]\!]$. Methods from the clustering ensemble literature already propose to solve such a problem. Mathematically, they amount to find a Frechet mean of the partitions (P_r) for various notions of divergence $\mathcal{D}$ between partitions

$$P \in \underset{\overline{P} \in \mathcal{P}_{N,\overline{k}}}{\arg\min} \frac{1}{R} \sum_{i=1}^{R} \mathcal{D}(P_r, \overline{P})\,. \tag{B1}$$

Depending on the choice of divergence $\mathcal{D}$, we can obtain different consensus methods, as detailed below.

Linear regression and many-to-one

These two methods are based on a similar notion of divergence between partitions. Given two partitions $P \in \mathcal{P}_{N,k}, \overline{P} \in \mathcal{P}_{N,\overline{k}}$, it is defined as

$$\mathcal{D}(P, \overline{P}) = \min_{Q \in \mathcal{Q}(k,\overline{k})} \|PQ - \overline{P}\|_F^2 \tag{B2}$$

Depending on the choice for the set $\mathcal{Q}(k, \overline{k})$ we get different methods:

- When we simply set $\mathcal{Q}(k, \overline{k}) = \mathcal{Q}_{\text{lin-reg}}(k, \overline{k}) = \mathbb{R}^{k \times \overline{k}}$, the optimal matrix Q is given by (see Lemma 2)

$$Q_{\text{lin-reg}} = (P^\top P)^{-1} P^\top \overline{P} = \text{diag}(P^\top \mathbf{1}_N)^{-1} P^\top \overline{P}\,, \tag{B3}$$

  where $\text{diag}(P^\top \mathbf{1}_N)^{-1}$ corresponds to the diagonal matrix containing the inverse of the cluster sizes. This realignment is proposed in [26]. Even though it allows to compute the divergence $\mathcal{D}(P, \overline{P})$, this matrix yields a "re-aligned partition" $PQ_{\text{lin-reg}}$

which is only a "soft-partition", with elements in $[0, 1]$. We refer to this solution as `lin-reg`.

- If one wants to obtain a true partition (with elements in $\{0, 1\}$), a solution is to restrict the matrix in $\mathcal{Q}(k, \overline{k})$ to send each cluster of P to at most one cluster of $\overline{P}$. For that we define $\mathcal{Q}_{\text{many-to-one}}(k, \overline{k}) = \{Q \in \{0, 1\}^{k \times \overline{k}}, Q\mathbf{1}_{\overline{k}} = \mathbf{1}_k\}$. The solution of (B2) with $\mathcal{Q}_{\text{many-to-one}}(k, \overline{k})$ is given (see Lemma 3) by row-bin$(Q_{\text{lin-reg}})$, where row-bin is the operator that returns a binary matrix of same shape, where each row contains only zeros except at the position of the maximum in the corresponding row of the input matrix. We refer to this solution as `many-to-one`.

OT-based method

To prevent empty clusters in the realigned partition, we also consider $\mathcal{Q}(k, \overline{k}) = \mathcal{Q}_{\text{ot}}(k, \overline{k})$ in (B2). With these constraints, the alignment problem becomes an OT problem which can be related to a specific *quadratic* regularized OT problem [27] which admits efficient convex solvers, as detailed in the following lemma (proof can be found in Section D).

Lemma 1. *Let $P \in \mathcal{P}_{N,k}$ and $D = \text{diag}(P^\top \mathbf{1}_N)$. Then problem (B2) with $\mathcal{Q}(k, \overline{k}) = \mathcal{Q}_{\text{ot}}(k, \overline{k})$ is equivalent to the problem*

$$\min_{Q \in \mathcal{Q}_{\text{ot}}(k, \overline{k})} \|D^{\frac{1}{2}} Q\|_F^2 - 2\langle C, Q \rangle \,, \tag{B4}$$

where $C = P^\top \overline{P}$. Assuming no empty cluster in the partition P, the solution of Equation (B4) is unique and can be solved by considering the dual problem

$$\max_{\mu \in \mathbb{R}^k,\ \nu \in \mathbb{R}^{\overline{k}}} \frac{1}{k} \sum_i \mu_i + \frac{1}{\overline{k}} \sum_j \nu_j - \frac{1}{4} \|D^{-\frac{1}{2}} [\mu \oplus \nu + 2C]_+\|_F^2 \,, \tag{B5}$$

where $\mu \oplus \nu := (\mu_i + \nu_j)_{ij}$ and for any matrix A, $[A]_+ := (\max\{A_{ij}, 0\})_{ij}$. More precisely, the optimal solution $Q^\star$ of Equation (B4) can be written as $Q^\star = \frac{1}{2} D^{-1} [\mu^\star \oplus \nu^\star + 2C]_+$ where $(\mu^\star, \nu^\star)$ is the optimal solution of Equation (B5).

Building upon this result we can solve Equation (B4) by tackling the dual Equation (B5) which is a convex unconstrained problem of two variables (maximization of a concave function). This expression allow us to use any convex solver, and, as suggested in [27], we rely on L-BFGS [35] that we find particularly effective in practice. We call this alignment procedure `quad-ot` which has roughly a $\mathcal{O}(NK^2)$ complexity.

Remark 1. *The only difference between Equation (B4) and standard quadratic OT problem of the form $\min_Q \langle M, Q \rangle + \frac{\gamma}{2} \|Q\|_F^2$ is that in our case the regularization term is $\|D^{1/2} Q\|_F^2$. This is equivalent to consider a Mahalanobis type regularization instead of a ℓ_2^2 one.*

Solving for the barycenter

To solve the barycenter problem in (B1) with a distance of the form of (B2), we alternate between finding the alignment matrices Q_r as explained above and updating the reference $\overline{P}$ as in (3). This corresponds to an alternating minimization algorithm,

where we alternate between (i) realigning the R partitions on the reference and (ii) updating the reference. The reference update is based on Lemma 4 which states that finding the closest partition matrix to a set a (realigned) matrices is given by the majority-vote update, see Eq. (3).

Appendix C Other edge sampling rules

In this section, we present edge sampling rules that we came up with while trying to design an fast coarsening procedure. They were not satisfying but we choose to present them here and explain their drawbacks, as we believe it is informative to better understand the coarsening algorithm we propose in the end. To the best of our knowledge, these strategies were not considered systematically in previous works to coarsen graphs.

Uniform edge sampling

This approach might be the most natural one. It consists in choosing edges uniformly at random in the graph and collapsing them. Doing so favors edges incident to high degree nodes, resulting after collapsing to an even higher degree hypernode. This amplification phenomenon yields an unbalanced final coarsened graph that contains one huge hypernode and all other hypernodes containing only a few nodes. This is problematic as it would not express well the community structure of the initial graph.

Uniform edge sampling with marked neighboring edges

This approach fixes the above issue. To avoid collapsing onto almost always the same hypernode, when an edge (i, j) is collapsed, we mark the edges incident to nodes i and j, and we sample edges uniformly at random among unmarked edges. This solves the issue of unbalancedness. However, the algorithm quickly runs out of unmarked edges and forces the early computation of the next current coarsened graph. While this can be done with sparse matrix products between the adjacency matrix and the matrix encoding the composition of the hypernodes, it remains costly and should be avoided as much as possible.

Uniform edge sampling with marked visited nodes

Here, we want to relax the limit imposed by the previous approach with marked edges. First, recall that sampling an edge uniformly at random is equivalent to sampling a node i with a probability proportional to its degree and sampling a neighbor j uniformly, see *e.g.*, [36, Section 6.14]. So instead of discarding edge (i, j) whenever either i or j has been used in a previous collapse, a natural relaxation is to reject the edge (i, j) only when i has been previously involved in a collapse, *irrespective of whether j was involved or not in such a collapse*. Simulations showed that when using this sampling strategy (in step 6 of Algorithm 2) to generate coarsening tables h^ℓ, the hypernodes size distribution was similar to the one with the edge marking strategy, but resulted in much less intermediate coarsened graph reconstructions.

Uniform node sampling with marked visited nodes

A final improvement to speedup the sampling procedure is to sample the first node i *uniformly at random* among non-visited nodes instead of according to its degree (the second node j being still draw uniformly among the neighbors of i). Computationally, this avoids updating the degrees after each collapse and further speeds up the coarsening procedure. The impact of sampling uniformly instead of according to the degrees stays limited thanks to the friendship paradox. This results in Algorithm 2.

Appendix D Relegated theoretical results

Lemma 2. *Let $P \in \mathcal{P}_{N,k}$ be a partition matrix (Definition 1), and assume that $\forall i \in [\![k]\!], [P^\top \mathbf{1}_N]_i \neq 0$. Then, for any integer $\overline{k}$ and any matrix $\overline{P} \in \mathbb{R}^{N\times\overline{k}}$, with $\mathcal{Q}(k,\overline{k}) = \mathbb{R}^{k\times\overline{k}}$, $Q^\star = \operatorname{diag}(P^\top \mathbf{1}_N)^{-1} P^\top \overline{P}$ is an optimal solution to problem (B2).*

Proof. Denoting $f(Q) = \|PQ - \overline{P}\|_F^2$. Since P is a partiton matrix its columns have pairwise disjoint support and we have $P^\top P = \operatorname{diag}(P^\top \mathbf{1}_N)$ hence

$$f(Q) = \|\overline{P}\|_F^2 - 2\langle Q, P^\top \overline{P}\rangle + \operatorname{tr}(Q^\top P^\top P Q) = \|\overline{P}\|_F^2 + \|\operatorname{diag}(P^\top \mathbf{1}_N)^{\frac{1}{2}} Q\|_F^2 - 2\langle Q, P^\top \overline{P}\rangle\,. \tag{D6}$$

The optimization problem is convex, setting the gradient of f to zero gives the solution. □

Lemma 3. *Let $P \in \mathcal{P}_{N,k}$ be a partition matrix (Definition 1), $\overline{P} \in \mathbb{R}^{N\times\overline{k}}$, and $\mathcal{Q}(k,\overline{k}) = \{Q \in \{0,1\}^{k\times\overline{k}} : Q\mathbf{1}_{\overline{k}} = \mathbf{1}_k\}$. Then $Q^\star$ defined by*

$$\forall i \in [\![k]\!], Q^\star_{ij} = \begin{cases} 1 & j \in \operatorname{argmax}_{p\in[\![\overline{k}]\!]} [P^\top \overline{P}]_{ip} \\ 0 & \textit{otherwise} \end{cases}, \tag{D7}$$

is an optimal solution to problem (B2).

Proof. Denoting $f(Q) = \|PQ - \overline{P}\|_F^2$ and using that P is a partition matrix, that $Q_{ij} \in \{0,1\}$ (hence $Q_{ij}^2 = Q_{ij}$) and $Q\mathbf{1}_{\overline{k}} = \mathbf{1}_k$ we can rewrite f as

$$\begin{aligned} f(Q) &= \|\overline{P}\|_F^2 + \|\operatorname{diag}(P^\top \mathbf{1}_N)^{\frac{1}{2}} Q\|_F^2 - 2\langle Q, P^\top \overline{P}\rangle \\ &= \|\overline{P}\|_F^2 + \sum_{ij} [P^\top \mathbf{1}_N]_i Q_{ij}^2 - 2\langle Q, P^\top \overline{P}\rangle \\ &= \|\overline{P}\|_F^2 + \sum_{i=1}^{k}\sum_{j=1}^{\overline{k}} [P^\top \mathbf{1}_N]_i Q_{ij} - 2\langle Q, P^\top \overline{P}\rangle \\ &= \|\overline{P}\|_F^2 + \sum_{i=1}^{k} [P^\top \mathbf{1}_N]_i (\sum_{j=1}^{\overline{k}} Q_{ij}) - 2\langle Q, P^\top \overline{P}\rangle \\ &= \|\overline{P}\|_F^2 + \sum_{i=1}^{k} [P^\top \mathbf{1}_N]_i - 2\langle Q, P^\top \overline{P}\rangle \\ &= \|\overline{P}\|_F^2 + N - 2\langle Q, P^\top \overline{P}\rangle \end{aligned} \tag{D8}$$

Denoting $C = -P^\top \overline{P}$, a solution to problem (B2) can thus be found by solving

$$\min_{Q \in \{0,1\}^{k \times \overline{k}} : Q\mathbf{1}_{\overline{k}} = \mathbf{1}_k} \langle Q, C\rangle \,. \tag{D9}$$

Now Equation (D9) is an optimization problem that decouples with respect to the rows of Q, *i.e.* there are k independent problems per row of Q. For each row $i \in [\![k]\!]$, a solution can be found by choosing any column index j such that $j \in \operatorname{argmin}_{p \in [\![\overline{k}]\!]} C_{ip}$ This condition is equivalent to find j such that $j \in \operatorname{argmax}_{p \in [\![\overline{k}]\!]} [P^\top \overline{P}]_{ip}$. □

Lemma 4. *Let $P^{(1)}, \cdots, P^{(R)}$ where each $P^{(r)} \in \mathbb{R}^{N \times \overline{k}}$. Then a solution to*

$$\min_{\overline{P} \in \mathcal{P}_{N,\overline{k}}} \frac{1}{R} \sum_{r=1}^{R} \|\overline{P} - P^{(r)}\|_F^2 \,. \tag{D10}$$

is given by

$$\forall i \in [\![N]\!],\ [\overline{P}]_{ij} = \begin{cases} 1 & j \in \operatorname*{argmax}_{p \in [\![\overline{k}]\!]} [\sum_{r=1}^{R} P^{(r)}]_{ip} \\ 0 & \textit{otherwise} \end{cases} \tag{D11}$$

Now let $P^{(1)}, \cdots, P^{(R)}$ where each $P^{(r)} \in \mathbb{R}^{N \times k_r}$ and $Q^{(1)}, \cdots, Q^{(R)}$ be coupling matrices such that each $Q^{(r)} \in \mathcal{Q}_{\text{ot}}(k_r, \overline{k})$. Then a solution to

$$\min_{\overline{P} \in \mathcal{P}_{N,\overline{k}}} \frac{1}{R} \sum_{r=1}^{R} \sum_{i,j=1}^{k,\overline{k}} \|P_{:,i}^{(r)} - \overline{P}_{:,j}\|_2^2 Q_{i,j}^{(r)} \,. \tag{D12}$$

is given by

$$\forall i \in [\![N]\!],\ [\overline{P}]_{ij} = \begin{cases} 1 & j \in \underset{p \in [\![\overline{k}]\!]}{\operatorname{argmax}} \left[\sum_{r=1}^{R} P^{(r)} Q^{(r)}\right]_{ip} \\ 0 & \textit{otherwise} \end{cases} \tag{D13}$$

Proof. For the first point, take $\overline{P}$ in the constraints. Since it is a partition matrix we have $\|\overline{P}\|_F^2 = \sum_{ij} \overline{P}_{ij}^2 = \sum_{ij} \overline{P}_{ij} = \mathbf{1}_N^\top \overline{P} \mathbf{1}_{\overline{k}} = N$. Thus problem (D10) is equivalent to

$$\min_{\overline{P} \in \mathcal{P}_{N,\overline{k}}} \langle \overline{P}, -\sum_{r=1}^{R} P^{(r)} \rangle \,. \tag{D14}$$

As detailed in the proof of Lemma 3 a solution can be found by choosing the index of the column j such that $j \in \operatorname{argmin}_{p \in [\![\overline{k}]\!]} [-\sum_{r=1}^{R} P^{(r)}]_{ip}$ which concludes the proof for the first point. For the second point, we use that $\overline{P}$ is a partition matrix and $Q^{(r)}$ are coupling matrices so that

$$\begin{aligned} \sum_{r=1}^{R} \sum_{i,j=1}^{k,\overline{k}} \|P_{:,i}^{(r)} - \overline{P}_{:,j}\|_2^2 Q_{i,j}^{(r)} &= \sum_{r=1}^{R} \sum_{i,j=1}^{k,\overline{k}} (\|P_{:,i}^{(r)}\|_2^2 - 2\langle P_{:,i}^{(r)}, \overline{P}_{:,j} \rangle + \|\overline{P}_{:,j}\|_2^2) Q_{i,j}^{(r)} \\ &= \text{cte} - 2 \sum_{r=1}^{R} \sum_{i,j=1}^{k,\overline{k}} \langle P_{:,i}^{(r)}, \overline{P}_{:,j} \rangle Q_{i,j}^{(r)} + \sum_{r=1}^{R} \sum_{i,j=1}^{k,\overline{k}} \|\overline{P}_{:,j}\|_2^2 Q_{i,j}^{(r)} \\ &= \text{cte} - 2 \sum_{r=1}^{R} \langle \overline{P}, P^{(r)} Q^{(r)} \rangle + \sum_{r=1}^{R} \sum_{j=1}^{\overline{k}} \|\overline{P}_{:,j}\|_2^2 \sum_{i=1}^{k} Q_{i,j}^{(r)} \\ &= \text{cte} - -2 \sum_{r=1}^{R} \langle \overline{P}, P^{(r)} Q^{(r)} \rangle + \frac{R}{\overline{k}} \|\overline{P}\|_F^2 \,. \end{aligned} \tag{D15}$$

Using that $\|\overline{P}\|_F^2 = N$ as previously proved, we get that the problem is equivalent to

$$\min_{\overline{P} \in \mathcal{P}_{N,\overline{k}}} \langle \overline{P}, -\sum_{r=1}^{R} P^{(r)} Q^{(r)} \rangle \,, \tag{D16}$$

hence the result. □

Lemma 1. *Let $P \in \mathcal{P}_{N,k}$ and $D = \operatorname{diag}(P^\top \mathbf{1}_N)$. Then problem (B2) with $\mathcal{Q}(k,\overline{k}) = \mathcal{Q}_{\text{ot}}(k,\overline{k})$ is equivalent to the problem*

$$\min_{Q \in \mathcal{Q}_{\text{ot}}(k,\overline{k})} \|D^{\frac{1}{2}} Q\|_F^2 - 2\langle C, Q \rangle \,, \tag{B4}$$

where $C = P^\top \overline{P}$. *Assuming no empty cluster in the partition* P, *the solution of Equation* (B4) *is unique and can be solved by considering the dual problem*

$$\max_{\mu \in \mathbb{R}^k,\ \nu \in \mathbb{R}^{\overline{k}}} \frac{1}{k}\sum_i \mu_i + \frac{1}{\overline{k}}\sum_j \nu_j - \frac{1}{4}\|D^{-\frac{1}{2}}[\mu \oplus \nu + 2C]_+\|_F^2\,, \tag{B5}$$

where $\mu \oplus \nu := (\mu_i + \nu_j)_{ij}$ *and for any matrix* A, $[A]_+ := (\max\{A_{ij}, 0\})_{ij}$. *More precisely, the optimal solution* $Q^\star$ *of Equation* (B4) *can be written as* $Q^\star = \frac{1}{2}D^{-1}[\mu^\star \oplus \nu^\star + 2C]_+$ *where* $(\mu^\star, \nu^\star)$ *is the optimal solution of Equation* (B5).

Proof. We will prove a slightly more general result by considering the problem

$$\min_{Q \in \mathcal{Q}_{\text{ot}}(k_r, \overline{k})} \langle M, Q\rangle + \frac{\gamma}{2}\|L^{\frac{1}{2}}Q\|_F^2\,, \tag{D17}$$

where $M \in \mathbb{R}^{p \times p_{\text{ref}}}, \gamma > 0$ and L is a symmetric positive definite matrix. We note $a = \frac{1}{k}\mathbf{1}_k, b = \frac{1}{\overline{k}}\mathbf{1}_{\overline{k}}$. We will then apply to $M = -2C, \gamma = 2$ and $L = D$ which is symmetric positive definite when there is no empty clusters (since in this case $\forall i \in [\![K]\!], D_{ii} \neq 0$). Most of our calculus are adapted from [27]. First, since L is a symmetric positive definite matrix, the problem Equation (D17) is a strongly convex problem, thus it admits a unique solution.

To look at the dual of Equation (D17), we consider the Lagrangian

$$\begin{aligned}\mathcal{L}(Q, \mu, \nu, \Gamma) &= \langle M, Q\rangle + \frac{\gamma}{2}\|L^{\frac{1}{2}}Q\|_F^2 + \langle \mu, a - Q\mathbf{1}_k\rangle + \langle \nu, b - Q^\top \mathbf{1}_{\overline{k}}\rangle - \langle \Gamma, Q\rangle \\ &= \langle M, Q\rangle + \frac{\gamma}{2}\|L^{\frac{1}{2}}Q\|_F^2 - \langle Q, \mu\mathbf{1}_{\overline{k}}^\top + \mathbf{1}_k\nu^\top + \Gamma\rangle + \langle \mu, a\rangle + \langle \nu, b\rangle\,,\end{aligned} \tag{D18}$$

where Γ is the variable accounting for the non-negativity constraints on Q. We have

$$\nabla_Q \mathcal{L}(Q, \mu, \nu, \Gamma) = 0 \iff M + \gamma LQ - \mu \oplus \nu - \Gamma = 0\,. \tag{D19}$$

This is statisfied when $Q = Q^\star = \frac{1}{\gamma}L^{-1}(\Gamma + \mu \oplus \nu - M)$. Moreover,

$$\begin{aligned}\langle M, Q^\star\rangle - \langle Q^\star, \mu \otimes \nu + \Gamma\rangle &= -\langle Q^\star, \mu \otimes \nu + \Gamma - M\rangle \\ &= -\langle \frac{1}{\gamma}L^{-1}(\Gamma + \mu \oplus \nu - M), \mu \otimes \nu + \Gamma - M\rangle \\ &= -\frac{1}{\gamma}\|L^{-\frac{1}{2}}(\Gamma + \mu \oplus \nu - M)\|_F^2\,.\end{aligned} \tag{D20}$$

Thus

$$\begin{aligned}\langle M, Q^\star\rangle - \langle Q^\star, \mu\otimes\nu+\Gamma\rangle + \frac{\gamma}{2}\|L^{\frac{1}{2}}Q^\star\|_F^2 &= -\frac{1}{\gamma}\|L^{-\frac{1}{2}}(\Gamma+\mu\oplus\nu-M)\|_F^2 \\ &+ \frac{\gamma}{2}\|L^{\frac{1}{2}}(\frac{1}{\gamma}L^{-1}(\Gamma+\mu\oplus\nu-M))\|_F^2 \\ &= -\frac{1}{\gamma}\|L^{-\frac{1}{2}}(\Gamma+\mu\oplus\nu-M)\|_F^2 \\ &+ \frac{1}{2\gamma}\|L^{-\frac{1}{2}}(\Gamma+\mu\oplus\nu-M)\|_F^2 \\ &= -\frac{1}{2\gamma}\|L^{-\frac{1}{2}}(\Gamma+\mu\oplus\nu-M)\|_F^2\end{aligned} \tag{D21}$$

Hence

$$\mathcal{L}(Q^\star,\mu,\nu,\Gamma) = -\frac{1}{2\gamma}\|L^{-\frac{1}{2}}(\Gamma+\mu\oplus\nu-M)\|_F^2 + \langle\mu,a\rangle + \langle\nu,b\rangle\,. \tag{D22}$$

Now we solve the problem over Γ that is we maximize the problem

$$\max_{\Gamma\geq 0}\ \mathcal{L}(Q^\star,\mu,\nu,\Gamma)\,, \tag{D23}$$

where ≥ 0 should be understood pointwise. This is equivalent to

$$\min_{\Gamma\geq 0}\|L^{-\frac{1}{2}}(\Gamma-A)\|_F^2\,. \tag{D24}$$

where $A := M-\mu\oplus\nu$. Writing $L = U\Delta U^\top$ where $\Delta = \mathrm{diag}(d_1,\cdots,d_n)$ with $d_i>0$, Equation (D24) equivalently writes

$$\min_{\Gamma\geq 0}\|\Delta^{-\frac{1}{2}}\Gamma - \Delta^{-\frac{1}{2}}A\|_F^2\,. \tag{D25}$$

With a change of variable $\tilde{\Gamma} = \Delta^{-1/2}\Gamma \geq 0$ this is equivalent to

$$\min_{\tilde{\Gamma}\geq 0}\|\tilde{\Gamma} - \Delta^{-\frac{1}{2}}A\|_F^2\,, \tag{D26}$$

whose minimum is given by $\tilde{\Gamma} = [\Delta^{-\frac{1}{2}}A]_+ = \Delta^{-\frac{1}{2}}[A]_+$ since $\Delta^{-\frac{1}{2}}$ is a diagonal matrix with positive entries. Thus the solution of (D24) is given by $\Gamma = \Delta^{1/2}\tilde{\Gamma} = \Delta^{1/2}[\Delta^{-1/2}A]_+ = [A]_+$. Also $[A]_+ - A = [-A]_+$. Thus

$$\min_{\Gamma\geq 0}\|L^{-\frac{1}{2}}(\Gamma-A)\|_F^2 = \|L^{-\frac{1}{2}}[-A]_+\|_F^2 = \|L^{-\frac{1}{2}}[\mu\oplus\nu-M]_+\|_F^2 \tag{D27}$$

Hence the dual problem of Equation (D17) is given by $\max_{\mu,\nu} \mathcal{L}(Q^\star, \mu, \nu, [-A]_+)$ which is

$$\max_{\mu,\nu} \langle \mu, a \rangle + \langle \nu, b \rangle - \frac{1}{2\gamma} \| L^{-\frac{1}{2}} (\mu \oplus \nu - M)_+ \|_F^2 \,. \tag{D28}$$

Applying this to $M = -2C, \gamma = 2$ and $L = D$ concludes. □

Appendix E Experiments details and extra results

E.1 Details about the implementation

The PASCO implementation relies on the `POT` library [24] for the fusion part. The heaviest experiments were performed with the support of the Centre Blaise Pascal's IT test platform at ENS de Lyon (Lyon, France) that operates the SIDUS solution [37]. We use an *Intel Xeon Gold 5218* machine.

E.2 Details of the coarsening experiment.

Table E1 provide some characteristics of the real graphs used in Figure 3.

	# nodes	# edges	avg degree	assortativity	avg clustering coef
Yeast	1.5k	1.9k	2	-0.21	0.07
Minnesota	2.6k	3.3k	2	-0.18	0.02
Airfoil	4.3k	12.3k	5	0.32	0.41

Table E1: Some characteristics of the real graphs used in Figure 3, extracted from [38].

The RSA experiments in Section 5.1 tested the conservation of spectral properties by coarsening algorithms, including PASCO. Figure E1 complete the initial figure Figure 3 with SBM graphs. Graphs were drawn under the SSBM$(1000, k, 2, \alpha)$, for $(k = 10, \alpha = 1/k)$, $(k = 100, \alpha = 1/k)$ and $(k = 10, \alpha = 1/(2k))$. Below, in Figure E2, we display computational times of the coarsening methods.

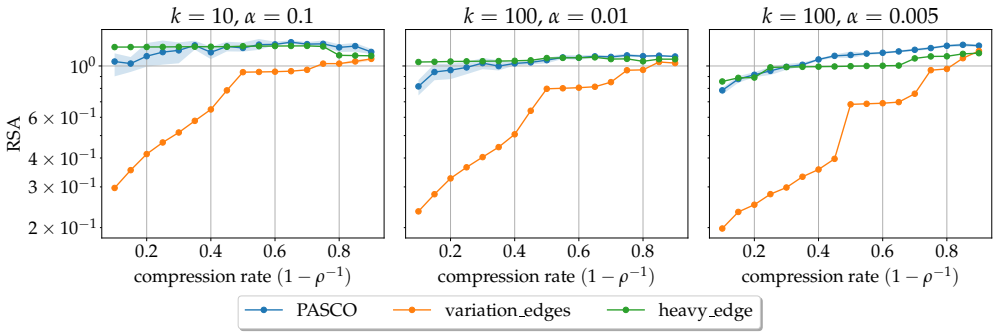


Fig. E1: We represent the RSA (the smaller, the better) of various coarsening schemes (including PASCO), as a function of the compression rate (the higher, the coarser the obtained graph). Shaded areas represent 0.2 upper- and lower-quantiles.

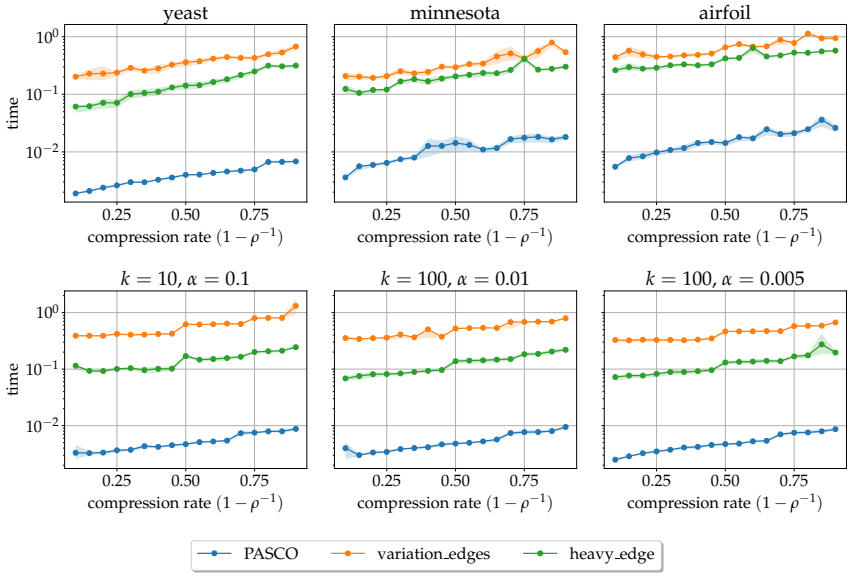


Fig. E2: We represent the computational time of various coarsening schemes (including PASCO), as a function of the compression rate (the higher, the coarser the obtained graph). Shaded areas represent 0.2 upper- and lower-quantiles over the 10 repetitions of the experiment. Top row: we reproduce a part of the experiment of [17, Figure 2] with the same real graphs and added PASCO. Bottom row: same experiment but with random graphs drawn from SSBM$(1000, k, 2, \alpha)$ for $(k = 10, \alpha = 1/k)$, $(k = 100, \alpha = 1/k)$ and $(k = 10, \alpha = 1/(2k))$.

E.3 Additional results on parameters influence

Here, we provide additional results on the experiments of Section 5.3. Figure E3 show the performance of the various alignment methods for different difficulty levels in the SSBM. Parameters are the same as in Figure 5.

In Figure 6, we showed the computational gains of using PASCO. As a sanity check, in Figure E4 we show that the performance *w.r.t.* the quality of the output partition are satisfying. Speed was not achieve at the cost of quality.

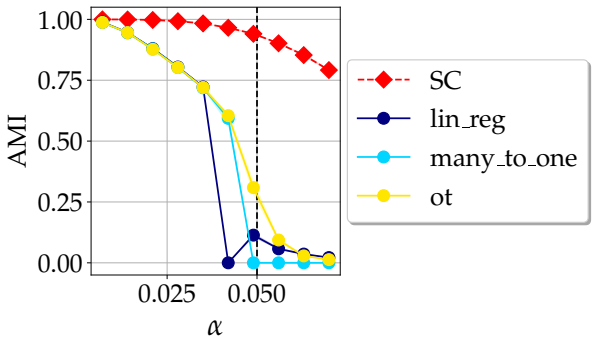


Fig. E3: Influence of the alignment method in PASCO. The AMI score is average over the 10 runs and shaded areas represent 0.2 upper- and lower-quantiles. Dashed lines correspond to PASCO threshold

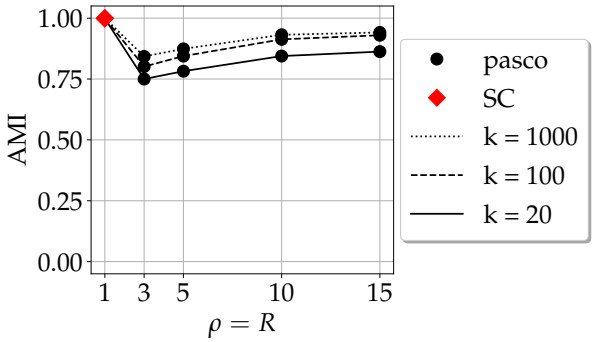


Fig. E4: Quality of the partitions w.r.t. ρ and R, when $R = \rho$. The number of communities vary in $\{20, 100, 1000\}$. AMI values are average over 10 runs and shaded areas represent 0.2 upper- and lower-quantiles.

E.4 Definition of the scores used to evaluate clustering quality

Definition 4 (Adjusted Mutual Information)**.** *The Adjusted Mutual Information (AMI) between two partitions $U = (U_1, \dots, U_k)$ and $V = (V_1, \dots, V_{k'})$ of the set of node $\mathbb{V}$ of size N is given by*

$$AMI(U,V) = \frac{MI(U,V) - \mathbb{E}[MI(U,V)]}{\max(H(U), H(V)) - \mathbb{E}[MI(U,V)]}\,, \tag{E29}$$

where

$$MI(U,V) = \sum_{i=1}^{k}\sum_{j=1}^{k'} P_{UV}(i,j)\log\frac{P_{UV}(i,j)}{P_U(i)P_V(j)}\,, \tag{E30}$$

with $P_U(i) = |U_i|/N$ (similarly for $P_V(j)$), $P_{UV}(i,j) = |U_i \cap V_j|/N$, and $H(U) = -\sum_i P_U(i)\log P_U(i)$ (similarly for $H(V)$). The expected mutual information (MI) $\mathbb{E}[MI(U,V)]$ in (E29) is computed by assuming hyper-geometric distribution for U and V, the parameters being estimated from the partitions. For the sake of conciseness and simplicity, we refer the reader to [29] for the precise formula.

Definition 5 (Generalized Normalized Cut)**.** *Given a graph $\mathbb{G}$ represented by its adjacency or weight matrix $\mathbb{A}$, consider a partition $(V_1, \dots, V_k)$ of the vertex set $\mathbb{V}$ of $\mathbb{G}$. The generalized normalized cut of the partition is defined as*

$$\frac{1}{k}\sum_{j=1}^{k}\frac{\sum_{u\in V_j, v\notin V_j}\mathbb{A}_{u,v}}{\sum_{u\in V_j, v\in\mathbb{V}}\mathbb{A}_{u,v}}\,. \tag{E31}$$

Definition 6 (Modularity)**.** *Given a graph $\mathbb{G}$ represented by its adjacency or weight matrix $\mathbb{A}$, consider a partition $(V_1, \dots, V_k)$ of the vertex set $\mathbb{V}$ of $\mathbb{G}$. Let d_u denote the degree of node u and m the weight of all the edges. Then, the modularity is given by*

$$\frac{1}{2m}\sum_{j=1}^{k}\sum_{u\in V_j, v\in V_j}\left(\mathbb{A}_{u,v} - \frac{d_u d_v}{2m}\right)\,. \tag{E32}$$

Definition 7 (Description Length)**.** *Let $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ be a graph and consider the partition $(V_1, \dots, V_k)$ of the vertex set $\mathbb{V}$. We denote by $e_{i,j}$ the number of edges between V_i and V_j. In mathematical terms, the description length is defined by $\texttt{dl} = \mathcal{S} + \mathcal{L}$, where $\mathcal{S}$ is the entropy of the fitted stochastic block model and $\mathcal{L}$ is the information required to describe the model. Their expressions are given by*

$$\mathcal{S} = |\mathbb{E}| - \frac{1}{2}\sum_{i,j=1}^{k} e_{i,j}\log\left(\frac{e_{i,j}}{|V_i|\,|V_j|}\right)$$
$$\mathcal{L} = |\mathbb{E}|\cdot h\left(\frac{k(k+1)}{2\,|\mathbb{E}|}\right) + |\mathbb{V}|\log(k),$$

with $h(x) = (1+x)\log(1+x) - x\log x$.

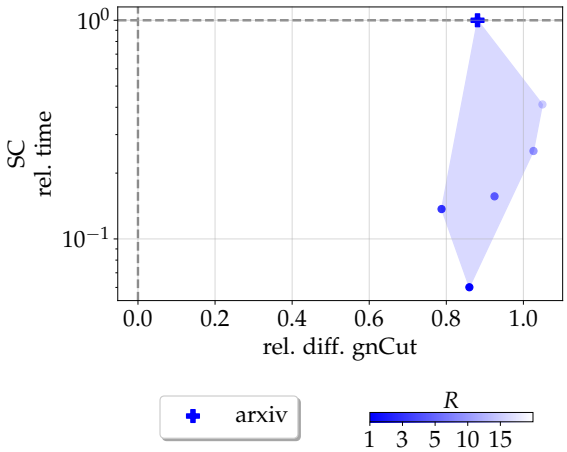


Fig. E5: An example to help read Figure 7. Each plot represents the performance of PASCO associated with a given clustering method (here SC). Specific markers (here a cross) represent the standalone clustering method's performance, while disk markers show PASCO's. Color (here blue) indicate the dataset (here `arxiv`). Performance is evaluated by computational time (y-axis, relative to standalone method, hence specific markers at $y = 1$) and clustering quality (x-axis, measured by AMI with the true partition or relative quality score difference; here, generalized normalized cut). Disk transparency varies with the number of coarsened graphs R (see color bar; $R \in 1, 3, 5, 10, 15$). The shaded area represents the convex hull of points associated to a dataset, ideally trending downward (speedup) and either rightward for AMI or closer to zero for relative quality differences.

E.5 Real data experiments

Table E2 provides a few caracteristics of the real graphs used in Section 5.4.

Name	# nodes	# edges	k	α_{est}	$1/(k-1)$
`arxiv`	169,343	2,315,598	40	0.044	0.026
`mag`	726,664	10,778,888	349	0.031	0.0029
`products`	2,385,902	123,612,734	47	0.028	0.022

Table E2: Name, number of nodes, number of edges and number of communities of the large real datasets used in the experiments.

Here, we present the results of the experiments on real graph with tables. Bold figures represent the best result for each criterion for a given clustering algorithm (`SC`, `CSC`, `louvain`, `leiden`, `MDL`, `infomap`).

Table E3: Results for the arxiv dataset.

methods	time ↓	ami ↑	modularity ↑	gnCut ↓	dl ↓
ground truth			0.493	0.436	9.23e6
SC	7.42e1	0.19	0.26	0.819	9.58e6
SC+PASCO ($t = 1$)	**4.46e0**	0.294	0.353	0.81	9.46e6
SC+PASCO ($t = 3$)	1.02e1	0.289	0.321	**0.779**	9.49e6
SC+PASCO ($t = 5$)	1.16e1	0.31	0.353	0.839	**9.43e6**
SC+PASCO ($t = 10$)	1.87e1	**0.311**	**0.354**	0.883	**9.43e6**
SC+PASCO ($t = 15$)	3.06e1	0.31	0.34	0.893	9.44e6
CSC	2.71e2	0.129	**0.361**	**0.4**	9.60e6
CSC+PASCO ($t = 1$)	**2.54e1**	0.185	0.325	0.481	9.56e6
CSC+PASCO ($t = 3$)	3.06e1	0.192	0.299	0.477	9.58e6
CSC+PASCO ($t = 5$)	3.28e1	0.21	0.291	0.48	9.55e6
CSC+PASCO ($t = 10$)	5.29e1	0.255	0.303	0.546	9.45e6
CSC+PASCO ($t = 15$)	7.46e1	**0.268**	0.311	0.572	**9.41e6**
louvain	1.77e0	0.39	0.704	0.897	8.39e6
louvain+PASCO ($t = 1$)	**1.14e0**	0.349	**0.581**	0.881	**9.05e6**
louvain+PASCO ($t = 3$)	5.00e0	0.37	0.615	**0.879**	8.94e6
louvain+PASCO ($t = 5$)	6.76e0	0.391	0.637	0.913	8.88e6
louvain+PASCO ($t = 10$)	1.22e1	0.393	0.648	0.914	8.85e6
louvain+PASCO ($t = 15$)	1.79e1	**0.405**	0.655	0.918	8.82e6
leiden	1.38e1	0.409	0.713	0.909	8.38e6
leiden+PASCO ($t = 1$)	**4.77e0**	0.359	**0.579**	0.891	**9.06e6**
leiden+PASCO ($t = 3$)	1.28e1	0.391	0.619	0.885	8.95e6
leiden+PASCO ($t = 5$)	1.66e1	0.4	0.64	**0.883**	8.85e6
leiden+PASCO ($t = 10$)	2.86e1	0.408	0.654	0.892	8.81e6
leiden+PASCO ($t = 15$)	3.67e1	**0.418**	0.665	0.922	8.75e6
MDL	7.51e2	0.351	0.651	0.705	8.03e6
MDL+PASCO ($t = 1$)	**3.82e2**	0.322	0.396	**0.438**	8.93e6
MDL+PASCO ($t = 3$)	5.24e2	0.338	0.384	0.442	**8.98e6**
MDL+PASCO ($t = 5$)	4.78e2	0.351	0.429	0.45	8.79e6
MDL+PASCO ($t = 10$)	6.52e2	0.379	0.462	0.476	8.61e6
MDL+PASCO ($t = 15$)	8.09e2	**0.387**	**0.496**	0.506	8.52e6
infomap	3.14e1	0.376	0.696	0.775	8.06e6
infomap+PASCO ($t = 1$)	**6.04e0**	0.358	**0.568**	**0.751**	**8.82e6**
infomap+PASCO ($t = 3$)	1.09e1	0.382	0.582	0.766	8.80e6
infomap+PASCO ($t = 5$)	1.22e1	0.395	0.605	0.782	8.67e6
infomap+PASCO ($t = 10$)	2.32e1	0.415	0.628	0.799	8.57e6
infomap+PASCO ($t = 15$)	2.81e1	**0.423**	0.639	0.809	8.53e6

Table E4: Results for the mag dataset.

methods	time ↓	ami ↑	modularity ↑	gnCut ↓	dl ↓
ground truth			0.268	0.217	5.35e7
SC	2.18e3	0.325	0.727	0.64	**4.92e7**
SC+PASCO ($t = 1$)	**2.90e2**	0.367	**0.672**	0.665	4.80e7
SC+PASCO ($t = 3$)	3.60e2	0.377	0.713	**0.608**	4.70e7
SC+PASCO ($t = 5$)	4.17e2	0.393	0.749	0.652	4.63e7
SC+PASCO ($t = 10$)	7.32e2	0.403	0.782	0.709	4.59e7
SC+PASCO ($t = 15$)	8.90e2	**0.406**	0.791	0.757	4.56e7
CSC	4.92e3	0.145	**0.438**	0.411	5.14e7
CSC+PASCO ($t = 1$)	**6.97e2**	0.235	0.505	0.424	5.22e7
CSC+PASCO ($t = 3$)	7.81e2	0.24	0.532	**0.401**	**5.24e7**
CSC+PASCO ($t = 5$)	9.03e2	0.26	0.566	0.435	5.15e7
CSC+PASCO ($t = 10$)	1.38e3	0.3	0.609	0.514	5.04e7
CSC+PASCO ($t = 15$)	2.20e3	**0.328**	0.646	0.56	4.96e7
louvain	1.36e1	**0.378**	0.842	0.931	4.71e7
louvain+PASCO ($t = 1$)	**8.47e0**	0.331	**0.748**	0.904	**5.03e7**
louvain+PASCO ($t = 3$)	2.32e1	0.352	0.773	0.891	4.98e7
louvain+PASCO ($t = 5$)	3.31e1	0.358	0.797	**0.824**	4.92e7
louvain+PASCO ($t = 10$)	8.07e1	0.364	0.804	0.903	4.91e7
louvain+PASCO ($t = 15$)	6.98e1	0.366	0.815	0.871	4.89e7
leiden	9.06e1	**0.379**	0.851	0.934	4.66e7
leiden+PASCO ($t = 1$)	**3.39e1**	0.343	**0.755**	0.918	**5.01e7**
leiden+PASCO ($t = 3$)	6.74e1	0.36	0.78	0.918	4.95e7
leiden+PASCO ($t = 5$)	8.18e1	0.372	0.803	**0.882**	4.90e7
leiden+PASCO ($t = 10$)	1.36e2	0.377	0.816	0.932	4.87e7
leiden+PASCO ($t = 15$)	1.92e2	**0.379**	0.817	0.943	4.87e7
MDL	3.91e3	0.357	0.701	0.709	3.95e7
MDL+PASCO ($t = 1$)	**3.32e3**	0.349	**0.446**	0.45	**4.53e7**
MDL+PASCO ($t = 3$)	4.54e3	0.355	0.474	**0.444**	4.50e7
MDL+PASCO ($t = 5$)	4.11e3	0.369	0.521	0.475	4.37e7
MDL+PASCO ($t = 10$)	4.95e3	0.382	0.583	0.526	4.22e7
MDL+PASCO ($t = 15$)	7.24e3	**0.386**	0.619	0.547	4.17e7
infomap	2.24e2	0.365	0.764	0.784	4.02e7
infomap+PASCO ($t = 1$)	**4.12e1**	0.359	**0.667**	**0.674**	**4.59e7**
infomap+PASCO ($t = 3$)	1.22e2	0.372	0.719	0.704	4.56e7
infomap+PASCO ($t = 5$)	1.63e2	0.384	0.752	0.715	4.45e7
infomap+PASCO ($t = 10$)	2.83e2	0.395	0.773	0.742	4.38e7
infomap+PASCO ($t = 15$)	4.06e2	**0.397**	0.784	0.757	4.35e7

Table E5: Results for the products dataset.

methods	time ↓	ami ↑	modularity ↑	gnCut ↓	dl ↓
ground truth			0.728	0.464	5.28e8
SC	6.37e2	0.202	**0.603**	0.722	**5.45e8**
SC+PASCO ($t = 1$)	**2.74e2**	0.327	0.41	0.717	5.97e8
SC+PASCO ($t = 3$)	4.37e2	0.283	0.412	**0.663**	5.98e8
SC+PASCO ($t = 5$)	5.48e2	**0.363**	0.48	0.801	5.92e8
SC+PASCO ($t = 10$)	7.45e2	0.332	0.444	0.885	5.95e8
SC+PASCO ($t = 15$)	9.99e2	0.32	0.42	0.824	5.96e8
CSC	2.08e4	0.206	0.585	0.601	5.43e8
CSC+PASCO ($t = 1$)	**3.32e3**	0.275	0.584	0.569	5.49e8
CSC+PASCO ($t = 3$)	5.07e3	0.262	0.552	**0.526**	5.59e8
CSC+PASCO ($t = 5$)	5.26e3	0.302	0.561	0.555	5.57e8
CSC+PASCO ($t = 10$)	6.45e3	0.406	0.669	0.625	**5.30e8**
CSC+PASCO ($t = 15$)	7.54e3	**0.436**	**0.715**	0.718	5.24e8
louvain	9.33e1	0.523	0.873	0.955	4.62e8
louvain+PASCO ($t = 1$)	**8.95e1**	0.49	**0.779**	0.937	**5.17e8**
louvain+PASCO ($t = 3$)	2.75e2	0.515	0.815	**0.896**	5.06e8
louvain+PASCO ($t = 5$)	3.76e2	0.531	0.834	0.906	5.01e8
louvain+PASCO ($t = 10$)	6.24e2	**0.537**	0.85	0.929	4.98e8
louvain+PASCO ($t = 15$)	8.23e2	0.535	0.849	0.897	4.97e8
leiden	7.93e2	**0.554**	0.881	0.957	4.55e8
leiden+PASCO ($t = 1$)	**3.61e2**	0.494	**0.786**	0.941	**5.14e8**
leiden+PASCO ($t = 3$)	8.07e2	0.526	0.826	**0.925**	5.05e8
leiden+PASCO ($t = 5$)	9.63e2	0.538	0.845	0.944	5.00e8
leiden+PASCO ($t = 10$)	1.65e3	0.549	0.851	0.94	4.95e8
leiden+PASCO ($t = 15$)	2.26e3	0.547	0.858	0.949	4.96e8
MDL	5.27e4	0.494	0.859	0.887	4.56e8
MDL+PASCO ($t = 1$)	**4.65e4**	0.491	**0.717**	**0.745**	**4.92e8**
MDL+PASCO ($t = 3$)	5.93e4	0.516	0.743	0.761	4.86e8
MDL+PASCO ($t = 5$)	6.55e4	0.533	0.785	0.803	4.75e8
MDL+PASCO ($t = 10$)	8.03e4	0.551	0.808	0.799	4.70e8
MDL+PASCO ($t = 15$)	9.58e4	**0.56**	0.827	0.837	4.65e8
infomap	2.71e3	0.504	0.87	0.896	4.54e8
infomap+PASCO ($t = 1$)	**4.78e2**	0.495	**0.765**	**0.797**	**4.89e8**
infomap+PASCO ($t = 3$)	7.61e2	0.523	0.794	0.832	4.80e8
infomap+PASCO ($t = 5$)	9.93e2	0.533	0.812	0.834	4.73e8
infomap+PASCO ($t = 10$)	1.18e3	0.546	0.841	0.872	4.66e8
infomap+PASCO ($t = 15$)	1.50e3	**0.553**	0.85	0.882	4.64e8