Reducing Million-Node Graphs to a Few Structural Patterns: A Unified Approach

Yike Liu
University of Michigan, Ann Arbor
yikeliu@umich.edu

Tara Safavi
University of Michigan, Ann Arbor
tsafavi@umich.edu

Neil Shah
Carnegie Mellon University
neilshah@cs.cmu.edu

Danai Koutra
University of Michigan, Ann Arbor
dkoutra@umich.edu

ABSTRACT

How do graph clustering techniques compare in terms of summarization power? How well can they summarize a million-node graph with a few representative structures? In this paper, we compare and contrast different techniques: METIS, LOUVAIN, SPECTRAL CLUSTERING, SLASHBURN, BIGCLAM, HYCOMFIT, and KCBC, our proposed k-core-based clustering method. Unlike prior work that focuses on various measures of cluster quality, we use vocabulary structures that often appear in real graphs and the Minimum Description Length (MDL) principle to obtain a graph summary per clustering method.

Our main contributions are: (i) *Formulation*: we propose a summarization-based evaluation of clustering methods. Our method, VOG-OVERLAP, concisely summarizes graphs in terms of their important structures with small edge overlap and large node/edge coverage; (ii) *Algorithm*: we introduce KCBC, a graph decomposition technique based on the k-core algorithm. We also introduce STEP, a summary assembly heuristic that produces compact summaries, as well as two parallel approximations thereof. (iii) *Evaluation*: we compare the summarization power of seven clustering techniques on large real graphs and analyze their compression rates, summary statistics, and runtimes.

1. INTRODUCTION

Summarization becomes increasingly crucial with the continuous generation of large amounts of data [17], as it can abstract away noise and help discover existing patterns, which in turn may inform the human decision process and the design of new large-scale analysis algorithms. In this paper we focus on the summarization of graphs, which are powerful structures that capture a host of phenomena from communication between people to interactions between neurons in the brain [15, 3, 26]. Graph summarization methods lead to the reduction of data volume, speedup of graph approaches, improved storage and query time, and interactive visualization. The graph mining community has mainly studied summarization techniques for the structure of static, plain graphs [8, 25] and, to a smaller extent, methods for attributed or dynamic networks [29]. Specifically, we study the summarization power of various graph clustering and community detection methods. Which graph clustering approach summarizes a graph in the best way? How expressive are the results?

Detecting clusters or communities in graphs is applicable to a variety of domains, including the social, biological, and web sciences [15, 4, 13]. Numerous objective functions have been proposed to detect clusters or communities, which are defined as "tightly-connected" subgraphs. These clusters provide a better understanding of the graph's underlying structure (e.g. functional units in biology, research communities in collaboration networks) and can also be seen as a summary of the original graph. In this paper we choose the latter interpretation and propose a novel approach to comparing graph clustering methods. Unlike the literature that compares clustering approaches in terms of cluster quality (e.g., modularity, average clustering coefficient, external and internal conductance, normalized mutual information), we compare and contrast the *output summaries* quantitatively and qualitatively. Our idea is that the method that leads to the best summary is the one that best guides a practitioner's attention to interesting and useful patterns.

For the graph summary generation, we leverage VOG [21, 22], a graph summarization algorithm that aims to succinctly describe million-node graphs. VOG formulates the graph summarization problem as an information-theoretic optimization problem in which the goal is to find the hidden local structures that collectively minimize the global description length of the graph. For compression, VOG uses the MDL principle:

$$min\ L(G, M) = min\{L(M) + L(\mathbf{E})\} \quad (1)$$

where $\mathbf{M}$ is an approximation of $\mathbf{A}$ deduced by the model M and $\mathbf{E} = \mathbf{M} \oplus \mathbf{A}$ is the error matrix. In addition to MDL, VOG uses a fixed vocabulary of structures Ω (cliques and near-cliques, stars, chains, and full or near-bipartite cores) that are ubiquitous in real-world graphs, and summarizes graphs using this vocabulary.

In [22], the first step of the summarization algorithm is to apply a subgraph extraction method, the output of which is then processed and refined by using MDL and a structure selection process. The authors use a node reordering method called SlashBurn [18], which biases the subgraphs to be stars or star-like (evidenced by the experiments in [22]). We propose to use VOG as a proxy to compare community detection and clustering methods with respect to a new metric, their summarization power: (i) SLASHBURN and its adaptation to graph clustering [18, 22]; (ii) k-core-based clustering (KCBC), which is formulated in a new way to provide multi-

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Table 1: Major symbols and definitions.

Notation	Description
$G(\mathcal{V}, \mathcal{E})$, $\mathbf{A}$	graph and its adjacency matrix, respectively
$\mathcal{V}, n = \lvert\mathcal{V}\rvert$	node-set and number of nodes of G, respectively
$\mathcal{E}, m = \lvert\mathcal{E}\rvert$	edge-set and number of edges of G, respectively
k	# of clusters or communities or subgraphs
t	# of iterations
h, m_h	size of hyperbolic community, and # of edges in it, respectively
d	average degree of nodes in G
M	a model for G = a list of node sets with associated structure types
s	structures in M
$\lvert S\rvert, \lvert s\rvert$	cardinality of set S and number of nodes in s, respectively
$\mathbf{E}$	error matrix, $\mathbf{E} = \mathbf{M} \oplus \mathbf{A}$
$L(G, M)$	# of bits to describe model M, and G using M
$L(M)$	# of bits to describe model M

ple strongly connected components; (iii) LOUVAIN community detection [5]; (iv) SPECTRAL CLUSTERING [16]; (v) METIS [19]; (vi) BIGCLAM [31]; (vii) HYCOM-FIT [2]. We note that each method employs a different objective function and biases the shape of the discovered subgraphs accordingly. Most of the methods detect well-connected clusters that correspond to full cliques or bipartite cores. We propose to use the MDL encoding cost as a quantitative means for comparing the methods: the most powerful method from a summarization perspective is the one that results in the most compressed graph summary. Our contributions are:

- *Formulation*: We are the first to evaluate clustering methods on the basis of MDL encoding and thoroughly compare different graph decomposition techniques in terms of summarization power. We also introduce an edge-overlap-aware graph summarization method.
- *Algorithm*: We propose KCBC, a scalable community detection algorithm based on k-cores. We also introduce STEP, a summary assembly method designed to produce compact summaries, as well as two parallel variants thereof.
- *Evaluation*: We conduct thorough empirical analysis of five graph decomposition methods on large, real-world data.

The paper is organized as follows: sections 2-4 introduce the clustering methods that we consider, our new summarization model that accounts for overlapping edges, and our summary assembly methods STEP, STEP-P, and STEP-PA. Our empirical evaluation, the related work, and conclusion are given in Sections 5-7. The major symbols are given in Table 1.

2. CLUSTERING METHODS FOR GRAPH SUMMARIZATION

One important part of the summarization approach that we leverage, VOG, is the graph decomposition method that is used to obtain candidate structures for the graph summary. In this section we study the effect of various clustering methods on the quality of the summary, and, in reverse, we use VOG-CONTRAST (Algorithm 1), a VOG-based approach, as a proxy to evaluate the summarization power of the clustering methods. We consider the following approaches: SLASHBURN, our proposed k-core-based clustering (KCBC), LOUVAIN, SPECTRAL CLUSTERING, METIS, BIGCLAM, and HYCOM-FIT. We first give a brief introduction of these algorithms and how we leverage them to generate candidate subgraphs for the VOG-style summaries, and then summarize their qualitative comparison in Table 2.

[1]Unlike VOG, in our algorithm we only consider full cliques, bipartite cores, stars, and chains. That is, we ignore the near-structures, because their MDL encoding incorporates error, which interferes with the global MDL cost in Equation (1).

Algorithm 1 VOG-CONTRAST: Summarization-based Comparison of Graph Decomposition Methods

Input: graph G

Step 1: Summary extraction by using one of {SLASHBURN, KCBC, LOUVAIN, SPECTRAL CLUSTERING, METIS, BIGCLAM, and HYCOM-FIT }.
Step 2: MDL-based subgraph labeling by using a reduced vocabulary (compared to VOG[1] that consists of full cliques, full bipartite cores, stars and chains.
Step 3: Summary assembly by employing the proposed heuristics STEP, STEP-P, and STEP-PA (Section 4).
return summary of the most important graph structures

SLASHBURN [18] is a node reordering algorithm initially developed for graph compression. It performs two steps iteratively: (i) It removes high-centrality nodes from the graph; (ii) It reorders nodes such that high-degree nodes are assigned the lowest IDs, and nodes from disconnected components get the highest IDs. The process is repeated on the giant connected component. We leverage this process by identifying structures from the egonet[2] of each central node, and the disconnected components, as subgraphs.

LOUVAIN community detection [5] is a modularity-based graph partitioning method for detecting hierarchical community structure. The method is composed of two phases which are applied iteratively. In the first phase, each node is placed in its own community. Next, the neighbors j of each node i are considered, and i is moved to j's community if the move produces the maximum modularity gain. The process is applied repeatedly until no further gain is possible. In the second phase, a new graph is built whose supernodes represent the communities of the first phase, and superedges are weighted by the sum of weights of links between the two communities. The first phase is then applied to the new graph and the algorithm typically converges in a few such passes.

SPECTRAL CLUSTERING refers to a class of algorithms which utilize eigendecomposition to identify community structure. We use one such spectral clustering algorithm [16], which partitions a graph by performing k-means clustering on the top-k eigenvectors of the input graph. The idea behind this method is that nodes with similar connectivity have similar eigen-scores in the top-k vectors and form clusters.

METIS [19] is a cut-based k-way multilevel graph partitioning scheme based on multilevel recursive bisection (MLRB). Until the graph size is substantially reduced, it first coarsens the input graph by collapsing connected nodes into supernodes iteratively so that the edge-cut is preserved. Next, the coarsened graph is partitioned using MLRB and the partitioning is projected onto the original input graph G through backtracking. The method produces k roughly equally sized partitions.

HYCOM-FIT [2] is a parameter-free algorithm that detects communities with hyperbolic structure. It approximates the optimal solution by iteratively detecting important communities. The key idea is to find in each step a single community that minimizes an MDL-based objective function, given the previously detected communities. The iterative procedure consists of three steps: community candidates, community construction, and matrix deflation.

BIGCLAM [31] is an overlapping community detection method that scales to large networks, built on the observation that overlaps between communities are densely connected. By explicitly modeling the affiliation strength per node-community pair, each node-community pair is assigned a nonnegative latent factor which repre-

[2]Egonet is the induced subgraph of a node and its neighbors.

Table 2: Qualitative comparison of the graph clustering techniques included in VoG-Overlap.

Properties	Clustering Techniques						
	SlashBurn	Louvain	Spectral Clustering	METIS	HyCoM-FIT	BigClam	Proposed: KCBC
Overlapping Clusters	✔	✘	✘	✘	✔	✔	✔
Similar-sized Clusters	✘	✘	✘	✘	✘	✘	✘
Complexity	$O(t(m+n\log n))$	$O(n\log n)$	$O(n^3)$	$O(m\cdot k)$	$O(k(m+h(\log h^2+m_h)))$	$O(d\cdot n\cdot t)$	$O(m+n)$
Parameter-free	✔	✘	✔	✘	✔	✔	✔(in our alg.)
Number of Clusters	High	Medium	Medium	Medium	High	High	Low
Summarization Power	Excellent	Good	Good	Good	Poor	Good	Poor
Cliques	✔	✔	✔	✔	✔	✔	✔
Bipartite Cores	✔	✔	✔	✔	✔	✔	✘
Stars	✔	✔	✔	✔	✔	✔	✔
Chains	✔	✘	✘	✘	✔	✘	✘

sents the degree of membership of a node to the community. Then the probability of an edge is modeled as a function of the shared community affiliations. The identification of network communities is done by fitting BigClam to a given undirected network G.

Proposed: KCBC. k-cores [14] have traditionally been used to unveil densely connected structures in graphs. A k-core can be defined as a maximal subgraph for which each node is connected to at least k other nodes in the subgraph. Though the existence of k-cores in social graphs has been studied previously, to our knowledge no previous works leverage the k-core algorithm (recursively delete all nodes and adjacent edges in the graph of degree less than k) to identify communities in graphs. In this work, we develop a k-core-based algorithm to identify notable graph structures. The method is described in Algorithm 2. The main advantages of this method are that it (i) is fast and scalable with time complexity $O(n+m)$ (ii) can identify multiple structures per node and (iii) produces concise listings of non-redundant structures.

Algorithm 2 KCBC: k-core-based Graph Clustering

Input: graph G

While the graph is nonempty

Step 1: Compute core numbers (max k for which the node is present in the decomposition) for all nodes in the graph.

Step 2: Choose the maximum k (k_{max}) for which the decomposition is non-empty, and identify the present nodes as the 'decomposition set'. Terminate when $k_{max}=1$.

Step 3: For the induced subgraph from the decomposition set, identify each connected component as a structure.

Step 4: Remove all edges in the graph between nodes in the decomposition set—they have been identified as structures already.

return set of all identified structures

In Table 2 we compare the main features of these clustering methods, where n is the number of nodes and m is the number of edges in a graph. Several of the approaches give clusters with overlapping edges. Ideally we would like to minimize this kind of overlap between structures–we handle this problem next.

3. ENCODING OVERLAPPING EDGES

In [22], the proposed summarization algorithm assumes that the candidate structures for the graph summary have node overlap, but no edge overlap. However, several graph decomposition methods (Table 2), including SlashBurn and KCBC, produce edge-overlapping subgraphs, and there is good chance that a group of densely overlapping subgraphs is included in the graph summary because they all *appear* to help reduce the encoding cost of the graph.

Our goal is to get a concise summary of the graph without explaining away edges multiple times – i.e., we want to minimize node and *edge* redundancy in our graph summary. We note that we are still interested in overlapping nodes that span multiple structures, as they can be seen as 'bridges' or 'connectors', and provide useful information about the network structure. To handle the above-mentioned issue, we propose VoG-Overlap, which minimizes the node/edge *overlap*, and maximizes the node/edge *coverage* of the summary. First, we give an illustrative example that shows the issue of overlapping edges that arises from some graph clustering methods, and then provide the details of VoG-Overlap, and show its better performance compared to VoG.

An Illustrative Example. Let us assume that the output of an edge-overlapping graph decomposition method is the following model which consists of three full cliques: *full clique 1* with nodes 1-20; *full clique 2* with nodes 11-30; and *full clique 3* with nodes 21-40. The VoG-based summary, which does not account for overlaps, includes in the summary all three structures, which clearly encode redundant nodes and edges. Despite the overlap, the description length of the graph given the model is only 441 bits, since it does not penalize edges that are covered multiple times. For reference, the graph needs 652 bits under the null model assumption. Ideally, we would want a method that penalizes extensive overlaps and tries to increase node/edge coverage.

Encoding the Overlapping Edges. We propose VoG-Overlap, which detects significant node and edge overlaps, and steers the structure selection process towards the desired output. We extend the optimization function for graph summarization by adding an overlap-related term (underlined):

$$min\ L(G,M) = min\{L(M)+L(\mathbf{E})+\underline{L(\mathbf{O})}\} \qquad (2)$$

where $\mathbf{M}$ is an approximation of $\mathbf{A}$ deduced by the model M, $\mathbf{E}=\mathbf{M}\oplus\mathbf{A}$ is the error matrix, and $\mathbf{O}$ is a weighted matrix with the number of times that each edge has been explained by M.

For consistency with VoG, we use the optimal prefix code [11] to encode the total number of overlapping edges. Following the literature [23], to encode the weights in matrix $\mathbf{O}$ which correspond to the number of times that each of the edges has been covered by the model, we use the MDL optimal encoding for integers [28]. The encoding for the overlaps is given by:

$$L(\mathbf{O}) = \log(|\mathbf{O}|) + ||\mathbf{O}||l_1 + ||\mathbf{O}||'l_0 + \sum_{o\in\mathcal{E}(\mathbf{O})} L_{\mathbb{N}}(|o|),$$

where $|\mathbf{O}|$ is the number of (distinct) overlapping edges, $||\mathbf{O}||$ and $||\mathbf{O}||'$ correspond to the number of present and missing entries in $\mathbf{O}$, $l_1 = -\log((||\mathbf{O}||/(||\mathbf{O}||+||\mathbf{O}||'))$, and analogue for l_0, are the lengths of the optimal prefix codes for the present and missing entries, respectively, and $\mathcal{E}(\mathbf{O})$ is the set of non-zero entries in matrix $\mathbf{O}$.

By applying VoG-Overlap to the example above, we obtain in the summary only the 1st and the 2nd clique, as desired. The encoding of our proposed method is 518 bits, which is higher

than the number of bits of VoG: The reason is that in the VoG-Overlap-based summary some edges have remained unexplained (edges from nodes 11-20 to nodes 21-40), and, thus, are encoded as error. On the other hand, the VoG-based model encodes all nodes and edges (without errors), but explains many edges twice (e.g., the clique 11-20, the edges between 11-20 and 21-30) without accounting for the redundancy-related bits twice.

4. COMPILING COMPACT SUMMARIES

The problem of selecting structures to include in the graph summary such that the graph encoding cost is minimized is hard because the objective function in Equation 2 does not admit a tractable solution; it is neither monotonic nor submodular. If the objective function were submodular, we would be guaranteed to find a greedy heuristic that gives a $(1-\frac{1}{e})$-approximation of the optimal solution.

Instead of considering all possible combinations of structures as summaries, which is not tractable, prior work [22] relies on the summary assembly heuristics Top10 and GnF. Top10, which chooses the 10 structures that reduce the encoding cost of the graph the most, is fast but does not provide a high compression rate nor high coverage of the graph with the model. GnF considers each candidate structure sequentially and includes in the summary (or model M) only the structures that help further decrease the current encoding cost of the graph. Compared to Top10, GnF yields a lower compression rate and better coverage. However, its performance heavily depends on the ordering of structures.

To overcome these shortcomings, we propose a new model selection method, Step. Motivated by Step's good qualitative performance but high runtime complexity, we also propose two parallel variants, Step-P and Step-PA.

• **Step.** Step is a 2-part iterative algorithm:

1. Iterate through all structures in M: for each structure, compute the encoding cost $L(G, M)$ with that structure included in the summary.
2. Choose the structure found in part 1 that lowers the encoding cost the most and add it to the summary. If no structure lowers the encoding cost, the algorithm has converged; else, go back to part 1.

In the i_{th} iteration Step solves the optimization problem:

$$s_i^* = \textit{argmin}_{s \in S_{not}} L(G, M_{i-1} \cup \{s\})$$

where $M_{i-1} = \{s_1, \ldots, s_{i-1}\}$ is the model generated by the $i-1_{th}$ iteration, $M_0 = \emptyset$ is the empty model, and S_{not} is the set of structures that are not included in the model. Unlike GnF, Step does *not* depend on the order of structures, rendering it more robust.

We will show in Section 5 that Step produces significantly more compact summaries than GnF. However, because Step considers all candidate structures in each iteration, its time complexity is $O(s^2)$ where s is the number of structures, making it unrealistically slow for graphs with a large number of structures. Considering the tradeoff between compression rate and runtime, we propose Step-P, a parallel heuristic that approximates Step while being significantly faster, as well as an optimization to Step-P that further decreases runtime while maintaining encoding cost.

• **Step-P.** The goal of Step-P is to speed up the computation of Step by iteratively solving smaller, "local" versions of Step in parallel. Step-P begins by dividing the nodes of the graph into p partitions using METIS partitioning. Next, each candidate structure is assigned to the partition with the maximal node overlap. Step-P then iterates until convergence:

1. Spawn a parallel process for every partition. Each process is tasked with finding the structure that would lower the encoding cost in Equation (1) the most out of all the structures in its partition. Note that for any given partition there may be no structure that lowers the encoding cost.
2. After all processes have terminated, choose the lowest-cost structure returned by the processes to be added to the model. If no structure lowers the encoding cost, the algorithm has converged; else, go back to part 1.

• **Step-PA.** In addition to parallelizing Step, we introduce the idea of "active" and "inactive" partitions, which is an optimization designed to reduce the number of processes that are spawned by Step-P. Step-PA works the same as Step-P, except that it begins by designating every partition of the graph as active. Then, if a partition fails to find a structure that would lower the encoding cost in Equation (1) x times (in step 1 of any iteration), that partition is declared inactive and is not visited in future iterations of Step-PA. Thus the partitions with structures not likely to decrease the overall encoding cost of the model get several "chances" before being eventually ruled out, effectively reducing the number of processes spawned for each iteration of Step-PA.

5. EXPERIMENTS

To compare and contrast the various decomposition methods used in VoG-Contrast, we use several real-world graphs, described in Table 3. We evaluate the clustering methods in terms of tradeoff between compression and coverage, qualitative properties of their resulting summaries, and runtime. For summary assembly we use the heuristics proposed in Section 4 and compare them to GnF, the best-performing heuristic in [22].

Table 3: Summary of graphs used in our empirical comparison.

Name	Nodes	Edges	Description
Flickr[3]	404,733	2,110,078	Friendship social network
Enron[4]	80,163	288,364	Enron email
AS-Oregon[5]	13,579	37,448	Router connections
Wikipedia-Chocolate[6]	2,899	5,467	Co-editor graph

5.1 Compression vs. Node/Edge Coverage

We start by addressing the tradeoff between compression rate and node/edge coverage for the five clustering methods considered. Tables 4 and 5 give the compression rate of VoG-Overlap with the Top10 and GnF selection heuristic, respectively. By compression rate we refer to the ratio between the number of bits needed by the final model over the number of bits required to describe the graph under the empty model (without any structures).

Figure 1 shows the node coverage (top) and edge coverage (bottom) of each VoG-Overlap + GnF summary model for the Chocolate (left) and AS-Oregon (right) graphs. The lighter and darker shades correspond to the node coverage before and after the structure selection, respectively (i.e., before/after step 3 of Algorithm 1). The node coverage for the non-overlapping clustering methods before the structure selection is not 100% because we ignore clusters with fewer than 3 nodes, which are not likely to be interesting from a practitioner's perspective.

[3]Flickr dataset: http://www.flickr.com

[4]Enron dataset: http://www.cs.cmu.edu/∼enron

[5]AS-Oregon dataset: http://topology.eecs.umich.edu/data.html.

[6]Data crawled by Jeffrey Rzeszotarski.

Table 4: VoG vs. VoG-Overlap + Top10: Compression rate of the clustering techniques with respect to the empty model. Lower is better.

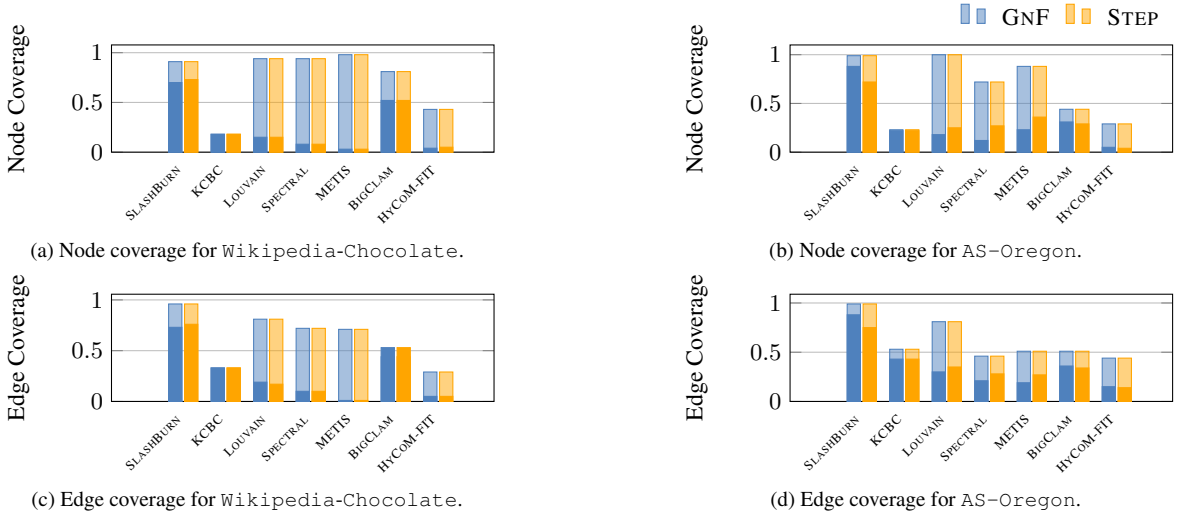
Dataset	VoG [21]	VoG-Overlap + Top10						
		SlashBurn	KCBC	Louvain	Spectral	METIS	BigClam	HyCoM-FIT
Flickr	99%	**99%**	100%	100%	100%	100%	100%	99%
Enron	98%	**98%**	100%	101%	100%	100%	100%	99%
AS-Oregon	87%	**87%**	96%	100%	100%	104%	101%	100%
Wikipedia-Chocolate	94%	94%	**78%**	101%	101%	104%	103%	100%

Table 5: VoG vs. VoG-Overlap + GnF: Compression rate of the clustering techniques with respect to the empty model.

Dataset	VoG [21]	VoG-Overlap + GnF						
		SlashBurn	KCBC	Louvain	Spectral	METIS	BigClam	HyCoM-FIT
Flickr	95%	80%	**33%**	88%	98%	98%	91%	95%
Enron	75%	75%	**41%**	100%	99%	100%	77%	95%
AS-Oregon	71%	71%	**65%**	95%	94%	96%	85%	98%
Wikipedia-Chocolate	88%	88%	**78%**	99%	99%	100%	89%	100%

Table 6: VoG vs. VoG-Overlap + Step: Compression rate of the clustering techniques with respect to the empty model.

Dataset	VoG [21]	VoG-Overlap + Step						
		SlashBurn	KCBC	Louvain	Spectral	METIS	BigClam	HyCoM-FIT
AS-Oregon	71%	76%	**65%**	94%	82%	85%	83%	98%
Wikipedia-Chocolate	88%	88%	**78%**	99%	99%	100%	87%	100%



(a) Node coverage for Wikipedia-Chocolate.

(b) Node coverage for AS-Oregon.

(c) Edge coverage for Wikipedia-Chocolate.

(d) Edge coverage for AS-Oregon.

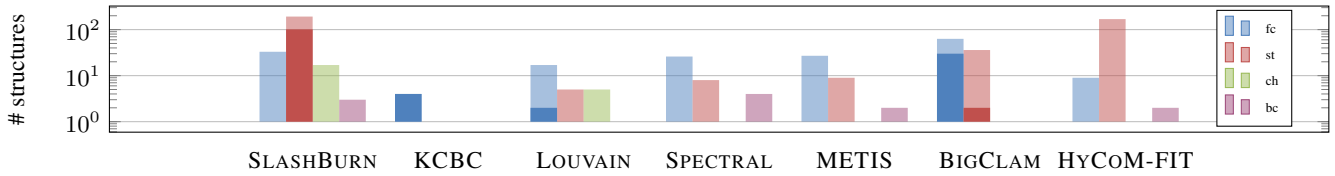
Figure 1: SlashBurn gives summaries with the best node/edge coverage. We show the node/edge coverage per clustering method with GnF and Step for Wikipedia-Chocolate (left) and AS-Oregon (right). *Light/dark shade = before/after structure selection.

Observation 1. *In terms of compression rate,* SlashBurn *and* KCBC *usually win over other methods, especially for the* GnF *heuristic. When* KCBC *has a much lower compression rate than* SlashBurn*, it is due to the fact that* KCBC *covers very few nodes (e.g., nodes with degree* $\geq k$*). Though lower compression is usually better, we still seek to find informative summaries with good coverage.*

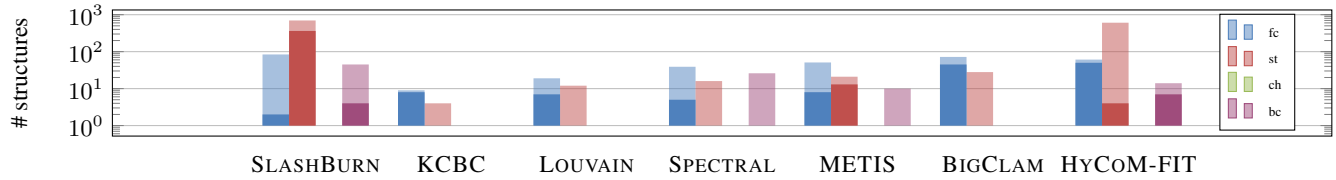
We see this tradeoff clearly with SlashBurn and KCBC. While KCBC has a significantly lower compression rate, SlashBurn achieves much higher node and edge coverage (Fig. 1). Louvain, Spectral Clustering and METIS lead to good coverage before the structure selection in step 3 of VoG-Contrast, but are poor after that. BigClam achieves smaller node/edge coverage, but also smaller difference in coverage between before and after structure selection. HyCoM-FIT has the lowest coverage in both datasets. Its coverage before structure selection is comparable to KCBC, but it drops significantly after the selection takes place.

5.2 Qualitative Comparison of Structures

What types of structures make up the summaries? Which method is the most expressive in terms of summarization? Figures 2 and 3 depict the number of structures found by each clustering method before and after the selection step (light and dark color, respectively). It is not surprising that SlashBurn tends to find more structures than others, especially for stars, because of the way it decomposes the input graph. SlashBurn and HyCoM-FIT contribute mainly stars and some cliques in the graph's final summary.

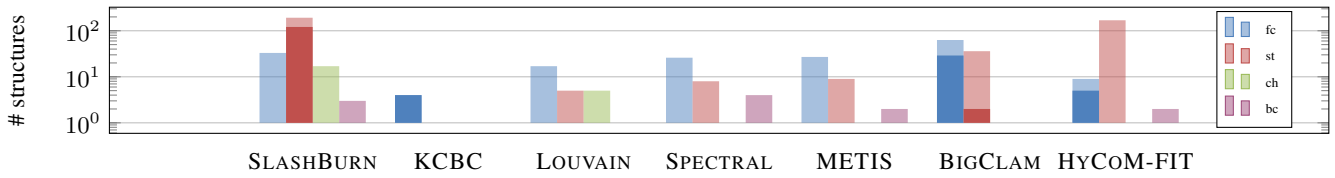


(a) Number of structures found by the clustering methods for dataset `Wikipedia-Chocolate`.

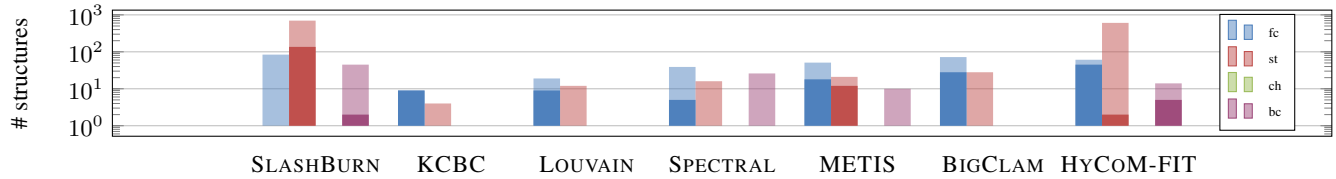


(b) Number of structures found by the clustering methods for dataset `AS-Oregon`.

Figure 2: Types of structures in `Wikipedia-Chocolate` and `AS-Oregon` graphs by GNF. Transparent/solid rectangles for before/after the structure selection step. Notation: 'fc': full clique, 'st': star, 'ch': chain, 'bc': bipartite core.



(a) Number of structures found by the clustering methods for dataset `Wikipedia-Chocolate`.



(b) Number of structures found by the clustering methods for dataset `AS-Oregon`.

Figure 3: Types of structures in `Wikipedia-Chocolate` and `AS-Oregon` graphs by STEP. Transparent/solid rectangles for before/after the structure selection step. Notation:: 'fc': full clique, 'st': star, 'ch': chain, 'bc': bipartite core.

Table 7: Runtime (in sec) of VOG-CONTRAST + GNF for the seven clustering techniques. We give the total runtime, and in parentheses the time division between structure generation/labeling (steps 1-2) and summary assembly (step 3). The fastest approaches are in bold.

Dataset	Clustering Techniques						
	SLASHBURN	KCBC	LOUVAIN	SPECTRAL	METIS	BIGCLAM	HYCOM-FIT
`Enron`	81737 (78108 + 3629)	11298 (**2860** + 8438)	12772 (5705 + 7067)	**5067** (3661 + 1406)	5444 (5039 + **405**)	19898 (19115 + 783)	8588 (6613 + 1975)
`AS-Oregon`	136 (107 + 29)	37 (**7** + 30)	238 (8 + 230)	26 (15 + 11)	**24** (18 + **6**)	44 (24 + 20)	388 (364 + 24)
`Wikipedia -Chocolate`	6 (5 + 1)	3.5 (3 + **0.5**)	6 (5 + 2)	**2** (**1** + 1)	5 (4 + 1)	12 (11 + 1)	29 (28 + 1)

KCBC's contribution is reverse. SPECTRAL CLUSTERING and METIS identify mainly full cliques and stars, but also some bipartite cores.

OBSERVATION 2. SLASHBURN *results in the most expressive summary which consists of all four types of structures. The other approaches find significantly fewer structures after selection, and the majority of them are full cliques and stars.*

5.3 Runtime Comparison

In Table 7 we provide the runtime of the subgraph generation (clustering) and summary assembly for each method in VOG-CONTRAST. Since GNF tends to give better compression than the TOP10 heuristic, we only report results for that.

OBSERVATION 3. *For Enron, the largest graph that we used in our experiment, the ordering of the subgraph generation methods from fastest to slowest is* SPECTRAL, METIS, HYCOM-FIT, KCBC, *followed by* LOUVAIN, BIGCLAM *and* SLASHBURN. *The differences in runtime increase as the size of the input graph in-*

Table 8: Runtime (in sec, unless stated otherwise) of STEP, STEP-P, STEP-PA, and GNF.

Dataset	STEP	STEP-P	STEP-PA	GNF
Enron	$>$ 2 weeks	$>$ 4 days	302737	12732
AS-Oregon	17653	8843	1996	407
Chocolate	249	138	16	8

Table 9: Compression rate (in bits) of STEP, STEP-P, STEP-PA, and GNF. Lower is better; N/A for runs that were terminated early.

Dataset	STEP	STEP-P	STEP-PA	GNF
Enron	N/A	N/A	25%	75%
AS-Oregon	35%	35%	35%	71%
Chocolate	56%	56%	56%	88%

creases. The summary assembly time is proportional to the number and size of candidate structures provided by the clustering method.

Considering the summarization power generally, SLASHBURN overpowers the other methods in quantity of discovered structures, especially stars. Cliques are identified by a selection of methods including KCBC, LOUVAIN, and SPECTRAL, which are often faster than SLASHBURN. All in all, each method has some advantages and disadvantages, and introduces different biases for the structure types that can compose the graph summaries.

5.4 Comparison of Summary Assembly Methods

Here we compare our proposed summary assembly heuristics STEP, STEP-P, and STEP-PA to the previously proposed GNF with respect to runtime and encoding cost. For these experiments we considered the set of candidate structures as the union of the subgraphs generated by all clustering methods described in Section 2. For our STEP-PA experiments, we chose $x = 3$ such that each active partition got three "chances" before being declared inactive. We ran our experiments on a single Ubuntu Linux server with 12 processors and 264 GB of RAM. In terms of parallel implementation, we set the maximum number of STEP-P processes running at any given moment equal to the number of processors on the machine to avoid thrashing. It is worth noting that the runtime of STEP-P is thus highly machine-dependent.

We give the runtime comparison of STEP, STEP-P, STEP-PA, and GNF in Table 8 and the encoding cost comparison in Table 9. For Enron, a graph of around 2.1 million edges with a set of candidate structures of around 11000 structures, the STEP heuristic was terminated after running for over two weeks. Although STEP-P is a parallelized version of STEP, it does not scale very well with input size, and it needs significantly more than 4 days to complete in the case of Enron. This highlights the significance of STEP-PA, our parallelized method that has additional optimizations, which completed in about 3.5 days.

OBSERVATION 4. *We see that while* GNF *is faster,* STEP *and its variants yield summaries that are 30-50% more compact. We also find that that the optimizations introduced with* STEP-PA *have no effect on the encoding cost of the final summary compared to* STEP*, but significantly reduce the runtime, rendering it more useful than* STEP *or* STEP-P *in practice as it retains quality while decreasing runtime.*

6. RELATED WORK

Work related to VOG-CONTRAST comprises MDL-based and graph clustering approaches.

MDL and Graph Mining. Many data mining problems are related to summarization and pattern discovery, and, thus, are intrinsically related to Kolmogorov complexity [12]. While not computable, it can be practically implemented by the Minimum Description Length principle [27] (lossless compression). Examples of applications in data mining include clustering [9], community detection in matrices [7], and outlier detection [1].

Graph Clustering. We have already presented several graph clustering and community detection methods [5, 19, 16] in Section 2, which are all biased toward heavily connected subgraphs, such as cliques and bipartite cores. Other methods include the blockmodels representation [6], and community detection algorithms tailored to social, biological, and web networks [15, 4, 13]. Work on clustering attributed graphs [20, 30, 32] is related but does not apply in our case. Leskovec et al.'s work [24] is relevant to ours since it compares several clustering methods, but their focus is on classic measures of cluster quality, while we propose an evaluation of these method in terms of summarization power. Some methods [25, 10] are related to VOG, but cannot be used as proxies to evaluate the summarization power of clustering techniques.

7. CONCLUSION

In this paper we evaluate various graph clustering and community detection methods in terms of summarization power, in contrast to the literature that focuses on measures of cluster quality. We propose an MDL-based graph summarization approach, VOG-CONTRAST, which leverages the clusters found by graph decomposition methods and is also edge-overlap aware (VOG-OVERLAP). Moreover, we present KCBC, a highly efficient, scalable, and parameter-free graph clustering algorithm based on k-cores. Finally, we introduce STEP, a model selection method that produces more compact summaries than GNF, and we parallelize STEP with STEP-P and STEP-PA to scale the algorithm to larger inputs. Our thorough experimental analysis on real-world graphs shows that each clustering approach has different strengths and weaknesses in terms of summarization power. Understanding their biases and combining them accordingly can give stronger better graph summaries with more diverse and high-quality structures.

8. REFERENCES

[1] L. Akoglu, H. Tong, J. Vreeken, and C. Faloutsos. Fast and Reliable Anomaly Detection in Categorical Data. In *CIKM*. ACM, 2012.

[2] M. Araujo, S. Günnemann, G. Mateos, and C. Faloutsos. Beyond blocks: Hyperbolic community detection. In *Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer, 2014.

[3] L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD*, pages 44–54, 2006.

[4] L. Backstrom, R. Kumar, C. Marlow, J. Novak, and A. Tomkins. Preferential behavior in online groups. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 117–128, New York, NY, USA, 2008. ACM.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[6] S. P. Borgatti. Special issue on blockmodels: Introduction. *Social Networks*, 14(1):1–3, 1992.

[7] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 79–88, 2004.

[8] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On Compressing Social Networks. In *KDD*, pages 219–228, 2009.

[9] R. Cilibrasi and P. Vitányi. Clustering by Compression. *IEEE TIT*, 51(4):1523–1545, 2005.

[10] D. J. Cook and L. B. Holder. Substructure Discovery Using Minimum Description Length and Background Knowledge. *JAIR*, 1:231–255, 1994.

[11] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[12] C. Faloutsos and V. Megalooikonomou. On Data Mining, Compression and Kolmogorov Complexity. In *Data Min. Knowl. Disc.*, volume 15, pages 3–20. Springer-Verlag, 2007.

[13] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[14] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 87–93. IEEE, 2011.

[15] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99:7821, 2002.

[16] J. P. Hespanha. An efficient matlab algorithm for graph partitioning. *Department of Electrical and Computer Engineering, University of California, Santa Barbara*, pages 25–67, 2004.

[17] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014.

[18] U. Kang and C. Faloutsos. Beyond 'Caveman Communities': Hubs and Spokes for Graph Compression and Mining. In *ICDM*, 2011.

[19] G. Karypis and V. Kumar. Multilevel k-way Hypergraph Partitioning. pages 343–348, 1999.

[20] A. Koopman and A. Siebes. Characteristic Relational Patterns. In *KDD*, pages 437–446, 2009.

[21] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. VoG: Summarizing and Understanding Large Graphs. In *SDM*, pages 91–99, 2014.

[22] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. Summarizing and Understanding Large Graphs. In *Statistical Analysis and Data Mining*. John Wiley & Sons, Inc., 2015.

[23] E. Kuzey, J. Vreeken, and G. Weikum. A fresh look on knowledge bases: Distilling named events from news. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1689–1698. ACM, 2014.

[24] J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.

[25] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph Summarization with Bounded Error. In *SIGMOD*, pages 419–432, 2008.

[26] OCP. Open Connectome Project. http://www.openconnectomeproject.org, 2014.

[27] J. Rissanen. Modeling by Shortest Data Description. *Automatica*, 14(1):465–471, 1978.

[28] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, pages 416–431, 1983.

[29] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1055–1064, New York, NY, USA, 2015. ACM.

[30] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 505–516. ACM, 2012.

[31] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.

[32] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2(1):718–729, 2009.