Algebraic Vertex Ordering of a Sparse Graph for Adjacency Access Locality and Graph Compression

Dimitris Floros* Nikos Pitsianis†‡ Xiaobai Sun‡

*Nicholas School of the Environment
Duke University
Durham, NC 27708, USA

†Department of Electrical and Computer Engineering
Aristotle University of Thessaloniki
Thessaloniki 54124, Greece

‡Department of Computer Science
Duke University
Durham, NC 27708, USA

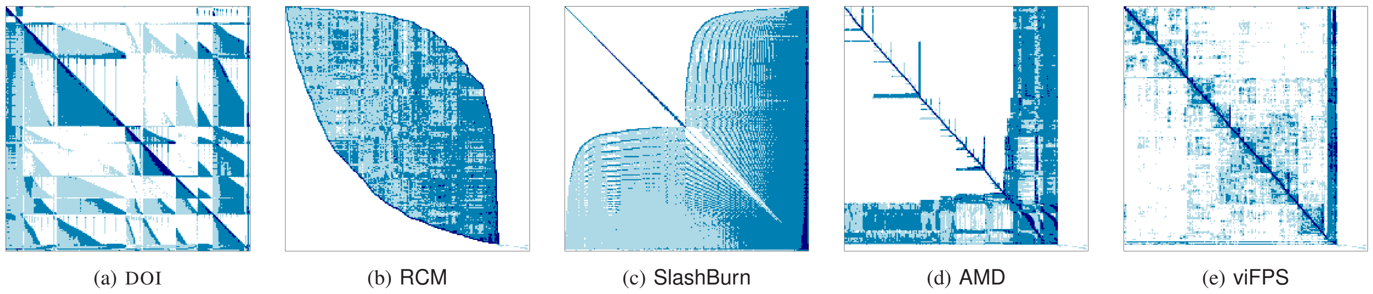


Fig. 1: The adjacency-matrix images of the citation graph $\mathsf{aps2020} = G(V,E)$ in five different vertex orderings. The graph is a representation of the American Physical Society (APS) publication up to the year 2020, with $|V| = 667{,}365$ articles and $|E| = 8{,}849{,}630$ citation links [2]. Each image pixel (i,j) represents a citation subgraph $G(V_i, V_j, E_{ij})$ with $|V_i| = |V_j| = 3{,}336$, $E_{ij} = (V_i \times V_j) \cap E$. A darker pixel indicates a denser subgraph. The five ordering methods are: (a) the lexicographical order of the digital object identifiers (DOIs) for the APS articles, (b) the reverse Cuthill-McKee (RCM) method [10], (c) the SlashBurn method [21], (d) the approximate minimum degree (AMD) method [3], and (e) the new method viFPS. The properties and performance assessment of these orderings with regard to graph compression are elaborated in the rest of the paper.

***Abstract*—In this work, we establish theoretical and practical connections between vertex indexing for sparse graph/network compression and matrix ordering for sparse matrix-vector multiplication and variable elimination. We present a fundamental analysis of adjacency access locality in vertex ordering from the perspective of graph composition of, or decomposition into, elementary compact graphs. We introduce an algebraic indexing approach that maintains the advantageous features of existing methods, mitigates their shortcomings, and adapts to the degree distribution. The new method demonstrates superior and versatile performance in graph compression across diverse types of graphs. It also renders proportional improvement in the efficiency of matrix-vector multiplications for subspace iterations in response to random walk queries on a large network.**

***Index Terms*—Graph compression, network compression, adjacency gap encoding, adjacency access locality, algebraic vertex ordering, sparse matrix computation.**

I. Introduction

In a modern data, knowledge, or information system, the datum entities (or feature vectors) are typically linked by a direct or induced pairwise adjacency relationship and represented as the vertices of a big graph/network $G(V,E)$ with vertex set V and edge set E. The edge set $E \subset V \times V$ represents the pairwise adjacency relation. The edges may be undirected or directed. The large graph is usually sparse and structured compared to Erdős-Rényi random graphs of the same sparsity. For data-information management and processing, the vertices/nodes are indexed from 1 to $n = |V|$ by a 1-to-1 mapping from the datum labels or identification numbers. How the vertices are sequentially indexed or ordered significantly impacts the space-time performance of the system in two related key aspects—the graph compression beyond the simple exclusion of absent links and the efficiency in response to frequent adjacency queries for retrieval and referral while the graph is accessed via a compressed representation. In this paper, we present our study and findings on vertex ordering for graph compression.

A graph compression maintaining real-time adjacency accesses (ideally in memory) is essentially an integral compression of individual adjacency lists on the graph with small compression/decompression windows. Every vertex v has an adjacency/neighbor list $\mathcal{N}(v)$ consisting of its incident edges or, in interchangeable terms, its adjacent nodes or immediate neighbors. With a vertex ordering, the adjacency list is represented by a subsequence of $\{1, 2, \ldots, n\}$. The degree of node v, $d(v)$, is the number of its neighbors, $d(v) = |\mathcal{N}(v)|$. If graph G is directed, $\mathcal{N}(v)$ has two sublists: $\mathcal{N}_{\text{in}}(v)$ of the incoming links and $\mathcal{N}_{\text{out}}(v)$ of the outgoing links. Invariant to vertex ordering, the total number of adjacency lists is n; the total number of the list items is $m = |E|$. Nonetheless, the integer subsequences representing the adjacency lists change from one vertex ordering to another, their compression is thereby affected first and foremost by vertex ordering.

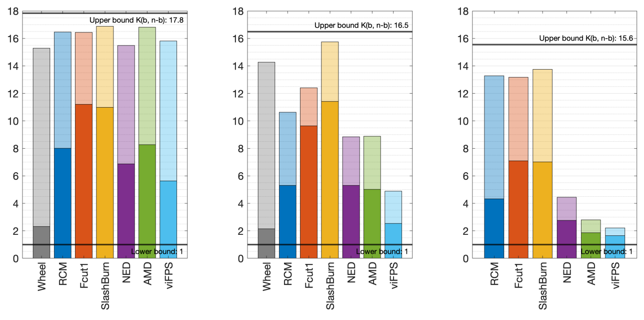
We also use $\mathcal{N}(v)$ to denote the 1-hop neighborhood graph centered at vertex v, it is a topological feature of vertex v on graph G. When topologically more similar or overlapped nodes are indexed closer to each other, the vertex ordering provides greater room for subsequent compression of the adjacency lists. Such ordering also potentially improves the spatial localities in overall adjacency accesses and, thereby, the efficiency in query response. Alternatively, a vertex ordering that increases the spatial localities in adjacency lists benefits graph compression. Briefly, grouping similar adjacency lists or minimizing their index distances is the key ingredient in an effective vertex ordering scheme.

There are several notable vertex ordering schemes for the compression of data-information graphs subject to the condition of maintaining real-time adjacency accesses [6], [11]. Some of them are specific to certain types of networks/graphs, followed by customized list compression techniques, or both. For instance, for web-graph compression, Silvestri suggested ordering the webpages by the lexicographic ordering of their uniform resource locator (URL) addresses [26]. The heuristic is that the similarity in adjacency patterns is well correlated with the URL proximity. Boldi and Vigna (BV) introduced a novel compression technique [7]. Their main idea is to encode the variation/gap in the neighborhood pattern of an adjacency list $\mathcal{N}(i)$ from a local prototype pattern among the adjacency lists $\mathcal{N}(j)$ within a small index block, say, 8 consecutive indices per block. The BV approach achieved a remarkable web-graph compression, 3 bits per web link, which was further reduced to 2 bits per link by others [6].

For social-network compression, Chierichetti, Kumar, Lattanzi et al. (CKL) advocated the use of the Shingle scores and ordering scheme, which were introduced by Gibson, Kumar, and Tomkins [8], [14]. The Shingle scores measure and encode the overlap/similarity among the 1-hop neighborhood graphs $\mathcal{N}(v)$, $v \in V$, and are subsequently used to order the vertices with a hashing function. We can expect the Shingle scores to be extended straightforwardly to h-hop neighborhood graphs $\mathcal{N}_h(v)$ with $h > 1$. Social networks tend to follow the power law in their degree distributions due to the preferential attachment as modeled or stylized by Barabasi and Albert (BA) [5]. A small number of nodes on a BA or BA-like network have very high degrees, their overreaching presence in many neighborhood graphs reduces the differentiation power by the Shingle scores. To overcome this problem, the CKL approach removes the nodes with high degrees, above d_τ, before ordering the other nodes by their neighborhood similarity in the remaining graph. The parameter d_τ is set above the average degree, and it can be determined from the degree distribution.

For BA-like networks/graphs, not necessarily social networks, an ordering scheme named SlashBurn was developed by Lim, Kang, and Faloutsos [21]. It first removes the top-ℓ high-degree nodes from the current graph. The removal criterion may be understood as the alternative to the CKL criterion. SlashBurn then decomposes the remaining graph into its connected components (CCs). The node-removal and CC-decomposition steps are applied to each of the connected components recursively. We may view SlashBurn extracting and expressing adjacency localities as layered connected components, not limited to neighborhood graphs with a fixed hop length as with the CKL approach. SlashBurn was shown superior to Shingle, by certain graph compression measures, in a benchmarking study with BA-like networks.

Interestingly, the Cuthill-McKee (CM) ordering or its reversal (RCM), and the Fiedler-spectral ordering, well known for their roles in sparse matrix computation [10], [13], are



(a) aps2020 (b) G_{ws} (c) T_{binomial}

Fig. 2: Pictorial description of the entries of Table I for three graphs—aps2020, G_{ws}, and T_{binomial}. The bars in solid colors represent the mLogGapA(G,π) scores: the lower, the better. They are between the lower and upper bounds in black lines by (7) and (8). The segments in lighter colors represent the $\Delta(G,\pi)$ values as structure indicators: a short light-colored segment over a short dark-colored bar indicates that more neighbors are placed within a short range on average. The grey bars represent the reference values in (9) and (10). It is a practical success when an ordering reaches close to the solid grey value on a non-elementary graph.

frequently used as base cases for performance comparisons among vertex ordering schemes for graph compression. Our first finding is that SlashBurn is outperformed, by and large, on BA-like graphs by AMD and NED, which are among the most effective ordering schemes for space-time efficient variable elimination [3], [4], [18], [22], although not obviously relevant to graph compression as RCM.

Our study of vertex ordering is comprehensive in two aspects: (a) we consider diverse types of sparse graphs, and (b) we analyze the match/mismatch between well-recognized graph types and popularly used ordering schemes in the context of adjacency access locality and graph compression. We establish theoretical and practical connections between vertex indexing for sparse graph/network compression and matrix ordering for sparse matrix-vector multiplication and variable elimination. These connections led us to the first comparison between SlashBurn and AMD.

With this work, we make two main contributions. First, we present a fundamental analysis of adjacency access locality (AAL) in vertex ordering from the perspective of graph/matrix composition of elementary compact graphs/matrices. The analysis is instrumental to vertex ordering studies in more than one aspect. It enables us to identify the lower and upper bounds on the feasible AAL scores, by the measures in (1) and (2), and, thereby, explore previously unknown potentials or limitations. We provide additional reference AAL values not only for assessing the performance of a vertex ordering but also for inferring the substructures captured by the ordering. Such a frame of reference was absent in previous performance assessments. More importantly, the analysis sheds light on new ways for obtaining better approximate solutions to the NP-hard vertex ordering problem.

Next, we present viFPS, a versatile indexing method for compression of diverse types of graphs. We maintain the advantageous features in the state-of-the-art indexing schemes and mitigate their shortcomings. The new method makes recursive *Fiedler* partition and ordering conditioned by what we refer to as the *Pareto Splits* in adaptation to the graph-degree statistics. We show in Sections III and IV the outstanding performance of viFPS on diverse types of graphs, competitive or superior to the state-of-the-art ordering schemes for each type of graph. As expected, the new method benefits sparse matrix computation as well. We show, in particular, improved efficiency for subspace iterations in responding to random-walk queries on a large and sparse network.

II. Adjacency access locality analysis

A. Measures

We adopt two well-established measures of vertex orderings for graph compression, with a slight modification. Denote by $G(V,E)$ the graph under consideration. One may assume that G is free of self-loops, which do not affect the ordering scores. Graph G is identified by its adjacency matrix A. Let $n = |V|$. Let $m = \mathrm{nnz}(A)$, the number of nonzeros in A. Then, $m = 2|E|$ when G is undirected and $m = |E|$ when G is directed. Denote by $\Pi(n)$ the set of all permutations of $(1:n)$. Chierichetti, Kumar and Lattanzi introduced in 2009 [8] the following two measures of adjacency access locality (AAL) captured by a vertex ordering $\pi \in \Pi(n)$,

$$\begin{aligned}\mathsf{mLogA}(G,\pi) &= \frac{1}{m}\sum_{(u,v)\in E}\log_2\left(1+|\pi(u)-\pi(v)|\right)\\ &= \frac{1}{m}\sum_{v\in V}\sum_{\substack{u\in\mathcal{N}(v)\\ u\neq v}}\log_2\left(1+|\pi(u)-\pi(v)|\right),\end{aligned} \tag{1}$$

$$\mathsf{mLogGapA}(G,\pi) = \frac{1}{m}\sum_{v\in V}1+\sum_{\substack{u_i\in\mathcal{N}[v]\\ i=2:d(v)}}\log_2\left(1+\pi(u_i)-\pi(u_{i-1})\right), \tag{2}$$

where the neighbors u_i of v are ordered by π, $\pi(u_i) > \pi(u_{i-1})$. We present the second expression in (1) in order to make the adjacency lists more explicit and to make the connection to and difference from (2) more salient. We make a slight modification in (2) by replacing $\log_2(1+|\pi(u_1)-\pi(v)|)$ with 1, effectively removing the host vertex v from its own adjacency list. This change makes the measure more closely related to general-purpose compression schemes, as shown in Figure 4. The measure mLogA is the average distance, in bit length, of the neighbors from each host vertex v; mLogGapA is the average gap in bit length between the successive neighbors of each host v. The goal is to find the vertex ordering that minimizes one of the measures or both. However, locating the optimum on an arbitrary graph is NP-hard [8] and computationally intractable. Practical ordering schemes resort to various heuristics.

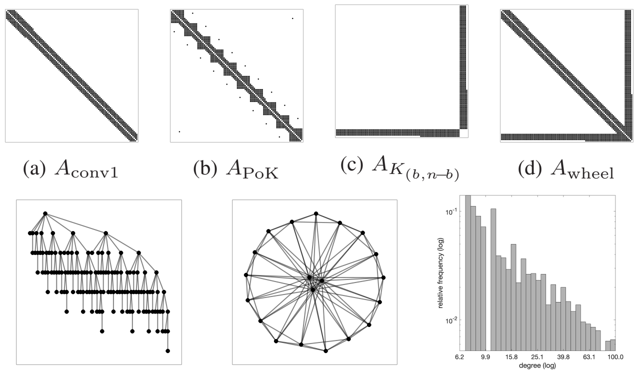
The average gap measure (2) is closely related to subsequent compression schemes. We make a novel use of the average distance measure (1) as well. The following two basic inequalities hold for any connected graph G,

$$1 \leqslant \mathsf{mLogGapA}(G,\pi) \leqslant \mathsf{mLogA}(G,\pi), \quad \forall\,\pi\in\Pi(n). \tag{3}$$

The absolute difference between the two measures is

$$\Delta(G,\pi) \triangleq \mathsf{mLogA}(G,\pi) - \mathsf{mLogGapA}(G,\pi). \tag{4}$$

We will demonstrate shortly how we utilize the differential information. We also use the basic statistics of the graph degrees. Frequently, we use the average degree $\bar{d}$. Conventionally, a graph is regarded as sparse if $\bar{d}\in O(\log_2 n)$.



(a) A_{conv1} (b) A_{PoK} (c) $A_{K_{(b,n-b)}}$ (d) A_{wheel}

(e) T_{binomial} (f) G_{wheel} (g) $d(G_{\mathrm{lfr}})$

Fig. 3: **Top:** The adjacency matrices of G_{conv1}, G_{PoK}, $K_{(b,n-b)}$ and G_{wheel} in their respective compact forms. **bottom:** T_{binomial} and G_{wheel} in 2D spatial displays, and the degree distribution of graph G_{LFR} in log-log scale.

B. Elementary compact sparse graphs

We make a formal AAL analysis of vertex ordering from the perspective of graph composition or decomposition. This perspective is not foreign. Beneath their differences in algorithms and results, existing vertex ordering methods are common in that each has a strategy for extracting certain subgraphs that are more compressible, and it is used in tandem with an ordering priority that such subgraphs be overlapped as much as possible. We make an explicit AAL analysis of three elementary types of sparse graphs that are frequently used to prototype and approximate subgraphs in a larger graph and are highly compressible. The corresponding adjacency matrices in their respective optimal vertex orderings are among the most compact sparse matrices.

First, the convolution network/graph with a narrowly banded adjacency matrix is perhaps a more familiar type for subgraphs of a large sparse graph. When $\bar{d} > 2$, the one-dimensional convolution graph G_{conv1} is next to the highest sum of local cluster coefficients. It is the base reference for modeling and generating (via edge rewiring) the small-world networks G_{ws} by the Watts-Strogatz model [28]. The adjacency matrix A in the RCM ordering π_{RCM} is banded with the minimal bandwidth. If G_{conv1} is not circulant, then

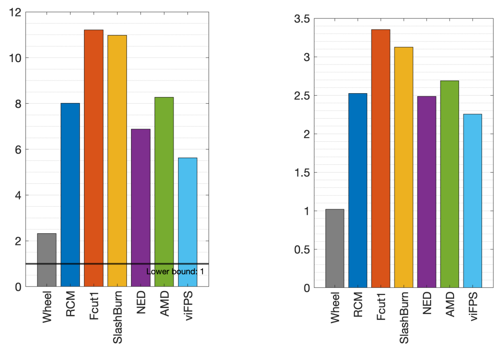
$$\begin{aligned}\mathsf{mLogGapA}(G_{\mathrm{conv1}},\pi_{\mathsf{RCM}}) &= \min_{\pi\in\Pi(n)}\mathsf{mLogGapA}(G_{\mathrm{conv1}},\pi),\\ &= 1+\gamma_1\frac{\log_2(3)-1}{\bar{d}}\end{aligned} \tag{5}$$

and

$$\begin{aligned}\mathsf{mLogA}(G_{\mathrm{conv1}},\pi_{\mathrm{rcm}}) &= \min_{\substack{G:m/n=\bar{d}\\ \pi\in\Pi(n)}}\mathsf{mLogA}(G,\pi),\\ \frac{b-1}{b}\log_2\left(\frac{b}{2}\right) &\leqslant \Delta(G_{\mathrm{conv1}},\pi_{\mathrm{rcm}}) \leqslant \log_2(b),\end{aligned} \tag{6}$$

where b is the semi-bandwidth, $b = \lceil\bar{d}/2\rceil$. Here, γ_1 denotes a value close to 1 and varies little with $\bar{d}$.

Next is the graph $G_{\mathrm{PoK}}(n,\bar{d})$ constructed as a linear path/chain of cliques $K_{\bar{d}}$, see Figure 3. It is a prototype sparse graph consisting of equally populated sub-communities with the maximal intra-community connection and the minimal inter-community connection [12], [27]. The removal of any



(a) mLogGapA (b) file sizes in HDF5

Fig. 4: A demonstration of the close correlation between mLogGapA of (2) on vertex ordering and HDF5 for general-purpose compression. **Left:** the mLogGapA scores of six different orderings on the citation graph aps2020; **Right:** the corresponding file sizes in HDF5 format. The file sizes are normalized by that for the reference graph G_{wheel} of the same size and sparsity in the AMD ordering, based on (9).

inter-clique link decouples the graph. The adjacency matrix A_{PoK} is of nearly block diagonal form by the path ordering, which is also of the minimal bandwidth. Although the semi-bandwidth is $\bar{d}$ instead of $\bar{d}/2$, the number of bits per link is doubled. The difference from the minimal on G_{conv1} is small in mLogA, no greater than $\log_2(b)/\bar{d}$ bits per link, and even smaller in mLogGapA, no greater than $\log_2(b)/(1+\bar{d}^2/2)$. In short, G_{PoK} can be treated almost the same as G_{conv1}.

Thirdly, the most primitive for modeling compact subgraphs of a sparse graph/network is the biclique $K_{(b,n-b)}$. In the special case $b = 1$, $K_{(1,n-1)}$ is the star graph. Any tree graph is composed of star subgraphs. For $b > 1$, the biclique $K_{(b,n-b)}$ can be viewed as a block version of a star graph with b center nodes, every center node is connected to all $n-b$ peripheral nodes. If the sparsity is specified by $\bar{d}$, the biclique is assumed to have $m = \bar{d}n$ links with $b = \lceil \bar{d}/2 \rceil$, it is incomplete if $\bar{d}$ is not an even integer. In the case of $\bar{d} = O(\log n)$, $K_{(b,n-b)}$ has a large gap in the degrees between the b center nodes and the rest. This degree inequality is a typical phenomenon in many social or biomolecular networks. The adjacency matrix of $K_{(b,n-b)}$ is of low rank, regardless of the ordering, while its counterparts for G_{conv1} and G_{PoK} are nearly regular and close to full rank. In its compact form, the adjacency matrix has a block row and a block column when the center nodes are indexed together, such as by π_{AMD}, the AMD ordering. There are several remarkable properties with the biclique compression.

Proposition 1. *Among all sparse graphs with the average degree $\bar{d}$ and over $\Pi(n)$ of any size n, the minimal* mLogGapA *score is achieved by π_{AMD} on $K_{(b,n-b)}$, $b = \lceil \bar{d}/2 \rceil$,*

$$\min_{\substack{G:m/n=\bar{d}\\ \pi\in\Pi(n)}} \text{mLogGapA}(G,\pi) = \text{mLogGapA}(K_{(b,n-b)},\pi_{\text{AMD}}) = 1. \quad (7)$$

Moreover, there is the minimum-equivalence set that admits any ordering π on $K_{(b,n-b)}$ if, and only if, π places the b center nodes together.

By the proposition, any non-circulant shift of π_{AMD} is an optimal ordering on $K_{(b.n-b)}$. Regarding the necessary condition, if the high-degree nodes are dispersed far apart by an ordering π, the score can be as large as, although bounded by, mLogA$(K_{(b.n-b)}, \pi_{\text{AMD}})$. We omit the special detail and present the general relations, beyond (3), between the two scores by mLogA and mLogGapA.

Proposition 2. *The feasible* mLogGapA *scores on all sparse graphs of the same average degree $\bar{d}$ and over $\Pi(n)$ of any size n are closely bounded from above as follows, $b = \lceil \bar{d}/2 \rceil$,*

$$\begin{aligned}\max_{\substack{G:m/n=\bar{d}\\ \pi\in\Pi(n)}} \text{mLogGapA}(G,\pi) &\leqslant \text{mLogA}(K_{(b,n-b)},\pi_{\text{AMD}}) \\ &= 1+\gamma_1\log_2(1+n-b).\end{aligned} \quad (8)$$

Proposition 3. *For an ordering π on graph G, the 2-tuple $[\,\text{mLogGapA}(G,\pi), \Delta(G,\pi)\,]$ is a descriptor of the substructures of $A(\pi,\pi)$, a larger relative difference indicates a larger portion of non-zero elements being off from the $\bar{d}(G)$ main diagonals.*

In more detail, when mLogGapA is close to the baseline at 1, then $A(\pi,\pi)$ is nearly banded or block-diagonal. If $\Delta(G,\pi)$ is close to 0 in addition, then $A(\pi,\pi)$ is narrowly banded or block-diagonal within a narrow band. At the other extreme, the following condition,

$$\text{mLogA}(G,\pi) > \text{mLogA}(K_{(b,n-b)},\pi_{\text{AMD}}) \quad (9)$$

suggests an improvement or replacement of the ordering π.

C. Composition effects

Among numerous ways of graph composition, we make a simple abstraction of their effects on the adjacency locality in vertex ordering. Fix the size $n = |V|$. Let graph $G_{\text{wheel}}(n, b_l, b_g)$ be the sum of $K_{(b_g,n-b_g)}$ and $G_{\text{conv1}}(b_l)$ of semi-bandwidth b_l, $b_l b_g > 0$. Then, $\bar{d}(G_{\text{wheel}}) = 2(b_l + b_g)$. Among the $n-b_g$ low-degree vertices, each has $2b_l$ neighbors that are locally connected and has direct links to the b_g global center nodes. The b_g center nodes have nearly half or more of the total links when $b_g \geqslant b_l$. Graph G_{wheel} is of a wheel shape, see Figure 3. In terms of local and global connectivities, G_{wheel} combines the best features from each component graph. In fact, it is the small-world graph with the maximal sum of the local cluster coefficients and the minimal diameter among all graphs of the same size and sparsity. In the context of graph compression, however, the wheel composition effect may be both startling and elucidating. The lowest gap score on G_{wheel} can be much greater than the sum of the individual components,

$$\text{mLogGapA}(G_{\text{wheel}},\pi_{\text{AMD}}) = 1+\gamma_1\frac{\log_2(n-\bar{d})}{\bar{d}}, \quad b_1b_2 > 0. \quad (10)$$

The least number of bits per link for G_{wheel} increases with $\log_2(n)/\bar{d}$ roughly, whereas it is 1 for $K_{(b_g,n-b_g)}$ and close to 1 for $G_{\text{conv1}}(b_l)$, invariant to variation in n. Due to the sequential nature of vertex ordering, the composition inevitably introduces gaps in every adjacency list between the b_l locally-connected grass-roots neighbors and the b_g global center nodes. The gap length varies from 1 up to $n-\bar{d}/2$.

The composition effect expressed by (10) also has brighter implications. First, if G_{wheel} is not super sparse, such as when $\bar{d} \geqslant \log_2(n)$, then it is highly compressible. Secondly, a large and sparser network in the real world is typically composed of

TABLE I: Empirical assessment of adjacency access localities and subgraph structures captured by 6 different vertex ordering schemes on 18 sparse graphs of diverse types. SlashBurn is the baseline. Each graph G has $|V|$ vertices and average degree $\bar{d}$. The number of nonzero elements in the adjacency matrix A is $\text{nnz}(A) = \bar{d}|V|$. For the synthetically generated graphs, except the binomial tree G_{binomial}, $|V| = 250K$ with K = 1,000 and $\bar{d} = 14$. Each table entry is a 2-tuple descriptor $[\,\mathsf{mLogGapA}(G,\pi) \mid \Delta(G,\pi)\,]$. The first element is the number of bits per link on average by ordering π on graph G, the sum of the two elements is the mLogA score, namely, the average neighbor distance in bit length, and the ratio of the first element to the second indicates the average portion of links within the $\bar{d}/2$ range in each adjacency list, as stated in Proposition 2. On each graph, the best mLogGapA score is highlighted in bold case, the runner-up is underlined. Figure 2 depicts the table entries for three of the graphs.

		$n = \|V\|$	nnz(A)	$\bar{d}$	RCM		Fcut1		SlashBurn		NED		AMD		viFPS	
synthetic	G_{conv1}	250 K	3.5 M	14	$\underline{1.0}$	1.1	3.3	1.8	11.4	4.0	1.7	1.6	$\underline{1.0}$	1.1	**1.0**	1.1
	G_{PoK}	250 K	3.5 M	14	$\underline{1.1}$	1.3	5.1	4.7	10.5	3.7	1.7	2.3	1.1	1.3	**1.1**	1.3
	$K_{(7,n-7)}$	250 K	3.5 M	14	1.0	15.6	1.0	15.6	1.0	15.6	1.0	15.6	1.0	15.6	1.0	15.6
	G_{wheel}	250 K	3.5 M	14	3.1	13.3	3.1	13.3	3.1	13.3	2.5	12.4	$\underline{2.2}$	12.1	**2.2**	12.1
	G_{ws}	250 K	3.5 M	14	5.3	5.3	9.6	2.8	11.4	4.3	5.3	3.5	$\underline{5.0}$	3.9	**2.5**	2.3
	G_{lfr}	250 K	3.5 M	14	$\underline{5.4}$	8.8	6.3	8.2	6.8	8.3	7.0	8.2	6.3	8.6	**5.1**	4.7
	T_{binomial}	262 K	0.52 M	2	4.6	9.7	7.6	6.6	8.1	7.8	2.8	1.7	$\underline{1.9}$	0.9	**1.7**	0.6
real-world	Polbooks [23]	105	882	8	**1.8**	1.4	1.9	1.7	2.3	2.3	2.0	1.9	2.0	2.1	$\underline{1.9}$	1.2
	Football [15]	115	613	5	2.3	1.7	$\underline{2.2}$	1.5	2.6	2.5	2.5	2.0	2.4	2.2	**1.9**	1.1
	URV email [17]	1.1 K	10.9 K	10	4.0	3.1	4.0	2.5	4.3	3.2	$\underline{3.8}$	3.0	$\underline{3.8}$	2.9	**3.3**	1.9
	Polblogs [1]	1.2 K	19 K	16	$\underline{1.0}$	5.9	3.2	4.0	3.4	5.3	3.5	5.1	2.9	5.6	**1.0**	5.9
	Powergrid [28]	4.9 K	13.2 K	3	3.5	3.3	4.0	3.0	4.1	2.8	2.9	2.0	$\underline{2.3}$	1.0	**2.1**	0.5
	Pothen-Barth [24]	6.7 K	46.2 K	7	2.9	1.3	4.8	1.1	6.8	1.9	3.1	1.1	$\underline{2.5}$	1.1	**2.1**	0.7
	Enron email [20]	36 K	361 K	11	4.8	5.7	5.5	5.3	5.7	5.3	4.7	5.4	$\underline{4.5}$	5.4	**3.9**	5.5
	APS-2020 [2]	0.67 M	8.85 M	13	8.0	8.5	11.2	5.2	11.0	5.9	$\underline{6.9}$	8.6	8.3	8.6	**5.6**	10.2
	Flickr [16]	0.82 M	9.84 M	12	$\underline{6.0}$	11.8	6.5	7.2	6.8	7.1	6.2	10.8	6.1	11.7	**5.3**	12.2
	YouTube [29]	1.1 M	6 M	5	7.5	9.1	7.8	8.4	8.6	7.8	8.0	6.6	$\underline{7.3}$	6.9	**7.3**	6.9
	LiveJournal [29]	4 M	69.4 M	17	9.8	7.3	12.2	5.6	12.7	5.5	$\underline{9.1}$	5.9	10.8	5.8	**9.1**	5.9

smaller subnetworks/subgraphs across multiple layers/levels, the grass-roots nodes do not necessarily have direct links to the elite nodes at the top. Consider the wheel graphs as subgraphs of a larger network. For instance, let G be a chain of wheel-shaped subgraphs of size n_s. Then, the number of extra bits per link is roughly $\log_2(n_s)/\bar{d}$ instead of $\log_2(n)/\bar{d}$. Here, the assumed homogeneity in the subgraph size and shape is for brevity in describing the composition/decomposition effects.

Our abstraction of graph composition and decomposition with regard to vertex ordering for graph compression helps algorithmic thinking or rethinking. A Delaunay triangulation graph can be viewed as a composition of wheel sub-graphs and star subgraphs, although not necessarily in a single chain. In general, a large, sparse graph can be decomposed into elementary subgraphs and wheel-like graphs, which may have missing spokes or missing arcs, the interconnection between the subgraphs is sparser.

We utilize the information and insight from the above analysis for evaluating and developing a vertex ordering scheme, as well as for inferring the local and global connectivity structures of a graph/network. We use particularly the important reference values in (7), (8), (9) and (10).

III. Performance assessment of existing methods

We report our empirical assessment of five existing vertex ordering schemes on their effectiveness and specificity/versatility. We use 18 sparse graphs for the assessment. An ordering scheme is effective for an intended class of graphs if it maintains decent performance over variations within the class. A scheme is more versatile if it is effective across multiple types or classes of graphs. In Table I, we list the schemes, the graph types and sizes, data sources, and tabulate the mLogGapA scores and the differential scores. We provide illustrations in Figure 2. We describe below the rationale behind the chosen schemes and graphs and present our findings.

The five vertex ordering schemes are RCM, Fcut1, SlashBurn, AMD, and NED, as discussed in Section I. We denote by Fcut1 the vertex ordering in the Fiedler eigenvector of the normalized graph Laplacian. It is commonly used for graph cut (binary partition) in parallel sparse matrix computation [25]. We introduce the comparisons with AMD and NED, besides RCM and Fcut1. The underlying idea of AMD is to approximately minimize the degrees in successive elimination graphs [3], [4]. In the context of vertex ordering for graph compression, the idea can be interpreted and understood as *ordering the vertices from the least shared connections/dependencies to the most common connections/dependencies.* NED (nested dissection) applies AMD to the graph minors contracted by recursive bisections, each node of a graph minor is a subgraph of the original graph G. The graph dissection and contraction are proven effective on graphs of geometrically two/three-dimensional connectivity [18]. The Shingle scheme [8] is not included as it behaves similarly and is reportedly inferior to SlashBurn [21]. Unlike the other four schemes, SlashBurn requires a hyper-parameter, which specifies the number of high-degree nodes to be removed from every connected component, recursively. We locate the best parameter value we could for SlashBurn in every experiment.

The graphs used for the assessment include 7 synthetic graphs for controlled studies and 11 real-world networks for applicability assessment, see Table I. The first four synthetic graphs are described in Section II. The additional three represent, respectively, well-recognized graph/network types for real-world networks and/or technological networks. Graph G_{WS} is a Watts-Strogatz graph with relatively high local cluster coefficients and a small diameter of $O(\log(n))$. Graph G_{LFR} is generated by the LFR simulator [19], with BA-like subcommunities of different sizes interconnected by sparser, random links. Graph T_{binomial} is a binomial tree typically used in algorithms/architectures for network routing, priority queuing, and option pricing, among other applications [9]. The real-world networks are among the most frequently used for benchmarking, except graph aps2020. Covering the APS publications over the time span longer than a century, the citation graph aps2020, as depicted in Figure 1, is incredibly valuable in its own right to network studies. For each graph, we shuffle its adjacency matrix randomly in order to reduce the influence of the given or generation sequence on RCM, AMD, and NED. The other schemes, SlashBurn and Fcut1, are invariant to the initial ordering.

There are four key takeaways from Table I. (D-i) On the three elementary graphs, all schemes reach the ideal score of (7) on the biclique, the first two graphs expose the specificity and limitation of SlashBurn and Fcut1, which are far short of reaching the ideal score of (5). (D-ii) On the small-world graphs, the performance of SlashBurn degrades more from G_{wheel} of diameter 2 to G_{ws} of diameter $O(\log(n))$, in comparison to the others. (D-iii) Even on its intended graphs with heavy-tail degree distributions, SlashBurn is consistently outperformed by AMD and NED. (D-iv) On networks with subcommunity structures, G_{PoK}, G_{LFR}, G_{football}, G_{polblogs}, G_{polbooks}, G_{aps2020} and G_{Flickr}, AMD is behind RCM. In summary, each scheme is limited to certain type(s) of graphs. By our analysis, each scheme is also short of reaching the theoretically expected in its favorite type(s) of graphs.

IV. The new method: viFPS

We introduce a new method, viFPS. In Table I, it demonstrates superior and versatile performance in graph compression across diverse types of networks and graphs. Figure 5 shows its additional benefits in improving the efficiency of subspace iterations with a sparse matrix. We delineate its principled properties and describe its simple procedure.

The development of viFPS originates from our better understanding of the advantageous features and shortcomings of the existing methods. RCM keeps locally connected neighbors as close as possible. AMD adaptively, although implicitly, separates the globally shared neighbors from the locally shared ones. SlashBurn is recursive but inflexible in adjusting its removal of high-degree nodes through its recursive division process, and it is unable to decouple the graph at the weakest links as Fcut1 does. Absent multi-resolution graph partitions, RCM, Fcut1 and AMD are limited in extracting substructures as NED and SlashBurn do. All but Fcut1 are combinatorial algorithms and sensitive to structural or random variation.

We educe three principled properties for a versatile vertex ordering method to acquire. (a) The persistent effectiveness in the presence of inevitable variations, structural or random, within an intended class or type of networks/graphs. (b) The generalizability to diverse types of networks/graphs. This property warrants adaptability to the degree distribution. According to the AAL analysis in Section II, this also entails the algorithmic capability to decompose a graph into locally closely connected sub-communities detached from their commonly affiliated nodes, at multiple resolution levels. (c) There is an explicit, easily interpretable, and computationally efficient approximation path, as with Fcut1, to the minimization of mLogGapA of (2) over $\Pi(n)$. Almost counterintuitively,

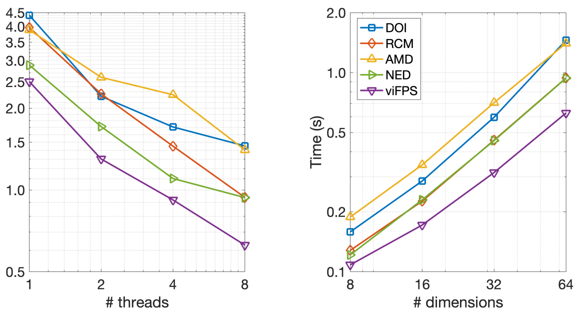


Fig. 5: The execution time in subspace iteration in responding to random-walk queries on the citation graph aps2020 with 5 different vertex orderings: DOI, RCM, AMD, NED and viFPS. The measured time (sec) is for 10 iterations $x_{k+1} = Ax_k$ with the adjacency matrix A and d iterate vectors x_k, d is the subspace dimension, on the Apple M2 Max processor, using the package SparseArrays in Julia. **Left:** The time variation with the number of threads $p \in \{1, 2, 4, 8\}$, at $d = 64$. **Right:** The time variation with the subspace dimension $d \in \{8, 16, 32, 64\}$, at $p = 8$. **Observation:** viFPS shows better parallel scalability and cache utilization.

making choices and modifications by these properties, we arrive at a remarkably simple, recursive procedure.

Provided at input the adjacency matrix A for a graph G, a pair of Pareto ratios (rvol, rminor), and a basecase vertex set size $n_{\min}$, viFPS returns a vertex permutation. The procedure is recursive, it takes the following steps on the current graph G. (1) If in the basecase $|V(G)| \leqslant n_{\min}$, apply AMD to G and return the permutation. (2) The Pareto Split. If rvol% of the total volume $\sum_{i\in V} d(i)$ is held by rminor% or less of the nodes, split the minority nodes from the majority. Denote by G_{major} the graph induced by the majority nodes. (3) The Fiedler cuts. Apply Fcut1 to every connected component of G_{major}, followed by a recursive call of viFPS to every divided subgraph. (4) Aggregation. Return the permutation aggregated from the split and cuts.

The procedure extends to any digraph G, such as the citation graph aps2020, or a bipartite, via the embedding matrix $[0, A; A^{\mathrm{T}}, 0]$. It returns both a row permutation and a column perturbation. We omit the rationale and nuance details due to the document length limit.

The time complexity of viFPS scales with $c\,m\log_2(n)$, where $m = |E|$, $n = |V|$, and c is a modest constant, proportional to the low dimension of a subspace iteration for obtaining the single Fiedler vector of a sparse subgraph.

In the viFPS approach, the algebraic Fiedler cuts extract the substructures, the statistical Pareto splits adapt to the degree distributions of the divided subgraphs. If the split condition by the ratio pair is not met on a subgraph, no split takes place. All splits can be deactivated by setting the ratio pair as $(\text{rvol}, \text{rminor}) = (100, 1)$. Only in the ideal case of a Pareto distribution, the split condition can be set by the single Pareto scale parameter. We find that an estimate of the scale parameter for a real-world network can be unreliable. For every graph/network in Table I, we use a global split condition $(\text{rvol}, \text{rminor}) = (20, 4)$, which guides the adaptive split in each divided subgraph. One can adjust or tune the split control, globally or recursively, in an attempt to further improve the compression, via parallel search or coordinated search. The granularity for such parameter tuning is to be further investigated in the cost-effectiveness aspect.

Acknowledgements

We thank Mark Doyle and the American Physical Society (APS) for providing the citation graph of APS publications up to the year 2020. We thank an anonymous reviewer for helpful suggestions on further clarification of the time-complexity scaling and the Pareto-split condition setting.

References

[1] L. A. Adamic and N. Glance. “The Political Blogosphere and the 2004 U.S. Election: Divided They Blog”. In: *Proceedings of the 3rd International Workshop on Link Discovery*. KDD05: The Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Chicago Illinois: ACM, Aug. 21, 2005, pp. 36–43.

[2] American Physical Society. *Dataset of Citing Article Pairs*. https://journals.aps.org/datasets. 2020.

[3] P. R. Amestoy, T. A. Davis, and I. S. Duff. “An Approximate Minimum Degree Ordering Algorithm”. In: *SIAM Journal on Matrix Analysis and Applications* 17.4 (Oct. 1996), pp. 886–905.

[4] P. R. Amestoy, T. A. Davis, and I. S. Duff. “Algorithm 837: AMD, an Approximate Minimum Degree Ordering Algorithm”. In: *ACM Transactions on Mathematical Software* 30.3 (Sept. 2004), pp. 381–388.

[5] A.-L. Barabási and R. Albert. “Emergence of Scaling in Random Networks”. In: *Science* 286.5439 (1999), pp. 509–512.

[6] M. Besta and T. Hoefler. *Survey and Taxonomy of Lossless Graph Compression and Space-Efficient Graph Representations*. Apr. 27, 2019. arXiv: 1806.01799 `[cs, math]`. URL: http://arxiv.org/abs/1806.01799 (visited on 07/05/2024). Pre-published.

[7] P. Boldi and S. Vigna. “The Webgraph Framework I: Compression Techniques”. In: *Proceedings of the 13th International Conference on World Wide Web*. WWW04: The 2004 World Wide Web Conference (in Conjunction with ACM Conference on Electronic Commerce [EC’04]). New York NY USA: ACM, May 17, 2004, pp. 595–602.

[8] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. “On Compressing Social Networks”. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD09: The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Paris France: ACM, June 28, 2009, pp. 219–228.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. Fourth edition. Cambridge, Massachusett: The MIT Press, 2022. 1291 pp.

[10] E. Cuthill and J. McKee. “Reducing the Bandwidth of Sparse Symmetric Matrices”. In: *Proceedings of the 1969 24th National Conference On -*. The 1969 24th National Conference. Not Known: ACM Press, 1969, pp. 157–172.

[11] L. Dhulipala, I. Kabiljo, B. Karrer, G. Ottaviano, S. Pupyrev, and A. Shalita. “Compressing Graphs and Indexes with Recursive Graph Bisection”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco California USA: ACM, Aug. 13, 2016, pp. 1535–1544.

[12] S. Fortunato and M. Barthelemy. “Resolution Limit in Community Detection”. In: *Proceedings of the National Academy of Sciences* 104.1 (2007), pp. 36–41.

[13] A. George, J. W. H. Liu, and J. W. Liu. *Computer solution of large sparse positive definite systems*. Prentice-Hall series in computational mathematics. Englewood Cliffs, NJ: Prentice-Hall, 1981. 324 pp.

[14] D. Gibson, R. Kumar, and A. Tomkins. “Discovering Large Dense Subgraphs in Massive Graphs”. In: *Proceedings of the 31st International Conference on Very Large Data Bases*. 2005, pp. 721–732.

[15] M. Girvan and M. E. J. Newman. “Community Structure in Social and Biological Networks”. In: *Proceedings of the National Academy of Sciences* 99.12 (June 11, 2002), pp. 7821–7826.

[16] D. Gleich. *Graph of Flickr Photo-Sharing Social Network Crawled in May 2006*. Version 1.1. Purdue University Research Repository, 2012.

[17] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. “Self-Similar Community Structure in a Network of Human Interactions”. In: *Physical Review E* 68.6 (Dec. 17, 2003), p. 065103.

[18] G. Karypis and V. Kumar. “A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs”. In: *SIAM Journal on Scientific Computing* 20.1 (Jan. 1998), pp. 359–392.

[19] A. Lancichinetti, S. Fortunato, and F. Radicchi. “Benchmark Graphs for Testing Community Detection Algorithms”. In: *Physical Review E* 78.4 (Oct. 24, 2008), p. 046110.

[20] J. Leskovec, J. Kleinberg, and C. Faloutsos. “Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations”. In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. KDD05: The Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Chicago Illinois USA: ACM, Aug. 21, 2005, pp. 177–187.

[21] Y. Lim, U. Kang, and C. Faloutsos. “SlashBurn: Graph Compression and Mining beyond Caveman Communities”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.12 (Dec. 1, 2014), pp. 3077–3089.

[22] D. Marx, ed. *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Philadelphia, PA: Society for Industrial and Applied Mathematics, Jan. 2021.

[23] M. E. J. Newman. “Modularity and Community Structure in Networks”. In: *Proceedings of the National Academy of Sciences* 103.23 (June 6, 2006), pp. 8577–8582.

[24] A. Pothen. “Graph Partitioning Algorithms with Applications to Scientific Computing”. In: *Parallel Numerical Algorithms*. Ed. by D. E. Keyes, A. Sameh, and V. Venkatakrishnan. Red. by M. D. Salas. Vol. 4. Dordrecht: Springer Netherlands, 1997, pp. 323–368.

[25] A. Pothen, H. D. Simon, and K.-P. Liou. “Partitioning Sparse Matrices with Eigenvectors of Graphs”. In: *SIAM Journal on Matrix Analysis and Applications* 11.3 (July 1990), pp. 430–452.

[26] F. Silvestri. “Sorting Out the Document Identifier Assignment Problem”. In: *Advances in Information Retrieval*. Ed. by G. Amati, C. Carpineto, and G. Romano. Vol. 4425. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 101–112.

[27] V. A. Traag, P. Van Dooren, and Y. Nesterov. “Narrow Scope for Resolution-Limit-Free Community Detection”. In: *Physical Review E* 84.1 (July 29, 2011), p. 016114.

[28] D. J. Watts and S. H. Strogatz. “Collective Dynamics of ‘Small-World’ Networks”. In: *Nature* 393.6684 (June 1998), pp. 440–442.

[29] J. Yang and J. Leskovec. “Defining and Evaluating Network Communities Based on Ground-Truth”. In: *Knowledge and Information Systems* 42.1 (2015), pp. 181–213.