SYMMETRY-BREAKING SYMMETRY IN DIRECTED SPECTRAL PARTITIONING

DIMOSTHENIS PASADAKIS[†], RAPHAEL S. STEINER[†], PÁL ANDRÁS PAPP, TONI BÖHNLEIN, AND ALBERT-JAN N. YZELMAN

Abstract. We break the symmetry in classical spectral bi-partitioning in order to incentivise the alignment of directed cut edges. We use this to generate acyclic bi-partitions and furthermore topological orders of directed acyclic graphs with superb locality. The new approach outperforms the state-of-the-art Gorder algorithm by up to 17× on total reuse distance and minimum linear arrangement.

Contents

1. Introduction

Graph partitioning plays a central role in a variety of applications, for example classification, recommendation systems, parallel processing, very-large-scale integration (VLSI), and routing to name a few. The basic premise is often the same: identifying tightly knit groups of roughly the same size with few interconnects. To this end, a plethora of methods and algorithms have been developed ranging from geometric, spectral, local-search, multi-level, evolutionary methods, and combinations of the aforementioned. We refer to the surveys [Fjä98, SKK+00, MPSG07, KHKM11, BS13, BMS+16, ÇDF+23].

A specialisation of the graph partitioning problem is acyclic partitioning, where one is given a directed acyclic graph and the partition itself is required to form a directed acyclic graph[1]. This stricter problem is required, for example, when the directed acyclic graph is modelling a computation with dependencies. As such, it has been used for pipeline computations, efficient uses of the memory hierarchy, and, in general, as a component of offline scheduling heuristics [AFK+12, FER+13, ERP+15, ÖBÇ19, PAKY24].

Between undirected and acyclic graph partitioning, there also lies a more relaxed partitioning problem: given a directed graph, one seeks to find a partition such that between any two parts the cut edges mostly point in one direction. This type of partitioning can be used as

Date: August 25, 2025.

Key words and phrases. Spectral partitioning, (nearly) acyclic partitioning, directed acyclic graph (DAG) partitioning, graph layout, linear ordering, linear arrangement, bandwidth, cut width, reuse distance.

[†]Joint first authors; listed in alphabetical order.

[1]Specifically the quotient graph with self-loops removed.

initialisers for acyclic partitions and for flow-based clustering [HAP22], such as hierarchy classifications [VLCD19], interbank debts [AOTS15], and population migration patterns [ZDC25].

In this paper, we present a modification to the classic spectral partitioning method of Fiedler [Fie73, Fie89] which yields an aforementioned nearly acyclic bi-partition. In the case of a directed acyclic graph, we furthermore augment the nearly acyclic bi-partition to an acyclic bi-partition. To the best of our knowledge, this marks the first time spectral methods have been used to address the problem of acyclic partitioning.

As the final and central application of our method in this paper, we turn to the graph-layout problem, also known as the linear-arrangement problem, of directed acyclic graphs. The latter asks for a topological order with 'good locality'. Specifications of locality take on a variety of shapes. The most well-known metrics to evaluate the locality of an ordering include the bandwidth, cut width, minimum linear arrangement (sum of edge lengths), and reuse distance. The associated problems have been well-studied, both from a theoretical and a practical point of view, as they are widely applicable. Applications range from VLSI design, single-core and multi-core scheduling, information retrieval, cache utilisation, network reliability, to speeding up graph and sparse matrix algorithms [Chu88, DPS02, NS12, Pet13].

In due course, we will show that our method excels at generating topological orders for data and temporal locality. Indeed, we report significant improvements over an acyclic adaptation of the state-of-the-art method Gorder [WYLL16]: up to 28× smaller minimal linear arrangement, up to 17× smaller total reuse distance, 11× smaller cut width, and 6× smaller bandwidth and maximum reuse distance. Figure 1.1 shows an example with 10× smaller minimal linear arrangement and total reuse distance.

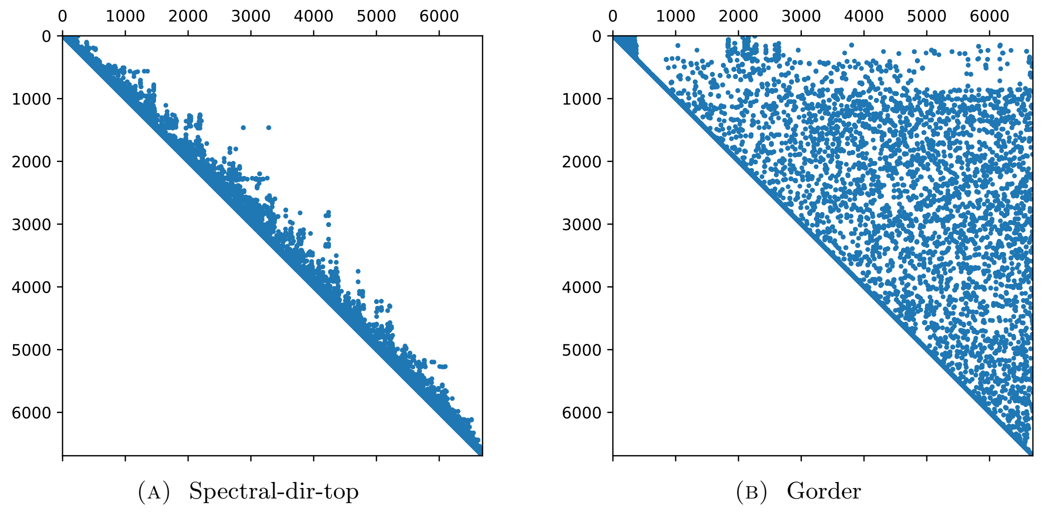


FIGURE 1.1. Topological orderings generated by (A) our direction-incentivised spectral topological-order method (Spectral-dir-top) and (B) an acyclic adaptation of Gorder (Gorder). The directed acyclic graph corresponds to the lower triangular part of the matrix 'barth'.

1.1. Nearly acyclic bi-partition. There are several spectral methods in the literature that deal with directed graphs. Broadly speaking, they can be categorised into two camps: flow-based and density-based [HAP22]. Examples for flow-based techniques include clustering on an embedding based on spectral information of the (dampened) page-rank transition matrix [VLCD19] or Hermitian variants [GM17, Moh20, LS20], bibliographic coupling matrix [Kes63], co-citation matrix [Sma73], and linear combinations thereof [SP11]. Their main feature is that they group vertices together if they have a lot of in- and/or out-neighbours in common, but disregard grouping vertices together which are tightly knit. This makes them suitable for detecting hierarchies and flows, but unsuitable for our purposes. On the other hand, density-based techniques such as using the directed Laplacian [ZHS05, HZS06, Gle06] group tightly knit vertices, but generally fail to have most cut edges between groups aligned.

In §2 of this paper, we address this gap. We achieve this by taking density-based spectral partitioning methods, such as the undirected Laplacian [Fie73, Fie89], p-Laplacian [BH09, SPK+18], or the directed Laplacian [Chu05], and modify them as to incentivise the alignment of cut edges. The general idea is as follows. Vaguely speaking, the aforementioned methods assign a vertex the value 1 if they belong to the first cluster and -1 if they belong to the second cluster. The value of a directed edge is given as the (signed) difference of the source and target of the edge. Thus, the value is 0 for a non-cut edge, 2 for a cut edge from the first cluster to the second, and -2 for a cut edge pointing in the reverse direction. Therefore, the product of the values of two cut edges tells us now whether they are aligned or not. Hence, summing up all these products over all pairs of edges gives us a term which penalises non-aligned cut edges. As this penalty term constitutes a quadratic form, we are able to incorporate it into spectral partitioning methods.

1.2. Acyclic bi-partition.

Like balanced undirected bi-partitioning, balanced acyclic bi-partitioning is NP-hard [GJS74, MPS17]. Yet, the latter is provably harder in the following sense. For ε-balanced[2] undirected bi-partitioning, there exists a polynomial-time $O(\log(|V|))$-approximation algorithm by Räcke [Räc08], whereas we show that no such algorithm exists for ε-balanced acyclic bi-partitioning under the exponential-time hypothesis [IP01].

Theorem 1.1. *Assuming the exponential-time hypothesis, there is a $\delta > 0$ such that for every $0 \leq \varepsilon < 1$ the ε-balanced acyclic bi-partition problem does not have a polynomial-time $n^{1/\log(\log(n))^{\delta}}$-factor approximation algorithm, where $n = |V|$ is the size of the input graph.*

Despite the NP-hardness result, there are a variety of fast heuristics and algorithms in the literature ranging from topological-order-based [Ker71], to acyclic adaptations of the Fiduccia–Mattheyses method [CLB94, PSSS21], evolutionary algorithms [SSSW17, MPS18], and multi-level algorithms [HKU+17, HOU+19, PSSS21]. We refer to the survey [ÇDF+23] and the references therein for further information.

In §3 of this paper, we present an acyclic bi-partitioner based on our nearly acyclic spectral bi-partitioner. To fix the acyclicity, we generate a topological order of the directed acyclic graph, which aims to preserve the bi-partition, and subsequently bisect said topological order. To the best of our knowledge, this marks the first acyclic bi-partitioner based on spectral methods.

1.3. Directed acyclic graph layout.

A multitude of different techniques and heuristics have been developed to address graph-layout or linear-arrangement problems. Methods range from (approximate) minimal degree orderings [ADD96, ADD04], (reverse) Cuthill–Mckee [CM69, LS76], highest recent relations heuristics such as Gorder [WYLL16], and recursively partitioning-based [SW04]. For a broader overview of the literature, we refer to [Chu88, DPS02, NS12, Pet13].

In §4 of this paper, we present a heuristic based on our spectral acyclic bi-partition algorithm for the graph-layout problem for directed acyclic graphs with the additional requirement that the linear arrangement must be a topological order. Our heuristic works similar to the one given by Schamberger–Wierum [SW04] for undirected graphs in the sense that it recursively (acyclicly bi-)partitions the graph into smaller parts which are then laid out in a (topological) order. We note that similar schemes have been employed to address numerous other problems, cf. [YB09, GBDD10, YB11, AAA18, BAvL+19]. Here, we make a subtle but notable change to these schemes. Namely, when we acyclicly bi-partition a subgraph, we take the whole graph into account and not just the subgraph. This allows us to not only minimise the edge lengths of the edges within the subgraph, but also the ones that have already been cut by previous rounds of acyclic bi-partitioning. This leads to an overall better graph layout.

[2]All parts of the partition are at most a factor $(1 + \varepsilon)$ larger than their mean, cf. Definition A.2.

1.4. Overview. In §2, we discuss the spectral bi-partitioning method with cut-edge direction incentive; in §3, we extend the method to produce acyclic bi-partitions; and in §4, we discuss how this can be used to generate a topological order with good locality properties. The evaluation of every method may be found in its respective section. In Appendix A, we give the proof of the inapproximability result, Theorem 1.1.

1.5. Acknowledgements. We thank Davide B. Bartolini and Mehdi Alipour for stimulating conversations on this and neighbouring topics. We are furthermore thankful to Davide B. Bartolini for comments on an earlier draft.

Dimosthenis Pasadakis was financially supported by the joint DFG-470857344 and SNSF-204817 project, and by the Huawei Zurich Research Center.

2. Nearly acyclic spectral bi-partitioning

2.1. Algorithm. Classical spectral bi-partitioning is a continuous relaxation of the discrete min-cut problem on an undirected graph $G = (V, E)$. The problem may be stated as

$$\operatorname*{argmin}_{\substack{\boldsymbol{x}\in\mathbb{R}^V \\ \|\boldsymbol{x}\|_2=1 \\ \boldsymbol{x}\perp\boldsymbol{1}}} \sum_{(u,v)\in E} (x_u - x_v)^2 \tag{2.1}$$

and its solution is given by the Fiedler eigenvector [Fie73, Fie89]. We refer to [Chu97] and references therein for an extensive overview of the literature.

In this paper, we consider a *directed* graph G and would like the cut edges to mostly align in the same direction. We achieve this by introducing a penalty to (2.1) when the cut edges are not aligned.

2.1.1. *Symmetry-breaking quadratic form.* Due to the symmetry of the quadratic form, we are unable to enforce that a cut edge $x_u - x_v \not\approx 0$, $(u, v) \in E$, is positive. However, by considering pairs of edges, we are able to construct a quadratic form which is large if and only if all cut edges point in the same direction; that is, they have the same sign:

$$\sum_{\substack{(u_1,v_1)\in E \\ (u_2,v_2)\in E}} (x_{u_1} - x_{v_1})(x_{u_2} - x_{v_2}). \tag{2.2}$$

We note that by Cauchy–Schwarz, we have

$$|E| \cdot \sum_{(u,v)\in E} (x_u - x_v)^2 \geq \left(\sum_{(u,v)\in E} (x_u - x_v) \right)^2 = \sum_{\substack{(u_1,v_1)\in E \\ (u_2,v_2)\in E}} (x_{u_1} - x_{v_1})(x_{u_2} - x_{v_2}). \tag{2.3}$$

Hence,

$$\sum_{(u,v)\in E} (x_u - x_v)^2 - c \left(\sum_{(u,v)\in E} (x_u - x_v) \right)^2 \tag{2.4}$$

is a symmetric positive semi-definite quadratic form for $0 \leq c \leq \frac{1}{|E|}$, which we can use for spectral bi-partitioning. More precisely, we can squeeze the quadratic form (2.4) between multiples of the quadratic form (2.1) used in classic spectral partitioning.

$$\sum_{(u,v)\in E} (x_u - x_v)^2 \geq \sum_{(u,v)\in E} (x_u - x_v)^2 - c \left(\sum_{(u,v)\in E} (x_u - x_v) \right)^2 \geq (1 - c|E|) \sum_{(u,v)\in E} (x_u - x_v)^2. \tag{2.5}$$

For $0 \leq c < \frac{1}{|E|}$, it follows that (2.4) is zero if and only if the Laplacian (2.1) is zero. Hence, like the Laplacian, the kernel of quadratic form (2.4) identifies the weakly connected components of the graph G. It furthermore follows that theoretical results on classic spectral partitioning on cut quality continue to hold with an additional multiplicative loss of at most $(1-c|E|)^{-1}$, e.g., (undirected) Cheeger's inequality [AM85, Alo86, SJ89].

The final optimisation problem reads:

$$\operatorname*{argmin}_{\substack{\boldsymbol{x}\in\mathbb{R}^V\\ \|\boldsymbol{x}\|_2=1\\ \boldsymbol{x}\perp\mathbf{1}}} \sum_{(u,v)\in E}(x_u-x_v)^2 - c\left(\sum_{(u,v)\in E}(x_u-x_v)\right)^2, \tag{2.6}$$

for some $0\le c\le \frac{1}{|E|}$.

Remark 2.1. The above ideas and methods are easily generalised to (non-negatively) edge-weighted graphs. We leave the details to the interested reader.

Remark 2.2. We have

$$\sum_{(u,v)\in E}(x_u-x_v) = \sum_{v\in V}\big(d^+(v)-d^-(v)\big)x_v, \tag{2.7}$$

where $d^+(v)$ denotes the out-degree and $d^-(v)$ the in-degree of a vertex $v\in V$. Hence, the alignment incentive can also be seen as the following heuristic: put vertices with larger difference of out-degree and in-degree in one part and the ones with smaller difference in the other.

Remark 2.3 (p-Laplacian). It has been demonstrated that using the p-Laplacian for p close to but strictly larger than 1 leads to better cut quality [BH09, SPK$^+$18]. One may generalise our penalty term (2.2) also to this setting. The optimisation problem reads now as follows:

$$\operatorname*{argmin}_{\substack{\boldsymbol{x}\in\mathbb{R}^V\\ \|\boldsymbol{x}\|_p=1\\ \boldsymbol{x}\perp\mathbf{1}}} \sum_{(u,v)\in E}|x_u-x_v|^p - c\left|\sum_{(u,v)\in E}(x_u-x_v)\right|^p, \tag{2.8}$$

for $0\le c\le \frac{1}{|E|^{p-1}}$. The range of c is again chosen in such a way that the expression in (2.8) is always non-negative, which follows from Hölder's inequality. Note that for $p=2$, this recovers the expression (2.6).

Remark 2.4 (Directed Laplacian). In a similar vein, one may also introduce an edge-alignment-incentive term to the directed Laplacian [Chu05]. Assume G is strongly connected and let P be the transition probability matrix of a random walk on G. Further, let $\boldsymbol{\varphi}$ be the Perron–Frobenius eigenvector of P normalised such that $\sum_{v\in V}\varphi_v=1$. The new optimisation problem reads:

$$\operatorname*{argmin}_{\substack{\boldsymbol{x}\in\mathbb{R}^V\\ \|\boldsymbol{x}\|_2=1\\ \boldsymbol{x}\perp\sqrt{\boldsymbol{\varphi}}}} \sum_{(u,v)\in E}\varphi_u P_{u,v}\left(\frac{x_u}{\sqrt{\varphi_u}}-\frac{x_v}{\sqrt{\varphi_v}}\right)^2 - c\left(\sum_{(u,v)\in E}\sqrt{\varphi_u P_{u,v}}\left(\frac{x_u}{\sqrt{\varphi_u}}-\frac{x_v}{\sqrt{\varphi_v}}\right)\right)^2, \tag{2.9}$$

for $0\le c\le\frac{1}{|E|}$, where non-negativity follows again from Cauchy–Schwarz. We further point out that a variant of the correction term, where the weighting $\sqrt{\varphi_u P_{u,v}}$ of an edge is replaced by $\varphi_u P_{u,v}$, is already incorporated in the directed Laplacian itself. This is because the correction term in that case is always zero. Indeed, we have

$$\sum_{(u,v)\in E}\varphi_u P_{u,v}\frac{x_u}{\sqrt{\varphi_u}} = \sum_{u\in V}x_u\sqrt{\varphi_u}\sum_{\substack{v\in V\\(u,v)\in E}}P_{u,v} = \sum_{u\in V}x_u\sqrt{\varphi_u}=0 \tag{2.10}$$

and

$$\sum_{(u,v)\in E}\varphi_u P_{u,v}\frac{x_v}{\sqrt{\varphi_v}} = \sum_{v\in V}\frac{x_v}{\sqrt{\varphi_v}}\sum_{\substack{u\in V\\(u,v)\in E}}\varphi_u P_{u,v} = \sum_{v\in V}x_v\sqrt{\varphi_v}=0. \tag{2.11}$$

2.1.2. *Main algorithm.* In order to get from the solution of the optimisation problem (2.6) to a bi-partition[3] $V = S \sqcup T$, one just takes the pointwise sign of the optimal vector. Details can be found in Algorithm 2.1.

Algorithm 2.1: Direction-incentivised spectral bi-partition.

Data: A finite directed graph $G = (V, E)$.
Result: A small cut bi-partition $S \sqcup T = V$ such that most edges between S and T are from S to T.

```
1 $\boldsymbol{x} \leftarrow$ solution to (2.6)
2 $S \leftarrow \{v \in V \mid x_v > 0\}$
3 $T \leftarrow \{v \in V \mid x_v \leq 0\}$
4 if $|(S \times T) \cap E| < |(T \times S) \cap E|$ then $(S, T) \leftarrow (T, S)$
5 return $(S, T)$
```

2.2. Evaluation. In this section, we shall demonstrate that Algorithm 2.1 identifies bi-partitions with more aligned cut edges than direction-oblivious algorithms. We further compare the cut quality against its classic counterpart, Equation (2.1), and state-of-the-art (undirected) partitioners from the literature and show that our algorithm remains competitive. In other words, the direction incentive from Algorithm 2.1 only has a minor effect on the cut quality.

2.2.1. *Algorithms.* In our evaluation, we shall be comparing our direction-incentivised spectral bi-partition, Algorithm 2.1 with $c = \frac{1}{2|E|}$, with the following bi-partitioners from the literature:

- Fiedler eigenvector bi-partitioning [Fie73, Fie89],
- METIS [KK98], and
- KaHIP [SS13].

For each algorithm which supports weight imbalance constraints, we set the maximum weight imbalance to 0.8, cf. Equation (2.14), in order to allow for a more fair comparison.

2.2.2. *Metrics.* Given a finite directed graph $G = (V, E)$ and bi-partition $U \sqcup W = V$ of its vertices, we consider the classic metrics: *conductance*, *cut edges*, and *weight imbalance*. Furthermore, in order to measure the impact of direction incentive of our method, we measure the *misaligned cut edges*.

Conductance. The conductance of the bi-partition is defined as

$$\mathrm{CON} = \frac{|E_{U\times W}| + |E_{W\times U}|}{|E_{U\times W}| + |E_{W\times U}| + \min\{|E_{U\times U}|, |E_{W\times W}|\}}, \tag{2.12}$$

where $E_{A\times B} = E \cap (A \times B)$ are the edges going from $A \subseteq V$ to $B \subseteq V$ in G.

Cut edges. We compute the proportion of cut edges relative to the number of edges in order to compare across graphs:

$$\mathrm{RCE} = \frac{|E_{U\times W}| + |E_{W\times U}|}{|E|}. \tag{2.13}$$

Weight imbalance. We calculate the weight imbalance as

$$\mathrm{WI} = \frac{||\mathrm{U}| - |\mathrm{W}||}{|\mathrm{V}|} = \frac{\max\{|U|, |W|\}}{|V|/2} - 1. \tag{2.14}$$

[3]Throughout the paper, we use the notation $S \sqcup T$ to denote a disjoint union.

Misaligned cut edges. We compute the proportion of misaligned cut edges relative to the number of cut edges:

$$\mathrm{RMCE} = \frac{\min\{|E_{U\times W}|, |E_{W\times U}|\}}{|E_{U\times W}| + |E_{W\times U}|}. \tag{2.15}$$

2.2.3. *Data sets.* We consider two sets of directed graphs, synthetic ones and graphs that come from a variety of real-world applications.

The synthetic ones are generated according to one of three undirected graph distributions: Erdős–Rényi [ER59], Watts–Strogatz [WS98], and stochastic block model [HLL83]. On each of these synthetic graphs, we artificially insert a nearly acyclic perfectly balanced bi-partition $\{A, B\}$. We do this by choosing the direction of edges as follows. For each edge which is totally within A, respectively B, we chose a direction independently and uniformly at random. For every edge between A and B, chose the direction B to A with probability α and the direction A to B with probability $1 - \alpha$ independently. Here, $\alpha \in [0.01, 0.1]$ is a parameter which we call the *alignment probability.*

Erdős–Rényi (ER). A graph with set of vertices $A \sqcup B$, where $|A| = |B| = 500$, and edges inserted between any pair of vertices with independent and uniform probability $p = 0.2$.

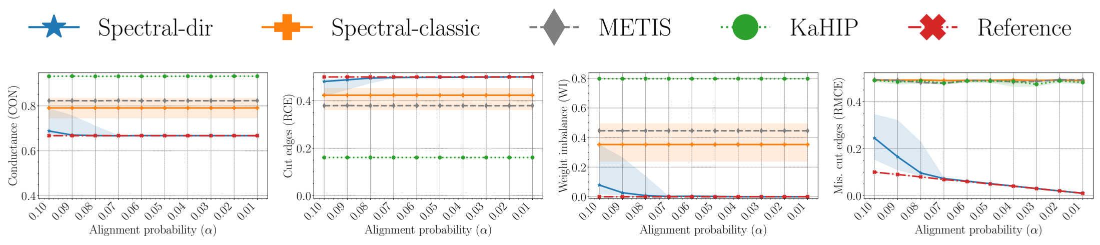
Watts–Strogatz (WS). A graph with $n = 1000$ vertices arranged as a circle. The sets A and B are chosen each as the vertices belonging to a half of the circle. Each vertex is connected to its nearest $k = 50$ neighbours via an edge. For each vertex, each edge to one of its clockwise $k/2$ nearest neighbours is rewired to a uniformly random vertex independently with uniform probability $p = 0.3$.

Stochastic Block Model (SBM). A graph with set of vertices $A \sqcup B$, where $|A| = |B| = 500$, and edges inserted between any pair of vertices in A, respectively B, with independent and uniform probability $p_{\text{int}} = 0.25$. Edges for a pair of vertices $(u, v) \in A \times B$ are inserted independently with uniform probability $p_{\text{ext}} = 0.2$.

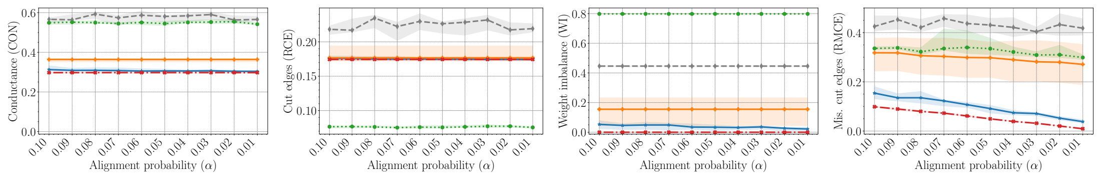
Real-world directed (RW). We subsequently consider a set of 36 real-world directed graphs from the university of Florida sparse matrix collection [DH11] and 4 fine-grained scheduling graphs generated by the HyperDAG_DB [PAKY22, PAKY24]. Together, these graphs emerge from a diverse set of applications, ranging from chemical process simulations to electromagnetics and fluid dynamics, and have number of vertices in the range $[10^2, 1.6\times10^4]$ and number of edges in the range $[10^3, 10^6]$. A complete list of the matrices with their associated statistics is offered in Table B.1.

2.2.4. *Results.* We present in Figure 2.1 the bi-partitioning results on the synthetic ER (A), WS (B), and SBM (C) graphs for decreasing alignment probability α. For each graph instance, we report the median and interquartile range (IQR), i.e., the range between the 25th and 75th percentiles, for each metric. The performance of the four considered algorithms—our proposed Algorithm 2.1 (in blue, Spectral-dir), classic spectral bisection [Fie73, Fie89] (in orange, Spectral-classic), Metis [KK98] (in gray), and KaHIP [SS13] (in green)—is compared against the ground-truth 'Reference' (in red), which corresponds to the bi-partition induced during the synthetic data generation. Perfect alignment with the reference indicates that the algorithm accurately recovered the induced partition.

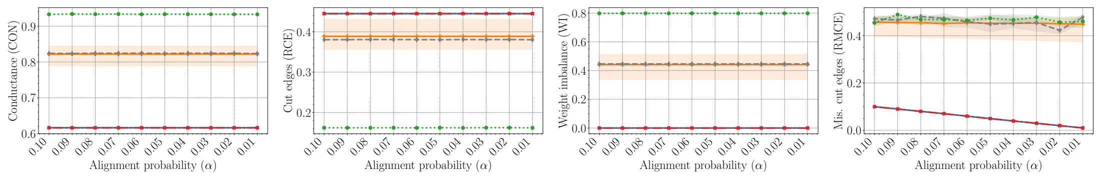
Our method achieves conductance (CON, first column) and cut edge values (RCE, second column) closest to the reference for all three considered graph distributions. The benefits of the direction-incentivised spectral bi-partitioning are most apparent in the weight imbalance (WI, third column) and misaligned cut edges (RMCE, fourth column) metrics. Spectral-dir is the only method that consistently identifies the induced partitions, as indicated by the proximity of WI and RMCE values to those of the reference. Both metrics improve (i.e., move closer to the reference) as α decreases, meaning fewer cut edges are oriented against the dominant direction. For the SBM graphs, Spectral-dir alone achieves exact correspondence



(A) Erdős–Rényi (ER)



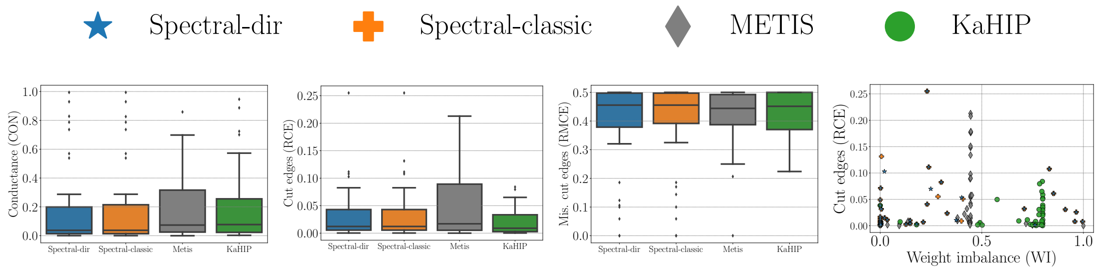
(B) Watts–Strogatz (WS)



(C) Stochastic Block Model (SBM)

FIGURE 2.1. Partitioning results for conductance (CON, first column), cut edges (RCE, second column), weight imbalance (WI, third column), and misaligned cut edges ratio (RMCE, right column), on the synthetic datasets of §2.2.3 with a decreasing aligment probability α. (A): Results for the Erdős–Rényi (ER) graphs. (B): Results for the Watts–Strogatz (WS) graphs. (C): Results for the Stochastic Block Model (SBM) graphs.

with the reference across all α, demonstrating its ability to recover the ground-truth partitions regardless of the cut-edge orientation.



(A) (B) (C) (D)

FIGURE 2.2. Partitioning results for the real-world directed graphs of §2.2.3. (A): Box plot distribution of the CON results, (B): Box plot distribution of the RCE results, (C): Box plot distribution of the RMCE results, (D): Cut edges (RCE) over weight imbalance (WI).

For the real-world graphs, we present in Figure 2.2 box plots of the CON (A), RCE (B), and RMCE (C) distribution results, and a scatter plot (D) to highlight the trade-off between cut edges (RCE) and weight imbalance (WI). With respect to CON, the classic spectral method achieves the best result (strictly or tied) in 62.5% of instances, while our proposed method does so in 55.0%. Our method obtains the best (lowest) score for RMCE in 50.0% of cases, while KaHIP reports the best results for minimizing RCE in 87.5% of the test graphs. Our results highlight that the proposed Algorithm 2.1 provides preferable partitioning solutions for applications where maintaining directional flow and achieving a low misalignment ratio is the primary objective.

3. Acyclic bi-partitioning

In this section, we discuss a simple heuristic to turn a bi-partition into an acyclic bi-partition, whilst attempting to preserve most of the original bi-partition.

3.1. Algorithm. The idea we present here to rectify misaligned edges in a bi-partition of a directed acyclic graph is based on a topological order and a bisection thereof. In order to preserve as much from the original bi-partition as possible, we generate a topological order via a priority, where priority is given to vertices of the first part over those in the second part. Furthermore, as a secondary priority, vertices with a cut outgoing edge in the right direction are deprioritised whereas ones with a cut outgoing edge in the wrong direction are prioritised, and vice versa for cut incoming edges. The details can be found in Algorithm 3.1, where we use the notation $\mathbf{1}_T$ as the indicator function of a set T. We point out that Herrmann *et al.* [HOU+19, §4.2.2] have a similar algorithm to fix acyclicity: they move all ancestors of vertices in S to S or all descendants of vertices in T to T.

Algorithm 3.1: Acyclic fix.

```
Data: A finite directed acyclic graph $G = (V, E)$, a bi-partition $S \sqcup T = V$ such that
      $|(S \times T) \cap E| \geq |(T \times S) \cap E|$, and a balance parameter $\beta \in ]0, 1[$.
Result: A bi-partition $S' \sqcup T' = V$ such that $(T' \times S') \cap E = \emptyset$.
prio[$v$] $\leftarrow 0, \quad \forall v \in V$
for $(u, v) \in (S \times T) \cap E$ do
    prio[$u$] $\leftarrow$ prio[$u$] $+ 1$
    prio[$v$] $\leftarrow$ prio[$v$] $- 1$
for $(u, v) \in (T \times S) \cap E$ do
    prio[$u$] $\leftarrow$ prio[$u$] $- 1$
    prio[$v$] $\leftarrow$ prio[$v$] $+ 1$
prioQ $\leftarrow \emptyset$
for $v \in V$ do
    if number of parents of $v \in G$ is 0 then prioQ.insert(($\mathbf{1}_T(v)$, prio[$v$], $v$))
TopOrdVec $\leftarrow \emptyset$
while not prioQ.empty() do
    ($\_, \_, v$) $\leftarrow$ prioQ.popMin()
    TopOrdVec.pushback($v$)
    for $(v, u) \in E$ do
        if all parents of $u \in G$ are in TopOrdVec then prioQ.insert(($\mathbf{1}_T(u)$, prio[$u$], $u$))
$s_{\min} \leftarrow$ maximal $s \in \mathbb{Z}_{\geq 0}$ such that the first $s$ vertices of TopOrdVec are contained in $S$
$s'_{\min} \leftarrow \max\{s_{\min}, \min\{|S|, \beta|V|\}\}$
$t_{\min} \leftarrow$ maximal $t \in \mathbb{Z}_{\geq 0}$ such that the last $t$ vertices of TopOrdVec are contained in $T$
$t'_{\min} \leftarrow \max\{t_{\min}, \min\{|T|, \beta|V|\}\}$
return minimal-cut bisection $(S', T')$ of TopOrdVec s.t. $|S'| \geq s'_{\min}$ and $|T'| \geq t'_{\min}$
```

Remark 3.1. In the case where the bi-partition came from the direction-incentivised spectral partitioning, see Algorithm 2.1, we are given more nuanced information via the solution to Equation (2.6). This could also be used as the first priority in Algorithm 3.1 instead, by either using the values directly or discretising them first. Note that the solution to (2.6) may need to be multiplied with -1 to have the edges in the right direction.

3.2. Evaluation. In this section, we shall demonstrate that the acyclic fix, Algorithm 3.1, preserves the initial partition better when it comes from our direction-incentivised spectral partitioning method opposed to the classical spectral partitioning method.

In a second step, we show that spectral partitioning methods in conjunction with the acyclic fix, Algorithm 3.1, produce acyclic bi-partitions that are on par if not better with ones produced by state-of-the-art algorithms from the literature and that this can be further improved by local-search algorithms. To this end, we compare our algorithm against ones from the literature on several graph classes using various metrics.

3.2.1. *Algorithms.* We compare our direction-incentivised spectral bi-partition, Algorithm 2.1 with $c = \frac{1}{2|E|}$, together with the acyclic fix, Algorithm 3.1, with the classic spectral partitioning also combined with our acyclic fix, and furthermore the following algorithms from the literature:

- acyclicity-adapted Fiduccia–Mattheyses [CLB94, PSSS21] with initial acyclic bi-partition given by our direction-incentivised spectral bi-partition, and
- dagP [HKU+17, HOU+19].

For each algorithm which supports weight imbalance constraints, we set the maximum weight imbalance to 0.8, cf. Equation (2.14), in order to allow for a more fair comparison.

3.2.2. *Metrics.* We are given a finite directed acyclic graph $G = (V, E)$ and an acyclic bi-partition $U \sqcup W = V$ of its set of vertices. Here, acyclic means that there is no edge going from a vertex in W to a vertex in U, that is $(W \times U) \cap E = \emptyset$.

As in §2.2.2, we measure conductance, cut edges, and weight imbalance. In order to further evaluate the acyclicity fix, see Algorithm 3.1, we measure the *preserved labels* from the original bi-partition.

Preserved labels. Given an original bi-partition $S \sqcup T = V$ of the set of vertices, such that $|(S \times T) \cap E| \geq |(T \times S) \cap E|$, we measure normalised number of preserved labels as:

$$\mathrm{NPL} = \frac{|S \cap U| + |T \cap W|}{|V|}. \tag{3.1}$$

3.2.3. *Data set.* We consider the same set of real-world directed input graphs as in §2.2.3, and convert them into acyclic instances. For an input unsymmetric adjacency matrix $A \in \mathbb{R}^{V \times V}$, we extract its strictly upper triangular part, A_U, and strictly lower triangular part, A_L. Let G_U and G_L be the directed graphs corresponding to A_U and A_L, respectively. To form the acyclic matrix A', we analyse the strictly upper (A_U) and lower (A_L) triangular components of the input matrix A. If only one of the corresponding graphs, G_U or G_L, is weakly connected, we select its matrix. When both are weakly connected, the selection is determined by edge density, with ties broken in favour of A_U. In the case where neither is weakly connected, we instead select the largest weakly connected component of whichever part, A_U or A_L, is denser.

3.2.4. *Results.* In Figure 3.1, we present in three scatter plots the effect of the acyclic fix, Algorithm 3.1, on the direction-incentivised spectral bi-partitioning (in blue, Spectral-dir) and on the classic variant (in orange, Spectral-classic). The classic spectral algorithm produces an inherently acyclic partition in 27.5% of cases, compared to 20.0% for the direction-incentivised method. However, for the partitions that do require correction, our direction-incentivised approach leverages the fix more effectively to improve the cut edges (RCE), achieving this in 43.8% of instances. In contrast, applying the fix to the classic method is more likely to worsen the RCE (65.5% of cases). Enforcing the acyclicity constraint results in a slight degradation in partition balance (WI) in 44.8% of classic and 43.8% of directed corrections, and in conductance (CON) in 72.4% and 75.0% of cases, respectively. For trivial reasons, we see that a higher percentage of preserved labels (NPL) generally corresponds to smaller change in absolute difference to conductance (CON), cut edges (RCE), and weight imbalance (WI). The acyclic fix was able to preserve an average of 95.8% of the labels over all instances when the initial bi-partition was given by Spectral-dir, compared to 93.1% in the case of Spectral–classic.

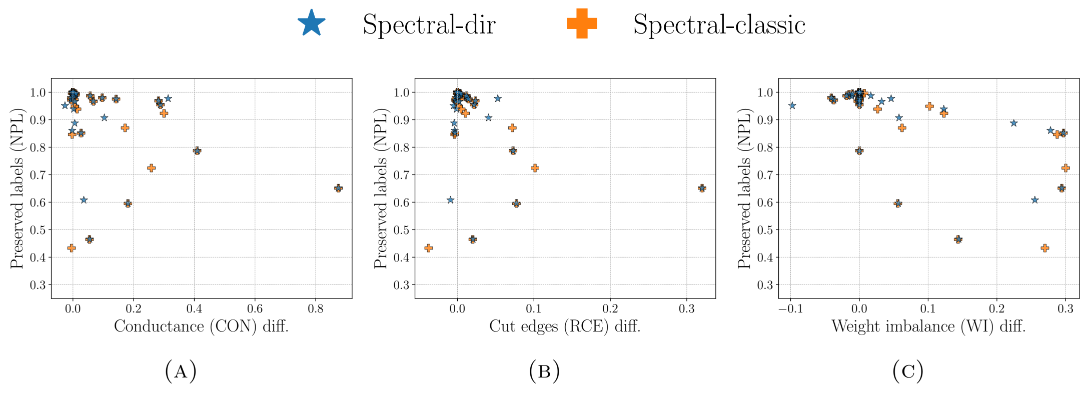


Figure 3.1. Impact of the acyclic-fix post-processing step, introduced in Algorithm 3.1, on the spectral base bi-partitioning algorithms. (A): Change in conductance (CON) over the fraction of correctly assigned vertices (NPL) (B): Change in cut edges (RCE) over the fraction of correctly assigned vertices (NPL), (C): Change in weight imbalance (WI) over the fraction of correctly assigned vertices.

In Figure 3.2, we present the bi-partitioning results on the real-world directed acyclic graphs. The performance of the four considered algorithms is shown: our proposed Algorithm 2.1 (in blue, Spectral-dir-acyc), classic spectral bisection (in orange, Spectral-classic-acyc), the acyclicity-adapted Fiduccia-Mattheyses variant using Algorithm 2.1 as initialiser (in gray, FM-spec-dir-acyc), and dagP (in green). The first three methods incorporate the acyclic fix introduced in Algorithm 3.1. In Subfigures (A), (B) we show box plots of the CON and RCE distribution results respectively, and in (A) the trade off between RCE and WI as a scatter plot. For CON, the two spectral methods exhibit the lowest median values, with the FM-based variant and classic spectral most frequently achieving the best score (47.5% each). Regarding RCE, the FM-based method achieves the lowest median, with dagP obtaining the best score in the highest number of instances (60.0%). The scatter plot in Subfigure (C) illustrates that the performance of dagP on RCE is associated with high weight imbalance (WI), a metric for which the classic spectral method attains the best result in 57.5% of cases, followed by the direction-incentivized proposed variant in 50.0%. These results highlight the effectiveness of the FM-based refinement of spectral methods to retrieve bi-partitions with low conductance and number of cut edges.

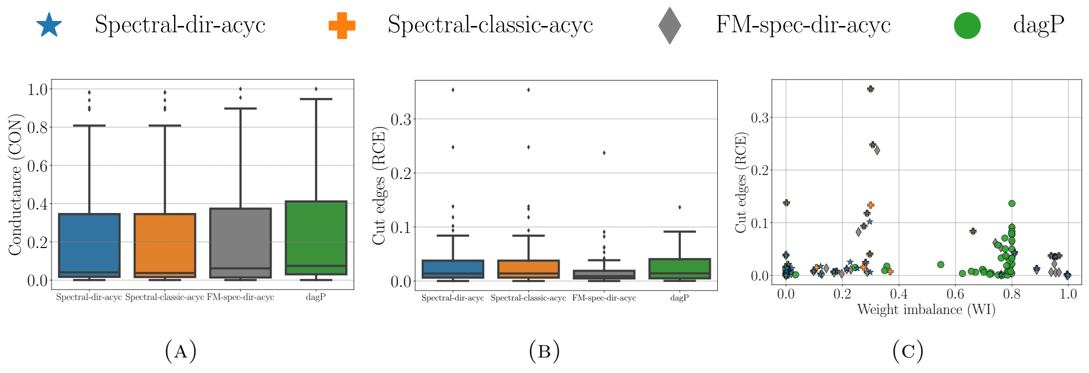


Figure 3.2. Results on obtaining acyclic bi-partitions from the real-world directed acyclic graphs of §3.2.3. (A): Box plot distribution of the CON results for each algorithm under consideration, (B): Box plot distribution of the RCE results for each algorithm under consideration, (C): Cut edges (RCE) over weight imbalance (WI).

4. Spectral topological order

4.1. Algorithm. In this section, we describe how the cut-edge direction-incentivised spectral bi-partitioning from §2.1 can be used to generate a topological order of a directed acyclic graph.

At its core, the bi-partition given by Algorithm 2.1 is used to separate the vertices into ones that come first in a topological order and ones that come later. On each of those parts, the algorithm is applied recursively. This results in an order of the vertices which is not necessarily a topological order. In order to remedy this, we apply a direction fix after each bi-partition $S \sqcup T = V$ such that cut edges only originate from the first part S, that is $(T \times S) \cap E = \emptyset$. Our scheme resembles the one by Schamberger–Wierum [SW04] for undirected graphs, but with the added difficulty of generating a linear layout which is also a topological order.

In §4.1.1, we describe the direction-fix algorithm and in §4.1.2, we describe the complete topological-order algorithm.

4.1.1. *Direction fix.* In order to state the algorithm, we require some notation. Given a directed acyclic graph $G = (V, E)$ and an (acyclic) partition $\bigsqcup_{i=0}^{\ell} L_i = V$ of its set of vertices, we write

$$L_1 \prec L_2 \prec \cdots \prec L_\ell \tag{4.1}$$

if and only if there are no edges originating from L_j and ending in L_i for all $j > i$. In other words, this is a topological order on the coarsened graph along the partition.

We present the direction-fix algorithm in Algorithm 4.1, where we use the notation $\mathbf{1}_T$ as the indicator function of a set T.

Algorithm 4.1: Direction fix.

```
Data: A finite directed acyclic graph $G = (V, E)$, a partition $K \sqcup L \sqcup M = V$ such
      that $K \prec L \prec M$, and a bi-partition $S \sqcup T = L$.
Result: A bi-partition $S' \sqcup T' = L$ such that $(T' \times S') \cap E = \emptyset$.
1  prio$[v] \leftarrow 0, \quad \forall v \in L$
2  for $v \in L$ do
3      prio$[v] \leftarrow$ prio$[v] + |(\{v\} \times M) \cap E|$
4      prio$[v] \leftarrow$ prio$[v] - |(K \times \{v\}) \cap E|$
5  for $(u, v) \in (S \times T) \cap E$ do
6      prio$[u] \leftarrow$ prio$[u] + 1$
7      prio$[v] \leftarrow$ prio$[v] - 1$
8  for $(u, v) \in (T \times S) \cap E$ do
9      prio$[u] \leftarrow$ prio$[u] - 1$
10     prio$[v] \leftarrow$ prio$[v] + 1$
11 prioQ $\leftarrow \emptyset$
12 for $v \in L$ do
13     if number of parents of $v \in G|_L$ is 0 then prioQ.insert(($\mathbf{1}_T(v)$, prio$[v]$, $v$))
14 TopOrdVec $\leftarrow \emptyset$
15 while not prioQ.empty() do
16     $(\_, \_, v) \leftarrow$ prioQ.popMin()
17     TopOrdVec.pushback($v$)
18     for $(v, u) \in (L \times L) \cap E$ do
19         if all parents of $u \in G|_L$ are in TopOrdVec then
               prioQ.insert(($\mathbf{1}_T(u)$, prio$[u]$, $u$))
20 $S' \leftarrow$ first $|S|$ vertices of TopOrdVec
21 $T' \leftarrow$ last $|T|$ vertices of TopOrdVec
22 return $(S', T')$
```

Algorithm 4.1 shares a lot of similarities with Algorithm 3.1. There are, however, two important differences. First, the secondary priority also takes into account the vertices which come before and after in the vertex precedence list VertPrecList. Second, the bisection of the topological order is done in a manner which preserves the cardinality of the initial bi-partition. The former change is an optimisation to keep cut edges short and the latter change is to preserve more of the spectral information.

4.1.2. *Main algorithm.* The main algorithm operates on a vertex precedence list VertPrecList:

$$L_1 \prec L_2 \prec \cdots \prec L_\ell, \tag{4.2}$$

where $\bigsqcup_{i=0}^{\ell} L_i = V$ is a partition of the set of vertices V. This list is initialised simply as V. In each step, a set of vertices L_i with $|L_i| \geq 2$ is chosen and refined as $S \prec T$, that is, $L_i = S \sqcup T$ with $S, T \neq \emptyset$ and $E \cap (T \times S) = \emptyset$. The vertex precedence list is subsequently updated as

$$\cdots \prec L_{i-1} \prec S \prec T \prec L_{i+1} \prec \cdots \, . \tag{4.3}$$

One easily verifies that this is indeed a valid vertex precedence list in the sense of Equation (4.1). The algorithm terminates as soon as all of the sets L_i consist of a single vertex. The details of the algorithm may be found in Algorithm 4.2.

Algorithm 4.2: Spectral topological order.

```
Data: A finite directed acyclic graph $G = (V, E)$.
Result: A topological order of $G$.
1  VertPrecList ← $\{V\}$
2  while $\exists L \in$ VertPrecList with $|L| > 1$ do
3      Let $L \in$ VertPrecList with $|L| > 1$
4      Let $K$ be the union of all sets $N \prec L$ in VertPrecList
5      Let $M$ be the union of all sets $N \succ L$ in VertPrecList
6      $\boldsymbol{x} \leftarrow$ solution to (2.6) with the restriction $\boldsymbol{x}|_K \equiv 1/|V|^{1/2}$ and $\boldsymbol{x}|_M \equiv -1/|V|^{1/2}$
7      $S, T \leftarrow \emptyset$
8      for $v \in L$ do
9          if $x_v > 0$ then $S$.insert($v$) else $T$.insert($v$)
10     if $L = V$ and $|(S \times T) \cap E| < |(T \times S) \cap E|$ then $(S, T) \leftarrow (T, S)$
11     $(S, T) \leftarrow$ DirectionFix$(G, K, L, M, S, T)$          // see Algorithm 4.1
12     Replace $L$ with $S \prec T$ in VertPrecList
13 return [$u$ for $u \in U$ for $U \in$ VertPrecList]
```

We note that the algorithm indeed produces a topological order as it maintains, via the vertex precedence list, a topological order of the coarsened graph (along the partition given by the vertex precedence list) and said coarsened graph is canonically isomorphic to the initial graph when the algorithm terminates. The latter is due to the fact that each set of vertices in the vertex precedence list consists of exactly a single vertex when the algorithm terminates.

The notable difference of Algorithm 4.2 to the recursive splitting algorithm of Schamberger–Wierum [SW04] lies in Line 6, where the bi-partition takes into account the previous bi-partitions and their effect on locality.

Remark 4.1. The spectral topological order algorithm, Algorithm 4.2, can be seen as a refinement of a recursive acyclic bi-partitioning algorithm. As such, it (or modifications thereof) can be used to generate more balanced acyclic bi-partitions and even k-way acyclic partitions with individually prescribed sizes for each part of the partition.

4.2. Evaluation. In this section, we will demonstrate that Algorithm 4.2 produces topological orders with excellent locality both from a data-centric and a temporal point of view. To this end, we compare our algorithm against ones from the literature on several graph classes using various metrics.

4.2.1. *Algorithms.* We compare our spectral topological order algorithm, Algorithm 4.2 with $c = \frac{1}{2|E|}$, and the classic variant ($c = 0$), with the following algorithms from the literature:

- depth-first search,
- breadth-first search with minimum-out-degree ordering,
- adapted Cuthill–Mckee [CM69],
- adapted Gorder [WYLL16] with (recommended) window size of 5, and
- recursive applications of dagP[4] [HKU+17, HOU+19] similar[5] to Algorithm 4.2.

The final algorithm does not exist *per se* in the literature, but may be seen as a naïve adaptation of the method by Schamberger–Wierum [SW04] to directed acyclic graphs using the state-of-the-art acyclic partitioner dagP.

4.2.2. *Metrics.* We are given a finite directed acyclic graph $G = (V, E)$ and a topological order $\preceq$ on G, which we shall represent as an enumeration σ of the vertices in this section, that is, a bijection $\sigma : V \to \{0, 1, \dots, |V| - 1\}$.

There are several metrics to evaluate the 'locality' of the topological order both from a data-centric view and from a temporal point of view. Most notably, there is the *bandwidth* [DPS02], *reuse distance* also known as *LRU-stack distance* [MGST70], and *cut width* [DPS02]. In the literature, this type of problem is known as a graph-layout problem. We refer to the surveys [Chu88, DPS02, Pet13].

Edge-length distribution. The length $\omega(e)$ of an edge $e = (u, v) \in E$ is defined as $\sigma(v) - \sigma(u)$. We note that this quantity is (strictly) positive. The edge-length distribution is the collection $\omega(e), e \in E$.

This distribution gives rise to data-locality metrics. For example, we note that the maximum over this distribution is known as the *bandwidth* and minimising the sum of the distribution as the *minimum-linear-arrangement* problem.

Reuse-distance distribution. In order to define reuse distance, one should think of each vertex producing an output which then gets accessed by its out-neighbours. This produces an access pattern, where we can measure the number of distinct memory accesses between any two accesses to the same piece of memory (output of a vertex). A precise definition of the access pattern and the reuse-distance distribution may be found in Algorithms 4.3 and 4.4. This distribution gives rise to types of temporal locality.

Algorithm 4.3: Access pattern.

```
Data: A directed graph $G = (V, E)$ and a topological order $\preceq$.
Result: An access pattern $A$.
1 $A \leftarrow \emptyset$
2 for $v \in V$ in topological order $\preceq$ do
3     for $u$ in-neighbour of $v$ in topological order $\preceq$ do
4         $A$.push_back($u$)
5     $A$.push_back($v$)
6 return $A$
```

[4]Minor modifications were made to recursively bisect using dagP, as it does not allow splitting a graph with less than three nodes and would cause a segmentation fault for subgraphs with no edges. Additionally, with loose balance constraints, dagP may assign all nodes to one part on small graphs to achieve a zero edge-cut; we therefore enforce a stricter weight imbalance of 0.3.

[5]More precisely, replace lines 4 through 11 with $(S, T) \leftarrow \text{dagP}(G|_M)$.

Algorithm 4.4: Reuse-distance distribution.

Data: An access pattern A.

Result: Corresponding reuse-distance distribution D.

```
1 $D \leftarrow \emptyset$
2 for $i = 0, 1, \ldots, A.\text{size}() - 1$ do
3     if $\exists j < i$ such that $A[j] = A[i]$ then
4         $k \leftarrow \max\{j < i \mid A[j] = A[i]\}$
5         $d \leftarrow |\{v \in V \mid \exists j$ such that $k < j < i$ and $A[j] = v\}|$
6         $D.\text{push\_back}(d)$
7 return $D$
```

Edge-cut distribution. The bisection-cut is the number of cut edges of a bisection of the enumeration σ of the topological order $\preceq$. That is

$$\beta(i) = |\{(u, v) \in E \mid \sigma(u) \leq i < \sigma(v)\}|. \tag{4.4}$$

The edge-cut distribution is the collection of the bisection-cuts $\beta(i)$ for $i = 0, 1, \ldots, |V| - 2$.

This distribution gives rise to data- and temporal-locality metrics. For example, we note that the maximum over this distribution is known as the *cut width* and minimising the sum of the distribution as the *minimum-linear-arrangement* problem.

Remark 4.2. The sum of all bisection-cuts is equal to the sum of all edge lengths, cf. [DPS02, Obs. 2, p. 3] and [Har66].

4.2.3. *Data set.* We use the same set of real-world directed graphs as in §2.2.3, and convert them once more into acyclic instances. Given an input adjacency matrix A, we consider its strictly upper (A_U) and lower (A_L) triangular components. The final acyclic matrix, denoted A', is chosen by identifying the component with the higher edge density, with ties being resolved in favour of A_U.

4.2.4. *Results.* In Figure 4.1, we present performance profiles [DM02] for the various metrics mentioned in §4.2.2.

We observe that the direction-incentivised spectral topological-order algorithm (Spectral-dir-top) outperforms or is on par with its classic variant (Spectral-classic-top) in all metrics. We attribute this to the fact that the direction-incentivised spectral acyclic bi-partition 3.1 is able to preserve more of the original bi-partition, cf. Figure 3.1. Spectral-dir-top further stands out as the single best algorithm for bandwidth, maximum reuse distance, and median edge cut. It is furthermore amongst the best algorithms for minimum linear arrangement and total reuse distance, and near the top in cut width. The recursive application of dagP (dagP-rec) outshines all algorithms on the median edge length, but this comes at the cost of neglecting the tail end of the edge length distribution, e.g., the bandwidth. Similarly, dagP does better on the median reuse distance than any other algorithm, but performs poorly on the maximum reuse distance. Furthermore, dagP is the best on cut width and amongst the best for total reuse distance. Gorder performs well on the median reuse distance. In general, Gorder performs better on temporal-locality metrics for which it was originally designed, but lacks in all other metrics. Cuthill–Mckee and breadth-first search (BFS) perform poorly except for bandwidth and maximum reuse distance where they land in the middle of the pack. Likewise, depth-first search (DFS) performs poorly except for median edge length and median reuse distance where it is above average.

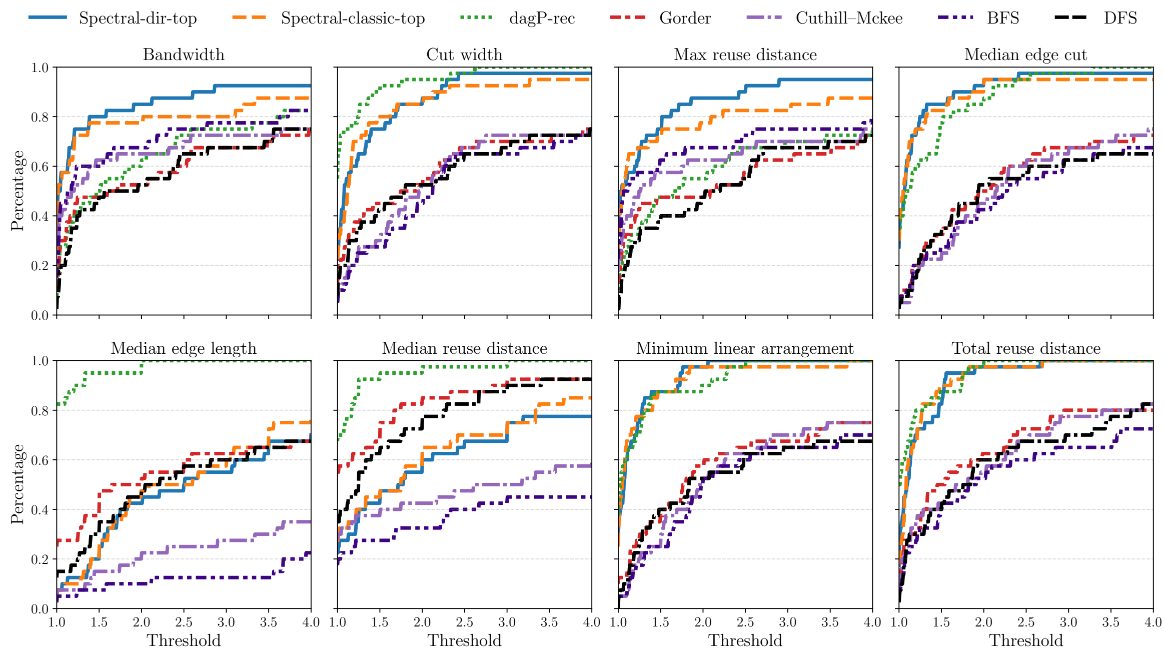


FIGURE 4.1. Performance profiles of the algorithms from §4.2.1 on the data set from §4.2.3 using the various metrics from §4.2.2. In each plot, the x-axis represents a threshold and the y-axis the percentage of graphs which are within this threshold times the result of the best algorithm.

5. CONCLUDING REMARKS

We have shown that our direction-incentivised spectral partitioning method, Algorithm 2.1, is able to find bi-partitions with more edges going in one direction. In order to do so, it may increase the edge cut by a small margin. As is well-known, the spectral methods performed better on conductance rather than edge cut and are thus better suited for classification purposes rather than balanced minimum cuts.

We have further demonstrated that spectral partitioning methods (with appropriate fixes) are viable for acyclic partitioning. In particular, when combined with modern multi-level and local search approaches, we believe it to be able to produce high-quality acyclic partitions. This extends to weight-balance constrained k-way acyclic partitioning, where we have shown that recursively splitting (for a topological order) leads to a good cut width, cf. Remark 4.1.

Most importantly, we have conclusively shown that our direction-incentivised spectral topological order, Algorithm 4.2, produces exquisite topological orders in terms of data and temporal locality and that the direction incentive improves the locality of the classical analogue even further. In particular, we have found our algorithm to be the best or amongst the best algorithm for bandwidth, maximum reuse distance, median edge cut, minimum linear arrangement, and total reuse distance.

We have further corroborated that the recursively partition method of Schamberger–Wierum [SW04] produces good orderings for the bulk of the graph, meaning it performs well on median edge length, edge cut, and reuse distance. We believe the method can be further improved by incorporating ideas mentioned in this paper, specifically by including information of previously cut edges into the subsequent partitioning iterations in order to also address the tail ends of the distributions. We leave this inquiry for future work.

We have thus far avoided talking about time. This is because our implementations are proofs of concepts and little effort has gone into engineering them to be efficient. In practice, spectral methods tend to be a bit slower, but modern implementations are suitably fast [HS04, BM13, NM16, PSVY23]. We further point out that Algorithm 4.2, which recursively spectrally splits parts of the graph, generically[6] only increases the complexity by a logarithmic factor over the spectral bi-partitioning, Algorithm 2.1.

At last, we bring up the question in what capacity the gains we observed transfer to the various applications, and leave it open as a research topic.

Appendix A. Inapproximability: proof of Theorem 1.1

We first begin with some formal definitions that are required to state and prove Theorem 1.1.

Definition A.1. Given a directed acyclic graph $G = (V, E)$, a disjoint partitioning $V_1 \sqcup V_2 = V$ is called an *acyclic bi-partition* if there is no directed edge $(u, v) \in E$ such that $u \in V_2$ and $v \in V_1$. The *cost* of the bi-partition is the number of edges $(u, v) \in E$ such that $u \in V_1$ and $v \in V_2$.

Definition A.2. Given a directed acyclic graph $G = (V, E)$ and a real parameter $0 \le \varepsilon < 1$, the acyclic bi-partition $V_1 \sqcup V_2 = V$ is ε*-balanced* if $\max(|V_1|, |V_2|) \le (1 + \varepsilon) \cdot \frac{|V|}{2}$. The goal of the ε*-balanced acyclic bi-partition problem* is to find an ε-balanced acyclic bi-partition of minimal cost.

For the undirected version of the problem, the special case of $\varepsilon = 0$ is sometimes also known as the graph bisection problem, and has also been extensively studied, see [FK02, Räc08] and references therein/-to.

Given these definitions, we first restate Theorem 1.1 for completeness.

Theorem 1.1. *Assuming the exponential-time hypothesis, there is a $\delta > 0$ such that for every $0 \le \varepsilon < 1$ the ε-balanced acyclic bi-partition problem does not have a polynomial-time $n^{1/\log(\log(n))^\delta}$-factor approximation algorithm, where $n = |V|$ is the size of the input graph.*

The theorem provides a particularly interesting contrast to the undirected version of the same problem, which is known to admit an $O(\log n)$-approximation algorithm [Räc08, LR88]. For the directed case, the only related inapproximability result we are aware of is a simple reduction which only applies when partitioning into a non-constant number of parts [MPS17].

We split the discussion of the proof into two parts: we first consider the case when the partitions must have equal size ($\varepsilon = 0$), and then we show how to extend this to any $0 < \varepsilon < 1$.

Proof for $\varepsilon = 0$. We provide a reduction from the smallest k-edge subgraph problem: given an undirected graph G_0 with N vertices and M edges, and an integer $1 \leqslant k \leqslant M$, we need to find the smallest subset of vertices that spans at least k edges. It is known that there exists a $\delta' > 0$ such that this problem is not approximable in polynomial time to a $n^{1/(\log\log n)^{\delta'}}$ factor, assuming ETH, see [Man17]. Let us select any $\delta > \delta'$.

Given a smallest k-edge subgraph problem with G_0 and k, we develop a directed acyclic graph as follows. We consider two parameters $t = 2 \cdot M$ and $\ell_0 = t \cdot N + M + 2$. The main ingredients of our construction will be *block gadgets*: a set of vertices $v_1, ..., v_\ell$ such that $(v_i, v_j) \in E$ for all $1 \le i < j \le \ell$. Our construction will ensure that we can trivially find a solution of cost at most $(\ell_0 - 2)$, and all our block gadgets have size $\ell \ge \ell_0$. This implies that if the vertices $v_1, ..., v_\ell$ are not all assigned to the same part, then there are at least $(\ell - 1) > (\ell_0 - 2)$ edges cut within the gadget, and hence we already have a higher cost than in the trivial solution. As such, any reasonable solution places each block gadget either entirely into V_1 or entirely into V_2.

Our directed acyclic graph will consist of the following block gadgets: two block gadgets B_1 and B_2, a block gadget B_e for each undirected edge e of G_0, and a single vertex v_u for each vertex u of G_0. We add a single edge from an arbitrary vertex of B_1 to an arbitrary vertex of B_2. For each vertex u incident to edge e in G_0, we add an edge from v_u to an arbitrary vertex of B_e. Finally, for all vertices v_u that represent a vertex in G_0, we add t distinct edges from

[6]We are assuming here that the spectral bi-partition always finds a δ-balanced bi-partition for a fixed $\delta < 1$, and that its time complexity is upper-bounded by a super-additive and monotone function.

v_u to t arbitrary vertices of B_2. As for the size of the block gadgets, the number of vertices will be

- $(M-k)\cdot \ell_0 + N + 1$ in B_1,
- $k \cdot \ell_0 + N + 1$ in B_2,
- ℓ_0 in each B_e.

Finally, we add N further isolated vertices to the directed acyclic graph. Altogether, the size of the construction is $n = 2M \cdot \ell_0 + 4 \cdot N + 2 = O(N \cdot M^2) = O(N^5)$.

We have $|B_1| + |B_2| > \frac{n}{2}$, so the two blocks cannot be in the same part. Due to the edge between them, this implies that B_1 is in V_1 and B_2 is in V_2. Since $\ell_0 > 2 \cdot N$, in order to have exactly $\frac{n}{2}$ vertices in both parts without splitting a block gadget, we need to assign exactly k block gadgets B_e to V_1, and the remaining $(M-k)$ block gadgets B_e to V_2. We can then assign any desired subset of the N vertices v_u to V_1, and the rest to V_2, since we can balance it afterwards with the N isolated vertices.

Selecting the k gadgets B_e to assign to V_1 corresponds to selecting k edges in G_0. If an edge e in G_0 is selected this way, and it is incident to a vertex u in G_0, then this implies that v_u must also be in V_1, since v_u has an edge to a vertex in B_e. As such, the vertices v_u corresponding to all the endpoints of the k selected edges must be in V_1. All these vertices v_u will have t outgoing edges to B_2 in V_2, and some further edges to the B_e incident to u; some of these might also be in V_2. On the other hand, if a vertex u has all of its incident edges in V_2, then no reasonable solution will place v_u in V_1, since this results in a lot of cut edges; indeed, any such solution can be easily improved by exchanging v_u with an isolated vertex.

This means that if a set of k selected edges in u_0 spans c vertices, then the resulting bi-partition will have cost $c \cdot t$ plus the sum of the degree of the spanned vertices, i.e. cost between $c \cdot t$ and $c \cdot t + M$. With $t = 2M$, this cost is always dominated by the term $c \cdot t$. The observation also holds the other way around: if we have a reasonable solution in the directed acyclic graph for the acyclic bi-partition problem (i.e., no blocks are cut, no v_u is placed in V_1 unless necessary), and the solution has cost between $c \cdot t$ and $(c+1) \cdot t$ for some $c \in \mathbb{Z}_{\geq 0}$, then this allows us to select a subset of c vertices in G_0 that induce at least k edges.

More formally, let $\mathrm{OPT}_{\mathrm{SkES}}$ and $\mathrm{OPT}_{\mathrm{ABP}}$ denote the optimum costs in the original smallest k-edge subgraph problem and the derived acyclic bi-partition problem, respectively. As discussed before, we have $\mathrm{OPT}_{\mathrm{ABP}} \leq t \cdot \mathrm{OPT}_{\mathrm{SkES}} + M$. Assume now that we have a polynomial-time algorithm that approximates the acyclic bi-partition problem to a $n^{1/(\log\log n)^\delta}$ factor. We can assume that this algorithm returns a reasonable solution (no blocks are cut, no v_u is placed in V_1 unless necessary), otherwise we begin by replacing it by a reasonable solution as discussed above; this only further decreases its cost. Assume that the algorithm returns a solution of cost $\mathrm{SOL}_{\mathrm{ABP}}$. Consider the derived solution of smallest k-edge subgraph problem, where we select the k edges whose edge gadgets are in V_1; let us denote its cost by $\mathrm{SOL}_{\mathrm{SkES}}$. Recall that by assumption, we have

$$\mathrm{SOL}_{\mathrm{ABP}} \leq n^{1/(\log\log n)^\delta} \cdot \mathrm{OPT}_{\mathrm{ABP}} \,. \tag{A.1}$$

Recall from before that $\mathrm{SOL}_{\mathrm{ABP}} \geq t \cdot \mathrm{SOL}_{\mathrm{SkES}}$, and hence

$$t \cdot \mathrm{SOL}_{\mathrm{SkES}} \leq n^{1/(\log\log n)^\delta} \cdot (t \cdot \mathrm{OPT}_{\mathrm{SkES}} + M) \,. \tag{A.2}$$

With $t > M$, we also get

$$\mathrm{SOL}_{\mathrm{SkES}} \leq n^{1/(\log\log n)^\delta} \cdot (\mathrm{OPT}_{\mathrm{SkES}} + 1) \leq 2 \cdot n^{1/(\log\log n)^\delta} \cdot \mathrm{OPT}_{\mathrm{SkES}} \,, \tag{A.3}$$

where the second inequality follows easily for $\mathrm{OPT}_{\mathrm{SkES}} \geq 1$. Recall that there is a constant c_0 such that $n \leq c_0 \cdot N^5$, so we can upper-bound $n^{1/(\log\log n)^\delta}$ by $(c_0 \cdot N^5)^{1/(\log\log N)^\delta}$. Then, $(2c_0 \cdot N^5)^{1/(\log\log N)^\delta}$ is in turn upper-bounded by $N^{6/(\log\log N)^\delta}$ for sufficiently large N. This implies

$$\mathrm{SOL}_{\mathrm{SkES}} \leq N^{6/(\log\log N)^\delta} \cdot \mathrm{OPT}_{\mathrm{SkES}} \,. \tag{A.4}$$

With $\delta > \delta'$ and N large enough, $N^{6/(\log\log N)^{\delta}}$ is upper-bounded by $N^{1/(\log\log N)^{\delta'}}$, hence

$$\mathrm{SOL}_{\mathrm{SkES}} \leq N^{1/(\log\log N)^{\delta'}} \cdot \mathrm{OPT}_{\mathrm{SkES}} \,. \tag{A.5}$$

As such, our algorithm allows to approximate the smallest k-edge subgraph problem in polynomial time to a factor that is not possible if ETH holds. □

The same reduction can easily be adapted to the case of more flexible balance constraints.

Proof for $0 < \varepsilon < 1$. In order to adapt the proof above to the ε-balanced case, we only need to adjust the size of B_1 and B_2 to ensure that we are still forced to place k edge gadgets into V_1, i.e., we can only fit B_2, at most $(M-k)$ edge gadgets and N further vertices into V_2. For simplicity, let s denote the total size of the edge and vertex gadgets and isolated vertices, i.e., $s = M \cdot \ell_0 + 2N$. Let s_0 denote the size of $(M-k)$ edge gadgets plus N further vertices, i.e., $s_0 = (M-k) \cdot \ell_0 + N$. We need to ensure that

$$|B_2| + s_0 \leq (1+\varepsilon) \cdot \frac{n}{2} \leq |B_2| + s_0 + N \,, \tag{A.6}$$

while $|B_1|+|B_2| > (1+\varepsilon)\cdot\frac{n}{2}$ also remains true. For this, we can first select n large enough such that $s < (1-\varepsilon)\cdot\frac{n}{2}$. We can then set $|B_2| = \lfloor(1+\varepsilon)\cdot\frac{n}{2}\rfloor - s_0$, and finally $|B_1| = n - |B_2| - s$.

Note that these choices also implicitly ensure for V_1 that $|B_1| + k\cdot\ell_0 + N \leq (1+\varepsilon)\cdot\frac{n}{2}$: after substituting our choices of $|B_1|$, $|B_2|$, s_0 and s, we get $n \leq (1+\varepsilon)\cdot\frac{n}{2} + \lfloor(1+\varepsilon)\cdot\frac{n}{2}\rfloor$, which holds for n large enough. □

Remark A.3. If one assumes the stronger Gap-ETH conjecture instead of ETH, then the same reduction shows an inapproximability to an $n^{o(1)}$ factor [Man17].

Appendix B. List of graphs

Graph	**Domain**	**Vertices**	**Edges**	**Sym. Edges**
arc130	materials science	130	1,282	75.9%
b2_ss	chemical process	1,089	4,228	1.5%
barth	structural simulation	6,691	26,439	0.0%
barth4	structural simulation	6,019	23,492	0.0%
barth5	structural simulation	15,606	61,484	0.0%
bayer03	chemical process	6,747	56,196	0.3%
bp_1600	optimisation	822	4,841	1.1%
CG_bcsstk05	scheduling	11,041	23,116	0.0%
CG_nos1	scheduling	10,256	19,384	0.0%
circuit204	circuit simulation	1,020	5,883	43.8%
conf5_0-4x4-10	optimisation	3,072	119,808	92.3%
cz5108	fluid dynamics	5,108	51,412	43.4%
dw2048	electromagnetics	2,048	10,114	98.5%
dw4096	electromagnetics	8,192	41,746	96.3%
epb0	thermodynamics	1,794	7,764	50.1%

Table B.1. All graphs from the SuiteSparse Matrix Collection [DH11] and generated by the HyperDAG_DB [PAKY22, PAKY24] used in the paper with source domain, number of vertices, number of edges, and percentage of symmetric edges listed. (Continued on next page)

Graph	**Domain**	**Vertices**	**Edges**	**Sym. Edges**
foldoc	networks	13,356	120,238	47.9%
fp	electromagnetics	7,548	834,222	76.8%
fpga_dcop_15	circuit simulation	1,220	5,892	81.8%
fs_541_1	3D problem	541	4,285	68.3%
g7jac050sc	economic problem	14,760	145,157	3.2%
gemat12	power network	4,929	33,111	0.1%
gre_512	chip simulation	512	2,192	0.0%
lhr14c	chemical process	14,270	307,858	0.7%
lshp1270	thermodynamics	1,270	8,668	100.0%
mahindas	economic problem	1,258	7,682	1.7%
mark3jac020sc	economic problem	9,129	52,883	6.1%
olm5000	fluid dynamics	5,000	19,996	66.7%
orani678	optimisation	2,529	90,158	7.1%
p2p-Gnutella06	networks	8,717	31,525	0.0%
pores_2	fluid dynamics	1,224	9,613	61.2%
POW3SPMV_bcspwr05	scheduling	5,904	9,297	0.0%
psmigr_3	economic problem	3,140	543,160	47.9%
qh882	networks	882	3,354	92.6%
rajat06	circuit simulation	10,922	28,922	0.0%
rdist3a	chemical process	2,398	61,896	14.0%
rw5151	optimisation	5,151	20,199	49.0%
SPMV_west0497	scheduling	4,448	5,181	0.0%
ted_A	thermodynamics	10,605	424,587	56.6%
utm5940	electromagnetics	5,940	83,842	52.9%
watt_1	fluid dynamics	1,856	11,360	98.7%

Table B.1. All graphs from the SuiteSparse Matrix Collection [DH11] and generated by the HyperDAG_DB [PAKY22, PAKY24] used in the paper with source domain, number of vertices, number of edges, and percentage of symmetric edges listed.

Appendix C. Individual results for topological orders

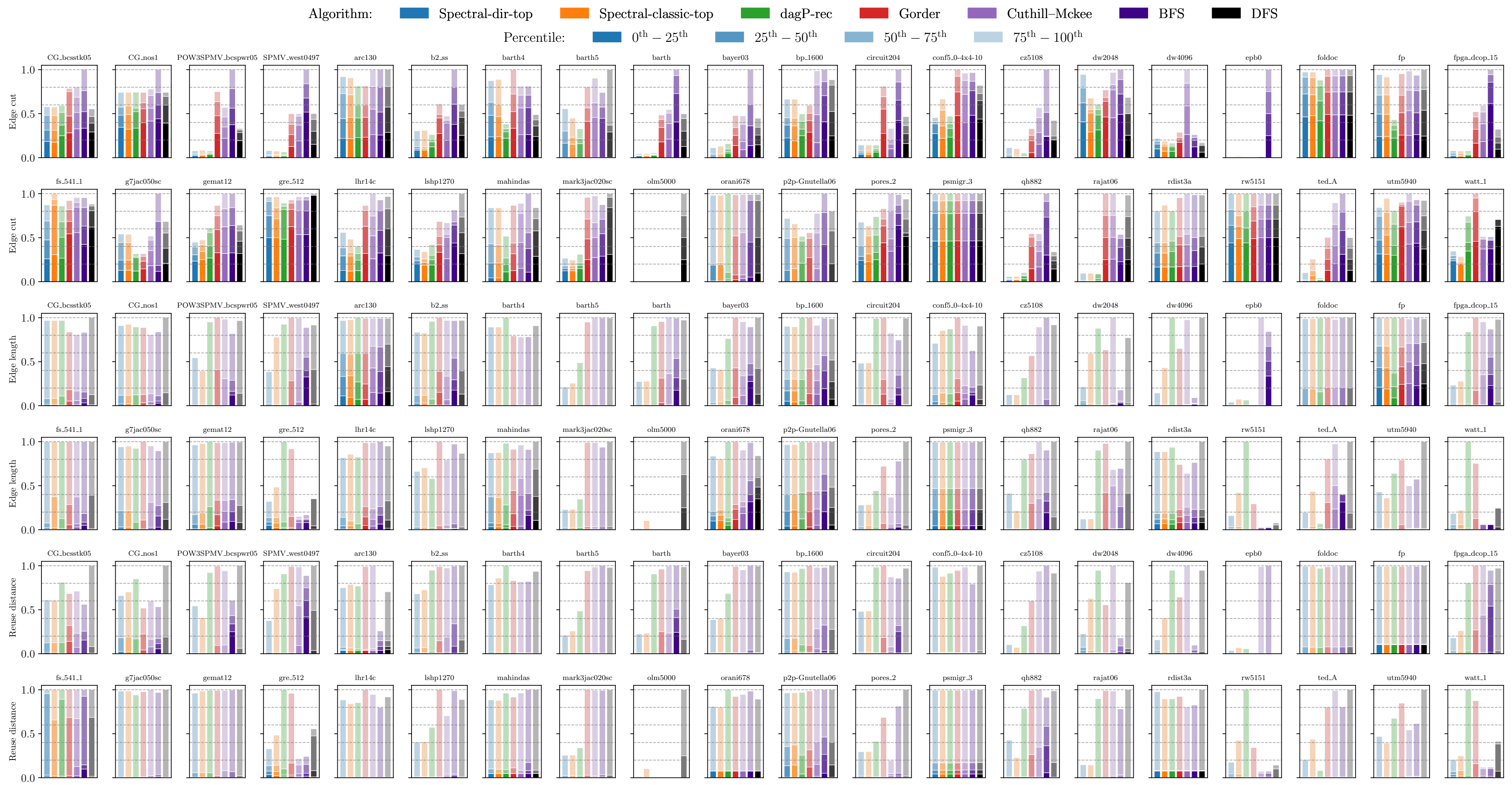


Figure C.1. Edge-length, reuse-distance, and edge-cut distributions of the topological orders generated by the algorithms in §4.2.1 on each individual graph from the data set in §4.2.3. For each graph and metric, the values have been rescaled such that the worst performing algorithm takes the value 1.

References

[AAA18] Nabil Abubaker, Kadir Akbudak, and Cevdet Aykanat. Spatiotemporal graph and hypergraph partitioning models for sparse matrix-vector multiplication on many-core architectures. *IEEE Transactions on Parallel and Distributed Systems*, 30(2):445–458, 2018.

[ADD96] Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[ADD04] Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. Algorithm 837: AMD, an approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):381–388, September 2004.

[AFK$^+$12] Kunal Agrawal, Jeremy T. Fineman, Jordan Krage, Charles E. Leiserson, and Sivan Toledo. Cache-conscious scheduling of streaming applications. In *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*, pages 236–245, 2012.

[Alo86] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[AM85] Noga Alon and Vitali D. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, 1985.

[AOTS15] Daron Acemoglu, Asuman Ozdaglar, and Alireza Tahbaz-Salehi. Systemic risk and stability in financial networks. *American Economic Review*, 105(2):564–608, February 2015.

[BAvL$^+$19] Rob H. Bisseling, Bas Fagginger Auer, Tristan van Leeuwen, Wouter Meesen, Marco van Oort, Daan Pelt, Brendan Vastenhouw, and Albert-Jan N. Yzelman. Mondriaan (version 4.2.1). http://www.staff.science.uu.nl/~bisse101/Mondriaan/mondriaan_v4.2.1.tar.gz, 2019.

[BH09] Thomas Bühler and Matthias Hein. Spectral clustering based on the graph p-Laplacian. In *Proceedings of the 26th annual international conference on machine learning*, pages 81–88, 2009.

[BM13] Alexander Bertrand and Marc Moonen. Distributed computation of the Fiedler vector with application to topology inference in ad hoc networks. *Signal Processing*, 93(5):1106–1117, 2013.

[BMS$^+$16] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. *Recent advances in graph partitioning.* Springer, 2016.

[BS13] Charles-Edmond Bichot and Patrick Siarry. *Graph partitioning.* John Wiley & Sons, 2013.

[ÇDF$^+$23] Ümit Çatalyürek, Karen Devine, Marcelo Faraj, Lars Gottesbüren, Tobias Heuer, Henning Meyerhenke, Peter Sanders, Sebastian Schlag, Christian Schulz, Daniel Seemaier, et al. More recent advances in (hyper) graph partitioning. *ACM Computing Surveys*, 55(12):1–38, 2023.

[Chu88] Fan R. K. Chung. Labelings of graphs. *Selected topics in graph theory*, 3(25):151–168, 1988.

[Chu97] Fan R. K. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[Chu05] Fan R. K. Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, 9:1–19, 2005.

[CLB94] Jason Cong, Zheng Li, and Rajive Bagrodia. Acyclic multi-way partitioning of Boolean networks. In *Proceedings of the 31st annual design automation conference*, pages 670–675, 1994.

[CM69] Elizabeth Cuthill and James McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172, 1969.

[DH11] Timothy A. Davis and Yifan Hu. The university of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1), December 2011.

[DM02] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91:201–213, 2002.

[DPS02] Josep Díaz, Jordi Petit, and Maria Serna. A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, 34(3):313–356, 2002.

[ER59] P. Erdős and A. Rényi. On random graphs. *Publ. Math. Debrecen*, 6:290–297, 1959. Reprinted in *The Art of Counting* (MIT Press, 1973).

[ERP$^+$15] Venmugil Elango, Fabrice Rastello, Louis-Noël Pouchet, Jagannathan Ramanujam, and Ponnuswamy Sadayappan. On characterizing the data access complexity of programs. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 567–580, 2015.

[FER$^+$13] Naznin Fauzia, Venmugil Elango, Mahesh Ravishankar, Jagannathan Ramanujam, Fabrice Rastello, Atanas Rountev, Louis-Noël Pouchet, and Ponnuswamy Sadayappan. Beyond reuse distance analysis: dynamic analysis for characterization of data locality potential. *ACM Transactions on Architecture and Code Optimization (TACO)*, 10(4):1–29, 2013.

[Fie73] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[Fie89] Miroslav Fiedler. Laplacian of graphs and algebraic connectivity. *Banach Center Publications*, 25(1):57–70, 1989.

[Fjä98] Per-Olof Fjällström. *Algorithms for graph partitioning: a survey.* Linköping University Electronic Press, 1998.

[FK02] Uriel Feige and Robert Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal on Computing*, 31(4):1090–1118, 2002.

[GBDD10] Laura Grigori, Erik G. Boman, Simplice Donfack, and Timothy A. Davis. Hypergraph-based unsymmetric nested dissection ordering for sparse LU factorization. *SIAM Journal on Scientific Computing*, 32(6):3426–3446, 2010.

[GJS74] Michael R. Garey, David S. Johnson, and Larry Stockmeyer. Some simplified NP-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, 1974.

[Gle06] David Gleich. Hierarchical directed spectral graph partitioning. *Information Networks*, 443:24, 2006.

[GM17] Krystal Guo and Bojan Mohar. Hermitian adjacency matrix of digraphs and mixed graphs. *Journal of Graph Theory*, 85(1):217–248, 2017.

[HAP22] Koby Hayashi, Sinan G. Aksoy, and Haesun Park. Skew-symmetric adjacency matrices for clustering directed graphs. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 555–564. IEEE, 2022.

[Har66] Lawrence H. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1(3):385–393, 1966.

[HKU$^+$17] Julien Herrmann, Jonathan Kho, Bora Uçar, Kamer Kaya, and Ümit V. Çatalyürek. Acyclic partitioning of large directed acyclic graphs. In *2017 17th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID)*, pages 371–380. IEEE, 2017.

[HLL83] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: first steps. *Social Networks*, 5(2):109–137, 1983.

[HOU$^+$19] Julien Herrmann, M. Yusuf Ozkaya, Bora Uçar, Kamer Kaya, and Ümit V. Çatalyürek. Multilevel algorithms for acyclic partitioning of directed acyclic graphs. *SIAM Journal on Scientific Computing*, 41(4):A2117–A2145, 2019.

[HS04] Yifan Hu and Jennifer Scott. Hsl_mc73: a fast multilevel Fiedler and profile reduction code. Technical report, CM-P00047969, 2004.

[HZS06] Jiayuan Huang, Tingshao Zhu, and Dale Schuurmans. Web communities identification from random walks. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 187–198. Springer, 2006.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[Ker71] Brian W. Kernighan. Optimal sequential partitions of graphs. *Journal of the ACM (JACM)*, 18(1):34–40, 1971.

[Kes63] Maxwell Mirton Kessler. Bibliographic coupling between scientific papers. *American documentation*, 14(1):10–25, 1963.

[KHKM11] Jin Kim, Inwook Hwang, Yong-Hyuk Kim, and Byung-Ro Moon. Genetic approaches for graph partitioning: a survey. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 473–480, 2011.

[KK98] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[LR88] Tom Leighton and Satish Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pages 422–431. IEEE Computer Society, 1988.

[LS76] Wai-Hung Liu and Andrew H. Sherman. Comparative analysis of the Cuthill–McKee and the reverse Cuthill–McKee ordering algorithms for sparse matrices. *SIAM Journal on Numerical Analysis*, 13(2):198–213, 1976.

[LS20] Steinar Laenen and He Sun. Higher-order spectral clustering of directed graphs. *Advances in neural information processing systems*, 33:941–951, 2020.

[Man17] Pasin Manurangsi. Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–961, 2017.

[MGST70] Richard L. Mattson, Jan Gecsei, Donald R. Slutz, and Irving L. Traiger. Evaluation techniques for storage hierarchies. *IBM Systems journal*, 9(2):78–117, 1970.

[Moh20] Bojan Mohar. A new kind of Hermitian matrices for digraphs. *Linear Algebra and its Applications*, 584:343–352, 2020.

[MPS17] Orlando Moreira, Merten Popp, and Christian Schulz. Graph Partitioning with Acyclicity Constraints. In Costas S. Iliopoulos, Solon P. Pissis, Simon J. Puglisi, and Rajeev Raman, editors, *16th International Symposium on Experimental Algorithms (SEA 2017)*, volume 75 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[MPS18] Orlando Moreira, Merten Popp, and Christian Schulz. Evolutionary multi-level acyclic graph partitioning. In *Proceedings of the genetic and evolutionary computation conference*, pages 332–339, 2018.

[MPSG07] Burkhard Monien, Robert Preis, Stefan Schamberger, and T. Gonzalez. Approximation algorithms for multilevel graph partitioning. *Handbook of approximation algorithms and Metaheuristics*, 10:1–60, 2007.

[NM16] Maxim Naumov and Timothy Moon. Parallel spectral graph partitioning. Technical Report NVR-2016-001, NVIDIA, June 2016.

[NS12] Uwe Naumann and Olaf Schenk. *Combinatorial Scientific Computing*. Chapman & Hall/CRC, 1st edition, 2012.

[ÖBÇ19] Yusuf M. Özkaya, Anne Benoit, and Ümit V. Çatalyürek. Is acyclic directed graph partitioning effective for locality-aware scheduling? In *PPAM 2019-13th International Conference on Parallel Processing and Applied Mathematics*, 2019.

[PAKY22] Pál András Papp, Georg Anegg, Aikaterini Karanasiou, and Albert-Jan N. Yzelman. HyperDAG_DB. https://github.com/Algebraic-Programming/HyperDAG_DB, 2022.

[PAKY24] Pál András Papp, Georg Anegg, Aikaterini Karanasiou, and Albert-Jan N. Yzelman. Efficient multi-processor scheduling in increasingly realistic models. In *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '24, pages 463–474, New York, NY, USA, 2024. Association for Computing Machinery.

[Pet13] Jordi Petit. Addenda to the survey of layout problems. *Bulletin of EATCS*, 3(105), 2013.

[PSSS21] Merten Popp, Sebastian Schlag, Christian Schulz, and Daniel Seemaier. Multilevel acyclic hypergraph partitioning. In *2021 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 1–15. SIAM, 2021.

[PSVY23] Dimosthenis Pasadakis, Olaf Schenk, Verner Vlačić, and Albert-Jan N. Yzelman. Nonlinear spectral clustering with C++ GraphBLAS, 2023.

[Räc08] Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 255–264, 2008.

[SJ89] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133, 1989.

[SKK$^+$00] Kirk Schloegel, George Karypis, Vipin Kumar, et al. *Graph partitioning for high performance scientific simulations*, volume 1. Citeseer, 2000.

[Sma73] Henry Small. Co-citation in the scientific literature: a new measure of the relationship between two documents. *Journal of the American Society for information Science*, 24(4):265–269, 1973.

[SP11] Venu Satuluri and Srinivasan Parthasarathy. Symmetrizations for clustering directed graphs. In *Proceedings of the 14th international conference on extending database technology*, pages 343–354, 2011.

[SPK$^+$18] Toby Simpson, Dimosthenis Pasadakis, Drosos Kourounis, Kohei Fujita, Takuma Yamaguchi, Tsuyoshi Ichimura, and Olaf Schenk. Balanced graph partition refinement using the graph p-Laplacian. In *Proceedings of the platform for advanced scientific computing conference*, pages 1–11, 2018.

[SS13] Peter Sanders and Christian Schulz. Think locally, act globally: highly balanced graph partitioning. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, volume 7933 of *LNCS*, pages 164–175. Springer, 2013.

[SSSW17] Peter Sanders, Christian Schulz, Darren Strash, and Robert Williger. Distributed evolutionary k-way node separators. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 345–352, 2017.

[SW04] Stefan Schamberger and Jens-Michael Wierum. A locality preserving graph ordering approach for implicit partitioning: graph-filing curves. In *PDCS*, pages 51–57, 2004.

[VLCD19] Hadrien Van Lierde, Tommy W. S. Chow, and Jean-Charles Delvenne. Spectral clustering algorithms for the detection of clusters in block-cyclic and block-acyclic graphs. *Journal of Complex Networks*, 7(1):1–53, 2019.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[WYLL16] Hao Wei, Jeffrey Xu Yu, Can Lu, and Xuemin Lin. Speedup graph processing by graph ordering. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1813–1828, 2016.

[YB09] Albert-Jan N. Yzelman and Rob H. Bisseling. Cache-oblivious sparse matrix–vector multiplication by using sparse matrix partitioning methods. *SIAM Journal on Scientific Computing*, 31(4):3128–3154, 2009.

[YB11] Albert-Jan N. Yzelman and Rob H. Bisseling. Two-dimensional cache-oblivious sparse matrix–vector multiplication. *Parallel Computing*, 37(12):806–819, 2011.

[ZDC25] Ning Zhang, Xiaowen Dong, and Mihai Cucuringu. Spectral clustering for directed graphs via likelihood estimation on stochastic block models. *Preprint*, 2025. arXiv:2403.19516.

[ZHS05] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on Machine learning*, pages 1036–1043, 2005.

(Dimosthenis Pasadakis) Institute of Computing, Università della Svizzera italiana, Via la Santa 1, 6962 Lugano, Switzerland
Email address: dimosthenis.pasadakis@usi.ch

(Raphael S. Steiner) Huawei Research Center Zurich, Computing Systems Lab, Thurgauerstrasse 80, 8050 Zürich, Switzerland
Email address: raphael.steiner@huawei.com

(Pál András Papp) Huawei Research Center Zurich, Computing Systems Lab, Thurgauerstrasse 80, 8050 Zürich, Switzerland
Email address: pal.andras.papp@huawei.com

(Toni Böhnlein) Huawei Research Center Zurich, Computing Systems Lab, Thurgauerstrasse 80, 8050 Zürich, Switzerland
Email address: toni.boehnlein@huawei.com

(Albert-Jan N. Yzelman) Huawei Research Center Zurich, Computing Systems Lab, Thurgauerstrasse 80, 8050 Zürich, Switzerland
Email address: albertjan.yzelman@huawei.com