COMPRESSING GRAPHS AND INDEXES [SIC] WITH RECURSIVE GRAPH BISECTION

Laxman Dhulipala*, Igor Kabilgo†, Brian Karrer†,
Guiseppe Ottaviano†, Gergey Pupyrev†, Alan Shalita†
*Carnegie Mellon University, †Facebook
Published at KDD 2016

Presented by Victor A. Ying
6.886 – April 11, 2018

Background and motivation

- Need quick lookups of adjacency information for any given vertex. For example:
  - *Given an individual, find friends in social network graph*
  - *Given a search term, find documents in inverted index*
- These graphs are huge. Compression fits bigger graphs into memory on fewer machines
- Prior work has shown reordering vertices according to community structure improves compression.
- Compression has a well-developed theory (information theory)

Graph reordering improves compression

1 → (408, 10214, 18422, 352225)
2 → (34, 408, 2243, 90899, 99244, 106888)
3 → (9842, 10332, 13248)

diffs

1 → (408, +9806, +18422, +333803)
2 → (34, +374, +1835, +88656, +8345, +7644)
3 → (9842, +1390, +2916)

reorder

1 → (4, 16662, 16664, 16940)
2 → (4, 770, 775, 11032, 11096, 11114)
3 → (80422, 85430, 85445)

diffs

1 → (4, +16658, +2, +276)
2 → (4, +776, +1, +10257, +64, +18)
3 → (80422, +5008, +15)

Small diffs compress well
with variable-length codes

Prior theory of graph reordering

- Given a graph G = (V, E)
- find a reordering π : V → {1, 2, 3, … , |V|} according to

$$\underset{\pi}{\operatorname{argmin}} \sum_{(u,v)\in E} cost(\pi(u), \pi(v))$$

Prior theory of graph reordering

- Given a graph G = (V, E)
- find a reordering π : V → {1, 2, 3, … , |V|} according to

$$\underset{\pi}{\operatorname{argmin}} \sum_{(u,v)\in E} |\pi(u) - \pi(v)|$$

Minimum linear arrangement (MLA) problem
Previously known to be NP-hard
and APX-hard under Unique Games Conjecture

Prior theory of graph reordering for compression

- Given a graph G = (V, E)
- find a reordering π : V → {1, 2, 3, … , |V|} according to

$$\operatorname*{argmin}_{\pi} \sum_{(u,v)\in E} \log|\pi(u) - \pi(v)|$$

Minimum logarithmic arrangement (MLogA) problem
Previously known to be NP-hard
Solutions are different than those of MLA

New theory of graph reordering for compression

- Given a graph G = (V, E)
- find a reordering π : V → {1, 2, 3, … , |V|} according to

$$\underset{\pi}{\operatorname{argmin}} \sum_{v \in V} \sum_{i=1}^{\deg_v - 1} \log\left|\pi(n_{v,i+1}) - \pi(n_{v,i})\right|$$

Minimum logarithmic gap arrangement (MLogGapA) problem
proven in this paper to be NP-hard

New theory of graph and index reordering for compression

- Given a graph G = (Q∪D, E)
- find a reordering π : D → {1, 2, 3, … , |D|} according to

$$\underset{\pi}{\operatorname{argmin}} \sum_{q \in Q} \sum_{i=1}^{\deg_q - 1} \log\left|\pi\left(d_{q,i+1}\right) - \pi\left(d_{q,i}\right)\right|$$

Bipartite minimum logarithmic arrangement (BiMLogA) problem proven in this paper to be NP-hard

Approximation algorithms

Input: graph G

1. Find a bisection (G_1, G_2) of G;
2. Recursively find linear arrangements for G_1 and G_2;
3. Concatenate the resulting orderings;

Algorithm 1: Graph Reordering using Graph Bisection

Finding a good bisection

Input : graph $G = (\mathcal{Q} \cup \mathcal{D}, E)$
Output: graphs $G_1 = (\mathcal{Q} \cup V_1, E_1), G_2 = (\mathcal{Q} \cup V_2, E_2)$

determine an initial partition of $\mathcal{D}$ into V_1 and V_2;
repeat
    for $v \in \mathcal{D}$ **do**
        $gains[v] \leftarrow ComputeMoveGain(v)$
    $S_1 \leftarrow$ sorted V_1 in descending order of $gains$;
    $S_2 \leftarrow$ sorted V_2 in descending order of $gains$;
    for $v \in S_1,\ u \in S_2$ **do**
        if $gains[v] + gains[u] > 0$ **then**
            exchange v and u in the sets;
        else break;
until *converged* **or** *iteration limit exceeded*;
return *graphs induced by* $\mathcal{Q} \cup V_1$ *and* $\mathcal{Q} \cup V_2$

Algorithm 2: Graph Bisection

Computing the gains of moving a vertex between partitions

$$\sum_{q \in \mathcal{Q}} \left(\deg_1(q) \log\left(\frac{n_1}{\deg_1(q)+1}\right) + \deg_2(q) \log\left(\frac{n_2}{\deg_2(q)+1}\right) \right)$$

Evaluation – Inputs

Graph	$\lvert\mathcal{Q}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert E\rvert$
Enron	9,660	9,660	224,896
AS-Oregon	13,579	13,579	74,896
FB-NewOrlean	63,392	63,392	1,633,662
web-Google	356,648	356,648	5,186,648
LiveJournal	4,847,571	4,847,571	85,702,474
Twitter	41,652,230	41,652,230	2,405,026,092
Gov2	39,187	24,618,755	5,322,924,226
ClueWeb09	96,741	50,130,884	14,858,911,083
FB-Posts-1B	60×10^3	1×10^9	20×10^9
FB-300M	300×10^6	300×10^6	90×10^9
FB-1B	1×10^9	1×10^9	300×10^9

Table 1: Basic properties of our dataset.

Evaluation – Graphs

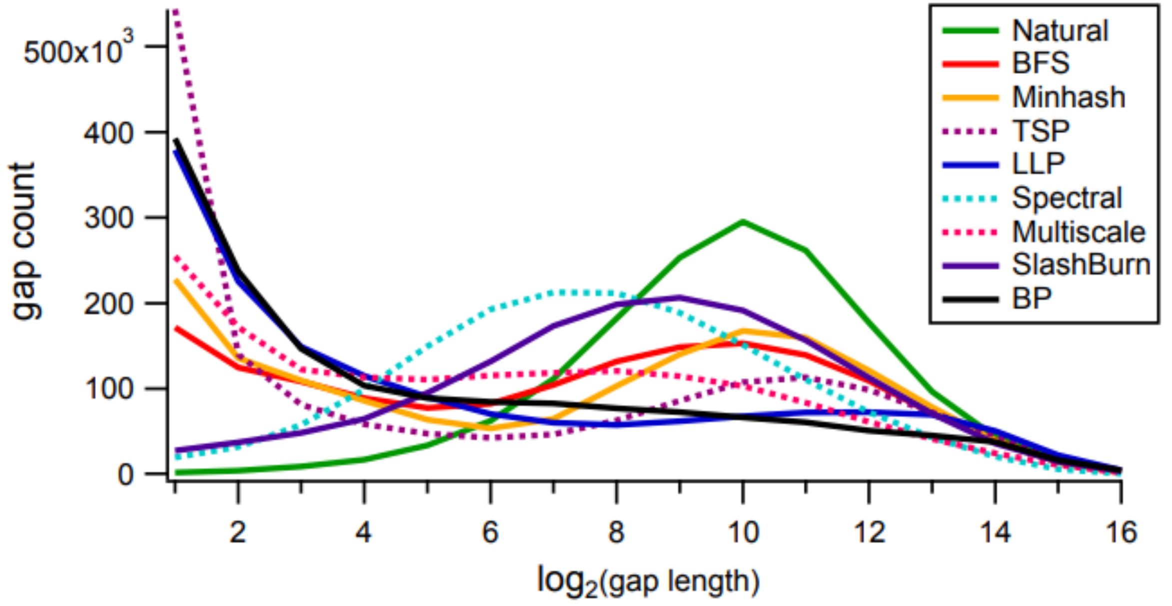
Graph	Algorithm	LogGap	Log	BV
Enron	Natural	5.01	9.82	7.80
	BFS	4.86	9.97	7.70
	Minhash	4.91	10.12	7.68
	TSP	3.95	9.46	6.58
	LLP	3.96	8.55	6.51
	Spectral	5.43	9.41	8.60
	Multiscale	4.23	**8.00**	6.90
	SlashBurn	5.11	10.18	8.05
	BP	**3.69**	8.26	**6.24**
AS-Oregon	Natural	7.88	12.06	13.34
	BFS	4.71	11.06	7.97
	Minhash	4.47	11.17	7.56
	TSP	3.59	10.39	6.66
	LLP	4.42	8.32	7.47
	Spectral	5.64	9.53	8.76
	Multiscale	4.53	**7.23**	7.31
	SlashBurn	4.50	10.66	8.74
	BP	**3.15**	9.21	**6.25**
FB-NewOrlean	Natural	9.74	14.29	14.64
	BFS	7.16	12.63	10.79
	Minhash	7.06	12.57	10.62
	TSP	5.62	11.61	8.96
	LLP	5.37	**9.41**	8.54
	Spectral	7.64	11.49	11.79
	Multiscale	5.90	9.58	9.25
	SlashBurn	8.37	13.06	12.65
	BP	**4.99**	9.45	**8.16**
web-Google	Natural	13.39	16.74	20.08
	BFS	5.57	11.21	7.69
	Minhash	5.65	13.14	6.87
	TSP	3.28	7.99	4.77
	LLP	3.75	6.70	5.13
	Spectral	6.68	10.25	9.16
	Multiscale	**2.72**	**4.82**	**4.10**
	SlashBurn	8.02	14.46	10.29
	BP	3.17	7.74	4.68
LiveJournal	Natural	10.43	17.44	14.61
	BFS	10.52	17.59	14.69
	Minhash	10.79	17.76	15.07
	LLP	7.46	**12.25**	11.12
	BP	**7.03**	12.79	**10.73**
Twitter	Natural	15.23	23.65	21.56
	BFS	12.87	22.69	17.99
	Minhash	10.43	21.98	14.76
	BP	**7.91**	**20.50**	**11.62**
FB-300M	Natural	17.65	25.34	
	Minhash	13.06	24.9	
	BP	**8.39**	**18.13**	
FB-1B	Natural	19.63	27.22	
	Minhash	14.60	26.89	
	BP	**8.66**	**18.36**	

Evaluation – Inverted Indexes

Index	Algorithm	LogGap	PEF	BIC
Gov2	Natural	2.12	3.12	2.52
	BFS	2.07	3.00	2.44
	Minhash	2.12	3.12	2.52
	BP	**1.81**	**2.44**	**1.95**
ClueWeb09	Natural	2.91	4.99	4.05
	BFS	2.91	4.99	4.06
	Minhash	2.91	4.99	4.05
	BP	**2.55**	**4.34**	**3.50**
FB-Posts-1B	Natural	8.03	10.19	9.95
	Minhash	3.41	4.96	4.24
	BP	**2.95**	**4.18**	**3.61**

Table 3: Reordering results of various algorithms on inverted indexes with highlighted best results.

Why does it work?



(a) FB-NewOrlean

LogGapA is well-correlated with actual compression rates

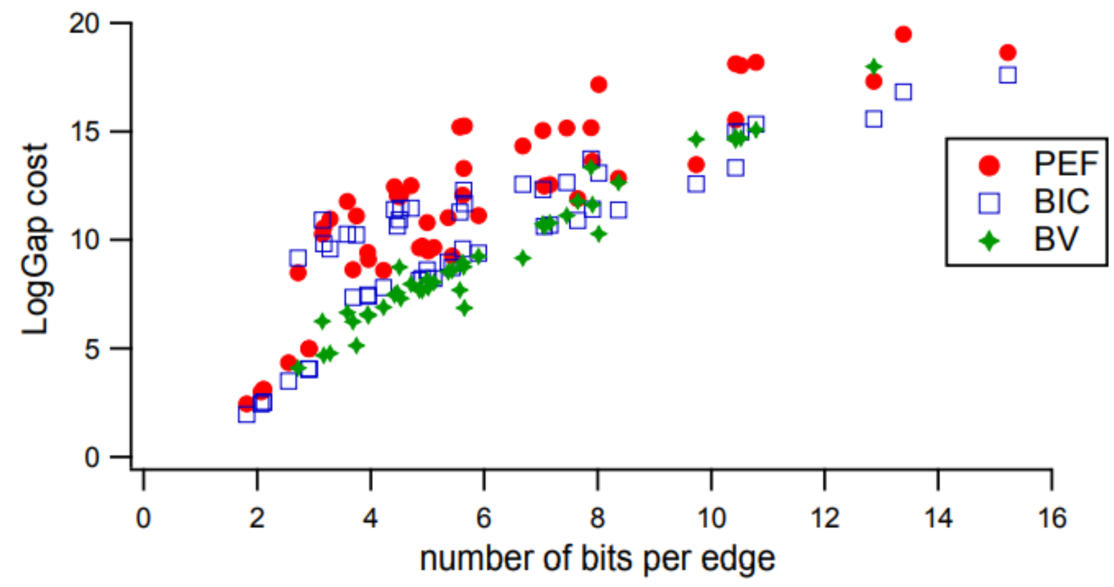


Figure 8: LogGap cost against the average number of bits per edge using various encoding schemes.

Visualizing adjacency matrices

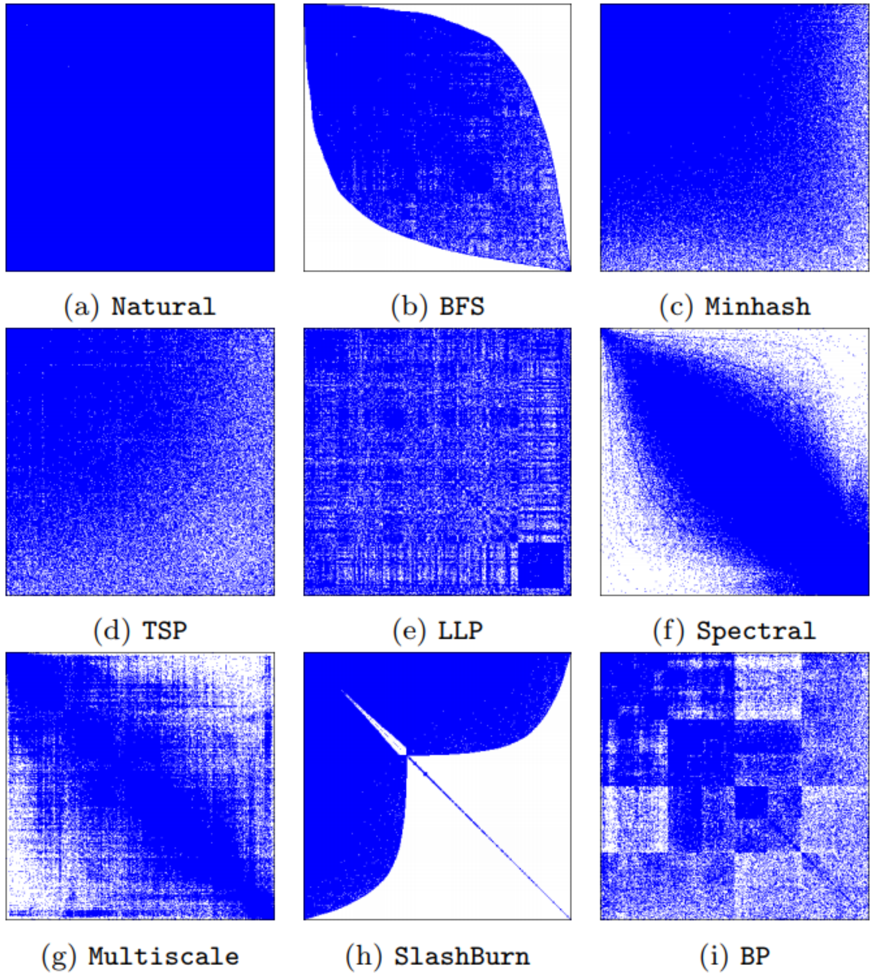


Figure 6: Adjacency matrices of FB-NewOrlean after applying various reordering algorithms; nonzero elements are blue.

Sensitivity studies – Initialization

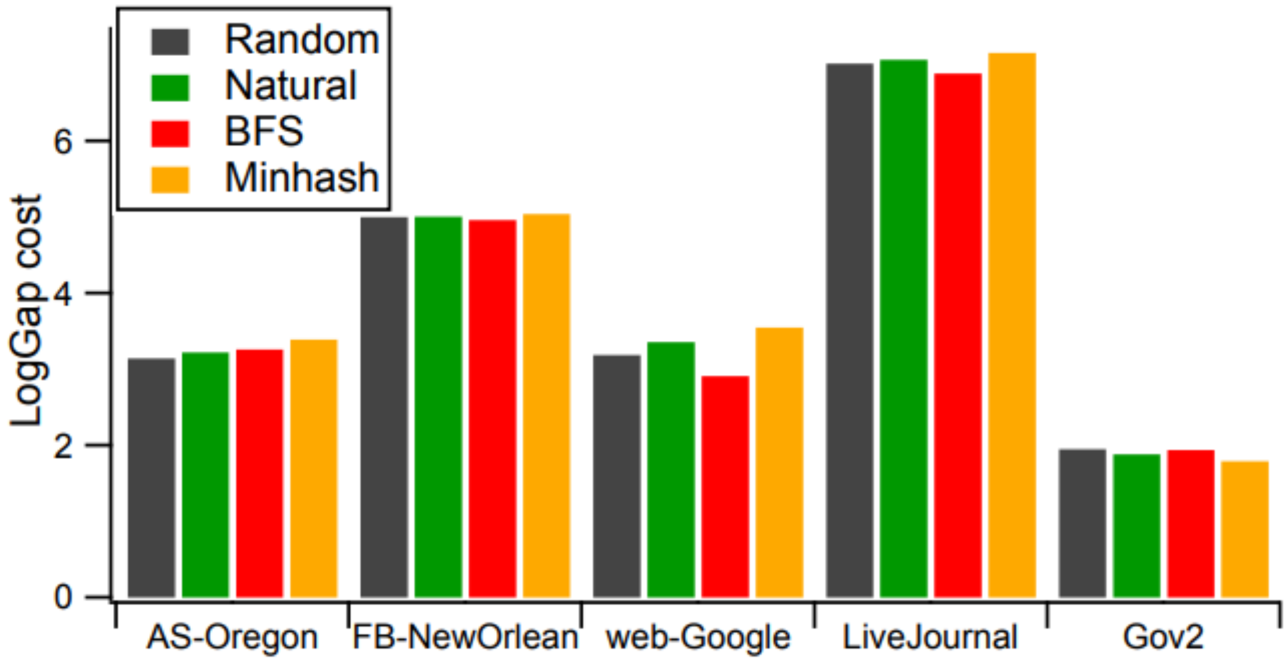


Figure 3: LogGap cost of the resulting order produced with different initialization approaches for graph bisection.

Sensitivity studies – Recursion depth

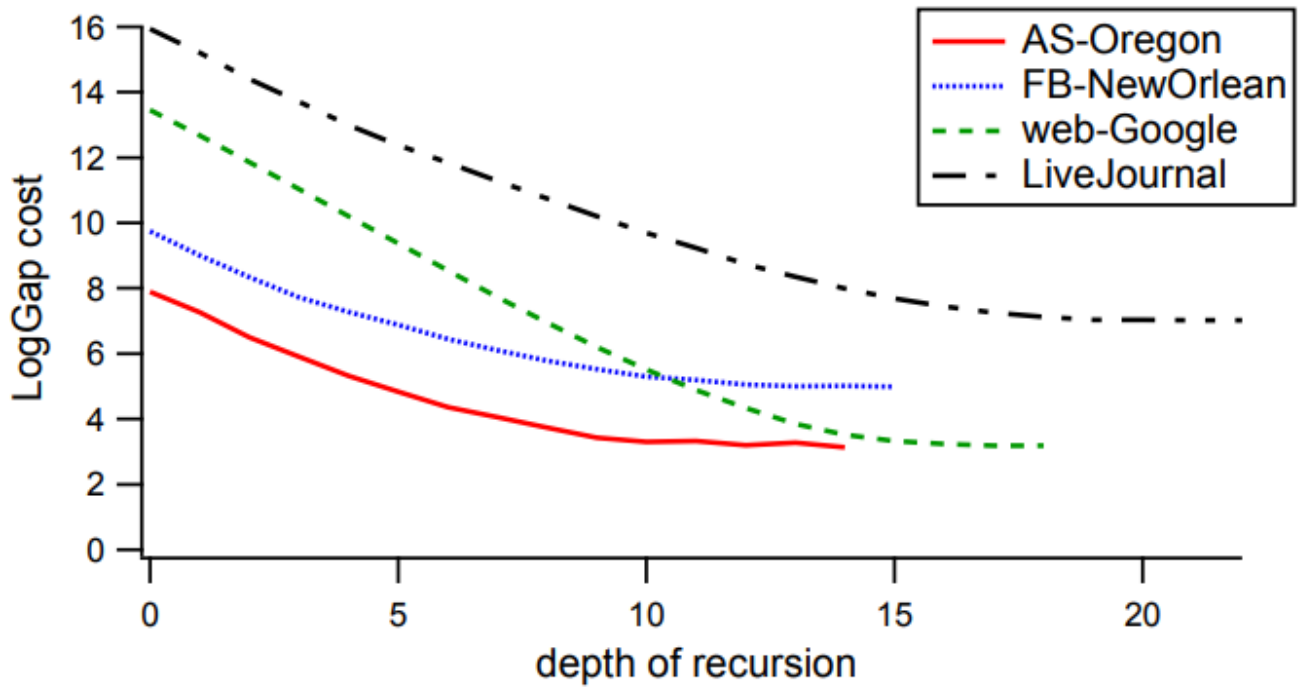


Figure 4: LogGap cost of the resulting order produced with a fixed depth of recursion. Note that the last few splits make insignificant contributions to the final quality.

Sensitivity studies – Refinement iterations

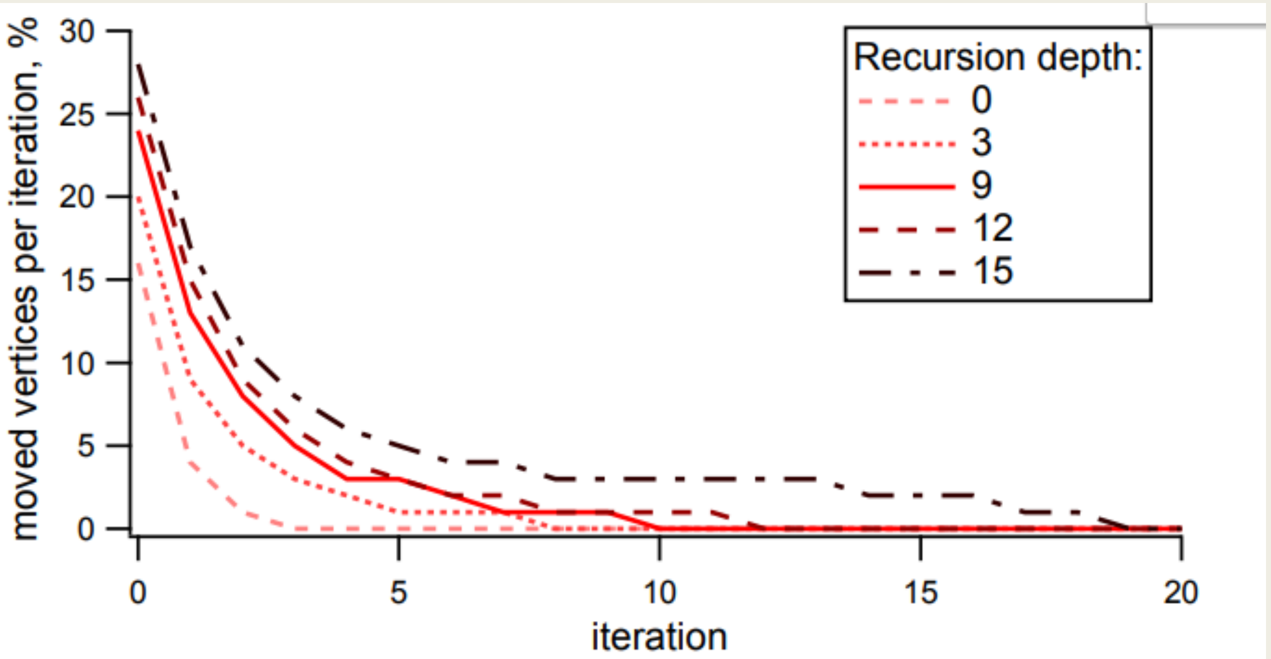


Figure 5: The average percentage of moved vertices on an iteration of Algorithm 2 for various levels of recursion. The data is computed for `LiveJournal`.

Conclusions

- Theory of compression and reordering gave rise to more effective heuristics for partitioning, resulting in better reorderings for compression.
- Connection between this work and reordering for locality?
  – *Maybe locality doesn’t care about gap sizes larger than a cache block?*
- Can we develop better approximation algorithms?
  – *Beyond balanced bisections?*
- Optimal algorithms for small graphs?