

Pushing the Envelope in Graph Compression

Panagiotis Liakos¹ **Katia Papakonstantinou**¹ Michael Sioutis²

¹University of Athens, ²Université Lille-Nord de France, CRIL-CNRS

23rd ACM International Conference on Information and Knowledge Management
Shanghai, November 6th, 2014

Motivation

- Explosive growth of large-scale systems modeled as graphs
 - Web Graphs
 - Social Networks
- Critical applications (e.g., in the field of Graph Mining) need **in-memory** graph representations
 - Serving adjacency queries
 - Maintaining snapshots for archival purposes

We need efficient compressed graph representations!

Graph compression / Compact graph representation:

Find a **compressed** graph representation that allows **mining without decompressing** the graph.

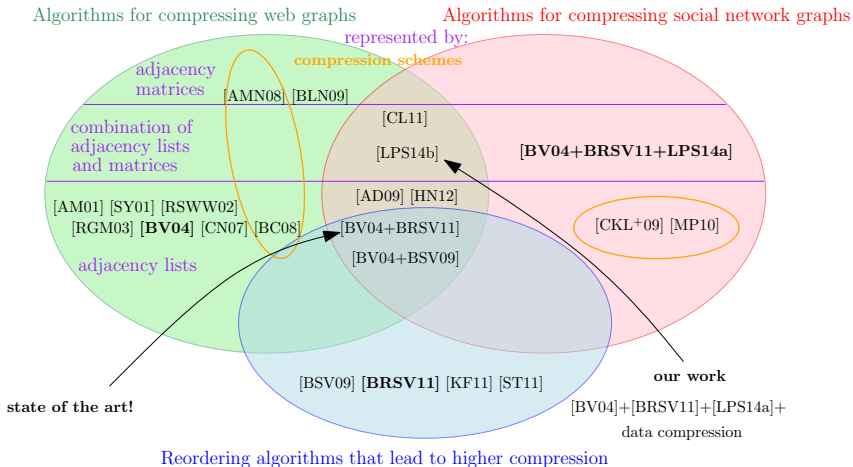
We are interested in **graphs created by human activity** (e.g., web graphs, social network graphs).

They exhibit, under certain orderings, **common properties**:

- power law distributed degrees,
- locality (of reference) and
- copy (or similarity) property

which induce **redundancy** in the graphs' representations, and are taken into account in the design of compression methods.

Graph compression methods



State of the art is BV: Further minimize space / access time

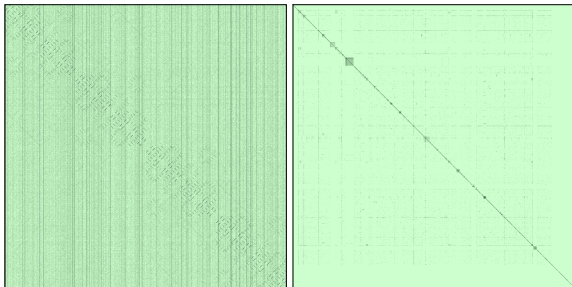
- 1 State of the art
- 2 Our method
- 3 Our results

Laboratory for Web Algorithmics' (LAW) techniques

- Modified gap representation [BV04] [▶ Details](#)
 - Represent each list of successors as a list of gaps to exploit the *locality of reference*
- Reference list [BV04] [▶ Details](#)
 - Code each adjacency list as a “modified” version of a previous list to exploit *similarity (copy-property)*
- Layered Label Propagation [BRSV11] [▶ Details](#)
 - Reorders very large graphs to provide a major increase in compression with respect to prior efforts

Layered Label Propagation effect

Graph cit-Patents before and after LLP reordering:

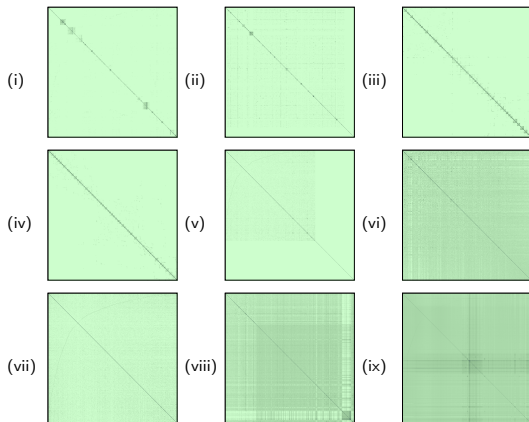


Observation: The area around the main diagonal is dense under certain orderings [LPS14a]

Main idea behind our method

Main idea

As the **area around the main diagonal** of the graph's adjacency matrix is **dense** after reordering, **isolate** it and **compress** it separately.



Heat maps of the adjacency matrices of:

- (i) cnr-2000
- (ii) web-Stanford
- (iii) roadNet-CA
- (iv) roadNet-PA
- (v) dblp-2010
- (vi) cit-Patents
- (vii) amazon-2008
- (viii) ljournal-2008
- (ix) twitter-2010

Our method in a glance

Algorithm 1: $BV+(G, k, b)$

input : A directed graph $G = (V, E)$ and parameters k and b .
output : A compressed representation of G .
begin

Isolate a dense subgraph S
around the main diagonal

Apply data compression on S

Compress $G \setminus S$ using the
state-of-the-art method of Boldi
et al.[BV04, BRSV11] (or any
other efficient graph compression
method)

```
setNonD ← set();  
k-diagonalStripe ← array(array([000 . . . 0]) × |V|);  
                                2k+1 bits
```

```
foreach  $(u, v) \in E$  do  
  if  $u - k \leq v \leq u + k$  then  
    |  $k$ -diagonalStripe[u][v] ← 1;  
  else  
    | setNonD ← setNonD ∪ (u, v);
```

```
seqDict ← dict();  
foreach seq ∈ k-diagonalStripe do  
  if seq ∉ seqDict then  
    | seqDict[seq] ← 1;  
  else  
    | seqDict[seq]++;
```

```
foreach (key, value) ∈ seqDict do  
  | seqDict[key] ← value × # of 1s ∈ key;
```

```
seqDict ← sort seqDict by value (desc. order);
```

```
seqSet ← {first  $2^b - 1$  sequences (keys) of seqDict};
```

```
foreach seq ∈ k-diagonalStripe do  
  if seq ∈ seqSet then  
    | use  $b$  bits to compress seq;  
  else  
    | bestSeq ← bestSubset(seqSet, seq, k);  
    | use  $b$  bits to compress bestSeq;  
    | setNonD ← setNonD ∪ {edges of seq that were left out of bestSeq};
```

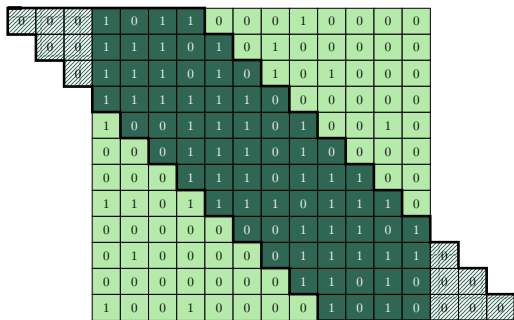
```
compress setNonD using BV;
```

Our techniques

- Isolating a dense subgraph
- Data compression

Isolating a dense subgraph

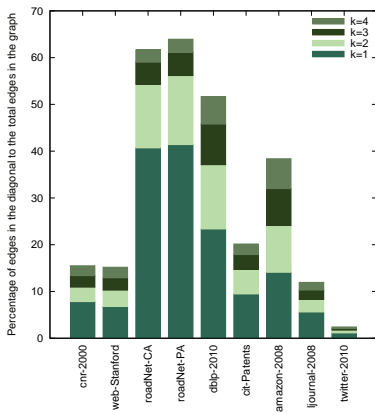
Diagonal stripe: For $G = (V, E)$ and $k \in \mathbb{Z}_+$, the k -diagonal stripe of G is the set: $\{(i, j) \mid i - k \leq j \leq i + k \text{ and } i, j \in \{0, \dots, |V|\}\}$.



The bits/edge required to store the stripe increases with k . However, even a sparser stripe may lead to higher compression of the graph!

Isolating a dense subgraph

Diagonal stripe: For $G = (V, E)$ and $k \in \mathbb{Z}_+$, the k -diagonal stripe of G is the set: $\{(i, j) \mid i - k \leq j \leq i + k \text{ and } i, j \in \{0, \dots, |V|\}\}$.



Web, road network, and social network graphs

The bits/edge required to store the stripe increases with k . However, even a sparser stripe may lead to higher compression of the graph!

Isolating a dense subgraph (cont.)

The computation of the bits/edge needed to represent the diagonal stripe of a given graph is straightforward:

roadNet-PA has 68.41% of its edges in the 7-diagonal

$$\text{uncompressed diagonal ratio} = \frac{(2k+1)|V|}{p|E|} = \frac{15|V|}{0.6841|E|} = 7.76 \text{ bits/edge.}$$

However, BV yields a compression ratio of 12.86 bits/edge for roadNet-PA.

Can we do better?

Isolating a dense subgraph (cont.)

Based on Shannon's source coding theorem, we get an indication of the expected compression ratio for the diagonal stripe.

Proposition 1.

Consider $k \in \mathbb{Z}_+$ and a graph $G = (V, E)$ with a percentage p of its edges belonging in the k -diagonal stripe. The minimum expected compression ratio of the diagonal stripe is upper bounded by $\frac{\log \left(\frac{(2k+1)|V|}{p|E|} \right)}{p|E|}$ bits/edge.

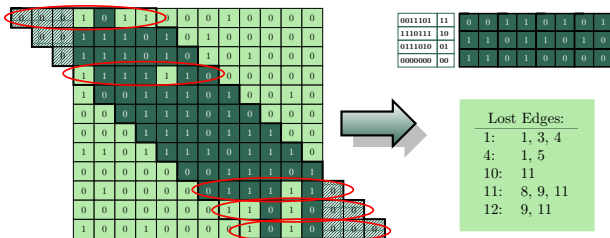
Estimation of the compressed roadNet-PA diagonal ratio

$$\text{expected compression ratio} = \frac{\log \left(\frac{15|V|}{0.6841|E|} \right)}{0.6841|E|} = 2.98 \text{ bits/edge} \ll 12.86 \text{ (BV)}$$

Our method is likely to offer a significant improvement!

Data compression on the dense subgraph

Consider the following graph and assume $k = 3$ and $b = 2$.

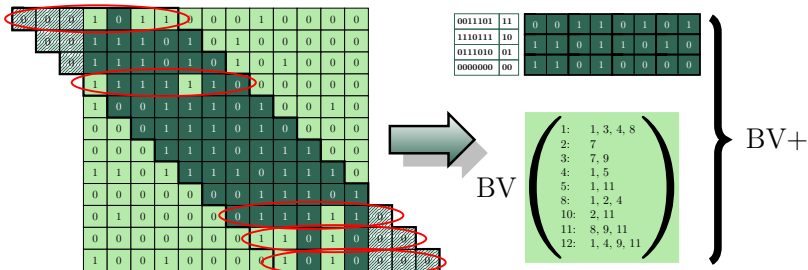


- $\frac{36}{47}$ edges are held in $\frac{b}{2k+1} = \frac{2}{7}$ of the original diagonal space!
- The “lost” edges are passed on to the next step (\rightsquigarrow lossless)

roadNet-PA compressed diagonal ratio

$k = 7, b = 2 \Rightarrow 1.95$ bits/edge.

Putting everything together



- Size of compressed graph: $b|V| + S_{BV}$

	Time Complexity	Edges in comp. diagonal	Edges outside comp. diagonal
• Edge Exists		$O(1)$	$< BV$
Successors		$O(b)$	$< BV$

Experimental evaluation: Compression ratios

graph	# nodes	# edges	% of edges in diagonal	compression ratio (bits/edge)		BV+ parameters		
				BV	BV+	<i>k</i>	<i>b</i>	
cnr-2000	325,557	3,216,152	6.05 %	3.71	3.62	17	2	▼
web-Stanford	281,903	3,985,272	6.78 %	4.06	3.90	1	2	▼
roadnet-CA	1,965,206	5,533,214	64.50 %	13.30	10.58	7	6	▼
roadnet-PA	1,088,092	3,083,796	66.89 %	12.86	10.07	7	6	▼
dblp-2010	326,186	1,615,400	54.49 %	8.63	7.2	24	7	▼
cit-Patents	3,774,767	33,037,894	19.07 %	14.72	14.25	9	6	▼
amazon-2008	735,323	5,158,388	59.27 %	10.77	10.07	23	15	▼
ljournal-2008	5,363,260	79,023,142	7.65 %	11.84	11.78	2	4	▼
twitter-2010	41,652,230	1,468,365,182	2.58 %	14.52	14.42	17	6	▼

- Large impact on social network graphs (dblp-2010: 16.6%)
- Impressive results for road network graphs (> 20%)
- Improvements on web graphs (2.4% – 3.9%)

Experimental evaluation: Access times (ms)

<i>graph</i>		cnr-2000	roadNet-CA	ljournal-2008
<i>Edge Exists</i>	BV+ (D)	645	600	663
	BV+ (Non-D)	2,286	832	4,373
	BV	2,397	923	4,518
<i>Successors</i>	BV+ (D)	643	668	627
	BV+ (Non-D)	1,947	849	1,940
	BV	2,159	891	2,009

- Significantly faster for the compressed diagonal part
- Faster than *BV* for the rest of the graph

We outscore *BV* in any non-single core environment

Effect of parameters

We can estimate a good k using Proposition 1.

Delicate balance between:

- Minimizing the diagonal stripe ratio
- Easing the task of compressing the rest of the graph

For our dataset: $k \in [2, 20]$ and $b \leq k$

Best choice for roadNet-PA

$k = 7, b = 6 \Rightarrow 3.27$ bits/edge (although it is > 1.95 bits/edge).

References I

- [AD09] Alberto Apostolico and Guido Drovandi, *Graph Compression by BFS*, Algorithms 2 (2009), no. 3, 1031–1044.
- [AM01] Micah Adler and Michael Mitzenmacher, *Towards Compressing Web Graphs*, DCC, 2001.
- [AMN08] Yasuhito Asano, Yuya Miyawaki, and Takao Nishizeki, *Efficient Compression of Web Graphs*, COCOON, 2008.
- [BC08] Gregory Buehrer and Kumar Chellapilla, *A scalable pattern mining approach to web graph compression with communities*, WSDM, 2008.
- [BLN09] Nieves R. Brisaboa, Susana Ladra, and Gonzalo Navarro, *k2-Trees for Compact Web Graph Representation*, SPIRE, 2009.
- [BRSV11] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna, *Layered Label Propagation: A MultiResolution Coordinate-Free Ordering for Compressing Social Networks*, WWW, 2011.
- [BSV09] Paolo Boldi, Massimo Santini, and Sebastiano Vigna, *Permuting Web and Social Graphs*, Internet Mathematics 6 (2009), no. 3, 257–283.
- [BV04] P. Boldi and S. Vigna, *The WebGraph Framework I: Compression Techniques*, WWW, 2004.
- [CKL⁺09] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan, *On compressing social networks*, KDD, 2009.
- [CL11] Francisco Claude and Susana Ladra, *Practical Representations for Web and Social Graphs*, CIKM, 2011.
- [CN07] Francisco Claude and Gonzalo Navarro, *A Fast and Compact Web Graph Representation*, SPIRE, 2007.
- [HN12] Cecilia Hernández and Gonzalo Navarro, *Compressed representation of web and social networks via dense subgraphs*, SPIRE, 2012.
- [KF11] U. Kang and Christos Faloutsos, *Beyond “caveman communities”: Hubs and spokes for graph compression and mining*, ICDM, 2011.
- [LPS14a] Panagiotis Liakos, Katia Papakonstantinou, and Michael Sioutis, *On the Effect of Locality in Compressing Social Networks*, ECIR, 2014.

References II

- [LPS14b] _____, *Pushing the envelope in graph compression*, CIKM, 2014.
- [MP10] Hossein Maserrat and Jian Pei, *Neighbor query friendly compression of social networks*, KDD, 2010.
- [RGM03] Sriram Raghavan and Hector Garcia-Molina, *Representing Web Graphs*, ICDE, 2003.
- [RSWW02] Keith H. Randall, Raymie Stata, Janet L. Wiener, and Rajiv G. Wickremesinghe, *The Link Database: Fast Access to Graphs of the Web*, DCC, 2002.
- [ST11] Ilya Safro and Boris Temkin, *Multiscale approach for the network compression-friendly ordering*, J. of Discrete Algorithms **9** (2011), no. 2, 190–202.
- [SY01] Torsten Suel and Jun Yuan, *Compressing the Graph Structure of the Web*, DCC, 2001.

Conclusion - Ongoing work

- We go beyond the state-of-the-art compressed data structure of Boldi et al. [BV04, BRSV11] for web and social graphs by further exploiting the clustering properties (*locality, similarity*) observed in these graphs.
- Our implementation can easily be employed to improve any compression method.
- We reduced the best recorded compression ratios of well-known datasets up to 21.7% and the corresponding access times up to 21.13%.
- We are currently examining labelings that favor our compression method.

Thank you!

For further details refer to:

<http://hive.di.uoa.gr/network-analysis>

or email me at: katia@di.uoa.gr

Modified gap representation

Represent each list of successors as a list of gaps to exploit the *locality of reference* [◀ Back](#)

Node	Outdegree	Successors
...
15	11	13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	15, 16, 17, 22, 23, 24, 315, 316, 317, 3041
17	0	
18	5	13, 15, 16, 17, 50
...

Node	Outdegree	Successors
...
15	11	3, 1, 0, 0, 0, 0, 3, 0, 178, 111, 718
16	10	1, 0, 0, 4, 0, 0, 290, 0, 0, 2723
17	0	
18	5	9, 1, 0, 0, 32
...

Reference list

Code each adjacency list as a “modified” version of a previous list to exploit *similarity (copy-property)* [◀ Back](#)

Node	Outdegree	Successors
...
15	11	13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	15, 16, 17, 22, 23, 24, 315, 316, 317, 3041
17	0	
18	5	13, 15, 16, 17, 50
...

Node	Outdegree	Reference	Copy list	Extra nodes
...		
15	11	0		13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	1	01110011010	22, 316, 317, 3041
17	0			
18	5	3	11110000000	50
...		