Berkeley
UNIVERSITY OF CALIFORNIA

Sparse Matrices for Scaling Data-Intensive Problems on Distributed-Memory Systems

Aydın Buluç
Computational Research Division, LBNL
EECS Department, UC Berkeley

SIAM CSE'21
March 1, 2021

Schedule

GraphBLAS: Tools, Algorithms, and Applications (Monday)
9:45 am: Aydin Buluc (Berkeley)
10:05 am: Tim Davis (Texas A&M)
10:25 am: Scott McMillan (CMU SEI)
10:45 am: Roger Pearce (LLNL)
11:05 am: Michel Pelletier (Graphegon)

2:15 pm: Ariful Azad (Indiana Univ.)
2:35 pm: Federico Busato (NVIDIA)
2:55 pm: Andrey Prokopenko (ORNL)
3:15 pm: Ben Brock (Berkeley)
3:35 pm: Vijay Thakkar (GATech)

Building Graph Algorithms with the GraphBLAS (Minitutorial)
Tuesday 2:15pm- 3:55pm (Part 1)
Wednesday 2:15pm- 3:55pm (Part 2)

The GraphBLAS forum

Standards for Graph Algorithm Primitives

Tim Mattson (Intel Corporation), David Bader (Georgia Institute of Technology), Jon Berry (Sandia National Laboratory), Aydin Buluc (Lawrence Berkeley National Laboratory), Jack Dongarra (University of Tennessee), Christos Faloutsos (Carnegie Melon University), John Feo (Pacific Northwest National Laboratory), John Gilbert (University of California at Santa Barbara), Joseph Gonzalez (University of California at Berkeley), Bruce Hendrickson (Sandia National Laboratory), Jeremy Kepner (Massachusetts Institute of Technology), Charles Leiserson (Massachusetts Institute of Technology), Andrew Lumsdaine (Indiana University), David Padua (University of Illinois at Urbana-Champaign), Stephen Poole (Oak Ridge National Laboratory), Steve Reinhardt (Cray Corporation), Mike Stonebraker (Massachusetts Institute of Technology), Steve Wallach (Convey Corporation), Andrew Yoo (Lawrence Livermore National Laboratory)

Abstract-- It is our view that the state of the art in constructing a large collection of graph algorithms in terms of linear algebraic operations is mature enough to support the emergence of a standard set of primitive building blocks. This paper is a position paper defining the problem and announcing our intention to launch an open effort to define this standard.

"If you want to go fast, go alone. If you want to go far, go together." -- unknown

https://graphblas.github.io/

GraphBLAS C API Specification

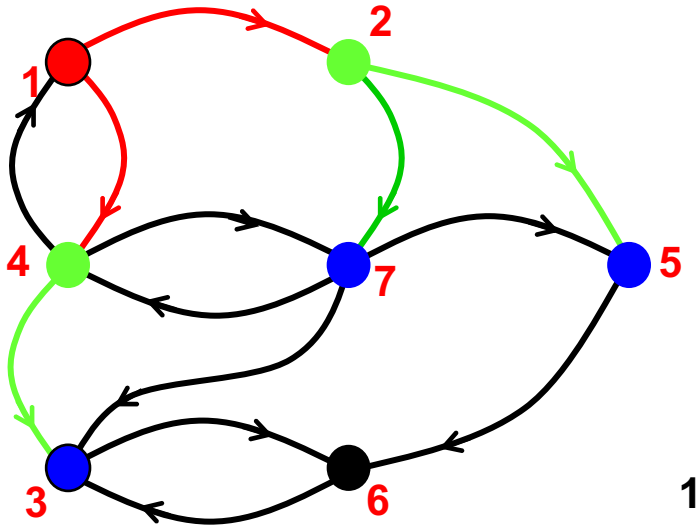
- **Goal:** A crucial piece of the GraphBLAS effort is to translate the mathematical specification to an actual Application Programming Interface (API) that
  i. is faithful to the mathematics as much as possible, and
  ii. enables efficient implementations on modern hardware.
- **Impact:** All graph and machine learning algorithms that can be expressed in the language of linear algebra
- **Innovation:** Function signatures (e.g. mxm, vxm, assign, extract), parallelism constructs (blocking v. non-blocking), fundamental objects (masks, matrices, vectors, descriptors), a hierarchy of algebras (functions, monoids, and semiring)

```
GrB_info GrB_mxm(GrB_Matrix          *C,      // destination
                 const GrB_Matrix     Mask,
                 const GrB_BinaryOp   accum,
                 const GrB_Semiring   op,
                 const GrB_Matrix     A,
                 const GrB_Matrix     B
                [, const Descriptor   desc]);
```

$$C(\neg M) \oplus= A^T \oplus.\otimes B^T$$

A.Buluç, T. Mattson, S. McMillan, J. Moreira, C. Yang. “The GraphBLAS C API Specification”, version 1.3.0

Pattern 1: Sparse matrix times sparse vector (SpMSpV)



Single-source traversal:

BFS, connected components, matching, ordering, etc.

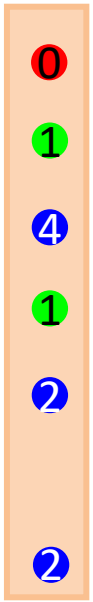
GrB_mxv(y, p, <semiring>, A, x, <desc>)

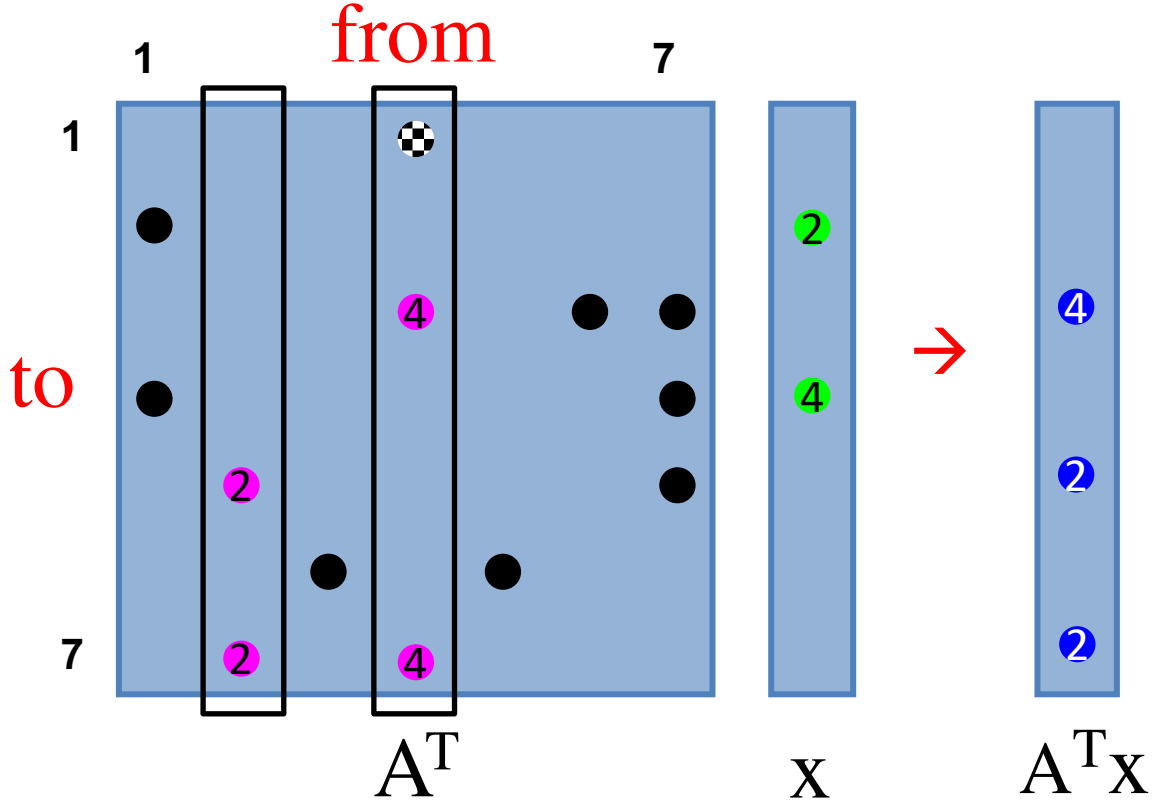
A: sparse adjacency matrix

x: sparse input vector (previous frontier)

p: mask (already discovered vertices)

parents (p):





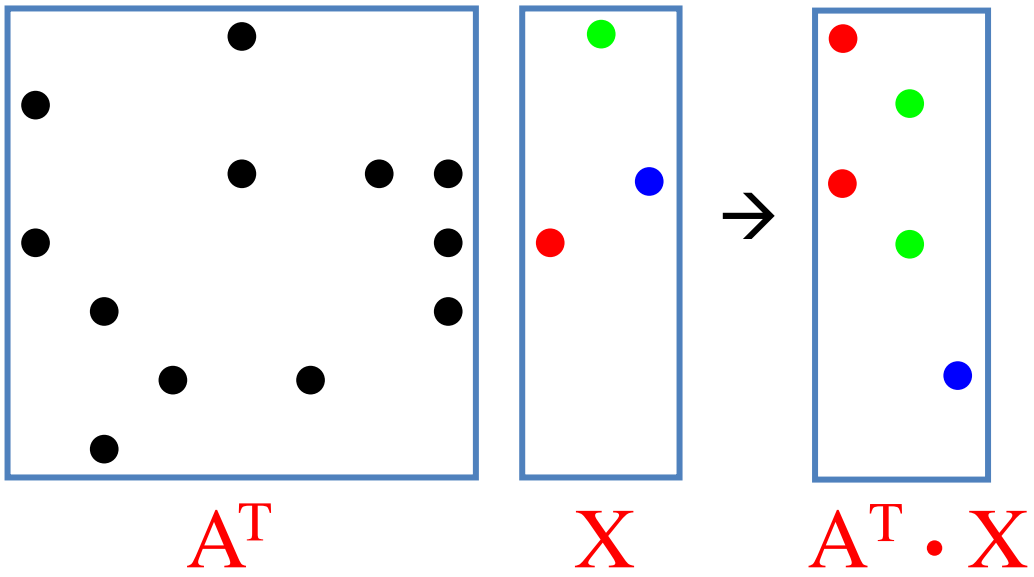
Pattern 2: Sparse matrix times sparse matrix (SpGEMM)

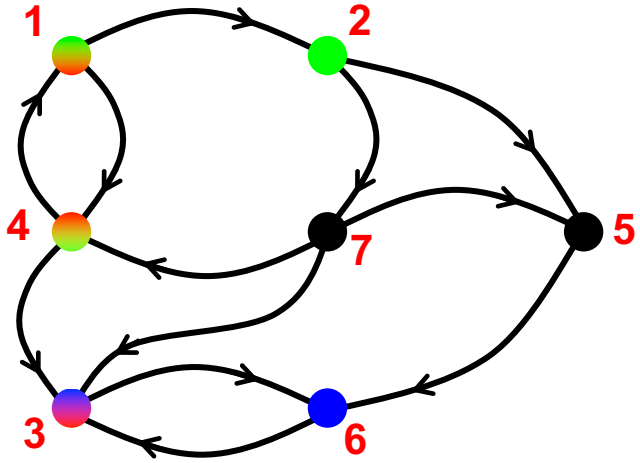
Multi-source traversal:

Ex: multi-source BFS, betweenness centrality, triangle counting*, Markov clustering*

GrB_mxm(Y, P, <semiring>, A, X, <desc>)

A: sparse adjacency matrix
X: sparse input matrix (previous frontier), n-by-b where b is the #sources
P: mask (already discovered vertices), multi-vector version of p from previous slide

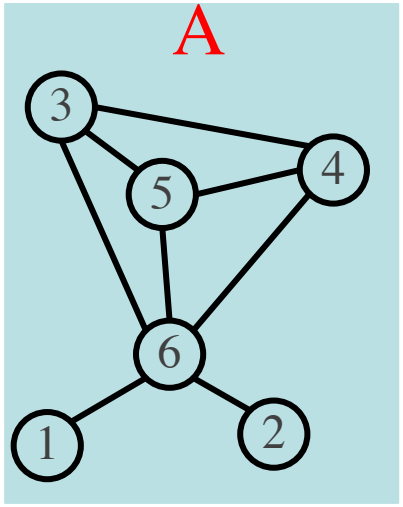




Pattern 2: Sparse matrix times sparse matrix (SpGEMM)

Triangle counting is also multi-source(in fact, all sources) traversal:
It just stops after one traversal iteration only, discovering all wedges

GrB_mxm(C, A, <semiring>, L, U, <desc>)

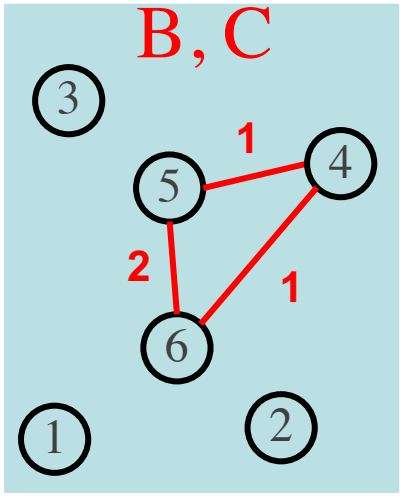


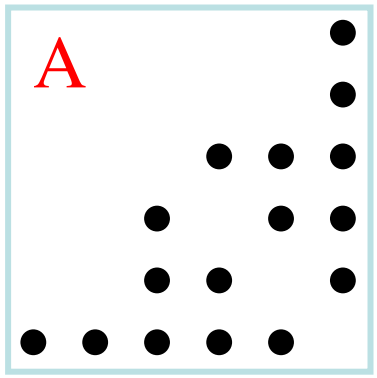
$A = L + U$ (hi->lo + lo->hi)

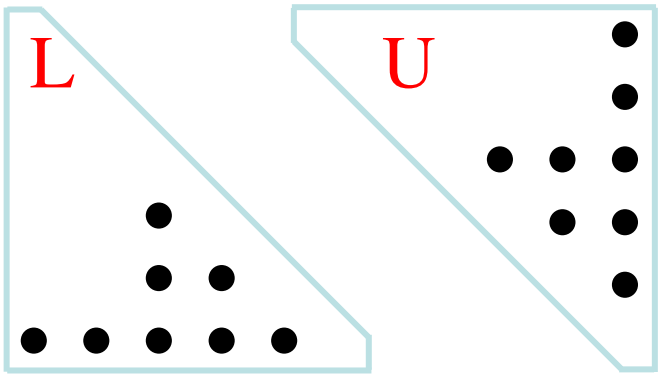
$L \times U = B$ (wedge, low hinge)

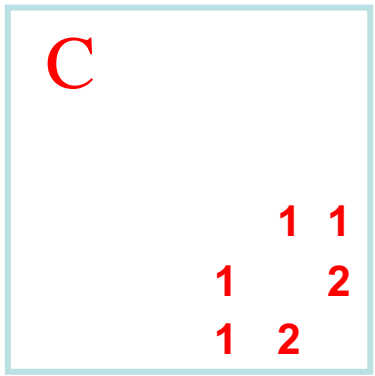
$A \wedge B = C$ (closed wedge)

$sum(C)/2 =$ **4 triangles**







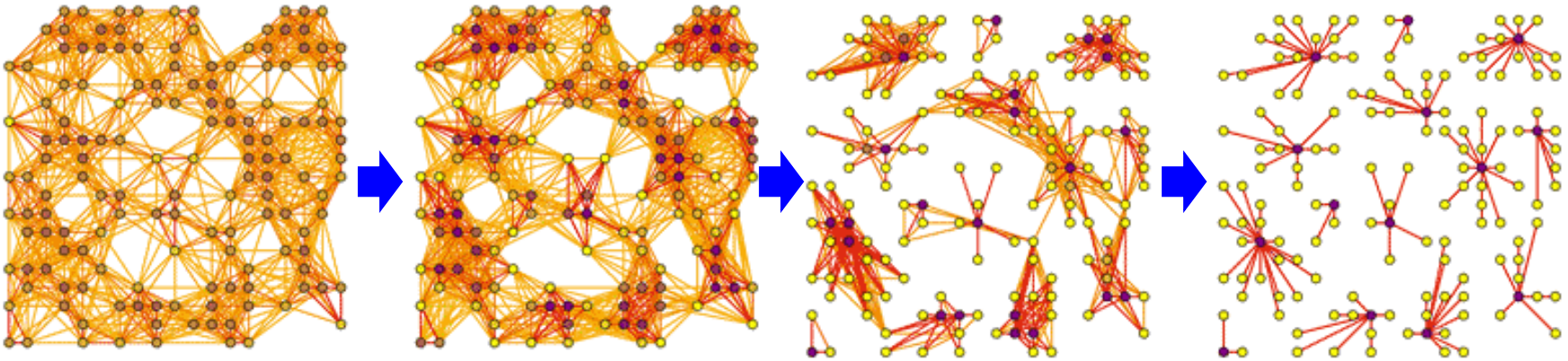


Pattern 2: Sparse matrix times sparse matrix (SpGEMM)

Markov clustering is also multi-source (in fact, all sources) traversal:
It alternates between SpGEMM and element-wise or column-wise pruning

GrB_mxm(C, GrB_NULL, <semiring>, A, A, <desc>)

A: sparse normalized adjacency matrix
C: denser (but still sparse) pre-pruned matrix for next iteration



Initial network | Iteration 1 | Iteration 2 | Iteration 3

At each iteration:
Step 1 (Expansion): Squaring the matrix while pruning (a) small entries, (b) denser columns
Naïve implementation: sparse matrix-matrix product (SpGEMM), followed by column-wise top-K selection and column-wise pruning
Step 2 (Inflation) : taking powers entry-wise

Pattern 3: Sparse matrix times tall-skinny dense matrix (SpMM)

Feature aggregation from neighbors:

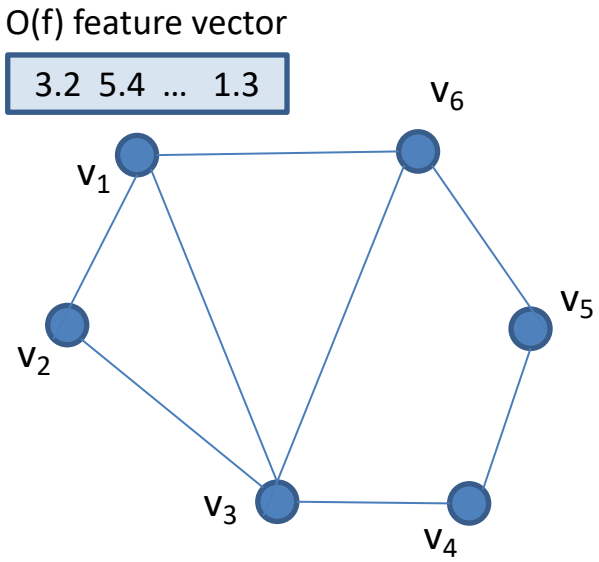
Used in Graph neural networks, graph embedding, etc.

GrB_mxm(W, GrB_NULL, <semiring>, A, H, <desc>)

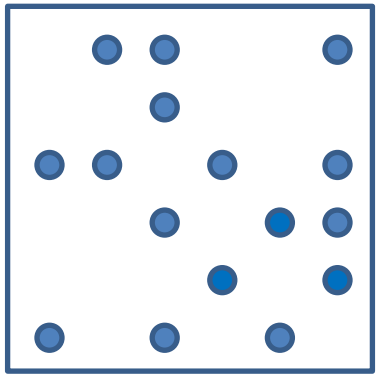
A: sparse adjacency matrix, n-by-n

H: input dense matrix, n-by-f where f << n is the feature dimension

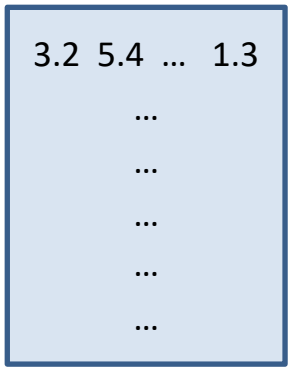
W: output dense matrix, new features



W =

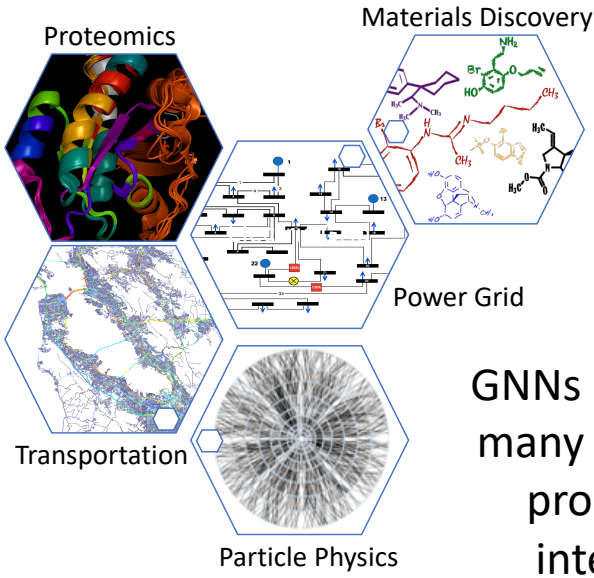


A^T



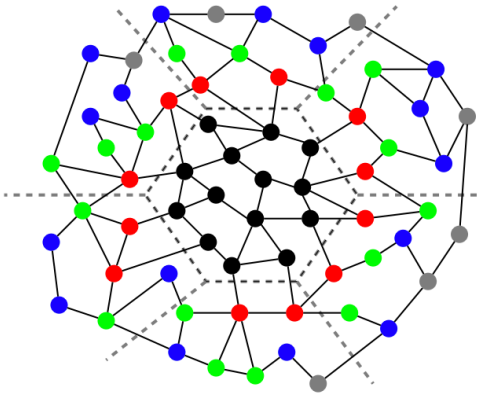
H

Graph Neural Networks (GNNs)



GNNs are finding success in many challenging scientific problems that involve interconnected data.

Interdependencies between samples (nodes of the graph) make stochastic gradient non-trivial without graph sampling



- GNNs are computationally intensive to train. Distributed training need to scale to large GPU/node counts despite challenging sparsity.
- CAGNET (Communication-Avoiding Graph Neural nETworks) full gradient descent to avoid inaccurate (and expensive) graph sampling

https://github.com/PASSIONLab/CAGNET/

GNN Training

- Each node is initialized with a feature vector
  - H^0 has initial feature vector per node $(n\ x\ f)$
- Each node aggregates vectors of its neighbors, applies a weight
- Each layer computes gradients

```
for i = 1 … E
    for l = 1 … L
        Z^l = A^T * H^(l-1) * W^l
        H^l = σ(Z^l)
    ...
    for l = L-1 … 1
        G^l = A * G^(l+1) * (W^(l+1))^T ⊙ σ'(Z^l)
        dH/dW = (H^(l-1))^T * A * G^l
```

$A \in n\ x\ n$

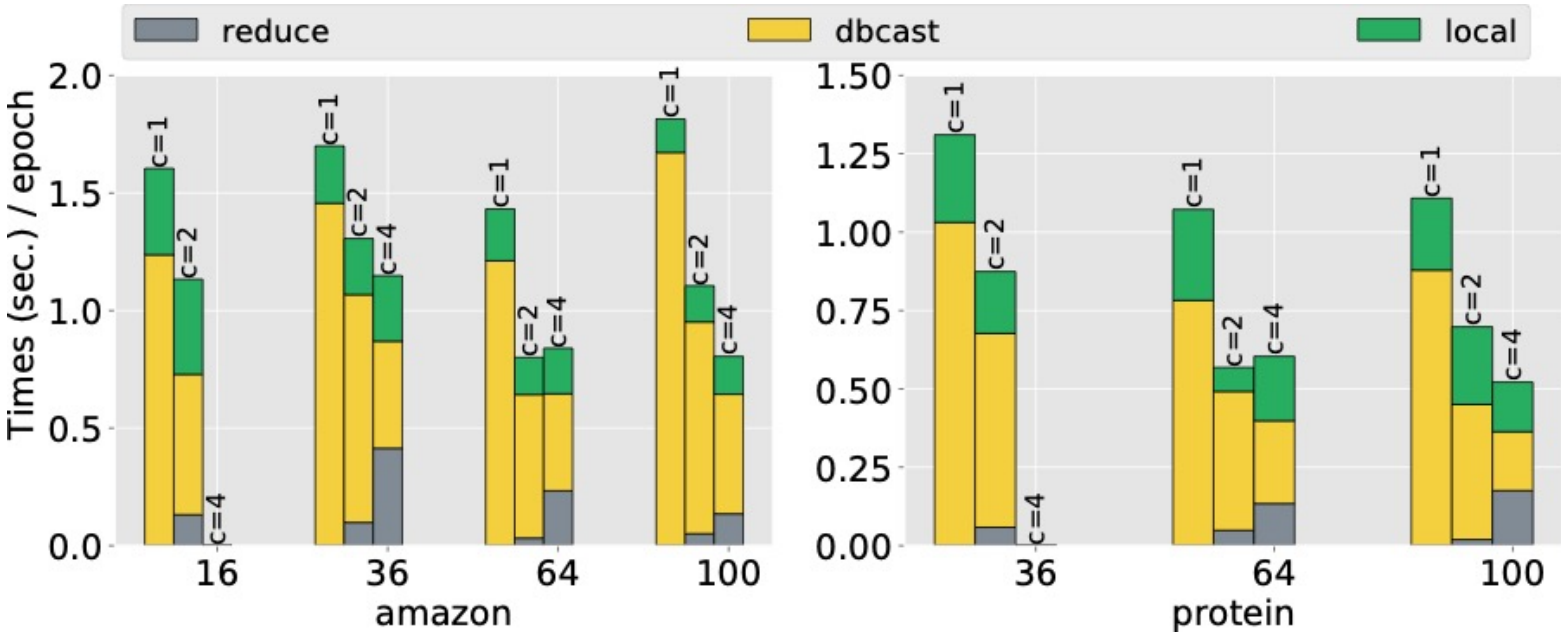
$H^l \in n\ x\ f^l$

$G^l \in n\ x\ f^l$

$W^l \in f^{l-1}\ x\ f^l$

- A is sparse and f << n, so the main workhorse is SpMM (sparse matrix times tall-skinny dense matrix)

Communication avoidance (CA) In GNN Training

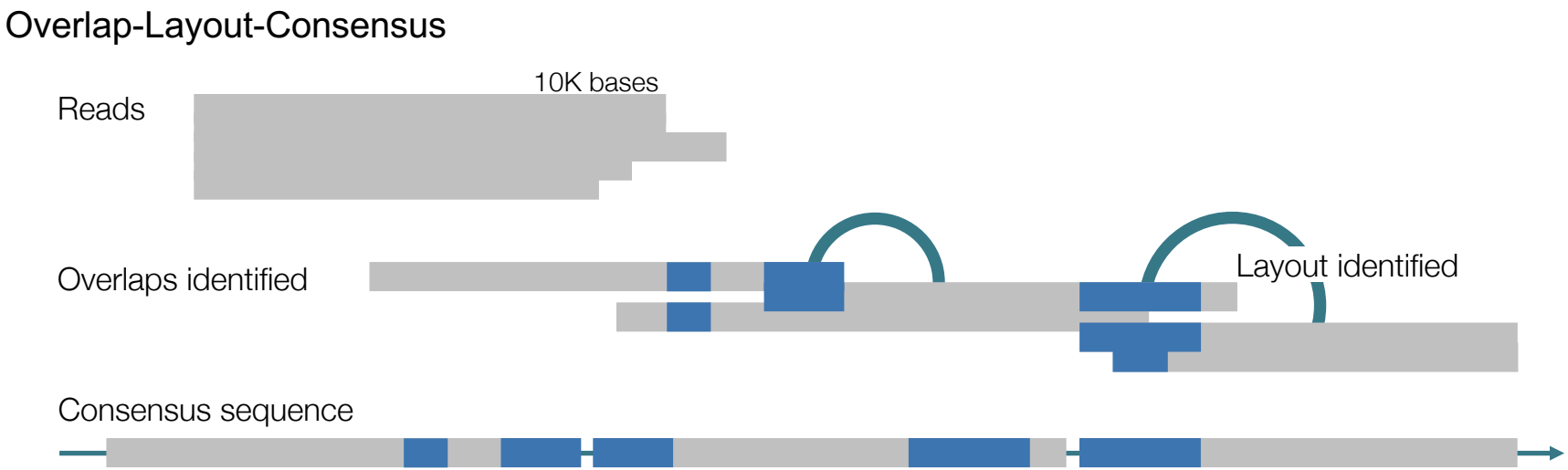


- Scales with both P (GPUs – x axis) and c (replication layers in CA algorithms)
- This is 1 GPU/node on Summit (all GPUs per node results in paper)
- Expect to scale with all GPUs / node with future architectures (e.g. Perlmutter)

Alok Tripathy, Katherine Yelick, Aydın Buluç. Reducing Communication in Graph Neural Network Training. SC'20

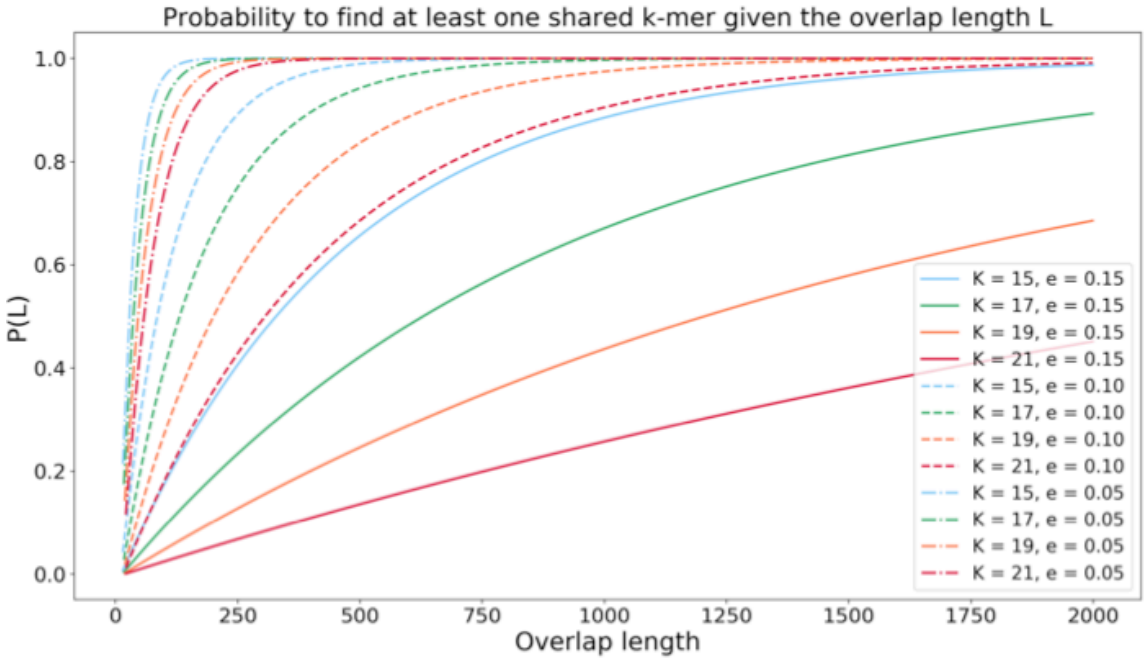
SpGEMM for DNA read overlapping

- **Long reads** from PacBio and Oxford Nanopore have the potential to revolutionize de-novo assembly
- **Overlap-Consensus-Layout** paradigm is more suitable than de Bruijn graph paradigm.
- **Overlapping** is the most computationally expensive step.

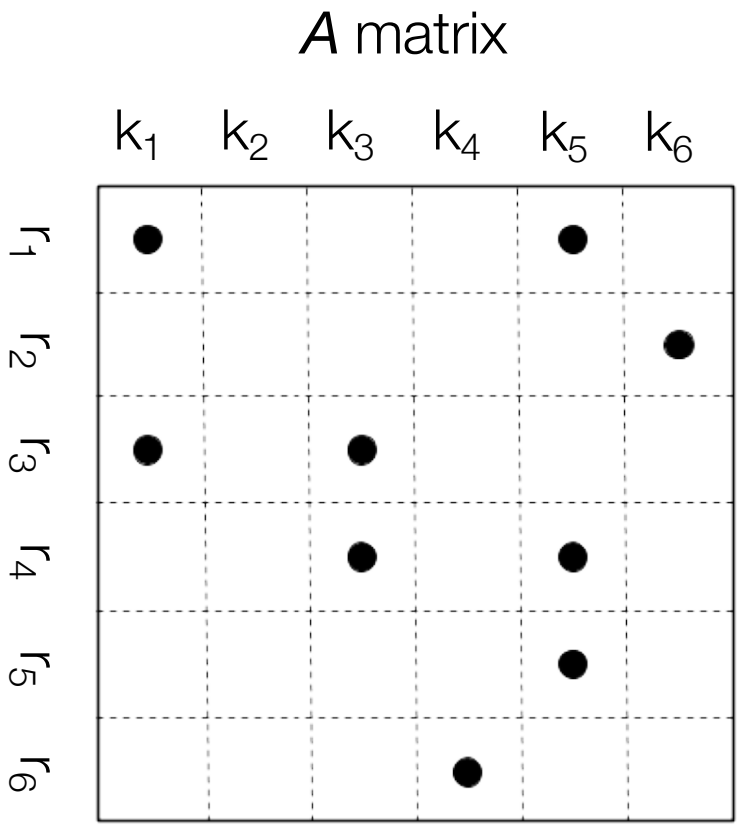


SpGEMM for DNA read overlapping

- We need to quickly determine pairs of reads that are *likely to* overlap, without resorting to $O(n^2)$ comparisons
- If two reads do not share any subsequence of length k (aka a k-mer) for a reasonably short k, then they are unlikely to overlap



SpGEMM for DNA read overlapping



- Suppose you have counted k-mers and only retained *reliable* k-mers
- Now you can generate this ***read-by-kmer*** sparse matrix **A**
- These are all linear time computations so far

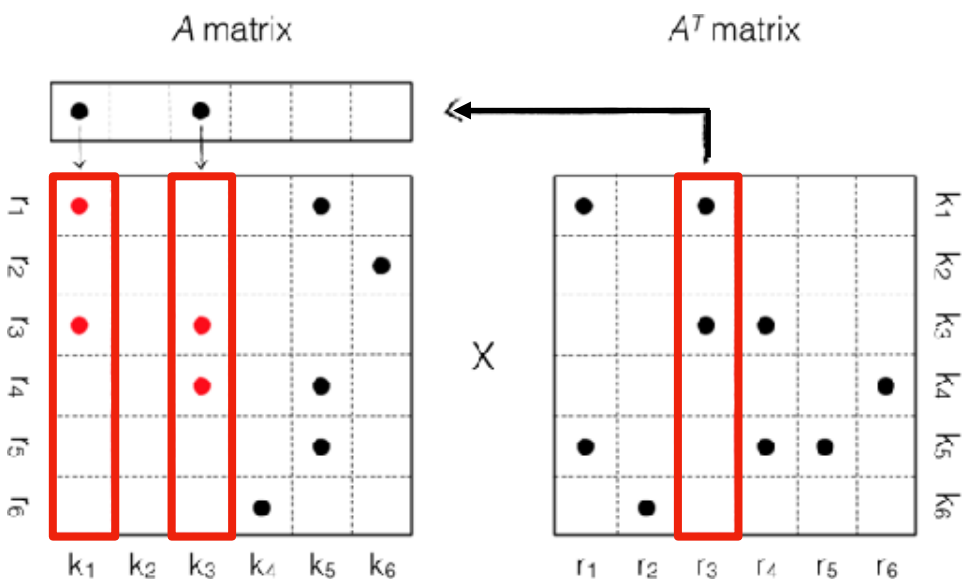
r_i = i^{th} read

k_j = j^{th} reliable ***k***-mer

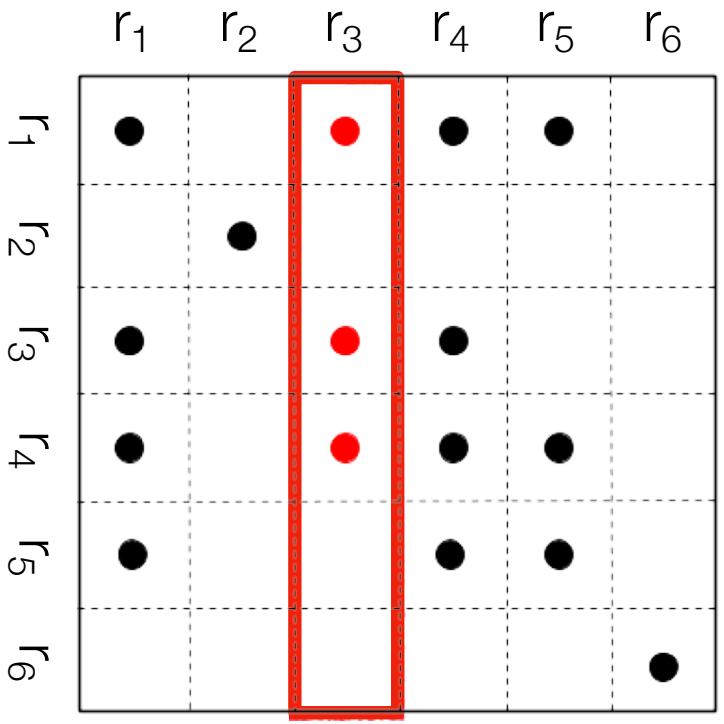
A(i,j) = presence of j^{th} reliable ***k***-mer in i^{th} read, plus its position

Giulia Guidi, Marquita Ellis, Daniel Rokhsar, Kathy Yelick, Aydın Buluç, BELLA: Berkeley Efficient Long-read to Long-Read Overlapper and Aligner, *Biorxiv*.

SpGEMM for DNA read overlapping



Read-by-read overlap matrix: AA^T

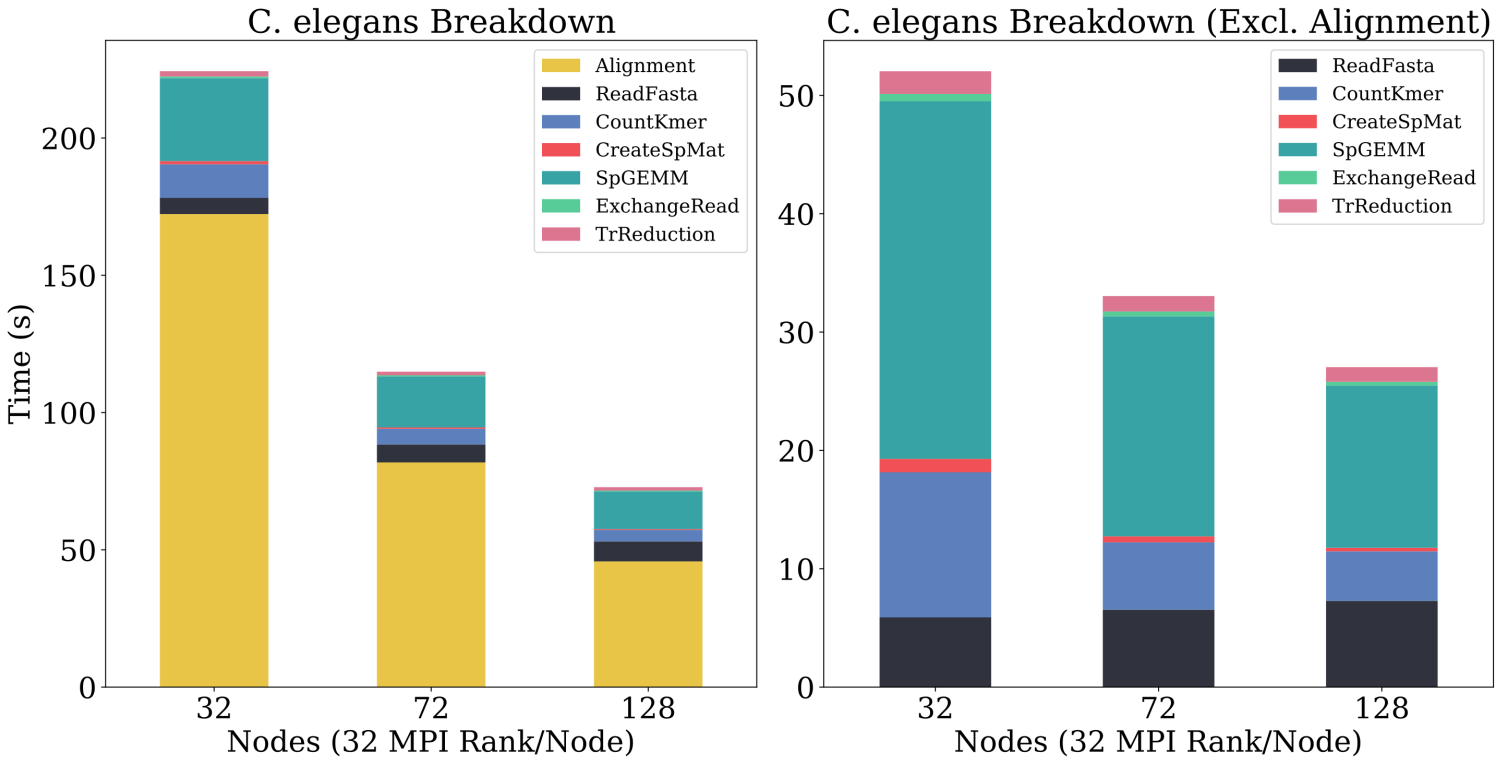


$AA^T(i,j)$ = # shared k-mers between reads i and j, plus their positions in the reads

Use any fast SpGEMM algorithm, as long as it runs on ***arbitrary semirings***

diBELLA.2D performance results

diBELLA.2D: distributed-memory version of BELLA on 2D process grid
Performs *overlap detection* plus *transitive reduction (overlap to string graph)*
https://github.com/PASSIONLab/diBELLA.2D



Giulia Guidi, Oguz Selvitopi, Marquita Ellis, Leonid Oliker, Katherine Yelick, Aydin Buluç. Parallel String Graph Construction and Transitive Reduction for De Novo Genome Assembly. *IPDPS* 2021

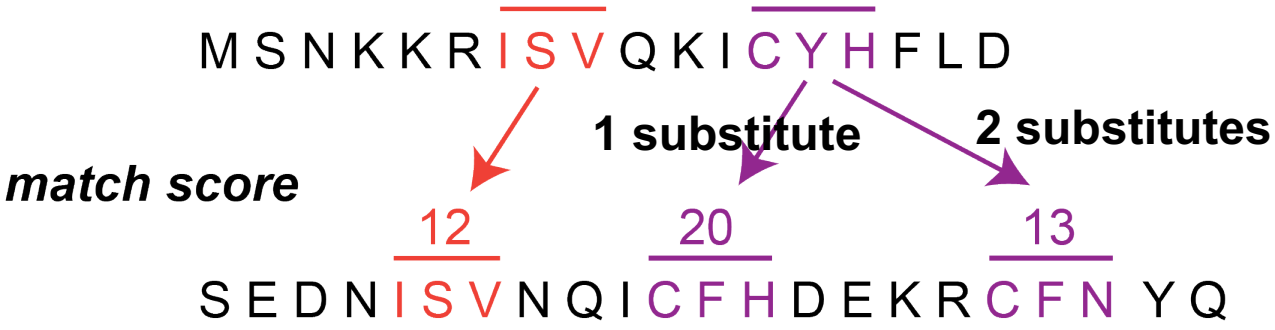
SpGEMM for many-to-many protein alignment

- Idea similar to BELLA, but removing the exact match restriction
- For homology detection, need to catch weaker signal (~30% ANI)
- K-mers with substitutes may be more valuable than exact matches!

BLOSUM 62 scoring matrix

(positive values are shaded)

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4																			
R	-1	5																		
N	-2	0	6																	
D	-2	-2	1	6																
C	0	-3	-3	-3	9															
Q	-1	1	0	0	-3	5														
E	-1	0	0	2	-4	2	5													
G	0	-2	0	-1	-3	-2	-2	6												
H	-2	0	1	-1	-3	0	0	-2	8											
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4



[MS282] Thursday 4:35-4:50 pm: *Distributed Many-to-Many Protein Sequence Alignment using Sparse Matrices* by **Oguz Selvitopi**

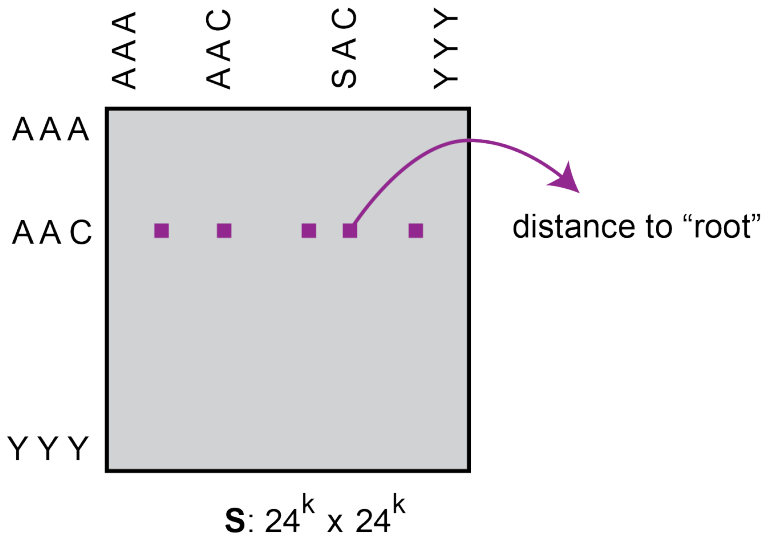
SpGEMM for many-to-many protein alignment

PASTIS (https://github.com/PASSIONLab/PASTIS) does distributed many-to-many protein sequence similarity search using sparse matrices

Introduce new sparse matrix **S**

Contains substitution information

Each entry has **substitution cost**



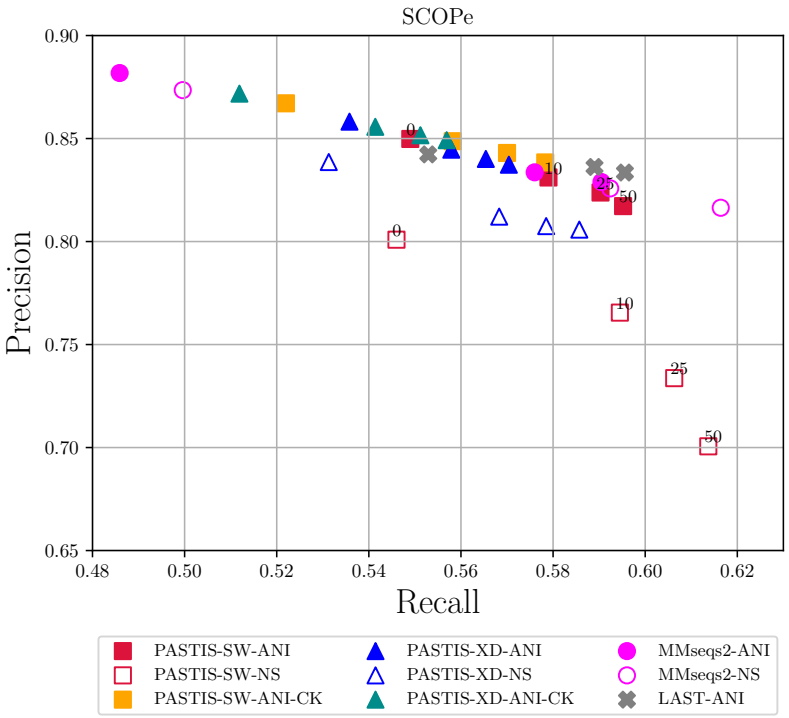
Exact k-mers → $C=AA^T$

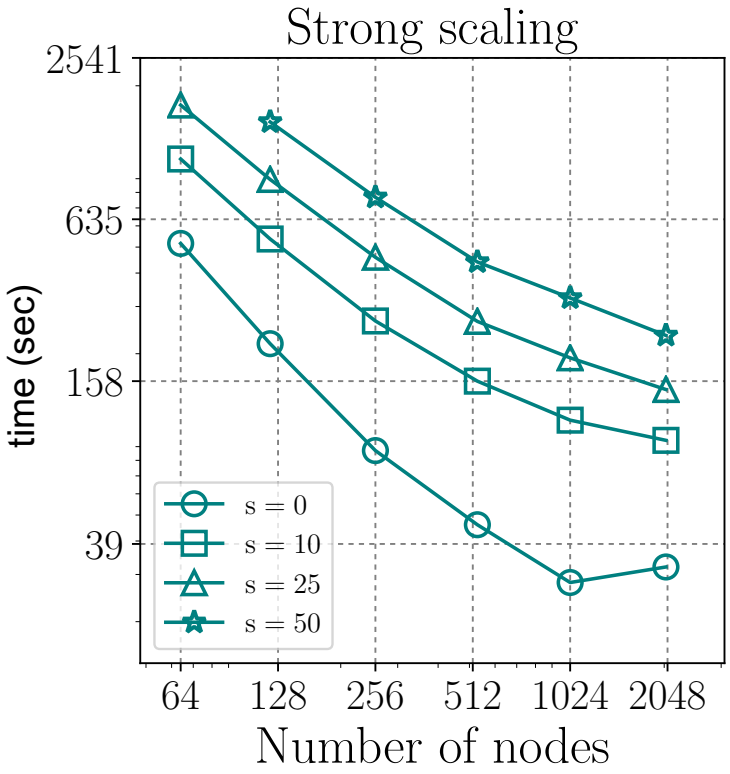
Substitute k-mers → $C=ASA^T$

New semiring

Oguz Selvitopi, Saliya Ekanayake, Giulia Guidi, Georgios Pavlopoulos, Ariful Azad, and Aydın Buluç. Distributed Many-to-Many Protein Sequence Alignment Using Sparse Matrices. SC'20.

PASTIS performance and accuracy





- *Protein similarity search* is the first and most time-consuming step in discovering protein families (proteins evolved from a common ancestor and who likely have the same function)
- *Protein family identification* is a key step in protein function discovery and taxonomic assignment of newly sequenced organisms

Acknowledgments

Ariful Azad, Ben Brock, Tim Davis, Saliya Ekanayake, Marquita Ellis, John Gilbert, Giulia Guidi, Jeremy Kepner, Nikos Krypides, Tim Mattson, Scott McMillan, Jose Moreira, Lenny Oliker, John Owens, Georgios Pavlopoulos, Dan Rokhsar, Oguz Selvitopi, Yu-Hang Tang, Alok Tripathy, Carl Yang, Kathy Yelick.

- My Research Team: http://passion.lbl.gov
- Our Youtube Channel: http://shorturl.at/lpFRY
- The GraphBLAS Forum: http://graphblas.org