SlimSell: A Vectorizable Graph Representation for Breadth-First Search

MACIEJ BESTA, FLORIAN MARENDING, EDGAR SOLOMONIK, TORSTEN HOEFLER

LARGE-SCALE IRREGULAR GRAPH PROCESSING

- Becoming more important [1]

[1] A. Lumsdaine et al. Challenges in Parallel Graph Processing. Parallel Processing Let. 2007.

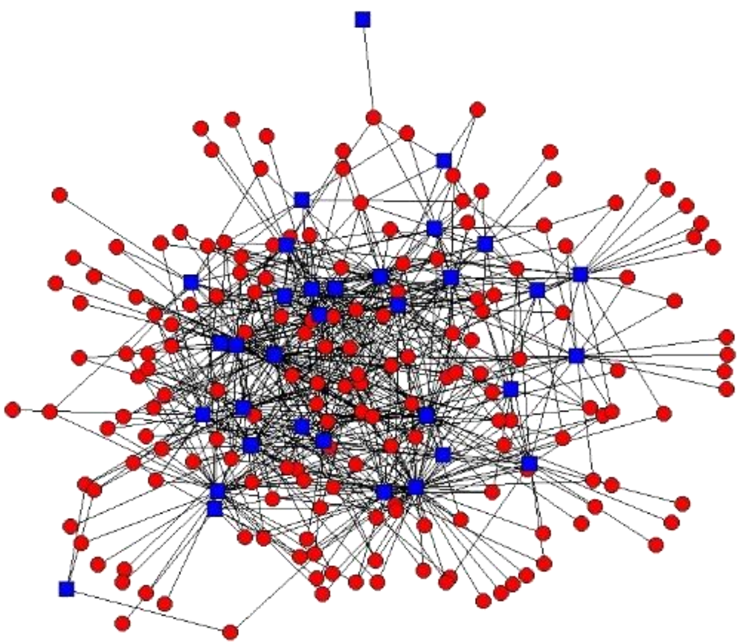
LARGE-SCALE IRREGULAR GRAPH PROCESSING

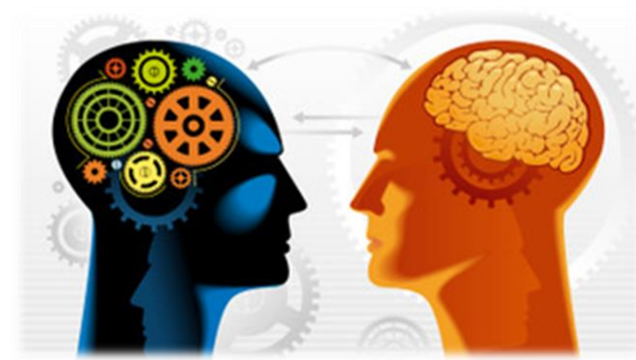
- Becoming more important [1]
  - Machine learning

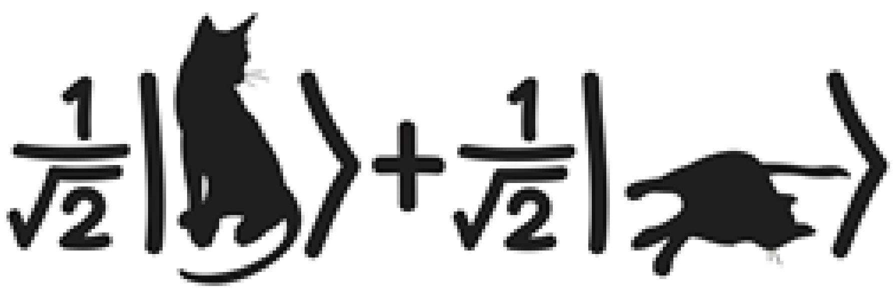
[1] A. Lumsdaine et al. Challenges in Parallel Graph Processing. Parallel Processing Let. 2007.

Large-Scale Irregular Graph Processing

- Becoming more important [1]
  - Machine learning
  - Computational science



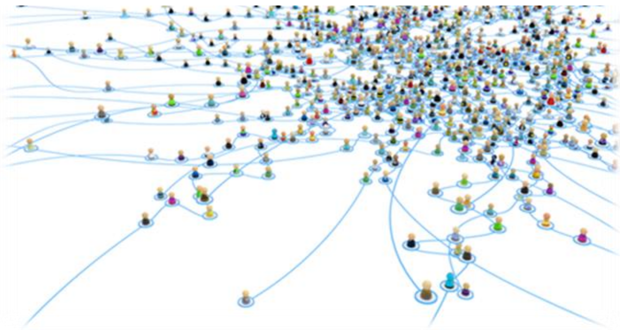


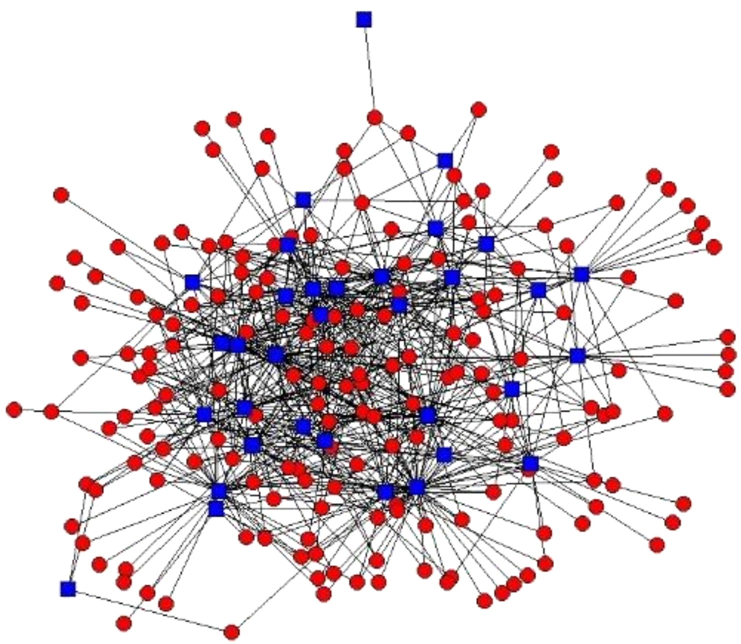


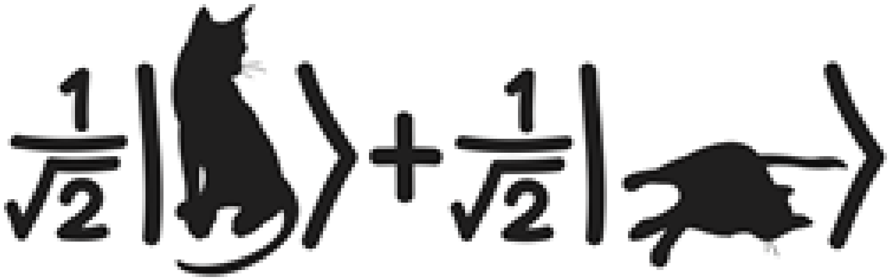
[1] A. Lumsdaine et al. Challenges in Parallel Graph Processing. Parallel Processing Let. 2007.

Large-Scale Irregular Graph Processing

- Becoming more important [1]
  - Machine learning
  - Computational science
  - Social network analysis





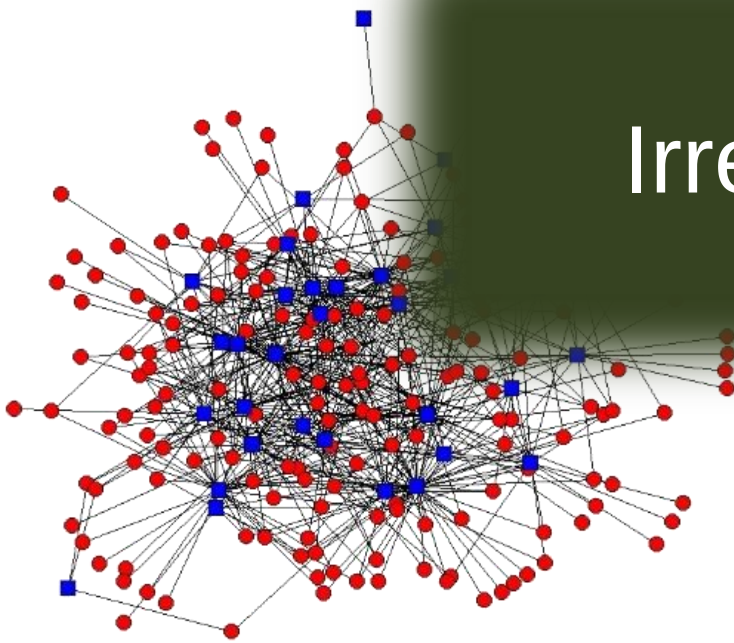


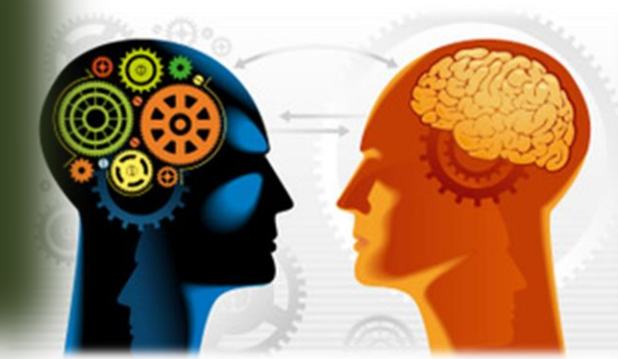
[1] A. Lumsdaine et al. Challenges in Parallel Graph Processing. Parallel Processing Let. 2007.

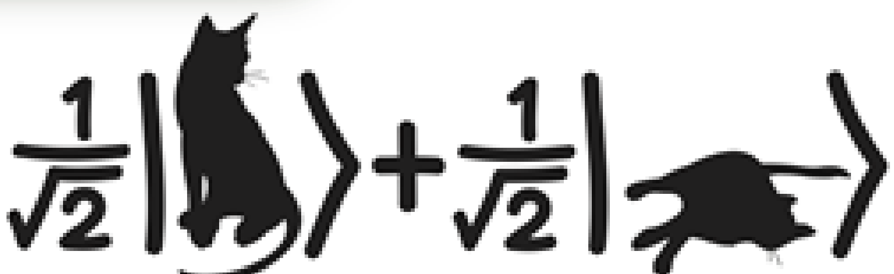
Large-Scale Irregular Graph Processing

- Becoming more important [1]
  - Machine learning
  - Computational science
  - Social network analysis

Irregular







[1] A. Lumsdaine et al. Challenges in Parallel Graph Processing. Parallel Processing Let. 2007.

VECTORIZATION

VECTORIZATION

- Deployed in various hardware

VECTORIZATION

- Deployed in various hardware
- Becoming more popular

Vectorization

- Deployed in various hardware
- Becoming more popular

AVX

$C = 8$ (SIMD width)

VECTORIZATION

- Deployed in various hardware
- Becoming more popular

AVX

$C = 16$ (SIMD width)

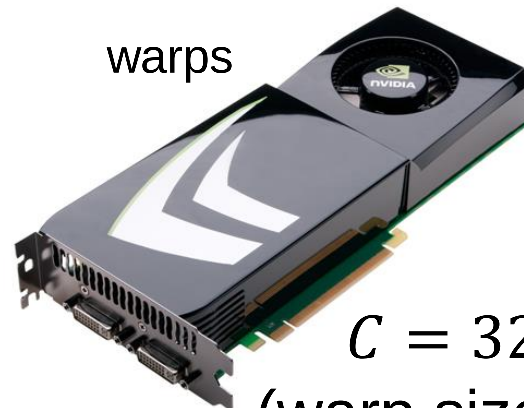
AVX

$C = 8$ (SIMD width)

VECTORIZATION

- Deployed in various hardware
- Becoming more popular

warps



$C = 32$
(warp size)

AVX

AVX

$C = 16$ (SIMD width)

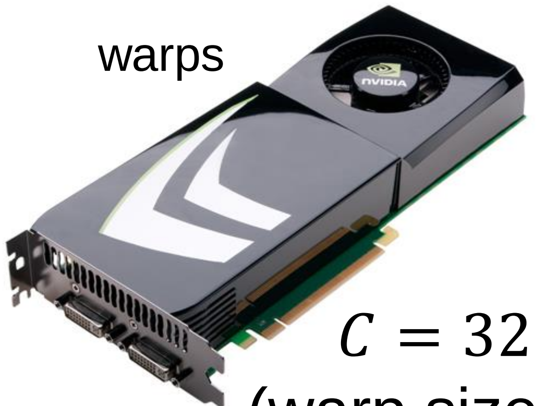
$C = 8$ (SIMD width)

C: „Chunk" size: SIMD width (CPUs, KNLs), warp size (GPUs)

VECTORIZATION

- Deployed in various hardware
- Becoming more popular
- Offers a lot of „regular" compute power

warps



$C = 32$
(warp size)

AVX

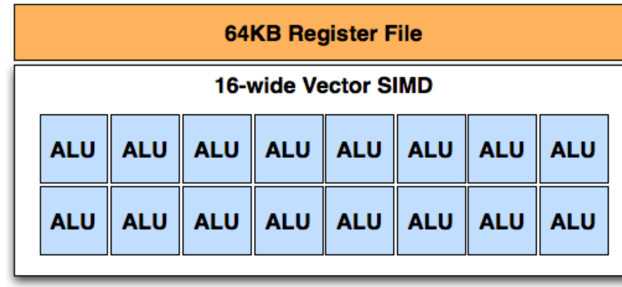
$C = 16$ (SIMD width)

AVX

$C = 8$ (SIMD width)

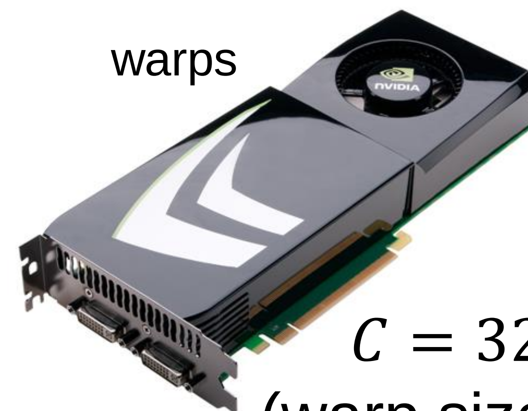
C: **„Chunk" size: SIMD width (CPUs, KNLs), warp size (GPUs)**

VECTORIZATION



- Deployed in various hardware
- Becoming more popular
- Offers a lot of „regular” compute power

warps



$C = 32$ (warp size)

AVX

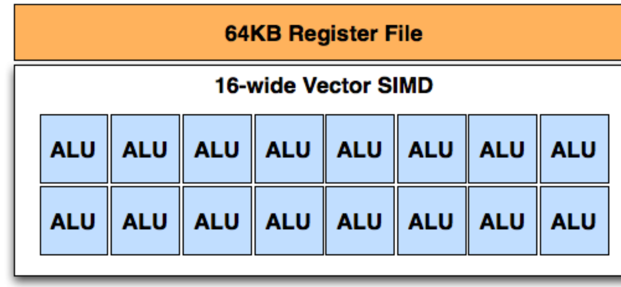
$C = 16$ (SIMD width)

AVX

$C = 8$ (SIMD width)

C: „Chunk” size: SIMD width (CPUs, KNLs), warp size (GPUs)

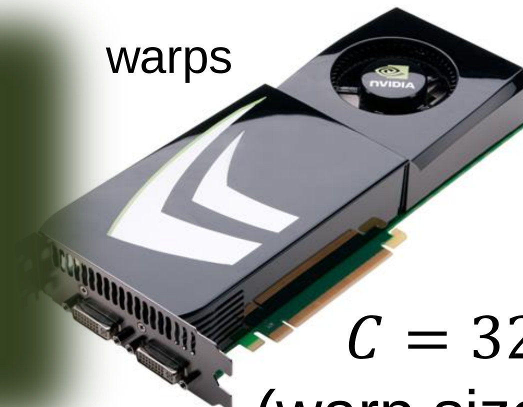
VECTORIZATION



- Deployed in various hardware
- Becoming more popular
- Offers a lot of „regular” compute power

Regular

warps



$C = 32$
(warp size)

AVX

$C = 16$ (SIMD width)

AVX

$C = 8$ (SIMD width)

C: „Chunk” size: SIMD width (CPUs, KNLs), warp size (GPUs)

VECTORIZATION

- Deployed in various hardware
- Becoming more popular
- Offers a lot of „regular” compute power

Regular

warps

$C = 32$
(warp size)

AVX

$C = 16$ (SIMD width)

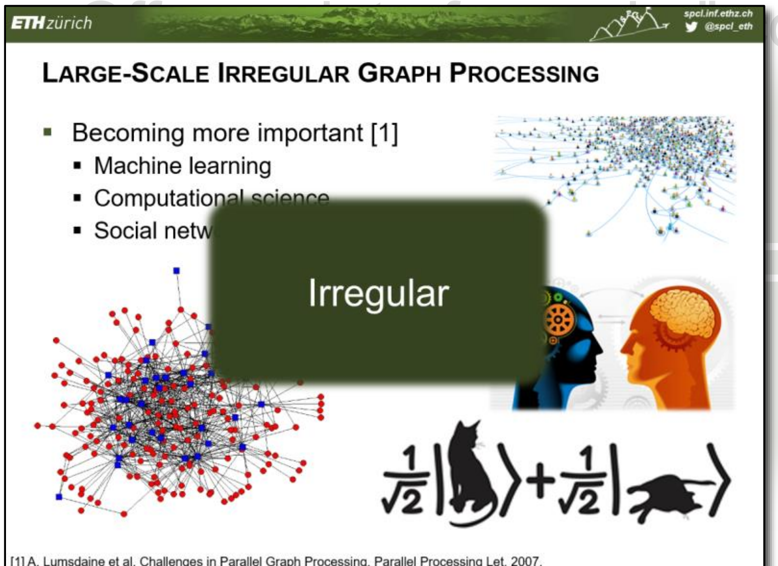
AVX

$C = 8$ (SIMD width)

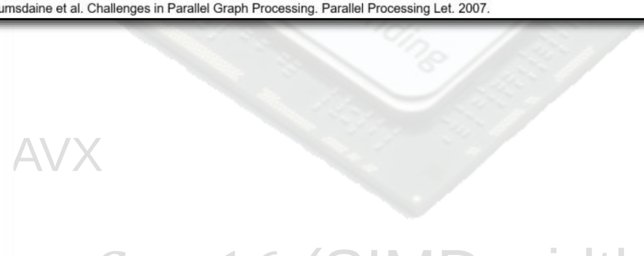
C: „Chunk” size: SIMD width (CPUs, KNLs), warp size (GPUs)

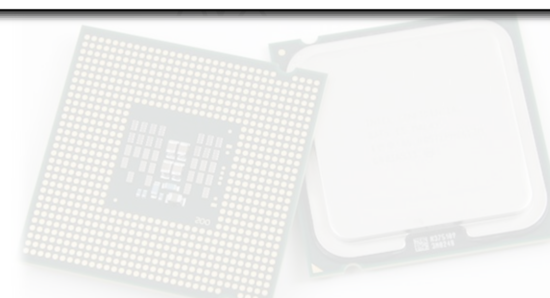
VECTORIZATION

- Deployed in various hardware
- Becoming more popular
- ompute



+





AVX

$C = 16$ (SIMD width)

$C = 8$ (SIMD width)

C: „Chunk" size: SIMD width (CPUs, KNLs), warp size (GPUs)

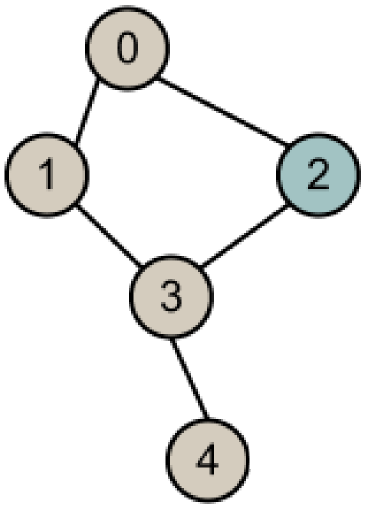
Breadth-First Search
Traditional Formulation

BREADTH-FIRST SEARCH
TRADITIONAL FORMULATION

- BFS is based on primitives such as queues

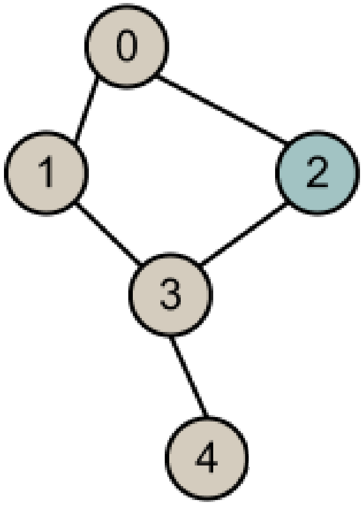
Breadth-First Search
Traditional Formulation

- BFS is based on primitives such as queues



Breadth-First Search
Traditional Formulation

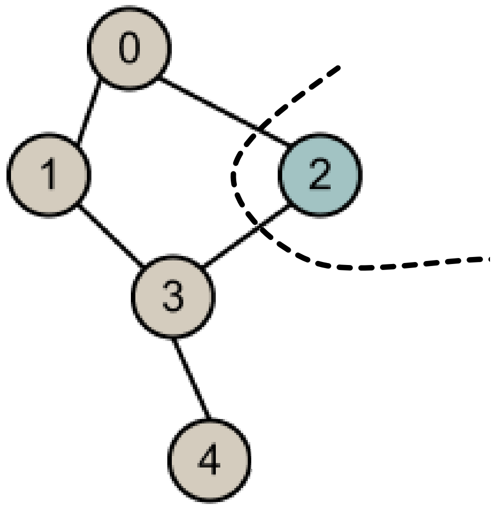
- BFS is based on primitives such as queues



1) F = {}

BREADTH-FIRST SEARCH
TRADITIONAL FORMULATION

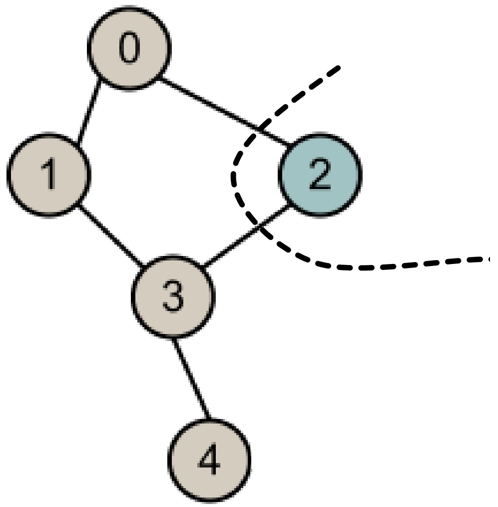
- BFS is based on primitives such as queues



1) F = {}

Breadth-First Search
Traditional Formulation

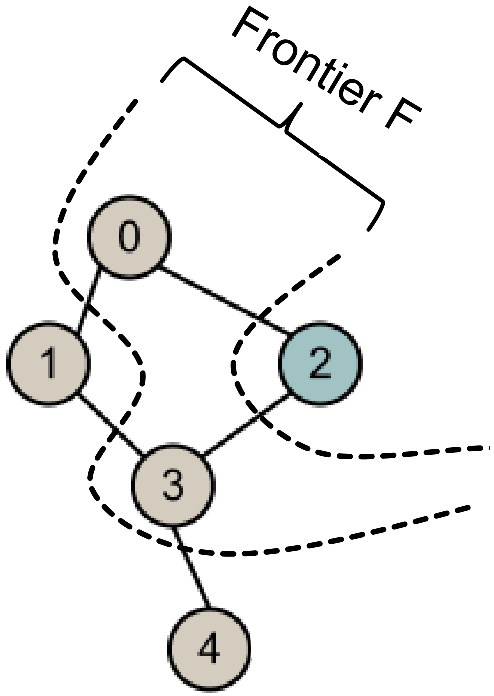
- BFS is based on primitives such as queues



1) F = {}
2) F = {2}

Breadth-First Search
Traditional Formulation

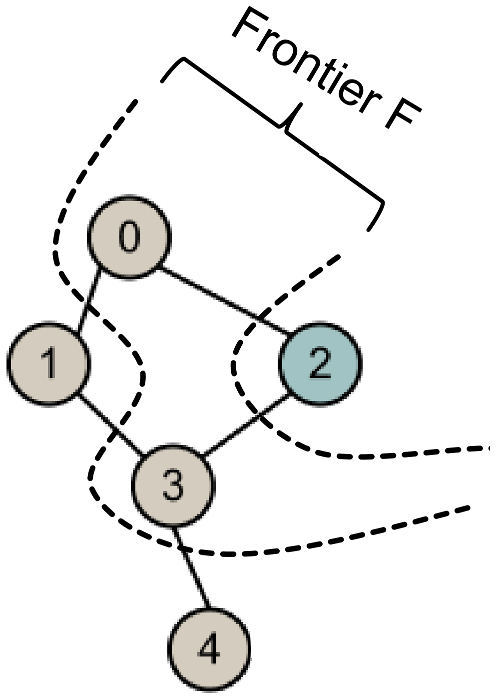
- BFS is based on primitives such as queues



1) F = {}
2) F = {2}

Breadth-First Search
Traditional Formulation

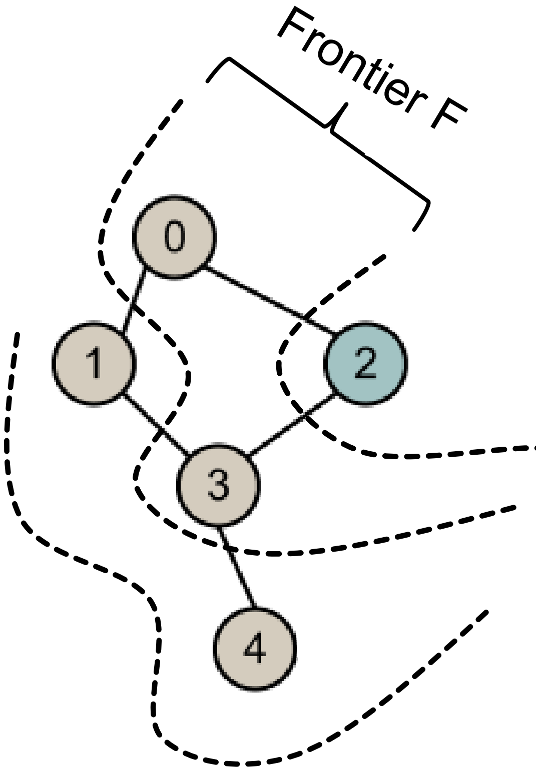
- BFS is based on primitives such as queues



1) F = {}
2) F = {2}
3) F = {0,3}

Breadth-First Search
Traditional Formulation

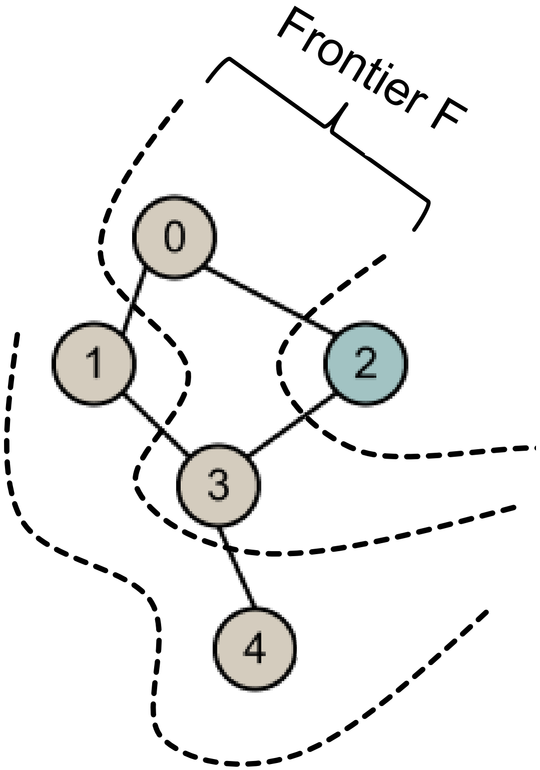
- BFS is based on primitives such as queues



1) F = {}
2) F = {2}
3) F = {0,3}

Breadth-First Search
Traditional Formulation

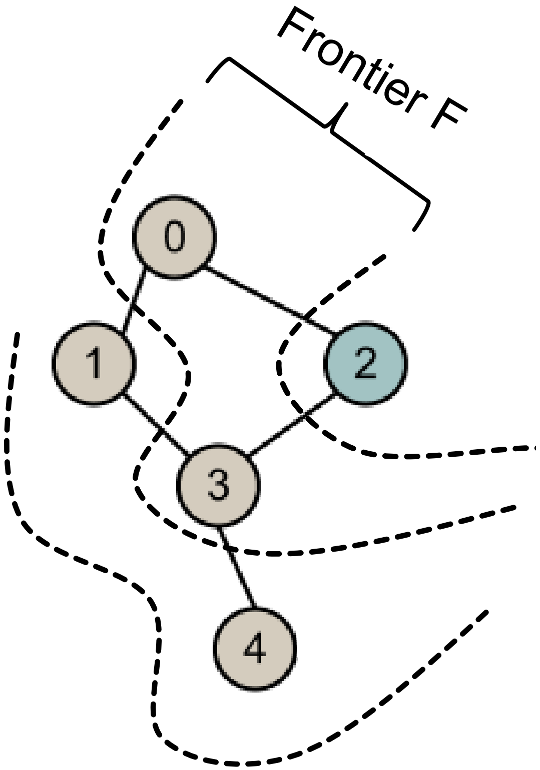
- BFS is based on primitives such as queues



1) F = {}
2) F = {2}
3) F = {0,3}
4) F = {1,4}

Breadth-First Search
Traditional Formulation

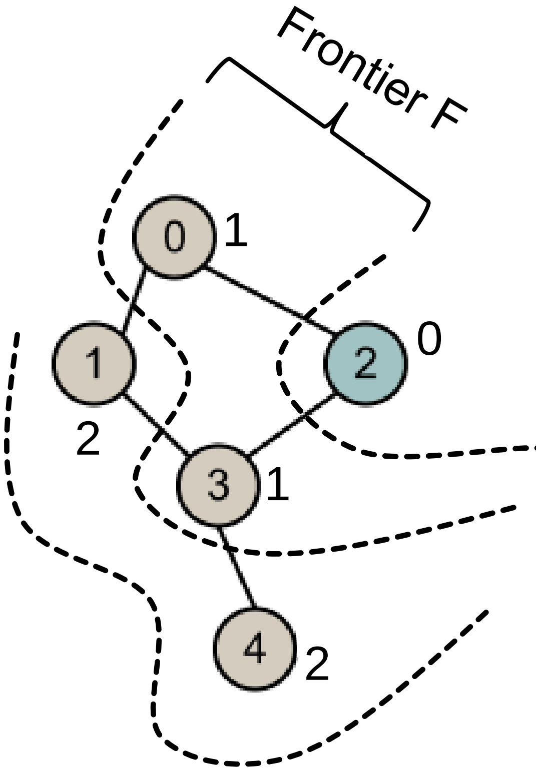
- BFS is based on primitives such as queues



1) F = {}
2) F = {2}
3) F = {0,3}
4) F = {1,4}

Breadth-First Search
Traditional Formulation

- BFS is based on primitives such as queues

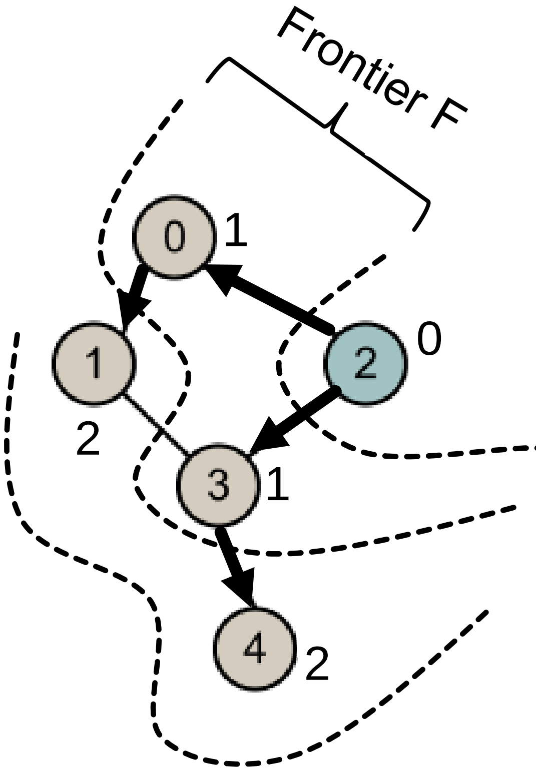


1) F = {}
2) F = {2}
3) F = {0,3}
4) F = {1,4}

Distances from the root

Breadth-First Search
Traditional Formulation

- BFS is based on primitives such as queues

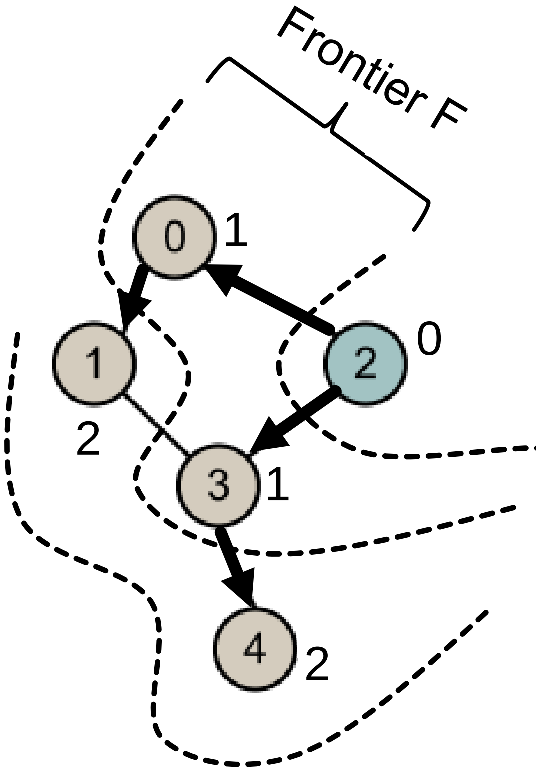


1) F = {}
2) F = {2}
3) F = {0,3}
4) F = {1,4}

Distances from the root

BREADTH-FIRST SEARCH
TRADITIONAL FORMULATION

- BFS is based on primitives such as queues



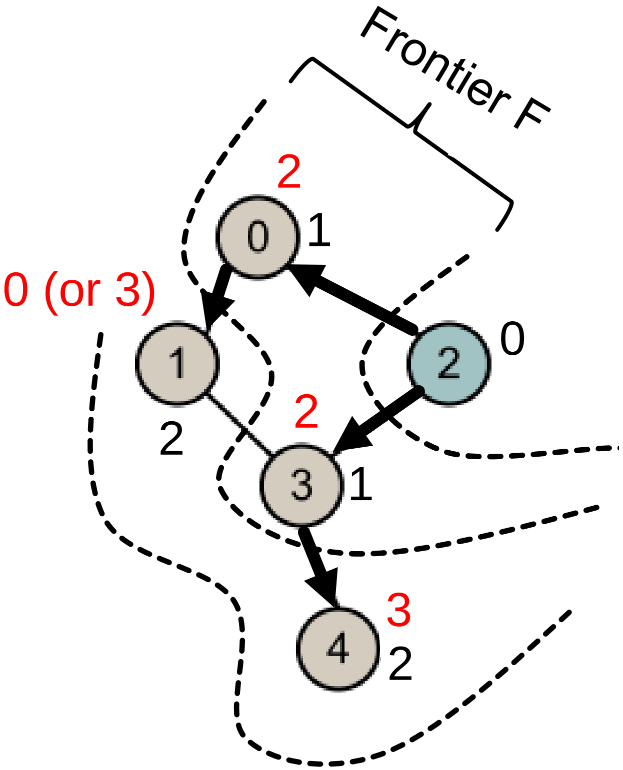
1) F = {}
2) F = {2}
3) F = {0,3}
4) F = {1,4}

Distances from the root

Parents (predecessors) in the traversal tree

BREADTH-FIRST SEARCH
TRADITIONAL FORMULATION

- BFS is based on primitives such as queues



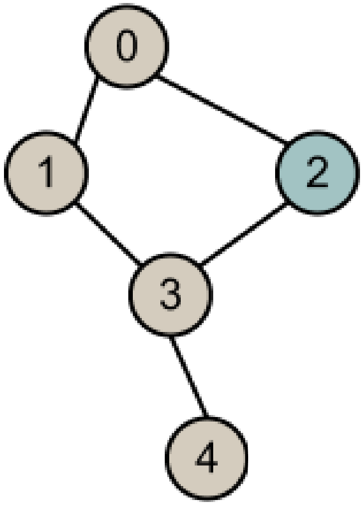
1) F = {}
2) F = {2}
3) F = {0,3}
4) F = {1,4}

Distances from the root

Parents (predecessors) in the traversal tree

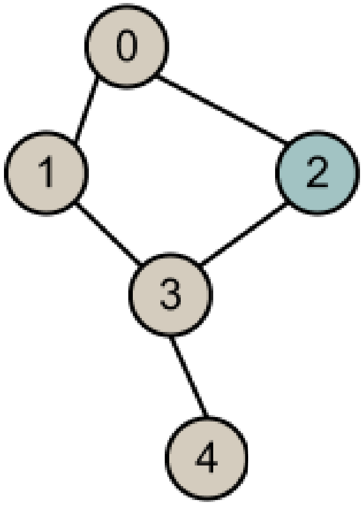
Breadth-First Search

Algebraic Formulation



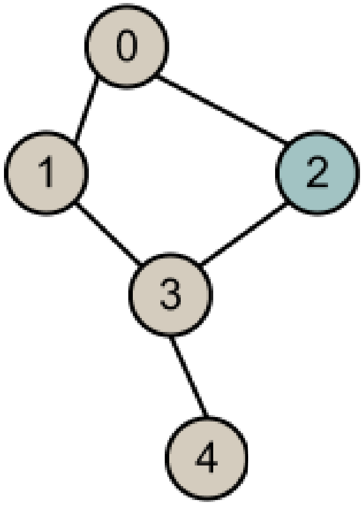
Breadth-First Search
Algebraic Formulation

- BFS is a series of matrix-vector products



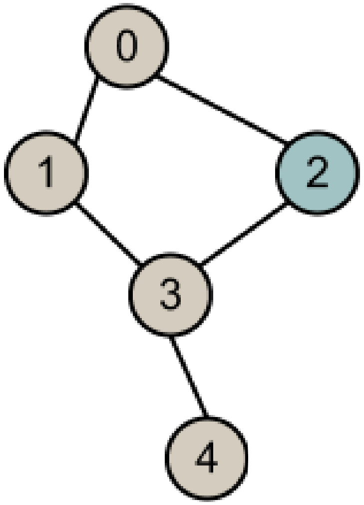
Breadth-First Search
Algebraic Formulation

- BFS is a series of matrix-vector products
- Graph is modeled by an adjacency matrix



Breadth-First Search
Algebraic Formulation

- BFS is a series of matrix-vector products
- Graph is modeled by an adjacency matrix

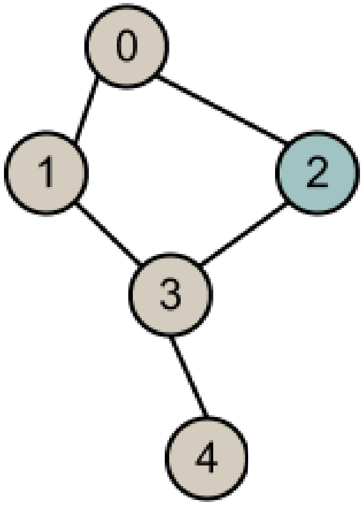


Adjacency Matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Breadth-First Search
Algebraic Formulation

- BFS is a series of matrix-vector products
- Graph is modeled by an adjacency matrix
- Multiplication is done over a semiring

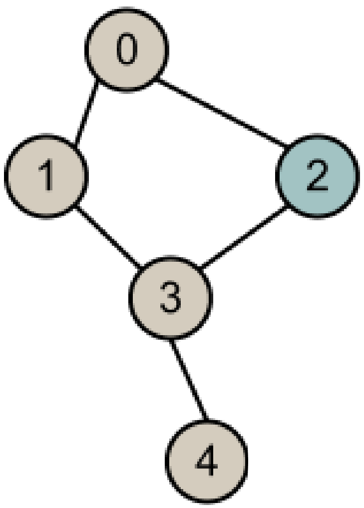


Adjacency Matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Breadth-First Search
Algebraic Formulation

- BFS is a series of matrix-vector products
- Graph is modeled by an adjacency matrix
- Multiplication is done over a semiring



Adjacency Matrix:

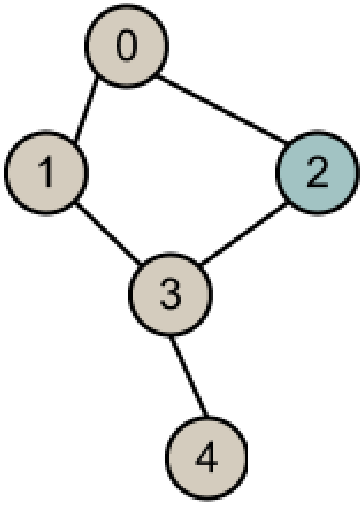
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Semiring:

$$(\mathbb{R}, op_1, op_2, el_1, el_2)$$

Breadth-First Search
Algebraic Formulation

- BFS is a series of matrix-vector products
- Graph is modeled by an adjacency matrix
- Multiplication is done over a semiring



Adjacency Matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Semiring:

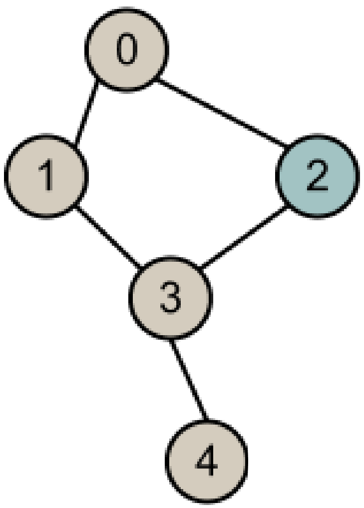
$$(\mathbb{R}, op_1, op_2, el_1, el_2)$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

Breadth-First Search

Algebraic Formulation

- BFS is a series of matrix-vector products
- Graph is modeled by an adjacency matrix
- Multiplication is done over a semiring



Adjacency Matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Semiring:

$$(\mathbb{R}\,, op_1, op_2, el_1, el_2)$$

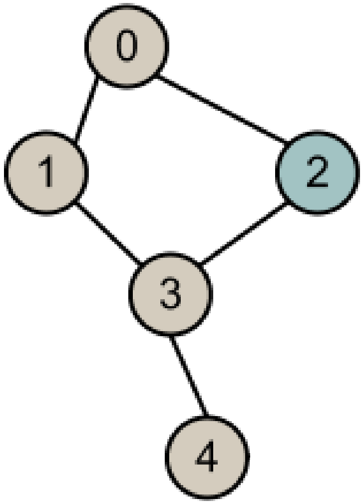
$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

Breadth-First Search
Algebraic Formulation

Tropical Semiring

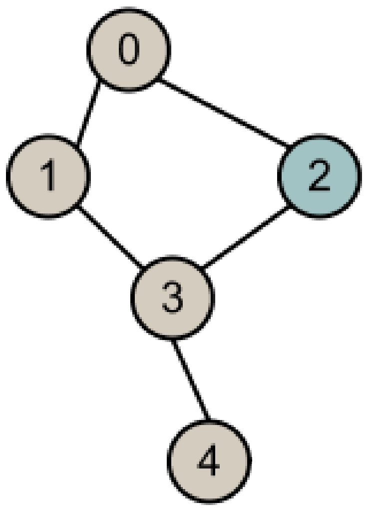
$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$



Breadth-First Search
Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$



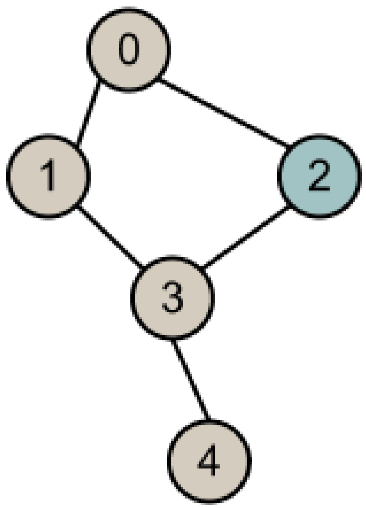
$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Breadth-First Search
Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Breadth-First Search
Algebraic Formulation

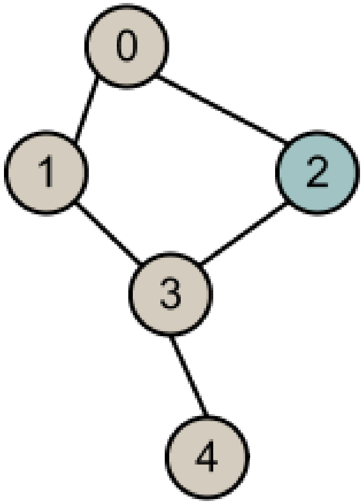
Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Usually stored using a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$



Breadth-First Search
Algebraic Formulation

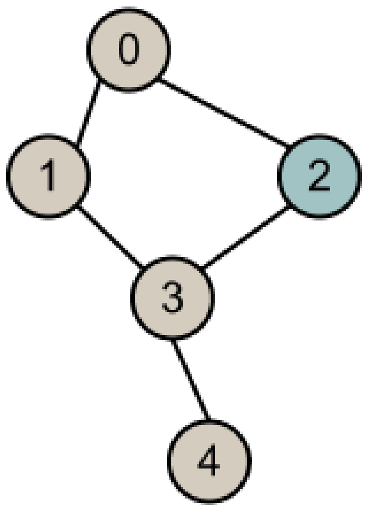
Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Usually stored using a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix} \qquad f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

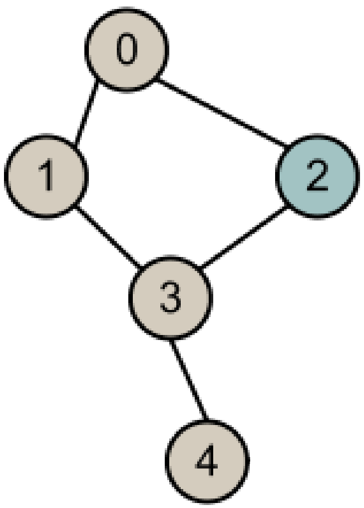


Breadth-First Search
Algebraic Formulation

Tropical Semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Usually stored using a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Stored with a dense or a sparse format

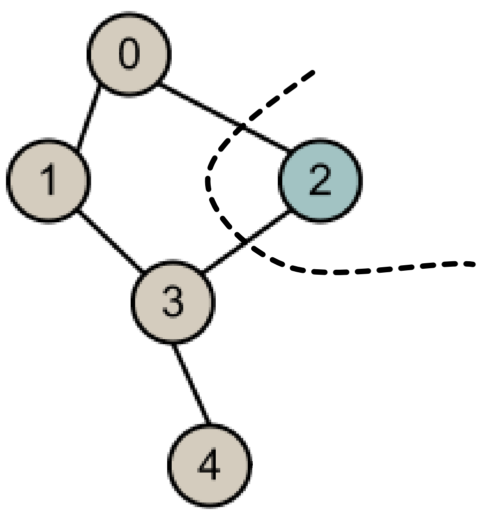
$$f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

Breadth-First Search
Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Usually stored using a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Stored with a dense or a sparse format

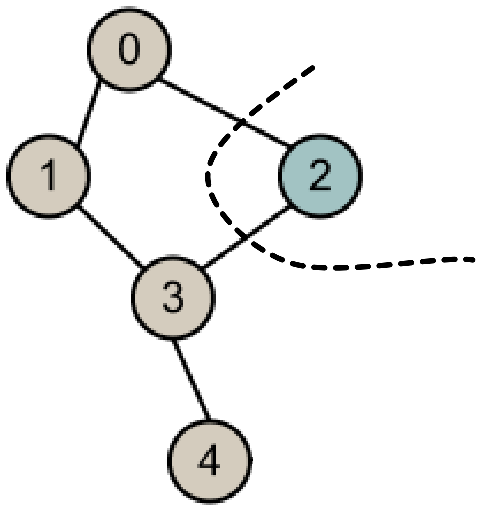
$$f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

Breadth-First Search
Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Usually stored using a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Stored with a dense or a sparse format

$$f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

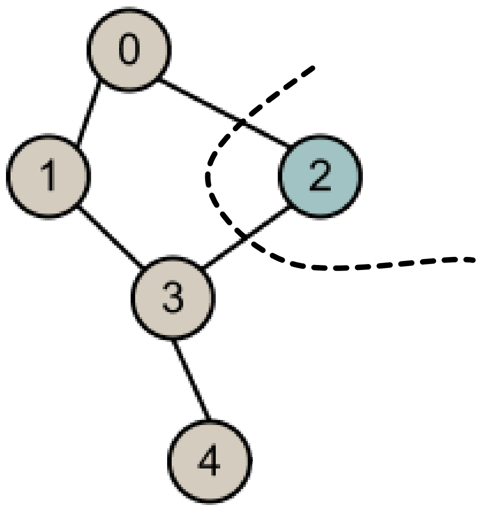
$$f_1 = A'^T \otimes_T f_0 = \begin{pmatrix} \\ \\ \\ \end{pmatrix}$$

Breadth-First Search
Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Usually stored using a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Stored with a dense or a sparse format

$$f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

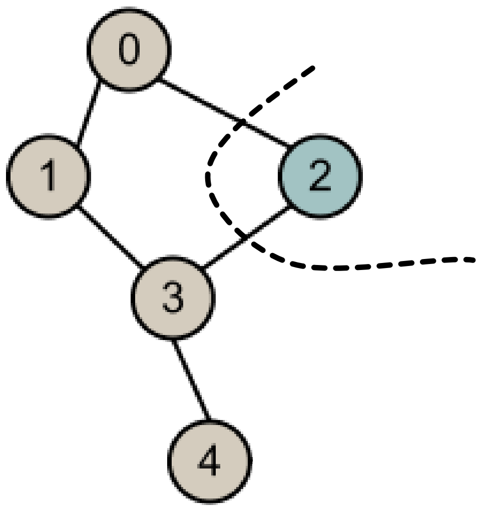
$$f_1 = A'^T \otimes_T f_0 = \begin{pmatrix} \\ \\ \\ \end{pmatrix}$$

Breadth-First Search
Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Usually stored using a sparse format

Stored with a dense or a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix} \quad f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

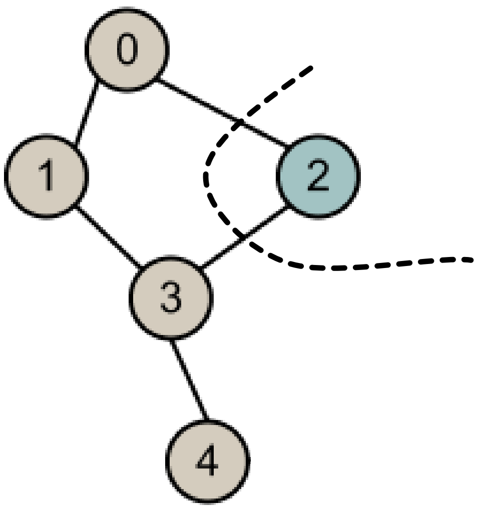
$$f_1 = A'^T \otimes_T f_0 = \begin{pmatrix} 1 \\ \\ \\ \\ \end{pmatrix}$$

Breadth-First Search
Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Usually stored using a sparse format

Stored with a dense or a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix} \qquad f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

$$f_1 = A'^T \otimes_T f_0 = \begin{pmatrix} 1 \\ \infty \\ 0 \\ 1 \\ \infty \end{pmatrix}$$

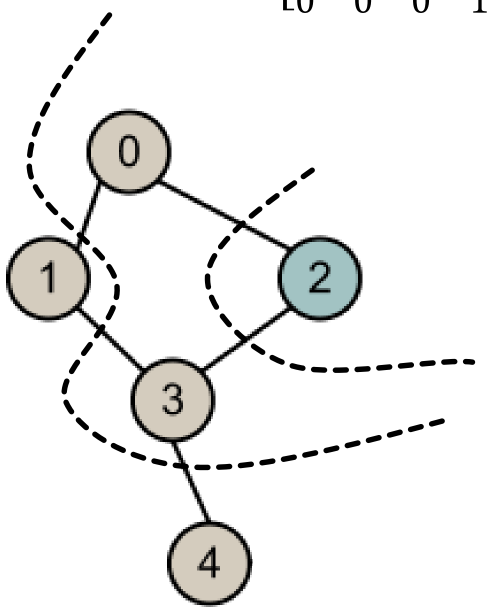
Breadth-First Search

Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Usually stored using a sparse format

Stored with a dense or a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix} \qquad f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

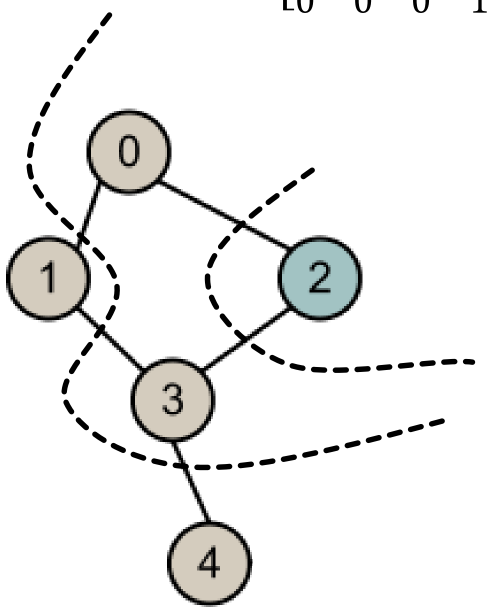
$$f_1 = A'^T \otimes_T f_0 = \begin{pmatrix} 1 \\ \infty \\ 0 \\ 1 \\ \infty \end{pmatrix}$$

Breadth-First Search
Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Usually stored using a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Stored with a dense or a sparse format

$$f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

$$f_1 = A'^T \otimes_T f_0 = \begin{pmatrix} 1 \\ \infty \\ 0 \\ 1 \\ \infty \end{pmatrix}$$

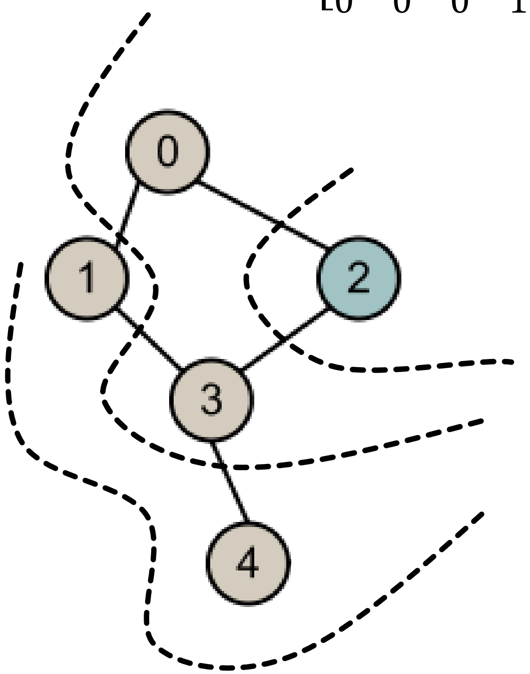
$$f_2 = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

Breadth-First Search
Algebraic Formulation

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Usually stored using a sparse format

Stored with a dense or a sparse format

$$A' = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & \infty & 1 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ \infty & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix} \quad f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

$$f_1 = A'^T \otimes_T f_0 = \begin{pmatrix} 1 \\ \infty \\ 0 \\ 1 \\ \infty \end{pmatrix} \quad f_2 = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

BREADTH-FIRST SEARCH
ALGEBRAIC FORMULATION

Tropical Semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

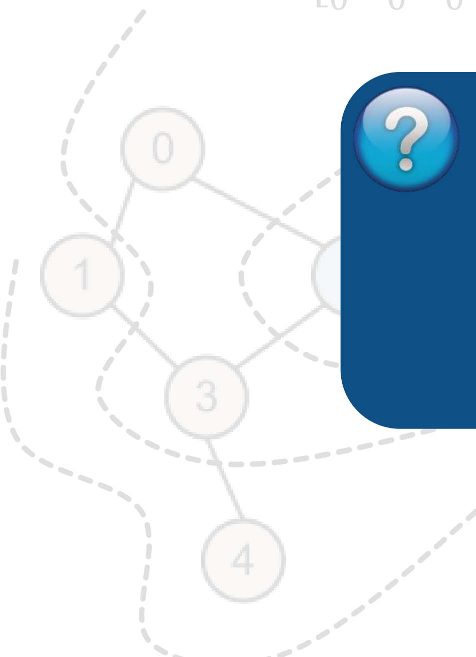
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Usually stored using a sparse format

$$\begin{bmatrix} 0 & 1 & 1 & \infty & \infty \end{bmatrix}$$

Stored with a dense or a sparse format

$$f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$



$$f_1 = A'^T \otimes_T f_0 = \begin{pmatrix} \infty \\ 0 \\ 1 \\ \infty \end{pmatrix}$$

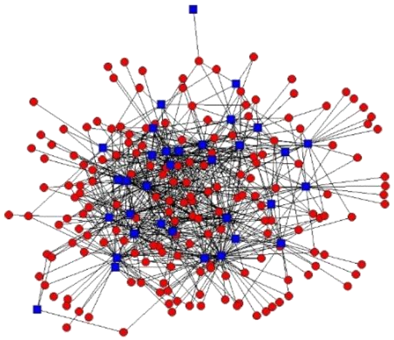
$$f_2 = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

How to do this in practice?

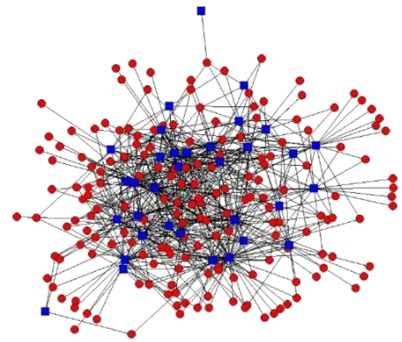
Graph Representations
Compressed Sparse Row (CSR)

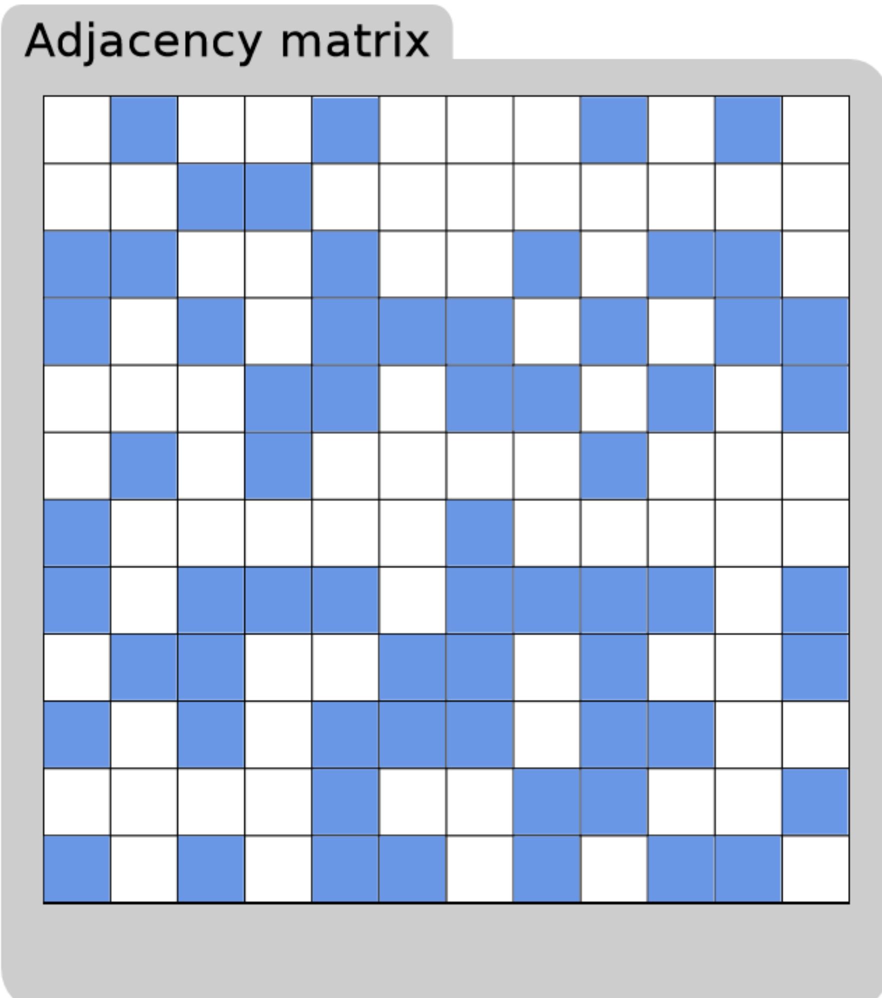
Graph Representations

Compressed Sparse Row (CSR)

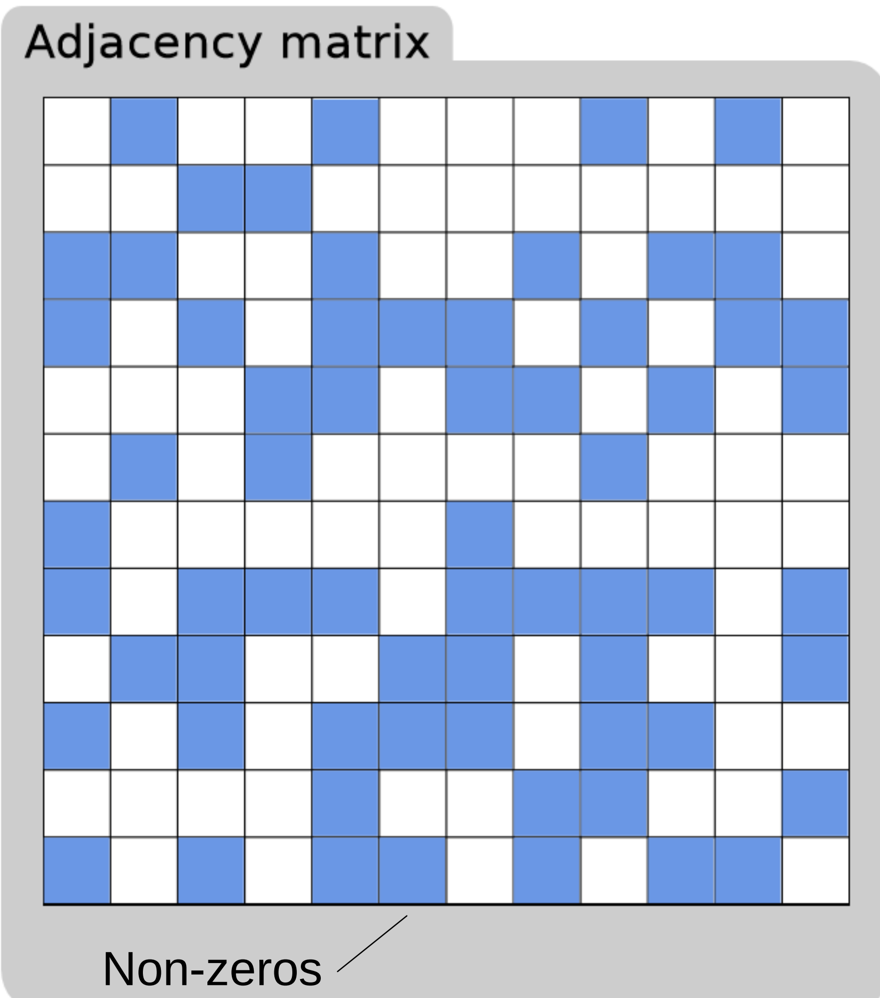


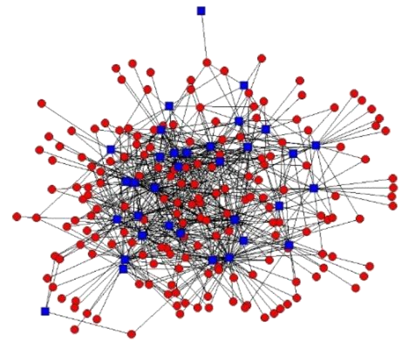
Graph Representations
Compressed Sparse Row (CSR)



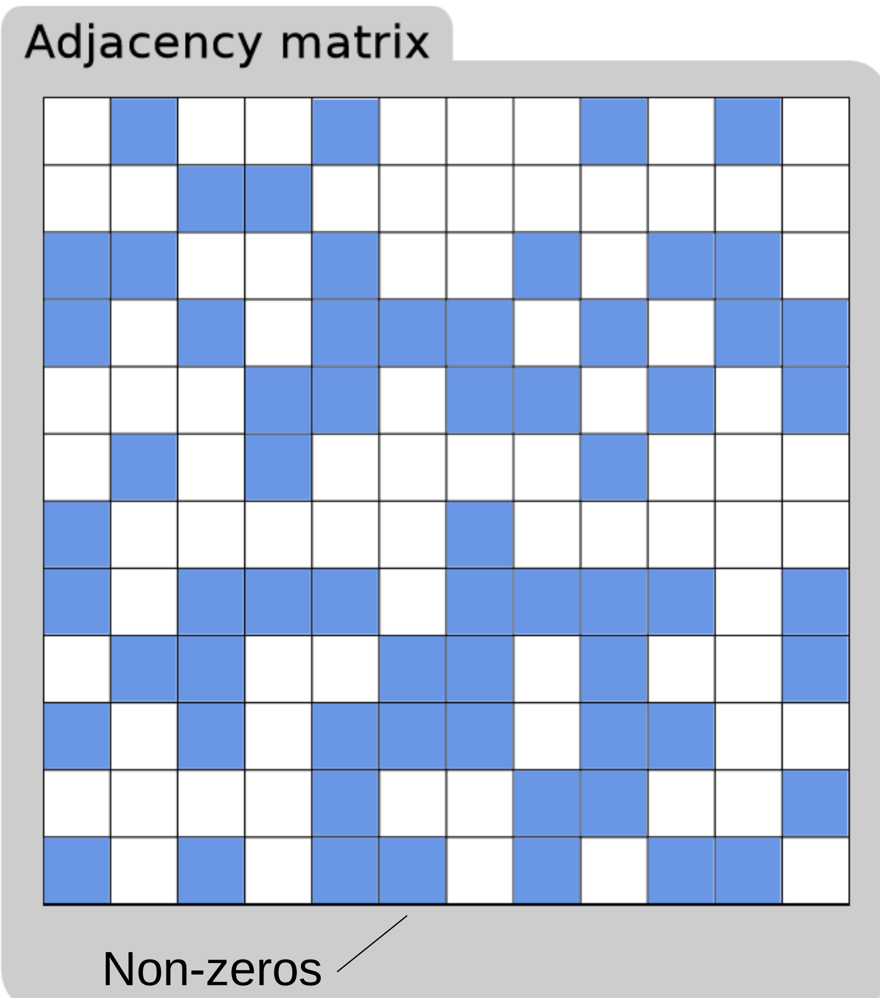


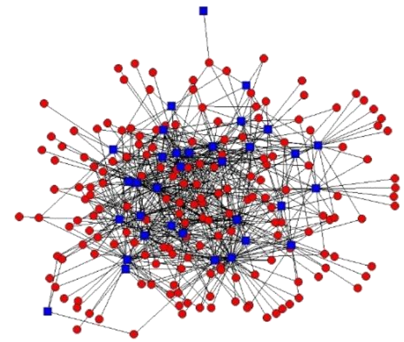
Graph Representations
Compressed Sparse Row (CSR)





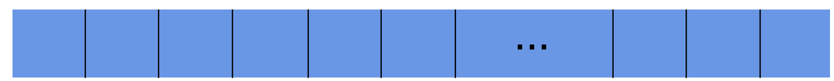
Graph Representations
Compressed Sparse Row (CSR)





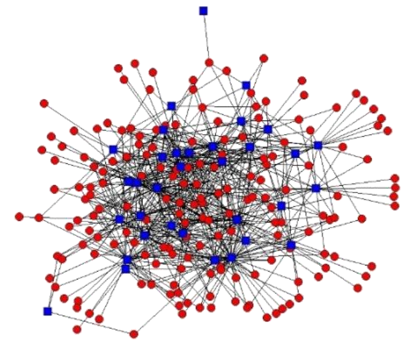
Non-zeros are stored in the *val* array

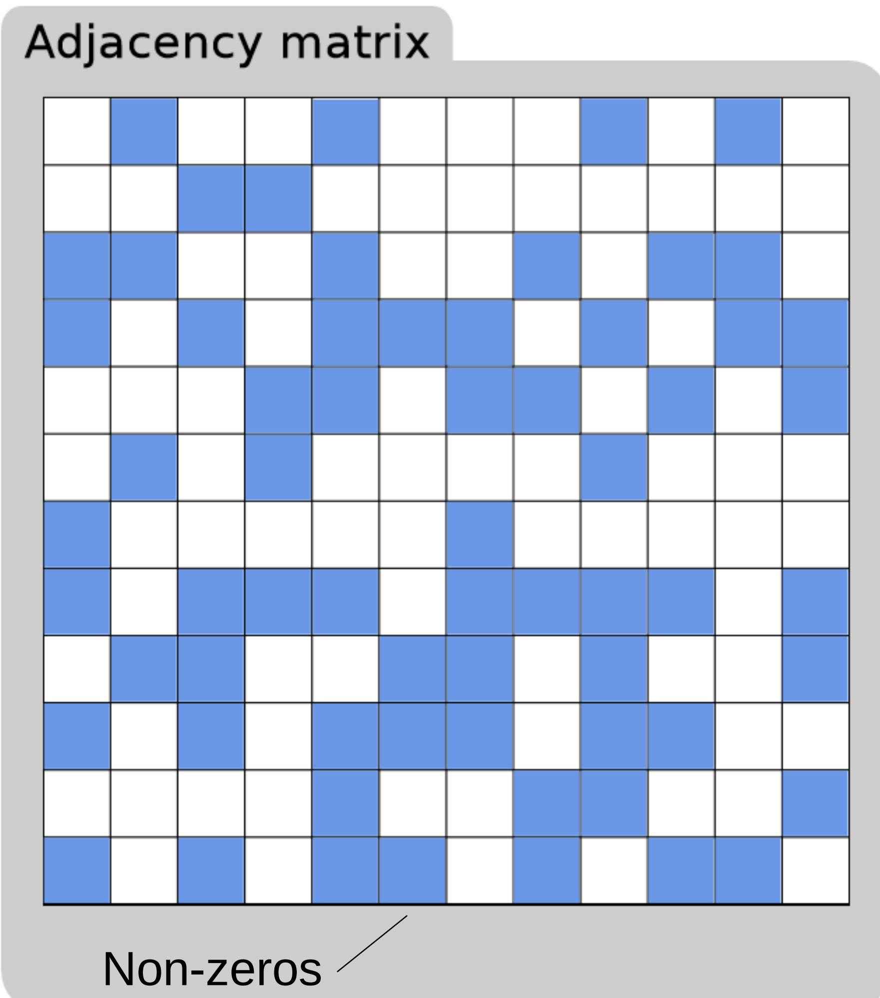
size: *2m* cells



n: number of vertices
m: number of edges

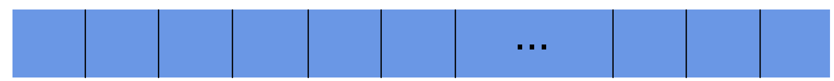
Graph Representations
Compressed Sparse Row (CSR)





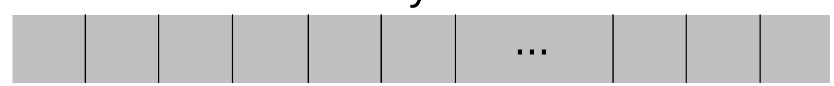
Non-zeros are stored in the *val* array

size: 2*m* cells



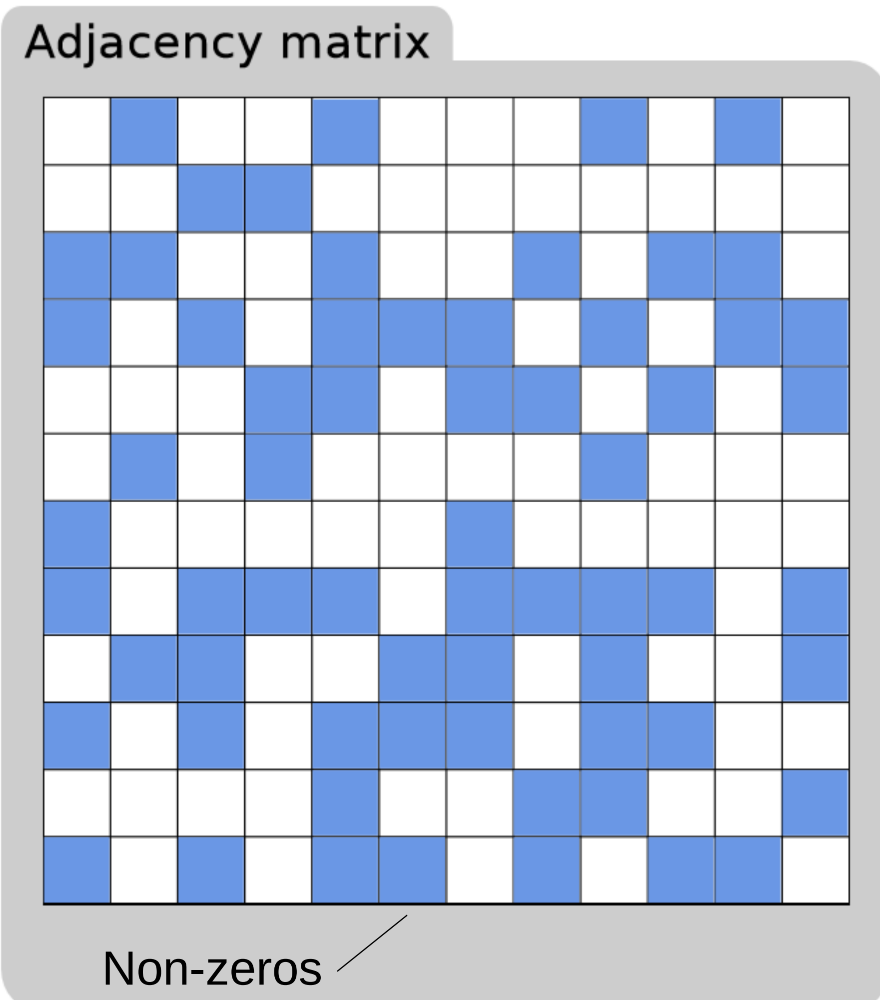
Column indices stored in the *col* array

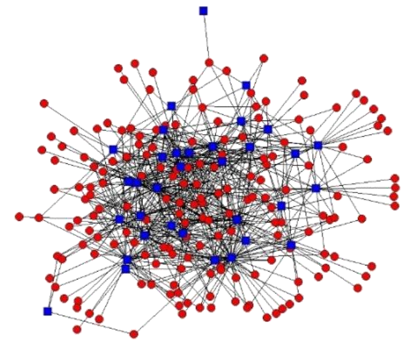
size: 2*m* cells



n: number of vertices
m: number of edges

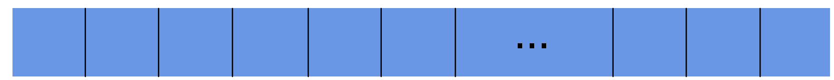
Graph Representations
Compressed Sparse Row (CSR)





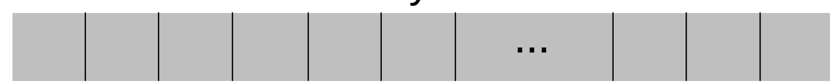
Non-zeros are stored in the *val* array

size: *2m* cells



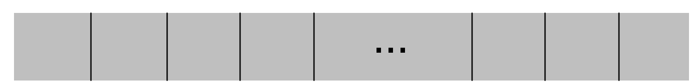
Column indices stored in the *col* array

size: *2m* cells



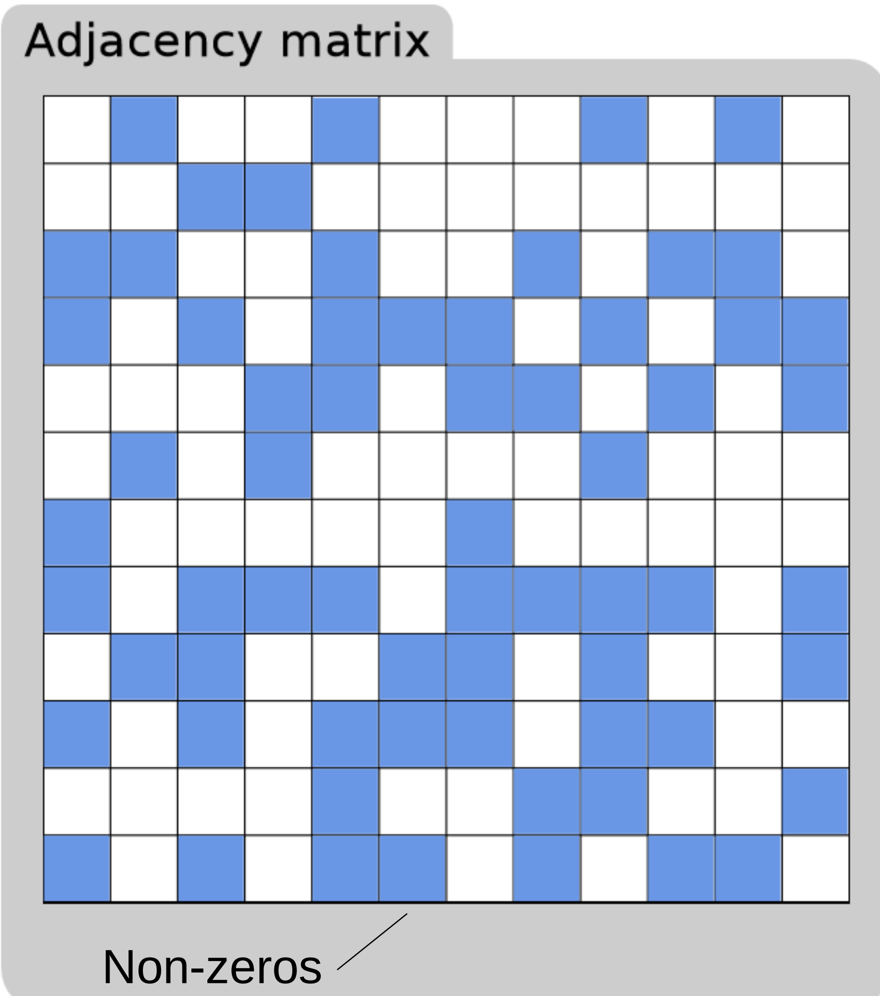
Row indices are stored in the *row* array

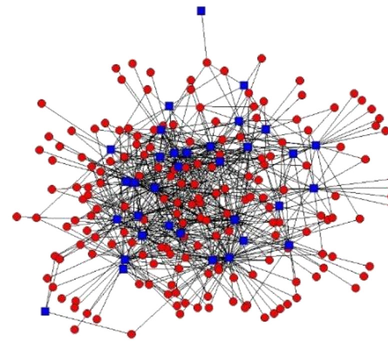
size: *n* cells



n: number of vertices
m: number of edges

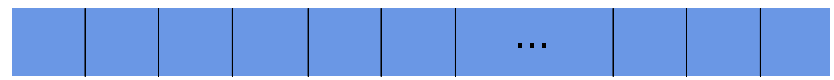
Graph Representations
Compressed Sparse Row (CSR)





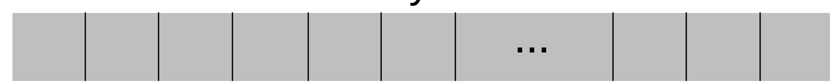
Non-zeros are stored in the *val* array

size: 2*m* cells



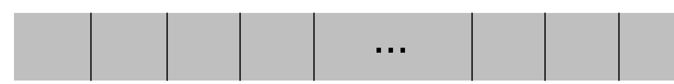
Column indices stored in the *col* array

size: 2*m* cells



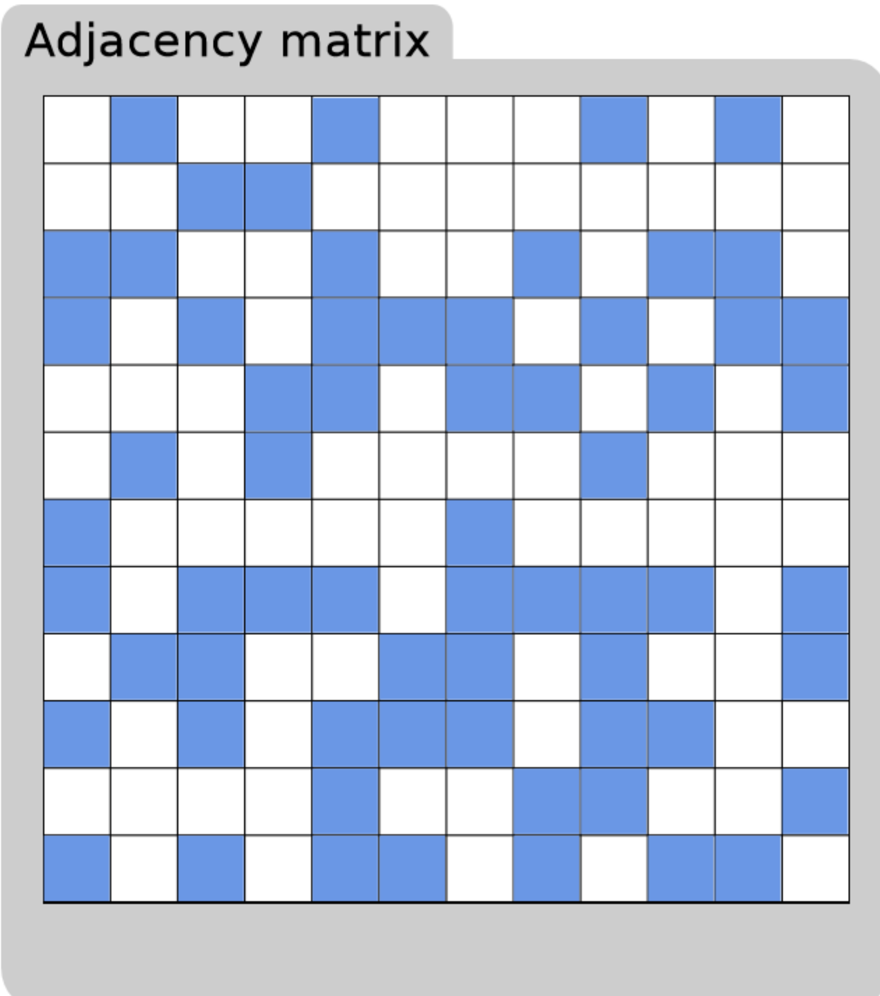
Row indices are stored in the *row* array

size: *n* cells



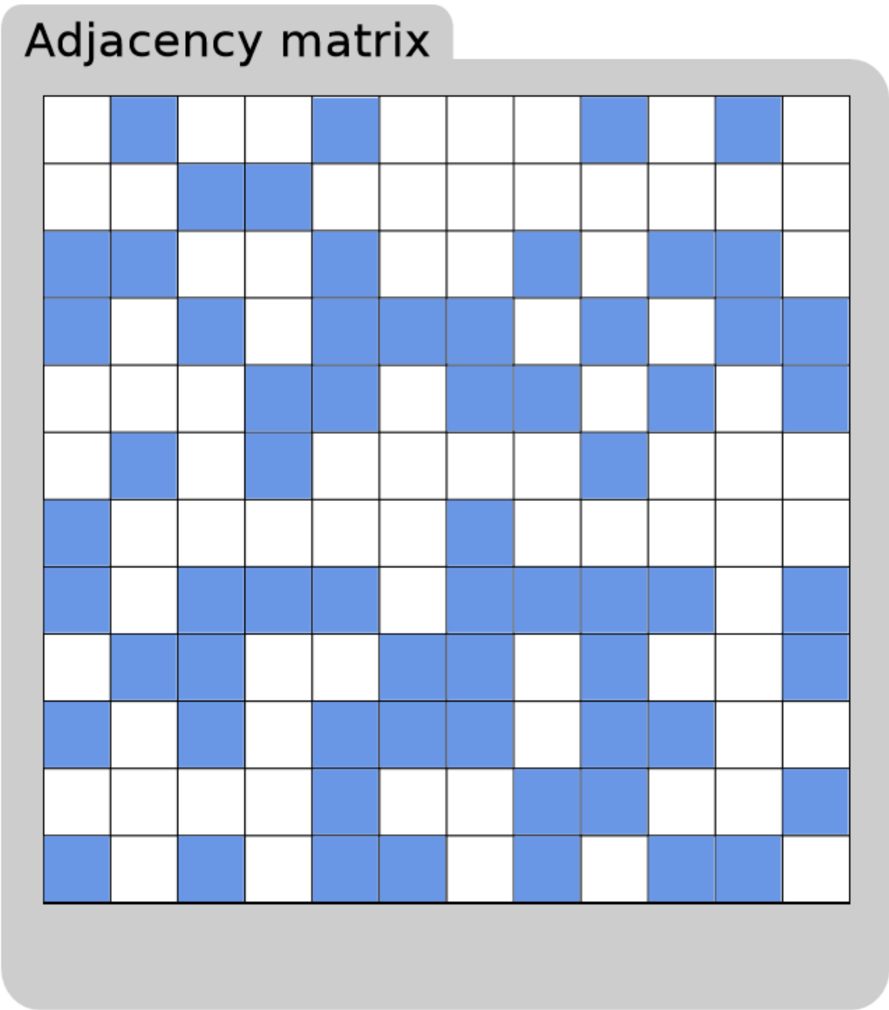
n: number of vertices
m: number of edges

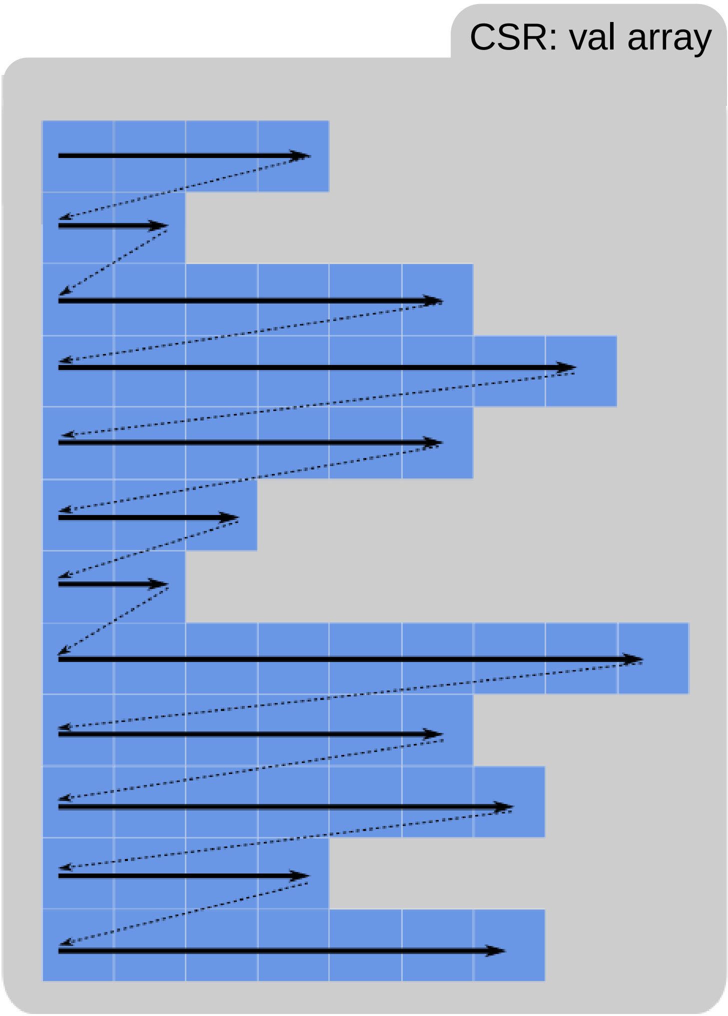
Graph Representations
Compressed Sparse Row (CSR)



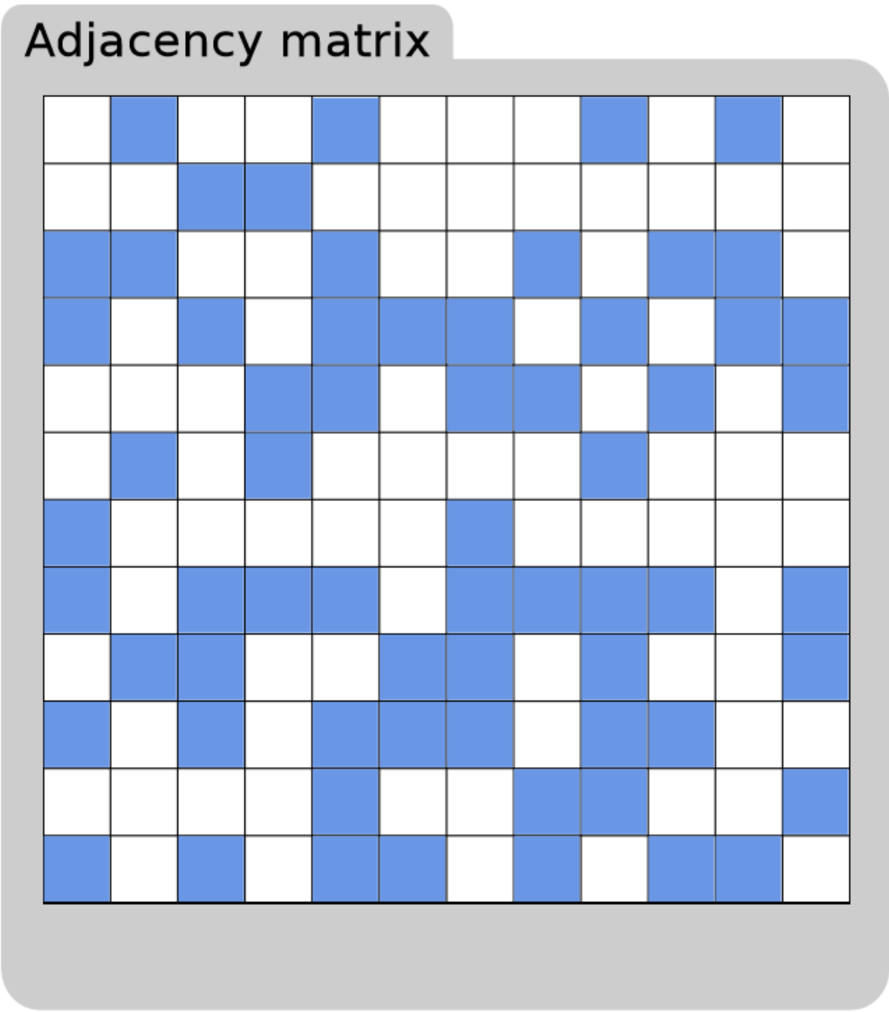
Graph Representations

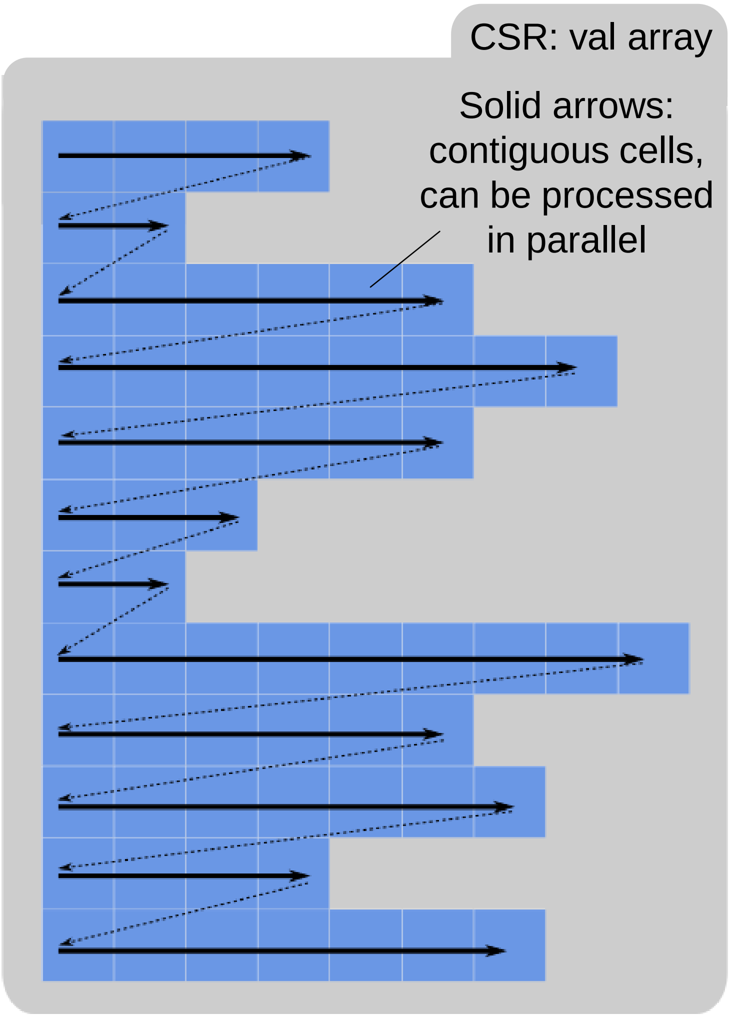
Compressed Sparse Row (CSR)





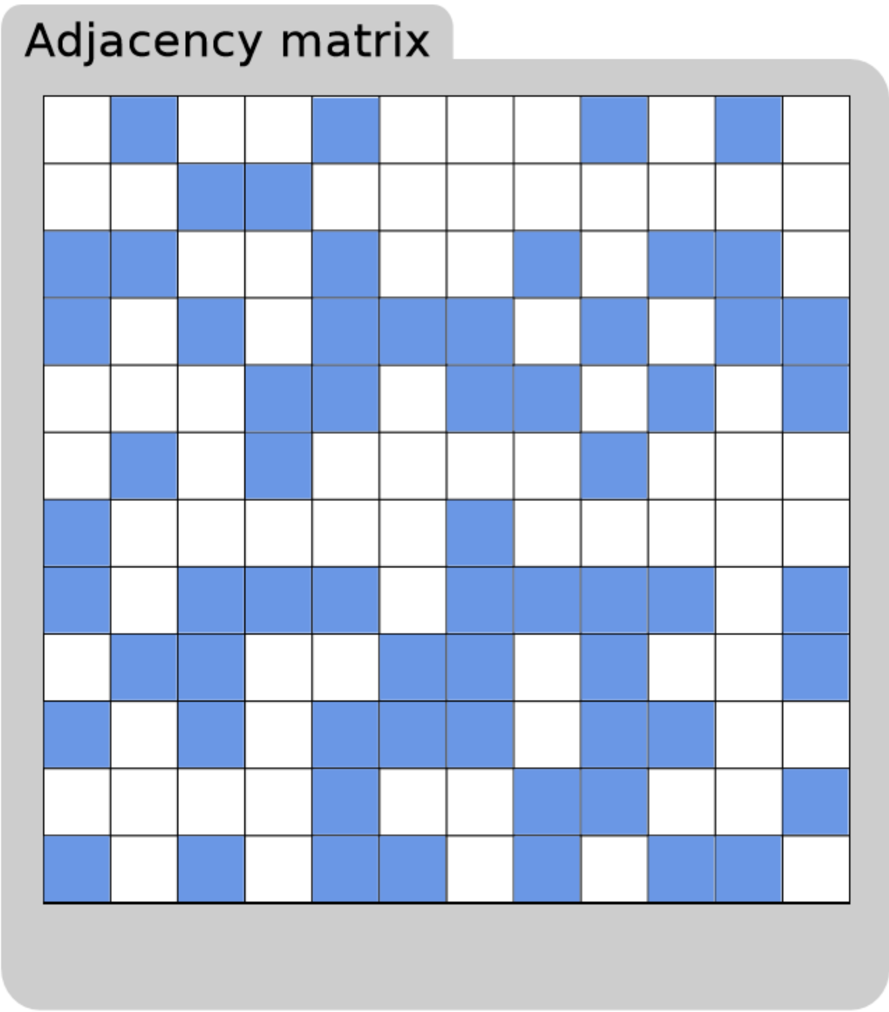
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

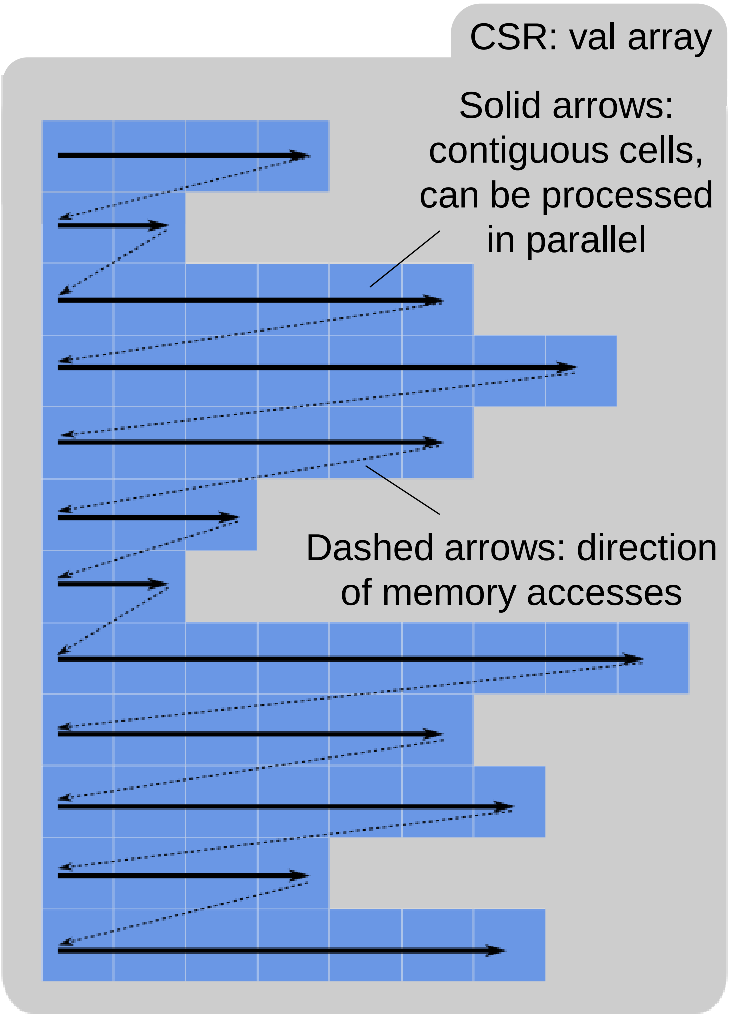




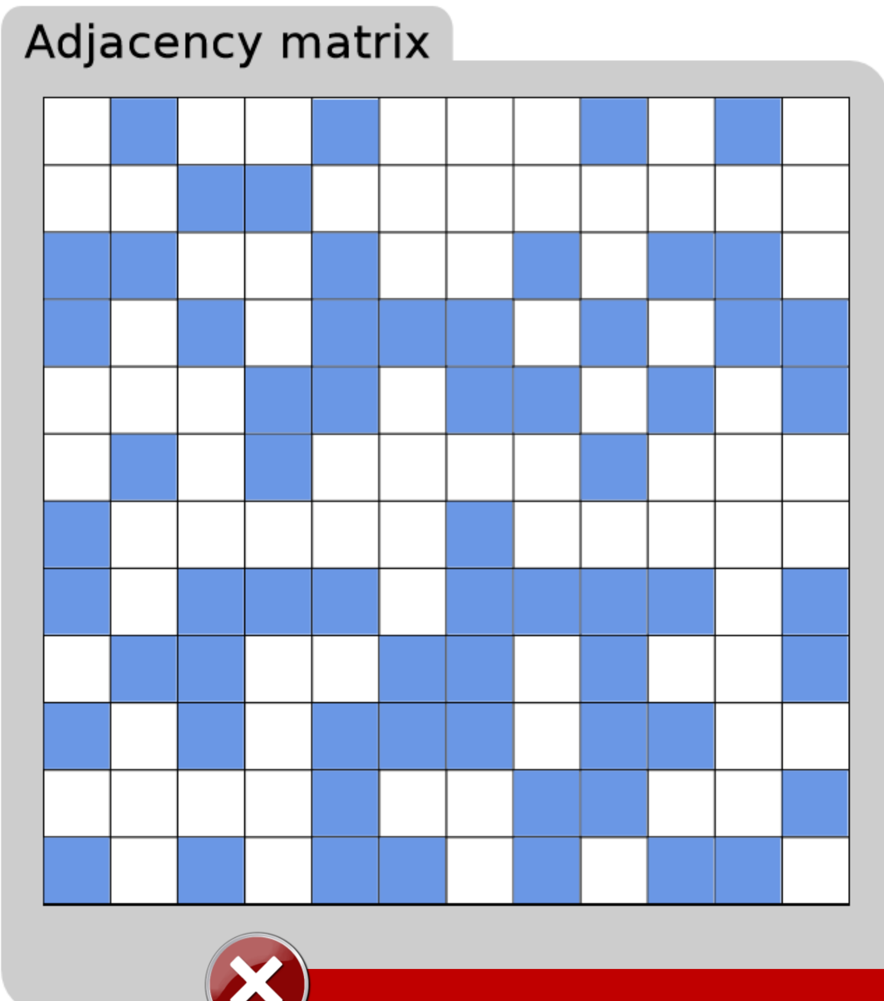
Graph Representations

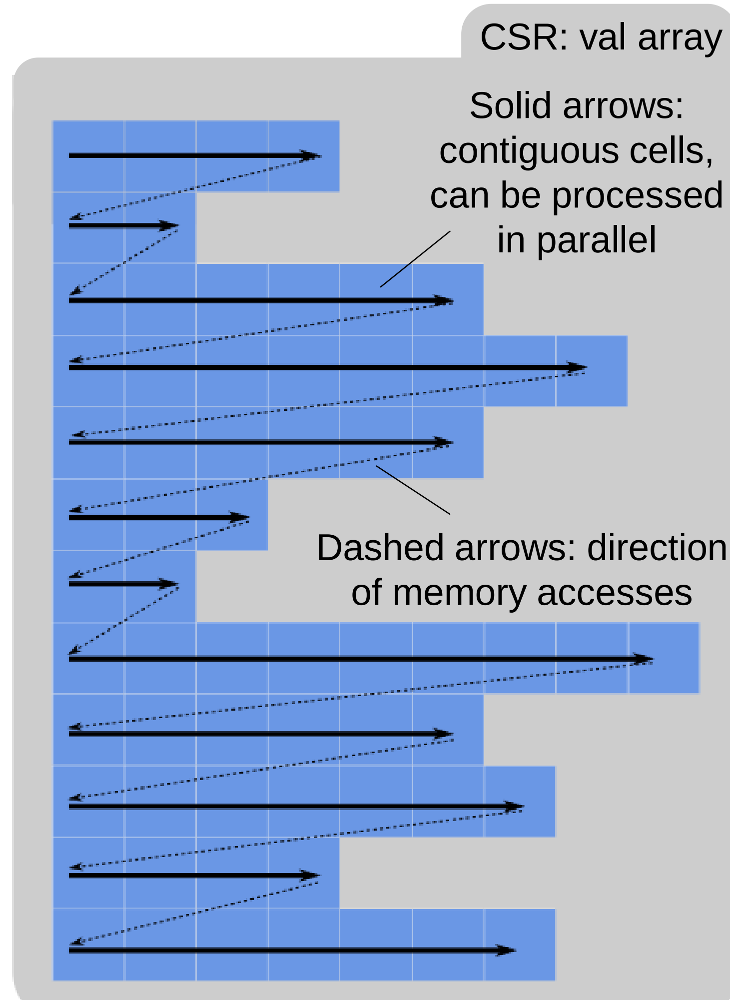
Compressed Sparse Row (CSR)





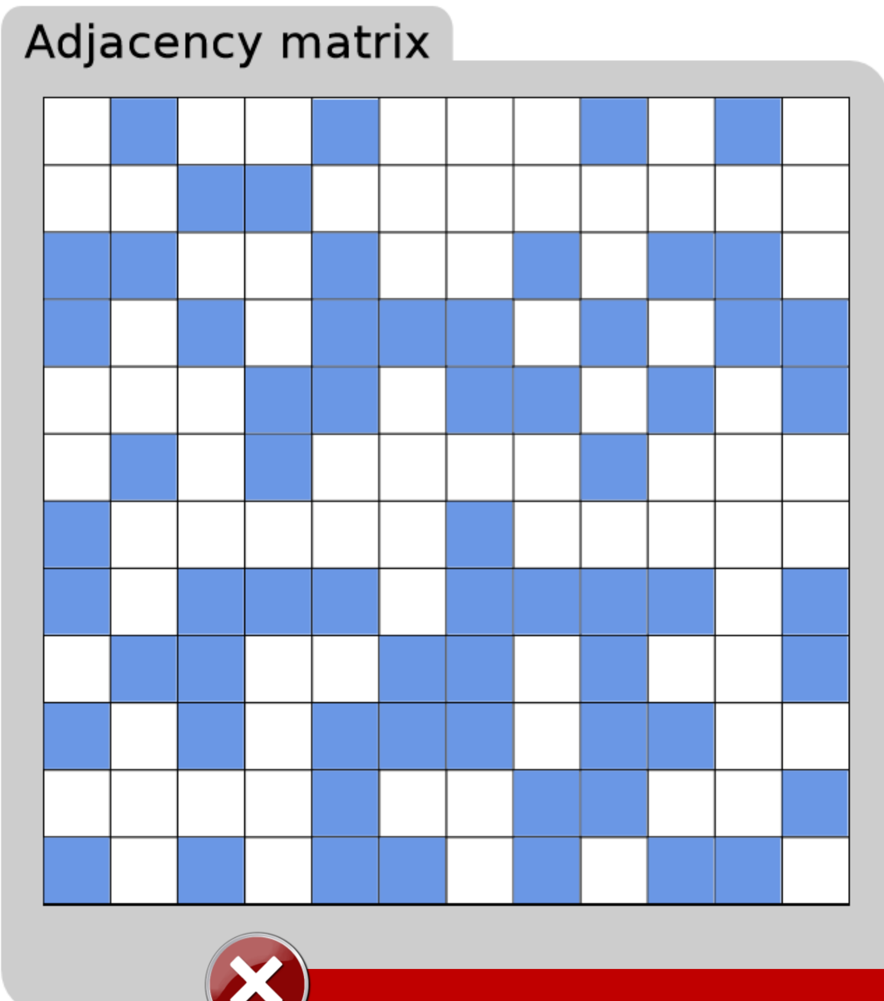
Graph Representations
Compressed Sparse Row (CSR)

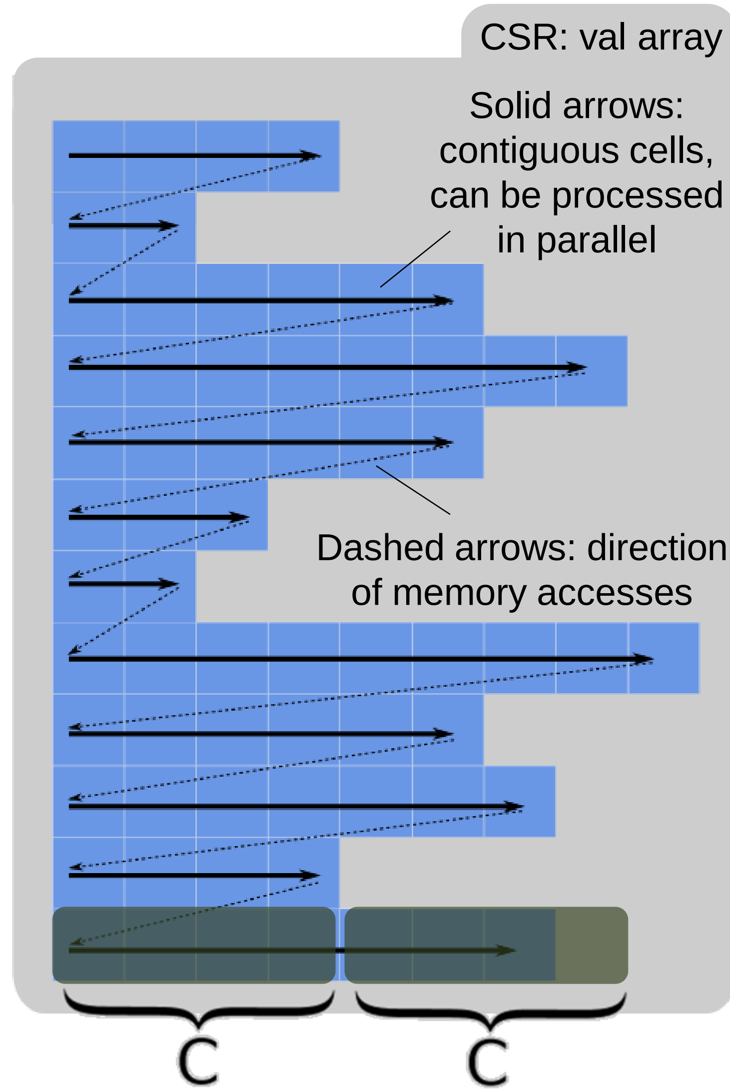




Row sizes incompatible with C

Graph Representations
Compressed Sparse Row (CSR)

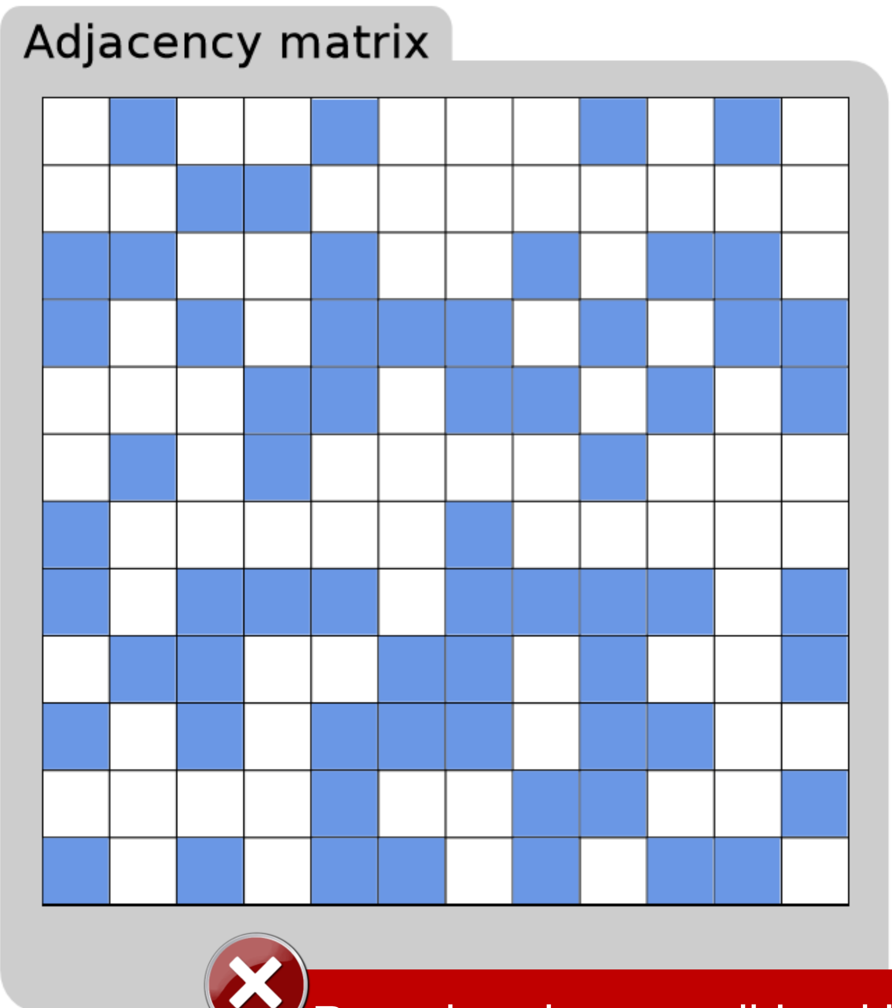


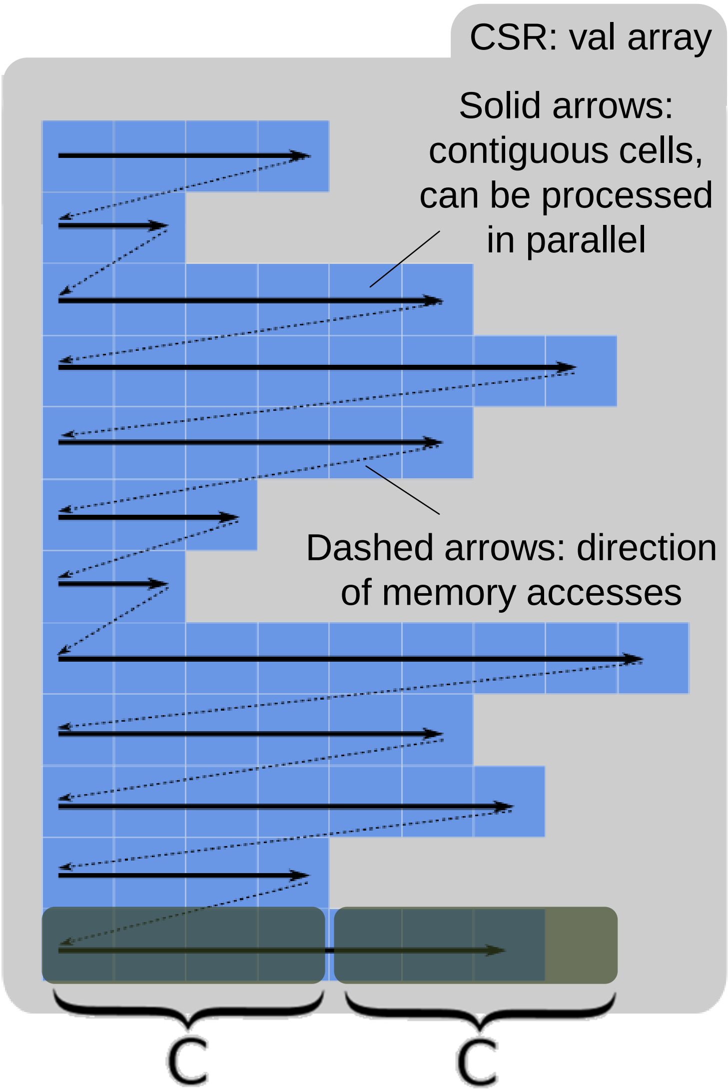


Row sizes incompatible with C

Graph Representations
Compressed Sparse Row (CSR)

Costly reductions within rows

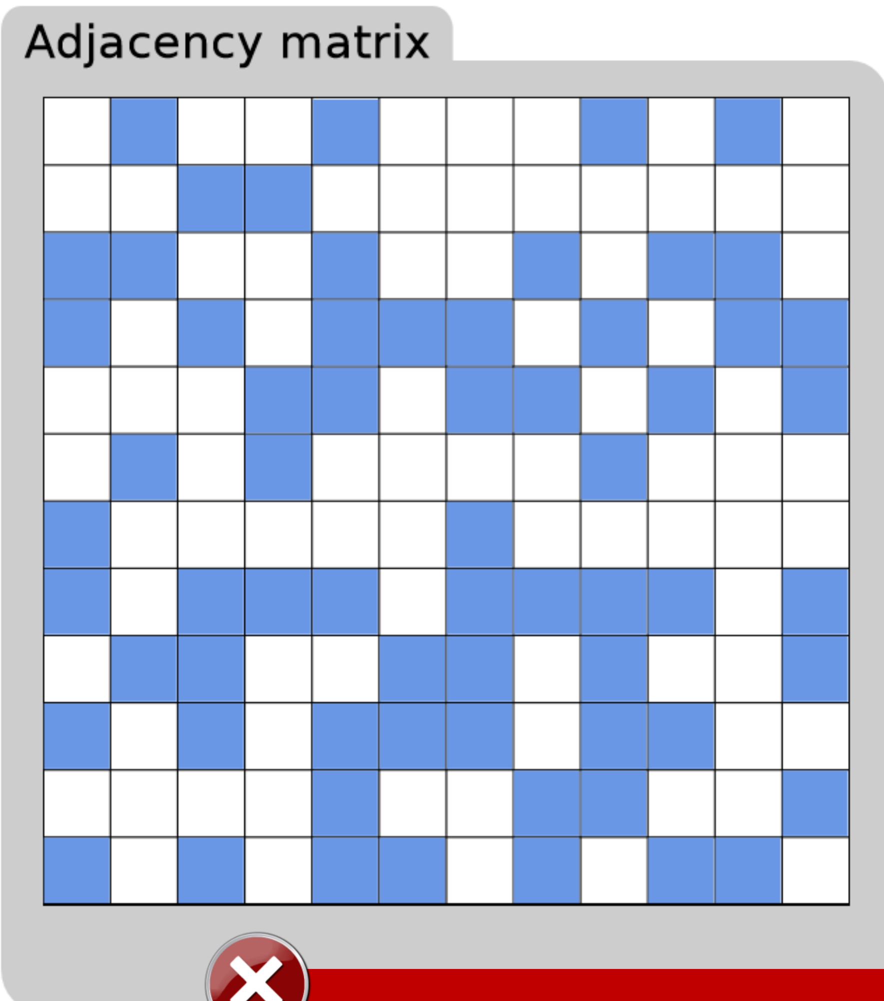


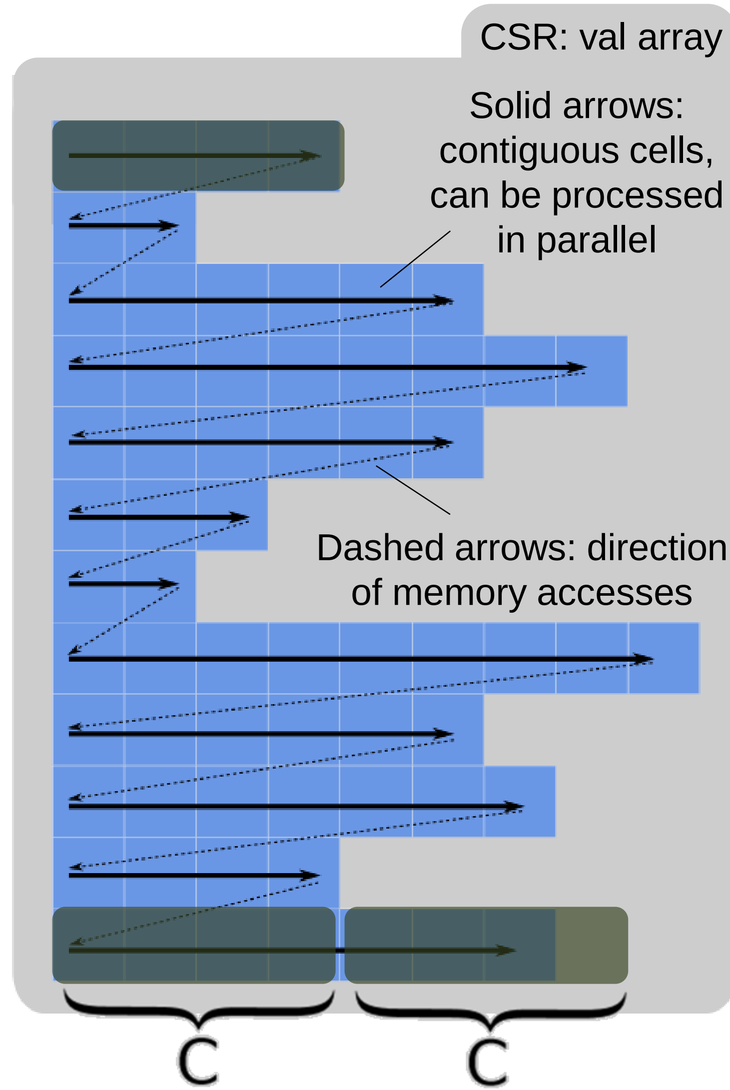


Row sizes incompatible with C

Graph Representations
Compressed Sparse Row (CSR)

Costly reductions within rows

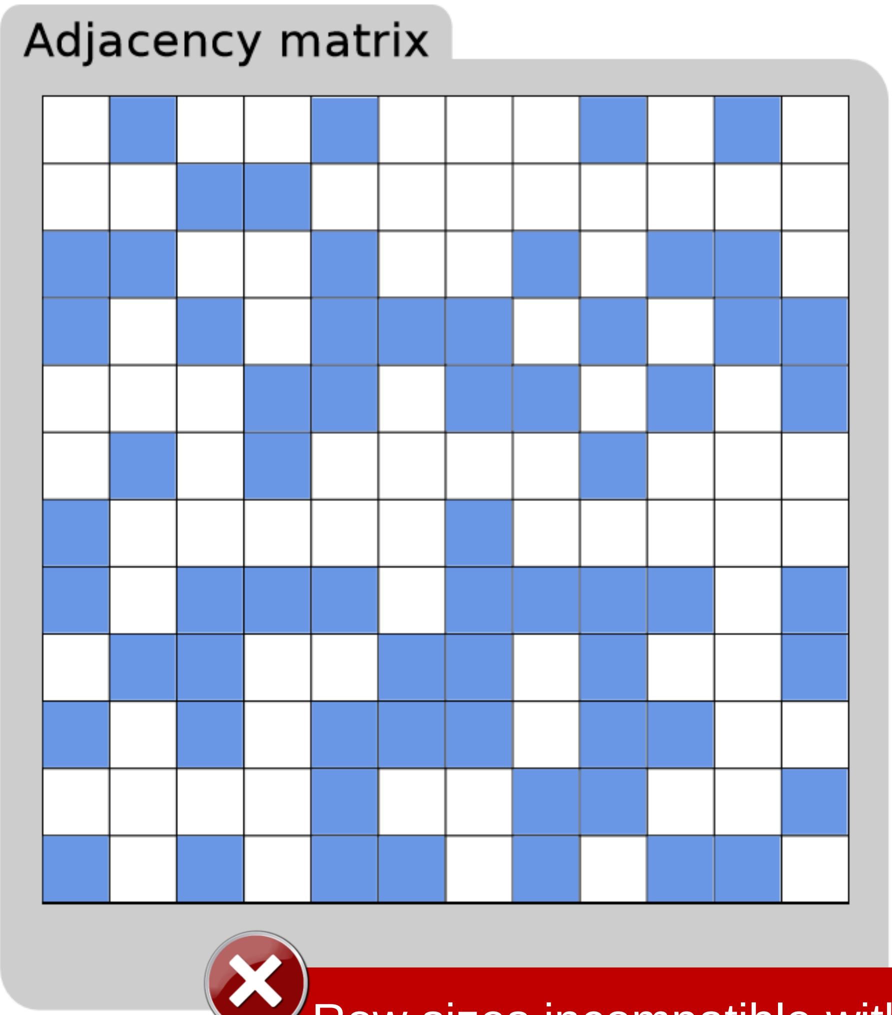


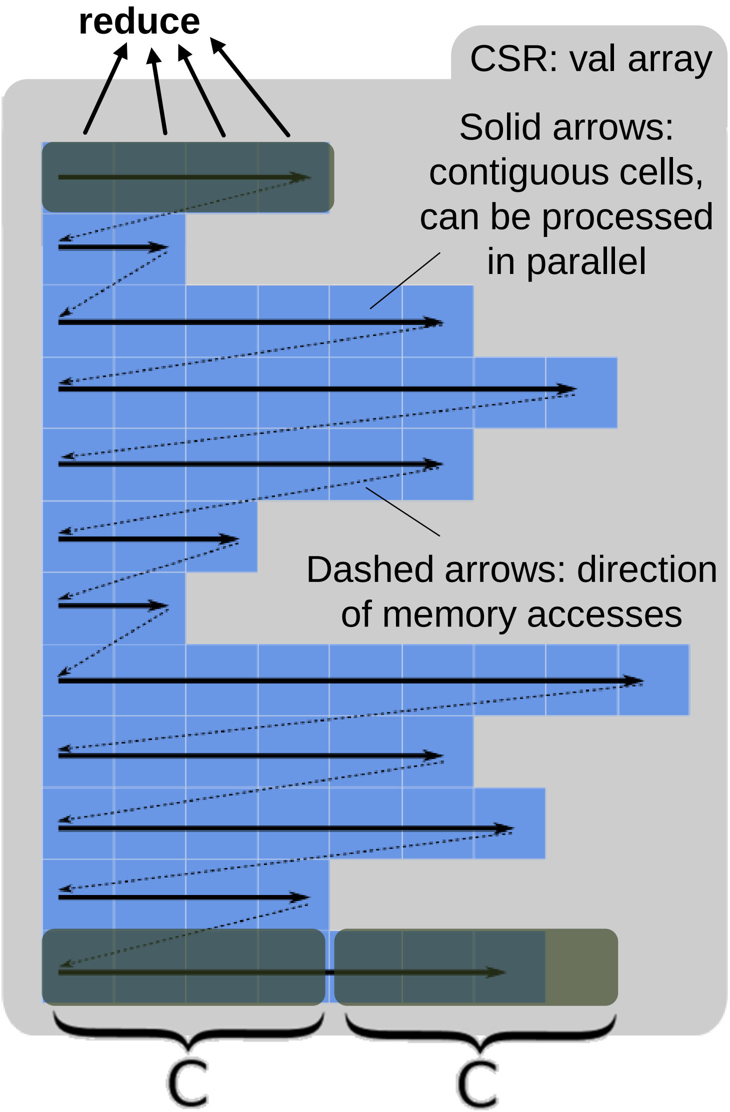


Row sizes incompatible with C

Graph Representations
Compressed Sparse Row (CSR)

Costly reductions within rows





Row sizes incompatible with C

Idea: utilize novel techniques used in numerical computations to accelerate graph processing

Idea: utilize novel techniques used in numerical computations to accelerate graph processing

ACSR [1]

ESB [3]

ELLPACK/ELL

SELL-P [4]

...

Sliced ELLPACK [2]

[1] A. Ashari et al. "Fast Sparse Matrix-vector Multiplication on GPUs for Graph Applications". SC14.
[2] A. Monakov et al. "Automatically tuning sparse matrix-vector multiplication for GPU architectures". ICHPEAC'10.
[3] X. Liu et al. "Efficient sparse matrix-vector multiplication on x86-based manycore processors". ICS'13.
[4] H. Anzt et al. "Acceleration of GPU-based Krylov Solvers via Data Transfer Reduction". Intl. J. HPCA 2015.

Specific to a given architecture

Idea: utilize novel techniques used in numerical computations to accelerate graph processing

ACSR [1]

ESB [3]

ELLPACK/ELL

SELL-P [4]

...

Sliced ELLPACK [2]

[1] A. Ashari et al. "Fast Sparse Matrix-vector Multiplication on GPUs for Graph Applications". SC14.
[2] A. Monakov et al. "Automatically tuning sparse matrix-vector multiplication for GPU architectures". ICHPEAC'10.
[3] X. Liu et al. "Efficient sparse matrix-vector multiplication on x86-based manycore processors". ICS'13.
[4] H. Anzt et al. "Acceleration of GPU-based Krylov Solvers via Data Transfer Reduction". Intl. J. HPCA 2015.

Specific to a given architecture

Idea: utilize novel techniques used in numerical computations to accelerate graph processing

Issues similar to the ones in CSR

ACSR [1]

ESB [3]

ELLPACK/ELL

SELL-P [4]

...

Sliced ELLPACK [2]

[1] A. Ashari et al. "Fast Sparse Matrix-vector Multiplication on GPUs for Graph Applications". SC14.
[2] A. Monakov et al. "Automatically tuning sparse matrix-vector multiplication for GPU architectures". ICHPEAC'10.
[3] X. Liu et al. "Efficient sparse matrix-vector multiplication on x86-based manycore processors". ICS'13.
[4] H. Anzt et al. "Acceleration of GPU-based Krylov Solvers via Data Transfer Reduction". Intl. J. HPCA 2015.

Idea: utilize novel techniques used in numerical computations to accelerate graph processing

Specific to a given architecture

Issues similar to the ones in CSR

ACSR [1]

ESB [3]

ELLPACK/ELL

SELL-P [4]

...

Sliced ELLPACK [2]

SELL-C-sigma [5]

[1] A. Ashari et al. "Fast Sparse Matrix-vector Multiplication on GPUs for Graph Applications". SC14.
[2] A. Monakov et al. "Automatically tuning sparse matrix-vector multiplication for GPU architectures". ICHPEAC'10.
[3] X. Liu et al. "Efficient sparse matrix-vector multiplication on x86-based manycore processors". ICS'13.
[4] H. Anzt et al. "Acceleration of GPU-based Krylov Solvers via Data Transfer Reduction". Intl. J. HPCA 2015.

Idea: utilize novel techniques used in numerical computations to accelerate graph processing

Specific to a given architecture

Issues similar to the ones in CSR

ACSR [1]

ESB [3]

ELLPACK/ELL

SELL-P [4]

...

Sliced ELLPACK [2]

SELL-C-sigma [5]

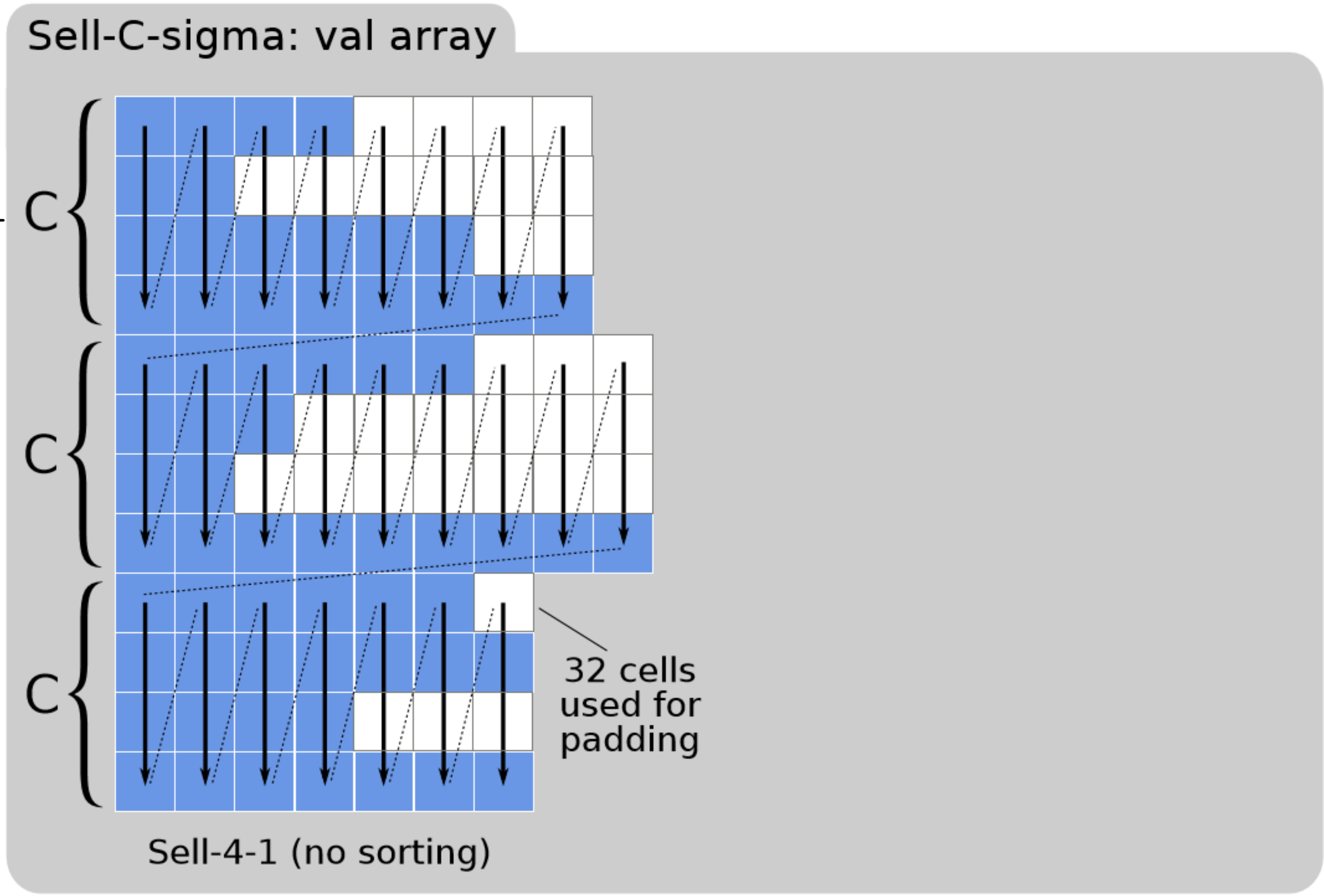
[1] A. Ashari et al. "Fast Sparse Matrix-vector Multiplication on GPUs for Graph Applications". SC14.
[2] A. Monakov et al. "Automatically tuning sparse matrix-vector multiplication for GPU architectures". ICHPEAC'10.
[3] X. Liu et al. "Efficient sparse matrix-vector multiplication on x86-based manycore processors". ICS'13.
[4] H. Anzt et al. "Acceleration of GPU-based Krylov Solvers via Data Transfer Reduction". Intl. J. HPCA 2015.
[5] M. Kreutzer et al. "A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units". SIAM J. of Scientific Computing.

Graph Representations

Sell-C-Sigma

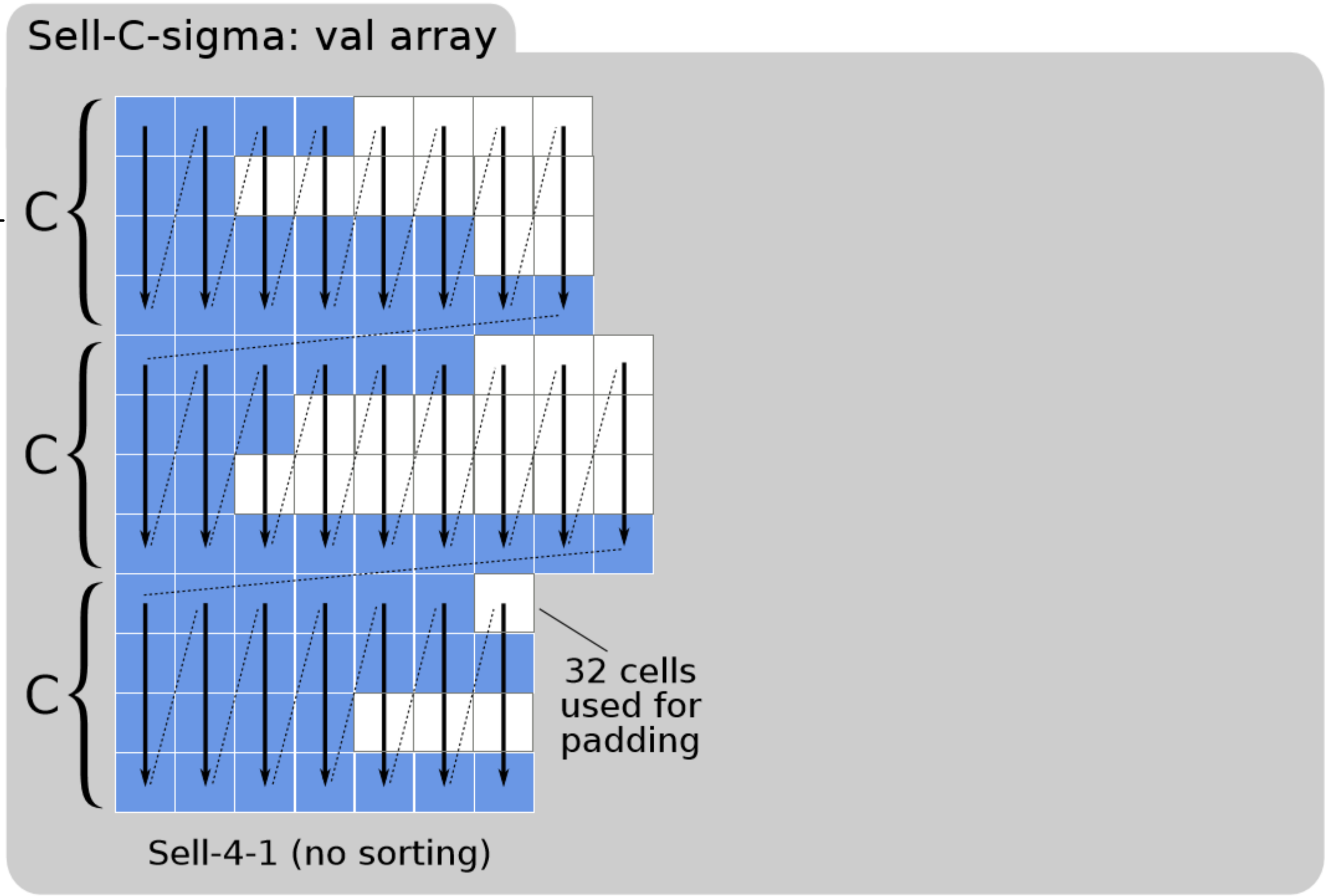
Graph Representations
Sell-C-Sigma

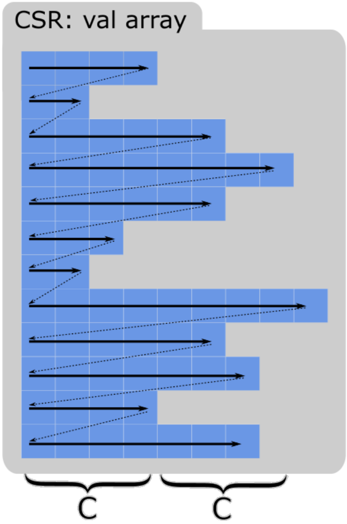
chunk size



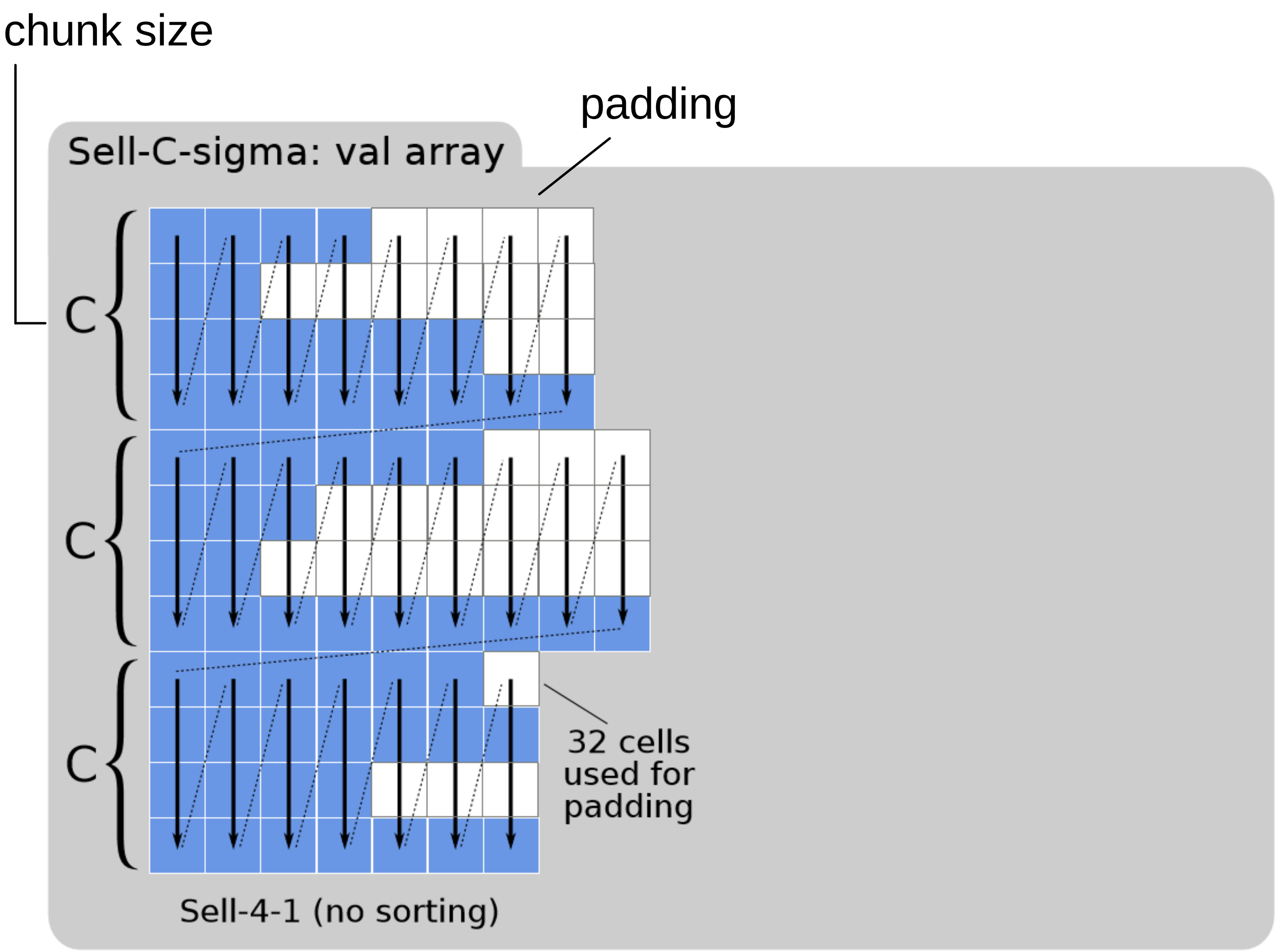
Graph Representations
Sell-C-Sigma

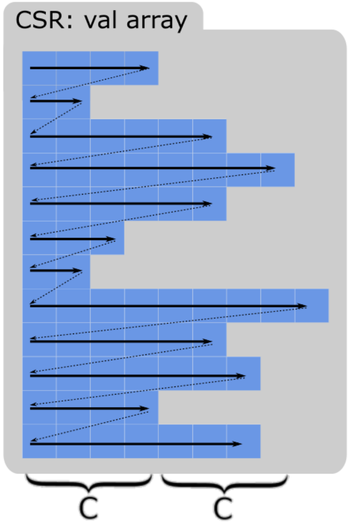
chunk size



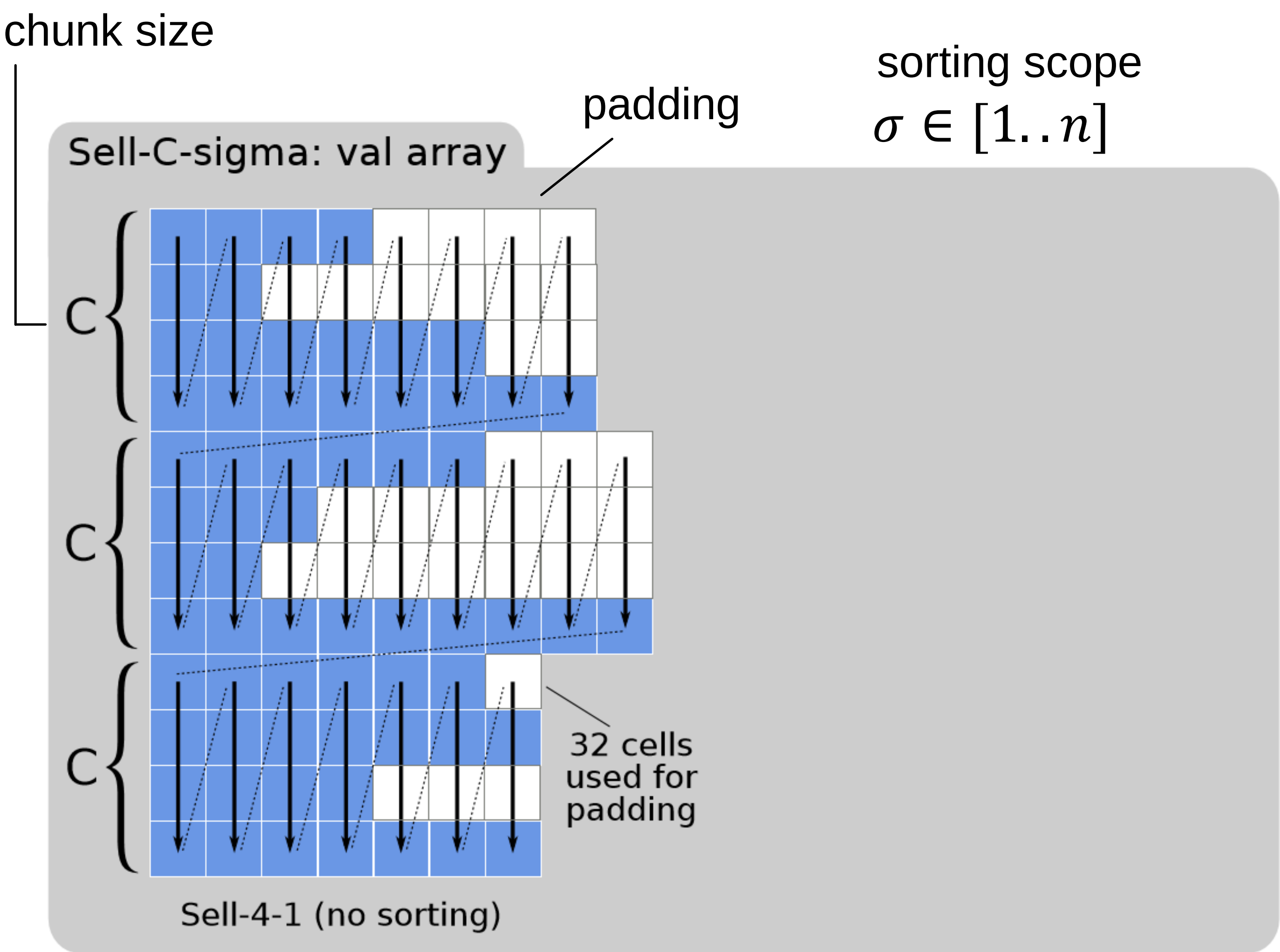


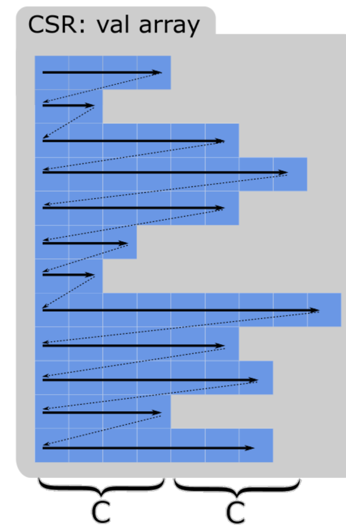
Graph Representations
Sell-C-Sigma





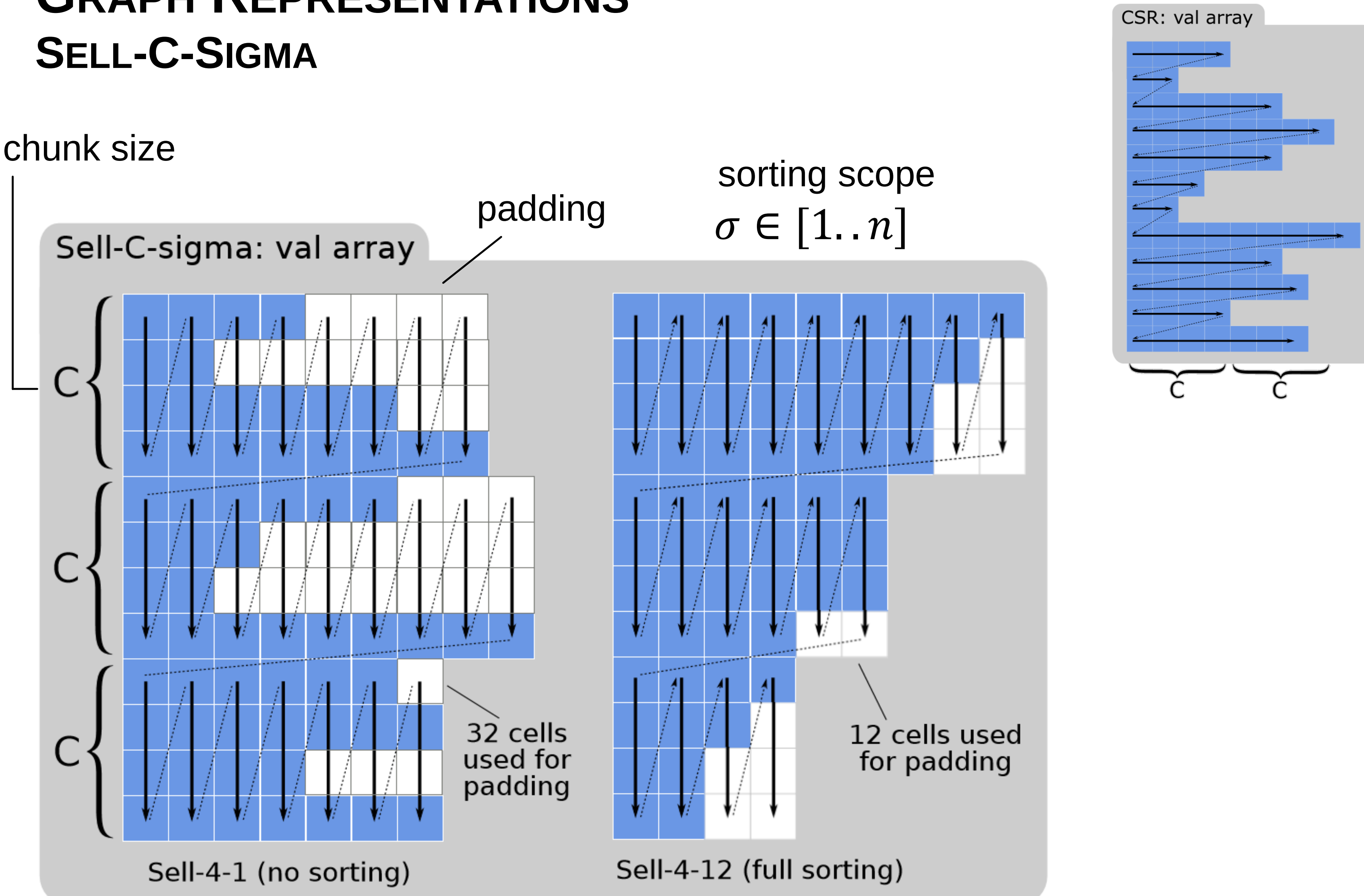
Graph Representations
Sell-C-Sigma





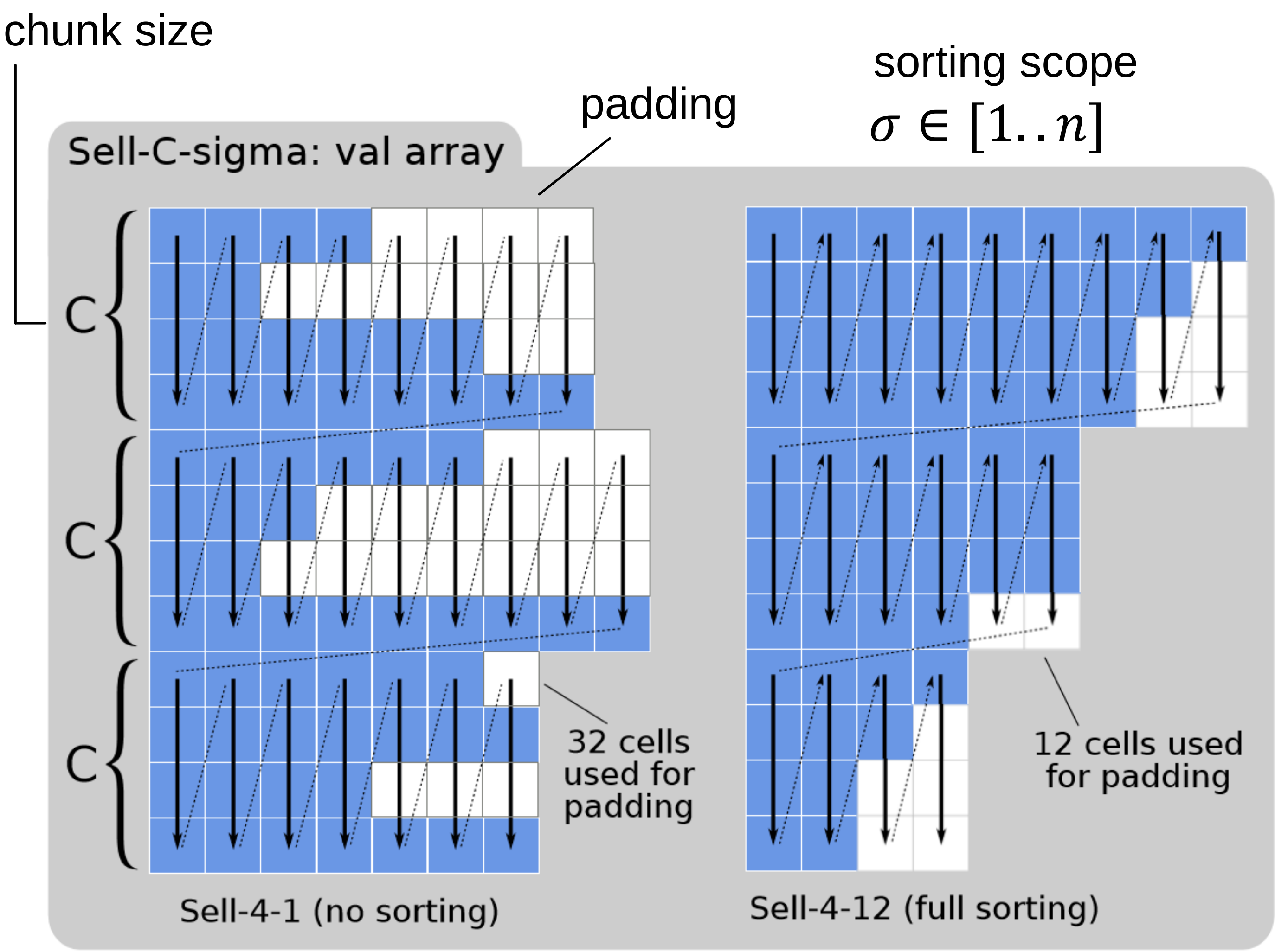
Graph Representations

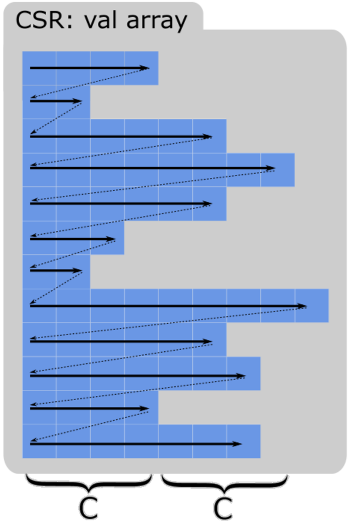
Sell-C-Sigma



Graph Representations

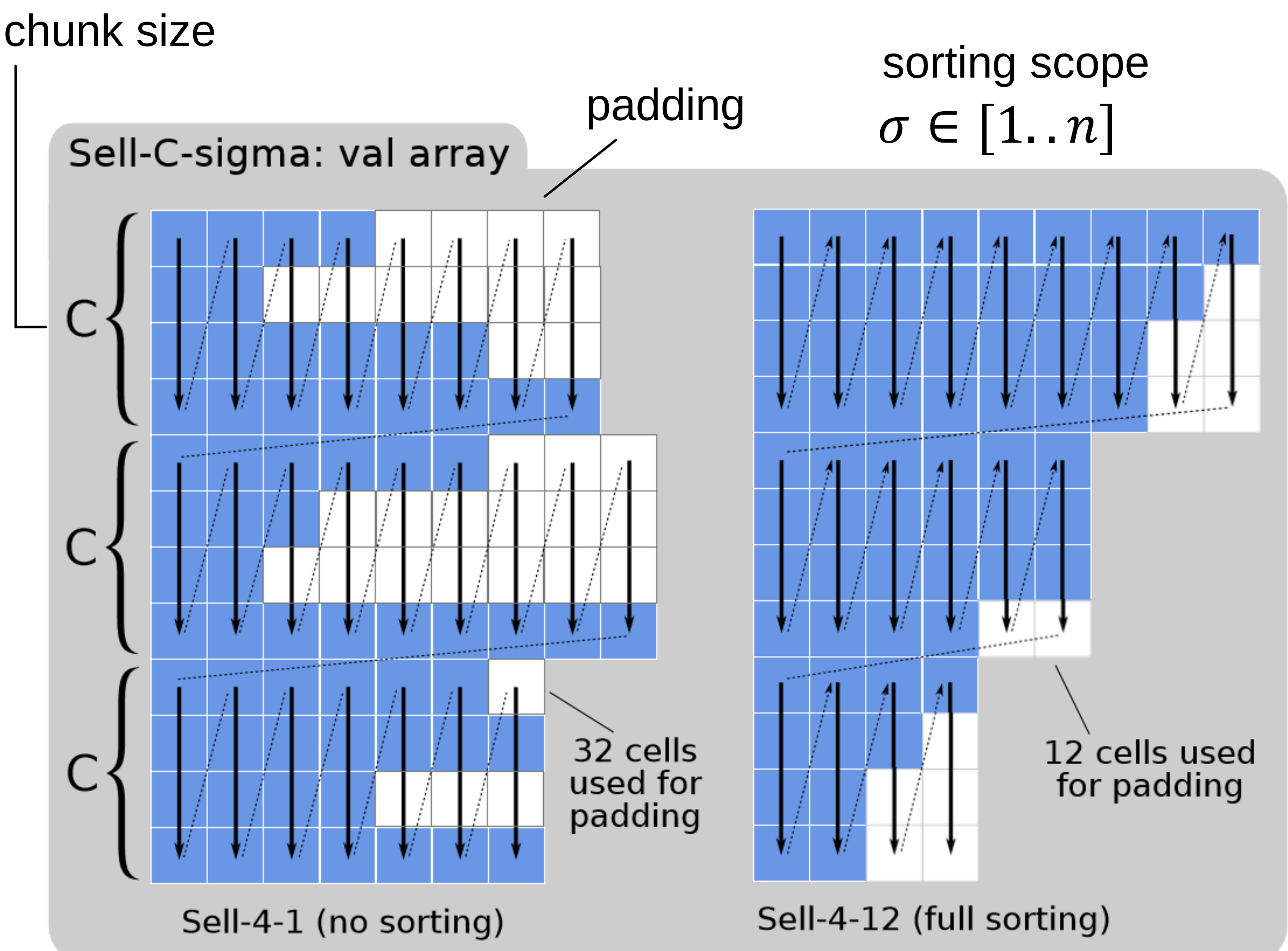
Sell-C-Sigma

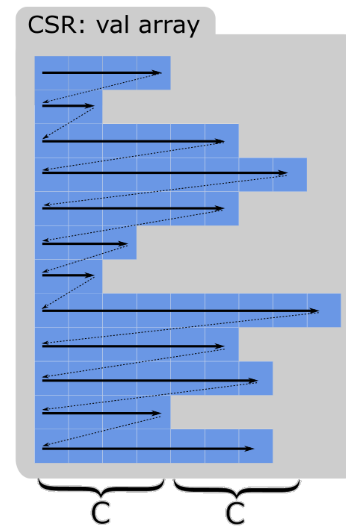




Reductions fast with SIMD operations

GRAPH REPRESENTATIONS
SELL-C-SIGMA





Reductions fast with SIMD operations

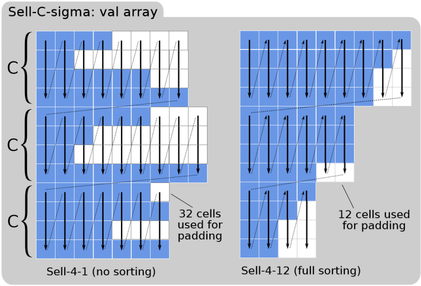
Portable

Sell-C-Sigma + Semirings

Systematic Analysis

Sell-C-Sigma + Semirings

Systematic Analysis



+

$(X, op_1, op_2, el_1, el_2)$

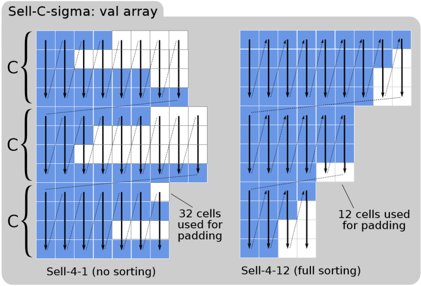
$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$(\mathbb{R}, max, \cdot, -\infty, 1)$

$(\mathbb{R}, +, \cdot, 0, 1)$

$(\{0,1\}, |, \&, 0, 1)$

Sell-C-Sigma + Semirings
Systematic Analysis



+

$(X, op_1, op_2, el_1, el_2)$

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

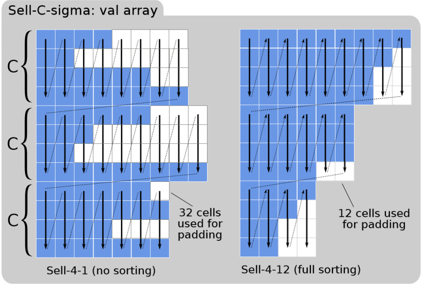
$(\mathbb{R}, max, \cdot, -\infty, 1)$

$(\mathbb{R}, +, \cdot, 0, 1)$

$(\{0,1\}, |, \&, 0, 1)$

What are the actual semirings and their formulations?

SELL-C-SIGMA + SEMIRINGS
SYSTEMATIC ANALYSIS



+

$(X, op_1, op_2, el_1, el_2)$

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$(\mathbb{R}, max, \cdot, -\infty, 1)$

$(\mathbb{R}, +, \cdot, 0, 1)$

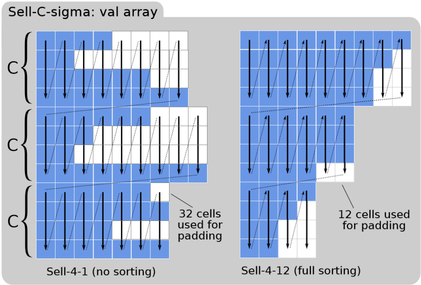
$(\{0,1\}, |, \&, 0, 1)$

What are the actual semirings and their formulations?

How to derive both distances and parents?

Sell-C-Sigma + Semirings

Systematic Analysis



+

$(X, op_1, op_2, el_1, el_2)$

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$(\mathbb{R}, max, \cdot, -\infty, 1)$

$(\mathbb{R}, +, \cdot, 0, 1)$

$(\{0,1\}, |, \&, 0, 1)$

What are the actual semirings and their formulations?

What is work complexity of BFS based on Sell-C-sigma?

How to derive both distances and parents?

SEMIRINGS FOR BFS

SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

Semirings for BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

$$(\mathbb{R}, +, \cdot, 0, 1)$$

Semirings for BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$f_k = A^T \otimes_R f_{k-1}$$

SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$f_k = \left(A^T \otimes_R f_{k-1}\right) \odot_R \left(\overline{\sum_{l=0}^{k-1} f_l}\right)$$

Hadamard product

SEMIRINGS FOR BFS

Hadamard product

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$f_k = \left(A^T \otimes_R f_{k-1}\right) \odot_R \left(\overline{\sum_{l=0}^{k-1} f_l}\right)$$

$distances = O(D)$

$parents \in O(m)$

After iterations

Semirings for BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

Hadamard product

$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\overline{\sum_{l=0}^{k-1} f_l}\right)$$

$distances = O(D)$

$parents \in O(m)$

After iterations

Boolean semiring

$$(\{0,1\}, |, \&, 0, 1)$$

$f_k =$ [similar to Real]

$distances \in O(D)$

$parents \in O(m)$

After iterations

Semirings for BFS

Tropical semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$f_k = A'^T \otimes_T f_{k-1}$

$distances \in O(1)$
$parents \in O(m)$

After iterations

Real semiring

Hadamard product

$(\mathbb{R}, +, \cdot, 0, 1)$

$f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\overline{\sum_{l=0}^{k-1} f_l}\right)$

$distances = O(D)$
$parents \in O(m)$

After iterations

Boolean semiring

$(\{0,1\}, |, \&, 0, 1)$

f_k = [similar to Real]

$distances \in O(D)$
$parents \in O(m)$

After iterations

Sel-max "semiring"

$(\mathbb{R}, max, \cdot, -\infty, 1)$

f_k =[more equations ☺]

$distances \in O(D)$
$parents \in O(1)$

After iterations

Semirings for BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$f_k = (A^T \otimes_R f_{k-1}) \odot_R \overline{\left(\sum_{l=0}^{k-1} f_l\right)}$$

Hadamard product

$distances = O(D)$

$parents \in O(m)$

After iterations

Boolean semiring

$$(\{0,1\}, |, \&, 0, 1)$$

f_k = [similar to Real]

$distances \in O(D)$

$parents \in O(m)$

After iterations

Sel-max “semiring”

$$(\mathbb{R}, max, \cdot, -\infty, 1)$$

f_k =[more equations ☺]

$distances \in O(D)$

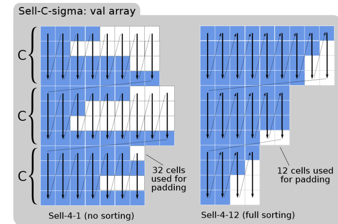
$parents \in O(1)$

After iterations

Sell-C-Sigma + Semirings

Formulations

SELL-C-SIGMA + SEMIRINGS FORMULATIONS



+

$(X, op_1, op_2, el_1, el_2)$

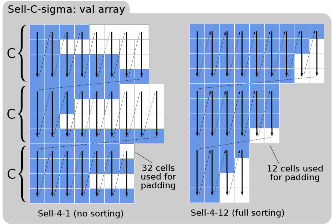
$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$(\mathbb{R}, +, \cdot, 0, 1)$

$(\{0,1\}, |, \&, 0, 1)$

$(\mathbb{R}, max, \cdot, -\infty, 1)$

SELL-C-SIGMA + SEMIRINGS FORMULATIONS



+

$(X, op_1, op_2, el_1, el_2)$

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

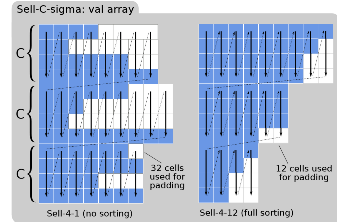
$(\mathbb{R}, +, \cdot, 0, 1)$

$(\{0,1\}, |, \&, 0, 1)$

$(\mathbb{R}, max, \cdot, -\infty, 1)$

```
    // Compute x_k (versions differ based on the used semiring):
#ifdef USE_TROPICAL_SEMIRING
    x = MIN(ADD(rhs, vals), x);
```
TROPICAL SEMIRING
```
#elif defined USE_BOOLEAN_SEMIRING
    x = OR(AND(rhs, vals), x);
```
BOOLEAN SEMIRING
```
#elif defined USE_SELMAX_SEMIRING
    x = MAX(MUL(rhs, vals), x);
```
SEL-MAX SEMIRING
```
#endif
    index += C;
  }
  // Now, derive f_k (versions differ based on the used semiring):
#ifdef USE_TROPICAL_SEMIRING
  STORE(&f_k[i*C], x); // Just a store.
```
TROPICAL SEMIRING
```
#elif defined USE_BOOLEAN_SEMIRING
  // First, derive f_k using filtering.
  V g = LOAD(&g_{k-1}[i*C]); // Load the filter g_{k-1}.
  x = CMP(AND(x, g), [0,0,...0], NEQ); STORE(&x_k[i*C], x);

  // Second, update distances d; depth is the iteration number.
  V x_mask = x; x = MUL(x, [depth,...,depth]);
  x = BLEND(LOAD(&d[i*C]), x, x_mask); STORE(&d[i*C], x);

  // Third, update the filtering term.
  g = AND(NOT(x_mask), g); STORE(&g_k[i*C], g);
```
BOOLEAN SEMIRING
```
#elif defined USE_SELMAX_SEMIRING:
  // Update parents.
  V pars = LOAD(&p_{k-1}[i*C]); // Load the required part of p_{k-1}
  V pnz = CMP(pars, [0,0,...,0], NEQ);
  pars = BLEND([0,0,...,0], pars, pnz); STORE(&p_k[i*C], pars);

  // Set new x_k vector.
  V tmpnz = CMP(x, [0,0,...,0], NEQ);
  x = BLEND(x, &v[i*C], tmpnz); STORE(&x_k[i*C], x);
#endif
```
SEL-MAX SEMIRING

Sell-C-Sigma + Semirings Formulations



+

$(X, op_1, op_2, el_1, el_2)$

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$(\mathbb{R}, +, \cdot, 0, 1)$

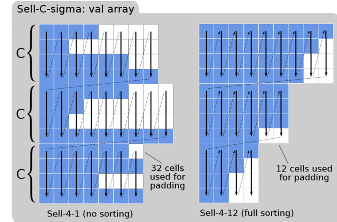
$(\{0,1\}, |, \&, 0, 1)$

$(\mathbb{R}, max, \cdot, -\infty, 1)$

```
    // Compute x_k (versions differ based on the used semiring):
#ifdef USE_TROPICAL_SEMIRING                                  TROPICAL SEMIRING
    x = MIN(ADD(rhs, vals), x);
#elif defined USE_BOOLEAN_SEMIRING                            BOOLEAN SEMIRING
    x = OR(AND(rhs, vals), x);
#elif defined USE_SELMAX_SEMIRING                             SEL-MAX SEMIRING
    x = MAX(MUL(rhs, vals), x);
#endif
    index += C;
  }
  // Now, derive f_k (versions differ based on the used semiring):
#ifdef USE_TROPICAL_SEMIRING                                  TROPICAL SEMIRING
  STORE(&f_k[i*C], x); // Just a store.
#elif defined USE_BOOLEAN_SEMIRING                            BOOLEAN SEMIRING
  // First, derive f_k using filtering.
  V g = LOAD(&g_{k-1}[i*C]); // Load the filter g_{k-1}.
  x = CMP(AND(x, g), [0,0,...0], NEQ); STORE(&x_k[i*C], x);

  // Second, update distances d; depth is the iteration number.
  V x_mask = x; x = MUL(x, [depth,...,depth]);
  x = BLEND(LOAD(&d[i*C]), x, x_mask); STORE(&d[i*C], x);

  // Third, update the filtering term.
  g = AND(NOT(x_mask), g); STORE(&g_k[i*C], g);
#elif defined USE_SELMAX_SEMIRING:                            SEL-MAX SEMIRING
  // Update parents.
  V pars = LOAD(&p_{k-1}[i*C]); // Load the required part of p_{k-1}
  V pnz = CMP(pars, [0,0,...,0], NEQ);
  pars = BLEND([0,0,...,0], pars, pnz); STORE(&p_k[i*C], pars);

  // Set new x_k vector.
  V tmpnz = CMP(x, [0,0,...,0], NEQ);
  x = BLEND(x, &v[i*C], tmpnz); STORE(&x_k[i*C], x);
#endif
```

Detailed formulations are in the paper ☺

Sell-C-Sigma + Semirings Formulations



+

$(X, op_1, op_2, el_1, el_2)$

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$(\mathbb{R}, +, \cdot, 0, 1)$

$(\{0,1\}, |, \&, 0, 1)$

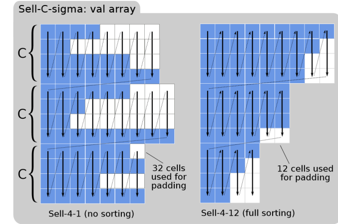
$(\mathbb{R}, max, \cdot, -\infty, 1)$

```
    // Compute x_k (versions differ based on the used semiring):
#ifdef USE_TROPICAL_SEMIRING                                    TROPICAL SEMIRING
    x = MIN(ADD(rhs, vals), x);
#elif defined USE_BOOLEAN_SEMIRING                              BOOLEAN SEMIRING
    x = OR(AND(rhs, vals), x);
#elif defined USE_SELMAX_SEMIRING                               SEL-MAX SEMIRING
    x = MAX(MUL(rhs, vals), x);
#endif
    index += C;
  }
  // Now, derive f_k (versions differ based on the used semiring):
#ifdef USE_TROPICAL_SEMIRING                                    TROPICAL SEMIRING
  STORE(&f_k[i*C], x); // Just a store.
#elif defined USE_BOOLEAN_SEMIRING                              BOOLEAN SEMIRING
  // First, derive f_k using filtering.
  V g = LOAD(&g_{k-1}[i*C]); // Load the filter g_{k-1}.
  x = CMP(AND(x, g), [0,0,...0], NEQ); STORE(&x_k[i*C], x);

  // Second, update distances d; depth is the iteration number.
  V x_mask = x; x = MUL(x, [depth,...,depth]);
  x = BLEND(LOAD(&d[i*C]), x, x_mask); STORE(&d[i*C], x);

  // Third, update the filtering term.
  g = AND(NOT(x_mask), g); STORE(&g_k[i*C], g);
#elif defined USE_SELMAX_SEMIRING:                              SEL-MAX SEMIRING
  // Update parents.
  V pars = LOAD(&p_{k-1}[i*C]); // Load the required part of p_{k-1}
  V pnz = CMP(pars, [0,0,...,0], NEQ);
  pars = BLEND([0,0,...,0], pars, pnz); STORE(&p_k[i*C], pars);

  // Set new x_k vector.
  V tmpnz = CMP(x, [0,0,...,0], NEQ);
  x = BLEND(x, &v[i*C], tmpnz); STORE(&x_k[i*C], x);
#endif
```

! What vector operations are required for each semiring when using Sell-C-sigma

! Detailed formulations are in the paper ☺

Sell-C-Sigma + Semirings Formulations



+

$(X, op_1, op_2, el_1, el_2)$

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$(\mathbb{R}, +, \cdot, 0, 1)$

$(\{0,1\}, |, \&, 0, 1)$

$(\mathbb{R}, max, \cdot, -\infty, 1)$

```
    // Compute x_k (versions differ based on the used semiring):
#ifdef USE_TROPICAL_SEMIRING
    x = MIN(ADD(rhs, vals), x);
#elif defined USE_BOOLEAN_SEMIRING
    x = OR(AND(rhs, vals), x);
#elif defined USE_SELMAX_SEMIRING
    x = MAX(MUL(rhs, vals), x);
#endif
    index += C;
  }
  // Now, derive f_k (versions differ based on the used semiring):
#ifdef USE_TROPICAL_SEMIRING
  STORE(&f_k[i*C], x); // Just a store.
#elif defined USE_BOOLEAN_SEMIRING
  // First, derive f_k using filtering.
  V g = LOAD(&g_{k-1}[i*C]); // Load the filter g_{k-1}.
  x = CMP(AND(x, g), [0,0,...0], NEQ); STORE(&x_k[i*C], x);

  // Second, update distances d; depth is the iteration number.
  V x_mask = x; x = MUL(x, [depth,...,depth]);
  x = BLEND(LOAD(&d[i*C]), x, x_mask); STORE(&d[i*C], x);

  // Third, update the filtering term.
  g = AND(NOT(x_mask), g); STORE(&g_k[i*C], g);
#elif defined USE_SELMAX_SEMIRING:
  // Update parents.
  V pars = LOAD(&p_{k-1}[i*C]); // Load the required part of p_{k-1}
  V pnz = CMP(pars, [0,0,...,0], NEQ);
  pars = BLEND([0,0,...,0], pars, pnz); STORE(&p_k[i*C], pars);

  // Set new x_k vector.
  V tmpnz = CMP(x, [0,0,...,0], NEQ);
  x = BLEND(x, &v[i*C], tmpnz); STORE(&x_k[i*C], x);
#endif
```

TROPICAL SEMIRING

BOOLEAN SEMIRING

SEL-MAX SEMIRING

TROPICAL SEMIRING

BOOLEAN SEMIRING

SEL-MAX SEMIRING

! What vector operations are required for each semiring when using Sell-C-sigma

! Detailed formulations are in the paper ☺

Graph Representations

Computational Complexity

GRAPH REPRESENTATIONS
COMPUTATIONAL COMPLEXITY

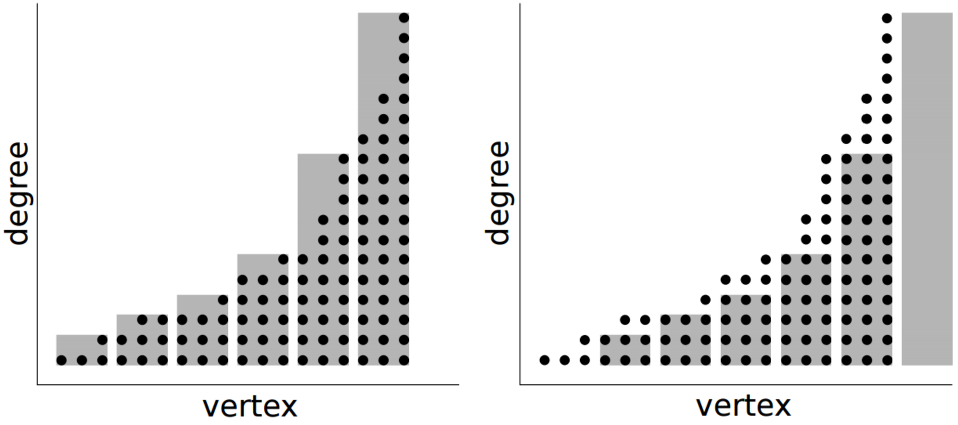
0

- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

GRAPH REPRESENTATIONS
COMPUTATIONAL COMPLEXITY

0

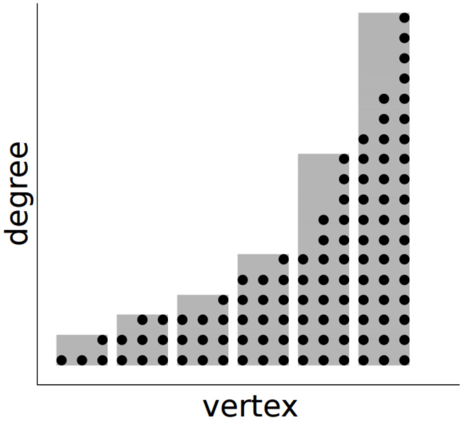
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

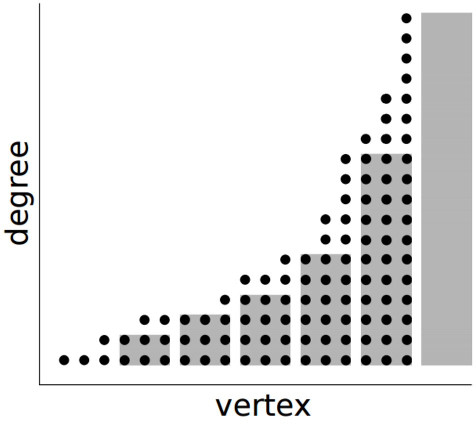


Graph Representations
Computational Complexity

0
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

1
The size of all the blocks (except the largest):



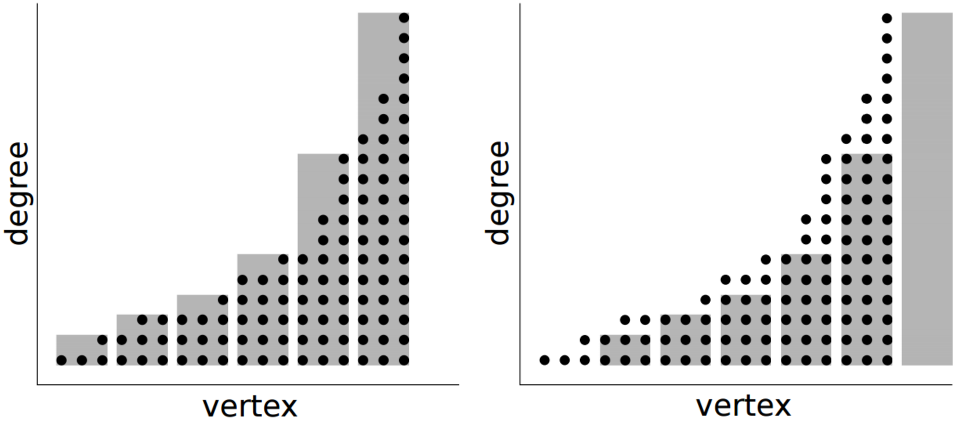


GRAPH REPRESENTATIONS
COMPUTATIONAL COMPLEXITY

0
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

1
The size of all the blocks (except the largest):

$$\sum_{i=2}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m$$



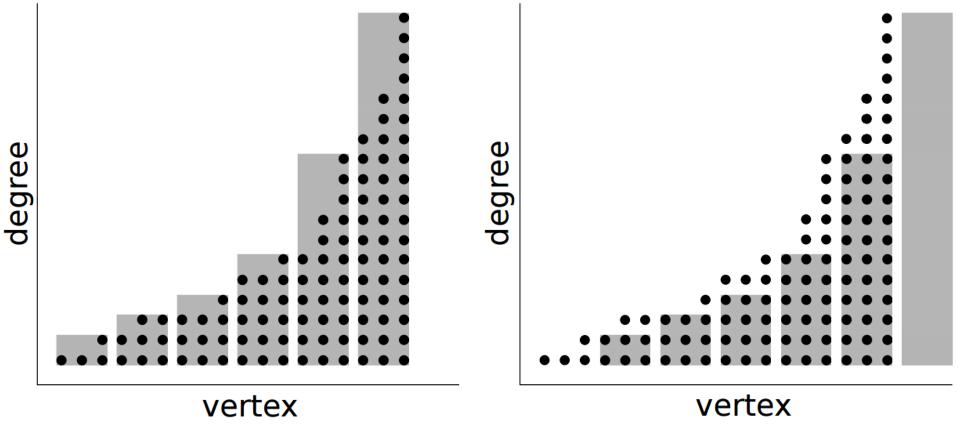
Graph Representations
Computational Complexity

0
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

1
The size of all the blocks (except the largest):

$$\sum_{i=2}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m$$

2
The size of the largest block:

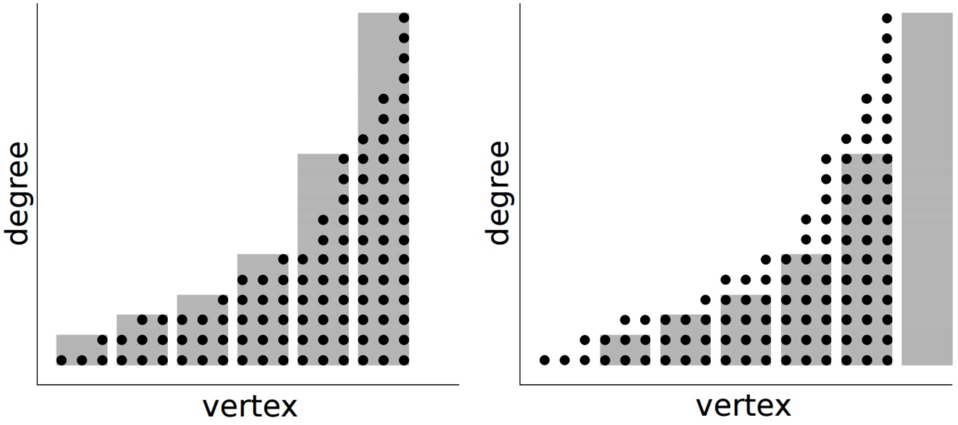


Graph Representations
Computational Complexity

0
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

1
The size of all the blocks (except the largest):

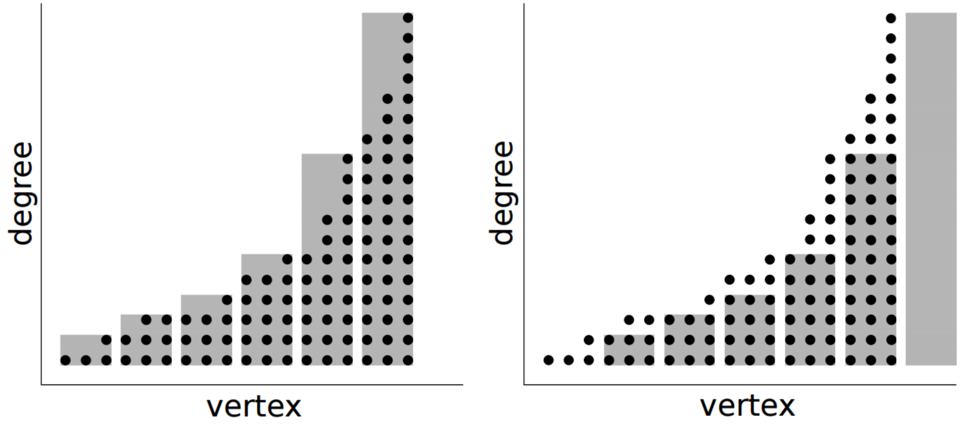
$$\sum_{i=2}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m$$



2
The size of the largest block: $\hat{\rho}C$

Graph Representations
Computational Complexity

0
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring



1 The size of all the blocks (except the largest):

$$\sum_{i=2}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m$$

2 The size of the largest block: $\hat{\rho} C$

3 **Storage bound**

Graph Representations
Computational Complexity

0
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

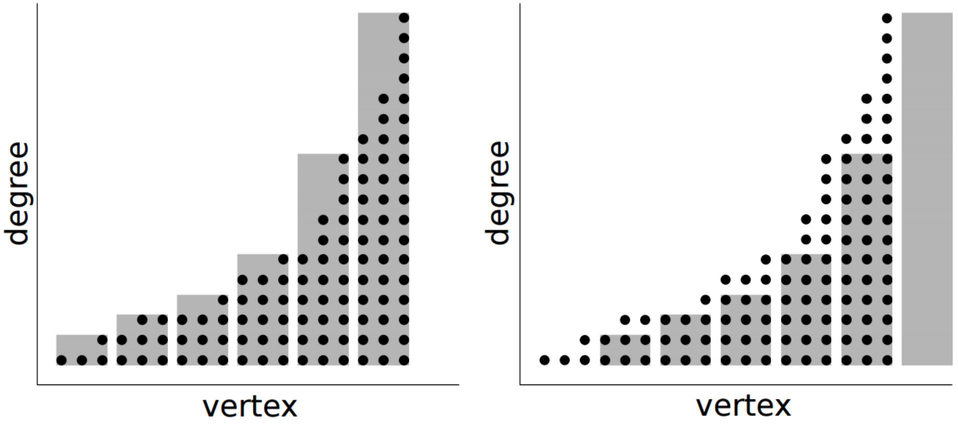
1 The size of all the blocks (except the largest):

$$\sum_{i=2}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m$$

2 The size of the largest block: $\hat{\rho}C$

3 **Storage bound**

$$\sum_{i=1}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m + \hat{\rho}C$$

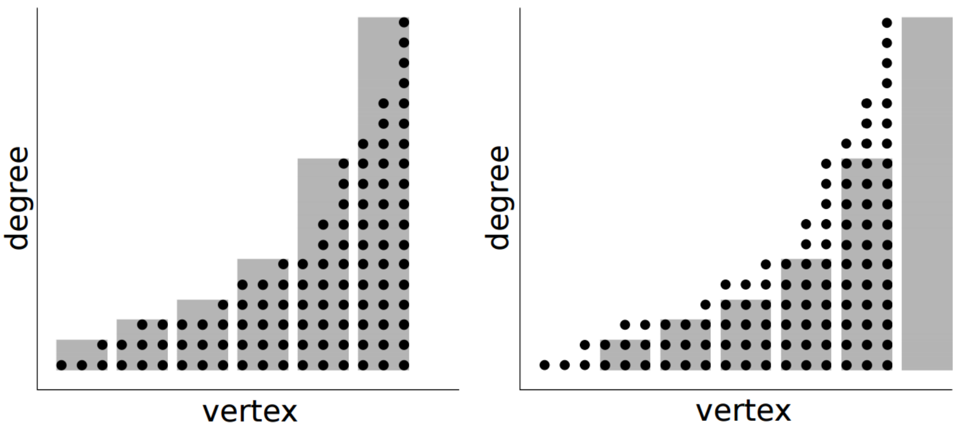


Graph Representations
Computational Complexity

0
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

1
The size of all the blocks (except the largest):

$$\sum_{i=2}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m$$



2
The size of the largest block: $\hat{\rho}C$

3
Storage bound

$$\sum_{i=1}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m + \hat{\rho}C$$

4
Computational complexity bound

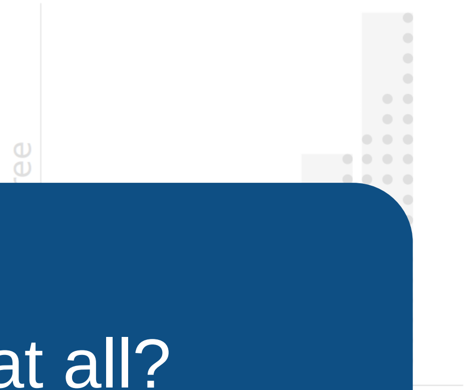
$$W = O(D(n + m + \hat{\rho}C))$$
$$= O(Dn + Dm + D\hat{\rho}C)$$

Graph Representations
Computational Complexity

0
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

1 The size of all the blocks (except the largest):

$$\sum_{i=2}^{\#chunks} C$$



2 The size of the larg

Is that all?

3 **Storage bound**

$$\sum_{i=1}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m + \hat{\rho}C$$

4 **Computational complexity bound**

$$W = O(D(n + m + \hat{\rho}C))$$
$$= O(Dn + Dm + D\hat{\rho}C)$$

Graph Representations
Computational Complexity

0
- Vertices are sorted by their degree
- ρ_i : the degree of the ith vertex
- $\hat{\rho}$: the maximum degree
- Assume tropical semiring

1 The size of all the blocks (except the largest):

$$\sum_{i=2}^{\#chunks} C$$

2 The size of the larg

Is that all?

3 **Storage bound**

$$\sum_{i=1}^{\#chunks} C \cdot \rho_{iC-1} \leq 2m + \hat{\rho}C$$

4 **Computat...und**

$$W = O(D$$

$$= O(Dn + Dm + D\hat{\rho}C)$$

Not really...

SlimSell

Reducing Storage Overheads

SlimSell

Reducing Storage Overheads

Sell-4-12

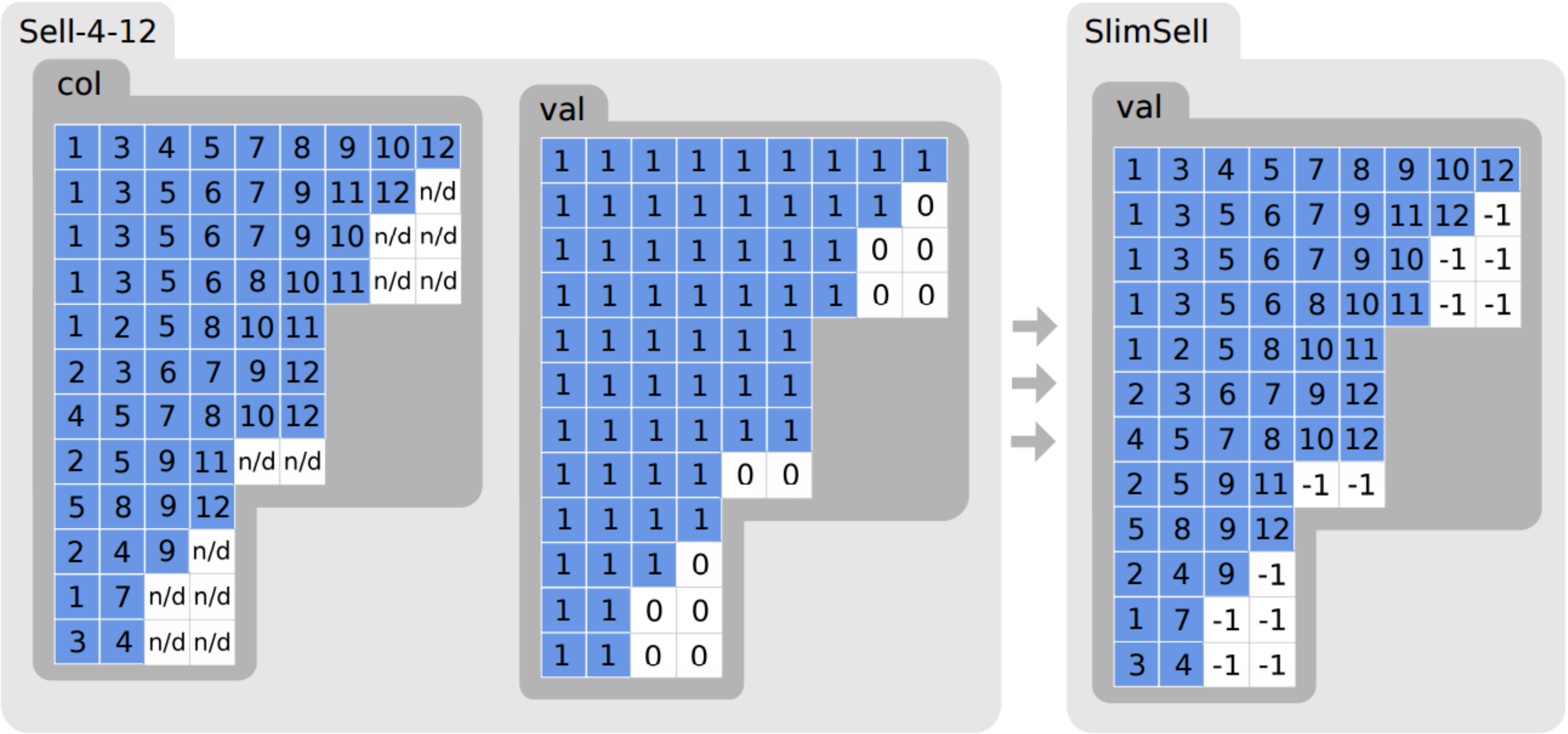
col

1	3	4	5	7	8	9	10	12
1	3	5	6	7	9	11	12	n/d
1	3	5	6	7	9	10	n/d	n/d
1	3	5	6	8	10	11	n/d	n/d
1	2	5	8	10	11			
2	3	6	7	9	12			
4	5	7	8	10	12			
2	5	9	11	n/d	n/d			
5	8	9	12					
2	4	9	n/d					
1	7	n/d	n/d					
3	4	n/d	n/d					

val

1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	0	0
1	1	1	1	1	1			
1	1	1	1	1	1			
1	1	1	1	1	1			
1	1	1	1	0	0			
1	1	1	1					
1	1	1	0					
1	1	0	0					
1	1	0	0					

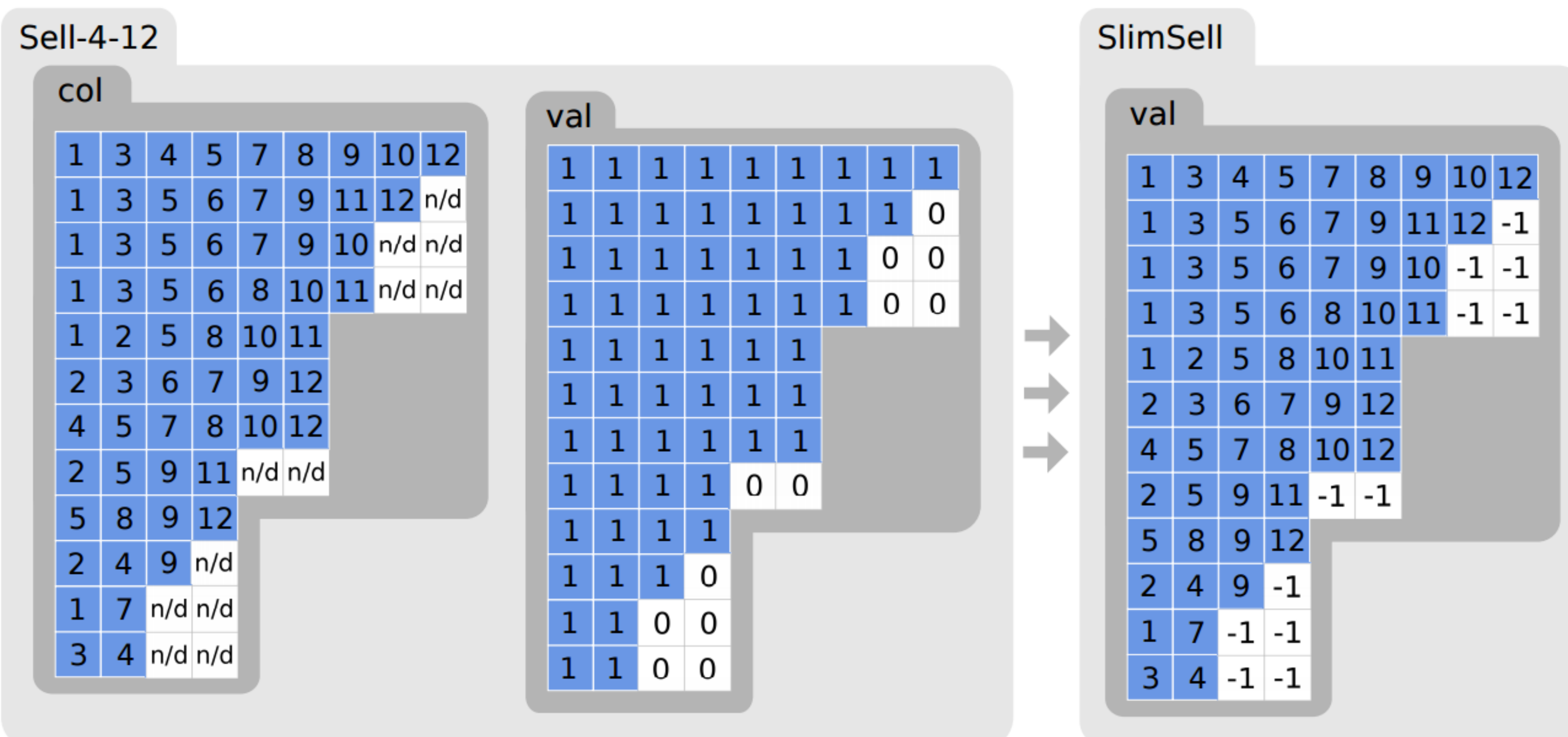
SlimSell
Reducing Storage Overheads



SlimSell

Reducing Storage Overheads

Representation	Sell-C-σ	CSR	AL	SlimSell
Size [cells]	$4m + \frac{2n}{C} + P$	$4m + n$	$2m + n$	$2m + \frac{2n}{C} + P$



SlimSell
Reducing Storage Overheads

SlimSell
Reducing Storage Overheads

Representation	Sell-C-σ	CSR	AL	SlimSell
Size [cells]	$4m + \frac{2n}{C} + P$	$4m + n$	$2m + n$	$2m + \frac{2n}{C} + P$

SlimSell
Reducing Storage Overheads

Representation	Sell-C-σ	CSR	AL	SlimSell
Size [cells]	$4m + \frac{2n}{C} + P$	$4m + n$	$2m + n$	$2m + \frac{2n}{C} + P$

$$2m + \frac{2n}{C} + P < n + 2m \Leftrightarrow P < n\left(1 - \frac{2}{C}\right)$$

SlimSell
Reducing Storage Overheads

Representation	Sell-C-σ	CSR	AL	SlimSell
Size [cells]	$4m + \frac{2n}{C} + P$	$4m + n$	$2m + n$	$2m + \frac{2n}{C} + P$

$$2m + \frac{2n}{C} + P < n + 2m \Leftrightarrow P < n\left(1 - \frac{2}{C}\right)$$

$C = 8$

$P < 3n/4$

$C = 16$

$P < 7n/8$

$C = 32$

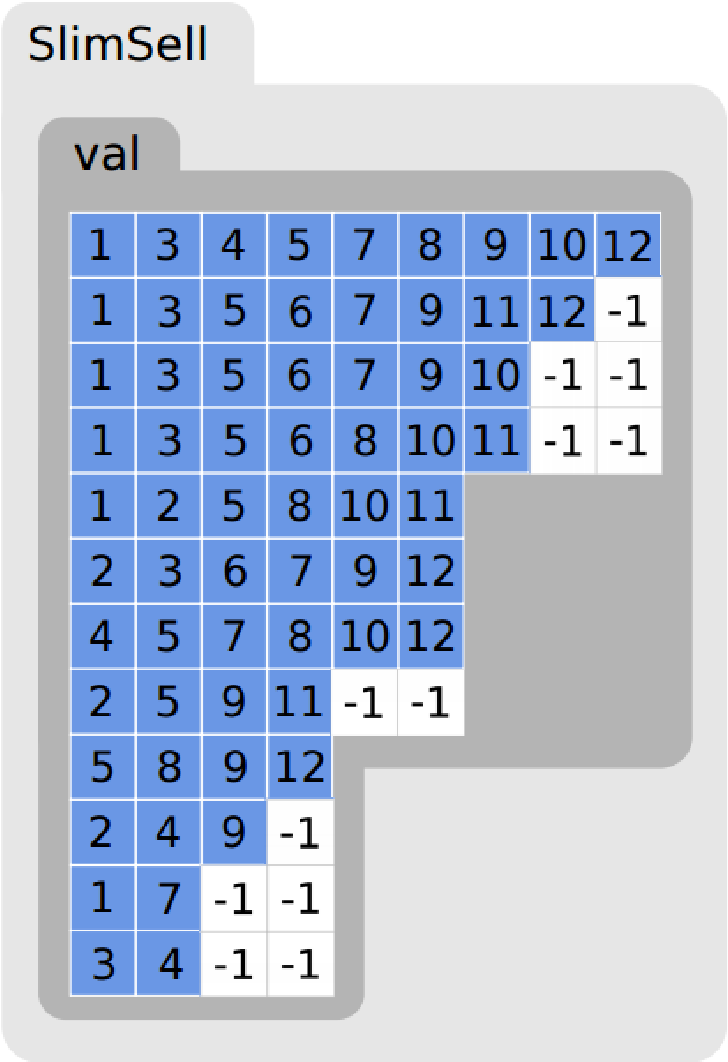
$P < 15n/16$

SlimSell

Further Optimizations: SlimWork

SlimSell

Further Optimizations: SlimWork



SlimSell

Further Optimizations: SlimWork

SlimSell

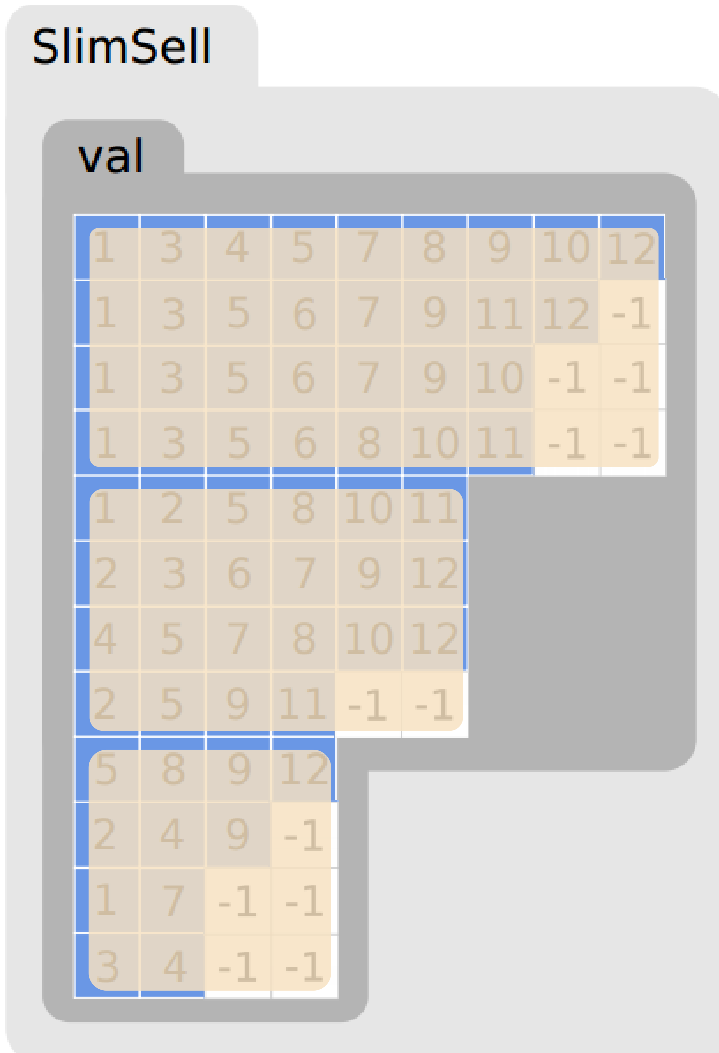
val

1	3	4	5	7	8	9	10	12
1	3	5	6	7	9	11	12	-1
1	3	5	6	7	9	10	-1	-1
1	3	5	6	8	10	11	-1	-1
1	2	5	8	10	11			
2	3	6	7	9	12			
4	5	7	8	10	12			
2	5	9	11	-1	-1			
5	8	9	12					
2	4	9	-1					
1	7	-1	-1					
3	4	-1	-1					

The corresponding traversal is label-setting, so...

SlimSell

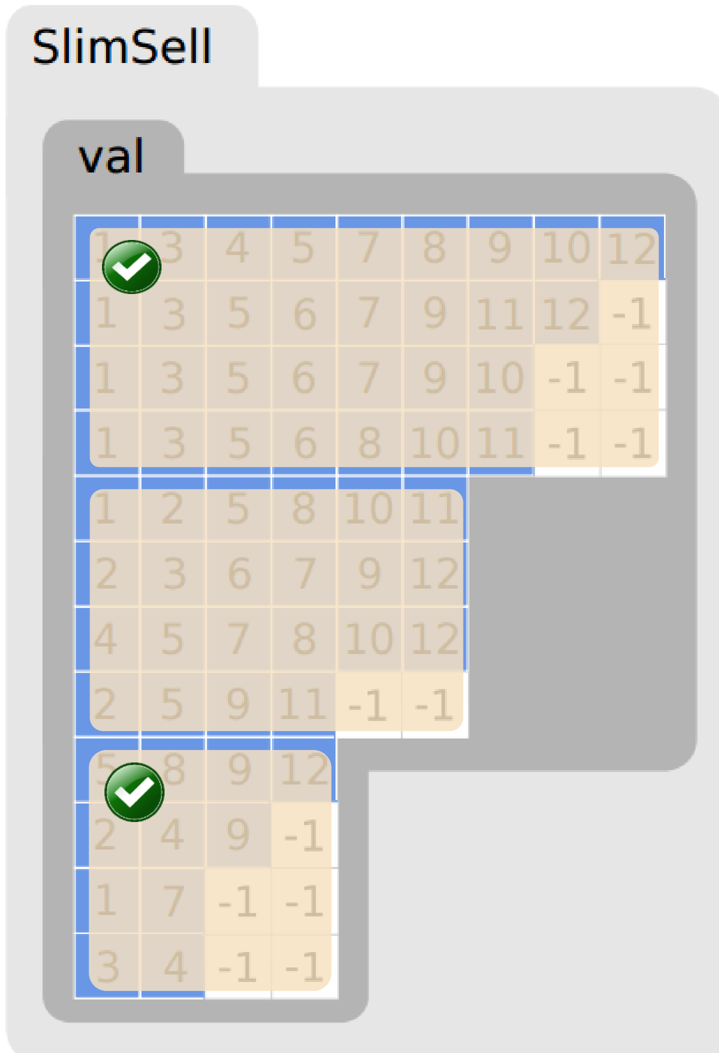
Further Optimizations: SlimWork



The corresponding traversal is label-setting, so...

SlimSell

Further Optimizations: SlimWork



The corresponding traversal is label-setting, so...

SlimSell

Further Optimizations: SlimChunk

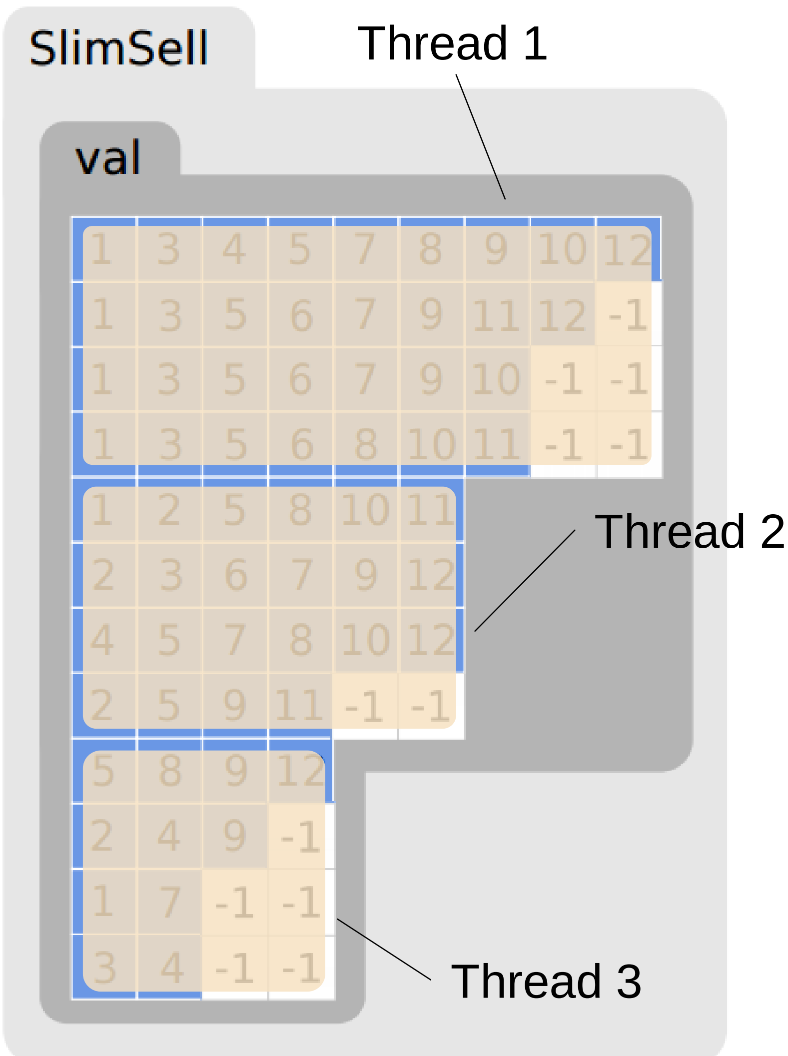
SlimSell

val

1	3	4	5	7	8	9	10	12
1	3	5	6	7	9	11	12	-1
1	3	5	6	7	9	10	-1	-1
1	3	5	6	8	10	11	-1	-1
1	2	5	8	10	11			
2	3	6	7	9	12			
4	5	7	8	10	12			
2	5	9	11	-1	-1			
5	8	9	12					
2	4	9	-1					
1	7	-1	-1					
3	4	-1	-1					

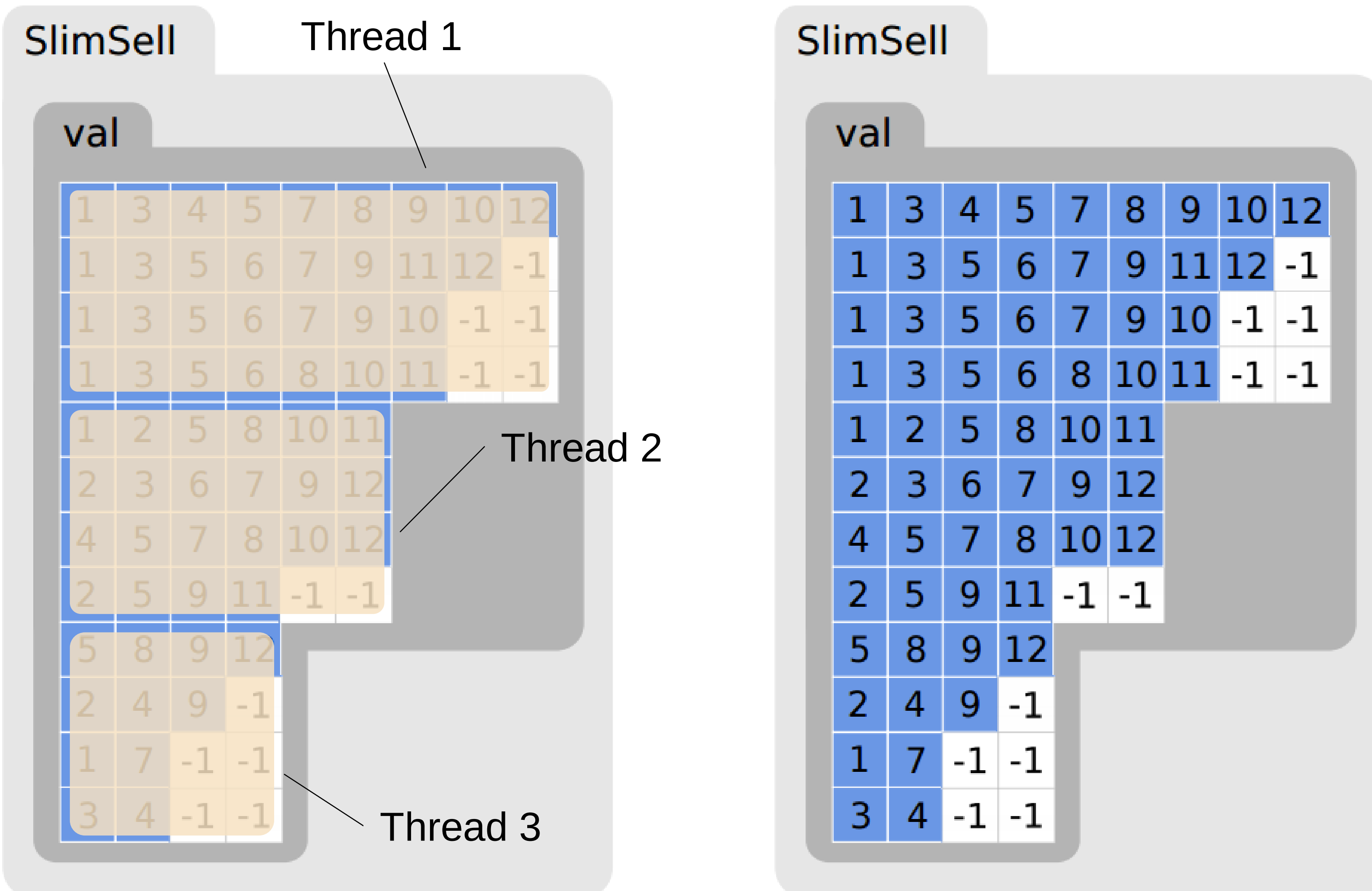
SlimSell

Further Optimizations: SlimChunk



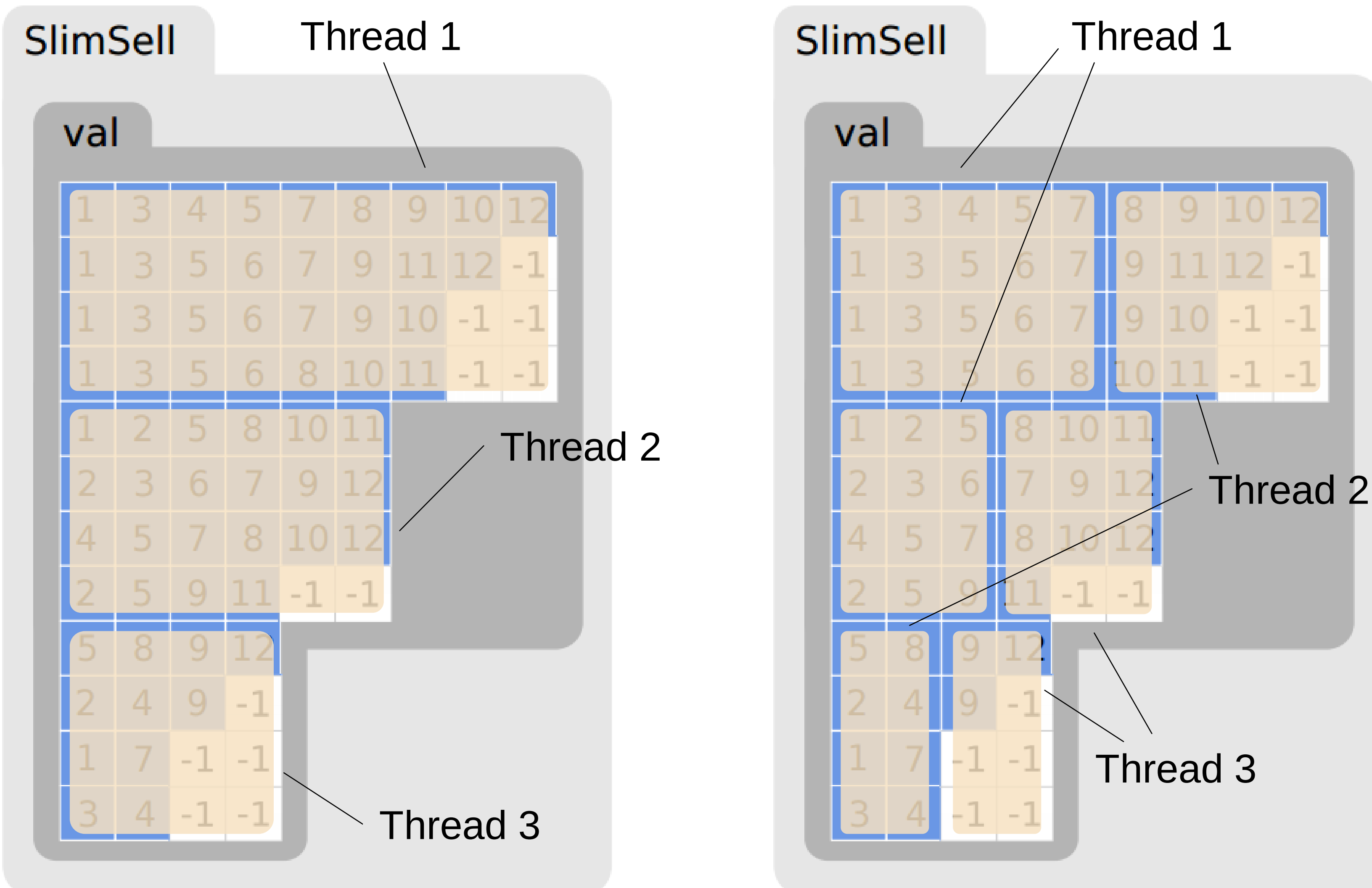
SLIMSELL

FURTHER OPTIMIZATIONS: SLIMCHUNK



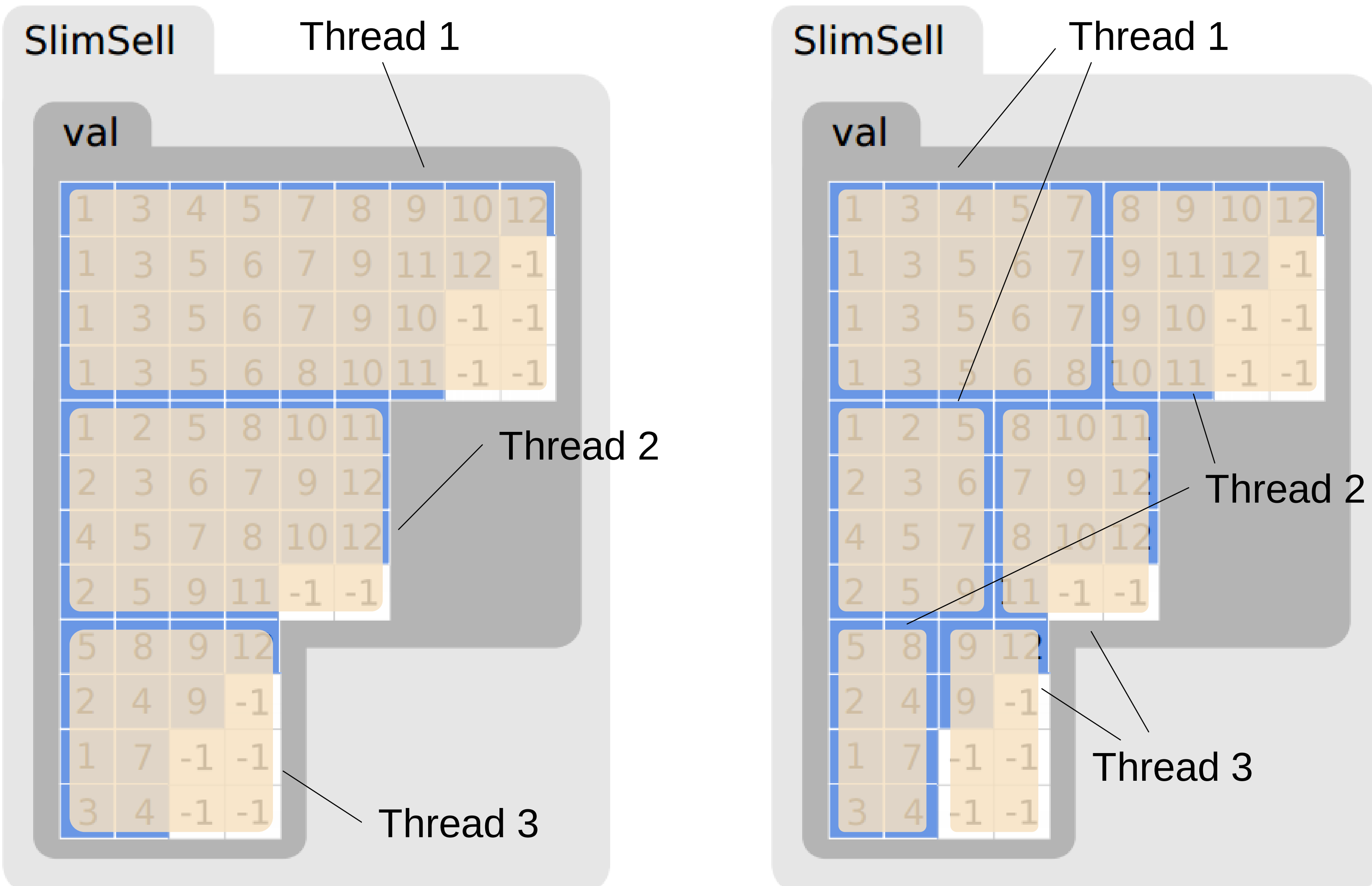
SlimSell

Further Optimizations: SlimChunk



SlimSell
Further Optimizations: SlimChunk

Additional reduction required



PERFORMANCE QUESTIONS

PERFORMANCE QUESTIONS

Does using semirings result in different performance?

Performance Questions

Does using semirings result in different performance?

What is the impact of various parameters (e.g., thread scheduling)?

Performance Questions

Does using semirings result in different performance?

What is the impact of various parameters (e.g., thread scheduling)?

What are storage and performance improvements from SlimSell?

Performance Analysis

Types of machines

Performance Analysis
Types of machines


CSCS Greina cluster


Trivium Intel Server


CSCS Piz Daint & Piz Dora

Performance Analysis
Types of machines

Performance Analysis
Types of machines

Performance Analysis
Types of machines

Performance Analysis

Types of graphs

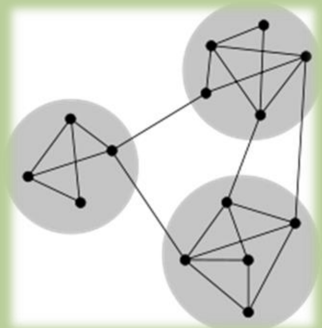
Performance Analysis
Types of graphs

Synthetic graphs

PERFORMANCE ANALYSIS
TYPES OF GRAPHS

Synthetic graphs

Kronecker [1]

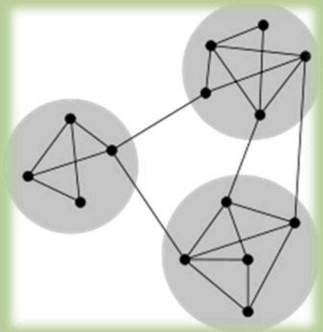


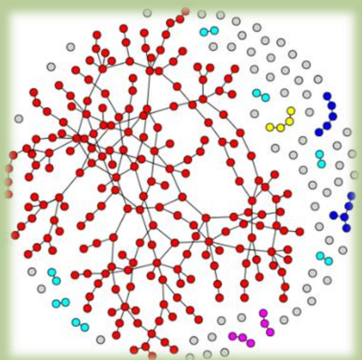
[1] J. Leskovec et al. Kronecker Graphs: An Approach to Modeling Networks. J. Mach. Learn. Research. 2010.

Performance Analysis
Types of graphs

Synthetic graphs

Kronecker [1]





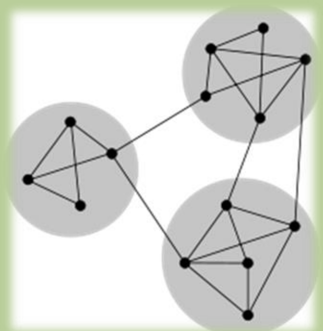
Erdös-Rényi [2]

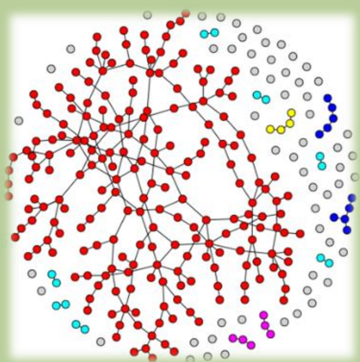
[1] J. Leskovec et al. Kronecker Graphs: An Approach to Modeling Networks. J. Mach. Learn. Research. 2010.
[2] P. Erdos and A. Renyi. On the evolution of random graphs. Pub. Math. Inst. Hun. A. Science. 1960.

PERFORMANCE ANALYSIS
TYPES OF GRAPHS

Synthetic graphs

Kronecker [1]





Erdös-Rényi [2]

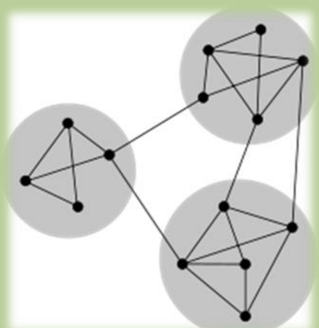
Real-world SNAP graphs [3]

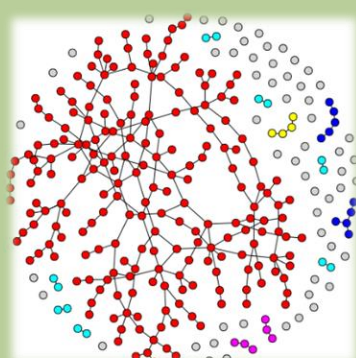
[1] J. Leskovec et al. Kronecker Graphs: An Approach to Modeling Networks. J. Mach. Learn. Research. 2010.
[2] P. Erdos and A. Renyi. On the evolution of random graphs. Pub. Math. Inst. Hun. A. Science. 1960.
[3] https://snap.stanford.edu

PERFORMANCE ANALYSIS
TYPES OF GRAPHS

Synthetic graphs

Kronecker [1]





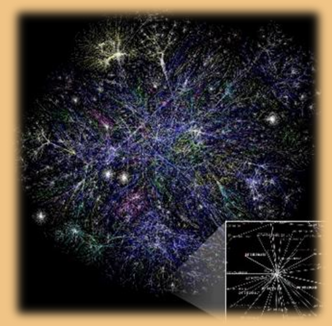
Erdös-Rényi [2]

Real-world SNAP graphs [3]

Road networks

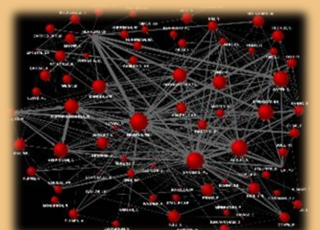
Social networks

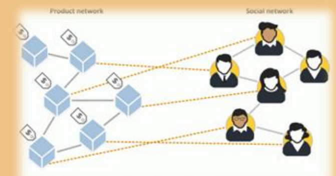
Comm. graphs



Web graphs

Citation graphs





Purchase networks

[1] J. Leskovec et al. Kronecker Graphs: An Approach to Modeling Networks. J. Mach. Learn. Research. 2010.
[2] P. Erdos and A. Renyi. On the evolution of random graphs. Pub. Math. Inst. Hun. A. Science. 1960.
[3] https://snap.stanford.edu

Performance Analysis

Other Parameters

Performance Analysis
Other Parameters

Semirings

Tropical: $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

Real: $(\mathbb{R}, +, \cdot, 0, 1)$

Boolean: $(\{0,1\}, |, \&, 0, 1)$

Sel-max: $(\mathbb{R}, max, \cdot, -\infty, 1)$

Performance Analysis
Other Parameters

OpenMP scheduling

Static
Dynamic

Semirings

Tropical: $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$
Real: $(\mathbb{R}, +, \cdot, 0, 1)$
Boolean: $(\{0,1\}, |, \&, 0, 1)$
Sel-max: $(\mathbb{R}, max, \cdot, -\infty, 1)$

Performance Analysis
Other Parameters

OpenMP scheduling

Static
Dynamic

Semirings

Tropical: $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$
Real: $(\mathbb{R}, +, \cdot, 0,1)$
Boolean: $(\{0,1\}, |, \&, 0,1)$
Sel-max: $(\mathbb{R}, max, \cdot, -\infty, 1)$

Scaling

Strong
Weak

Performance Analysis
Other Parameters

OpenMP scheduling

Static
Dynamic

Semirings

Tropical: $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$
Real: $(\mathbb{R}, +, \cdot, 0, 1)$
Boolean: $(\{0,1\}, |, \&, 0, 1)$
Sel-max: $(\mathbb{R}, max, \cdot, -\infty, 1)$

Scaling

Strong
Weak

Sell-C-sigma parameters

Sorting
Chunk size

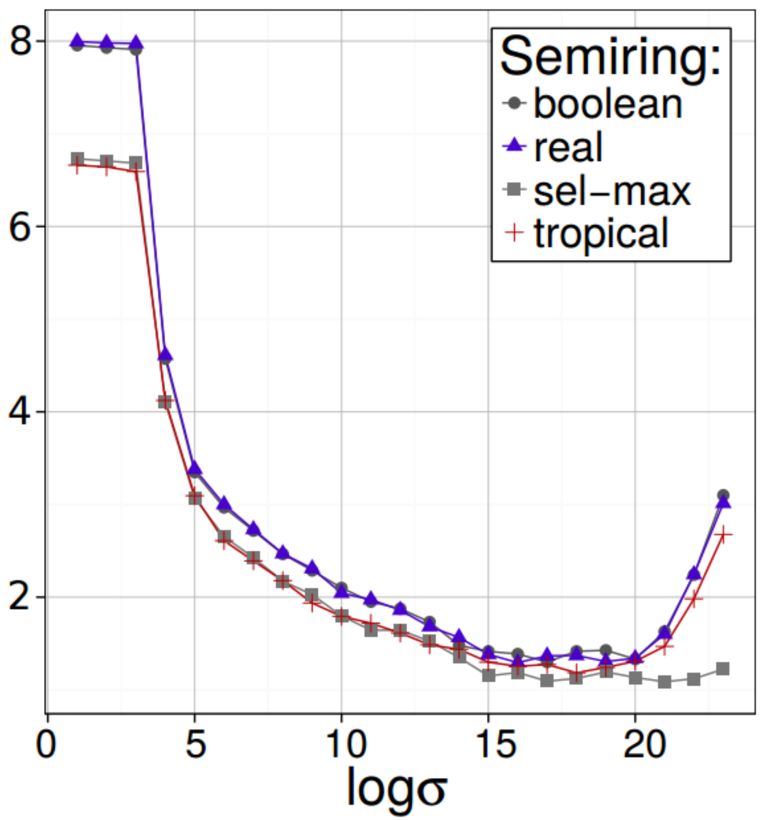
Performance Analysis
Semiring Comparison

Kronecker power-law graphs

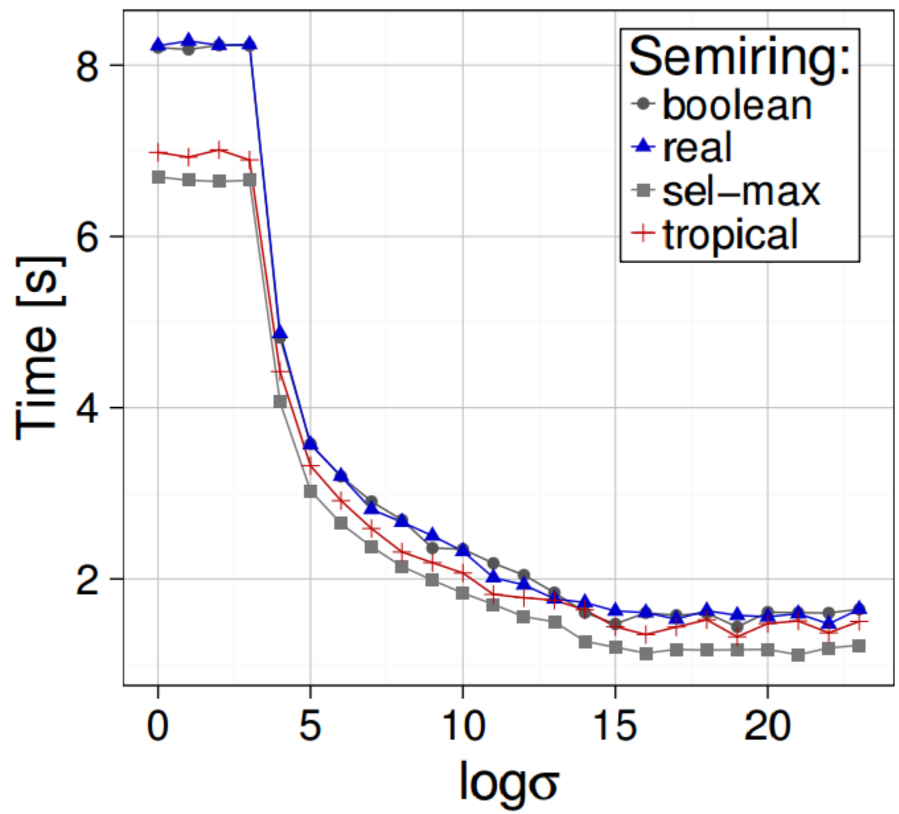
$n = 2^{23}, \bar{\rho} = 16$

Xeon CPU, $C = 8$

Static scheduling



Dynamic scheduling



Performance Analysis
Impact from SlimWork

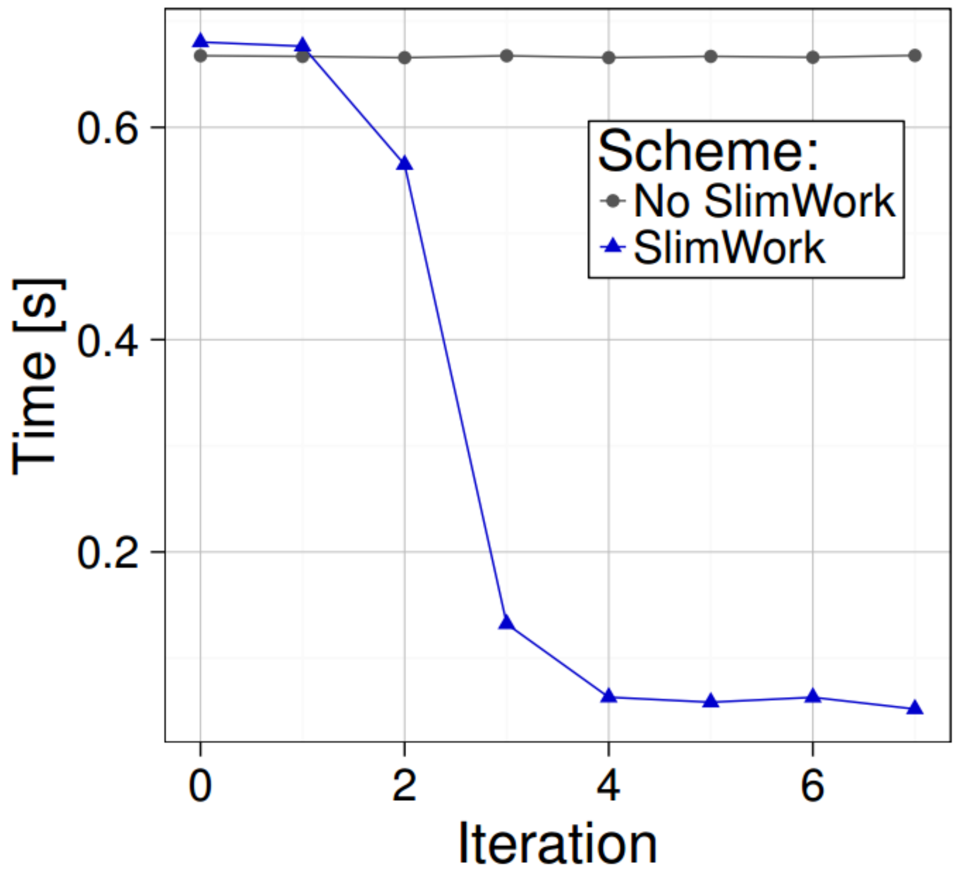
Kronecker power-law graphs

$n = 2^{23}, \bar{\rho} = 16$

Xeon CPU, $C = 8$

$\log \sigma = 23$

Dynamic scheduling



Performance Analysis
Impact from SlimChunk

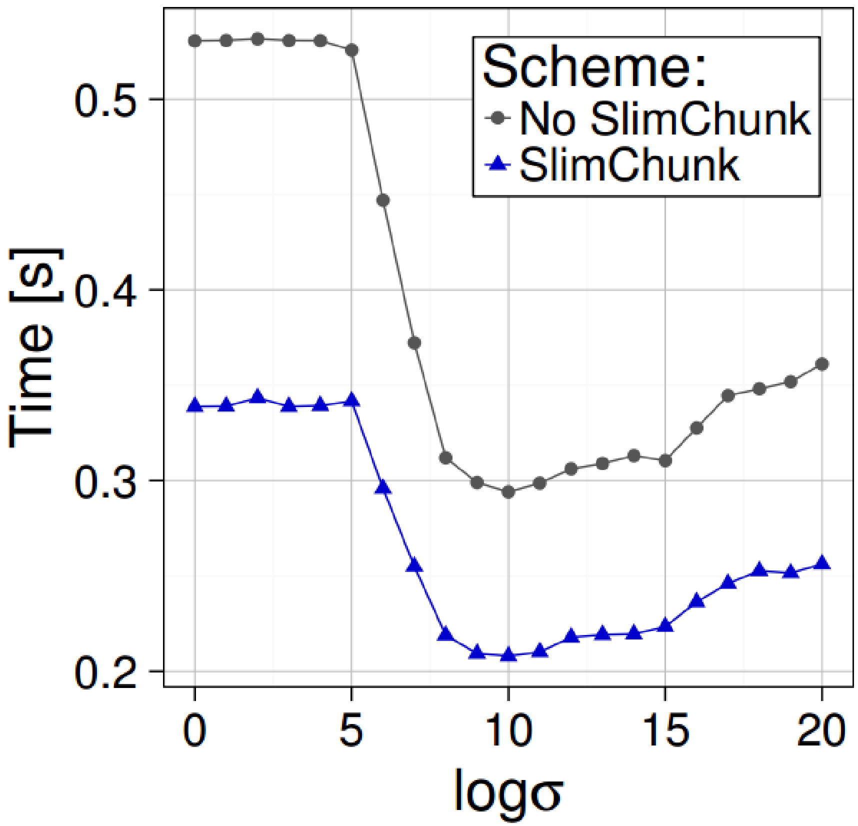
Kronecker power-law graphs

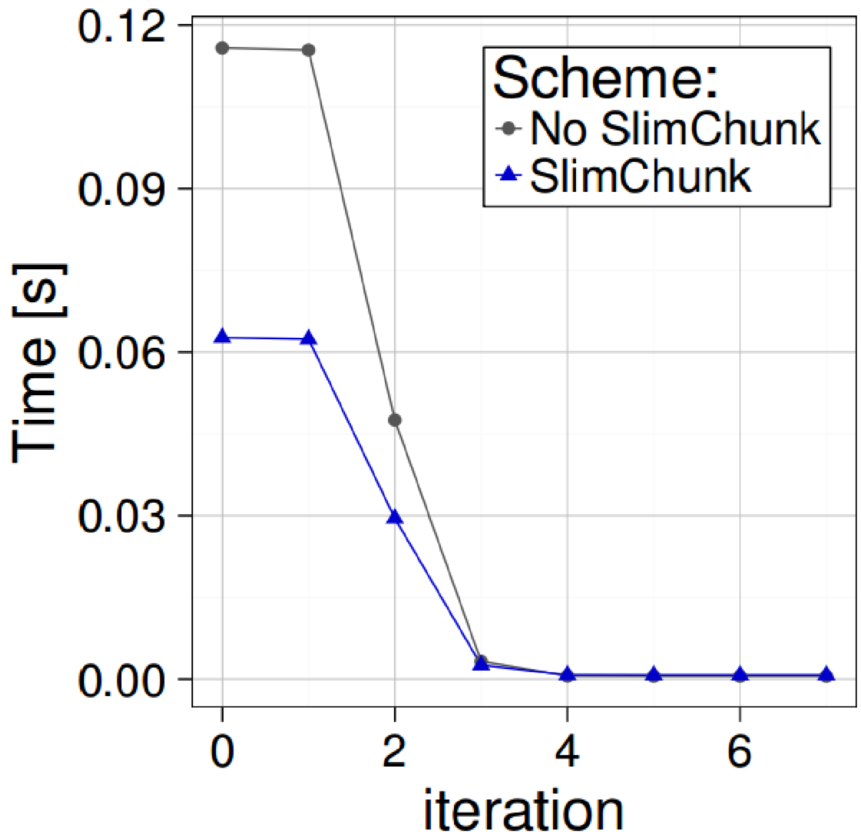
$n = 2^{20}, \bar{\rho} = 16$

Tesla K80 GPU, $C = 32$

$\log \sigma = 20$

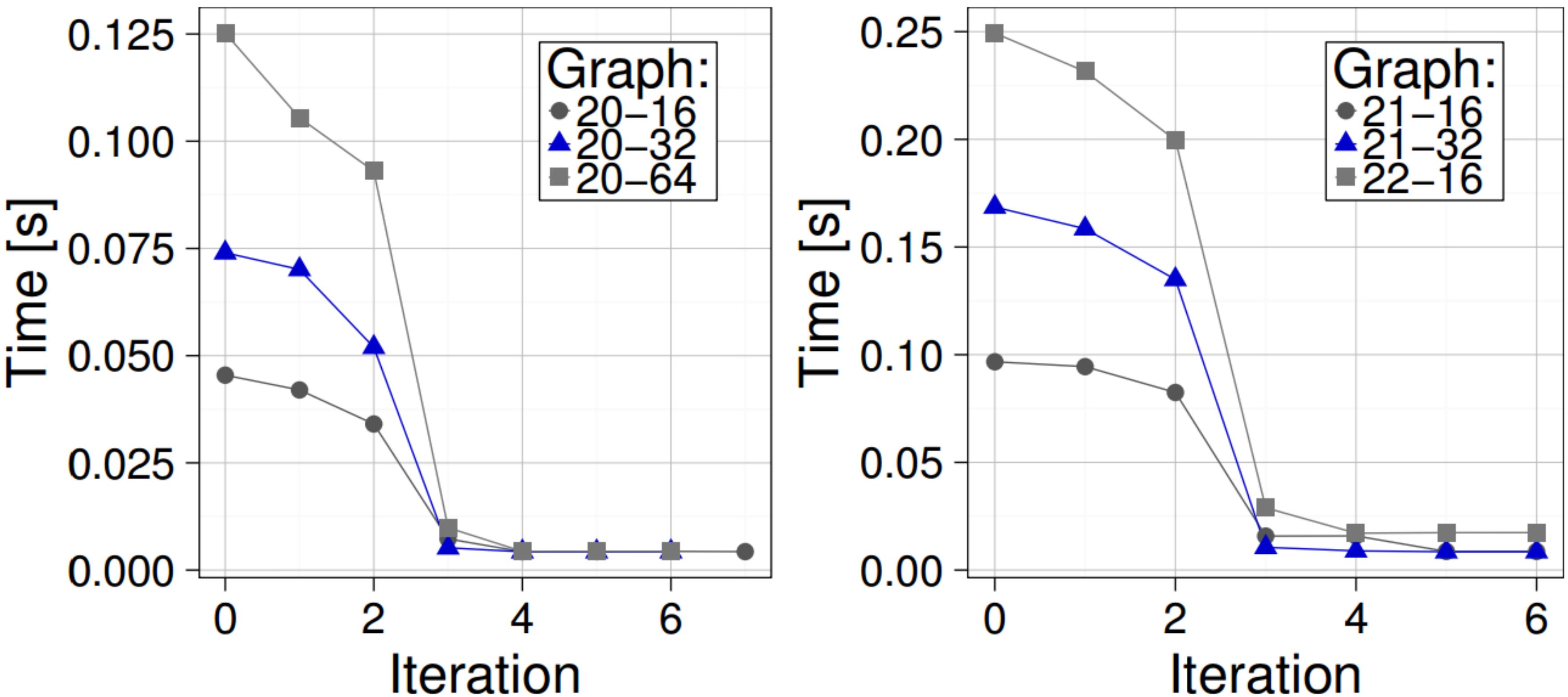
Dynamic scheduling





Performance Analysis
KNL Analysis

Kronecker power-law graphs
Intel KNL, $C = 16$
$\log \sigma \in \{20,21,22\}$
Dynamic scheduling



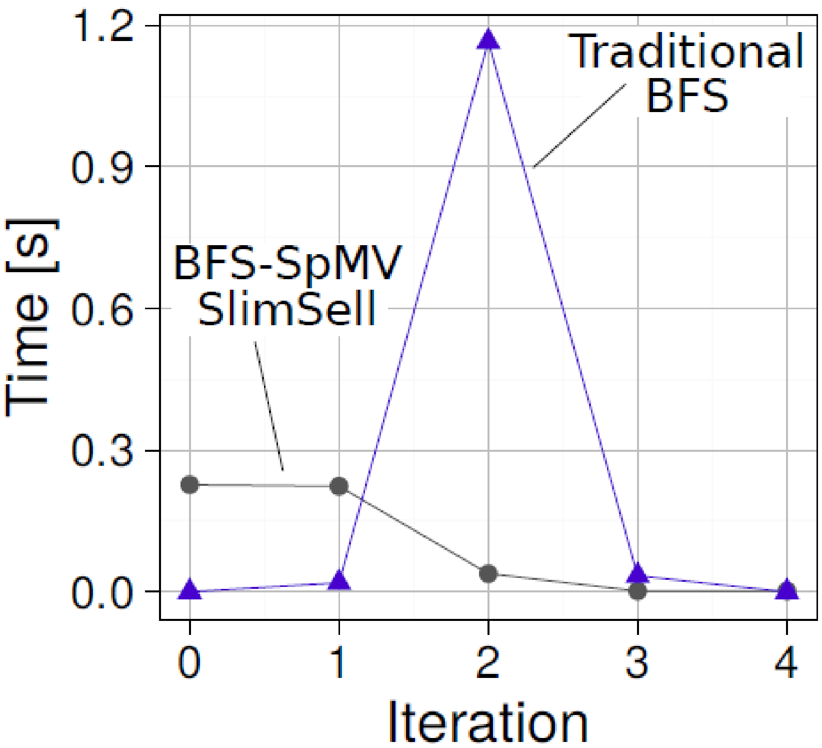
Performance Analysis
Comparison to Graph500

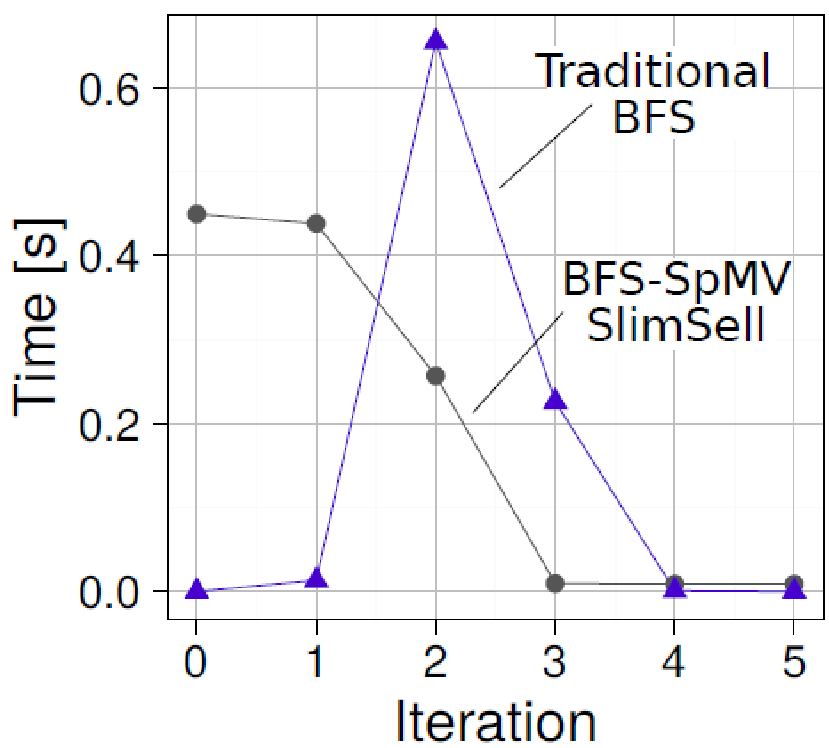
Kronecker power-law graphs

Intel KNL, $C = 16$

$\log \sigma \in \{20,21,22\}$

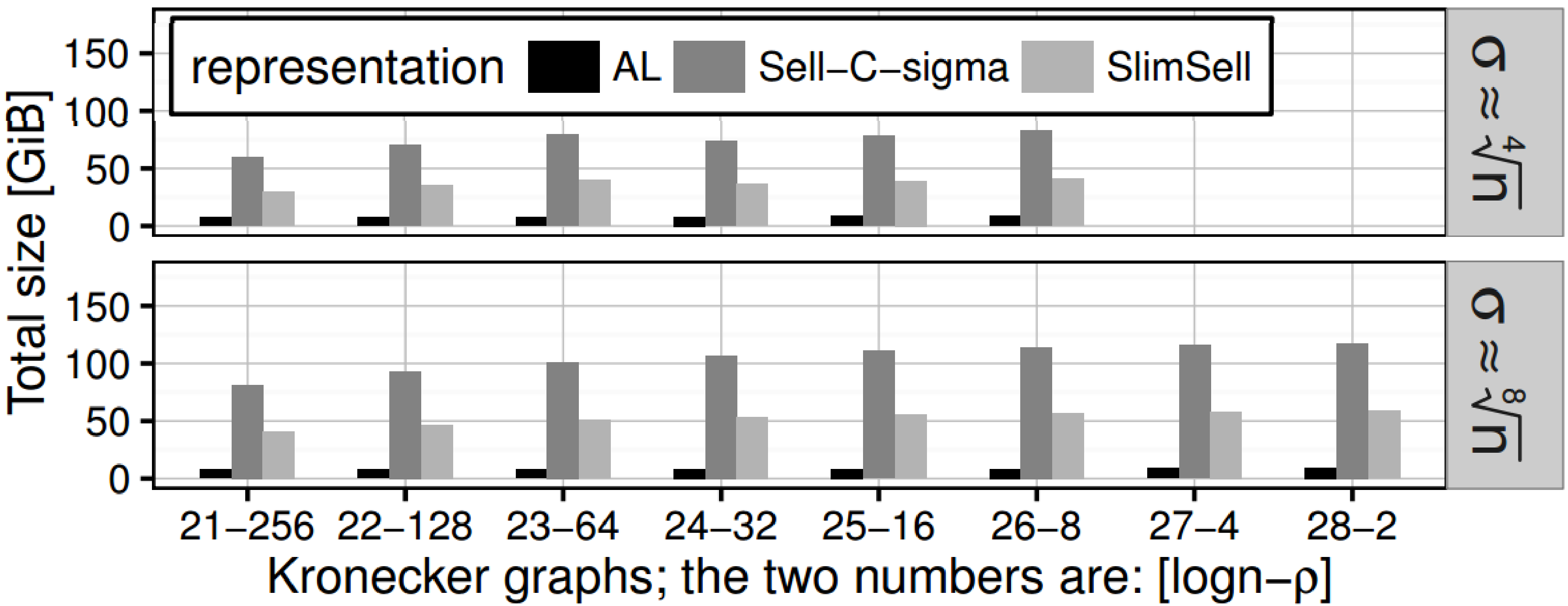
Dynamic scheduling





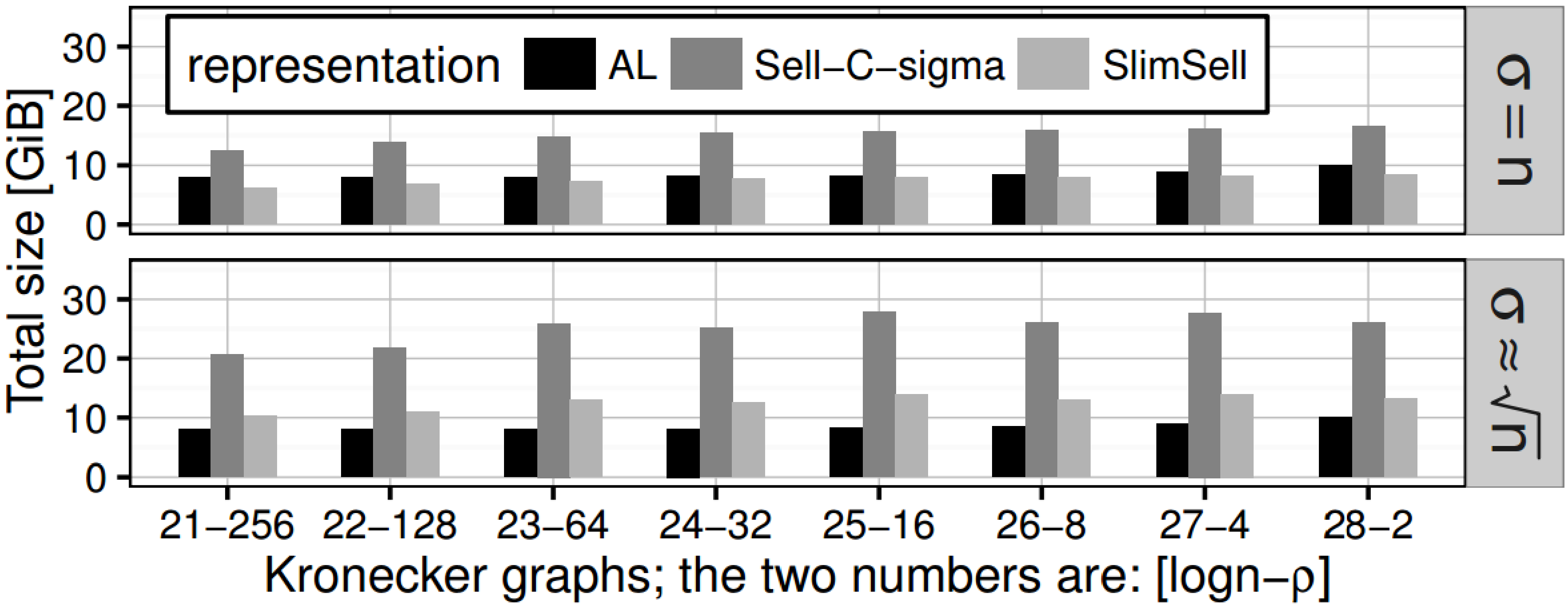
Performance Analysis
Size Analysis

Kronecker power-law graphs



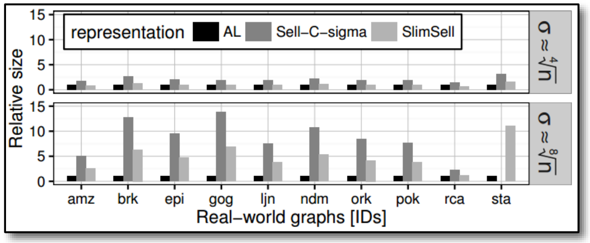
PERFORMANCE ANALYSIS
SIZE ANALYSIS

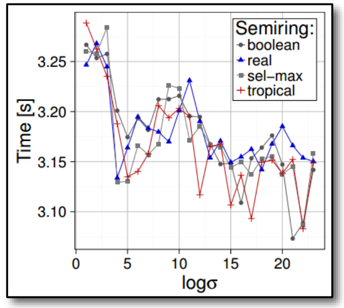
Kronecker power-law graphs

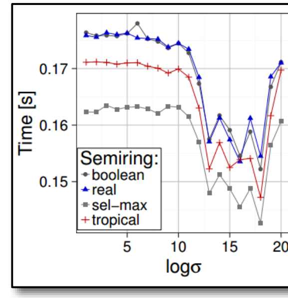


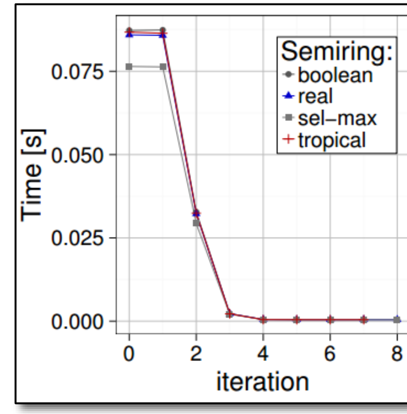
OTHER ANALYSES

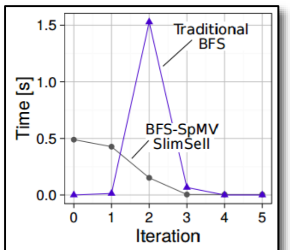
Other Analyses

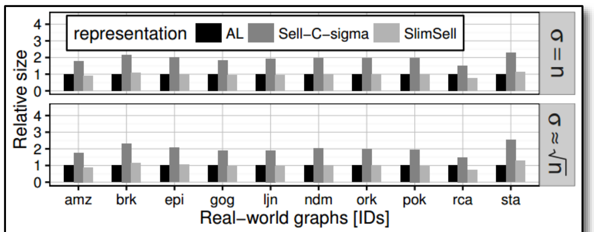


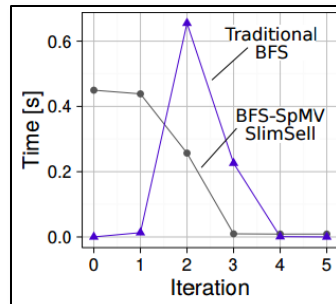


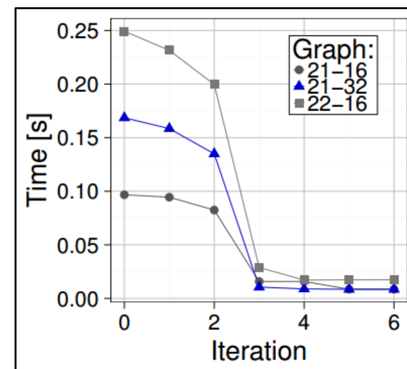


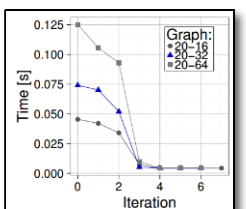


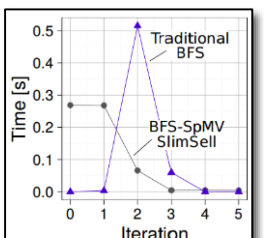


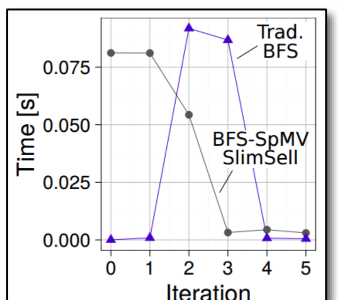




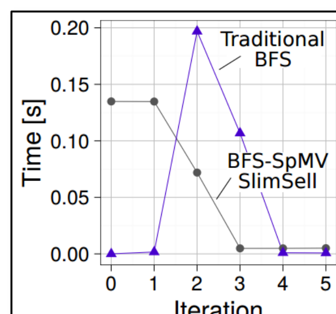


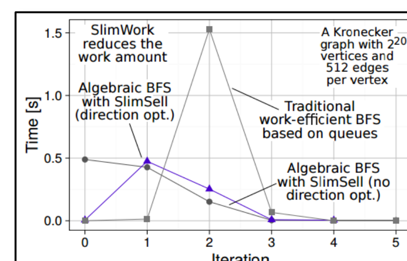


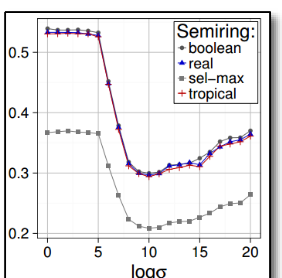


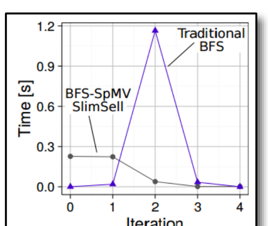


σ	Boolean	Real	Tropical	Sel-max
2^4	1.17	1.17	1.21	1.18
2^{18}	1.00	1.04	1.04	1.03





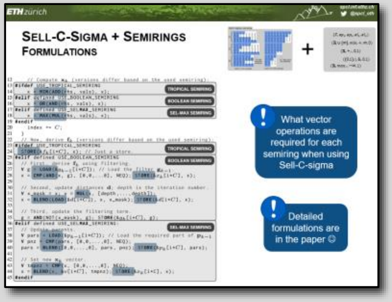


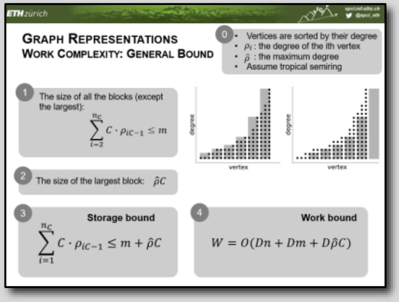


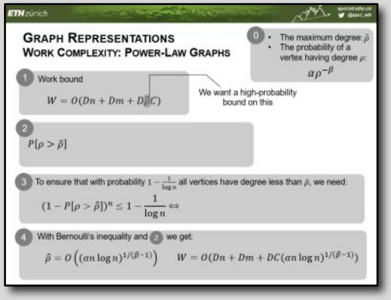
CONCLUSIONS

Conclusions

Sell-C-sigma for graphs

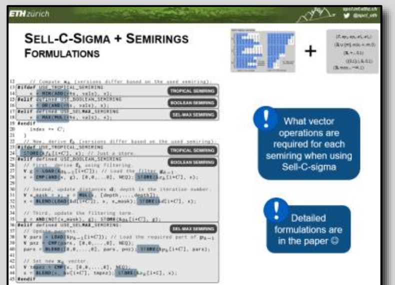


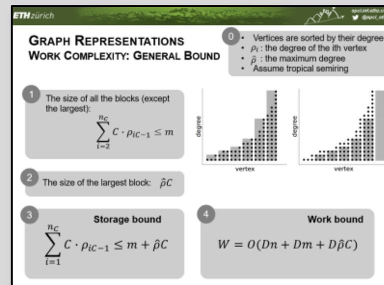


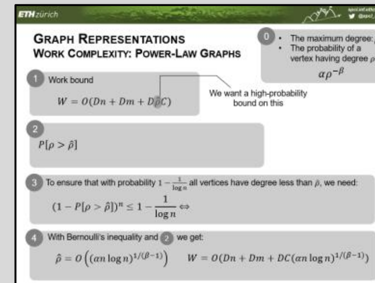


Conclusions

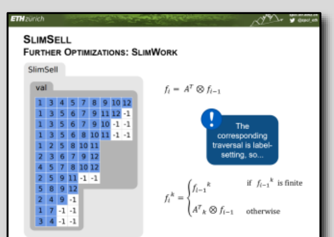
Sell-C-sigma for graphs

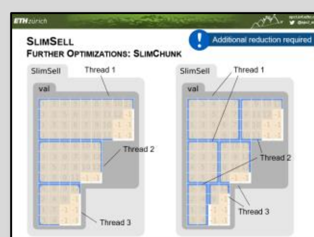






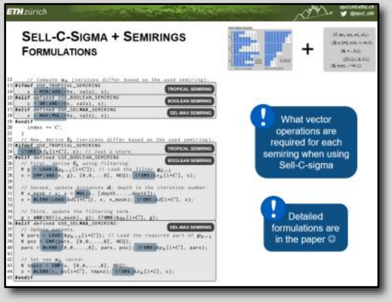
SlimSell: vectorizable representation

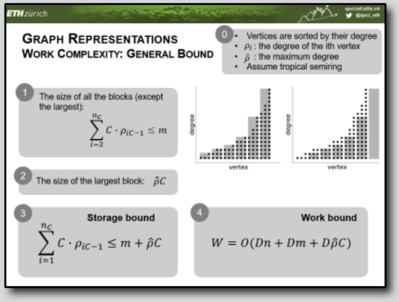


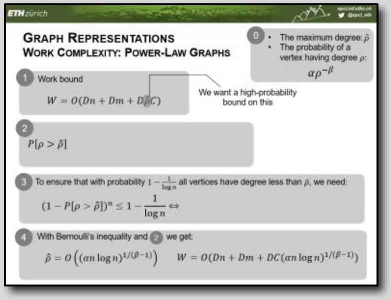


Conclusions

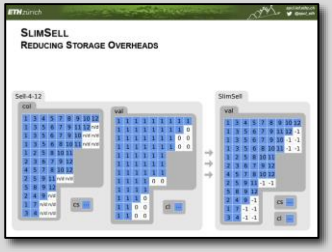
Sell-C-sigma for graphs

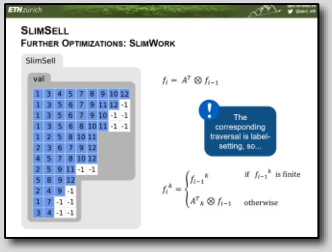




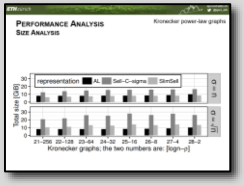


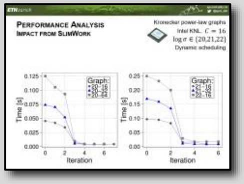
SlimSell: vectorizable representation





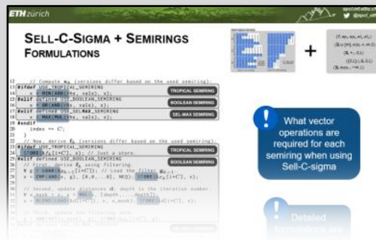
Performance & space analysis

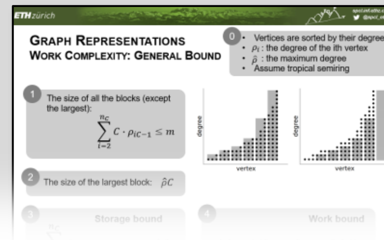


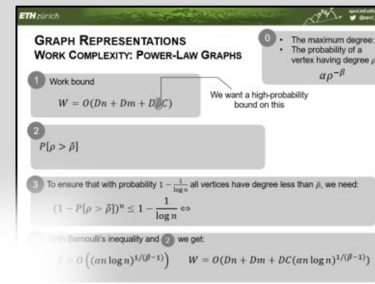


Conclusions

Sell-C-sigma for graphs

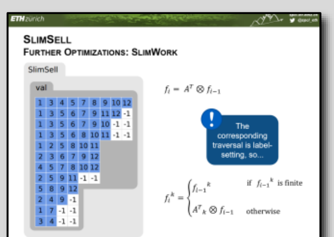


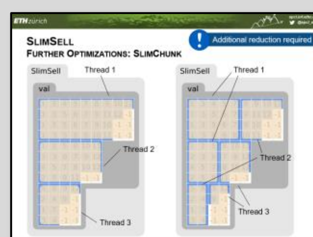




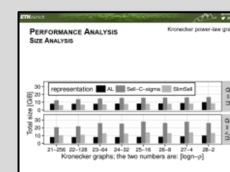
Thank you
for your attention

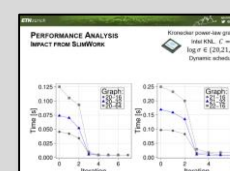
SlimSell: vectorizable representation





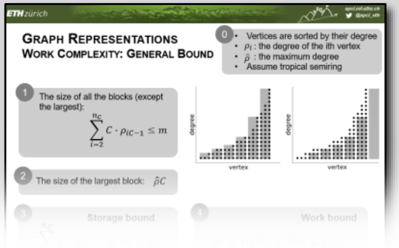
Performance & space analysis

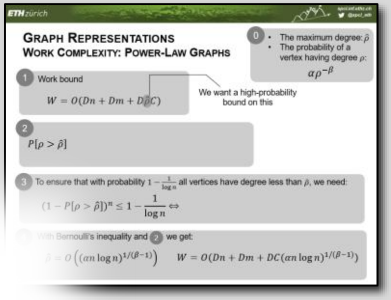




Conclusions

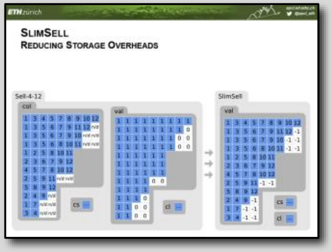
Sell-C-sigma for graphs

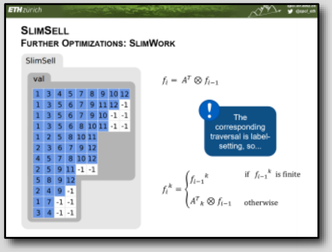




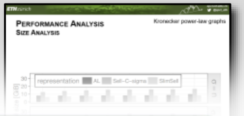
Thank you for your attention

SlimSell: vectorizable representation





Performance & space analysis



spcl.inf.ethz.ch/jobs

...is hiring ☺

SEMIRINGS FOR BFS

SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Semirings for BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

$$(\mathbb{R}, +, \cdot, 0, 1)$$

SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$f_k = A^T \otimes_R f_{k-1}$$

Semirings for BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$f_k = \left(A^T \otimes_R f_{k-1}\right) \odot_R \left(\overline{\sum_{l=0}^{k-1} f_l}\right)$$

Hadamard product

Semirings for BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$f_k = \left(A^T \otimes_R f_{k-1}\right) \odot_R \left(\overline{\sum_{l=0}^{k-1} f_l}\right)$$

Hadamard product

$distances = O(D)$

$parents \in O(m)$

After iterations

Semirings for BFS

Hadamard product

Real semiring

$(\mathbb{R}, +, \cdot, 0, 1)$

$f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\overline{\sum_{l=0}^{k-1} f_l}\right)$

$distances = O(D)$

$parents \in O(m)$

After iterations

Tropical semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$f_k = A'^T \otimes_T f_{k-1}$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Boolean semiring

$(\{0,1\}, |, \&, 0, 1)$

$f_k =$ [similar to Real]

$distances \in O(D)$

$parents \in O(m)$

After iterations

Semirings for BFS

Hadamard product

Real semiring

$(\mathbb{R}, +, \cdot, 0, 1)$

$f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\overline{\sum_{l=0}^{k-1} f_l}\right)$

$distances = O(D)$

$parents \in O(m)$

After iterations

Tropical semiring

$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$

$f_k = A'^T \otimes_T f_{k-1}$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Boolean semiring

$(\{0,1\}, |, \&, 0, 1)$

f_k = [similar to Real]

$distances \in O(D)$

$parents \in O(m)$

After iterations

Sel-max "semiring"

$(\mathbb{R}, max, \cdot, -\infty, 1)$

$f_k = \left(\overline{\overline{A^T \otimes_R f_{k-1}}}\right) - \left(\sum_{l=0}^{k-1} f_l\right)$

$distances \in O(D)$

$parents \in O(1)$

After iterations

SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$$

$$f_k = A'^T \otimes_T f_{k-1}$$

$distances \in O(1)$

$parents \in O(m)$

After iterations

Real semiring

Hadamard product

$$(\mathbb{R}, +, \cdot, 0, 1)$$

$$f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\overline{\sum_{l=0}^{k-1} f_l}\right)$$

$distances = O(D)$

$parents \in O(m)$

After iterations

Boolean semiring

$$(\{0,1\}, |, \&, 0, 1)$$

$f_k =$ [similar to Real]

$distances \in O(D)$

$parents \in O(m)$

After iterations

Sel-max "semiring"

$$(\mathbb{R}, max, \cdot, -\infty, 1)$$

$$f_k = \left(\overline{\overline{A^T \otimes_R f_{k-1}}}\right) - \left(\sum_{l=0}^{k-1} f_l\right)$$

$distances \in O(D)$

$parents \in O(1)$

After iterations

Graph Representations

Work Complexity: Power-Law Graphs

GRAPH REPRESENTATIONS
WORK COMPLEXITY: POWER-LAW GRAPHS

0

- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

GRAPH REPRESENTATIONS
WORK COMPLEXITY: POWER-LAW GRAPHS

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1
Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha\rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}]$$

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta}$$

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}$$

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}$$

3

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}$$

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need:

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}$$

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need:

$$(1 - P[\rho > \hat{\rho}])^n \leq 1 - \frac{1}{\log n} \Leftrightarrow$$

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}$$

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need:

$$(1 - P[\rho > \hat{\rho}])^n \leq 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \geq 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$$

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}$$

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need:

$$(1 - P[\rho > \hat{\rho}])^n \leq 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \geq 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$$

4

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha\rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}$$

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need:

$$(1 - P[\rho > \hat{\rho}])^n \leq 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \geq 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$$

4 With Bernoulli's inequality and 2 we get:

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}$$

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need:

$$(1 - P[\rho > \hat{\rho}])^n \leq 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \geq 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$$

4 With Bernoulli's inequality and 2 we get:

$$\hat{\rho} = O\left((\alpha n \log n)^{1/(\beta - 1)}\right)$$

Graph Representations
Work Complexity: Power-Law Graphs

0
- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree ρ:

$$\alpha \rho^{-\beta}$$

1 Work bound

$$W = O(Dn + Dm + D\hat{\rho}C)$$

We want a high-probability bound on this

2

$$P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}$$

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need:

$$(1 - P[\rho > \hat{\rho}])^n \leq 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \geq 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$$

4 With Bernoulli's inequality and 2 we get:

$$\hat{\rho} = O\left((\alpha n \log n)^{1/(\beta-1)}\right) \qquad W = O(Dn + Dm + DC(\alpha n \log n)^{1/(\beta-1)})$$