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]

LARGE-SCALE IRREGULAR GRAPH PROCESSING

- Becoming more important [1]
  - Machine learning

LARGE-SCALE IRREGULAR GRAPH PROCESSING

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

LARGE-SCALE IRREGULAR GRAPH PROCESSING

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

LARGE-SCALE IRREGULAR GRAPH PROCESSING

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

VECTORIZATION
VECTORIZATION

- Deployed in various hardware
VECTORIZATION

- Deployed in various hardware
- Becoming more popular
VECTORIZATION

- Deployed in various hardware
- Becoming more popular

\[ C = 8 \text{ (SIMD width)} \]
VECTORIZATION

- Deployed in various hardware
- Becoming more popular

\[ C = 16 \text{ (SIMD width)} \]

\[ C = 8 \text{ (SIMD width)} \]
**VECTORIZATION**

- Deployed in various hardware
- Becoming more popular

\[ C = 16 \text{ (SIMD width)} \]

\[ C = 32 \text{ (warp size)} \]

\[ C = 8 \text{ (SIMD width)} \]

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

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

\[ C = 16 \] (SIMD width)

AVX

\[ C = 32 \]

(warp size)

\[ C = 8 \] (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

$C = 16$ (SIMD width)

$C = 32$ (warp size)

$C = 8$ (SIMD width)

$C : \text{"Chunk" size: SIMD width (CPUs, KNLs), warp size (GPUs)}$
VECTORIZATION

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

\[ C = 16 \text{ (SIMD width)} \]

\[ C = 32 \text{ (warp size)} \]

\[ C = 8 \text{ (SIMD width)} \]

\[ C : \text{"Chunk" size: SIMD width (CPUs, KNLs), warp size (GPUs)} \]
VECTORIZATION

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

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

$C = 16$ (SIMD width)

$C = 32$ (warp size)

$C = 8$ (SIMD width)

Regular

AVX

warps
**VECTORIZATION**

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

**LARGE-SCALE IRREGULAR GRAPH PROCESSING**
- Becoming more important [1]
  - Machine learning
  - Computational science
  - Social networks

---

**Vectorization**

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

\[ C = \begin{cases} 8 & \text{(SIMD width)} \\ 16 & \text{(SIMD width)} \\ 32 & \text{(warp size)} \end{cases} \]

\[ C \text{ : } \text{“Chunk” size: SIMD width (CPUs, KNLs), warp size (GPUs)} \]

---

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) $F = \emptyset$
2) $F = \{2\}$
3) $F = \{0,3\}$
4) $F = \{1,4\}$
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) \( F = \{\} \)
2) \( F = \{2\} \)
3) \( F = \{0,3\} \)
4) \( F = \{1,4\} \)

Frontier \( F \)

Distances from the root
**Breadth-First Search**

**Traditional Formulation**

- BFS is based on primitives such as queues

![Diagram](image)

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
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:**

\((\mathbb{R}, \text{op}_1, \text{op}_2, \text{el}_1, \text{el}_2)\)

\[
\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}
\]
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:
\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

Semiring:
\((\mathbb{R}, \text{op}_1, \text{op}_2, \text{el}_1, \text{el}_2)\)
\((\mathbb{R}, +, \cdot, 0, 1)\)

\[
\begin{bmatrix}
1 & 2 \\
3 & 1 \\
\end{bmatrix}
\begin{bmatrix}
0 \\
2 \\
\end{bmatrix}
= \begin{bmatrix}
4 \\
2 \\
\end{bmatrix}
\]
Breadth-First Search
Algebraic Formulation

Tropical Semiring
$\left( \mathbb{R} \cup \{\infty\}, \min, +, \infty, 0 \right)$
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

\[
A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

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

Tropical Semiring

\((\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)\)
**BREADTH-FIRST SEARCH**

**ALGEBRAIC FORMULATION**

Tropical Semiring

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

```
A =
\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

A' =
\[
\begin{bmatrix}
0 & 1 & 1 & \infty & \infty \\
1 & 0 & \infty & 1 & \infty \\
1 & \infty & 0 & 1 & \infty \\
\infty & 1 & 1 & 0 & 1 \\
\infty & \infty & \infty & 1 & 0
\end{bmatrix}
\]
```

Usually stored using a sparse format.
BREADTH-FIRST SEARCH
ALGEBRAIC FORMULATION

\[ A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix} \]

Usually stored using a sparse format

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

\[ f_0 = \begin{pmatrix}
\infty \\
\infty \\
0 \\
\infty
\end{pmatrix} \]

Tropical Semiring
\((\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)\)
BREADTH-FIRST SEARCH
ALGEBRAIC FORMULATION

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

\[ A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix} \]

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

\[ f_0 = \begin{pmatrix}
\infty \\
\infty \\
0 \\
\infty
\end{pmatrix} \]
BREADTH-FIRST SEARCH
ALGEBRAIC FORMULATION

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

Usually stored using a sparse format

\[ A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \]

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

\[ f_0 = \begin{pmatrix} \infty \\ \infty \\ 0 \\ \infty \end{pmatrix} \]

Stored with a dense or a sparse format
Breadth-First Search
Algebraic Formulation

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

Usually stored using a sparse format

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

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

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

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

Usually stored using a sparse format

$\otimes_T$ represents tropical multiplication.
**Breadth-First Search**

**Algebraic Formulation**

\[
A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

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

\[
f_0 = \begin{bmatrix}
\infty \\
\infty \\
0 \\
\infty \\
\infty
\end{bmatrix}
\]

\[
f_1 = A'^T \otimes_T f_0 = \begin{bmatrix}
\end{bmatrix}
\]

**Tropical Semiring**

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

Usually stored using a sparse format

Stored with a dense or a sparse format
**Breadth-First Search**

**Algebraic Formulation**

Tropical Semiring

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

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

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

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

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

Usually stored using a sparse format

Stored with a dense or a sparse format
**BREADTH-FIRST SEARCH**

**ALGEBRAIC FORMULATION**

\[
A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

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

\[
f_0 = \begin{pmatrix}
\infty \\
\infty \\
0 \\
\infty \\
\infty \\
\end{pmatrix}
\]

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

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

Usually stored using a sparse format

Stored with a dense or a sparse format

TYPICAL STORAGE OPTIONS:

- **Sparse format**
- **Dense format**

**READTH-FIRST SEARCH**

**SEARCH**

**ALGEBRAIC FORMULATION**

\[
A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

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

\[
f_0 = \begin{pmatrix}
\infty \\
\infty \\
0 \\
\infty \\
\infty \\
\end{pmatrix}
\]

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

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

Usually stored using a sparse format

Stored with a dense or a sparse format

**SEARCH**

**ALGEBRAIC FORMULATION**

\[
A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

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

\[
f_0 = \begin{pmatrix}
\infty \\
\infty \\
0 \\
\infty \\
\infty \\
\end{pmatrix}
\]

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

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

Usually stored using a sparse format

Stored with a dense or a sparse format

**SEARCH**

**ALGEBRAIC FORMULATION**

\[
A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

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

\[
f_0 = \begin{pmatrix}
\infty \\
\infty \\
0 \\
\infty \\
\infty \\
\end{pmatrix}
\]

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

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

Usually stored using a sparse format

Stored with a dense or a sparse format
**Breadth-First Search**

**Algebraic Formulation**

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

Usually stored using a sparse format

```
A = [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]
```

```
A' = [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]
```

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

\[ A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix} \]

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

\[ f_0 = \begin{pmatrix}
\infty \\
\infty \\
0 \\
\infty \\
\infty
\end{pmatrix} \]

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

\[ f_2 = \begin{pmatrix}
1 \\
2 \\
0 \\
1 \\
2
\end{pmatrix} \]
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

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

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

\[
f_0 = \begin{pmatrix}
\infty \\
\infty \\
0 \\
\infty \\
\infty \\
\end{pmatrix}
\]

Stored with a dense or a sparse format

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

\[
f_2 = \begin{pmatrix}
1 \\
2 \\
0 \\
1 \\
2 \\
\end{pmatrix}
\]

How to do this in practice?
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

Adjacency matrix

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graph representation: [Graph Image]
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

Adjacency matrix

Non-zeros
**GRAPH REPRESENTATIONS**
**COMPRESSED SPARSE ROW (CSR)**

Non-zeros are stored in the `val` array.

- **Size**: `2m` cells
- **Non-zeros**: `n` number of vertices
- **Edges**: `m` number of edges
**Graph Representations**

**Compressed Sparse Row (CSR)**

Adjacency matrix

- Non-zeros are stored in the $val$ array: size: $2m$ cells
- Column indices stored in the $col$ array: size: $2m$ cells

$n$: number of vertices
$m$: number of edges
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

Non-zeros are stored in the \textit{val} array
\begin{itemize}
  \item size: 2m cells
\end{itemize}

Column indices stored in the \textit{col} array
\begin{itemize}
  \item size: 2m cells
\end{itemize}

Row indices are stored in the \textit{row} array
\begin{itemize}
  \item size: n cells
\end{itemize}

\begin{itemize}
  \item \( n \): number of vertices
  \item \( m \): number of edges
\end{itemize}
**Graph Representations**

**Compressed Sparse Row (CSR)**

- **Adjacency matrix**

  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)

Adjacency matrix
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

Adjacency matrix

CSR: val array
Solid arrows: contiguous cells, can be processed in parallel
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

Adjacency matrix

CSR: val array

- Solid arrows: contiguous cells, can be processed in parallel
- Dashed arrows: direction of memory accesses
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

Adjacency matrix

CSR: val array
Solid arrows: contiguous cells, can be processed in parallel
Dashed arrows: direction of memory accesses

Row sizes incompatible with C
GRAPH REPRESENTATIONS

COMPRESSED SPARSE ROW (CSR)

Adjacency matrix

CSR: val array

Solid arrows: contiguous cells, can be processed in parallel

Dashed arrows: direction of memory accesses

Row sizes incompatible with C
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

Adjacency matrix

Costly reductions within rows

Row sizes incompatible with C

CSR: val array
Solid arrows: contiguous cells, can be processed in parallel
Dashed arrows: direction of memory accesses

CSR

C

C
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

Costly reductions within rows

Row sizes incompatible with C

CSR: val array

Solid arrows: contiguous cells, can be processed in parallel

Dashed arrows: direction of memory accesses
GRAPH REPRESENTATIONS
COMPRESSED SPARSE ROW (CSR)

Costly reductions within rows

Solid arrows: contiguous cells, can be processed in parallel
Dashed arrows: direction of memory accesses

Adjacency matrix

Row sizes incompatible with C

CSR: val array
reduce

C
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]

ELLPACK/ELL

ESB [3]

SELL-P [4]

Sliced ELLPACK [2]

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

ACSR [1]
ELLPACK/ELL

ESB [3]
SELL-P [4]

Sliced ELLPACK [2]

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

ACSR [1]
ELLPACK/ELL
ESB [3]
SELL-P [4]
Sliced ELLPACK [2]

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

ACSR [1]

ELLPACK/ELL

ESB [3]

SELL-P [4]

Sliced ELLPACK [2]

SELL-C-sigma [5]

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

- ACSR [1]
- ELLPACK/ELL
- ESB [3]
- Sliced ELLPACK [2]
- SELL-P [4]
- SELL-C-sigma [5]

GRAPH REPRESENTATIONS
SELL-C-SIGMA
**GRAPH REPRESENTATIONS**

**SELL-C-SIGMA**

chunk size

Sell-C-sigma: val array

Sell-4-1 (no sorting)

32 cells used for padding
GRAPH REPRESENTATIONS
SELL-C-SIGMA

chunk size

Sell-C-sigma: val array

C

C

C

C

32 cells used for padding

Sell-4-1 (no sorting)
GRAPH REPRESENTATIONS
SELL-C-SIGMA

chunk size

Sell-C-sigma: val array

padding

32 cells used for padding

Sell-4-1 (no sorting)
GRAPH REPRESENTATIONS
SELL-C-SIGMA

chunk size

padding

sorting scope
\(\sigma \in [1..n]\)

Sell-C-sigma: val array

32 cells used for padding

Sell-4-1 (no sorting)
GRAPH REPRESENTATIONS
SELL-C-SIGMA

chunk size

Sell-C-sigma: val array

padding

sorting scope
\( \sigma \in [1..n] \)

CSR: val array

C

32 cells used for padding

12 cells used for padding

Sell-4-1 (no sorting)

Sell-4-12 (full sorting)
GRAPH REPRESENTATIONS
SELL-C-SIGMA

chunk size

Sell-C-sigma: val array

sorting scope
$$\sigma \in [1..n]$$

padding

32 cells used for padding

12 cells used for padding

Sell-4-1 (no sorting)

Sell-4-12 (full sorting)

Reductions fast with SIMD operations
GRAPH REPRESENTATIONS
SELL-C-SIGMA

chunk size

padding

sorting scope \( \sigma \in [1..n] \)

Sell-C-sigma: val array

32 cells used for padding

12 cells used for padding

Sell-4-1 (no sorting)

Sell-4-12 (full sorting)

Portable

Reductions fast with SIMD operations
SELL-C-SIGMA + SEMIRINGS
SYSTEMATIC ANALYSIS
SELL-C-SIGMA + SEMIRINGS
SYSTEMATIC ANALYSIS

\[(X, op_1, op_2, el_1, el_2) \]
\[(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)\]
\[(\mathbb{R}, max, -, \infty, 1)\]
\[(\{0,1\}, |, \& , 0,1)\]
What are the actual semirings and their formulations?
SELL-C-SIGMA + SEMIRINGS
SYSTEMATIC ANALYSIS

What are the actual semirings and their formulations?

How to derive both distances and parents?

\[(X, op_1, op_2, el_1, el_2)\]
\[(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)\]
\[(\mathbb{R}, \max, -, -\infty, 1)\]
\[(\mathbb{R}, +, 0, 1)\]
\[(\{0,1\}, |, \&, 0, 1)\]
SELL-C-SIGMA + SEMIRINGS
SYSTEMATIC ANALYSIS

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?

\[(X, \text{op}_1, \text{op}_2, \text{el}_1, \text{el}_2)
(\mathbb{R} \cup \{\infty\}, \text{min}, +, \infty, 0)
(\mathbb{R}, \text{max}, ;, -\infty, 1)
(\{0,1\}, |, \&
0,1)\]
SEMIRINGS FOR BFS
SEMIRINGS FOR BFS

Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$$
**SEMIRINGS FOR BFS**

Tropical semiring

\[(\mathbb{R} \cup \{\infty\}, \text{min}, +, \infty, 0)\]

\[f_k = A'^T \otimes_T f_{k-1}\]
SEMRINGS FOR BFS

Tropical semiring

\[(\mathbb{R} \cup \{\infty\}, \text{min}, +, \infty, 0)\]

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

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 $\in O(m)$

**Real semiring**

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

After iterations
**Semirings for BFS**

**Tropical semiring**

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

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

*distances* ∈ \(O(1)\)

*parents* ∈ \(O(m)\)

**Real semiring**

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

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

After iterations
**SEMIRINGS FOR BFS**

**Tropical semiring**

\[(\mathbb{R} \cup \{\infty\}, \text{min}, +, \infty, 0)\]

\[f_k = A'T \otimes_T f_{k-1}\]

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

**Real semiring**

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

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

Hadamard product

After iterations

Distances are $O(1)$

Parents are $O(m)$
**Semirings for BFS**

**Tropical semiring**

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

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

- **distances** ∈ $O(1)$
- **parents** ∈ $O(m)$

**Real semiring**

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

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

- **distances** = $O(D)$
- **parents** ∈ $O(m)$

Hadamard product

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 \( \in O(m) \)

**Real semiring**

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

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

- Distances \( = O(D) \)
- Parents \( \in O(m) \)

**Boolean semiring**

\[ (\{0,1\}, |, \& , 0,1) \]

\[ f_k = [\text{similar to Real}] \]

- Distances \( \in O(D) \)
- Parents \( \in O(m) \)
**SEMIRINGS FOR BFS**

**Tropical semiring**

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

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

- distances ∈ $O(1)$
- parents ∈ $O(m)$

**Real semiring**

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

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

- distances = $O(D)$
- parents ∈ $O(m)$

**Boolean semiring**

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

$f_k = [\text{similar to Real}]$

- distances ∈ $O(D)$
- parents ∈ $O(m)$

**Sel-max “semiring”**

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

$f_k = [\text{more equations } \smile ]$

- distances ∈ $O(D)$
- parents ∈ $O(1)$

- Hadamard product

- 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** $\in O(m)$

**Real semiring**

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

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

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

**Boolean semiring**

$$([0,1], |, &, 0,1)$$

$$f_k = \text{[similar to Real]}$$

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

**Sel-max “semiring”**

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

$$f_k = \text{[more equations 😊 ]}$$

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

**Hadamard product**

After iterations
SELL-C-SIGMA + SEMIRINGS
FORMULATIONS
SELL-C-SIGMA + SEMIRINGS FORMULATIONS

\[(X, \text{op}_1, \text{op}_2, e_1, e_2)\]  
\[\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0\]  
\[\mathbb{R}, +, \cdot, 0, 1\]  
\[\{0, 1\}, \& , 0, 1\]  
\[\mathbb{R}, \max, +, -\infty, 1\]
SELL-C-SIGMA + SEMIRINGS FORMULATIONS

```c
// Compute \( x_k \) (versions differ based on the used semiring):
#ifdef USE_TROPICAL_SEMIRING
  x = MIN(ADD(rhs, vals), x);
#else defined USE_BOOLEAN_SEMIRING
  x = OR(AND(rhs, vals), x);
#else 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.
#else 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);
#else 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
```
SELL-C-SIGMA + SEMIRINGS FORMULATIONS

```c
// Compute x_k (versions differ based on the used semiring):
#endif 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):
#endif 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(0,0,...,0), NEQ); x = BLEND(x, &v[i*C], tmpnz); STORE(&x_k[i*C], x);
#endif
```

Details formulations are in the paper 😊

$$(X, \text{op}_1, \text{op}_2, \text{el}_1, \text{el}_2) \quad (\mathbb{R} \cup \{\infty\}, \text{min}, +, \infty, 0)$$

$$((0,1), \& , 0, 1) \quad (\mathbb{R}, \max, -, \infty, 1)$$
**SELL-C-SIGMA + SEMIRINGS FORMULATIONS**

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

- **Tropical Semiring**
  - Operation: \( \min(\text{ADD}(\text{rhs}, \text{vals}), x) \)
  - Structure: \( (\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0) \)
  - Example: \((0,1], \& , 0,1\) (\(\mathbb{R}, \max, \cdot, -\infty, 1\))

- **Boolean Semiring**
  - Operation: \( \text{OR}(\text{AND}(\text{rhs}, \text{vals}), x) \)
  - Structure: \( (\{0,1\}, \| , 0,1\) \)

- **Sel-Max Semiring**
  - Operation: \( \max(\text{MUL}(\text{rhs}, \text{vals}), x) \)

Detailed formulations are in the paper 😊
SELL-C-SIGMA + SEMIRINGS FORMULATIONS

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

- Vertices are sorted by their degree
- \( \rho_i \) : the degree of the \( i \)th vertex
- \( \hat{\rho} \) : the maximum degree
- Assume tropical semiring
Graph Representations
Computational Complexity

0

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

![Graph degree distribution](image)

- X-axis: vertex
- Y-axis: degree

Legend:
- Grey dots: degree distribution
- Black dots: sorted vertices by degree
GRAPH REPRESENTATIONS
COMPUTATIONAL COMPLEXITY

0. Vertices are sorted by their degree
   - $\rho_i$ : the degree of the $i$th vertex
   - $\hat{\rho}$ : the maximum degree
   - Assume tropical semiring

1. The size of all the blocks (except the largest):
**Graph Representations**

**Computational Complexity**

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

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

0. Vertices are sorted by their degree
- \(\rho_i\): the degree of the \(i\)th vertex
- \(\hat{\rho}\): the maximum degree
- Assume tropical semiring
The size of all the blocks (except the largest):

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

The size of the largest block:

- Vertices are sorted by their degree
- \( \rho_i \): the degree of the \( i \)th vertex
- \( \hat{\rho} \): the maximum degree
- Assume tropical semiring
**Graph Representations**

**Computational Complexity**

1. The size of all the blocks (except the largest):
   
   \[ \# \text{chunks} \sum_{i=2}^{C \cdot \rho_i C-1} \leq 2m \]

2. The size of the largest block: \( \hat{\rho} C \)

- Vertices are sorted by their degree
- \( \rho_i \): the degree of the \( i \)th vertex
- \( \hat{\rho} \): the maximum degree
- Assume tropical semiring
**Graph Representations**

**Computational Complexity**

1. The size of all the blocks (except the largest):
   \[
   \sum_{i=2}^{\text{#chunks}} C \cdot \rho_{iC-1} \leq 2m
   \]

2. The size of the largest block: \( \hat{\rho} C \)

3. Storage bound

0. Vertices are sorted by their degree
   - \( \rho_i \): the degree of the ith vertex
   - \( \hat{\rho} \): the maximum degree
   - Assume tropical semiring
**Graph Representations**

**Computational Complexity**

1. The size of all the blocks (except the largest):
   \[
   \sum_{i=2}^{\text{\#chunks}} C \cdot \rho_{iC-1} \leq 2m
   \]

2. The size of the largest block:
   \[\hat{\rho}C\]

3. Storage bound
   \[
   \sum_{i=1}^{\text{\#chunks}} C \cdot \rho_{iC-1} \leq 2m + \hat{\rho}C
   \]

- Vertices are sorted by their degree
- \(\rho_i\): the degree of the \(i\)th vertex
- \(\hat{\rho}\): the maximum degree
- Assume tropical semiring
Graph Representations
Computational Complexity

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

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

2. The size of the largest block: \( \hat{\rho} C \)

3. Storage bound

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

4. Computational complexity bound

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

- Vertices are sorted by their degree
- \( \rho_i \) : the degree of the ith vertex
- \( \hat{\rho} \) : the maximum degree
- Assume tropical semiring
GRAPH REPRESENTATIONS
COMPUTATIONAL COMPLEXITY

1. The size of all the blocks (except the largest):
   \[\sum_{i=2}^{\text{#chunks}} C \cdot \rho_{iC-1} \leq 2m + \hat{\rho}C\]

2. The size of the largest block:
   \[\sum_{i=1}^{\text{#chunks}} C \cdot \rho_{iC-1} \leq 2m + \hat{\rho}C\]

3. Storage bound
   \[W = O(D(n + m + \hat{\rho}C)) = O(Dn + Dm + D\hat{\rho}C)\]

4. Computational complexity bound

- Vertices are sorted by their degree
- \(\rho_i\): the degree of the ith vertex
- \(\hat{\rho}\): the maximum degree
- Assume tropical semiring
**Graph Representations**

**Computational Complexity**

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

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

2. The size of the largest block:

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

3. Storage bound

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

4. Computational complexity bound

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

Is that all?

Not really...
SLIMSELL
REDUCING STORAGE OVERHEADS
**SLIMSSELL**

**REDUCING STORAGE OVERHEADS**

<table>
<thead>
<tr>
<th>col</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>n/d</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>n/d</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>n/d</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>n/d</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>n/d</td>
<td>n/d</td>
</tr>
</tbody>
</table>

---

This table represents a matrix used in SLIMSSELL to illustrate the reduction of storage overheads. The matrix shows a comparison between the original data (col) and the compressed data (val), where each cell indicates whether a value is preserved or deleted. The 'n/d' indicates a value that is not determined or not applicable.
SLIMSELL
REDUCING STORAGE OVERHEADS
# SLIMSELL

## REDUCING STORAGE OVERHEADS

<table>
<thead>
<tr>
<th>Representation</th>
<th>Sell-(C-\sigma)</th>
<th>CSR</th>
<th>AL</th>
<th>SlimSell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size [cells]</td>
<td>(4m + \frac{2n}{C} + P)</td>
<td>(4m + n)</td>
<td>(2m + n)</td>
<td>(2m + \frac{2n}{C} + P)</td>
</tr>
</tbody>
</table>

**Sell-4-12**

<table>
<thead>
<tr>
<th>col</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
</tbody>
</table>

**SlimSell**

<table>
<thead>
<tr>
<th>val</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
</tbody>
</table>

**SlimSell**

<table>
<thead>
<tr>
<th>val</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>n/d</td>
<td>n/d</td>
</tr>
</tbody>
</table>
SLIMSELL
REDUCING STORAGE OVERHEADS
# SlimSell

## Reducing Storage Overheads

<table>
<thead>
<tr>
<th>Representation</th>
<th>Sell-$C-\sigma$</th>
<th>CSR</th>
<th>AL</th>
<th>SlimSell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size [cells]</td>
<td>$4m + \frac{2n}{C} + P$</td>
<td>$4m + n$</td>
<td>$2m + n$</td>
<td>$2m + \frac{2n}{C} + P$</td>
</tr>
</tbody>
</table>
SLIMSELL
REDUCING STORAGE OVERHEADS

<table>
<thead>
<tr>
<th>Representation</th>
<th>Sell-C-σ</th>
<th>CSR</th>
<th>AL</th>
<th>SlimSell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size [cells]</td>
<td>$4m + \frac{2n}{C} + P$</td>
<td>$4m + n$</td>
<td>$2m + n$</td>
<td>$2m + \frac{2n}{C} + P$</td>
</tr>
</tbody>
</table>

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

**REDUCING STORAGE OVERHEADS**

<table>
<thead>
<tr>
<th>Representation</th>
<th>Sell-$C$-$\sigma$</th>
<th>CSR</th>
<th>AL</th>
<th>SlimSell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size [cells]</td>
<td>$4m + \frac{2n}{C} + P$</td>
<td>$4m + n$</td>
<td>$2m + n$</td>
<td>$2m + \frac{2n}{C} + P$</td>
</tr>
</tbody>
</table>

\[
2m + \frac{2n}{C} + P < n + 2m \iff P < n \left(1 - \frac{2}{C}\right)
\]

- $C = 8$
  - $P < \frac{3n}{4}$
- $C = 16$
  - $P < \frac{7n}{8}$
- $C = 32$
  - $P < \frac{15n}{16}$
SLIMSELL

FURTHER OPTIMIZATIONS: SLIMWORK
SLIMSELL

FURTHER OPTIMIZATIONS: SLIMWORK
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: SLIMWORK

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

**FURTHER OPTIMIZATIONS: SLIMCHUNK**

<table>
<thead>
<tr>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
</tbody>
</table>

```
1 3 5 6 7 9 10 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**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>-1</td>
</tr>
<tr>
<td>Thread 2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Thread 3</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
**SlimSell**

**Further Optimizations: SlimChunk**

Thread 1

<table>
<thead>
<tr>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3 4 5 7 8 9 10 12</td>
</tr>
<tr>
<td>1 3 5 6 7 9 11 12 -1</td>
</tr>
<tr>
<td>1 3 5 6 7 9 10 -1 -1</td>
</tr>
<tr>
<td>1 3 5 6 8 10 11 -1 -1</td>
</tr>
<tr>
<td>1 2 5 8 10 11</td>
</tr>
<tr>
<td>2 3 6 7 9 12</td>
</tr>
<tr>
<td>4 5 7 8 10 12</td>
</tr>
<tr>
<td>2 5 9 11 -1 -1</td>
</tr>
<tr>
<td>5 8 9 12</td>
</tr>
<tr>
<td>2 4 9 -1</td>
</tr>
<tr>
<td>1 7 -1 -1</td>
</tr>
<tr>
<td>3 4 -1 -1</td>
</tr>
</tbody>
</table>

Thread 2

<table>
<thead>
<tr>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3 4 5 7 8 9 10 12</td>
</tr>
<tr>
<td>1 3 5 6 7 9 11 12 -1</td>
</tr>
<tr>
<td>1 3 5 6 7 9 10 -1 -1</td>
</tr>
<tr>
<td>1 3 5 6 8 10 11 -1 -1</td>
</tr>
<tr>
<td>1 2 5 8 10 11</td>
</tr>
<tr>
<td>2 3 6 7 9 12</td>
</tr>
<tr>
<td>4 5 7 8 10 12</td>
</tr>
<tr>
<td>2 5 9 11 -1 -1</td>
</tr>
<tr>
<td>5 8 9 12</td>
</tr>
<tr>
<td>2 4 9 -1</td>
</tr>
<tr>
<td>1 7 -1 -1</td>
</tr>
<tr>
<td>3 4 -1 -1</td>
</tr>
</tbody>
</table>

Thread 3
SLIMSELL

FURTHER OPTIMIZATIONS: SLIMCHUNK

Thread 1

Thread 2

Thread 3

Thread 1

Thread 2

Thread 3
**SLIMSELL**

**FURTHER OPTIMIZATIONS: SLIMCHUNK**

<table>
<thead>
<tr>
<th>SlimSell</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>val</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 3 4 5 7 8 9 10 12</td>
<td>1 3 5 6 7 9 11 12 -1</td>
<td>1 3 5 6 7 9 10 -1 -1</td>
<td>1 3 5 6 8 10 11 -1 -1</td>
</tr>
<tr>
<td>1 3 5 6 7 9 10 -1 -1</td>
<td>1 3 5 6 7 9 11 12</td>
<td>1 3 5 6 7 9 10 -1 -1</td>
<td>1 3 5 6 8 10 11 -1 -1</td>
</tr>
<tr>
<td>1 2 5 8 10 11</td>
<td>2 3 6 7 9 12</td>
<td>2 3 6 7 9 12</td>
<td>2 3 6 7 9 12</td>
</tr>
<tr>
<td>5 8 9 12</td>
<td>4 5 7 8 10 12</td>
<td>4 5 7 8 10 12</td>
<td>4 5 7 8 10 12</td>
</tr>
<tr>
<td>2 5 9 11 -1 -1</td>
<td>2 5 9 11 -1 -1</td>
<td>2 5 9 11 -1 -1</td>
<td>2 5 9 11 -1 -1</td>
</tr>
<tr>
<td>3 4 -1 -1</td>
<td>1 7 -1 -1</td>
<td>1 7 -1 -1</td>
<td>1 7 -1 -1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 are storage and performance improvements from SlimSell?
- What is the impact of various parameters (e.g., thread scheduling)?
PERFORMANCE ANALYSIS

TYPES OF MACHINES
PERFORMANCE ANALYSIS
TYPES OF MACHINES

Trivium Intel Server

CSCS Greina cluster

CSCS Piz Daint & Piz Dora
PERFORMANCE ANALYSIS
TYPES OF MACHINES

Intel Xeon CPU
CSCS Greina cluster

Intel Haswell CPU
Trivium Intel Server

Intel Xeon CPU
CSCS Piz Daint & Piz Dora
PERFORMANCE ANALYSIS
TYPES OF MACHINES

Intel Haswell CPU
NVIDIA GTX 670 GPU
Trivium Intel Server

NVIDIA Tesla K80 GPU
Intel Xeon CPU
CSCS Greina cluster

Intel Xeon CPU
NVIDIA Tesla K20X GPU
CSCS Piz Daint & Piz Dora
**Performance Analysis**

Types of Machines

- Nvidia Tesla K80 GPU
- Intel Xeon Phi KNL
- Intel Xeon CPU

- Intel Haswell CPU
- Nvidia GTX 670 GPU

- Trivium Intel Server

CSCS Greina cluster

Intel Xeon CPU
Nvidia Tesla K20X GPU
CSCS Piz Daint & Piz Dora
PERFORMANCE ANALYSIS
TYPES OF GRAPHS
Performance Analysis
Types of Graphs

Synthetic graphs
PERFORMANCE ANALYSIS
TYPES OF GRAPHS

Synthetic graphs

Kronecker [1]

PERFORMANCE ANALYSIS

TYPES OF GRAPHS

Synthetic graphs

Kronecker [1]

Erdös-Rényi [2]

PERFORMANCE ANALYSIS
TYPES OF GRAPHS

Real-world SNAP graphs [3]

Synthetic graphs

Kronecker [1]

Erdős-Rényi [2]

**Performance Analysis**

**Types of Graphs**

**Synthetic graphs**

1. Kronecker [1]

2. Erdös-Rényi [2]

**Real-world SNAP graphs [3]**

1. Social networks
2. Road networks
3. Communication graphs
4. Citation graphs
5. Web graphs
6. Purchase networks

---

PERFORMANCE ANALYSIS
OTHER PARAMETERS
### PERFORMANCE ANALYSIS

### OTHER PARAMETERS

<table>
<thead>
<tr>
<th>Semirings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tropical</td>
<td>$(\mathbb{R} \cup {\infty}, \text{min}, +, \infty, 0)$</td>
</tr>
<tr>
<td>Real</td>
<td>$(\mathbb{R}, +, \cdot, 0, 1)$</td>
</tr>
<tr>
<td>Boolean</td>
<td>$({0, 1},</td>
</tr>
<tr>
<td>Sel-max</td>
<td>$(\mathbb{R}, \text{max}, \cdot, -\infty, 1)$</td>
</tr>
</tbody>
</table>
**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: \((\mathbb{R}, \max, \cdot, -\infty, 1)\)

**OpenMP scheduling**
- Static
- Dynamic
**Performance Analysis**

**Other Parameters**

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

**OpenMP scheduling**
- Static
- Dynamic

**Scaling**
- Strong
- Weak
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: \((\mathbb{R}, \max, \cdot, -\infty, 1)\)

OpenMP scheduling
Static
Dynamic

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

\[ n = 2^{23}, \quad \bar{\rho} = 16 \]

Xeon CPU, \( C = 8 \)

\[ \log \sigma = 23 \]
**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

---

Graph: 20–16, 20–32, 20–64

Graph: 21–16, 21–32, 22–16

---

Time [s] vs. Iteration
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

Total size [GiB]

representation
AL
Sell-C-sigma
SlimSell


Kronecker graphs; the two numbers are: [logn−ρ]
Performance Analysis

Size Analysis

Kronecker power-law graphs

![Bar chart showing the total size of Kronecker graphs for different ranges of logn−ρ. The bars are color-coded as follows: representation, AL, Sell−C−sigma, and SlimSell.](image_url)
OTHER ANALYSES
OTHER ANALYSES
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

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

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

*distances* $\in O(1)$

*parents* $\in O(m)$

After iterations
**SEMIRINGS FOR BFS**

Tropical semiring

\[(\mathbb{R} \cup \{\infty\}, \text{min}, +, \infty, 0)\]

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

- distances ∈ \(O(1)\)
- parents ∈ \(O(m)\)

Real semiring

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

After iterations

After iterations
SEMI RINGS FOR BFS

Tropical semiring

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

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

\(\text{distances} \in O(1)\)

\(\text{parents} \in O(m)\)

Real semiring

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

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

\text{After iterations}

\(\text{distances} \in O(1)\)

\(\text{parents} \in O(m)\)
**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* \( \in O(m) \)

**Real semiring**

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

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

- *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 $\in O(m)$

**Real semiring**

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

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

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

**Hadamard product**

- After iterations

**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 \( \in O(m) \)

**Real semiring**

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

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

distances \( = O(D) \)

parents \( \in O(m) \)

**Boolean semiring**

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

\( f_k = \text{[similar to Real]} \)

distances \( \in O(D) \)

parents \( \in O(m) \)

Hadamard product
**Semirings for BFS**

**Tropical semiring**

\[
(\mathbb{R} \cup \{\infty\}, \text{min}, +, \infty, 0)
\]

\[
f_k = A^T \otimes_T f_{k-1}
\]

- \text{distances} \in O(1)
- \text{parents} \in O(m)

**Boolean semiring**

\[
(\{0,1\}, |, \&, 0,1)
\]

\[
f_k = [\text{similar to Real}]
\]

- \text{distances} \in O(D)
- \text{parents} \in O(m)

**Real semiring**

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

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

- \text{distances} = O(D)
- \text{parents} \in O(m)

**Sel-max “semiring”**

\[
(\mathbb{R}, \text{max}, \cdot, -\infty, 1)
\]

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

- \text{distances} \in O(D)
- \text{parents} \in O(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* \(\in O(m)\)

**Real semiring**

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

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

*distances* \(\in O(D)\)

*parents* \(\in O(m)\)

**Boolean semiring**

\[ (\{0,1\}, |, \& , 0, 1) \]

\[ f_k = [\text{similar to Real}] \]

*distances* \(\in O(D)\)

*parents* \(\in O(m)\)

**Sel-max “semiring”**

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

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

*distances* \(\in O(D)\)

*parents* \(\in O(1)\)
GRAPH REPRESENTATIONS
WORK COMPLEXITY: POWER-LAW GRAPHS
GRAPH REPRESENTATIONS
WORK COMPLEXITY: POWER-LAW GRAPHS

- The maximum degree: $\hat{\rho}$
- The probability of a vertex having degree $\rho$:
  \[ \alpha \rho^{-\beta} \]
Graph Representations

Work Complexity: Power-Law Graphs

1. Work bound

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

0. The maximum degree: \( \hat{\rho} \)
- The probability of a vertex having degree \( \rho \):

\[ \alpha \rho^{-\beta} \]
Graph Representations
Work Complexity: Power-Law Graphs

1. Work bound

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

0. The maximum degree: \( \hat{\rho} \)
   - The probability of a vertex having degree \( \rho \):
     \[ \alpha \rho^{-\beta} \]

We want a high-probability bound on this
0  • The maximum degree: $\hat{\rho}$
• The probability of a vertex having degree $\rho$: $\alpha \rho^{-\beta}$

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

1. Work bound

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

2. \( P[\rho > \hat{\rho}] \)

0. The maximum degree: \( \hat{\rho} \)

The probability of a vertex having degree \( \rho \):

\[ \alpha \rho^{-\beta} \]

We want a high-probability bound on this.
Graph Representations

Work Complexity: Power-Law Graphs

1. Work bound

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

We want a high-probability bound on this

2. Probability bound

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

We want a high-probability bound on this

0. The maximum degree: \( \hat{\rho} \)

- The probability of a vertex having degree \( \rho \): \( \alpha \rho^{-\beta} \)
GRAPH REPRESENTATIONS
WORK COMPLEXITY: POWER-LAW GRAPHS

1. Work bound

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

We want a high-probability bound on this

2. 

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

0. The maximum degree: \( \hat{\rho} \)

• The probability of a vertex having degree \( \rho \):

\[ \alpha \rho^{-\beta} \]
Graph Representations

Work Complexity: Power-Law Graphs

1. Work bound

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

2. Probability bound

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

3. We want a high-probability bound on this

- The maximum degree: \( \hat{\rho} \)
- The probability of a vertex having degree \( \rho \):

\[ \alpha \rho^{-\beta} \]
The maximum degree: \( \hat{\rho} \)

- The probability of a vertex having degree \( \rho \):
  \[
  P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta} \approx \alpha \int_{\hat{\rho}}^{\infty} x^{-\beta} dx = \alpha \frac{\hat{\rho}^{1-\beta}}{\beta - 1}
  \]

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

1. Work bound

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

2. Probability of a vertex having degree \( \rho \):

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

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

\[ (1 - P[\rho > \hat{\rho}])^n \leq 1 - \frac{1}{\log n} \iff \]
Graph Representations

Work Complexity: Power-Law Graphs

1. Work bound

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

2. Probability bound

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

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

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

- The maximum degree: \( \hat{\rho} \)
- The probability of a vertex having degree \( \rho \):

\[ \alpha \rho^{-\beta} \]
**Graph Representations**

**Work Complexity: Power-Law Graphs**

1. Work bound

\[ W = O(Dn + Dm + D\rho C) \]

2. Probability of a vertex having degree \( \rho \):

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

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

\[ (1 - P[\rho > \hat{\rho}])^n \leq 1 - \frac{1}{\log n} \iff P[\rho > \hat{\rho}] \geq 1 - \left(1 - \frac{1}{\log n}\right)^{1/n} \]
**GRAPH REPRESENTATIONS**

**WORK COMPLEXITY: POWER-LAW GRAPHS**

1. **Work bound**

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

2. **Probability bound**

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

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

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

4. **With Bernoulli’s inequality and 2, we get:**
Graph Representations

Work Complexity: Power-Law Graphs

1. Work bound

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

We want a high-probability bound on this

2. Probability bound

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

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

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

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

1. Work bound

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

We want a high-probability bound on this

2. Probability bound

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

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

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

4. With Bernoulli’s inequality and 2, we get:

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