EHzürich



SlimSell: A Vectorizable Graph Representation for Breadth-First Search

MACIEJ BESTA, FLORIAN MARENDING, EDGAR SOLOMONIK, TORSTEN HOEFLER





Becoming more important [1]





- Becoming more important [1]
 - Machine learning





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



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



- 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.

- 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.



1 1 2 2 2 4 K - ---

VECTORIZATION





Deployed in various hardware





- Deployed in various hardware
- Becoming more popular



- Deployed in various hardware
- Becoming more popular



C = 8 (SIMD width)



spcl.inf.ethz.ch

VECTORIZATION

- Deployed in various hardware
- Becoming more popular





C = 8 (SIMD width)



- Deployed in various hardware
- Becoming more popular





C: ", Chunk" size: SIMD width (CPUs, KNLs), warp size (GPUs) C = 8 (SIMD width)



- 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 = 8 (SIMD width)

ਤ┥ዸ▎▅▏▙

- 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 = 8 (SIMD width)



- 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 = 8 (SIMD width)



AVX

VECTORIZATION

	16-wide Vector SIMD							
	ALU	ALU	ALU	ALU	ALU	ALU	ALU	ALU
	ALU	ALU	ALU	ALU	ALU	ALU	ALU	ALU

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



and the sector

C: "Chunk" size: SIMD width (CPUs, KNLs), warp size (GPUs) C = 8 (SIMD width)

Deployed in various hardware

Becoming more popular

64KB Register File16-wide Vector SIMDALU

= Hzürich ETHzürich AND T Specimic LARGE-SCALE IRREGULAR GRAPH PROCESSING VECTORIZATION 64KB Register File ALU ALU ALU ALU ALU ALU ALU Becoming more important [1] Deployed in various hardware ALU ALU ALU ALU ALU ALU ALU Machine learning Becoming more popular Computational science Offers a lot of ... Social netw Irregular Regular C = 32 $\left| + \frac{1}{\sqrt{2}} \right|$ 닀 C = 16C = 8[1] A. Lumsdaine et al. Challenges in Parallel Graph Processing. Parallel Processing Let. 2003 AVX C = 16 (SIMD width)

C: ", Chunk" size: SIMD width (CPUs, KNLs), warp size (GPUs) C = 8 (SIMD width)







































1)
$$F = \{\}$$

2) $F = \{2\}$
3) $F = \{0,3\}$





1)
$$F = \{\}$$

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





















BFS is based on primitives such as queues



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



Parents (predecessors) in the traversal tree



BFS is based on primitives such as queues





Parents (predecessors) in the traversal tree









BFS is a series of matrix-vector products





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





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




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





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



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

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



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

 $(\mathbb{R}, +, \cdot, 0, 1)$ $\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$









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



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

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























3

































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



Paradon









Adjacency matrix







Adjacency matrix







Adjacency matrix





Non-zeros are stored									
in the val array						size: 2 <i>m</i> cells			



Adjacency matrix







Adjacency matrix





Row indices are stored in the *row* array size: *n* cells ...



Adjacency matrix







State of the second state

GRAPH REPRESENTATIONS COMPRESSED SPARSE ROW (CSR)

Adjacency matrix





Adjacency matrix





The second second



Adjacency matrix







Adjacency matrix







Adjacency matrix



Row sizes incompatible with C





Adjacency matrix



Row sizes incompatible with C





Costly reductions within rows







Costly reductions within rows







Costly reductions within rows



Row sizes incompatible with C




all of the store was



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



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



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



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.





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.





[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.





[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.



A TOPPE

GRAPH REPRESENTATIONS Sell-C-Sigma



chunk size





chunk size







chunk size







chunk size



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













chunk size





Reductions fast with SIMD operations



chunk size





Reductions fast with SIMD operations















 $(X, op_1, op_2, el_1, el_2)$ $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$ $(\mathbb{R}, max, \cdot, -\infty, 1)$ $(\mathbb{R}, +, \cdot, 0, 1)$ $(\{0,1\}, |, \&, 0, 1)$

What are the actual semirings and their formulations?





 $(X, op_1, op_2, el_1, el_2)$ $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$ $(\mathbb{R}, max, \cdot, -\infty, 1)$ $(\mathbb{R}, +, \cdot, 0, 1)$ $(\{0,1\}, |, \&, 0, 1)$

What are the actual semirings and their formulations?

How to derive both distances and parents?





 $(X, op_1, op_2, el_1, el_2)$ $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$ $(\mathbb{R}, max, \cdot, -\infty, 1)$ $(\mathbb{R}, +, \cdot, 0, 1)$ $(\{0,1\}, |, \&, 0, 1)$

What are the actual semirings and their formulations?

What is work complexity of BFS based on Sell-Csigma? How to derive both distances and parents?



COME THE

SEMIRINGS FOR BFS



Tropical semiring

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



Tropical semiring

 $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ $f_k = A'^T \otimes_T f_{k-1}$



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



Tropical semiring

 $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ $f_k = A'^T \otimes_T f_{k-1}$ $\underset{\text{distances} \in O(1)}{\text{distances} \in O(n)}$

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



Tropical semiring

 $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ $f_k = A'^T \otimes_T f_{k-1}$ $\underset{\text{distances} \in O(1)}{\text{distances} \in O(n)}$

Real semiring $(\mathbb{R}, +, \cdot, 0, 1)$ $f_k = A^T \otimes_R f_{k-1}$



Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$$
$$f_k = A'^T \otimes_T f_{k-1}$$
$$After \\ iterations \\ parents \in O(m)$$

Hadamard product Real semiring $(\mathbb{R}, +, \cdot, 0, 1)$ $f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\sum_{l=0}^{k-1} f_l \right)$



Tropical semiring

$$(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$$
$$f_k = A'^T \otimes_T f_{k-1}$$
$$\underset{\text{distances} \in O(1)}{\text{distances} \in O(n)}$$

Hadamard product 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) $parents \in O(m)$ After iterations



Tropical semiring

 $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ $f_k = A'^T \otimes_T f_{k-1}$ $\underset{distances \in O(1)}{distances \in O(n)}$ After iterations parents \in O(m)

Hadamard product
Real semiring

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

 $f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\sum_{l=0}^{k-1} f_l \right)$
 $distances = O(D)$
 $parents \in O(m)$
After
iterations

Boolean semiring $(\{0,1\}, |, \&, 0,1)$ $f_k = [similar to Real]$ $f_{k} = [similar to Real]$ $f_{k} = [similar to Real]$ $f_{k} = [similar to Real]$ $f_{k} = [similar to Real]$



Tropical semiring

 $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ $f_k = A'^T \otimes_T f_{k-1}$ $\underset{\text{distances} \in O(1)}{\text{distances} \in O(n)}$

Hadamard product
Real semiring

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

 $f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\sum_{l=0}^{k-1} f_l \right)$
 $distances = O(D)$
 $parents \in O(m)$
After
iterations

Boolean semiring $(\{0,1\}, |, \&, 0,1)$ $f_k = [similar to Real]$ $distances \in O(D)$ $parents \in O(m)$ 

Tropical semiring

 $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ $f_k = A'^T \otimes_T f_{k-1}$ $\underset{\text{distances} \in O(1)}{\text{distances} \in O(n)}$

Hadamard product
Real semiring

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

 $f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\sum_{l=0}^{k-1} f_l \right)$
 $distances = O(D)$
 $parents \in O(m)$
After
iterations

Boolean semiring $(\{0,1\}, |, \&, 0,1)$ $f_k = [similar to Real]$ $distances \in O(D)$ $parents \in O(m)$ Sel-max "semiring" $(\mathbb{R}, max, \cdot, -\infty, 1)$ $f_k = [\text{more equations } \odot]$ $distances \in O(D) \qquad \text{After} \\ parents \in O(1) \qquad - \text{iterations}$







 $(X, op_1, op_2, el_1, el_2)$ $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$ $(\mathbb{R}, +, \cdot, 0, 1)$ ({0,1}, |, &, 0,1) $(\mathbb{R}, max, \cdot, -\infty, 1)$

MS PCL



 $(X, op_1, op_2, el_1, el_2)$ $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$ $(\mathbb{R}, +, \cdot, 0, 1)$ ({0,1}, |, &, 0,1) $(\mathbb{R}, max, \cdot, -\infty, 1)$

13 #ifdef USE_TROPICAL_SEMIRING 14 $x = MIN(ADD(rhs, vals), x);$ TROPICAL SEMIRING 15 #elif defined USE_BOOLEAN_SEMIRING 16 $x = OR(AND(rhs, vals), x);$ BOOLEAN SEMIRING 17 #elif defined USE_SELMAX_SEMIRING 18 $x = MAX(MUL(rhs, vals), x);$ BEL-MAX SEMIRING 19 #endif 20 index += C; 21 } 22 // Now, derive f_k (versions differ based on the used semiring); 23 #ifdef USE_TROPICAL_SEMIRING 24 STORE(δf_k [i*C], x); // Just a store. 25 #elif defined USE_BOOLEAN_SEMIRING 25 #elif defined USE_BOOLEAN_SEMIRING 26 // First, derive f_k using filtering. 27 V g = LOAD(δg_{k-1} [i*C]); // Load the filter g_{k-1} . 28 x = CMP(AND(x, g), [0,0,0], NEQ); STORE(δx_k [i*C], x); 30 // Second, update distances d; depth is the iteration number. 31 V x_mask = x; x = MUL(x, [depth,, depth]); 32 x = BLEND(LOAD(δd [i*C]), x, x_mask); STORE(δd [i*C], x); 33 // Third, update the filtering term. 34 // Third, update the filtering term. 35 #elif defined USE_SELMAX_SEMIRING: 36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD(δp_{k-1} [i*C]); // Load the required part of \mathbf{p}_{k-1} 39 V pars = DLAD($[0,0,,0]$, NEQ); 30 // Set new x_k vector. 31 V tmpnz = CMP(pars, [0,0,,0], NEQ); 32 #elEND ([0,0,,0], pars, pnz); STORE(δp_k [i*C], pars); 34 // Set new x_k vector. 35 W tmpnz = CMP(x, [0,0,,0], NEQ); 36 #elif defined USE_SELMAX_SEMIRING: 37 // Set new x_k vector. 37 // Set new $x_$	12	// Compute \mathbf{x}_k (versions differ based on the used semiring):
<pre>14 x = MIN(ADD(rhs, vals), x);</pre>	13	#ifdef USE_TROPICAL_SEMIRING
<pre>15 #elif defined USE_BOOLEAN_SEMIRING x = OR(AND(rhs, vals), x); 17 #elif defined USE_SELMAX_SEMIRING x = MAX(MUL(rhs, vals), x); 19 #endif 20 index += C; 21 } 23 // Now, derive f_k (versions differ based on the used semiring); 24 #ifdef USE_TROPICAL_SEMIRING 25 #elif defined USE_BOOLEAN_SEMIRING 26 #elif defined USE_BOOLEAN_SEMIRING 27 V g = LOAD(&g_{k-1}[i*C]); // Just a store. 28 #elif defined USE_BOOLEAN_SEMIRING 29 // First, derive f_k using filtering. 20 W g = LOAD(&g_{k-1}[i*C]); // Load the filter g_{k-1}. 28 x = CMP(AND(x, g), [0,0,0], NEQ); STORE($&x_k[i*C]$, x); 29 // Second, update distances d; depth is the iteration number. 31 V x_mask = x; x = MUL(x, [depth,,depth]); 32 x = BLEND(LOAD(&d[i*C]), x, x_mask); STORE(&d[i*C], x); 33 // Third, update the filtering term. 34 // Third, update the filtering term. 35 g = AND(NOT(x_mask), g); STORE($&g_k[i*C]$, g); 36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD(&p_{k-1}[i*C]); // Load the required part of P_{k-1} 39 V pnz = CMP(pars, [0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE($&p_k[i*C]$, pars); 41 // Set new x_k vector. 42 // Set new x_k vector. 44 x = BLEND(x, &v[i*C], tmpnz); STORE($&x_k[i*C], x)$; 45 #endif</pre>	14	x = MIN(ADD(rhs, vals), x);
16 $x = OR(AND(rhs, vals), x);$ 17 #elif defined USE_SELMAX_SEMIRING 17 #elif defined USE_SELMAX_SEMIRING 18 $x = MAX(MUL(rhs, vals), x);$ 19 #endif 20 index += C; 21 } 22 // Now, derive f_k (versions differ based on the used semiring); 23 #ifdef USE_TROPICAL_SEMIRING 24 STORE(& f_k [i*C], x); // Just a store. 25 #elif defined USE_BOOLEAN_SEMIRING 25 #elif defined USE_BOOLEAN_SEMIRING 26 // First, derive f_k using filtering. 27 V g = LOAD(& g_{k-1} [i*C]); // Load the filter g_{k-1} . 28 x = CMP(AND(x, g), [0,0,0], NEQ); STORE(& x_k [i*C], x); 29 30 // Second, update distances d; depth is the iteration number. 31 V x_mask = x; x = MUL(x, [depth,,depth]); 32 x = BLEND(LOAD(& d[i*C]), x, x_mask); STORE(& d[i*C], x); 34 // Third, update the filtering term. 35 g = AND(NOT(x_mask), g); STORE(& g_k [i*C], g); 36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD($& p_{k-1}$ [i*C]); // Load the required part of P_{k-1} 39 V pnz = CMP(pars, [0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE(& p_k [i*C], pars); 41 42 // Set new x_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, & v [i*C], tmpnz); STORE(& x_k [i*C], x); 45 #endif	15	#elif defined USE_BOOLEAN_SEMIRING
17 #elif defined USE_SELMAX_SEMIRING x = MAX(MUL(rhs, vals), x); 19 #endif 10 index += C; 11 } 22 // Now, derive f_k (versions differ based on the used semiring): 23 #ifdef USE_TROPICAL_SEMIRING 24 STORE(δf_k [i*C], x); // Just a store. 25 #elif defined USE_BOOLEAN_SEMIRING 25 #elif defined USE_BOOLEAN_SEMIRING 26 // First, derive f_k using filtering. 27 V g = LOAD(δg_{k-1} [i*C]); // Load the filter g_{k-1} . 28 x = CMP(AND(x, g), [0,0,0], NEQ); STORE(δx_k [i*C], x); 29 30 // Second, update distances d; depth is the iteration number. 31 V x_mask = x; x = MUL(x, [depth,,depth]); 32 x = BLEND(LOAD(δd [i*C]), x, x_mask); STORE(δd [i*C], x); 33 44 // Third, update the filtering term. 35 g = AND(NOT(x_mask), g); STORE(δg_k [i*C], g); 36 #elif defined USE_SELMAX_SEMIRING: 37 V pars = LOAD(δp_{k-1} [i*C]); // Load the required part of \mathbf{p}_{k-1} 38 V pars = BLEND([0,0,,0], NEQ); 39 pars = BLEND([0,0,,0], pars, pnz); STORE(δp_k [i*C], pars); 40 // Set new x_k vector. 41 // Set new x_k vector. 42 // Set new x_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, δv [i*C], tmpnz); STORE(δx_k [i*C], x); 45 #endif	16	x = OR(AND(rhs, vals), x);
<pre>18 x = MAX(MUL(rhs, vals), x); 19 #endif 20 index += C; 21 } 22 // Now, derive f_k (versions differ based on the used semiring): 23 #ifdef USE_TROPICAL_SEMIRING 24 STORE(& f_k[i*C], x); // Just a store. 25 #elif defined USE_BOOLEAN_SEMIRING 26 #lif defined USE_BOOLEAN_SEMIRING 27 V g = LOAD($\& g_{k-1}$[i*C]); // Load the filter g_{k-1}. 28 x = CMP(AND(x, g), [0,0,0], NEQ); STORE($\& x_k$[i*C], x); 29 30 // Second, update distances d; depth is the iteration number. 31 V x_mask = x; x = MUL(x, [depth,,depth]); 32 x = BLEND(LOAD($\& d[i*C]$), x, x_mask); STORE($\& d[i*C]$, x); 33 4 // Third, update the filtering term. 35 g = AND(NOT(x_mask), g); STORE($\& g_k$[i*C], g); 4 #elif defined USE_SELMAX_SEMIRING: 5 V pars = LOAD($\& p_{k-1}$[i*C]); // Load the required part of P_{k-1} 36 V pars = BLEND([0,0,,0], NEQ); 37 pars = BLEND([0,0,,0], NEQ); 38 pars = BLEND([0,0,,0], pars, pnz); STORE($\& p_k$[i*C], pars); 41 42 // Set new x_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, $\& v$[i*C], tmpnz); STORE($\& x_k$[i*C], x); 45 #endif 45 46 47 Complete the filtering term. 46 Complete the filtering term. 47 Pars = BLEND([0,0,,0], NEQ); 48 Pars = BLEND([0,0,,0], NEQ); 49 Pars = BLEND([0,0,,0], NEQ); 40 Pars = BLEND([0,0,,0], pars, pnz); STORE($\& p_k$[i*C], pars); 41 // Set new x_k vector. 42 // Set new x_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, $\& v$[i*C], tmpnz); STORE($\& x_k$[i*C], x); 45 #endif</pre>	17	#elif defined USE_SELMAX_SEMIRING SEL-MAX SEMIRING
<pre>19 #endif 20 index += C; 21 } 22 // Now, derive f_k (versions differ based on the used semiring): #ifdef USE_TROPICAL_SEMIRING 23 #ifdef USE_TROPICAL_SEMIRING 24 STORE(& f_k[i*C], x); // Just a store. 25 #elif defined USE_BOOLEAN_SEMIRING 26 // First, derive f_k using filtering. 27 V g = LOAD($\&g_{k-1}$[i*C]); // Load the filter g_{k-1}. 28 x = CMP(AND(x, g), [0,0,0], NEQ); STORE($\&x_k$[i*C], x); 30 // Second, update distances d; depth is the iteration number. 31 V x_mask = x; x = MUL(x, [depth,,depth]); 32 x = BLEND(LOAD($\&d$[i*C]), x, x_mask); STORE($\&d$[i*C], x); 33 // Third, update the filtering term. 35 g = AND(NOT(x_mask), g); STORE($\&g_k$[i*C], g); 36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD($\&p_{k-1}$[i*C]); // Load the required part of P_{k-1} 39 V pnz = CMP(pars, [0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE($\&p_k$[i*C], pars); 41 // Set new \mathbf{x}_k vector. 42 // Set new \mathbf{x}_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, $\&v$[i*C], tmpnz); STORE($\&x_k$[i*C], x); 45 #endif</pre>	18	x = MAX(MUL(rhs, vals), x);
<pre>1ndex += C; 1 } 1 // 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 2 // First, derive f_k using filtering. 2 // First, derive f_k using filtering. 2 // Store(&g_{k-1}[i*C]); // Load the filter g_{k-1}. 2 x = CMP(AND(x, g), [0,0,0], NEQ); STORE(&x_k[i*C], x); 3 // Second, update distances d; depth is the iteration number. 3 // Second, update distances d; depth is the iteration number. 4 // Third, update the filtering term. 5 g = AND(LOAD(&d[i*C]), x, x_mask); STORE(&d[i*C], x); 3 // Third, update the filtering term. 5 g = AND(NOT(x_mask), g); STORE(&g_k[i*C], g); 5 #elif defined USE_SELMAX_SEMIRING: 5 // Update parents. 5 // update parents. 5 // pars = LOAD(&p_{k-1}[i*C]); // Load the required part of P_{k-1} 5 // Set new x_k vector. 5 // Set new</pre>	19	#endlf
<pre>21 } 22 // Now, derive f_k (versions differ based on the used semiring): 23 #ifdef USE_TROPICAL_SEMIRING 24 STORE(&$f_k[i*C]$, x); // Just a store. 25 #elif defined USE_BOOLEAN_SEMIRING 26 // First, derive f_k using filtering. 27 V g = LOAD($\&g_{k-1}[i*C]$); // Load the filter g_{k-1}. 28 x = CMP(AND(x, g), [0,0,0], NEQ); STORE($\&x_k[i*C]$, x); 29 30 // Second, update distances d; depth is the iteration number. 31 V x_mask = x; x = MUL(x, [depth,,depth]); 32 x = BLEND(LOAD($\&d[i*C]$), x, x_mask); STORE($\&d[i*C]$, x); 34 34 // Third, update the filtering term. 35 g = AND(NOT(x_mask), g); STORE($\&g_k[i*C]$, g); 36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD($\&p_{k-1}[i*C]$); // Load the required part of P_{k-1} 39 V pars = BLEND([0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE($\&p_k[i*C]$, pars); 41 42 // Set new \mathbf{x}_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, $\&v[i*C]$, tmpnz); STORE($\&x_k[i*C]$, x); 45 #endif</pre>	20	index += C;
<pre>22 // Now, derive I_k (versions differ based on the used semiring): 23 #ifdef USE_TROPICAL_SEMIRING 24 STORE(&$f_k[i*C]$, x); // Just a store. 25 #elif defined USE_BOOLEAN_SEMIRING 26 // First, derive f_k using filtering. 27 V g = LOAD($\&g_{k-1}[i*C]$); // Load the filter g_{k-1}. 28 x = CMP(AND(x, g), [0,0,0], NEQ); STORE($\&x_k[i*C]$, x); 29 30 // Second, update distances d; depth is the iteration number. 31 V x_mask = x; x = MUL(x, [depth,,depth]); 32 x = BLEND(LOAD($\&d[i*C]$), x, x_mask); STORE($\&d[i*C]$, x); 34 34 // Third, update the filtering term. 35 g = AND(NOT(x_mask), g); STORE($\&g_k[i*C]$, g); 36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD($\&p_{k-1}[i*C]$); // Load the required part of P_{k-1} 39 V pars = BLEND([0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE($\&p_k[i*C]$, pars); 41 42 // Set new \mathbf{x}_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, $\&v[i*C]$, tmpnz); STORE($\&x_k[i*C]$, x); 45 #endif</pre>	21	$\{$
TROPICAL SEMIRING STORE ($\&f_k[i*C]$, x); // Just a store. TROPICAL SEMIRING $i = 1 \text{ for ef } (i = 1 \text{ for ef } (i = 2 \text$	22	// Now, derive \mathbf{I}_k (versions differ based on the used semiring):
<pre>STORE($a_{j_k}[i \times C], x$); // Just a store. #elif defined USE_BOOLEAN_SEMIRING f // First, derive f_k using filtering. V g = LOAD($a_{g_{k-1}}[i \times C]$); // Load the filter g_{k-1}. x = CMP(AND(x, g), [0,0,0], NEQ); STORE($a_{k_k}[i \times C], x$); // Second, update distances d; depth is the iteration number. V x_mask = x; x = MUL(x, [depth,,depth]); x = BLEND(LOAD($a_{d}[i \times C]$), x, x_mask); STORE($a_{d}[i \times C], x$); // Third, update the filtering term. g = AND(NOT(x_mask), g); STORE($a_{g_k}[i \times C], g$); #elif defined USE_SELMAX_SEMIRING: // Update parents. V pars = LOAD($a_{p_{k-1}}[i \times C]$); // Load the required part of p_{k-1} V pnz = CMP(pars, [0,0,,0], NEQ); pars = BLEND([0,0,,0], pars, pnz); STORE($a_{p_k}[i \times C], pars$); // Set new x_k vector. V tmpnz = CMP(x, [0,0,,0], NEQ); x = BLEND(x, $a_{U}[i \times C], tmpnz$); STORE($a_{k_k}[i \times C], x$); #endif</pre>	25	TROPICAL SEMIRING
BOOLEAN SEMIRING $//$ First, derive f_k using filtering. $Y = LOAD(\&g_{k-1}[i*C]); // Load the filter g_{k-1}.x = CMP(AND(x, g), [0,0,0], NEQ); STORE(\&x_k[i*C], x);//$ Second, update distances d ; depth is the iteration number. $Y x_mask = x; x = MUL(x, [depth,,depth]);$ $x = BLEND(LOAD(\&d[i*C]), x, x_mask); STORE(\&d[i*C], x);$ // Third, update the filtering term. $g = AND(NOT(x_mask), g); STORE(\&g_k[i*C], g);$ $#elif defined USE_SELMAX_SEMIRING:$ $Y pars = LOAD(\&p_{k-1}[i*C]); // Load the required part of P_{k-1}Y pnz = CMP(pars, [0,0,,0], NEQ);pars = BLEND([0,0,,0], pars, pnz); STORE(\&p_k[i*C], pars);// Set new x_k vector.Y tmpnz = CMP(x, [0,0,,0], NEQ);x = BLEND(x, &v[i*C], tmpnz); STORE(\&x_k[i*C], x);#endif$	24	#elif defined USE BOOLEAN SEMIRING
<pre>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 \mathbf{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</pre>	26	// First derive f, using filtering BOOLEAN SEMIRING
$x = CMP(AND(x, g), [0, 0, 0], NEQ); STORE(\&x_k[i*C], x);$ $y = CMP(AND(x, g), [0, 0, 0], NEQ); STORE(&x_k[i*C], x);$ $y = CMP(AND(x, g), [0, 0, 0], NEQ); STORE(&x_k[i*C], x);$ $x = BLEND(LOAD(&d[i*C]), x, x_mask); STORE(&d[i*C], x);$ $y = AND(NOT(x_mask), g); STORE(&g_k[i*C], g);$ $f = Iif defined USE_SELMAX_SEMIRING:$ $y = AND(NOT(x_mask), g); STORE(&g_k[i*C], g);$ $g = AND(NOT(x_mask), g); STORE(&g_k[i*C], g);$ $f = Iif defined USE_SELMAX_SEMIRING:$ $y = BLEND([0, 0,, 0], NEQ);$ $p = S = BLEND([0, 0,, 0], NEQ);$ $y = S = BLEND(x, &vector.$ $Y = BLEND(x, &vector.$	27	$V g = LOAD(8a_{L-1}[i*C])$; // load the filter g_{L-1} .
29 29 29 29 20 29 20 20 20 20 20 20 20 20 20 20	28	$x = CMP(AND(x, g), [0, 0,, 0], NEO); STORE(8x_1[i*C], x);$
<pre>30 // Second, update distances d; depth is the iteration number. 31 V x_mask = x; x = MUL(x, [depth,,depth]); 32 x = BLEND(LOAD(&d[i*C]), x, x_mask); STORE(&d[i*C], x); 33 // Third, update the filtering term. 34 g = AND(NOT(x_mask), g); STORE(&g_k[i*C], g); 36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD(&$p_{k-1}[i*C]$); // Load the required part of p_{k-1} 39 V pars = LOAD(&$p_{k-1}[i*C]$); // Load the required part of p_{k-1} 39 V pars = BLEND([0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE(&$p_k[i*C]$, pars); 41 42 // Set new \mathbf{x}_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, &$v[i*C]$, tmpnz); STORE(&$x_k[i*C]$, x); 45 #endif</pre>	29	(, (, (.), (,., (,,,,(,,,(,,,(,,,(,,,
31 V x_mask = x; x = MUL(x, [depth,,depth]); x = BLEND(LOAD(&d[i*C]), x, x_mask); STORE(&d[i*C], x); 33 4 // Third, update the filtering term. g = AND(NOT(x_mask), g); STORE(&g_k[i*C], g); 36 #elif defined USE_SELMAX_SEMIRING: 7 7 7 8 9 9 9 9 9 9 9 9 9 9 9 9 9	30	// Second. update distances \mathbf{d} : depth is the iteration number.
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	31	<pre>V x_mask = x; x = MUL(x, [depth,,depth]);</pre>
33 34 35 36 37 36 #elif defined USE_SELMAX_SEMIRING: 37 38 39 40 40 40 41 42 43 44 44 45 45 46 47 47 47 47 47 47 47 47 47 47	32	$x = BLEND(LOAD(\&d[i*C]), x, x_mask); STORE(\&d[i*C], x);$
<pre>34 // Third, update the filtering term. 35 g = AND(NOT(x_mask), g); STORE(&g_k[i*C], g); 36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD(&$p_{k-1}[i*C]$); // Load the required part of \mathbf{p}_{k-1} 39 V pnz = CMP(pars, [0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE(&$p_k[i*C]$, pars); 41 42 // Set new \mathbf{x}_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, &$v[i*C]$, tmpnz); STORE(&$x_k[i*C]$, x); 45 #endif</pre>	33	
35 $g = AND(NOT(x_mask), g); STORE(\&g_k[i*C], g);$ 36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD(& $p_{k-1}[i*C]$); // Load the required part of p_{k-1} 39 V pnz = CMP(pars, [0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE(& p_k [i*C], pars); 41 42 // Set new x_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 $x = BLEND(x, &v[i*C], tmpnz); STORE(&x_k[i*C], x);$ 45 #endif	34	// Third, update the filtering term.
36 #elif defined USE_SELMAX_SEMIRING: 37 // Update parents. 38 V pars = LOAD($&p_{k-1}[i*C]$); // Load the required part of \mathbf{p}_{k-1} 39 V pnz = CMP(pars, [0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE($&p_k[i*C]$, pars); 41 42 // Set new \mathbf{x}_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, $&v[i*C]$, tmpnz); STORE($&x_k[i*C]$, x); 45 #endif	35	$g = AND(NOT(x_mask), g); STORE(\&g_k[i*C], g);$
37 // Update parents. 38 V pars = LOAD($&p_{k-1}[i*C]$); // Load the required part of \mathbf{p}_{k-1} 39 V pnz = CMP(pars, [0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE($&p_k[i*C]$, pars); 41 42 // Set new \mathbf{x}_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, $&v[i*C]$, tmpnz); STORE($&x_k[i*C]$, x); 45 #endif	36	#elif defined USE_SELMAX_SEMIRING:
38 V pars = LOAD($p_{k-1}[i*C]$); // Load the required part of \mathbf{p}_{k-1} 39 V pnz = CMP(pars, [0,0,,0], NEQ); 40 pars = BLEND([0,0,,0], pars, pnz); STORE($p_k[i*C]$, pars); 41 42 // Set new \mathbf{x}_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, $v[i*C]$, tmpnz); STORE($x_k[i*C]$, x); 45 #endif	37	// Update parents.
<pre>39 V pnz = CMP(pars, [0,0,,0], NEQ); 940 pars = BLEND([0,0,,0], pars, pnz); STORE(&p_k[i*C], pars); 941 942 // Set new x_k vector. 943 V tmpnz = CMP(x, [0,0,,0], NEQ); 944 x = BLEND(x, &v[i*C], tmpnz); STORE(&x_k[i*C], x); 95 #endif</pre>	38	V pars = LOAD(& p_{k-1} [i*C]); // Load the required part of \mathbf{p}_{k-1}
<pre>40 pars = BLEND([0,0,,0], pars, pnz); STORE(&p_k[i*C], pars); 41 42 // Set new x_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, &v[i*C], tmpnz); STORE(&x_k[i*C], x); 45 #endif</pre>	39	<pre>V pnz = CMP(pars, [0,0,,0], NEQ);</pre>
<pre>41 42 44 45 44 44 44 45 45 45 45 45 45 45 45</pre>	40	pars = BLEND ([0,0,,0], pars, pnz); ST ORE (&p _k [i*C], pars);
<pre>42 // Set new x_k vector. 43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, &v[i*C], tmpnz); STORE(&x_k[i*C], x); 45 #endif</pre>	41	
43 V tmpnz = CMP(x, [0,0,,0], NEQ); 44 x = BLEND(x, &v[i*C], tmpnz); STORE(& x_k [i*C], x); 45 #endif	42	// Set new \mathbf{x}_k vector.
44 x = BLEND(x, &v[i*C], tmpnz); STORE(& x_k [i*C], x); 45 #endif	43	<pre>V tmpnz = CMP(x, [0,0,,0], NEQ);</pre>
45 #endif	44	$x = BLEND(x, \&v[i*C], tmpnz); STORE(\&x_k[i*C], x);$
	45	#endif

Detailed

formulations are

in the paper \odot

SELL-C-SIGMA + SEMIRINGS FORMULATIONS





 $\begin{array}{l} (X, op_1, op_2, el_1, el_2) \\ (\mathbb{R} \cup \{\infty\}, min, +, \infty, 0) \\ (\mathbb{R}, +, \cdot, 0, 1) \\ (\{0, 1\}, |, \&, 0, 1) \\ (\mathbb{R}, max, \cdot, -\infty, 1) \end{array}$





 $(X, op_1, op_2, el_1, el_2)$ $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$ $(\mathbb{R}, +, \cdot, 0, 1)$ $(\{0, 1\}, |, \&, 0, 1)$ $(\mathbb{R}, max, \cdot, -\infty, 1)$

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

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





 $(X, op_1, op_2, el_1, el_2)$ $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$ $(\mathbb{R}, +, \cdot, 0, 1)$ $(\{0, 1\}, |, \&, 0, 1)$ $(\mathbb{R}, max, \cdot, -\infty, 1)$

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

> Detailed formulations are in the paper ©







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

()

•



- Vertices are sorted by their degree
 - ρ_i : the degree of the ith vertex
 - $\hat{
 ho}$: the maximum degree

0

Assume tropical semiring





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



()



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



()



vertex

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





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





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





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





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





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

The state

Assume tropical semiring



3 Storage bound #*chunks* $\sum_{i=1}^{4} C \cdot \rho_{iC-1} \leq 2m + \hat{\rho}C$

Computational complexity bound

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



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

and the state

Assume tropical semiring









A COMPANY AND









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









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

10000 Mar 10



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

E with a second with the

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



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

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

C = 8



P < 3n/4



and a start of the





SlimSell

val	n.							
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

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	F					
2	4	9	-1						
1	7	-1	-1						
3	4	-1	-1						

The corresponding traversal is labelsetting, so...



SlimSell



the state of the

The corresponding traversal is labelsetting, so...



SlimSell



and and and the

The corresponding traversal is labelsetting, so...



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						









SI	im	S	el	L
SI	im	S	el	L

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					







Persona



Additional reduction required

SLIMSELL FURTHER OPTIMIZATIONS: SLIMCHUNK

Thread 1 SlimSell val Thread 2 Thread 3



200



PERFORMANCE QUESTIONS



PERFORMANCE QUESTIONS

Does using semirings result in different performance?



spcl.inf.ethz.ch

PERFORMANCE QUESTIONS

Does using semirings result in different performance?

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



spcl.inf.ethz.ch

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












CSCS Piz Daint & Piz Dora















NVIDIA Tesla K80 GPU Intel Xeon Phi KNL Intel Xeon CPU

CSCS Greina cluster

Intel Xeon CPU NVIDIA Tesla K20X GPU







PERFORMANCE ANALYSIS

TYPES OF GRAPHS



spcl.inf.ethz.ch ず @spcl_eth ETHZÜRICh

PERFORMANCE ANALYSIS TYPES OF GRAPHS

Synthetic graphs









[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.





[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] <u>https://snap.stanford.edu</u>

Real-world SNAP graphs [3]





Real-world SNAP graphs [3]





Web graphs



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







Semirings

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

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

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

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



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



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



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

$$n=2^{23}, ar{
ho}=16$$
 Xeon CPU, ${\cal C}=8$





PERFORMANCE ANALYSIS IMPACT FROM SLIMWORK

Kronecker power-law graphs

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

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 = 16log $\sigma \in \{20, 21, 22\}$ Dynamic scheduling





PERFORMANCE ANALYSIS COMPARISON TO GRAPH500

Kronecker power-law graphs



Intel KNL, C = 16log $\sigma \in \{20, 21, 22\}$ Dynamic scheduling





Kronecker power-law graphs

PERFORMANCE ANALYSIS SIZE ANALYSIS





Kronecker power-law graphs

PERFORMANCE ANALYSIS SIZE ANALYSIS





OTHER ANALYSES



OTHER ANALYSES









0.5

0.4

ر الا

e 0.3

0.1

0.0

0

1

Traditional

BFS

BFS-SpMV / SlimSell

2 3 4

Iteration











the Participant of the second















CONCLUSIONS



Sell-C-sigma for graphs







In a start is the local





Sell-C-sigma for graphs





WORK COMPL	RESENTATIONS EXITY: POWER-LAW G	IRAPHS	The maximum degree The probability of a vertex having degree $\alpha \alpha^{-\beta}$
Work bound W = O(D)	$n + Dm + D\overline{a}C$	We want a high-prob	ability
0		sound on this	
$P[\rho > \hat{\rho}]$			
A -			
the set of the second sec	with probability $1 - \frac{1}{\log n}$ all ve	artices have degree let	is than $\hat{\rho}$, we need:
o ensure inal	Cardena de C		
$(1 - P[\rho >$	$(\hat{\rho}])^n \le 1 - \frac{1}{\log n} \Leftrightarrow$		
$(1 - P[\rho >$	$(\hat{\rho}])^n \leq 1 - \frac{1}{\log n} \Leftrightarrow$ s inequality and (2) we get:		

SlimSell: vectorizable representation









Sell-C-sigma for graphs





A CONTRACTOR OF STREET, ST

GRAPH REPRESENTATIONS WORK COMPLEXITY: POWER-LAW	GRAPHS	The maximum degree: The probability of a vertex having degree ρ $\alpha \rho^{-\beta}$
$W = O(Dn + Dm + D \overline{D}C)$	We want a high-prol bound on this	sability
$r_1p > p_1$		
3 To ensure that with probability $1 - \frac{1}{\log n}$ $(1 - P[\rho > \hat{\rho}])^n \le 1 - \frac{1}{\log n} \Leftrightarrow$	Il vertices have degree k	iss than $\hat{\rho}_i$ we need:

SlimSell: vectorizable representation



SL Fu	.IM RT	SI	EL R (L DP	TIN	IZ/	ATIO	DN	s: 5	
SI	im	Sel	I							
	va	θ.								$f_i = A^T \otimes f_{i-1}$
	1	3	4	5	7	8	9	10	12	
- 18	1	3	5	6	7	9	11	12	-1	
- 18	1	3	5	6	7	9	10	-1	-1	The
- 18	1	3	5	6	8	10	11	-1	-1	corresponding
- 18	1	2	5	8	10	11				traversal is label-
- 18	2	3	6	7	9	12				setting, so
- 18	4	5	7	8	10	12				
- 18	2	5	9	11	-1	-1				() if f k is finite
- 18	5	8	9	12						f_{l-1} $h = h = h$
- 18	2	4	9	-1	L					$f_i^n = \{$
	1	7	-1	-1	н					$A^{T}_{k} \otimes f_{i-1}$ otherwise
	3	4	-1	-1	н					

SLIMSELL	PTIMIZATIONS: SLIMC	HUNK
SlimSell	Thread 1	SlimSell Thread 1
val		val
1	Thread 2	Thread 2 Thread 3

Performance

& space analysis

L 1				
30 (01B) 9 12 12 10 10 10 10 10 10 10 10 10 10 10 10 10	representation	iel-C-signa Sim		0 = D
Total siz	法法法法	di di s	h de	$\sigma=\sqrt{n}$





Sell-C-sigma for graphs





GRAPH REPRESENTATIONS WORK COMPLEXITY: GENERAL	BOUND 0 Vertices ρ_l : the r $\hat{\rho}$: the r Assume	are sorted by the degree of the ith maximum degree tropical semiring
1 The size of all the blocks (except the largest): $\sum_{l=2}^{n_{C}} C \cdot \rho_{lC-1} \leq m$	ea/cag	gedree
2 The size of the largest block: $\hat{\rho}C$	vertex	ve
0	0	

ETH zürich	and the spaces	-	NOT .	and and a second
GRAPH REPRESENTATION WORK COMPLEXITY: POWER-L	S LAW GRAPHS	•	The maximum The probability vertex having $\alpha \rho^{-\beta}$	degree:ρ / of a degree ρ:
$W = O(Dn + Dm + D\rho C)$ $P[\rho > \hat{\rho}]$	We want a hi bound	gh-prob on this	ability	
3 To ensure that with probability $1 - \frac{1}{\log n}$ $(1 - P[\rho > \hat{\rho}])^n \le 1 - \frac{1}{\log n}$	t ∉e all vertices have de ⇔	egree le	ss than $\hat{\rho}$, we n	eed:
with Bernouli's inequality and (2) $eta=O\left((lpha \log n)^{1/(eta-1)} ight)$	we get: W = O(Dn + Dr	n + D	$C(\alpha n \log n)^{1/2}$	^(β-1))

Thank you for your attention

SlimSell: vectorizable representation



SLIM Furti	SI	EL R (L DP	тім	z	ATI	DN	s: 5	
Slim	5el	I							
val	٥.								$f_i = A^T \otimes f_{i-1}$
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	The
1	3	5	6	8	10	11	-1	-1	corresponding
1	2	5	8	10	11				traversal is label-
2	3	6	7	9	12				setting, so
4	5	7	8	10	12				
2	5	9	11	-1	-1				(. * if f * is finite
5	8	9	12						f _{l-1} , , , , , , , , , , , , , , , , , , ,
2	4	9	-1						$h^{-} = \{$
1	7	-1	-1						$(A^r_k \otimes f_{i-1} \text{ otherwise})$
3	4	-1	-1						

SLIMSELL	PTIMIZATIONS: SLIMCH	UNK Additional reduction required
SlimSell	Thread 1	SlimSell Thread 1
val		val
	Thread 2	Thread 2

Performance

& space analysis

8 (3B) 0 3 2 5 2 5 2 5 2 5 2 5 2 5 2 5 2 5 2 5 2 5	representation AL Sel-C-al	igra Skridel g
Total siz	ն հեհե	





Sell-C-sigma for graphs



& space









COME THE

SEMIRINGS FOR BFS



Tropical semiring

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



Tropical semiring

 $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ $f_k = A'^T \otimes_T f_{k-1}$







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





Real semiring $(\mathbb{R}, +, \cdot, 0, 1)$ $f_k = A^T \otimes_R f_{k-1}$




Hadamard product
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
Real semiring

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

 $f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\sum_{l=0}^{k-1} f_l \right)$
 $distances = O(D)$
 $parents \in O(m)$
After
iterations





Hadamard product
Real semiring

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

 $f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\sum_{l=0}^{k-1} f_l\right)$
 $distances = O(D)$
 $parents \in O(m)$
After
iterations

Boolean semiring
({0,1}, , &, 0,1)
$f_k = [similar to Real]$
$distances \in O(D)$ iterations
parents $\in O(m)$





Hadamard product
Real semiring

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

 $f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\sum_{l=0}^{k-1} f_l\right)$
 $distances = O(D)$
 $parents \in O(m)$
After
iterations



Sel-max "semiring"

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

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

$$distances \in O(D)$$

$$parents \in O(1)$$
After iterations





Hadamard product
Real semiring

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

 $f_k = (A^T \otimes_R f_{k-1}) \odot_R \left(\sum_{l=0}^{k-1} f_l\right)$
 $distances = O(D)$
 $parents \in O(m)$
After
iterations



Sel-max "semiring"

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

$$f_{k} = \left(\overline{A^{T} \otimes_{\mathbb{R}} f_{k-1}}\right) - \left(\sum_{l=0}^{k-1} f_{l}\right)$$

$$distances \in O(D)$$

$$after$$

$$parents \in O(1)$$





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

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



Work bound

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

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

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



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

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

Work bound

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

We want a high-probability bound on this



• The maximum degree: $\hat{
ho}$

The probability of a vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

2



The maximum degree: $\hat{
ho}$

The probability of a vertex having degree ρ :

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

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

We want a high-probability bound on this

2

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



The maximum degree: $\hat{
ho}$ The probability of a

vertex having degree ρ :

1 Work bound $W = O(Dn + Dm + D\hat{\rho}C)$ We want a high-probability bound on this $P[\rho > \hat{\rho}] = \alpha \sum_{x=\hat{\rho}+1}^{n-1} x^{-\beta}$



The maximum degree: $\hat{\rho}$ The probability of a

vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

•

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



The maximum degree: $\hat{
ho}$ The probability of a

vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

•

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



The maximum degree: $\hat{\rho}$ The probability of a

vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

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

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



The maximum degree: $\hat{\rho}$ The probability of a

vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

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

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need: $(1 - P[\rho > \hat{\rho}])^n \le 1 - \frac{1}{\log n} \Leftrightarrow$



The maximum degree: $\hat{
ho}$ The probability of a

vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

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

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need: $(1 - P[\rho > \hat{\rho}])^n \le 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \ge 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$



The maximum degree: $\hat{\rho}$ The probability of a

vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

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

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need: $(1 - P[\rho > \hat{\rho}])^n \le 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \ge 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$

4



The maximum degree: $\hat{
ho}$ The probability of a

vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

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

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need: $(1 - P[\rho > \hat{\rho}])^n \le 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \ge 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$

With Bernoulli's inequality and 2 we get:



The maximum degree: $\hat{
ho}$ The probability of a

vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

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

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need: $(1 - P[\rho > \hat{\rho}])^n \le 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \ge 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$

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

The maximum degree: $\hat{\rho}$ The probability of a

vertex having degree ρ :

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

Work bound

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

We want a high-probability bound on this

2 - 2 - 2 - 2 - F 16

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

3 To ensure that with probability $1 - \frac{1}{\log n}$ all vertices have degree less than $\hat{\rho}$, we need: $(1 - P[\rho > \hat{\rho}])^n \le 1 - \frac{1}{\log n} \Leftrightarrow P[\rho > \hat{\rho}] \ge 1 - \left(1 - \frac{1}{\log n}\right)^{1/n}$

With Bernoulli's inequality and 2 we get:

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