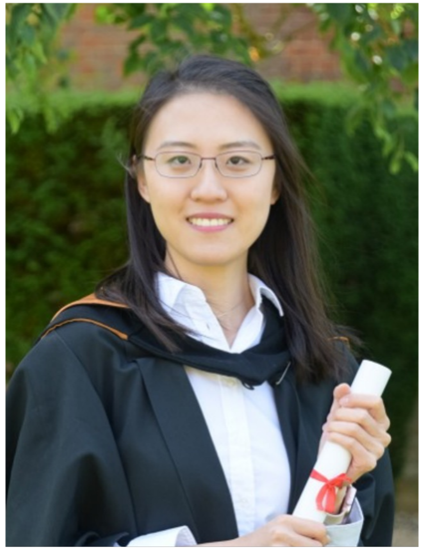
Sketching sparse low-rank matrices with near-optimal sample- and time-complexity

Xiaoqi (Shirley) Liu, Ramji Venkataramanan
University of Cambridge

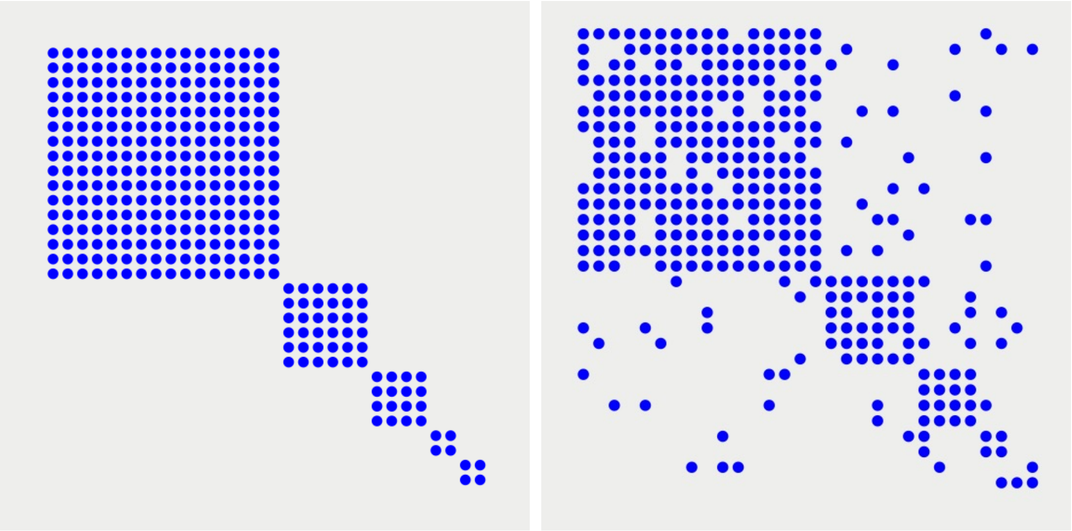


Motivating examples

Sparse and low-rank matrices arise in a variety of applications

Community detection

- Protein interaction data

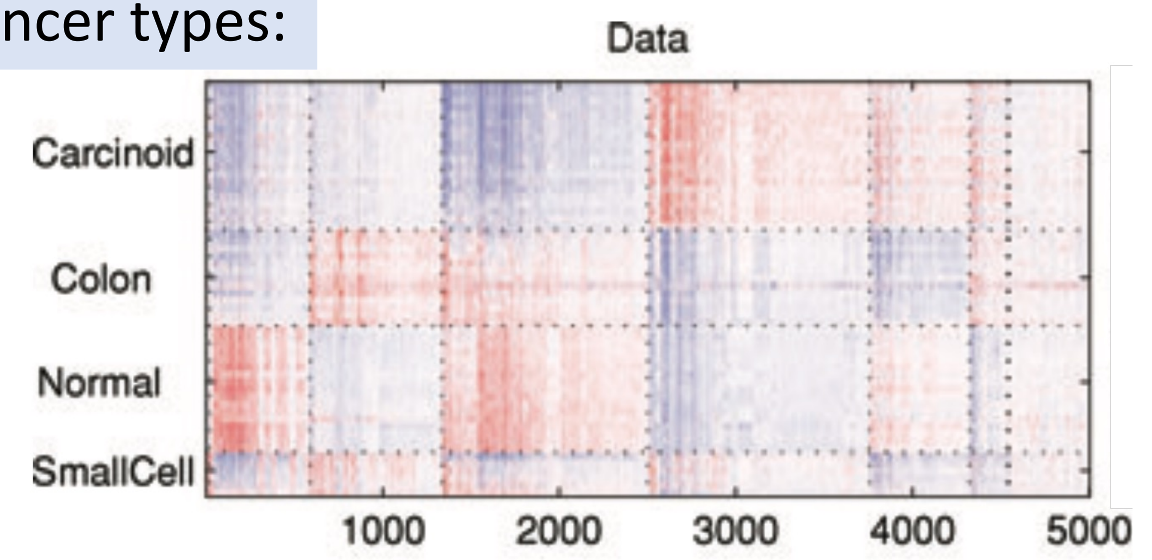


[Lee et al., 2010]

Bi-clustering

- Gene expression microarray data

Clustered cancer types:



Expression levels of genes

Matrix sketching problem

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{X})$$

Sketch vector $\boldsymbol{y} \in \mathbb{R}^m$

$m \ll n_1, n_2$

Data matrix $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$

$\mathcal{A}$: Pre-determined sketching operator

Recover

Rank-r

$$\boldsymbol{X} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

$\{\boldsymbol{u}_i, \boldsymbol{v}_i\}$ k-sparse

Goal of this work:

Design a **good** sketching operator $\mathcal{A}$ & an **efficient** algorithm to recover $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}$ from the sketch $\boldsymbol{y}$

[Woodruff, 2014]

$\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$

Rank-r

$\{\boldsymbol{u}_i, \boldsymbol{v}_i\}$ k-sparse

Related work

Sketching low-rank and sparse matrices:

- Convex optimization-based algorithms: [Candès and Plan, 2011]
  - Running time $\propto$ poly(n_1, n_2)
  - Sample complexity $\propto$ polylog(n_1, n_2)

Goal of this work:

- Design a **good** sketching operator $\mathcal{A}$ & an **efficient** algorithm to recover $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}$ from the sketch $\boldsymbol{y} = \mathcal{A}(\boldsymbol{X})$
- Sample complexity & running time $\propto k$ (near optimal compared to degrees of freedom)

Compressed sensing:

- Sketching operator adapted from [Bakshi et al., 2016] [Li et al., 2019]

Sketching operator (for symmetric matrices)

$$\boldsymbol{X} = \sum_{i=1}^{r} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^T \in \mathbb{R}^{n \times n}$$

Rank-r

$\{\boldsymbol{v}_i\}$ k-sparse

Vectorised matrix: $\boldsymbol{x} \in \mathbb{R}^{\tilde{n}}$ with $\tilde{k}$ nonzeros

Sketching matrix: $\boldsymbol{B} \in \mathbb{C}^{2R \times \tilde{n}}$

Sketch: $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{x} \in \mathbb{C}^{2R}$

$\boldsymbol{B} \in \mathbb{C}^{2R\times\tilde{n}}$ = Combines 1) A sparse parity check matrix $\boldsymbol{H} \in \{0,1\}^{R\times\tilde{n}}$

2) A two-row DFT matrix $\boldsymbol{S} \in \mathbb{C}^{2\times\tilde{n}}$

Example: $\tilde{n} = 6, R = 4$

entries in $\boldsymbol{x}$

$$\boldsymbol{H} = \begin{bmatrix} \boxed{1 & 1 & 1 & 0 & 1 & 1} \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\boxed{\boldsymbol{S} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & W & \dots & W^5 \end{bmatrix}}$$

$$W = \exp\left(\frac{2\pi i}{6}\right)$$

2 ones in each column

[Li et al., 2019]

$\boldsymbol{B} \in \mathbb{C}^{2R\times\tilde{n}}$ = Combines 1) A sparse parity check matrix $\boldsymbol{H} \in \{0,1\}^{R\times\tilde{n}}$

2) A two-row DFT matrix $\boldsymbol{S} \in \mathbb{C}^{2\times\tilde{n}}$

Example: $\tilde{n} = 6, R = 4$

entries in $\boldsymbol{x}$

$$\boldsymbol{B} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & W & W^2 & 0 & W^4 & W^5 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & W^3 & W^4 & W^5 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & W & W^2 & W^3 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{S} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & W & \dots & W^5 \end{bmatrix}$$

$$W = \exp\left(\frac{2\pi \mathrm{i}}{6}\right)$$

[Li et al., 2019]

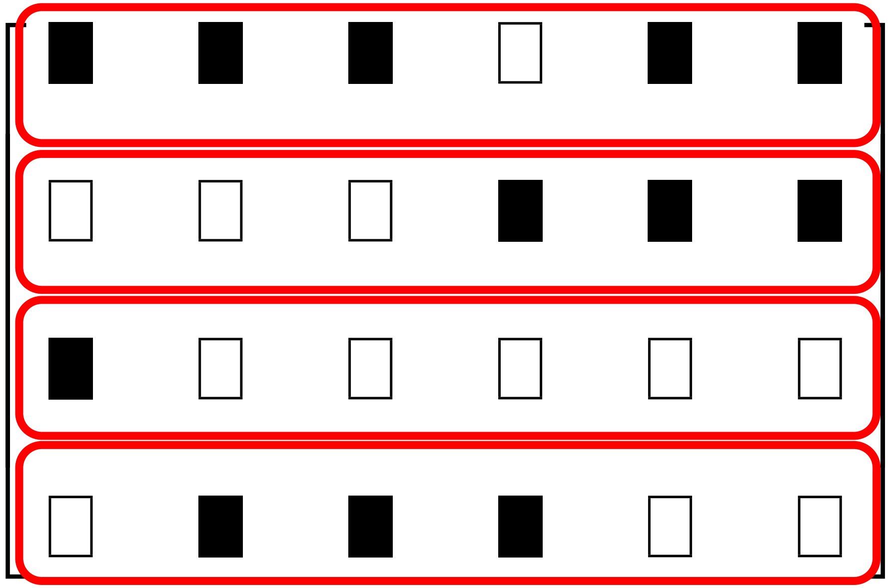
$\boldsymbol{B} \in \mathbb{C}^{2R\times\tilde{n}}$ = Combines 1) A sparse parity check matrix $\boldsymbol{H} \in \{0,1\}^{R\times\tilde{n}}$

2) A two-row DFT matrix $\boldsymbol{S} \in \mathbb{C}^{2\times\tilde{n}}$

Example: $\tilde{n} = 6, R = 4$

entries in $\boldsymbol{x}$

$B =$



$\square$: zeros $\blacksquare$: $1, W^i$

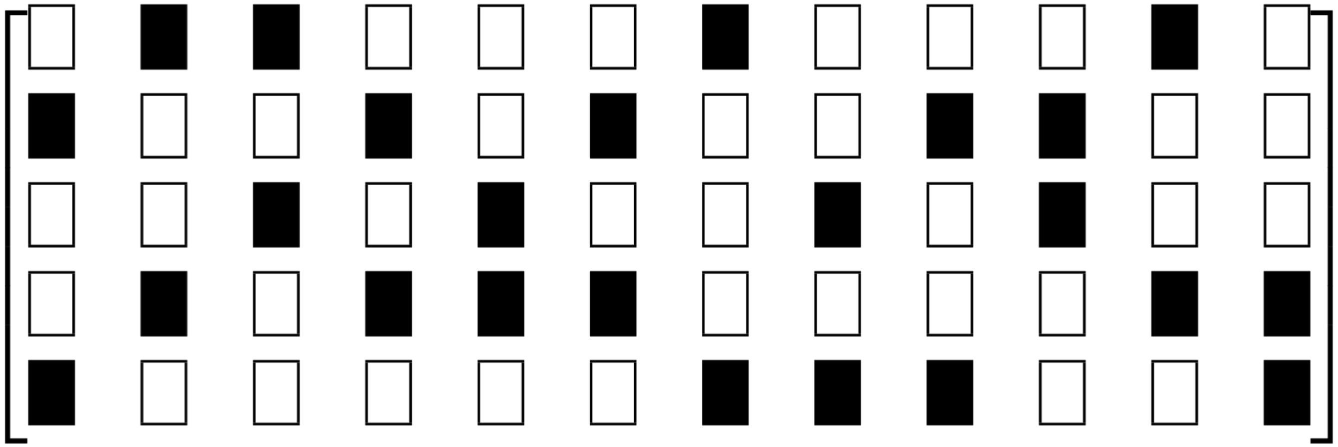
$\boldsymbol{B} \in \mathbb{C}^{2R\times\tilde{n}}$ = Combines 1) A sparse parity check matrix $\boldsymbol{H} \in \{0,1\}^{R\times\tilde{n}}$

2) A two-row DFT matrix $\boldsymbol{S} \in \mathbb{C}^{2\times\tilde{n}}$

Example: $\tilde{n} = 12, R = 5$

entries in $\boldsymbol{x}$

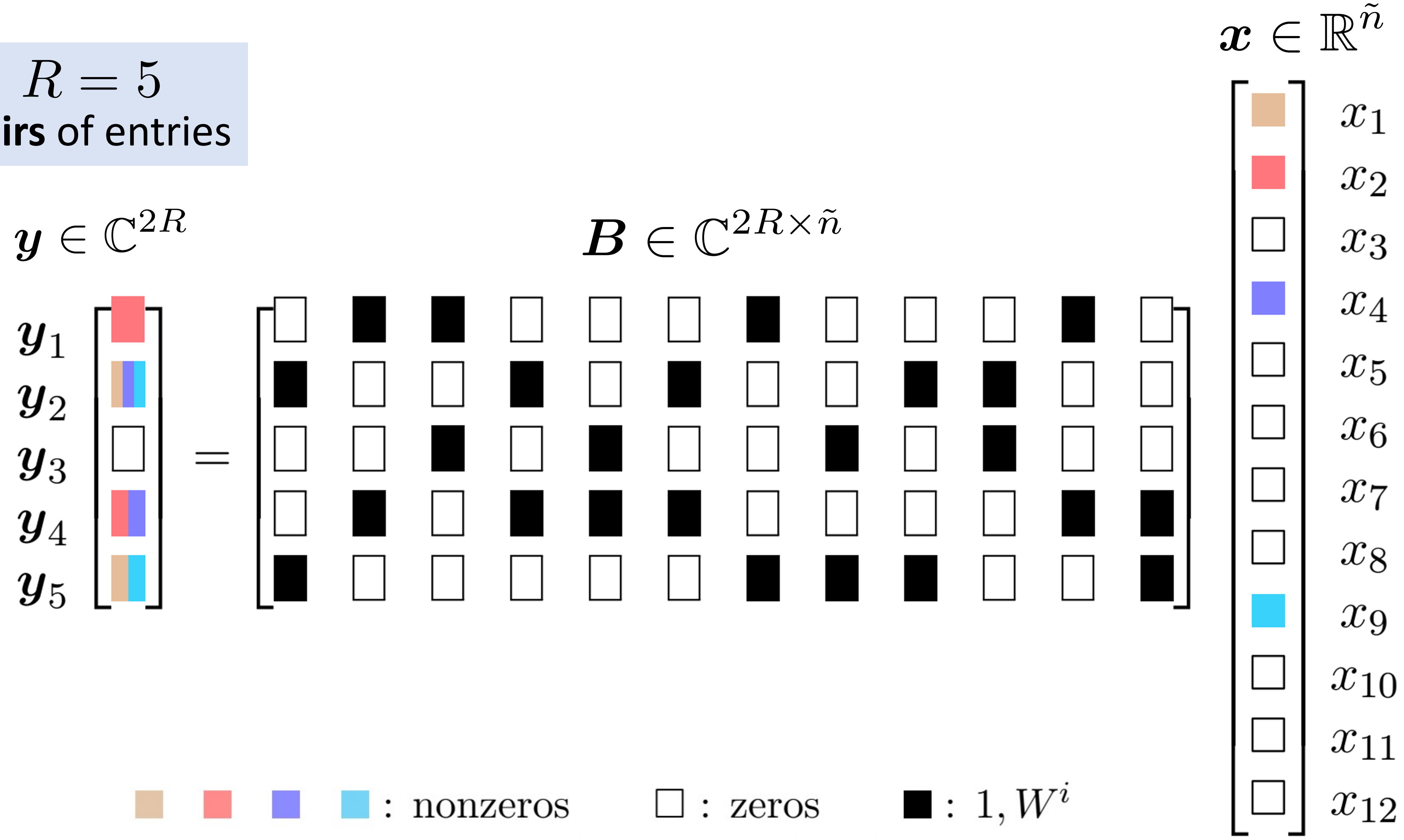
$\boldsymbol{B} =$



□ : zeros ■ : $1, W^i$

Vectorised matrix

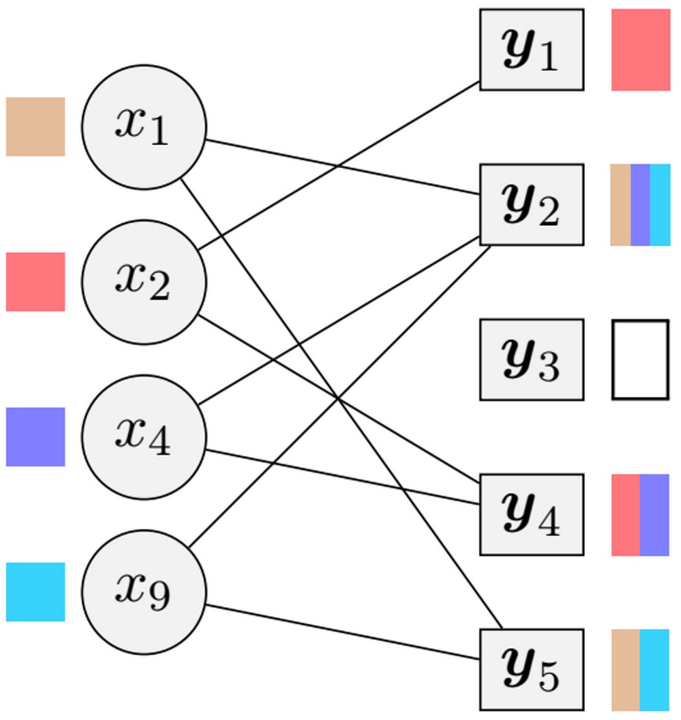
$R = 5$
pairs of entries



Stage A of the algorithm (illustrated on a sparse graph)

$\tilde{k} = 4$
nonzero entries of $\boldsymbol{x}$
(vectorised matrix)

$R = 5$
pairs of entries
in the sketch $\boldsymbol{y}$



$t = 0$

Analogous to the peeling decoder for LDPC codes over a BEC:

Identifies a singleton
(i.e. a degree-1 right node)

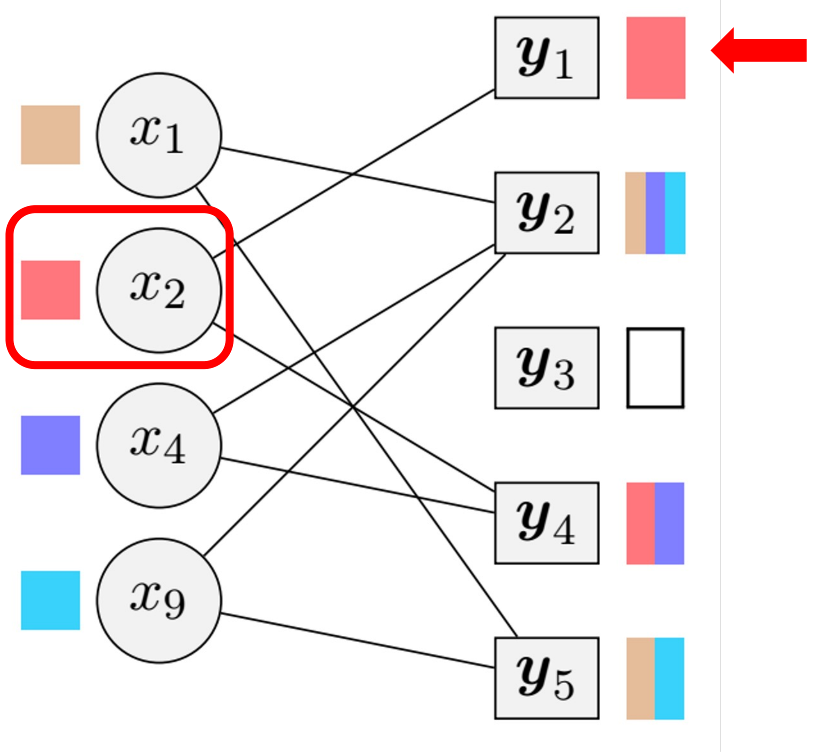
Recovers the nonzero entry

Peels off the nonzero entry

Stage A of the algorithm (illustrated on a sparse graph)

$\tilde{k} = 4$
nonzero entries of $\boldsymbol{x}$
(vectorised matrix)

$R = 5$
pairs of entries
in the sketch $\boldsymbol{y}$



$t = 0$

Analogous to the peeling decoder for LDPC codes over a BEC:

Identifies a singleton (i.e. a degree-1 right node)

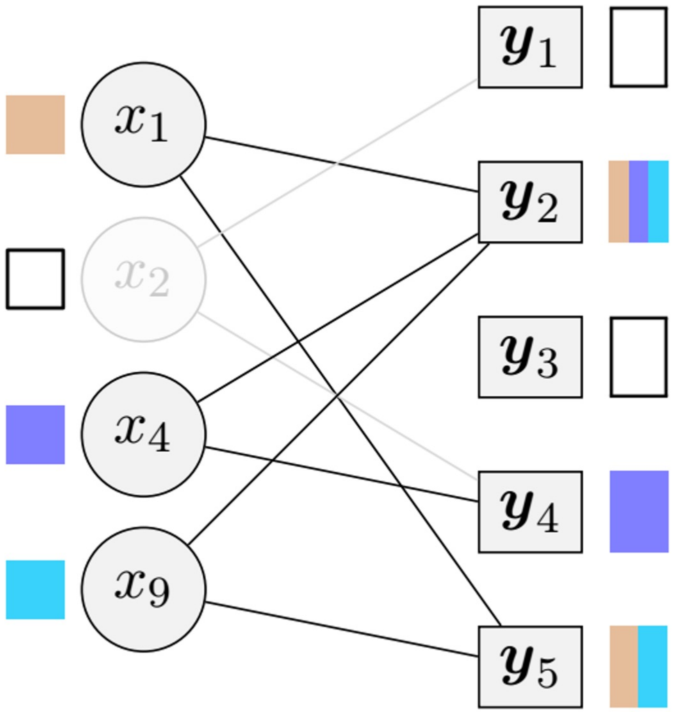
Recovers the nonzero entry

Peels off the nonzero entry

Stage A of the algorithm (illustrated on a sparse graph)

$\tilde{k} = 4$
nonzero entries of $\boldsymbol{x}$
(vectorised matrix)

$R = 5$
pairs of entries
in the sketch $\boldsymbol{y}$



$t = 1$

Analogous to the peeling decoder for LDPC codes over a BEC:

Identifies a singleton (i.e. a degree-1 right node)

↓

Recovers the nonzero entry

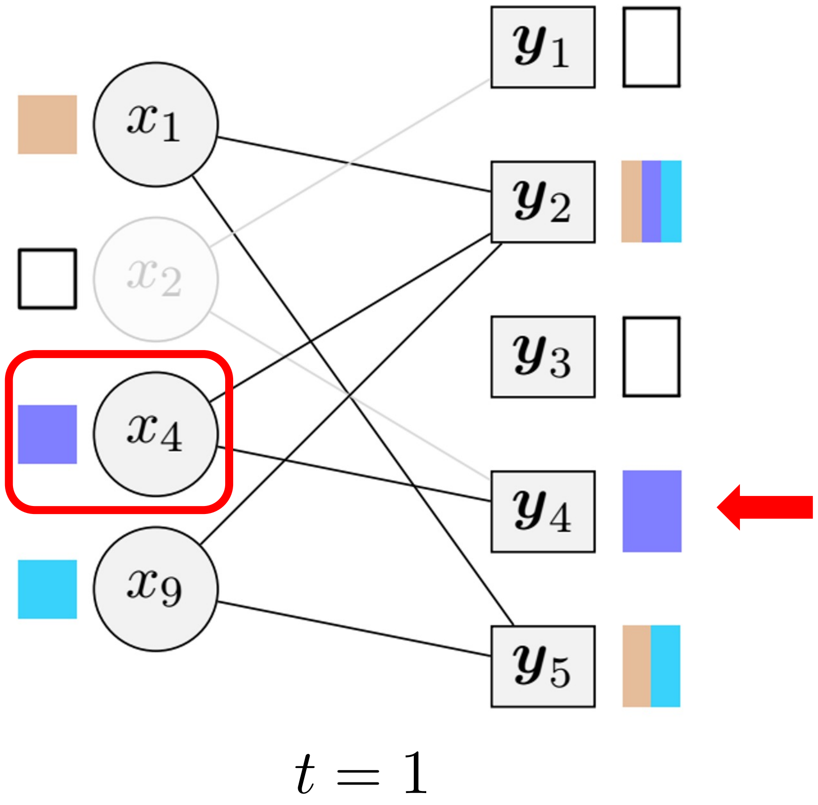
↓

Peels off the nonzero entry

Stage A of the algorithm (illustrated on a sparse graph)

$\tilde{k} = 4$
nonzero entries of $\boldsymbol{x}$
(vectorised matrix)

$R = 5$
pairs of entries
in the sketch $\boldsymbol{y}$



Analogous to the peeling decoder for LDPC codes over a BEC:

Identifies a singleton (i.e. a degree-1 right node)

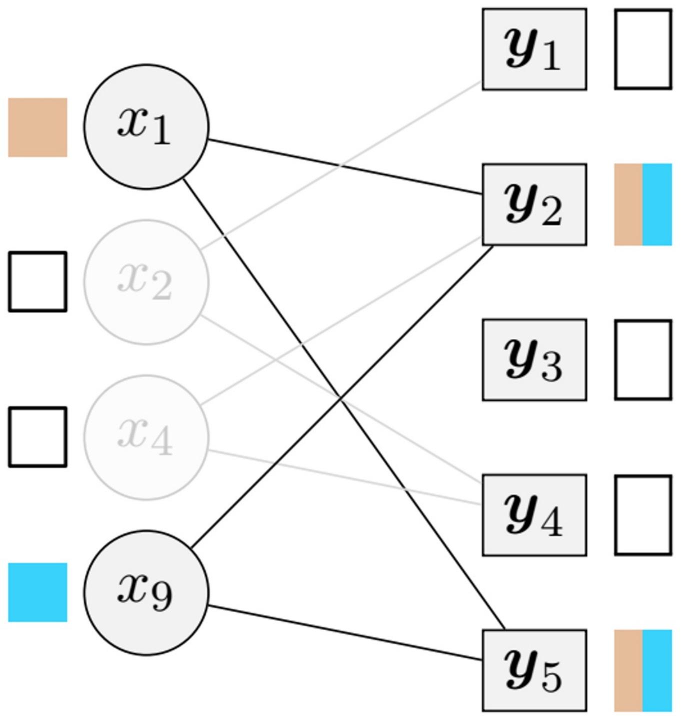
Recovers the nonzero entry

Peels off the nonzero entry

Stage A of the algorithm (illustrated on a sparse graph)

$\tilde{k} = 4$
nonzero entries of $\boldsymbol{x}$
(vectorised matrix)

$R = 5$
pairs of entries
in the sketch $\boldsymbol{y}$



$t = 2$

Analogous to the peeling decoder for LDPC codes over a BEC:

Identifies a singleton (i.e. a degree-1 right node)

↓

Recovers the nonzero entry

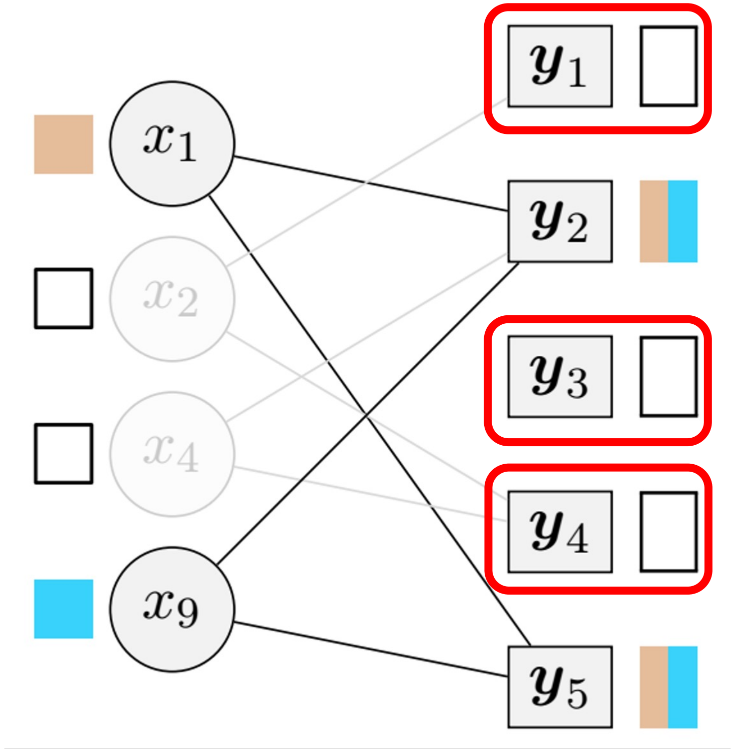
↓

Peels off the nonzero entry

Stage A of the algorithm (illustrated on a sparse graph)

$\tilde{k} = 4$
nonzero entries of $\boldsymbol{x}$
(vectorised matrix)

$R = 5$
pairs of entries in the sketch $\boldsymbol{y}$



$t = 2$

Degree-0 right nodes involve only zero-valued x_i's

Analogous to the peeling decoder for LDPC codes over a BEC:

Identifies a singleton (i.e. a degree-1 right node)

Recovers the nonzero entry

Peels off the nonzero entry

Stage B of the algorithm

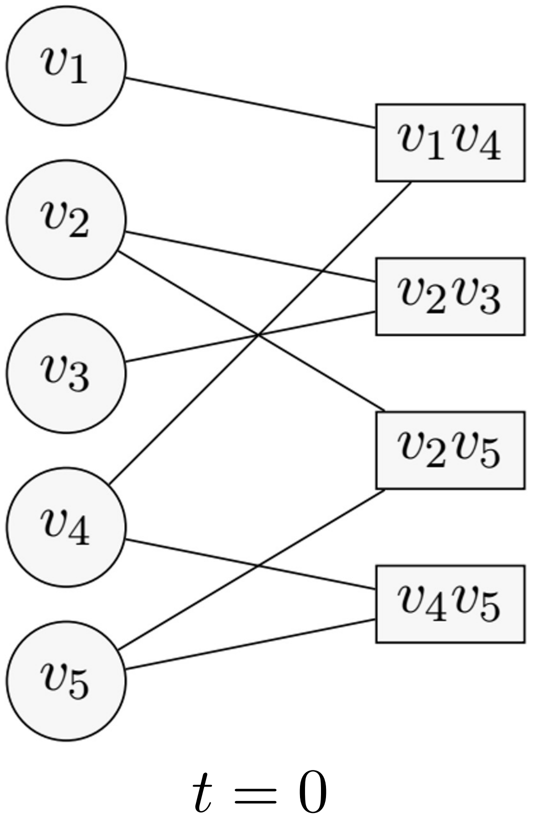
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

4 nonzero pairwise products recovered in Stage A



Stage B of the algorithm

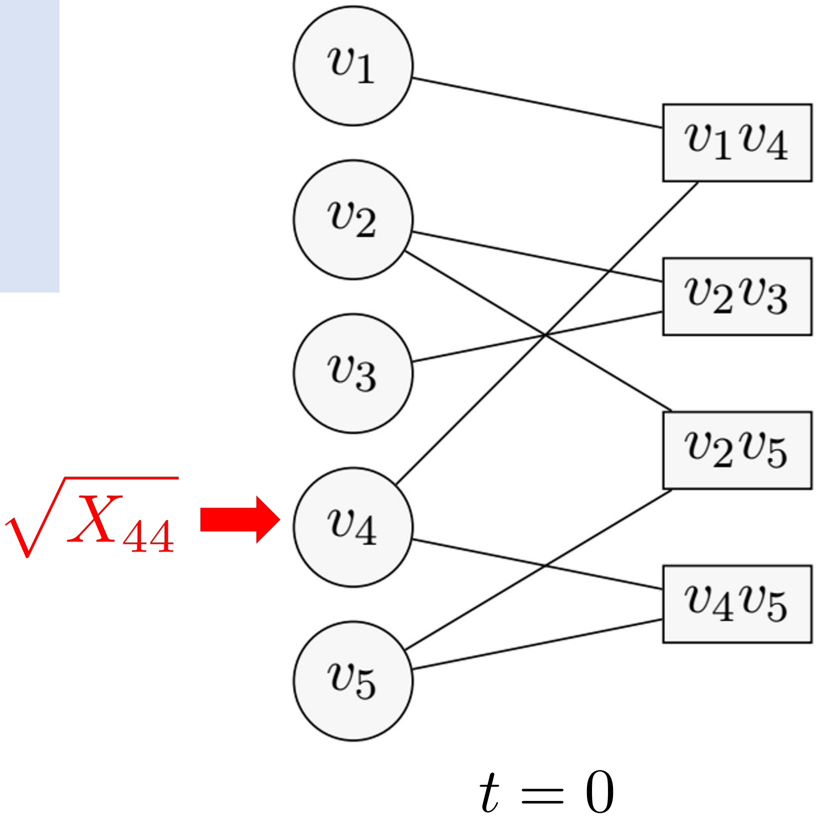
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

4 nonzero pairwise products recovered in Stage A



Stage B of the algorithm

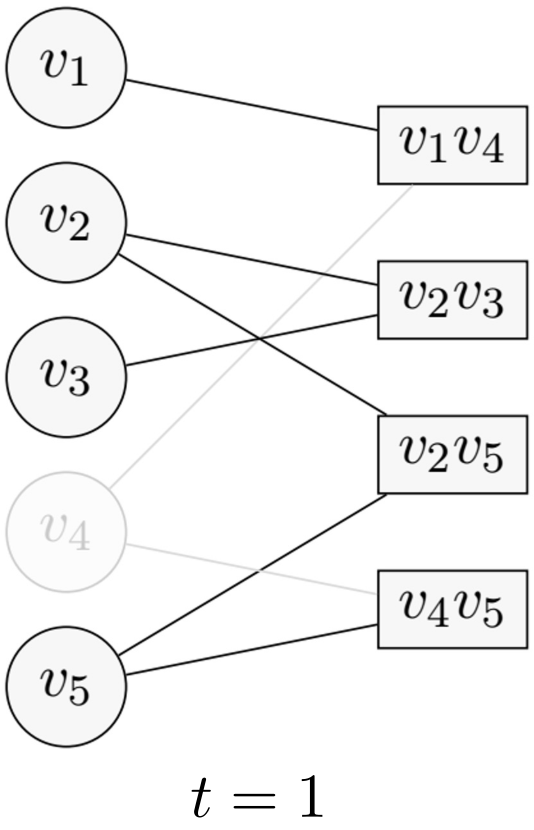
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



Stage B of the algorithm

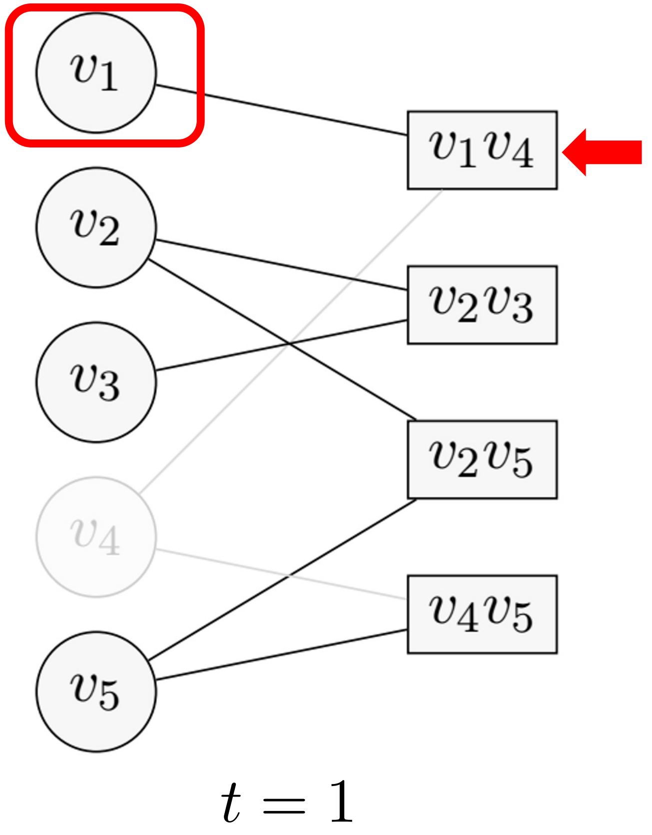
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



Stage B of the algorithm

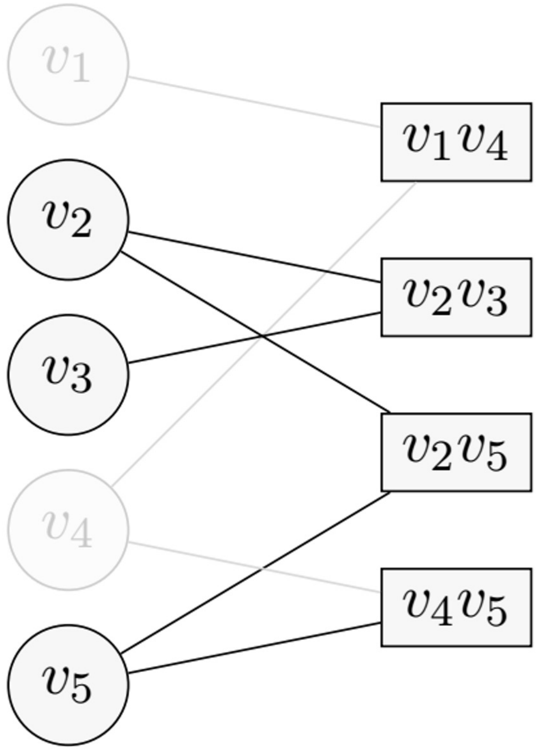
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



$t = 2$

Stage B of the algorithm

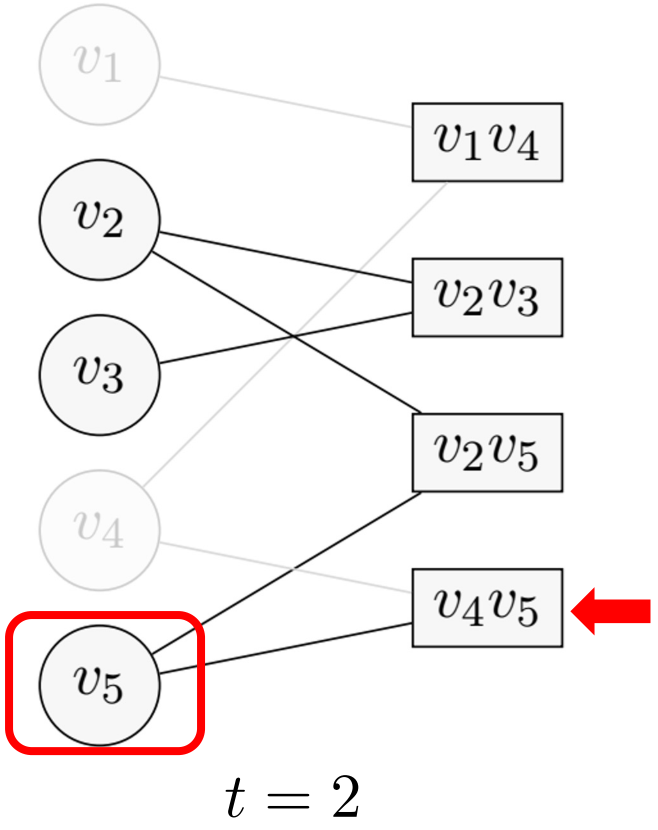
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



Stage B of the algorithm

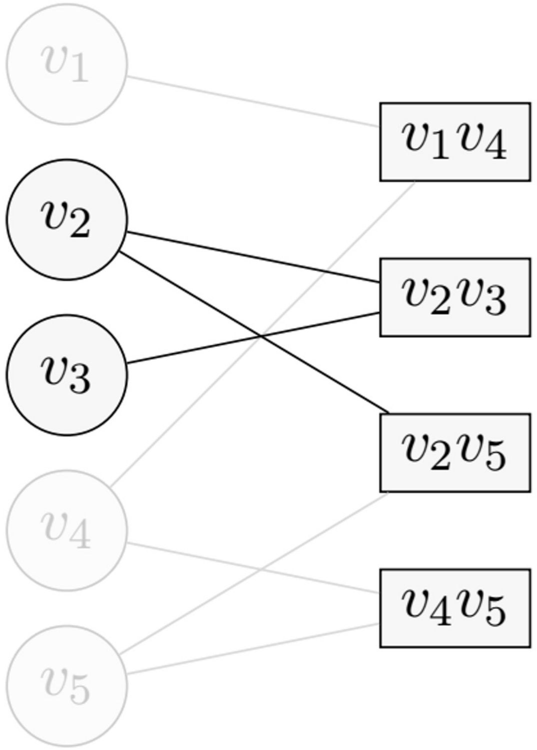
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



$t = 3$

Stage B of the algorithm

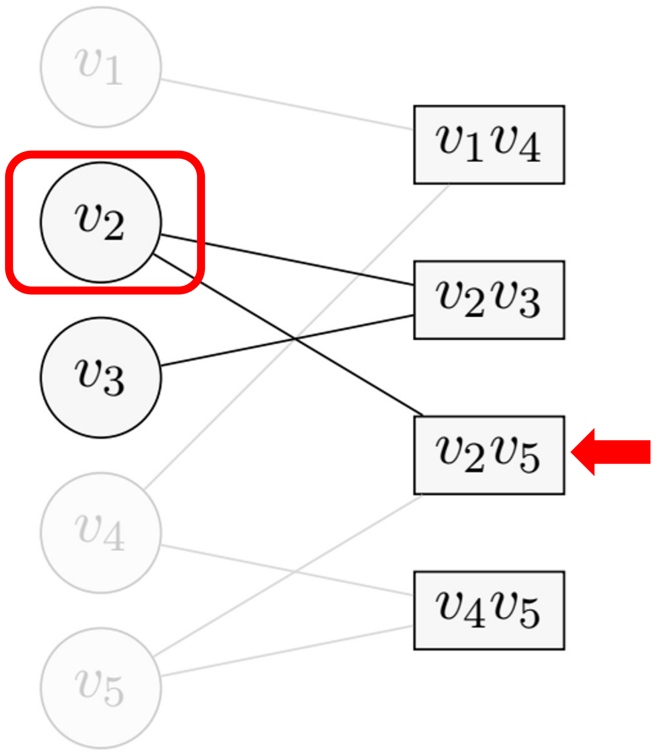
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



$t = 3$

Stage B of the algorithm

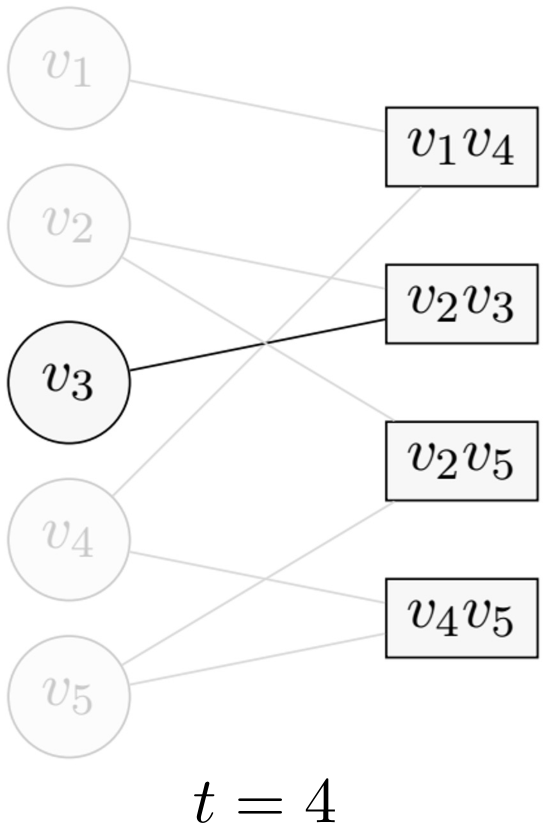
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



Stage B of the algorithm

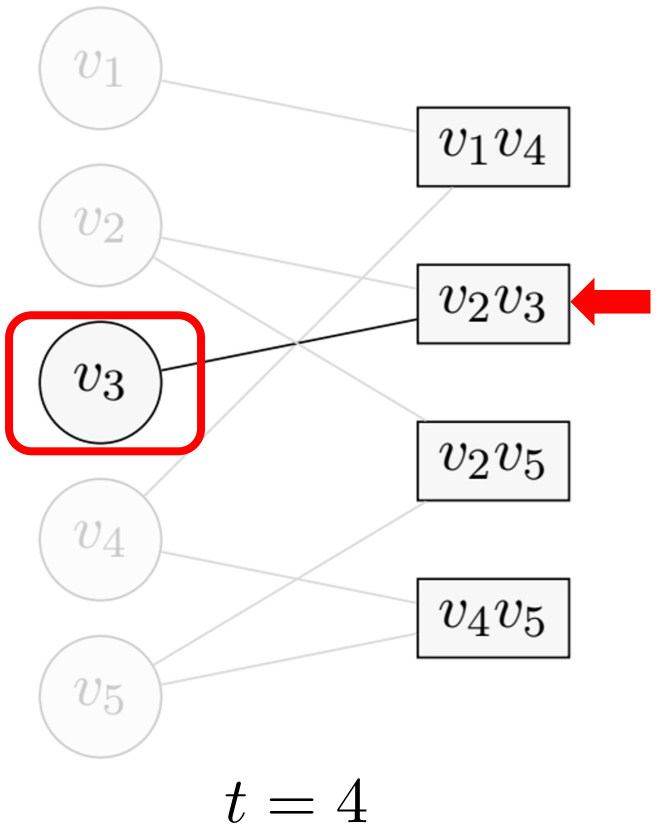
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



Stage B of the algorithm

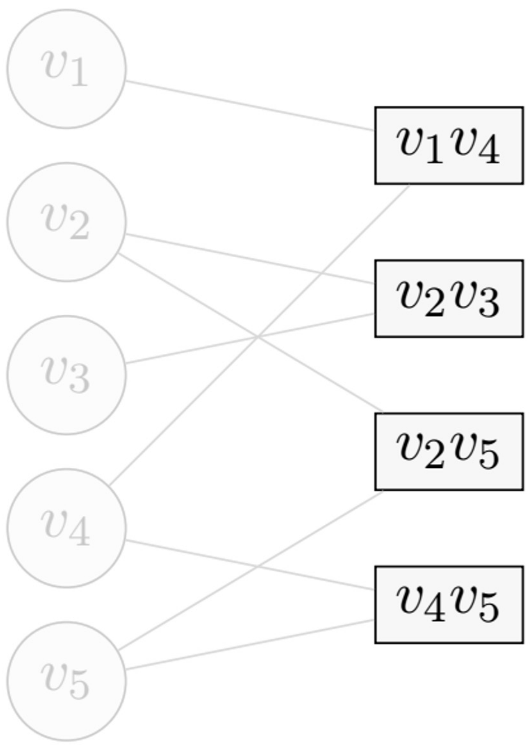
Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



$t = 5$

Stage B of the algorithm

Rank-1 case:

- $\boldsymbol{X} = \boldsymbol{v}\boldsymbol{v}^T, \boldsymbol{v}$ is k-sparse
- Matrix entries are of the form $\{v_i v_j\}$

Goal: solve for the nonzeros of $\boldsymbol{v}$ i.e. $\{v_i\}$ from the $\{v_i v_j\}$ recovered in Stage A

General rank-r case:

Locations of the nonzeros of $\{\boldsymbol{v}_i\}$ disjoint:

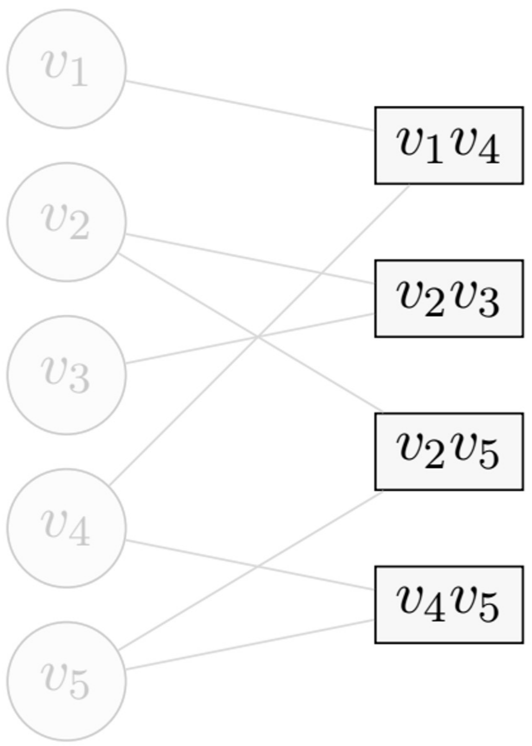
- r peeling processes in parallel

Locations of the nonzeros of $\{\boldsymbol{v}_i\}$ may overlap:

- Stage B **not** applicable
- Stage A + SVD on nonzero submatrix

$k = 5$ nonzero entries of $\boldsymbol{v}$

3 nonzero pairwise products recovered in Stage A



$t = 5$

Main result: non-asymptotic guarantees

Degrees of freedom of $\boldsymbol{X}$: $\mathcal{O}(rk)$

Theorem 1 (Symmetric case). Consider $\boldsymbol{X} = \sum_{i=1}^{r} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^T$, where each $\boldsymbol{v}_i$ has k nonzero entries with the locations of the nonzeros being uniformly random. For sufficiently large k, the proposed sketching scheme has the following guarantees.

1) For $r = 1$ or $r > 1$ with the supports of $\{\boldsymbol{v}_i\}$ disjoint, fix any $\delta \in (0, 1)$, the two-stage algorithm recovers $\{\boldsymbol{v}_i\}$ (up to a sign ambiguity) from the sketch of size

$$\boxed{m = \frac{3rk^2}{\delta \ln k}}$$

Near-optimal; only $\propto k$

with probability at least $\boxed{1 - 2r\exp(-\frac{1}{30}k^{1-\delta})}$ with running time $\boxed{\mathcal{O}(rk^2/\ln k)}$.

2) When $r > 1$ and the supports of $\{\boldsymbol{v}_i\}$ may overlap, with probability at least $\boxed{1 - \mathcal{O}(k^{-2})}$, stage A of the algorithm followed by eigendecomposition of the recovered nonzero submatrix recovers $\{\boldsymbol{v}_i\}$ from a sketch of size $\boxed{m = 2rk^2}$ with running time $\boxed{\mathcal{O}\left((rk)^3\right)}$.

Analogous theorem for general non-symmetric matrices: [Liu and Venkataramanan, 2022] arXiv:2205.06228.

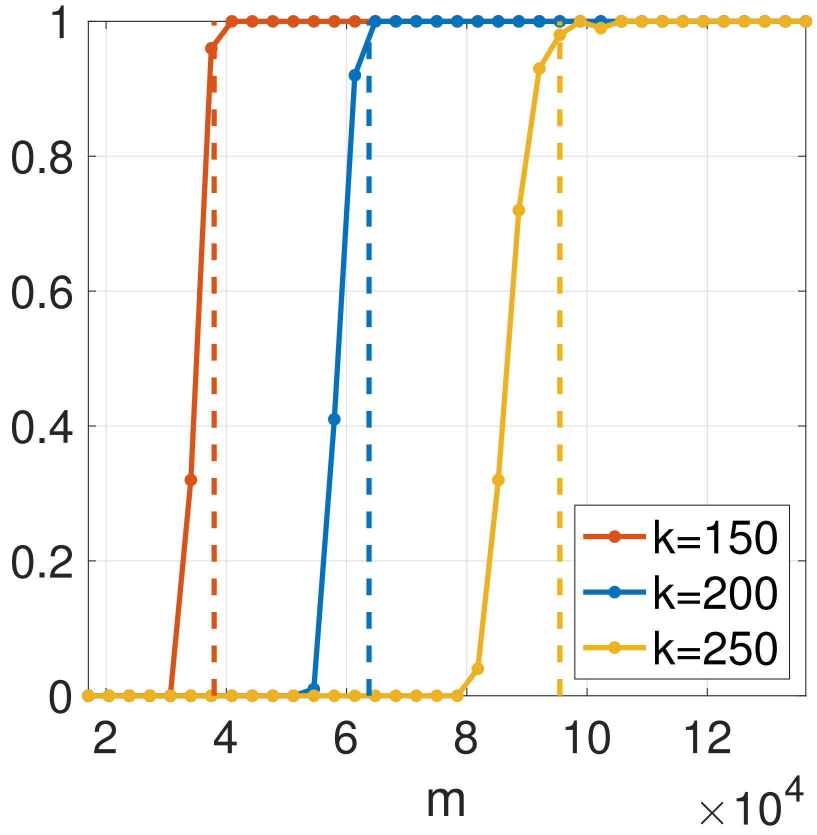
Proof ideas

- Analyse random graph processes representing the two stages of the algorithm
- Track the number of degree-1 right nodes over time
- Handle intricate dependencies among graph variables using Negative Association

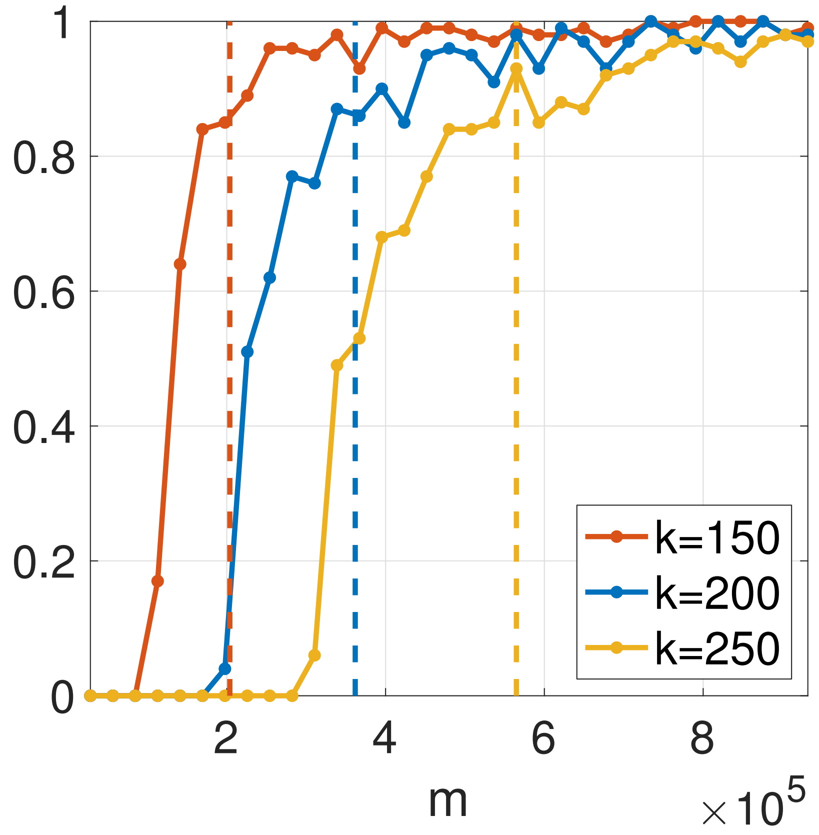
- Stage A similar to peeling decoder for LDPC codes over BEC but graph degrees grow with k

Numerical results (of symmetric matrices)

Disjoint supports:



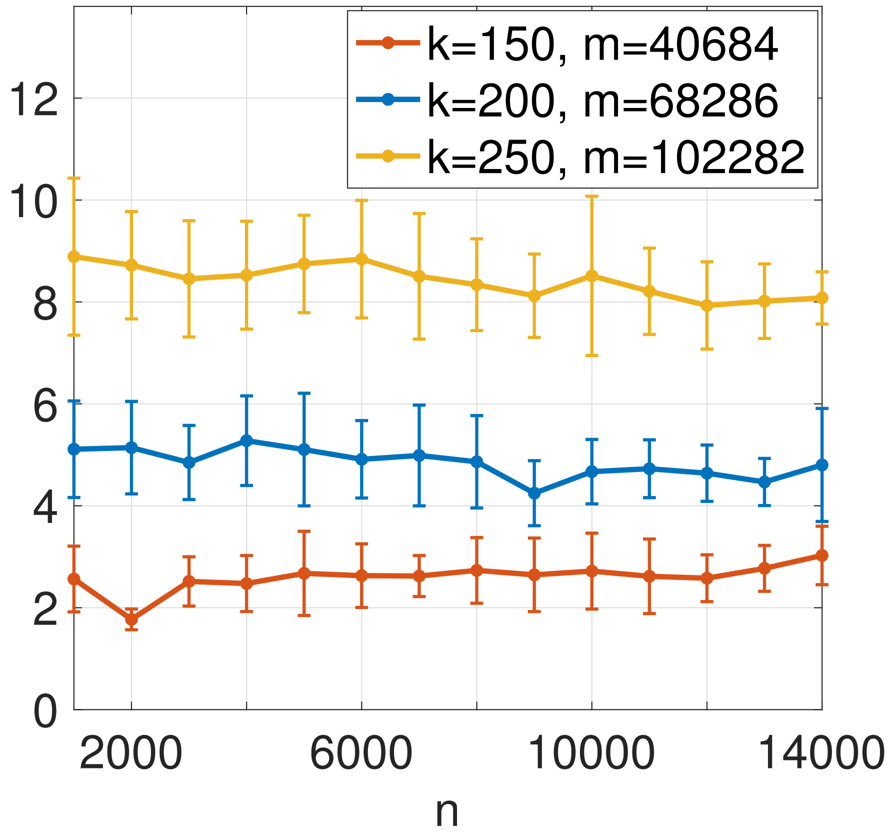
Overlapping supports:



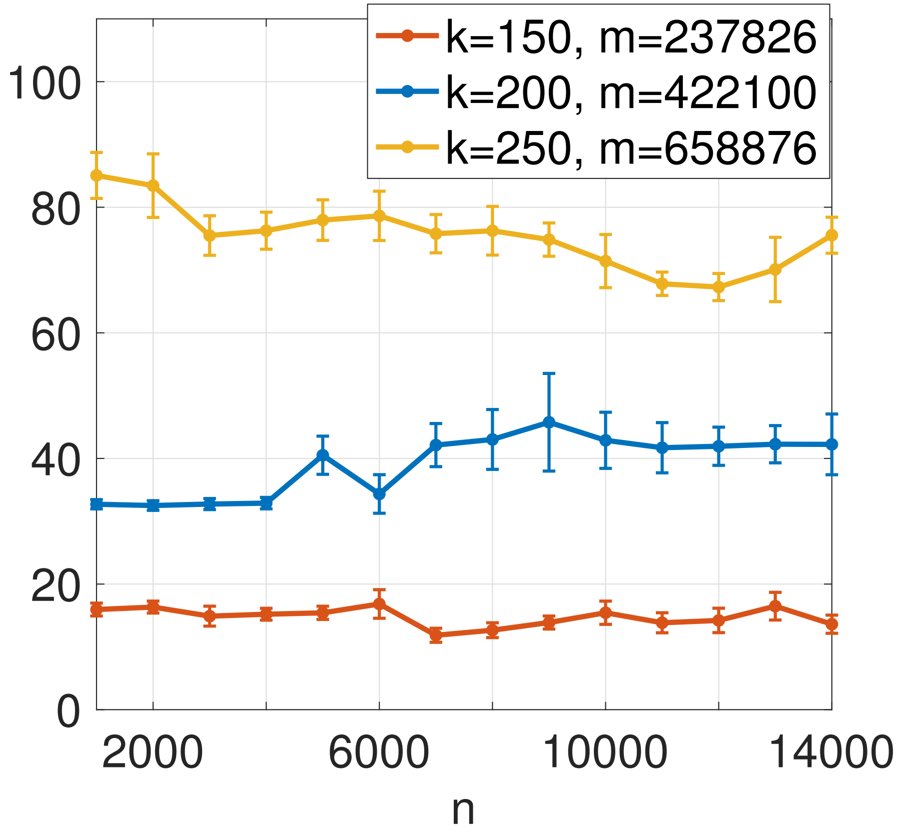
Probability of exact recovery of $\{\boldsymbol{v}_i\}$ versus # measurements m for different sparsity levels k. ($n = 10^4$, $r = 3$, 100 trials). Dashed lines indicate the # measurements m prescribed by Theorem 1.

Numerical results (of symmetric matrices)

Disjoint supports:



Overlapping supports:



Running time of recovery algorithm versus ambient dimension n, for different sparsity levels k and # measurements m. ($r = 3$, 50 trials) Error bars indicate one standard deviation.

Xiaoqi (Shirley) Liu: xl394@cam.ac.uk

Summary & Future work

Main contribution: For $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$, $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}$ k-sparse

- Proposed a sketching operator $\mathcal{A}$ based on LDPC codes & an efficient iterative algorithm to recover $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}$ from the sketch $\boldsymbol{y} = \mathcal{A}(\boldsymbol{X})$
- Proved finite-k theoretical guarantees: near-optimal sample and time complexities which only $\propto k$ (corroborated by numerical results)

Extension to tackle sparse low-rank matrices + **small** noise:

Liu, X. and Venkataramanan, R. (2022). Sketching sparse low-rank matrices with near-optimal sample- and time-complexity. **arXiv:2205.06228**.

Future directions:

- Extend Stage B of the algorithm when $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}$ have overlapping supports
- For the noisy case: reduce complexity; derive theoretical guarantees