Near-Optimal Distance Emulator for Planar Graphs

Hsien-Chih Chang, Paweł Gawrychowski, Shay Mozes, and Oren Weimann

Slides by Shay Mozes

IDC HERZLIYA

Distance Emulators

- let G be a graph with n nodes,
  let T be a subset of k vertices of G (terminals).

- a distance emulator is a small graph H with $T \subseteq V(H)$ that preserves the distances between all pairs of terminals.

Distance Emulators

- let G be a graph with n nodes,
  let T be a subset of k vertices of G (terminals).

- a distance emulator is a small graph H with $T \subseteq V(H)$ that preserves the distances between all pairs of terminals.

- related concepts:
  - small distance preserving minor/subgraph.
  - compressed representation of the distances (not a graph).

Prior Results

- minor preservers: $\Omega(k^2)$ lower bound even for unweighted grids [Krauthgamer, Nguyen, and Zondiner ICALP'12]
- compressed representation:
  - naive representation using $O(\min(n, k^2))$ bits is optimal, even for weighted grids [Gavoille, Peleg, Pérennes, and Raz SODA'01]
  - can compress unweighted undirected planar graphs using $\tilde{O}(\min(k^2, \sqrt{nk}))$ bits [Abboud, Gawrychowski, Mozes, Weimann SODA'18]

Our results

- we turn the compressed representation of Abboud et. al into a distance emulator:

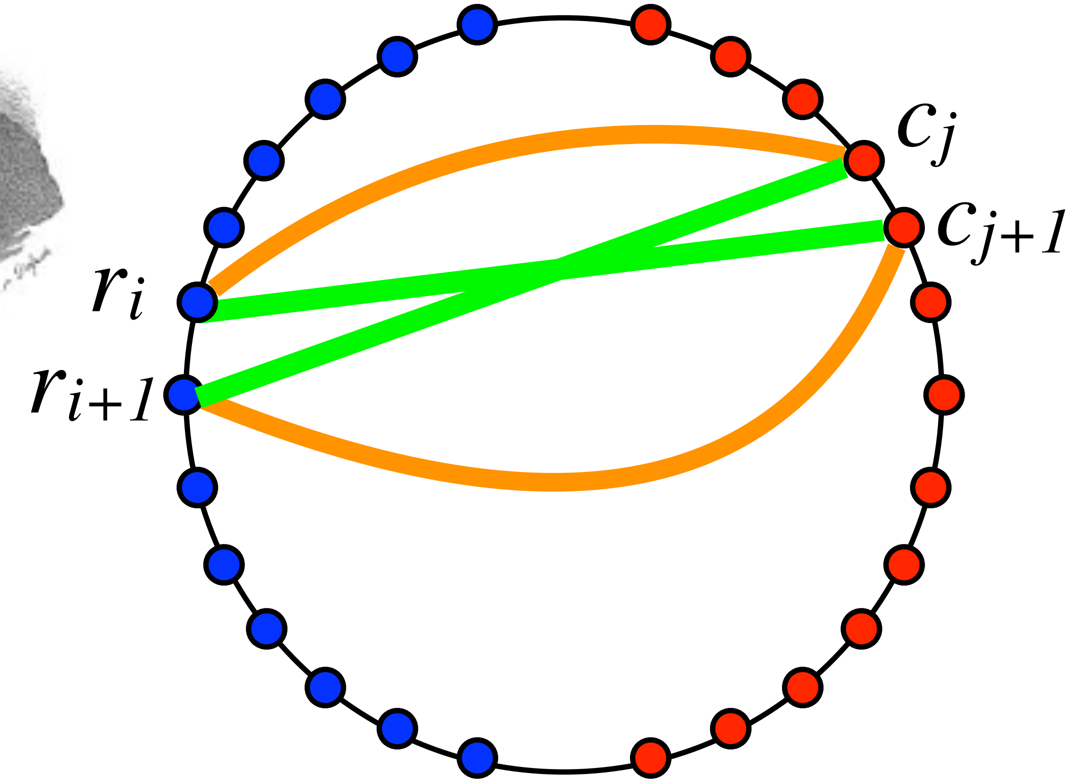
  for an undirected unweighted planar graph G with n vertices and k terminals, we construct a directed weighted (non-planar) emulator with $\tilde{O}(\min(k^2, \sqrt{n \cdot k}))$ vertices and edges in $\tilde{O}(n)$ time.

- as a corollary, one can compute all-pairs distances among the terminals in $\tilde{O}(n)$ time when $k = O(n^{1/3})$
  (just run Dijkstra on the emulator k times)

Converting Abboud et al's compression into an emulator

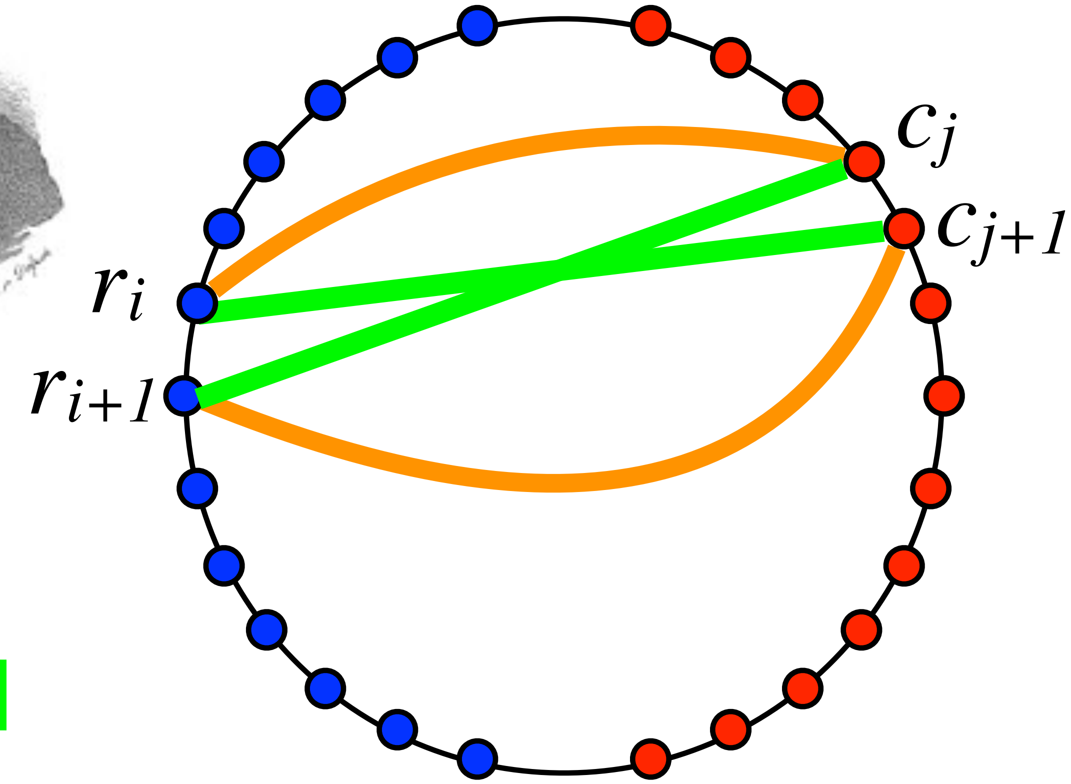
- the main building block in the compression scheme of Abboud et al. is a compression of m-by-m unit-Monge matrices into $\tilde{O}(m)$ bits
- our main technical tool emulates an m-by-m unit-Monge matrix by a graph with $\tilde{O}(m)$ vertices and edges

Monge matrices



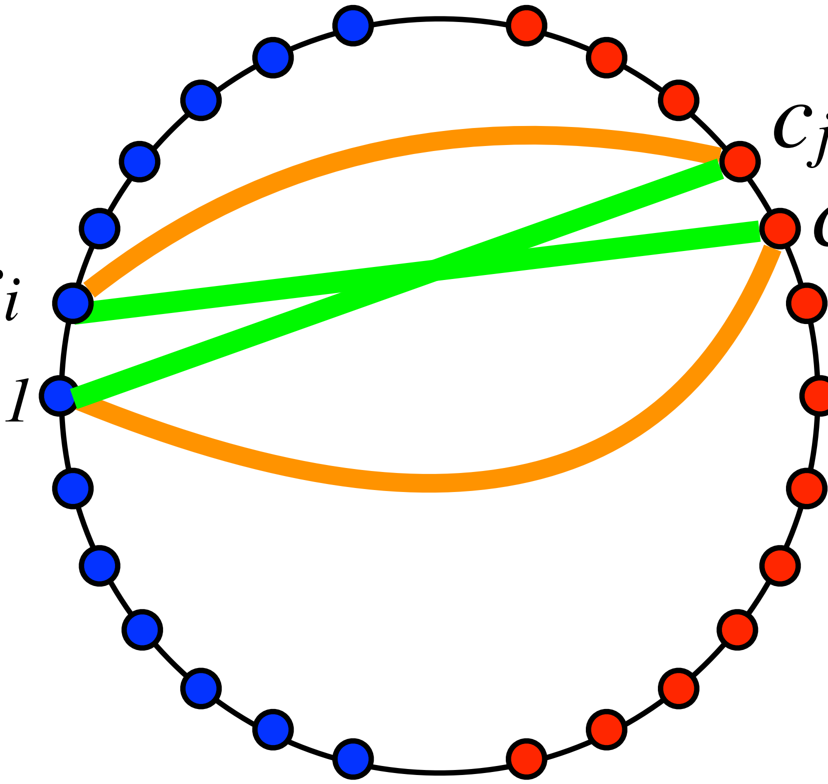
- $M[i,j]$ - distance from r_i to c_j

Monge matrices



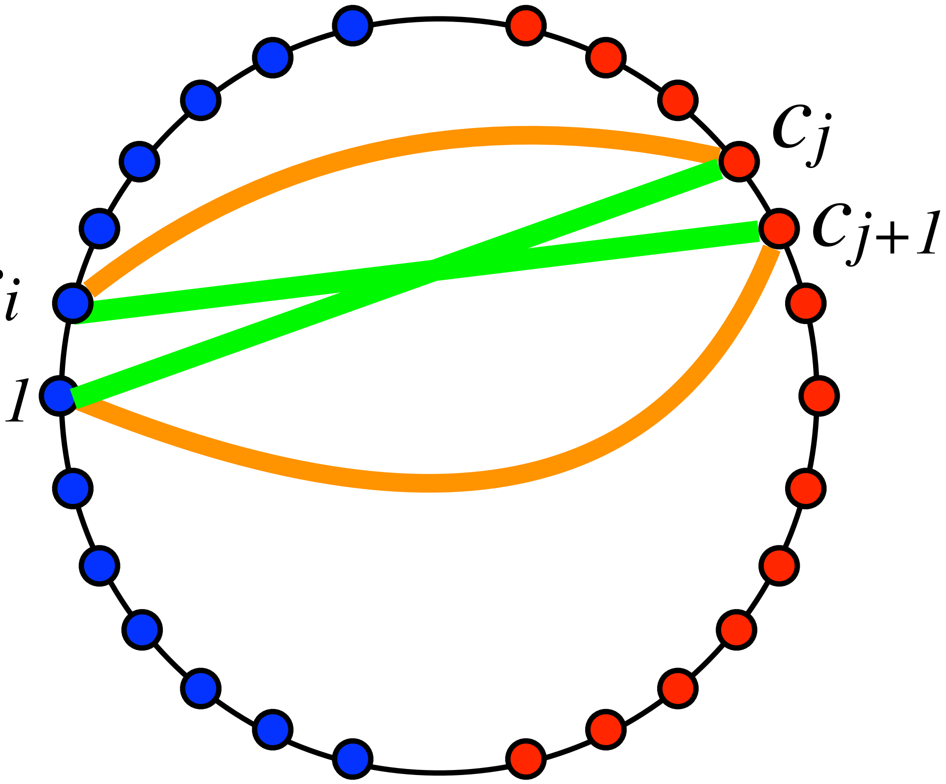
- $M[i,j]$ - distance from r_i to c_j
- Monge: paths must cross, so
  $M[i,j]+M[i+1,j+1] \leq M[i,j+1]+M[i+1,j]$

Monge matrices



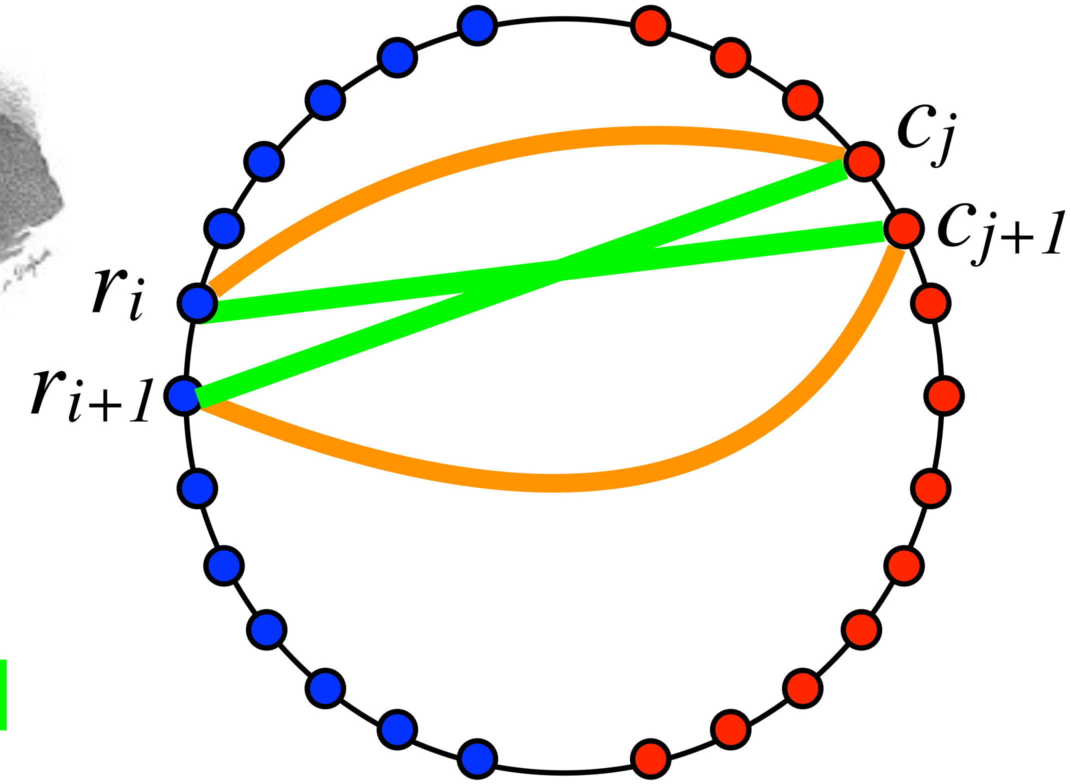
- $M[i,j]$ - distance from r_i to c_j
- Monge: paths must cross, so
  $\mathbf{M[i,j]+M[i+1,j+1]} \leq \mathbf{M[i,j+1]+M[i+1,j]}$
- so difference between consecutive rows is monotone:
  $M[i,j]-M[i+1,j] \leq M[i,j+1]-M[i+1,j+1]$

Monge matrices



- $M[i,j]$ - distance from r_i to c_j
- Monge: paths must cross, so
  $M[i,j]+M[i+1,j+1] \leq M[i,j+1]+M[i+1,j]$
- so difference between consecutive rows is monotone:
  $M[i,j]-M[i+1,j] \leq M[i,j+1]-M[i+1,j+1]$
- unit Monge: r_i and r_{i+1} are neighbors, so
  $-1 \leq M[i,j] - M[i+1,j] \leq 1$

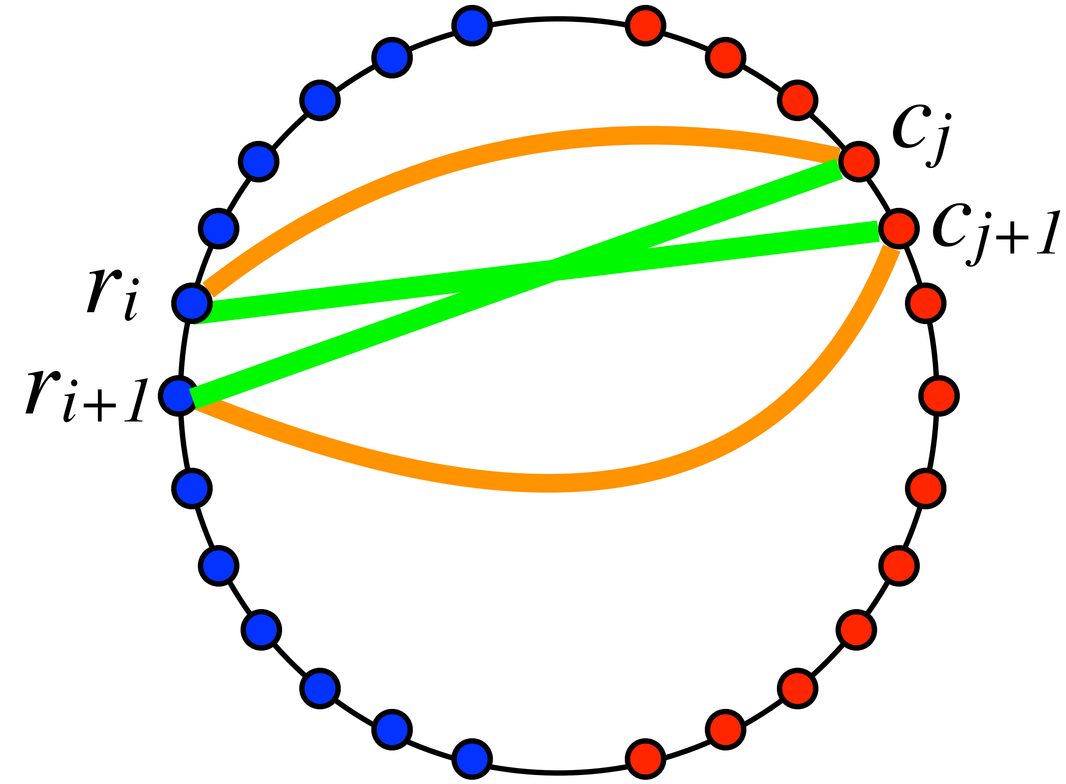
Monge matrices



- $M[i,j]$ - distance from r_i to c_j
- Monge: paths must cross, so
  $M[i,j]+M[i+1,j+1] \leq M[i,j+1]+M[i+1,j]$
- so difference between consecutive rows is monotone:
  $M[i,j]-M[i+1,j] \leq M[i,j+1]-M[i+1,j+1]$
- unit Monge: r_i and r_{i+1} are neighbors, so
  $-1 \leq M[i,j] - M[i+1,j] \leq 1$
- so difference between consecutive rows is monotone and bounded. Looks like:

-1 -1 -1 -1 0 0 0 0 0 0 0 0 1 1 1 1 1

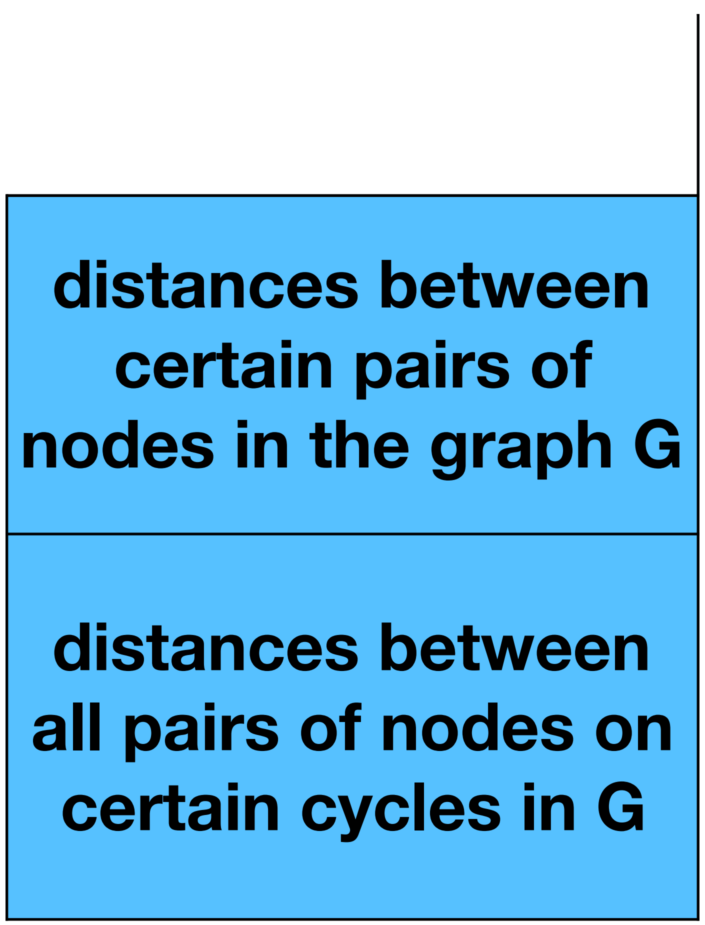
Unit-Monge matrix compression



- store the first row of an x-by-y matrix explicitly
- encode difference between each pair of consecutive rows by storing the two locations where -1 changes to 0 or 0 changes to 1
- total space for x-by-y matrix is $\tilde{O}(x+y)$ instead of $\tilde{O}(xy)$

Converting Abboud et al's compression into an emulator

- the compression scheme of Abboud et al. combines unit-Monge matrix compression with a slicing technique and small cycle separators.



Converting Abboud et al's compression into an emulator

- the compression scheme of Abboud et al. combines unit-Monge matrix compression with a slicing technique and small cycle separators.

	metric compression of Abboud et al.
distances between certain pairs of nodes in the graph G	represented explicitly
distances between all pairs of nodes on certain cycles in G	represented using **unit-Monge matrix compression** **(not a graph)**

Converting Abboud et al's compression into an emulator

- the compression scheme of Abboud et al. combines unit-Monge matrix compression with a slicing technique and small cycle separators.

	metric compression of Abboud et al.	**our subset emulator**
distances between certain pairs of nodes in the graph G	represented explicitly	represented as weighted edges
distances between all pairs of nodes on certain cycles in G	represented using **unit-Monge matrix compression** **(not a graph)**	represented using **emulators for unit-Monge matrices** **(a graph)**

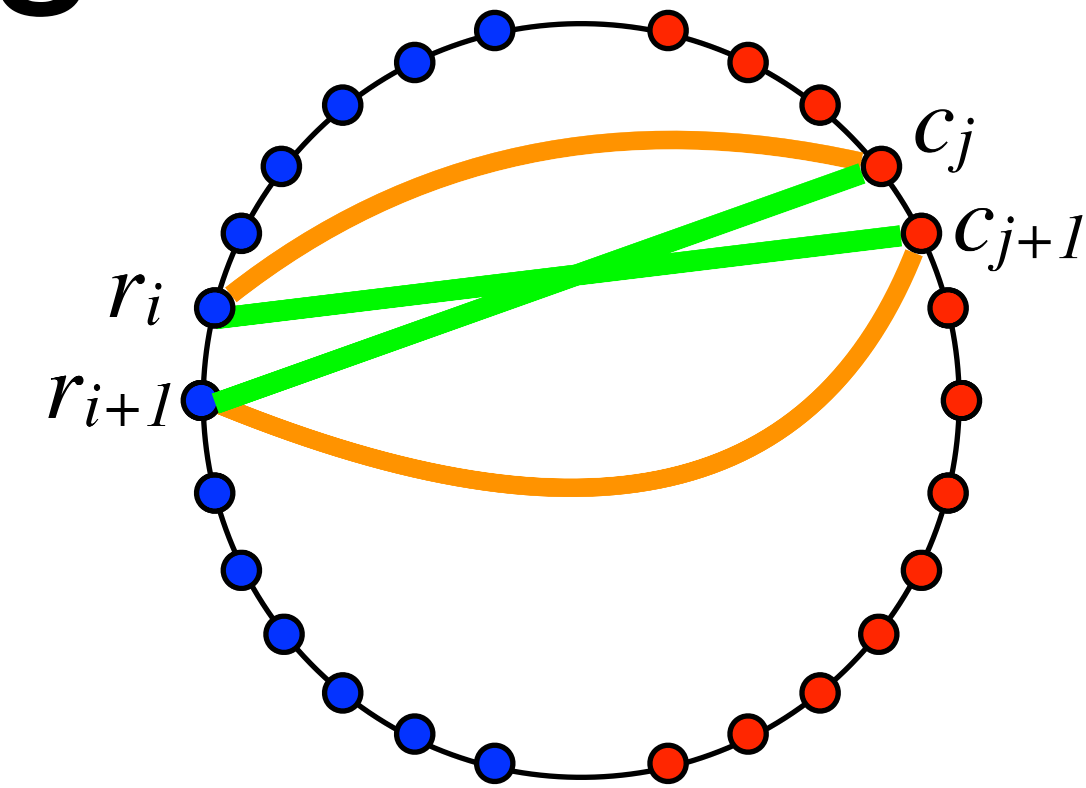
Converting Abboud et al's compression into an emulator

- the compression scheme of Abboud et al. combines unit-Monge matrix compression with a slicing technique and small cycle separators.

	metric compression of Abboud et al.	**our subset emulator**
distances between certain pairs of nodes in the graph G	represented explicitly	represented as weighted edges
distances between all pairs of nodes on certain cycles in G	represented using **unit-Monge matrix compression** **(not a graph)**	represented using **emulators for unit-Monge matrices** **(a graph)**

Emulating unit-Monge matrices

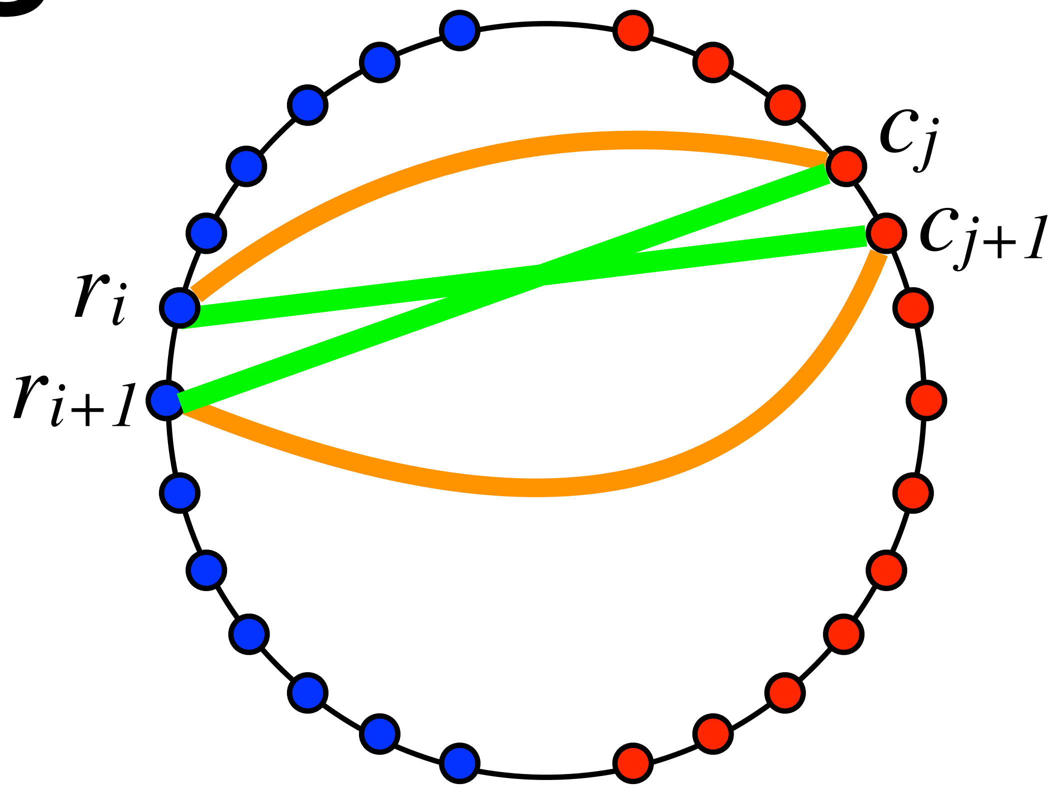
- we would like to construct a small graph H with vertices $\{r_i\}$, $\{c_j\}$ and possibly new vertices, such that $dist_H(r_i,c_j) = dist_G(r_i,c_j)$



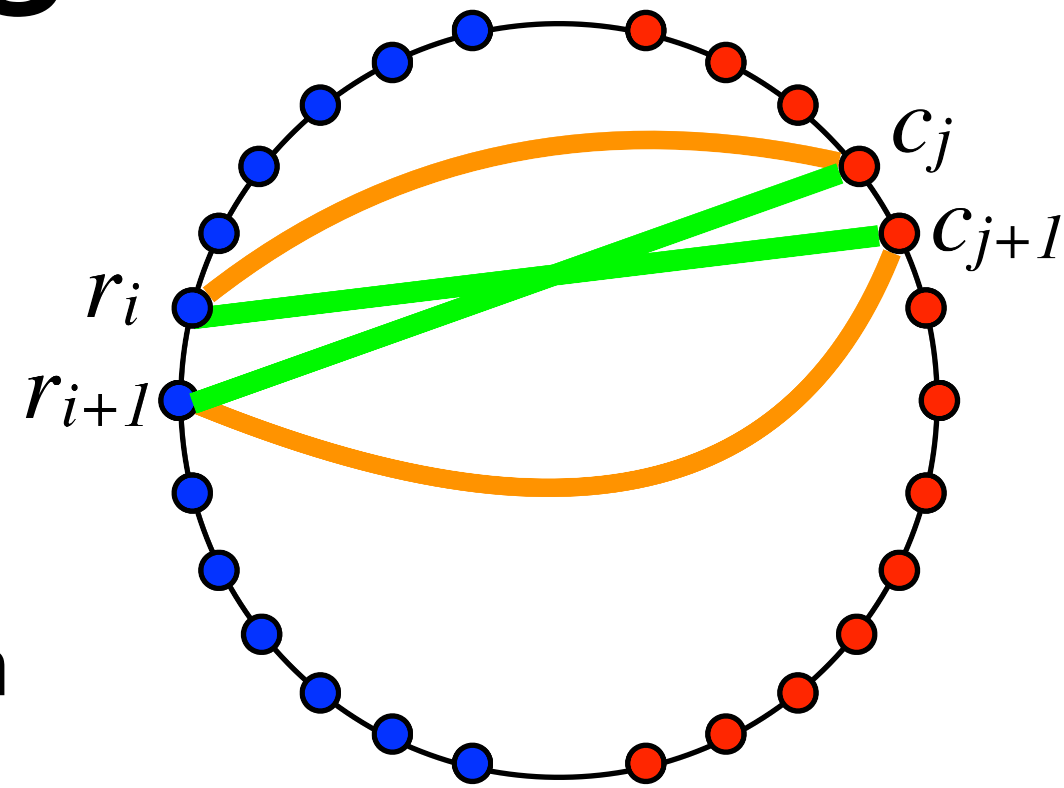
Emulating unit-Monge matrices

- we would like to construct a small graph H with vertices $\{r_i\}$, $\{c_j\}$ and possibly new vertices, such that $dist_H(r_i,c_j) = dist_G(r_i,c_j)$

- equivalently, the distance from r_i to c_j in H is $M[i,j]$



Emulating unit-Monge matrices



- we would like to construct a small graph H with vertices $\{r_i\}$, $\{c_j\}$ and possibly new vertices, such that $dist_H(r_i,c_j) = dist_G(r_i,c_j)$
- equivalently, the distance from r_i to c_j in H is $M[i,j]$
- we first convert the problem into yet another equivalent one

  (this is not essential, but the presentation is cleaner)

From unit-Monge to right-stochastic

- If M is unit Monge then each row in matrix of difference between consecutive rows looks like:

-1 -1 -1 -1 0 0 0 0 0 0 0 0 1 1 1 1 1

From unit-Monge to right-stochastic

- If M is unit Monge then each row in matrix of difference between consecutive rows looks like:

  -1 -1 -1 -1 0 0 0 0 0 0 0 0 1 1 1 1 1

- if we also take differences between consecutive columns we get a 0/1 matrix P with at most two 1's in each row

From unit-Monge to right-stochastic

- If M is unit Monge then each row in matrix of difference between consecutive rows looks like:

  -1 -1 -1 -1 0 0 0 0 0 0 0 0 1 1 1 1 1

- if we also take differences between consecutive columns we get a 0/1 matrix P with at most two 1's in each row
- for simplicity think of P as a right-stochastic matrix (at most a single 1 entry in each row)

From unit-Monge to right-stochastic

- If M is unit Monge then each row in matrix of difference between consecutive rows looks like:

  -1 -1 -1 -1 0 0 0 0 0 0 0 0 1 1 1 1 1

- if we also take differences between consecutive columns we get a 0/1 matrix P with at most two 1's in each row
- for simplicity think of P as a right-stochastic matrix (at most a single 1 entry in each row)
- can express M as:

$$M[i,j] = R[i] + C[j] + \boxed{\sum_{i' \geq i, j' \geq j} P[i',j']}$$

Dominance query

$$\begin{array}{cc} & \begin{matrix} & & & & j & & & \end{matrix} \\ \begin{matrix} \\ \\ \\ i \\ \\ \\ \\ \\ \end{matrix} & \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

P

Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.

Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.

divide

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

divide

Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.

divide

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

divide

Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.

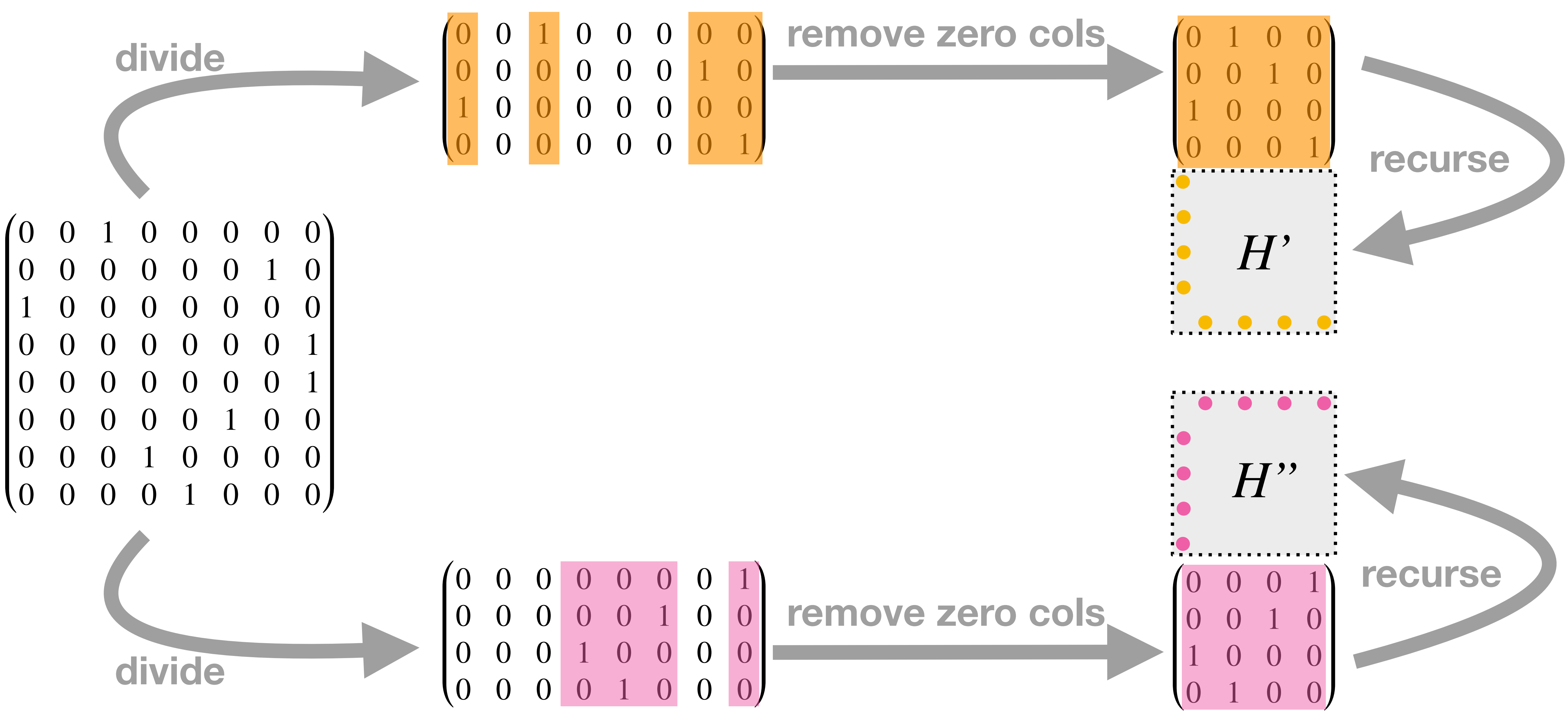
$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\xrightarrow{\text{divide}} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{\text{remove zero cols}} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\xrightarrow{\text{divide}} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\text{remove zero cols}} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

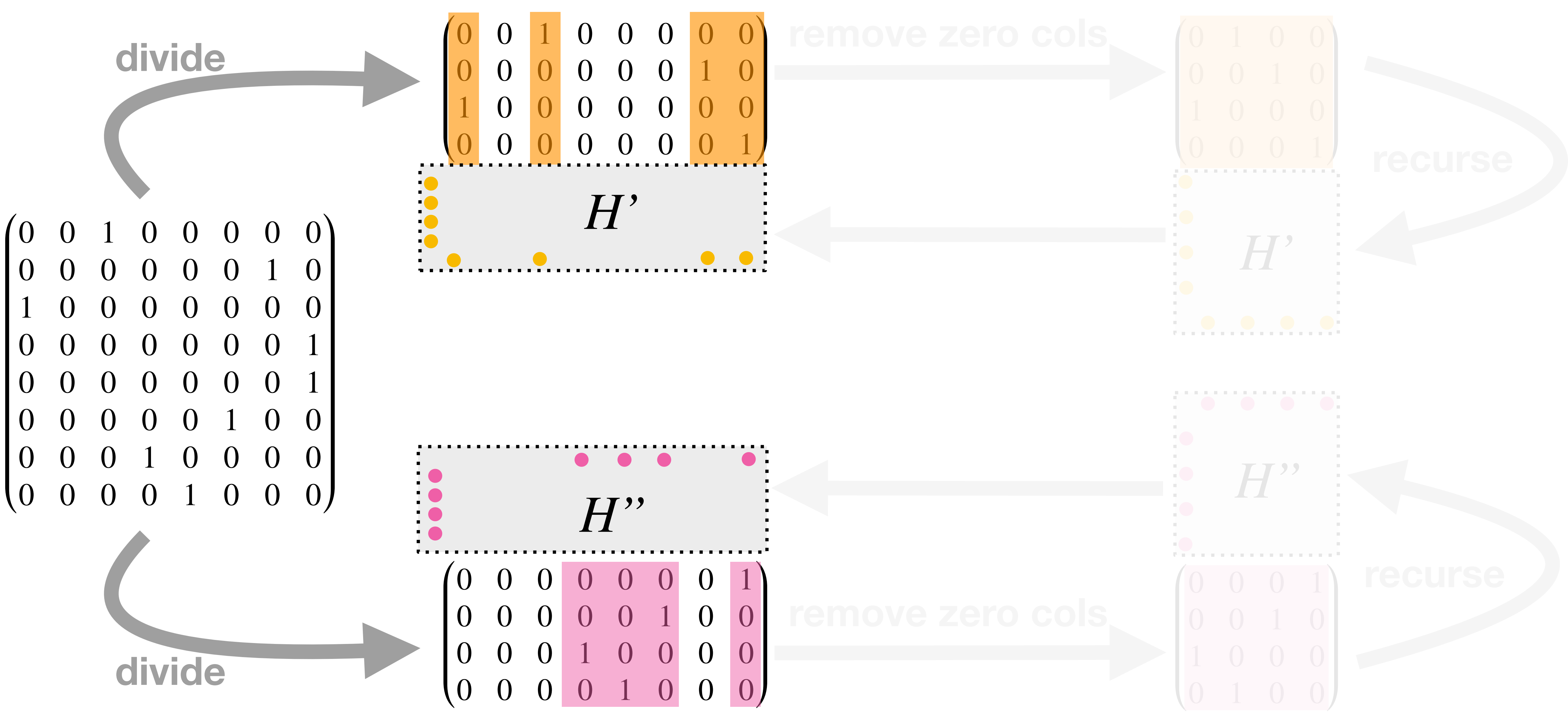
Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.



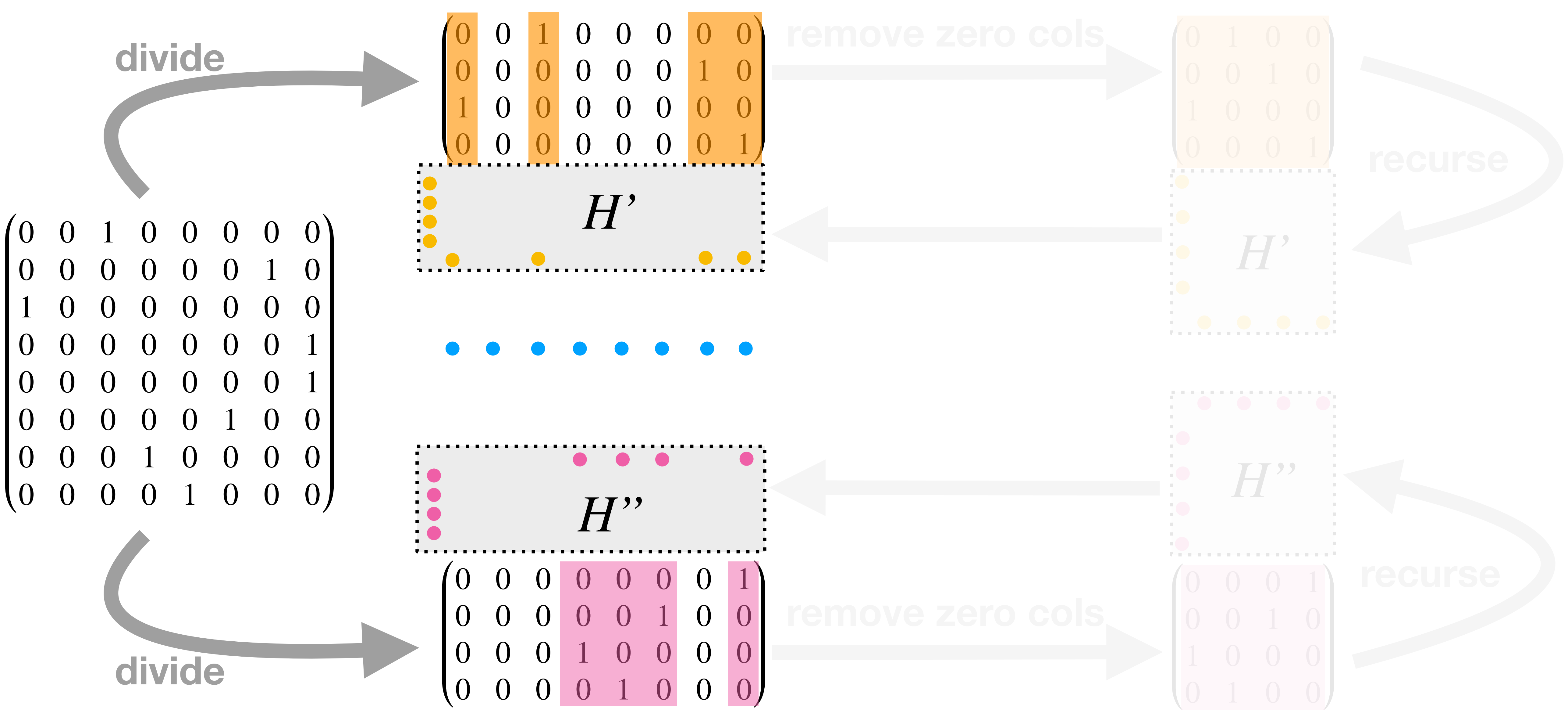
Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.



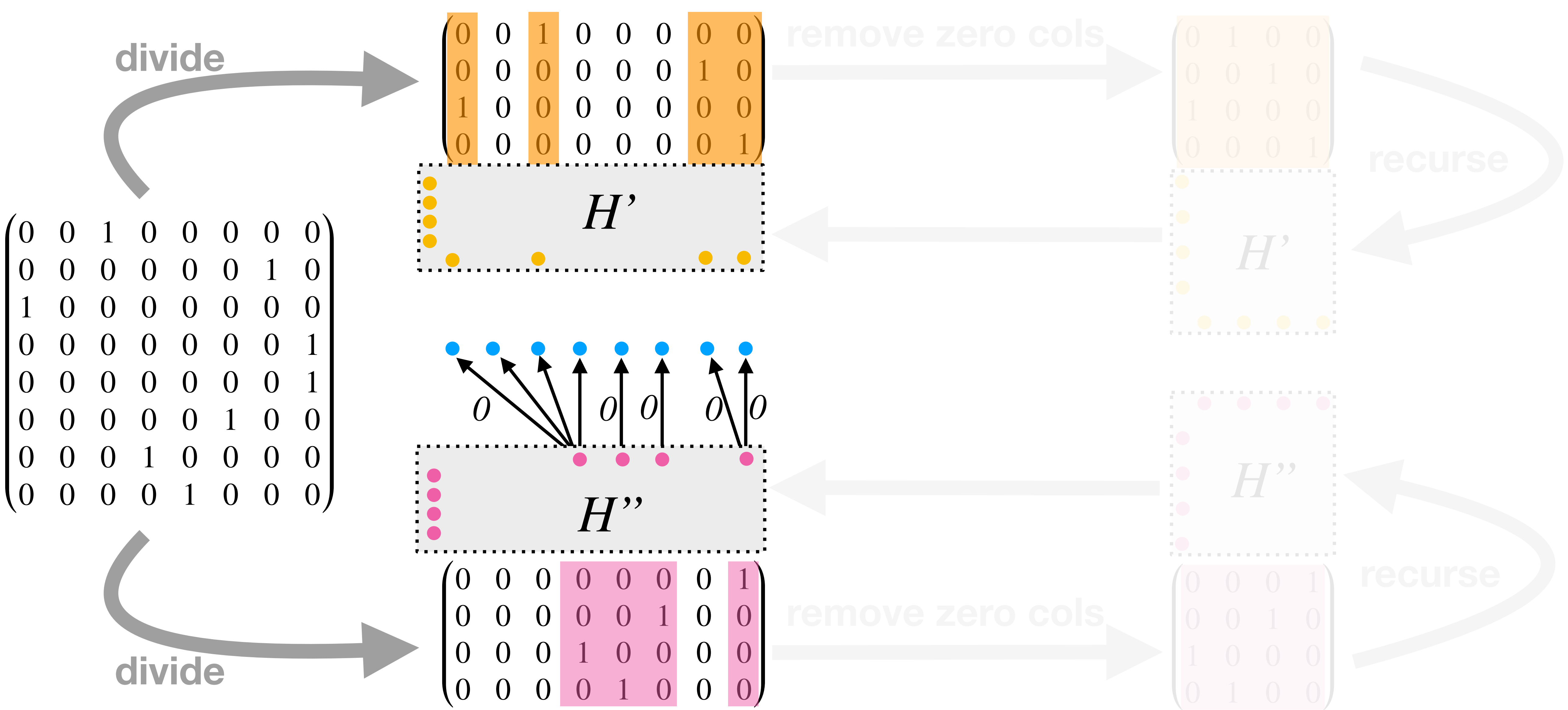
Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.



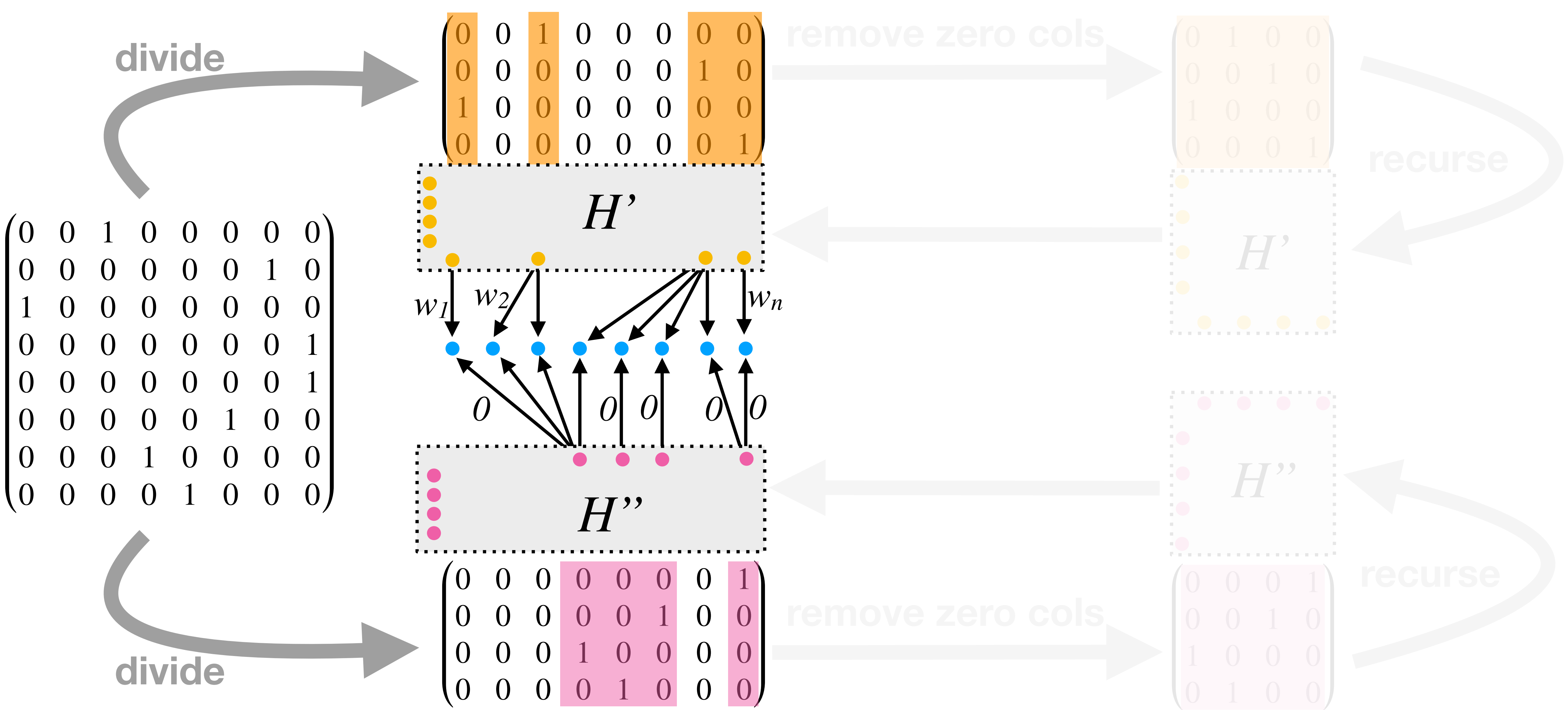
Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.



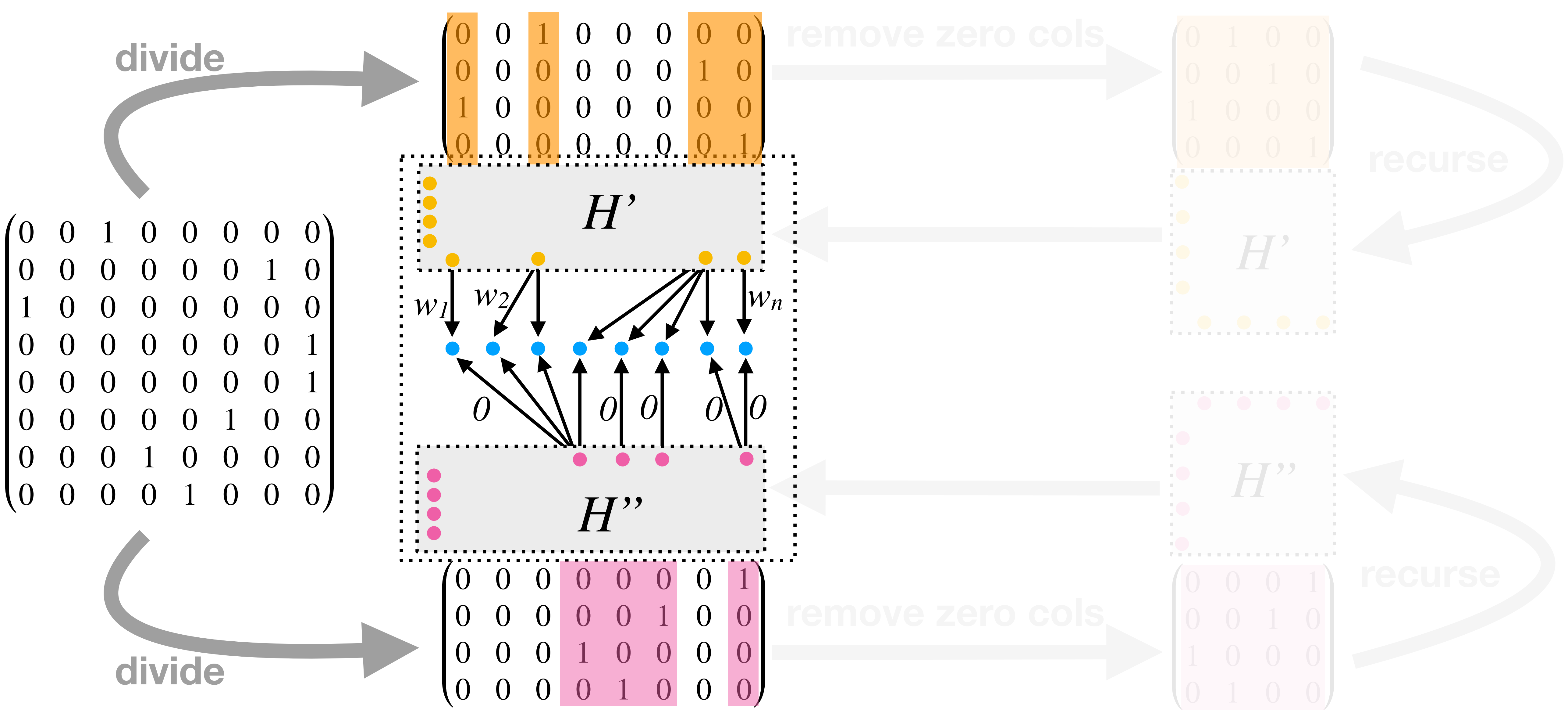
Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.



Graphical encoding of dominance queries on a right-stochastic matrix

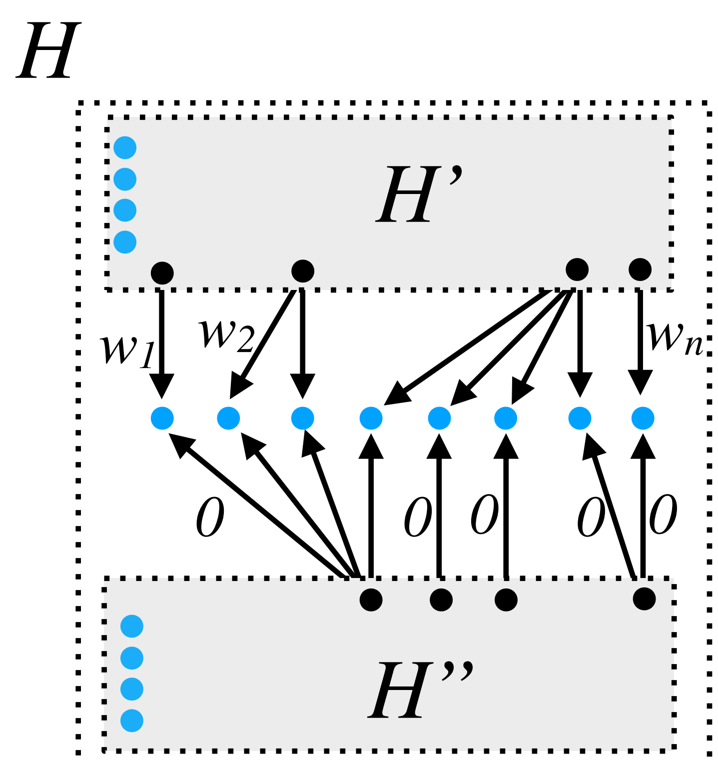
given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.



Graphical encoding of dominance queries on a right-stochastic matrix

given a n-by-n right-stochastic matrix M construct a small graph H (size $\tilde{O}(n)$) with a vertex r_i for each row, and a vertex c_j for each column, such that $dist_H(r_i,c_j)$ is the (i,j)-dominance query in M.

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$



the resulting emulator H is a directed acyclic (non-planar) graph with non-negative edge weights with $O(n \log n)$ vertices and edges

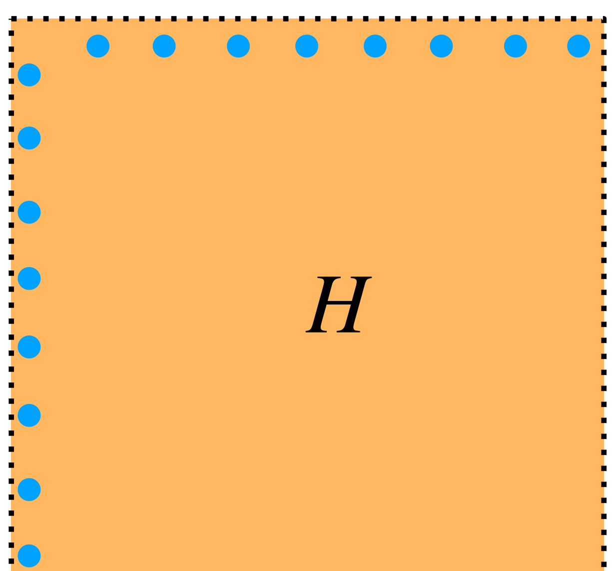
Back from right-stochastic to unit-Monge

- can express unit-Monge M as:

$$M[i,j] = R[i] + C[j] + \underbrace{\boxed{\sum_{i' \geq i, j' \geq j} P[i',j']}}_{\text{Dominance query}}$$

$$\begin{array}{cc} & \begin{matrix} & & & & & j & & & \end{matrix} \\ \begin{matrix} \\ \\ \\ \\ i \\ \\ \\ \\ \\ \end{matrix} & \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

P



Back from right-stochastic to unit-Monge

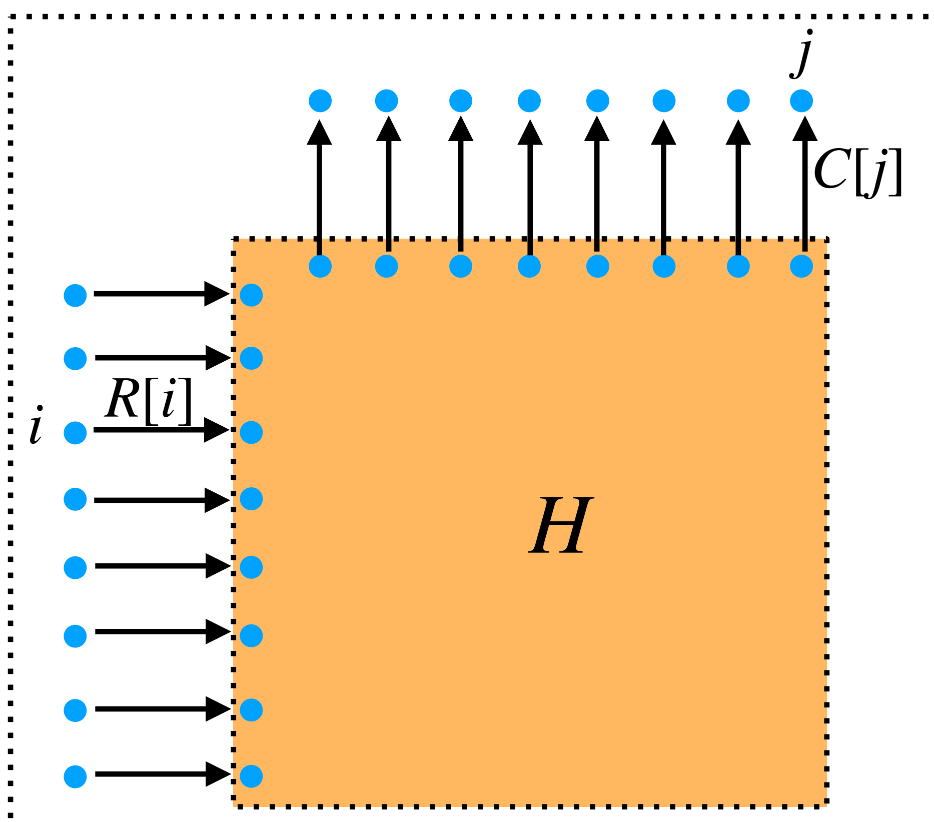
- can express unit-Monge M as:

$$M[i,j] = R[i] + C[j] + \boxed{\sum_{i' \geq i, j' \geq j} P[i', j']}$$

Dominance query

$$\begin{array}{c|ccccccccc}
 & & & & & & j & & & \\
 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
i & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{array}$$

P



Converting Abboud et al's compression into an emulator

- the compression scheme of Abboud et al. combines unit-Monge matrix compression with a slicing technique and small cycle separators.

	metric compression of Abboud et al.	**our subset emulator**
distances between certain pairs of nodes in the graph G	represented explicitly	represented as weighted edges
distances between all pairs of nodes on certain cycles in G	represented using **unit-Monge matrix compression**	represented using **emulators for unit-Monge matrices** ✓

Our results

- we turn the compressed representation of Abboud et. al into a distance emulator:

  for an undirected unweighted planar graph G with n vertices and k terminals, we construct a directed weighted (non-planar) emulator with $\tilde{O}(\min(k^2, \sqrt{n \cdot k}))$ vertices and edges in $\tilde{O}(n)$ time.

- as a corollary, one can compute all-pairs distances among the terminals in $\tilde{O}(n)$ time when $k = O(n^{1/3})$
  (just run Dijkstra on the emulator k times)

Things I swept under the rug

- dealing with triangular unit-Monge matrices
- construction time of emulator
- construction time (and slight improvement) of the compression scheme of Abboud et al.

Open Questions

- does a small planar emulator exist?
- lower bounds?
  preliminary result: $O(n \log n)$ is tight for emulating by DAGs
- other applications

We are looking for postdocs and PhD interns

- jointly hosted by Oren Weimann (Haifa U.) and Shay Mozes (IDC Herzliya)
- planar graphs, data structures, string algorithms
- contact me in person or at smozes@idc.ac.il , oren@cs.haifa.ac.il

אוניברסיטת חיפה
University of Haifa
جامعة حيفا

IDC
HERZLIYA