Near-Optimal Compression for the Planar Graph Metric

Amir Abboud, Paweł Gawrychowski, Shay Mozes, Oren Weimann

How compressible is the shortest path metric of planar graphs?

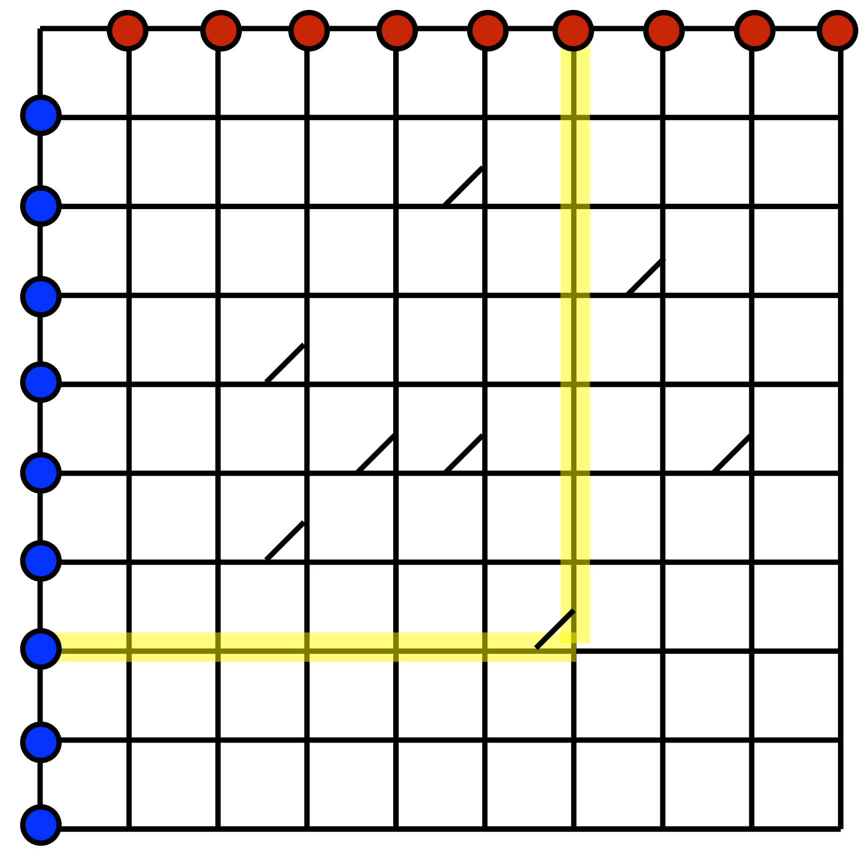
- input: a planar graph G with n nodes and subset of k terminals.
- output: a compressed encoding of the exact distances between all pairs of terminals.
- naive (ignoring logs):
  - $O(n)$ bits - explicitly store G
  - $O(k^2)$ bits - explicitly store all pairwise distances

Related problems

- sparsification - subgraph/minor preserving distances
  - Krauthgamer, Nguyen, and Zondiner [ICALP’12]: naive solution is optimal even for unweighted grids
- labeling schemes - deduce distance between u,v given just the labels of u and of v
  - Gavoille, Peleg, Prennes, and Raz [SODA’01]: naive solution is optimal even for weighted grids
  - their lower bound is not tight for unweighted grids

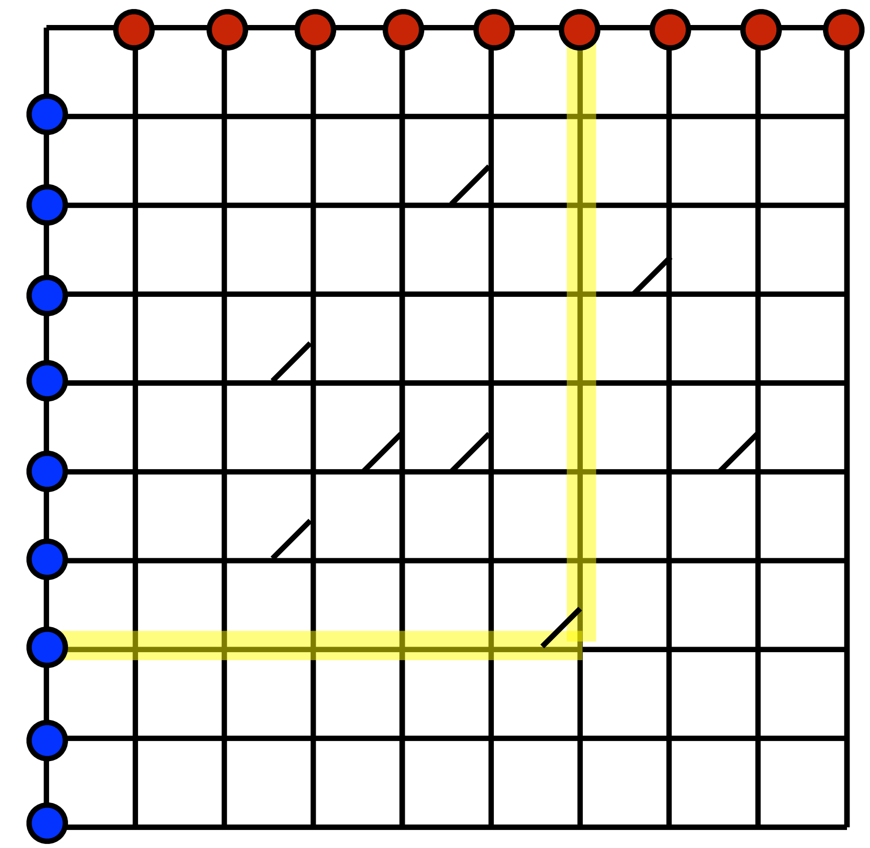
$O(\min(n,k^2))$ is optimal for weighted graphs

Gavoille, Peleg, Prennes, and Raz [SODA'01]

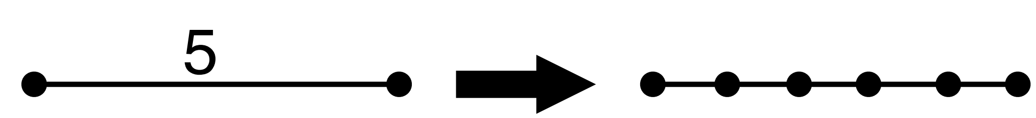


- k-by-k grid (so $k=n^{1/2}$)
- can choose edge weights in [1,k] so that shortest path go first right then up
- can encode any k-by-k boolean matrix by adding shortcuts
- so must use $\Omega(k^2) = \Omega(n)$ bits for this grid
- for general planar $\Omega(\min(n,k^2))$ **weighted!**

What about unweighted graphs?



- k-by-k grid
- subdivide edges to emulate weights
- n increases to $\Theta(k^3)$
- so must use $\Omega(k^2) = \Omega(n^{2/3})$ bits for this grid
- using multiple smaller grids easy to show that $\Omega(\min((nk)^{1/2}, k^2))$ bits required for general unweighted planar graphs



State of affairs

- in the weighted case: naive $min(n,k^2)$ is optimal

- in the unweighted case:

  upper bound $\min(n,k^2)$

  lower bound $\min((nk)^{1/2}, k^2)$

- can it be that weights account for this gap?

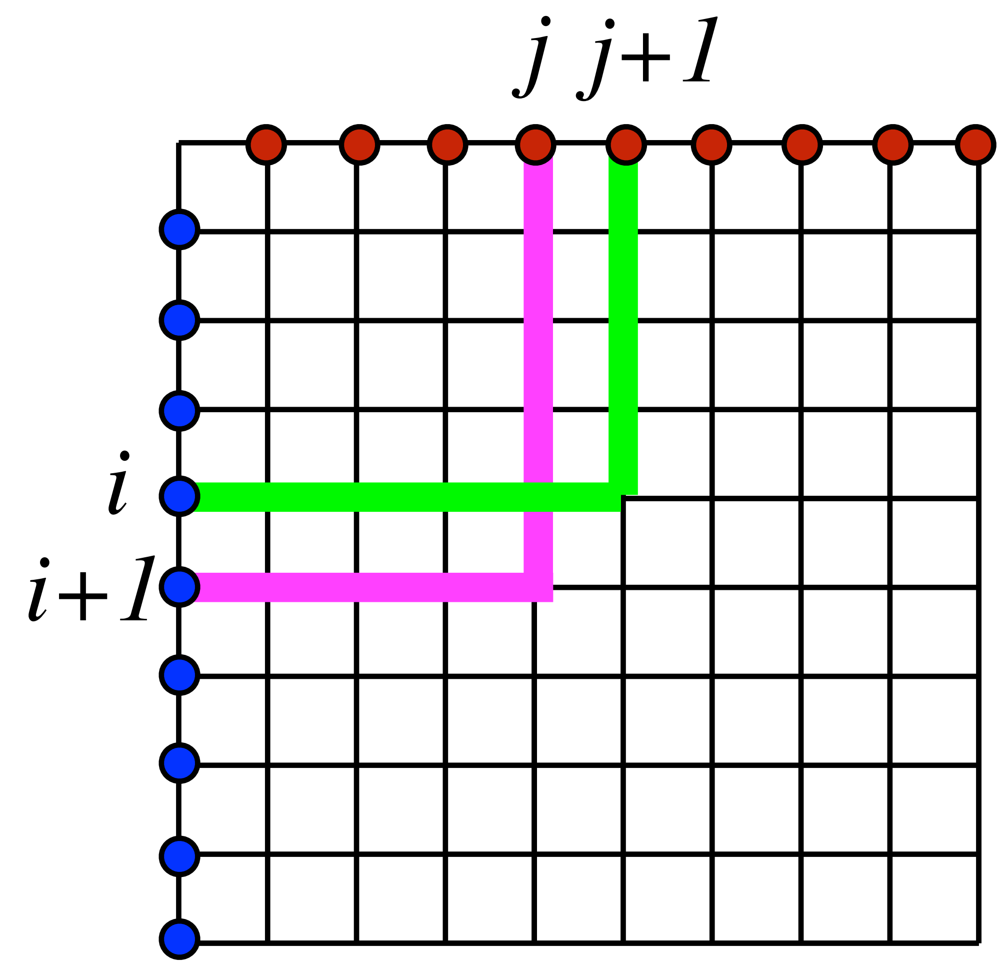
- is the lower bound too naive? (terminals on same face, graph is grid, subdivision is naive)

Our results

- the shortest path metric of undirected unweighted planar graphs can be encoded using $\tilde{O}((nk)^{1/2})$ bits
  - this is optimal up to polylogarithmic factors
  - shows that weights make the problem significantly harder

Monge and unit-Monge

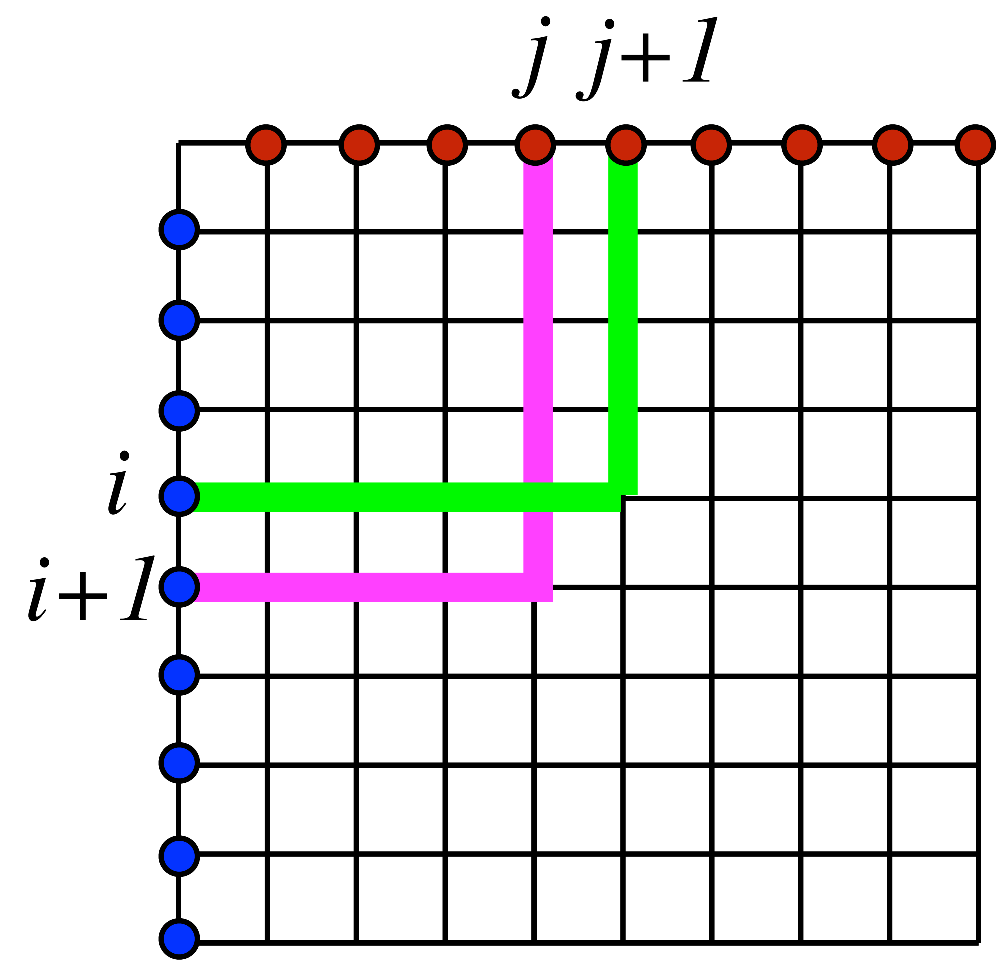
- $M[i,j]$ - distance from i'th blue to j'th red



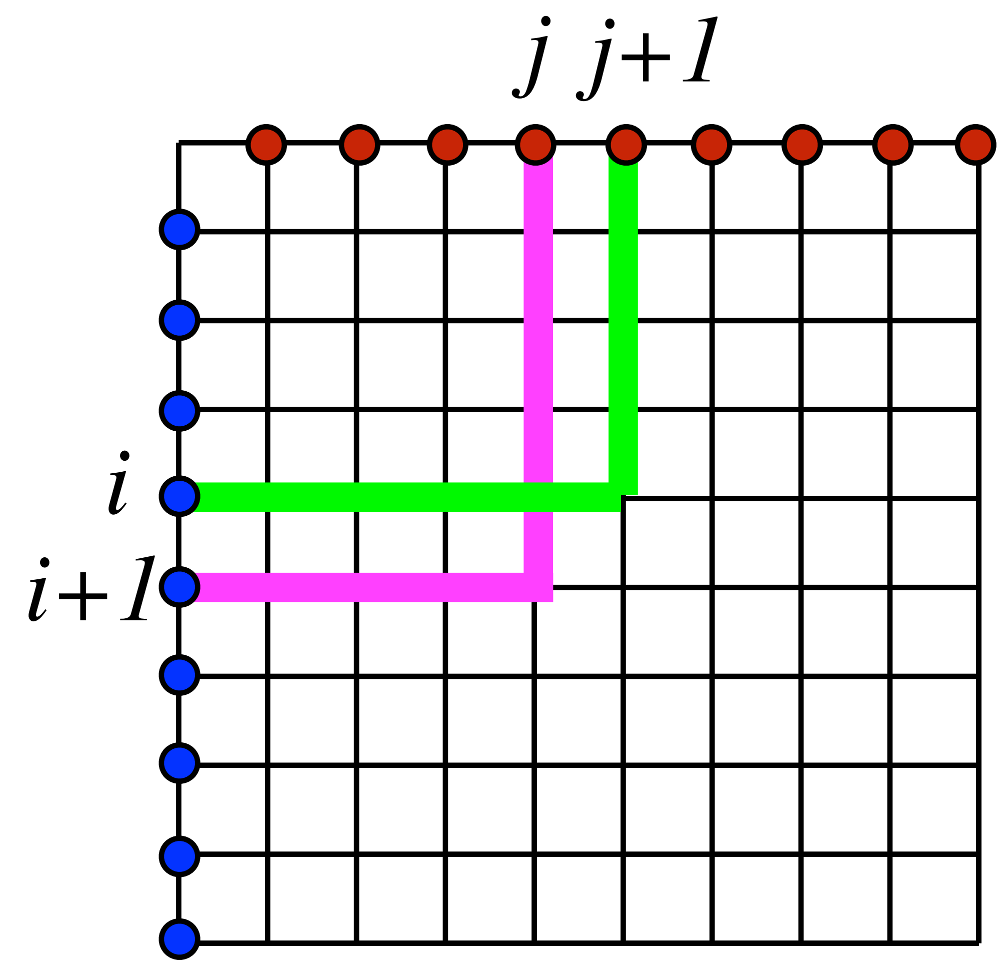
Monge and unit-Monge

- $M[i,j]$ - distance from i'th blue to j'th red
- Monge: paths must cross, so

  $M[i, j]+M[i+1, j+1] \leq M[i,j+1]+M[i+1, j]$

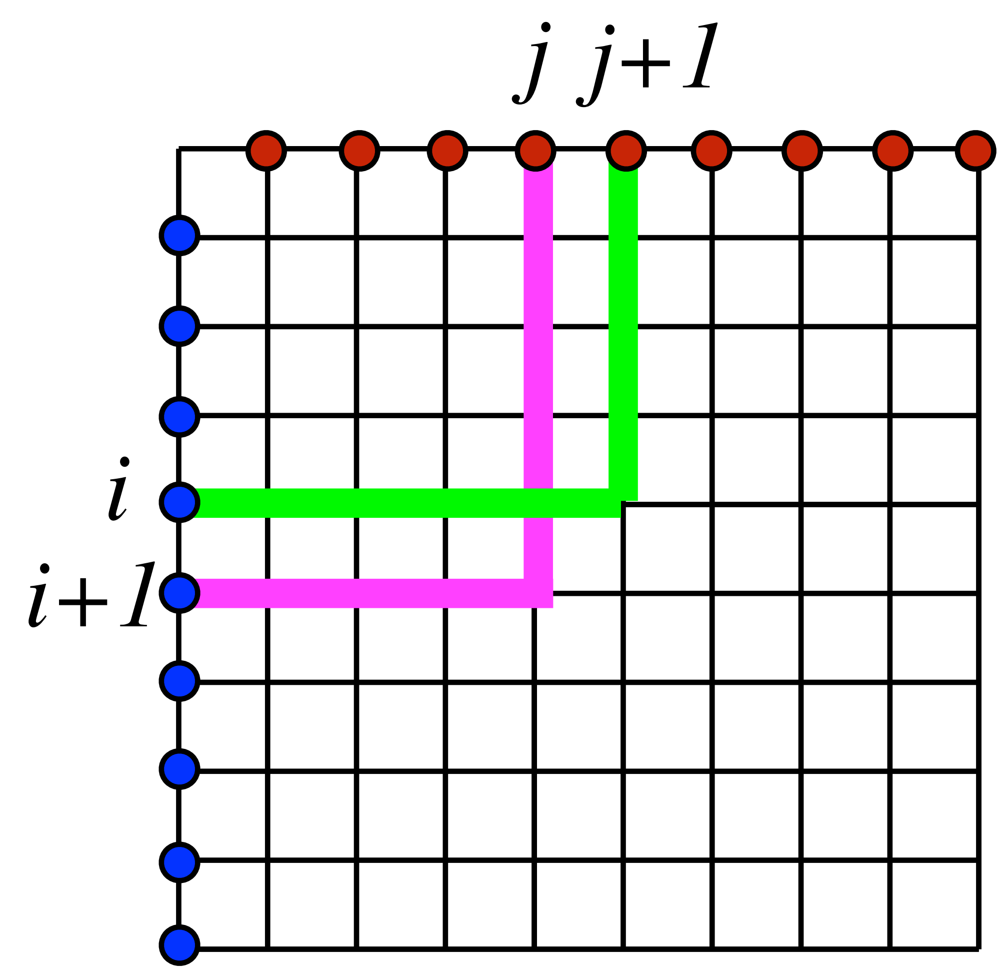


Monge and unit-Monge



- $M[i,j]$ - distance from i'th blue to j'th red
- Monge: paths must cross, so
  $M[i, j]+M[i+1, j+1] \leq M[i,j+1]+M[i+1, j]$
- unit Monge: i and $i+1$ are neighbors, so
  $-1 \leq M[i, j] - M[i+1, j] \leq 1$

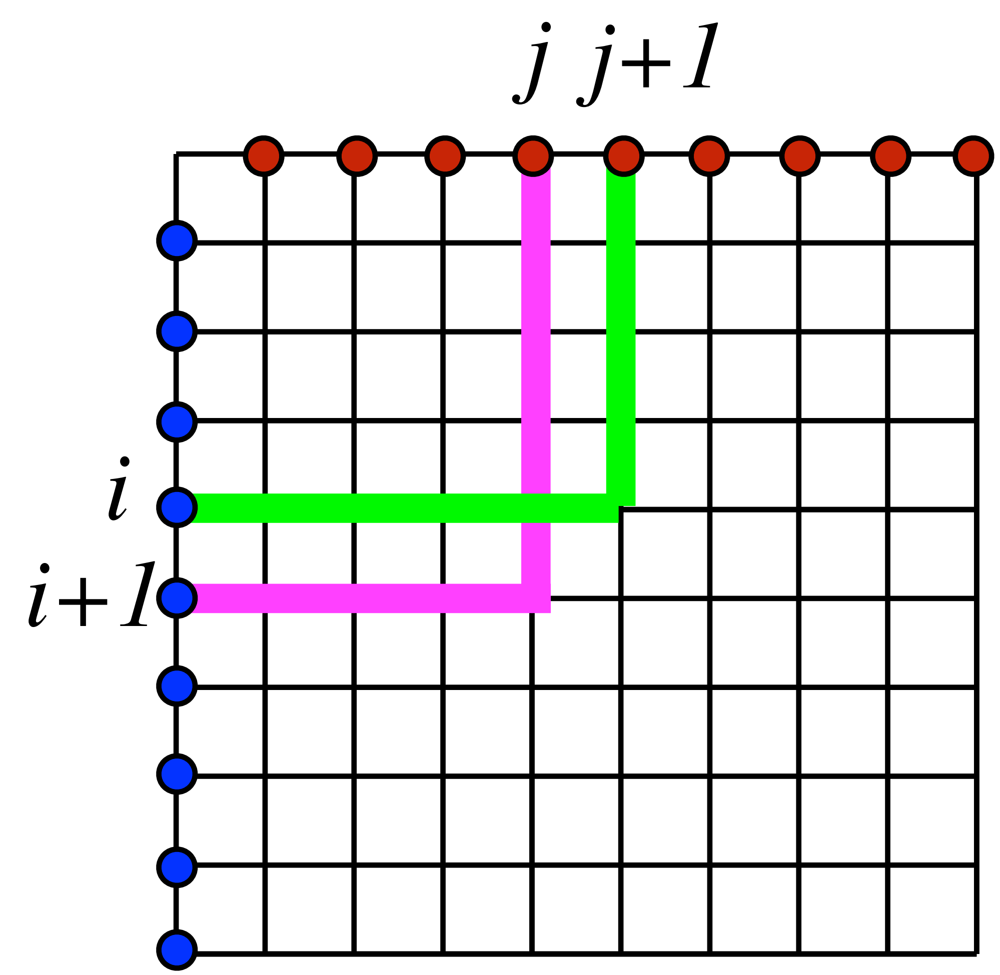
Monge and unit-Monge



- $M[i,j]$ - distance from i'th blue to j'th red
- Monge: paths must cross, so
  $$M[i,j]+M[i+1,j+1] \leq M[i,j+1]+M[i+1,j]$$
- unit Monge: i and $i+1$ are neighbors, so
  $$-1 \leq M[i,j] - M[i+1,j] \leq 1$$
- so difference between consecutive rows looks like:

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

Unit-Monge matrices compress well



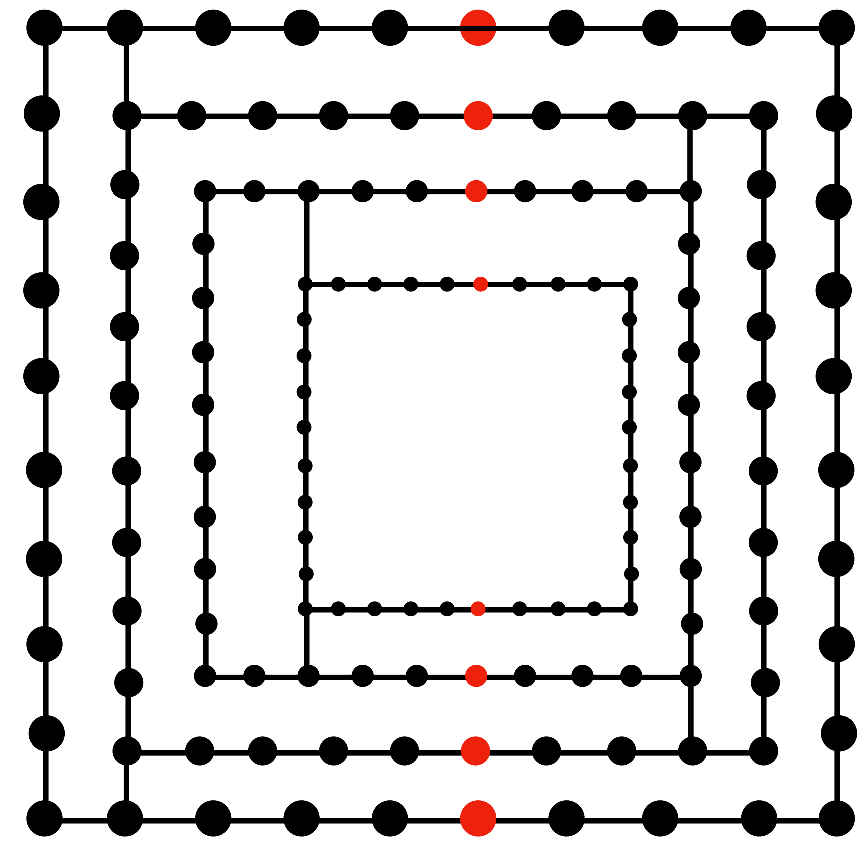
- encode the first row of an x-by-y matrix using $\tilde{O}(y)$ bits
- encode difference between each pair of consecutive rows using $\tilde{O}(1)$ bits by storing the locations where -1 changes to 0 or 0 changes to 1
- total space for x-by-y matrix is $\tilde{O}(x+y)$
- works not just for grids, but as long as red and blue nodes are on a single face and blue nodes are consecutive on the face

Difficulties in applying the unit-Monge property

- usually, the Monge property is used to handle distances among nodes of small cycle separators

- if face size is not bounded there may not exist small cycle separators

- since the graph is unweighted, cannot triangulate

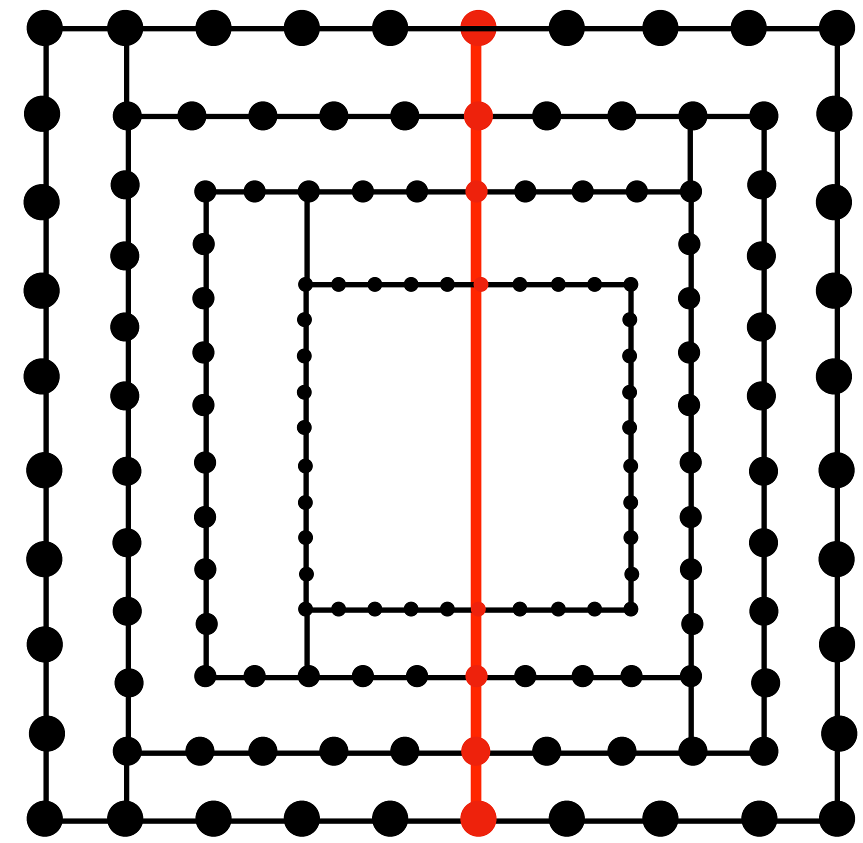


Difficulties in applying the unit-Monge property

- usually, the Monge property is used to handle distances among nodes of small cycle separators

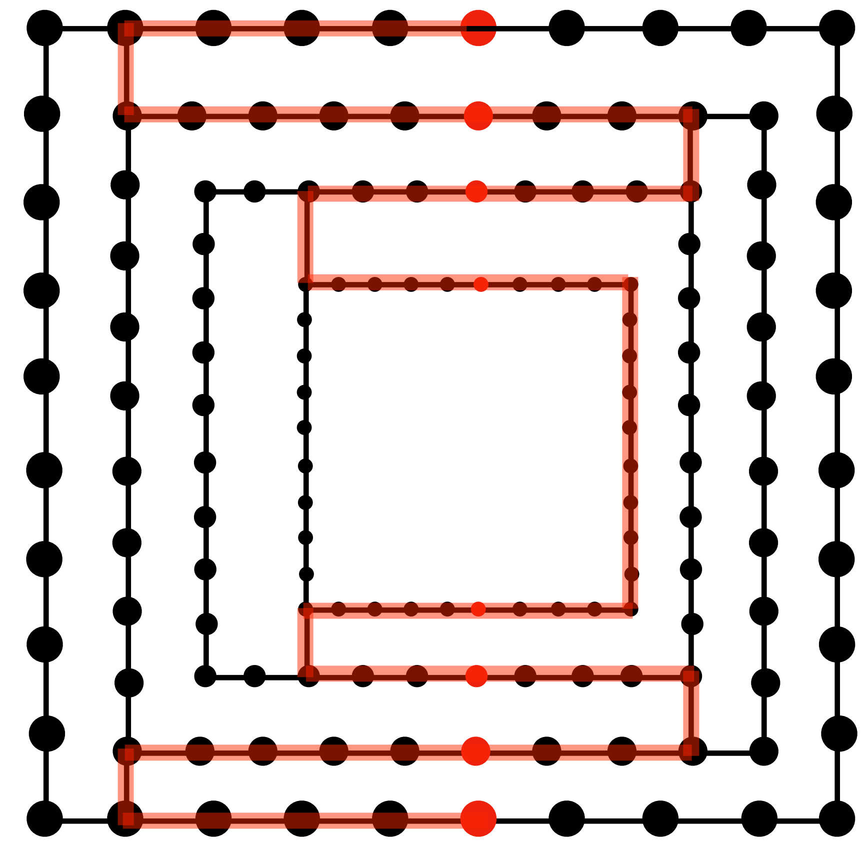
- if face size is not bounded there may not exist small cycle separators

- since the graph is unweighted, cannot triangulate



Difficulties in applying the unit-Monge property

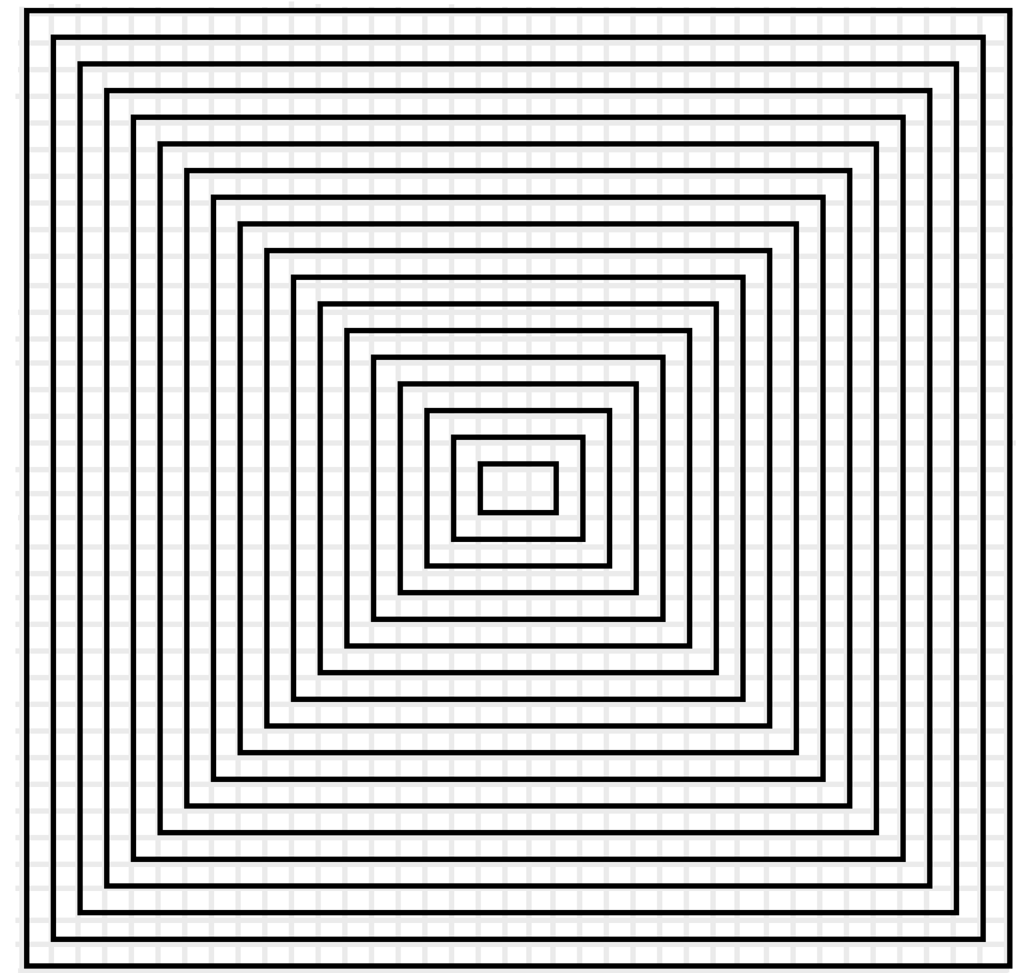
- usually, the Monge property is used to handle distances among nodes of small cycle separators
- if face size is not bounded there may not exist small cycle separators
- since the graph is unweighted, cannot triangulate



Slices

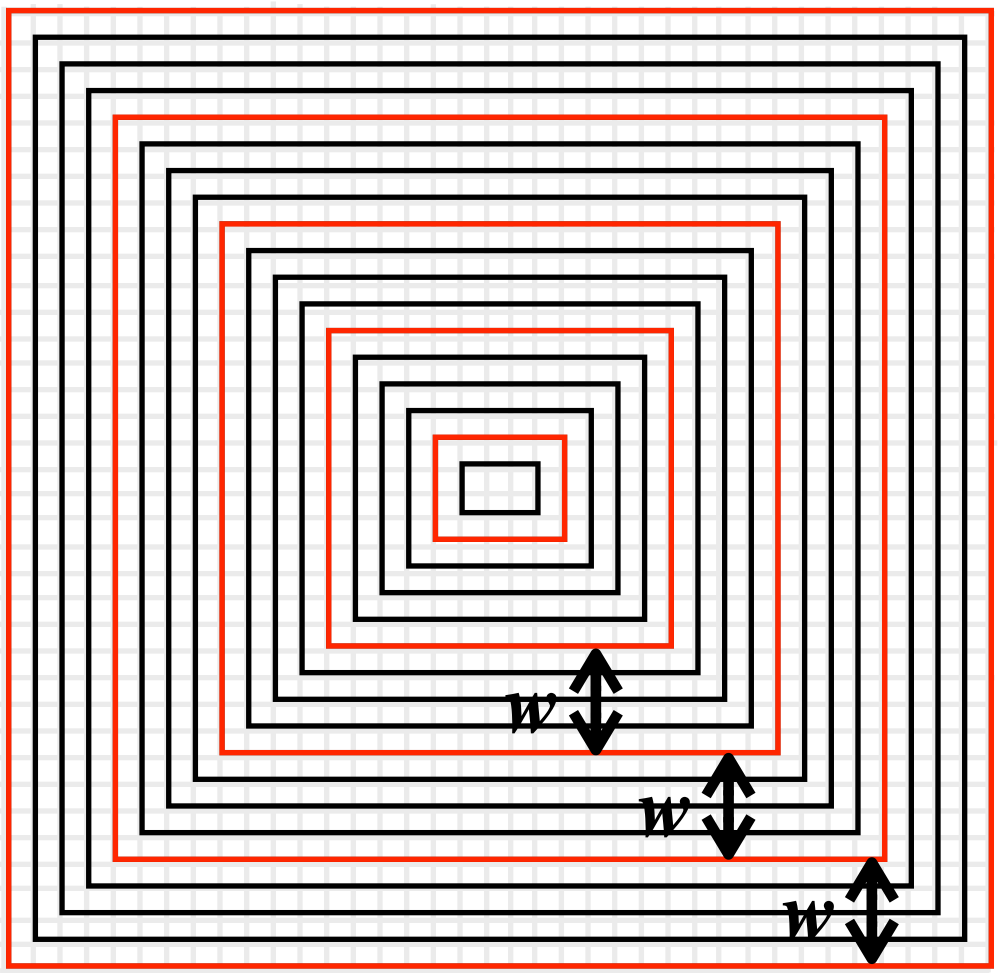
Slices

- can define BFS levels that form nesting cycles.



Slices

- can define BFS levels that form nesting cycles.
- let w be a parameter to be set later
- for some i, the levels congruent to i modulo w consist of at most n/w nodes in total (shifting argument)



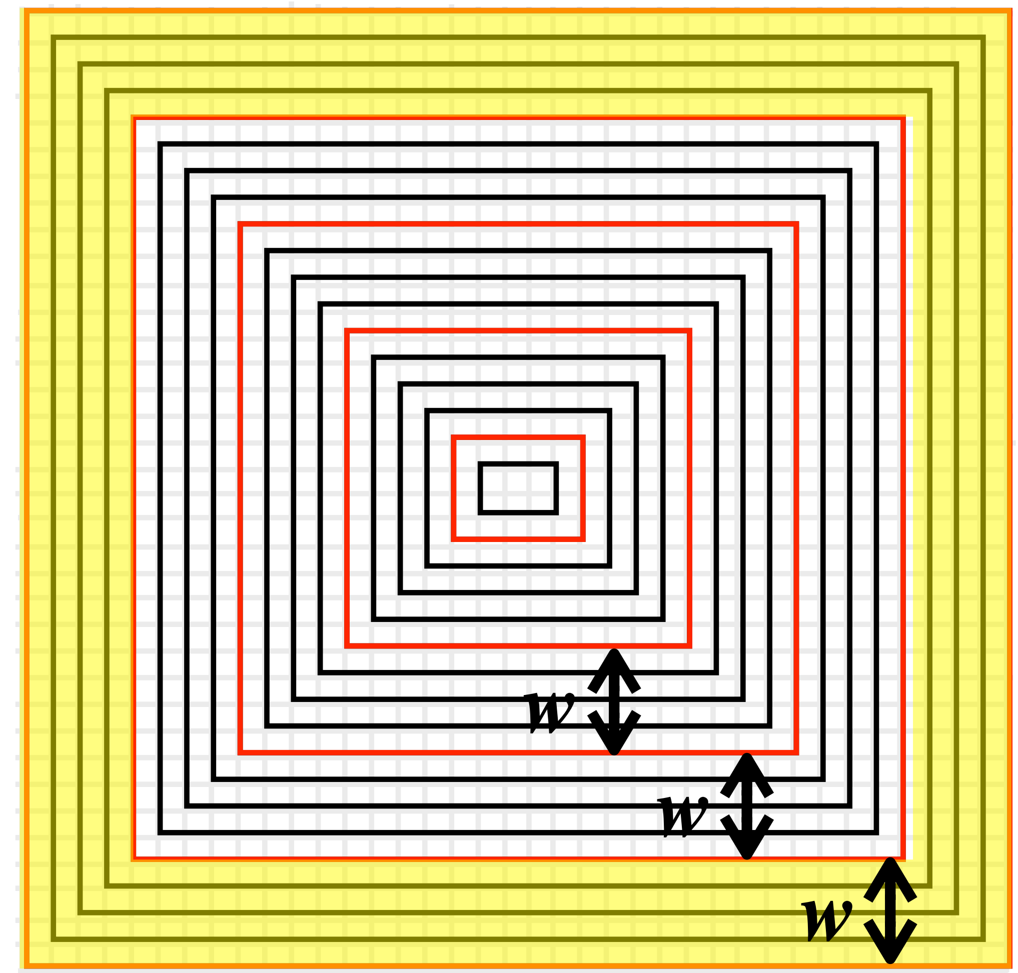
Slices

- can define BFS levels that form nesting cycles.

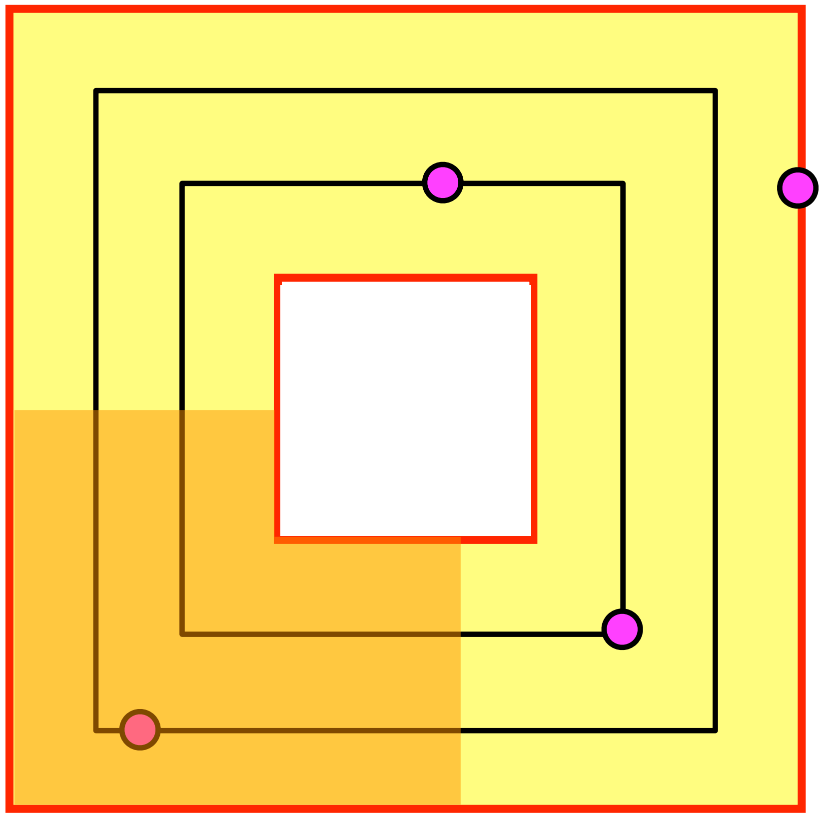
- let w be a parameter to be set later

- for some i, the levels congruent to i modulo w consist of at most n/w nodes in total (shifting argument)

- defines slices in a natural way, each consisting of nodes in w consecutive levels

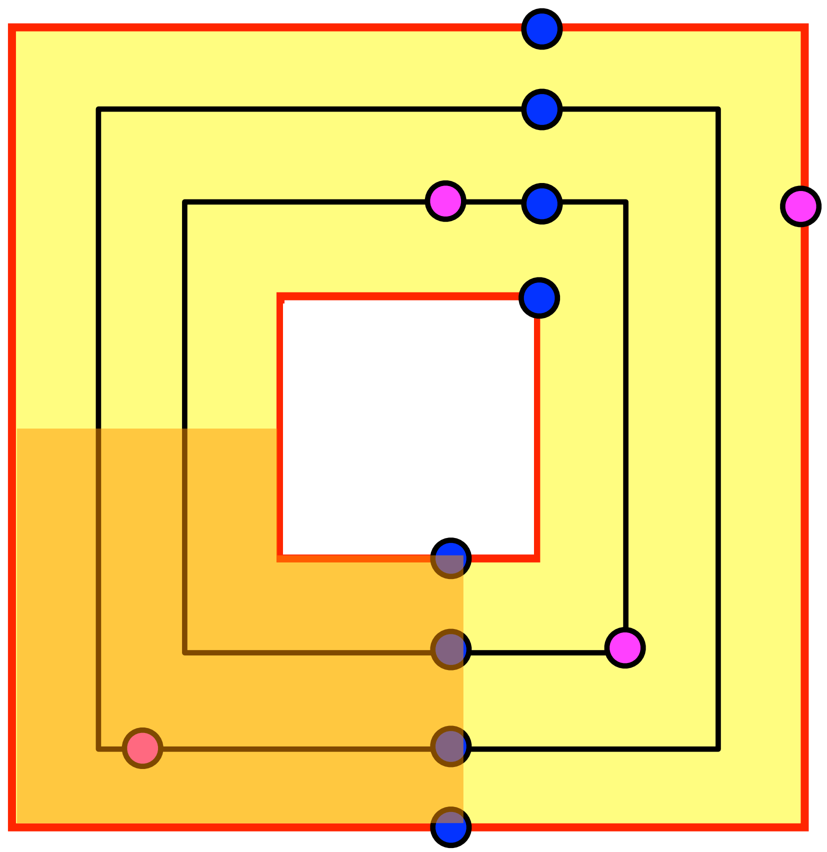


separators in slices



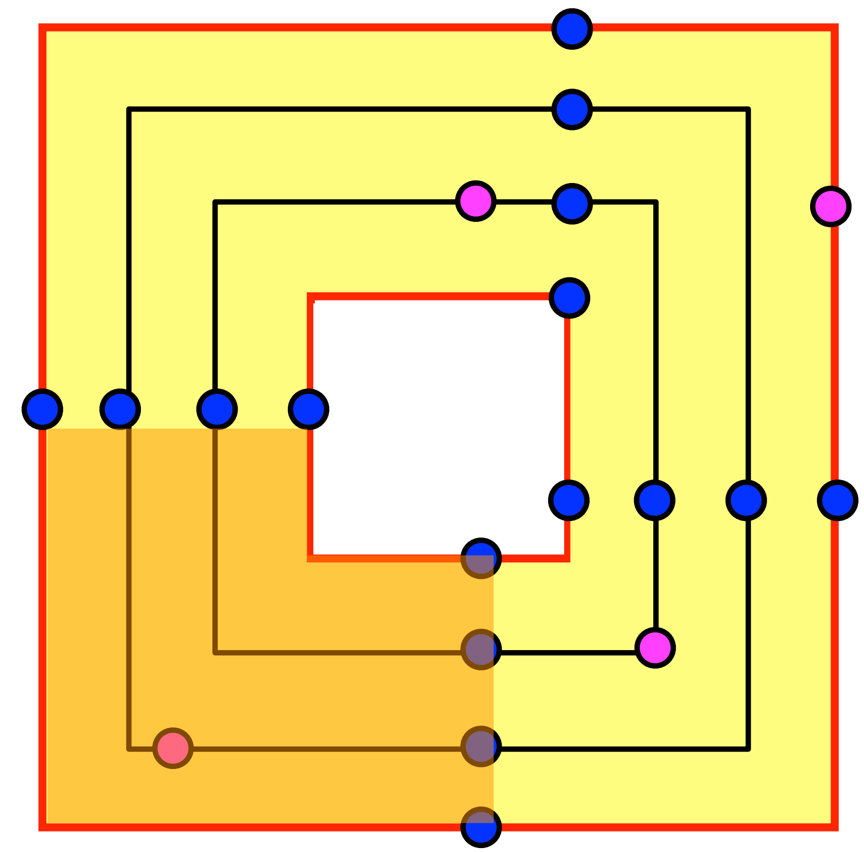
- since a slice consists of w consecutive layers, it has balanced separators:
  - $O(w)$ nodes whose removal partitions the graph into balanced components
  - all on a simple cycle — so Monge
  - but cycle may contain other nodes — so not unit-Monge
- can recursively divide slice using separators into regions, until each region has at most a single terminal. recursion depth is $O(\log k)$

separators in slices



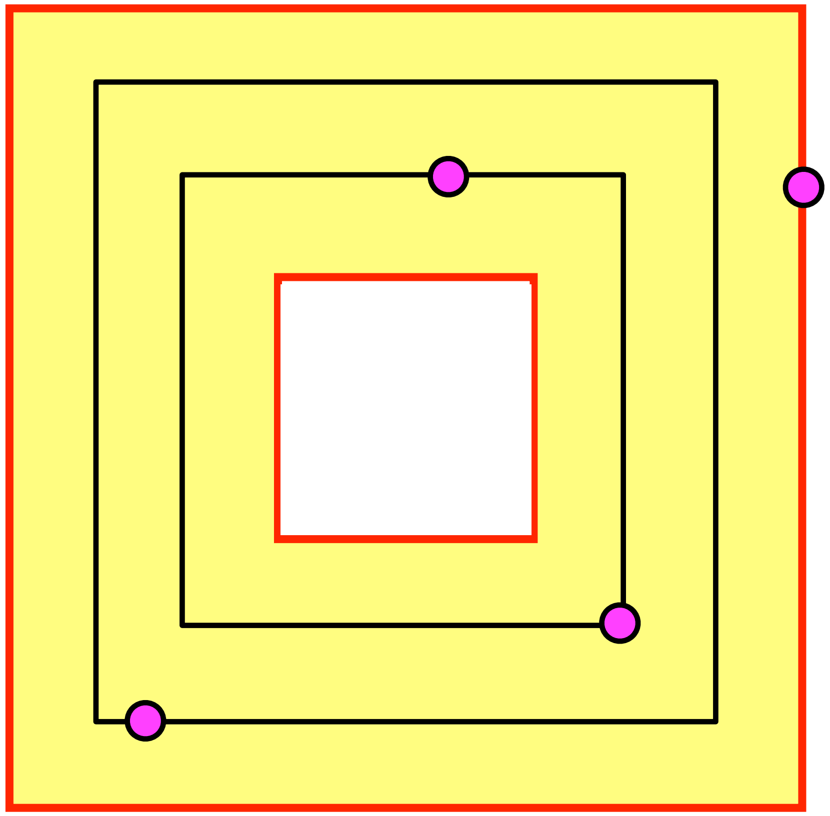
- since a slice consists of w consecutive layers, it has balanced separators:
    - $O(w)$ nodes whose removal partitions the graph into balanced components
    - all on a simple cycle — so Monge
    - but cycle may contain other nodes — so not unit-Monge
- can recursively divide slice using separators into regions, until each region has at most a single terminal. recursion depth is $O(\log k)$

separators in slices



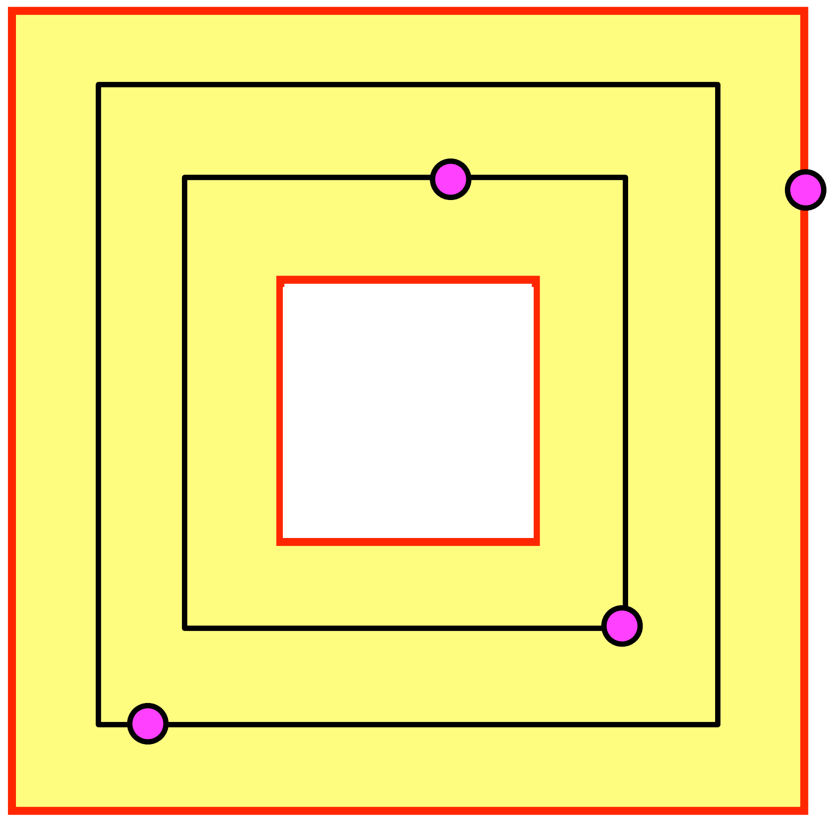
- since a slice consists of w consecutive layers, it has balanced separators:
  - $O(w)$ nodes whose removal partitions the graph into balanced components
  - all on a simple cycle — so Monge
  - but cycle may contain other nodes — so not unit-Monge
- can recursively divide slice using separators into regions, until each region has at most a single terminal. recursion depth is $O(\log k)$

encoding distances

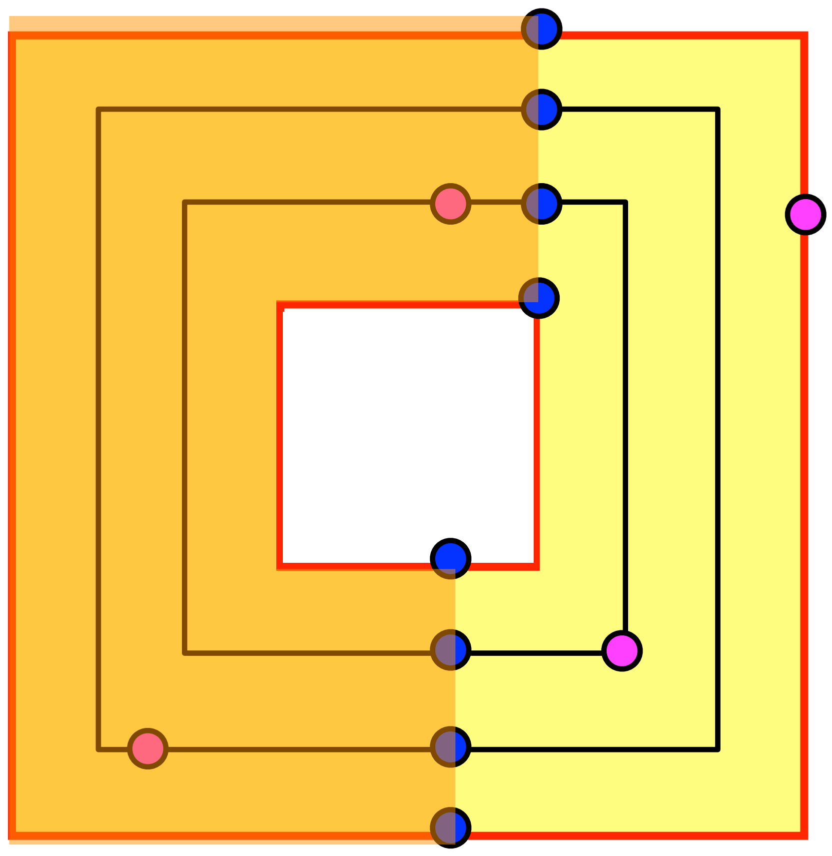


encoding distances

- boundary-to-boundary (unit-Monge) $\tilde{O}(n/w)$



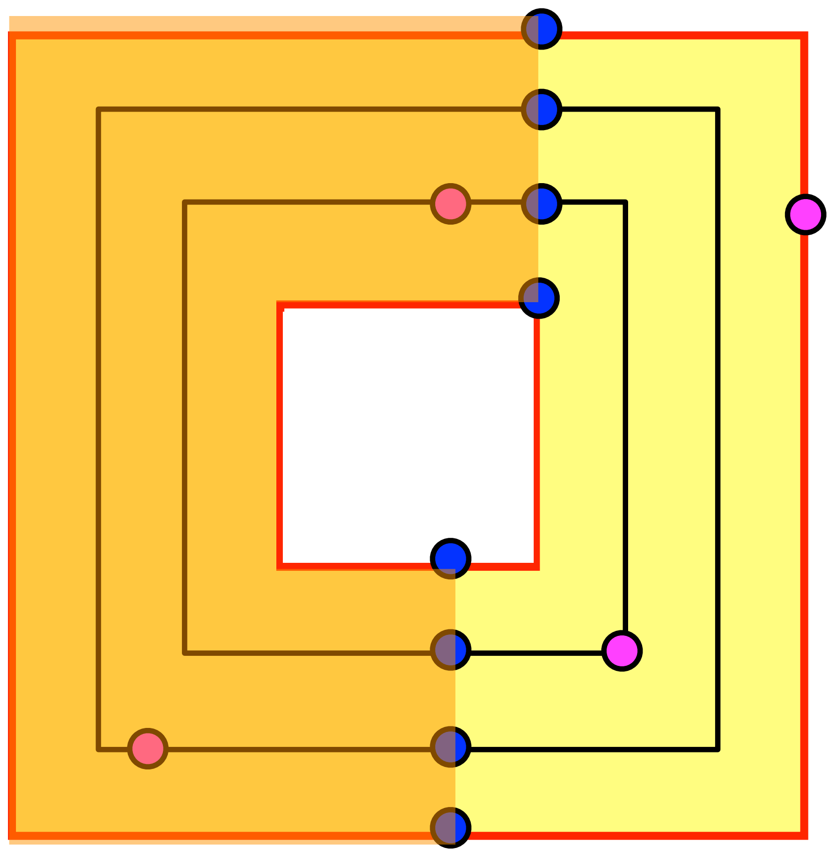
encoding distances



- boundary-to-boundary (unit-Monge) $\tilde{O}(n/w)$
- boundary-to-separator (unit-Monge) $\tilde{O}(n/w+kw)$

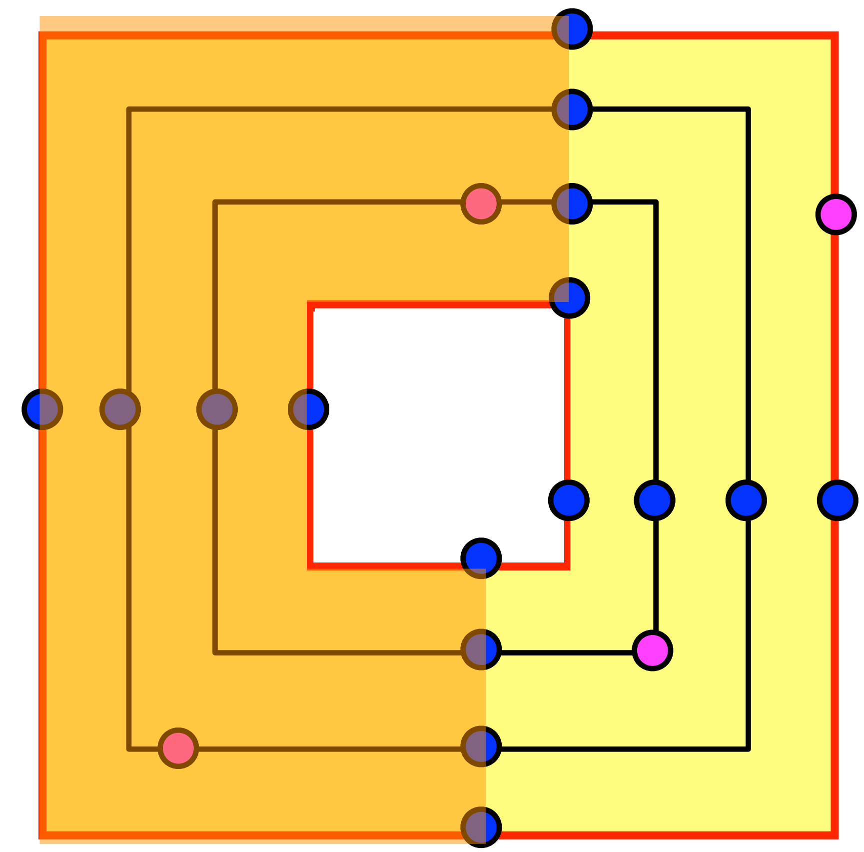
encoding distances

- boundary-to-boundary (unit-Monge) $\tilde{O}(n/w)$
- boundary-to-separator (unit-Monge) $\tilde{O}(n/w+kw)$
- terminals-to-separator (explicitly) $\tilde{O}(kw)$



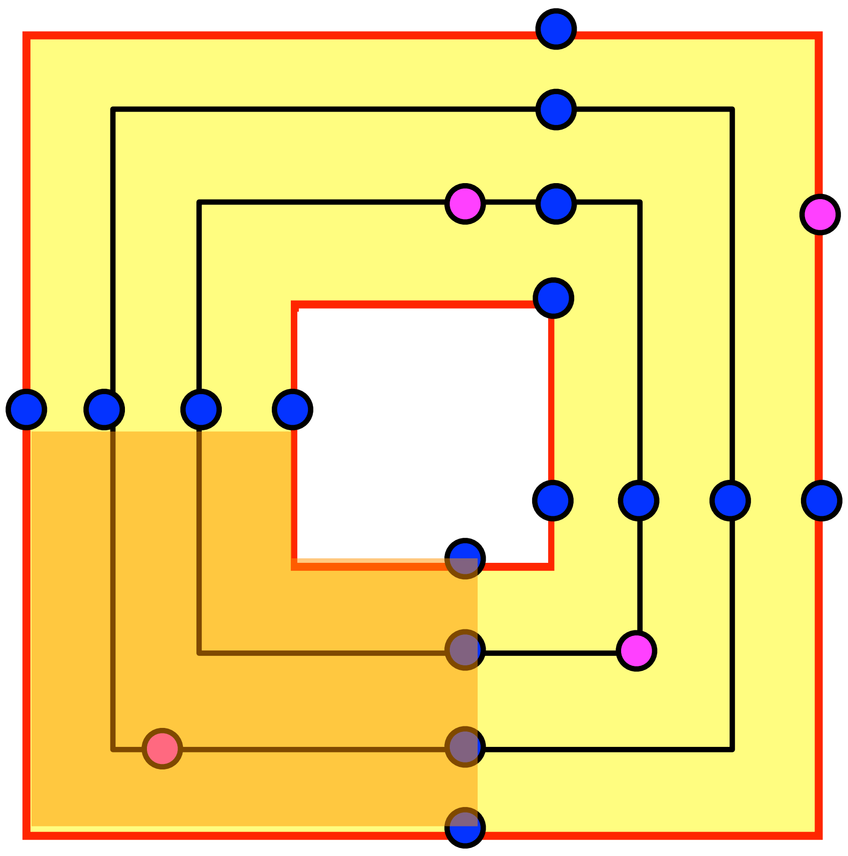
encoding distances

- boundary-to-boundary (unit-Monge) $\tilde{O}(n/w)$
- boundary-to-separator (unit-Monge) $\tilde{O}(n/w+kw)$
- terminals-to-separator (explicitly) $\tilde{O}(kw)$

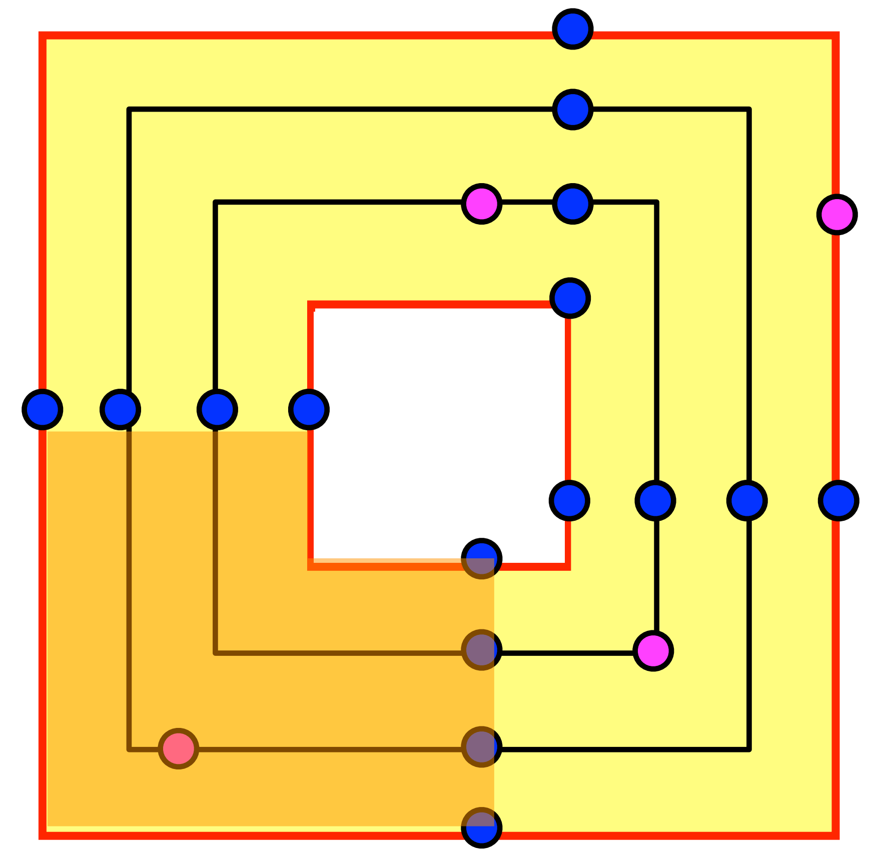


encoding distances

- boundary-to-boundary (unit-Monge) $\tilde{O}(n/w)$
- boundary-to-separator (unit-Monge) $\tilde{O}(n/w+kw)$
- terminals-to-separator (explicitly) $\tilde{O}(kw)$
- terminal-to-boundary in its region (explicitly) $\tilde{O}(n/w)$



encoding distances

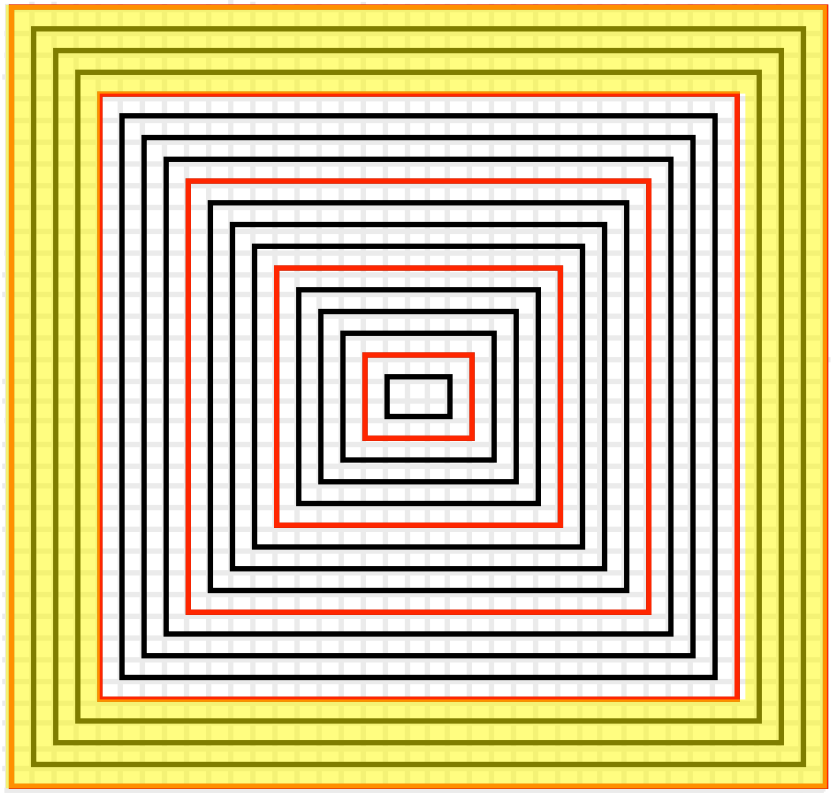


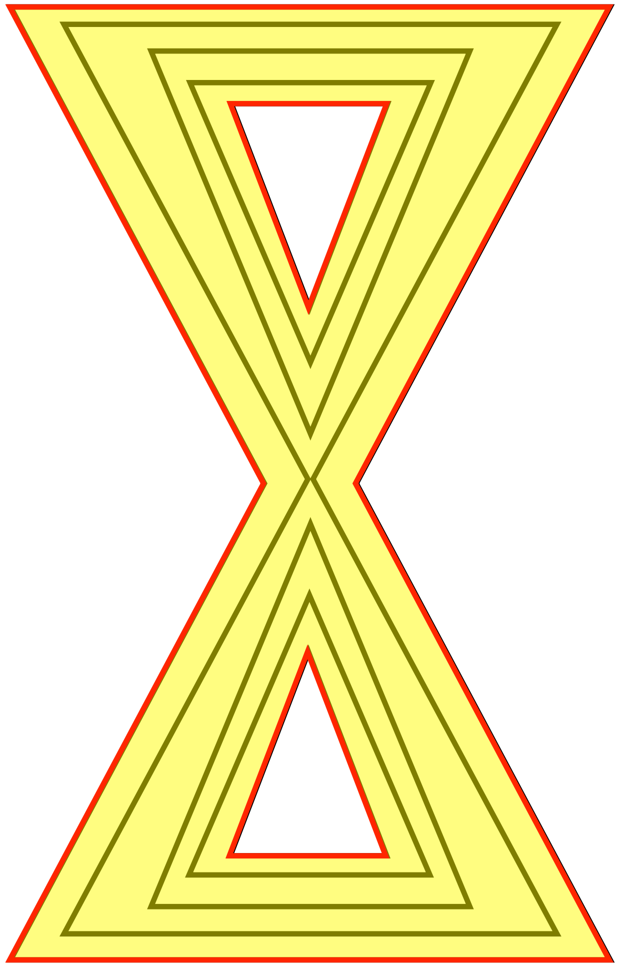
- boundary-to-boundary (unit-Monge) $\tilde{O}(n/w)$
- boundary-to-separator (unit-Monge) $\tilde{O}(n/w+kw)$
- terminals-to-separator (explicitly) $\tilde{O}(kw)$
- terminal-to-boundary in its region (explicitly) $\tilde{O}(n/w)$

choosing $w = (n/k)^{1/2}$ yields $\tilde{O}((nk)^{1/2})$

The curse of holes

- a slice can have many holes
- then cannot afford to store boundary-to-boundary distances
- instead, take terminals enclosed by holes into account
- if a single hole contains most of the terminals, there is no longer a balanced separator
- a special mechanism handles such holes reminiscent of heavy-path decomposition





Subset distance oracle

- can regard encoding as a union of:
  - cliques (distances stored via unit-Monge matrices)
  - stars (distances stored explicitly)
- Fakcharoenphol and Rao [FOCS'01] have an efficient implementation of Dijkstra's algorithm in this scenario
- combining the two immediately yields a distance oracle with $\tilde{O}((nk)^{1/2})$ space and query time
- using separators we get query time $\tilde{O}(\min(n^{3/4} , (nk)^{1/2})$

Our results

- the shortest path metric of undirected unweighted planar graphs can be encoded using $\tilde{O}((nk)^{1/2})$ bits (nearly optimal)
- first nontrivial subset oracle for undirected unweighted planar graphs
- weights (even linearly bounded) significantly change the complexity of the planar graph metric

Discussion

- we show it is unlikely that our encoding can be used to get a distance labeling scheme with $o(n^{1/2})$ bits per label

- the $\Omega(n^{1/3})$ lower bound (undirected) of Gavoille et al. for distance labeling cannot be improved by current techniques

- what about directed unweighted planar graphs?

- can the unit-Monge property be further exploited?
  e.g. distance-product of two n-by-n unit-Monge matrices can be computed in $O(n \log n)$ time (compared to $O(n^2)$ for Monge matrices and $O(n^3)$ for general matrices)

- can we get an oracle with query time better than $\tilde{O}(\min(n^{3/4} , (nk)^{1/2})$?