

# Incorporating distributed Dijkstra's algorithm into variogram calculation with locally varying anisotropy

Oscar Peredo, Felipe Navarro, Mauricio Garrido and Julián M. Ortiz

Department of Mining Engineering – FCFM – Universidad de Chile  
Advanced Mining Technology Center – Universidad de Chile

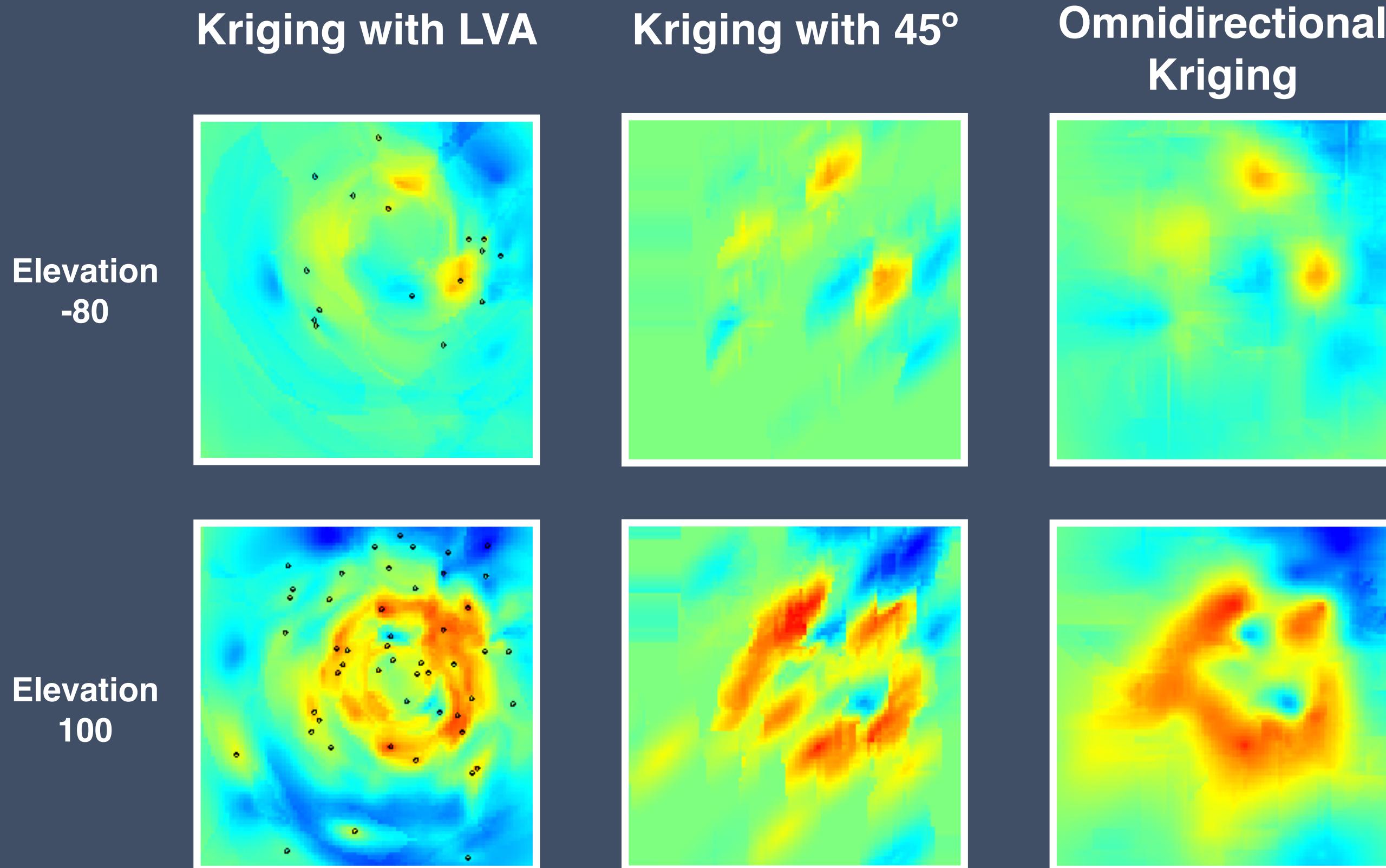
**37th APCOM Conference**  
**Fairbanks, Alaska 2015**



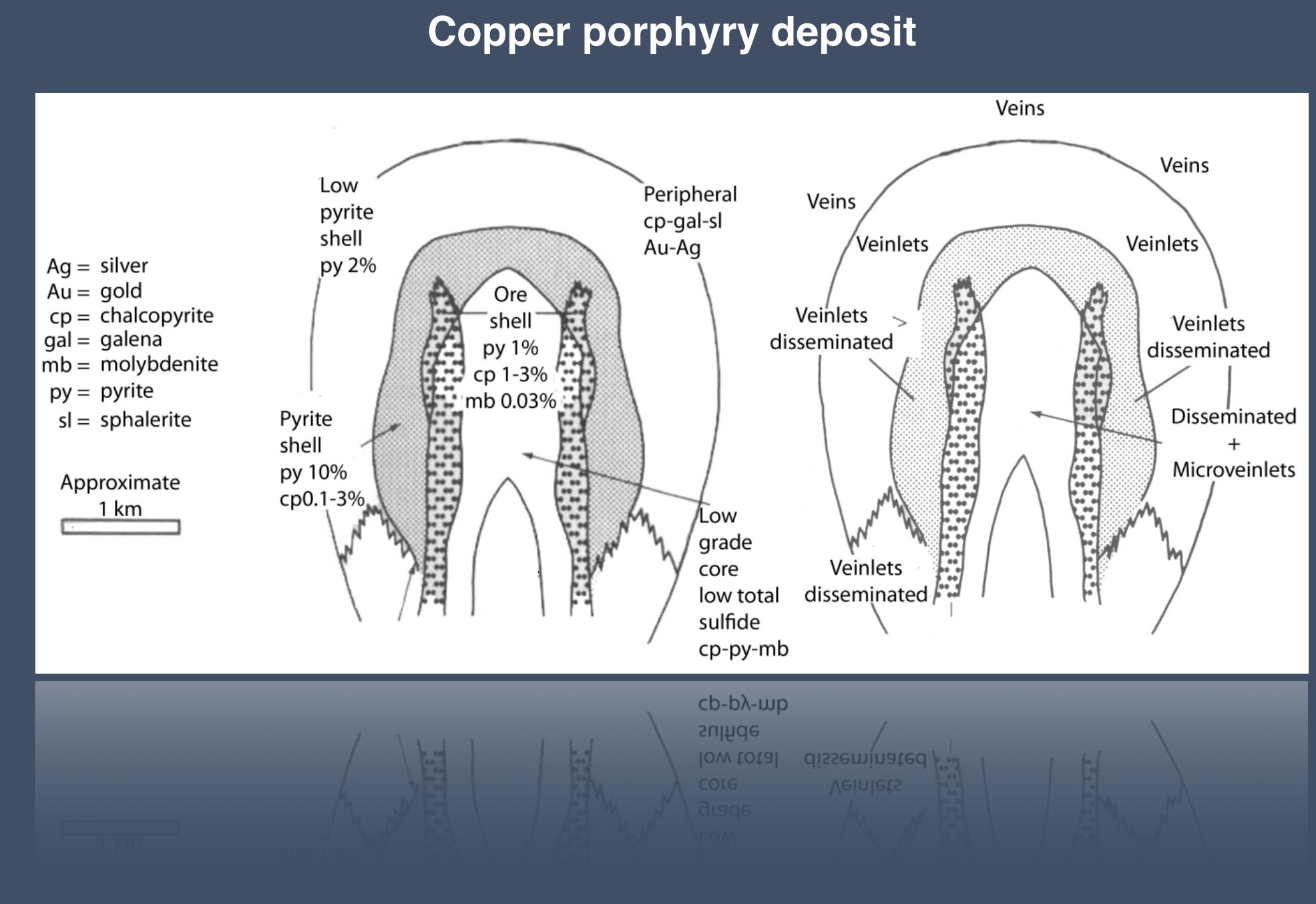
Ingeniería de Minas  
FACULTAD DE CIENCIAS  
FÍSICAS Y MATEMÁTICAS  
UNIVERSIDAD DE CHILE

**amtc**  
ADVANCED MINING TECHNOLOGY CENTER

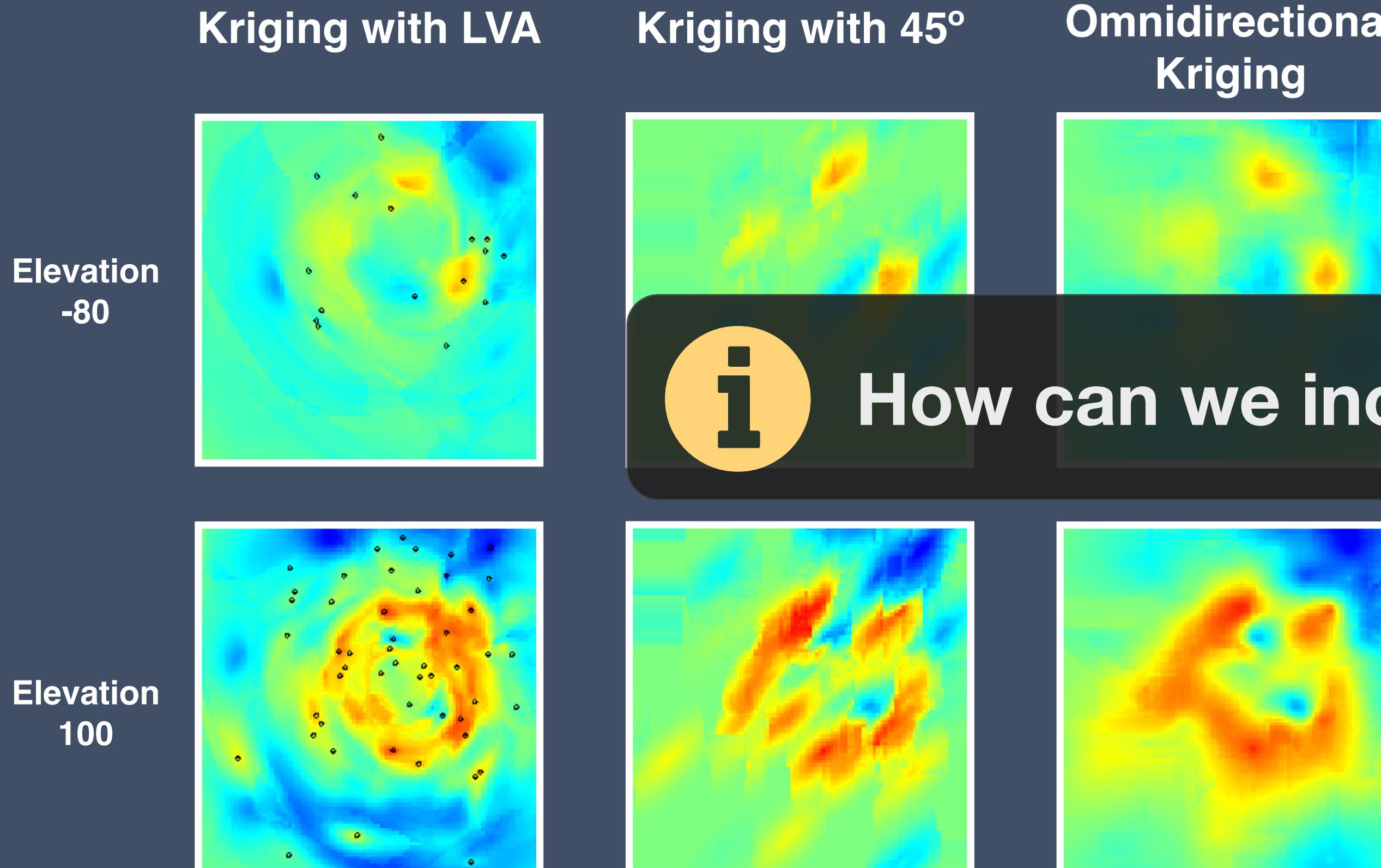
# Locally varying anisotropy (LVA)



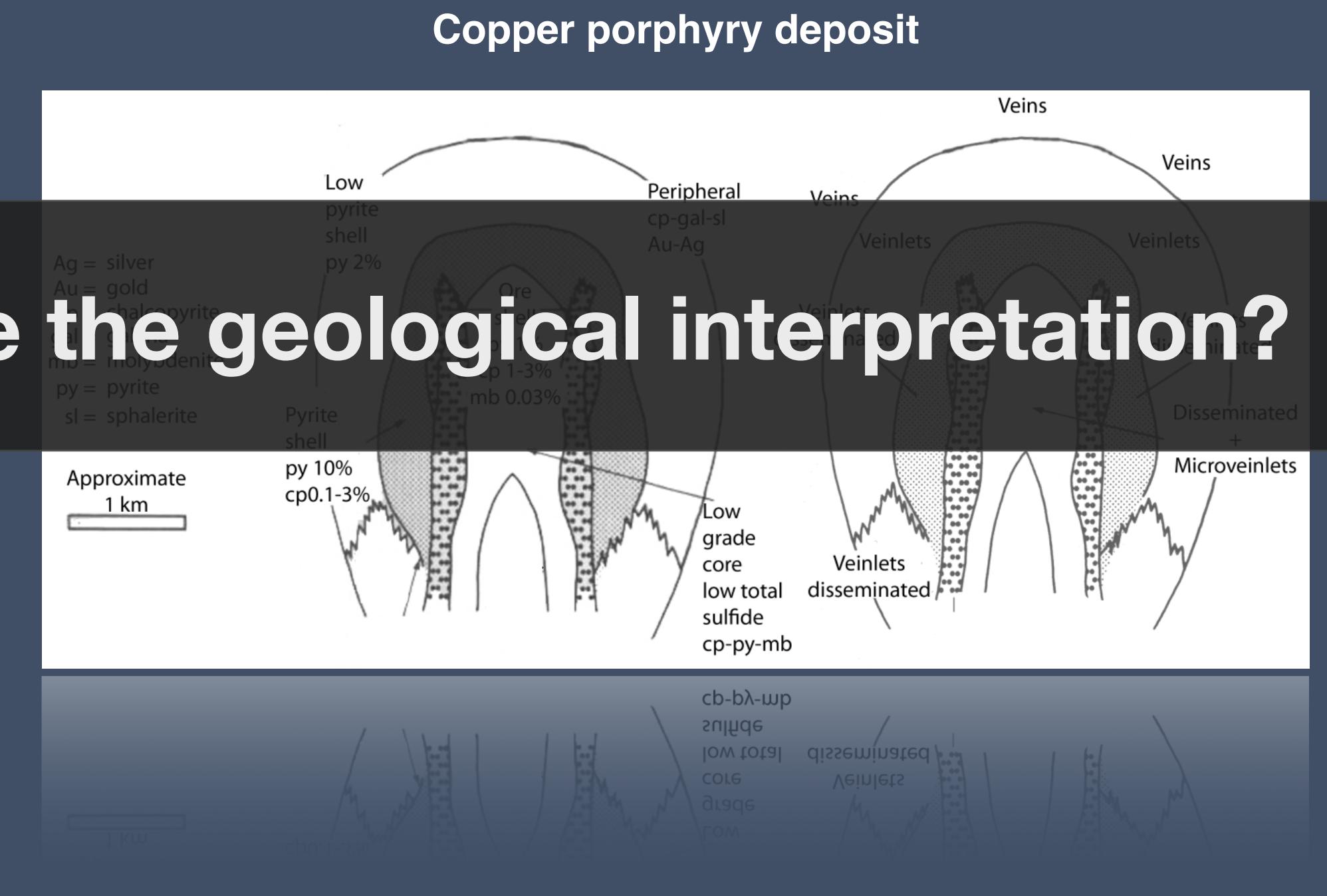
Source: Boisvert (2010)



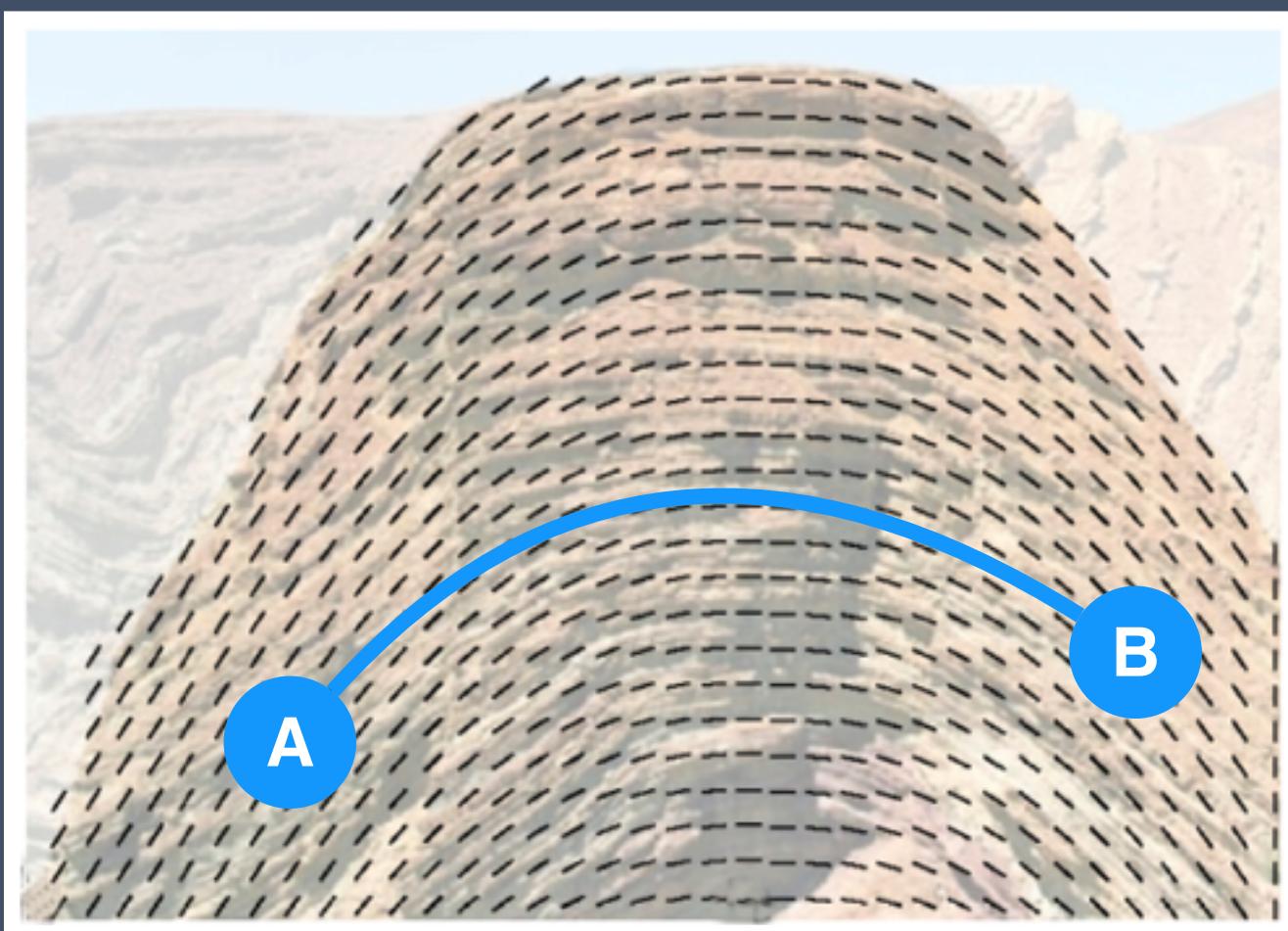
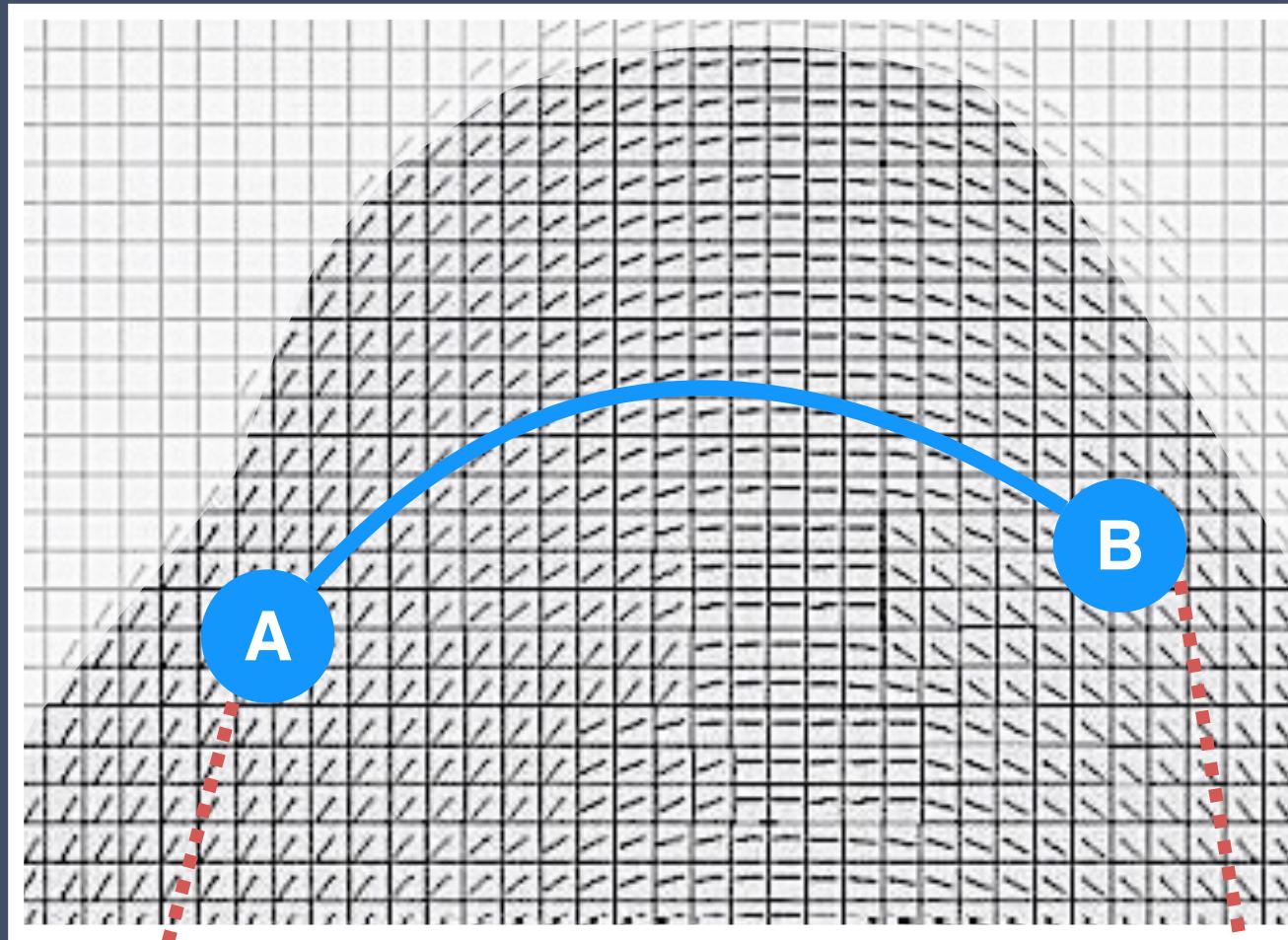
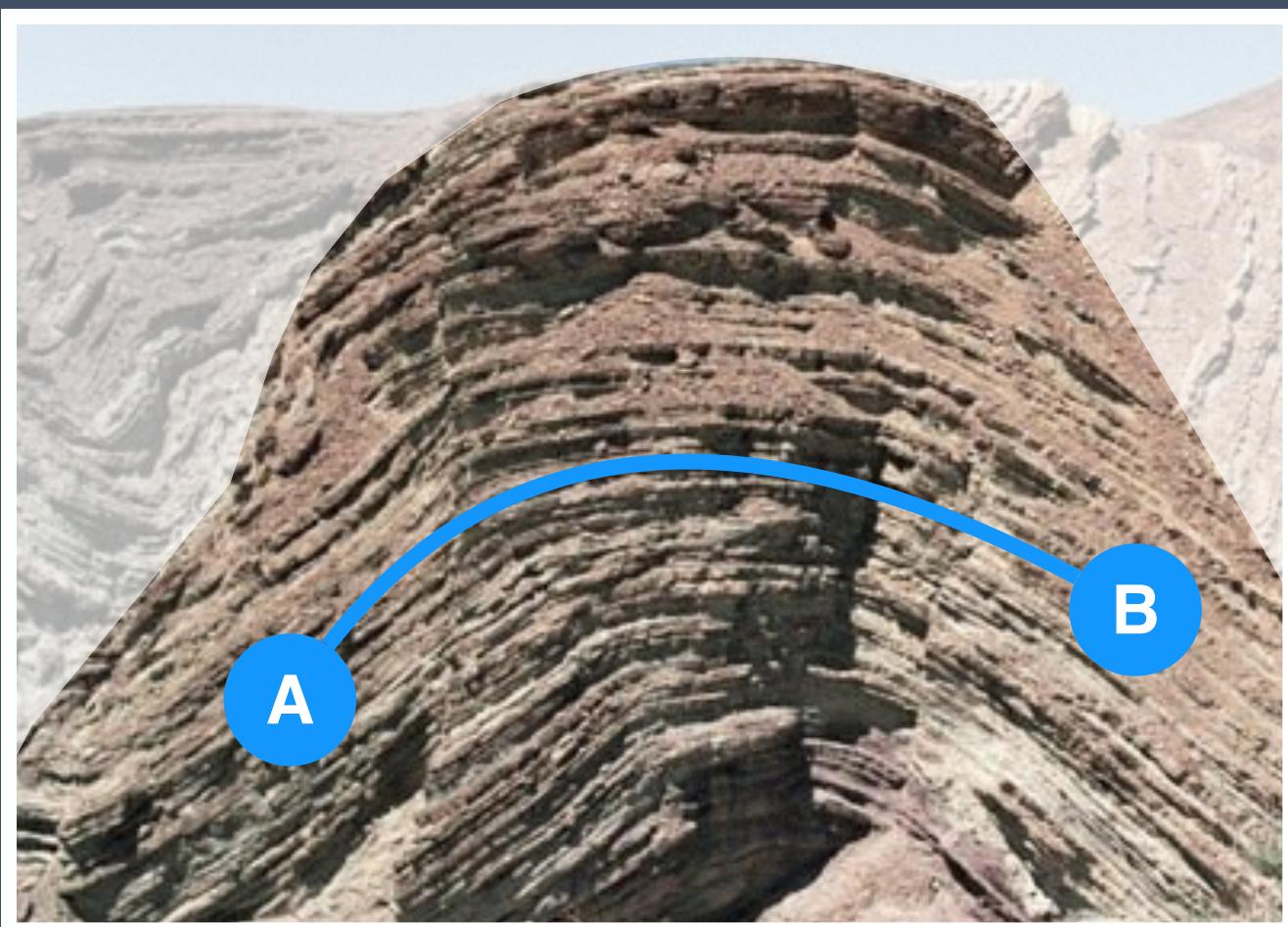
# Locally varying anisotropy (LVA)



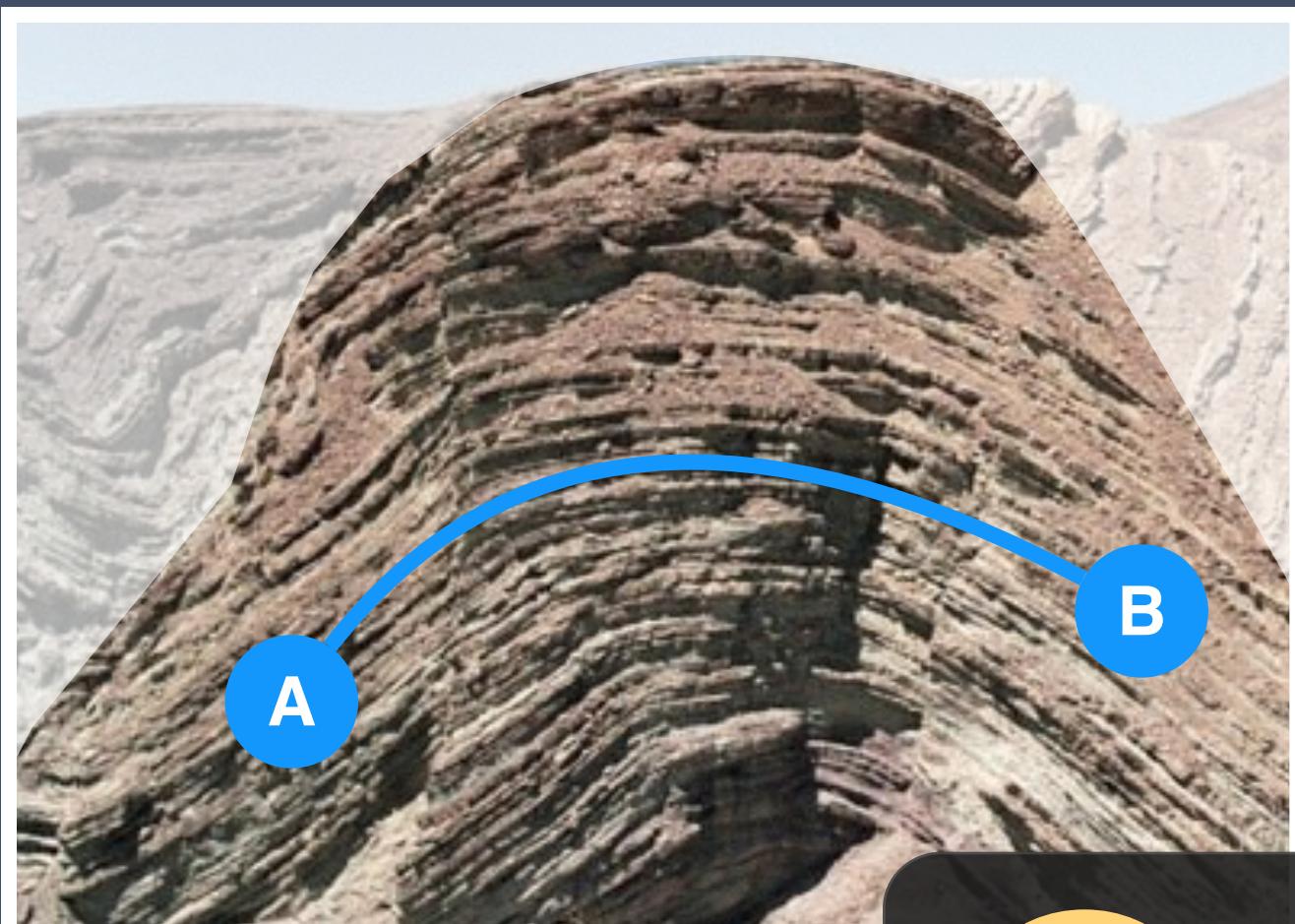
**How can we include the geological interpretation?**



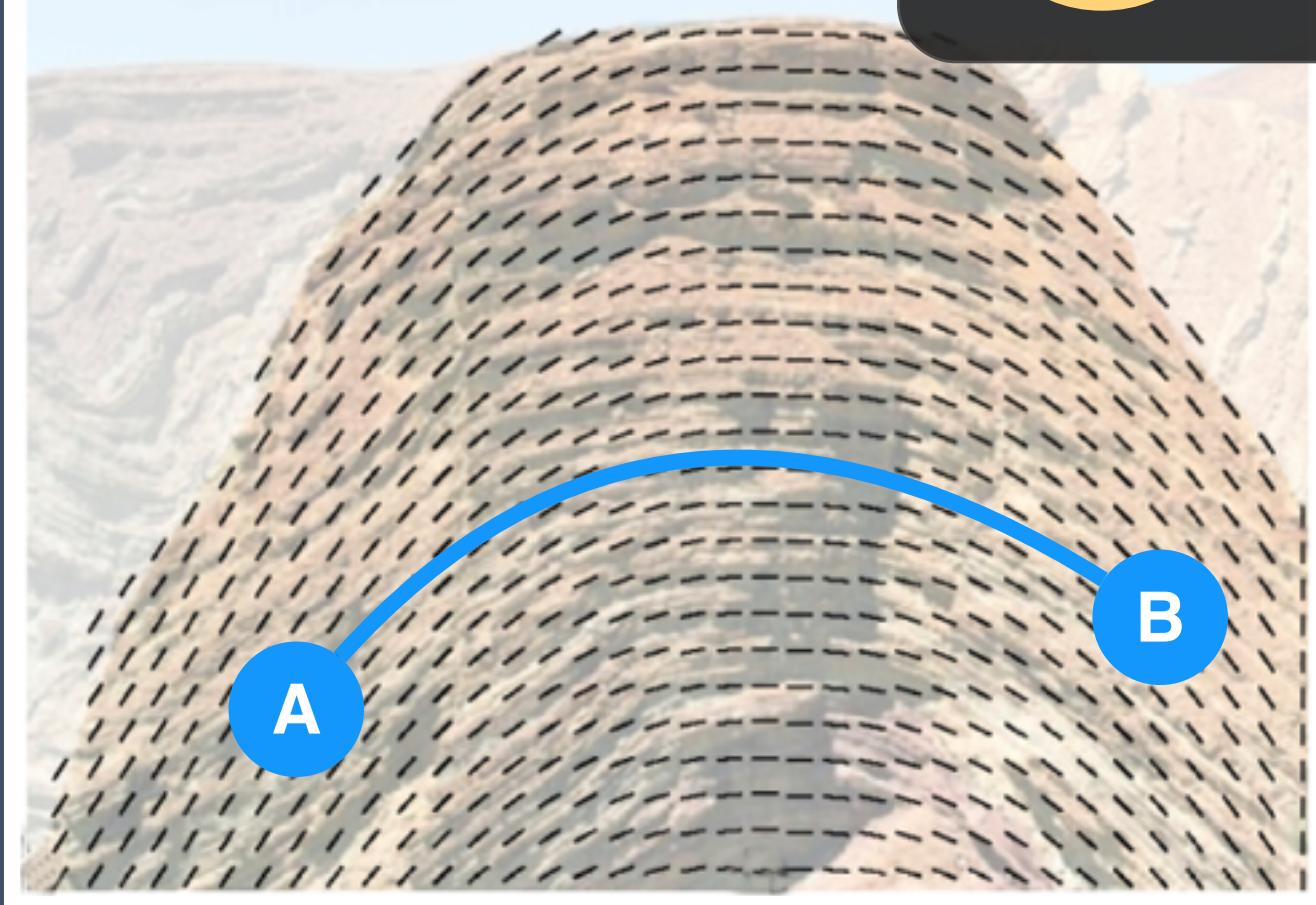
Source: Boisvert (2010)



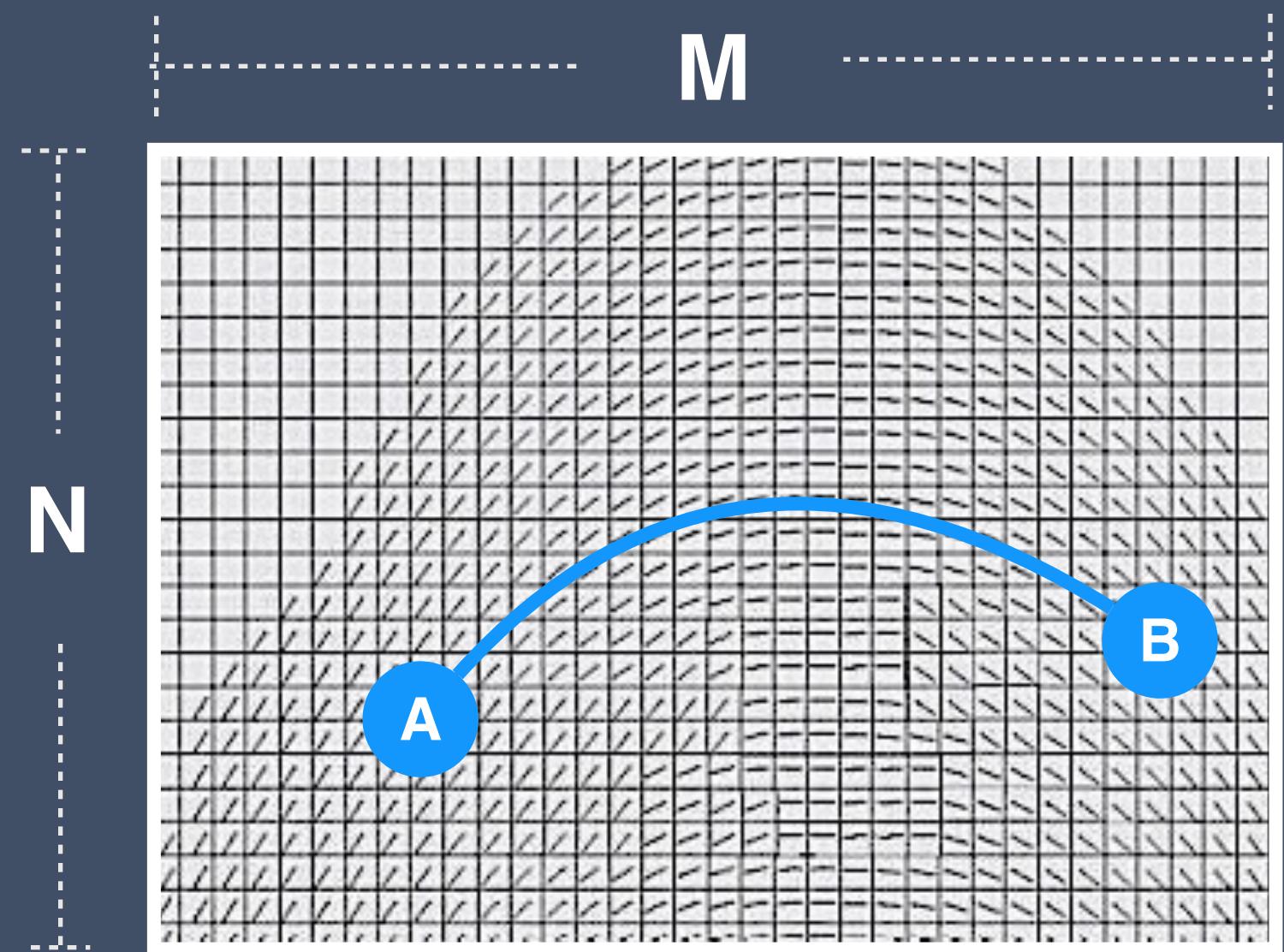
(\*) Picture from an anticline with LVA ([www.geology.about.com](http://www.geology.about.com))



**How can we calculate the anisotropic distance?**

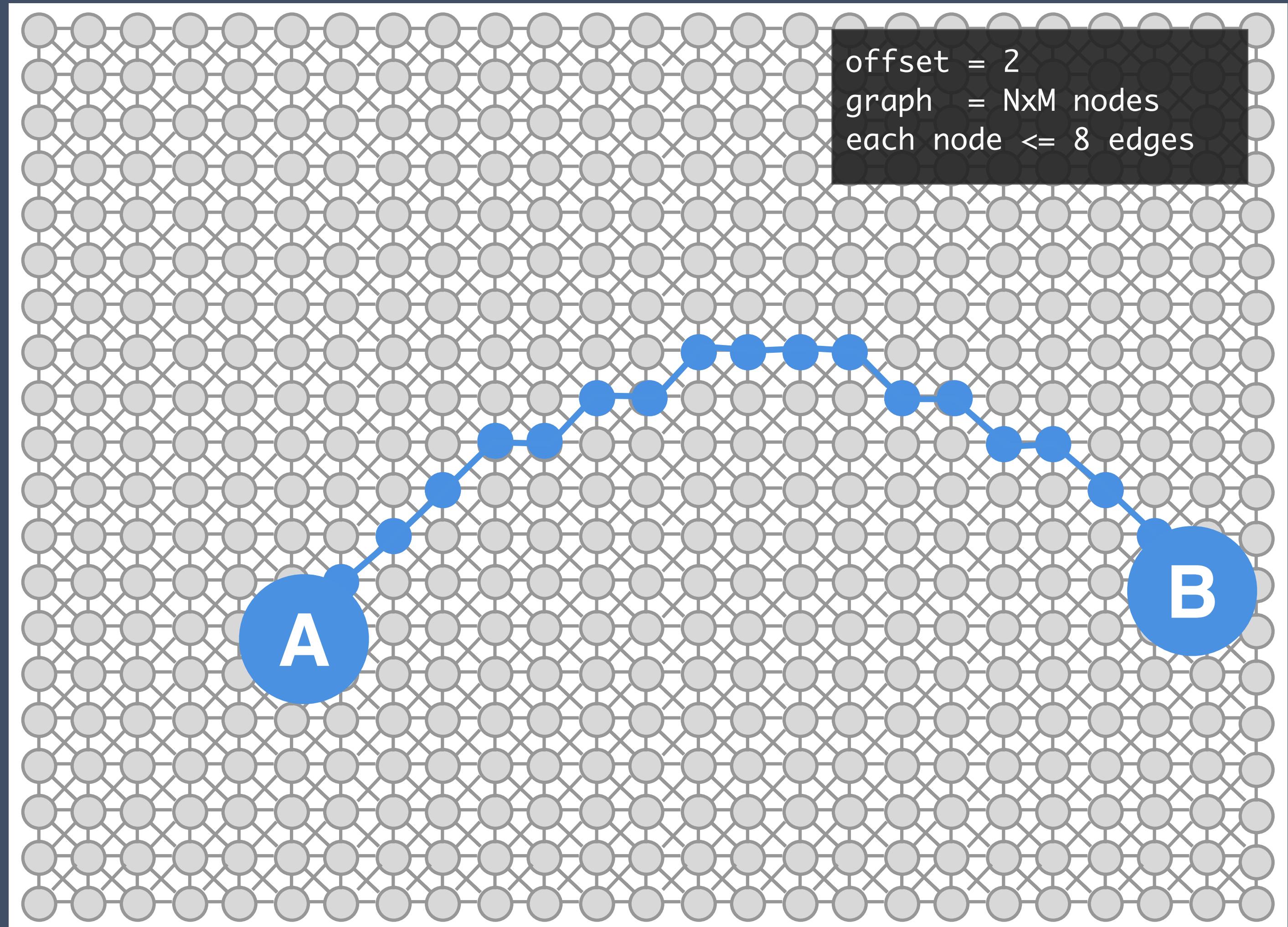


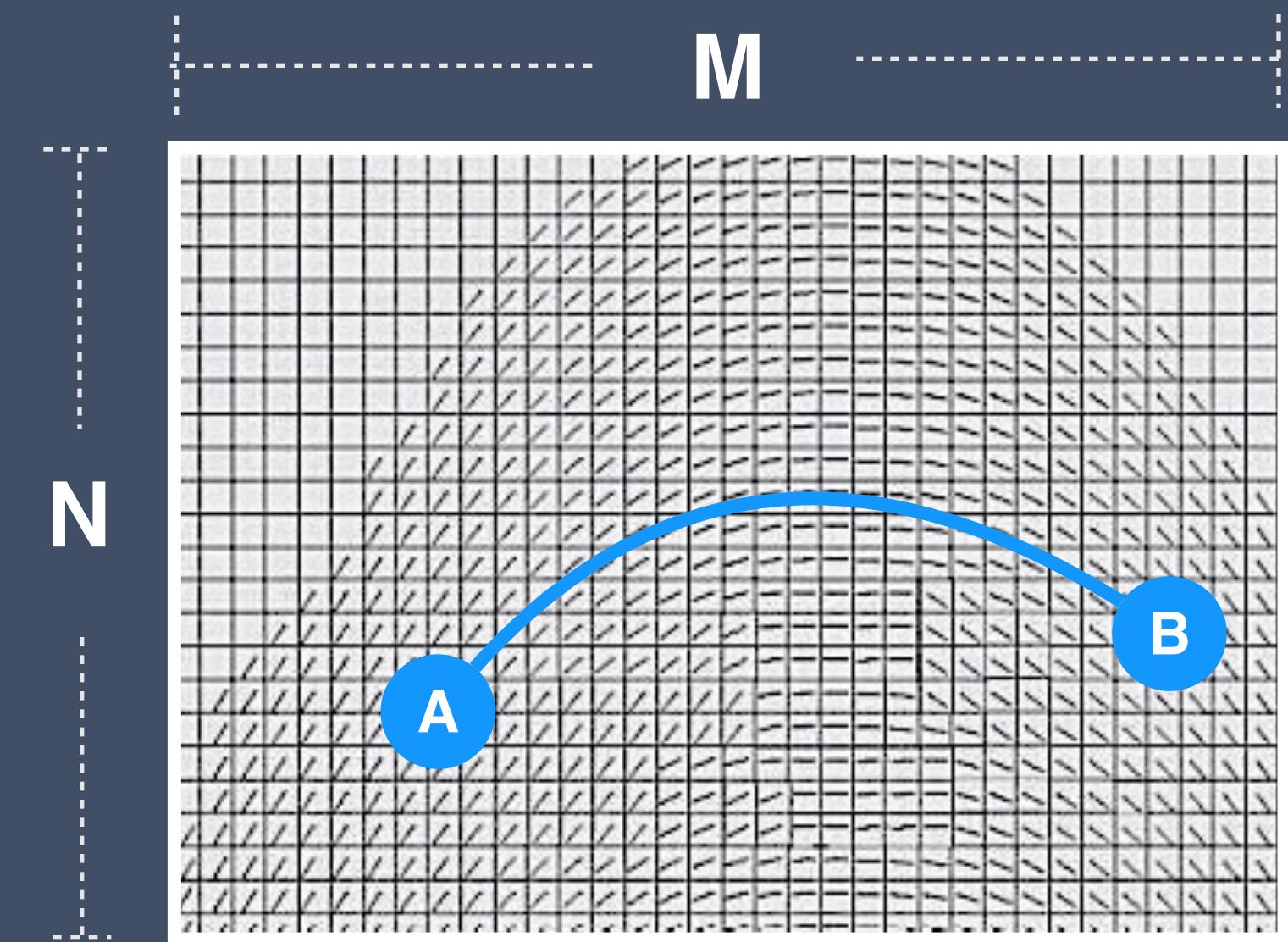
(\*) Picture from an anticline with LVA ([www.geology.about.com](http://www.geology.about.com))



## How to build the anisotropic distance matrix

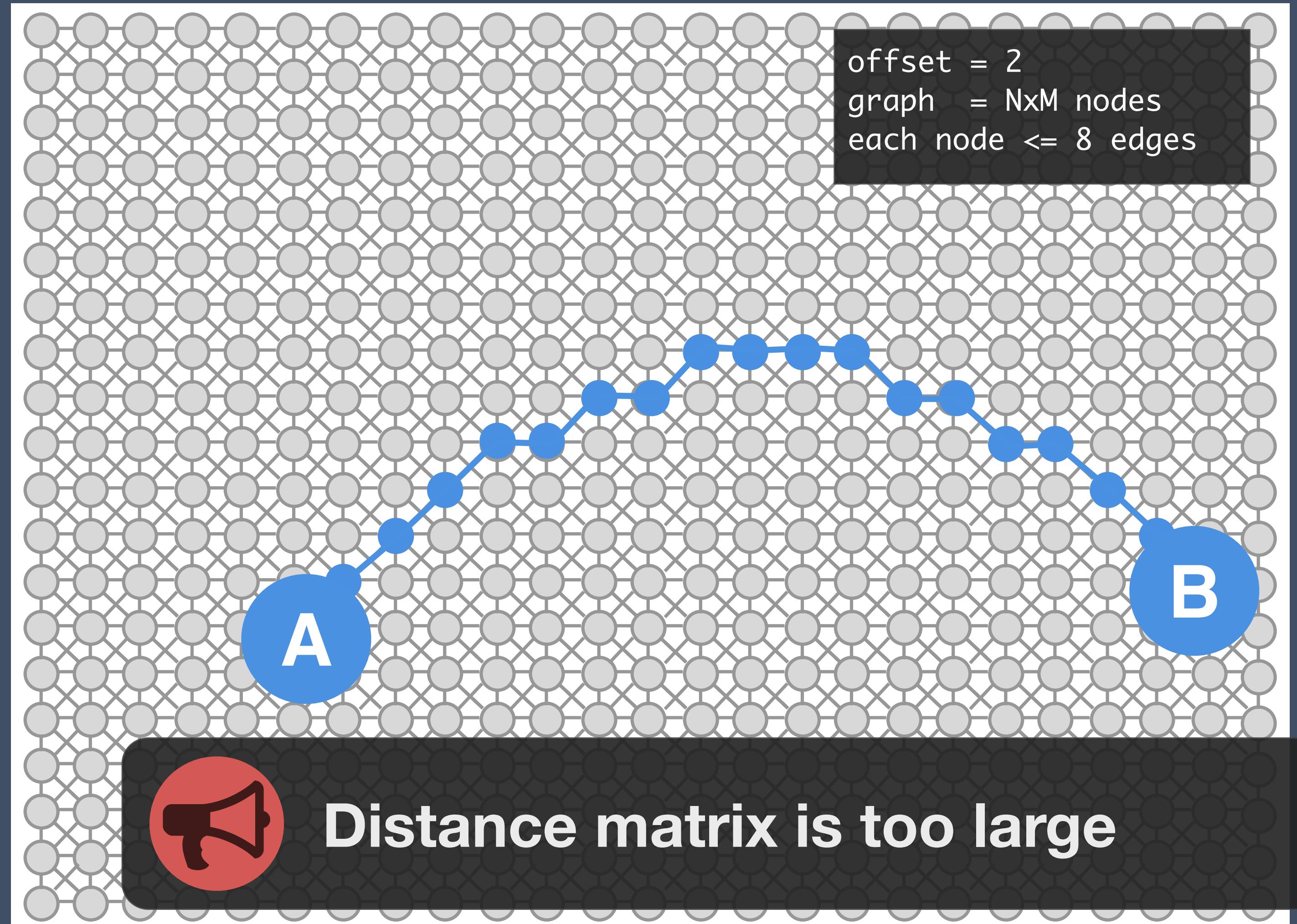
1. Define Graph offset (local connectivity)
2. Build Graph
3. Find shortest path between A and B (using single source Dijkstra)
4. Repeat for each node ( $N \times M$  Dijkstra runs)





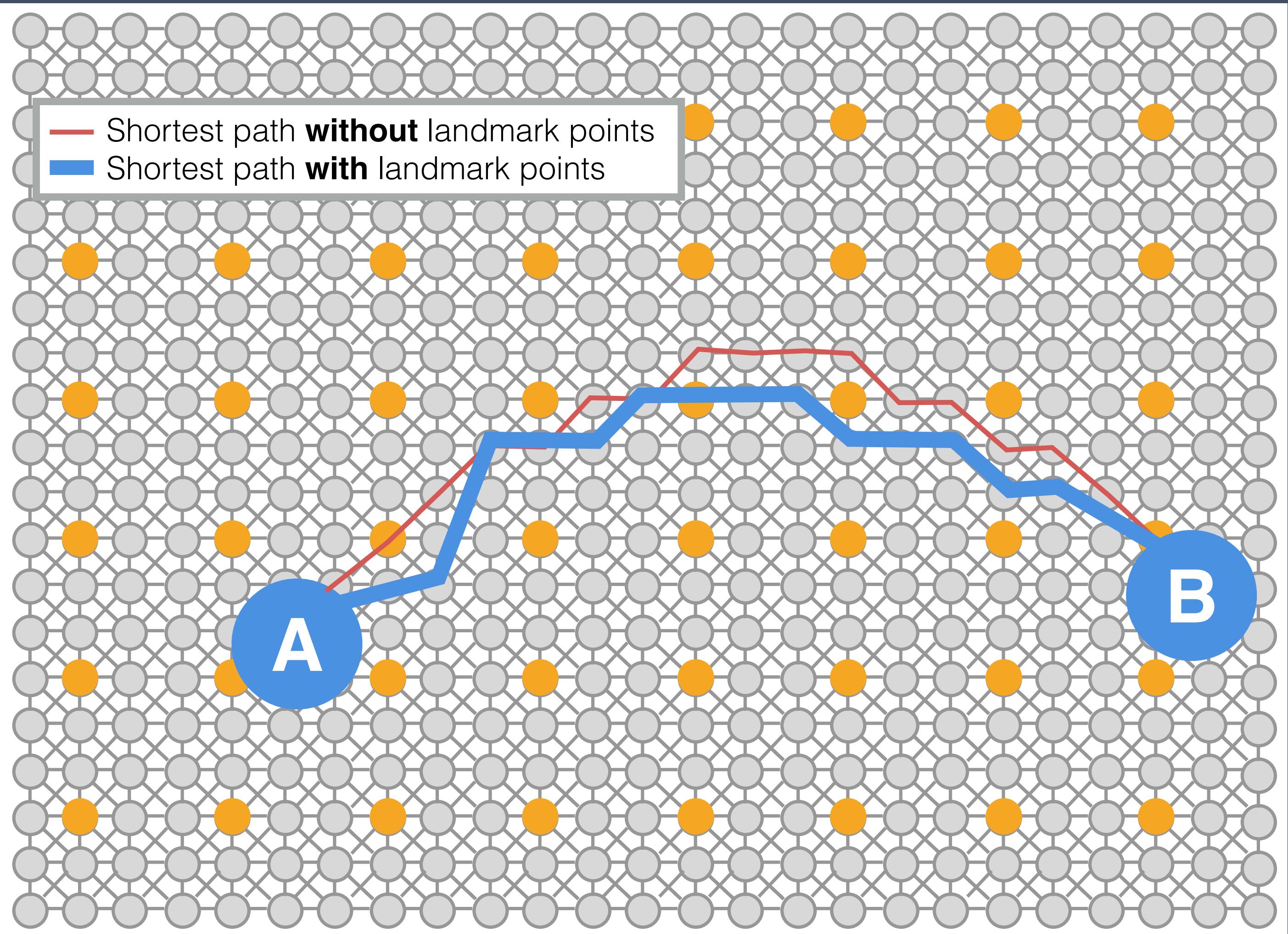
## How to build the anisotropic distance matrix

1. Define Graph offset (local connectivity)
2. Build Graph
3. Find shortest path between A and B (using single source Dijkstra)
4. Repeat for each node ( $N \times M$  Dijkstra runs)



# Dimensionality reduction using landmark points

- L-ISOMAP is used
- Distance matrix size is reduced  
 $L \times (N \times M) \ll (N \times M)^2$
- Trade-off: Numerical accuracy of anisotropic distance
- Shortest path calculation between landmark points and nodes (instead of between every node,  $L$  Dijkstra runs)

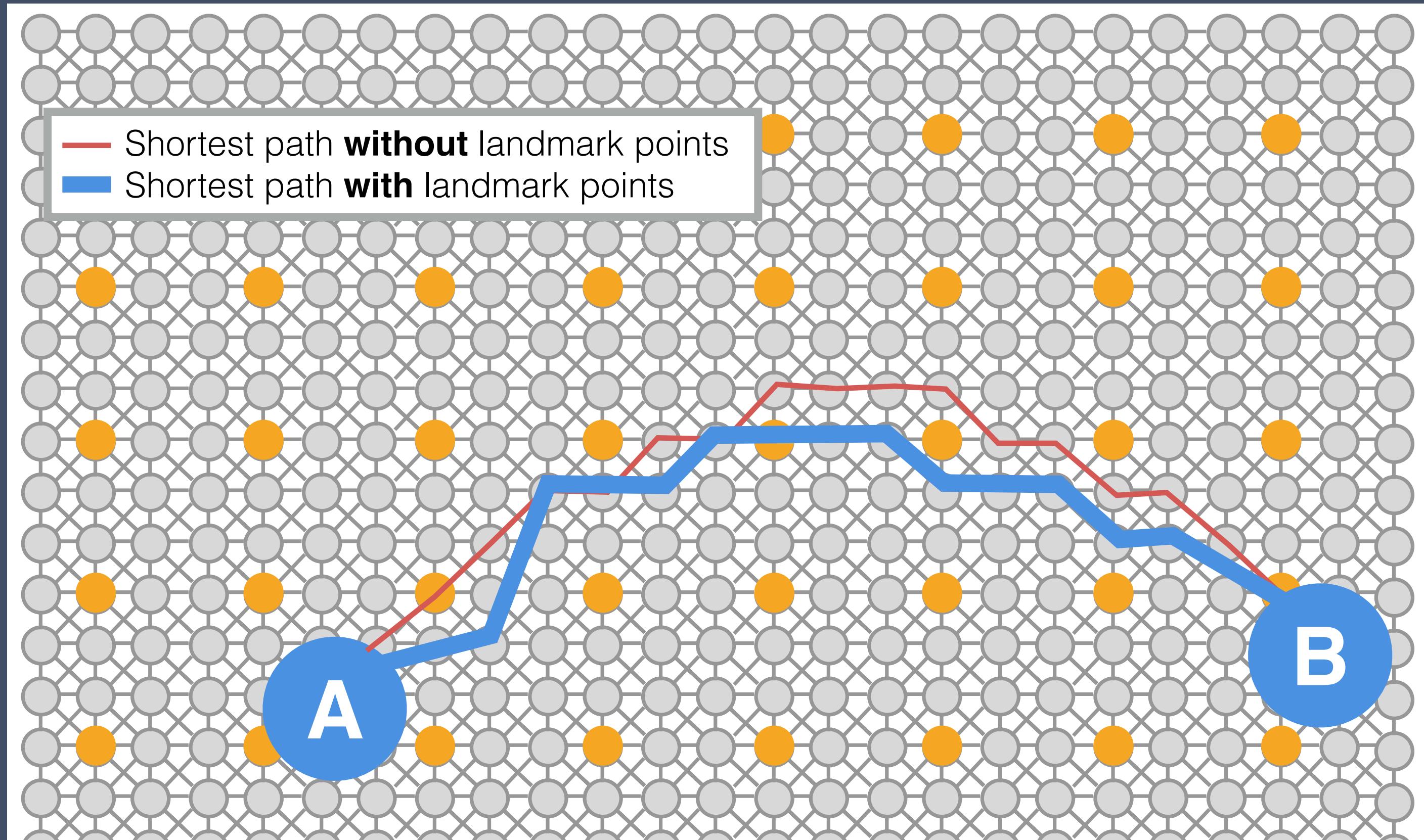


# Dimensionality reduction using landmark points

- L-ISOMAP is used
- Distance matrix size is reduced  
 $L \times (N \times M) \ll (N \times M)^2$
- Trade-off: Numerical accuracy of anisotropic distance
- Shortest path calculation between landmark points and nodes (instead of between every node,  $L$  Dijkstra runs)

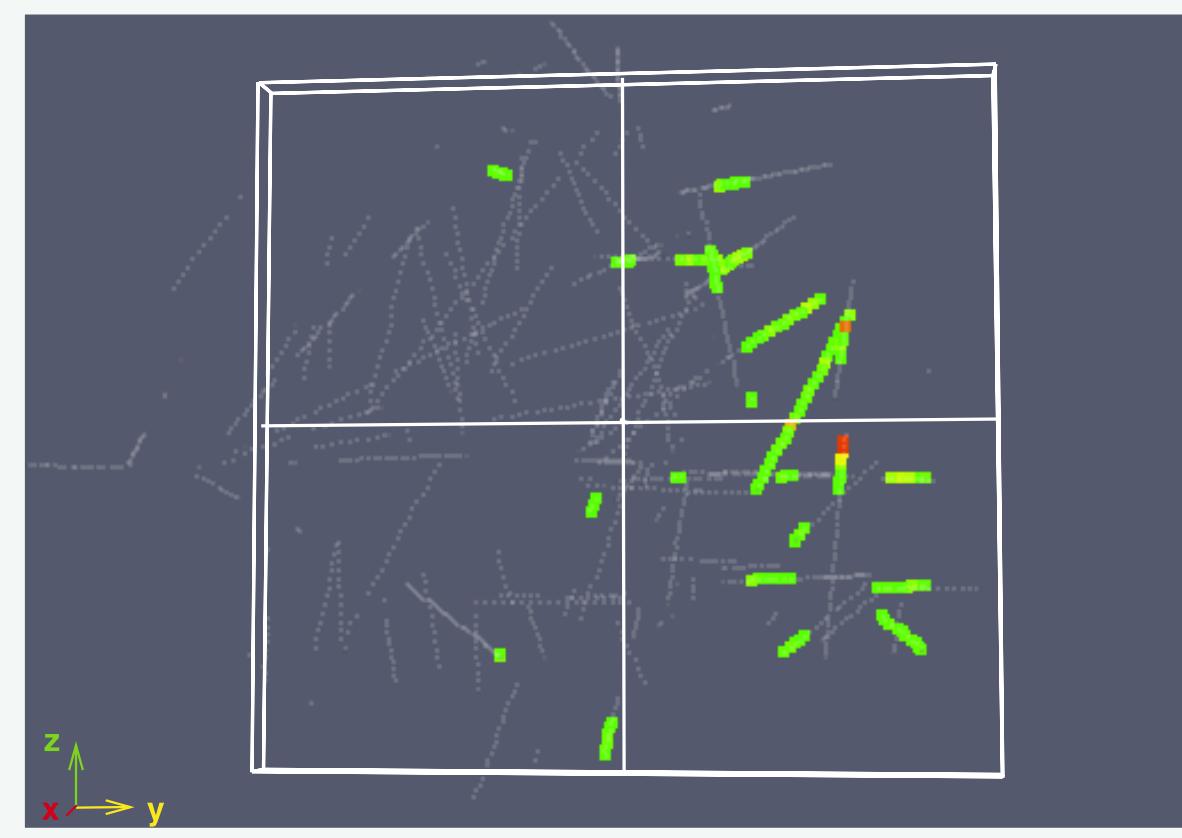
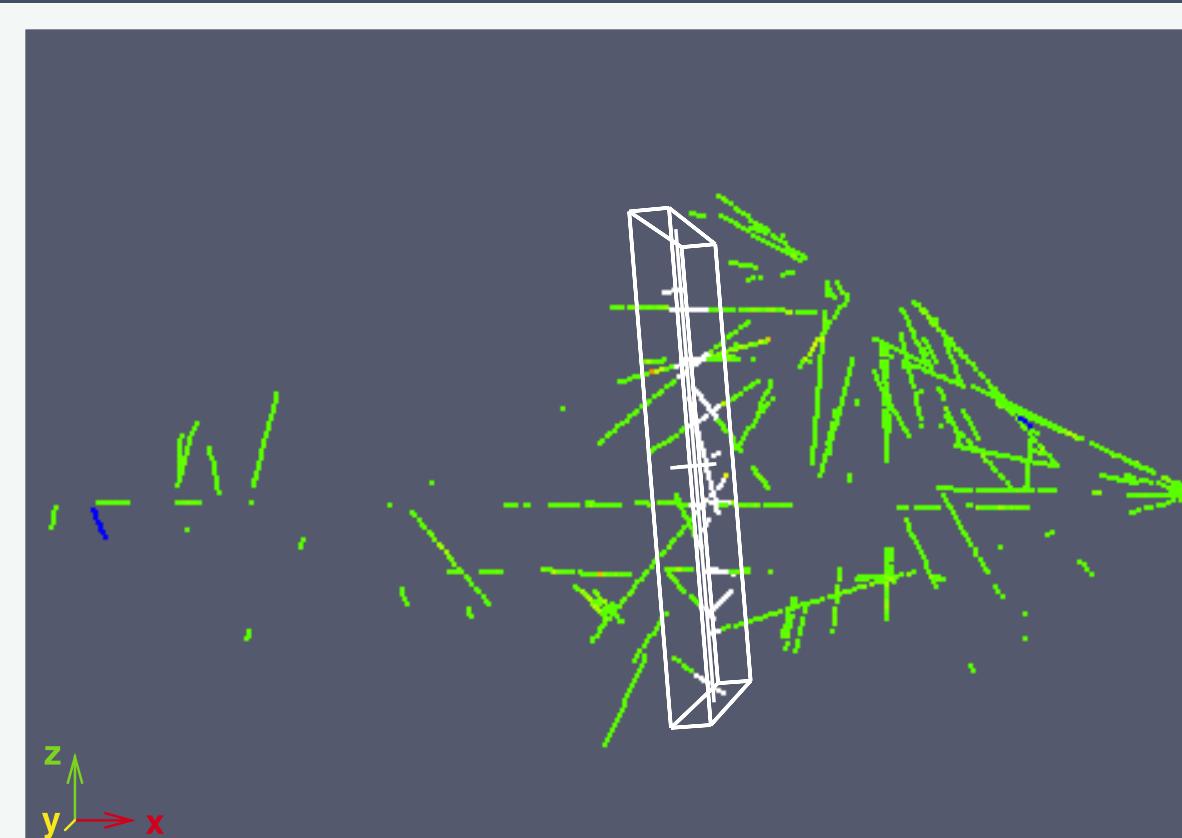


Shortest path

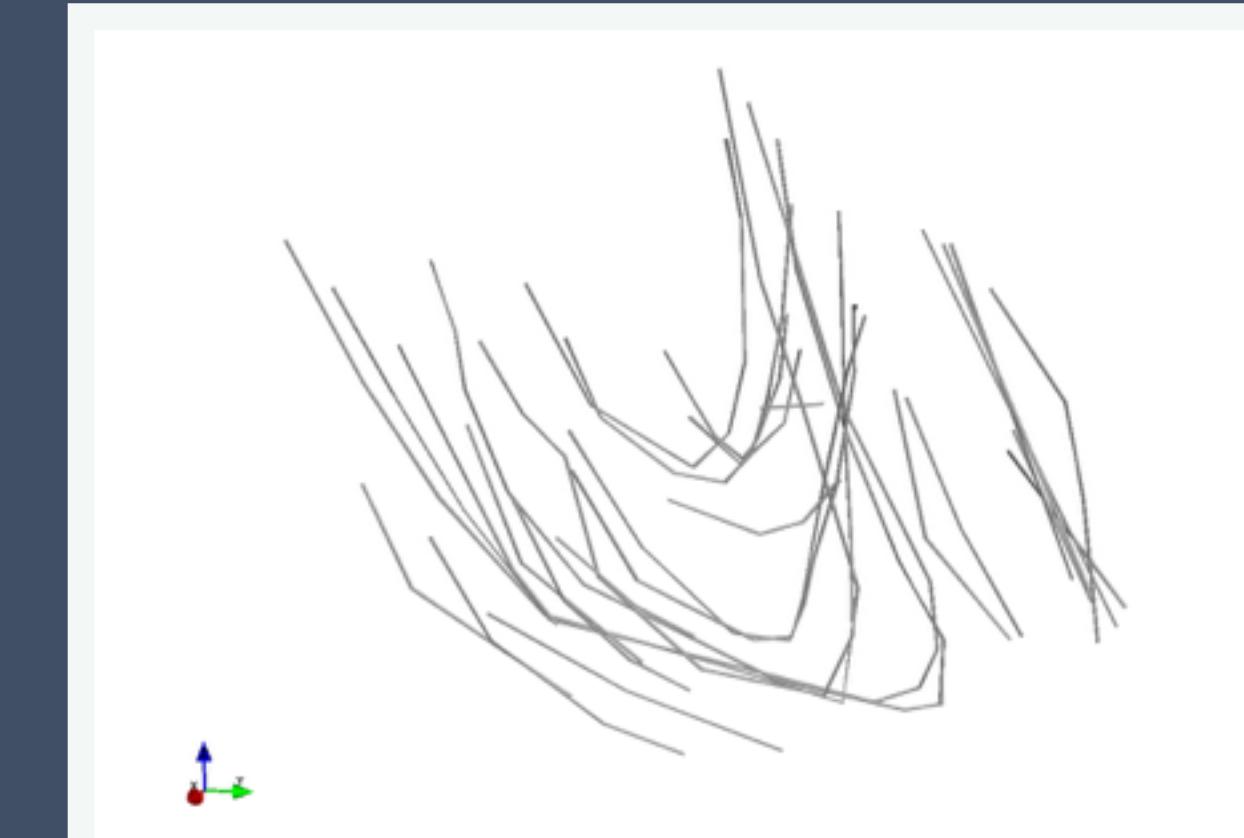


calculation is still intensive (RAM and CPU)

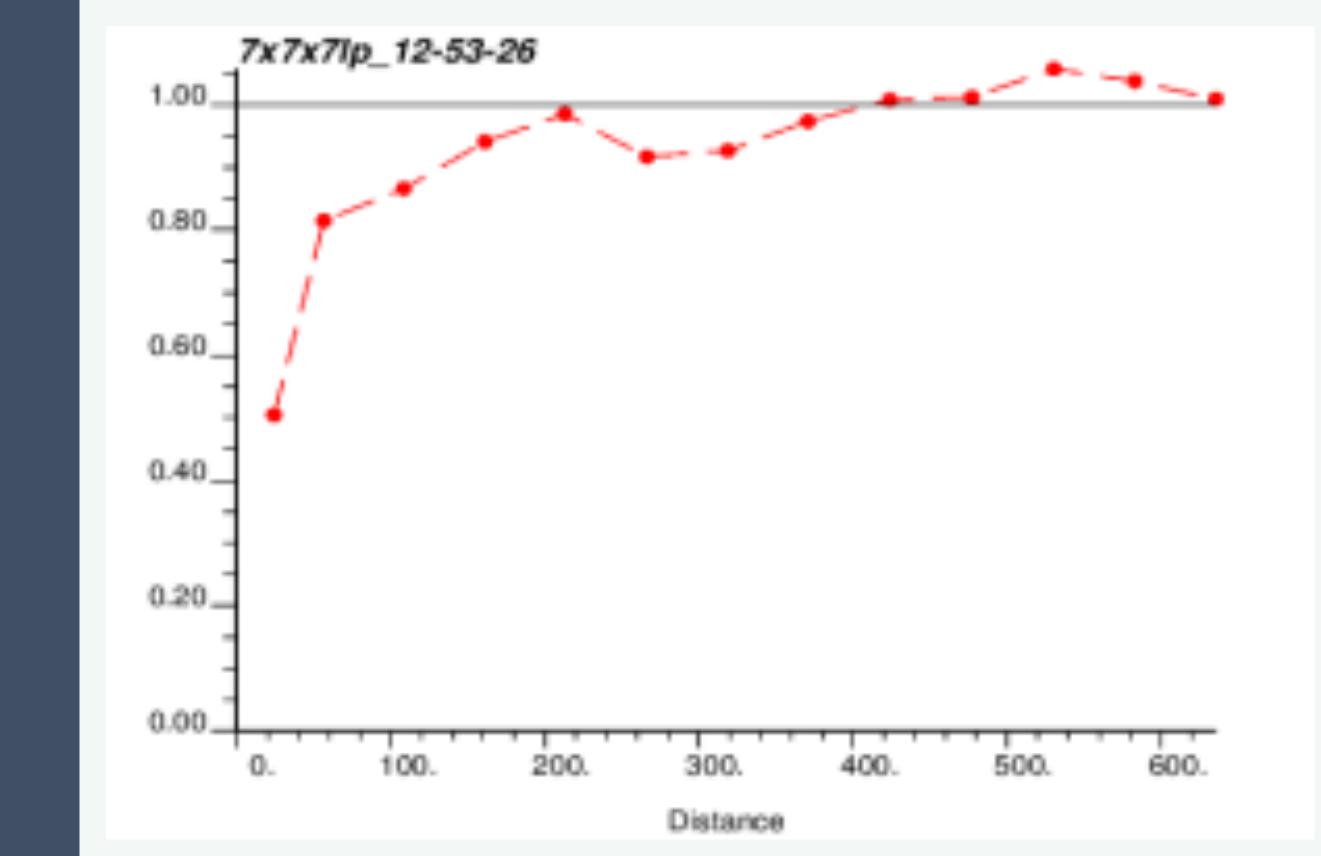
# Case study



Drillholes  
(800 gold grade samples)

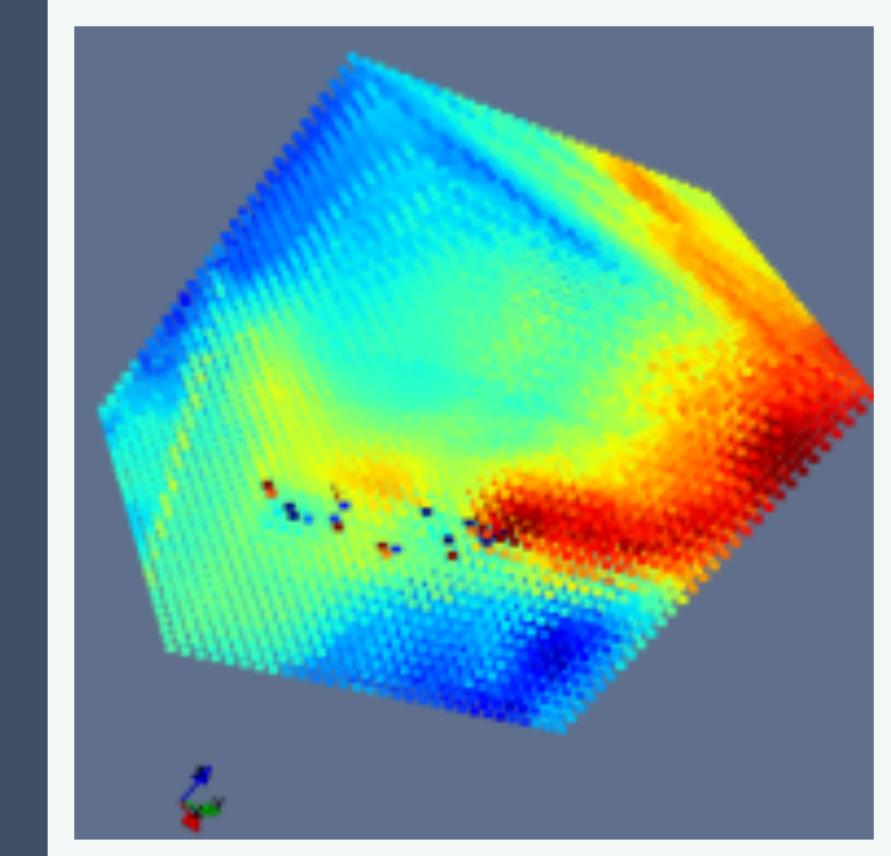


DXF lines  
LVA field ( $30 \times 30 \times 30$ )



Semi Variogram

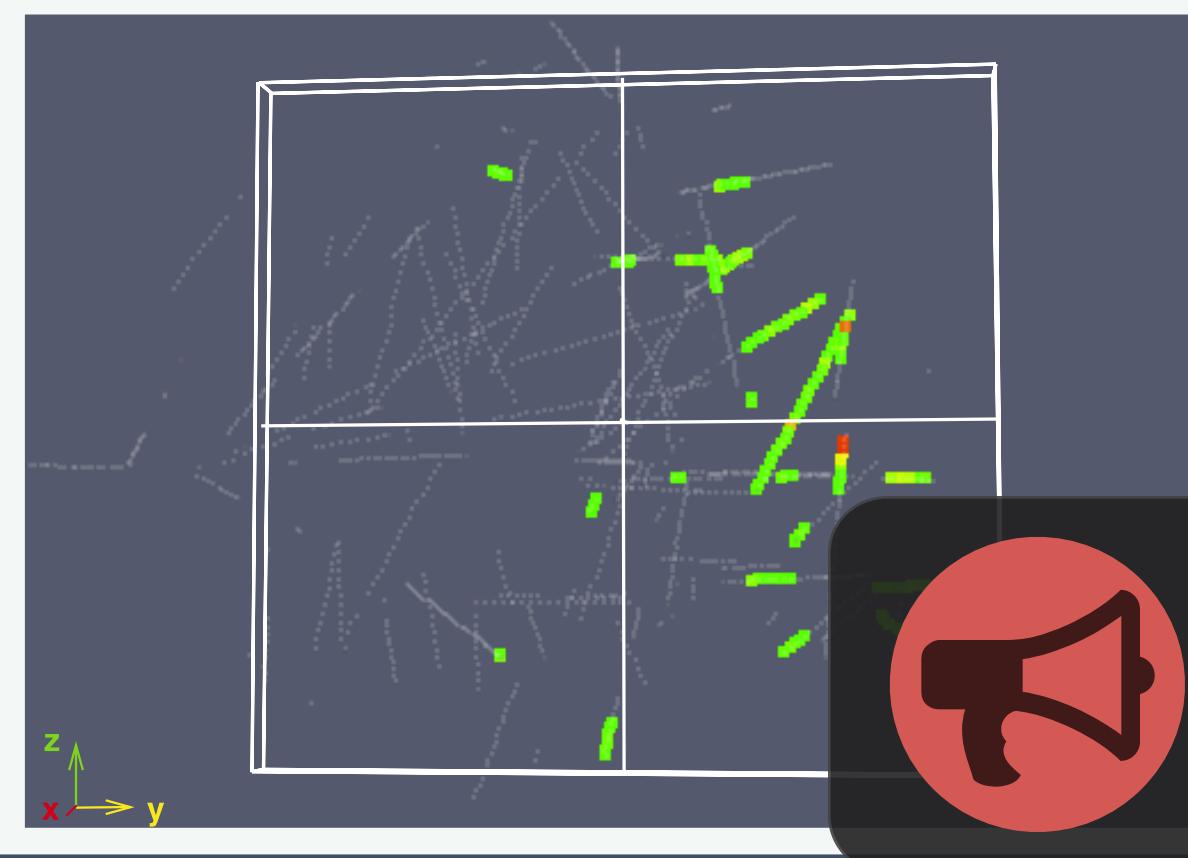
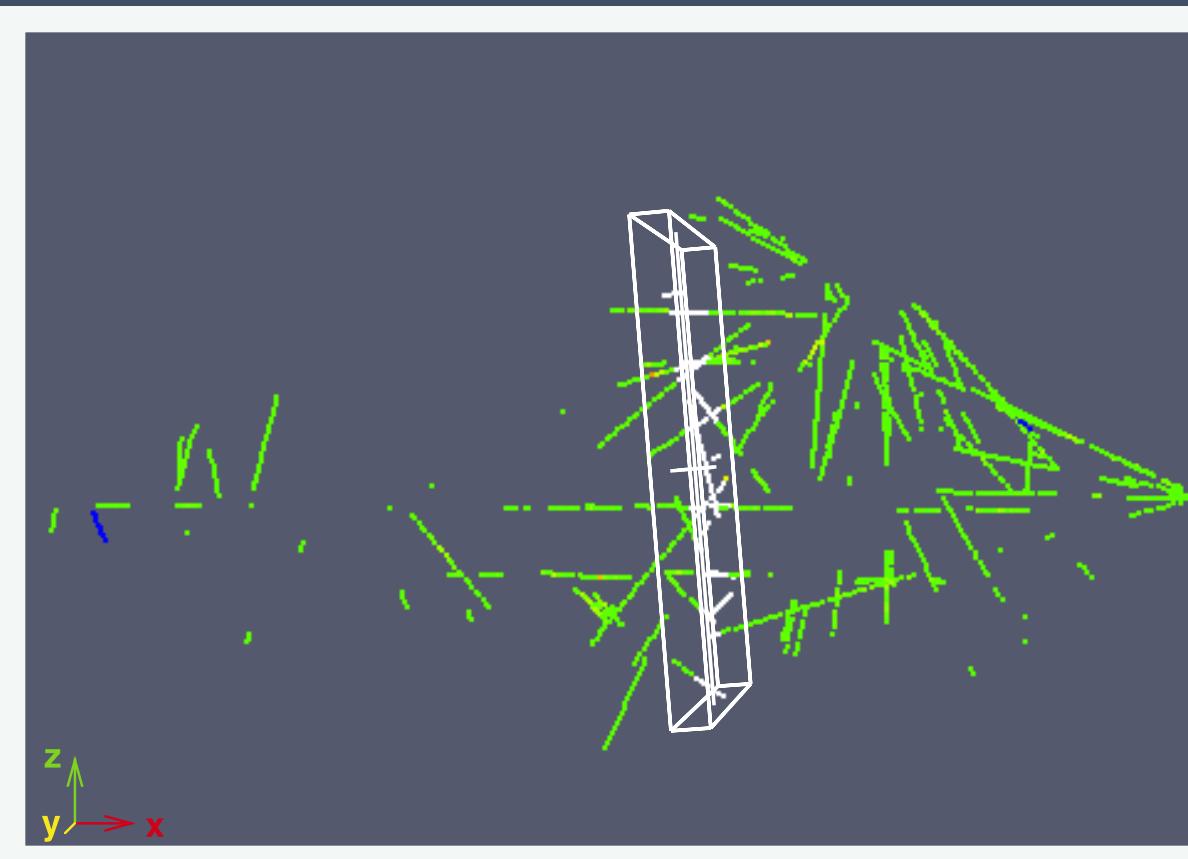
offset = 2  
Landmark grid ( $7 \times 7 \times 7$ )  
Estimation grid ( $30 \times 30 \times 30$ )



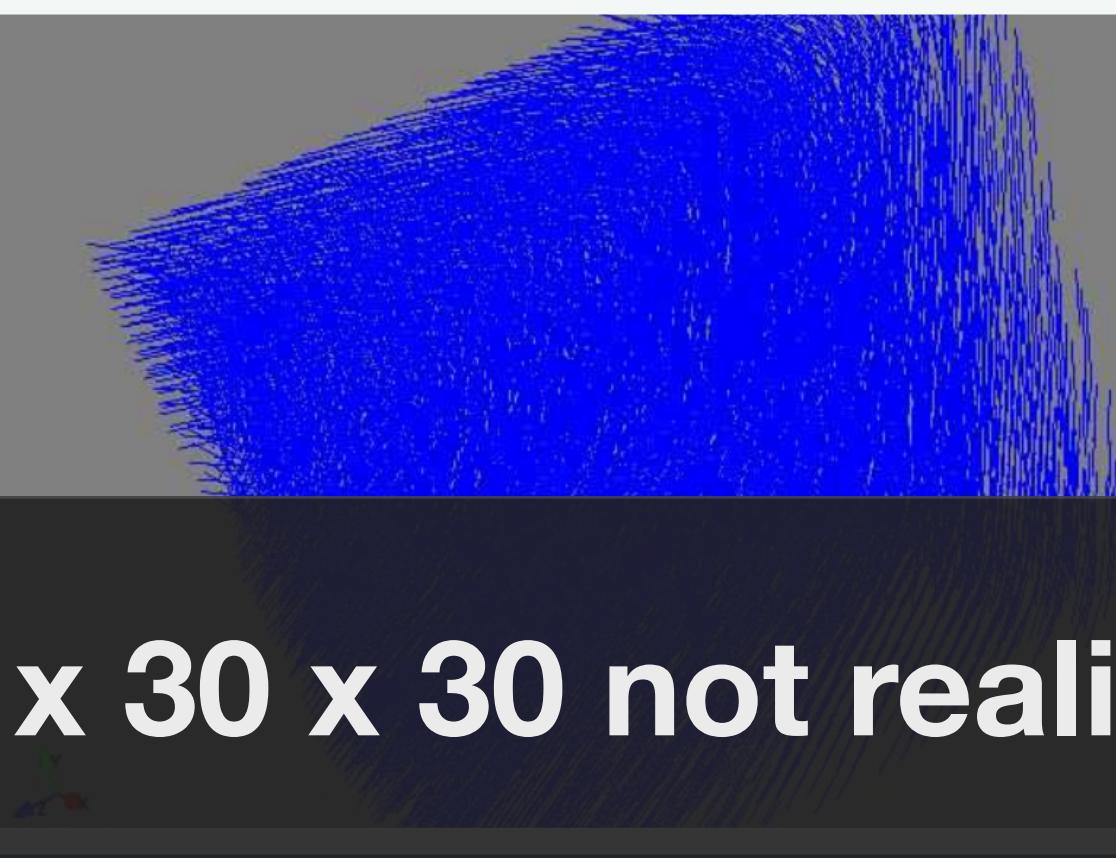
Simple Kriging

**NOTE:** Carlin type deposit containing gold, copper, molybdenum, zinc, iron and lead associated with dolomitic skarns

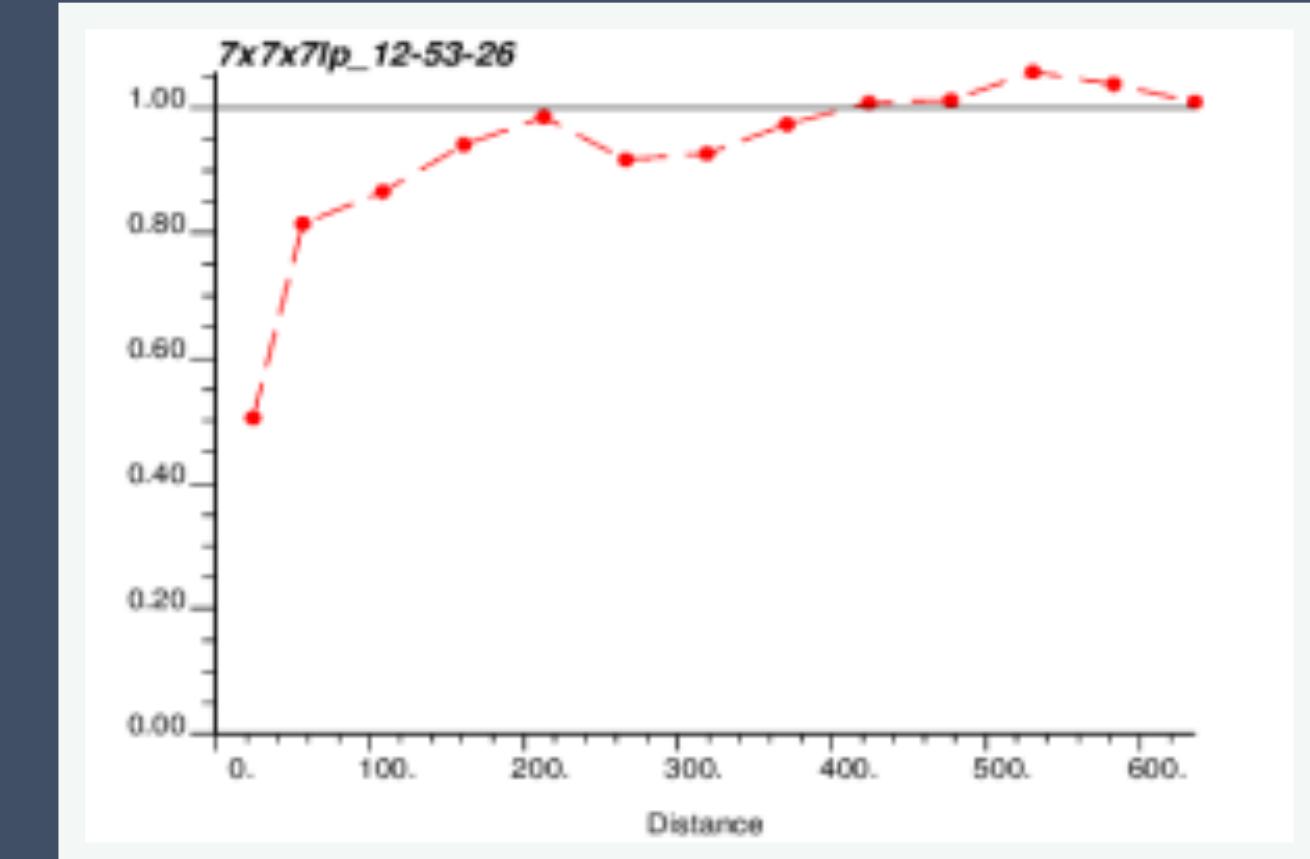
# Case study



Drillholes  
(800 gold grade samples)



**30 x 30 x 30 not realistic. Can we scale to larger grids?**



Semi Variogram

offset = 2

Landmark grid ( $7 \times 7 \times 7$ )

Estimation grid ( $30 \times 30 \times 30$ )

Simple Kriging

Drillholes  
(800 gold grade samples)

DXF lines  
LVA field ( $30 \times 30 \times 30$ )

**NOTE:** Carlin type deposit containing gold, copper, molybdenum, zinc, iron and lead associated with dolomitic skarns

# Case study

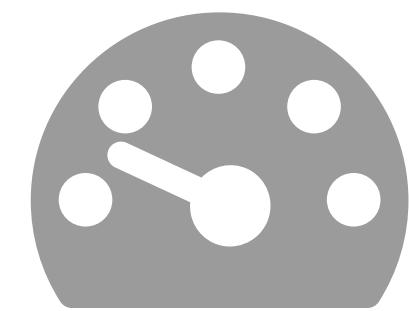


Drillholes  
(800 gold grade samples)

Shortest path calculations  
(Dijkstra)

Sequential

14 GB

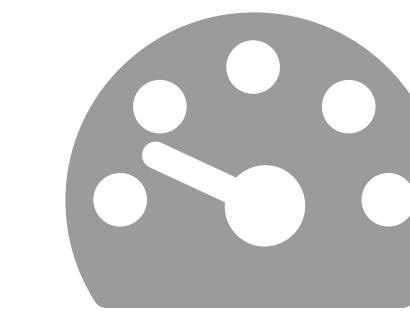


1900 s

$120^3$  grid

Sequential

34 GB

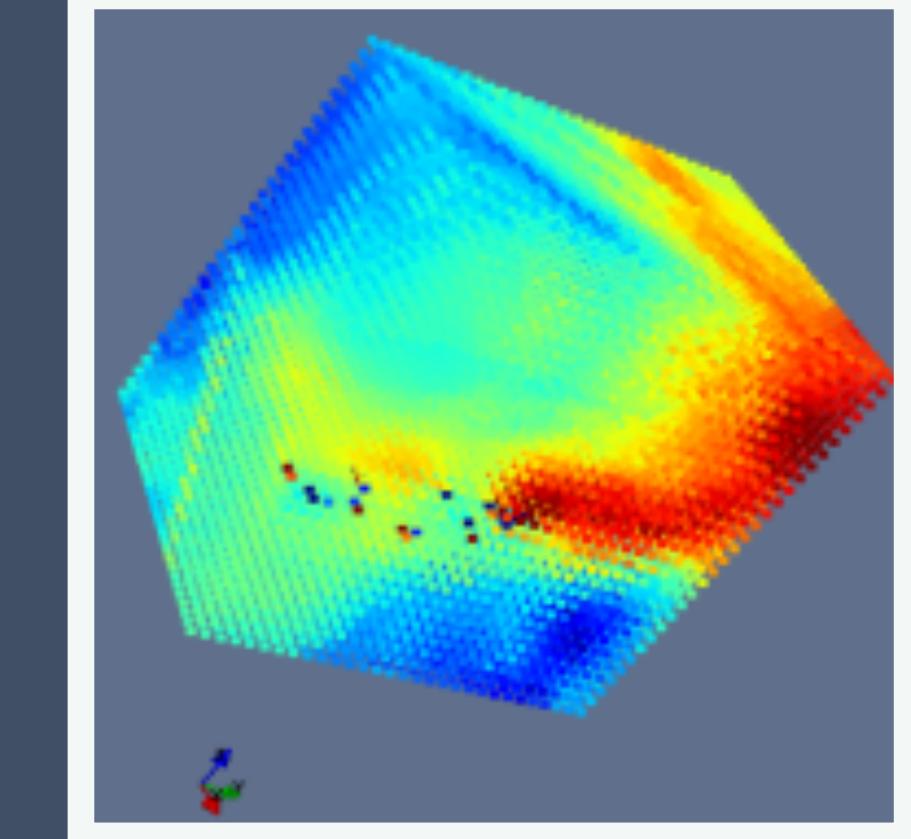


9300 s

$160^3$  grid

\* time in seconds

offset = 2  
Landmark grid ( $7 \times 7 \times 7$ )



Simple Kriging

$\times 7 \times 7)$   
 $0 \times 30 \times 30)$



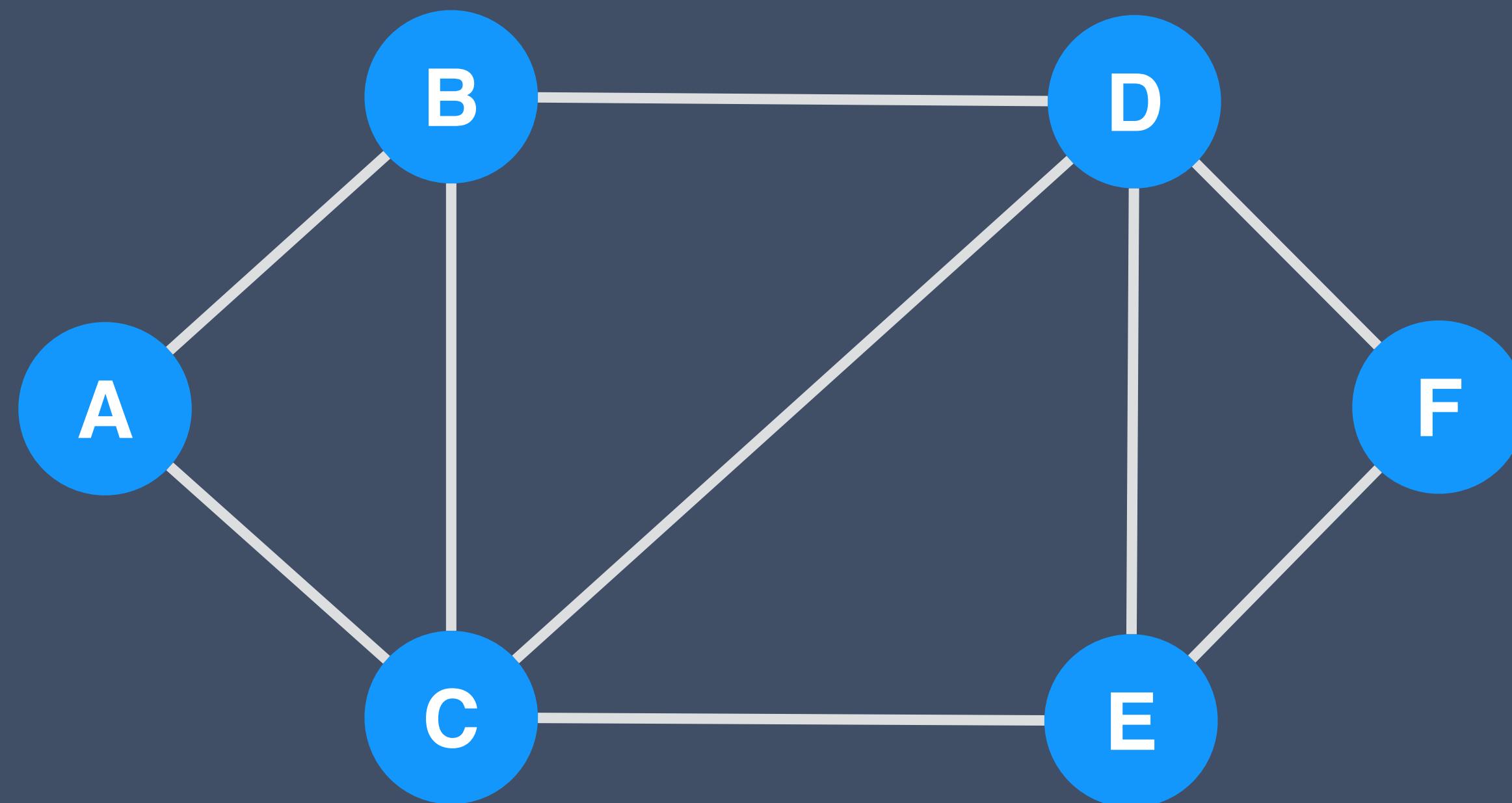
**Dimensionality curse: larger grids require too much RAM**

**NOTE:** Carlin type deposit containing gold, copper, molybdenum, zinc, iron and lead associated with dolomitic skarns



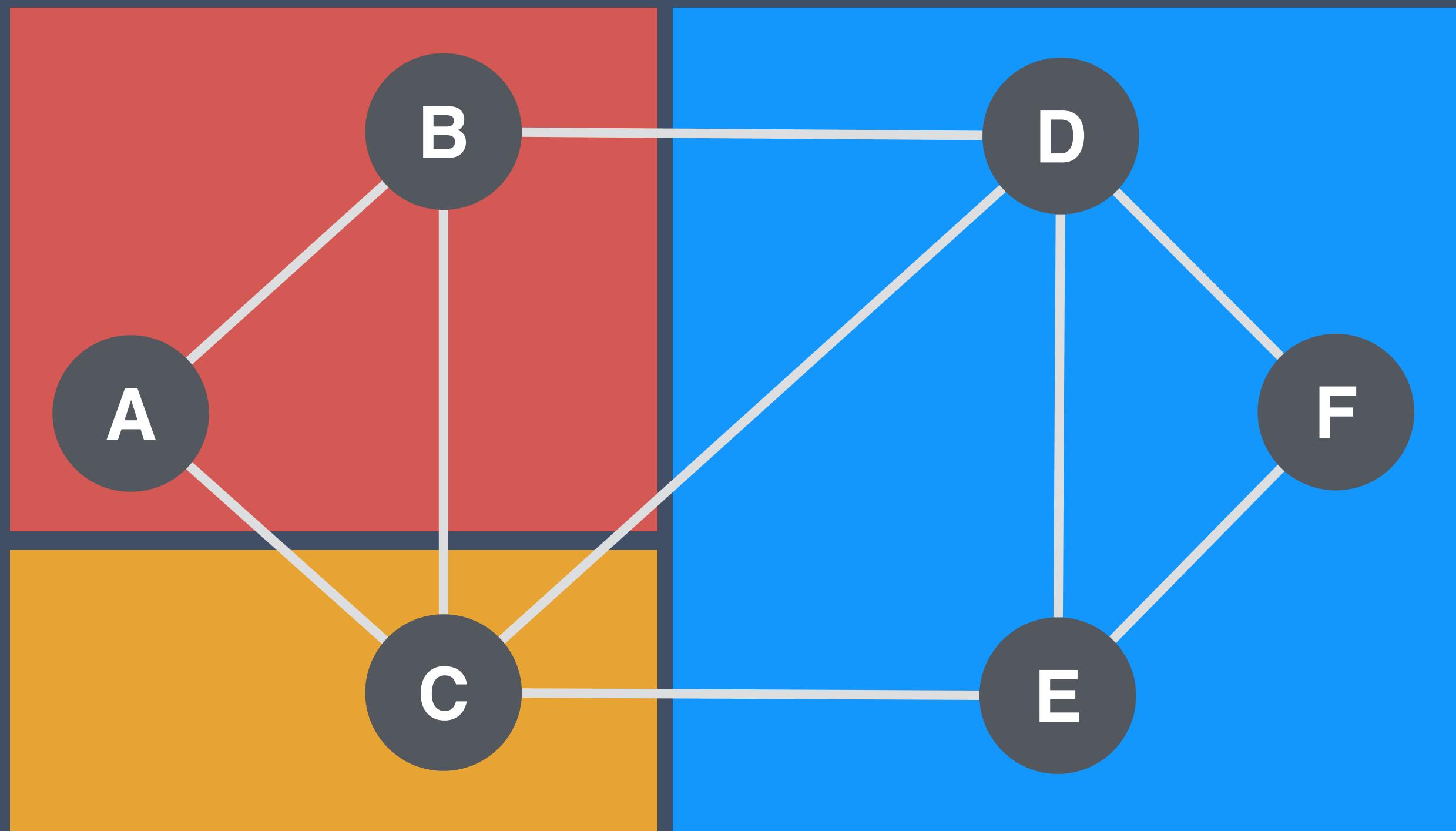
# First Problem: RAM

# Distributed graph



- When working with large estimation grids **huge** graphs must be stored in memory
- LVA methods generates *undirected graphs*
- A grid of  $120 \times 120 \times 120$  uses more than 41GB (offset = 3)

# Distributed graph

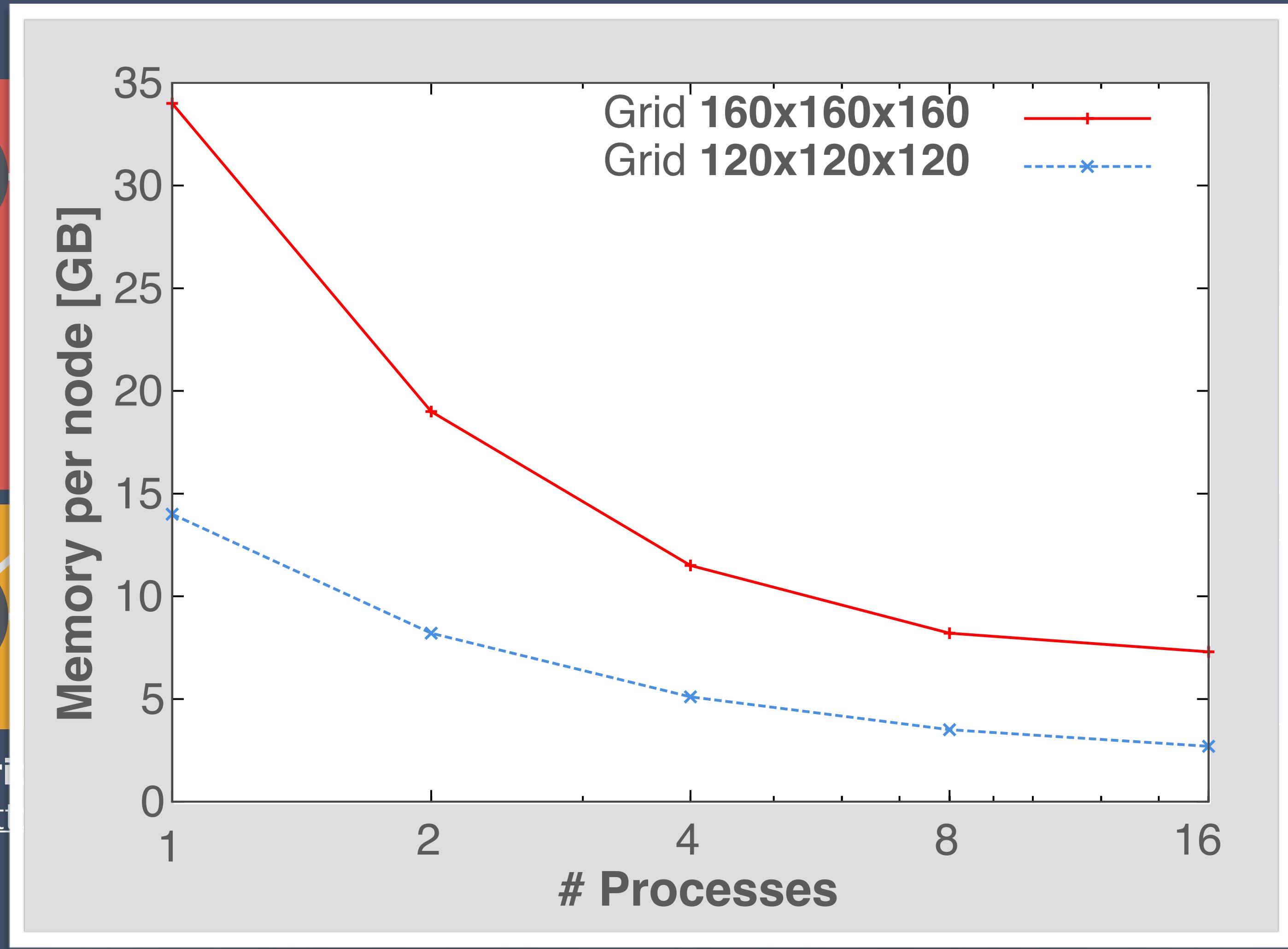
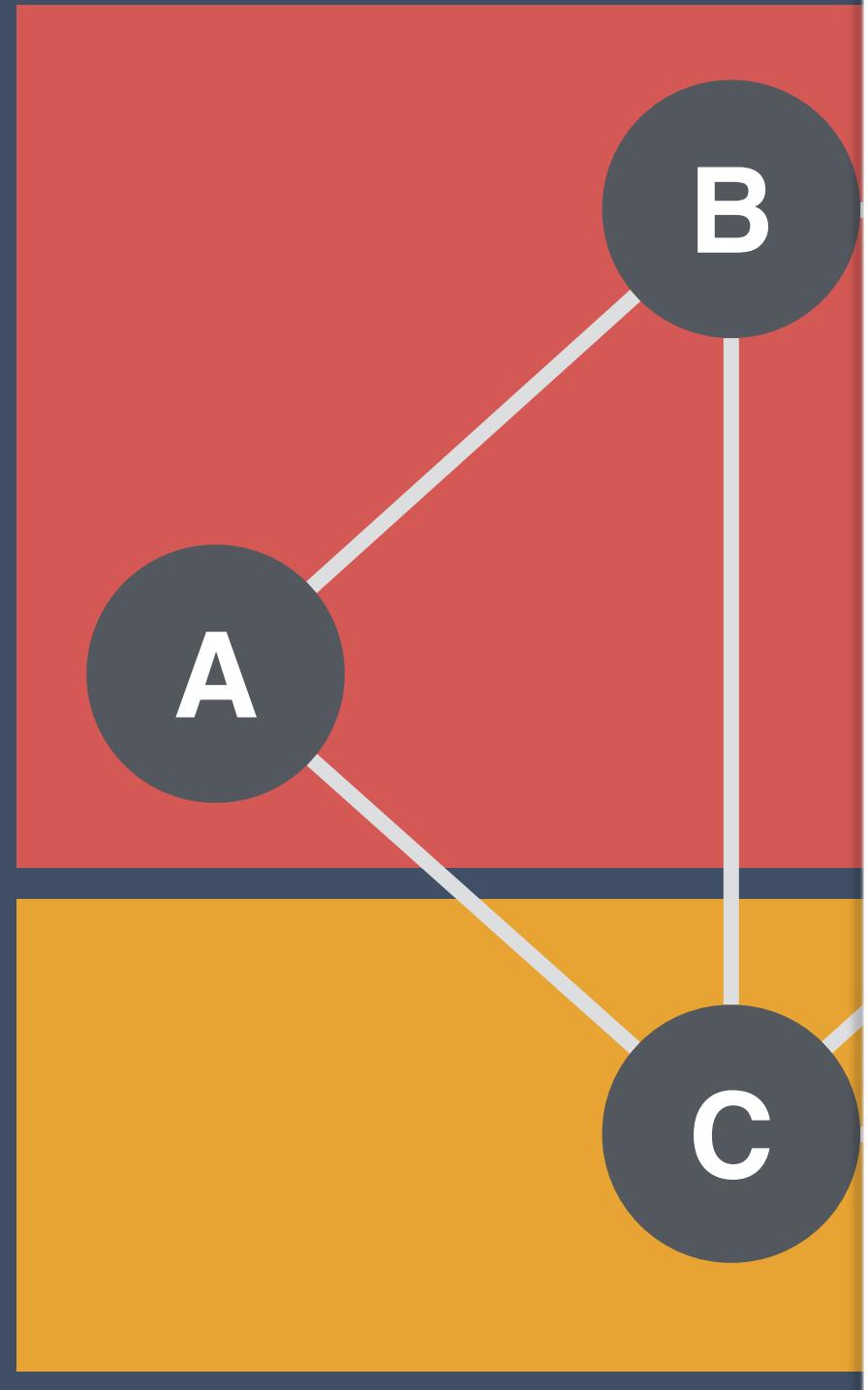


processor 0	vert: A, B edges: 4
processor 1	vert: C edges: 2
processor 2	vert: D, F, E edges: 3

Distribution using *Parallel Boost Graph Library*

<http://www.boost.org/libs/graph/doc/index.html>

# Distributed graph



vert: A, B  
edges: 4

vert: C  
edges: 2

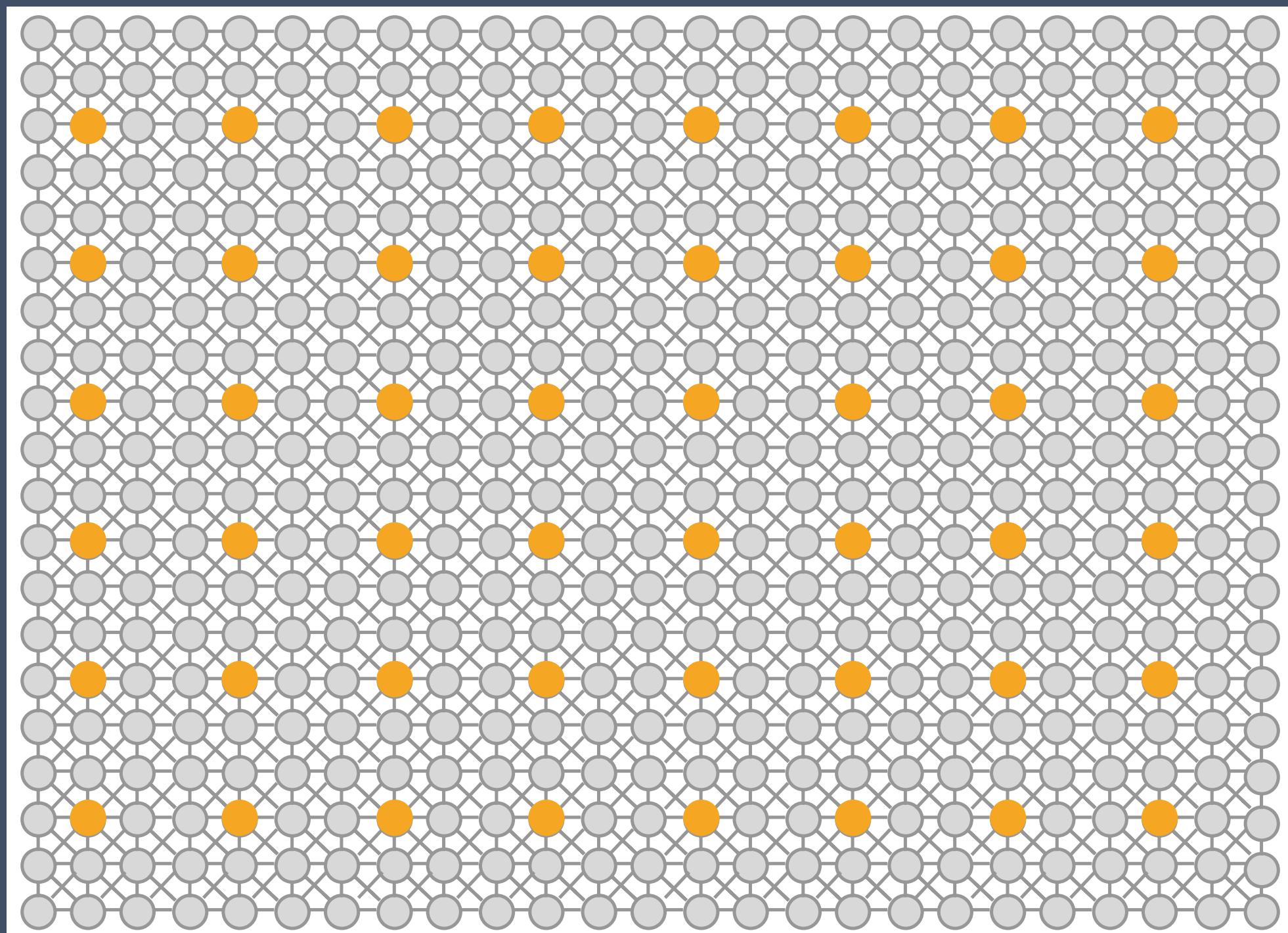
vert: D, F, E  
edges: 3



## Second problem: CPU

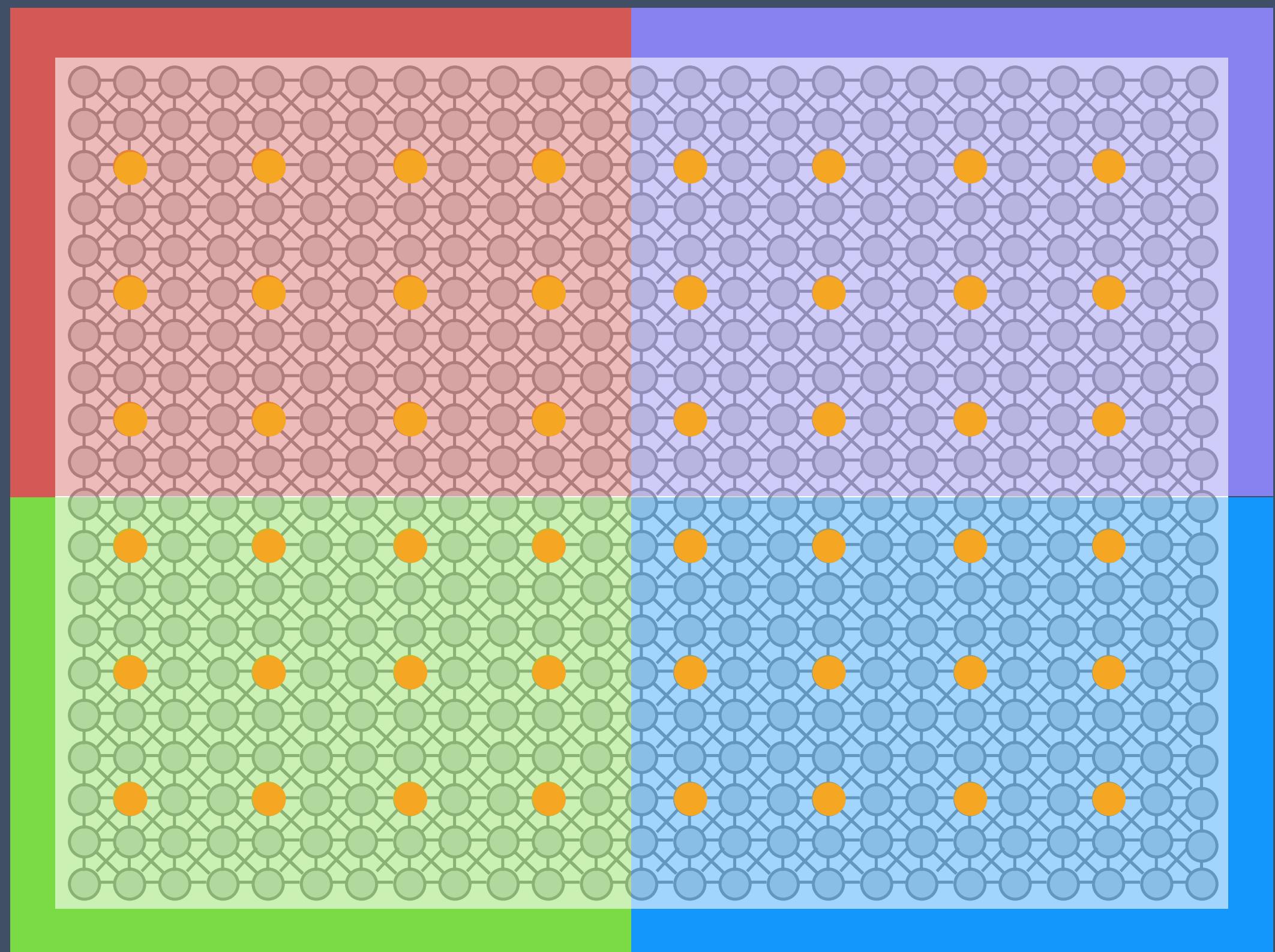


# Parallel Dijkstra runs



- For each landmark point, we must run a single source shortest path calculation
- Every calculation can be executed in parallel by different processes
- Each run can use the distributed graph explained before

# Parallel Dijkstra runs



set 0: 12 Dijkstra runs

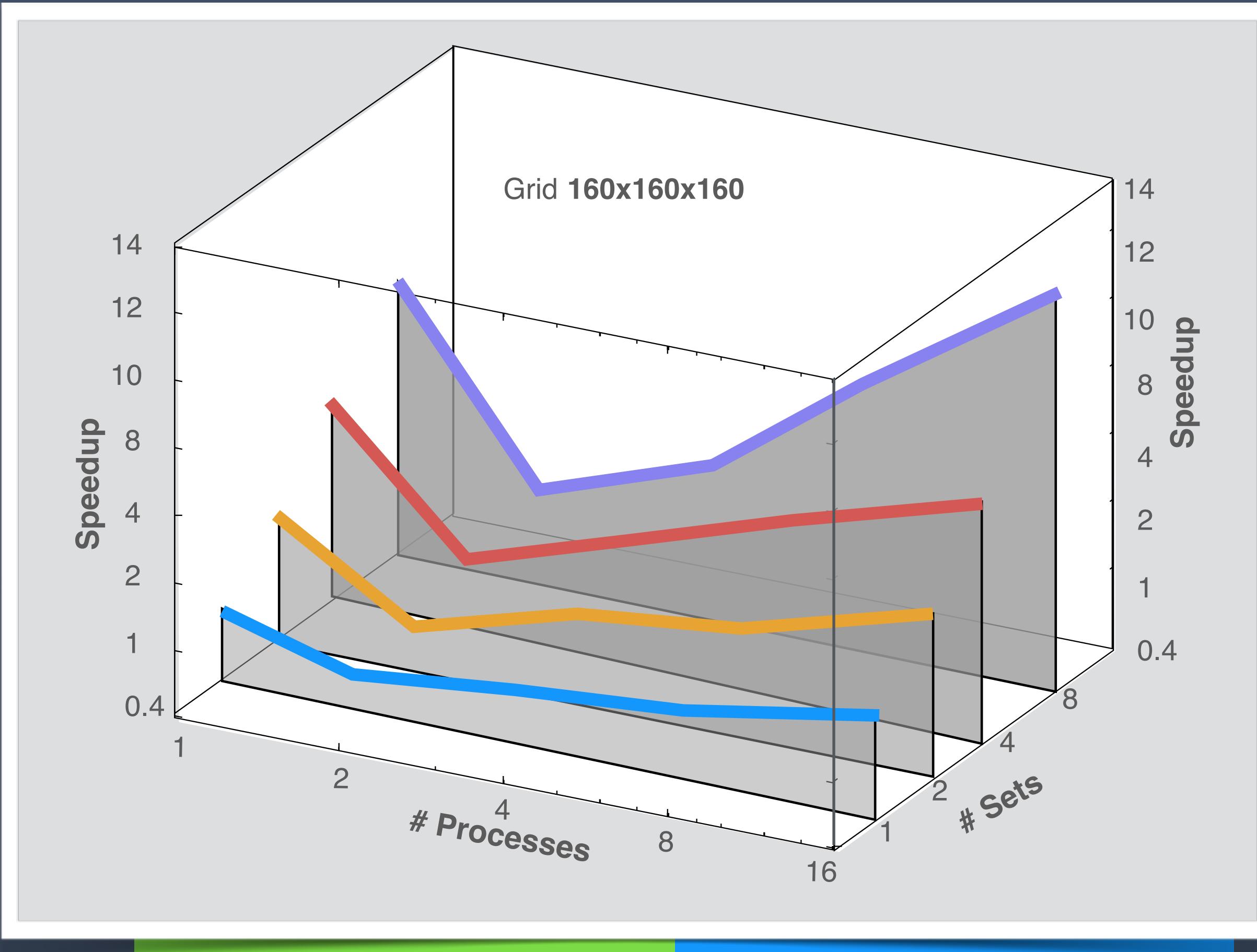
set 1: 12 Dijkstra runs

set 2: 12 Dijkstra runs

set 3: 12 Dijkstra runs

Distribution using *Parallel Boost Graph Library + Parallel MPI Executions*

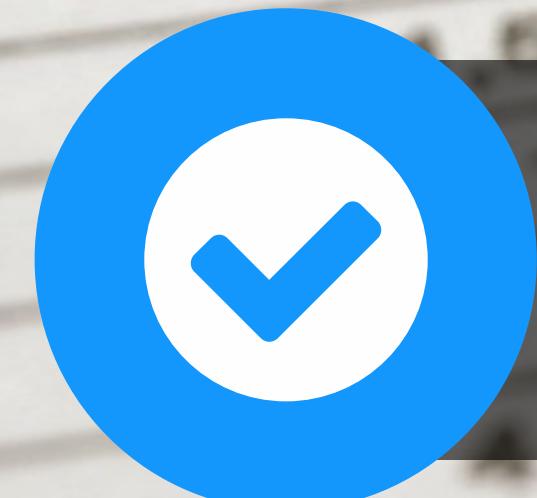
# Parallel Dijkstra runs



#Processes	1	2	4	8	16		
# Sets	1	<b>1x</b>	<b>0.44x</b>	<b>0.67x</b>	<b>1.07x</b>	<b>1.49x</b>	
1	1	<b>2x</b>	<b>0.88x</b>	<b>1.34x</b>	<b>2.13x</b>	<b>2.97x</b>	
2	2	1	<b>4x</b>	<b>1.73x</b>	<b>2.68x</b>	<b>4.27x</b>	<b>5.96x</b>
4	4	2	1.73	2.68	4.27	5.96	
8	8	7.5	3.29	5.03	8.04	11.19	

Distribution using *Parallel Boost Graph Library + Parallel MPI Executions*

Grid 160x160x160

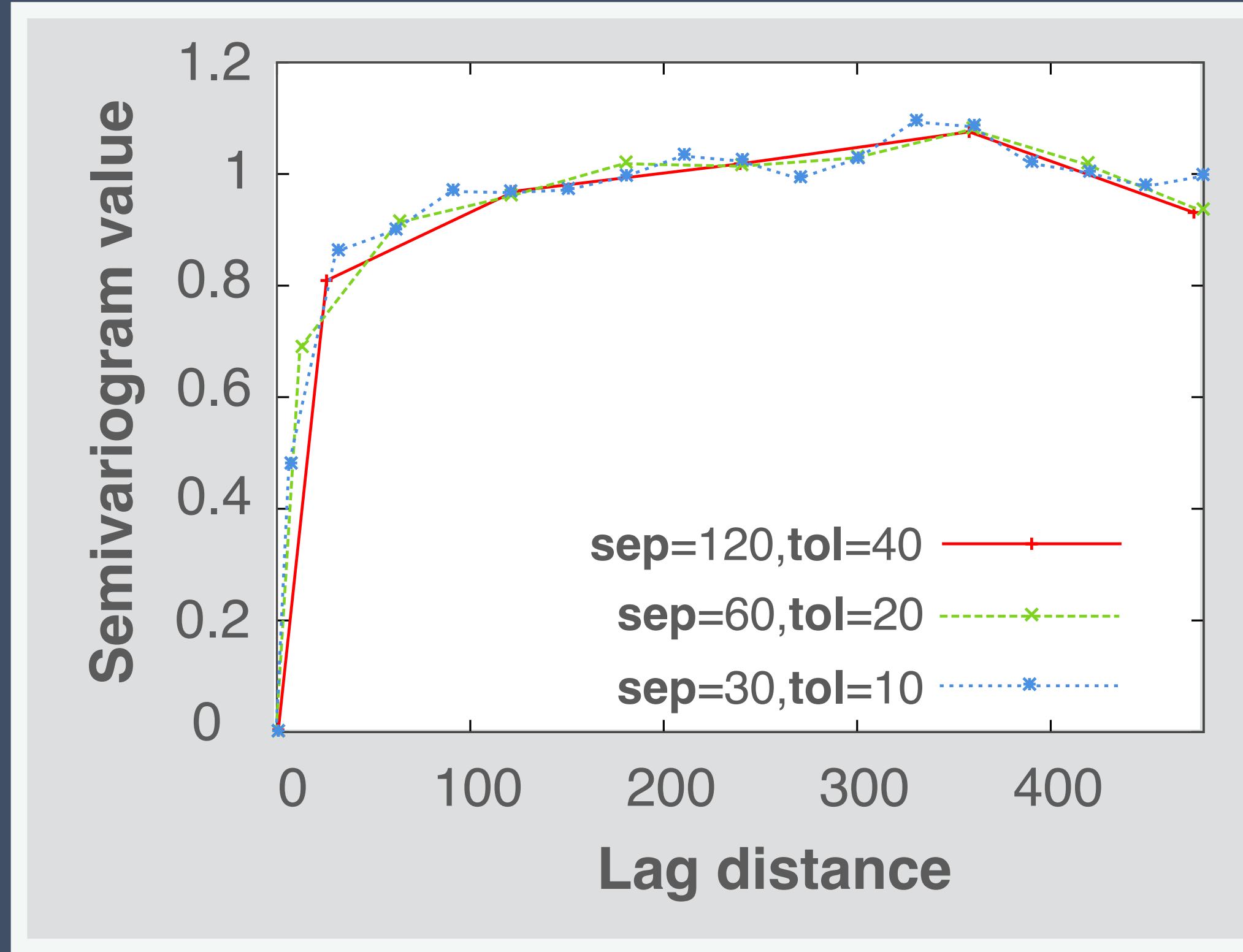


# Results



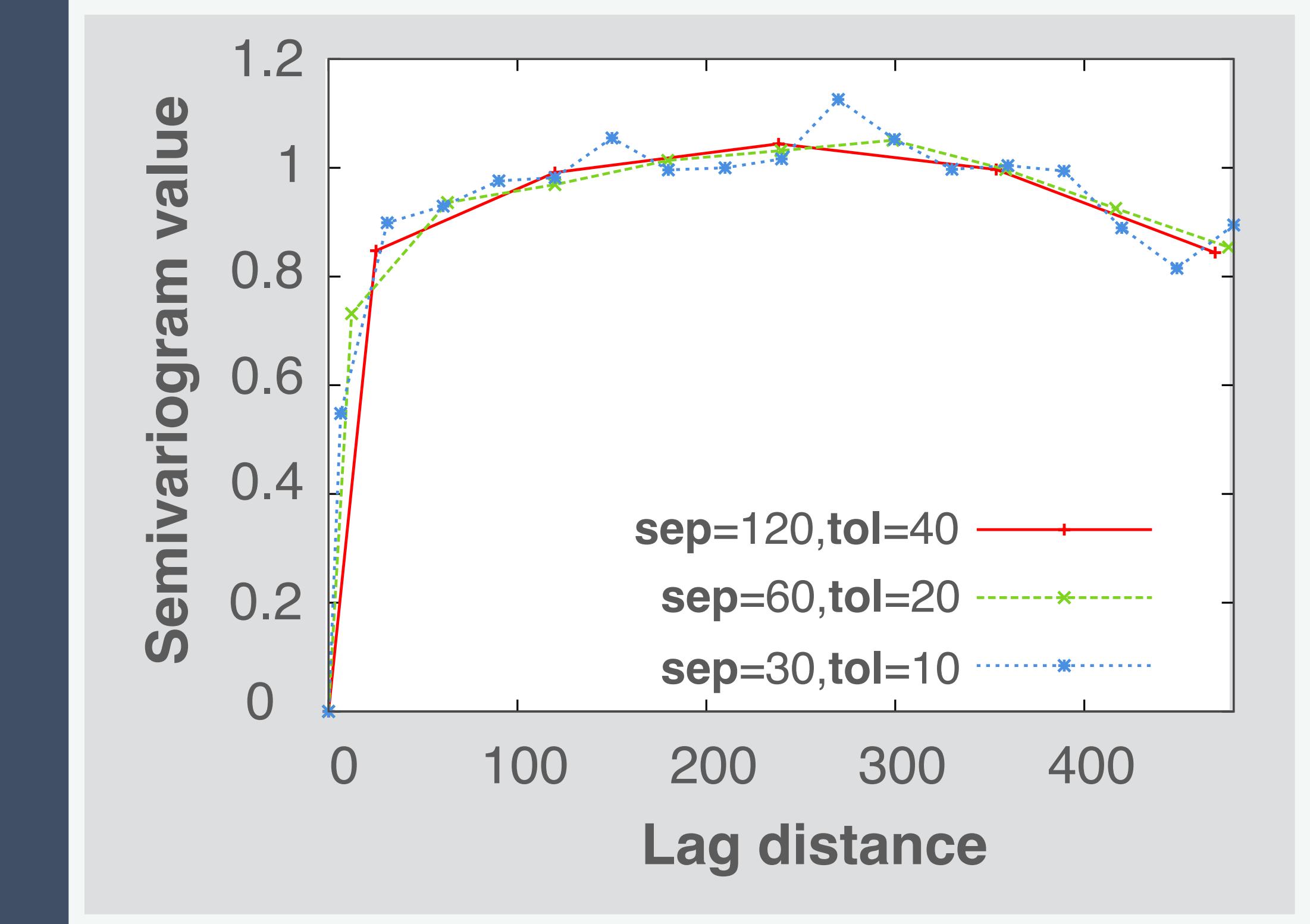
1,443.03	6,517.11	3,978.59	6,066.89	2,51	3,456.31
4,887.11	2,960.78	4,537.89	4,394.91	9,389.11	9,389.11
1,880.78	1,680.53	210.78	1,889.93	2,804.03	2,804.03
1,680.54	4,248.81	3,130.91	3,590.35	1,239.03	1,239.03
6,889.89	1,558.04	6,890.43	4,555.89	1,902.3	1,902.3
1,563.04	6,118.81	4,835.95	3,896.79	7,890.00	7,890.00
6,993.83	3,808.88	2,884.99	2,565.39	8,700.78	8,700.78
908.33	4,585.14	3,930.22	3,425.89	1,781.97	1,781.97
4,685.96	8,118.38	623.09	3,210.78	8,302.04	8,302.04
3,918.88	2,884.54	3,478.31	2,565.39	8,822.11	8,822.11
734.69	8,121.84	459.93	5,108.03	2,981.98	2,981.98
3,929.02	5,810.18	989.09	9,920.88	9,090.28	9,090.28
5,890.93	3,985.47	8,392.71	5,091.99	1,883.03	1,883.03
1,781.07	4,818.78	9,930.77	5,000.21	3,107.00	3,107.00
4,318.73	4,885.18	3,091.99	8,398.91	898.31	898.31
4,885.93	571.01	946.18	3,890.00	9,934.93	9,934.93
671.09	8,410.81	3,110.91	1,881.93	3,470.37	3,470.37
3,490.31	3,184.08	3,630.90	811.46	2,430.00	2,430.00
2,984.03	6,904.51	7,890.83	8,437.04	6,941.77	6,941.77
5,744.69	4,340.44	8,557.97	955.48	6,441.38	6,441.38
4,230.00	5,189.84	9,738.95	1,890.99	3,881.03	3,881.03
1,252.34	1,811.11	7,093.09	6,450.38	3,310.41	3,310.41
499.11	7,410.18	879.93	3,450.38	8,867.80	8,867.80
7,890.93	1,571.47	3,989.08	3,310.41	2,319.87	2,319.87
1,781.27	6,810.18	9,279.03	2,128.25	2,128.25	2,128.25
6,890.93	8,322.81	3,909.88	6,971.99	6,971.99	6,971.99
8,912.88	6,581.33	3,772.21	5,779.44	5,779.44	5,779.44
1,766.37	1,106.66	8,100.20	6,407.00	1,370.37	1,370.37
2,981.28	4,814.39	2,319.83	3,371.93	1,370.37	1,370.37
2,106.66	4,814.39	8,371.93	6,407.00	1,370.37	1,370.37
3,106.66	4,814.39	3,371.93	3,371.93	1,370.37	1,370.37

# Results



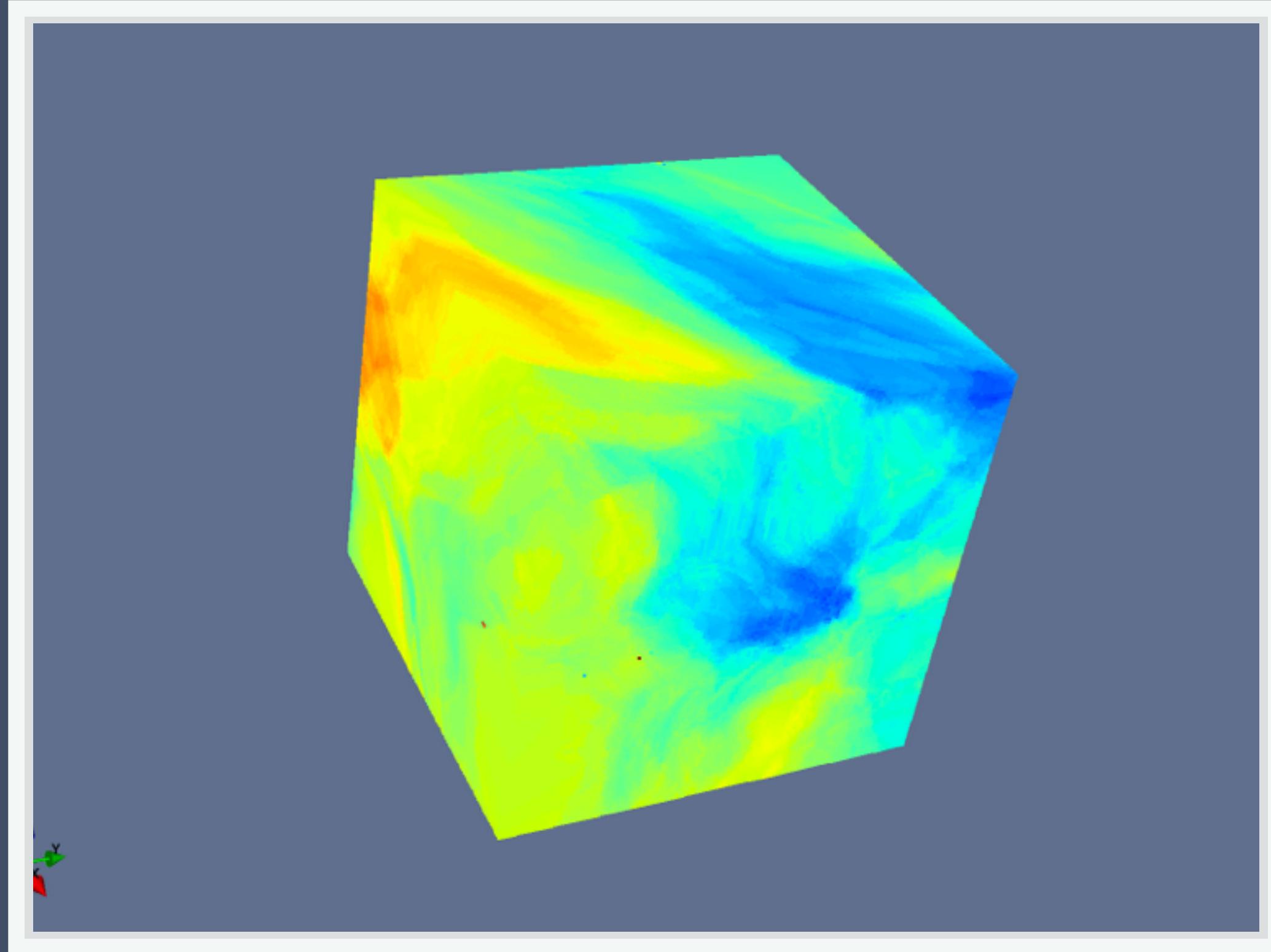
Estimation grid of  
**1,728,000** nodes  
( $120 \times 120 \times 120$ )

**Semi Variogram**  
offset = 2  
landmark = 7x7x7



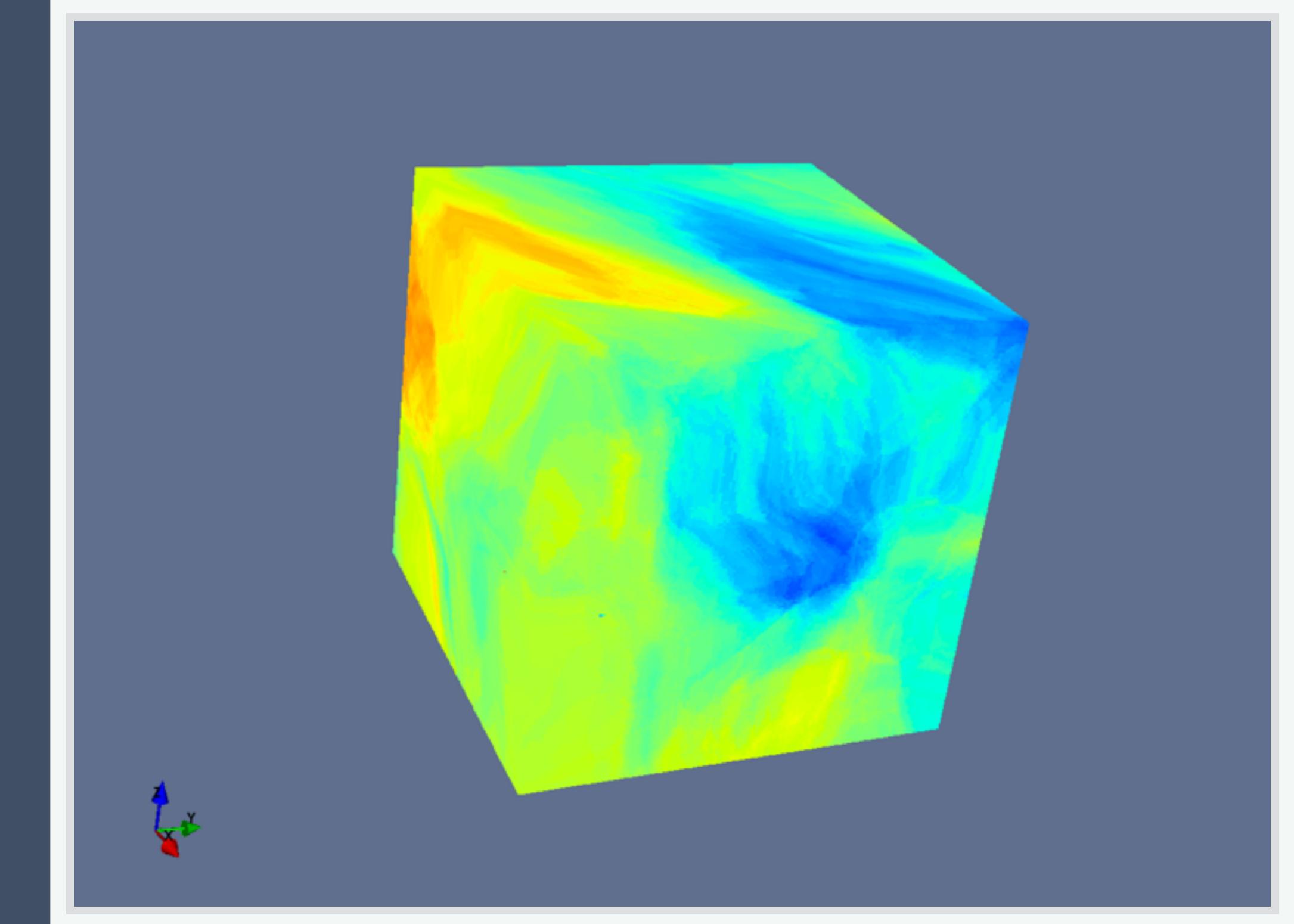
Estimation grid of  
**4,096,000** nodes  
( $160 \times 160 \times 160$ )

# Results



Estimation grid of  
**1,728,000** nodes  
( $120 \times 120 \times 120$ )

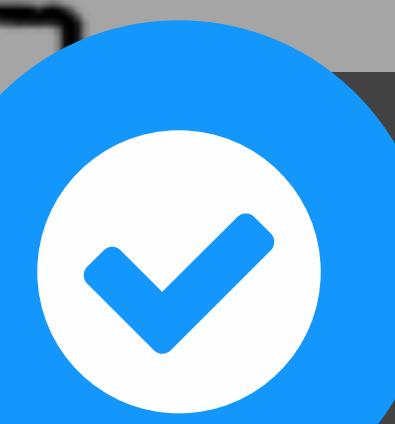
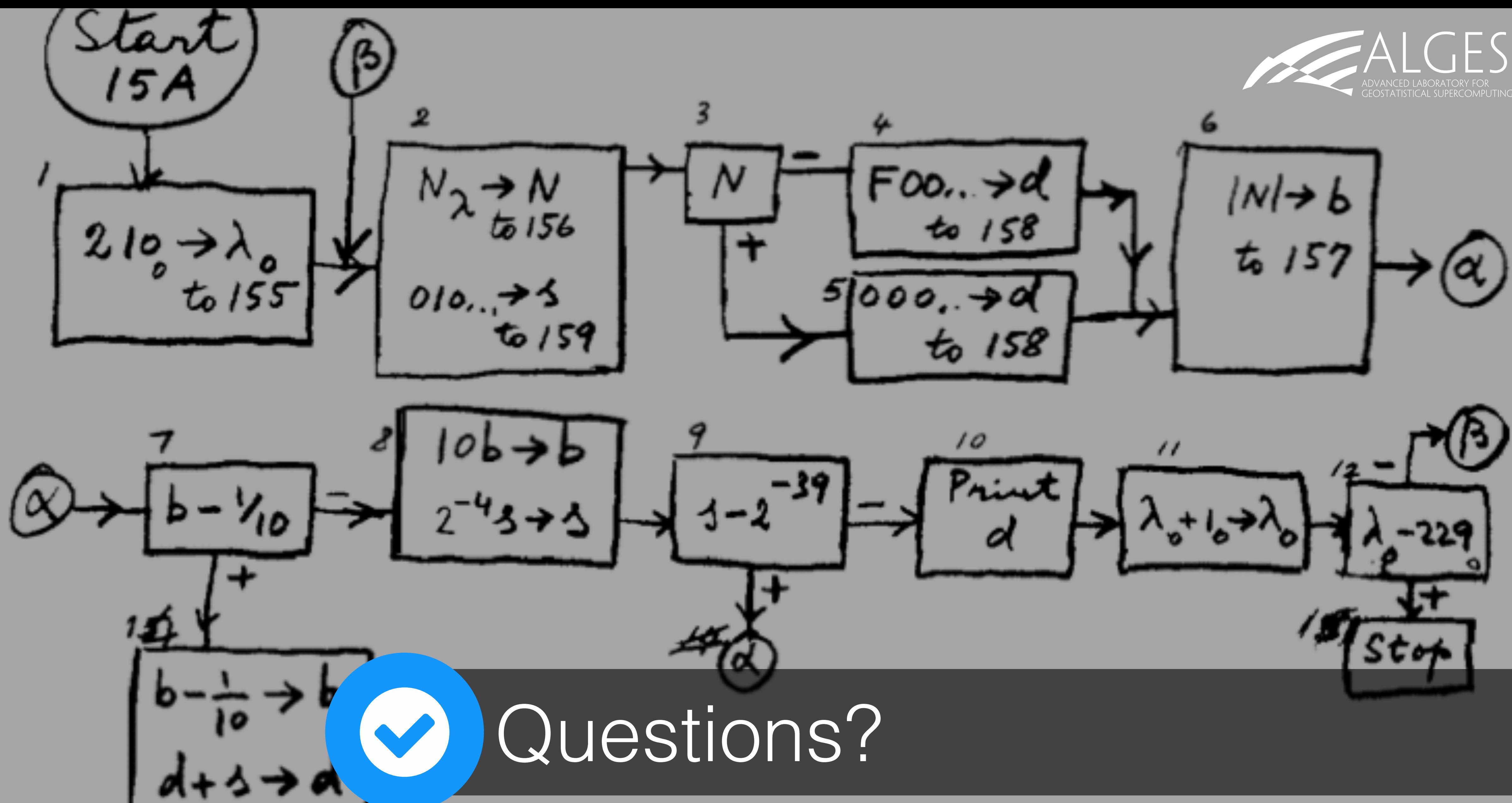
**Simple Kriging**  
offset = 2  
landmark = 7x7x7



Estimation grid of  
**4,096,000** nodes  
( $160 \times 160 \times 160$ )

# Conclusions

- The scope and possibilities that the parallel approach can bring in the LVA context are related with the size of the problems to be solved
- Due to the possibility to distribute the memory usage among many computational nodes, very large graphs can be stored in memory in order to be processed by all workers
- As a future work, a new version of the original LVA codes will be developed, enabling the parallel processing capabilities of the current hardware



# Questions?

# Incorporating distributed Dijkstra's algorithm into variogram calculation with locally varying anisotropy

Oscar Peredo, Felipe Navarro, Mauricio Garrido and Julián M. Ortiz

Department of Mining Engineering – FCFM – Universidad de Chile  
Advanced Mining Technology Center – Universidad de Chile

[jortiz@ing.uchile.cl](mailto:jortiz@ing.uchile.cl)

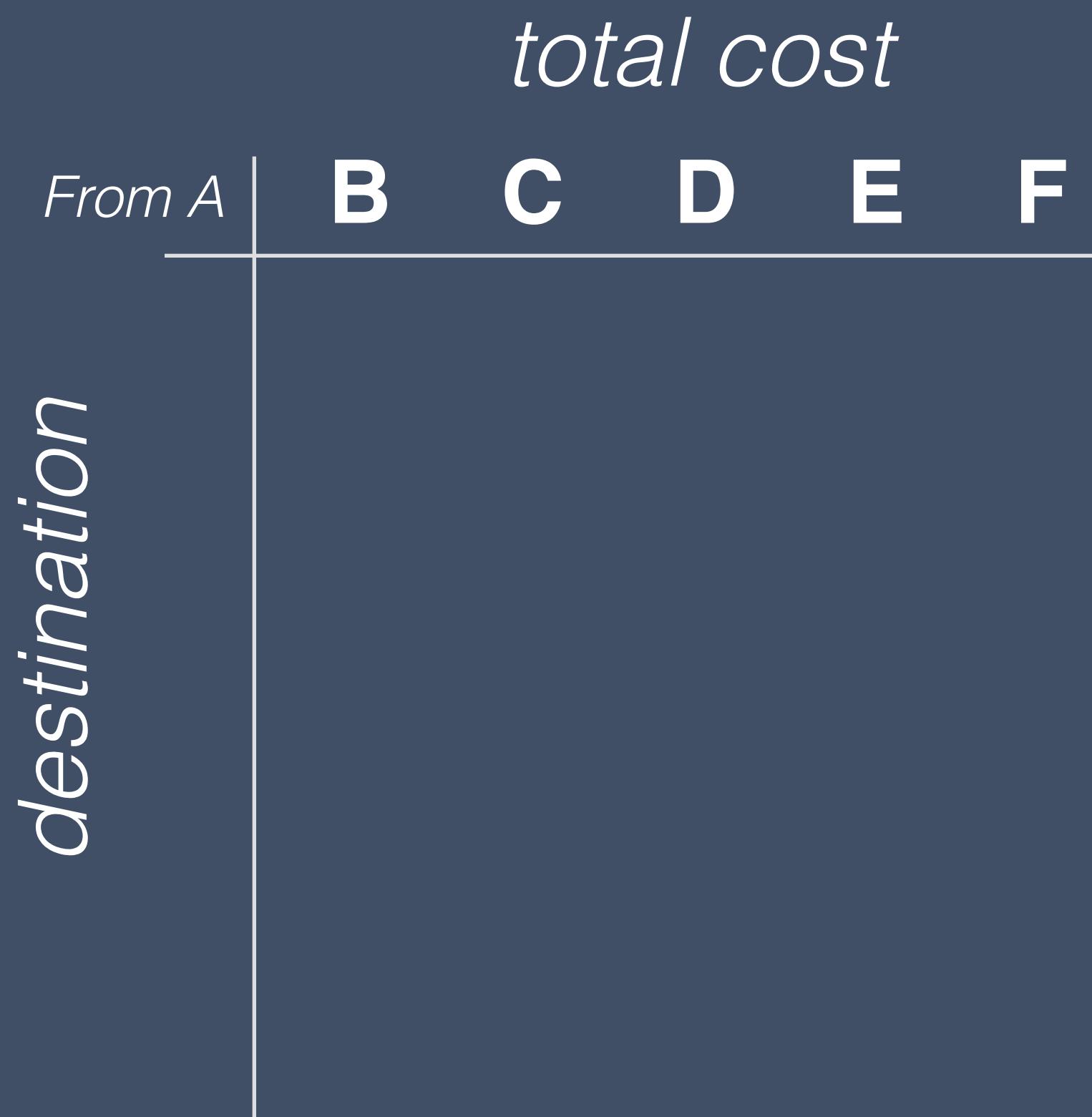
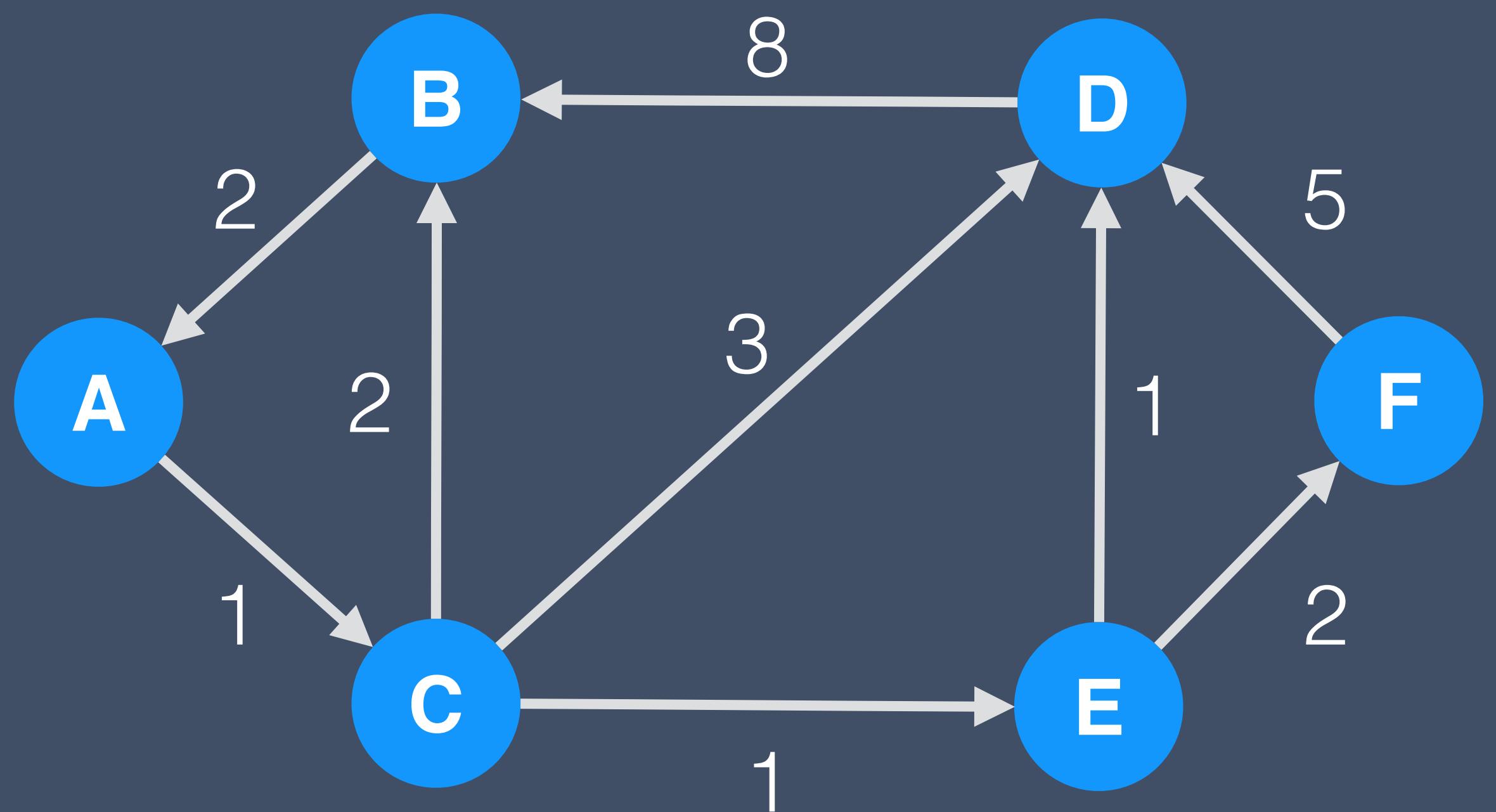
**37th APCOM Conference**  
**Fairbanks, Alaska 2015**

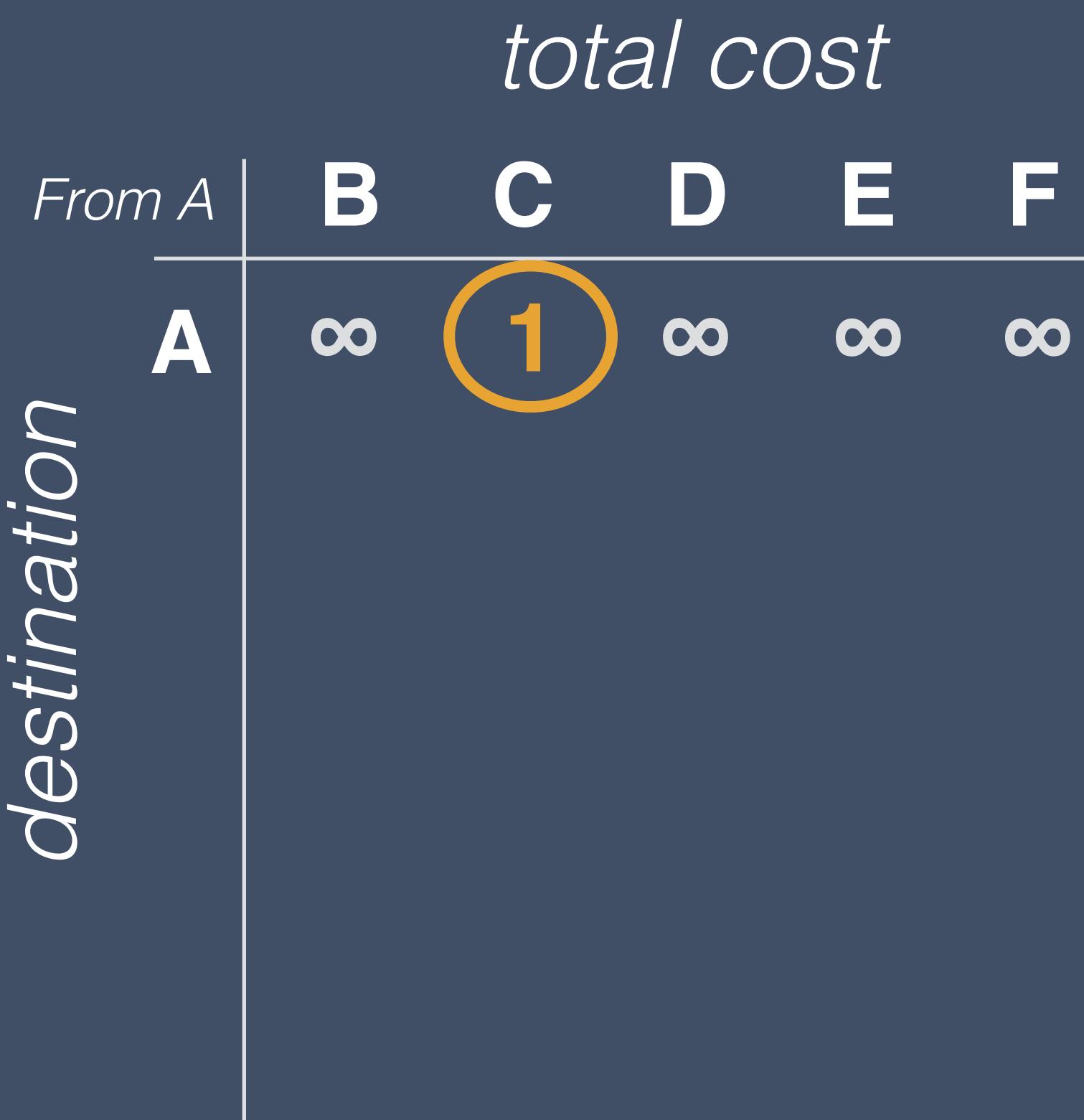
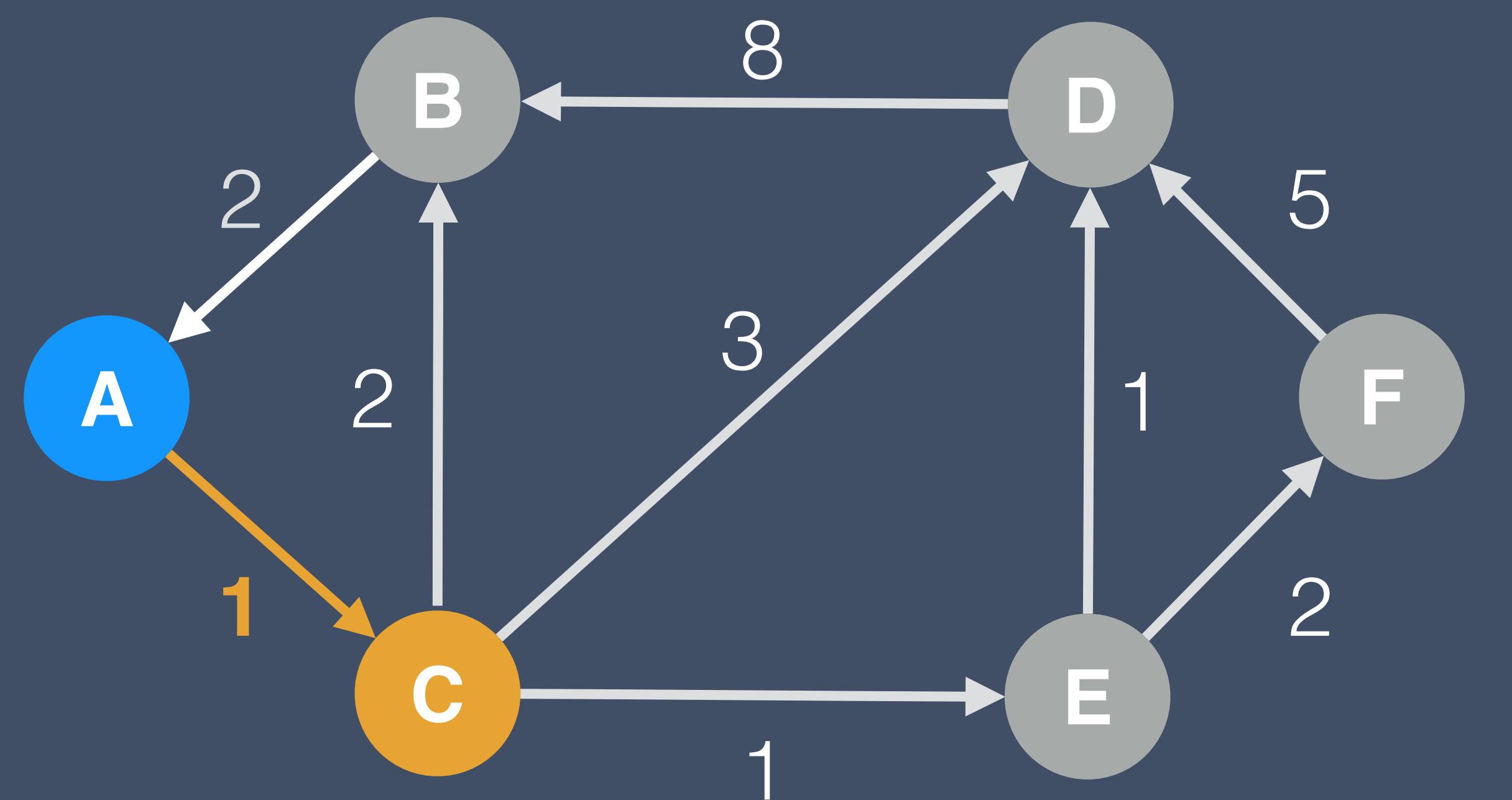


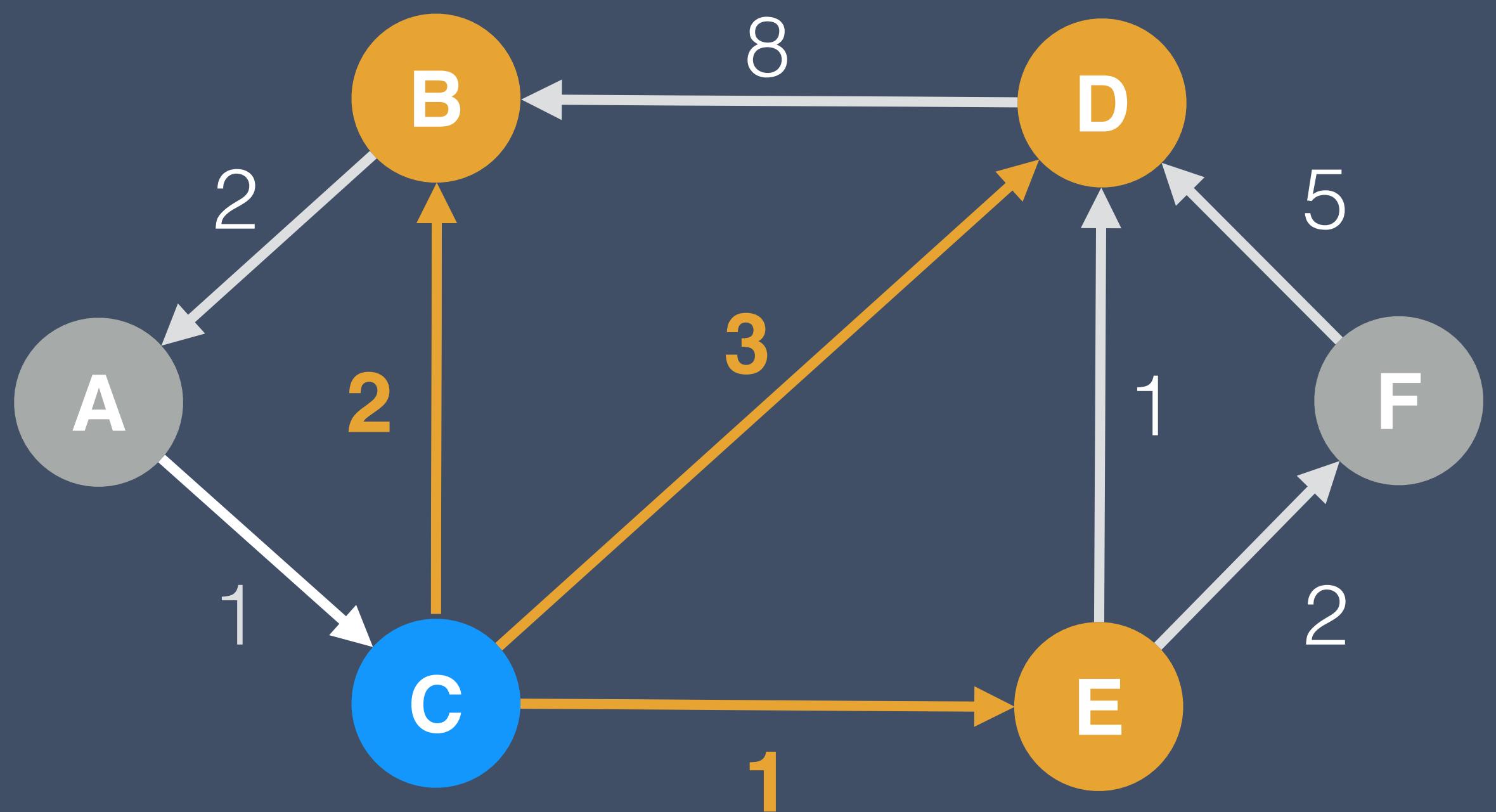
Ingeniería de Minas  
FACULTAD DE CIENCIAS  
FÍSICAS Y MATEMÁTICAS  
UNIVERSIDAD DE CHILE

**amtc**  
ADVANCED MINING TECHNOLOGY CENTER

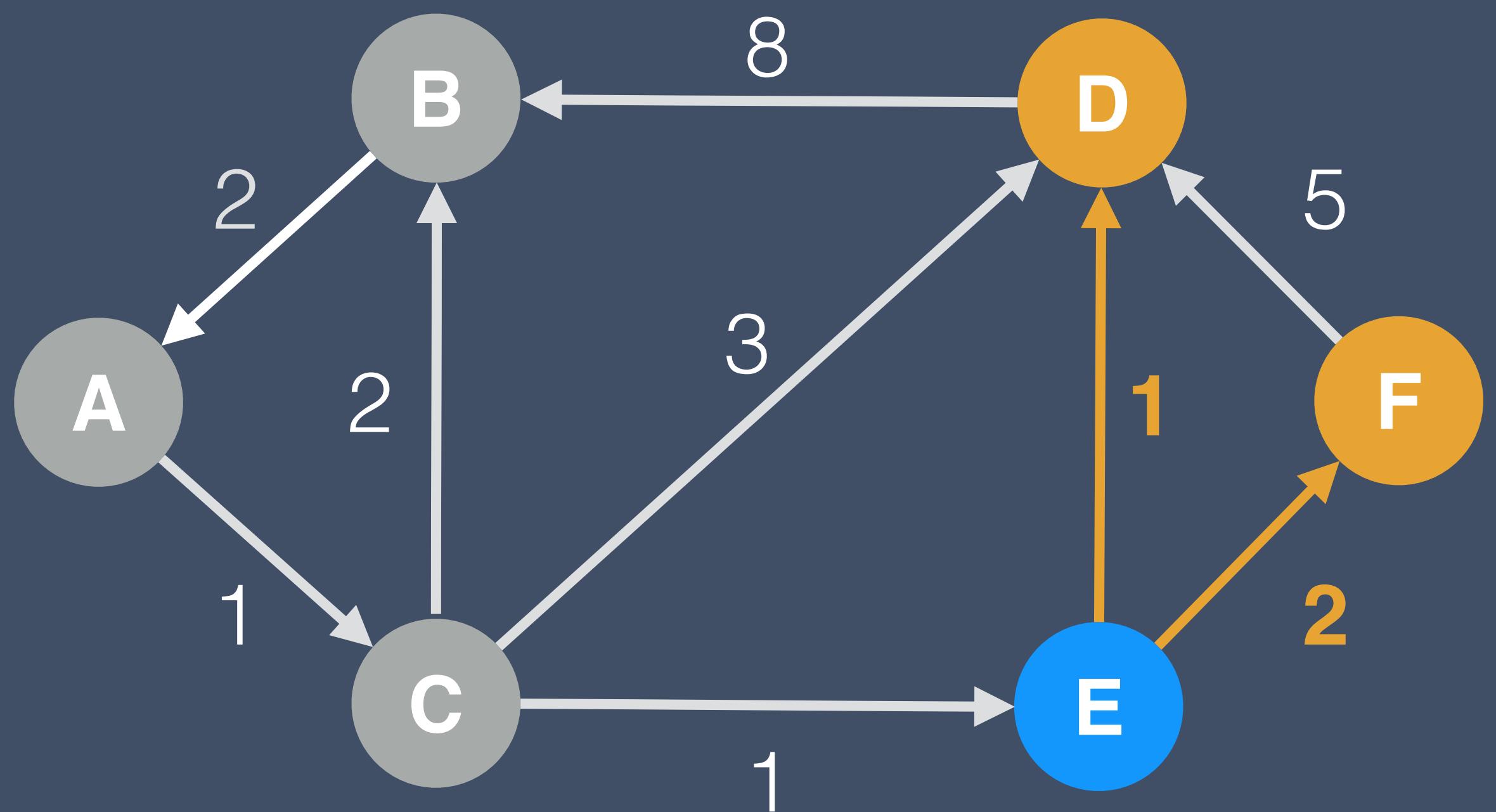
# Dijkstra calculation



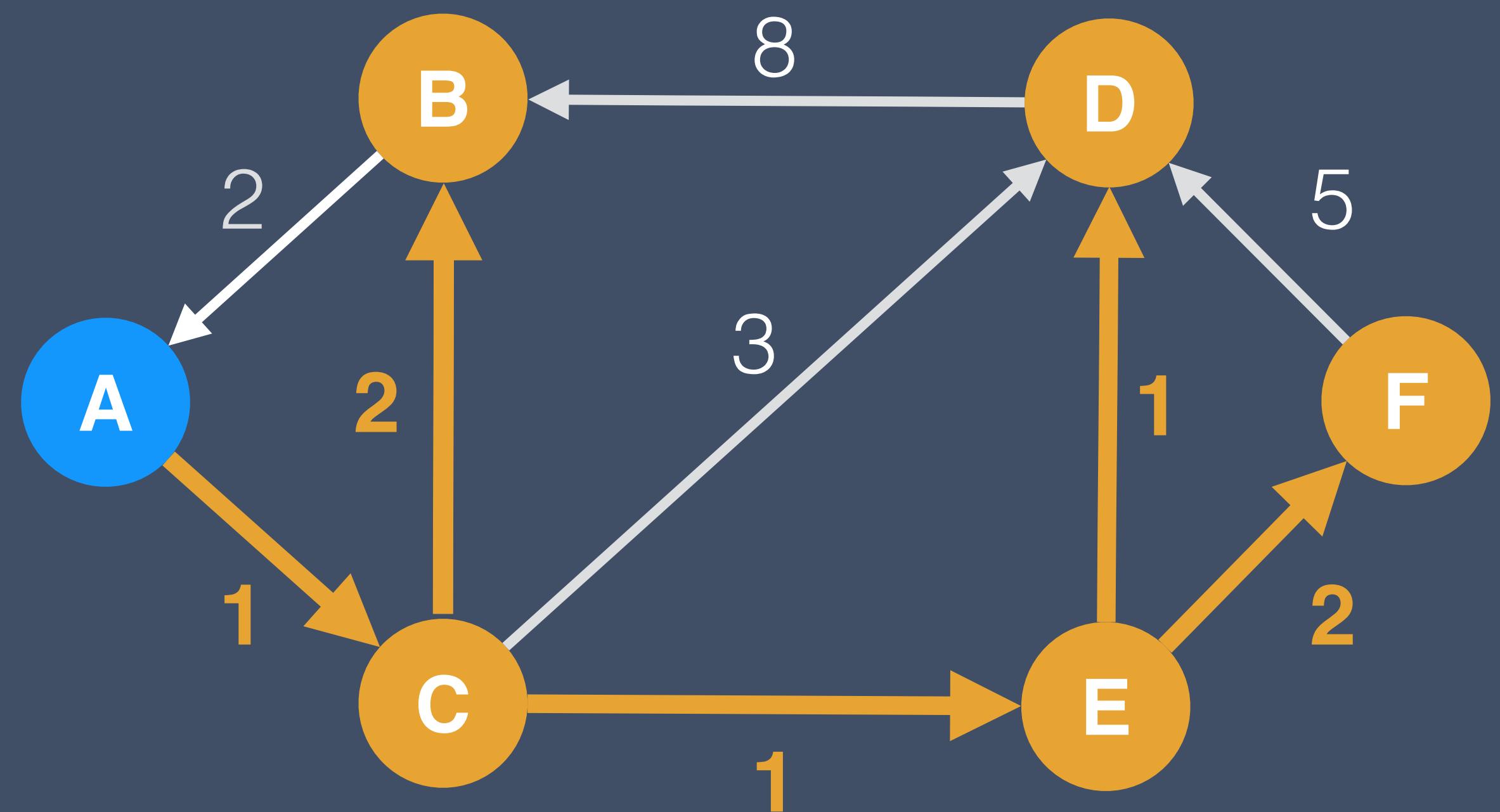




	total cost				
From A	B	C	D	E	F
A	$\infty$	1	$\infty$	$\infty$	8
C	3	1	4	2	8



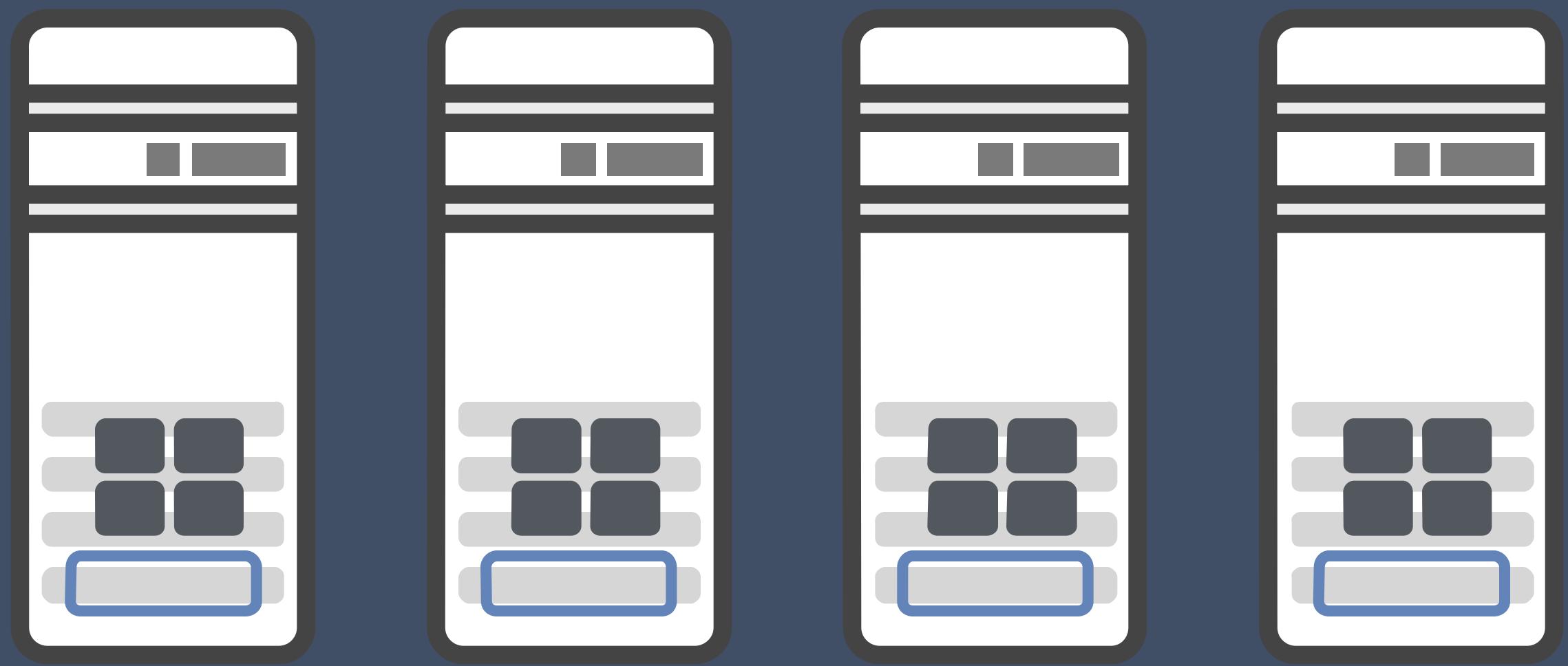
	total cost				
From A	B	C	D	E	F
A	$\infty$	1	$\infty$	$\infty$	$\infty$
C	3	1	4	2	$\infty$
E	3	1	3	2	4



	total cost				
From A	B	C	D	E	F
A	$\infty$	1	$\infty$	$\infty$	$\infty$
C	3	1	4	2	$\infty$
E	3	1	3	2	4

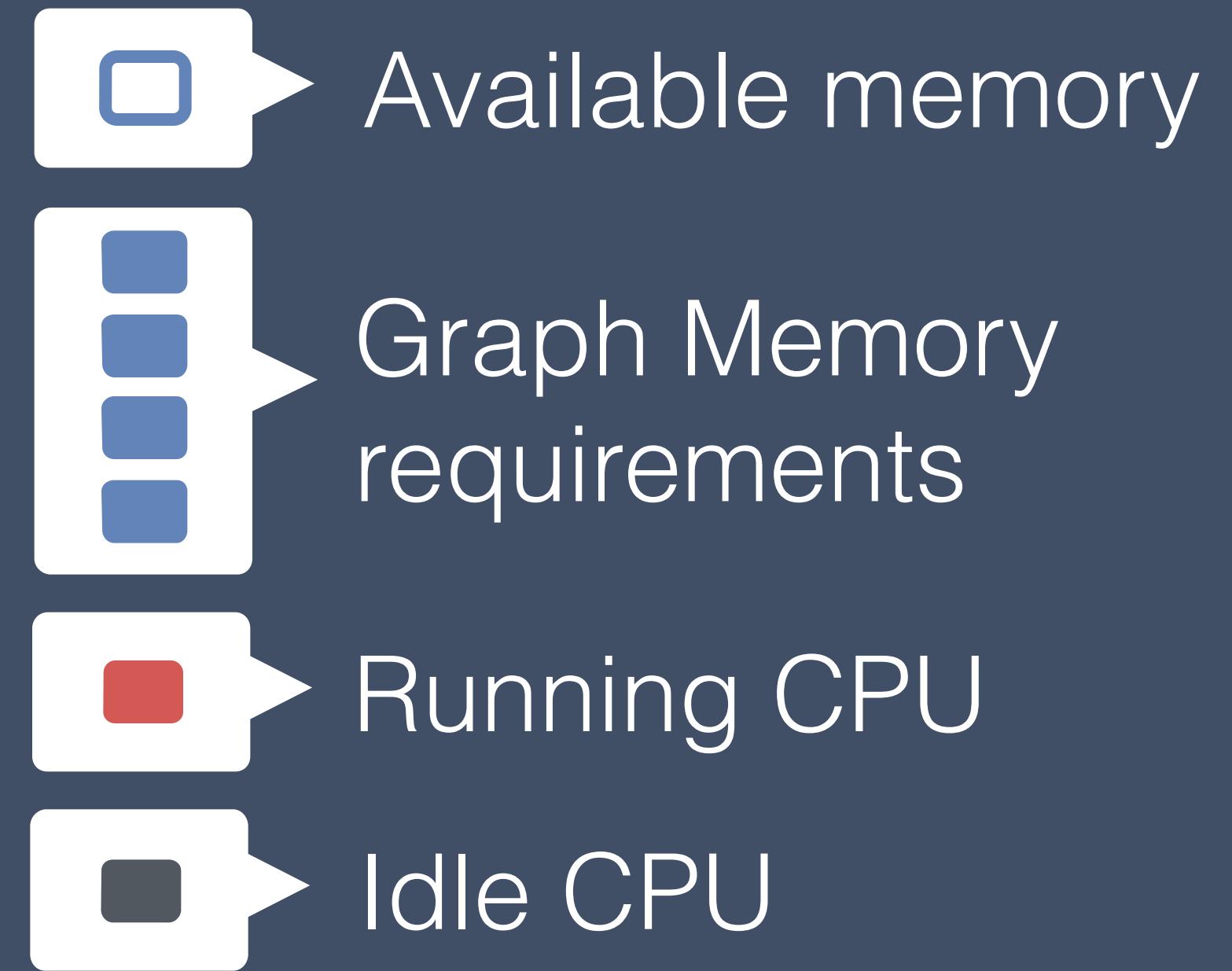
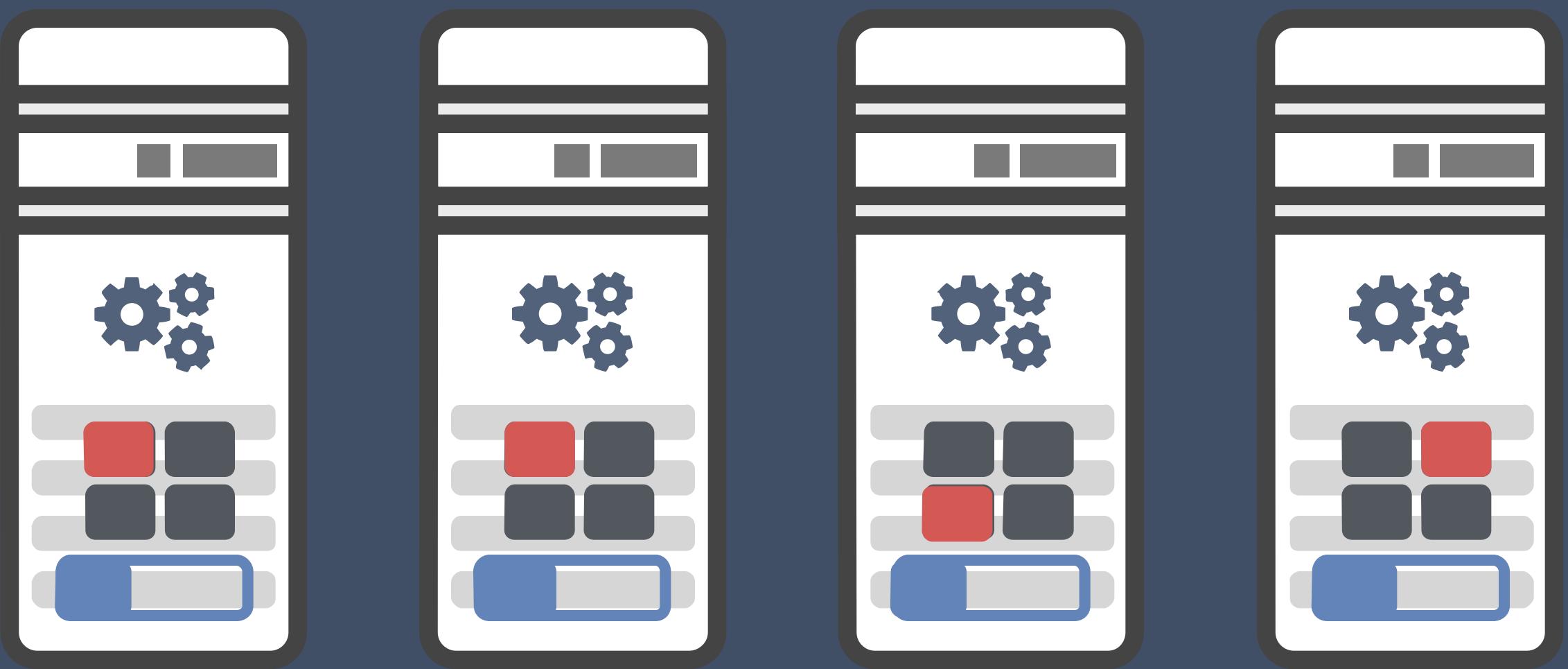
# How does it work?

# How does it work?



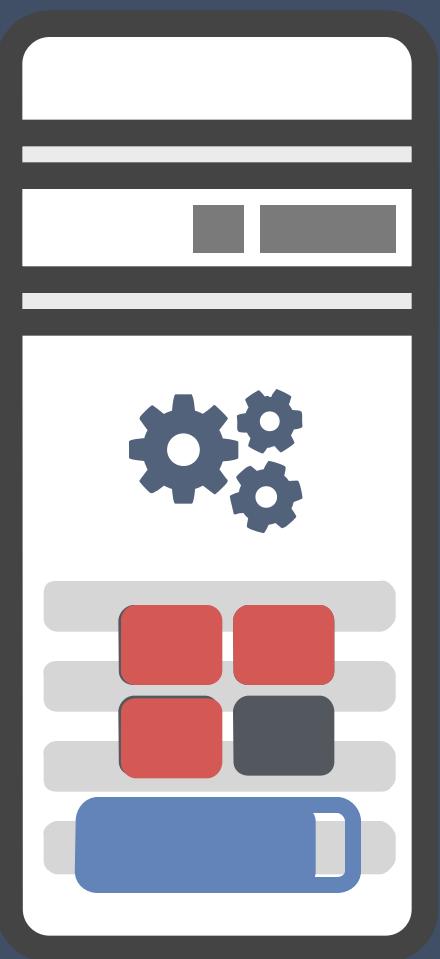
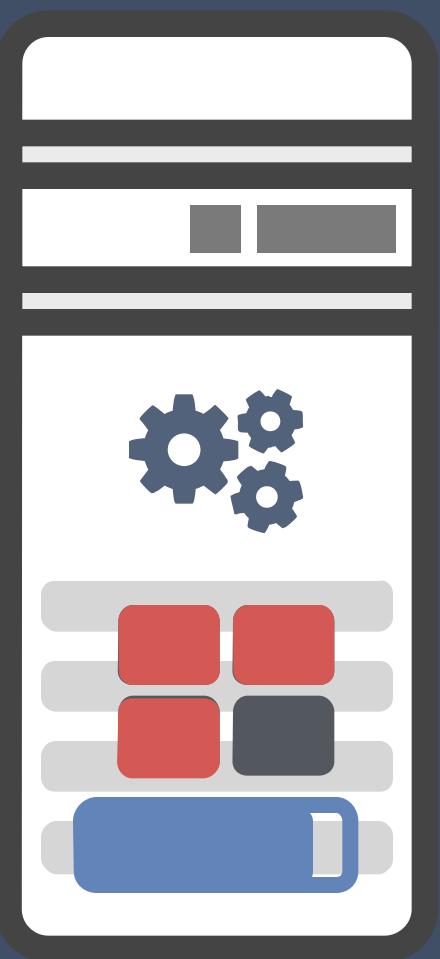
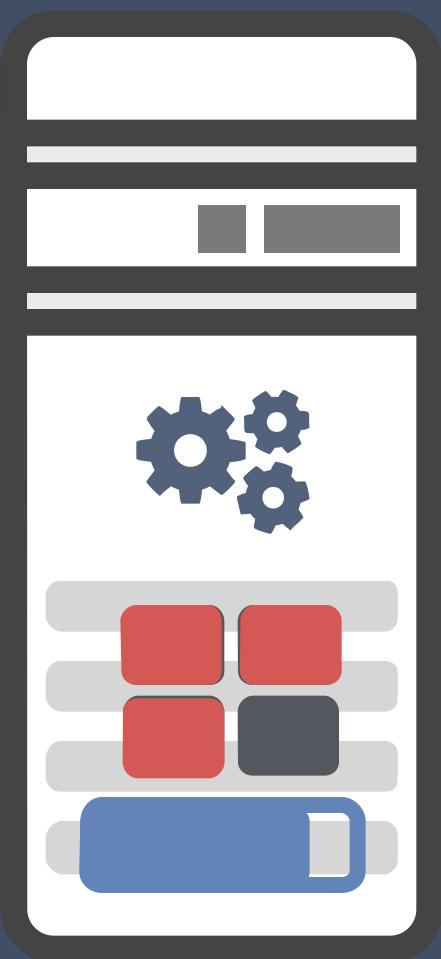
-  Available memory
-  Graph Memory requirements
-  Running CPU
-  Idle CPU

# How does it work?



**The graph is stored in 4 distributed machines**

# How does it work?



Available memory  
Graph Memory requirements  
Running CPU  
Idle CPU



**3 sets of landmark points are being processed in parallel**

