

Graph Partitioning

Mar 30, 2021

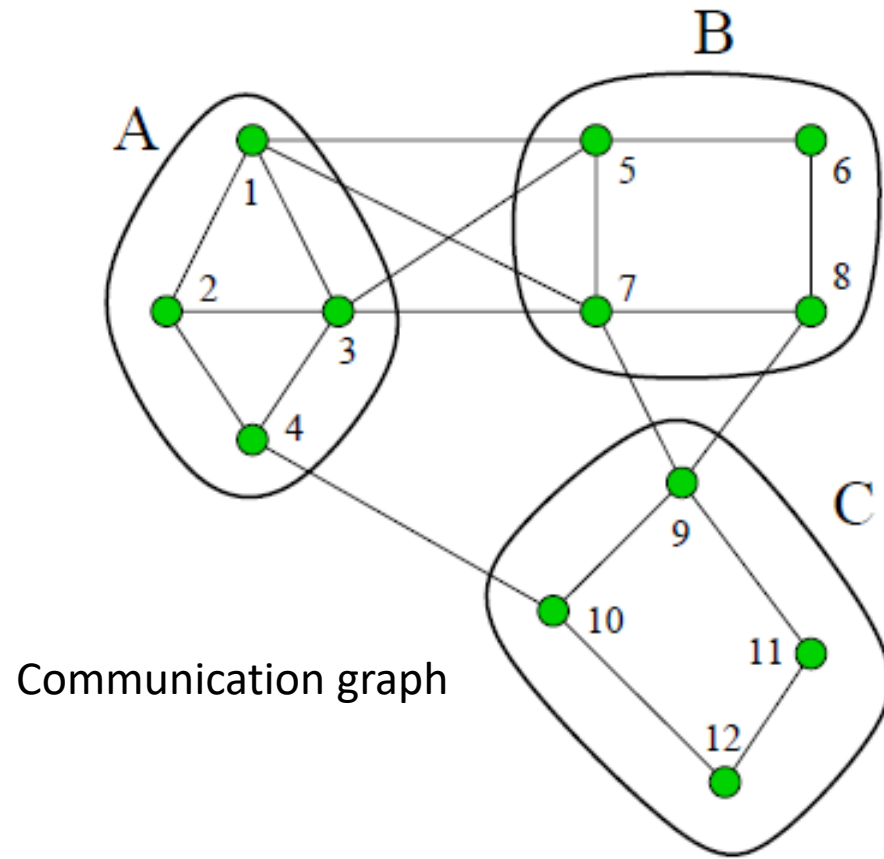


References

- Catalyurek et al., “Hypergraph-Partitioning-Based Decomposition for Parallel Sparse-Matrix Vector Multiplication”
- Karypis et al., “METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System”



Graph Partitioning



Partition the graph vertices such that the sum of weights of edges connecting the different parts is minimized.

Edge is a *cut* when its vertex pairs belong to two parts

Communication cost \propto edge-cut

Three subdomains/parts: A, B, C

Edge cut = 7

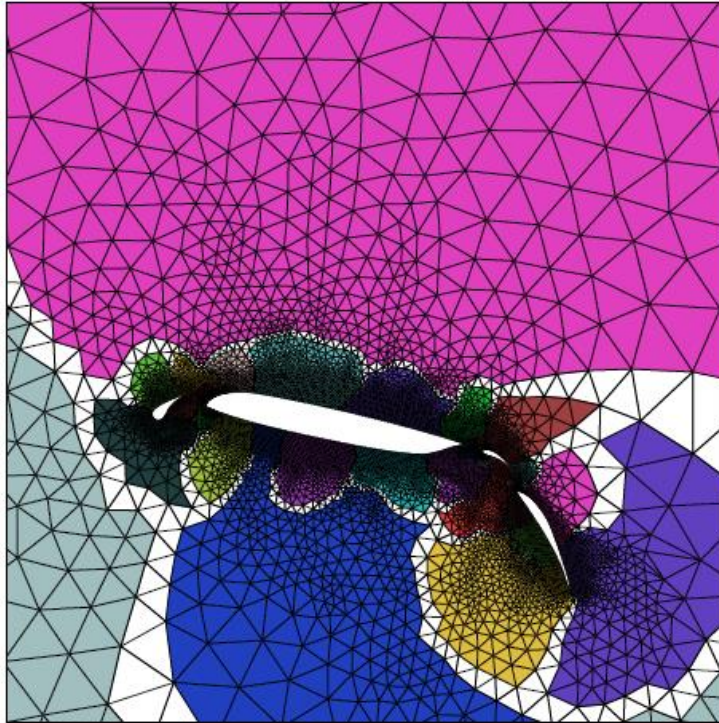


Applications

- Sparse Matrix Vector Multiplication
- Unstructured Mesh
- VLSI design



Meshing



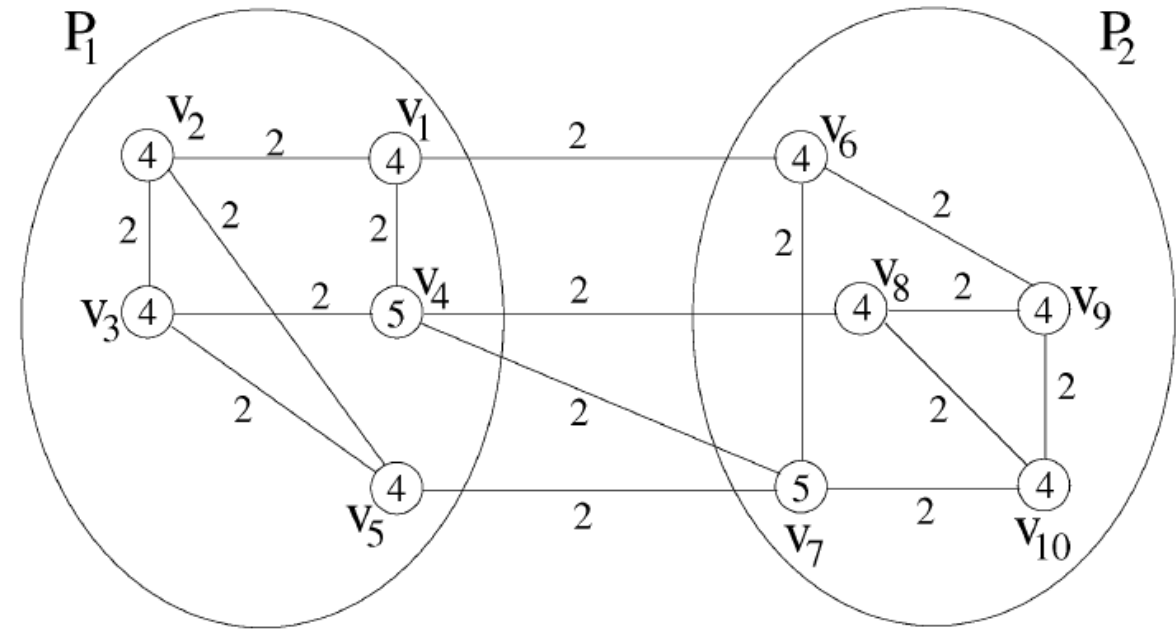
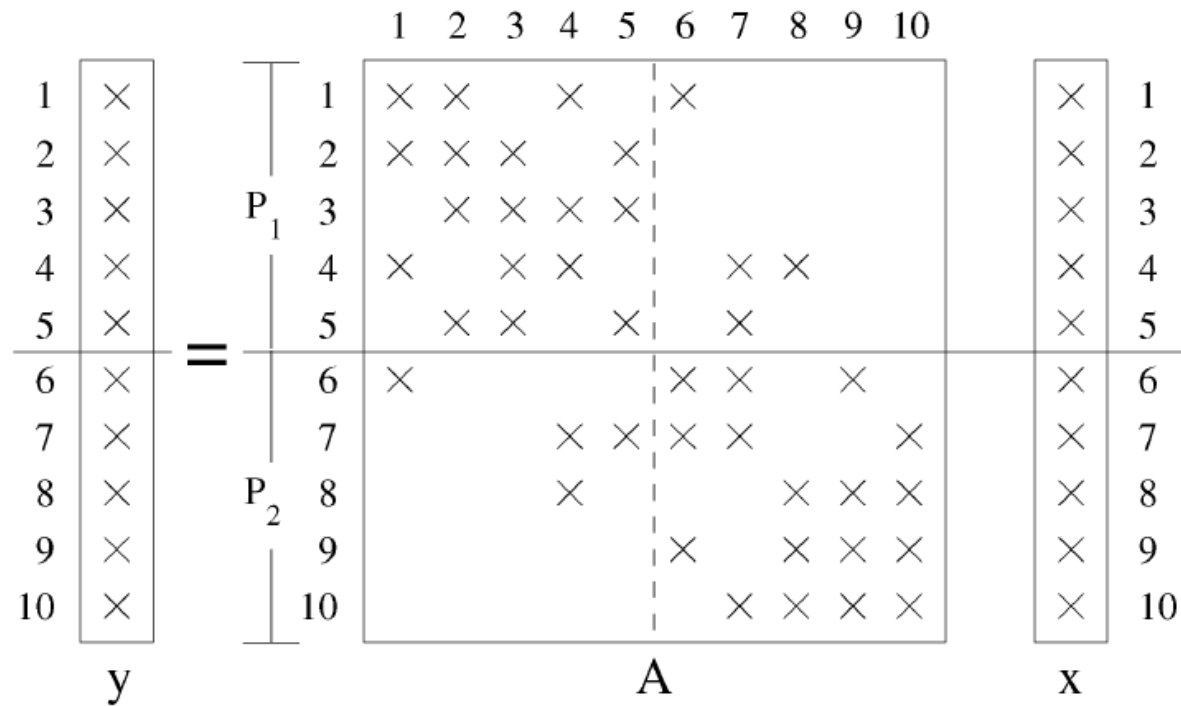
2D irregular mesh of an airfoil *

Q: How can you ensure efficient execution of these simulations?

Q: How do you model irregular mesh computations/communications as graphs?



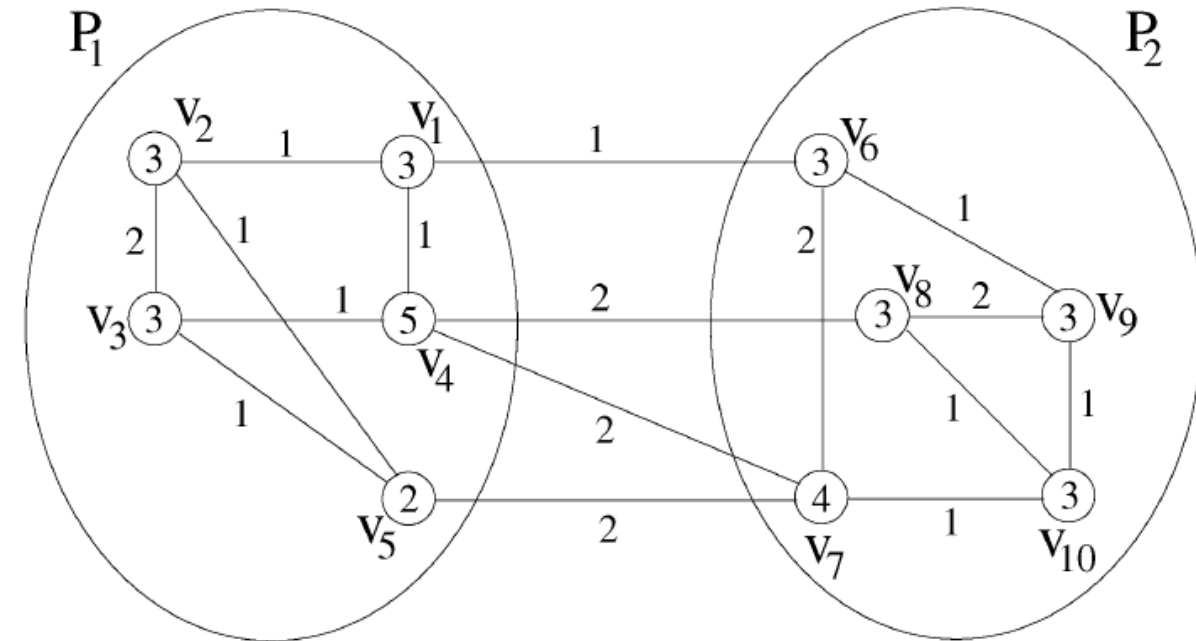
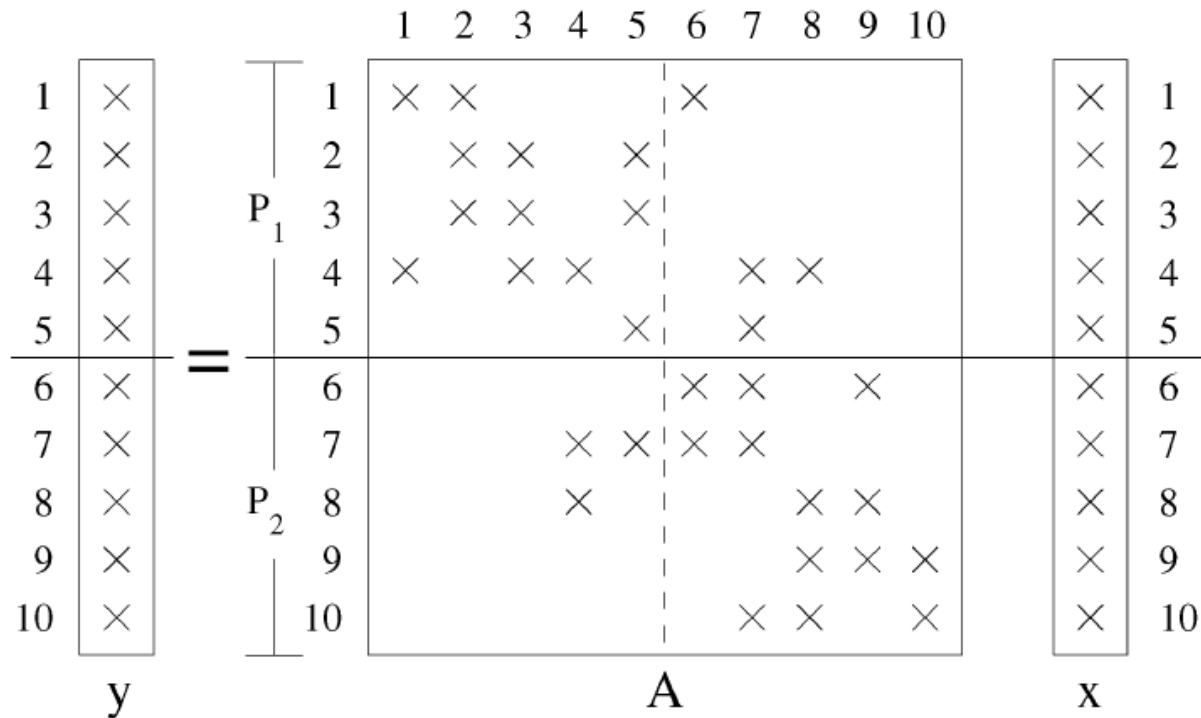
Graph Model for SpMV



Two-way partitioning of symmetric matrix for SpMV computation
 Computational load w_i = Number of non-zero entries in row/column i



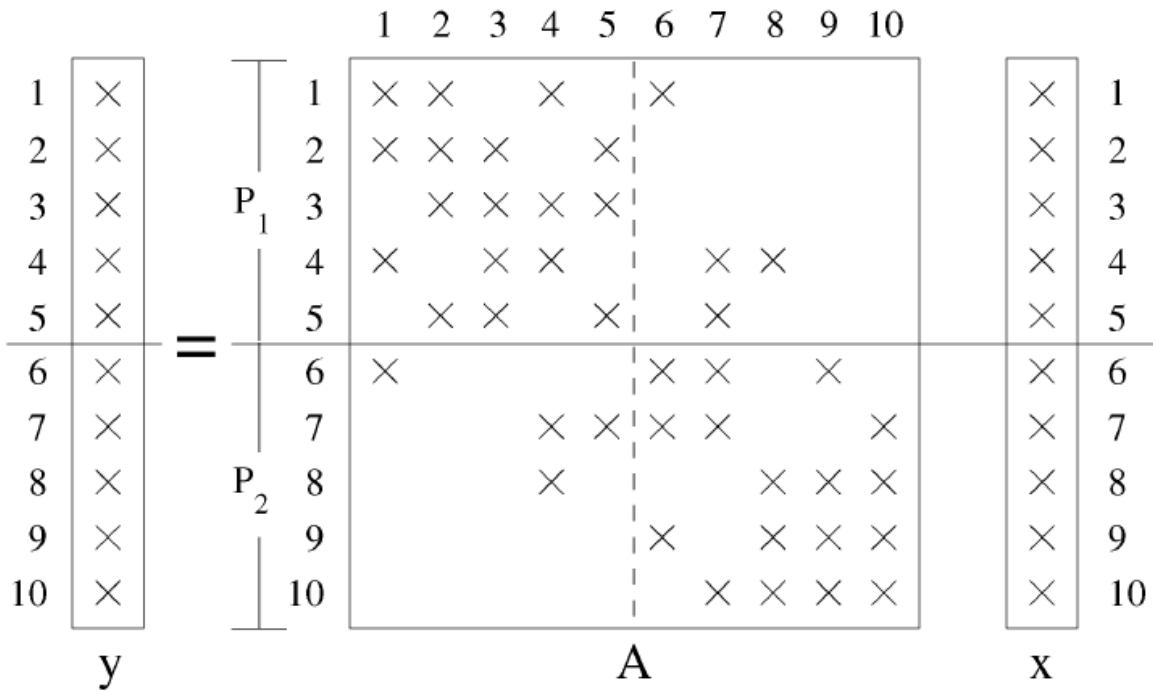
Graph Model for SpMV (Nonsymmetric Matrix)



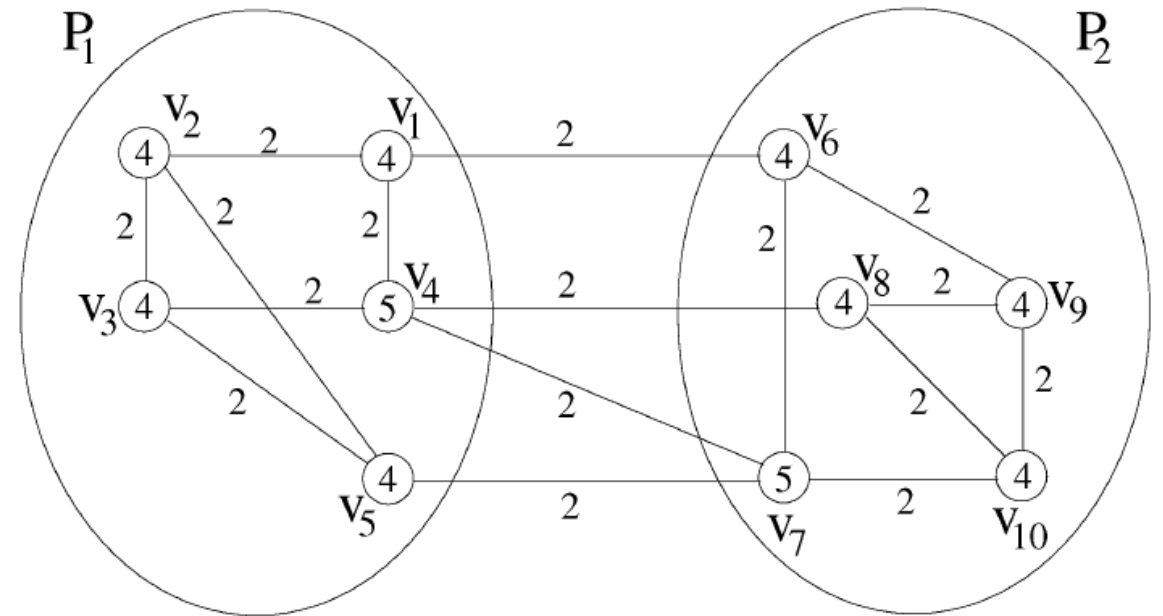
Two-way partitioning of nonsymmetric matrix for SpMV computation
 Computational load w_i = Number of non-zero entries in row/column i



Graph Model for SpMV



- Computation load?
 - Similar for both processors



- Number of communications?
 - 8 as per this graph
 - Actually 6
 - E.g. $Y'_7 = A_{7,4} X_7 + A_{7,5} X_7$



Graph Partitioning

Problem: Partition the graph such that the edge-cut is minimized and partitions are equally-sized.

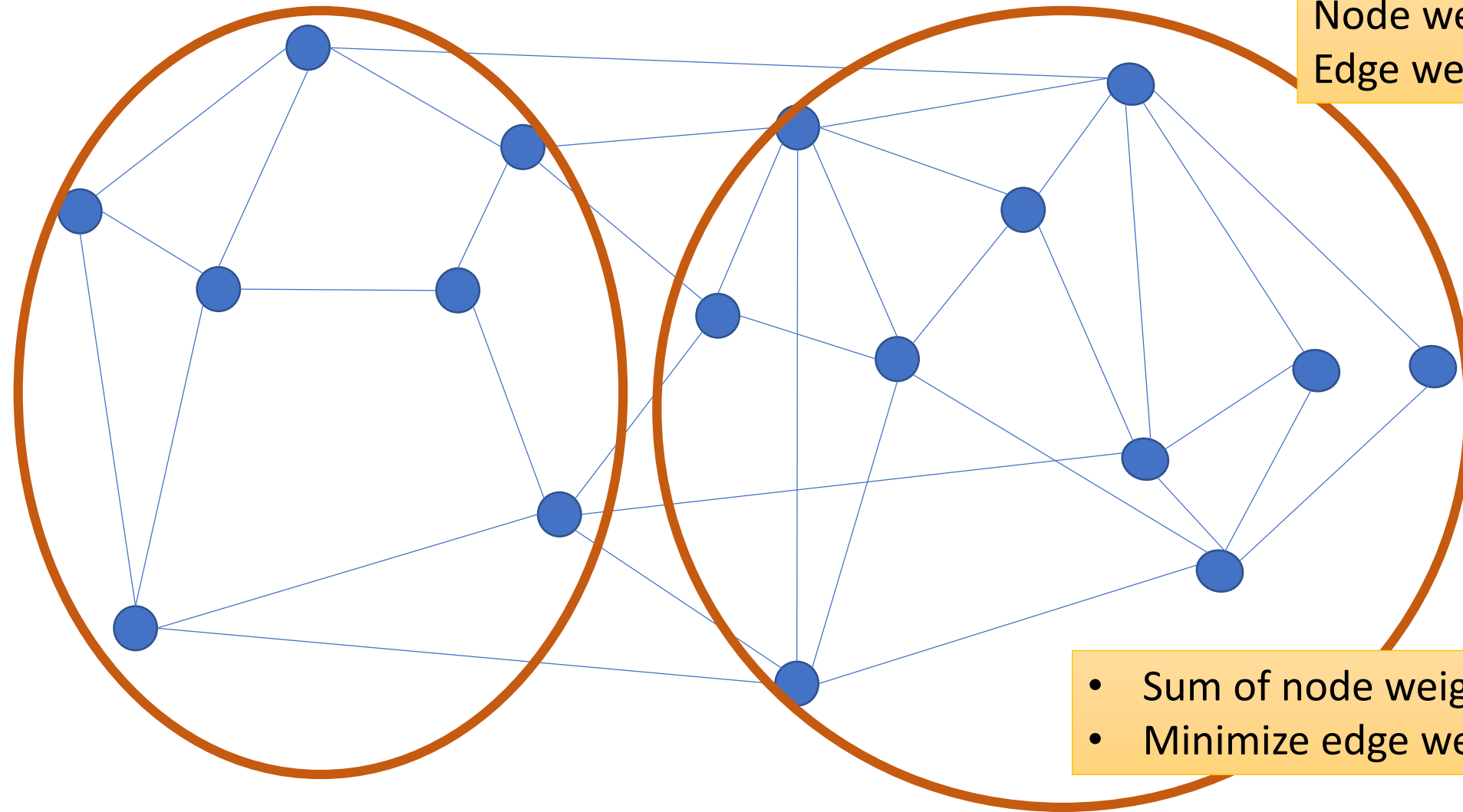
Techniques

1. Geometric (applicable when coordinate information available)
 - Group together based on neighbourhood
 - Often directly used on meshes
2. Multilevel
 - Group together based on connectivity



2-way partitioning (NP-hard)

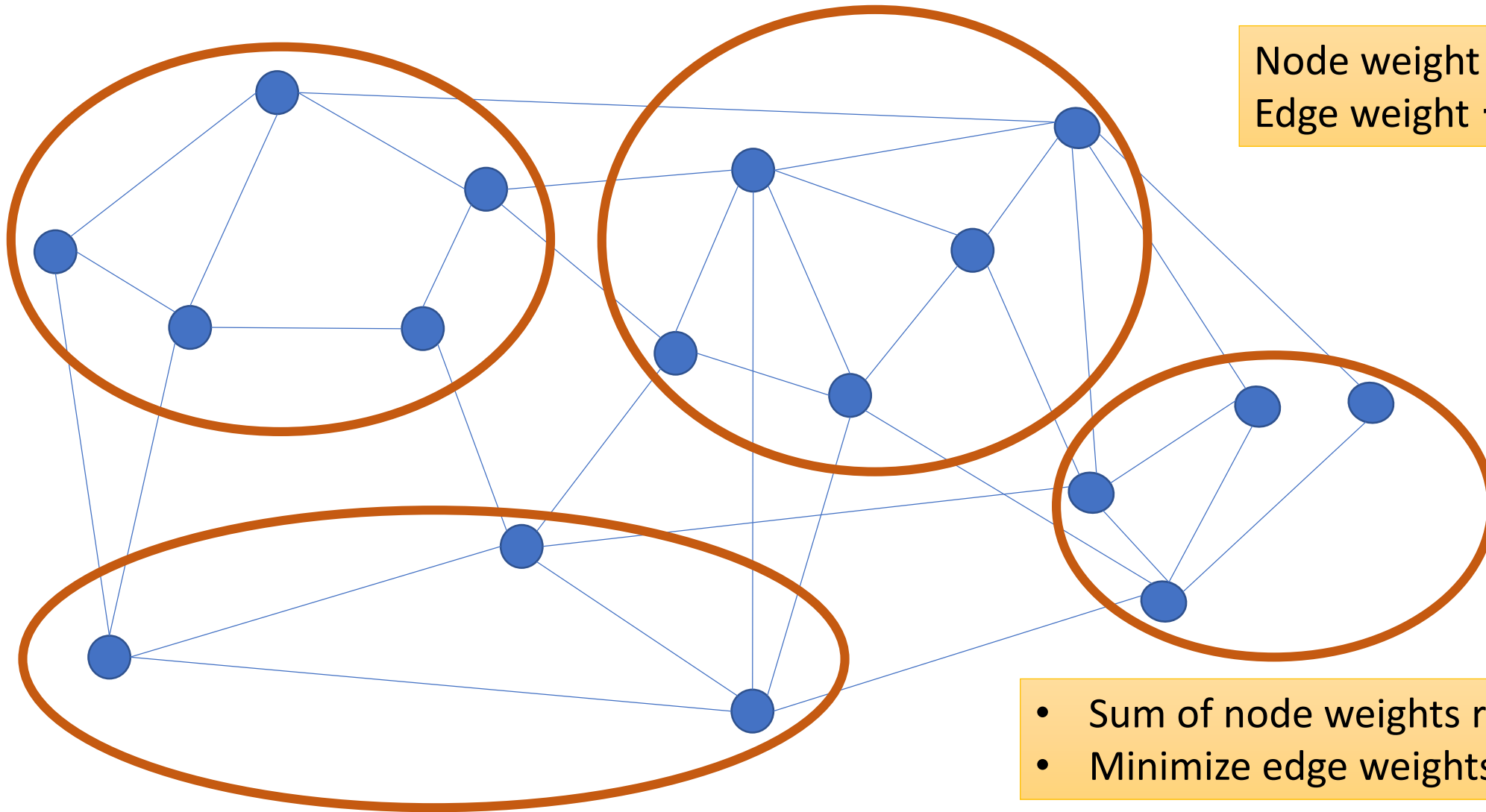
Node weight \rightarrow Computation
Edge weight \rightarrow Communication



- Sum of node weights roughly same
- Minimize edge weights across partitions



k-way partitioning



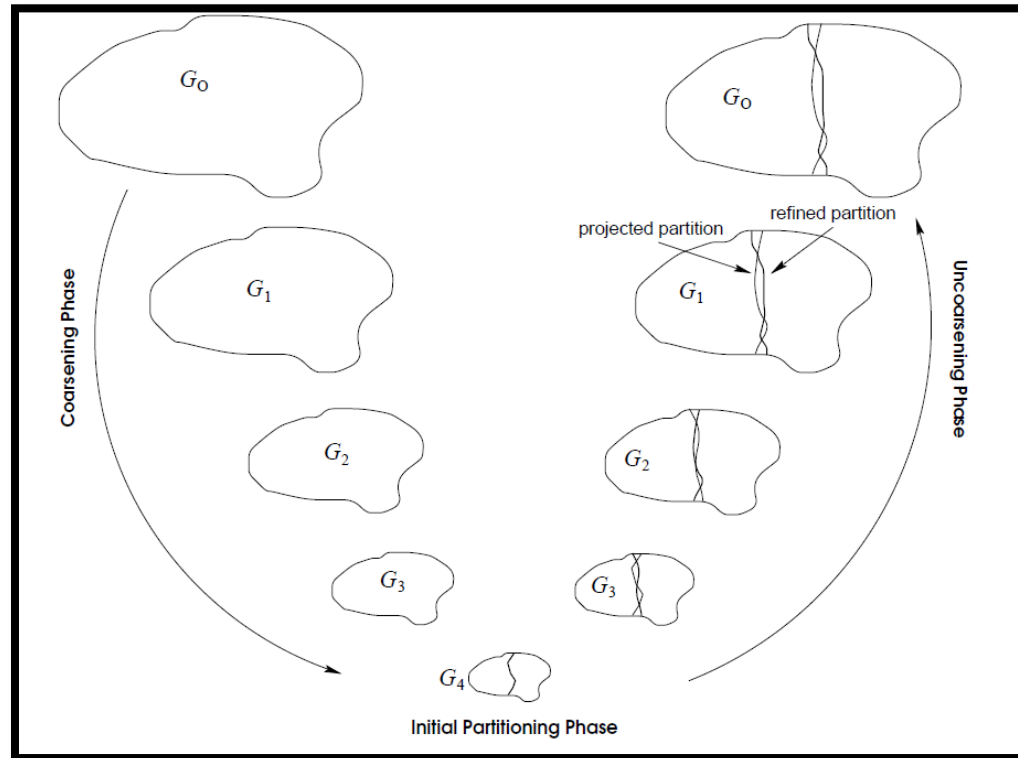
Node weight \rightarrow Computation
Edge weight \rightarrow Communication

k partitions after
log k phases

- Sum of node weights roughly same
- Minimize edge weights across partitions



Multilevel Partitioning for Bisections



Coarsen



G_0 is coarsened to a few hundred vertices

A set of vertices of G_i is combined to form a single vertex of G_{i+1}

G_{i+1} is constructed from G_i by finding a matching of G_i

Maximal matching will result in smaller graph

RM

HEM

KL

High-quality bisection (2-way partition) of smaller graph is computed

KL refinement

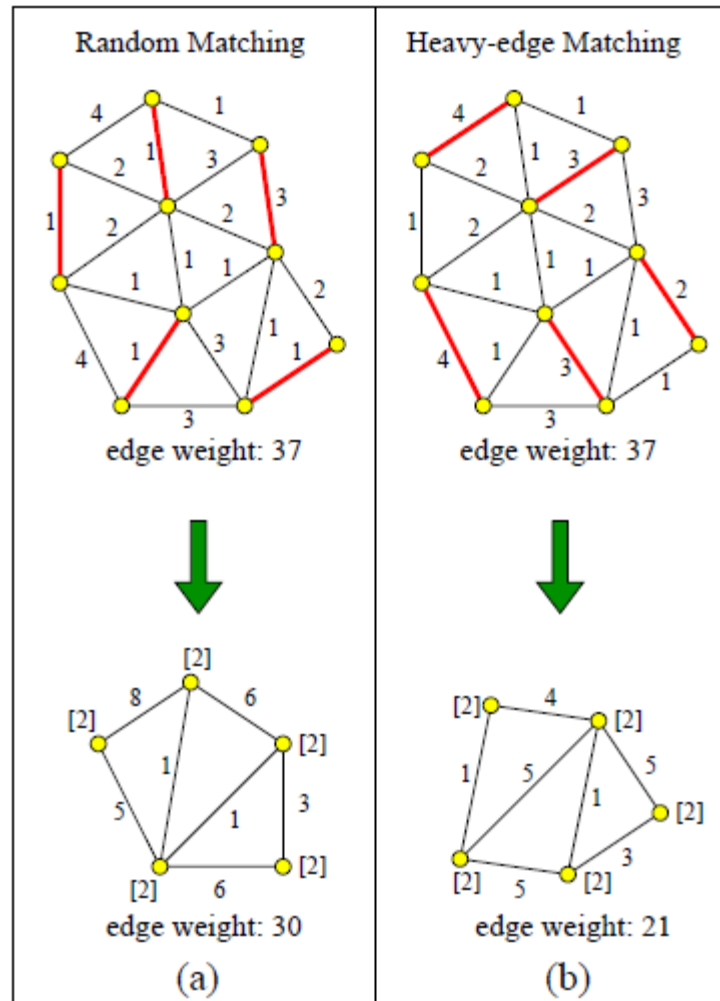
Smaller graph is uncoarsened – may lead to better edge cuts

REFERENCES:

KARYPIS ET AL. A FAST AND HIGH QUALITY MULTILEVEL SCHEME FOR PARTITIONING IRREGULAR GRAPHS
SCHLOEGEL ET AL., GRAPH PARTITIONING FOR HIGH PERFORMANCE SCIENTIFIC SIMULATIONS



Random and Heavy-edge Matching



K-way partition

- Recursive bisection is used
- Coarsen
 - RM/HEM
- Subdivide each part using 2-way partitions
 - KL algorithm
- Uncoarsen
 - KL refinement algorithm
 - Iterative algorithm (5 – 10 on average, in practice)
 - Move subsets of vertices if it leads to decrease in edge cuts
- $\log k$ phases
- Used extensively due to its simplicity



Some Results for 256-way partitioning

Matrix	MSB	MSB-KL	Chaco-ML	Our multilevel
144	607.27	650.76	95.59	48.14
4ELT	24.95	26.56	7.01	3.13
598A	420.12	450.93	67.27	35.05
ADD32	18.72	21.88	4.23	1.63
AUTO	2214.24	2361.03	322.31	179.15
BCSSTK30	426.45	430.43	51.41	22.08
BCSSTK31	309.06	268.09	39.68	15.21
BCSSTK32	474.64	540.60	53.10	22.50
BBMAT	474.23	504.68	55.51	25.51
BRACK2	218.36	222.92	31.61	16.52
CANT	978.48	1167.87	108.38	47.70
COPTER2	185.39	194.71	31.92	16.11
CYLINDER93	671.33	697.85	91.41	39.10
FINAN512	311.01	340.01	31.00	17.98
FLAP	279.67	331.37	35.96	16.50
INPRO1	341.88	352.11	56.05	24.60
KEN-11	121.94	137.73	13.69	4.09



ParMETIS (Parallel METIS)

<http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview>

```
class time mpirun -np 24 -hosts csews22,csews24,csews26,csews36 parmetis Graphs/rotor.graph 1 6 1 1 6 1
reading file: Graphs/rotor.graph
finished reading file: Graphs/rotor.graph
[ 99617 1324862 4150 4151] [600] [ 0.000] [ 0.000]
[ 53191 788584 2154 2347] [600] [ 0.000] [ 0.000]
[ 28506 428506 1135 1257] [600] [ 0.000] [ 0.000]
[ 15384 232572 600 670] [600] [ 0.000] [ 0.000]
[ 8399 126748 321 375] [600] [ 0.000] [ 0.000]
[ 4614 68864 172 221] [600] [ 0.000] [ 0.000]
[ 2564 37344 95 119] [600] [ 0.000] [ 0.001]
[ 1448 20118 50 68] [600] [ 0.000] [ 0.001]
[ 845 11004 28 45] [600] [ 0.000] [ 0.002]
[ 742 9762 25 38] [600] [ 0.000] [ 0.002]
nvtxs: 742, cut: 20515, balance: 1.018
nvtxs: 845, cut: 19852, balance: 1.008
nvtxs: 1448, cut: 18818, balance: 1.026
nvtxs: 2564, cut: 17857, balance: 1.045
nvtxs: 4614, cut: 16685, balance: 1.032
nvtxs: 8399, cut: 15778, balance: 1.027
nvtxs: 15384, cut: 14979, balance: 1.023
nvtxs: 28506, cut: 13827, balance: 1.019
nvtxs: 53191, cut: 12810, balance: 1.019
nvtxs: 99617, cut: 11564, balance: 1.016
Final 6-way Cut: 11564 Balance: 1.016

real 0m26.993s
user 0m42.232s
sys 0m25.888s
```

```
class time mpirun -np 12 parmetis Graphs/rotor.graph 1 6 1 1 6 1
reading file: Graphs/rotor.graph
finished reading file: Graphs/rotor.graph
[ 99617 1324862 8301 8302] [300] [ 0.000] [ 0.000]
[ 53114 788078 4290 4699] [300] [ 0.000] [ 0.000]
[ 28389 426478 2279 2487] [300] [ 0.000] [ 0.000]
[ 15307 231214 1225 1353] [300] [ 0.000] [ 0.000]
[ 8360 126112 639 736] [300] [ 0.000] [ 0.000]
[ 4623 68992 360 421] [300] [ 0.000] [ 0.000]
[ 2598 37404 189 258] [300] [ 0.000] [ 0.001]
[ 1501 20490 114 151] [300] [ 0.000] [ 0.001]
[ 893 11246 60 93] [300] [ 0.000] [ 0.002]
[ 555 6280 37 66] [300] [ 0.000] [ 0.005]
[ 467 5238 30 55] [300] [ 0.000] [ 0.005]
nvtxs: 467, cut: 22248, balance: 1.024
nvtxs: 555, cut: 20933, balance: 1.036
nvtxs: 893, cut: 20317, balance: 1.051
nvtxs: 1501, cut: 18857, balance: 1.047
nvtxs: 2598, cut: 17298, balance: 1.051
nvtxs: 4623, cut: 16238, balance: 1.051
nvtxs: 8360, cut: 15091, balance: 1.048
nvtxs: 15307, cut: 13861, balance: 1.048
nvtxs: 28389, cut: 12859, balance: 1.050
nvtxs: 53114, cut: 11976, balance: 1.050
nvtxs: 99617, cut: 10609, balance: 1.049
Final 6-way Cut: 10609 Balance: 1.049

real 0m57.833s
user 1m32.244s
sys 0m3.836s
```



Output

```
for i in `seq 0 5` ; do grep ^${i} rotor.graph.part | wc -l ; done
```

16599

16222

15842

16147

17415

17392

```
class head -30 rotor.graph.part
5
5
5
5
5
5
5
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
3
3
3
3
3
3
3
3
3
```

