Graph Partitioning

T. E. KALAYCI

machine Learning and Intelligent OptimizatioN (LION) Research Group
Trento University

20 April 2016

Table of contents

Graph Partitioning Problem

- An important problem in Computer Science
- We try to produce smaller components with specific properties from graph $G = (V, E)$
- For example $k - way$ partition divides the vertex set into k smaller components
- It is one of the fundamental algorithmic operations
- Partitioning large graphs is often an important subproblem for complexity reduction or parallelism [3]

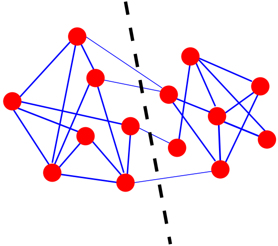


Image Source: S. Fortunato, C. Castellano, Community Structure in Graphs, 2007, http://arxiv.org/pdf/0712.2716v1.pdf

Application Areas and Solving Approaches

- Many areas of Computer Science like parallel processing, complex networks, road networks, image processing, sparse matrix factorization, network partitioning, VLSI physical design, etc [2, 3].
- Many different approaches like global optimization, iterative improvement heuristics, multilevel graph partitioning, evolutionary methods and further meta-heuristics for solving the problem [3].

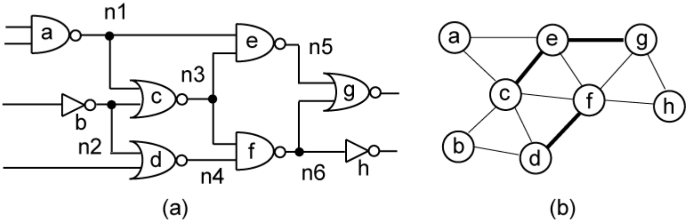


Image Source: S. K. Lim, Fig. 2-1 in Practical Problems in VLSI Physical Design, 2008, `http://users.ece.gatech.edu/limsk/book/`

Problem Definition

- Graph is defined as $G = (V, E)$ where V are set of vertices and E are set of edges.
- We are going to divide vertices into disjoint subsets
- Number of edges whose endpoints are in different subsets would be minimized
- Also we can have balance property and create balanced partitions.
- Let's define bipartitioning according to these objectives:
  - A balanced bipartition of the graph $G = (V, E)$ is an unordered pair (Set_0, Set_1) of subsets of V such that $Set_0 \cup Set_1 = V$ and $Set_0 \cap Set_1 = \emptyset$.
  - Difference between cardinalities of the two sets (i.e., $||Set_0| - |Set_1||$) is as small as possible (*zero* if V contains an even number of vertices, *one* otherwise)
  - The cut size (denoted as $f(Set_0, Set_1)$) is minimized.
- Many possible partitioning to search: n choose n/2, $\binom{n}{n/2} = \frac{n!}{((n/2)!)^2}$
- Finding optimal partition is a NP-complete problem.

Algorithms

- Best known and widely used bipartitioning heuristics are:
  - Kernighan-Lin heuristic
  - Fiduccia-Mattheyses variant of Kernighan-Lin
  - Karypis, Kumar: another improvement to KL, only consider vertices on boundary

Kernighan-Lin

- Most popular heuristic for balanced bipartitioning [5]
- It is an iterative improvement algorithm
- It starts with an initial partition and improves it iteratively
- As long as cut size keeps decreasing
  - Vertex pairs with largest decrease (or the smallest increase) in cut size are exchanged
  - Exchanged vertices are then locked and prohibited to participate in further exchanges
  - Process will continue until all vertices are locked
- It is a $O(n^2 log(n))$ algorithm
- Drawbacks
  - Only handles unit vertex weights
  - Handles only exact bisections
  - Cannot handle hypergraphs
  - One pass of the algorithm is expensive

Fiduccia-Mattheyses

- A classical approach to solve the hypergraph bipartitioning problem [4]
- Improves K-L heuristic:
  - Aims at reducing net-cut costs; the concept of cutsize is extended to hypergraphs.
  - Only a single vertex is moved across the cut in a single move.
  - Vertices are weighted.
  - Can handle "unbalanced" partitions; a balance factor is introduced.
  - A special data structure (bucket list) is used to select vertices to be moved across the cut to improve running time.
- It is a linear time algorithm ($O(N)$)

Min-Max Greedy

- A simple greedy construction for bipartitioning [2]
- Iteratively adds vertex to a set based on $E(i, set) \equiv |\{(i,j) \in E$ such that $j \in set\}|$ (number of edges incident on vertex i whose other endpoint is in the set)
- Selects the vertex to a set which maximizes internal edges and minimizes external edges.
- Linear time algorithm: $O(|E|)$
- It is implemented using two arrays of buckets: one for each of the partition set

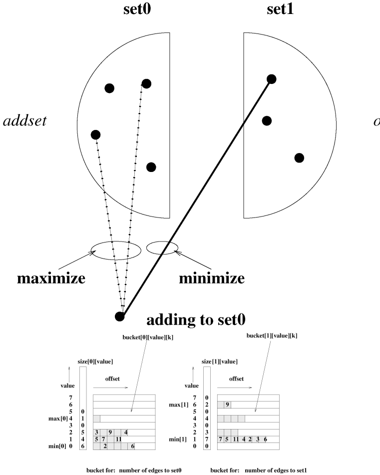


Image Source: Battiti and Bertossi [2]

Differential Greedy

- Can we use a single array of buckets?
- What if we minimize the *difference* between new edge across the cut and new internal edges?
- Diff-Greedy is the answer of these questions [1]
- It tracks only $E(i,0) - E(i,1)$ with a single buckets array which is a max-min priority queue.

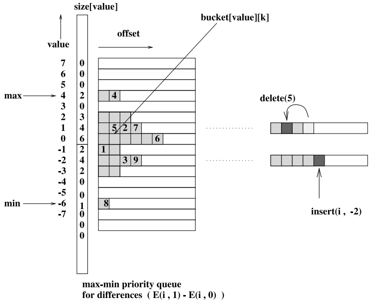
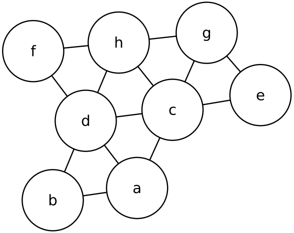


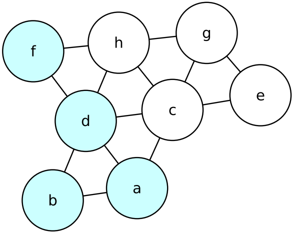
Image Source: Battiti and Bertossi [1]

MMG Example

Starting graph



Resulting graph

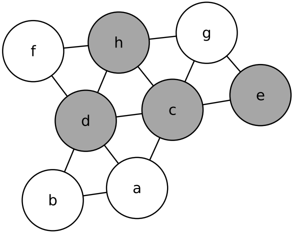


First step: We have selected A and H as random nodes for Set_0 and Set_1

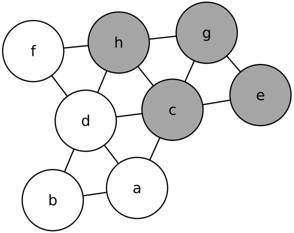
		2	3	4	5	6	7
	addset	0	1	0	1	0	1
B	**Set0**	1					
	Set1	0					
C	**Set0**	1	1	1	2	2	
	Set1	1	1	2	2	3	
D	**Set0**	1	2	2			
	Set1	1	1	1			
E	**Set0**	0	0	0	0		
	Set1	0	0	1	1		
F	**Set0**	0	0	0	1	1	
	Set1	1	1	1	1	1	
G	**Set0**	0	0				
	Set1	1	1				
MIN(otherset)		B,E	E,F,G	D,E,F	E	F	C
MAX(addset)		**B**	**G**	**D**	**E**	**F**	**C**

K-L Example

Cut size at the beginning = 8



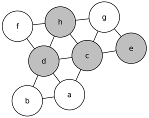
Cut size at the end = 4



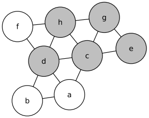
Pairs	Dx = Ex-Ix	Dy = Ey-Iy	c(x,y)	Gain = Dx+Dy − 2c(x,y)
a,c	2-1	2-3	1	-2
a,d	2-1	3-2	1	0
a,e	2-1	1-1	0	1
a,h	2-1	2-2	0	1
b,c	1-1	2-3	0	-1
b,d	1-1	3-2	1	-1
b,e	1-1	1-1	0	0
b,h	1-1	2-2	0	0
f,c	2-0	2-3	0	1
f,d	2-0	3-2	1	1
f,e	2-0	1-1	0	2
f,h	2-0	2-2	1	0
g,c	3-0	2-3	1	0
g,d	3-0	3-2	0	4
g,e	3-0	1-1	1	1
g,h	3-0	2-2	1	1

F-M Example

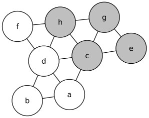
Cut size at the beginning $= 8$



Cut size after first move $= 5$



Cut size at the end $= 4$



	From Set	A	B	C	D	E	F	G	H	Result Cut Size
1	0	1	0	-1	1	0	2	3	0	5
2	1	1	0	-3	1	-2	2		0	4
3	0	-1	-2	-1		-2	0		0	4
4		-1	-2	-1		-2			-2	

Roberto Battiti and A. Alberto Bertossi.
Differential greedy for the 0-1 equicut problem.
In P. M. Pardalos and D. Du, editors, *in Proceedings of the DIMACS Workshop on Network Design: Connectivity and Facilities Location*, pages 3–21. American Mathematical Society, 1997.

Roberto Battiti and A. Alberto Bertossi.
Greedy, prohibition, and reactive heuristics for graph partitioning.
IEEE Transactions on Computers, 48(4):361–385, 1999.

Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz.
Recent advances in graph partitioning.
In *Algorithm Engineering Selected Results and Surveys*, pages 117–158. Springer International Publishing, 2016.

Charles M. Fiduccia and Robert M. Mattheyses.
A linear-time heuristic for improving network partitions.
In *19th Conference on Design Automation*, pages 175–181, 1982.

Brian W. Kernighan and Shen Lin.
An efficient heuristic procedure for partitioning graphs.
The Bell Systems Technical Journal, 49(2):291–307, 1970.