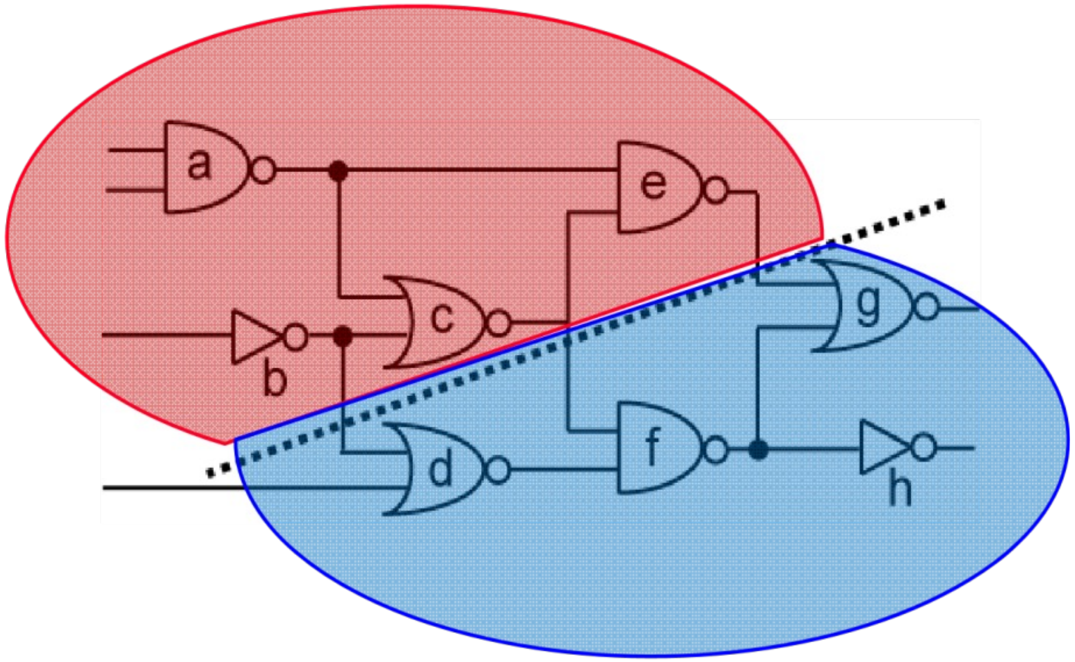
Lecture 4: Circuit Partitioning – II

Tsung-Wei (TW) Huang
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT

Recap: Circuit Partition

- **An essential step for reducing algorithm design complexity**
  - Divide and conquer (D&C)
- **Input**
  - A circuit graph
- **Output**
  - A set of partitioned subgraphs
- **Objective**
  - Minimize cross-connection



Recap: KL Algorithm

1. Pair-wise exchange of nodes to reduce cut size
2. Allow cut size to increase temporarily within a pass
3. Compute the gain of a swap

    Repeat

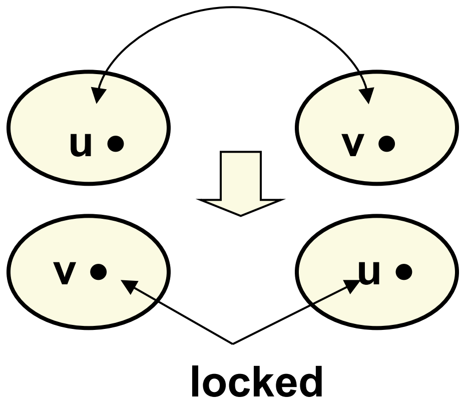
        Perform a feasible swap of max gain

        Mark swapped nodes "locked"

        Update swap gains

    Until no feasible swap

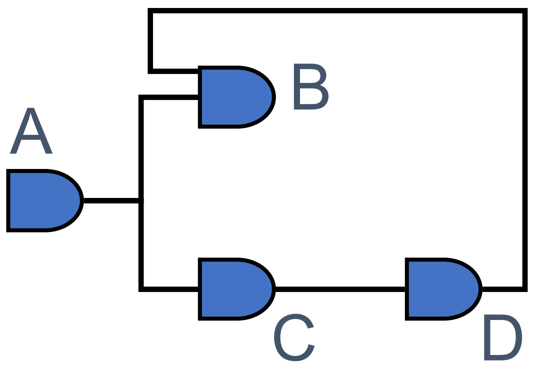
4. Find max prefix partial sum in gain sequence $g_1, g_2, \ldots, g_m$
5. Make corresponding swaps permanent
6. Start another pass if current pass reduces the cut size



Recap: Drawbacks of KL Algorithm

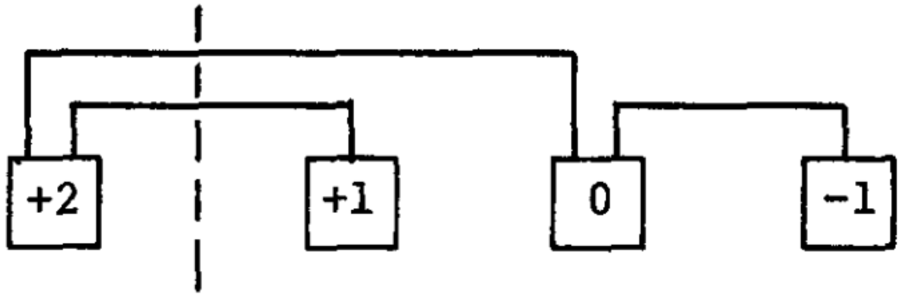
- **Handle only unit vertex weights**
  - Vertex weights might represent block sizes, different from blocks to blocks in real situation
- **Handle only exact bisection**
  - Need dummy vertices to handle the unbalanced problem
- **Handle only non-hypergraphs**
  - Practical circuits have many terminal nodes for each cell output
  - Need to handle multi-terminal nets directly

A has two terminals, B and C!

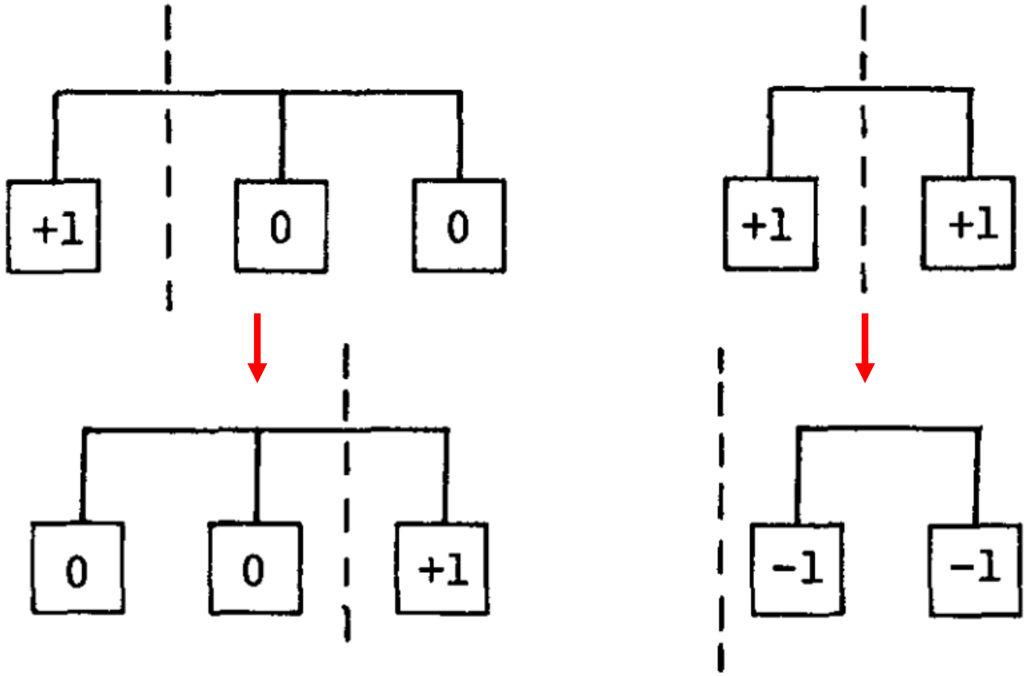


Fiduccia-Mattheyses (FM) Algorithm

- Fiduccia and Mattheyses, “A linear time heuristic for improving network partitions,” *ACM Design Automation Conference (DAC)*, 1982



Cell gains based on “net”

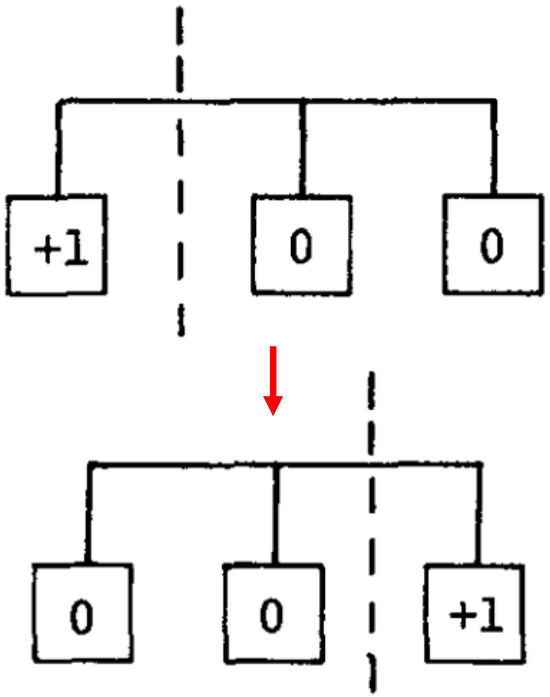


Hypergraph Partition Problem Formulation

- **Input: A hypergraph with**
  - Set vertices V. (|V| = n)
  - Set of hyperedges E (total # pins in netlist = p)
  - Area a_u for each vertex u in V
  - Cost c_e for each hyperedge in e
  - An area ratio r
- **Output: two partitions X & Y such that**
  - Total cost of hyperedges cut is minimized
  - area(X) / (area(X) + area(Y)) is about *r*
- **This problem has been proven to be NP-complete**

Ideas of FM Algorithm

- **Similar to KL:**
  - Work in passes
  - Lock vertices after moved
  - Only move those vertices up to the maximum partial sum of gain
- **Difference from KL:**
  - Not exchanging pairs of vertices
    - Move only one vertex at each time
  - Calculate the gain value based on connected nets
  - The use of gain bucket data structure



FM Partitioning Algorithm

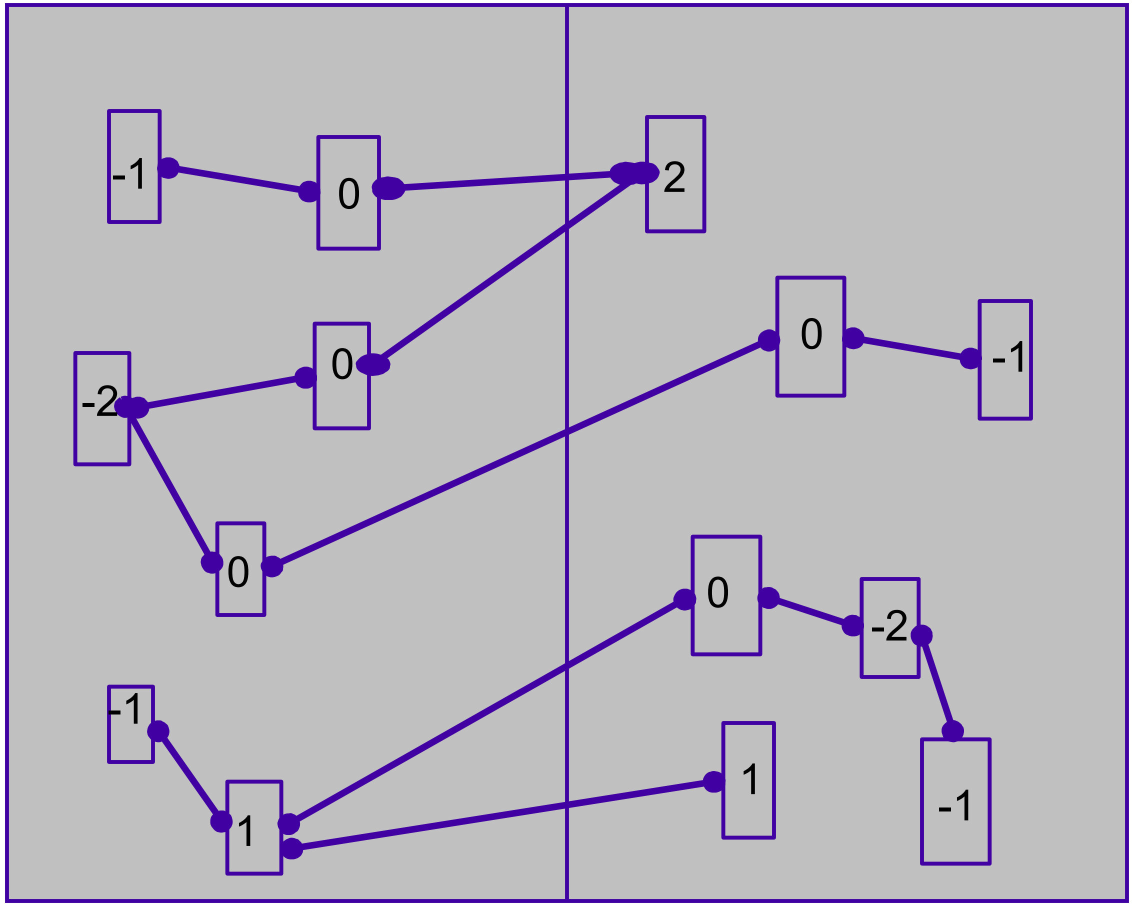
- Moves are made based on object gain
- Object Gain: The amount of change in cut crossings that will occur if an object is moved from its current partition into the other partition
- A **pass** description

While there is unlocked object

1. Each object is assigned a gain
2. Objects are put into a sorted gain list
3. The object with the highest gain from the larger of the two sides is selected and moved
4. The moved object is "locked"
5. Gains of "touched" objects are recomputed
6. Gain lists are resorted

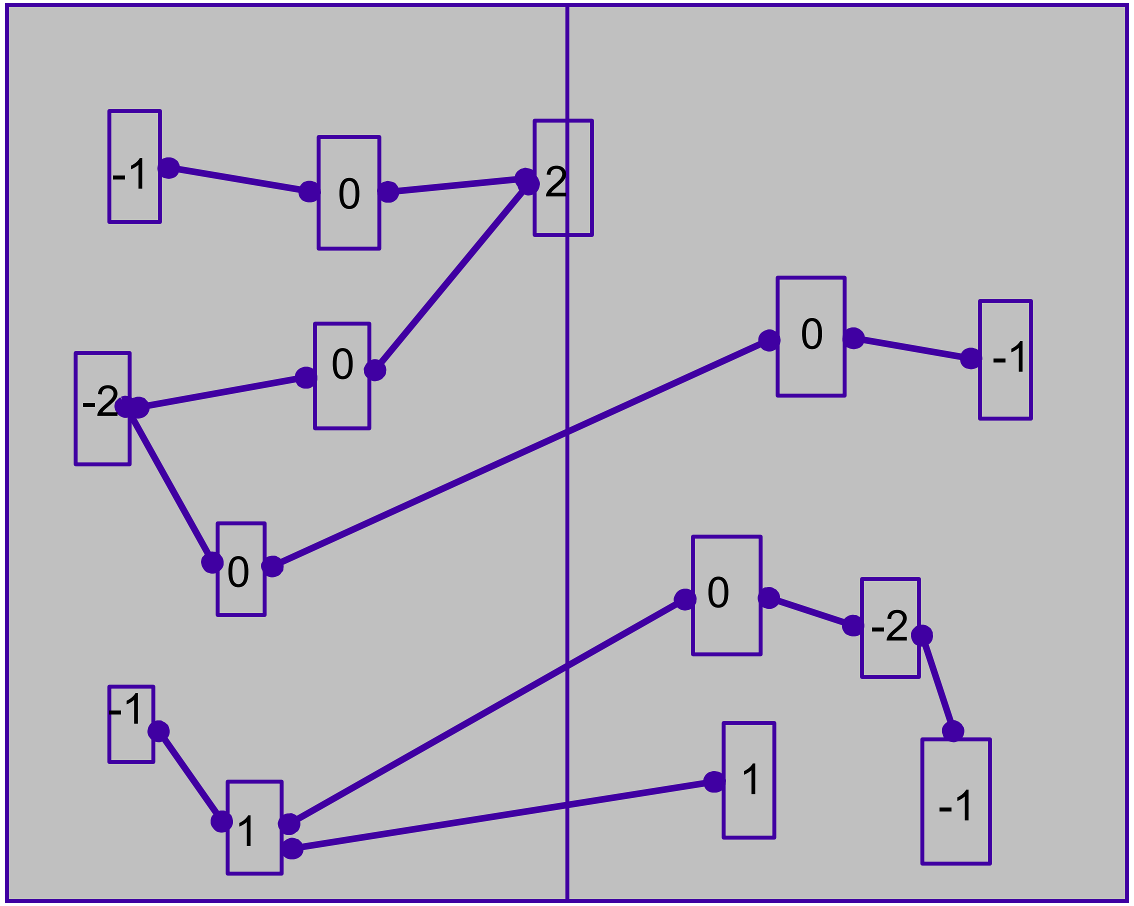
- Repeat the pass until there is no improvement

FM Algorithm Walkthrough – 1

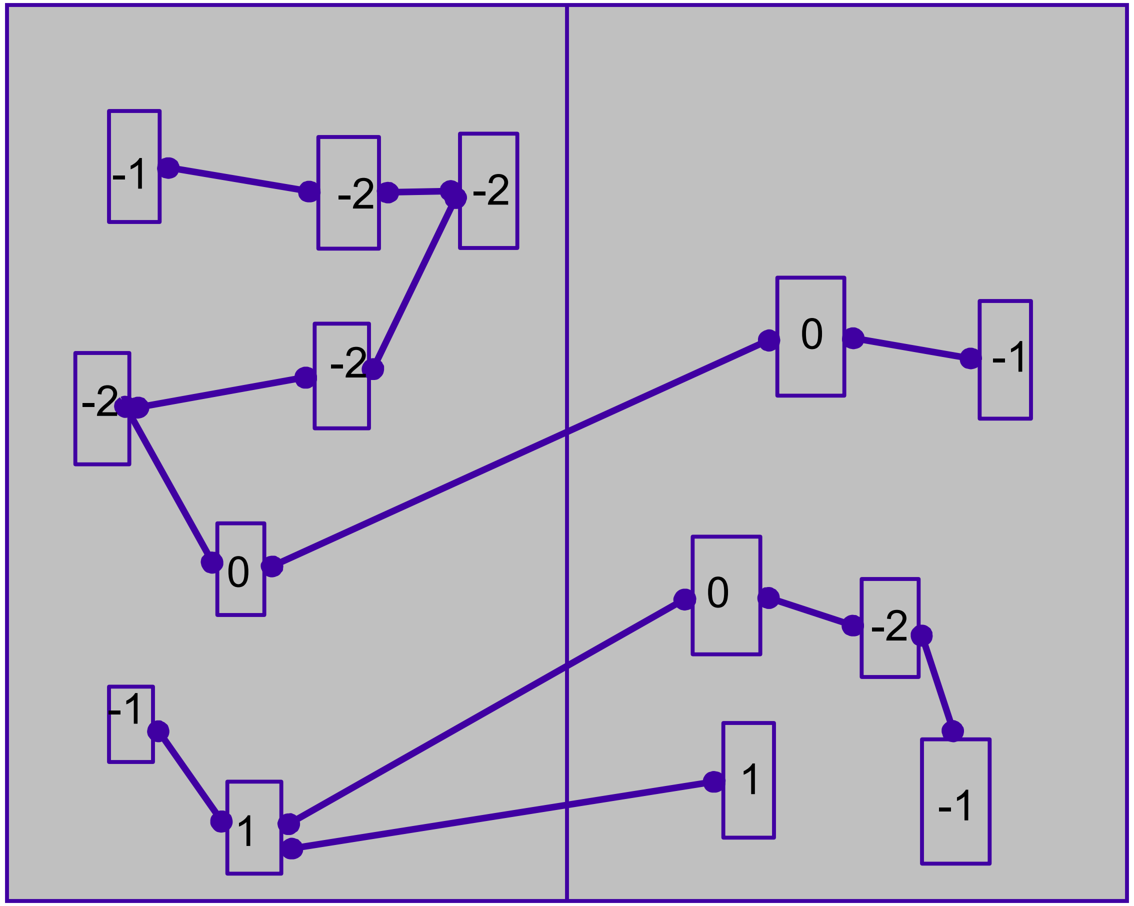


Let's focus on 2-pin net first for simplicity (degenerate to KL)

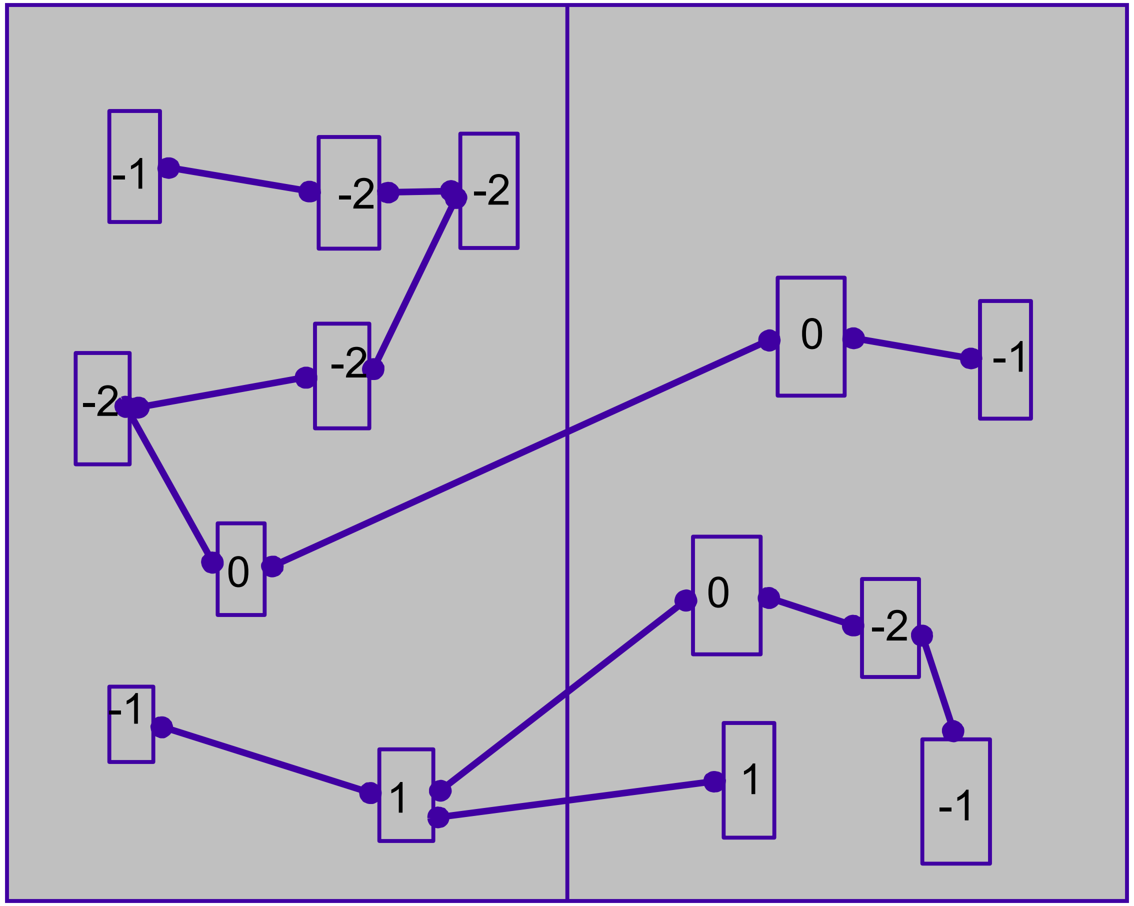
FM Algorithm Walkthrough – 2



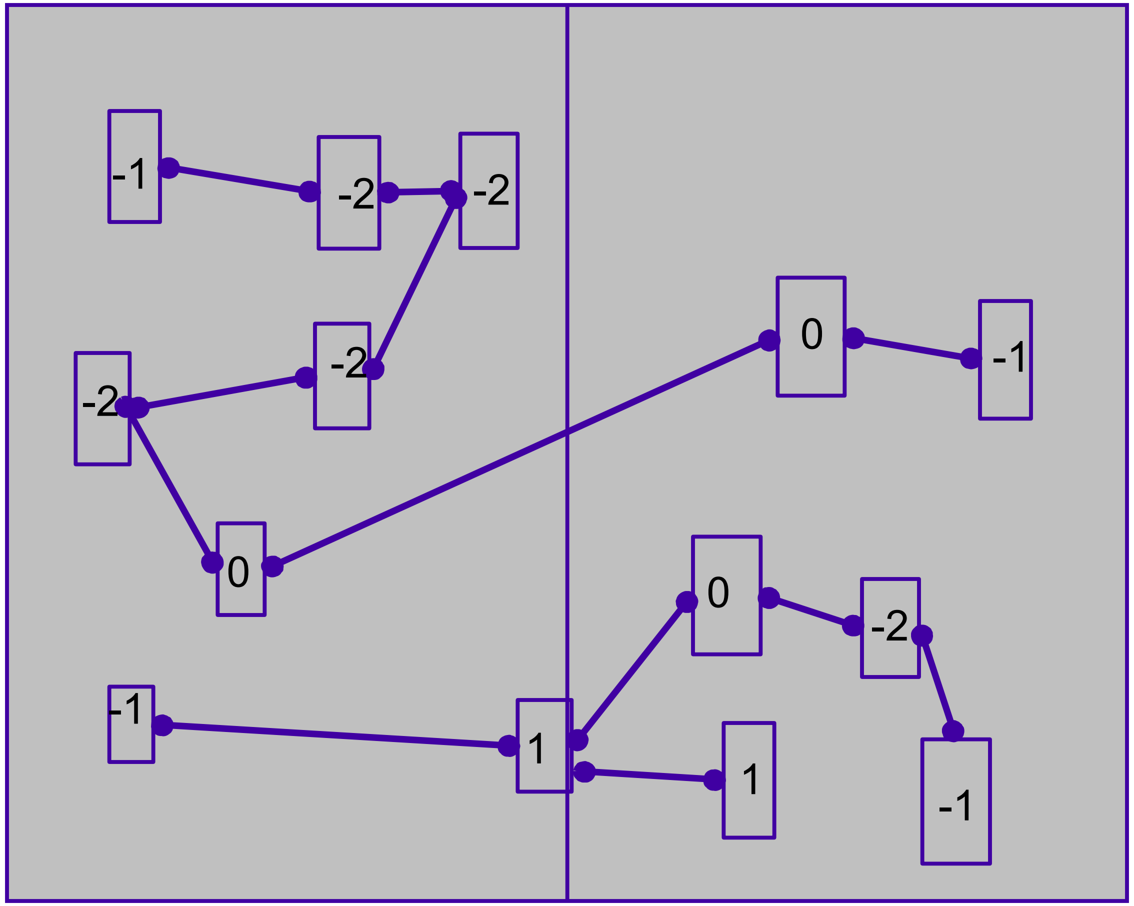
FM Algorithm Walkthrough – 3



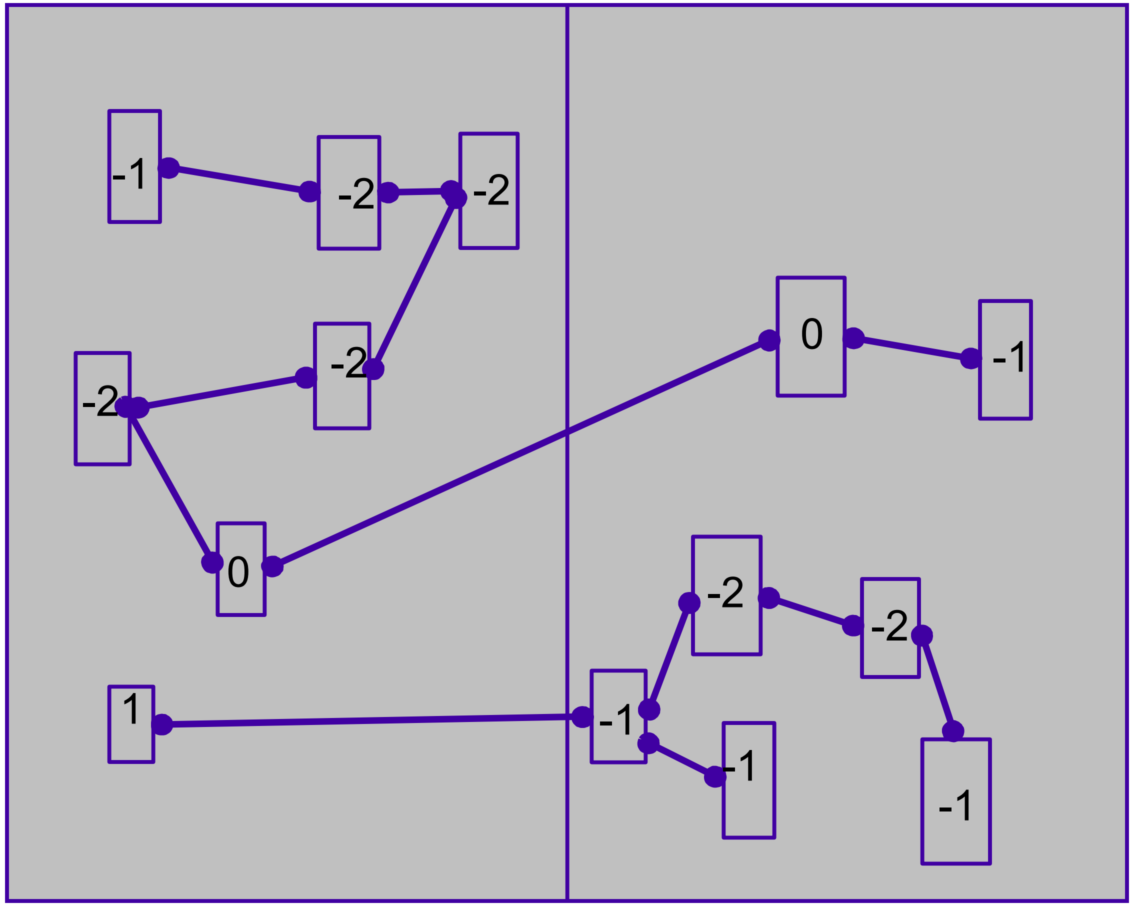
FM Algorithm Walkthrough – 4



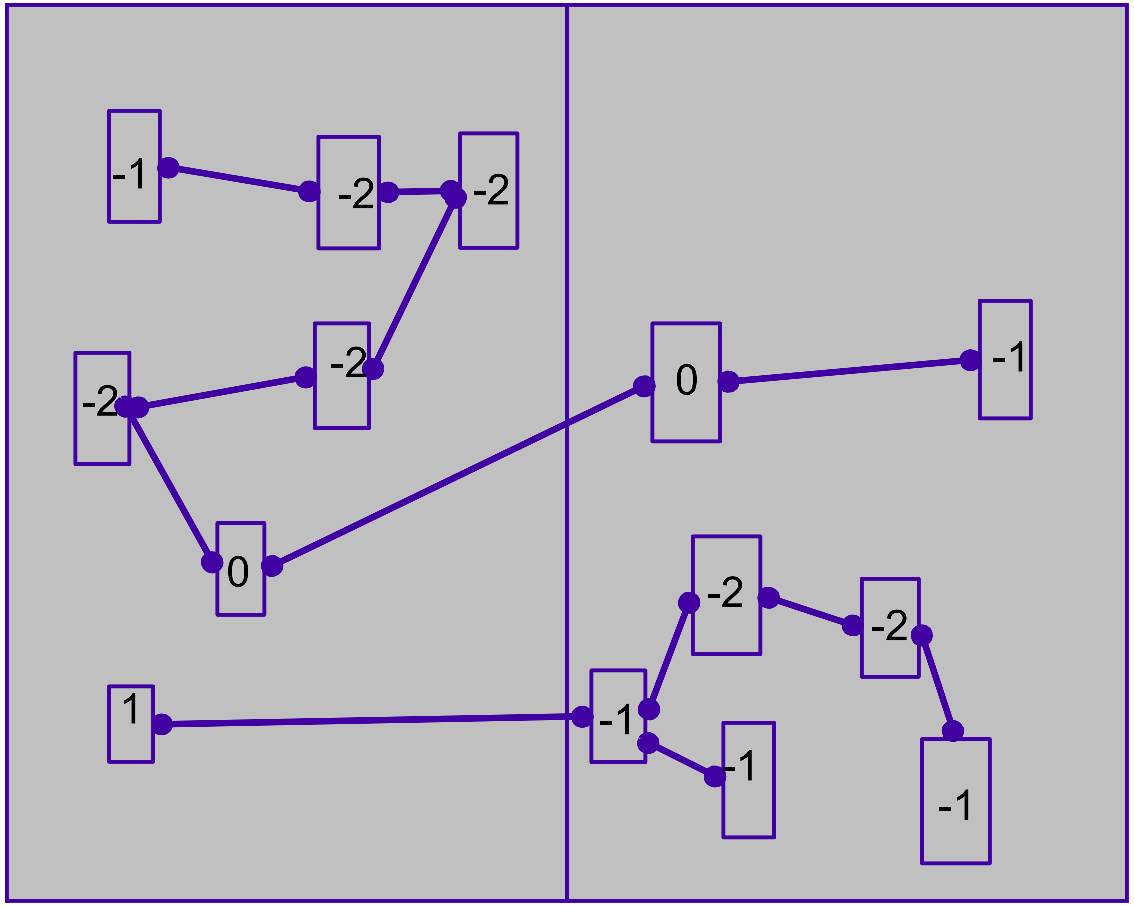
FM Algorithm Walkthrough – 5



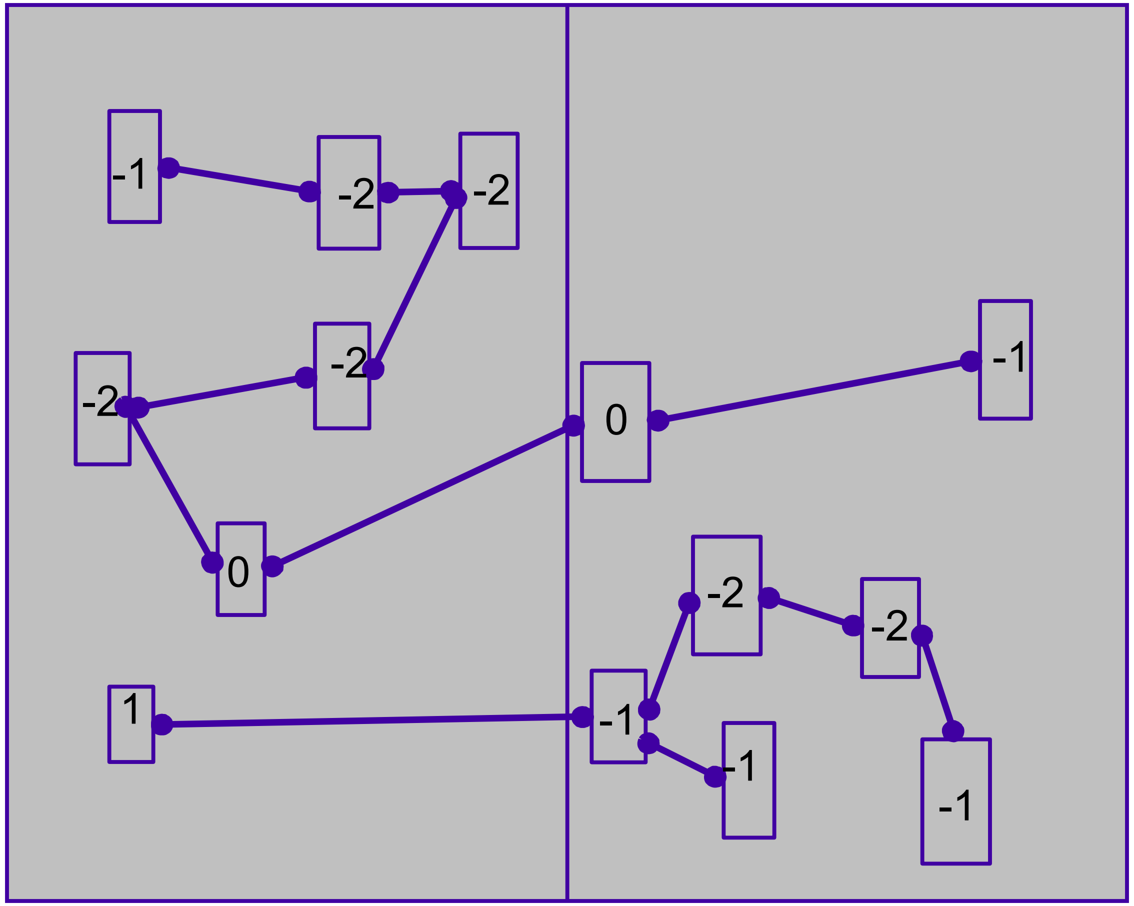
FM Algorithm Walkthrough – 6



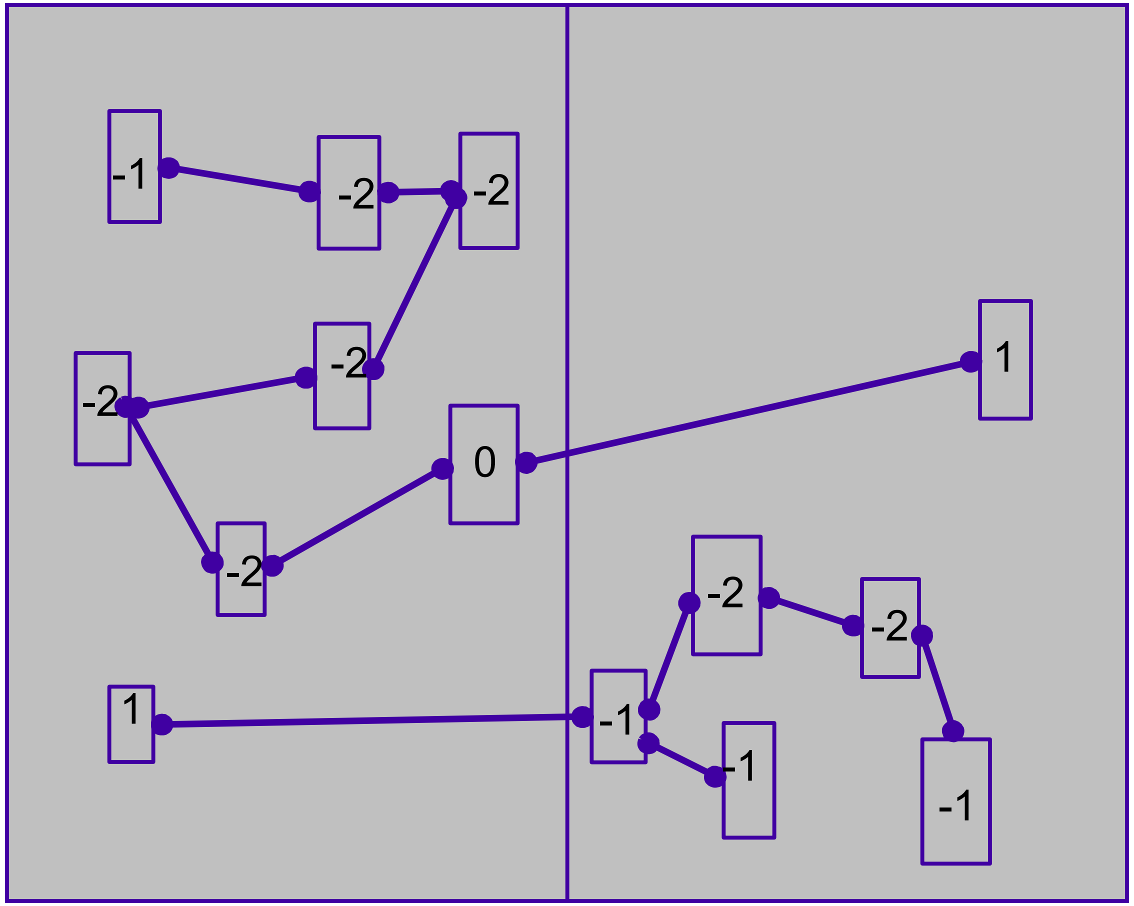
FM Algorithm Walkthrough – 7



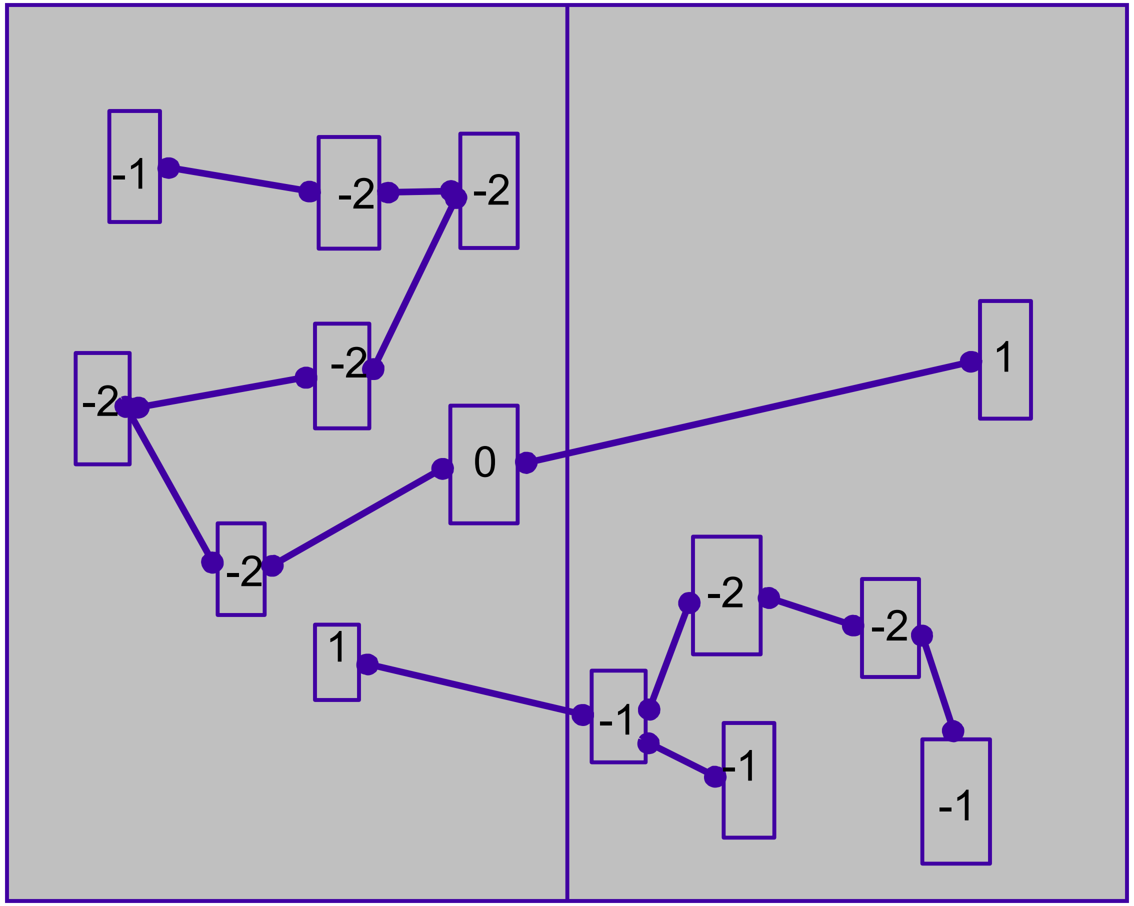
FM Algorithm Walkthrough – 8



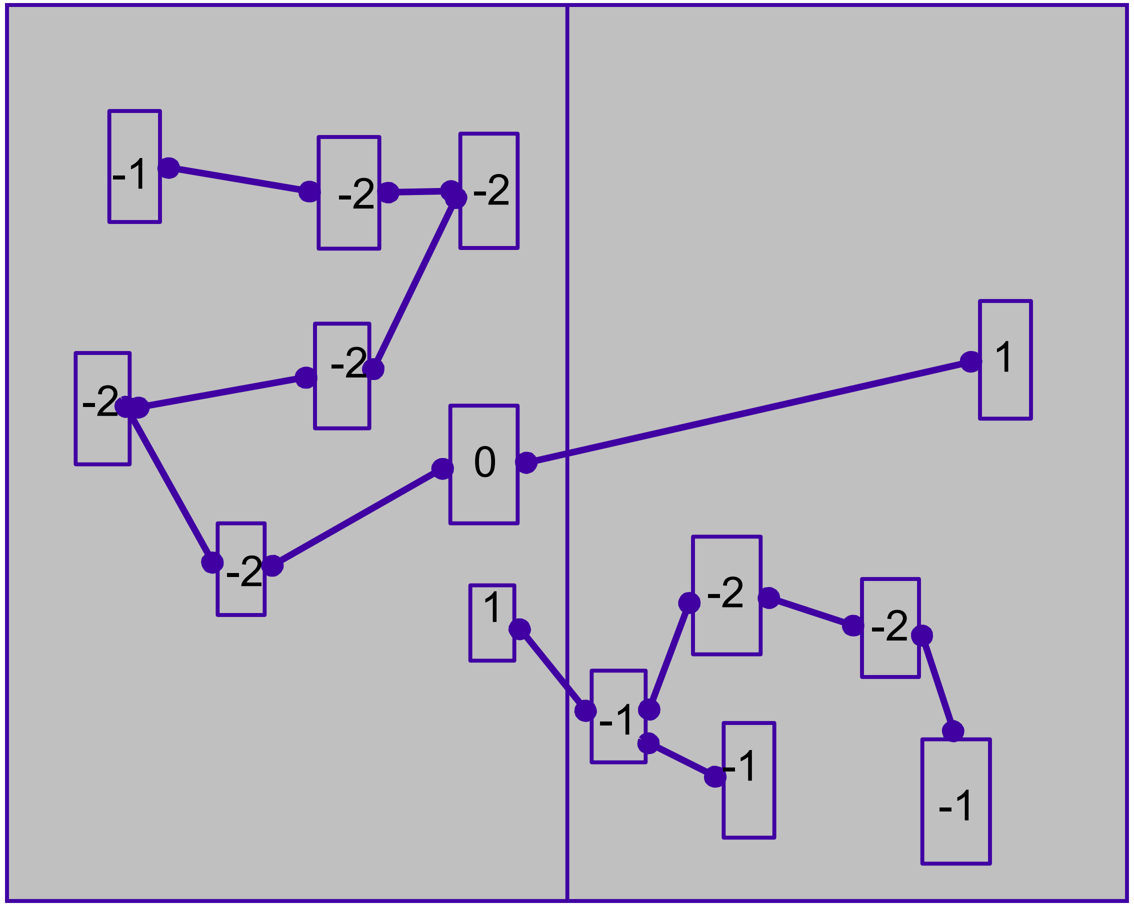
FM Algorithm Walkthrough – 9



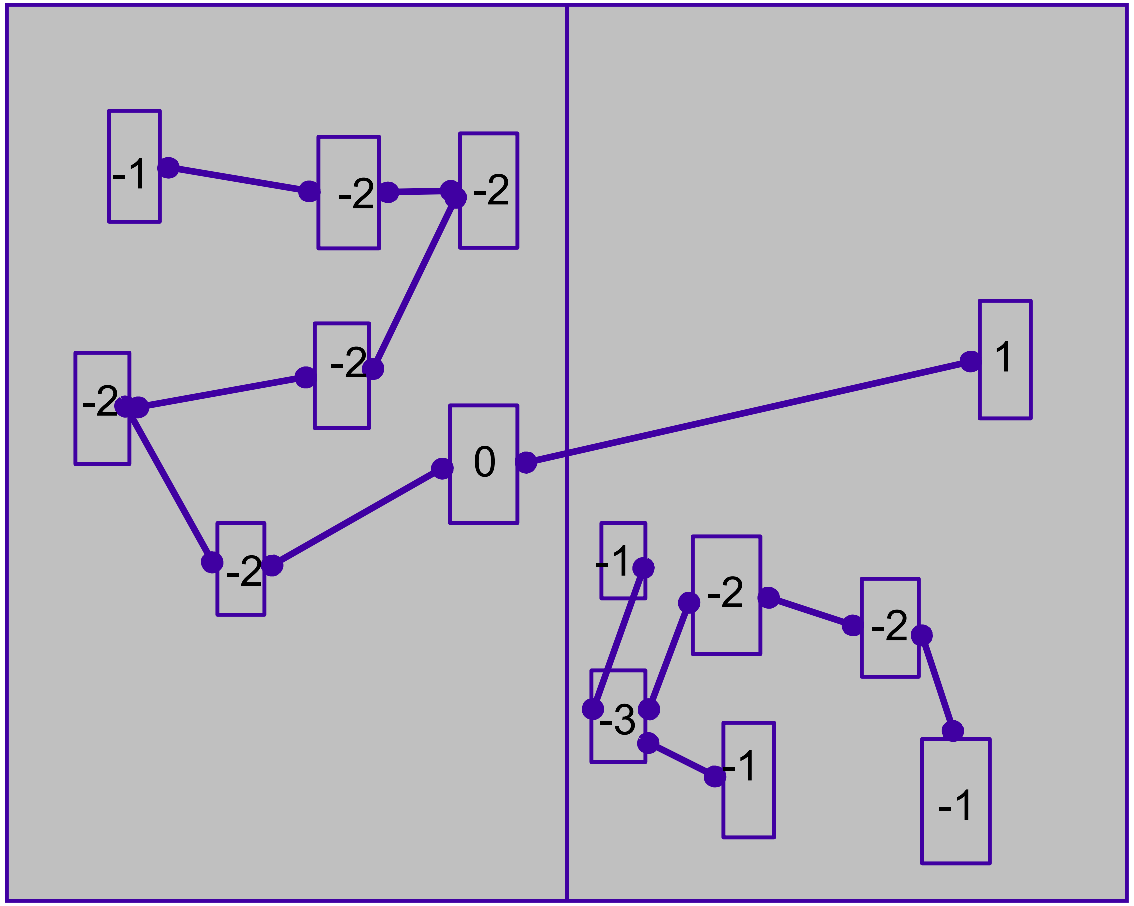
FM Algorithm Walkthrough – 10



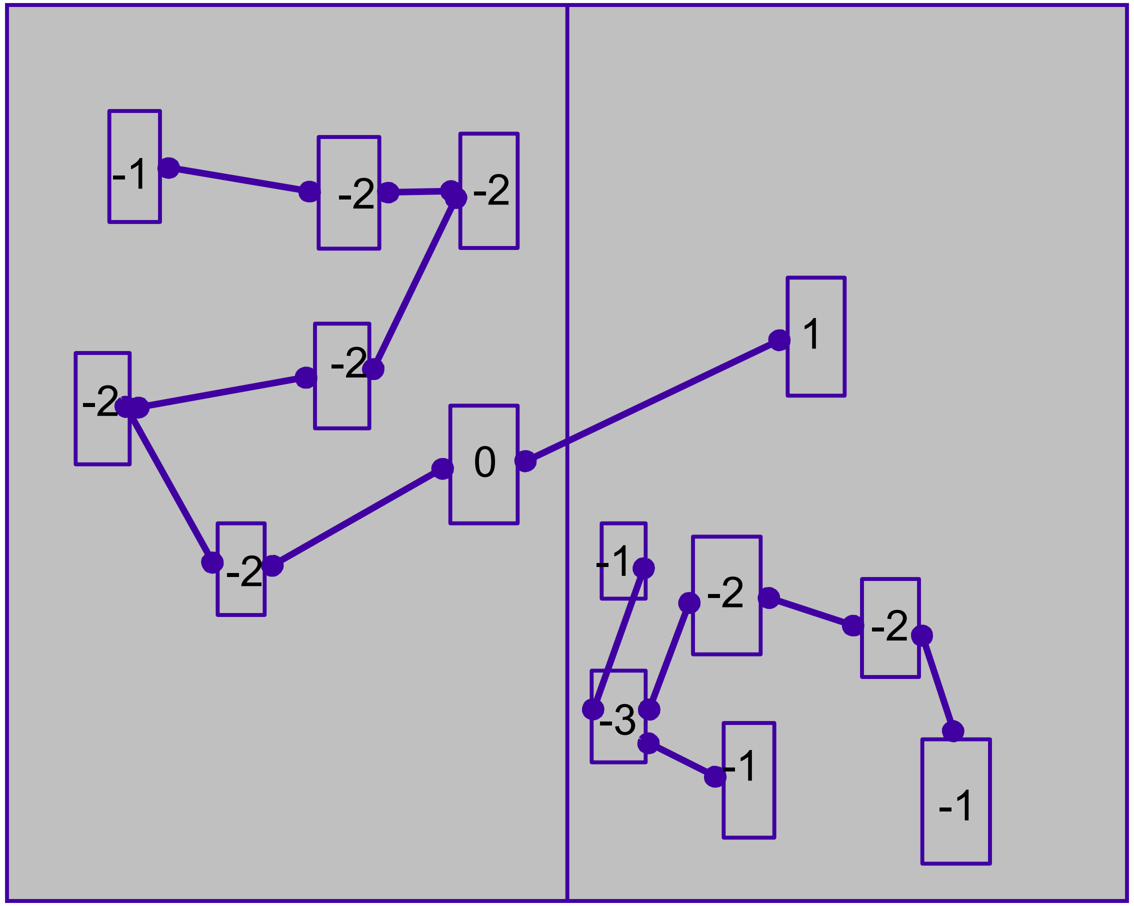
FM Algorithm Walkthrough – 11



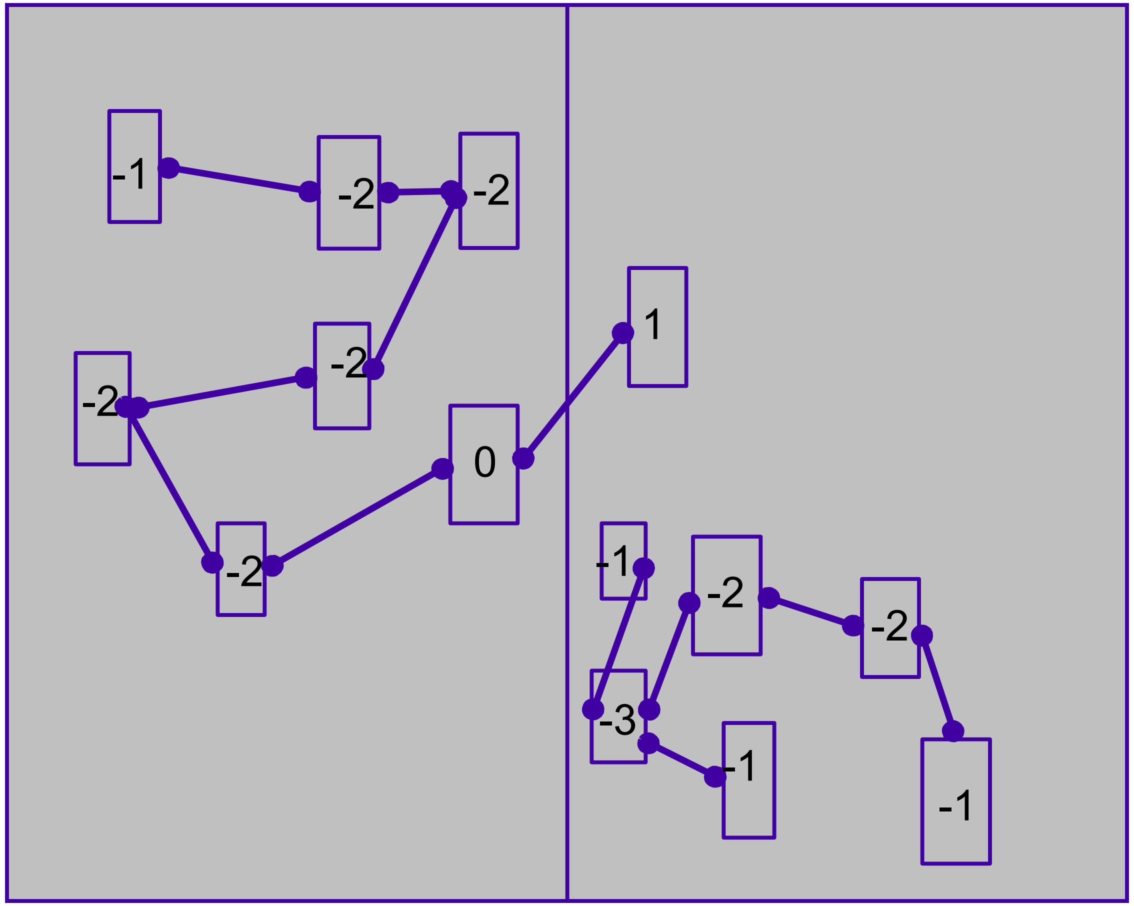
FM Algorithm Walkthrough – 12



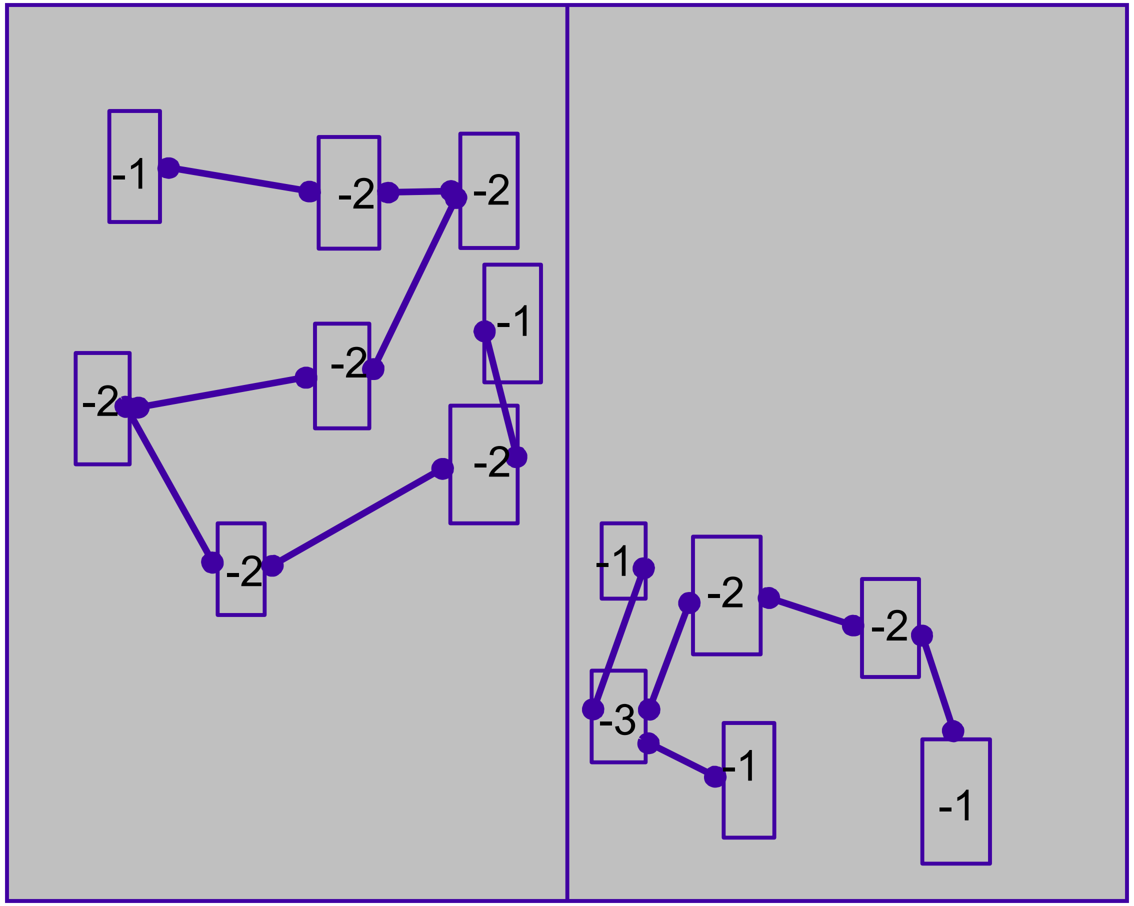
FM Algorithm Walkthrough – 13



FM Algorithm Walkthrough – 14



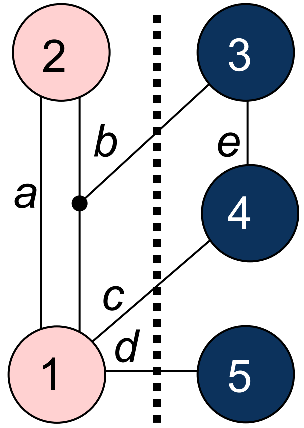
FM Algorithm Walkthrough – 15



Implementation Details

Gain $\Delta g(c)$ for cell c

Netlist: a {1, 2}, b {1, 2, 3}, c {1, 4}, d {1, 5}, e {3, 4}



$\Delta g(c) = FS(c) - TE(c)$,

where the "moving force" $FS(c)$ is the number of nets connected to c but not connected to any other cells within c's partition, i.e., cut nets that connect only to c, and

the "retention force" $TE(c)$ is the number of *uncut* nets connected to c.

The higher the gain $\Delta g(c)$, the higher is the priority to move the cell c to the other partition.

Implementation Details (cont'd)

Gain $\Delta g(c)$ for cell c

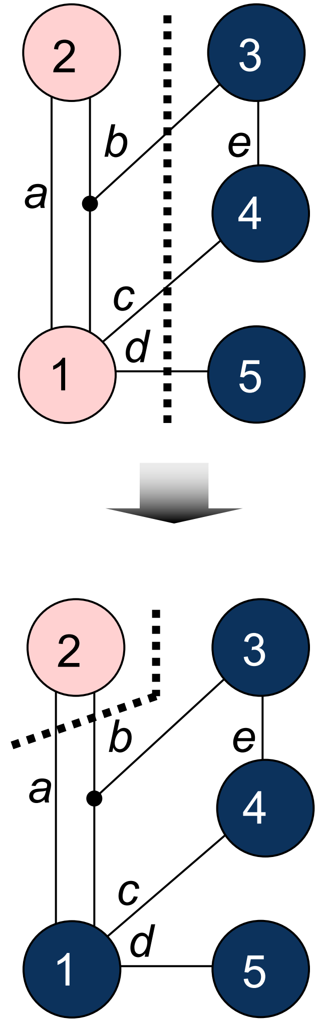
Netlist: a {1, 2}, b {1, 2, 3}, c {1, 4}, d {1, 5}, e {3, 4}

$\Delta g(c) = FS(c) - TE(c)$,

where the "moving force" $FS(c)$ is the number of nets connected to c but not connected to any other cells within c's partition, i.e., cut nets that connect only to c, and

the "retention force" $TE(c)$ is the number of *uncut* nets connected to c.

Cell 1:	$FS(1) = 2$	$TE(1) = 1$	$\Delta g(1) = 1$
Cell 2:	$FS(2) = 0$	$TE(2) = 1$	$\Delta g(2) = -1$
Cell 3:	$FS(3) = 1$	$TE(3) = 1$	$\Delta g(3) = 0$
Cell 4:	$FS(4) = 1$	$TE(4) = 1$	$\Delta g(4) = 0$
Cell 5:	$FS(5) = 1$	$TE(5) = 0$	$\Delta g(5) = 1$



Maximum Positive Gain

- Find the maximum positive gain G_m for each pass

$$G_m = \sum_{i=1}^{m} \Delta g_i$$

- The maximum positive gain G_m is the cumulative cell gain of m moves that produce a minimum cut cost.
- G_m is determined by the maximum sum of cell gains Δg over a prefix of m moves in a pass

Area Ratio Factor

- The *ratio factor* is the relative balance between the two partitions with respect to cell area
  It is used to prevent all cells from clustering into one partition.

- The ratio factor *r* is defined as $r = \dfrac{area(A)}{area(A) + area(B)}$

  where *area*(*A*) and *area*(*B*) are the total respective areas of partition *A* and partition *B*

Balanced Partition

- The balance criterion enforces the ratio factor
- To ensure feasibility, the maximum cell area $area_{max}(V)$ must be taken into account
- A partitioning of V into two partitions A and B is said to be balanced if

$$[\ r \cdot area(V) - area_{max}(V)\ ] \leq area(A) \leq [\ r \cdot area(V) + area_{max}(V)\ ]$$

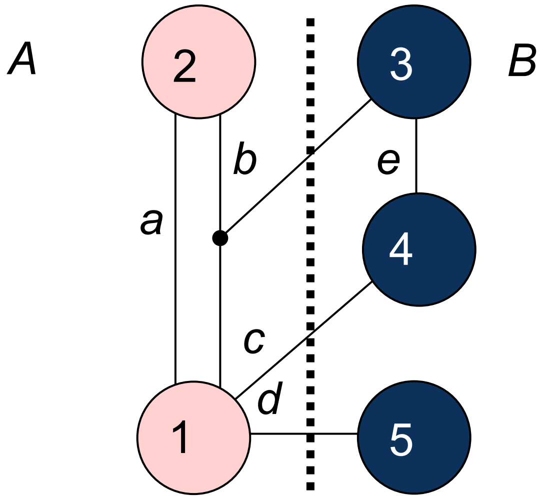
Base Cell

- A base cell is a cell c that has the greatest cell gain $\Delta g(c)$ among all free cells, and whose move does not violate the balance criterion

Base cell

Cell 1:	FS(1) = 2	TE(1) = 1	Δg(1) = 1
Cell 2:	FS(2) = 0	TE(2) = 1	Δg(2) = -1
Cell 3:	FS(3) = 1	TE(3) = 1	Δg(3) = 0
Cell 4:	FS(4) = 1	TE(4) = 1	Δg(4) = 0

FM Algorithm (One Pass)



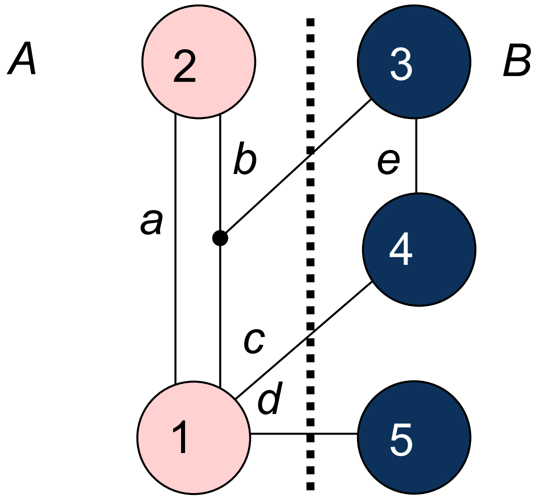
Given: Ratio factor $r = 0.375$
area(Cell_1) = 2
area(Cell_2) = 4
area(Cell_3) = 1
area(Cell_4) = 4
area(Cell_5) = 5

Step 0: Compute the balance criterion

$$[\, r \cdot area(V) - area_{max}(V) \,] \leq area(A) \leq [\, r \cdot area(V) + area_{max}(V) \,]$$

$$0.375 * 16 - 5 = 1 \leq area(A) \leq 11 = 0.375 * 16 + 5.$$

FM Algorithm (One Pass)



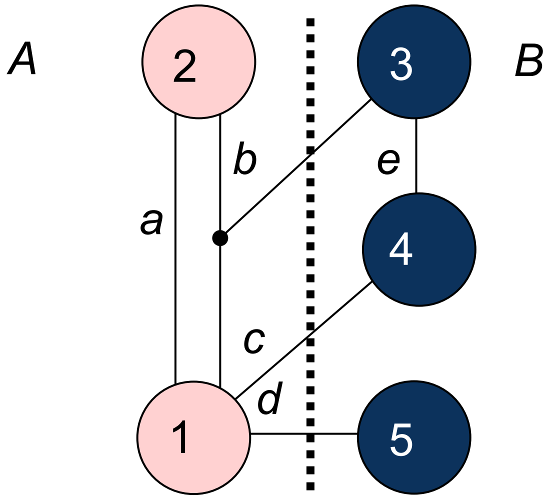
$FS(c)$: # *cut* nets that connect only to c

$TE(c)$: # *uncut* nets connected to c

Step 1: Compute the gains of each cell

Cell 1:	$FS(\text{Cell\_1}) = 2$	$TE(\text{Cell\_1}) = 1$	$\Delta g(\text{Cell\_1}) = 1$
Cell 2:	$FS(\text{Cell\_2}) = 0$	$TE(\text{Cell\_2}) = 1$	$\Delta g(\text{Cell\_2}) = -1$
Cell 3:	$FS(\text{Cell\_3}) = 1$	$TE(\text{Cell\_3}) = 1$	$\Delta g(\text{Cell\_3}) = 0$
Cell 4:	$FS(\text{Cell\_4}) = 1$	$TE(\text{Cell\_4}) = 1$	$\Delta g(\text{Cell\_4}) = 0$
Cell 5:	$FS(\text{Cell\_5}) = 1$	$TE(\text{Cell\_5}) = 0$	$\Delta g(\text{Cell\_5}) = 1$

FM Algorithm (One Pass)

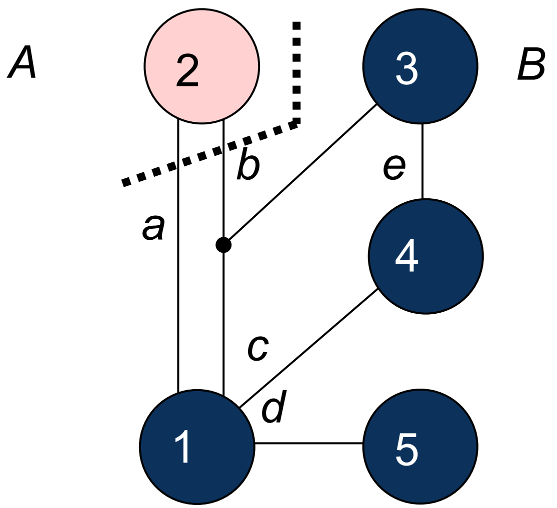


Cell 1:	FS(Cell_1) = 2	TE(Cell_1) = 1	Δg(Cell_1) = 1
Cell 2:	FS(Cell_2) = 0	TE(Cell_2) = 1	Δg(Cell_2) = -1
Cell 3:	FS(Cell_3) = 1	TE(Cell_3) = 1	Δg(Cell_3) = 0
Cell 4:	FS(Cell_4) = 1	TE(Cell_4) = 1	Δg(Cell_4) = 0
Cell 5:	FS(Cell_5) = 1	TE(Cell_5) = 0	Δg(Cell_5) = 1

Ratio factor r = 0.375
area(Cell_1) = 2 *area*(Cell_2) = 4
area(Cell_3) = 1 *area*(Cell_4) = 4
area(Cell_5) = 5

Step 2: Select the base cell
Possible base cells are Cell 1 and Cell 5
Balance criterion after moving Cell 1: *area*(*A*) = *area*(Cell_2) = 4
Balance criterion after moving Cell 5: *area*(*A*) = *area*(*Cell_1*) + *area*(*Cell_2*) + *area*(*Cell_5*) = 11
Both moves respect the balance criterion. Let's select Cell 1 (can also do Cell 5).

FM Algorithm (One Pass)



$FS(c)$: # *cut* nets that connect only to c

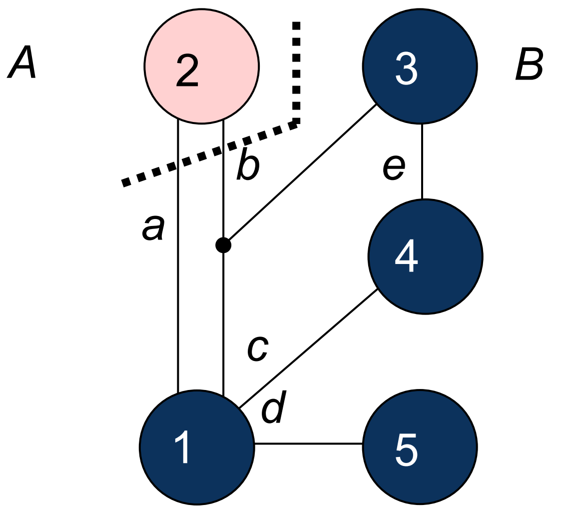
$TE(c)$: # *uncut* nets connected to c

Step 3: Fix base cell, update Δg values

Cell 2:	$FS(\text{Cell\_2}) = 2$	$TE(\text{Cell\_2}) = 0$	$\Delta g(\text{Cell\_2}) = 2$
Cell 3:	$FS(\text{Cell\_3}) = 0$	$TE(\text{Cell\_3}) = 1$	$\Delta g(\text{Cell\_3}) = -1$
Cell 4:	$FS(\text{Cell\_4}) = 0$	$TE(\text{Cell\_4}) = 2$	$\Delta g(\text{Cell\_4}) = -2$
Cell 5:	$FS(\text{Cell\_5}) = 0$	$TE(\text{Cell\_5}) = 1$	$\Delta g(\text{Cell\_5}) = -1$

After Iteration $i = 1$: Partition $A_1 = \{2\}$, Partition $B_1 = \{1,3,4,5\}$, with fixed cell $\{1\}$.

FM Algorithm (One Pass)



Lock Cell 1

Cell 2:	FS(Cell_2) = 2	TE(Cell_2) = 0	Δg(Cell_2) = 2
Cell 3:	FS(Cell_3) = 0	TE(Cell_3) = 1	Δg(Cell_3) = -1
Cell 4:	FS(Cell_4) = 0	TE(Cell_4) = 2	Δg(Cell_4) = -2
Cell 5:	FS(Cell_5) = 0	TE(Cell_5) = 1	Δg(Cell_5) = -1

Next iteration …

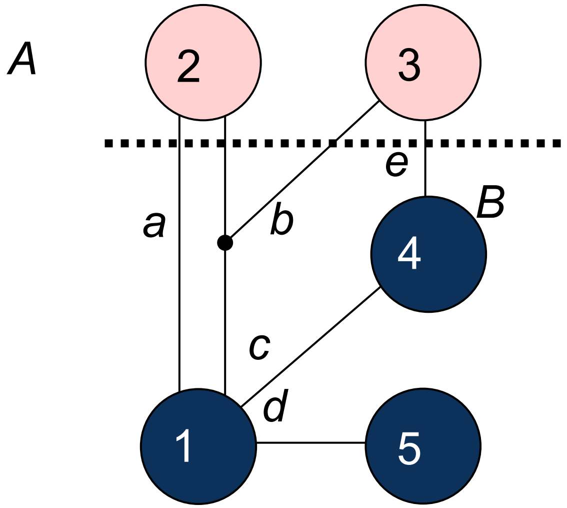
Cell *2* has maximum gain $\Delta g_2 = 2$, $area(A) = 0$, balance criterion is violated

Cell *3* has next maximum gain $\Delta g_2 = -1$, $area(A) = 5$, balance criterion is met

Cell *5* has next maximum gain $\Delta g_2 = -1$, $area(A) = 9$, balance criterion is met

Move cell *3*, updated partitions: $A_2 = \{2,3\}$, $B_2 = \{1,4,5\}$, with fixed cells {1,3}

FM Algorithm (One Pass)



Lock Cell 1 and Cell 3

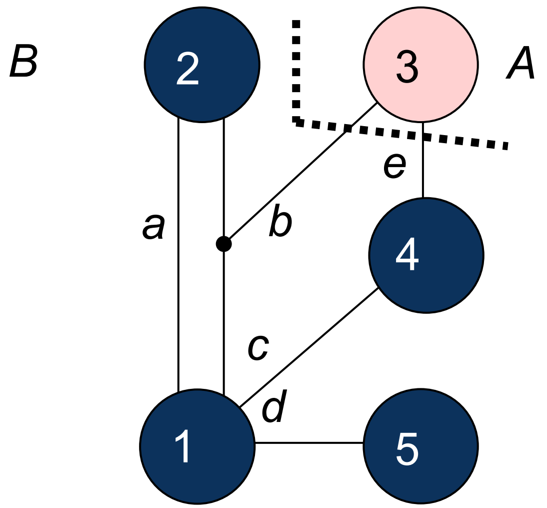
Cell 2: $\Delta g(\text{Cell\_2}) = 1$
Cell 4: $\Delta g(\text{Cell\_4}) = 0$
Cell 5: $\Delta g(\text{Cell\_5}) = -1$

Next iteration …

Cell *2* has maximum gain $\Delta g_3 = 1$, $area(A) = 1$, balance criterion is met.

Move cell *2*, updated partitions: $A_3 = \{3\}$, $B_3 = \{1,2,4,5\}$, with fixed cells $\{1,2,3\}$

FM Algorithm (One Pass)



Lock Cell 1, Cell 3, and Cell 2

Cell 4: $\Delta g(\text{Cell\_4}) = 0$

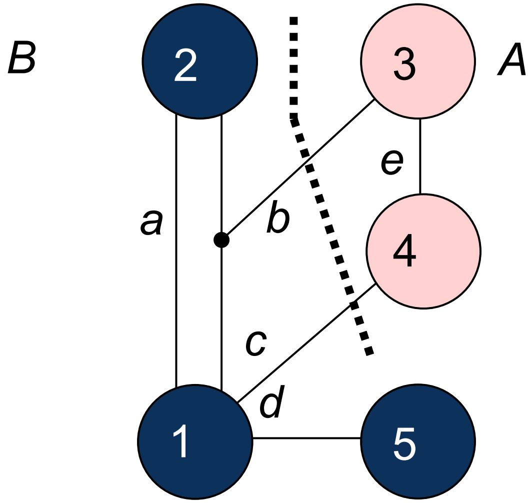
Cell 5: $\Delta g(\text{Cell\_5}) = -1$

Next iteration …

Cell 4 has maximum gain $\Delta g_4 = 0$, $area(A) = 5$, balance criterion is met.

Move cell 4, updated partitions: $A_4 = \{3,4\}$, $B_3 = \{1,2,5\}$, with fixed cells $\{1,2,3,4\}$

FM Algorithm (One Pass)



Lock Cell 1, Cell 3, Cell 2, and Cell 4

Cell 5: $\Delta g(\text{Cell\_5}) = -1$

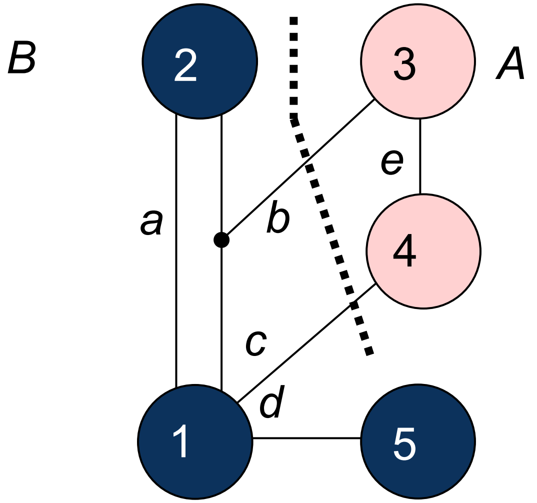
Next iteration …

Cell 5 has maximum gain $\Delta g_5 = -1$, $area(A) = 10$, balance criterion is met.

Move cell 5, updated partitions: $A_4 = \{3,4,5\}$, $B_3 = \{1,2\}$, all cells {1,2,3,4,5} fixed.

FM Algorithm (One Pass)

Ratio factor r = 0.375
area(Cell_1) = 2 *area*(Cell_2) = 4
area(Cell_3) = 1 *area*(Cell_4) = 4
area(Cell_5) = 5



End of each pass: find best move sequence $c_1 \ldots c_m$

$G_1 = \Delta g_1 = 1$

$G_2 = \Delta g_1 + \Delta g_2 = 0$

$G_3 = \Delta g_1 + \Delta g_2 + \Delta g_3 = 1$

$G_4 = \Delta g_1 + \Delta g_2 + \Delta g_3 + \Delta g_4 = 1$

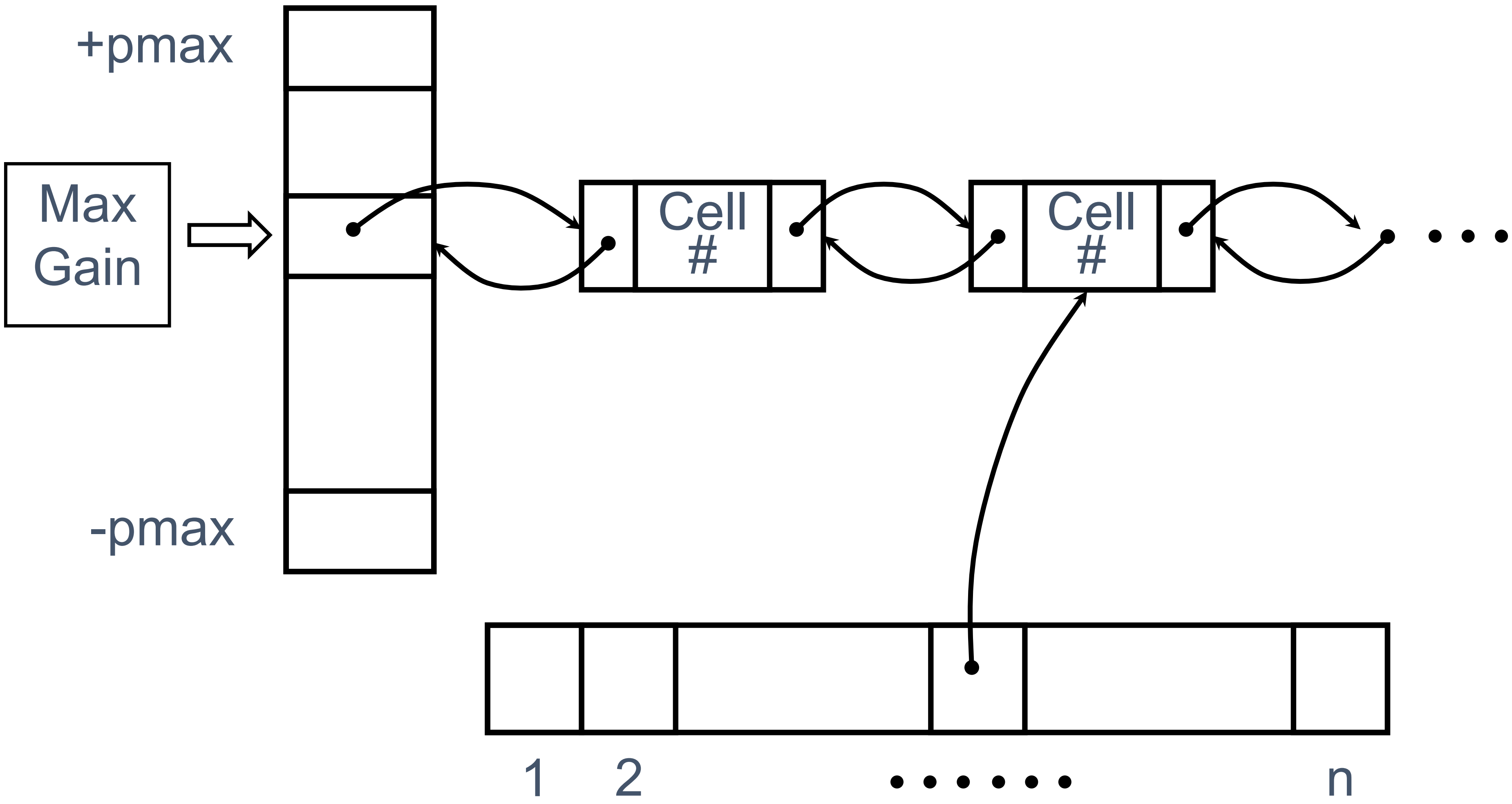
$G_5 = \Delta g_1 + \Delta g_2 + \Delta g_3 + \Delta g_4 + \Delta g_5 = 0.$

Maximum positive cumulative gain $G_m = \sum_{i=1}^{m} \Delta g_i = 1$ found in iterations 1, 3 and 4.

The move prefix m = 4 is selected due to the better balance ratio (*area*(*A*) = 5); the four cells 1, 2, 3 and 4 are then moved.

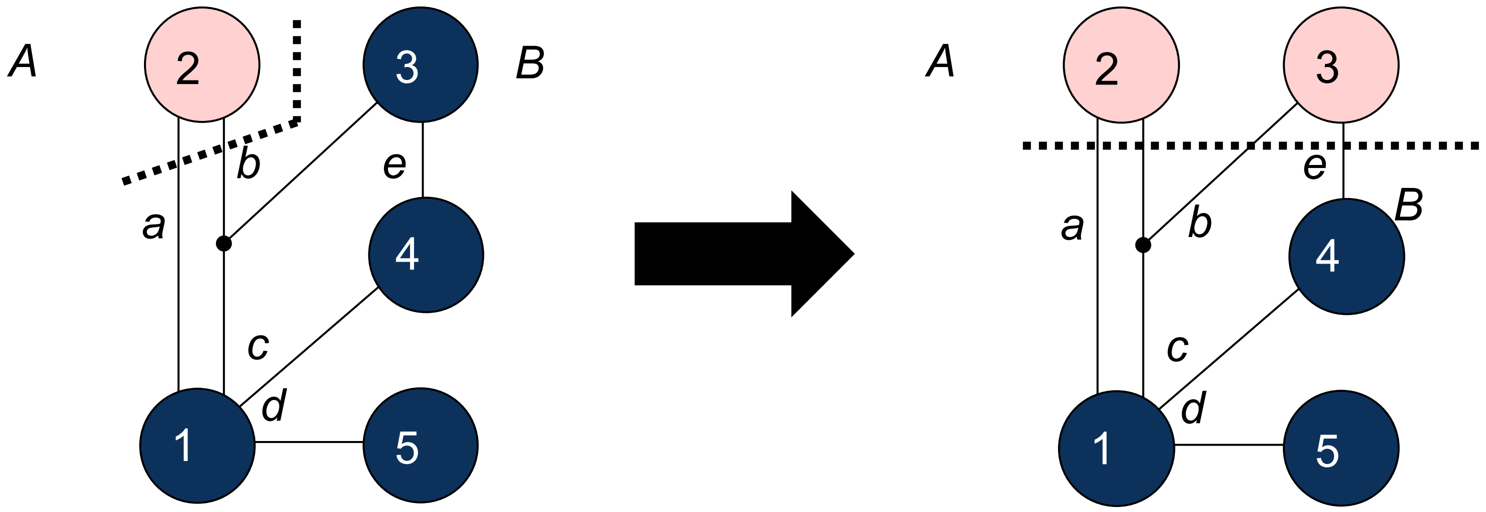
Result of Pass 1: Current partitions: *A* = {3,4}, *B* = {*1,2,5*}, cut cost reduced from 3 to 2.

Gain-based Bucket Data Structure



Incremental Update

- Which gain values need update after moving Cell 3?



Time Complexity of FM Algorithm

- **For each pass**
  - Constant time to find the best vertex to move
  - After each move, time to update gain buckets is proportional to degree of vertex moved
  - Total time is O(p), where *p* is total number of pins
- **Number of passes is usually small**
  - Gain values converge very quickly

Programming Assignment #1

- **Implement FM partitioning algorithm**
  - https://github.com/tsung-wei-huang/ece5960-physical-design/tree/main/PA1
- **Two checkpoint dues, 9/7 and 9/14 23:59 PM**
  - https://github.com/tsung-wei-huang/ece5960-physical-design/issues/2
- **Final Due on 9/21 (Wed) 23:59 PM**
  - Upload your solutions to twhuang-server-01.ece.utah.edu
    - Account: ece6960-fall22
  - Place your source code + README under PA1/your_uid/
  - README should contain instruction to compile & run your code

Programming Assignment #1 (cont'd)

- **In addition to source code + README, upload a report with:**
  - A table showing your results of each benchmark
  - A section discussing what challenges you encounter
  - A section discussing how you overcome those challenges
    - Also discuss unsolved challenges
- **The report needs to be just a one- or two-page pdf**
  - No need to be lengthy …
- **Upload your report to the class GitHub page**
  - https://github.com/tsung-wei-huang/ece5960-physical-design/issues/1
  - Due 9/21 23:59 PM

In-class Presentation: 9/14

- **Circuit partition research presentation on 9/14 (in class)**
  - George Karypis and Vipin Kumar, "Multilevel k-way Hypergraph Partitioning," *1999 ACM/IEEE Design Automation Conference (DAC)*
  - Honghua Yang and Martin Wong, "Efficient Network Flow Based Min-Cut Balanced Partitioning," *1994 ACM/IEEE International Conference on Computer-aided Design (ICCAD)*
  - Masahiro Tanaka, Kenjiro Taura, Toshihiro Hanawa, Kentaro Torisawa, "Automatic Graph Partitioning for Very Large-scale Deep Learning," *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS 2021)*
- https://github.com/tsung-wei-huang/ece5960-physical-design/issues/9

Summary

- **We have discussed FM partitioning algorithm**
  - Inherited greedy movement ideas from KL algorithm
  - Extended to hypergraph
  - Incorporated area constraint (balanced partition)
- **We have discussed implementation details of FM algorithm**
  - Bucket list data structure to keep track of gains and cells
  - Incremental update for gain values