Max Flow, Matching, and Min Cost Max Flow

Topics We Will Cover:

- Designing Max Flow & Min Cut Solutions
- Variants: Lower Bound on flow graphs, Circular flow
- Different Covering Problems solvable with BPM
- Weighted BPM Problem
- Designing MCMF solutions
- MCMF Cost Minimization instead of Flow Maximization
- Complexity Analysis of different Flow, BPM, MCMF problems

A Flow Graph and Max Flow

- A source and a sink
- Each edge has a capacity denoting amount of flow passing through it
- Max Flow = Maximum flow we can send from the source to the sink
- Max Flow = Number of edge disjoint paths from source to sink if all edges have unit capacity

League Game

- N team league (N <= 100)
- In each match, the winner gets 1 point and loser gets 0 point
- Given the current point table and the number of remaining matches between each pair
- Find out if your team can win the league in the best possible case?
- If multiple teams have the highest points, all of them become winners of the league.

Min Cut = Max Flow

- Cut Set = A Set of edges which when removed, the source and sink gets disconnected
- Cost of Cut Set = Summation of the capacities of edges in the cut set
- Min Cut = The cut set with the minimum cost
- Min Cut Cost = Max Flow of the graph

Project Selection Problem

- Given N Projects and M instruments
- i-th project has profit P[i]
- j-th instrument has cost C[i]
- i-th project requires set S[i] of instruments
- If an instrument is bought once, it can be used for any number of projects
- Maximize (Profit - Cost)

Project Selection Problem Solution

- Let A be the set of projects we decide to do
- Let B be the set of instruments we need to buy
- Maximize (profit(A) - cost(B))
- But min cut can solve minimization problems

Project Selection Problem Solution

- Let E be the complete set of projects
- Let A be the set of projects we decide to do
- Let X be the set of projects we decide to sacrifice doing
- E = A union X
- Let B be the set of instruments we need to buy
- Maximize (profit(A) - cost(B)
- => Maximize (profit(E) - profit(X) - cost(B))
- => Minimize (Profit(X) + cost (B))

Project Selection Problem Solution

- A cut set denotes a valid solution i.e. a valid choice of X and B
- Min Cut denotes the best possible valid solution
- Maximization -> Minimization -> Min Cut

Image Segmentation Problem

- Given N pixels
- Each pixel needs to be colored in black or white
- Coloring i-th pixel in black gives profit B[i]
- Coloring i-th pixel in white gives profit W[i]
- Some pairs of pixels are related
- If u and v are related, and assigned different color, then penalty is P[u][v]
- Maximize (profit - penalty)

Image Segmentation Problem Solution

- Maximization -> Minimization -> Min Cut
- Let R be pixels colored in white
- Let S be pixels colored in black
- Maximize (W(R) + B(S) - penalty)
- => Maximize (W(All) - W(S) + B(All) - B(R) - penalty)
- => Minimize (W(S) + B(R) + penalty)

Binary Matrix

- Given the dimensions of a nxm binary matrix
- Given the row sums and the column sums
- We need to reconstruct the matrix

Binary Matrix (Continued)

- Reconstruct the Lexicographically Smallest Binary Matrix
- Run n x m Max flows
- Can we do it with one max flow?

Problem: Bring them here

- Given an undirected graph with a source and a destination
- Travelling an edge takes one day
- We need to send K soldiers from the source to the destination in minimum number of days
- An edge can be used by only one soldier each day

Lower Bound on Edges

- Given an acyclic flow graph with one source and one sink
- Each edge has an upper bound (capacity) and a lower bound
- We have to satisfy the bounds by passing flow from source to sink
- No maximization of flow

MinFlow and MaxFlow satisfying Lower Bound

- Maximize the flow while satisfying both lower and upper bounds
- Minimize the flow
- Binary Search x Max Flow Solution
- Constant number of Max flow Solution

Circular Flow Satisfying with Lower Bounds

- Given a flow graph with no source and no sink
- Satisfy upper and lower bounds on the edges
- Incoming flow(u) = outgoing flow(u)
- Circular flow

Cut Set Cost Definition Change

- Given a flow graph with a source and a sink
- Find the min cut where,
- Cost of cut set = average capacity of the edges in the cut set

BPM

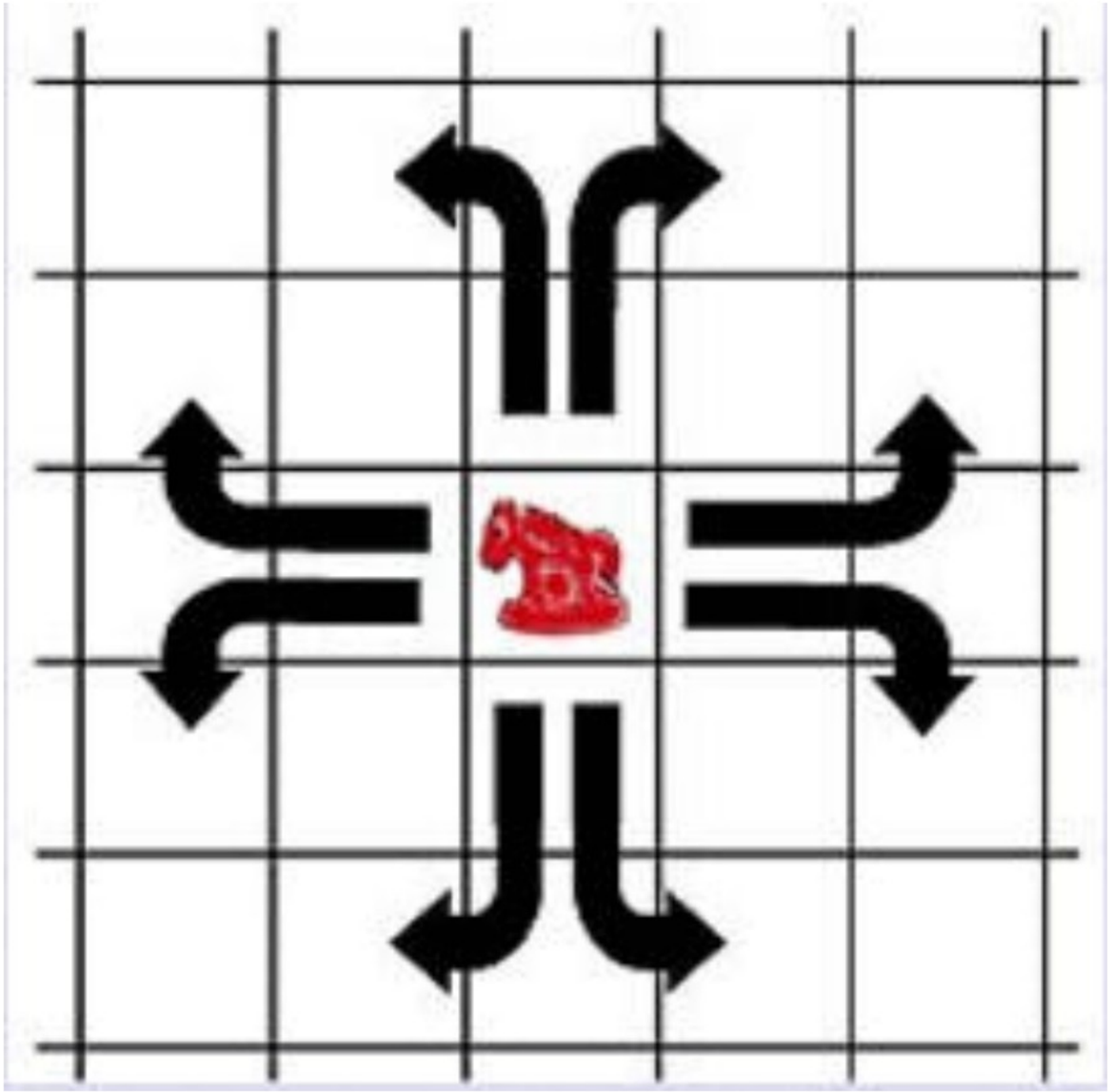
- Given a bipartite graph
- Pick largest set of edges where each vertex is part of at most one edge in the set
- i.e. we want to find the maximum matching

Covering Problems Solvable in Polynomial Time

- Maximum Independent Set in Bipartite Graph
  - Largest set of nodes who do not have any edge between themselves
  - Solution: V - Max Matching
- Minimum Vertex Cover in Bipartite Graph
  - Smallest set of nodes where at least one end-point of each edge is present
  - Solution: Max Matching

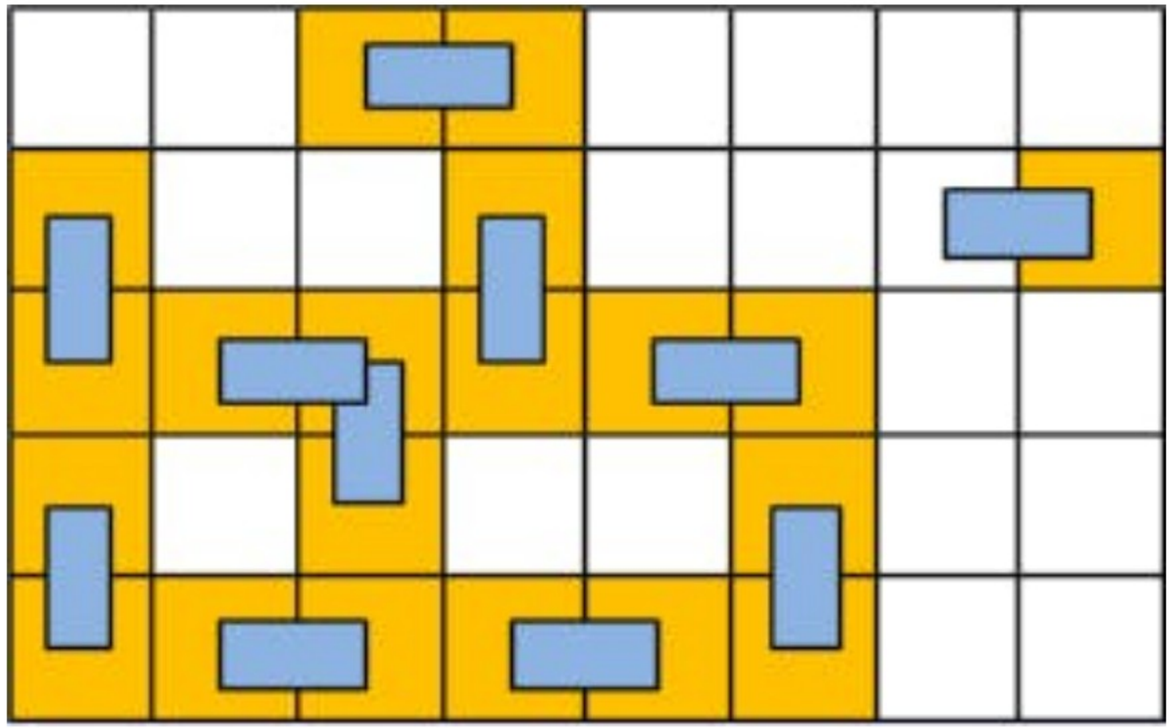
Covering Problems Solvable in Polynomial Time

- Minimum Edge Cover in General Graph
  - Smallest set of edges where each vertex is end-point of at least one edge
  - V - matching (if edge cover exists)
- Minimum Path Cover (Vertex Disjoint) in DAG
  - Minimum number of vertex disjoint paths that visit all nodes
- Minimum Path Cover (Vertex not-disjoint) in General Graph
  - Minimum number of paths that visit all nodes



Knights on a Chessboard

- Given mxn chessboard, and k blocked cells
- Place the maximum number of knights in the board who do not attack eachother
- The knight moves are shown in the left figure



Dominoes

- Given an n x m grid with k golden cells
- We need to cover all the golden cells with minimum number of 2x1 dominoes
- Dominoes can overlap
- 11 dominoes were used in the picture

Muddy Fields

- Given a nxm grid (field)
- Each cell is either good or muddy
- We need to cover all muddy cells using boards of unit width and any length
- We can place the boards vertically or horizontally
- We cannot cover any good cell
- Find out the minimum number of boards required

Soldiers & Archers

- You have A archers and B soldiers initially
- There are C enemies
- The i-th enemy becomes active on the i-th day and remains active for eternity
- The i-th enemy can be neutralized by the X[i]-th archer or the Y[i]-th soldier
- No two enemies have the exact same pair of X and Y values
- An archer or soldier can be used for neutralizing multiple enemies
- After the i-th enemy becomes active, you need to find out the minimum total number of archers and soldiers you need
- This is an online problem, where you have to output the solution for the i-th enemy for getting the information of the (i+1)-th enemy
- A, B <= 10^3, C <= 10^4

Minimum Path Cover in DAG (Vertex Disjoint)

- Given a DAG
- Put at most one soldier in a node
- A soldier can traverse a path
- Each node should be visited by exactly one soldier
- Use minimum number of soldiers

Solution: Minimum Path Cover in DAG (Vertex Disjoint)

- Create a bipartite graph
- Each side contains the same set of nodes as the original graph
- If u->v edge existed in original graph, give an edge from u of the left part to v of the right part.
- Solution: number of nodes in original graph - matching

Minimum Path Cover in general graph

- Given a general graph
- Put at most one soldier in a node
- A soldier can traverse a path
- Each node should be visited by at least one soldier
- Use minimum number of soldiers

Solution: Minimum Path Cover in general graph

- Find out the strongly connected components of the original graph
- Using SCCs as nodes, create a DAG (SCC Graph)
- Create a bipartite graph where each side contains the same set of nodes as the SCC graph
- If v is reachable from u in SCC graph, give an edge from u of the left part to v of the right part.
- Solution: number of nodes in SCC graph - matching

Minimum Path Cover in General Graph (Vertex Disjoint)

- Not solvable in polynomial time
- Previous solution using SCC will not work
- Why: Different parts of a single SCC can be used by multiple soldiers in the optimal solution

Weighted BPM (Problem: The Great Merger)

- Given two rooted trees
- You can add nodes to any of the trees
- You cannot change the roots
- Target: Make the two trees have exactly the same shape
- Do it with minimum number of additions
- Number of nodes <= 100

Complexity Analysis of different Flow and Matching Algorithms

- Dinic Max Flow: O(V^2 x E)
- Dinic on unit graph: O(sqrt(V) x E)
- Kuhn BPM: O(VE)
- Hopcroft Karp BPM: O(sqrt(V) x E)
- HK BPM complexity = Dinic BPM complexity
- Hungarian Weighted BPM: O(V^3)

Min cost max flow

- Given a flow graph where each edge has a cost associated with it
- Cost[i] denotes the cost of passing one flow through the i-th edge
- We need to first maximize the flow and then minimize the cost.

Shortest Cycle

- Given an undirected graph with a source and a destination
- Find the shortest path starting from the source and ending at the source which visits the destination
- No edge can be visited twice in the path

Olympiad in Programming and Sports

- N students (N <= 3000)
- A[i] = i-th student's programming capability
- B[i] = i-th student's sports capability
- P students in programming team
- S students in sports team (S+P = N)
- maximize the summation of ability of both teams

Solution Complexity: Olympiad Problem

- MCMF Complexity: min(VE x T(V, E), F x T(V,E))
- T(V, E) = Complexity of shortest path algorithm, F = Flow passed
- MCMF(Bellman Ford) complexity: min (VE x VE, F x VE)
- V = 3000, E = 9000, F = 3000
- Final Worst Case = 8.1 x 10^10
- MCMF(SPFA) average complexity: min (VE x E, F x E)
- Final Average Case = 2.7 x 10^7
- SPFA = Shortest Path Faster Algorithm

MCMF cost minimization instead of flow

- N different colors, M boxes
- A[i] number of balls of i-th color
- B[i][j] = profit of putting a ball of i-th color into the j-th box
- A box can contain at most one ball of a particular color
- j-th box can contain C[j] balls without any cost
- X penalty is added for every extra ball that is kept in a box exceeding its capacity
- Maximize profit. No need for putting every ball in a box