Algorithms

Topics: Minimum Cost Maximum Flow

Minimum Cost Maximum Flow

TUTORIAL PROBLEMS

Minimum Cost flow problem is a way of minimizing the cost required to deliver maximum amount of flow possible in the network. It can be said as an extension of maximum flow problem with an added constraint on cost(per unit flow) of flow for each edge. One other difference in min-cost flow from a normal max flow is that, here, the source and sink have a strict bound on the limits on the flow they can produce or take in respectively, $B(s) > 0, B(t) < 0$. Intermediate nodes have no bounds or can be represented as $B(x) = 0$.

Cycle Cancelling Algorithm:
This algorithm is used to find min cost flow along the flow network. Pseudo code for this algorithm is provided below.

```
function: CostNetwork(Graph G, Graph Gf):
    Gc <- empty graph
    for i in edges E:
        if E(u,v) in G:
            cf(u,v) = c(u,v)
        else if E(u,v) in Gf:
            cf(u,v) = -c(u,v)

function: MinCost(Graph G):
    Find a feasible maximum flow of G using Ford Fulkerson and construct
```

```
residual graph(Gf)
    Gc = CostNetwork(G, Gf)
    while(negativeCycle(Gc)):
        Increase the flow along each edge in cycle C by minimum capacity in
the cycle C
        Update residual graph(Gf)
        Gc = CostNetwork(G,Gf)
    mincost = sum of Cij*Fij for each of the flow in residual graph
    return mincost
```

Negative cycle in cost network(G_c) are cycle with sum of costs of all the edges in the cycle is negative. They can be detected using Bellman Ford algorithm. They should be eliminated because, practically, flow through such cycles cannot be allowed. Consider a negative cost cycle, if at all flow has to pass through this cycle, the total cost is always reducing for every cycle completed. And so would result in an infinite loop in desire of minimizing the total cost. So, whenever a cost network includes a negative cycle, it implies, the cost can further be minimized (By flowing through other side of cycle instead of the side currently considered). A negative cycle once detected are removed by flowing a bottleneck capacity through all the edges in the cycle.

There are various applications of minimum cost flow problem. One of which is solving the minimum weighted bipartite matching. Bipartite graph(B) is a graph whose nodes can be divided into two disjoint sets$(P$ and $Q)$ and all the edges of graph joins a node in P to node in Q. Matching means that no two edges in final flow touch each other(share common node). It can be considered as a multi-source multi-destination graph. Convert such graphs to single source and single destination by creating a source node S and join all nodes in set P with S and a destination node T and join all nodes in set Q with T. Now, above algorithm can be applied to find min cost max flow in graph B.

Hungarian Algorithm:
A variant of weighted bipartite matching problem is known as assignment problem. In simple terms, assignment problem can be described as having N jobs and N workers, each worker does a job for particular cost. Also, each worker should be given only one job and each job should be assigned to only worker. This can be solved using Hungarian algorithm. Pseudocode for this problem is given below.

Input will be an $N \times N$ matrix showing cost charged by each worker for each job.

```
function: HungarianAlgorithm(Matrix C):
    Copy C into X
    for i from 1 to N:
        subtract elements in row(i) of X with min(row(i))
    for j from 1 to N:
        subtract elements in column(j) of X with min(column(j))
    L = minimum number of lines(horizontal or vertical) to join all 0s in X
    while L != N:
        M = minimum number among the cells that are not crossed by the
```

```
lines
            Subtract M from all the cells that are not crossed by lines
            Add M to all cells that have intersection of lines
            L = minimum number of lines(horizontal or vertical) to join all 0s
in X
      return FindMinCost(X,C)
```

FindMinCost does an optimal selection of 0s in matrix X such that N cells are selected and non of them lie in same row or column. The values in cells in C corresponding to selected cells in X, are added and are returned as answer for minimum cost that is to be calculated.

Contributed by: Vinay Kumar