Minimum Cost Flow Primal-Dual Algorithm

Presentation - CS602: Applied Algorithms

Ayyapu Ruthvika, Dhananjay Kejriwal, Mothika Gayathri Khyathi

Department of Computer Science and Engineering

Indian Institute of Technology Bombay

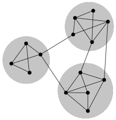
Contents

Terminology

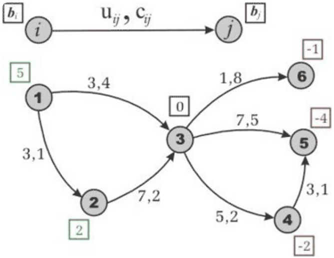
Transportation Network

$G = (V, E, u, c, b)$

- Directed graph $G = (V, E)$ where
- V is the set of n nodes and E is the set of m edges
- u_{ij} is the flow capacity for each edge $(i, j) \in E$ and
- c_{ij} is the cost per unit flow for each edge $(i, j) \in E$
- b_i is the supply (if positive)/demand (if negative) for each vertex $i \in V$



Example of the Transportation Network



In this we have 2 supply vertexes ($b_i > 0$), 3 demand vertexes ($b_i < 0$), and 1 transshipment node ($b_i = 0$).

Statement of the Problem

Minimum-cost flow problem

The minimum-cost flow problem (MCFP) is an optimization problem to find the cheapest possible way of sending a certain amount of flow through a transportation network.

Representing the flow on edge $(i, j) \in E$ as x_{ij}, let us see the optimization problem as linear program.

Linear Program

Primal LP for MCFP

$$\min \sum_{(i,j)\in E} c_{ij} x_{ij}$$

$$\forall i \in V, \quad \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = b_i$$

$$\forall (i,j) \in E, \quad 0 \leq x_{ij} \leq u_{ij}$$

First set of constraints are mass balance constraints while second set of constraints are flow bound constraints.

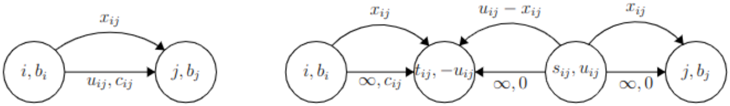
Reducing MCFP

Transshipment problem

Special case of MCFP is a Transshipment problem (TP) where all capacities in transporation network are infinite.

Claim 1

A MCFP can be reduced to a TP in O(m) time.



Reducing MCFP (contd.)

Algorithm for reduction of MCFP to TP

For a given transportation network, for every edge $(i, j) \in E$ with capacity u_{ij} and cost c_{ij}, do the following

1. Introduce two new vertices t_{ij} and s_{ij}
2. Replace (i, j) with with the $(i, t_{ij}), (s_{ij}, t_{ij}), (s_{ij}, j)$ with infinite capacities and set costs to be $c_{ij}, 0, 0$ respectively
3. Set demand of t_e to be u_{ij} and supply of s_e to be u_{ij}

Any valid flow in the original MCFP can be converted to a flow in the new TP with the same cost and vice versa. Henceforth, we will solely deal with Transshipment problems.

Transshipment problem

Abusing notation for transportation network, we define a transshipment problem as $G = (V, E, c, b)$

Primal LP for TP

$$\min \sum_{(i,j)\in E} c_{ij} x_{ij}$$

$$\forall i \in V, \quad \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = b_i$$

$$\forall (i,j) \in E, \quad x_{ij} \geq 0$$

There are m primal variables, one for each edge and n equality constraints, one for each vertex

Transshipment problem (contd.)

Dual LP for TP

$$\max \sum_{i \in V} b_i y_i$$

$$\forall (i,j) \in E, \quad y_j - y_i \leq c_{ij}$$

$$\forall i \in V, \quad y_i \geq 0$$

There are n dual variables, one for each vertex and m dual constraints, one for each edge

Transshipment problem (contd.)

Complementary Slackness

$$\forall (i,j) \in E, \quad x_{ij} \neq 0 \implies y_j - y_i = c_{ij}$$

Interpretation of Dual LP

The y_i's in the dual satisfy a shortest path ”type” condition, where edge weights are costs.

Feasibility

Is Dual always feasible?

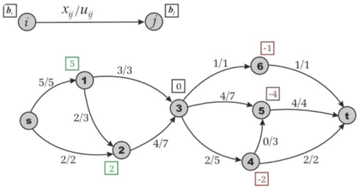
Costs can be negative in TP, so there may not exist feasible ”shortest path” for Dual

We need to answer two questions before solving TP.

1. Is the given TP feasible?
2. If yes, does an optimum exist?

Feasibility (contd.)

Consider a TP $G = (V, E, c, b)$. Let S be the set of sources ($S = \{i \in V : b_i > 0\}$) and T be the set of sinks ($T = \{i \in V : b_i < 0\}$). Consider the max-flow instance (G', b) constructed from TP, we add supersource s connected to all vertices in S with edge (s, i) of capacity b_i and we add supersink t connected to all vertices in T with edge (i, t) of capacity $-b_i$. All other edges have same capacity as in G



Feasibility (contd.)

For any $U \subseteq V$, $b(U) = \sum_{i \in U} b_i$

Claim 2

The TP $G = (V, E, c, b)$ is feasible iff $b(V) = b(S) + b(T) = 0$ and the maximum flow in instance (G', b) is exactly $b(S)$

Thus, we can determine feasibility by a single max-flow computation (Ford-Fulkerson algorithm), which brings us to our first assumption in further analysis.

Assumption 1

The TP G is feasible.

Existence of Optimal value

Circulation

A circulation is a flow that satisfies exact flow conservation constraints at all vertices. This means, for all vertices, the flow going in is equal to the flow going out.

Simplest circulation is flow along dicircuit(directed cycle). Observe that we can add a circulation to any feasible flow, to get a new feasible flow.

Claim 3

A feasible TP $G = (V, E, c, b)$ has an optimal solution iff G has no negative cost dicircuit.

Thus, we can determine existence of optimal solution by checking for negative cost dicircuit (Bellman-Ford), which brings us to our second assumption in further analysis.

Existence of Optimal value (contd.)

Assumption 2

The TP G has an optimal solution.

Proof for claim 3

Forward implication is straightforward. If G has a negative cost dicircuit, then we can reduce the objective of any feasible solution, by routing a circulation in this dicircuit. For backward implication, suppose G has no negative cost dicircuit. Create a new vertex s that connects to all vertices with costs 0. Observe that shortest path distances from s are defined. For vertex $i \in V$, let y_i be the shortest path distance from s. Note that $\{y_i\}$ values are dual feasible. Since G is feasible and we have obtained a dual feasible too, we can say that an optimum solution exists too.

Residual networks

Let us introduce notion of Residual networks. Given TP $G = (V, E, c, b)$ and feasible flow x

Residual network

The residual network G_x is defined as follows. For all $(i, j) \in E$

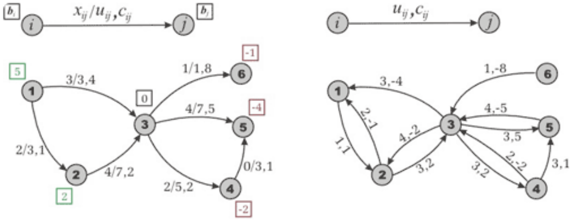
- If both x_{ij} and x_{ji} are zero, then G_x has the exact structure of G between i and j
- If $x_{ij} \neq 0$, then add the edge (i, j) with cost c_{ij} and infinite capacity (as in G). Also add the backward edge (j, i) with cost $-c_{ij}$ and capacity x_{ij}

These residual networks have some nice properties which we can use.

Residual networks (contd.)

Properties of Residual network G_x for feasible flow x

- Any circulation on G_x can be added to x to get a new feasible flow
- Every edge $(i, j) \in E$ of G is present in G_x. Moreover, the cost of this edge in G_x is at most c_{ij}.



Idea for Algorithm

- We maintain an infeasible primal flow x (which can be thought of as partial flow), a feasible dual point y, such that they satisfy complementary slackness
- This infeasible primal flow x respects all capacity constraints and for all vertices $i \in V$, the flow sent/received is at most $|b_i|$
- Each iteration will reduce "in-feasibility" of primal flow x while maintaining other conditions
- Finally, we will end up at a feasible primal flow x which will be optimal
- Optimality comes from the fact that the pair (x, y) are primal/dual feasible and will be satisfying complementary slackness

Idea for Algorithm (contd.)

- Since x is partial flow, we will have some sources S_x and sinks T_x in residual network G_x (with supply/demand as residual supply/demand)
- We can work conveniently with complementary slackness if we express it as dual feasibility w.r.t. G_x

Dual feasibility w.r.t $G_x = (V, E_x, c_x, b_x)$

We say that dual point y is dual feasible w.r.t. to G_x if for $(i,j) \in E_x$, $y_j - y_i \leq (c_x)_{ij}$

Since $E \subseteq E_x$, dual feasibility w.r.t G_x implies dual feasibility of y. This leads to following claim

Claim 4

If $(i,j) \in E$ and $(i,j) \in E_x$ and $(j,i) \in E_x$, then $y_j - y_i = c_{ij}$. Equivalently, if y is dual feasible w.r.t. G_x, then x and y satisfy complementary slackness.

Idea for Algorithm (contd.)

- Preceding claims makes it easy to capture complementary slackness, now we only have to maintain x as partial flow and dual point y as dual feasible w.r.t. G_x
- Now define $(i,j) \in E$ as equality edge if dual constraint corresponding to this edge is tight.

Claim 5

Suppose a path P from a source in S_x to a sink in T_x consists solely of equality edges, and we route flow along P to get a flow x'. If y was dual feasible w.r.t. G_x, then it is also dual feasible w.r.t. $G_{x'}$

- So algorithm should find a path of equality edges and route along it. If no such path exists then let R_x be set of vertices reachable by S_x using equality edges.
- Increase all dual variables in $V \backslash R_x$ by same amount till some constraints gets tight

Primal-Dual Algorithm

Algorithm 1 to solve TP $G = (V, E, c, b)$ for optimal flow x

- Initialize x to zero, calculate y using Bellman-Ford's algorithm
- while x is a partial flow
  - P = Path consisting of equality edges from S_x to T_x
  - If(P exists) route flow along P, update x and continue
  - R_x = set of vertices reachable by S_x using equality edges
  - Increase all dual variables in $V \backslash R_x$ by same amount till some constraint gets tight

Improving algorithm

- We can build on previous ideas and try to combine the case when P exists or not.
- Let us define the weight of an edge $(i, j) \in E_x$ to be $w_{ij} = (c_x)_{ij} - y_j + y_i$, note that weights are non-negative
- For $i \in V$, define σ_i as shortest path distance in G_x from any vertex in S_x.

Claim 6

Let $\sigma_m = \min_{t \in T_x} \sigma_t$. Then updating $y_i' = y_i + \min(\sigma_i, \sigma_m)$ maintains dual feasibility w.r.t $G_{x'}$

Primal-Dual Algorithm

Algorithm 2 to solve TP $G = (V, E, c, b)$ for optimal flow x

- Initialize x to zero, calculate y using Bellman-Ford's algorithm
- while x is a partial flow
  - For all edges $(i, j) \in E_x$, calculate w_{ij}
  - For all vertices $i \in V$, calculate $\{\sigma_i\}$
  - Determine σ_m, and corresponding shortest path P_m
  - Route as much flow as possible along P_m, and update x
  - For all vertices $i \in V$, update $y_i = y_i + min(\sigma_i, \sigma_m)$

Conclusion

- This algorithm has multiple real-life applications for solving problems like Discrete Location Problems or Transportation Problem
- Algorithm run-time can even be improved further by considering successive scaling and can be presented as future work

Questions?

References

- Lecture notes of CSE 202, 2021, University of California
- 3-part article from topcoder.com on Minimum Cost Flow