# Optimal Transport Using Network Flows

Leo Chang and Estella Xu

## Introduction

Our problem is stated roughly as follows: Let $A, B$ be two equal areas such that $A$ is filled with earth and $B$ is empty. How do we transport the earth from $A$ to $B$ with the least amount of effort? [1]. While the general transportation problem formulated by Monge has seen much advancement by functional analysis, we will be concerned with the much simpler linear programming formulation.

## Basic network theory

We begin with a short discussion of the prerequisite material on network flows. Consider a digraph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. where $\mathcal{N}$ and $\mathcal{A}$ are the set of nodes and the set of directed arcs of $\mathcal{G}$, respectively. We call $\mathcal{G}$ a *network*. Suppose $\mathcal{G}$ is such that each node has an associated value. For each $i \in \mathcal{N}$, we call this value $b_i$ the *supply* (in this convention, negative values are *demands*). Assume that the system is closed; that is, material being transported cannot enter or leave the system. Then we may assume that total supply and total demand are equal, or

$$\sum_{i \in \mathcal{N}} b_i = 0.$$

Assign each directed arc $(i, j)$ to a value $c_{ij}$ which denotes the *cost* to move 1 unit of material along $(i, j)$. For each $(i, j)$, let $x_{ij}$ denote the quantity transported from $i$ to $j$ along $(i, j)$. This quantity must be nonnegative since each arc is directed. The objective is to minimize total cost while meeting demand with the available supply, or to

$$\text{minimize} \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij}.$$

For any node $k \in \mathcal{N}$, the *total flow into* $k$ is

$$\sum_{i:(i,k) \in \mathcal{A}} x_{ik},$$

while the *total flow out from* $k$ is

$$\sum_{j:(k,j) \in \mathcal{A}} x_{kj}.$$

The net inflow is thus the difference between these sums, and this gives the *flow balance constraints*:

$$\sum_{i:(i,k) \in \mathcal{A}} x_{ik} - \sum_{j:(k,j) \in \mathcal{A}} x_{kj} = -b_k \text{ for } k \in \mathcal{N}.$$

We may write the optimization problem in matrix notation as

$$\begin{aligned} \text{minimize } & c^T x \\ \text{subject to } & Ax = -b \\ & x \geq 0, \end{aligned}$$

where $c$ is the cost vector, $x$ is the quantity transport vector, $A$ is the node-arc incidence matrix, and $b$ is the supply vector, with the elements of each ordered accordingly and corresponding to each other.

We may also give the dual problem in matrix notation as

$$\begin{aligned} &\text{minimize } b^T y \\ &\text{subject to } A^T y + z = c \\ &z \geq 0 \end{aligned}$$

and in network notation as

$$\begin{aligned} &\text{minimize } \sum_{i \in \mathcal{N}} b_i y_i \\ &\text{subject to } y_j - y_i + z_{ij} = c_{ij} \qquad && (i,j) \in \mathcal{A} \\ &z \geq 0 && (i,j) \in \mathcal{A}. \end{aligned}$$

The complementary conditions require

$$x_{ij} z_{ij} = 0 \qquad (i,j) \in \mathcal{A}.$$

We call the primal variables the *primal flows.*

A *path* is an ordered list of nodes such that each adjacent pair of nodes in the list is connected by an arc, regardless of direction. A network is *connected* if every pair of nodes is connected by a path. We will assume that all networks henceforth are connected. A *tree* is a network that is connected and acyclic. If a subnetwork is a tree and contains all the nodes of the larger network, it is a *spanning tree.*

In a network flow problem, a selection of primal flow values that satisfy the balance equations at each node is a *balanced flow*, and if all the flows are nonnegative in a balanced flow, it is a *feasible flow.* Given a spanning tree in the network, a balanced flow assigning 0 to every arc not on the spanning tree is a *tree solution.* It turns out that a square submatrix of the matrix representation of a network is a basis if and only if the arcs in the columns of the matrix form a spanning tree [2]. Hence, we can solve a network flow optimization problem by allowing arcs to enter and leave the spanning tree.

We return to the dual problem. We have that

$$y_j - y_i + z_{ij} = c_{ij} \qquad (i,j) \in \mathcal{A}.$$

By complementarity, $z_{ij} = 0$ for all arcs in the spanning tree. Note that a spanning tree with $m$ nodes has $m-1$ arcs since we can fix a root and the other $m-1$ nodes must connect to the spanning tree with an arc. So, there exists a system of $m-1$ equations with $m$ unknowns. Since the primal problem contains a redundant equation associated to the root node, the dual variable for the root node is 0. That is, we have a system of $m$ equations with $m$ unknowns, which we can certainly solve. The dual slacks for the arcs not in the spanning tree then just follow from the dual feasibility condition solved for $z_{ij}$.

Duality theory gives that an optimal solution is reached when all flows and all dual slacks are nonnegative. We will discuss this problem in the example below.

## The transportation problem

The transportation problem is a specific type of network flow, called a *bipartite graph*. Here, the set of nodes, $\mathcal{N}$, can be partitioned into a set of supply nodes, $\mathcal{S}$, and a set of demand nodes, $\mathcal{D}$. Every arc $\mathcal{A}$ has a tail in $\mathcal{S}$ and a head in $\mathcal{D}$. Note that in order for the bipartite graph to be a valid representation of supply and demand, in addition to the previously mentioned requirements of networks, all supply values must be nonnegative and all demand values must be nonpositive.

The canonical linear programming optimization problem is the Hitchcock transportation problem, a special case of the bipartite graph where each supply node must have an arc to each demand node. Thus, the notation of the general network problem is simplified. Denote the cost of traveling from node $i \in \mathcal{S}$ to $j \in \mathcal{D}$ as $c_{ij}$, the amount transported from $i$ to $j$ as $x_{ij}$, the supplies at the supply nodes $s_i$ for $i \in \mathcal{S}$, and the demands at the demand nodes $d_j$ for $j \in \mathcal{D}$. Then, we write the problem as

$$\begin{aligned} \text{minimize } & \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{D}} c_{ij}x_{ij} \\ \text{subject to } & \sum_{j\in\mathcal{D}} x_{ij} = s_i \quad i \in \mathcal{S} \\ & \sum_{i\in\mathcal{S}} x_{ij} = d_j \quad j \in \mathcal{D} \\ & x_{ij} \geq 0 \quad i \in \mathcal{S}, j \in \mathcal{D}. \end{aligned}$$

## Example

Consider the following transportation problem[1]:

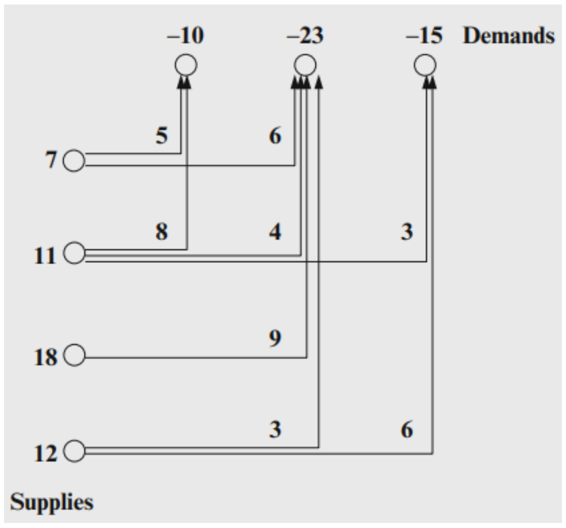


Figure 1: Given problem.

[1]Problem 15.1 in [2].

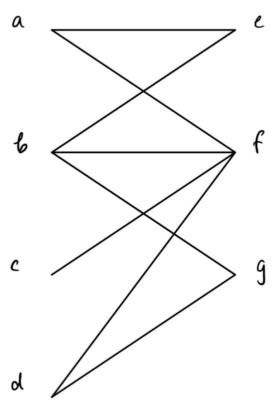
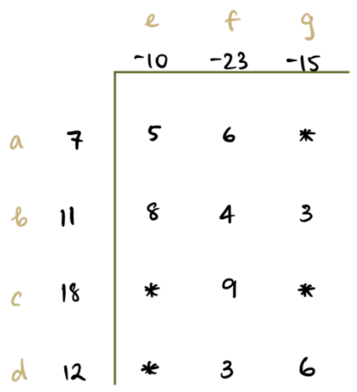


Figure 2: Alternate representation: supplies a-d (top to bottom), demands e-g (left to right).

In Figure 1, the numbers next to the node denote the amount available to supply or the amount demanded, and the numbers on the arcs denote the cost to transport between the two corresponding nodes. It must hold that

(i) the graph is connected;
(ii) all supply nodes have nonnegative value and all demand nodes have nonpositive value;
(iii) the total amount supplied and demanded equals 0, i.e., the system is closed;
(iv) all quantities transported are nonnegative.

Now, we will organize the information in an adjacency matrix:



| | | e | f | g |
|---|---|---|---|---|
| | | -10 | -23 | -15 |
| a | 7 | 5 | 6 | * |
| b | 11 | 8 | 4 | 3 |
| c | 18 | * | 9 | * |
| d | 12 | * | 3 | 6 |

Figure 3: Tabular representation of supplies, demands, and costs.

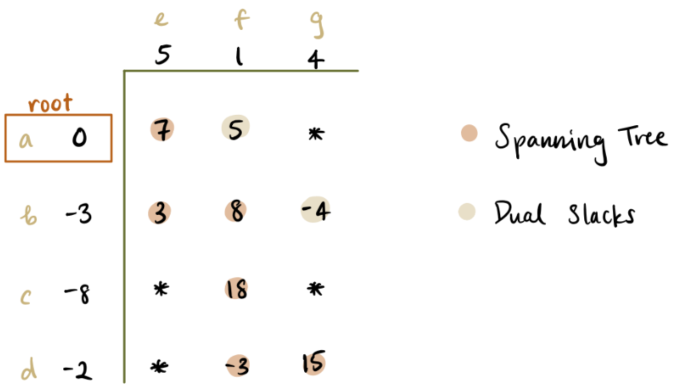
The stars denote arcs that do not exist, but one may also set them to an arbitrarily large number to represent infinite cost.

We first choose an arbitrary tree solution, which automatically puts the problem in its dual form. We find one by taking a spanning tree (any will work), picking a “root” node, and balancing the spanning tree accordingly. Letting the root be $a$, we assume that all the non-spanning tree arcs have no flow so that the spanning tree is a solution by itself. The first arc is $(a, e)$ and it must take from node $a$ 7 units, since the rest of the arcs connecting to node $a$ are not in the spanning tree and thus 0. So, $(a, e)$ has a flow of 7. Then, looking at arc $(b, e)$, notice that in total, $(a, e)$ and $(b, e)$ must supply node $e$ with 10 units (from the $-10$ demand). Since $(a, e)$ has flow 7, $(b, e)$ must have flow 3. We continue along the spanning tree in this manner calculating how we can distribute flow in the spanning tree to satisfy the supply and demand independent of the non-spanning tree arcs.

In the remaining entries of the matrix (that is, the entries that are not part of the spanning tree), there is no flow, so we use these entries to represent the dual problem. To find the dual slacks, or the flow of the arcs not in the spanning tree, we first need to calculate the dual variables. First, set the dual variable at the root node $a$, denoted $y_a$, equal to 0 (recall that the root node is associated with the redundant equation). Then, we calculate the other dual variables using the constraint

$$y_j - y_i + z_{ij} = c_{ij} \qquad (i, j) \in \mathcal{A}.$$

By complementarity, $z_{ij} = 0$ for each $(i, j)$ in the spanning tree. Consider arc $(a, e)$. Using the constraint, we have $y_e - y_a = 5$, and since $y_a = 0$, $y_e = 5$. We continue in this manner through the spanning tree to get all the dual variables. We can then calculate the dual slacks using the complementarity mentioned above.



| | | e | f | g |
|---|---|---|---|---|
| | | 5 | 1 | 4 |
| root a | 0 | 7 | 5 | * |
| b | -3 | 3 | 8 | -4 |
| c | -8 | * | 18 | * |
| d | -2 | * | -3 | 15 |

● Spanning Tree
● Dual Slacks

Figure 4: Table containing the dual variables, the spanning tree, and the dual slacks.

Now that we have the spanning tree, we can begin what is called the *self-dual network simplex method*. Like the regular simplex method, this optimization is based on entering and then removing an arc. First, we perturb any negative value by adding $\mu$, which represents the amount needed to make an edge feasible.[2] Notice that to reach the optimal solution, we need the minimum value of $\mu$ to be less than or equal to zero. Since we want each value in the table to be nonnegative, we must have $\mu \geq 4$. We let the arc that determined this inequality enter the spanning tree (this is synonymous with picking the "worst" or the farthest away value in the tableaus we've seen in class—the $(b, g)$ arc has the most negative value, which is the one that gives $\mu \geq 4$, so we therefore let it enter). We call this arc the *primal pivot*.

Once we let the $(b, g)$ arc enter the spanning tree, notice that the spanning tree now consists of a cycle. This follows by the fact that spanning trees include every node.

Now, we have added the flow of $(b, g)$ to the cycle; for now, we will call this amount $t$. Since we must maintain flow balance in the cycle, all other arcs must be adjusted (Figure 5). In particular, the arcs flowing in the same cyclical direction as the entering arc must be incremented by $t$ and the opposite decremented (since we need to, for example, let node $b$ give node $f$ $t$ fewer units in order for it to give $t$ units to node $g$).

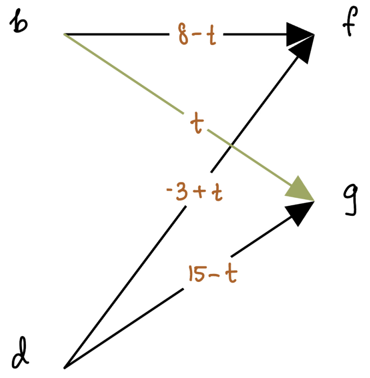


Figure 5: Cycle created by entering $(b, g)$.

To choose which arc leaves the spanning tree, we choose the element that that gives the tightest bound—as $t$ increases, $(b, f)$ eventually decreases to zero (and it's the first edge to do so), and since we need all edges to be nonnegative, we let $t = 8$. Therefore, we let $(b, g)$ enter and let $(b, f)$ leave (Figure 6-7). From here on out, the dual variables do not matter for the problem; as long as we keep track of our incrementing and decrementing, the spanning tree and the dual slacks are all we need. Update all other values in the cycle based on $t$.

[2]Technically, $\mu$ is optional. It merely allows us to confirm which edge is the "worst."

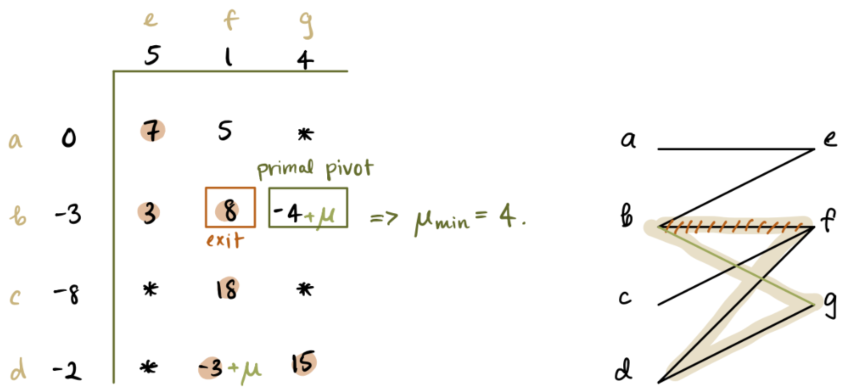


Figure 6: The first iteration.

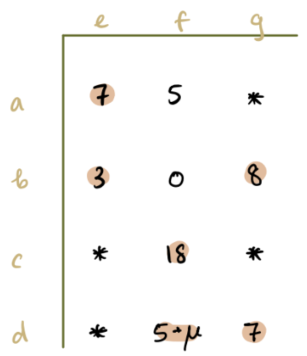

e f g
a 7 5 *
b 3 0 8
c * 18 *
d * 5+μ 7

Figure 7: Updating the cycle.

Notice that when we remove the leaving arc and before we add the entering arc, we create a two disjoint networks, highlighted in blue and yellow in Figure 8, with one containing the tail of the leaving arc and the other containing the head. Now, the two orange arcs are the ones no longer in the spanning tree, and because they bridge the two disjoint trees, they must be updated. In other words, since we changed the original bridge between the two disjoint trees, we must change the other bridges as well. In particular, they must get decremented by the old value of the entering arc, giving us the table in Figure 8.

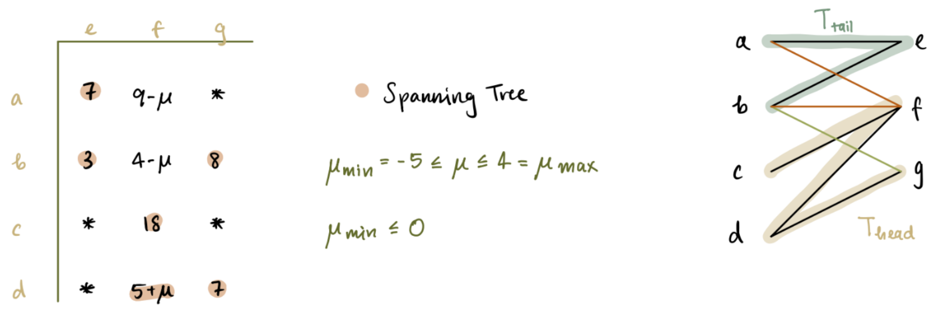


Figure 8: Updating flows outside of the spanning tree.

Finally, all of the flow values in the matrix are positive (and, if we are using the $\mu$ notation, $\mu_{\text{min}} = -5$), meaning that we have found the optimal value.[3] Then, taking the flows (which denotes the amount delivered through an arc) and multiplying by the original cost values, we get 302. Therefore, the optimal cost is 302.

## Summary

In conclusion, a linear programming optimal transport problem can be solved by the self-dual simplex method as follows:

1. Pick a spanning tree and find a possible solution. Update its flows assuming that all other arcs have zero flow. Then, find the dual variables and the dual slacks.
2. Begin optimizing by letting the "worst" arc enter the spanning tree. Then, pick the leaving arc by finding the least flow in the opposite direction of the entering arc in the cycle.
3. Adjust the flows of the other arcs in the spanning tree based on the flow of the entering arc.
4. Adjust the flows of the dual slacks (non-spanning tree arcs) based on how they bridge the two disjoint trees formed by removing the leaving arc.
5. Repeat steps 2-4 until all flows are positive. Then, plug in the flow values into the objective function.

[3]Often times, we must perform more than one iteration of this algorithm by picking a new pivot.

## References

[1] Arthur Cayley. “On Monge’s ’Mèmiore sur la Thèorie des Dèblais et des Remblais.” In: *Proceedings of the London Mathematical Society* s1-14.10 (1882), pp. 139–143. DOI: `http://doi.org/10.1112/plms/s1-14.1.139`.

[2] Robert J. Vanderbei. *Linear Programming.* International Series in Operations Research  Management Science. Springer New York, NY, 2013. ISBN: 978-1-4614-7630-6.