A Tutorial on Network Optimization

Gérard Cornuéjols and Michael A. Trick

Summer, 1994

1 Introduction

Seervada Park has recently been opened to sightseeing and backpacking. Cars are not permitted in the park, but there is a narrow, winding, set of trails that are used by the rangers' jeeps and by a tram system that takes tourists from the entry point to a spectacular set of falls. The trail system looks like Figure 1, where a trail is represented by a line, and the meeting of two or more trails is represented by a circle. At every such meeting point there is a ranger station. The number beside a line is the length of that path in miles.

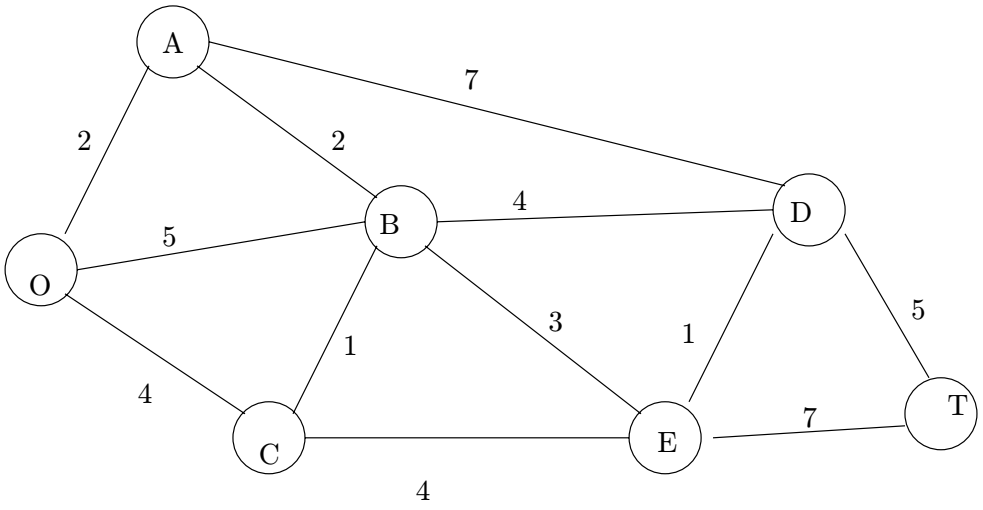


Figure 1: Seervada Park

The park management has three problems to solve. The first is to find the best path from the entrance (O) to the waterfalls (T). Here, "best" is defined to be the shortest path from O to T. This is an example of the *shortest path problem*.

The second problem is that a telephone system must be installed under the road to link all the stations (including the entrance and the waterfall's station). Because installation is expensive and disruptive to the unique ecology of the park, only enough lines will be installed so that there is some connection between every pair of stations. The objective is to minimize the miles of lines that must be installed. This is an example of the *minimum spanning tree problem*.

Finally, the third problem is that during certain periods (like the Labor Day weekend) more people wish to use the tram than can be accommodated. To avoid unduly disturbing the wildlife and ecology of the park, strict limitations on the number of trams that can use each trail segment have been placed. Therefore, during the peak season, various routes might be used by the trams in order to maximize the number of tourists served. The problem is to determine the maximum number of trips per day that can be done without violating the capacity restrictions. This is an example of the *maximum flow problem*.

2 Terminology

A **network** or **graph** consists of points, and lines connecting pairs of points. The points are called **nodes** or vertices. The lines are called **arcs**. The arcs may have a direction on them, in which case they are called **directed arcs**. If an arc has no direction, it is often called an **edge**. If all the arcs in a network are directed, the network is a **directed network**. If all the arcs are undirected, the network is an **undirected network**.

Two nodes may be connected by a series of arcs. A **path** is a sequence of distinct arcs (no nodes repeated) connecting the nodes. A **directed path** from node i to node j is a sequence of arcs, each of whose direction (if any) is towards j. An **undirected path** may have directed arcs pointed in either direction.

A path that begins and ends at the same node is a **cycle** and may be either directed or undirected.

A network is **connected** if there exists an undirected path between any

pair of nodes. A connected network without any cycle is called a **tree**, mainly because it looks like one.

3 Shortest Paths

Consider an *undirected* and *connected* network with two special nodes, called the *origin* and *destination*. Associated with each edge is a *distance*, a non-negative number. The objective is to find a shortest path between the origin and destination.

There is a very fast algorithm for this problem. The essence is to explore outward from the origin, successively finding the shortest paths to nodes in the network until the destination is reached.

Consider the park example, with origin O and destination T. Looking just locally, we can determine that we can reach the following nodes at the given distances:

- A: distance 2
- B: distance 5
- C: distance 4

Now consider the closest node, in this case node A. We can reach it directly in distance 2. Might we find another path from O to A that is shorter? Of course not! We would have to go through another node, and we already know that all other nodes are at least distance 2 away from O. Therefore we know that A is the closest node to O and has distance 2.

What is the second closest node to O? The key to finding this is to know the following (due to Dijkstra):

If we have found the kth closest nodes to O, then the $k+1$st closest has a path that goes from O to one of the k closest and then a single edge from that node to the $k+1$st.

This assumes we treat node O as the 0th-closest node to O. In other words, to determine the second closest, look at edges going out from O and the first closest:

- B has distance 4 (2 to get from O to A and 2 to get from A to B)
- C has distance 4 (direct from O)

- D has distance 9 (2 to get from O to A and 7 to get from A to D)

and those are all the possibilities. The shortest of these is the second closest. There is a tie, so we will arbitrarily make B the second closest (distance 4) and C the third closest (distance 4).

Which node is the fourth closest? The possibilities are:

- E at distance 7 (through B)
- D at distance 8 (through B)

so the fourth closest is E. The fifth closest is either

- D at distance 8 (through B or E), or
- T at distance 14 (through E).

The fifth closest is D. So the sixth closest is T at distance 13 through D. Working our way backwards gives us that the shortest path from O to T is

O-A-B-D-T (or, alternatively, O-A-B-E-D-T).

In general the algorithm is as follows:

> *To find the $k+1$st closest given the k closest:* Call the k closest (including the origin) the "solved" nodes and the others the unsolved nodes. For each solved node, find the unsolved node connected to it by the shortest edge. This is a candidate node. For each candidate, add the distance on the edge to the distance from the origin to the solved node. The candidate with the shortest such distance is the $k+1$st closest node.

You can find the shortest path from the origin to the destination by repeatedly using the above step until the destination becomes a solved node. Ties may be broken arbitrarily, and may provide information about alternative shortest paths.

Discussion. It is important to recognize that the shortest path problem is a *model* that can be used even if no distances are involved. For instance, the numbers on the edges might represent costs, so the problem is finding a sequence of tasks that completes some goal at minimum cost. Or the numbers might be time, and the problem is minimizing the amount of time required to accomplish some goal. This is illustrated by the following exercise:

Exercise 1 *At a small but growing airport, the local airline company is purchasing a new tractor for a tractor-trailer train to bring luggage to and from the airplanes. A new mechanized luggage system will be installed in 3 years, so the tractor will not be needed after that. However, because it will receive heavy use, and maintenance costs are high, it may still be more economical to replace the tractor after 1 or 2 years. The following table gives the total net discounted cost associated with purchasing a tractor in year i and trading it in in year j (where year 0 is now):*

		j		
		1	*2*	*3*
	0	*8*	*18*	*31*
i	*1*	—	*10*	*21*
	2	—	—	*12*

The problem is to determine at what times the tractor should be replaced (if ever) to minimize the total costs for tractors.

(a) Formulate this problem as a shortest path problem.

(b) Use the algorithm presented to solve this problem.

Exercise 2 *Solve the shortest path problem for the network in figure 2.*

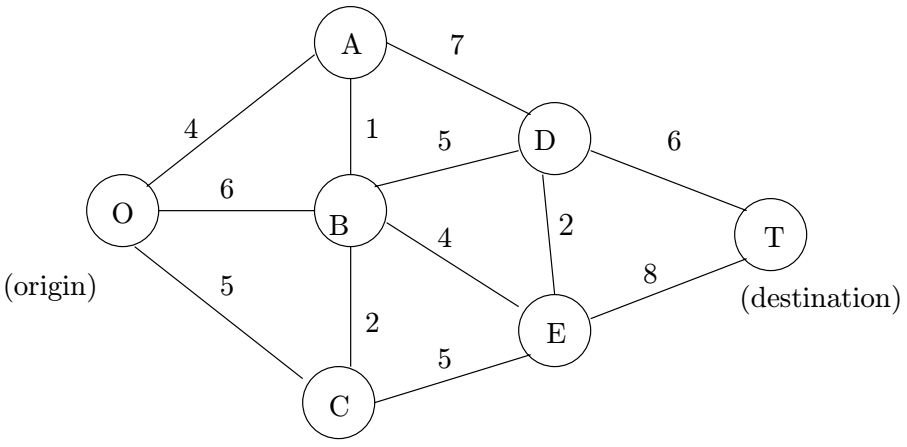


Figure 2: Distances

4 Minimum Spanning Tree

The minimum spanning tree problem is to choose edges from an undirected connected network so that every two nodes are connected by a path. The objective is to choose edges with minimum total distance. It is intuitively clear that such a set will not contain a cycle. Therefore, the set will be a tree. This tree will connect (*span*) all the nodes.

This problem has a number of important practical applications. For example, it can sometimes be helpful in planning *transportation networks* that will not be used much. There, the primary consideration is to provide some link between all pairs of nodes in the most economical way. Other examples include the planning of large-scale communication models and distribution networks.

The minimum spanning tree problem can be solved in a very straightforward way because it is one of the few problems where being *greedy* at each stage of a solution procedure still results in an optimal solution at the end! The algorithm is as follows:

Greedy algorithm

1. *Initialization* The algorithm starts with no edge in the tree.

2. *Iterative Step* The edges are considered in increasing order of their length. If an edge does not form a cycle with edges already in the tree, add it to the tree. Otherwise, discard it.

3. *Stopping Criterion:* Stop when all nodes are connected.

Proof that the greedy algorithm finds a minimum spanning tree: Let T^* denote the tree found by the greedy algorithm and suppose some other tree is shorter. We will show that this leads to a contradiction. Among the trees which are shorter than the greedy solution T^*, denote by T one which has the largest number of common edges with T^*. Let a be a shortest edge in T^* but not in T and let C be the unique cycle formed when edge a is added to the edges of T. The cycle C contains at least one edge not in T^*, say b, since T^* contains no cycle. The length of a is less than or equal to that of b, since edge a must have been considered before b by the greedy algorithm. Now the tree T' obtained from T by adding edge a and removing edge b is of the same length as or shorter than T. Therefore T' is shorter than T^*. But it has one more edge in common with T^*, a contradiction. This completes the proof.

Here's a different algorithm for finding a minimum spanning tree:

1. Select a node arbitrarily, and connect it to a node closest to it.
2. Identify an unconnected node that is closest to a connected node and add the edge between them. Repeat until all nodes have been connected.

It may seem that it makes a difference which node to start at. It turns out it doesn't (except for possibly finding alternative minimum spanning trees). We omit the proof that this algorithm also gives an optimum solution to the minimum spanning tree problem. In the homework and exam, you can use either of these two algorithms to solve the problem.

Exercise 3 *A bank is hooking up computer terminals at each of its branch offices using special phone lines. The phone line from a branch does not need to be directly connected to the main office, but there must be a sequence of phone lines from any office to the main office.*

The charge for the special phone lines is directly proportional to the mileage involved. The distance between the offices are given in the following table.

	Main	*B1*	*B2*	*B3*	*B4*	*B5*
Main Office	—	*190*	*70*	*115*	*270*	*160*
Branch 1	*190*	—	*100*	*240*	*215*	*50*
Branch 2	*70*	*100*	—	*140*	*120*	*220*
Branch 3	*115*	*240*	*140*	—	*175*	*80*
Branch 4	*270*	*215*	*120*	*175*	—	*310*
Branch 5	*160*	*50*	*220*	*80*	*310*	—

(a) Explain how this problem fits in with the definition of the minimum spanning tree problem.

(b) Solve this problem.

5 Maximum Flow

Now we address the final question of the Seervada Park management: how to route the various tram trips from the entrance to the waterfalls to maximize the number of trips per day. There are strict upper bounds on the number of

outgoing trips per day (we will assume that trams return by the same route, so we can concentrate only on the outgoing trips) on each arc. The limits are represented in figure 3. In general, the limit (or *capacity*) of arc (i, j) will be denoted by c_{ij}.

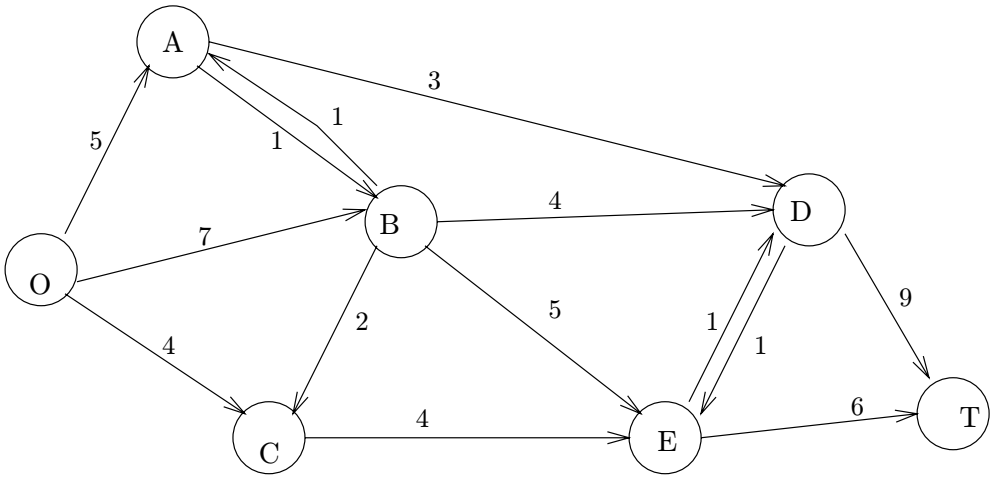


Figure 3: Daily limits

Note that the arcs are directed and it is possible to have oppositely directed arcs between the same nodes. Given the limits, one possible solution is to send twelve trams per day, with five using the route O-B-E-T, one using O-C-E-T, one using O-C-E-D-T, three more on O-A-D-T and two on O-B-D-T. Is there a way to do better?

A very efficient algorithm for this problem is the *augmenting path algorithm*, due to Ford and Fulkerson. To describe this algorithm, it will be useful to introduce the following notation. The traffic (or *flow*) on arc (i, j) is denoted by f_{ij}. For example, in the twelve tram solution described above, $f_{OA} = 3$ (3 trams use arc OA), $f_{OB} = 5+2 = 7$ (7 trams use arc OB, namely 5 from route O-B-E-T and two from O-B-D-T), $f_{OC} = 1+1 = 2$ and so forth.

An *augmenting path* is a undirected path from the origin to the destination such that every arc directed in the forward direction of the path has positive *residual capacity* $c_{ij} - f_{ij}$ and every arc directed in the backward direction of the path has positive flow f_{ij}. Each augmenting path represents an opportunity to increase the amount of flow sent from the origin to the destination. Note that such a path may include both arcs that represent increasing flow (the forward arcs) and arcs that represent cancelling flow (the

backward arcs). The maximum amount of flow that can be feasibly sent along the path is determined by the smallest of the residual capacities along forward arcs *and* of the flows along backward arcs.

This gives the augmenting path algorithm:

1. Identify an augmenting path from the origin to the destination.
2. Find the maximum amount of flow that can be feasibly sent along this augmenting path. Let this value be m.
3. Increase the flow on forward arcs of this path by m and reduce the flow on backward arcs by m.

Note that step 1 is not completely defined. There are many alternatives for finding an augmenting path. When we solve the example, the paths will be chosen arbitrarily (and you may also choose them arbitrarily in the homework and on exams). Note, however, that you must be organized enough to always find an augmenting path when one exists. So it is in fact a good idea to apply a labeling procedure (such as Ford and Fulkerson's method, described in many textbooks) when you cannot find more augmenting paths by inspection.

Let's work through our example: is the twelve tram solution described above an optimal solution?

By inspection, we discover that the path O-A-B-D-T, which only contains forward arcs, has positive residual capacity on every arc. Namely, the residual capacity on arc OA is 2, it is 1 on AB, 2 on BD and 3 on DT. The smallest of these values is 1. Therefore we have found an augmenting path and we can obtain a better solution by sending one unit of flow along this path. This yields a thirteen tram solution. Is it optimal?

Again, by inspection, we discover the following augmenting path: O-C-E-B-D-T. It contains four forward arcs and one backward arc (BE). The residual capacities on the forward arcs are 2, 2, 1 and 2 respectively. The flow on the backward arc is 5. The smallest of these values is 1. Therefore we can increase our total flow by 1 (to a total of fourteen trams). This is acheived by increasing the flow on each of the forward arcs by one unit and decreasing it by one unit on the backward arc. The resulting arc flows are shown in figure 4.

Note that routes for the trams can easily be constructed from the flow solution. For example, three trams can use route O-C-E-T, three O-B-E-T,

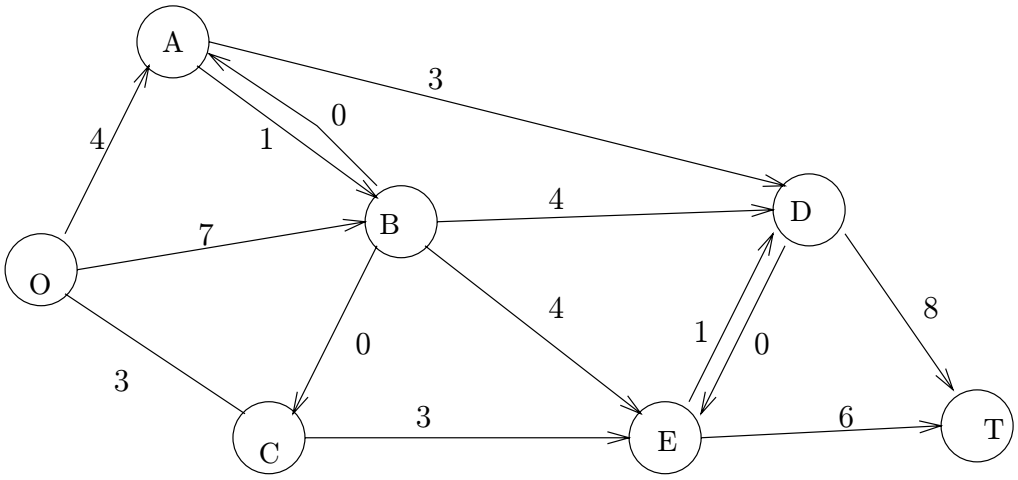


Figure 4: Optimal flow

one O-B-E-D-T, three O-B-D-T, three O-A-D-T and one O-A-B-D-T. There are many other ways of routing the fourteen trams. We are happy with just one. Can we do better than fourteen trams?

We apply the Ford-Fulkerson labeling procedure. The origin O is labeled. Then A is labeled since there is a positive residual capacity on arc OA. Similarly, C can be labeled and then E (since OC and CE have positive residual capacities). Next B can be labeled since BE has positive flow and E is already labeled. Now all arcs going from a labeled node to an unlabeled one have zero residual capacity and all arcs going from an unlabeled node to a labeled one have zero flow. So no further labeling is possible. Since the destination could not be labeled, there is no augmenting path to be found. The current solution is optimum.

There is a simple way to see that, when the destination cannot be labeled by the labeling procedure, we have found an optimum flow: look at the arcs going out of the labeled nodes, namely O-A-B-C-E. The total capacity of the arcs leaving this set is 14. Therefore, we know that we cannot send more than 14 units of flow from O to T. Since we found a way to send 14 units, we know we have the maximum flow. This is not a coincidence. Call a **cut** a set of arcs leaving a node set that contains the origin and doesn't contain the destination. Let the capacity of a cut be the sum of the capacities of arcs in the cut. The **max-flow min-cut theorem** states that the maximum feasible flow from the origin to the destination equals the capacity of a cut

with minimum capacity. To prove this theorem, we could write the maximum flow problem as a linear program (we will actually do this in the minimum cost flow section) and observe that the dual of this linear program is the minimum cut problem. The max flow- min cut theorem then follows from the strong duality theorem of linear programming. (The only tricky part of the proof is to show that the max-flow and the min-cut problems can indeed be formulated as linear programs instead of integer programs. The key property here is known as *(total) unimodularity*. We will not go into it in this course.

Exercise 4 *Consider a network with nodes s, a, b, c, d, t where s is the origin, t is the destination, and arcs (s,a), (s,b), (s,c), (a,b), (a,d), (b,c), (b,t), (c,t), (d,b), (d,t). The following table contains arc capacities as well as arc flows.*

	Arcs									
	(s,a)	*(s,b)*	*(s,c)*	*(a,b)*	*(a,d)*	*(b,c)*	*(b,t)*	*(c,t)*	*(d,b)*	*(d,t)*
Capacities	*10*	*4*	*6*	*2*	*6*	*2*	*7*	*4*	*3*	*6*
Flows	*8*	*4*	*2*	*2*	*6*	*2*	*7*	*4*	*3*	*3*

(a) Verify that the solution given by the above arc flows is feasible, that is it satisfies the capacity restrictions and there is conservation of flow at each node of the network other than the origin and destination. (Conservation of flow at node a means that the total flow going into node a equals the total flow leaving it.)

(b) Is the above solution a maximum flow from s to t ? If not, find an augmenting path and construct a maximum flow.

(c) Find a cut of minimum capacity.

6 Other Network Models

So far we have considered three network models: shortest path, minimum spanning tree, and maximum flow, and you have learned some algorithms to solve these problems. The purpose of learning these algorithms is twofold: to impress upon you the simplicity of the algorithms, and to allow you to solve smallish examples quickly and accurately.

We will now move on to some more complicated models. These models can all be solved very effectively by specializing the simplex method for them. It is important for you to know how to model within the restrictions required by each model. The models we will look at are the transportation, assignment, transshipment and minimum cost flow models.

6.1 Transportation Problem

One of the main products of P&T Company is canned peas. The peas are prepared at three canneries (near Bellingham, Washington; Eugene, Oregon; and Albert Lea, Minnesota) and are then shipped by truck to four distributing warehouses in Sacremento, California; Salt Lake City, Utah; Rapid City, South Dakota; and Albuquerque, New Mexico. Because shipping costs are a major expense, management has begun a study to reduce them. For the upcoming season, an estimate has been made of what the output will be from each cannery, and how much each warehouse will require to satisfy its customers. The shipping costs from each cannery to each warehouse has also been determined. This is summarized in the next table.

Shipping cost	Warehouse				
per truckload	1	2	3	4	Output
1	464	513	654	867	75
Cannery 2	352	416	690	791	125
3	995	682	388	685	100
Requirement	80	65	70	85	

You should find it an easy exercise to model this as a linear program. If we let x_{ij} be the number of truckloads shipped from cannery i to warehouse j, the problem is to

$$
\begin{aligned}
\text{minimize} \quad & 464x_{11} + 513x_{12} + 654x_{13} + 867x_{14} + 352x_{21} + \ldots + 685x_{34} \\
\text{subject to} \quad & \\
& x_{11} + x_{12} + x_{13} + x_{14} = 75 \\
& x_{21} + x_{22} + x_{23} + x_{24} = 125 \\
& x_{31} + x_{32} + x_{33} + x_{34} = 100 \\
& x_{11} + x_{21} + x_{31} = 80 \\
& x_{12} + x_{22} + x_{32} = 65 \\
& x_{13} + x_{23} + x_{33} = 70 \\
& x_{14} + x_{24} + x_{34} = 85 \\
& x_{ij} \geq 0 \text{ for all } i \text{ and } j.
\end{aligned}
$$

This is an example of the **transportation model**. As has been pointed out, this problem has a lot of nice structure. All the coefficients are 1 and every variable appears in exactly two constraints. It is this structure that lets the simplex algorithm be specialized into an extremely efficient algorithm.

What defines a transportation model? In general, the transportation model is concerned with distributing (literally or figuratively) a commodity from a group of supply centers, called **sources** to a group of receiving centers, called **destinations** to minimize total cost.

In general, source i has a supply of s_i units, and destination j has a demand for d_j units. The cost of distributing items from a source to a destination is proportional to the number of units. This data can be conveniently represented in a table like that for the sample problem.

We will generally assume that the total supply equals the total demand. If this is not true for a particular problem, *dummy* sources or destinations can be added to make it true. The text refers to such a problem as a *balanced transportation problem*. These dummy centers may have zero distribution costs, or costs may be assigned to represent unmet supply or demand.

For example, suppose that cannery 3 makes only 75 truckloads. The total supply is now 25 units too little. A dummy supply node can be added with supply 25 to balance the problem, and the cost from the dummy to each warehouse can be added to represent the cost of not meeting the warehouse's demand.

The transportation problem has a couple of nice properties:

Feasibility As long as the supply equals demand, there exists a feasible solution to the problem.

Integrality If the supplies and demands are integer, every basic solution (including optimal ones) have integer values. Therefore, it is not neces-

sary to resort to integer programming to find integer solutions. Linear programming suffices. Note that this does not mean that each destination will be supplied by exactly one source.

In the pea shipping example, a basic solution might be to ship 20 truckloads from cannery 1 to warehouse 2 and the remaining 55 to warehouse 4, 80 from cannery 2 to warehouse 1 and 45 to warehouse 2 and, finally, 70 truckloads from cannery 3 to warehouse 3 and 30 to warehouse 4. Even though the linear programming formulation of the pea shipping example has seven constraints other than nonnegativity, a basic solution has only six basic variables! This is because the constraints are linearly dependent: the sum of the first three is identical to the sum of the last four. As a consequence, the feasible region defined by the constraints would remain the same if we only kept six of them. In general, a basic solution to the transportation model will have a number of basic variables equal to the number of sources plus the number of destinations minus one.

We close this section by showing how the dual problem can be used to verify whether a solution, such as the one given above, is optimal. In our example, the dual linear program is

$$\begin{aligned}
&\text{Maximize} && 75u_1 + 125u_2 + 100u_3 + 80v_1 + 65v_2 + 70v_3 + 85v_4 \\
&\text{Subject to} \\
&&& u_1 + v_1 \leq 464 \\
&&& u_1 + v_2 \leq 513 \\
&&& u_1 + v_3 \leq 654 \\
&&& u_1 + v_4 \leq 867 \\
&&& u_2 + v_1 \leq 352 \\
&&& \ldots \\
&&& u_3 + v_4 \leq 685 \\
&&& u_i, v_j \text{ unrestricted for all } i \text{ and } j.
\end{aligned}$$

The fact that one of the primal constraints is redundant means that one of the dual variables can be set to zero. For example we can set $u_1 = 0$. Now we apply the complementary slackness conditions. If the solution $x_{12} = 20$, $x_{14} = 55$, $x_{21} = 80$, $x_{22} = 45$, $x_{33} = 70$, $x_{34} = 30$ is optimal, then the dual constraints corresponding to these basic variables must be tight:

$$
\begin{aligned}
u_1 + v_2 &= 513 \\
u_1 + v_4 &= 867 \\
u_2 + v_1 &= 352 \\
u_2 + v_2 &= 416 \\
u_3 + v_3 &= 388 \\
u_3 + v_4 &= 685.
\end{aligned}
$$

Since we can choose $u_1 = 0$, this linear system of equations is very easy to solve. We get $v_2 = 513$, $v_4 = 867$. This implies $u_2 = -97$ and $u_3 = -182$. Finally $v_1 = 449$ and $v_3 = 570$. Is this dual solution a feasible dual solution? We must check the constraints.

$$
\begin{aligned}
u_1 + v_1 &= 0 + 449 && \leq 464 \\
u_1 + v_3 &= 0 + 570 && \leq 654 \\
u_2 + v_3 &= -97 + 570 && \leq 690 \\
u_2 + v_4 &= -97 + 867 && \leq 791 \\
u_3 + v_1 &= -182 + 449 && \leq 95 \\
u_3 + v_2 &= -182 + 513 && \leq 682.
\end{aligned}
$$

Since we have primal and dual feasible solutions which satisfy the complementary slackness conditions, these solutions are optimal.

Exercise 5 *Consider the following transportation problem*

Shipping cost	*Destination*				
	1	*2*	*3*	*4*	*Supply*
1	*6*	*0*	*2*	*11*	*15*
Source 2	*12*	*7*	*9*	*20*	*30*
3	*0*	*4*	*0*	*8*	*5*
Demand	*10*	*15*	*15*	*10*	

and the following solution: $x_{12} = 5$, $x_{14} = 10$, $x_{21} = 5$, $x_{22} = 10$, $x_{23} = 15$, $x_{31} = 5$. *Use duality to determine whether this is an optimal solution.*

6.2 Assignment Problem

A special case of the transportation problem is the *assignment problem* which occurs when each supply is 1 and each demand is 1. In this case, the integrality implies that every supplier will be assigned one destination and every destination will have one supplier. The costs give the charge for assigning a supplier and destination to each other.

Example. A company has three new machines of different types. There are four different plants that could receive the machines. Each plant can only receive one machine, and each machine can only be given to one plant. The expected profit that can be made by each plant if assigned a machine is as follows:

	Plant			
	1	2	3	4
1	13	16	12	11
Machine 2	15	0	13	20
3	5	7	10	6

This is a transportation problem with all supplies and demands equal to 1, so it is an assignment problem.

Note that a balanced problem must have the same number of supplies and demands, so we must add a dummy machine (corresponding to receiving no machine) and assign a zero cost for assigning the dummy machine to a plant.

Exercise 6 *Albert, Bob, Carl, and David are stranded on a desert island with Elaine, Francine, Gert, and Holly. The "compatibility measures" in the next table indicate the happiness each couple would experience if they spent all their time together. If a couple spends only a partial amount of time together, then the happiness is proportional to the fraction of time spent. So if Albert and Elaine spend half their time together, they earn happiness of 7/2.*

	E	*F*	*G*	*H*
A	*7*	*5*	*8*	*2*
B	*7*	*8*	*9*	*4*
C	*3*	*5*	*7*	*9*
D	*5*	*5*	*6*	*7*

(a) Let x_{ij} be the fraction of time that man i spends with woman j. Formulate a linear program that maximizes the total happiness of the island (assume that no man spends time with any other man, no woman spends time with any other woman, and no one spends time alone).

(b) Explain why exactly four x_{ij} will be 1 and the rest will be 0 in an optimal solution to this linear program. This result is called the Marriage Theorem *for obvious reasons.*

(c) Do you think that this theorem will hold if we allow men to spend time with men and women to spend time with women?

6.3 The Transshipment Problem

One requirement of the transportation problem is advance knowledge of the method of distributing flow from each source to each destination. This determines the cost. Sometimes, however, the best method of routing the flow is not clear because of the possibility of transshipments.

Continuing our pea shipping example, suppose our company decided to look at using common carriers to ship our goods (common carriers are trucking companies for hire, as opposed to internal trucking divisions). Since no single trucking company serves all of our area, many shipments will need to be transfered to another truck at least once along the way. These transfers can be made at an intermediate cannery or warehouse, or be made at one of three other *junctions*. These are at Butte, Montana; Boise, Idaho; and Cheyenne, Wyoming. The shipping cost per truck is given in the following table:

		Cannery			Junction			Warehouse				Output
		1	2	3	1	2	3	1	2	3	4	
Cannery	1		146	—	324	286	—	452	505	—	871	75
	2	146		—	373	212	570	335	407	688	784	125
	3	—	—		658	—	405	—	685	359	673	100
Junction	1	322	371	656		262	398	503	234	329	—	
	2	284	210	—	262		406	305	207	464	558	
	3	—	569	403	398	406		597	253	171	282	
Warehouse	1	453	336	—	505	307	599		359	706	587	
	2	505	407	683	235	208	254	357		362	341	
	3	—	687	357	329	464	171	705	362		457	
	4	868	781	670	—	558	282	587	340	457		
Demand								80	65	70	85	

So, for instance, peas can go from cannery 1 to warehouse 4 directly at cost 871. They can also go from cannery 1, to junction 2, to warehouse 2, and then to warehouse 4 at cost of $286 + 207 + 341 = 834$.

The overall problem is to determine how the output from each cannery should be shipped to the warehouses to meet demand and minimize shipping costs.

Note that the above table does not represent a transportation problem. It is a *transshipment problem*. We can, however, represent any transshipment problem as an equivalent transportation problem.

The fundamental idea is to interpret the individual truck trips as being the shipment from a source to a destination. Therefore, there is both a source and a destination for all of the places in our problem. There is a source node for cannery 1 (representing everything leaving cannery 1) and a destination node (representing everything entering the node). This gives rise to the following cost table (ignore supplies and demands for now):

		Cannery			Junction			Warehouse				Output
		1	2	3	1	2	3	1	2	3	4	
Cannery	1	0	146	M	324	286	M	452	505	M	871	375
	2	146	0	M	373	212	570	335	407	688	784	425
	3	M	M	0	658	M	405	M	685	359	673	400
Junction	1	322	371	656	0	262	398	503	234	329	M	300
	2	284	210	M	262	0	406	305	207	464	558	300
	3	M	569	403	398	406	0	597	253	171	282	300
Warehouse	1	453	336	M	505	307	599	0	359	706	587	300
	2	505	407	683	235	208	254	357	0	362	341	300
	3	M	687	357	329	464	171	705	362	0	457	300
	4	868	781	670	M	558	282	587	340	457	0	300
Demand		300	300	300	300	300	300	380	365	370	385	

Impossible shipments are represented by a large cost M. Shipments from a place to itself are assigned cost 0.

The final step is to give every source node a supply and every destination node a demand. Now suppose we knew how much would be shipped through a place. If that amount was 50, then we would increase the supply for the source version of that place by 50 and we would increase the demand for the destination version of that place by 50. This would ensure that 50 units were shipped through that place.

We do not know the amount shipped through each node, but we can place an upper bound on the amount. Since it would not pay to ship something through a place twice, a safe upper bound on the amount shipped through the place is the total supply (300 in this case). This value is added to the supply or demand of each node. So if 50 flow is sent through junction 3, then 250 is sent from the source of junction 3 to the destination of junction 3 (at zero cost). In order to meet supply and demand, 50 units of flow must be sent to the demand node of junction 3, and 50 units of flow are available at the source node and must be sent somewhere else. But the flow through junction 3 can be any value from 0 to 300.

By solving this transportation problem, we get the following optimal solution:

300 units from cannery 1 to cannery 1
75 from cannery 1 to junction 2
300 from cannery 2 to cannery 2
80 from cannery 2 to warehouse 1
45 from cannery 2 to warehouse 2
300 from cannery 3 to cannery 3
70 from cannery 3 to warehouse 3
30 from cannery 3 to warehouse 4
300 from junction 1 to junction 1
225 from junction 2 to junction 2
75 from junction 2 to warehouse 2
300 from junction 3 to junction 3
300 from warehouse 1 to warehouse 1
245 from warehouse 2 to warehouse 2
55 from warehouse 2 to warehouse 4
300 from warehouse 3 to warehouse 3
300 from warehouse 4 to warehouse 4.

Removing the shipments that remain in place leaves:

75 from cannery 1 to junction 2
80 from cannery 2 to warehouse 1
45 from cannery 2 to warehouse 2
70 from cannery 3 to warehouse 3
30 from cannery 3 to warehouse 4
75 from junction 2 to warehouse 2
55 from warehouse 2 to warehouse 4

so only junction 2 and warehouse 3 are used to transfer goods.

General structure. The transshipment problem is concerned with how to allocate and route flow (peas in our example) from supply centers to destinations via intermediate points. In addition to transshiping flow, supply centers generate a surplus that must be distributed and each destination generates a given deficit. Intermediate points (or transshipment nodes) neither generate nor absorb flow. The total supply must equal the total demand, so dummy nodes should be added appropriately. No connection may have a capacity, and all costs should be nonnegative. This defines a transshipment problem.

Transshipment problems can be solved via a transformation to the transportation problem.

Supplementary Exercise 1 *A company must meet the following demands for cash at the beginning of each of the next six months:*

Month	*1*	*2*	*3*	*4*	*5*	*6*
Needs	*\$200*	*\$100*	*\$50*	*\$80*	*\$160*	*\$140*

At the beginning of month 1, the company has \$150 in cash and \$200 worth of bond 1, \$100 worth of bond 2 and \$400 worth of bond 3. Of course, the company will have to sell some bonds to meet demands, but a penalty will be charged for any bonds sold before the end of month 6. The penalties for selling \$1 worth of each bond are shown in the table below.

		Month of sale					
		1	*2*	*3*	*4*	*5*	*6*
	1	*\$0.07*	*\$0.06*	*\$0.06*	*\$0.04*	*\$0.03*	*\$0.03*
Bond	*2*	*\$0.17*	*\$0.17*	*\$0.17*	*\$0.11*	*\$0*	*\$0*
	3	*\$0.33*	*\$0.33*	*\$0.33*	*\$0.33*	*\$0.33*	*\$0*

(a) Assuming that all bills must be paid on time, formulate a balanced transportation problem that can be used to minimize the cost of meeting the cash demands for the next six months.

(b) Assume that payment of bills can be made after they are due, but a penalty of \$0.02 per month is assessed for each dollar of cash demands that is postponed for one month. Assuming all bills must be paid by the end of month 6, develop a transshipment model that can be used to minimize the cost of paying the next six months' bills.

[Hint: *Transshipment points are needed, in the form C_t = cash available at beginning of month t after bonds for month t have been sold, but before month t demand is met. Shipments into C_t occur from bond sales and C_{t-1}. Shipments out of C_t occur to C_{t+1} and demands for months $1, 2, \ldots, t$.*]

6.4 Minimum Cost Flow

This is the most general model we will look at. Like the maximum flow problem, it considers flows in networks with capacities. Like the shortest path problem, it considers a cost for flow through an arc. Like the transportation problem, it allows multiple sources and destinations. Like the transshipment problem, it allows nodes between sources and destinations. In fact, all of these problems (all the models you have seen except minimum spanning tree) can be seen as *special cases* of the minimum cost flow problem.

Consider a directed network with n nodes. The decision variables are x_{ij}, the flow through arc (i, j). The given information includes:

- c_{ij}: cost per unit of flow from i to j (may be negative),
- u_{ij}: capacity (or upper bound) on flow from i to j,
- b_i: net flow generated at i.

This last value has a sign convention:

- $b_i > 0$ if i is a supply node,
- $b_i < 0$ if i is a demand node,
- $b_i = 0$ if i is a transshipment node.

The objective is to minimize the total cost of sending the supply through the network to satisfy the demand.

Note that for this model, it is not necessary that every arc exists. We will use the convention that summations are only taken over arcs that exist. The linear programming formulation for this problem is:

$$\begin{array}{ll} \text{Minimize} & \sum_i \sum_j c_{ij} x_{ij} \\ \text{Subject to} & \sum_j x_{ij} - \sum_j x_{ji} = b_i \text{ for all nodes } i, \\ & 0 \le x_{ij} \le u_{ij} \text{ for all arcs } (i, j). \end{array}$$

Again, we will assume that the network is balanced, so $\sum_i b_i = 0$, since dummies can be added as needed. We also still have a nice integrality property. If all the b_i and u_{ij} are integral, then the resulting solution to the linear program is also integral.

Minimum cost network flows are solved by a variation of the simplex algorithm and can be solved more than 100 times faster than equivalently sized linear programs. From a modeling point of view, it is most important to know the sort of things that can and cannot be modeled in a single network flow problem:

Can do

1. Lower bounds on arcs. If a variable x_{ij} has a lower bound of l_{ij}, upper bound of u_{ij}, and cost of c_{ij}, change the problem as follows:
   - Replace the upper bound with $u_{ij} - l_{ij}$,
   - Replace the supply at i with $b_i - l_{ij}$,
   - Replace the supply at j with $b_i + l_{ij}$,

   Now you have a minimum cost flow problem. Add $c_{ij}l_{ij}$ to the objective after solving and l_{ij} to the flow on arc (i, j) to obtain a solution of the original problem.
2. Upper bounds on flow through a node. Replace the node i with nodes i' and i''. Create an arc from i' to i'' with the appropriate capacity, and cost 0. Replace every arc (j, i) with one from j to i' and every arc (i, j) with one from i'' to j. Lower bounds can also be handled this way.
3. Convex, piecewise linear costs on arc flows (for minimization). This is handled by introducing multiple arcs between the nodes, one for each portion of the piecewise linear function. The convexity will assure that costs are handled correctly in an optimal solution.

Can't do

1. Fixed cost to use a node.
2. Fixed cost to use an arc.
3. "Proportionality of flow." That is, if one unit enters node i, then you insist that .5 units go to node j and .5 to node k.

4. Gains and losses of flow along arcs, as in power distribution.

Note that although these cannot be done in a single network, it may be possible to use the solutions to multiple networks to give you an answer. For instance, if there is only one arc with a fixed cost, you can solve both with and without the arc to see if it is advantageous to pay the fixed cost.

7 PERT/CPM

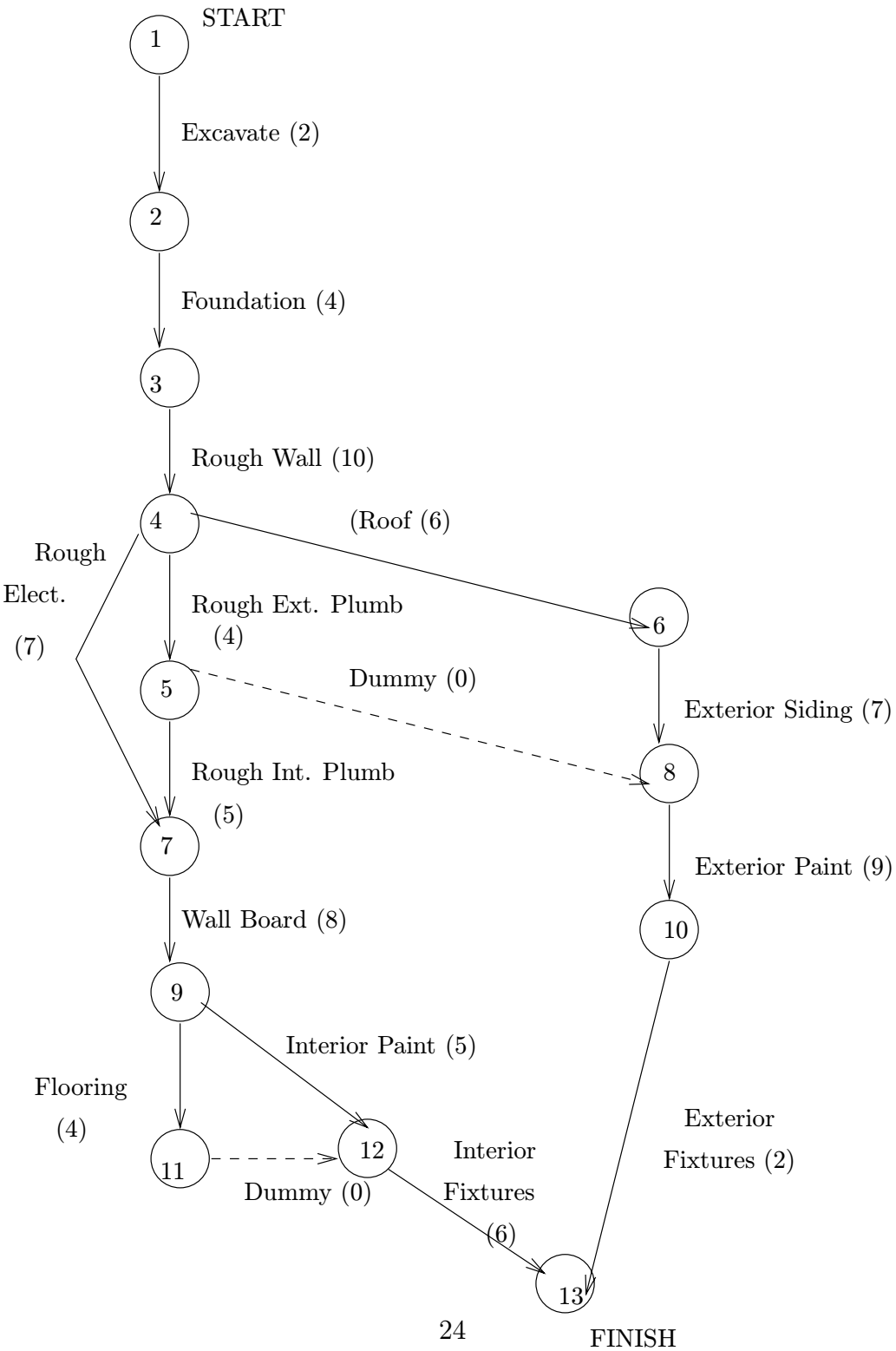
The final network models we discuss are the PERT and CPM models for project scheduling. The successful management of large projects, be they construction, transportation, financial, or what-have-you, relies on careful scheduling and coordinating of various tasks. CPM (Critical Path Method) and PERT (Project Evaluation and Review Technique) attempt to analyze project scheduling. This allows for better control and evaluation of the project. For instance, CPM allows you to answer such questions as:

- How long will the project take?
- When will we be able to start a particular task?
- If this task is not completed on time, will the entire project be delayed?
- Which tasks should we speed up (crash) in order to finish the project earlier?

CPM and PERT have been used in a wide variety of projects, from designing and producing computers to scheduling a complex surgery. The essential ideas behind these techniques are quite straightforward (not as complicated as maximum flow for instance). We will begin with CPM.

CPM uses a **project network** to portray graphically the relationships among the tasks in a project. This is illustrated in the following diagram for building a house. Here excavation must be done before foundation and the foundation done before the rough wall. After that, three tasks may be done (rough electrical, rough exterior plumbing, and roof).

Each arc of the project represents a task, or **activity**. This is the activity on arc (AOA) representation. Each node represents an event (like finishing foundation is node 3). The direction of the arcs gives the sequences in which


START
1
Excavate (2)
2
Foundation (4)
3
Rough Wall (10)
4
(Roof (6)
Rough
Elect.
(7)
Rough Ext. Plumb
(4)
6
5
Dummy (0)
Exterior Siding (7)
8
Rough Int. Plumb
(5)
7
Exterior Paint (9)
10
Wall Board (8)
9
Interior Paint (5)
Flooring
(4)
Exterior
Fixtures (2)
11
12
Interior
Fixtures
(6)
Dummy (0)
13
FINISH

activities must be achieved. An event must precede the beginning of successive activities, and an event occurs only when all the activities immediately preceding it occur.

There are two special nodes, the start node and the finish node. The start node has no activities enter it; the finish node has no activity leave it.

Each arc has two roles: it represents an activity and it defines the precedence relationships among the activities. Sometimes it is necessary to add arcs that only represent precedence relationships. These **dummy arcs** are represented by dashed arrows. In our example, the arc from 5 to 8 represents the fact that exterior plumbing must be completed in order to begin exterior painting.

There are generally rules restricting the form of a project network. Some of these are:

- An activity is represented by no more than one arc,
- Two nodes can be connected by no more than one arc,
- The project network can have no directed cycles.

To satisfy the second requirement, dummy nodes can be added (like node 11). Because of the third requirement, we can number the nodes so that if one node precedes another then the first gets a lower number than the second (as in our example).

Once we have the form of a project network, we can estimate the time required by each activity. Dummy activities get time 0. This number is in parentheses on the preceding diagram. These times are used to calculate two basic quantities: the **early time** and the **late time**.

The early time is the earliest time for an event assuming each activity is started as soon as possible. This is obtained by making a forward pass through the network. The start of the project is set to have early time 0. Every following event gets an early time based on the latest time a preceding event finishes. This is illustrated in table 3.1.

The late time for an event is the latest time an event can occur without delaying the completion of the project beyond its early time. In this project, it takes 44 days to build a house. What is the latest the roofing can finish (or equivalently the latest exterior siding can start) without delaying the entire project?

The late time is calculated in a manner analogous to the early time. This time, begin at the end and work backwards. Here is the result for this project.

Event	Preceding Event	Early Time + Activity Time	Early Time
1	—	—	0
2	1	0+2	2
3	2	2+4	6
4	3	6+10	16
5	4	16+4	20
6	4	16+6	22
7	4	16+7	
	5	20+5	25
8	5	20+0	
	6	22+7	29
9	7	25+8	33
10	8	29+9	38
11	9	33+4	37
12	9	33+5	38
	11	37+0	
13	10	38+2	
	12	38+6	44

Table 1: Calculation of Early Time

Event	Following Event	Late Time - Activity Time	Late Time
13	—	—	44
12	13	44–6	38
11	12	38–0	38
10	13	44–2	42
9	12	38–5	33
	11	38–4	
8	10	42–9	33
7	9	33–8	25
6	8	33–7	26
5	8	33–0	
	7	25–5	20
4	7	25–7	
	6	26–6	
	5	20–4	16
3	4	16–10	6
2	3	6–4	2
1	2	2–2	0

Table 2: Calculation of Late Time

Therefore, roofing must be done by time 26 or the completion of the building will be delayed. Note that it can finish as early as 22 days and as late as 26 days without affecting the completion time. Associated with each activity is a value called the **total float** for the activity. The total float for an activity is the amount of time an activity can be delayed beyond its earliest time without affecting the final completion time of the project. The total float for an arc from i to j is the difference between the late time of job j and (the early time for job i plus the activity time).

An alternative measure for the flexibility of an activity is the **free float**. This is the amount of time an activity can be delayed without pushing *any* job past its early time. This is calculated as the difference between the early time of job j and (the early time of job i plus the activity time). The following table illustrates this concept.

Activity	Total Float	Free Float
(1,2)	2–(0+2) = 0	2–(0+2) = 0
(2,3)	6–(2+4) = 0	6–(2+4) = 0
(3,4)	16–(6+10) = 0	16–(6+10 = 0
(4,5)	20–(16+4) = 0	20–(16+4) = 0
(4,6)	26–(16+6) = 4	22–(16+6) = 0
(4,7)	25–(16+7) = 2	25–(16+7) = 2
(5,7)	25–(20+5) = 0	25–(20+5) = 0
(6,8)	33–(22+7) = 4	29–(22+7) = 0
(7,9)	33–(25+8) = 0	33–(25+8) = 0
(8,10)	42–(29+9) = 4	38–(29+9) = 0
(9,11)	38–(33+4) = 1	37–(33+4) = 0
(9,12)	38–(33+5) = 0	38–(33+5) = 0
(10,13)	44–(38+2) = 4	44–(38+2) = 4
(12,13)	44–(38+6) = 0	44–(38+6) = 0

Table 3: Calculation of Float

If an activity with a total float of 0 is delayed for whatever reason, then the entire project is delayed. Such an activity is called a **critical activity**. Any path (there may be more than one) from the start node to the finish node made up solely of critical activities is a **critical path**. In our example, a critical path is 1-2-3-4-5-7-9-12-13.

This information on early and late times, critical jobs, and critical paths in invaluable to a manager, for it allows her to investigate the effect of possible improvements to the project plan.

Using Linear Programming

It is possible to use linear programming to find a critical path and to determine the overall completion time. Although this approach is much slower than the above algorithm, we will be able to extend the model to identify activities that should be "crashed" (speeded up, perhaps at some cost).

To do this, let x_j be the time that event j occurs. The objective, for this example is to minimize $x_{13} - x_1$. Each activity corresponds to a constraint. For instance, the activity between 1 and 2 is the constraint

$$x_2 \geq x_1 + 2.$$

The other constraints are

$$\begin{aligned}
x_3 &\geq x_2 + 4 \\
x_4 &\geq x_3 + 10 \\
x_5 &\geq x_4 + 4 \\
x_6 &\geq x_4 + 6 \\
x_7 &\geq x_4 + 7 \\
x_7 &\geq x_5 + 5 \\
x_8 &\geq x_5 + 0 \\
x_8 &\geq x_6 + 7 \\
x_9 &\geq x_7 + 8 \\
x_{10} &\geq x_8 + 9 \\
x_{11} &\geq x_9 + 4 \\
x_{12} &\geq x_9 + 5 \\
x_{12} &\geq x_{11} \\
x_{13} &\geq x_{10} + 2 \\
x_{13} &\geq x_{12} + 6
\end{aligned}$$

If x_1 is set to 0, optimizing this linear program will give the optimal completion time. Furthermore, one critical path will be identified. The arcs on this path will correspond to constraint with a shadow price of -1.

Crashing the Project It is easy then to use the above linear programming formulation to determine which projects to decrease the time for. Suppose we

need to complete the house in 40 days, and we can decrease the excavation by at most 1 day at cost \$500/day, foundation by at most 2 days at cost \$600 per day, rough walling by at most 2 days at cost \$400 per day, and exterior siding by at most 4 days at cost \$300 per day decreased. How can we minimize cost so that the house is finished in 40 days?

If we let new variables z_1, z_2, z_3, and z_4 represent the number of days we decrease the time needed for excavation, foundation, walling, and siding, respectively, then we get the L.P.:

$$
\begin{array}{ll}
\text{Minimize} & 500z_1 + 600z_2 + 400z_3 + 300z_4 \\
\text{subject to} & x_2 \geq x_1 + 2 - z_1 \\
& x_3 \geq x_2 + 4 - z_2 \\
& x_4 \geq x_3 + 10 - z_3 \\
& x_5 \geq x_4 + 4 \\
& x_6 \geq x_4 + 6 \\
& x_7 \geq x_4 + 7 \\
& x_7 \geq x_5 + 5 \\
& x_8 \geq x_5 + 0 \\
& x_8 \geq x_6 + 7 - z_4 \\
& x_9 \geq x_7 + 8 \\
& x_{10} \geq x_8 + 9 \\
& x_{11} \geq x_9 + 4 \\
& x_{12} \geq x_9 + 5 \\
& x_{12} \geq x_{11} \\
& x_{13} \geq x_{10} + 2 \\
& x_{13} \geq x_{12} + 6 \\
& x_1 = 0 \\
& x_{13} - x_1 \leq 40 \\
& z_1 \leq 1 \\
& z_2 \leq 2 \\
& z_3 \leq 2 \\
& z_4 \leq 4
\end{array}
$$

PERT

So far we have assumed that reasonably accurate estimates can be made of the time required for each activity in the project network. In reality, ther is frequently some uncertainty about the time an activity can take.

In a PERT network, this uncertainty is summed up in three numbers about each activity: the most likely value for the duration (m), a pessimistic

value (b) and an optimistic value (a). PERT then fits a particular type of probability distribution to these values. This distribution (the *beta* distribution) assumes that the range from a and b encompasses 6 standard deviations (3 on either side of the mean). The mean itself is calculated as

$$\text{Mean} = \frac{a + 4m + b}{6}$$

and the variance:

$$\text{Variance} = \frac{(b-a)^2}{36}.$$

Based on these values, PERT will use an activity network to calculate a mean finishing time along with a variance about that finishing time. There are two critical assumptions: the times for the activities are *independent* of each other, and the critical path identified is always the longest path in the network, no matter how the activity lengths turn out.

With these assumptions, you can solve a PERT network as follows. Find a critical path using the CPM method with the mean activity times on the arcs. This gives the mean finishing time. The variance of the finishing time is simply the sum of the variances of the activities on the critical path. The overall finishing value is assumed to be normally distributed, so quantiles are based on the normal distribution.

PERT allows you to answer such questions as:

- What is the probability the total time is less than 40 days?
- What is the probability the total time is more than 50 days?
- How much should the project budget be increased in order to ensure with 99% probability that the project will finish by December 10?

Unfortunately, the assumptions behind PERT are very stringent. Be careful when using any canned package for it hides the assumptions very well. Used correctly, both PERT and CPM greatly aid in the control and analysis of large projects.

Exercise 7 *Consider the following project.*

Activity	*Duration*	*Predecessors*
A	6	—
B	3	—
C	5	—
D	4	A
E	5	A
F	7	B
G	4	C
H	6	C
I	3	D
J	4	F, G
K	3	F, G
L	2	H
M	6	E, I, J
N	5	K, L

Find the early time and late time for each event as well as the total float for each activity. Also identify the critical path.