Eg 5.8

$$\max \quad \overset{c_1}{2x_1} + \overset{c_2}{7x_2} + \overset{c_3}{3x_3} \qquad \overset{②}{\vec{c}}^T\vec{x}$$

$$\text{s.t.} \begin{cases} 1x_1 + 3x_2 + 4x_3 \le 30 \\ 1x_1 + 4x_2 - 1x_3 \le 10 \end{cases} \qquad \text{s.t.} \begin{cases} \overset{③}{A}\vec{x} = \overset{①}{\vec{b}} \\ \vec{x} \ge 0 \end{cases}$$

$$x_1, x_2, x_3 \ge 0$$

Initial Tableau

$$\begin{array}{c|ccccc|c} & x_1 & x_2 & x_3 & x_4 & x_5 & \vec{b} \\ \hline x_4 & 1 & 3 & 4 & 1 & 0 & 30 \\ x_5 & 1 & 4 & -1 & 0 & 1 & 10 \\ \hline x_0 & -2 & -7 & 3 & 0 & 0 & 0 \end{array}$$

Optimal Tableau

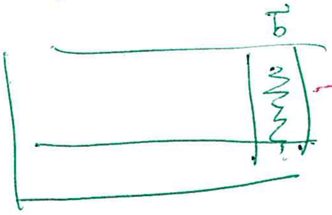
$$\begin{array}{c|ccccc|c} & x_1 & x_2 & \downarrow x_3 & x_4 & x_5 & \vec{b} \\ \hline x_4 & 0 & -1 & 5 & 1 & -1 & 20 \\ x_1 & 1 & 4 & -1 & 0 & 1 & 10 \\ \hline x_0 & 0 & 1 & 1 & 0 & 2 & 20 \end{array}$$

$$B = \begin{array}{cc} \vec{a}_4 & \vec{a}_1 \\ \end{array} \hspace{-4em} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad \vec{x}_B = \begin{pmatrix} 20 \\ 10 \end{pmatrix} \begin{matrix} x_4 \\ x_1 \end{matrix} \qquad \vec{c}_B = (\underset{c_4}{0}, \underset{c_1}{2})$$

$$\vec{x}_B = B^{-1}\vec{b}$$

$$B^{-1} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$$

(i) Change the $\vec{b}$ ⇒ $\vec{x}_B$ ⇒ x_0 Changed (feasiblity) last column



if ≥ 0 then done

if < 0 then not feasible but optimal

→ dual simplex method

(ii) Change in $\vec{c}^T \Longrightarrow$ (optimality) last row x_0-row.

Eg $\vec{c}_{original} = (2\ 7\ 3\ 0\ 0) \to \hat{\vec{c}}_{new} = (3\ 6\ -3\ 0\ 0)$

$\vec{c}_B = (0,2)$ $\qquad \hat{c}_B = (0, 3)$

$\qquad\qquad\qquad\qquad\qquad\quad \uparrow$
$\qquad\qquad\qquad\qquad\qquad\quad \hat{c}_4 \ \ \hat{c}_1$

$$\hat{Z}^T = \hat{C}_B Y = (0,3)\,Y$$
$$= (3\ 12\ -3\ 0\ 3)$$
$$\hat{Z}^T - \hat{c}^T = (3\ 12\ -3\ 0\ 3) - (3\ 6\ -3\ 0\ 0)$$
$$= (0, 6, 0, 0, 3) \geq 0$$

$$\begin{array}{ccccc|c} & & & & & 20 \\ & & & & & 10 \\ \hline 0 & 6 & 0 & 0 & 3 & 30 \end{array}$$

$$\underset{\hat{C}_B^T}{(0,3)} \underset{\vec{X}_B}{\begin{pmatrix} 20 \\ 10 \end{pmatrix}} = X_{0\,new}.$$

(iii) Case 1 entries in B are changed

(i.e. entries in A underneath basic variables are unchanged)

Eg If $\vec{a}_4$ or $\vec{a}_1$ is changed

then Redo the whole thing

Case 2 entries in A underneath the nonbasic variables are changed. (i.e. B is unchanged) g 5.8 (iii)

initial tableau

	x_1	x_2	x_3	x_4	x_5	$\vec{b}$
x_4	1	3	4	1	0	30
x_5	1	4	-1	0	1	10
	-2	-7	3	0	0	0

optimal tableau

	x_1	x_2	x_3	x_4	x_5	$\vec{b}$
x_4	0	-1	5	1	-1	20
x_1	1	4	-1	0	1	10
x_0	0	1	1	0	2	20

$$B = \begin{array}{c} \begin{matrix} \vec{a}_4 & \vec{a}_1 \end{matrix} \\ \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \end{array} \qquad B^{-1} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$$

Suppose

$$A = \begin{pmatrix} 1 & 3 & 4 & 1 & 0 \\ 1 & 4 & -1 & 0 & 1 \end{pmatrix} \longrightarrow \hat{A} = \begin{pmatrix} 1 & \overset{\hat{a}_2}{1} & 4 & 1 & 0 \\ 1 & 3 & -1 & 0 & 1 \end{pmatrix}$$

$$Y = \begin{pmatrix} 0 & -1 & 5 & 1 & -1 \\ 1 & 4 & -1 & 0 & 1 \end{pmatrix} \longrightarrow \hat{Y} = \begin{pmatrix} 1 & \overset{\hat{y}_2}{-2} & 4 & 1 & 0 \\ 1 & 3 & -1 & 0 & 1 \end{pmatrix}$$

$$\left(\because \ \hat{A} = B\hat{Y} \ \Rightarrow \ \hat{a}_2 = B\hat{y}_2 \ \Rightarrow \ \hat{y}_2 = B^{-1}\hat{a}_2 = \begin{pmatrix} -2 \\ 3 \end{pmatrix} \right)$$

$$\begin{aligned} \vec{z} - \vec{c} &= \vec{c}_B^{\,T} Y - \vec{c} \\ &= (0,2)\,Y - \vec{c} \\ &= (2\ 8\ -2\ 0\ 0) - (2\ 7\ -3\ 0\ 0) \\ &= (0\ 1\ 1\ 0\ 0) \end{aligned} \quad \longrightarrow \quad \begin{aligned} \hat{z} - \vec{c} &= \vec{c}_B^{\,T} \hat{Y} - \vec{c} \\ &= (0,2)\,\hat{Y} - \vec{c} \\ &= (2\ 6\ -2\ 0\ 0) - (2\ 7\ -3\ 0\ 0) \\ &= (0,-1,1,0,0) \end{aligned}$$

Thus the new tableau is

	x_1	x_2	x_3	x_4	x_5	$\vec{b}$
x_4	0	-2	5	1	-1	20
x_1	1	3^*	-1	0	1	10
x_0	0	-1	1	0	2	20

After 1 iteration, we get the optimal tableau

	x_1	x_2	x_3	x_4	x_5	$\vec{b}$
x_4	$2/3$	0	$13/3$	1	$-1/3$	$80/3$
x_2	$1/3$	1	$-1/3$	0	$1/3$	$10/3$
x_0	$1/3$	0	$2/3$	0	$7/3$	$70/3$

#

Adding a new variable:

original

$$\max \ \vec{c}^T\vec{x}$$
$$\begin{cases} A\vec{x} = \vec{b} \\ \vec{x} \geq \vec{0} \end{cases}$$

new

$$\max \ \vec{c}^T\vec{x} + c_n \cdot x_n$$
$$\begin{cases} A\vec{x} + \vec{a}_n x_n = \vec{b} \\ \vec{x}, x_n \geq 0 \end{cases}$$

$$\begin{array}{c|c} A & \vec{b} \\ \hline -\vec{c}^T & 0 \end{array} \qquad\qquad \begin{array}{c|c|c} A & \vec{a}_n & \vec{b} \\ \hline -\vec{c}^T & -c_n & 0 \end{array}$$

$$\Big\downarrow \qquad\qquad\qquad\qquad \Big\downarrow$$

$$\begin{array}{c|c} Y & \vec{x}_B = B^{-1}\vec{b} \\ \hline -(\vec{c}^T - \vec{z}^T) & x_0 = \vec{c}_B^T\vec{x}_B \end{array} \qquad\qquad \begin{array}{c|c|c} Y & \vec{y}_n & \vec{x}_B = B^{-1}b \\ \hline -(\vec{c}^T - \vec{z}^T) & (c_n - z_n) & x_0 \end{array}$$

$x_n = 0$

$\Downarrow$

$A\vec{x} = \vec{b}$

$\Downarrow$

$\vec{x}_B$ is still feasible

$$\begin{cases} \vec{y}_n = B^{-1}\vec{a}_n, \\ z_n = \vec{c}_B^T B^{-1}\vec{a}_n \end{cases}$$

feasible but not neccessarily optimal.

Eg 5.8 Suppose

initial

$$\begin{array}{c|ccccc|c|c} & & & & & & x_6 & \\ \hline x_4 & 1 & 3 & 4 & 1 & 0 & 1 & 30 \\ x_5 & 1 & 4 & -1 & 0 & 1 & 1 & 10 \\ \hline & -2 & -7 & 3 & 0 & 0 & -4 & 0 \end{array} \longrightarrow$$

optimal

$$\begin{array}{c|ccccc|c|c} & & & & & & x_6 & \\ \hline x_4 & 0 & -1 & 5 & 1 & -1 & 0 & 20 \\ x_1 & 1 & 4 & -1 & 0 & 1 & 1 & 10 \\ \hline & 0 & 1 & 1 & 0 & 2 & -2 & 20 \end{array}$$

feasible but not optimal

$$\vec{y}_6 = B^{-1}\vec{a}_6 = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$z_6 = \vec{c}_B^T\vec{y}_6 = (0, 2)\begin{pmatrix} 0 \\ 1 \end{pmatrix} = 2$$

$$-(c_6 - z_6) = -(4-2) = -2$$

P5

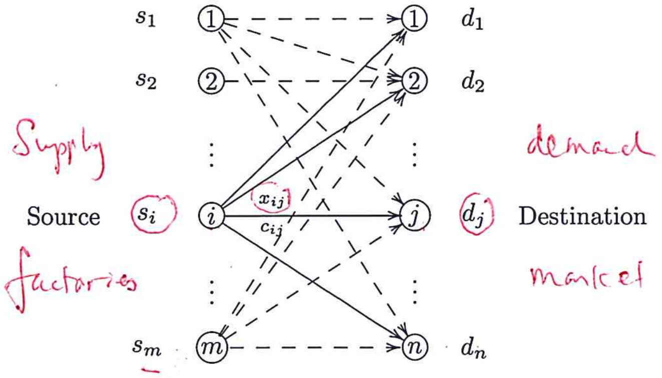
Chapter 6

TRANSPORTATION PROBLEMS

LPP

6.1 Transportation Model

Transportation models deal with the determination of a minimum-cost plan for transporting a commodity from a number of sources to a number of destinations. To be more specific, let there be m *sources* (or *origins*) that produce the commodity and n *destinations* (or *sinks*) that demand the commodity. At the i-th source, $i = 1, 2, \cdots, m$, there are s_i units of the commodity available. The demand at the j-th destination, $j = 1, 2, \cdots n$, is denoted by d_j. The cost of transporting one unit of the commodity from the i-th source to the j-th destination is c_{ij}. Let x_{ij}, $1 \leq i \leq m, 1 \leq j \leq n$, be the numbers of the commodity that are being transported from the i-th source to the j-th destination. Our problem is to determine those x_{ij} that will minimize the overall transportation cost. An optimal solution x_{ij} to the problem is called a *transportation plan.*



We note that at the i-th source, we have the i-th source equation

$$\sum_{j=1}^{n} x_{ij} \overset{\leq}{=} s_i, \qquad 1 \leq i \leq m,$$

while at the j-th destination, we have the j-th destination equation

$$\sum_{i=1}^{m} x_{ij} \overset{\leq}{=} d_j, \qquad 1 \leq j \leq n.$$

Notice that if the total demand equals the total supply, then we have the following *balanced trans-*

P6

portation equation:

$$\sum_{i=1}^{m} s_i = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij} = \sum_{j=1}^{n} d_j .$$

and the model is said to be *balanced.*

In the case of an unbalanced model, i.e. the total demand is not equal to the total supply, we can always add dummy source or dummy destination to complement the difference. In the following, we only consider balanced transportation models. They can be written as the following linear programming problem:

$$\begin{array}{ll} \min & x_0 = \underline{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}} \\ \text{subject to} & \left\{ \begin{array}{ll} \sum_{j=1}^{n} x_{ij} = s_i & 1 \le i \le m, \\ \sum_{i=1}^{m} x_{ij} = d_j & 1 \le j \le n, \\ x_{ij} \ge 0 & 1 \le i \le m, 1 \le j \le n, \end{array} \right. \end{array} \tag{6.1}$$

LPP

where $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$.

Notice that there are mn variables but only $m+n$ equations. To initiate the simplex method, we have to add $m+n$ more artificial variables and solving the problem by the simplex method seems to be a very tedious task even for moderate values of m and n. However, the transportation models possess some important properties that make the calculation easier to be handled.

Using the vector notations

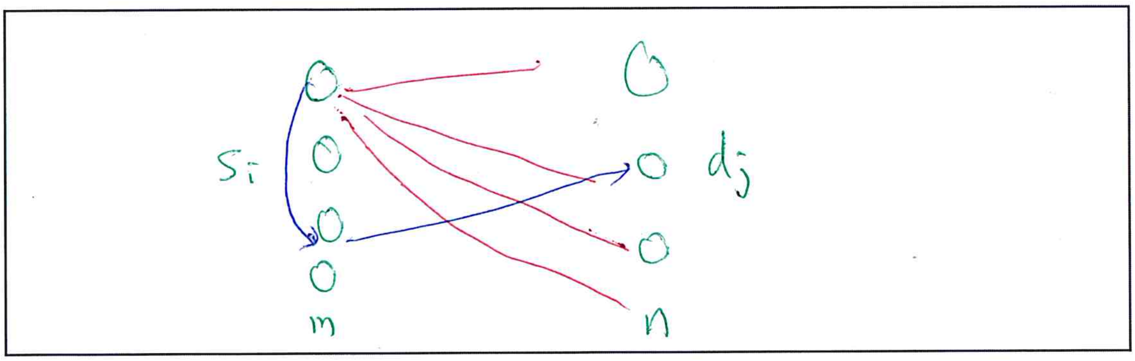
$$\begin{aligned} \mathbf{x} &= [x_{11}, x_{12}, x_{13}, \cdots, x_{1n}, x_{21}, \cdots, x_{2n}, \cdots, x_{m1}, \cdots, x_{mn}]^T, \\ \mathbf{c} &= [c_{11}, c_{12}, c_{13}, \cdots, c_{1n}, c_{21}, \cdots, c_{2n}, \cdots, c_{m1}, \cdots, c_{mn}]^T, \\ \mathbf{b} &= [s_1, s_2, \cdots, s_m, d_1, d_2, \cdots, d_n]^T, \end{aligned}$$

the transportation model can be stated as the following linear programming problem:

$$\begin{array}{ll} \min & x_0 = \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \left\{ \begin{array}{l} A\mathbf{x} = \mathbf{b}, \\ \mathbf{x} \ge \mathbf{0}. \end{array} \right. \end{array} \tag{6.2}$$

where the technology matrix A of the model is of the form:

$$A = \left[\begin{array}{cccccccccccccccccc}
1 & 1 & \cdots & 1 & & & & & & & & & & & & & & 0 \\
 & & & & 1 & 1 & \cdots & 1 & & & & & & & & & & \\
 & & & & & & & & \ddots & & & & & & & & & \\
 & & & & & & & & & 1 & 1 & \cdots & 1 & & & & & \\
 & & & & & & & & & & & & & \ddots & & & & \\
 & & & & & & & & & & & & & & 1 & 1 & \cdots & 1 \\
1 & & & & 1 & & & & & 1 & & & & & 1 & & & \\
 & 1 & & & & 1 & & & & & 1 & & & & & 1 & & \\
 & & \ddots & & & & \ddots & & \cdots & & & \ddots & & \cdots & & & \ddots & \\
0 & & & 1 & & & & 1 & & & & & 1 & & & & & 1
\end{array}\right] \tag{6.3}$$



6.6 Transshipment Model

The standard transportation model assumes that the direct route between a source and a destination is a minimum-cost route. However, in actual application, the minimum-cost route is not known a priori. In fact, the minimum-cost route from one source to another destination may well pass through another source first. The transportation techniques we developed in the previous sections can be adapted to find the minimum-cost route systematically.

The idea is to formulate the problem of finding the minimum-cost route as a *transshipment model* and then solve the transshipment model by transportation techniques. In transshipment model, commodities are allowed to pass transiently through other sources and destinations before it ultimately reaches its designated destination. It therefore is capable of seeking the minimum-cost route between a source and a destination.

To put the transshipment model in the context of transportation problem, we note that in transshipment model, the entire supply from all sources could potentially pass through any source or destination before it is redistributed again. This means that each source or destination node in the transportation network can be considered both as a transient source and a transient destination. Thus the number of sources equals to the number of destinations in the transshipment models and that number is equal to the sum of sources and destinations in the corresponding transportation models.

To allow transient passing of the commodity, an additional buffer stock B has to be allowed at each source and destination. Since potentially, the entire supply from all sources could pass through any one of the node, the size of the buffer stock has to be at least equals to the sum of supply or demand of the transportation model, i.e.

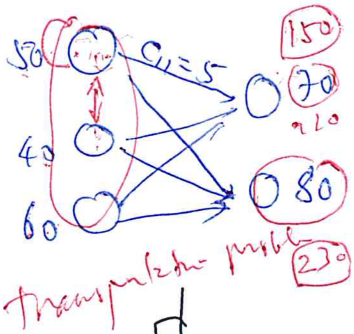
$$B \geq \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j.$$

This amount of buffer stock has to be added to each source and destination nodes in the transportation network.

Before one can use the transportation technique to solve the transshipment model, one has to determine the unit cost of shipping the commodities through the transient nodes. In general, the shipping cost from one location to itself should be zero and the shipping cost from the source S_i to the destination D_j should be the same as the shipping cost from D_j to S_i, but that may change depending on the problem. However, one should note that the unit shipping cost from a source to another source or from a destination to another destination is in general not given in the original transportation problem. Thus these figures have to be given before the transshipment models is completed.

P8

Example 6.7. Consider the following transportation problem.



	D_1	D_2	
S_1	5	7	50
S_2	12	8	40
S_3	7	9	60
	70	80	

c_{ij}

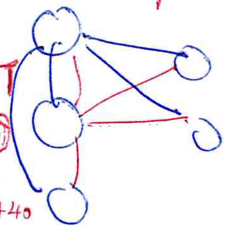
S_i

d_j

To formulate the problem as a transshipment problem, we assume that the commodities can pass through any one of the nodes in the network before they finally reach their destinations. We suppose further that the cost is the same for shipments in opposite directions and unit cost of shipment amongst the sources is 10 while amongst destinations is 5. The transshipment problem is thus changed into the following transportation problem.

	S_1	S_2	S_3	D_1	D_2	
S_1	0	10	10	5	7	200
S_2	10	0	10	12	8	190
S_3	10	10	0	7	9	210
D_1	5	12	7	0	5	150
D_2	7	8	9	5	0	150
	150	150	150	220	230	

~~Transsh~~



50+T

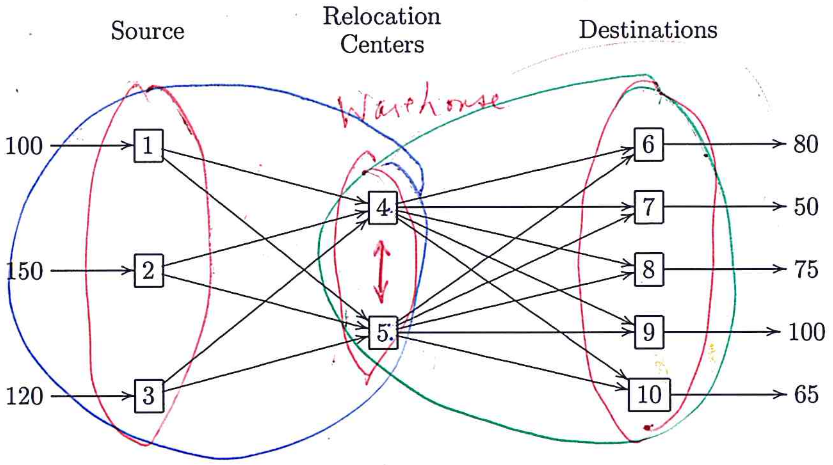
150+40

Transshipment problem

Sometimes in a transportation problem, the commodities have to be shipped to an relocation centers before they are shipped to their final destinations. In that case, only the relocation centers can act as both a destination and a source. Clearly, the size of the buffer stock at these relocation centers should be the same as the total supply of the transportation problem. Notice that if the commodities are not allowed to be shipped from source S_i to destination D_j directly, then the unit cost for such a shipment should be set to an arbitrarily large number, i.e. $c_{ij} = M \gg 1$.

p9

Example 6.8. Consider the following transportation network.



The buffer size at the relocation centers should be set to 370. The corresponding transportation tableau is given as follows:

	4	5	6	7	8	9	10	
1			M	M	M	M	M	100
2			M	M	M	M	M	150
3			M	M	M	M	M	120
4	0							370
5		0						370
	370	370	80	50	75	100	65	

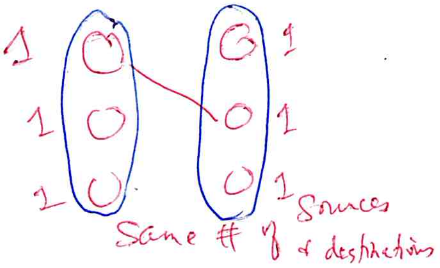
$M >> 1$

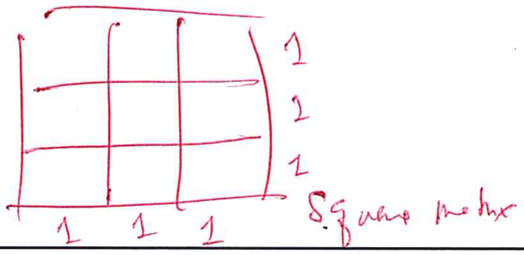
LPP

6.7 Assignment Problems

Consider assigning n jobs to n machines such that one job is assigned to one machine and one machine gets only one job. Thus the total number of possible assignments is $n!$. A cost c_{ij} is associated with assigning job i to machine j, $i = 1, 2, \cdots, n$; $j = 1, 2, \cdots, n$. The least total cost

P10





assignment is then a (zero-one) linear program as follows:

$$\max \qquad x_0 = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{subject to} \begin{cases} \displaystyle\sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, 2, \cdots, n) \quad \text{(one job sets one machines)} \\ \displaystyle\sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, 2, \cdots, n) \quad \text{(one machine gets one job)} \\ \quad x_{ij} = 0, 1 \quad (1 = 1, 2, \cdots, n;\ j = 1, 2, \cdots, n) \end{cases}$$

Notice that both the transportation problem and the assignment problem have *totally unimodualr* coefficient matrices, i.e. the determinants of all their square submatrices are equal to $0, +1$ or -1. This implies that both problems have the integer value property and hence all BFS solutions, in particular, the optimal solutions, are integer-valued even if the integral constraints are discarded, provided that all s_i and d_j are integers in the case of transportation problems.

Standard LP methods and the transportation techniques are applicable for assignment problems with $x_{ij} = 0, 1$ replaced by $0 \le x_{ij} \le 1$. However, there is a much more efficient direct method generally known as the *assignment algorithm*. The key observation of the assignment algorithm is that without loss of generality, we may assume that the cost $c_{ij} \ge 0$ for all i, j.

Theorem 6.5. *Let C be the cost matrix with entries c_{ij}, $1 \le i \le m$ and $1 \le j \le n$. If a constant is added to or subtracted from any row or column of C, giving C'; the minimization of the modified objective function $x_0{}' = \sum_{i,j} c_{ij}{}' x_{ij}$ yields the same solution x_{ij} as the original objective function $x_0 = \sum_{i,j} c_{ij}x_{ij}$.*

Proof. Suppose p_i is added to row i and q_j is subtracted from column j. Then

$$\begin{aligned} x_0{}' &= \sum_{i,j} c_{ij}{}' x_{ij} = \sum_{i,j} (c_{ij} \pm p_i \pm q_j) x_{ij} \\ &= \sum_{i,j} c_{ij}x_{ij} \pm \sum_i p_i \sum_j x_{ij} \pm \sum_j q_j \sum_i x_{ij} \\ &= \sum_{i,j} c_{ij}x_{ij} \pm (\sum_i p_i \pm \sum_j q_i) \\ &= x_0 + \text{constant} \end{aligned}$$

□

We use this idea to create a new coefficient matrix C' with at least one zero element in each row and in each column, and if using *zero elements only* (or a subset of which) yields a feasible assignment (with total cost = 0, of course), then this assignment is optimal because the total cost of any feasible assignment is nonnegative, since $c'_{ij} \ge 0$ for all ij.

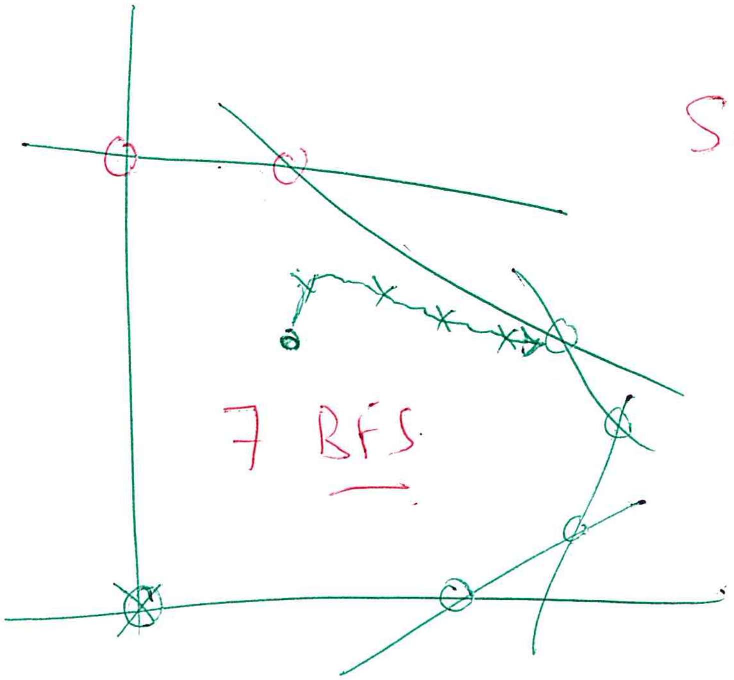
To determine if the zero elements alone can yield a feasible assignment solution, we first cover the cost matrix C' by lines. Define *cover* c to be the minimum number of lines that can cover all zero elements. Then $c \le n$. If $c = n$, then we have an assignment on only the zero elements. The actual assignment of jobs to machines is obtained by a *trace-back* as follows:

Let

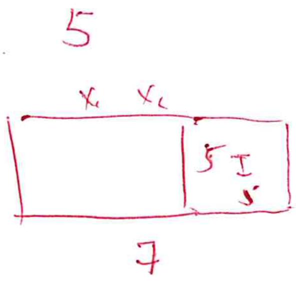
$$z_{ij} \equiv \text{number of zeros in row } i + \text{ column } j,$$

where the (i, j)th entry is zero. Make successive assignments in *increasing* z_{ij} order. Delete row i and column j upon assignment i-j is made.

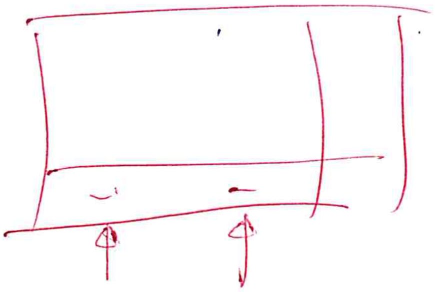
Simplex Method

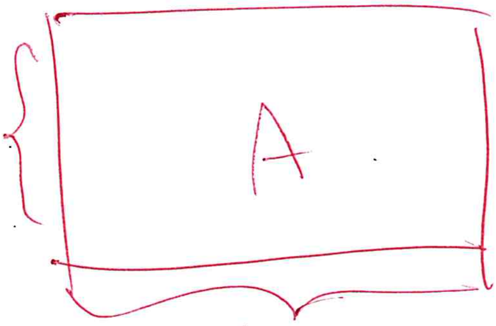


FCF



$C^7_5 = \frac{7\cdot\cancel{6}\cancel{5}3}{2\cdot 1} = 21$





m constraints

max # (Basic Soln) C^n_m

n variables

Basic Feasible Soln

$B = \underbrace{\begin{bmatrix} \quad \end{bmatrix}}_{m} m$

$$\overset{A}{\begin{bmatrix} B & R \end{bmatrix}} \begin{bmatrix} X_B \\ X_N \end{bmatrix} = \vec{b} \Rightarrow B\vec{X}_B = \vec{b}$$

$X_N = 0 \Rightarrow \vec{X}_B = B^{-1}\vec{b}$

$\vec{X}_B < 0$ non feasible

P12

Chapter 10

Taha

Interior-Point Methods for Linear Programming

We studied two pivoting algorithms for linear programming in Chapter 4. These algorithms are finite and the simplex method in particular is known to be very efficient practically. Yet, there is no known pivoting algorithm that is polynomial. There are pathological examples of linear programs for which the simplex method (or the criss-cross method) behaves badly, i.e., generates an exponential number of pivots to find an optimal solution. There is a famous example due to Klee and Minty, [28, 35]. An explicit description is given by

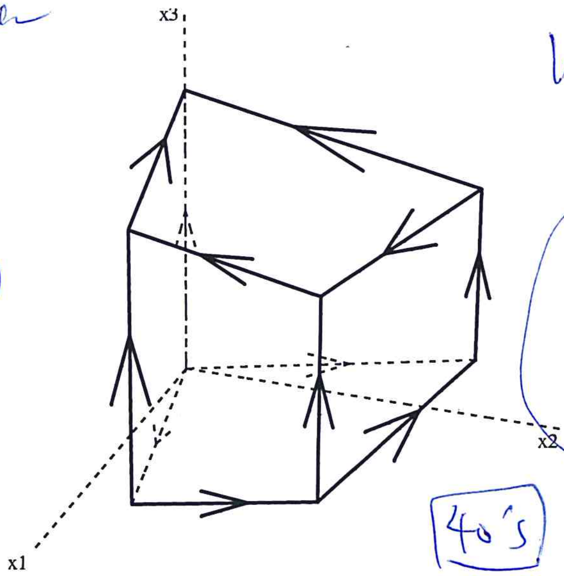
$$
\begin{array}{llrcl}
 & \max & & x_n & \\
(10.1) & \text{subject to} & 0 \leq & x_1 & \leq 1, \\
 & & \epsilon\, x_{j-1} \leq & x_j & \leq 1 - \epsilon\, x_{j-1}, \forall j = 2, \ldots n
\end{array}
$$

where $0 < \epsilon < 1/2$.

Real-life problem
Average-Case
Sc.

$O(m^p n^b)$

$p = 2, 3$



Worst-case
scenario

LPP simplex
Method
$O(n!)$

40's

2nd World War

70

Figure 10.1: Three Dimensional Klee-Minty's Example with $\epsilon = 1/5$

Interior-pt method $O(m^p n^8)$ proven

A B

M^{-1}

As you can see in Figure 10.1, the feasible region is combinatorially a n dimensional hypercube with 2^n extreme points and the simplex method can visit all 2^n extreme points. This example is given by a system of $2n$ inequalities in n variables and in particular the binary encoding length of the problem is $O(n)$. Thus, any polynomial algorithm uses at most a polynomial number of arithmetic operations in terms of n. The orientation by the objective function has a recursive structure. Namely, the orientation restricted to the (bottom) facet associated with $\epsilon x_{n-1} \leq x_n$ is reversed on the opposite facet associated with $x_n \leq 1 - \epsilon x_{n-1}$. Each of these orientations are isomorphic to the orientation of an $(n-1)$ dimensional Klee-Minty's example. A typical exponential behavior of the simplex method goes though all vertices of the bottom facet first, moves up to the top facet, and then goes through all vertices of the facet to reach the optimal vertex.

The first polynomial algorithm for linear programming was proposed by Khachiyan [27] in 1979. The algorithm is known as the *ellipsoid method.* Although it was considered a breakthrough, the algorithm is not known to be practical because it is difficult to implement it with a reasonable arithmetic precision. The first practical polynomial algorithm (class), known as interior-point methods, was invented by Karmarker [26] in 1984, and many variations have been proposed afterwards including the primal-dual interior-point methods.

The purpose of this chapter is to give a brief description of the primal-dual interior-point methods. Our main goals are to give a basic theoretical framework and to explain key algorithmic components which were developed in the field of nonlinear continuous optimization.

10.1 Notions

For a function $f : \mathbb{R}^n \to \mathbb{R}$, the *gradient* ∇f of f is defined as the vector of its partial derivatives

$$\nabla f(x) := \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \tag{10.2}$$

Of course, for the gradient to be defined, the function must be differentiable.

The matrix of second partial derivatives of f is the *Hessian matrix* denoted by $H(x)$ or $\nabla^2 f(x)$ defined by

$$\nabla^2 f(x) = H(x) := \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} \cdots \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \ddots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} \cdots \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{bmatrix} \tag{10.3}$$

In this chapter, for simplicity, we assume that any function f is twice continuously differentiable, i.e. it has a second derivative that is continuous. This class of function is denoted by C^2.