3.1 IP's Solvable directly thro' LP Relaxation

This includes IPs in which the coeff. matrix is TU. So when RHS vector is integral, all BFSs of LP relaxation are also integral. So, IP can be solved directly by solving for an extreme pt. sol. of LP relaxation.

This includes assignment problem, transportation problems, and min cost network flow problems.

As an example we now discuss the Hungarian method for the assignment problem, a primal-dual algorithm.

Primal-Dual Algorithms

Here we discuss special min cost flow problems on bipartite networks, the **assignment and transportation problems**. Algorithms based on an approach called the **Primal-Dual approach** are discussed.

Strategy of P-D methods to solve an LP

1. Methods need an **initial dual feasible solution** to start. *Hence methods suitable to solve problems for which a dual feasible sol. can be found easily (these include assignment, transportation, matching & other network problems).*
2. Always maintains:

- a **(primal vector, dual feasible sol.) pair** that together satisfies **C. S. opt. conds.**
- dual feasibility and C.S. opt. conds.

Primal vector is primal infeasible till end. So when primal feasibility attained, the pair becomes opt. & method terminates.

3. Two main steps carried out in this particular order:

Primal vector change step: Keeping dual sol. fixed, obtain primal vector closest to primal feasibility among all primal vectors satisfying C.S. conds. together with present dual feasible sol.

So, a measure of **primal infeasibility** is minimized during this step.

Dual solution change step: Do this only when primal vector cannot be changed as above. Changes to a new dual feasible solution satisfying:

a) C.S. conds with present primal vector.

b) Using it, it is possible to resume primal vector change & make some progress in it.

The primal infeasibility measure is decreasing $\downarrow$ during algo. Method can terminate two ways:

1. With an optimum primal, dual pair of sols.
2. With primal infeasibility conclusion.

Assignment Problem

Input: $n =$ order of problem; $c = (c_{ij})$, square cost matrix of order n.

Output needed: Solution $x = (x_{ij})$ that solves:

$$\begin{aligned} \min \quad z(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & & & \\ \text{s. to} \quad \sum_{j=1}^{n} x_{ij} &= 1 & i = 1, \ldots, n \\ \sum_{i=1}^{n} x_{ij} &= 1 & j = 1, \ldots, n \\ x_{ij} &\in \{0,1\} & \forall i, j \end{aligned}$$

Put up an $n \times n$ 2-D array, and associate variable x_{ij} with cell (i, j). Cell (i, j) said to have an **allocation** in solution x iff $x_{ij} = 1$. Exactly one allocation in each row & col of array in any sol. So, sol. x is a **permutation matrix**. Also called **assignment** or **integer doubly stochastic matrix**.

Problem is **Minimum cost perfect matching problem** in an $n \times n$ bipartite network. Each assignment is a perfect matching in this network.

EXAMPLE: $n = 4$, one assignment is $\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Forbidden cells: Cells where allocations forbidden. So, x_{ij} must $= 0$ in all for bidden cells (i, j). Problem called:

Assignment problem on complete network if no forbidden cells

Assignment problem on dense network if number of forbidden cells small

Assignment problem on sparse network if number of forbidden cells large.

We assume:

$$c_{ij} = \begin{cases} \infty & \text{if } (i,j) \text{ forbidden} \\ \text{cost of allocating in } (i,j) & \text{otherwise} \end{cases}$$

Partial assignment: A $0-1$ matrix with at most one allocation in any row or col. Example: $\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Each partial assignment is a **matching** that is not perfect in the network.

Dual problem: max $w(u, v) = \Sigma_{i=1}^{n} u_i + \Sigma_{j=1}^{n} v_j$ s. to $\bar{c}_{ij} = c_{ij} - u_i - v_j \geq 0 \quad \forall i, j$.

$\bar{c}_{ij}$ called **reduced** or **relative cost coeff.** of cell (i, j) WRT dual vector (u, v). (u, v) **dual feasible** iff $\bar{c}_{ij} \geq 0 \forall i, j$.

WRT dual sol. (u, v), cell (i, j) called **admissible** or **equality cell** if $\bar{c}_{ij} = 0$; **inadmissible cell** otherwise.

The set of admissible cells constitutes **admissible** or **equality subnetwork**.

Strategy of the Hungarian method: Maintains $(x, (u, v))$ partial assignment, dual sol. pair always satisfying:

- **Dual feasibility** $\bar{c}_{ij} \geq 0 \; \forall i, j$.
- **C.S. conds.** $\bar{c}_{ij} x_{ij} = 0 \; \forall i, j$, i.e., allocations occur only in admissible cells.

Primal vector change step maximizes $\Sigma\Sigma\, x_{ij}$ subject to (1) C. S. conds., and (2) at most one allocation in any row or col. This problem is the maximum value flow (from row nodes to col. nodes) problem in admissible subnetwork, can be solved by the labeling algo. It finds maximum partial assignment satisfying C. S. conds. with current dual feasible sol.

When max value of $\Sigma\Sigma\, x_{ij}$ becomes n, primal feasibility attained, and method terminates with final x as an opt. assignment.

EXAMPLES:

$j =$	1	2	3	u_i
$i = 1$	$c_{11} = 10$	9	10	3
2	22	5	12	-1
3	9	20	15	5
v_j	4	6	7	

$j =$	1	2	3	u_i
$i = 1$	$c_{11} = 3$	9	10	10
2	4	5	19	6
3	6	8	8	3
v_j	5	8	4	

$j =$	1	2	3	u_i
$i = 1$	$c_{11} = 1$	6	8	1
2	12	33	45	12
3	14	84	93	14
v_j	0	5	7	

Concept of Lower Bounding the Objective value

THEOREM: If (u, v) dual feasible, dual obj. func. value $w(u, v) = \sum u_i + \sum v_j$ called **Total reduction** is a **lower bound** for the cost $z(x)$ of any assignment x.

Hungarian Method

Step 1: Finding initial dual feasible sol. by row & col. reduction:

$u_i =$ min in row i of original cost matrix, $\forall i$.

Row reduction: For each i subtract u_i from each entry of row i of c.

$v_j =$ min in col. j after row reduction.

Col. reduction: For each j subtract v_j from each entry in col. j of row reduced matrix, leading to 1st reduced cost matrix.

Step 2: Finding initial allocations: Take any row or col. without an allocation but with an uncrossed admissible cell. Put an allocation in that cell, & cross out all other admissible cells in its row & col. Repeat until all admissible cells crossed out.

If allocations in all rows, they define an optimum assignment, terminate. Otherwise continue.

Step 3: Labeling routine to check max flow:

3.1: Initial labeling: For each i, if no allocation in row i, label it $(s, +)$. Put all labeled rows in *list*.

3.2: Delete a labeled row or col from list to scan: Use FIFO. If list $= \emptyset$ **nonbreakthrough**, go to Step 5.

Scanning labeled row i: Label all unlabeled cols. j with an admissible cell in row i with (Row $i, +$); & put these labeled cols. in list. Return to Step 3.2.

Scanning labeled col. j: Look for an allocation in this col. j. If none, col. j has **breakthrough**, go to Step 4.

If col. j has an allocation, suppose it is in row i. If this row i unlabeled, label it with (Col. $j, -$), and put this labeled row in list. Return to Step 3.2.

Step 4: Allocation change: Use labels to trace the predecessor path of col. in which breakthro' occured to a row with label $(s, +)$. It will be an **alternating path** of unallocated, allocated cells. Exchange allocated, unallocated cells on this path. Chop down trees (i.e., erase labels on all rows, cols.).

If all rows allocated, they define an opt. assignment, terminate. Otherwise go to Step 3.

Step 5: Dual sol. change: At this time verify that all cells in the Labeled row-Unlabeled col. block are inadmissible. The change is carried out to create at least one new admissible in this block so more cols. can be labeled.

$\delta = \min \bar{c}_{ij}$ in Labeled row-Unlabeled col block.

If $\delta = \infty$, no feasible assignment (i.e., one without an allocation in a forbidden cell), terminate.

If δ finite, define new dual sol. to be:

$$\text{new } u_i = \begin{cases} \text{present } u_i + \delta & \text{if row } i \text{ labeled} \\ \text{present } u_i & \text{if row } i \text{ unlabeled} \end{cases}$$

$$\text{new } v_j = \begin{cases} \text{present } v_j - \delta & \text{if col } j \text{ labeled} \\ \text{present } v_j & \text{if col } j \text{ unlabeled} \end{cases}$$

Compute new reduced costs, identify new admissible cells, make list = set of all labeled rows, and go to Step 3.2.

Array 5.2

$j =$	1	2	3	4	5	6
$i = 1$		0 □				
2			0 □			
3	0			0 □		
4			0	0		
5			0	0		
6	0 □				0	0

$$c = \begin{pmatrix} 15 & 22 & 13 & 4 \\ 12 & 21 & 15 & 7 \\ 16 & 20 & 22 & 6 \\ 6 & 11 & 8 & 5 \end{pmatrix}$$

THEOREM: δ is > 0 in every dual sol. change step.

THEOREM: If $\delta = \infty$ in Step 5, present matching is a maximum cardinality matching in network & there is no feasible assignment.

THEOREM: **Konig-Egervary Theorem** Whenever non-breakthrough occurs, we have:

no. of allocations = no. of unlabeled rows + no. of labeled cols.

If lines are drawn through each unlabeled row & each labeled col. these lines cover all admissible cells. This set of lines is a smallest cardinality set of lines thro' rows & cols. that cover all admissible cells.

THEOREM: In Step 5, dual feasibility and C. S. property with current primal vector are always maintained.

THEOREM: After Step 5 is carried out & labeling resumed, at least one new col. can be labeled.

THEOREM: In this algo. the no. of consecutive occurences of nonbreakthro' before a breakthro' occurs is at most n.

THEOREM: If implemented directly, the complexity of the algo. is $O(n^4)$.

Data structures to reduce complexity to $O(n^3)$

Need to avoid computing all n^2 relative cost coeffs. after every dual sol. change.

Amazingly, this is possible by introducing an index for each unlabeled col. !

So, good **Data structures** very imp. in implementing opt. algos.

Divide the algo. into **stages**. A new stage begins after every allocation change step. By above results, dual sol. change step can occur at most n times in a stage. But the efficient implementation computes the reduced cost coeffs. in all cells only once in a stage, at the beginning.

At the 1st nonbreakthrough in a stage, it defines an **index** $[t_j, p_j]$ for each unlabeled col. where:

p_j = min current reduced cost coeff. in this col. among labeled rows

t_j = no. of a labeled row in which above min occurs (in case of tie, select any one).

Index only defined for unlabeled cols., and it is erased when the col gets labeled. Indices of all unlabeled cols. updated whenever a new row is labeled, or dual sol changed, so p_j in unlabeled cols. is always true to its definition.

$O(n^3)$ Version of Hungarian Method

Step 1: Finding initial dual feasible sol. by row & col. reduction:

u_i = min in row i of original cost matrix, $\forall i$.

Row reduction: For each i subtract u_i from each entry of row i of c.

v_j = min in col. j after row reduction.

Col. reduction: For each j subtract v_j from each entry in col. j of row reduced matrix, leading to 1st reduced cost matrix.

Step 2: Finding initial allocations: Take any row or col. without an allocation but with an uncrossed admissible cell. Put an allocation in that cell, & cross out all other admissible cells in its row & col. Repeat until all admissible cells crossed out.

If allocations in all rows, they define an optimum assignment, terminate. Otherwise set Stage number $k = 1$.

Step 3: Begin Stage k. Compute and store $\forall i, j$ the relative cost coeff. $\bar{c}_{ij} = c_{ij} - u_i - v_j$ where $(u_i), (v_j)$ is the present dual solution. Identify present admisssible cells as those with $\bar{c}_{ij} = 0$.

Step 4: Labeling routine to check max flow:

4.1: Initial labeling: For each i, if no allocation in row i, label it $(s, +)$. Put all labeled rows in *list*.

4.2: Delete a labeled row or col from list to scan: Use FIFO. If list $= \emptyset$ **nonbreakthrough**, go to Step 4.3 if this is the first nonbreakthrough in this Stage k, otherwise go to Step 6.

Scanning labeled row i: Label all unlabeled cols. j with an admissible cell in row i with (Row $i, +$), and erase the indices on these cols. if they have any; & put these labeled cols. in list. Return to Step 4.2.

Scanning labeled col. j: Look for an allocation in this col. j. If none, col. j has **breakthrough**, go to Step 5.

If col. j has an allocation, suppose it is in row i. If this row i unlabeled, label it with (Col. $j, -$), and put this labeled row in list and go to Step 4.4.

4.3: Defining indices for unlabeled cols.: For each unlabeled col. q at this time, define its index to be $[t_q, p_q]$ where:

$p_q = \min\ \{\bar{c}_{iq} : i \text{ a labeled row at this time}\}$, $t_q =$ an i that ties for the min in the definition of p_q selected arbitrarily among those tied. Go to Step 6.

4.4: Updating indices on unlabeled cols. when a new row i is labeled: For each unlabeled col h at this time with index $[t_h, p_h]$,if:

- the stored reduced cost coeff. $\bar{c}_{ih} = 0$, label col h with (Row $i, +$), erase the index on col h and put col h in the list. Go to Step 4.2.

- $p_h > \bar{c}_{ih} > 0$, change index on col h to $[i, \bar{c}_{ih}]$ and go to Step 4.2.

- $p_h \leq \bar{c}_{ih} > 0$, leave index on col. h unchanged, go to Step 4.2.

Step 5: Allocation change: Use labels to trace the predecessor path of col. in which breakthro' occured to a row with label $(s, +)$. It will be an **alternating path** of unallocated, allocated cells. Exchange allocated, unallocated cells on this path.

Chop down trees (i.e., erase labels on all rows, cols., and erase any indices on cols.).

If all rows allocated, they define an opt. assignment, terminate. Otherwise increment k by 1 and go to Step 3.

Step 6: Dual sol. change:

$\delta = \min \{p_j :$ over unlabeled cols. j at this time$\}$.

If $\delta = \infty$, no feasible assignment (i.e., one without an allocation in a forbidden cell), terminate.

If δ finite, define new dual sol. to be:

$$\text{new } u_i = \begin{cases} \text{present } u_i + \delta & \text{if row } i \text{ labeled} \\ \text{present } u_i & \text{if row } i \text{ unlabeled} \end{cases}$$

$$\text{new } v_j = \begin{cases} \text{present } v_j - \delta & \text{if col } j \text{ labeled} \\ \text{present } v_j & \text{if col } j \text{ unlabeled} \end{cases}$$

Subtract δ from the p_j index of each unlabeled col. j. For each unlabeled col. j in which p_j became 0 as a result of this subtraction, record (t_j, j) as a new admissible cell, label col. j with (Row $t_j, +$), erase the index on it, and put col. j in the list.

Put all labeled rows in the list. Go to Step 4.2.

THEOREM: In this version, the p_j index on an unlabeled col. j is always the min reduced cost coeff WRT present dual sol. in cells among labeled rows in this col. The value of δ found out in every dual sol. change step is always > 0 and correct as defined in original algo. The complexity of this version is $O(n^3)$.