Note Title 30-03-2021

Integer linear programs

An LP in which some or all variables are restricted to take integer values only.

eg. max $3x + 4y$

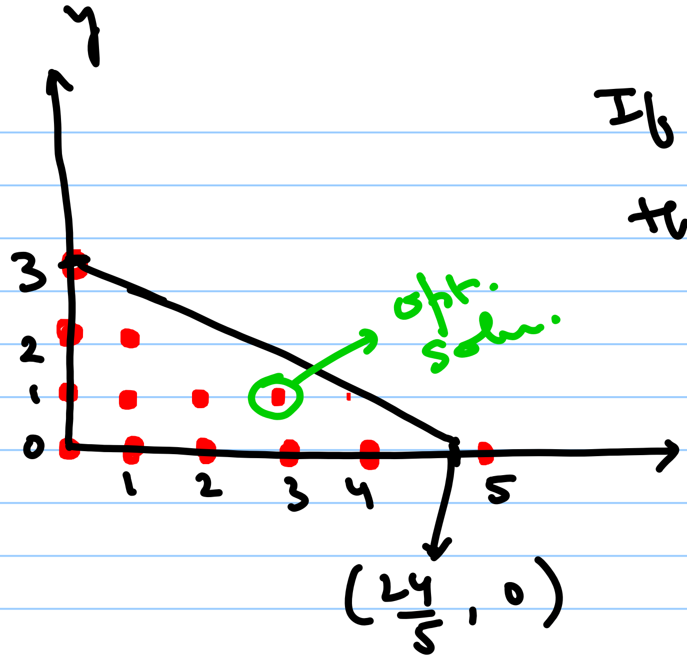
subject to $5x + 8y \leq 24$

$x, y \geq 0$, x, y integers.

Pure integer programs.

Mixed integer programs

- what is the optimal soln? what are the challenges?



If we ignore the integer restriction then the optimal soln of the relaxed LPP is $x^* = \frac{24}{5}$, $y^* = 0$; $z^* = \frac{72}{5}$

If we round it off to upper side $\hat{x} = 5$, $\hat{y} = 0$, then $(5, 0)$ is not feasible for problem.

$\therefore$ round off to lower side to get $(4, 0)$ with $z = 12$

But $z = 12$ at $(0, 3)$ also. However, optimal soln of integer program is $(3, 1)$ with optimal value $= 13$.

eg : max $z = 21x + 11y$

s. to $7x + 4y \le 13$

$x, y \geq 0$ & integers

The relaxed associated LPP has optimal soln $\left(\frac{13}{7}, 0\right)$

Round off gives $(2, 0) \rightarrow$ not feasible

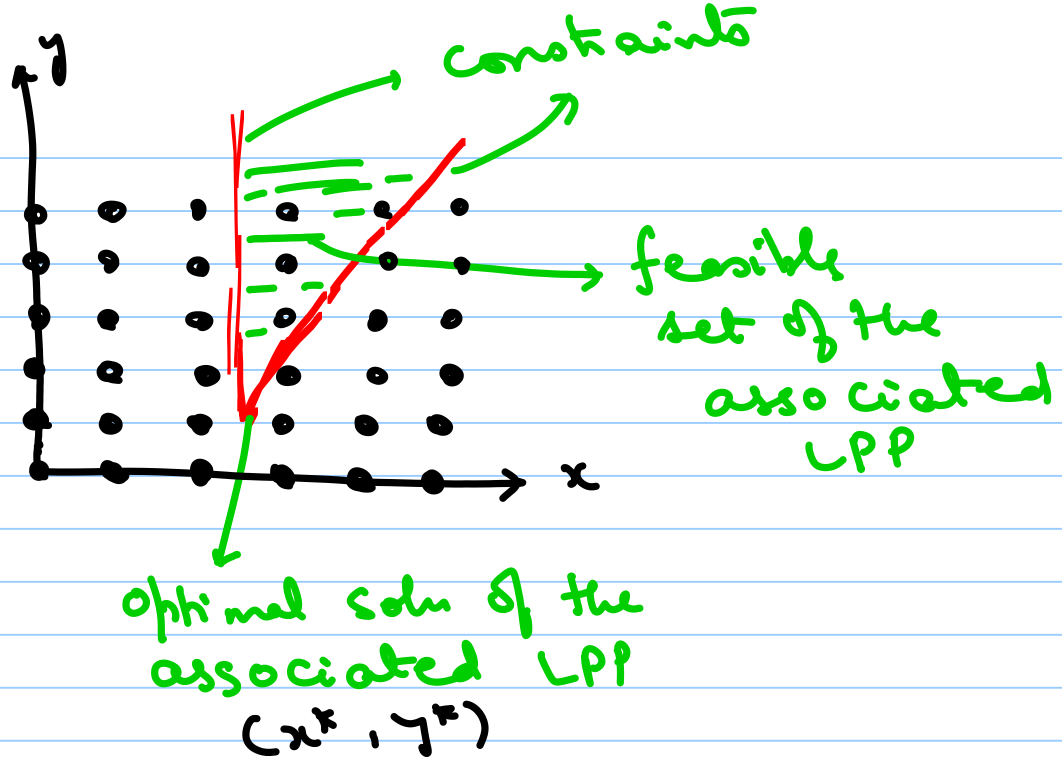
Round off to lower side $(1, 0)$ feasible with $z = 21$

While the optimal soln of the problem is $(0, 3)$ with

$z^* = 33$ too far away from 21.

Note : The feasible set of ILP is discrete so non-convex, nor connected in the sense of paths.

eg



Note that rounding off of (x^*, y^*) to their nearest 4 neighbours always result in pts which are not even feasible for the given ILP

$\therefore$ Rounding off the optimal soln of the associated (relaxed) LPP do not work.

LP with binary variables

$$\max \; z = c^T x$$

s.to $\quad Ax \leq b$

$\quad x_i \in \{0,1\}, \; \forall i = 1, 2, \ldots, n$

$$\Downarrow$$

$\quad 0 \leq x_i \leq 1, \; x_i$ integer

$\therefore$ Binary variables LPP is equivalent to ILP.

eg. Investment Optimization Problem

Assets	1	2	3	4	5
Expected return	12	11	13	8	15
Cash	500	400	700	300	600

Either one invest in an asset or do not invest

Problem: $\max \; 12x_1 + 11x_2 + 13x_3 + 8x_4 + 15x_5$

subject to

$$5x_1 + 4x_2 + 7x_3 + 3x_4 + 6x_5 \le 14$$

$$x_i \in \{0,1\} \;\; \forall i.$$

where the investment budget is 1400/-

Logical constraints:

- Only one of the five investments are acceptable

$$x_1 + x_2 + x_3 + x_4 + x_5 \le 1$$

- We can make investment in atmost 2 assets

$x_1 + x_2 + x_3 + x_4 + x_5 \leq 2$

- If investment in asset 1 is made then investment in asset 3 can not be made

$$x_1 + x_3 \leq 1$$

- If investment in 1 is made then investment in 4 must also be made

$$x_4 \geq x_1 \quad \text{or} \quad x_1 - x_4 \leq 0$$

- Either investment in 2 or 5 are made by not both

$$x_2 + x_5 = 1$$

- One can make investment in exactly 3 out of 5

$x_1 + x_2 + x_3 + x_4 + x_5 = 3.$

eg. Knapsack Problem

You have n objects to put in a shoulder bag

Each item j has weight w_j and value c_j

The maximum capicity of bag to hold is say b.

$$(P) \quad \max \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \le b$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n.$$

eg Warehouse location problem

The aim is to find that out of n warehouses handling the demand of m customers, which warehouse to operate and how much is to ship from a warehouse to any customer. Define:

$$y_i = \begin{cases} 1 & \text{if warehouse } i \text{ operates} \\ 0 & \text{if warehouse } i \text{ do not operate} \end{cases}$$

x_{ij} : Amount to be delivered from i^{th} warehouse to the j^{th} customer.

Let L_i be the fixed cost of opening the i^{th} warehouse and c_{ij} is the cost of transporting one unit of item from i^{th} warehouse to j^{th} customer

The demand d_j of the j^{th} customer must be fulfilled

$$\therefore \ (P) \quad \min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} + \sum_{i=1}^{n} L_i y_i$$

$$\text{s.to} \quad \sum_{i=1}^{n} x_{ij} = d_j \ , \quad x_{ij} \geq 0 \ , \ \text{and}$$

$$\left.\begin{array}{l} \text{If } y_j = 1 \text{ then } x_{ij} \leq d_j \\ \text{If } y_j = 0 \text{ then } x_{ij} = 0 \end{array}\right\} \text{No transportation from non-operational warehouses}$$

$\therefore$ we get

$$\min\ z = \sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{n} L_i y_i$$

subject to $\displaystyle\sum_{i=1}^{n} x_{ij} = d_j \quad , j = 1, 2, \ldots, m$

$$0 \le y_i \le 1 \quad , \quad i = 1, 2, \ldots, n$$

$$x_{ij} \le d_j y_i \quad , \forall\ i = 1, 2, \ldots, n$$
$$j = 1, 2, \ldots, m$$

y_i are integers $\forall\ i$

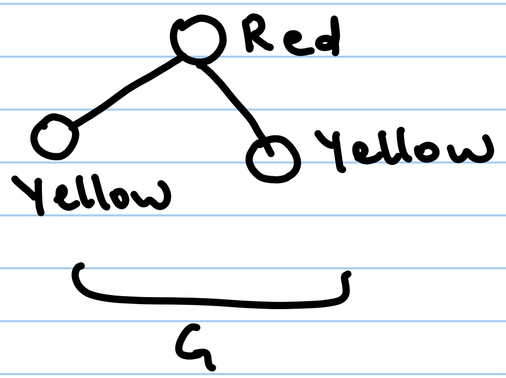
$x_{ij} \geq 0, \ \forall\ i, \ \forall\ j$

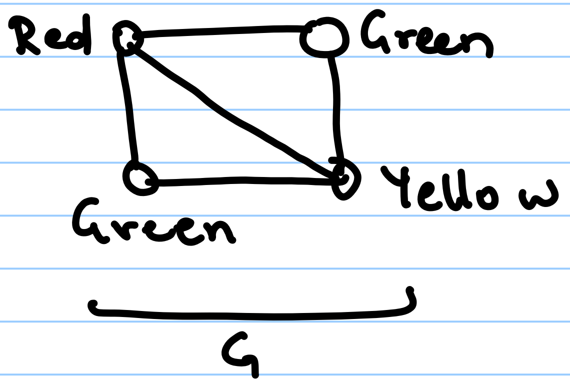
Graph Coloring

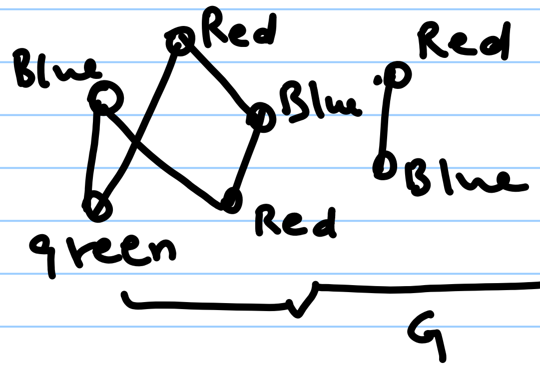
vertices

Graph $G = (V, E) \rightarrow$ edges

Assign colors to the nodes so that every node must be colored, no two adjacent nodes get the same color and accomplish this task using as few different colors as possible.







Define

x_{ij} : variable is true if i^{th} node gets j^{th} color

w_j : variable is true if atleast one node gets j^{th} color

$x_{ij} \in \{0,1\}$, $w_j \in \{0,1\}$, $\forall\, i \in V$, $j \in C$ ↪ color set

minimize $\sum_{j \in C} w_j$ ⟶ as few color as possible

subject to $\sum_{j \in C} x_{ij} = 1, \ \forall\, i \in V$ ⟶ each node get exactly one color.

no two adjacent vertices get same color ⟵ $x_{uj} + x_{vj} \leq 1, \ \forall\, (u,v) \in E, \ \forall\, j \in C$

if any node get color j then $w_j = 1$ ⟵ $x_{ij} \leq w_j, \ \forall\, i \in V, \ j \in C$

There are many problems in graphs, networks, combinatorics, AI & ML that can be formulated as linear integer programs or binary problems.

Either-or constraints

$$2x + 3y \geq 14 \quad \text{or} \quad 5x - 7y \leq 3$$

We introduce a binary variable δ such that

$$\left.\begin{array}{l} \text{if } \delta = 1 \text{ then } 2x + 3y \geq 14 \\ \text{if } \delta = 0 \text{ then } 5x - 7y \leq 3 \end{array}\right\} \Rightarrow \begin{array}{l} 2x + 3y \geq 14 - M(1-\delta) \\ 5x - 7y \leq 3 + M\delta \\ \delta \in \{0,1\} \end{array}$$

where $M >> 0$ is prefixed.

If - then constraints

If $f(x_1, x_2, \ldots, x_n) > b_1$ then $g(x_1, x_2, \ldots, x_n) \le b_2$

equivalent to (If p then q $\equiv$ $p \to q$)

$$f(x_1, x_2, \ldots, x_n) \le b_1 \text{ or } g(x_1, \ldots, x_n) \le b_2$$

$$\Longleftrightarrow \quad f(x_1, x_2, \ldots, x_n) \le b_1 + M\delta$$

$$g(x_1, x_2, \ldots, x_n) \le b_2 + M(1-\delta)$$

$$\delta \in \{0, 1\}$$

where $M \gg 0$ is prefined.

eg. Atleast one of the 3 constraints to hold:

$$x + 4y + 2z \geq 9 \quad \text{or} \quad 3x - 5y \leq 15 \quad \text{or} \quad 2y + z \geq 8$$

Define binary variables $\delta_1, \delta_2, \delta_3$ s.t.

$$\left.\begin{array}{lll} \text{if } \delta_1 = 1 & \text{then} & x + 4y + 2z \geq 9 \\ \text{if } \delta_2 = 1 & \text{then} & 3x - 5y \leq 15 \\ \text{if } \delta_3 = 1 & \text{then} & 2y + z \geq 8 \end{array}\right\} :\equiv \begin{array}{l} x + 4y + 2z \geq 9 - M(1-\delta_1) \\ 3x - 5y \leq 15 + M(1-\delta_2) \\ 2y + z \geq 8 - M(1-\delta_3) \\ \delta_1 + \delta_2 + \delta_3 \geq 1 \\ \delta_i \in \{0,1\}, \; i = 1, 2, 3 \end{array}$$

where $M >> 0$ is predefined.

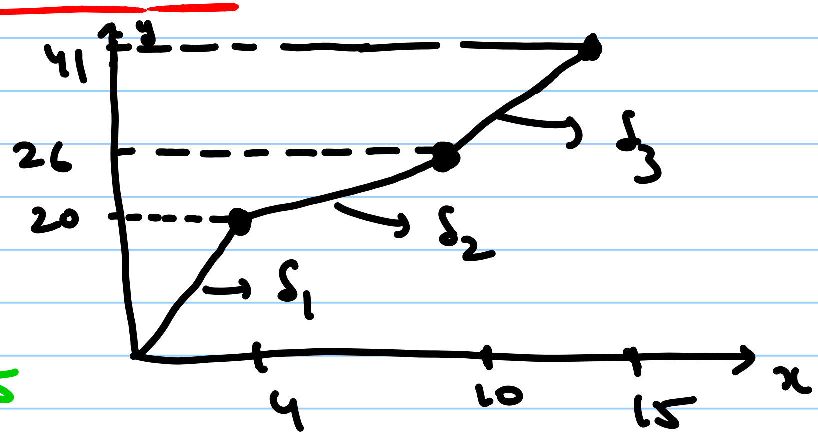
Piecewise linear function representation

$5x_1$

$$y = \begin{cases} 5x & 0 \le x \le 4 \\ x+16 & 4 \le x \le 10 \\ 3x-4 & 10 \le x \le 15 \end{cases}$$

$x_2 + 20$
$0 \le x_2 \le 6$

$3x_3 + 26, \quad 0 \le x_3 \le 5$



Take $\quad x = x_1 + x_2 + x_3$

$$0 \le x_1 \le 4, \quad 0 \le x_2 \le 6, \quad 0 \le x_3 \le 5$$

If $x_2 > 0$ then $x_1 = 4$

If $x_3 > 0$ then $x_2 = 6$

Define $\delta_1 = \begin{cases} 1 & \text{if } x_1 \text{ is at its upper bound} \\ 0 & \text{otherwise} \end{cases}$

$\delta_2 = \begin{cases} 1 & \text{if } x_2 \text{ is at its upper bound} \\ 0 & \text{otherwise} \end{cases}$

$\therefore$ If constraints are

$$\left.\begin{aligned} 4\delta_1 &\le x_1 \le 4 \\ 6\delta_2 &\le x_2 \le 6\delta_1 \\ 0 &\le x_3 \le 5\delta_2 \end{aligned}\right\} \begin{aligned} &\delta_1 = \delta_2 = 0 \Rightarrow 0 \le x \le 4 \\ &\delta_1 = 1,\ \delta_2 = 0 \Rightarrow 4 \le x \le 10 \\ &\delta_1 = 1,\ \delta_2 = 1 \Rightarrow 10 \le x \le 15 \end{aligned}$$

Note $\delta_2 \le \delta_1$

$\therefore \underbrace{\delta_2 = 1,\ \delta_1 = 0}_{\text{not possible}}$.

$$y = 5x_1 + x_2 + 3x_3$$

$$
\left.\begin{aligned}
& y = 5x_1 + x_2 + 3x_3 \\
& 0 \le x_1 \le 4 \\
& 0 \le x_2 \le 6 \\
& 0 \le x_3 \le 5 \\
& 4\delta_1 \le x_1 \le 4 \\
& 6\delta_2 \le x_2 \le 6\delta_1 \\
& 0 \le x_3 \le \delta_2 \\
& \delta_1, \delta_2 \in \{0,1\}.
\end{aligned}\right\} \text{ final formulation.}
$$

Such formulations are useful to model problems in supply chain, pricing | discounting, inventory management, using binary var.

eg. Suppose we have x is one of the element in the set $\{4, 8, 12\}$. To formulate this constraint, we write

$$x = 4\delta_1 + 8\delta_2 + 12\delta_3$$

$$\delta_1 + \delta_2 + \delta_3 = 1$$

$$\delta_i \in \{0, 1\}, \ i = 1, 2, 3.$$

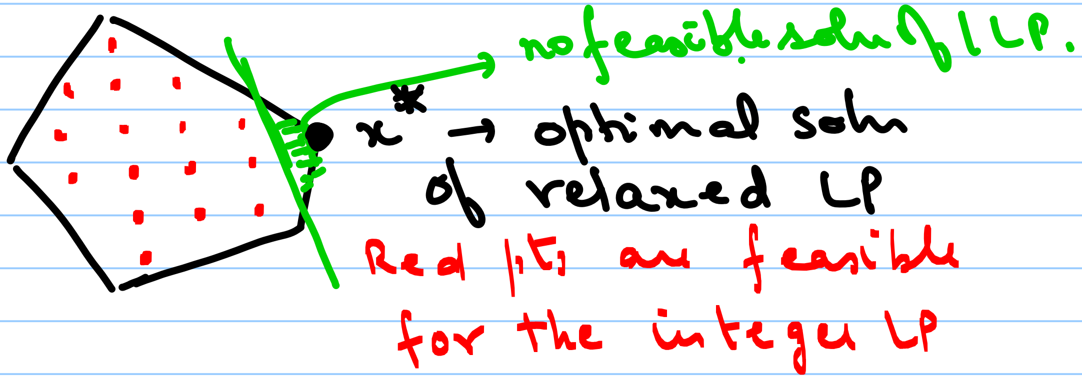
→ Let $x \in \{0, 4, 8, 12\}$, then write it as

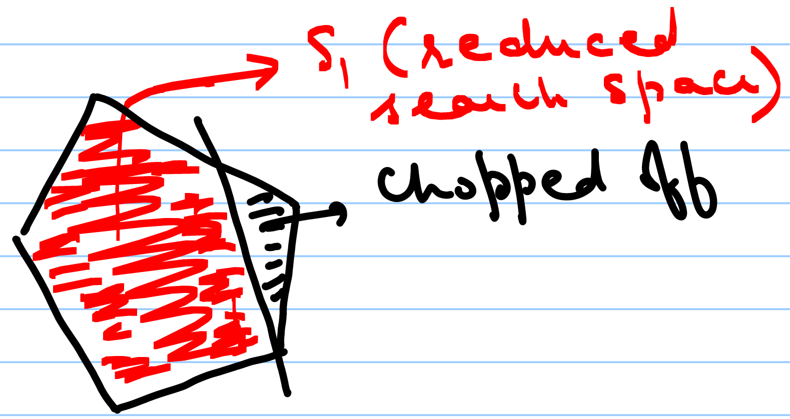
$$x = 4\delta_1 + 8\delta_2 + 12\delta_3$$

$$\delta_1 + \delta_2 + \delta_3 \leq 1, \quad \delta_1, \delta_2, \delta_3 \in \{0, 1\}.$$

Cutting Plane Technique

Consider the associated relaxed LPP, and let its feasible set be S. The idea in cutting plane is to find a hyperplane which chop off that region of S which contains no feasible solution of the integer LP.





Gomory cutting Plane technique

Step 1 : Solve the relaxed LPP to get optimal soln x^*

Step 2 : If x^* is integral, stop

Step 3 : Find a valid linear inequality (or cut) using x^* such that the cut removes x^* and does not remove any feasible pt of the original ILP. Insert this cut as a constraint in the relaxed LPP to get a new LPP, and go to step 1 to resolve it to get fresh x^*.

Dual simplex method can be used to resolve the new LPP.

Let us consider a pure integer LP,

$$\max \ z = c^T x$$

$$\text{subject to} \quad Ax \le b$$
$$x \text{ integers}$$

In system, A and b are inserted as rationals (even if some of the entries are irrational), and all rationals can be converted into integers by taking LCM multipliers.

$\therefore$ without loss of generality, we take all entries in A and b as integrals, and hence all slack, surplus, artificial

variables inserted in the given problem will be integers.
Let x^* be the soln of the associated relaxed LPP (with no integer restriction). Let its r^{th} component x_r^* be non integer. In the optimal table, let J be index set of optimal basic variables ($\rightarrow I$ matrix in final table).
Let S_I be the feasible region of the original ILP.
For any $x \in S_I$, from the r^{th} constraint we will have

$$x_r + \sum_{i \notin J} y_{ri} x_i = x_r^* \quad \text{———} \quad (1)$$

$$
\begin{array}{llllllll}
v_B & x_B & y_1 & y_2 & \cdots & y_r & \cdots & y_p \\
 & \vdots & & & & & & \\
x_{B_r} & x_r^* & y_{r1} & y_{r2} & \cdots & 1 & \cdots & y_{rp} \rightarrow r^{th} \text{ row}
\end{array}
$$

and $y_{ir} = 0 \ \forall\, i \neq r$, $y_{rr} = 1$, $i \in J$

① $\Rightarrow$

$$x_r + \sum_{i \notin J} \lfloor y_{ri} \rfloor x_i + \sum_{i \notin J} f_{ri}\, x_i = \lfloor x_r^* \rfloor + f_r \quad \text{—} ②$$

$0 \leq f_{ri} \leq 1$, $0 < f_r < 1$

Since, $x \geq 0$, ② $\Rightarrow$

$$x_r + \sum_{i \notin J} \lfloor y_{ri} \rfloor x_i \leq \lfloor x_r^* \rfloor + f_r \quad \text{—} ③$$

Now, for $x \in S_I$, LHS of ③ is integer, and $\lfloor x_r^* \rfloor$ is also integer, and $f_r \in (0,1)$, so

$$x_r + \sum_{i \notin J} \lfloor y_{ri} \rfloor x_i \leq \lfloor x_r^* \rfloor$$

$$\Rightarrow \; -x_r - \sum_{i \notin J} \lfloor y_{ri} \rfloor x_i \geq -\lfloor x_r^* \rfloor \quad \text{— (4)}$$

(1) + (4)

$$\sum_{i \notin J} \left(y_{ri} - \lfloor y_{ri} \rfloor \right) x_i \geq x_r^* - \lfloor x_r^* \rfloor$$

$$\Rightarrow \; \sum_{i \notin J} f_{ri} x_i \geq f_r \; , \quad \forall \; x \in S_I \longrightarrow \text{cut equation (5)}$$

$$\Rightarrow \; -\sum_{i \notin J} f_{ri} x_i + s = -f_r \, , \quad s \geq 0 \;\; (\text{slack}) \quad \text{— (6)}$$

Note: i) x^* does not satisfy (5), as $x_i^* = 0 \ \forall i \notin J$

ii) s is integer $\forall x \in S_I$ because from (2), we have

$$x_r = \Big(f_r - \sum_{i \notin J} f_{ri} x_i\Big) + \Big(\lfloor x_r^* \rfloor - \sum_{i \notin J} \lfloor y_{ri} \rfloor x_i\Big)$$

$$\Rightarrow \underbrace{x_r}_{\text{integer as } x \in S_I} = -s + \underbrace{\Big(\lfloor x_r^* \rfloor - \sum_{i \notin J} \lfloor y_{ri} \rfloor x_i\Big)}_{\text{integer as } x \in S_I} \Rightarrow s \text{ must be integer.}$$

Insert the cut constraint (6) in the relaxed LPP with cost coefficient zero with slack s in the objective fn. & solve the problem. You can use dual simplex method rather than solving the new LPP afresh from scratch.

eg. max $z = 2x_1 + x_2$

s. to $x_1 + x_2 + x_3 = 5$

$-x_1 + x_2 + x_4 = 0$

$6x_1 + 2x_2 + x_5 = 21$

x_1, x_2, x_3, x_4, x_5 integers and ≥ 0.

Solve the relaxed LPP (drop integer condition) and the optimal table is as follows

C_B	V_B	x_B	y_1	y_2	y_3	y_4	y_5
2	x_1	$11/4$	1	0	$-1/2$	0	$1/4$
1	x_2	$9/4$	0	1	$3/2$	0	$-1/4$
0	x_4	$1/2$	0	0	-2	1	$1/2$
		$z_j - c_j \rightarrow$	0	0	$1/2$	0	$1/4$

Each variable can act as source of cut

$J = \{1, 2, 4\}$, $i \notin J$ means $i \in \{3, 5\}$. Use $\sum_{i \notin J} f_{ri} x_i \geq f_r$

x_1 row $\rightarrow$ $\frac{x_3}{2} + \frac{x_5}{4} \geq \frac{3}{4}$

x_4 row $\rightarrow$ $\frac{x_5}{2} \geq \frac{1}{2}$

x_2 row $\rightarrow$ $\frac{x_3}{2} + \frac{3x_5}{4} \geq \frac{1}{4}$

Suppose, we take $r=4$ and x_4 for introducing a cut.

$$-\frac{x_5}{2} + S = -\frac{1}{2}, \quad S \geq 0$$

C_B	V_B	x_B	y_1	y_2	y_3	y_4	y_5	y_s
	x_1	$11/4$	1	0	$-1/2$	0	$1/4$	0
	x_2	$9/4$	0	1	$3/2$	0	$-1/4$	0
	x_4	$1/2$	0	0	-2	1	$1/2$	0
	S	$-1/2$	0	0	0	0	$-1/2$	1
	$z_j - c_j \to$		0	0	$1/2$	0	$1/4$	0

Note: $z_j - c_j \geq 0$ by $S < 0$ $\therefore$ invoke the dual simplex.

v_B	x_B	y_1	y_2	y_3	y_4	y_5	y_s
x_1	$10/4$	1	0	$-1/2$	0	0	$1/2$
x_2	$10/4$	0	1	$3/2$	0	0	$-1/2$
x_4	0	0	0	-2	1	0	1
x_5	1	0	0	0	0	1	-2
	$z_j - c_j \rightarrow$	0	0	$1/2$	0	0	$1/2$

We can choose row of x_1 or x_2 for next cut. Let $r=1$.

cut is $\dfrac{x_3}{2} + \dfrac{s}{2} \geq \dfrac{1}{2}$

or $-x_3/2 - s/2 + s_1 = -\dfrac{1}{2}$, $s_2 \geq 0$, s_2 slack

y_B	x_B	y_1	y_2	y_3	y_4	y_5	y_s	y_{s_1}
x_1	$10/4$	1	0	$-1/2$	0	0	$1/2$	0
x_2	$10/4$	0	1	$3/2$	0	0	$-1/2$	0
x_4	0	0	0	-2	1	0	1	0
x_5	1	0	0	0	0	1	-2	0
s_1	$-1/2$	0	0	$-1/2$	0	0	$-1/2$	1 →
	$z_j - c_j$ →	0	0	$1/2$ ↑	0	0	$1/2$	0

Invoke the dual simplex

Next table gives optimal soln ($x_1 = 3$, $x_2 = 1$, $x_3 = 1$, $x_4 = 2$, $x_5 = 1$) — all integers. So stop optimal value = 7.

Graphically:

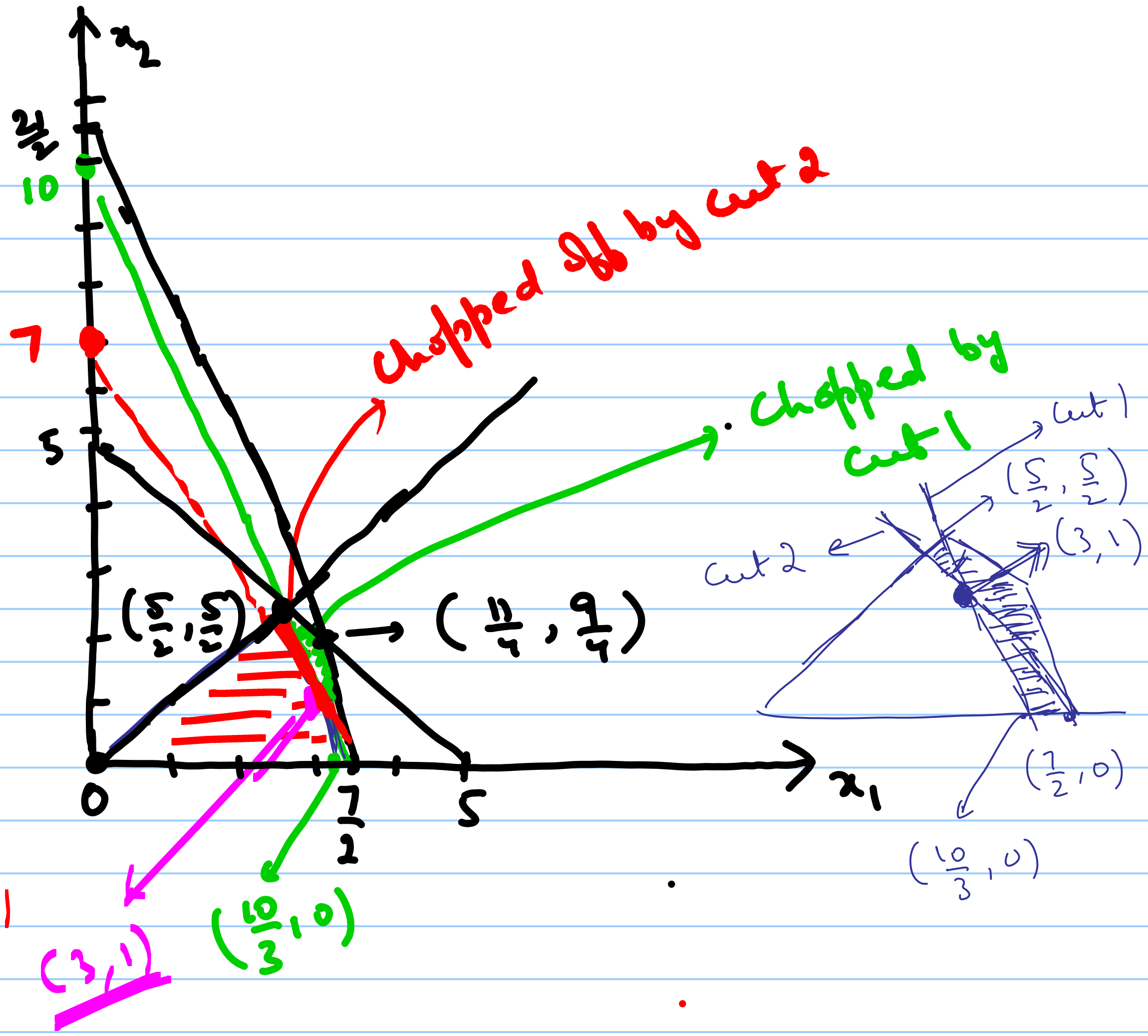
$$\max z = 2x_1 + x_2$$

s.to $\quad x_1 + x_2 \le 5$

$\quad -x_1 + x_2 \le 0$

$\quad 6x_1 + 2x_2 \le 21$

$\quad x_1, x_2 \ge 0$



Cut 1: $\frac{x_5}{2} \ge \frac{1}{2}$

$\Rightarrow x_5 \ge 1 \Rightarrow 21 - 6x_1 - 2x_2 \ge 1$

$\Rightarrow 3x_1 + x_2 \le 10$

cut 2 $\quad \dfrac{x_3}{2} + \dfrac{s}{2} \geq \dfrac{1}{2}$

$$\Rightarrow \quad x_3 + s \geq 1$$

$$\Rightarrow \quad (5 - x_1 - x_2) + \left(\frac{21 - 6x_1 - 2x_2 - 1}{2}\right) \geq 1$$

$$\Rightarrow \quad 2x_1 + x_2 \leq 7$$

Gomory cut for mixed ILP:

Let x_r be an integer restricted variable in MILP

From the optimal table of the relaxed associated LPP,

let x_r^* be optimal value & not integer right now. From the row in the optimal table, we have

$$x_r + \sum_{i \in K} y_{ri} x_i = x_r^* \quad \text{(pivot row)}$$

or $$x_r + \sum_{i \in K} y_{ri} x_i = \lfloor x_r^* \rfloor + f_r \quad , \quad 0 < f_r < 1. \quad — (1)$$

For x_r to be integer, either $x_r \leq \lfloor x_r^* \rfloor$ or $x_r \geq \lfloor x_r^* \rfloor + 1$

$\therefore$ (1) $\Rightarrow$ $$\sum_{i \in K} y_{ri} x_i \geq f_r \quad — (2)$$

$$\sum_{i \in K} y_{ri} x_i \leq f_r - 1 \quad — (3)$$

K is the index set of all non-basic variables. Define

K^+ : set of indices in K for which $y_{ri} \geq 0$

K^- : set of indices in K for which $y_{ri} < 0$

$\therefore$ 2 & 3 $\Rightarrow$

$$\sum_{i \in K^+} y_{ri} x_i + \sum_{i \in K^-} y_{ri} x_i \geq f_r$$

$$\Rightarrow \sum_{i \in K^+} y_{ri} x_i \geq f_r \quad — (4)$$

or

$$\sum_{i \in K^+} y_{ri} \overbrace{x_i}^{\geq 0} + \sum_{i \in K^-} y_{ri} x_i \leq \overbrace{f_r - 1}^{< 0}$$

$$\Rightarrow \frac{f_r}{f_r - 1} \sum_{i \in K^-} y_{ri} x_i \geq f_r \quad — (5)$$

One of these inequalities must hold, and both LHS expressions are non-negative, so we combine them

$$\sum_{i \in K^+} y_{ri} x_i + \frac{f_r}{f_r - 1} \sum_{i \in K^-} y_{ri} x_i \geq f_r$$

$\forall\, x$ feasible for ILP

Add a slack

$$s - \left(\sum_{i \in K^+} y_{ri} x_i + \frac{f_r}{f_r - 1} \sum_{i \in K^-} y_{ri} x_i \right) = -f_r \,, \quad s \geq 0$$

is the cut equation for mixed ILP.

We can invoke dual simplex to solve the problem.

eg $\max z = x_1 + x_2$

Subject to $2x_1 + 5x_2 \le 16 \longrightarrow 2x_1 + 5x_2 + x_3 = 16$

$6x_1 + 5x_2 \le 30 \longrightarrow 6x_1 + 5x_2 + x_4 = 30$

$x_1, x_2 \geq 0$, x_1 : integer $\quad x_1, x_2, x_3, x_4 \geq 0$

x_1: integer

c_B	v_B	x_B	y_1	y_2	y_3	y_4	
1	x_2	9/5	0	1	3/10	-1/10	
1	x_1	7/2	1	0	$\underline{-1/4}$	$\underline{1/4}$	— pivot row
		$z_j - c_j \rightarrow$	0	0	1/20	3/20	

x_1 is required to be integer. Pivot row is that of x_1

$r = 1$. Also, $K = \{3, 4\}$ & $K^+ = \{4\}$, $K^- = \{3\}$, $f_r = 1/2$

The cut inequality is

$$\frac{1}{4} x_4 + \frac{\frac{1}{2}}{\frac{1}{2}-1}\left(-\frac{1}{4} x_3\right) \geq \frac{1}{2}$$

$$\Rightarrow \frac{1}{4} x_4 + \frac{1}{4} x_3 \geq \frac{1}{2}$$

$$\Rightarrow S - \frac{1}{4} x_3 - \frac{1}{4} x_4 = -\frac{1}{2}, \quad S \geq 0$$

c_B	v_B	x_B	y_1	y_2	y_3	y_4	y_s
1	x_2	9/5	0	1	3/10	-1/10	0
1	x_1	7/2	1	0	-1/4	1/4	0
0	s	-1/2	0	0	-1/4	-1/4	1
		$z_j - c_j \rightarrow$	0	0	1/20	3/20	0

Apply the dual simplex method.

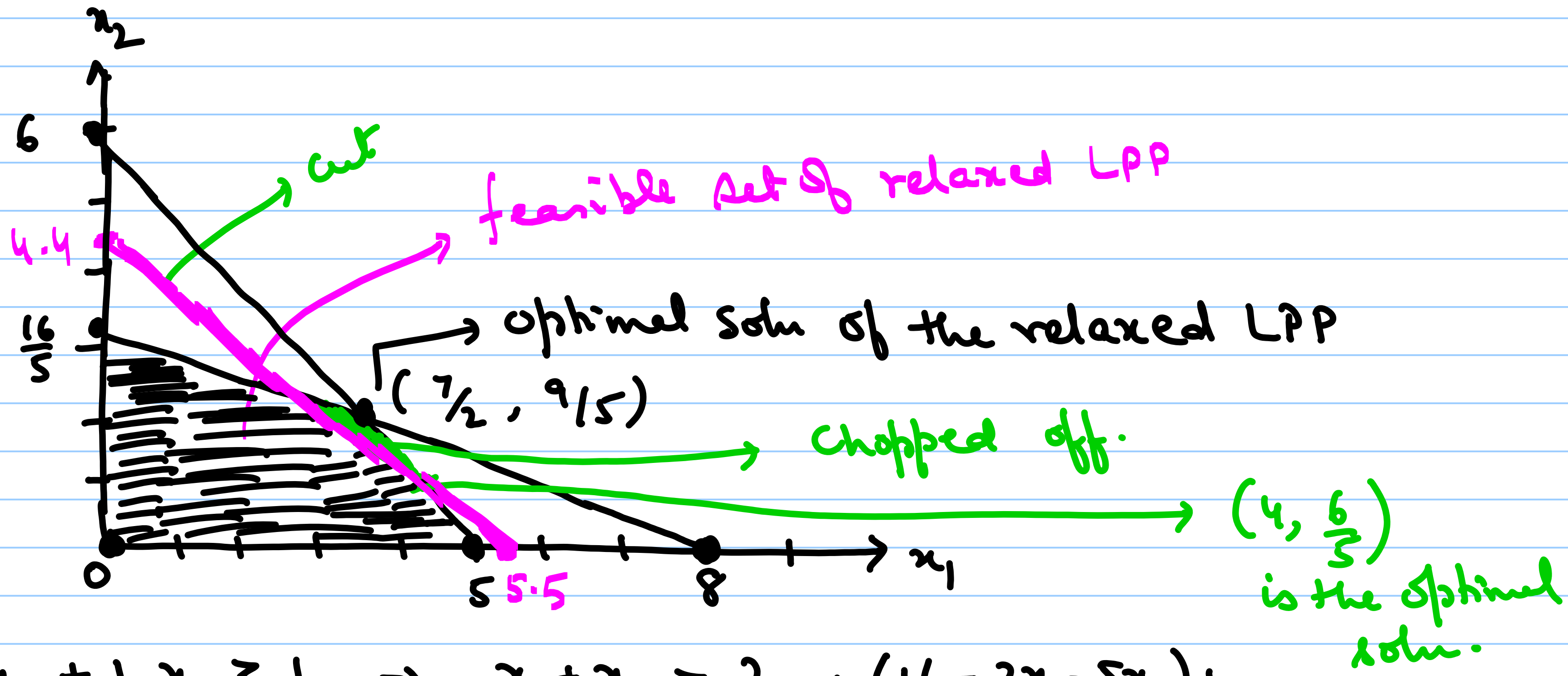
$$\max\left\{ \frac{z_j - c_j}{y_{ij}} \;\middle|\; y_{ij} < 0 \right\}$$

$$= \max\left\{ (1/20)/(-1/4),\ (3/20)\ /\ (-\tfrac{1}{4}) \right\} = -\frac{1}{5} \Rightarrow x_3 \text{ enters.}$$

C_B	V_B	x_B	y_1	y_2	y_3	y_4	y_5
1	x_2	$6/5$	0	1	0	$-2/5$	$6/5$
1	x_1	4	1	0	0	$1/2$	-1
0	x_3	2	0	0	1	1	-4
		$z_j - c_j$	0	0	0	$1/10$	$4/5$

$\therefore$ optimal soln is $x_1^* = 4$, $x_2^* = 6/5$ & $Z = 26/5$

Graphical:



Cut $\quad \frac{1}{4} x_3 + \frac{1}{4} x_4 \geq \frac{1}{2} \Rightarrow x_3 + x_4 \geq 2 \Rightarrow (16 - 2x_1 - 5x_2) +$
$(30 - 6x_1 - 5x_2) \geq 2$

$\Rightarrow 4x_1 + 5x_2 \leq 22$

One of the major disadvantage is that if the calculations are stopped prematurely during the implementation of this method in a large scale problem due to time limitation or cost effectiveness, then we will have "no good" integer solution (even to some approximation) in hand. Another problem is while working on computing devices, the round-off or truncation errors can cause issue in correctly distinguishing b/w integer and non-integer valued-variables.

Branch and Bound Method

Consider a linear binary variable program

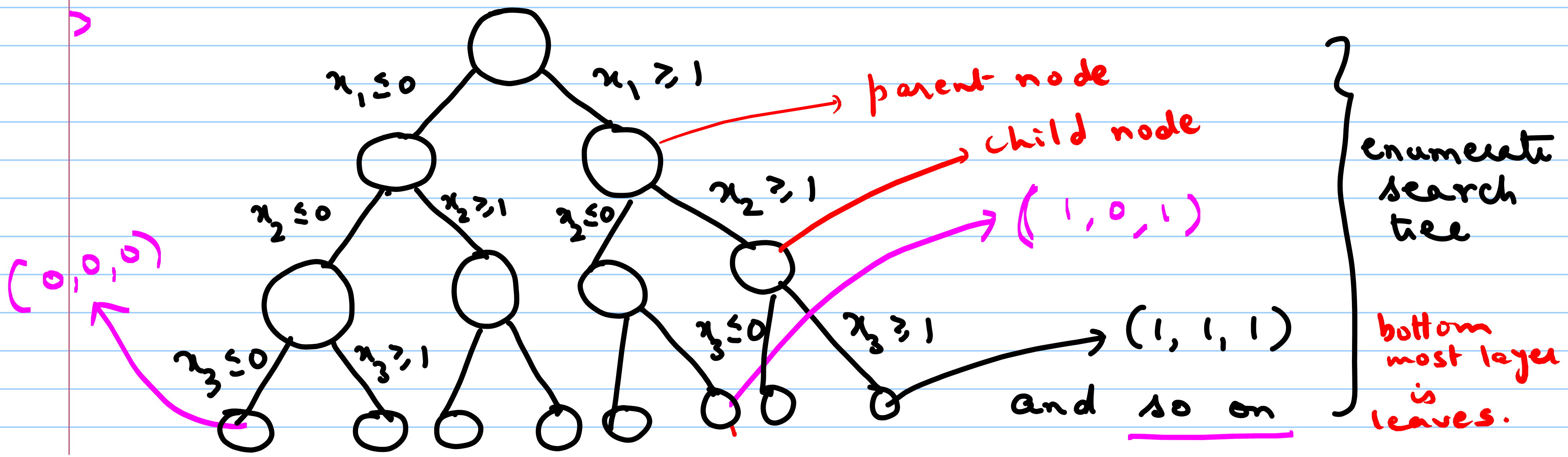
$$(P) \quad \max \ c^T x$$
$$\text{s.to} \quad Ax \le b$$
$$x \in \{0,1\}^n \iff 0 \le x \le 1, \ x \text{ integer}$$

One way is to look for complete enumeration of 2^n tupples of 0, 1 entries (brute force), and then obtain the one which is feasible and optimal among them.

A recursive way of enumerating all solutions is create search tree using one variable at a time.

For instance, let the variables for enumeration be ordered as $x_1, x_2, \ldots, x_n$

First solve LPP max $c^T x$ s.to $Ax \le b$, $0 \le x \le 1$.



But for large n, we can not afford to enumerate all 2^n 0,1 tupples. We have to build some logic to reduce the branching search. This is done by eliminating a subtree from further enumeration. We eliminate a subtree or prune the branch of the tree when we realize that the ILP soln at the root of the subtree can not be optimal.

First solve the relaxed LP problem, LP(1), and let $x(1)$ be its optimal soln.

Incumbent: It is a feasible solution of ILP. It is also the

best solution for ILP so far (with the best objective value)
We denote it by z_I.

Typically, when we start the B&B algo, we may not have an incumbent in hand (though in certain special cases, we have subroutines that find out a feasible integer solutions with a large objective values. The best of these is the initial incumbant);
we may have to wait till we get initial incumbant. We till then take $z_I = -\infty$.

Once we have an optimal soln $x(1)$ of $LP(1)$ at node 1, We identify the variable (say x_1 in $x(1)$) taking fractional value to branch the node into two child nodes LP(2) and LP(3) by inserting the constraints $x_1 \leq 0$ and $x_1 \geq 1$, respectively, in LP

Let LP(j) be the relaxed LP at j^{th} node

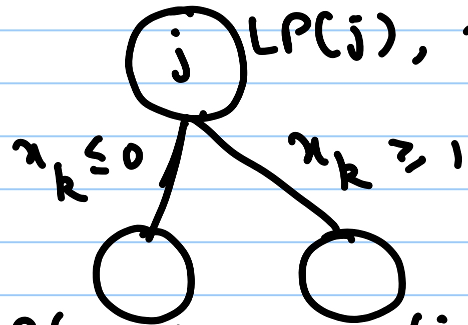
$z_{LP}(j)$ be optimal value of LP(j)

IP(j) be the integer problem with LP(j)

$z_{IP}(j)$ be the optimal value of IP(j)

Note : $z_{IP}(j) \leq z_{LP}(j), \forall j$

If any LP(j) is infeasible then prune that branch and subsequent subtree. If at any node j, $z_{LP}(j) \le z_I$ then at subsequent nodes with root node j, the objective value will decrease; means



$LP(j)$, $x(j)$, $z_{LP}(j) \le z_I$

$LP(j+1)$ $\quad$ $LP(j+2)$; $z_{LP}(j+2) \le z_{LP}(j) \le z_I$

$z_{LP}(j+1) \le z_{LP}(j) \le z_I$.

no improvement will happen in the objective fn of ILP, so prune such node.

The nodes which are not explored in the tree are called active nodes. Each active node must be enumerated.

If at any node j, $LP(j)$ is infeasible then $ILP(j)$ is also infeasible so prune the node and stop enumerating the subtree from this pruned node.

In case $LP(j)$ is feasible, possesses optimal solution $x(j)$ with optimal objective value $z_{LP}(j) > z_I$ (say), then enumerate node j. Identify the variable for branching and create two subnodes with new LPP's listed on them.

→ Select an active node j

→ Solve LP (j). Let optimal soln be $x(j)$, optimal value $z_{LP}(j)$

→ If $z_{LP}(j) \leq z_I$ then prune node j.

→ If $z_{LP}(j) > z_I$ and $x(j)$ is feasible for ILP (j), then set incumbant soln by $x(j)$ and $z_I := z_{LP}(j)$

Fathom node j

→ If $z_{LP}(j) > z_I$ and $x(j)$ is not feasible for ILP (j) then identify the variable to branch at node j.

and mark the two children nodes of j as "active".

→ Mark node j inactive.

<u>For eg</u>: max $24x_1 + 2x_2 + 20x_3 + 4x_4$

s.to $8x_1 + x_2 + 5x_3 + 4x_4 \leq 9$

$x_i \in \{0,1\}, \forall i$

Suppose incumbant is $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$. Then, $Z_I = 26$

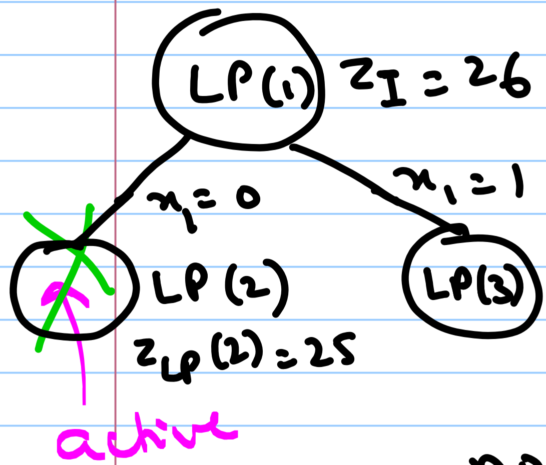
Now, suppose active node $j = 2$ and we are solving the following LP(2) problem

$LP(2)$ $\quad \max \; 24x_1 + 2x_2 + 20x_3 + 4x_4$

$\quad$ s.to $\quad 8x_1 + x_2 + 5x_3 + 4x_4 \le 9$

$$x_1 = 0$$

$$0 \le x_2 \le 1, \quad 0 \le x_3 \le 1, \quad 0 \le x_4 \le 1$$



Optimal soln is $(0, 1, 1, 3/4)$ & $Z_{LP}(2) = 25$

Since $Z_{LP}(2) < Z_I = 26$, if we further branch node 2, then the objective value comes down but we donot want it. So prune the node 2.

We then go to the next active node LP(3). And so on.

when choosing the next node (among the list of active nodes), we want

- do not jump too much to preserve and handle the datastructure flow and keep in mind the limitation on memory required.
- to converge to the optimal soln quickly.

Depth first search (DFS):– found to be very efficient and fast; though can stuck up in suboptimal search tree.

Best first search (BFS) :– look at the node with best relaxed value next. But found to be ill-worked.

Eg. $\max \; 9x_1 + 3x_2 + 5x_3 + 3x_4$

s.to $5x_1 + 2x_2 + 5x_3 + 4x_4 \le 10$

$x_i \in \{0,1\}, \; \forall i$

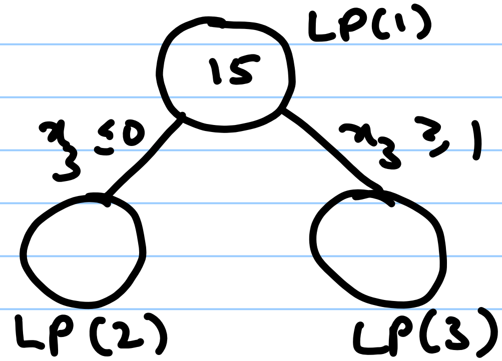
Initially, let $Z_I = -\infty$

LP(1) $\max \; 9x_1 + 3x_2 + 5x_3 + 3x_4$

s.to $5x_1 + 2x_2 + 5x_3 + 4x_4 \le 10$

$0 \le x_i \le 1, \quad \forall \; i = 1, 2, 3, 4$

$x(1) = (1, 1, \cdot 6, 0)$ & $z_{LP}(1) = 15$



LP(2) LP(1) and $x_3 = 0$

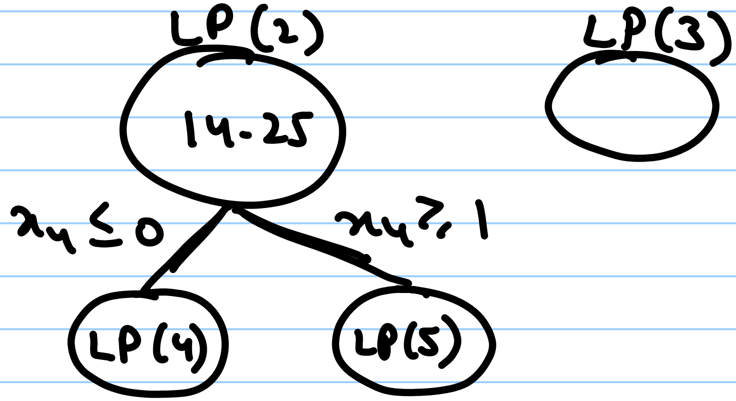
LP(3) LP(1) and $x_3 = 1$

Mark node 1 as inactive. There are two active nodes now.

Let us enumerate node 2 for LP(2)

LP(2) : optimal soln $x(2) = (1, 1, 0, 0.75)$ and

$$Z_{LP}(2) = 14.25 > Z_I$$



Branch via variable x_4 at LP(2) and make node 2 inactive.

We have 3 active nodes – node 3, node 4, node 5.

Let us enumerate node 4

or LP(4) := LP(2) with $x_4 \leq 0$.

LP(4) : optimal soln $x(4) = (1, 1, 0, 0)$ and optimal value $Z_{LP}(4) = 12$.

Note this solution is feasible for ILP(4); so change incumbant $Z_I = 12$, and fathom node 4. (mark it inactive).

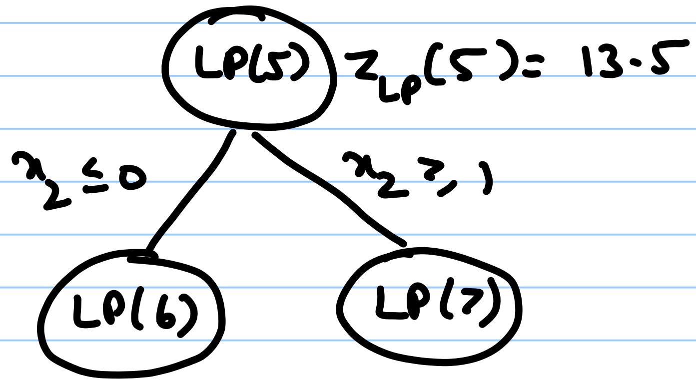
Now, the inactive nodes are node 5 and node 3.

let us backtrack. Choose node 5 or LP(5)

$\therefore$ LP(5) is LP(2) with $x_4 \geq 1$

optimal soln is $x(5) = (1, \frac{1}{2}, 0, 1)$, and optimal value $Z_{LP}(5) = 13.5 > Z_I = 12$.

So, branch node 5 by variable x_2 and then mark node 5 inactive. Enumerate LP(6)



LP(6) is

$$\max \; 9x_1 + 3x_2 + 5x_3 + 3x_4$$

$$\text{s.to} \quad 5x_1 + 2x_2 + 5x_3 + 4x_4 \le 10$$

$$\left\{ \begin{array}{l} 0 \le x_i \le 1, \quad \forall\, i = 1,2,3,4 \\ x_3 \le 0 \\ x_4 \ge 1 \\ x_2 \le 0 \end{array} \right.$$

$x_3 = 0$
$x_4 = 1$
$x_2 = 0$

Optimal soln $x(6) = (1, 0, 0, 1)$ and $Z_{LP}(6) = 12$

This being integer feasible soln so fathom the node and

$Z_I = 12$ (still).

Inactive nodes are now node 7 and node 3.

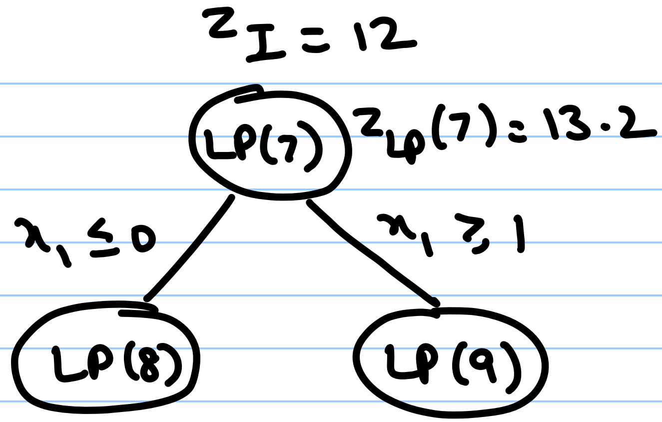
let us enumerate LP(7)

i.e. same as LP(6) but with $x_2 \le 0$ replaced by $x_2 \ge 1$

optimal soln $x(7) = (0.8, 1, 0, 1)$ and $Z_{LP}(7) = 13.2 > Z_I = 12$

$\therefore$ Branch out from node 7 into two branches via variable x_1 and mark node 7 inactive.

$\therefore$ Next, we have 3 nodes active: node 8, node 9, and node 3.



Consider
LP(8) : LP(7) with $x_1 \le 0$

$\therefore$ $x(8) = (0, 1, 0, 1)$ and $z_{LP}(8) = 6$

Integer feasible soln, so fathom the node 8, but $z_{LP}(8) < z_I$

so do not change the incumbent value or bound z_I.

Consider
LP(9) : LP(7) with $x_1 \ge 1$

This model is infeasible, so prune the node and mark node 9 inactive.

The only active node is 3.

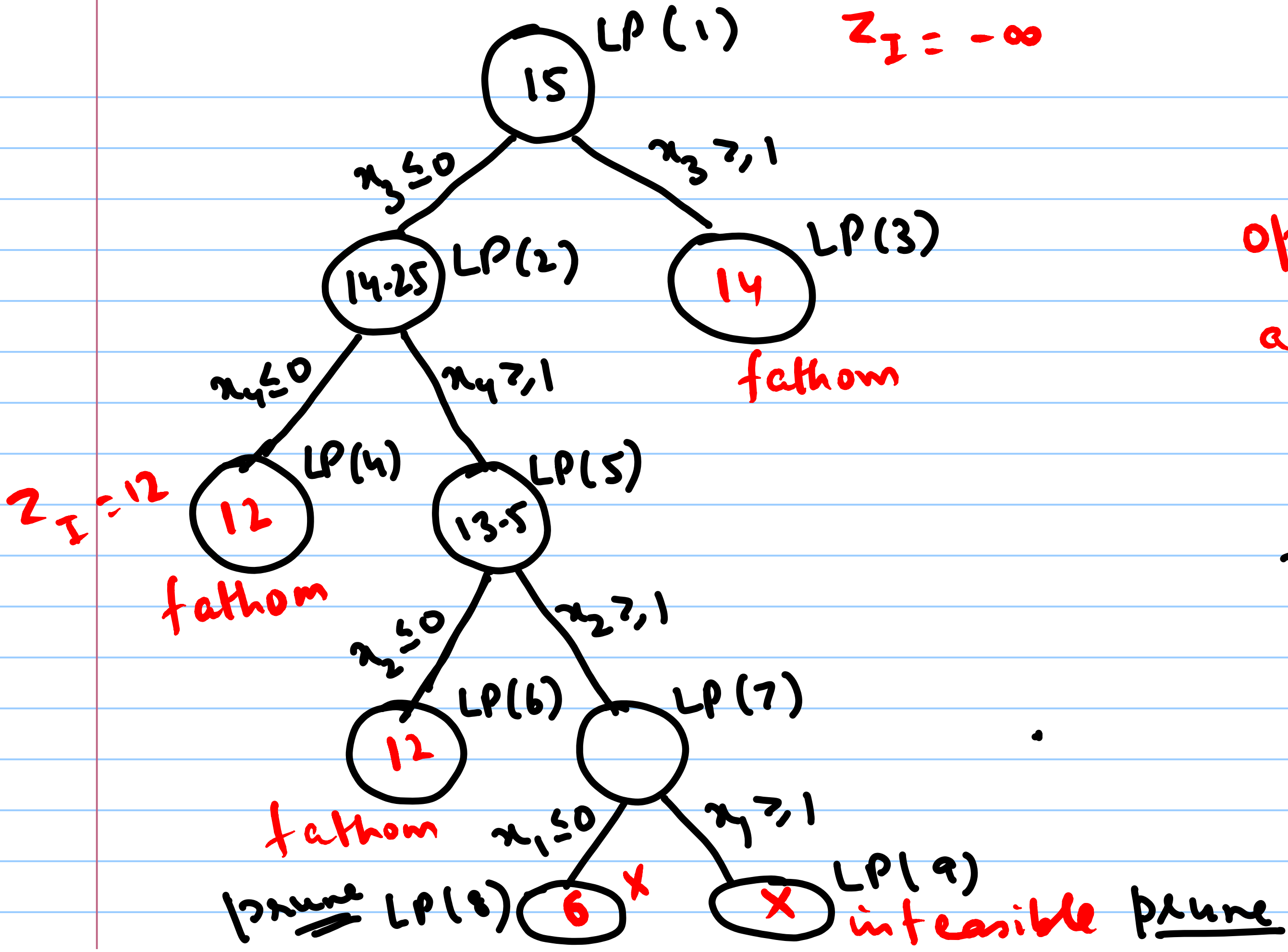
$\therefore$ Consider $LP(3) = LP(1)$ with $x_3 \geq 1$.

optimal soln $x(3) = (1,0,1,0)$ and $Z_{LP}(3) = 14$

This is feasible soln of ILP and $Z_{LP}(3) > Z_I$.

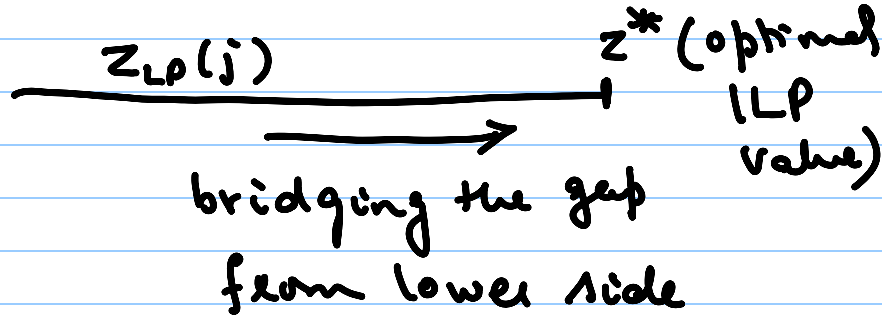
$\therefore$ change $Z_I = 14$ and mark node 3 as inactive.

Now the active node set is Φ; so stop and the final optimal soln of ILP is $(1,0,1,0)$ with optimal value $= 14$.



optimal value = 14 = z_I

and optimal soln (1, 0, 1, 0)



Eg: $\max\ z = 7x_1 + 9x_2$

subject to $-x_1 + 3x_2 \le 6$

$$7x_1 + x_2 \le 35$$

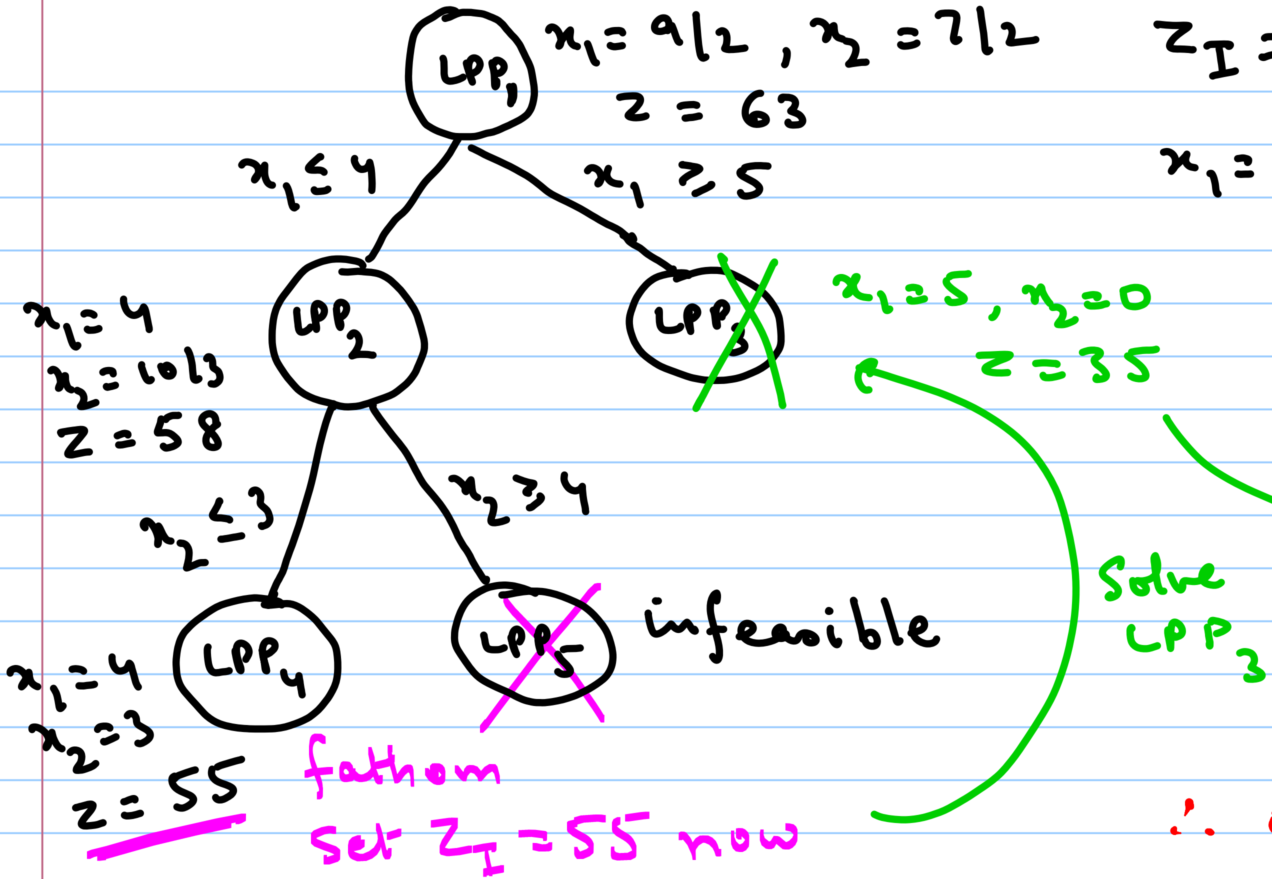
$$x_2 \le 7$$

$x_1, x_2 \ge 0$ & both integers.

Solve the associated relaxed LPP.

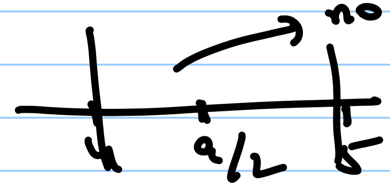
Optimal soln is $x = (x_1 = 9/2,\ x_2 = 7/2)$ and $z = 63$

If we can identify incumbant say $x_1 = 1, x_2 = 1$, integer feasible, then set $z_I = 16$, else simply set $z_I = -\infty$.



$Z_I = -\infty$ (say)

$x_1 = 9/2 \quad \lfloor x_1 \rfloor = 4$



no integer value of x_1

$z < Z_I = 55$

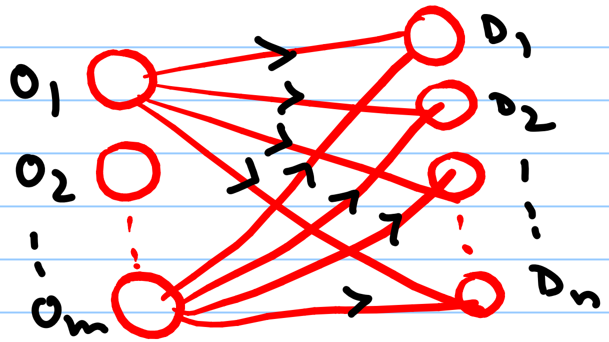
$\Rightarrow$ prune the node

$\therefore$ optimal soln is

$x_1^* = 4, \; x_2^* = 3, \; z^* = 55$

Transportation Problems

It is a special case of network probleme. Here, there are finite number of nodes called origins and a finite number of nodes called receivers / destinations. A single homogenous commodity is to be transported from the origins to the destinations.



Each origin can transport to any destination and any destination can receive product from any origin.

There is no transport b/w origins and b/w destinations for this commodity.

One can see it as a graph problem where the graph $G = (V, E)$ is a complete bipartite graph.

A graph $G=(V,E)$ is called a bipartite graph if we can express $V = V_1 \cup V_2$, $V_1 \cap V_2 = \Phi$, and any edge $(u, v) \in E$, the vertex $u \in V_1$ and vertex $v \in V_2$.

Also, it is complete for each node in V_1 and node in V_2 are connected by an edge in E.

We can associate some weight with each edge in the frame-work. This weight can be treated as a cost of transportation of 1 unit of product. So, we have a cost matrix $[c_{ij}]$, c_{ij} : cost of transporting one unit of product from O_i to D_j. Let each origin O_i has a_i units available and each destination D_j has a minimum demand of b_j units.

The aim is to find $x_{ij} \geq 0$, units to be transported from O_i to D_j, so as to minimize the overall cost of transportation in the network.

$$(P) \quad \text{Min} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \, x_{ij}$$

	D_1	D_2	$\cdots$	D_n	a_i
O_1	c_{11}	c_{12}		c_{1n}	a_1
O_2	c_{21}	c_{22}	$\cdots$	c_{2n}	a_2
$\vdots$					
O_m	c_{m1}	c_{m2}	$\cdots$	c_{mn}	a_m
b_j	b_1	b_2		b_n	

s.to
$$\sum_{j=1}^{n} x_{ij} \leq a_i \ , \quad i = 1, 2, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} \geq b_j \ , \quad j = 1, 2, \ldots, n$$

$$x_{ij} \geq 0 \ , \ \forall \ i, j.$$

<u>**Balanced TP**</u>: If all constraints hold as equality in (P)

means $\sum_{j=1}^{n} x_{ij} = a_i$, $\sum_{i=1}^{m} x_{ij} = b_j$ $\forall$ $i=1,\ldots,m$
$j=1.\ldots n$

The BTP is feasible iff

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

<u>Proof</u> :- Let $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = M > 0$

Take $x_{ij} = \frac{a_i b_j}{M} \geq 0 \quad \forall (i,j)$

Then, $[x_{ij}]$ is feasible for BTP as

$$\sum_{j=1}^{n} x_{ij} = \frac{a_i}{M} \sum_{j=1}^{n} b_j = a_i \quad \forall\, i$$

& $$\sum_{i=1}^{m} x_{ij} = \frac{b_j}{M} \sum_{i=1}^{m} a_i = b_j \quad \forall\, j$$

$\therefore$ BTP is feasible.

Converse follows from the constraints of BTP naturally.

Also, a feasible BTP is always bounded because

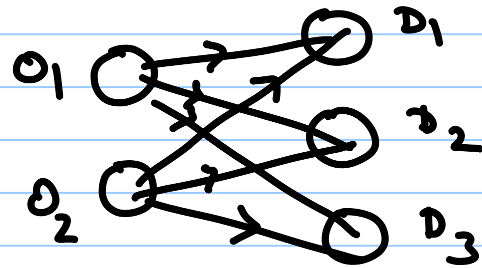
$$0 \le x_{ij} \le \min\{a_i, b_j\}, \quad \forall\, i, j.$$

$\Rightarrow$ $\sum \sum c_{ij} x_{ij}$ is finite in the feasible set of BTP.

Without loss of generality we can assume each a_i and b_j is a non-negative integers with atleast one $a_i > 0$, $b_j > 0$.

Let us see the structure of the coefficient matrix A from the constraints of BTP.

Suppose we have 2 origins and 3 destinations.



$$\begin{cases} x_{11} + x_{12} + x_{13} = a_1 \\ x_{21} + x_{22} + x_{23} = a_2 \end{cases}$$

$$\begin{cases} x_{11} + x_{21} = b_1 \\ x_{12} + x_{22} = b_2 \\ x_{13} + x_{23} = b_3 \end{cases}$$

$$
A = \begin{array}{c}
\begin{array}{cccccc} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} \end{array} \\
\left[\begin{array}{ccc|ccc}
1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 \\
\hline
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1
\end{array}\right]_{5 \times 6}
\end{array}
= \left[\begin{array}{c|c} E_1 & E_2 \\ I_3 & I_3 \end{array}\right]
$$

In general, with m origins and n destinations

$$
A = \begin{array}{c}
\begin{array}{ccccccccccc} x_{11} & x_{12} & \cdots & x_{1n} & x_{21} & \cdots & x_{2n} & & x_{m1} & \cdots & x_{mn} \end{array} \\
\left[\begin{array}{cccc|cccc|c|cccc}
1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & 0 & & 0 & 1 & 1 & \cdots & 1 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & & & & & & & & & & & & \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 & 1 & & 1 \\
\hline
1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & \cdots & 1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 & \cdots & 0 & 1 & \cdots & 0 \\
0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 & \cdots & 0 & 0 & \cdots & 1
\end{array}\right]_{(m+n) \times mn}
\end{array}
$$

m

n

$$A = \left[\begin{array}{c|c|c|c} E_1 & E_2 & --- & E_m \\ I_n & I_n & & I_n \end{array}\right]_{(m+n)\times(mn)}$$

Result: Rank of A is $m+n-1$.

Proof :- Let R_k denote the k^{th} row of A. Then, note

$$\sum_{k=1}^{m} R_k - \sum_{k=m+1}^{n} R_k = 0 \Rightarrow$$ one of the row is a linear combination of the other $m+n-1$ rows.

$\Rightarrow \operatorname{rank}(A) < m+n$ (note, typically $m+n \leq mn$, for $m \neq 1, n \neq 1$)

Next we produce a submatrix B of A such that B is $(m+n-1)\times(m+n-1)$ and $|B| \neq 0 \Rightarrow \operatorname{rank}(A) = m+n-1$

Define a submatrix

$$\hat{B} = \begin{bmatrix} a_{1n} & a_{2n} & \cdots & a_{mn} & a_{11} & a_{12} & \cdots & a_{1n-1} \end{bmatrix}_{(m+n)\times(m+n-1)}$$

and delete the last row from $\hat{B}$ to get

$$B = \begin{array}{c} \\ m \\ \\ \\ \\ n-1 \\ \\ \end{array} \left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ \hline 0 & 0 & & 0 & 1 & 0 & & 0 \\ 0 & 0 & & \vdots & 0 & 1 & & \vdots \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & & 0 & 0 & 0 & & 1 \end{array}\right] = \begin{bmatrix} I_m & E_1 \\ 0 & I_n \end{bmatrix}$$

$|B| = 1 \neq 0$ and

B is $(m+n-1)\times(m+n-1)$

$\therefore$ In a BTP, the no. of variables are mn and basic variables are $m+n-1$. So, $mn - m - n + 1 = (m-1)(n-1)$

variables are non-basic and this number could be very large.

Result: A is a unimodular matrix.

Before proving it, let us define a <u>unimodular</u> matrix.

Let M be any matrix with entries from the set $\{0, 1, -1\}$ only.

M is called a unimodular matrix if every square sub-matrix of M has determinant 1, −1, or 0.

eg. $M = \begin{bmatrix} 1 & 1 & 0 \\ -1 & -1 & 0 \\ 1 & -1 & -1 \end{bmatrix}$, $|M| = -1(-1+1) = 0$

But det. $= 2$

Not a unimodular matrix.

eg. $M = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ is unimodular.

Proof : Let A_k be any $k \times k$ submatrix of A. Three cases arise:

(i) $\exists$ a column in A_k with all entries zero $\Rightarrow |A_k| = 0$

(ii) every column of A_k has exactly 2 ones $\Rightarrow |A_k| = 0$
(rank result argument)

(iii) $\exists$ a column in A_k with exactly 1 one.

Open the determinant of A_k with this column

$\Rightarrow |A_k| = \pm |A_{k-1}|$

Apply now the same argument of 3 cases to A_{k-1} to get

$|A_k| = 0$ or $|A_k| = \pm |A_{k-2}|$ & so on.

Finally, we get a 1×1 submatrix A_1 which is 1 or 0.

$\therefore \quad |A_k| = 0 \quad$ or $\quad |A_k| = \pm 1$

$\Rightarrow$ Each square submatrix of A has determinant 0 or ± 1, giving A is a unimodular matrix.

Note as rank of A is $m+n-1$; every basis matrix of BTP will have $m+n-1$ basic columns. By unimodularity, $|B| = \pm 1$.

Let a_{ij} by $((i-1)n+j)^{th}$ column of A

a_{11} : 1st column of A - - a_{21} : $(n+1)$th column of A

a_{12} : 2^{nd} column of A $\quad a_{2n}$: $2n^{th}$ column of A

a_{mn} : mn^{th} column of A .

Each $a_{ij} = e_i + e_{m+j}$

where e_k is the $(m+n)\times 1$ column with k^{th} entry 1 and all other entries 0.

Let $B = [a_{\alpha\beta}]$ be a basis matrix. If a_{ij} is a column in A which is not in B, then we can write

$$a_{ij} = \sum_{\alpha} \sum_{\beta} y_{\alpha\beta}\, a_{\alpha\beta}$$

Note: each $y_{\alpha\beta} \in \{0, 1, -1\}$. So, the pivot element (if we apply the simplex to solve BTP) have value +1 always. In that case, when we move from B to next $\hat{B}$

the basic variables formula becomes

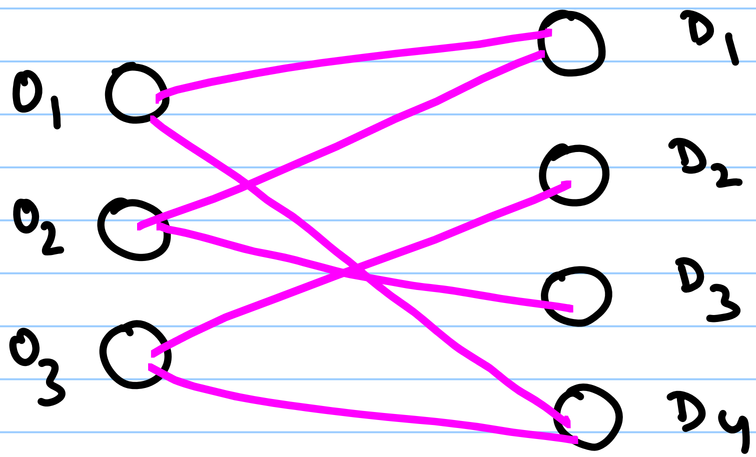
$$\hat{x}_{\hat{B}_i} = \begin{cases} x_{B_i} + x_{B_r} & \text{if } y_{ij} = -1 \\ x_{B_i} - x_{B_r} & \text{if } y_{ij} = +1 \\ x_{B_i} & \text{if } y_{ij} = 0 \end{cases}$$
$i \neq r$

& $\hat{x}_{\hat{B}_r} = x_{B_r}$.

Note in this formula one do not require any multiplication or division ; and thus pivoting step and computing new BFS is much more easier by addition / subtraction only.

Let us see what is happening from graph point of view

let $m = 3$, $n = 4$



No. of basic variables

$= m+n-1 = 6$

and these are independent.

In graph the 6 edges of basic variable forms a tree.

A <u>tree</u> in graph theory is a graph $G = (V, E)$, which is connected and acyclic.

We have just randomly took 6 edges (violet) to depict

BFS in hand; basic variables are $x_{11}, x_{14}, x_{21}, x_{23}, x_{32}, x_{34}$.

If we write a symbolic matrix of A as

$$\begin{bmatrix} a_{11} & a_{12} & \boxed{a_{13}} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$$

The columns forming current B are circled. We can find atleast one circle in each row and atleast one circle cell in each column $\therefore$ A is a unimodular matrix.

Now if the current BFS is not optimal we have to identify a variable (or column) to enter in B. Suppose that variable is x_{13} or column a_{13} depicted in red in the matrix (back pg). This will lead to a formation of unique loop. We can see

$$a_{13} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = a_{11} - a_{21} + a_{23}$$

or $a_{13} - a_{11} + a_{21} - a_{23} = 0$. (The +, − sign in loop)

Instead of x_{13} or a_{13} suppose a_{33} (or x_{33}) enters into the basis

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & \boxed{a_{33}} & a_{34} \end{bmatrix}$$

$$a_{33} - a_{34} + a_{14} - a_{11} + a_{21} - a_{23} = 0$$

$$\Rightarrow a_{33} = a_{34} - a_{14} + a_{11} - a_{21} + a_{23}$$

$$\left\{ \begin{array}{l} \text{loop in graph is} \\ D_3 \to O_3 \to D_4 \to O_1 \to D_1 \to O_2 \to D_3. \end{array} \right.$$

$\therefore$ $y_{34} = 1$

$y_{11} = 1$

$y_{23} = 1$

Thereafter, we have to find the outgoing variable by taking min. ratio $\left\{ \frac{x_{Bi}}{y_{ij}} \mid y_{ij} > 0 \right\}$. $\therefore$ Here, we would be taking the min value of $x_{B_{st}}$ $\forall$ (s,t) : $y_{st} = +1$ in the branches of the cycle created.

Also recall to find entering variable we need to compute $z_{ij} - c_{ij}$ and for minimization, the optimality criterion is that $z_{ij} - c_{ij} \leq 0$. These opportunity costs are computed using the dual problem. So, we have to write the dual of BTP.

We write the dual of BTP

$$(P) \quad \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_{ij} = a_i, \quad i = 1, \dots, m \quad \leftarrow u_i$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, \dots, n \quad \leftarrow v_j$$

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

dual var.

Dual

$$(D) \quad \max \sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j$$

$$\text{subject to} \quad u_i + v_j \leq c_{ij} \quad \forall (i,j)$$

$$u_i, v_j \in \mathbb{R} \quad \forall (i,j)$$

At optimality

$$\sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\Rightarrow \sum_{i=1}^{m} u_i \sum_{j=1}^{n} x_{ij} + \sum_{j=1}^{n} v_j \sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\Rightarrow \sum_{i=1}^{m} \sum_{j=1}^{n} (u_i + v_j - c_{ij}) x_{ij} = 0 \quad \text{— } (1)$$

For $x_{ij} > 0$, we must have $u_i + v_j - c_{ij} = 0$ — (2)

otherwise $u_i + v_j - c_{ij} \leq 0$ by dual feasibility.

In (2), we have $m+n-1$ eqns and $m+n$ variables. To determine them we pick one of them say $u_1 = 0$ and computes all other u_i & v_j. So, the dual soln is fully known.

For non-basic cells compute $z_{ij} - c_{ij} = u_i + v_j - c_{ij}$

If all are ≤ 0, then current BFS is optimal soln. else we find the cell (s,t) such that $u_s + v_t - c_{st} > 0$ and most +ve. Choose the cell (s,t) to enter in the basis. In case of multiple choices for (s,t), pick the one arbitrarily.

$\therefore$ In the next iteration x_{st} is a basic variable.

Thereafter, starting from cell (s,t) construct a unique loop & this will be used to identify the outgoing pair (i, j).

To find the first BFS: There are quite a few methods for finding initial BFS. We discuss Vogel's approximation method (VAM).

In the cost matrix $[c_{ij}]$ of BTP, in each row i, find the minimum cost and the second minimum cost. Take the absolute difference of the two & note it down for the i^{th} row. Repeat this exercise with each column also.

Among all differences say α_i for the i^{th} row and β_j for each column, choose the $\max\{\alpha_i, \beta_j\} > 0$. Suppose this maxium happens at the k^{th} row. So, go in the k^{th} row and in the cell with least cost. Fill this cell x_{kr} by $\min\{a_k, b_r\}$ exhusting either row availability or column requirement. Cancel that row/column. Repeat these steps with the reduced cost matrix. Till all rows & columns availability & requirement fulfilled.

* At the end of each assignment of value, the a_i/b_r must be adjusted.

	D_1	D_2	D_3	D_4	a_i
O_1	$\boxed{10}$ 1	3	9	$\boxed{2}$ 1	12
O_2	6	$\boxed{8}$ 5	$\boxed{4}$ 10	$\boxed{6}$ 5	18
O_3	2	4	$\boxed{11}$ 3	6	11
b_j	10	8	15	8	41

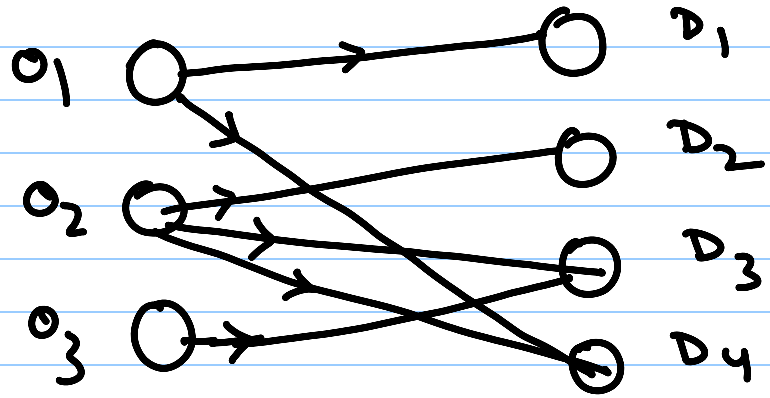
BFS : $x_{11} = 10$, $x_{14} = 2$

$x_{22} = 8$, $x_{23} = 4$, $x_{24} = 6$

$x_{33} = 11$

and $z = 155$.

Once BFS is with us we have to check its optimality.



Solve

$u_i + v_j = c_{ij}$

$\forall$ cells (i, j) which are basic.

$\therefore \quad u_1 + v_1 = 1, \quad u_1 + v_4 = 1$

$u_2 + v_2 = 5, \quad u_2 + v_3 = 10$

$u_2 + v_4 = 5, \quad u_3 + v_3 = 3$

Set $u_1 = 0 \Rightarrow v_1 = 1, \; v_4 = 1, \; u_2 = 4, \; v_2 = 1, \; v_3 = 6, \; u_3 = -3$

	D_1	D_2	D_3	D_4	a_i	u_i
O_1	$\boxed{10}$ 1	3 $\boxed{-2}$	9 $\boxed{-3}$	$\boxed{2}$ 1	12	0
O_2	6 $\boxed{-1}$	$\boxed{8}$ 5	$\boxed{4}$ 10	$\boxed{6}$ 5	18	4
O_3	2 $\boxed{-4}$	4 $\boxed{-6}$	$\boxed{11}$ 3	6 $\boxed{-8}$	11	-3
b_j	10	8	15	8		
v_j	1	1	6	1		

$z_{ij} - c_{ij} = u_i + v_j - c_{ij} \leq 0, \forall$ non basic cells (i, j)

$\therefore$ Current BFS is optimal. for BTP.

	D_1	D_2	D_3	D_4	D_5	a_i	u_i
O_1	10 (0)	$\boxed{4}$ 15	10 (3)	$\boxed{4}$ 12	20 (−5)	8	0
O_2	$\boxed{5}$ 5	10 (0)	$\boxed{2}$ 8	15 (−8)	$\boxed{0}$ 10	7	−5
O_3	15 (−10)	$\boxed{5}$ 10	12 (−4)	12 (−5)	$\boxed{5}$ 10	10	−5
b_j	5	9	2	4	5		
v_j	10	15	13	12	15		

$$z = 249$$

$z_{13} - c_{13} = u_1 + v_3 - c_{13} = 3 > 0 \Rightarrow x_{13}$ enters in basis.

	D_1	D_2	D_3	D_4	D_5
O_1	10	$4 \rfloor - \theta$ 15	$\theta \rfloor$ 10	$4 \rfloor$ 12	20
O_2	$5 \rfloor$ 5	10	$2 \rfloor - \theta$ 8	15	$0 \rfloor + \theta$ 10
O_3	15	$5 \rfloor + \theta$ 10	12	12	$5 \rfloor - \theta$ 10

assign $x_{13} = \theta$ and then alternatively adjust this θ. Choose

$\theta = \min\{4, 2, 5\} = 2$ (corresponding to $y_{ij} = +1$ in loop)

						u_i
	10	$\boxed{2}$ 15	$\boxed{2}$ 10	$\boxed{4}$ 12	20	0
	$\boxed{5}$ 5	10	8	15	$\boxed{2}$ 10	-5
	15	$\boxed{7}$ 10	12	12	$\boxed{3}$ 10	-5
v_j	10	15	10	12	15	

$z = 243$

Compute $u_i + v_j - c_{ij}$. Here, $u_i + v_j - c_{ij} \le 0 \; \forall$ non-basic cells (i, j). $\therefore$ Current BFS is optimal and optimal cost is 243.

Unbalanced TP : (P) is called unbalanced TP if

(1) $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$ or

(2) $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$

In case (1): supply is more than demand. We create a dummy destination D_{n+1} with requirement $b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$ and associate cost

$$c_{i\,n+1} = \min\{0, c_{i1}, c_{i2}, \dots, c_{in}\}, \quad i=1,2,\dots,m$$

In case (2) when supply is less than the demand, we create a dummy origin O_{m+1} with availability $a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i > 0$

The costs in this O_{m+1} rows are

$$c_{m+1\,j} = \min\{0, c_{1j}, c_{2j}, \dots, c_{mj}\}, \; j = 1, \dots, n.$$

Then solve the resultant BTP.

eg.

	D_1	D_2	D_3	D_4	a_i
O_1	1	3	0	1	20
O_2	-6	5	10	-5	34
O_3	2	-4	3	0	11
b_j	10	8	15	8	

$\sum b_j = 41$

$\sum a_i = 65$

	D_1	D_2	D_3	D_4	D_5	a_i
O_1	1	3	[15] 0	1	[5] 0	20
O_2	[10] −6	5	10	[8] −5	[16] −6	34
O_3	2	[8] −4	3	0	[3] −4	11
b_j	10	8	15	8	24	65

optimal soln of BTP is

$x_{13} = 20$, $x_{21} = 26$, $x_{24} = 8$, $x_{32} = 11$

& optimal $z = -240$

Assignment Problem

\# of jobs $= n$

\# of men $= n$

→ Each job is to be assigned to one person & one person can be assigned one job only. Jobs can not be partitioned and each person is capable of doing any job assigned to him/her.

We are looking at a bijection in complete bipartite graph with a total of n-nodes in each partition of V.

There are $n!$ such bijection maps b/w O_i and D_j $i = 1, \ldots, n$, $j = 1, \ldots, n$, out of which we have to choose the one with the least cost of getting all jobs done. Mathematically, we want to find x_{ij} (which job is assigned to which machine)

$$x_{ij} = \begin{cases} 1 & \text{if } i^{th} \text{ job is assign to } j^{th} \text{ machine} \\ 0 & \text{if } i^{th} \text{ job is not assign to } j^{th} \text{ machine} \end{cases}$$

(AP) $\quad \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$ subject to $\sum_{i=1}^{n} x_{ij} = 1 \;\; \forall j$, $\sum_{j=1}^{n} x_{ij} = 1 \;\; \forall i$

$x_{ij} \in \{0, 1\}$.

We can see the problem (AP) as a special case of BTP with all a_i's $=1$ and all b_j's $=1$.

\# of basic variables $= n + n - 1 = 2n-1$

\# of nonbasic variables $= (n-1)^2 = n^2 - (2n-1)$

Out of n^2 variables x_{ij} only $2n-1$ are basic variables And out of $2n-1$ such variables only n variables can take value 1 in basic feasible solns and remaining $n-1$ other variables taking value 0. This $n-1$ could be a large number $\therefore$ there is a large no. of degenerate basic variables (or any BFS see a large degeneracy). For this we did not apply the tool of solving BTP to solve AP problem.

Before we move on, the (AP) problem involves binary variables as all $x_{ij} \in \{0, 1\}$. So, one can think of applying the B & B method. But truely we would not be requiring it because of the special structure inherited by the matrix A in (AP).

Doubly stochastic matrix: An $n \times n$ matrix $M = [m_{ij}]$ is called a doubly stochastic matrix if all its entries are non-negative and $\sum_{i=1}^{n} m_{ij} = 1, \ \forall j$, $\sum_{j=1}^{n} m_{ij} = 1, \ \forall i$

eg $M = \begin{pmatrix} 2/3 & 2/3 \\ 2/3 & 2/3 \end{pmatrix}$, $M = \begin{pmatrix} 1/5 & 2/5 & 2/5 \\ 1/5 & 3/5 & 1/5 \\ 3/5 & 0 & 2/5 \end{pmatrix}$

Let $\mathcal{D}$ be the set of all doubly stochastic matrices.

→ $\mathcal{D}$ is a convex set and its extreme points are the matrices $X = [x_{ij}] \in \mathcal{D}$, with each $x_{ij} \in \{0,1\}$.

like

$$\mathcal{D} = \left\{ \begin{bmatrix} m_{11} & 1-m_{11} \\ 1-m_{11} & m_{11} \end{bmatrix} : m_{11} \in [0,1] \right\}$$

Extreme points of $\mathcal{D}$ are

$$\left\{ \begin{bmatrix} m_{11} & 1-m_{11} \\ 1-m_{11} & m_{11} \end{bmatrix} : m_{11} \in \{0,1\} \right\}$$

$$= \left\{ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\}$$

or

$$\mathcal{D} = \left\{ \begin{bmatrix} m_{11} & m_{12} & 1-m_{11}-m_{12} \\ m_{21} & m_{22} & 1-m_{21}-m_{22} \\ 1-m_{11}-m_{21} & 1-m_{12}-m_{22} & -1+m_{11}+m_{12}+m_{21}+m_{22} \end{bmatrix} : m_{ij} \in [0,1] \right\}$$

Extreme points of $\mathcal{D}$ are

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad - 6 \text{ in number}$$

In (AP), we ignore the binary condition on x_{ij} & solve it like LPP. An optimal soln of (AP) will always exist and naturally at an extreme point of $\mathcal{D}$, where $x_{ij} \in \{0, 1\}$, $\forall\ (i, j)$ holds.

Let (AP) be an assignment problem with cost matrix $C = [c_{ij}]$. Let transformation happens in rows/columns of C to get $\hat{C} = [\hat{c}_{ij}]$, where $\hat{c}_{ij} = c_{ij} + \alpha_i + \beta_j$, $\alpha_i \in \mathbb{R}$, $\beta_j \in \mathbb{R}$. Let we form an AP with matrix $\hat{C}$ and call it as $(\hat{AP})$.

Result: The two assignment problems (AP) and $(\hat{AP})$ have the same optimal solns (or assignment).

Proof:- Note the feasible sets of both (AP) and $(\hat{AP})$ are the same.

Let $[x_{ij}]$ be any feasible soln of (AP) or $(\hat{AP})$

Note:

$$\hat{z} = \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{c}_{ij} x_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} (c_{ij} + \alpha_i + \beta_j) x_{ij}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{n} x_{ij} \right) + \sum_{j=1}^{n} \beta_j \left(\sum_{i=1}^{n} x_{ij} \right)$$

$$= z + \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{n} \beta_j \quad \text{(using feasibility of } [x_{ij}])$$

$$= z + \text{constant.}$$

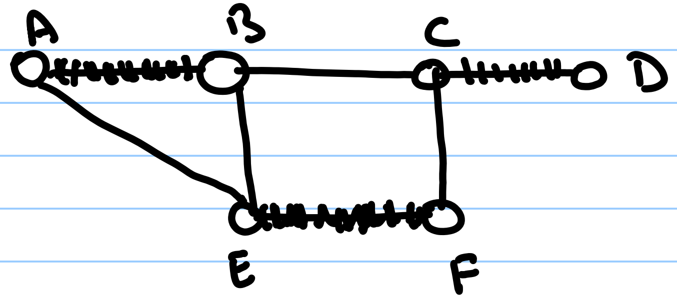
$\therefore$ optimal soln of (AP) and $(\hat{AP})$ would be the same though the optimal values of (AP) and $(\hat{AP})$ would differ by some constant.

Hungarian Method: It is a combinatorial algo which solves the assignment problem in a polynomial time. It was developed by Harold Kuhn. The name Hungarian was given by Kuhn to acknowledge the works of two Hungarian researchers Dénes Köing and Jenõ Egerváry, who contributed immensely to the theory of AP from graph theory perspective.

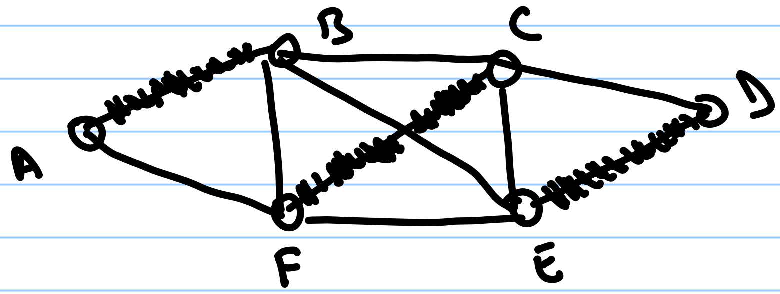
König Egerváry Theorem: In a matrix of 0's and 1's, the maximum number of zeros such that no two of them are in same line is the minimum number of lines which collectively contains all 0's. Here, by lines means row or column.

→ In any bipartite graph, the maximum size of a matching equals the minimum size of cover

→ Given a graph $G = (V, E)$, a matching M in G is a set of pairwise non-adjacent edges.



matching in G.



matching in G

In (AP), we are trying to find maximum matching of least cost.

Algorithm :

1) choose the minimum cost in each row and subtract it from each element of each cost of this row.

2) Repeat 1) for each column.

After these two steps each row and each column will have atleast one zero. let this transformed cost matrix be $\hat{C}$.

3) Find maximum no. of independent zeros in $\hat{C}$. Let these be r

Naturally, $r \leq n$ and this r is called index of $\hat{C}$.

(Any two zeros of $\hat{C}$ are independent if they are not in the same row or in the same column; and more than two zeros in $\hat{C}$ are called independent if every two of them are independent)

4) If $r = n$ then stop and we have optimal assignment by assigning $x_{ij} = 1$ $\forall$ cells (i, j) corresponding to independent zeros. Else go to next step

5) Find the minimum number of lines to cover all zeros of $\hat{C}$. By König Egeivary thm this line no. is equal to maximum no. of independent zeros in $\hat{C}$.
Choose the smallest elt among the uncovered zeros, say γ.

Subtract this r from all uncovered elements, add r to all elements which are at the intersection of lines drawn, and leave all other elements unchanged to get a new $\hat{C}$.

Repeat the steps with $\hat{\hat{C}}$.

Note: we have develop algo with $c_{ij} \geq 0 \ \forall (i,j)$ in C

In case entries in C are $-$ve then we can choose some $\xi > 0$ and add it to all elements of C to get C_ξ with all entries ≥ 0. This will not alter the optimal soln by lemma though the optimal value with C and C_ξ will differ by a constant ξ

Eg:

$$\text{Jobs}\quad \overset{\text{Machines}}{\begin{pmatrix} 7 & 5 & 9 & 8 & 11 \\ 9 & 12 & 7 & 11 & 10 \\ 8 & 5 & 4 & 6 & 9 \\ 7 & 3 & 6 & 9 & 5 \\ 4 & 6 & 7 & 5 & 11 \end{pmatrix}} = C$$

$$\hat{C} : \begin{array}{ccccc|c} 2 & 0 & 4 & 2 & 4 & \\ 2 & 5 & 0 & 3 & 1 & \checkmark \\ 4 & 1 & 0 & 1 & 3 & \checkmark \\ 4 & 0 & 3 & 5 & 0 & \\ 0 & 2 & 3 & 0 & 5 & \\ & & \checkmark & & & \end{array}$$

$r = 4$

$n = 5$

min elt = 1 (uncovered)

2	0	5	2	4
1	4	0	2	0
3	0	0	0	2
4	0	4	5	0
0	2	4	0	5

$r = n = 5$

Optimal assignment

$J_1 \to M_2$; $J_2 \to M_3$; $J_3 \to M_4$, $J_4 \to M_5$

$J_5 \to M_1$ and optimal cost = 27.

"To find the minimum no. of lines to cover zeros"

(1) Mark all rows for which the assignment is not made

(2) Mark all columns not already marked with assigned zeros in the marked rows.

(3) Mark all rows not already marked which have assigned zeros in the marked columns

Draw cover / grid through unmarked rows and marked columns to get r.

Machines

J	90	75	75	80
O	45	85	55	65
B	99	95	90	105
S	75	100	95	85

$C :=$

$\hat{C}$

15	0	0	0
0	40	10	15
9	5	0	10
0	25	20	5

$\hat{C}:$

15	0	0	0
0	40	10	15
9	5	0	10
0	25	20	5

$r = 3$, $n = 4$, min uncovered $= 5$

Transformed cost

20	0	0	0
0	35	5	10
14	5	0	10
0	20	15	0

20	0	0	0
0	35	5	10
14	5	0	10
0	20	15	0

$r = n = 4$

Optimality holds.

Optimal assignment

$J_1 \rightarrow M_2$

$J_2 \rightarrow M_1$

$J_3 \rightarrow M_3$

$J_4 \rightarrow M_4$

$Z = 75 + 45 + 90 + 85 = 295$

Dual of (AP)

(AP) $\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

s.to $\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \dots, n$

$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \dots, n$

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

(Dual) $\max \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$

s.to $u_i + v_j \leq c_{ij}$

$\forall \; i = 1, \dots, n$

$j = 1, \dots, n$

$u_i, v_j \in \mathbb{R}, \forall (i, j)$

u_i and v_j are dual variables.

At optimality $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} = \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} = \sum_{i=1}^{n} u_i \left(\sum_{j=1}^{n} x_{ij} \right) + \sum_{j=1}^{n} v_j \left(\sum_{i=1}^{n} x_{ij} \right)$$

$$\Rightarrow \sum_{i=1}^{n} \sum_{j=1}^{n} (c_{ij} - u_i - v_j) x_{ij} = 0 .$$

Now, $c_{ij} - u_i - v_j \geq 0$, $x_{ij} \geq 0$ & sum $= 0$ means

$$\underbrace{(c_{ij} - u_i - v_j) x_{ij} = 0}_{\text{complimentarity condition.}} \quad \forall \ (i, j) \quad \text{—} \quad 1$$

Now in step 1 of the method we find the least elt rowwise that is precisely

Let $u_i = \min_{1 \le j \le n} c_{ij}$

And we get $C_1 = [c_{ij}^{(1)}]$, $c_{ij}^{(1)} = c_{ij} - u_i$

Similarly, in step 2, take

$$v_j = \min_{1 \le i \le n} c_{ij}^{(1)}$$

to get $\hat{C} = [\hat{c}_{ij}]$, $\hat{c}_{ij} = c_{ij}^{(1)} - v_j = c_{ij} - u_i - v_j \geq 0$

$\therefore$ By end of step 2, we have the dual feasible soln (u_i, v_j) in hand. When we have $k < n$, the steps update the dual feasible soln to say $(u_i^{(1)}, v_j^{(1)})$. we continue this updation to get $(u_i^{(k)}, v_j^{(k)})$ dual feasible at which the CS condition 1 holds. Once C-S holds, we get the optimal soln of (AP), and its dual.

Updating dual variables :

let r be the least cost among all uncovered elements (when $r < n$).

$$u_i^{(1)} = \begin{cases} r & \text{if } i^{th} \text{ row is uncovered (or marked row)} \\ 0 & \text{otherwise} \end{cases}$$

$$v_j^{(1)} = \begin{cases} -r & \text{if } j^{th} \text{ column is covered (marked column)} \\ 0 & \text{otherwise} \end{cases}$$

Remember the cover drawn is of marked column and unmarked rows, and $\hat{\hat{C}}_{ij} = \hat{c}_{ij} - u_i^{(1)} - v_j^{(1)}$

→ look at cell (i,j) which is uncovered in $\hat{C}$

$\Rightarrow$ it is in marked row and unmarked column so $u_i^{(1)} = r$, $v_j^{(1)} = 0 \Rightarrow \hat{\hat{c}}_{ij} = \hat{c}_{ij} - u_i^{(1)} - v_j^{(1)} = \hat{c}_{ij} - r$

→ look at cell (i,j) which is at the intersection of row and column in cover.

$\Rightarrow$ (i, j) is in the unmarked row & marked column

$\Rightarrow$ $u_i^{(1)} = 0$, $v_j^{(1)} = -r$

$\therefore$ $\hat{\hat{c}}_{ij} = \hat{c}_{ij} - u_i^{(1)} - v_j^{(1)} = \hat{c}_{ij} + r$.

→ Let (i, j) be the cell on cover but not at intersection.

$\Rightarrow$ (i, j) is in unmarked row & unmarked column

$\Rightarrow$ $u_i^{(1)} = 0 = v_j^{(1)}$

$\therefore$ $\hat{\hat{c}}_{ij} = \hat{c}_{ij}$ (no change)

Convergence of the method :-

$C = [c_{ij}]$

$\hat{C} = [\hat{c}_{ij}]$, $\hat{c}_{ij} = c_{ij} - u_i - v_j \geq 0$

After $l < n$ step, we get $u_i^{(1)}$, $v_j^{(1)}$ and $\hat{\hat{C}} = [\hat{\hat{c}}_{ij}]$.

$$\hat{\hat{c}}_{ij} = \hat{c}_{ij} - u_i^{(1)} - v_j^{(1)} \geq 0$$

Now, $\hat{C} - C^{(1)}$ is a matrix entries $u_i^{(1)} + v_j^{(1)}$

$$= \begin{cases} r & \text{row } i \text{ marked \& col. } j \text{ is unmarked} \\ -r & \text{row } i \text{ unmarked \& col. } j \text{ is marked} \\ 0 & \text{otherwise} \end{cases}$$

$$\Rightarrow \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{c}_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{(1)} = \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$$

$$= p(n-q)r - q(n-p)r$$

$$= n(p-q)r$$

$p :=$ # of marked rows

$q :=$ # of unmarked columns

Now, $r = (n-p)+q \Rightarrow p-q = n-r.$

$$\therefore \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{c}_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{(1)} = n(n-r)r \geq 0 \text{ \& integer}$$

we can take to be integer simply $\because$ we can assume all cost entries c_{ij} are integers & non-negative. (we can use LCM of fractional quantities and multiply all entries by this LCM to get integer entries in C).

$\therefore$ when we move from one iteration to the next in the method the difference of sum of all costs is non-negative integers (countable), and as $r \to n$, the difference $\to 0$. (or convergence)

AP in maximization form : Let $c_{ij} \geq 0 \ \forall (i,j)$

We can find the $\max\limits_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \{c_{ij}\} = \theta$ (say). Transform

the C matrix to $\Theta - C$, where $\Theta = [\theta]_{n \times n}$.

Now, with the new matrix $\Theta - C$, the AP is in the minimization form to find the optimal assignment.

AP with negative cost

If any of the $c_{ij} < 0$ then choose a sufficiently large

+ve number say $\eta > 0$ and add this η to every element or $C + \underline{\eta}$, where $\underline{\eta} = [\eta]_{n \times n}$; and solve the new cost matrix AP to get its optimal assignment. This assignment is optimal for the original problem also.

<u>AP with some additional conditions:</u>

There is no unique way; but one has to logically see the information and make appropriate modifications in C.

For eg: If we say job J_i has to be done by machine M_k then allocate $x_{ik} = 1$ and cross off the ith job row and the k^{th} machine column, and solve the reduced cost matrix AP for other Jobs & Machines optimal assignment.

Similarly, if we say that machine M_i can not do job J_i then set $c_{ik} = M >> 0$ (almost $+\infty$ for minimization AP), to ensure $x_{ik} = 0$ in the final assignment.

Unbalanced AP

When no. of jobs $\neq$ no. of machines, then the AP is called the unbalanced AP. We can have two cases -

(i) no. of jobs > no. of machines

(ii) no. of jobs < no. of machines.

In case (i), if no additional information is available, then some of the jobs will be left undone. In case (ii), in the absence of any information, some machines will be idle.

To solve (i) or (ii) we convert the UAP to BAP by adding fictitious machines or fictitious jobs respectively with corresponding costs zero.

Any allocation in the dummy row/ column indicate the undone job / idle machine

	P_1	P_2	P_3	P_4	P_5
J_1	8	12	18	16	18
J_2	26	18	19	24	12
J_3	17	14	18	10	8
J_4	11	26	15	13	20

UAP

$\rightarrow$

	P_1	P_2	P_3	P_4	P_5
J_1	8	12	18	16	18
J_2	26	18	19	24	12
J_3	17	14	18	10	8
J_4	11	26	15	13	20
J_5	0	0	0	0	0

BAP

	P_1	P_2	P_3	P_4	P_5
J_1	0	4	10	8	10
J_2	14	6	7	12	0
J_3	9	6	10	2	0
J_4	0	15	4	2	9
J_5	0	0	0	0	0

$r = 3 < n = 5$

min unmarked $= 2$

J_1	0	2	8	6	10
J_2	14	4	5	10	0
J_3	9	4	8	0	0
J_4	0	13	2	0	9
J_5	2	0	0	0	2

$r = 4 < n = 5$

min unmarked $= 2$

	P_1	P_2	P_3	P_4	P_5
J_1	0	0	6	6	10
J_2	14	2	3	10	0
J_3	9	2	6	0	0
J_4	0	11	0	0	9
J_5	4	0	0	2	4

Tie case

$J_1 \rightarrow P_1$

$J_2 \rightarrow P_5$ Optimal cost = 45

$J_3 \rightarrow P_4$

$J_4 \rightarrow P_3$ and P_2 remains idle.

In an UAP, some additional information could be provided. For instance, P_1 must do atleast two jobs or P_3 must do two jobs or all jobs must be done, or P_2 must do a job or P_2, P_3, P_4 must be doing 4 jobs at totals. One has to suitably create the BAP with cost entries and then solve and interpret the optimal assignment for the original problem.

We are here presenting only one such case.

	J_1	J_2	J_3	J_4	J_5
P_1	4	5	2	3	3
P_2	5	4	1	6	2
P_3	3	2	1	4	5

→ P_1 must do exactly 2 jobs.

→ All jobs must be done.

	J_1	J_2	J_3	J_4	J_5
P_1	4	5	2	3	3
P_1	4	5	2	3	3
$\min\{P_2, P_3\}$	3	2	1	4	2
$\min\{P_2, P_3\}$	3	2	1	4	2
$\min\{P_2, P_3\}$	3	2	1	4	2

J_1	J_2	J_3	J_4	J_5
0	2	0	0	0
0	2	0	0	0
0	0	0	2	0
0	0	0	2	0
0	0	0	2	0

There are several alternative Solns to break tie

P_1 will do J_1 and J_4
P_3 will do J_2
P_3 | P_2 any one can do job J_3
P_2 will do J_5

This is not the unique soln.

Optimal cost = 12.