Notes for OR

1 Linear Programming

1.1 Some formulations

P1 To obtain your recommended daily allowances of Vitamins A, C, and K, you decide to eat apricots, bananas and cucumbers. The percentage of the recommended daily allowance of each vitamin contained in a serving of a given food, along with the cost of one serving of each food is given in the table below.

	A	C	K	cost
apricot	60	26	6	\$ 1.53
banana	3	33	1	\$ 0.37
cucumber	2	7	12	\$ 0.18

$$\begin{aligned} \text{Minimise } & 1.53x + 0.37y + 0.18z \\ \text{Subject to } & 60x + 3y + 2z \geq 100. \\ & 26x + 33y + 7z \geq 100. \\ & 6x + y + 12z \geq 100. \\ & x, y, z \geq 0. \end{aligned}$$

Solution: $x = 1.39728..., y = 0.315510..., z = 7.60840....$ Cost is 3.62409....

More generally, if there are n foods and m nutrients and each unit of food j costs c_j and contains $a_{i,j}$ units of nutrient i and we require r_i nutrients per day, then we need to solve

$$\begin{aligned} \text{Minimise } & \sum_{j=1}^{n} c_j x_j \\ \text{Subject to } & \sum_{j=1}^{n} a_{i,j} x_j \geq r_i, \quad i = 1, 2, \ldots, m. \\ & x_j \geq 0, \quad j = 1, 2, \ldots, n. \end{aligned}$$

P2 A company makes two products (say, P and Q) using two machines (say, A and B). Each unit of P that is produced requires 50 minutes processing time on machine A and 30 minutes processing time on machine B. Each unit of Q that is produced requires 24 minutes processing time on machine A and 33 minutes processing time on machine B. Machine A is going to be available for 40 hours and machine B is available for 35 hours. The profit per unit of P is \$25 and the profit per unit of Q is \$30. Company policy is to

determine the production quantity of each product in such a way as to maximize the total profit given that the available resources should not be exceeded.

$$\begin{aligned} \text{Maximise } & 25x + 30y \\ \text{Subject to } & 50x + 24y \leq 2400. \\ & 30x + 33y \leq 2100. \\ & x, y \geq 0. \end{aligned}$$

P3 An operations manager is trying to determine a production plan for the next week. There are three products (say, P, Q, and R) to produce using four machines (say, A and B, C, and D). Each of the four machines performs a unique process. There is one machine of each type, and each machine is available for 2400 minutes per week.

Problem Data

Machine	Product P	Product Q	Product R	Availability
Processing time per unit on A	20	10	10	2400
Processing time per unit on B	12	28	16	2400
Processing time per unit on C	15	6	16	2400
Processing time per unit on D	10	15	0	2400
Profit per unit	45	60	50	
Maximum sales	100	40	60	

$$\begin{aligned} \text{Maximise } & 45p + 60q + 50r \\ \text{Subject to } & 20p + 10q + 10r \leq 2400. \\ & 12p + 28q + 16r \leq 2400. \\ & 15p + 6q + 16r \leq 2400. \\ & 10p + 15q + 0r \leq 2400. \\ & 0 \leq p \leq 100,\ 0 \leq q \leq 40,\ 0 \leq r \leq 60. \end{aligned}$$

P4 Suppose you run an ice cream factory, and you anticipate monthly demand (in tons) for the next n periods to be $d_i, i = 1, 2, \ldots, n$. Suppose it costs c per ton to change production from one month to the next, and s per ton to store ice cream for a month. What is the minimum cost production schedule that meets demand?

Suppose that we produce x_i tons of ice-cream in period i and have y_i tons in storage at the beginning of period $i = 0, 1, \ldots, n+1$. Let $x_0 = x_{n+1} = y_1 = 0$.

$$\begin{aligned} \text{Minimise } & \sum_{i=1}^{n} (c|x_i - x_{i-1}| + sy_i). \\ \text{Subject to } & x_i + y_i \geq d_i, i = 1, 2, \ldots, n. \\ & y_{i+1} = x_i + y_i - d_i, i = 1, 2, \ldots, n-1. \\ & x_i, y_i \geq 0, i = 1, 2, \ldots, n. \end{aligned}$$

The objective function is non-linear. Introduce another variable z_i. Replace the above by

$$\begin{aligned}
\text{Minimise } & \sum_{i=1}^{n}(cz_i + sy_i).\\
\text{Subject to } & x_i + y_i \geq d_i, i = 1, 2, \ldots, n.\\
& y_{i+1} = x_i + y_i - d_i, i = 1, 2, \ldots, n-1.\\
& z_i \geq x_i - x_{i-1}, i = 1, 2, \ldots, n. \qquad (1)\\
& z_i \geq x_{i-1} - x_i, i = 1, 2, \ldots, n. \qquad (2)\\
& x_i, y_i \geq 0, i = 1, 2, \ldots, n.
\end{aligned}$$

In an optimum solution, z_i will be as small as possible, given that it has to satisfy (1),(2), giving $z_i = |x_i - x_{i-1}|$.

P5 The problem is to minimise $f(x) = \max\{a_i x + b_i : i = 1, 2, \ldots, n\}$.

$$\begin{aligned}
\text{Minimise } & z.\\
\text{Subject to } & z \geq a_i x + b_i, i = 1, 2, \ldots, n.
\end{aligned}$$

P6 A company is planning its investment strategy over the next n periods. It has m choices of investment. If investment i is operated at level y in period t then it generates $r_{i,t}y$ dollars and needs $\rho_{i,j,t}y$ units of resource j. There will be $a_{j,t}$ units of resource j available in total during period t. The cost of resource $j, j = 1, 2, \ldots, r$ in period t is $b_{j,t}$ dollars per unit. The company starts with A dollars to spend. Money generated in periods $1, 2, \ldots, t$ can be used in period $t+1$, but must be available at the start of the period. The company wishes to maximise the total amount it has in hand at the end of period n.

$$\begin{aligned}
\text{Maximise } & \sum_{t=1}^{n}\sum_{i=1}^{m}\left(r_{i,t} - \sum_{j=1}^{r}\rho_{i,j,t}b_{j,t}\right)y_{i,t}\\
\text{Subject to } & \sum_{i=1}^{m}\rho_{i,j,t}y_{i,t} \leq a_{j,t} \text{ for all } j, t.\\
& A + \sum_{\tau=1}^{t-1}\sum_{i=1}^{m}\left(r_{i,\tau} - \sum_{j=1}^{r}\rho_{i,j,\tau}b_{j,\tau}\right)y_{i,\tau} \geq \sum_{i=1}^{m}\sum_{j=1}^{r}\rho_{i,j,t}b_{j,t}y_{i,t}, \quad t = 1, 2, \ldots, n\\
& y_{i,t} \geq 0 \text{ for all } i, t.
\end{aligned}$$

Standard Form $\mathbf{A}$ is an $m \times n$ matrix of full row rank. The columns of $\mathbf{A}$ are $\mathbf{a}_j, j = 1, 2, \ldots, n$.

$$\begin{aligned}
\text{Maximise } & \mathbf{c}^T\mathbf{x} \qquad (3)\\
\text{Subject to } & \mathbf{A}\mathbf{x} = \mathbf{b}\\
& \mathbf{x} \geq 0.
\end{aligned}$$

Transforming to standard form:

$$\begin{aligned}
\mathbf{a}^T\mathbf{x} \leq b &\longrightarrow \mathbf{a}^T\mathbf{x} + s = b.\\
\mathbf{a}^T\mathbf{x} \geq b &\longrightarrow \mathbf{a}^T\mathbf{x} - s = b.\\
x_j \leq 0 &\quad \text{Replace } x_j \text{ by } -x'_j,\ x'_j \geq 0.\\
x_j \text{ free} &\quad \text{Replace } x_j \text{ by } x'_j - x''_j,\ x'_j, x''_j \geq 0.\\
\text{Minimise } \mathbf{c}^T\mathbf{x} &\quad \text{Maximise } (-\mathbf{c})^T\mathbf{x}.
\end{aligned}$$

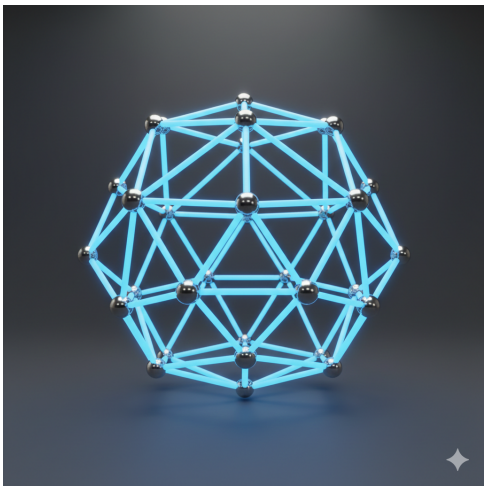
Linear programs can be

1. Solvable: e.g. maximise x: subject to $0 \leq x \leq 1$.
2. Infeasible: e.g. maximise x: subject to $x \leq 1, x \geq 2$.
3. Unbounded: maximise x: subject to $x \geq 0$.

The *feasible region* is the set $\{\mathbf{x} \in \mathbb{R}^n : \ \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq 0\}$. The feasible region is a *polyhedron*:



(It isn't necessarily bounded in size.)

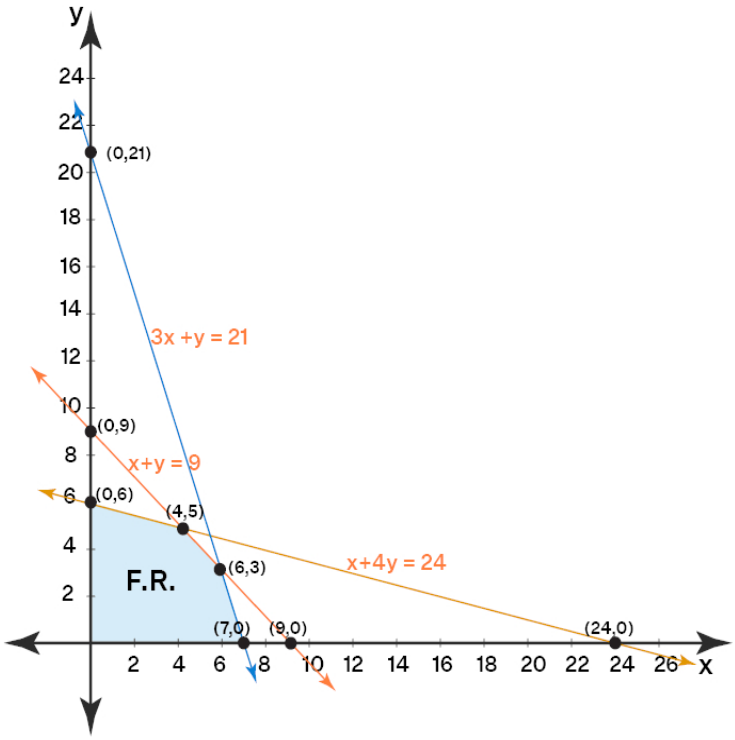


The "corners" are called *extreme points*. If there is an optimal solution, then there is an extreme point with optimal value.

The simplex algorithm starts at an extreme point and moves along the edges from extreme point to neighboring extreme point until it finds the optimum.

All we need now is (i) an algebraic equivalent of vertex and (ii) a procedure for moving from vertex to neighboring vertex.

The algebraic equivalent is *Basic Feasible Solution* and the procedure (ii) is a *pivot*.



The feasible region F here is

$$\begin{aligned} x + 4y &\leq 24. \\ 3x + y &\leq 21. \\ x + y &\leq 9. \\ x, y &\geq 0. \end{aligned}$$

The extreme points are $(0,0), (7,0), (6,3), (4,5), (0,6)$.

Now add *slack* variables.

$$\begin{aligned} x + 4y \quad +s_1 \qquad\qquad &= 24. \\ 3x + y \qquad + s_2 \qquad &= 21. \\ x + y \qquad\qquad +s_3 \quad &= 9. \\ x, y, s_1, s_2, s_3 &\geq 0. \end{aligned}$$

The extreme points are
$(0,0,24,21,9), (7,0,17,0,2), (6,3,6,0,0),$
$(4,5,0,4,0), (0,6,0,15,3)$.

Suppose we want to maximise $z = 2x + 3y$ over F.

Start at $(0,0)$. Choose an edge to move along that increases z, e.g. move to $(7,0)$, then to $(6,3)$, then to $(4,5)$. Now a move along an adjacent edge decreases z – we found the maximum.

Now we keep track of which variables are *basic* or *non-basic*. Initial basis $\{s_1, s_2, s_3\}$, then $\{x, s_1, s_3\}$, then $\{x, y, s_1\}$ and finally $\{x, y, s_2\}$. Put the non-basic variables equal to 0 and then solve the equations for the basic variables.

A Basic Feasible Solution (BFS) is one with $n - m$ zero *non-basic* variables $x_j, j \in J \subseteq [n] \setminus I$ such that the remaining m *basic* variables $x_i, i \in I = [n] \setminus J$ are (i) the unique solution to the remaining equations and (ii) they have non-negative values. BFS's defined by I_1, I_2 are neighbors if $|I_1 \setminus I_2| = |I_2 \setminus I_1| = 1$. The simplex algorithm proceeds as follows: if I defines the current basis, see if there exists $k \in I, \ell \in [n] \setminus I$ such that $I' = (I \cup \{\ell\}) \setminus \{k\}$ is an improvement. If there is, replace I by I'. Otherwise the problem is solved. All we need now is machinery to implement this efficiently.

Example 1:

$$\begin{aligned} \text{Maximise } & 2x_1 + 3x_2 \\ \text{Subject to } & x_1 + 4x_2 \leq 24. \\ & 3x_1 + x_2 \leq 21. \\ & x_1 + x_2 \leq 9. \\ & x_1, x_2 \geq 0. \end{aligned}$$

$B.V.$	x_1	x_2	x_3	x_4	x_5	RHS
x_0	-2	-3				0
x_3	1	4	1			24
x_4	**3**	1		1		21
x_5	1	1			1	9
x_0		$-7/3$		$2/3$		14
x_3		$11/3$	1	$-1/3$		17
x_1	1	$1/3$		$1/3$		7
x_5		**2/3**		$-1/3$	1	2
x_0				$-1/2$	$7/2$	21
x_3			1	**3/2**	$-11/2$	6
x_1	1			$1/2$	$-1/2$	6
x_2		1		$-1/2$	$3/2$	3
x_0			$1/3$		$5/3$	23
x_4			$2/3$	1	$-11/3$	4
x_1	1		$-1/3$		$4/3$	4
x_2		1	$1/3$		$-1/3$	5

Vertex to neighboring vertex becomes basic solution to neighboring basic solution.

Sequence of bases:

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

$$B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 3 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad B^{-1} = \begin{bmatrix} 1 & -1/3 & 0 \\ 0 & 1/3 & 0 \\ 0 & -1/3 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & 4 \\ 0 & 3 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad B^{-1} = \begin{bmatrix} 1 & 3/2 & -11/2 \\ 0 & 1/2 & -1/2 \\ 0 & -1/2 & 3/2 \end{bmatrix}$$

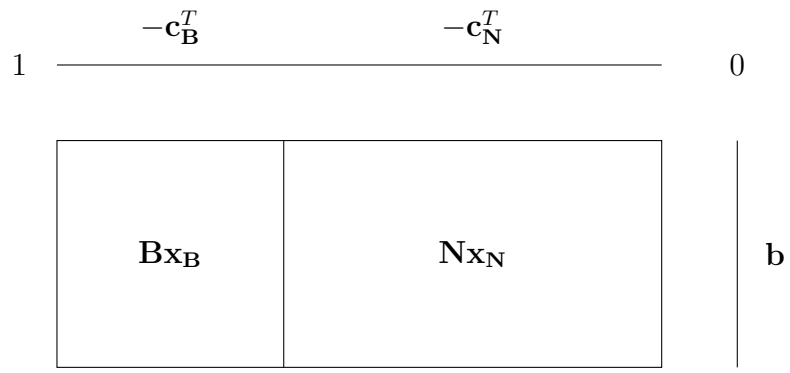
$$B = \begin{bmatrix} 0 & 1 & 4 \\ 1 & 3 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad B^{-1} = \begin{bmatrix} 2/3 & 1 & -11/3 \\ -1/3 & 0 & 4/3 \\ 1/3 & 0 & -1/3 \end{bmatrix}$$

Shadow prices: $c_{\mathbf{B}}^T \mathbf{B}^{-1} = [0, 2, 3] \begin{bmatrix} 2/3 & 1 & -11/3 \\ -1/3 & 0 & 4/3 \\ 1/3 & 0 & -1/3 \end{bmatrix} = [1/3, 0, 5/3].$

Basic Solutions Algebraically, the extreme points are the *Basic Feasible Solutions* to $\mathbf{Ax} = \mathbf{b}$. The simplex algorithm moves from one BFS (extreme point) to a neighbor by a *pivot*.

Suppose that $\mathbf{B}$ is an $m \times m$ non-singular sub-matrix of $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2\, \cdots \mathbf{a}_n]$.

$$x_0 - \mathbf{c}_\mathbf{B}^T\mathbf{x}_\mathbf{B} - \mathbf{c}_\mathbf{N}^T\mathbf{x}_\mathbf{N} = 0.$$
$$\mathbf{B}\mathbf{x}_\mathbf{B} + \mathbf{N}\mathbf{x}_\mathbf{N} = \mathbf{b}.$$

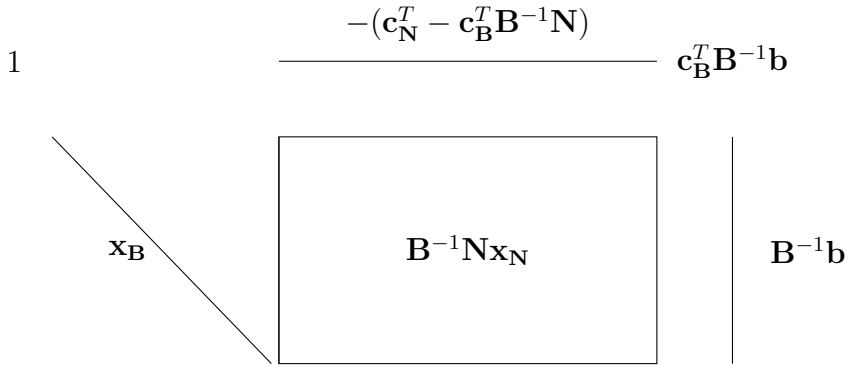


Multiply matrix $\mathbf{A}$ on the left by $\mathbf{B}^{-1}$ and substitute $\mathbf{x}_\mathbf{B} = \mathbf{B}^{-1}(\mathbf{b} - \mathbf{N}\mathbf{x}_\mathbf{N})$. To get an equivalent set of linear equations. Solutions to $\mathbf{Ax} = \mathbf{b}$ satisfy the following and vice-versa:

$$x_0 - (\mathbf{c}_\mathbf{N}^T - \mathbf{c}_\mathbf{B}^T\mathbf{B}^{-1}\mathbf{N})\mathbf{x}_\mathbf{N} = \mathbf{c}_\mathbf{B}^T\mathbf{B}^{-1}\mathbf{b}.$$
$$\mathbf{x}_\mathbf{B} + \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_\mathbf{N} = \mathbf{B}^{-1}\mathbf{b}.$$

The values $c_j - \mathbf{c}_\mathbf{B}^T\mathbf{B}^{-1}\mathbf{a}_j$ are called the *reduced costs*.

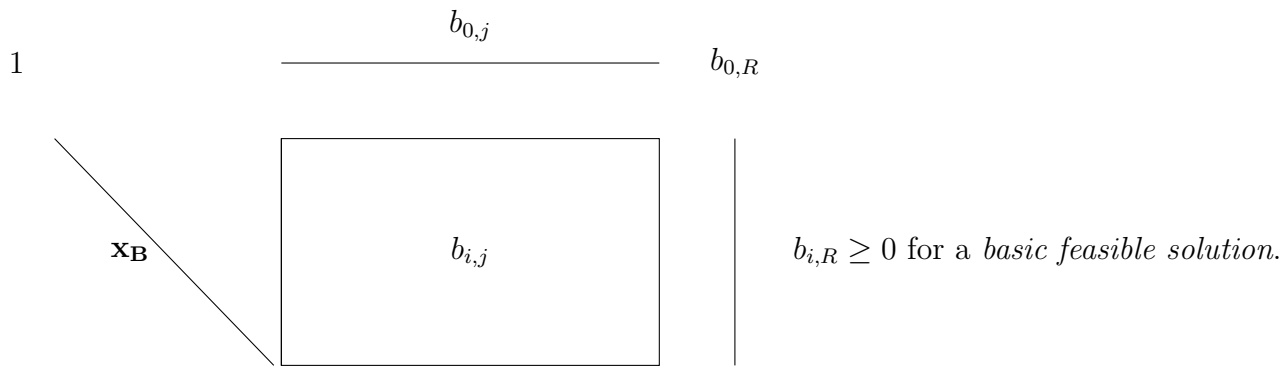
Simplex tableau



I denotes the index set of the *basic* columns $\mathbf{B}$ and J denotes the index set of the *non-basic* columns $\mathbf{N}$. Variables $x_i, i \in I$ are referred to as the *basic variables* and variables $x_j, j \in J$ are referred to as the *non-basic variables*. We index the rows by which basic variable they contain. The tableau represents the equations

$$x_i + \sum_{j \in J} b_{i,j}x_j = b_{i,R}, \qquad i \in I \cup \{0\}\,. \tag{4}$$

The associated *basic solution* is obtained by putting $\mathbf{x}_\mathbf{N} = 0$ and $\mathbf{x}_\mathbf{B} = \mathbf{B}^{-1}\mathbf{b}$. This is *feasible* if $\mathbf{B}^{-1}\mathbf{b} \geq 0$ and we will refer to such a solution as a *Basic Feasible Solution – BFS*.
Note that solutions to $\mathbf{Ax} = \mathbf{b}$ are obtained by giving values to $\mathbf{x}_\mathbf{N}$ and then putting $\mathbf{x}_\mathbf{B} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_\mathbf{N}$.



Optimality condition: if $b_{0,j} \geq 0$ for all $j \in J$ and $\mathbf{x}$ is a BFS then $\mathbf{x}$ maximises the objective function x_0. Indeed, $\mathbf{x}$ has objective value $b_{0,R}$ and any other feasible solution $\mathbf{y} \geq 0$ satisfies

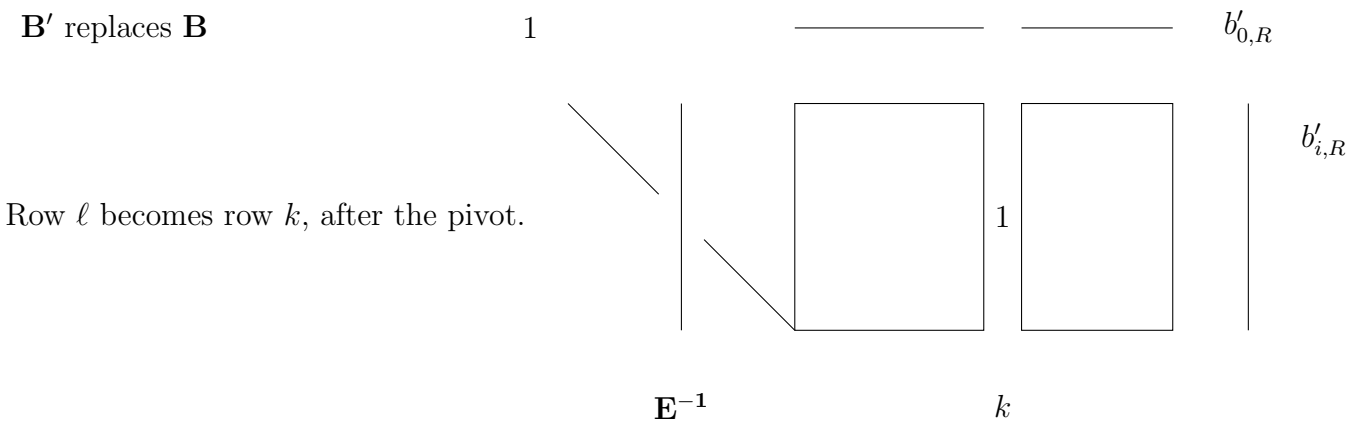
$$y_0 = b_{0,R} - \sum_{j \in J} b_{0,j} y_j \leq b_{0,R}.$$

The simplex algorithm starts with an initial BFS and does a sequence of pivots and stops when $b_{0,j} \geq 0, j \in J$. To find an initial BFS we use a 2-phase version of the simplex algorithm that is described below. (This is not a circular argument!)

Unboundedness: if at some stage there is a BFS and a non-basic variable x_ℓ such that (i) $b_{0,\ell} < 0$ and (ii) $b_{i,\ell} \leq 0$ for $i \in I$ then the problem is unbounded. Put $x_\ell = t$ and $x_j = 0, j \in J \setminus \{\ell\}$ and compute the values of the basic variables from (4). Then for large t

$$x_0 = \mathbf{c_B^T B^{-1} b} - t b_{0,\ell} \nearrow \infty \text{ with } t \text{ and } x_i = b_{i,R} - t b_{i,\ell} \geq 0.$$

Simplex pivot: We choose k, ℓ (row $\ell \in I$, column $k \in J$) where $b_{\ell,k} \neq 0$.
$I \leftarrow I + k - \ell$ and $J \leftarrow J + \ell - k$. Let $\mathbf{B}'$ be obtained from $\mathbf{B}$ by deleting column $\mathbf{a}_\ell$ and replacing it by $\mathbf{a}_k$.



Divide row ℓ by $b_{\ell,k}$ and then subtract $b_{i,k}$ times row ℓ from row $i \neq \ell$.

$$b'_{k,j} \leftarrow \frac{b_{\ell,j}}{b_{\ell,k}} \qquad \text{and} \qquad b'_{i,j} \leftarrow b_{i,j} - b_{i,k} \times \frac{b_{\ell,j}}{b_{\ell,k}} \text{ for } i \neq k, \qquad \text{Note that } b'_{i,k} = 0, i \neq k.$$

Remember: row operations do not change the set of solutions to a set of linear equations.

Then

$$\mathbf{B}' = \mathbf{B} + [0 \overset{\mathbf{a}_k}{|} 0] - [0 \overset{\mathbf{a}_\ell}{|} 0] \text{ and so } \mathbf{B}^{-1}\mathbf{B}' = \mathbf{I} + \left[0\ \mathbf{B}^{-1}\mathbf{a}_k\ 0\right] - [0\ \mathbf{e}_\ell\ 0]\,,$$

and

$$(\mathbf{B}')^{-1} = \mathbf{E^{-1}B^{-1}} \text{ where } \mathbf{E} = \mathbf{I} - [\mathbf{0}\ \mathbf{e}_\ell\ \mathbf{0}] + \left[\mathbf{0}\ \mathbf{B^{-1}a_k}\ \mathbf{0}\right].$$

Note that

$$\begin{bmatrix} 1 & 0 & 0 & a_1 & 0 & 0 & 0 \\ 0 & 1 & 0 & a_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & a_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_5 & 1 & 0 & 0 \\ 0 & 0 & 0 & a_6 & 0 & 1 & 0 \\ 0 & 0 & 0 & a_7 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -a_1/a_4 & 0 & 0 & 0 \\ 0 & 1 & 0 & -a_2/a_4 & 0 & 0 & 0 \\ 0 & 0 & 1 & -a_3/a_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/a_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & -a_5/a_4 & 1 & 0 & 0 \\ 0 & 0 & 0 & -a_6/a_4 & 0 & 1 & 0 \\ 0 & 0 & 0 & -a_7/a_4 & 0 & 0 & 1 \end{bmatrix}$$

From this we can see what replaces the identity in the columns associated with I.

Preservation of solution: It is important to realise that row operations do not change the set of solutions to $\mathbf{Ax} = \mathbf{b}$. So, after a pivot, the set of solutions to the new set of equations is the same as the previous set.

Choice of pivot: we choose one so that the new basic solution is feasible and has a larger objective value
Feasibility: $b'_{k,R} = b_{\ell,R}/b_{\ell,k}$ so we need $b_{\ell,k} > 0$, assuming that $b_{\ell,R} > 0$.
We need $b'_{i,R} = b_{i,R} - b_{i,k} \times \frac{b_{\ell,R}}{b_{\ell,k}} \geq 0$. This is automatically true if $b_{i,k} \leq 0$.
Otherwise we need $\frac{b_{\ell,R}}{b_{\ell,k}} \leq \frac{b_{i,R}}{b_{i,k}}$ – *ratio test.*

Objective improvement – maximization: $b'_{0,R} = b_{0,R} - b_{0,k} \times \frac{b_{\ell,R}}{b_{\ell,k}}$ and so we choose $b_{0,k} < 0$. Note that $-b_{0,k}$ is the *reduced cost.* We have $x_0 = b_{0,R} - \sum_{j \in J} b_{0,j} x_j$ for $\mathbf{Ax} = \mathbf{b}$.

Another perspective: current solution, $x_i = b_{i,R}, i \in I$ and $x_j = 0, j \in J$. Suppose we increase x_k, keeping $x_j = 0, j \in J \setminus k$ and satisfying (4). Then $x_i \leftarrow b_{i,R} - b_{i,k}x_k$ and so the maximum we can increase x_k to is $\min\{b_{i,R}/b_{i,k} : b_{i,k} > 0\}$. The solution we get is the same as the basic solution after the pivot. We now have $x_j = 0$ for $j \in (J \setminus \{k\}) \cup \{\ell\}$ and $\mathbf{a}_i, i \in (I \cup \{k\}) \setminus \{\ell\}$ forms our new basis.

Degeneracy: A BFS is degenerate if $b_{\ell,R} = 0$ for some $\ell \in I$. If you pivot on $b_{\ell,k}$ in such a row, then the RHS $\mathbf{b}_R$ does not change. After the pivot we still have $x_k = x_\ell = 0$ and all other variables have the same value. Only the status of x_k, x_ℓ has changed.

Finding a starting basis: 2-phase method.
We can assume that $\mathbf{b} \geq 0$ since we can multiply a row with $b_i < 0$ by -1. First solve

$$\begin{aligned} \text{Maximise} \quad & (-\mathbf{1})^{\mathbf{T}}\boldsymbol{\xi} \\ \text{Subject to} \quad & \mathbf{Ax} + \boldsymbol{\xi} = \mathbf{b} \\ & x_0 - \mathbf{c}^T\mathbf{x} = 0. \\ & \mathbf{x}, \boldsymbol{\xi} \geq 0. \end{aligned}$$

The starting basis has basic variables $\xi_1, \xi_2, \ldots, \xi_m$ and the original LP has feasible solutions iff the optimum above has $\boldsymbol{\xi} = 0$. Once we get to $\boldsymbol{\xi} = 0$, we can remove the columns corresponding to the *artificial variables* $\boldsymbol{\xi}$. More precisely, if ξ_t is non-basic then we just delete the ξ_t column. If ξ_t is basic then there are two possibilities: first assume that there is a non-zero in the ξ_t row that corresponds to a non-basic x_j. In this case we pivot on this non-zero. Because the current value of ξ_t is zero, the actual solution does not change and we swap an artificial basic variable for a regular variable and then we can remove the artificial variable ξ_t. If there are no non-zeros in the ξ_t row, then we can remove the ξ_t row, the corresponding row in $\mathbf{A}$ is linearly dependent on other rows.

The LP is infeasible, i.e. the feasible region is empty, if the minimum sum of artificials is strictly positive.
Example 2:

$$\begin{aligned} \text{Maximise } & 2x_1 + 3x_2 + 4x_3 \\ \text{Subject to } & x_1 + 4x_2 - 3x_3 = 7. \\ & 2x_1 + x_2 + 2x_3 \geq 8. \\ & x_1 + x_2 + x_3 \leq 9. \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

We add artificials ξ_1, ξ_2 to the first two constraints. Then Phase 1 becomes

$$\begin{aligned} \text{Minimise } & \zeta = \xi_1 + \xi_2 \\ \text{Subject to } & x_0 - 2x_1 - 3x_2 - 4x_3 = 0. \\ & x_1 + 4x_2 - 3x_3 + \xi_1 = 7. \\ & 2x_1 + x_2 + 2x_3 - s_1 + \xi_2 = 8. \\ & x_1 + x_2 + x_3 + s_2 = 9. \\ & x_1, x_2, x_3, s_1, s_2, \xi_1, \xi_2 \geq 0. \end{aligned}$$

Note that $\zeta = 15 - 3x_1 - 5x_2 + x_3 + s_1$.

$B.V.$	x_0	x_1	x_2	x_3	s_1	s_2	ξ_1	ξ_2	RHS	
ζ		3	5	-1	-1				15	
x_0	1	-2	-3	-4					0	
ξ_1		1	4	-3			1		7	
ξ_2		**2**	1	2	-1			1	8	
s_2		1	1	1		1			9	
ζ			7/2	-4	1/2			$-3/2$	3	
x_0	1		-2	-2	-1			1	8	
ξ_1			**7/2**	-4	1/2		1	$-1/2$	3	
x_1		1	1/2	1	$-1/2$			1/2	4	
s_2			1/2		1/2	1		$-1/2$	5	
ζ							-1	-1	0	
x_0	1			$-30/7$	$-5/7$		4/7	5/7	68/7	
x_2			1	$-8/7$	1/7		2/7	$-1/7$	6/7	$FEASIBLE$
x_1		1		**11/7**	$-4/7$		$-1/7$	4/7	25/7	
s_2				4/7	3/7	1	$-1/7$	$-3/7$	32/7	
x_0	1	30/11			$-25/11$		2/11	25/11	214/11	
x_2		8/11	1		$-3/11$		2/11	3/11	38/11	
x_3		7/11		1	$-4/11$		$-1/11$	4/11	25/11	
s_2		$-4/11$			**7/11**	1	$-1/11$	$-7/11$	36/11	
x_0	1	10/7				25/7	$-1/7$		218/7	$OPTIMAL$
x_2		4/7	1			3/7	1/7		34/7	*Ignore reduced*
x_3		3/7		1		4/7	$-1/7$		29/7	*Cost of artificial*
s_1		$-4/7$			1	11/7	$-1/7$	-1	36/7	*variable*

Optimal solution is: $x_1 = 0, x_2 = 34/7, x_3 = 29/7, x_0 = 218/7$.

Termination: If an LP problem is *non-degenerate* then for every vertex, we have $b_{i,R} > 0$ for all $i \in I$. This means that $b'_{0,R} = b_{0,R} - b_{0,k} \times \frac{b_{\ell,R}}{b_{\ell,k}} > b_{0,R}$ and so we never repeat a basis. As there are at most $\binom{n}{m}$ bases, we must eventually terminate.

If there are degenerate bases, then this can lead to the algorithm endlessly cycling through a fixed finite sequence of bases without actually changing the current feasible solution.

Bland's Rule: Choose the lowest indexed non-basic variable x_j for which $b_{0,j} < 0$ and then the lowest indexed basic variable x_i with $b_{i,j} > 0$ that minimises the ratio $b_{i,R}/b_{i,j}$ and pivot on (i, j).

If the Simplex Algorithm does not terminate then there must be a sub-sequence of bases $I_t = I_{t-1} + j_t - i_t, t = 1, 2, \ldots, p$ that repeats indefinitely from some point. Note that $b_{i_t,R} = 0, t = 1, 2, \ldots, p$ and $L = \{i_1, i_2, \ldots, i_p\} = \{j_1, j_2, \ldots, j_p\}$. And note that the actual BFS $\mathbf{x}$ does not change.

We reduce the tableaus by deleting rows and columns corresponding to variables not in L. Applying the Simplex algorithms to these tableaus will result in the same endless sequence of pivots. Let $t = \max L$ and let T_1 be a reduced tableau in this sequence where x_t is the variable that leaves the basis and let x_s be the variable that enters the basis. Next let T_2 be a reduced tableau in the sequence where x_t enters the basis. Now let $\mathbf{y}$ be the solution to $\mathbf{Ay} = \mathbf{b}$ obtained from $\mathbf{x}$ by changing x_s to -1 and keeping all other non-basic variables at 0. Then we have $y_i = b^1_{i,R} + b^1_{i,s} = b^1_{i,s}$ for $i \in L$, where the suffix $l = 1, 2$ refers to tableau T_l. Computing the objective value of $\mathbf{y}$ in T_1 and T_2 we get

$$b^1_{0,R} + b^1_{0,s} = b^2_{0,R} + b^2_{0,s} - \sum_{i \in L} b^2_{0,i} y_i = b^2_{0,R} + b^2_{0,s} - \sum_{i \in L} b^2_{0,i} b^1_{i,s}.$$

Now $b^1_{0,R} = b^2_{0,R}$ (no change in objective value in the sequence) and $b^1_{0,s} < 0$ (because x_s enters the basis) and $b^2_{0,s} \geq 0$ (because of Bland's rule). So, $\sum_{i \in L} b^2_{0,i} b^1_{i,s} > 0$ and there exists $\ell \in L$ such that $b^2_{0\ell} b^1_{\ell,s} > 0$. Now $\ell \neq t$ since the pivot element $b^1_{t,s} > 0$ and $b^2_{0,t} < 0$ (t enters basis at T_2). But in T_2 we have $b^2_{0,i} \geq 0, i \in L \setminus \{t\}$ (Bland's rule) and so we have $b^1_{0,s} < 0, b^1_{\ell,s} > 0$ implying that we should have selected x_ℓ instead of x_t as the variable to leave the basis in T_1, contradiction.

Worst-case performance of the simplex algorithm While we can guarantee that the simplex algorithm will terminate eventually, we may not be here to see it. In a famous paper Klee and Minty showed that a natural choice of pivot column on a specially constructed problem with $2n$ constraints can result in $2^n - 1$ pivots. It is still unknown whether or not there is a choice of pivot column that gives an algorithm that runs in time poly$(m + n)$.

Shadow prices: The vector $\boldsymbol{\pi}^T = \mathbf{c}_\mathbf{B}^T\mathbf{B}^{-1}$ is often called the vector of *shadow prices*. (It is also the solution to the dual problem, see below.) Suppose that we perturb the RHS $\mathbf{b}$ to $\mathbf{b}' = \mathbf{b} + \boldsymbol{\delta}$. Suppose that $\mathbf{B}$ is the optimal basis. If $\mathbf{B}^{-1}\mathbf{b}' \geq 0$ then the reduced costs have not changed and so $\mathbf{B}$ remains as the optimal basis. The new optimal value will be $x_0' = \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{b}' = \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{b} + \mathbf{c}_B^T\mathbf{B}^{-1}\boldsymbol{\delta} = b_{0,R} + \sum_{i=1}^m \pi_i\delta_i$.

Reading off the shadow prices: The initial tableau will have an $m \times m$ identity matrix $\mathbf{I}$, where the ith cost is 0, either because it corresponds to a slack or to an artificial variable. The matrix $\mathbf{I}$ will be replaced by $\mathbf{B}^{-1}$ in the final tableau and the entry in the ith column of the objective row will therefore be $-(0 - \boldsymbol{\pi}^T\mathbf{e}_i) = \boldsymbol{\pi}_i$.

So, in Example 1, $\boldsymbol{\pi} = [1/3, 0, 5/3]^T$ and for $\boldsymbol{\delta}$ small, $x_0' = 23 + \delta_1/3 + 5\delta_3/3$. Notice that increasing 21 to $21 + \varepsilon$ does not lead to an increase in the optimal objective value. This is because the constraint $3x + y \leq 21$ is not tight at the optimum $(4, 5)$.

In Example 2, $\boldsymbol{\pi} = [-1/7, 0, 25/7]^T$ and for $\boldsymbol{\delta}$ small, $x_0' = 218/7 - \delta_1/7 + 25\delta_3/7$. Notice that increasing 8 to $8+\varepsilon$ does not lead to an increase in the optimal objective value. This is because the constraint $2x_1 + x_2 + 2x_3 \geq 8$ is not tight at the optimum $(0, 34/7, 29/7)$.

In the context of maximising the profit $\mathbf{c}^T\mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ where b_i represents the amount of some resource available, π_i represents the maximum price one should pay to buy one more unit of the ith resource. In the context of minimising the cost $\mathbf{c}^T\mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ where b_i represents the amount of some resource you need, π_i represents the extra cost if you need one more unit of the ith resource.

Note that in general, if we remove the requirement that $\boldsymbol{\delta}$ be small, then we have $x_0' \leq b_{0,R} + \sum_{i\in I} \pi_i\delta_i$. Indeed,

$$x_0' = b_{0,R} + \sum_{i\in I} \pi_i\delta_i + \sum_{j\in J} (c_j - \boldsymbol{\pi}^T\mathbf{a}_j)x_j \leq b_{0,R} + \sum_{i\in I} \pi_i\delta_i,$$

because at the maximum, $(c_j - \boldsymbol{\pi}^T\mathbf{a}_j)x_j \leq 0$ for $j \in J$.

Sensitivity Analysis: RHS This refers to computing the maximum by which we can change just b_ℓ without changing the optimal basis. If $\mathbf{e}_\ell = [0, 0, \ldots, 1, \ldots, 0]^T$ is the vector which all 0's except in the ℓth position where it equals 1, then this amounts to finding the interval for λ_ℓ such that $\mathbf{B}^{-1}(\mathbf{b} + \lambda_\ell\mathbf{e}_\ell) \geq 0$ i.e. such that for all $i \in I$, $b_{i,R} + \lambda_\ell\beta_{i,\ell} \geq 0$ where $\beta_{i,\ell}$ is the element in row i, column ℓ of $\mathbf{B}^{-1}$.

In Example 1, $\mathbf{B}^{-1} = \begin{bmatrix} 2/3 & 1 & -11/3 \\ -1/3 & 0 & 4/3 \\ 1/3 & 0 & -1/3 \end{bmatrix}$ and $\mathbf{B}^{-1}\mathbf{b} = [4, 4, 5]^T$ and $\lambda_1 \in [-6, 12], \lambda_2 \in [-4, \infty)$, $\lambda_3 \in [-3, 12/11]$.

In Example 2, $\mathbf{B}^{-1} = \begin{bmatrix} 1/7 & 0 & 3/7 \\ -1/7 & 0 & 4/7 \\ -1/7 & -1 & 11/7 \end{bmatrix}$ and $\mathbf{B}^{-1}\mathbf{b} = [34/7, 29/7, 36/7]^T$ and $\lambda_1 \in [-34, 29]$, $\lambda_2 \in (-\infty, 36/7], \lambda_3 \in [-36/11, \infty)$.

Sensitivity Analysis: objective row This refers to computing the maximum by which we can change just c_j without changing the optimal basis. For non-basic variables x_k this amounts to finding the interval for

λ_k such that $c_k + \lambda_k - \boldsymbol{\pi}^T \mathbf{a}_k \leq 0$ i.e. $\lambda_k \leq b_{0,k} = \boldsymbol{\pi}^T \mathbf{a}_k - c_k$. Note that the optimal solution does not change for λ_k this small.

In Example 1, x_1, x_2 are all basic and there is nothing to calculate.

In Example 2, $\lambda_1 \leq 10/7$.

For basic variables we must see the effect on the shadow prices and then check the change in reduced costs. Suppose that c_ℓ is replaced by $c_\ell + \lambda_\ell$. Then $\boldsymbol{\pi} = \mathbf{c}_\mathbf{B}^T \mathbf{B}^{-1}$ is replaced by $\boldsymbol{\pi} + \lambda_\ell \boldsymbol{\beta}_\ell$, where $\boldsymbol{\beta}_\ell$ is the ℓth row of $\mathbf{B}^{-1}$. Note that $\boldsymbol{\beta}_\ell^T \mathbf{a}_j = b_{\ell,j}$. The reduced costs $\widehat{c}_j = c_j - \boldsymbol{\pi}^T \mathbf{a}_j, j \in J$ then become $\widehat{c}_j - \lambda_\ell b_{\ell,j}, j \in J$ and we must keep the changed reduced costs non-positive. So the interval for λ_ℓ is

$$\lambda_\ell \in [\max \{\widehat{c}_j / b_{\ell,j} : b_{\ell,j} > 0, j \in J\}, \min \{\widehat{c}_j / b_{\ell,j} : b_{\ell,j} < 0, j \in J\}].$$

(In this calculation we use $c_j = 0$ for a non-basic slack variable.)

In Example 1, $\lambda_1 \in [-5/4, 1], \lambda_2 \in [-1, 5]$.

In Example 2, $\lambda_2 \in [-5/2, \infty), \lambda_3 \in [-10/3, \infty)$.

1.2 Duality

Linear programs come in pairs. One will be called the *primal* and the other will be called the *dual*. Which is which is not well-defined, since the dual of the dual is the primal.

The dual of the LP in standard form (3) is

$$\begin{aligned} \text{Minimise} \quad & \mathbf{b}^T\mathbf{y} \\ \text{Subject to} \quad & \mathbf{A}^T\mathbf{y} \geq \mathbf{c}. \end{aligned} \tag{5}$$

The dual of

$$\begin{aligned} \text{Maximise} \quad & \mathbf{c}_1^T\mathbf{x}_1 + \mathbf{c}_2^T\mathbf{x}_2 + \mathbf{c}_3^T\mathbf{x}_3 \\ \text{Subject to} \quad & \mathbf{A}_{1,1}\mathbf{x}_1 + \mathbf{A}_{1,2}\mathbf{x}_2 + \mathbf{A}_{1,3}\mathbf{x}_3 = \mathbf{b}_1 \\ & \mathbf{A}_{2,1}\mathbf{x}_1 + \mathbf{A}_{2,2}\mathbf{x}_2 + \mathbf{A}_{2,3}\mathbf{x}_3 \geq \mathbf{b}_2 \\ & \mathbf{A}_{3,1}\mathbf{x}_1 + \mathbf{A}_{3,2}\mathbf{x}_2 + \mathbf{A}_{3,3}\mathbf{x}_3 \leq \mathbf{b}_3 \\ & \mathbf{x}_1 \geq 0,\ \mathbf{x}_2 \leq 0. \end{aligned} \tag{6}$$

is

$$\begin{aligned} \text{Minimise} \quad & \mathbf{b}_1^T\mathbf{y}_1 + \mathbf{b}_2^T\mathbf{y}_2 + \mathbf{b}_3^T\mathbf{y}_3 \\ \text{Subject to} \quad & \mathbf{A}_{1,1}^T\mathbf{y}_1 + \mathbf{A}_{2,1}^T\mathbf{y}_2 + \mathbf{A}_{3,1}^T\mathbf{y}_3 \geq \mathbf{c}_1 \\ & \mathbf{A}_{1,2}^T\mathbf{y}_1 + \mathbf{A}_{2,2}^T\mathbf{y}_2 + \mathbf{A}_{3,2}^T\mathbf{y}_3 \leq \mathbf{c}_2 \\ & \mathbf{A}_{1,3}^T\mathbf{y}_1 + \mathbf{A}_{2,3}^T\mathbf{y}_2 + \mathbf{A}_{3,3}^T\mathbf{y}_3 = \mathbf{c}_3 \\ & \mathbf{y}_2 \leq 0,\ \mathbf{y}_3 \geq 0. \end{aligned} \tag{7}$$

To see this, we first reduce (6) to the standard form:

$$\begin{aligned} \text{Maximise} \quad & \mathbf{c}_1^T\mathbf{x}_1 - \mathbf{c}_2^T\mathbf{x}_2' + \mathbf{c}_3^T\mathbf{x}_3' - \mathbf{c}_3^T\mathbf{x}_3'' \\ \text{Subject to} \quad & \mathbf{A}_{1,1}\mathbf{x}_1 - \mathbf{A}_{1,2}\mathbf{x}_2' + \mathbf{A}_{1,3}\mathbf{x}_3' - \mathbf{A}_{1,3}\mathbf{x}_3'' = \mathbf{b}_1 \\ & \mathbf{A}_{2,1}\mathbf{x}_1 - \mathbf{A}_{2,2}\mathbf{x}_2' + \mathbf{A}_{2,3}\mathbf{x}_3' - \mathbf{A}_{2,3}\mathbf{x}_3'' - \mathbf{x}_4 = \mathbf{b}_2 \\ & \mathbf{A}_{3,1}\mathbf{x}_1 - \mathbf{A}_{3,2}\mathbf{x}_2' + \mathbf{A}_{3,3}\mathbf{x}_3' - \mathbf{A}_{3,3}\mathbf{x}_3'' + \mathbf{x}_5 = \mathbf{b}_3 \\ & \mathbf{x}_1 \geq 0,\ \mathbf{x}_2' \geq 0,\ \mathbf{x}_3' \geq 0,\ \mathbf{x}_3'' \geq 0,\ \mathbf{x}_4 \geq 0,\ \mathbf{x}_5 \geq 0. \end{aligned} \tag{8}$$

(We have added slack variables and replaced $\mathbf{x}_2$ by $-\mathbf{x}_2'$ and $\mathbf{x}_3$ by $\mathbf{x}_3' - \mathbf{x}_3''$.)
We can now use (5) to obtain the dual:

$$\begin{aligned} \text{Minimise} \quad & \mathbf{b}_1^T\mathbf{z}_1 + \mathbf{b}_2^T\mathbf{z}_2 + \mathbf{b}_3^T\mathbf{z}_3 \\ \text{Subject to} \quad & \mathbf{A}_{1,1}^T\mathbf{z}_1 + \mathbf{A}_{2,1}^T\mathbf{z}_2 + \mathbf{A}_{3,1}^T\mathbf{z}_3 \geq \mathbf{c}_1 \\ & -\mathbf{A}_{1,2}^T\mathbf{z}_1 - \mathbf{A}_{2,2}^T\mathbf{z}_2 - \mathbf{A}_{3,2}^T\mathbf{z}_3 \geq -\mathbf{c}_2 \\ & \mathbf{A}_{1,3}^T\mathbf{z}_1 + \mathbf{A}_{2,3}^T\mathbf{z}_2 + \mathbf{A}_{3,3}^T\mathbf{z}_3 \geq \mathbf{c}_3 \\ & -\mathbf{A}_{1,3}^T\mathbf{z}_1 - \mathbf{A}_{2,3}^T\mathbf{z}_2 - \mathbf{A}_{3,3}^T\mathbf{z}_3 \geq -\mathbf{c}_3 \\ & -\mathbf{z}_2 \geq 0,\ \mathbf{z}_3 \geq 0. \end{aligned} \tag{9}$$

Now to obtain (7), $\mathbf{y}_1 \leftarrow \mathbf{z}_1, \mathbf{y}_2 \leftarrow \mathbf{z}_2, \mathbf{y}_3 \leftarrow \mathbf{z}_3$. Then multiply the second inequality through by -1 and replace the third and fourth inequalities by an equation.

The relationship can be summarised: primal variables gives rise to dual constraints and vice-versa.
The primal constraint i is associated with a dual variable y_i and the dual constraint i is associated with primal variable x_i.
Equations give rise to free variables and vice-versa; in the context of primal maximisation, $\leq$ constraints are *normal* and give rise to normal non-negative dual variables and $\geq$ constraints are *abnormal* and give rise to abnormal non-positive dual variables.

As another example, suppose that the primal is

$$\begin{aligned} \text{Maximise} \quad & 10x_1 + 11x_2 + 12x_3 \\ \text{Subject to} \quad & x_1 + 2x_2 + 3x_3 = 20 \\ & 4x_1 + 5x_2 + 6x_3 \geq 21 \\ & 7x_1 + 8x_2 + 9x_3 \leq 22 \\ & x_1 \geq 0,\ x_2 \leq 0. \end{aligned}$$

then the dual is

$$\begin{aligned} \text{Minimise} \quad & 20y_1 + 21y_2 + 22y_3 \\ \text{Subject to} \quad & y_1 + 4y_2 + 7y_3 \geq 10 \\ & 2y_1 + 5y_2 + 8y_3 \leq 11 \\ & 3y_1 + 6y_2 + 9y_3 = 12 \\ & y_2 \leq 0,\ y_3 \geq 0. \end{aligned}$$

The dual of the dual is the primal If one takes the dual of (7) then one is back to (6).

Weak duality, standard form: if $\mathbf{x}$ is feasible for the primal (3) (primal feasible) and $\mathbf{y}$ is feasible for the dual (5) (dual feasible) then

$$\mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{y} = \mathbf{c}^T\mathbf{x} - (\mathbf{A}\mathbf{x})^T\mathbf{y} = \mathbf{x}^T\mathbf{c} - \mathbf{x}^T\mathbf{A}^T\mathbf{y} = \mathbf{x}^T(\mathbf{c} - \mathbf{A}^T\mathbf{y}) \leq 0.$$

So, if $\mathbf{x}_0$ is primal feasible and $\mathbf{y}_0$ is dual feasible and $\mathbf{c}^T\mathbf{x}_0 = \mathbf{b}^T\mathbf{y}_0$ then $\mathbf{x}_0$ solves the primal problem and $\mathbf{y}_0$ solves the dual problem.

Strong duality, standard form: if $\mathbf{x}_0$ solves the primal problem and $\mathbf{y}_0$ solves the dual problem then $\mathbf{c}^T\mathbf{x}_0 = \mathbf{b}^T\mathbf{y}_0$.
Let $\mathbf{B}$ be an optimal basis and let $\mathbf{y}_0^T = \mathbf{c}_\mathbf{B}^T\mathbf{B}^{-1}$. Then (i) $\mathbf{y}_0$ is dual feasible since the reduced costs $c_j - \mathbf{y}_0^T\mathbf{a}_j$ are all non-positive; (ii) the objective value is $\mathbf{c}_\mathbf{B}^T\mathbf{B}^{-1}\mathbf{b}$ for both primal and dual.

The following states what is possible for primal and dual:

1. Primal solvable & Dual Solvable.
2. Primal unbounded & Dual infeasible.
3. Primal infeasible &Dual infeasible.
4. Dual unbounded & Primal infeasible.

Complementary slackness: A primal feasible $\mathbf{x}_0$ and a dual feasible $\mathbf{y}_0$ satisfy *complementary slackness* if whenever a variable is positive then the corresponding dual constraint is satisfied with equality (tight). In which case $\mathbf{x}_0$ solves the primal and $\mathbf{y}_0$ solves the dual. Indeed, for our standard form,

$$\mathbf{c}^T\mathbf{x}_0 - \mathbf{b}^T\mathbf{y}_0 = \mathbf{x}_0^T(\mathbf{c} - \mathbf{A}^T\mathbf{y}_0) = 0.$$

Thus strong duality implies complementary slackness holds for $\mathbf{x}_0, \mathbf{y}_0$ optimal.

For an LP in the form:

$$\begin{aligned} \text{Maximise} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

which has a dual

$$\begin{aligned} \text{Minimise} \quad & \mathbf{b}^T\mathbf{y} \\ \text{subject to} \quad & \mathbf{A}^T\mathbf{y} = \mathbf{c} \\ & \mathbf{y} \geq 0 \end{aligned}$$

complementary slackness becomes $\mathbf{y}_0^T(\mathbf{b} - \mathbf{A}\mathbf{x}_0) = 0$ or $y_i > 0$ implies $\mathbf{r}_i^T\mathbf{x} = b_i$ where $\mathbf{r}_i$ denotes the ith row of $\mathbf{A}$. Another way of interpreting this is that $\mathbf{x}_0$ solves the primal if and only if $\mathbf{c}$ is a non-negative linear combination of the normals $\mathbf{r}_i$ of the tight constraints i such that $\mathbf{r}_i^T\mathbf{x}_0 = b_i$ i.e. $\mathbf{c}$ is in the *cone* generated by these normals.

Lemma 1 (Farkas Lemma)**.** *For an $m \times n$ matrix and $\mathbf{b} \in \mathbb{R}^m$, exactly one of the following holds:*

(i) There exists $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} \geq \mathbf{0}, \mathbf{A}\mathbf{x} = \mathbf{b}$.

(ii) There exists $\mathbf{u} \in \mathbb{R}^m$ such that $\mathbf{u}^T\mathbf{A} \geq \mathbf{0}$ and $\mathbf{u}^T\mathbf{b} < 0$.

Given an $m \times n$ real matrix $\mathbf{A}$ and $\mathbf{b} \in \mathbb{R}^m$ either (i) there exists $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} \geq 0$ and $\mathbf{A}\mathbf{x} = \mathbf{b}$ or (ii) there exists $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y}^T\mathbf{A} \geq 0$ and $\mathbf{y}^T\mathbf{b} < 0$.

First observe that we cannot have (i) and (ii). Indeed then

$$0 > \mathbf{y}^T\mathbf{b} = \mathbf{y}^T\mathbf{A}\mathbf{x} \geq 0, \qquad \text{contradiction.}$$

Suppose (i) fails and so the LP in standard form with $\mathbf{c} = 0$ is infeasible. Now $\mathbf{u} = 0$ satifies the dual constraint $\mathbf{A}^T\mathbf{u} \geq \mathbf{c}$ and so the dual is not infeasible. So, it is unbounded and (ii) holds.

Dual simplex algorithm: We work with the primal (3). A tableau is primal feasible if $b_{i,R} \geq 0$ for all $i \in I$. A tableau is dual feasible if $b_{0,j} \geq 0$ for all $j \in [n]$. The (primal) simplex algorithm starts with a primal feasible tableau (perhaps using artificial variables) and does pivots that maintain primal feasibility and improve the objective value until a tableau is reached that is both primal and dual feasible.

The dual simplex algorithm starts with a dual feasible tableau and does pivots that maintain dual feasibility and improve the (dual) objective value until a tableau is reached that is both primal and dual feasible.

We only need to discuss the choice of pivot:
Dual feasibility: we need $0 \leq b'_{0,\ell} = b_{0,\ell} - b_{0,k} \times \frac{b_{\ell,\ell}}{b_{\ell,k}} = -\frac{b_{0,k}}{b_{\ell,k}}$ or $b_{\ell,k} < 0$. Also need, for $j \in J \setminus \{k\}$, that

$0 \leq b'_{0,j} = b_{0,j} - b_{0,k} \times \frac{b_{\ell,j}}{b_{\ell,k}}$. This is automatically true if $b_{\ell,j} \geq 0$.
Otherwise we need $\frac{b_{0,j}}{b_{\ell,j}} \geq \frac{b_{0,k}}{b_{\ell,k}}$ – dual ratio test.
Improve the dual objective: this means reduce $b_{0,R}$. Now $b'_{0,R} = b_{0,R} - b_{0,k} \times \frac{b_{\ell,R}}{b_{\ell,k}}$ and so we want $b_{0,k} > 0$.

Summarising: we choose ℓ such that $b_{\ell,R} < 0$. Then we choose k by minimising $|b_{0,j}|/|b_{\ell,j}|$ over $b_{\ell,j} < 0$.

Example 3:

$$\begin{aligned} \text{Minimise } & 2x_1 + 3x_2 + 4x_3 \\ \text{Subject to } & x_1 + 4x_2 + 3x_3 \geq 7. \\ & 2x_1 + x_2 + 2x_3 \geq 8. \\ & x_1 + x_2 + x_3 \geq 9. \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

B.V.	x_1	x_2	x_3	x_4	x_5	x_6	*RHS*
x_0	-2	-3	-4				0
x_4	-1	$\mathbf{-4}$	-3	1			-7
x_5	-2	-1	-2		1		-8
x_6	-1	-1	-1			1	-9
x_0	$-5/4$		$-7/4$	$-3/4$			$21/4$
x_2	$1/4$	1	$3/4$	$-1/4$			$7/4$
x_5	$-7/4$		$-5/4$	$-1/4$	1		$-25/4$
x_6	$\mathbf{-3/4}$		$-1/4$	$-1/4$		1	$-29/4$
x_0			$-4/3$	$-1/3$		$-5/3$	$52/3$
x_2		1	$2/3$	$\mathbf{-1/3}$		$-1/3$	$-2/3$
x_5			$-2/3$	$1/3$	1	$-7/3$	$32/3$
x_1	1		$1/3$	$1/3$		$-4/3$	$29/3$
x_0		-1	-2			-2	18
x_4		-3	-2	1		-1	2
x_5		1			1	-2	10
x_1	1	1	1			-1	9

So the minimum is achieved for $x_1 = 9, x_2 = x_3 = 0$ and $x_0 = 18$.
Furthermore, the shadow prices/dual solution is (0,0,2).
(Note that we multiplied the top row entry -2 by -1, because we began by multiplying the rows through by -1. This has the effect of multiplying the optimal dual variables by -1.)

If we find during the execution of the dual algorithm that $b_{i,R} < 0$ and $b_{i,j} \geq 0$ for $j \in J$ then there is no feasible choice of pivot. But if this happens, then the primal is infeasible since feasibility would imply $0 \leq x_i + \sum_{j=1}^{n} b_{i,j} x_j = b_{i,R} < 0$. Because the dual is feasible, it must be unbounded.

It is instructive to determine the dual solution associated with a dual feasible basis. The dual constraints can be expressed $c_j - \mathbf{y}^T \mathbf{a}_j \leq 0$ for $j \in [n]$. Because $c_i - \mathbf{y}^T \mathbf{a}_i = 0$ for $i \in I$, the dual solution $\mathbf{y}$ is given by $\mathbf{y}^T = \mathbf{c}_\mathbf{B}^T \mathbf{B}^{-1}$ and $z_j = \mathbf{y}^T \mathbf{a}_j - c_j$ for $j \in [n]$.

Suppose now that $b_{\ell,R} < 0$ and $b_{\ell,j} \geq 0, j \in [n]$. z_ℓ is currently equal to 0. Suppose that we increase z_ℓ to λ, keeping $z_i = 0, i \in I \setminus \{\ell\}$. Then, the new solution is

$$\mathbf{y}^T = c_\mathbf{B}^T \mathbf{B}^{-1} + \lambda \mathbf{e}_\ell^T \mathbf{B}^{-1}, \quad z_\ell = \lambda, \quad z_i = 0, i \in I \setminus \{\ell\}, \quad z_j = c_\mathbf{B}^T \mathbf{B}^{-1} \mathbf{a}_j + \lambda \mathbf{e}_\ell^T \mathbf{B}^{-1} \mathbf{a}_j - c_j, j \in J.$$

Now $\mathbf{e}_\ell^T\mathbf{B}^{-1}\mathbf{a}_j = b_{\ell,j} \geq 0$ and so the above is dual feasible for arbitrary $\lambda > 0$. The value of the dual objective is $(c_\mathbf{B}^T\mathbf{B}^{-1} + \lambda\mathbf{e}_\ell^T\mathbf{B}^{-1})\mathbf{b} = b_{0,R} + \lambda b_{\ell,R} \to -\infty$ as $\lambda \to \infty$.

1.3 Column generation

In some circumstances we need to solve an LP where the number of columns is much too large for the constraint matrix to be stored in a computer. We will begin with an example of the *Cutting Stock Problem.*

The manufacturer of rolls of paper sells rolls in width $\ell_1 < \ell_2 < \cdots < \ell_m$. These are obtained by cutting *jumbo* rolls of width $\ell \geq \ell_m$. Suppose then that the manufacturer receives orders for d_i rolls of width $\ell_i, i = 1, 2, \ldots, m$. The manufacturer wishes to find a way to meet demand and to cut as few jumbo rolls as possible. This is an integer program, but it will suffice to consider its LP relaxation.

A *cutting pattern* $\Pi = (\xi_{\Pi,1}, \xi_{\Pi,2}, \ldots, \xi_{\Pi,m})$ implies cutting a jumbo roll into $\xi_{\Pi,i}$ rolls of width ℓ_i for $i = 1, 2, \ldots, m$. There will be some waste. Let x_Π denote the number of jumbo rolls cut into pattern Π. Then the problem becomes

$$\begin{aligned} \text{Minimise } & \sum_\Pi x_\Pi \\ \text{Subject to } & \sum_\Pi \xi_{\Pi,i}\, x_\Pi \geq d_i, i = 1, 2, \ldots, m \\ & x_\Pi \geq 0, \forall \Pi. \end{aligned} \tag{10}$$

Note that the number of possible patterns is $|\{(\xi_1, \ldots, \xi_m) : \xi_1\ell_1 + \cdots + \xi_m\ell_m \leq \ell\}| \geq \binom{\lfloor \ell/\ell_m \rfloor + m - 1}{m-1}$, which in principle can be extremely large, if $\ell \gg \ell_m$.

Let $\mathbf{A} = (\mathbf{a}_\Pi)$ be the $m \times n$ coefficient matrix and let $\mathbf{A}_S$ be the submatrix of $\mathbf{A}$ made up of the columns $\Pi \in S$. Suppose further that we have solved the problem

$$\begin{aligned} LP_S = \text{Minimise } & \sum_{\Pi \in S} x_\Pi \\ \text{Subject to } & \sum_{\Pi \in S} \xi_{\Pi,i}\, x_\Pi \geq d_i, i = 1, 2, \ldots, m \\ & x_\Pi \geq 0, \Pi \in S. \end{aligned}$$

For example, if we take $\Pi_i = (0, \ldots, 0, \lfloor \ell/\ell_i \rfloor, 0, \ldots 0)$ and $S = \{\Pi_i, i = 1, 2, \ldots, m\}$ then LP_S has a unique solution where we cut all rolls of width ℓ_i from the same roll.

Having solved LP_S we will have obtained optimal shadow prices $\boldsymbol{\pi}_S = [\pi_1, \pi_2, \ldots, \pi_m] \geq 0$ and normally we would look for a non-basic variable x_Π with reduced cost $1 - \boldsymbol{\pi}_S^T\mathbf{a}_\Pi < 0$. We cannot access all the reduced costs, but instead we solve the Knapsack problem

$$\text{Maximise } \sum_{i=1}^m \pi_i z_i. \tag{11}$$

$$\text{Subject to } \sum_{i=1}^m \ell_i z_i \leq \ell. \tag{12}$$

$$z_i \geq 0 \text{ and integer for } i = 1, 2, \ldots, m. \tag{13}$$

Note that each solution to (12),(13) defines a pattern and the objective (11) is 1 minus the reduced cost of the pattern. So there is pattern with negative reduced cost iff the optimal value to (11) is greater than 1. In which case we add the corresponding pattern to S and re-solve LP_S. Failing this, the solution to LP_S solves the LP relaxation of the cutting stock problem.

Note that the total amount of wasted roll is $\ell \sum_{\Pi} x_{\Pi} - \sum_{i=1}^{m} d_i \ell_i$ and so minimising the number of rolls cut also minimises the total waste.

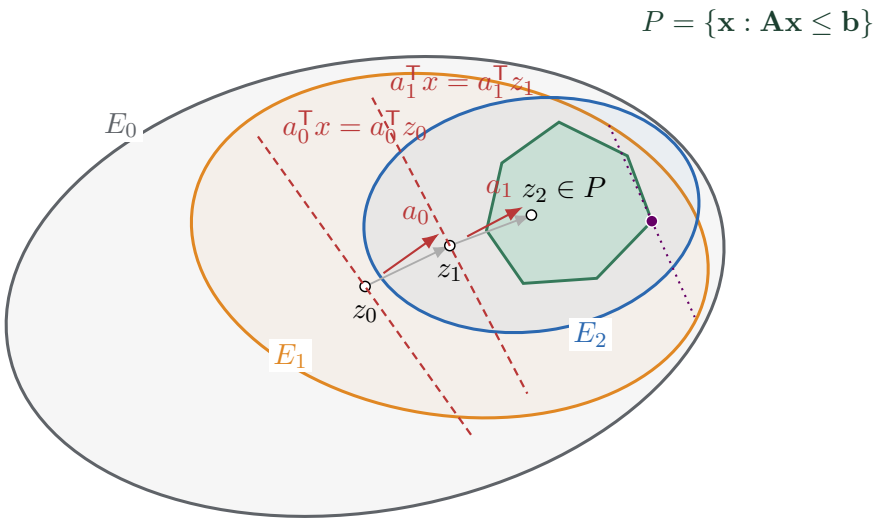
1.4 The Ellipsoid Algorithm – sketch

In the worst-case the simplex method takes exponential time. The Ellipsoid Algorithm was the first polynomial time algorithm for solving LP problems, discovered by the Russian mathematician Khachiyan. Suppose that $P = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ is either empty or is a full-dimensional bounded polytope. The algorithm runs in time $poly(L)$ where L is the number of bits needed to describe $\mathbf{A}, \mathbf{b}$. The algorithm aims to find a point in P or prove that $P = \emptyset$. Optimising $\mathbf{c}^T\mathbf{x}$ over P can be done by using binary search over problems that check for the emptiness of polytopes of the form $Q = \left\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{c}^T\mathbf{x} \leq c\right\}$ or $\left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{c}^T\mathbf{x} \geq c\right\}$.

The first step is to find an *ellipsoid* E that contains P. Here an ellipsoid is a set of the form $\left\{\mathbf{x} : (\mathbf{x}-\mathbf{z})^T\mathbf{M}^{-1}(\mathbf{x}-\mathbf{z}) \leq 1\right\}$ where $\mathbf{M}$ is a positive definite matrix. Here $\mathbf{z}$ will be the centre of E. For example we can take $\mathbf{M} = n(n!2^{nL})^2\mathbf{I}_n$.

The algorithm proceeds as follows: it generates a sequence of ellipsoids $E_k = \left\{\mathbf{x} \in \mathbb{R}^n : (\mathbf{x}-\mathbf{z}_k)^T\mathbf{M}_k^{-1}(\mathbf{x}-\mathbf{z}_k) \leq 1\right\} \supseteq P$, $k \geq 0$ until either we find that $\mathbf{z}_k \in P$ or that the volume of E_k is so small that P must be empty.

If $\mathbf{a}^T\mathbf{z}_k > b_k$, then the algorithm constructs an ellipsoid E_{k+1} that contains the intersection of the half-space $\left\{\mathbf{x} : \mathbf{a}_k^T\mathbf{x} \leq \mathbf{a}_k^T\mathbf{z}_k\right\}$ and the retained half of E_k.



Thus the first step of iteration k is to check whether or not the center $\mathbf{z}_k \in P$ i.e. does $\mathbf{z}_k$ satisfy all the constraints of P. So, suppose for example that $\mathbf{a}_1^T\mathbf{z}_k > b_1$ and that H be the half-space $\left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_1^T\mathbf{x}_k \leq b_1\right\}$. So, $P \subseteq E_k \cap$

H and the crucial step is to find another ellipsoid E_{k+1} such that (i) $E_{k+1} \supseteq E_k \cap H$ and $vol(E_{k+1})/vol(E_k) \leq \exp\left\{-\frac{1}{2(n+1)}\right\}$. Thus after t unsuccessful steps, we find that $vol(E_t) \leq vol(E_0)\exp\left\{-\frac{t}{2(n+1)}\right\}$.

Volume of E_0 Assume that all the entries of $\mathbf{A}, \mathbf{b}$ are rationals. If P is non-empty then crudely the absolute values of the basic variables of any basis will lie in $[2^{-Ln}, 2^{Ln}n!]$ and so we can take $\mathbf{M}_0 = 2^L n\mathbf{I}_n$ and then $vol(E_0) \leq 2^{Ln}$.

Smallest possible volume of P If P is non-empty then it will contain a simplex, and so its volume will be at least $2^{-Ln}/(n+1)!$.

It follows that the ellipsoid algorithm will finish after at most $2(n+1)\log(2^{2Ln}(n+1)!) \leq poly(L)$ steps.

Updating the ellipsoid Volume ratios are preserved by translation and linear transformations. Furthermore, since $\mathbf{M}_k$ is positive definite, we can write $\mathbf{M}_k = \mathbf{N}_k\mathbf{N}_k^T$ for some non-singular matrix $\mathbf{N}_k$. So, putting $\mathbf{y} = \mathbf{N}_k^{-1}(\mathbf{x} - \mathbf{z}_k)$ (and $\mathbf{x} = \mathbf{N}_k\mathbf{y} + \mathbf{z}_k$) we see that the bounding ellipsoid is transformed into the ball $B(0,1) = \{\mathbf{y} : \mathbf{y}^T\mathbf{y} \leq 1\}$ and $\mathbf{z}_k$ becomes the origin. We can rotate our "picture" so that $H = \{\mathbf{y} : y_1 \geq 0\}$.

This is all reversible and so once we find a new ellipsoid w.r.t. the ball, we can find the corresponding ellipsoid for $\mathbf{M}_k, \mathbf{z}_k$ etc.

Now let

$$\mathbf{Q}^{-1} = \frac{n^2-1}{n^2}\begin{bmatrix} \frac{n+1}{n-1} & & 0 \\ & 1 & \\ 0 & & 1 \\ & & & \ddots \end{bmatrix} \qquad \text{and } \mathbf{z}^T = \left(\frac{1}{n+1}, 0, \ldots, 0\right).$$

Now consider the ellipsoid $E = \{\mathbf{y} : (\mathbf{y}-\mathbf{z})^T\mathbf{Q}^{-1}(\mathbf{y}-\mathbf{z}) \leq 1\}$. All we need to do now is check that (i) $vol(E)/vol(B(0,1)) \leq \exp\{-1/(2(n+1))\}$ and (ii) E contains $H \cap B(0,1)$. For (i) we have

$$\begin{aligned}\frac{vol(E)}{vol(B(0,1))} = (\det(\mathbf{Q}))^{1/2} &= \left(\left(\frac{n^2}{n^2-1}\right)^n \frac{n-1}{n+1}\right)^{1/2} = \left(\left(1+\frac{1}{n^2-1}\right)^n\left(1-\frac{2}{n+1}\right)\right)^{1/2} \\ &\leq \exp\left\{\frac{n}{2(n^2-1)} - \frac{1}{n+1}\right\} = \exp\left\{-\frac{1}{2(n+1)}\right\}.\end{aligned}$$

For (ii), suppose that $y_1 \geq 0$ and $\|\mathbf{y}\| \leq 1$ then, where $\mathbf{y} = (y_1, \widehat{\mathbf{y}})$,

$$\begin{aligned}
(\mathbf{y}-\mathbf{z})^T\mathbf{Q}^{-1}(\mathbf{y}-\mathbf{z}) &= \frac{n^2-1}{n^2}\left(y_1 - \frac{1}{n+1}, \widehat{\mathbf{y}}\right)^T \begin{bmatrix} \frac{n+1}{n-1} & 0 & \\ 0 & 1 & \\ & & \ddots \end{bmatrix} \left(y_1 - \frac{1}{n+1}, \widehat{\mathbf{y}}\right) \\
&= \frac{n^2-1}{n^2}\frac{n+1}{n-1}\left(y_1 - \frac{1}{n+1}\right)^2 + \frac{(n^2-1)}{n^2}\|\widehat{\mathbf{y}}\|^2 \\
&= \frac{1}{n^2}((n+1)y_1 - 1)^2 + \frac{(n^2-1)}{n^2}\|\widehat{\mathbf{y}}\|^2 \\
&\leq \frac{1}{n^2}((n+1)y_1 - 1)^2 + \frac{(n^2-1)}{n^2}(1-y_1^2) \quad \text{since } y_1^2 + \|\widehat{\mathbf{y}}\|^2 \leq 1 \\
&= \frac{(n+1)^2}{n^2}y_1^2 - 2\frac{n+1}{n^2}y_1 + \frac{1}{n^2} + \frac{n^2-1}{n^2} - \frac{n^2-1}{n^2}y_1^2 \\
&= \frac{2(n+1)}{n^2}(y_1^2 - y_1) + 1 \\
&\leq 1, \quad \text{since } 0 \leq y_1 \leq 1.
\end{aligned}$$

We can put these transformations altogether and get

$$\mathbf{d}_k = \frac{\mathbf{M}_k\mathbf{a}_1}{\sqrt{\mathbf{a}_1^T\mathbf{M}_k\mathbf{a}_1}}; \quad \mathbf{z}_{k+1} = \mathbf{z}_k - \frac{\mathbf{d}_k}{n+1}; \quad \mathbf{M}_{k+1} = \frac{n^2}{n^2-1}\left(\mathbf{M}_k - \frac{2}{n+1}\mathbf{d}_k\mathbf{d}_k^T\right).$$

There is one point that we do not cover. That is the number of bits of accuracy that we need to keep in the calculation. We must check that this number remains polynomial in L.

1.5 Interior point method

While the simplex algorithm moves along the edges of the boundary, one intuitively feels that one can make better progress by moving in the direction of the objective function in the interior of the polyhedron P defining the feasible region. In this section we descibe a primal-dual algorithm that moves close to the *central path*. The central path is a continuous curve of increasingly accurate primal-dual solutions. We use *Newton's method* to supply the discrete step used to move from a neighborhood of one point on this curve to a neighborhood of a point with a smaller duality gap.

Consider the standard primal-dual linear programming pair:

$$\text{(P)} \quad \min_{\mathbf{x}}\{\mathbf{c}^T\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b},\, \mathbf{x} \geq 0\} \qquad \text{(D)} \quad \max_{\mathbf{y},\mathbf{s}}\{\mathbf{b}^T\mathbf{y} : \mathbf{A}^T\mathbf{y} + \mathbf{s} = \mathbf{c},\, \mathbf{s} \geq 0\}, \tag{14}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ has full row rank, $\mathbf{x}, \mathbf{s}, \mathbf{c} \in \mathbb{R}^n$, and $\mathbf{y}, \mathbf{b} \in \mathbb{R}^m$. We assume that the primal and dual have strictly feasible points and that an optimal solution exists. Write

$$\mathbf{X} = \operatorname{diag}(\mathbf{x}), \qquad \mathbf{S} = \operatorname{diag}(\mathbf{s}), \qquad \mathbf{e} = (1, \ldots, 1)^T,$$

and let

$$\mathcal{F}^\circ = \{(\mathbf{x}, \mathbf{y}, \mathbf{s}) : \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{A}^T\mathbf{y} + \mathbf{s} = \mathbf{c},\ \mathbf{x} > 0,\ \mathbf{s} > 0\}$$

denote the set of strictly feasible primal-dual points.

For any primal-dual feasible point,

$$\mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{y} = \mathbf{x}^T(\mathbf{A}^T\mathbf{y} + \mathbf{s}) - (\mathbf{A}\mathbf{x})^T\mathbf{y} = \mathbf{x}^T\mathbf{s}. \tag{15}$$

Thus $\mathbf{x}^T\mathbf{s}$ is the primal-dual gap. Since every term $x_i s_i$ is nonnegative, a feasible point is optimal precisely when

$$x_i s_i = 0, \qquad i = 1, \ldots, n.$$

These are the complementary-slackness equations. They place an optimal solution on the boundary, because for every i at least one of x_i and s_i must be zero.

The central path The central path replaces the boundary equations $x_i s_i = 0$ by the perturbed equations

$$x_i s_i = \tau, \qquad i = 1, \ldots, n,$$

where $\tau > 0$. More precisely, for each $\tau > 0$ the central point $(\mathbf{x}(\tau), \mathbf{y}(\tau), \mathbf{s}(\tau))$ is the solution of

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{16}$$
$$\mathbf{A}^T\mathbf{y} + \mathbf{s} = \mathbf{c}, \tag{17}$$
$$\mathbf{X}\mathbf{S}\mathbf{e} = \tau\mathbf{e}, \tag{18}$$
$$\mathbf{x} > 0, \qquad \mathbf{s} > 0. \tag{19}$$

The set

$$\mathcal{C} = \{(\mathbf{x}(\tau), \mathbf{y}(\tau), \mathbf{s}(\tau)) : \tau > 0\}$$

is called the *central path*. Along this path every complementarity product has the same value, so no coordinate pair (x_i, s_i) approaches the boundary substantially faster than the others. If $x_i(\tau)$ becomes small, then $s_i(\tau) = \tau / x_i(\tau)$ compensates for it, and conversely.

The central path also has a barrier interpretation. For fixed $\tau > 0$, consider the logarithmic-barrier problem

$$\min \left\{ \mathbf{c}^T\mathbf{x} - \tau \sum_{i=1}^{n} \log x_i : \mathbf{A}\mathbf{x} = \mathbf{b},\ x > 0 \right\}. \tag{20}$$

Using the Lagrangian

$$\mathcal{L}_\tau(\mathbf{x}, \mathbf{y}) = \mathbf{c}^T\mathbf{x} - \tau \sum_{i=1}^{n} \log x_i - \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b}),$$

the stationarity equation is

$$\mathbf{c} - \mathbf{A}^T\mathbf{y} - \tau\mathbf{X}^{-1}\mathbf{e} = 0.$$

Defining $\mathbf{s} = \mathbf{c} - \mathbf{A}^T\mathbf{y}$ gives $\mathbf{s} = \tau\mathbf{X}^{-1}\mathbf{e}$, which is exactly $\mathbf{X}\mathbf{S}\mathbf{e} = \tau\mathbf{e}$. Thus the central point is the minimizer of the primal objective with a logarithmic penalty that becomes infinite at the boundary $x_i = 0$.

By (15), the duality gap at a central point is

$$\mathbf{c}^T\mathbf{x}(\tau) - \mathbf{b}^T\mathbf{y}(\tau) = \mathbf{x}(\tau)^T\mathbf{s}(\tau) = n\tau. \tag{21}$$

Consequently, reducing τ drives the duality gap to zero. As $\tau \downarrow 0$, every accumulation point of the central path is primal-dual optimal. The role of a path-following method is therefore to approximately follow $\mathcal{C}$ while repeatedly reducing its parameter.

Neighborhood of the central path At a general strictly feasible point, let

$$\mu = \frac{\mathbf{x}^T\mathbf{s}}{n} > 0$$

denote the normalized duality gap. On the central path, $\mu = \tau$. The algorithm does not require its iterates to satisfy $\mathbf{XSe} = \mu\mathbf{e}$ exactly. Instead, it keeps them in the 2-norm neighborhood

$$\mathcal{N}_2(\theta) = \{(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{F}^\circ : \|\mathbf{XSe} - \mu\mathbf{e}\|_2 \leq \theta\mu\}, \tag{22}$$

where $\theta \in (0, 1)$ is a fixed scalar parameter, typically a small constant such as $\theta = 0.4$.

This is a measure of centrality rather than Euclidean distance from a particular point. The primal and dual equations are satisfied exactly, while the vector of complementarity products is required to be nearly constant. In particular, membership in $\mathcal{N}_2(\theta)$ implies, for every i,

$$(1 - \theta)\mu \leq x_i s_i \leq (1 + \theta)\mu, \tag{23}$$

because $|x_i s_i - \mu| \leq \|\mathbf{XSe} - \mu\mathbf{e}\|_2$.

1.5.1 Newton's Method

Newton's method for a system of equations For a scalar equation $f(z) = 0$, Newton's method replaces f near the current point z by its first-order approximation

$$f(z + \Delta z) \approx f(z) + f'(z)\Delta z$$

and chooses Δz so that the approximation is zero. For a vector equation $F(z) = 0$, the corresponding Newton equation is

$$J_F(z)\Delta z = -F(z), \tag{24}$$

where $J_F(z)$ is the Jacobian. The new point is $z^+ = z + \Delta z$. Newton's equation cancels the current residual and all of its first-order change; the residual left at the new point consists of second-order terms. This quadratic residual is the key reason that a full Newton step can remain close to the central path when the target parameter is changed only slightly.

For a prescribed target parameter $\tau > 0$, define the perturbed KKT map (we are trying to solve $F_\tau = 0$):

$$F_\tau(\mathbf{x}, \mathbf{y}, \mathbf{s}) = \begin{bmatrix} \mathbf{Ax} - \mathbf{b} \\ \mathbf{A}^T\mathbf{y} + \mathbf{s} - \mathbf{c} \\ \mathbf{XSe} - \tau\mathbf{e} \end{bmatrix}. \tag{25}$$

Its Jacobian, with variables ordered as $(\mathbf{x}, \mathbf{y}, \mathbf{s})$, is

$$J_{F_\tau}(\mathbf{x}, \mathbf{y}, \mathbf{s}) = \begin{bmatrix} \mathbf{A} & 0 & 0 \\ 0 & \mathbf{A}^T & \mathbf{I} \\ \mathbf{S} & 0 & \mathbf{X} \end{bmatrix}. \tag{26}$$

The first two block rows are linear feasibility equations. The only nonlinear equation is $\mathbf{XSe} = \tau\mathbf{e}$.

Path-following and centering in one step Suppose the current point belongs to $\mathcal{N}_2(\theta)$ and has normalized gap μ. At each iteration, we reduce the target barrier parameter by choosing

$$\mu^+ = (1-\beta)\mu, \qquad \beta = \frac{\gamma}{\sqrt{n}}, \tag{27}$$

where $\gamma > 0$ is a sufficiently small constant. We then apply Newton's method to $F_{\mu^+} = 0$. From (16) – (18) and (26), we see that the Newton equations are

$$\mathbf{A}\Delta\mathbf{x} = 0, \tag{28}$$

$$\mathbf{A}^T\Delta\mathbf{y} + \Delta\mathbf{s} = 0, \tag{29}$$

$$\mathbf{S}\Delta\mathbf{x} + \mathbf{X}\Delta\mathbf{s} = \mu^+\mathbf{e} - \mathbf{XSe}. \tag{30}$$

The RHS of equation (30) can be re-written as

$$\mu^+\mathbf{e} - \mathbf{XSe} = (\mu\mathbf{e} - \mathbf{XSe}) - \beta\mu\mathbf{e}. \tag{31}$$

The first term is the centering component: it acts to remove, to first order, the deviation of the current point from the central path corresponding to the current value of μ. The second term is the progress component: it shifts the target to the smaller parameter $\mu^+ = (1-\beta)\mu$. The short-step choice $\beta = O(1/\sqrt{n})$ keeps the combined Newton direction small enough that a full Newton step preserves positivity and remains within the prescribed neighborhood of the central path.

The third equation is obtained by linearizing complementarity. Indeed,

$$(\mathbf{X}+\Delta\mathbf{X})(\mathbf{S}+\Delta\mathbf{S})\mathbf{e} = \mathbf{XSe} + \mathbf{S}\Delta\mathbf{x} + \mathbf{X}\Delta\mathbf{s} + \Delta\mathbf{X}\Delta\mathbf{Se}. \tag{32}$$

Newton's method omits the quadratic term $\Delta\mathbf{X}\Delta\mathbf{Se}$ and chooses the linear terms so that the target $\mu^+ e$ is reached. Consequently, after a full Newton step, the complementarity residual is exactly the omitted second-order term.

Solving the Newton system Let

$$r_c = \mu^+ e - \mathbf{XSe}, \qquad \mathbf{D} = \mathbf{X}^{1/2}\mathbf{S}^{-1/2}.$$

From (29),

$$\Delta\mathbf{s} = -\mathbf{A}^T\Delta\mathbf{y}.$$

Substituting this into (30) gives

$$\mathbf{S}\Delta\mathbf{x} - \mathbf{X}\mathbf{A}^T\Delta\mathbf{y} = r_c,$$

and hence

$$\Delta\mathbf{x} = \mathbf{S}^{-1}r_c + \mathbf{D}^2\mathbf{A}^T\Delta\mathbf{y}, \qquad \mathbf{D}^2 = \mathbf{XS}^{-1} = \mathbf{S}^{-1}\mathbf{X}. \tag{33}$$

Using $\mathbf{A}\Delta\mathbf{x} = 0$, we obtain the normal equations

$$\mathbf{AD}^2\mathbf{A}^T\Delta\mathbf{y} = -\mathbf{AS}^{-1}r_\mathbf{c}. \tag{34}$$

The matrix $\mathbf{AD}^2\mathbf{A}^T$ is symmetric positive definite. In fact, for $\mathbf{v} \neq 0$,

$$\mathbf{v}^T\mathbf{AD}^2\mathbf{A}^T\mathbf{v} = \|\mathbf{DA}^T\mathbf{v}\|_2^2 > 0,$$

because $\mathbf{D}$ is nonsingular and the full row rank of $\mathbf{A}$ implies that $\mathbf{A}^T$ is also nonsingular. Thus (34) has a unique solution. After finding $\Delta\mathbf{y}$, one recovers $\Delta\mathbf{s} = -\mathbf{A}^T\Delta\mathbf{y}$ and then $\Delta\mathbf{x}$ from (33).

Orthogonality and Search Direction Bounds Premultiplying (30) by $(\mathbf{XS})^{-1/2}$ yields

$$\mathbf{D}^{-1}\Delta\mathbf{x} + \mathbf{D}\Delta\mathbf{s} = (\mathbf{XS})^{-1/2}(\mu^{+}\mathbf{e} - \mathbf{XSe}). \tag{35}$$

Notice that

$$\Delta\mathbf{x}^{T}\Delta\mathbf{s} = \Delta\mathbf{x}^{T}(-\mathbf{A}^{T}\Delta\mathbf{y}) = -(\mathbf{A}\Delta\mathbf{x})^{T}\Delta\mathbf{y} = 0$$

due to the orthogonality of $\ker(\mathbf{A})$ and $\mathrm{range}(\mathbf{A}^{T})$. Taking the squared 2-norm of both sides of (35) gives

$$\|\mathbf{D}^{-1}\Delta\mathbf{x}\|_2^2 + \|\mathbf{D}\Delta\mathbf{s}\|_2^2 = \|(\mathbf{XS})^{-1/2}(\mu^{+}\mathbf{e} - \mathbf{XSe})\|_2^2. \tag{36}$$

Using the Cauchy–Schwarz and the arithmetic-geometric mean inequality, we obtain

$$\begin{aligned}\|\Delta\mathbf{X}\Delta\mathbf{Se}\|_2 &\le \|\mathbf{D}^{-1}\Delta\mathbf{x}\|_2\,\|\mathbf{D}\Delta\mathbf{s}\|_2 \\ &\le \frac{1}{2}\left(\|\mathbf{D}^{-1}\Delta\mathbf{x}\|_2^2 + \|\mathbf{D}\Delta\mathbf{s}\|_2^2\right) \\ &= \frac{1}{2}\|(\mathbf{XS})^{-1/2}(\mu^{+}\mathbf{e} - \mathbf{XSe})\|_2^2. \end{aligned} \tag{37}$$

Neighborhood Invariance Let

$$\mathbf{r} = (\mathbf{XS})^{-1/2}(\mu^{+}\mathbf{e} - \mathbf{XSe}).$$

By (31), the neighborhood assumption, and $\beta\sqrt{n} = \gamma$,

$$\|\mu^{+}\mathbf{e} - \mathbf{XSe}\|_2 \le \|\mu\mathbf{e} - \mathbf{XSe}\|_2 + \beta\mu\|\mathbf{e}\|_2 \le (\theta + \gamma)\mu. \tag{38}$$

Equation (23) then implies

$$\|\mathbf{r}\|_2 \le \frac{\theta + \gamma}{\sqrt{1 - \theta}}\sqrt{\mu}. \tag{39}$$

The first two Newton equations preserve primal and dual feasibility exactly. To check strict positivity, observe from (36) that

$$\|\mathbf{D}^{-1}\Delta\mathbf{x}\|_2 \le \|\mathbf{r}\|_2, \qquad \|\mathbf{D}\Delta\mathbf{s}\|_2 \le \|\mathbf{r}\|_2.$$

For every coordinate,

$$\frac{|\Delta x_i|}{x_i} = \frac{|\mathbf{D}^{-1}\Delta(\mathbf{x})_i|}{\sqrt{x_i s_i}} \le \frac{\theta + \gamma}{1 - \theta}, \tag{40}$$

$$\frac{|\Delta s_i|}{s_i} = \frac{|(\mathbf{D}\Delta\mathbf{s})_i|}{\sqrt{x_i s_i}} \le \frac{\theta + \gamma}{1 - \theta}. \tag{41}$$

Thus, if $\theta + \gamma < 1 - \theta$, a full Newton step satisfies $\mathbf{x} + \Delta\mathbf{x} > 0$ and $\mathbf{s} + \Delta\mathbf{s} > 0$.

Now put

$$\mathbf{x}^{+} = \mathbf{x} + \Delta\mathbf{x}, \qquad \mathbf{y}^{+} = \mathbf{y} + \Delta\mathbf{y}, \qquad \mathbf{s}^{+} = \mathbf{s} + \Delta\mathbf{s}.$$

By (32) and the Newton equation,

$$\mathbf{X}^{+}\mathbf{S}^{+}\mathbf{e} = \mu^{+}\mathbf{e} + \Delta\mathbf{X}\Delta\mathbf{Se}. \tag{42}$$

Therefore, using (37) and (39),

$$\|\mathbf{X}^{+}\mathbf{S}^{+}\mathbf{e} - \mu^{+}\mathbf{e}\|_2 \le \frac{1}{2}\|\mathbf{r}\|_2^2 \le \frac{(\theta + \gamma)^2}{2(1 - \theta)}\mu = \frac{(\theta + \gamma)^2}{2(1 - \theta)(1 - \beta)}\mu^{+}. \tag{43}$$

For example, if $\theta = 0.4$ and $\gamma = 0.1$, then $\beta \leq 0.1$ and

$$\frac{(\theta+\gamma)^2}{2(1-\theta)(1-\beta)} \leq \frac{0.25}{2(0.6)(0.9)} < 0.4 = \theta.$$

It follows that $(\mathbf{x}^+, \mathbf{y}^+, \mathbf{s}^+) \in \mathcal{N}_2(\theta)$. This establishes both strict feasibility and neighborhood membership for the full Newton step.

Global Linear Convergence Rate Since

$$\mathbf{e}^T \Delta\mathbf{X}\Delta\mathbf{S}\mathbf{e} = \Delta\mathbf{x}^T\Delta\mathbf{s} = 0,$$

we have

$$\begin{aligned}\mu^{k+1} &= \frac{(\mathbf{x}^+)^T\mathbf{s}^+}{n} = \frac{\mathbf{e}^T\mathbf{X}^+\mathbf{S}^+\mathbf{e}}{n} \\ &= \frac{\mathbf{e}^T(\mu^+\mathbf{e} + \Delta\mathbf{X}\Delta\mathbf{S}\mathbf{e})}{n} = \mu^+ = \left(1 - \frac{\gamma}{\sqrt{n}}\right)\mu^k. \end{aligned} \tag{44}$$

This proves that the duality gap shrinks by the factor $1 - \gamma/\sqrt{n}$ at every iteration. Geometrically, the Newton correction restores centrality after the target parameter has been moved forward, while the orthogonality relation ensures that the average complementarity is exactly the prescribed value μ^+.

Overall Complexity Derivation To reduce the duality gap from an initial μ^0 to a tolerance ϵ, it is enough to choose K so that

$$\mu^K = \left(1 - \frac{\gamma}{\sqrt{n}}\right)^K \mu^0 \leq \epsilon. \tag{45}$$

Taking logarithms and using $\ln(1-z) \leq -z$ gives

$$K = \mathcal{O}\left(\sqrt{n}\log\left(\frac{\mu^0}{\epsilon}\right)\right). \tag{46}$$

1.6 Total Arithmetic Complexity

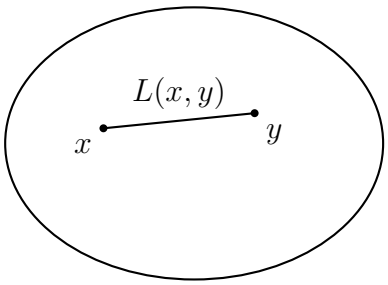
1. **Iteration count:** For accuracy $\epsilon = 2^{-L}$ and $\mu^0 = 2^{\mathcal{O}(L)}$, the iteration count is $K = \mathcal{O}(\sqrt{n}L)$.
2. **Cost per step:** For a dense problem, constructing and factorizing AD^2A^T costs $\mathcal{O}(m^2n+m^3)$ operations, which is $\mathcal{O}(n^3)$ when $m \leq n$.

Hence the total arithmetic complexity is $\mathcal{O}(n^{3.5}L)$ under this dense-matrix cost model.

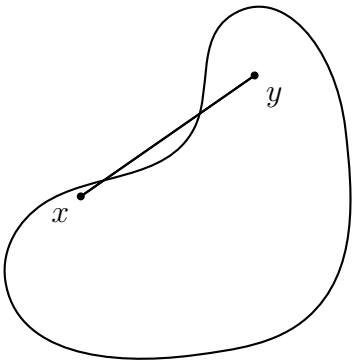
2 Convex Sets

A set $S \subseteq \mathbb{R}^n$ is said to be *convex* if for distinct $\mathbf{x}, \mathbf{y} \in S$ the *line segment*

$$L(\mathbf{x}, \mathbf{y}) = \{\lambda\mathbf{x} + (1-\lambda)\mathbf{y} : 0 \leq \lambda \leq 1\} \subseteq S.$$



Convex Set



Non-Convex Set

Examples of convex sets:

C1 $S = \{\mathbf{x} : \mathbf{a}^T\mathbf{x} = 1\}$. $\mathbf{x}, \mathbf{y} \in S$ implies that

$$\mathbf{a}^T(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) = \lambda\mathbf{a}^T\mathbf{x} + (1-\lambda)\mathbf{a}^T\mathbf{y} = \lambda + (1-\lambda) = 1.$$

C2 $S = \{\mathbf{x} : \mathbf{a}^T\mathbf{x} \leq 1\}$. Proof similar to C1.

C3 $S = B(0,\delta)$: $\mathbf{x}, \mathbf{y} \in S$ implies that

$$|\lambda\mathbf{x} + (1-\lambda)\mathbf{y}| \leq |\lambda\mathbf{x}| + |(1-\lambda)\mathbf{y}| \leq \lambda\delta + (1-\lambda)\delta = \delta.$$

C4 $S = B(0,\delta)$: $\mathbf{x}, \mathbf{y} \in S$ implies that

$$|\lambda\mathbf{x} + (1-\lambda)\mathbf{y}| \leq |\lambda\mathbf{x}| + |(1-\lambda)\mathbf{y}| \leq \lambda\delta + (1-\lambda)\delta = \delta.$$

C5 If f is convex, then the *level set* $\{\mathbf{x} : f(\mathbf{x}) \leq 0\}$ is convex.
$f(\mathbf{x}), f(\mathbf{y}) \leq 0$ implies that $f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}) \leq 0$.

Operations on convex sets:

O1 S convex and $\mathbf{x} \in \mathbb{R}^n$ implies that $\mathbf{x} + S = \{\mathbf{x} + \mathbf{y} : \mathbf{y} \in S\}$ is convex.

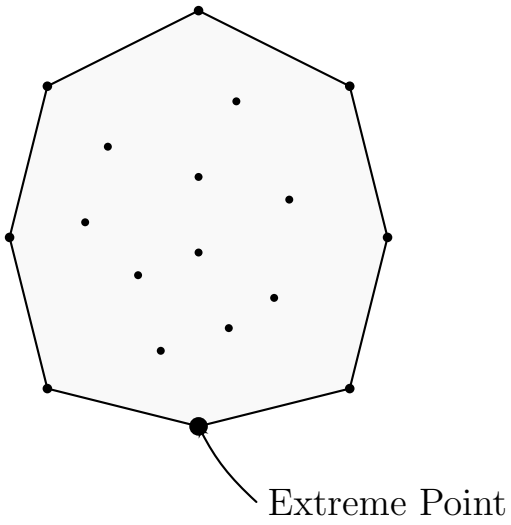
O2 S, T convex implies that $A = S \cap T$ is convex. $\mathbf{x}, \mathbf{y} \in A$ implies that $\mathbf{x}, \mathbf{y} \in S$ and so $L = L(\mathbf{x}, \mathbf{y}) \subseteq S$. Similarly, $L \subseteq T$ and so $L \subseteq S \cap T$.

O3 Using induction we see that if $S_i, 1 \leq i \leq k$ are convex then so is $\bigcap_{i=1}^{k} S_i$.

O4 If S, T are convex sets and $\alpha, \beta \in \mathbb{R}$ then $\alpha S + \beta T = \{\alpha \mathbf{x} + \beta \mathbf{y} : \mathbf{x} \in S, \mathbf{y} \in T\}$ is convex.
If $\mathbf{z}_i = \alpha \mathbf{x}_i + \beta \mathbf{y}_i \in \alpha S + \beta T, i = 1, 2$ then

$$\lambda \mathbf{z}_1 + (1-\lambda)\mathbf{z}_2 = \alpha(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) + \beta(\lambda \mathbf{y}_1 + (1-\lambda)\mathbf{y}_2) \in \alpha S + \beta T.$$

It follows from C1,C2 and O3 that an affine subspace $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$ and a halfspace $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ are convex for any matrix $\mathbf{A}$ any vector $\mathbf{b}$. So the feasible region of a linear program is a convex set.



Clearly $S \subseteq conv(S)$, since we can take $|I| = 1$.

Lemma 2. *$conv(S)$ is a convex set.*

Proof. Let $\mathbf{x} = \sum_{i \in I} \lambda_i \mathbf{x}_i, \mathbf{y} = \sum_{j \in J} \mu_j \mathbf{y}_j \in conv(S)$. Let $K = I \cup J$ and put $\lambda_i = 0, i \in J \setminus I$ and $\mu_j = 0, j \in I \setminus J$. Then for $0 \leq \alpha \leq 1$ we see that

$$\alpha \mathbf{x} + (1-\alpha)\mathbf{y} = \sum_{i \in K} (\alpha \lambda_i \mathbf{x}_i + (1-\alpha)\mu_i \mathbf{y}_i) \text{ and } \sum_{i \in K} (\alpha \lambda_i + (1-\alpha)\mu_i) = 1$$

implying that $\alpha \mathbf{x} + (1-\alpha)\mathbf{y} \in conv(S)$ i.e. $conv(S)$ is convex. □

Lemma 3. *If S is convex, then $S = conv(S)$.*

Proof. Exercise. □

Corollary 4. *$conv(conv(S)) = conv(S)$ for all $S \subseteq \mathbb{R}^n$.*

Proof. Exercise. □

2.0.1 Extreme Points

A point $\mathbf{x}$ of a convex set S is said to be an *extreme point* if **THERE DO NOT EXIST** distinct $\mathbf{y}, \mathbf{z} \in S \setminus \{\mathbf{x}\}$ such that $\mathbf{x} \in L(\mathbf{y}, \mathbf{z})$. We let $ext(S)$ denote the set of extreme points of S.

EX1 If $n = 1$ and $S = [a, b]$ then $ext(S) = \{a, b\}$.

EX2 If $S = B(0, 1)$ then $ext(S) = \{\mathbf{x} : |\mathbf{x}| = 1\}$.

EX3 If $S = \{\mathbf{x} : A\mathbf{x} = \mathbf{b}\}$ is the set of solutions to a set of linear equations and $|S| \neq 1$ then $ext(S) = \emptyset$.

Extreme points and BFS's Let $C = \{\mathbf{x} : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq 0\}$ where $\mathbf{A}$ has full row rank. Then the extreme points of C are BFS's.
Suppose first that $\mathbf{x}$ is a BFS and that $\mathbf{x} = \lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2$ where $0 < \lambda < 1$. If $j \notin I$ then we have $0 = x_j = \lambda x_{1,j} + (1-\lambda)x_{2,j}$, which implies that $x_{1,j} = x_{2,j} = 0$ (remember that $\lambda, 1-\lambda > 0, x_{1,j}, x_{2,j} \geq 0$). This implies that $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{x}$.
Now suppose that $\mathbf{x}$ is not a BFS. Then we can can choose a basis $\mathbf{B}$ and partition $\mathbf{x}$ as $\mathbf{x_B} : \mathbf{x_N}$ where $\mathbf{x_N} \neq 0$. Let $I_0 = \{i \in I : x_i = 0\}$ and let $\mathbf{r}_i^T = \mathbf{e}_i^T\mathbf{B}^{-1}\mathbf{N}$ denote row i of $\mathbf{B}^{-1}\mathbf{N}$. Suppose first that there exists a non-zero $\boldsymbol{\xi} \in \mathbb{R}^N$ such that $\mathbf{r}_i^T\boldsymbol{\xi} = 0$ for $i \in I_0$. Then clearly $\mathbf{x} = (\mathbf{x}_1 + \mathbf{x}_2)/2$ where $\mathbf{x}_1 = \mathbf{x} - \varepsilon[0_\mathbf{B}, \mathbf{B}^{-1}\mathbf{N}\boldsymbol{\xi}]$ and $\mathbf{x}_2 = \mathbf{x} + \varepsilon[0_\mathbf{B}, \mathbf{B}^{-1}\mathbf{N}\boldsymbol{\xi}]$ for ε small enough that $\mathbf{x}_1, \mathbf{x}_2 \in C$. So, $\mathbf{x}$ is not an extreme point.
If $\boldsymbol{\xi}$ does not exist then we must have $|I_0| \geq n_1 = n - m$, the number of columns in $\mathbf{N}$. Furthermore, there must be an $n_1 \times n_1$ non-singular matrix $\mathbf{N}_1$ made up of rows $\mathbf{r}_i \in I_1 \subseteq I_0$. But then we see that in fact $\mathbf{x}$ is a BFS with basis $(I \setminus I_1) \cup J$. ($x_i, i \in I_1$ becomes non-basic at value 0, instead of basic at value 0.)

Optimal extreme points Assume that C is bounded. Then $\mathbf{z}$ is an extreme point of C iff there exists $\mathbf{c}$ such that $\mathbf{z}$ is the unique maximiser of $\mathbf{c}^T\mathbf{x}$ over points in $\mathbf{x} \in C$. Indeed, if $\mathbf{z} = \lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2$ where $0 < \lambda < 1$ then $\mathbf{c}^T\mathbf{z} \leq \max\left\{\mathbf{c}^T\mathbf{x}_1, \mathbf{c}^T\mathbf{x}_2\right\}$. Conversely, if $\mathbf{z}$ is an extreme point, BFS, let $\mathbf{B}$ be a basis matrix for $\mathbf{x}$. Then put $c_i = 0, i \in I$ and $c_j = -1, j \in J$. Then, $\mathbf{c}^T\mathbf{z} = 0$ and $\mathbf{c}^T\mathbf{x} = -\sum_{j\in J} x_j < 0$ if $\mathbf{x} \neq \mathbf{z}$.
This argument also shows that there is always an extreme point that yields a maximum of a linear function over C.

Neighboring extreme points Let $C = \{\mathbf{x} \geq 0 : \mathbf{Ax} = \mathbf{b}\}$. Two extreme points $\mathbf{v}_1, \mathbf{v}_2 \in C$ are said to be neighbors if there exists $\mathbf{c}$ such that $\mathbf{c}^T\mathbf{v}_1 = \mathbf{c}^T\mathbf{v}_2$ and $\mathbf{c}^T\mathbf{x} < \mathbf{c}^T\mathbf{v}_1$ for all $\mathbf{x} \in C \setminus L(\mathbf{v}_1, \mathbf{v}_2)$.

We show next that if $\mathbf{v}_1, \mathbf{v}_2$ are BFS's with basic index sets I_1, I_2 where $|I_1 \setminus I_2| = 1$ (i.e. one can be obtained from the other by a pivot) then $\mathbf{v}_1, \mathbf{v}_2$ are neighbors on C. Indeed, Put $c_i = 0, i \in I_1 \cup I_2$ and $c_j = -1, j \notin I_1 \cup I_2$. Then $\mathbf{c}^T\mathbf{v}_1 = \mathbf{c}^T\mathbf{v}_2 = 0$ and if $\mathbf{x} \notin L(\mathbf{v}_1, \mathbf{v}_2)$ then there exists $j \notin I_1 \cup I_2$ such that $x_j > 0$ implying that $\mathbf{c}^T\mathbf{x} < 0$.

Convex Hull Suppose that $S \subseteq \mathbb{R}^n$. Then the *convex hull* $\text{conv}(S)$ is defined

$$\text{conv}(S) = \left\{\sum_{i\in I} \lambda_i\mathbf{x}_i : (i)\ |I| < \infty,\ (ii)\ \sum_{i\in I}\lambda_i = 1,\ (iii)\ \lambda_i \geq 0, i \in I,\ (iv)\ \mathbf{x}_i \in S, i \in I\right\} = \bigcap_{\substack{C \supseteq S \\ C \text{ convex}}} C, \quad (47)$$

is the smallest convex set containing S. It follows as in O2, O3, that $\text{conv}(S)$ is always convex.

(Carathéodory's theorem implies that we can take $|I| \leq n+1$.)

Theorem 5. *If $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s\}$ is a finite set then conv(S) is a polyhedron. (Assume that dim$(S) = n$.)*

Proof. We show that

$$\text{conv}(S) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \leq \mathbf{b}\} = \bigcap_{i=1}^{m} \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_i^T \mathbf{x} \leq b_i\right\}, \tag{48}$$

where the half-spaces $\left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_i^T \mathbf{x} \leq b_i\right\}$ are obtained by, choosing $I \in \binom{[s]}{n}$ where $\mathbf{x}_i, i \in I$ are affinely independent and letting the hyperplane defined by $\mathbf{x}_i, i \in I$ be $\left\{\mathbf{x} : \mathbf{a}^T \mathbf{x} = b\right\}$ and letting $b_I = \max\left\{\mathbf{a}^T \mathbf{x}_j : j \in [s]\right\}$. This defines a polytope P containing S and hence conv(S).

Suppose that $P \not\subseteq$ *conv*(S), then there exists $\mathbf{x}_0 \in P$ and a hyperplane $H = \left\{\mathbf{x} : \mathbf{a}^T \mathbf{x} = b\right\}$ such that $\mathbf{a}^T \mathbf{x}_j \leq b, j \in [s]$ while $\mathbf{a}^T \mathbf{x}_0 > b$, see Section 2.1. We show that we can choose H so that it contains n affinely independent points $I \subseteq S$. This gives us a contradiction, $\mathbf{a}^T \mathbf{x}_0 > b_I$. Choose H to maximise the number of affinely independent points in S it contains and suppose that this is less than n. Suppose that H contains $\mathbf{x}_i, i = 1, 2, \ldots, m < n$. (We can assume that $m \geq 1$ because we can shift H until it touches *conv*(S).) Because $m < n$, we can find $\mathbf{h} \neq 0$ such that $\mathbf{h}^T(\mathbf{x}_0 - \mathbf{x}_1) = \mathbf{h}^T(\mathbf{x}_i - \mathbf{x}_1) = 0, i = 2, 3, \ldots, m$. (Note that $b = \mathbf{a}^T \mathbf{x}_1$.) Now let $H_\xi = \left\{\mathbf{x} : (\mathbf{a} + \xi\mathbf{h})^T(\mathbf{x} - \mathbf{x}_1) = 0\right\}$, so that $H = H_0$. Then for any ξ, we have $\mathbf{x}_i \in H_\xi, i = 1, 2, \ldots, m$ and $(\mathbf{a} + \xi\mathbf{h})^T(\mathbf{x}_0 - \mathbf{x}_1) > 0$. Replacing $\mathbf{h}$ by $-\mathbf{h}$ if necessary, assume that $S_+ = \left\{\mathbf{x} \in S : \mathbf{h}^T(\mathbf{x} - \mathbf{x}_1) > 0\right\}$ is non-empty. Then putting $\xi = \min_{\mathbf{x} \in S_+} \frac{-\mathbf{a}^T(\mathbf{x} - \mathbf{x}_1)}{\mathbf{h}^T(\mathbf{x} - \mathbf{x}_1)}$ we obtain a supporting separating hyperplane H_ξ containing an additional point of S, contradiction. □

Diameter of polytopes Given C there is a graph $G_C = (V, E)$ where $V = \{\text{extreme points of } C\}$ and $\{\mathbf{v}_1, \mathbf{v}_2\} \in E$ if $\mathbf{v}_1, \mathbf{v}_2$ are neighbors on C. The *diameter* of G_C is the maximum over pairs $\mathbf{v}_1, \mathbf{v}_2 \in V$ of the minimum number of edges in a path from $\mathbf{v}_1$ to $\mathbf{v}_2$ in G_C. As such the diameter of G_C is a lower bound on the maximum number of pivots for an LP based on C. (It takes at least this number of pivots if we start at $\mathbf{v}_1$ and $\mathbf{v}_2$ is the optimum.)

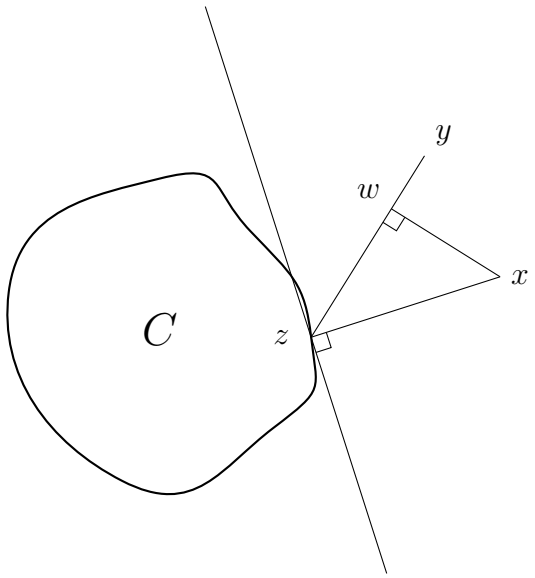
Diameter of $(0/1)$ **polytopes** It is not known whether or not the diameter of C is bounded by a polynomial in m, n, $n^{O(\log n)}$ is what we know at present. The n-cube $Q_n = \{0,1\}^n$ and $L_n = [0,1]^n = \text{conv}(Q_n)$. If $S \subseteq Q_n$, we let $P_S = \text{conv}(S)$. Many interesting polytopes are of this form. They are called $(0,1)$-polytopes as their extreme points are subsets of Q_n. In the special case of $(0,1)$-polytopes of dimension n, we can prove a bound of n on the diameter. Let P be a $(0,1)$ polytope and let $P^i = \{\mathbf{x} \in P : x_1 = i\}, i = 0, 1$. Then relies on

Fact 1 If $\mathbf{v}, \mathbf{w} \in P^0$ then they are neighbors in P^0 if and only if they are neighbors in P. (This because $v_1 = w_1 = 0$ implies that $x_1 = 0$ for all $\mathbf{x} \in L(\mathbf{v}, \mathbf{w})$ and we can use the $\mathbf{c}$ that tells us that they are neighbors in P.)

Fact 2 If $P \neq P^0$ and $\mathbf{x} \in P^0$ then $\mathbf{x}$ has a P-neighbor in P^1. (Assume w.l.o.g. that $\mathbf{x} = 0$ and that x_1 is a non-basic variable. Then increasing x_1 will lead to a neighboring vertex in P^1.
This needs a bit more explanation. We suppose that $P = \{\mathbf{x} \geq 0 : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ and define
$Q = \{(\mathbf{x}, \mathbf{y}) \geq 0 : \mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{b}\}$. We apply our argument to Q and argue that $\mathbf{x}$ is not a vertex of P iff $(\mathbf{x}, \mathbf{b} - \mathbf{A}\mathbf{x})$ is not a vertex of Q. We know that increasing x_1 leads to a vertex of Q and this will correspond to a vertex in P^1.

We can now prove that the diameter of G_P is at most n. We use induction on the dimension. This is trivial for dimension 1 and assume it is true for dimension $n-1$. Let $\mathbf{u}$ and $\mathbf{v}$ be the extremities of a path defining the diameter of P. If $\mathbf{u}$ and $\mathbf{v}$ belong to the same $P^i, i = 0$ then the result is immediate by induction. So assume that $\mathbf{u} \in P^0$ and $\mathbf{v} \in P^1$ and let $\mathbf{w} \in P^0$ be a vertex adjacent to $\mathbf{v}$. By our induction hypothesis a shortest path from $\mathbf{w}$ to $\mathbf{u}$ contains at most $n-1$ edges and therefore the diameter of P is at most n.

2.1 Separating Hyperplane



Theorem 6. *Let $C \neq \emptyset$ be a convex set in $\mathbb{R}^n$ and suppose $\mathbf{x} \notin C$. Then there exists $\mathbf{0} \neq \mathbf{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that (i) $\mathbf{a}^T\mathbf{x} \geq b$ and (ii) $C \subseteq \{\mathbf{y} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{y} \leq b\}$.*

Proof.
Case 1: C is closed.
Let $\mathbf{z}$ be the closest point in C to $\mathbf{x}$. Let $\mathbf{a} = \mathbf{x} - \mathbf{z} \neq \mathbf{0}$ and $b = (\mathbf{x} - \mathbf{z})^T\mathbf{z}$. Then

$$\mathbf{a}^T\mathbf{x} - b = (\mathbf{x} - \mathbf{z})^T\mathbf{x} - (\mathbf{x} - \mathbf{z})^T\mathbf{z} = |\mathbf{x} - \mathbf{z}|^2 > 0.$$

This verifies (i). Suppose (ii) fails and there exists $\mathbf{y} \in C$ such that $\mathbf{a}^T\mathbf{y} > b$. Let $\mathbf{w} \in C$ be the closest point to $\mathbf{x}$ on the line segment $L(\mathbf{y}, \mathbf{z}) \subseteq C$. The triangle formed by $\mathbf{x}, \mathbf{w}, \mathbf{z}$ has a right angle at $\mathbf{w}$ and an acute angle at $\mathbf{z}$. This implies that $|\mathbf{x} - \mathbf{w}| < |\mathbf{x} - \mathbf{z}|$, a contradiction.

Notice that here we have the stronger statement that (i) $\mathbf{a}^T\mathbf{x} > b$ and (ii) $C \subseteq \{\mathbf{y} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{y} \leq b\}$.

Case 2: $\mathbf{x} \notin \bar{C}$.
We observe that $\bar{C} \supseteq C$ and is convex (exercise). We can thus apply Case 1, with $\bar{C}$ replacing C.

Case 3: $\mathbf{x} \in \bar{C} \setminus C$. Every ball $B(\mathbf{x}, \delta)$ contains a point of $\mathbb{R}^n \setminus \bar{C}$ that is distinct from $\mathbf{x}$. Choose a sequence $\mathbf{x}_m, \notin \bar{C}, m \geq 1$ that tends to $\mathbf{x}$. For each m, let $\mathbf{a}_m, b_m$ define a hyperplane that separates $\mathbf{x}_m$ from $\bar{C}$, as in Case 2. We can assume that $|\mathbf{a}_m| = 1$ (scaling) and that b_m is in some bounded set and so there must be a convergent subsequence of $(\mathbf{a}_m, b_m), m \geq 1$ that converges to $(\mathbf{a}, b), |\mathbf{a}| = 1$. Assume that we re-label so that this subsequence is $(\mathbf{a}_m, b_m), m \geq 1$. Then for $\mathbf{y} \in \bar{C}$ we have $\mathbf{a}_m^T\mathbf{y} \leq b_m$ for all m. Taking limits we see that $\mathbf{a}^T\mathbf{y} \leq b$. Similarly, taking limits we see that $\mathbf{a}^T\mathbf{x} \geq b$. □

Corollary 7. *Suppose that $S, T \subseteq \mathbb{R}^n$ are convex and that $S \cap T = \emptyset$. Then there exists $\mathbf{a}, b$ such that $\mathbf{a}^T\mathbf{x} \leq b$ for all $\mathbf{x} \in S$ and $\mathbf{a}^T\mathbf{x} \geq b$ for all $\mathbf{x} \in T$.*

Proof. Let $W = S + (-1)T$. Then $\mathbf{0} \notin W$ and applying Theorem 6 we see that there exists $\mathbf{a}$ such that $\mathbf{a}^T\mathbf{z} \leq 0$ for all $\mathbf{z} \in W$. Now put

$$b = \frac{1}{2}\left(\sup_{\mathbf{x}\in S} \mathbf{a}^T\mathbf{x} + \inf_{\mathbf{x}\in T} \mathbf{a}^T\mathbf{x}\right).$$

□

Farkas Lemma We can reprove the Farkas lemma using hyperplane separation: Suppose then that (i) fails to hold. Let $S = \{\mathbf{y} : \mathbf{y} = \mathbf{A}\mathbf{x} \text{ for some } \mathbf{x} \geq \mathbf{0}\}$. Then $\mathbf{b} \notin S$ and since S is closed there exists $\boldsymbol{\alpha}, \beta$ such that (a) $\boldsymbol{\alpha}^T\mathbf{b} > \beta$ and (b) $\boldsymbol{\alpha}^T\mathbf{A}\mathbf{x} \leq \beta$ for all $\mathbf{x} \geq 0$. Now $\mathbf{0} \in S$ and so we have $\beta \geq 0$ and $(-\boldsymbol{\alpha})^T\mathbf{b} < 0$. Second, if there exists a column $\mathbf{a}_j$ of $\mathbf{A}$ such that $\boldsymbol{\alpha}^T\mathbf{a}_j > 0$ then choosing $x_j > \beta/\boldsymbol{\alpha}^T\mathbf{a}_j$ and $x_k = 0, k \neq j$ gives $\mathbf{x} \geq 0$ such that $\boldsymbol{\alpha}^T\mathbf{A}\mathbf{x} > \beta$, contradiction. This then implies that $(-\boldsymbol{\alpha})^T\mathbf{A} \geq 0$. Thus $-\boldsymbol{\alpha}$ satisfies (ii).

Theorem 8. *Let S be a closed, bounded convex set in $\mathbb{R}^n$. Then $S = conv(ext(S))$.*

Proof. We prove this by induction on the dimension n. For $n = 1$ the result is trivial, since then S must be an interval $[a, b]$. Inductively assume the result for dimensions less than n. Clearly, $S \supseteq T = conv(ext(S))$.

Suppose there exists $\mathbf{x} \in S \setminus T$.
Case 1: $\mathbf{x}$ lies on the boundary ∂S of S. Let $\mathbf{x} \in H = \left\{\mathbf{y} : \mathbf{a}^T\mathbf{y} = b\right\}$ be the hyperplane defined in Theorem 6, applied to $\mathbf{x}$ and the interior of S. (We can rule out $S \subseteq H$ because of the induction hypothesis.) We observe that if $\mathbf{w}$ is a vertex of $S^* = S \cap H$ then it is also a vertex of S. For if $\mathbf{w} = \lambda\mathbf{w}_1 + (1-\lambda)\mathbf{w}_2, \mathbf{w}_1, \mathbf{w}_2 \in S, 0 < \lambda < 1$ then we have $b = \mathbf{a}^T\mathbf{w} = \lambda\mathbf{a}^T\mathbf{w}_1 + (1-\lambda)\mathbf{a}^T\mathbf{w}_2 \leq \lambda b + (1-\lambda)b = b$. This implies that $\mathbf{a}^T\mathbf{w}_1 = \mathbf{a}^T\mathbf{w}_2 = b$ and so $\mathbf{w}_1, \mathbf{w}_2 \in S^*$, contradiction. Then we have by induction that $\mathbf{x}$ is in the convex hull of $ext(H \cap S)$ which is contained in T.

Case 2: $\mathbf{x} \notin \partial S$. By extending an arbitrary line through $\mathbf{x}$ in both directions we see that $\mathbf{x} = \lambda\mathbf{w}_1 + (1-\lambda)\mathbf{w}_2$ for $\mathbf{w}_1, \mathbf{w}_2 \in \partial S$ and the result follows from Case 1 applied to $\mathbf{w}_1, \mathbf{w}_2$. □

3 Primal-Dual Algorithms

Here is the general idea. Suppose we have the primal (3) and a solution $\mathbf{y}$ to the dual (5). Let $\mathbf{A}_\mathbf{y}$ be the submatrix of $\mathbf{A}$ formed by the columns $\mathbf{a}_j$ for which $c_j = \mathbf{a}_j^T\mathbf{y}$. If we can find $\mathbf{z} \geq 0$ such that $\mathbf{A}_\mathbf{y}\mathbf{z} = \mathbf{b}$ then complementary slackness will tell us that by padding out the missing components of $\mathbf{z}$ with zeroes to create $\mathbf{x}$ we will have that $\mathbf{x}, \mathbf{y}$ are optimal for (3), (5) respectively.

If $\mathbf{z}$ does not exist then the Farkas Lemma implies that there exists $\mathbf{u}$ such that $\mathbf{A}_\mathbf{y}^T\mathbf{u} \geq 0$ and $\mathbf{b}^T\mathbf{u} < 0$. We replace $\mathbf{y}$ by $\mathbf{y} + \varepsilon\mathbf{u}$. If $c_j = \mathbf{a}_j^T\mathbf{y}$ then $\mathbf{a}_j^T\mathbf{u} \geq 0$ and so $c_j \leq \mathbf{a}_j^T(\mathbf{y} + \varepsilon\mathbf{u})$ and if we ensure that ε is small enough such that $c_j \leq \mathbf{a}_j^T(\mathbf{y} + \varepsilon\mathbf{u})$ for those j for which $c_j < \mathbf{a}_j^T\mathbf{y}$ then $\mathbf{y} + \varepsilon\mathbf{u}$ is feasible for (5). Also, $\mathbf{b}^T(\mathbf{y} + \varepsilon\mathbf{u}) < \mathbf{b}^T\mathbf{y}$ and so the dual value has improved. In the following, we describe a specific LP for which this idea leads to a fast algorithm.

3.1 Primal-Dual Algorithm for the Assignment Problem

A *matching* M in a graph is a set of vertex disjoint edges. A vertex v is covered by M if there exists $e \in M$ such that $v \in e$. A matching M is *perfect* if every vertex of G is covered by M. For the complete bipartite graph $K_{A,B}$ on vertex set $A = \{a_i : i \in [n]\}, B = \{b_i : i \in [n]\}$, perfect matchings can be represented by permutations of n i.e $M = \{(a_i, b_{\pi(i)}) : i \in [n]\}$. Given a cost matrix $(c(i,j))$, the cost of a perfect matching $M = M(\pi)$ is given by

$$c(M) = \sum_{i=1}^{n} c(i, \pi(i)).$$

The assignment problem is that of finding a perfect matching of minimum cost.

Consider the linear program ALP:

$$\begin{aligned} &\text{Minimize} && \textstyle\sum_{i=1}^n \sum_{j=1}^n c_{i,j} x_{i,j} && (49)\\ &\text{Subject to} \\ &&& \textstyle\sum_{j=1}^n x_{i,j} = 1 \qquad \text{for } i = 1, 2, \ldots, n. && (50)\\ &&& \textstyle\sum_{i=1}^n x_{i,j} = 1 \qquad \text{for } j = 1, 2, \ldots, n. && (51)\\ &&& x_{i,j} \geq 0 \qquad \text{for } i, j = 1, 2, \ldots, n. && (52) \end{aligned}$$

The assignment problem is the solution to ALP where we replace (52) by

$$x_{i,j} = 0 \text{ or } 1 \text{ for } i, j = 1, 2, \ldots, n. \tag{53}$$

This is because (50), (51) force the set $\{(i,j) : x_{i,j} = 1\}$ to be a perfect matching and (49) is then the cost of this matching.

In general replacing non-negativity constraints (52) by *integer* constraints (53) makes an LP hard to solve. Not however in this case.

The dual of ALP is the linear program DLP:

$$\begin{aligned} &\text{Maximize} && \textstyle\sum_{i=1}^n u_i + \sum_{j=1}^n v_j && (54)\\ &\text{Subject to} \\ &&& u_i + v_j \leq c(i,j) \qquad \text{for } i, j = 1, 2, \ldots, n. && (55) \end{aligned}$$

The *primal-dual* algorithm that we describe relies on *complementary slackness* to find a solution.

Complementary Slackness: If a feasible solution $\mathbf{x}$ to ALP and a feasible solution $\mathbf{u}$,$\mathbf{v}$, to DLP satisfy

$$x_{i,j} > 0 \text{ implies that } u_i + v_j = c(i,j). \tag{56}$$

then $\mathbf{x}$ solves ALP and $\mathbf{u}$,$\mathbf{v}$, solves DLP. For then

$$0 = \sum_{i=1}^{n} \sum_{j=1}^{n} (c(i,j) - u_i - v_j) x_{i,j} = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j} - \left(\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \right), \tag{57}$$

and the two solutions have the same objective value.

(We have used $\sum_{i=1}^n u_i \sum_{j=1}^n x_{i,j} = \sum_{i=1}^n u_i$, which follows from (50) etc.)

The steps of the Primal-Dual algorithm are as follows:

Step 1 Choose an initial dual feasible solution. E.g. $v_j = 0, j \in [n]$ and $u_i = \min_j c(i,j)$.

Step 2 Given a dual feasible solution, $\mathbf{u}$,$\mathbf{v}$, define the graph $K_{\mathbf{u},\mathbf{v}}$ to be the bipartite graph with vertex set A, B and an edge (i,j) whenever $u_i + v_j = c(i,j)$.

Step 3 Find a maximum size matching M in $K_{\mathbf{u},\mathbf{v}}$.

Step 4 If M is perfect then (56) holds and M provides a solution to the assignment problem.

Step 5 If M is not perfect, update $\mathbf{u}$,$\mathbf{v}$ and go to Step 3.

To carry out Step 3, we proceed as follows:

Step 3a Begin with an arbitrary matching M of $K_{\mathbf{u},\mathbf{v}}$.

Step 3b Let A_U denote the set of vertices in A not covered by M.

Step 3c Let $\vec{K}_{\mathbf{u},\mathbf{v}}$ be the digraph obtained from $K_{\mathbf{u},\mathbf{v}}$ by orienting matching edges from B to A and other edges from A to B.

Step 3d Let A_M, B_M denote the set of vertices in A, B that are reachable by a path in $\vec{K}_{\mathbf{u},\mathbf{v}}$ from A_U. Such paths are necessarily alternating.

Step 3e If there is a vertex $b \in B_M$ that is not covered by M then there is an augmenting path P from some $a \in A_U$ to b. In this case we use P to construct a matching M' with $|M'| > |M|$. We then go to Step 3b, with M replaced by M'. Otherwise, Step 3 is finished.

To carry out Step 5, we assume that we have finished Step 3 with M, A_M, B_M. We then let

$$\theta = \min\{c_{i,j} - u_i - v_j : a_i \in A_M, b_j \notin B_M\} > 0.$$

We know that $\theta > 0$. Otherwise, if a_i, b_j is the minimising pair, then we should have put $b_j \in B_M$.

We then amend $\mathbf{u}$,$\mathbf{v}$ to $\mathbf{u}^*, \mathbf{v}^*$ via

$$u_i^* = \begin{cases} u_i + \theta & a_i \in A_M. \\ u_i & \text{Otherwise.} \end{cases} \text{ and } v_j^* = \begin{cases} v_j - \theta & j \in B_M. \\ v_j & \text{Otherwise.} \end{cases}$$

Observe the following:

1. $\mathbf{u}^*, \mathbf{v}^*$ is feasible for DLP. $u_i^* + v_j^* \le u_i + v_j$ except for the case where $a_i \in A_M, b_j \notin B_M$ and θ is chosen so that the increase maintains feasibility.
2. If $b \in B_M$ for the pair $\mathbf{u}$,$\mathbf{v}$ then it will stay in B_M when we replace $\mathbf{u}$,$\mathbf{v}$ by $\mathbf{u}^*, \mathbf{v}^*$. This is because there is a path $P = (a_{i_1} \in A_U, b_{i_1}, \ldots, a_{i_k}, b_{i_k} = b)$ such that each edge of P contains one vertex in A_M and one vertex in B_M. Hence the sum $u_i + v_j$ is unchanged for edges along P.
3. A vertex $b \notin B_M$ contained in a pair that defines θ will be in B_M when we replace $\mathbf{u}$,$\mathbf{v}$ by $\mathbf{u}^*, \mathbf{v}^*$.

In summary: if we reach Step 4 with a perfect matching then we have solved ALP. After at most n changes of $\mathbf{u}$,$\mathbf{v}$ in Step 5, the size of M increases by at least one. This is because updating $\mathbf{u}$,$\mathbf{v}$ increases B_M by at least one. Thus the algorithm finishes in $O(n^4)$ time. ($O(n^3)$ time if done carefully.)

Note also that $|A_M| > |B_M|$ and so the dual objective increases in Step 5.

Total Unimodularity Notice that the above algorithm solves the LP (49) and satisfies the integrality constraints (53). The reason for this is that all of the extreme points of the feasible region defined by (50), (51) and (52) are integral. The reason for this is that the constraint matrix of (49) is *totally unimodular*. A matrix $\mathbf{A}$ is totally unimodular if every square submatrix of $\mathbf{A}$ has determinant $0, \pm 1$. This for example implies that if $\mathbf{B}$ is a basis matrix of $\mathbf{A}$ then the entries of $\mathbf{B}^{-1}$ are also $0, \pm 1$ and so $\mathbf{B}^{-1}\mathbf{b}$ is integer for every integer $\mathbf{b}$. Recall that $\mathbf{B}^{-1} = (adjoint\ \mathbf{B})/det\mathbf{B}$ where *adjoint* $\mathbf{B}$ is the matrix whose (i,j)th entry is equal to $(-1)^{i+j}$ times the determinant of the submatrix of $\mathbf{B}$ obtained by deleting row j and column i.

Vertex-edge incidence matrix The matrix of coefficients for the LP (49) has the following property: the rows can be divided into two sets X, Y. Each column has two non-zeros, a 1 in a row of X and a 1 in a row of Y. We argue by induction on s that an $s \times s$ submatrix of $\mathbf{A}$ has determinant $0, \pm 1$. This is trivially true if $s = 1$ and so assume it is true for $1 \leq t < s$. Let $\mathbf{C}$ be an $s \times s$ sub-matrix. If $\mathbf{C}$ has a column of 0's then $\mathbf{C}$ has determinant 0. If $\mathbf{C}$ has a column with at most one 1 then we can expand its determinant by this column and use induction. If all columns have two 1's then the sum of $\mathbf{C}$'s rows in X equals the sum of $\mathbf{C}$'s rows in Y and so $\mathbf{C}$ is singular and has determinant 0.

4 Stable marriage

We are given two sets X, Y of size n and for each $x \in X$ there is a permutation ρ_x of Y and similarly for each $y \in Y$ there is a permutation of X. The two classic examples are

(i) X is a set of women and Y is a set of men. Each $x \in X$ ranks the men in Y by attraction, $\rho_x(1) > \rho_x(2) > \cdots > \rho_x(n)$. Here $\rho_x(i) = j$ means that man i is the jth most attractive. Similarly for each $y \in Y$. Let $\sigma_x = \rho_x^{-1}$.

(ii) X is a set of hospital internships and Y is a set of aspiring doctors.

Thinking in terms of the complete bipartite graph $K_{X,Y}$, a perfect matching $\{\{x, \mu(x)\} : x \in X\}$ is *stable* if **there does not exist** an unstable pair x, y: here $y = \rho_x(a_1), \mu(x) = \rho_x(a_2), a_1 < a_2$ and $x = \rho_y(b_1), \mu^{-1}(y) = \rho_y(b_2),\ b_1 < b_2$. I.e. x prefers y to their currently matched partner and vice-versa.

Theorem 9 (Gale-Shapley)**.** *There is always at least one stable matching and one can be found efficiently.*

Proof. The construction of the matching goes in rounds. At the start of round t there is a matching M_t that covers $X_t \subseteq X, Y_t \subseteq Y$. In addition $\phi_t(x), x \in X$ denotes the number of proposals made by x so far. These will have been to $\sigma_x(1), \sigma_x(2), \ldots, \sigma_x(\phi_t(x))$. And $\psi_t(y), y \in Y$ denotes the rank of y's partner in M_t, if $y \in Y_t$. Otherwise, $\psi_t(y) = n + 1$. Initially, $X_0 = Y_0 = \emptyset$ and the process continues until $X_t = X, Y_t = Y$.

Step 1 Each $x \notin X_t$ proposes to the next member of Y on their list, i.e. proposes to $\rho_x(\phi_t(x) + 1)$ and $\phi_{t+1}(x) \leftarrow \phi_t(x) + 1$.

Step 2 For each $y \in Y$ we do the following: suppose first that $y \in Y_t$. y compares all the proposals $S(y,t) = \{x : \rho_x(\phi_t(x) + 1) = y\}$. Suppose that z is the most attractive member of $S(y,t)$. If $\rho_y(z) < \psi(y) \leq n$, i.e. $y = y_i$ prefers z to their currently matched partner x_i then we replace $\{x_i, y_i\}$ by $\{z, y\}$ and then $X_{t+1} \leftarrow (X_t \cup \{z\}) \setminus \{x_i\}$. If $y \notin Y_t$ then $X_{t+1} \leftarrow (X_t \cup \{z\}$ and $Y_{t+1} \leftarrow (Y_t \cup \{y\}$ and $\{z, y\}$ is added to M_t.

Example: Consider a set of four proposers $P = \{1, 2, 3, 4\}$ and four receivers $R = \{A, B, C, D\}$. We apply the Proposer-Oriented Version of the Gale-Shapley Algorithm.

Proposers' Preferences (highest to lowest rank):

- **1:** $A > B > C > D$
- **2:** $B > A > C > D$
- **3:** $A > B > C > D$
- **4:** $B > A > D > C$

Receivers' Preferences (highest to lowest rank):

- **A:** $2 > 3 > 1 > 4$
- **B:** $1 > 2 > 3 > 4$
- **C:** $3 > 1 > 2 > 4$
- **D:** $1 > 2 > 3 > 4$

Execution Steps

- **Round 1:**
  - **1** proposes to **A**. (A tentatively accepts 1; current pair: $(A, 1)$)
  - **2** proposes to **B**. (B tentatively accepts 2; current pair: $(B, 2)$)
  - **3** proposes to **A**. *Conflict at A:* A prefers 3 over 1. A accepts 3 and rejects 1.
  - **4** proposes to **B**. *Conflict at B:* B prefers 2 over 4. B keeps 2 and rejects 4.
  - *Unmatched Proposers:* $\{1, 4\}$
- **Round 2:**
  - **1** proposes to next choice **B**. *Conflict at B:* B prefers 1 over current partner 2. B accepts 1 and rejects 2.
  - **4** proposes to next choice **A**. *Conflict at A:* A already has 3. A prefers 3 over 4. A rejects 4.
  - *Unmatched Proposers:* $\{2, 4\}$
- **Round 3:**
  - **2** proposes to next choice **A**. *Conflict at A:* A already has 3. A prefers 2 over 3. A accepts 2 and rejects 3.
  - **4** proposes to next choice **D**. (D is free; tentatively accepts 4; current pair: $(D, 4)$)
  - *Unmatched Proposers:* $\{3\}$
- **Round 4:**

- **3** proposes to next choice **B**. *Conflict at B:* B already has 1. B prefers 1 over 3. B rejects 3.
- *Unmatched Proposers:* $\{3\}$

- **Round 5:**
  - **3** proposes to next choice **C**. (C is free; tentatively accepts 3; current pair: $(C, 3)$)
  - *Unmatched Proposers:* $\emptyset$

Final stable matching: $(1, B)$ $(2, A)$ $(3, C)$ $(4, D)$.

There can be at most n^2 rounds. Each of the n members of X can be rejected up to $n-1$ times before their final accepted match, leading to a maximum of $n(n-1)+1$ total proposals. If $x \in X$ was rejected n times then all $y \in Y$ are matched but x isn't, impossible.

Let $M_n = \{\{x, \mu(x)\} : x \in X\}$. Assume for the sake of a contradiction, that there is an unstable pair (x, y). This means that x prefers y to $\mu(x)$ and y prefers x to $\mu^{-1}(y)$. Because y comes before $\mu(x)$ in x's list, this means at some point during the run of the algorithm y must have been "proposed to" by x. So, y had the opportunity to keep x or potentially reject x for better and better matches. But y ended up with a match that's worse than x; this is a contradiction, since y's matches must have only gotten better throughout the algorithm. □

X-optimality Let $M = \{\{x, \mu(x)\}\}$ be a stable matching. We say that it is X-optimal if for any other stable matching $\widehat{M} = \{\{x, \widehat{\mu}(x)\}\}$ and any $x \in X$, $\sigma_x(\mu(x)) \leq \sigma_x(\widehat{\mu}(x))$. (Note that a priori it is not clear that X-optimal stable matchings even exist.)

Theorem 10. *The output of the above algorithm is X-optimal.*

Proof. . For each $x \in X$, let $\mu^*(x)$ be the most desirable among anything x can be matched to in any stable matching. Assume by contradiction that the above algorithm is not X-optimal. This means some $x \in X$ gets rejected by her best possible choice $y^* = \mu^*(x)$ at some point during the algorithm. Consider the first time during the run of the algorithm where this happens. This means that y^* rejects x because some $\widehat{x} \in X$ tries to match with y^* and y^* prefers $\widehat{x}$ over x. Now there are two cases:
Case 1: $\mu^*(x) = \mu^*(\widehat{x})$**:**
Now y^* prefers $\widehat{x}$ over x and $\widehat{x}$ prefers y^* to any other match in any stable matching. Therefore, $\widehat{x}, y^*$ will be unstable in any matching where they are not paired. But this contradicts the definition of y^*.

Case 2: $\mu^*(x) \neq \mu^*(\widehat{x})$**:**
In this case $\widehat{x}$ would have first proposed to $\mu^*(\widehat{x})$ before proposing to y^*. If $\widehat{x}$ proposes to y^* instead, it must have been rejected by $\mu^*(\widehat{x})$ in an earlier round. But this is a contradiction to x being the first to be rejected by their best feasible choice of $y \in Y$. □

Y-pessimality Let $M = \{\{\nu(y), y\}\}$ be a stable matching. We say that it is Y-pessimal if for any other stable matching $\widehat{M} = \{\{\widehat{\nu}(y), y\}\}$ and any $y \in Y$, $\sigma_y(\nu(y)) \geq \sigma_y(\widehat{\nu}(y))$. (Note that a priori it is not clear that Y-pessimal stable matchings even exist.)

Theorem 11. *The output of the above algorithm is Y-pessimal.*

Proof. Suppose there exists $y \in Y$ that is matched with $x \in X$ by the algorithm and yet there is another stable matching σ where y is matched with x' where y prefers x to x'. Let x be matched with $y' \neq y$ in σ. Then because the algorithm produces an X-optimal matching, we know that x prefers y to y'. Thus x, y form an unstable pair in σ, contradiction. □

5 Game theory

5.1 Two person zero-sum games

We discuss here an application of linear programming to the theory of games. This theory is an attempt to provide an analysis of situations involving conflict and competition. *Solving the game* means determining the optimal strategies for the players.

Game 1: There are two players A and B and to play the game they each choose a number 1,2,3 or 4 without the other's knowledge and then they both simultaneously announce their numbers. If A calls i and B calls j then B pays A $a_{i,j}$ – the *payoff* – given in the matrix below. (if $a_{i,j} < 0$, this is equivalent to A paying B $-a_{i,j}$.)

$$\begin{bmatrix} 2 & 4 & 2 & 1 \\ -2 & 5 & 1 & -1 \\ 1 & -5 & 3 & 0 \\ 6 & 2 & -3 & -2 \end{bmatrix}$$

This is a *two person zero-sum game*, zero sum because the algebraic sum of the player's winnings is always zero.

Game 2: (Penalty kicks) Suppose that A and B play the following game of soccer. A plays in goal and B takes penalty kicks. B can kick the ball into the left hand corner, the right hand corner or into the middle. If A guesses correctly where B will kick then A will make a save. The payoff to A is given by the following matrix.

$$\begin{bmatrix} & KR & KL & KM \\ DR & 2 & -1 & -2 \\ DL & -1 & 2 & -2 \\ M & -1 & -1 & 1 \end{bmatrix}$$

We will be considering $m \times n$ generalisations of Game 1 and other games like Game 2 that can be reduced to this form.

Thus there is given some $m \times n$ payoff $\mathbf{A}$. In a *play* of the game, A chooses $i \in M = \{1, 2, \ldots, m\}$ and B chooses $j \in N = \{1, 2, \ldots, n\}$. These choices are made independently without either player knowing what the other has chosen. They then announce their choices and B pays $a_{i,j}$ to A.

M, N will be referred to as the sets of *tactics* for A,B respectively.

A *match* is an unending sequence of plays. A's objective is to maximize her expected winnings from the match and B's objective is to minimize his expected loss.

A *strategy* for the match is some rule for selecting the tactic for the next play.

Let S_A, S_B be sets of strategies for A, B respectively. We shall initially consider the case where $S_A =$

$\{(1),\ldots,(m)\}$ and $S_B = \{(1),\ldots,(n)\}$ where (t) is the *pure* strategy of using tactic t in each play We Shall subsequently be enlarging S_A and S_B and we therefore introduce new notation to allow for this possibility.

Thus for each $u \in S_A$ and $v \in S_B$ let $PAY(u,v)$ denote the average payment of B to A.

Stable Solutions $(u_0, v_0) \in S_A \times S_B$ is a *stable solution* if

$$PAY(u, v_0) \leq PAY(u_0, v_0) \leq PAY(u_0, v) \tag{58}$$

holds for all u, v.

If (58) holds then neither A nor B has any incentive to change strategy if each assumes his opponent is not going to change his or hers.

The subsequent analysis is concerned with finding a stable solution.

Thinking of S_A as the row indices and S_B as the column indices of some matrix we define

$$ROWMIN(u) = \min_{v \in S_B} PAY(u,v), \qquad u \in S_A.$$
$$COLMAX(v) = \max_{u \in S_A} PAY(u,v), \qquad v \in S_B.$$

Suppose now that A chooses u. We assume that after some finite time, B will be able to deduce this choice. B will then choose his strategy v to minimize $PAY(u,v)$. Thus if A chooses u then she can expect her average winnings to be $ROWMIN(u)$.

Similarly if B chooses v he can expect his average losses to be $COLMAX(v)$.

Thus if $P_A = ROWMIN(u_0) = \max_{u \in S_A} ROWMIN(u)$ and $P_B = COLMAX(v_0) = \min_{v \in S_B} COLMAX(v)$ then A can by choosing u_0 ensure that her average winnings are at least P_A and B by choosing v_0 can ensure that his losses are at most P_B. If $P_A = P_B$ then this seems to solve the game but is $P_A = P_B$ always?

Theorem 12.

(a) $P_A \leq P_B$.

(b) $S_A \times S_B$ *contains a stable solution iff* $P_A = P_B$.

Proof. (a)

$$P_A = ROWMIN(u_0) \leq PAY(u_0, v_0) \leq COLMAX(v_0) = P_B. \tag{59}$$

(b) Suppose first that (u_0, v_0) is stable. Then, from (58), we have

$$COLMAX(v_0) = PAY(u_0, v_0) = ROWMIN(u_0)$$

and hence

$$P_B \leq COLMAX(v_0) = ROWMIN(u_0) \leq P_A,$$

which from (a) implies that $P_A = P_B$.

Conversely, if $P_A = P_B$ then from (59) we deduce that

$$ROWMIN(u_0) = PAY(u_0, v_0) = COLMAX(v_0)$$

which implies (58). □

We now consider specifically the case $S_A = \{(1), \ldots, (m)\}$ and $S_B = \{(1), \ldots, (n)\}$.

For Game one we have $P_A = P_B = 1 = a_{1,4}$ and hence A plays 1 and B plays 4 solves the game and A can guarantee to win at least 1 and B can guarantee to lose at most 1 on average.

The matrix of this game is said to have a *saddle point* (i_0, j_0) which means that $(i_0), (j_0)$ satisfies (58).

For a game whose matrix does not have a saddle point things are more complex. Consider for example Game two. $P_A = -1$ and $P_B = 1$. It follows from Theorem 12 that no pair of pure strategies solves the game. A knows she can average at least -1 by playing (3) and B knows he need lose no more than 1 on average by playing (3) but note that if A plays (3) then B has an incentive to play (1) or (2) but if he plays (1) then A will play (1) and so on.

Mixed strategies:
To break this seeming deadlock we allow the players to choose mixed strategies. A mixed strategy for A is a vector of probabilities $\mathbf{p} = (p_1, \ldots, p_m)$ where $p_i \geq 0$ for $i \in M$ and $p_1 + \cdots + p_m = 1$. A then chooses tactic i with probability p_i for $i \in M$ i.e. before each play A carries out a statistical experiment that has an outcome $i \in M$ with probability p_i. A then plays the corresponding tactic. Similarly B's mixed strategies are vectors $\mathbf{q} = (q_1, \ldots, q_n)$ satisfying $q_j \geq 0, j \in N$ and $q_1 + \cdots + q_n = 1$.

Pure strategies can be represented as vectors with a single non-zero component equal to 1. We now enlarge S_A, S_B to

$$\begin{aligned} S_A &= \{\mathbf{p} \in \mathbb{R}^m : \mathbf{p} \geq 0 \text{ and } p_1 + \cdots + p_m = 1\}. \\ S_B &= \{\mathbf{q} \in \mathbb{R}^n : \mathbf{q} \geq 0 \text{ and } q_1 + \cdots + q_n = 1\}. \end{aligned} \tag{60}$$

We now show using the duality theory of linear programming that $S_A \times S_B$ as defined in (60) contains a stable solution.

We shall first show how to compute P_A. Let $c_j(\mathbf{p}) = \sum_{i \in M} a_{i,j} p_i$. Then

$$P_A = \max_{\mathbf{p} \in S_A} \left(\min_{\mathbf{q} \in S_B} \sum_{j=1}^{n} c_j(\mathbf{p}) q_j \right) \tag{61}$$

Lemma 13.

$$\min_{\mathbf{q} \in S_B} \sum_{j=1}^{n} \xi_j q_j = \min \{\xi_1, \ldots, \xi_n\}. \tag{62}$$

Proof. Let $\xi_t = \min \{\xi_1, \ldots, \xi_n\}$ and let L be the LHS of (62). Putting $\widehat{q}_j = 0$ for $j \neq t$ and $\widehat{q}_t = 1$ we have $\widehat{\mathbf{q}} \in S_B$ and $\sum_{j=1}^{n} \xi_j \widehat{q}_j = \xi_t$. Thus $L \leq \xi_t$. However, for any $\mathbf{q} \in S_B$,

$$\sum_{j=1}^{n} \xi_j q_j \geq \sum_{j=1}^{n} \xi_t q_j = \xi_t \sum_{j=1}^{n} q_j = \xi_t.$$

□

It follows from the lemma and (61) that

$$\begin{aligned} P_A &= \max_{\mathbf{p}\in S_A} \min\{c_1(\mathbf{p}),\ldots,c_n(\mathbf{p})\} \\ &= \max \min\{c_1(\mathbf{p}),\ldots,c_n(\mathbf{p})\} \end{aligned}$$

Subject to

$$\begin{aligned} & p_1+\cdots+p_m = 1 \\ & p_1,\ldots,p_m \geq 0 \\ &= \max\ \xi \end{aligned} \tag{63}$$

Subject to

$$\begin{aligned} & \xi \leq \sum_{i=1}^{m} a_{i,j}p_i, \qquad j=1,\ldots,n \\ & p_1+\cdots+p_m = 1 \\ & p_1,\ldots,p_m \geq 0 \end{aligned}$$

Using similar arguments we can show that

$$P_B = \quad \min\ \eta \tag{64}$$

Subject to

$$\begin{aligned} & \eta \geq \sum_{j=1}^{n} a_{i,j}q_j, \qquad i=1,\ldots,m \\ & q_1+\cdots+q_n = 1 \\ & q_1,\ldots,q_n \geq 0 \end{aligned}$$

We note next that (63), (64) are a pair of dual linear programs. They are both feasible and hence $P_A = P_B$ and stable solutions exist. In fact if $\mathbf{p}^0$ solves (63) and $\mathbf{q}^0$ solves (64) $(\mathbf{p}^0, \mathbf{q}^0)$ is stable as $PAY(\mathbf{p}^0, \mathbf{q}^0) = P_A = P_B$. The *value* of the game is $P_A = P_B$.

Random payoff: We note that the above analysis goes through unchanged if A, B having selected tactics I, J, the payoff to A is a random variable whose expected value is $a_{I,J}$.

The above LP formulations can be slightly simplified. We can assume that $a_{i,j} > 0$ for all i, j. This is because adding a positive constant c to each entry of A will not change the optimal strategies. It will merely increase the optimal payoff by c. It follows that the maximum value of ξ in (63) is positive. We can therefore replace p_i by $x_i = p_i/\xi$ and then we find

$$P_A^{-1} = \text{Minimum } \xi^{-1} = \text{Minimum } \sum_{i=1}^{m} x_i \text{ subject to } \sum_{i=1}^{m} a_{i,j}x_i \geq 1 \text{ for all } j,\ x_i \geq 0 \text{ for all } i. \tag{65}$$

$$P_B^{-1} = \text{Maximum } \eta^{-1} = \text{Maximum } \sum_{j=1}^{n} y_j \text{ subject to } \sum_{j=1}^{n} a_{i,j}y_j \leq 1 \text{ for all } i,\ y_j \geq 0 \text{ for all } j. \tag{66}$$

5.1.1 Dominance

If $A(i,j) \geq A(i,j')$ for all i then player B will never use strategy j. It is preferable for her/him to use strategy j' instead. So, column j can be removed from the matrix A.

Similarly, if $A(i,j) \leq A(i',j)$ for all j then player A will never use strategy i. It is preferable for her/him to use strategy i' instead. So, row i can be removed from the matrix A.

Repeated use of this idea can reduce a game substantially.

5.1.2 Latin Square Game

Suppose that every row sum is equal to $R > 0$ and every column sum is equal to $C > 0$ where $mR = nC$. Then both players can choose uniformly. Consider the two LP's (65), (66) that solve the game: putting $x_i = 1/C$ and $y_j = 1/R$ gives two feasible solutions with the same objective value.

5.1.3 Non-singular games

Suppose that $\mathbf{A}$ is non-singular and that $\mathbf{1^T A^{-1} 1} > \mathbf{0}$. Then, $\mathbf{x}^T = \mathbf{1^T A^{-1}}$ and $\mathbf{y} = \mathbf{A^{-1} 1}$ solve (65), (66) respectively, if they are non-negative. The value of the game is $V = 1/\mathbf{1^T A^{-1} 1}$.

5.1.4 Symmetric games

A game is symmetric if $\mathbf{A}^T = -\mathbf{A}$ i.e if A is anti-symmetric. Then the game has value 0. If A and B both use strategy $\mathbf{p}$ then because $\mathbf{p}^T A \mathbf{p} = 0$ for anti-symmetric $\mathbf{A}$, we see that $PAY(\mathbf{p}, \mathbf{p}) = 0$. This implies that $0 \geq P_A = P_B \geq 0$.

5.2 Nash equilibrium

We give a proof of Nash's Theorem based on the Fixed Point Theorem of L. E. J. Brouwer. Given a set C and a mapping T of C into itself, a point $z \in C$ is said to be a fixed point of T, if $T(z) = z$.

Brouwer's Fixed Point Theorem. *Let C be a nonempty, compact, convex set in a finite dimensional Euclidean space, and let T be a continuous map of C into itself. Then there exists a point $z \in C$ such that $T(z) = z$.*

Now consider a finite n-person game. The pure strategy sets are denoted by $X_1, \ldots, X_n$, with X_k consisting of $m_k \geq 1$ elements, say $X_k = \{1, \ldots, m_k\}$. The space of mixed strategies of Player k is given by X_k^*,

$$X_k^* = \left\{ p_k = (p_{k,1}, \ldots, p_{k,m_k}) : p_{k,i} \geq 0 \text{ for } i = 1, \ldots, m_k, \text{and } \sum_{i=1}^{m_k} p_{k,i} = 1 \right\}. \tag{1}$$

For a given joint pure strategy selection, $x = (i_1, \ldots, i_n)$ with $i_j \in X_j$ for all j, the payoff, or utility, to Player k is denoted by $u_k(i_1, \ldots, i_n)$ for $k = 1, \ldots, n$. For a given joint mixed strategy selection, $(p_1, \ldots, p_n)$ with $p_j \in X_j^*$ for $j = 1, \ldots, n$, the corresponding expected payoff to Player k is given by $g_k(p_1, \ldots, p_n)$,

$$g_k(p_1, \ldots, p_n) = \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} p_{1,i_1} \cdots p_{n,i_n} u_k(i_1, \ldots, i_n). \tag{2}$$

Let us use the notation $g_k(p_1, \ldots, p_n \mid i)$ to denote the expected payoff to Player k if Player k changes strategy from p_k to the pure strategy $i \in X_k$,

$$g_k(p_1, \ldots, p_n \mid i) = g_k(p_1, \ldots, p_{k-1}, \delta_i, p_{k+1}, \ldots, p_n). \tag{3}$$

where δ_i represents the probability distribution giving probability 1 to the point i. Note that $g_k(p_1, \ldots, p_n)$ can be reconstructed from the $g_k(p_1, \ldots, p_n \mid i)$ by

$$g_k(p_1, \ldots, p_n) = \sum_{i=1}^{m_k} p_{k,i} g_k(p_1, \ldots, p_n \mid i). \tag{4}$$

A vector of mixed strategies, $(p_1, \ldots, p_n)$, is a strategic equilibrium if for all $k = 1, \ldots, n$, and all $i \in X_k$,

$$g_k(p_1, \ldots, p_n \mid i) \le g_k(p_1, \ldots, p_n). \tag{5}$$

Theorem. *Every finite n-person game in strategic form has at least one strategic equilibrium.*

Proof. For each k, X_k^* is a compact convex subset of m_k dimensional Euclidean space, and so the product, $C = X_1^* \times \cdots \times X_n^*$, is a compact convex subset of a Euclidean space of dimension $\sum_{i=1}^n m_i$. For $z = (p_1, \ldots, p_n) \in C$, define the mapping $T(z)$ of C into C by

$$T(z) = z' = (p_1', \ldots, p_n') \tag{6}$$

where

$$p_{k,i}' = \frac{p_{k,i} + \max\big(0, g_k(p_1, \ldots, p_n \mid i) - g_k(p_1, \ldots, p_n)\big)}{1 + \sum_{j=1}^{m_k} \max\big(0, g_k(p_1, \ldots, p_n \mid j) - g_k(p_1, \ldots, p_n)\big)}. \tag{7}$$

Note that $p_{k,i} \ge 0$, and the denominator is chosen so that $\sum_{i=1}^{m_k} p_{k,i}' = 1$. Thus $z' \in C$. Moreover the function $f(z)$ is continuous since each $g_k(p_1, \ldots, p_n)$ is continuous. Therefore, by the Brouwer Fixed Point Theorem, there is a point, $z' = (q_1, \ldots, q_n) \in C$ such that $T(z') = z'$. Thus from (7)

$$q_{k,i} = \frac{q_{k,i} + \max\big(0, g_k(z' \mid i) - g_k(z')\big)}{1 + \sum_{j=1}^{m_k} \max\big(0, g_k(z' \mid j) - g_k(z')\big)}. \tag{8}$$

for all $k = 1, \ldots, n$ and $i = 1, \ldots, m_n$. Since from (4) $g_k(z')$ is an average of the numbers $g_k(z' \mid i)$, we must have $g_k(z' \mid i) \le g_k(z')$ for at least one i for which $q_{k,i} > 0$, so that $\max(0, g_k(z' \mid i) - g_k(z')) = 0$ for that i. But then (8) implies that

$$\sum_{j=1}^{m_k} \max\big(0, g_k(z' \mid j) - g_k(z')\big) = 0,$$

so that $g_k(z' \mid i) \le g_k(z')$ for all k and i. From (5) this shows that $z' = (q_1, \ldots, q_n)$ is a strategic equilibrium.■

Remark. From the definition of $T(z)$, we see that $z = (p_1, \ldots, p_n)$ is a strategic equilibrium if and only if z is a fixed point of T. In other words, the set of strategic equilibria is given by $\{z : T(z) = z\}$. If we could solve the equation $T(z) = z$ we could find the equilibria. Unfortunately, the equation is not easily solved.

5.3 Auctions

Bids are made on a single item. There are N bidders. Each bidder has their own private valuation of the item for sale.

5.3.1 Auction Formats

English auction. Bids increase. The last person to bid wins and pays the last bid.

Dutch auction. Start at a high price. The first bidder to bid wins and pays that bid.

Sealed-bid first-price auction. Bids are placed in envelopes. The highest bidder wins and pays their bid.

Sealed-bid second-price auction. Bids are placed in envelopes. The highest bidder wins and pays the second-highest bid.

The English auction and the sealed-bid second-price auction are essentially the same. It is optimal to bid one's valuation. More discussion will be given later.

The Dutch auction and the sealed-bid first-price auction are also essentially equivalent.

5.3.2 Analysis of Symmetric Nash Equilibria

We analyze symmetric Nash equilibria in the sealed-bid first-price and second-price auctions, (c) and (d).

Bidder i values the object at x_i, where $\mathbb{P}(x_i \le x) = F(x)$, $x \in [0, \bar{w}]$. Bidder i uses a bidding strategy $\beta_i(x_i)$ and bids b_i. The equilibrium is *symmetric* if $\beta_1 = \beta_2 = \cdots = \beta_N$.

Second-Price Auction The payoff to bidder i is

$$\pi_i = \begin{cases} x_i - \max\limits_{j \neq i} b_j, & b_i > \max\limits_{j \neq i} b_j, \\ 0, & \text{otherwise.} \end{cases}$$

Proposition 14. *The strategy $\beta^{\text{II}}(x) = x$ is a dominant strategy.*

Proof. Fix bidder 1 and put

$$p_1 = \max_{j \neq 1} b_j.$$

First consider a deviation to a bid $z_1 < x_1$. The payoffs from bidding z_1 and from bidding x_1 are as follows:

	$x_1 > z_1 \ge p_1$	$p_1 > x_1 > z_1$	$x_1 > p_1 > z_1$
Payoff from bidding z_1	$x_1 - p_1$	0	0
Payoff from bidding x_1	$x_1 - p_1$	0	$x_1 - p_1$

Thus bidding x_1 is at least as good as bidding $z_1 < x_1$.

Now consider a deviation to a bid $z_1 > x_1$:

	$z_1 > x_1 \geq p_1$	$p_1 > z_1 > x_1$	$z_1 > p_1 > x_1$
Payoff from bidding z_1	$x_1 - p_1$	0	$x_1 - p_1$
Payoff from bidding x_1	$x_1 - p_1$	0	0

In the last case $x_1 - p_1 < 0$, so bidding x_1 is again at least as good as bidding z_1. Hence truthful bidding is a dominant strategy. □

Let

$$y_1 = \max\{x_2, x_3, \ldots, x_N\}.$$

Then, assuming that bids are made independently,

$$\mathbb{P}(y_1 \leq y) = F(y)^{N-1} = G(y).$$

The expected payment by a bidder of value x in the second-price auction is

$$m^{\mathrm{II}}(x) = G(x)\,\mathbb{E}\big[y_1 \mid y_1 < x\big].$$

5.3.3 First-Price Auction

The payoff to bidder i is

$$\pi_i = \begin{cases} x_i - b_i, & b_i > \max\limits_{j \neq i} b_j, \\ 0, & \text{otherwise.} \end{cases}$$

Assume that bidders $j \neq 1$ follow an increasing strategy β, and let

$$y_1 = \max_{j \neq 1} x_j.$$

Suppose bidder 1, whose valuation is x, bids b, where $b \leq \beta(\bar{w})$ and $\beta(0) = 0$.. Then

$$\mathbb{E}(\text{payoff}) = G\big(\beta^{-1}(b)\big)(x - b) = \mathbb{P}\big(\beta(y_1) < b\big)(x - b)$$

Maximizing with respect to b, and using $G' = g$, gives the first-order condition

$$\frac{g\big(\beta^{-1}(b)\big)}{\beta'\big(\beta^{-1}(b)\big)}(x - b) - G\big(\beta^{-1}(b)\big) = 0.$$

(If $y = \beta^{-1}(b), b = \beta(y)$ then $\frac{db}{dy} = \beta'(y) = \beta'\left(\beta^{-1}(b)\right) = \frac{1}{dy/db}$.)

In a symmetric equilibrium $b = \beta(x)$. Hence

$$G(x)\beta'(x) + g(x)\beta(x) = xg(x),$$

or equivalently

$$\frac{d}{dx}\big(G(x)\beta(x)\big) = xg(x).$$

It follows that

$$\beta(x) = \frac{1}{G(x)} \int_0^x yg(y)\,dy = \mathbb{E}\big[y_1 \mid y_1 < x\big].$$

Proposition 15. *The strategy*

$$\beta^{\mathrm{I}}(x) = \mathbb{E}\big[y_1 \mid y_1 < x\big]$$

defines a symmetric equilibrium of the first-price auction.

Proof. Assume that bidders $2, 3, \ldots, N$ use this strategy. Consider the expected payoff to bidder 1 when their value is x and they bid b. Put

$$z = \beta^{-1}(b).$$

Then

$$\begin{aligned}\Pi(b,x) &= G(z)\big(x - \beta(z)\big) = G(z)x - G(z)\mathbb{E}\big[y_1 \mid y_1 < z\big] \\ &= G(z)x - \int_0^z yg(y)\,dy = G(z)x - G(z)z + \int_0^z G(y)\,dy.\end{aligned}$$

Consequently,

$$\begin{aligned}&\Pi\big(\beta(x), x\big) - \Pi(b,x) \\ &= G(x)x - G(x)x + \int_0^x G(y)\,dy - \left(G(z)x - G(z)z + \int_0^z G(y)\,dy\right) = G(z)(z-x) - \int_x^z G(y)\,dy \\ &= \int_x^z \big(G(z) - G(y)\big)\,dy \geq 0.\end{aligned}$$

Thus bidder 1 maximizes their expected payoff by choosing $z = x$, which is equivalent to bidding $b = \beta(x)$. □

Integration by parts gives the alternative expression

$$\beta^{\mathrm{I}}(x) = \frac{1}{G(x)}\int_0^x yg(y)\,dy = \frac{1}{G(x)}\left([yG(y)]_0^x - \int_0^x G(y)\,dy\right) = x - \int_0^x \frac{G(y)}{G(x)}\,dy = x - \int_0^x \left(\frac{F(y)}{F(x)}\right)^{N-1} dy.$$

5.3.4 Examples

1. If

   $$F(x) = x, \qquad x \in [0,1],$$

   so that valuations are uniform on $[0,1]$, then

   $$\beta^{\mathrm{I}}(x) = \frac{N-1}{N}x.$$

2. Suppose $\bar{w} = \infty$ and

   $$F(x) = 1 - e^{-\lambda x} \qquad \text{and} \qquad N = 2.$$

   Then

   $$\beta^{\mathrm{I}}(x) = x - \int_0^x \frac{F(y)}{F(x)}\,dy = x - \frac{1}{1-e^{-\lambda x}}\left[y + \lambda^{-1}e^{-\lambda y}\right]_0^x = \frac{1}{\lambda} - \frac{xe^{-\lambda x}}{1-e^{-\lambda x}}.$$

   In particular, if $\lambda = 2$, a bidder never bids more than $1/2$; neither does the other bidder.

5.3.5 Revenue

The expected payment by a bidder of value x is the same in the first-price and second-price auctions:

$$m^{\mathrm{I}}(x) = G(x)\,\mathbb{E}\big[y_1 \mid y_1 < x\big] = m^{\mathrm{II}}(x).$$

5.4 Richman game

The game takes place on a finite digraph $D = (V, A)$ where $|V| < \infty$, featuring two distinguished vertices, r and b. Two players, RED and BLUE, compete using a token that starts at a vertex $v \notin \{r, b\}$. A path exists from every vertex to at least one of r, b.

RED possesses X_R money, while BLUE possesses X_B money. During a move, RED makes a bid $x \leq X_R$ and BLUE makes a bid $y \leq X_B$. If $x > y$, RED wins the bid, RED's money becomes $X_R \leftarrow X_R - x$, BLUE's money becomes $X_B \leftarrow X_B + x$, and RED moves the token. If $x < y$, BLUE wins the bid, RED's money becomes $X_R \leftarrow X_R + y$, BLUE's money becomes $X_B \leftarrow X_B - y$, and BLUE moves the token. If $x = y$, the choice of who moves the token is randomized. RED's objective is to move the token to r, whereas BLUE's objective is to move the token to b.

Notation For a given vertex v, its set of successors (out-neighbors) is denoted as $S(v)$. For a function $f : V \to \mathbb{R}$, $f^+(v) = \max_{w \in S(v)} f(w)$ and $f^-(v) = \min_{w \in S(v)} f(w)$.

R is a RICHMAN function if (i) $R(b) = 0$, (ii) $R(r) = 1$, and (iii) $R(v) = \frac{R^+(v)+R^-(v)}{2}$ for all $v \in V$, where

Lemma 16. *RICHMAN functions exist.*

Proof. We will define $R(v)$ as the limit of $R(v, t)$ as $t \to \infty$. We define $R(v, t)$ as follows: $R(b, t) = 0$ and $R(r, t) = 1$ for $t = 0, 1, 2, \ldots$ and $R(v, 0) = 1$ for all $v \in V \setminus \{r, b\}$. Then we let $R(v, t) = \frac{1}{2}(R^+(v, t-1) + R^-(v, t-1))$ for $t > 0, v \neq r, b$.

We have $R(v, 1) \leq R(v, 0)$ for all $v \in V$ and then

$$R(v, t+1) = \frac{R^+(v, t) + R^-(v, t)}{2} \leq \frac{R^+(v, t-1) + R^-(v, t-1)}{2} = R(v, t)$$

follows by induction. This implies that $R(v) = \lim_{t\to\infty} R(v, t)$ exists and since $\lim_{t\to\infty} R^{\pm}(v, t) = R^{\pm}(t)$, it follows easily that R is a RICHMAN function. □

Theorem 17. *Assume that $X_B + X_R = 1$. If $X_B > R(v, t)$, where $R(v, t)$ is as in Lemma 16 then BLUE wins in at most t moves, starting from v.*

Proof. The proof is by induction on t. Trivial for $t = 0$, since $X_B > R(v, 0)$ implies that $v = b$. Assume the truth of the lemma for $t - 1$. Let u and w be defined such that $R(u, t-1) = R^-(v, t-1)$ and $R(w, t-1) = R^+(v, t-1)$. Blue bids $\Delta = (R(w, t-1) - R(u, t-1))/2$. If Blue wins the bid, Blue moves to u and has $\geq X_B - \Delta = R(u, t-1)$, winning in $t - 1$ moves. If Red wins the bid and moves to w, Blue now has $\geq R(v, t) + \Delta = R(w, t-1)$ and will again win. □

Theorem 18. *The RICHMAN function is unique.*

Proof. If $R(u) = R^-(v)$ then the edge (v, u) is said to be of steepest descent from v. Let $A(v)$ be the set of vertices reachable from v using only edges of steepest descent.

Claim 1. *If $R(v) < 1$, then $b \in A(v)$.*

Proof of Claim 1
Choose $z \in A(v)$ such that $R(z) = \min\{R(u) : u \in A(v)\}$. Note that $R(z) \leq R(v) < 1$. Assume that $z \neq b$ and let $w \in S(z)$. The definition of z implies that $R^-(z) = R(z)$ which forces $R^+(z) = R(z)$ and then $R^-(z) \leq R(w) \leq R^+(z)$ implies that $w \in A(v)$ and $R(w) = R(z)$, so any vertex reachable from z is in $A(v)$. Now because $R(r) = 1$ we know that $r \notin A(z)$ and so by assumption $b \in A(z) \subseteq A(v)$.
End of proof of Claim 1

Assume that R_1 and R_2 are both RICHMAN functions. Choose v such that the difference is maximized and let $M = R_1(v) - R_2(v)$. Let u_i and w_i correspond to a local minimum and maximum for R_i from v.

$$R_1(u_1) - R_2(u_2) \leq R_1(u_2) - R_2(u_2) \leq M \text{ and } R_1(w_1) - R_2(w_2) \leq R_1(w_1) - R_2(w_1) \leq M.$$

Summing, yields

$$M = R_1(v) - R_2(v) = \frac{R_1(u_1) + R_1(w_1)}{2} - \frac{R_2(u_2) + R_2(w_2)}{2} \leq M.$$

This implies that $R_1(u_2) - R_2(u_2) = M$ and because u_2 is any vertex corresponding to a local minimum, we find that all values along paths of steepest descent maintain the maximum difference M. If $R_2(v) < 1$, then b is in the steepest descent path, forcing $M \leq 0$. The same argument with $M' = R_2(v) - R_1(v)$ shows $M = -M' \geq 0$, proving uniqueness. □

Linear Program The values for $R(v)$ can be computed by solving a linear program.

$$\begin{aligned} \text{Minimize } & \sum_v x_v^+ - \sum_v x_v^-. \\ \text{Subject to } & x_v^+ \geq x_w \text{ for } (v,w) \in A. \\ & x_v^- \leq x_w \text{ for } (v,w) \in A. \\ & x_v = \frac{x_v^+ + x_v^-}{2}. \\ & 0 \leq x_w \leq 1, x_b = 0, x_r = 1. \end{aligned}$$

6 Integer Programming

This is the name given to Linear Programming problems which have an extra constraint in that some or all of the variables have to be integer.

6.1 Examples

Capital budgeting A firm has n projects that it would like to undertake but because of budget limitations not all can be selected. In particular project j is expected to produce a revenue of c_j but requires an investment of $a_{i,j}$ in time period i for$i = 1, \ldots, m$. The capital available in time period i is b_i. The problem of maximising revenue subject to the budget constraints can be formulated as follows: let $x_j = 0/1$ correspond to not proceeding or respectively proceeding with project j then we have to

$$\begin{aligned} \text{Maximise } & \sum_{j=1}^{n} c_j x_j \\ \text{Subject to } & \sum_{j=1}^{n} a_{i,j} x_j \leq b_i, \qquad i = 1, \ldots, m. \\ & 0 \leq x_j \leq 1, \ x_j \text{ integer for } j = 1, 2, \ldots, n. \end{aligned}$$

Depot location We consider here a simple problem of this type: a company has selected m possible sites for distribution of its products in a certain area. There are n customers in the area and the transportation cost of supplying the whole of customer j's requirements over the given planning period from potential site i is $c_{i,j}$. Should site i be developed it will cost f_i to construct a depot there. Which sites should be selected to minimise the total construction plus transport cost?

To do this we introduce variables $y_1, \ldots, y_m$ which can only take values 0 or 1 and correspond to a particular site being not developed or developed respectively. We next define $x_{i,j}$ to be the fraction of customer j's requirements supplied from depot i in a given solution. The problem can then be expressed,

$$\begin{aligned} \text{Minimise } & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j} + \sum_{i=1}^{m} f_i y_i \\ \text{Subject to } & \sum_{i=1}^{m} x_{i,j} = 1, \qquad j = 1, 2, \ldots, n \\ & x_{i,j} \leq y_i, \quad i = 1, 2, \ldots, m, \ j = 1, 2, \ldots, n \\ & x_{i,j} \geq 0, \ 0 \leq y_i \leq 1, y_i \text{ integer}, \qquad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n. \end{aligned}$$

Note that if $y_i = 0$ then $f_i y_i = 0$ and there is no contribution to the total cost. Also, $x_{i,j} \leq y_i$ implies $x_{i,j} = 0$ and no goods are distributed from site i. This corresponds exactly to there not being a depot at location i.

On the other hand, if $y_i = 1$, then $f_i y_i = f_i$ which is the cost of constructing depot i. Also, $x_{i,j} \leq y_i$ becomes $x_{i,j} \leq 1$ which holds anyway from the first constraint.

Set Covering Let $S_1, S_2, \ldots, S_n$ be a family of subsets of a set $S = \{1, 2, \ldots, m\}$. A covering of S is a subfamily S_j for $j \in I$ such that $S = \bigcup_{j \in I} S_j$. Assume that each subset S_j has a cost $c_j > 0$ associated with

it. We define the cost of a cover to be the sum of the costs of the subsets included in the cover.

The problem of finding a cover of minimum cost is of particular practical significance. As an integer program it can be specified as follows: define the $m \times n$ matrix $A = [a_{i,j}]$ by

$$a_{i,j} = \begin{cases} 1 & i \in S_j. \\ 0 & i \notin S_j. \end{cases}$$

Let $x_j, j = 1, 2, \ldots, n$ be 0 /1 variables with $x_j = 1(0)$ to mean set S_j is included (respectively not included) in the cover. The problem is to

$$\begin{aligned} \text{Minimize} & \sum_{j=1}^{n} c_j x_j \\ \text{Subject to} & \sum_{j=1}^{n} a_{i,j} x_j \geq 1, \qquad i = 1, 2, \ldots, m. \\ & x_j = 0 \text{ or } 1, j = 1, 2, \ldots, n. \end{aligned} \tag{67}$$

The m inequality constraints have the following significance: since $x_j = 0$ or 1 and the coefficients $a_{i,j}$ are also 0 or 1 we see that $\sum_{j=1}^{n} a_{i,j} x_j$ can be zero only if $x_j = 0$ for all j such that $a_{i,j} = 1$. In other words only if no set S_j, is chosen such that $i \in S_j$. The inequalities are put in to avoid this.

As an example consider the following simplified airline crew scheduling problem. An airline has m scheduled flight-legs per week in its current service. A flight-leg being a single flight flown by a single crew e.g. London - Paris leaving Heathrow at 10.30 am. Let $S_j, j = 1, 2, \ldots, n$ be the collection of all possible weekly sets of flight-legs that can be flown by a single crew. Such a subset must take account of restrictions like a crew arriving in Paris at 11.30 am, cannot take a flight out of New York at 12.00 pm. and so if c_j is the cost of set S_j of flight-legs then the problem of minimising cost subject to covering all flight-legs is a set covering problem. Note that if crews are not allowed to be passengers on a Flight e.g. so that they can be flown to their next flight, then we have to make (67) an equality – the set partitioning problem.

General terminology The most general problem called the *mixed integer programming problem* can be specified as

$$\begin{aligned} \text{Minimise } & x_0 = \mathbf{c}^T \mathbf{x} \\ \text{Subject to } & A\mathbf{x} = \mathbf{b} \\ & x_j \geq 0, \qquad j = 1, 2, \ldots, n. \\ & x_j \text{ integer for } j \in I \end{aligned}$$

where $I \subseteq [n]$.

When $I = [n]$ and all the quantities $c_j, a_{i,j}, b_i$ are integer then we have a *pure integer programming problem.*

Further uses of integer variables

(i) If a variable x can only take a finite number of values $p_1, p_2, \ldots, p_m$, then we can replace x by the expression

$$x = p_1 w_1 + p_2 w_2 + \cdots + p_m w_m, \ w_1 + w_2 + \cdots + w_m = 1, \ w_i = 0 \text{ or } 1 \text{ for } i = 1, 2, \ldots, m.$$

For example X might be the output of a plant which can be small p_1, medium p_2 or large p_3. The cost $c(x)$ of the plant could be represented by $c_1w_1 + c_2w_2 + c_3w_3$ where c_1 is the cost of a small plant etc.

(ii) In L.P. one generally consider all constraints to be holding simultaneously. It is possible that the variable might have to satisfy one or other of a set of constraints.

$$0 \leq x \leq M \text{ and } (0 \leq x \leq 1 \text{ OR } x \geq 2).$$

We replace this by

$$x \leq 1 + (M-1)(1-\delta) \text{ and } x \geq 2 - M\delta \text{ and } x \geq 0, \delta = 0/1.$$

Hardness Integer programming problems generally take much longer to solve than the corresponding linear program obtained by ignoring integrality. It is wise therefore to consider the possibility of solving as a straight forward L.P. and then rounding e.g, in the trim-loss problem. This is not always possible for example if x is a $0/1$ variable such that $x = 0$ means do build a plant and $x = 1$ means build a plant then rounding $x = 1/2$ is not very satisfactory.

6.2 A cutting plane algorithm for the pure problem

The rationale behind this approach is:-

Step 1 Solve the continuous problem as an L.P. i.e. ignore integrality.

Step 2 If by chance the optimal basic variables are all integer then the optimum solution has been found. Otherwise,

Step 3 Generate a cut i.e. a constraint which is satisfied by all integer solutions to the problem but not by the current L.P. solution.

Step 4 Add this new constraint and go to Step 1.

It is straight forward to show that if at any stage the current L.P. Solution $\mathbf{x}$ is integer it is the optimal integer solution. This is because $\mathbf{x}$ is optimal over a region containing all feasible integer solutions. The problem is to define cuts that ensure the convergence of the algorithm in a finite number of steps. The first finite algorithm was devised by R.E, Gomory. It is based on the following construction: let

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

be an equation which is to be satisfied by non-negative integers $x_1, x_2, \ldots, x_n$ and let S be the set of possible integer solutions.

For a real number ξ we define $\lfloor \xi \rfloor$ to be the largest integer which is less than or equal to ξ. Thus $\xi = \lfloor \xi \rfloor + \varepsilon$ where $0 \leq \varepsilon < 1$.

$$\lfloor 6.5 \rfloor = 6, \ \lfloor 3 \rfloor = 3, \ \lfloor -4.5 \rfloor = -5.$$

Now let $a_j = \lfloor a_j \rfloor + f_j$ and $b = \lfloor b \rfloor + f$. Then we have

$$\sum_{j=1}^{n} (\lfloor a_j \rfloor + f_j) x_j = \lfloor b \rfloor + f$$

and hence

$$\sum_{j=1}^{n} f_j x_j - f = \lfloor b \rfloor - \sum_{j=1}^{n} \lfloor a_j \rfloor x_j. \tag{68}$$

Now for $x \in S$, the RHS of (68) is an integer and the LHS is at least $-f > -1$. This implies that

$$\sum_{j=1}^{n} f_j x_j - f \in \{0, 1, \ldots\}.$$

Suppose now that one has solved the LP relaxation and the solution is not integer. Therefore there is a basic variable x_i with

$$x_i + \sum_{j \notin I} b_{i,j} x_j = b_{i,0}$$

where $b_{i,0}$ is not an integer. (Here I is the set of indices of basic variables and the $b_{i,j}$ are the coefficients of the simplex tableaux.)

Putting $f_j = b_{i,j} - \lfloor b_{i,j} \rfloor$ for $j \notin I$ and $f = b_{i,0} - \lfloor b_{i,0} \rfloor$ we see that

$$\sum_{j \notin I} f_j x_j \geq f \tag{69}$$

for all integer solutions to our problem.

Now $f > 0$ since $b_{i,0}$ is not an integer and so (69) is not satisfied by the current L.P. solution since $x_j = 0$ for $j \notin I$ and so (69) is a cut.

The initial continuous problem solved by the algorithm is the L.P. problem obtained by ignoring integrality.

Statement of the Algorithm

Step 1 Solve current continuous problem.

Step 2 If the solution is integral it is the optimal integer solution, otherwise.

Step 3 Choose a basic variable x_i, which is currently non-integer, construct the corresponding constraint (69) and add it to the problem. Go to step 1.

We note that the tableau obtained after adding the cut is dual feasible and so the dual simplex algorithm can be used to re-optimise.

Example:

$$\text{Maximise } x_1 + 4x_2$$
$$\text{Subject to } 2x_1 + 4x_2 \leq 7.$$
$$10x_1 + 3x_2 \leq 14.$$
$$x_1, x_2 \geq 0 \text{ and integer.}$$

$B.V.$	x_1	x_2	x_3	x_4	ξ_1	ξ_2	RHS	
x_0	-1	-4					0	
x_3	2	**4**	1				7	
x_4	10	3		1			14	
x_0	1		1				7	
x_2	1/2	1	1/4				7/4	cut made from this row
x_4	17/2		$-3/4$	1			35/4	
ξ_1	$\mathbf{-1/2}$		$-1/4$		1		$-3/4$	
x_0			1/2		2		11/2	
x_2		1			1		1	
x_4			$\mathbf{-5}$	1	17		-4	
x_1	1		1/2		-2		3/2	
x_0				1/10	37/10		51/10	cut made from this row
x_2		1			1		1	
x_3			1	$-1/5$	$-17/5$		4/5	
x_1	1			1/10	$-3/10$		11/10	
ξ_2				$\mathbf{-1/10}$	$-7/10$	1	$-1/10$	
x_0					3		5	
x_2		1			1		1	
x_3			1		-1	-2	1	
x_1					-1	1	1	
x_4					7	-10	1	

- One can show that the Gomory cuts $\sum_j f_j > f$ when expressed in terms of the original non-basic variables have the form $\sum_j w_j x_j \leq W$ where the w_j, W are integer and the value of $\sum_j w_j x_j$ after solving the current continuous problem is $W + \varepsilon$ where $0 < \varepsilon < 1$ assuming the current solution is non-integer, Thus the cut is obtained by moving a hyperplane parallel to itself to an extent which cannot exclude an integer solution. It is worth noting that the plane can usually be moved further without excluding integer points thus generating deeper cuts. For a discussion on how this can be done see the reference given for integer programming,

- After adding a cut and carrying out one iteration of the dual simplex algorithm the slack variable corresponding to this cut becomes non-basic, If during a succeeding iteration this slack variable becomes basic then it may be discarded along with its current row without affecting termination. This means that the tableau never has more than $n + 1$ rows or $m + n$ columns.

- A valid cut can be generated from any row containing a non-integral variable, One strategy is to choose the variable with the largest fractional part as this 'helps' to produce a 'large' change in the objective value. It is interesting that finiteness of the algorithm has not been proved for this strategy although finiteness has been proved for the strategy of always choosing the 'topmost' row the tableau with a non-integer variable.

- The behaviour of this algorithm has been erratic. It has for example worked well on set covering problems but in other cases the algorithm has to be terminated because of excessive use of computer time, This

raises an important point; if the algorithm is stopped prematurely then one does not have a good sub-optimal solution to use. Thus in some sense the algorithm is unreliable,

7 Branch and bound

We consider the problem P_0:

$$\text{Minimize } f(x) \text{ subject to } x \in S_0.$$

Here S_0 is our set of feasible solutions and $f : S_0 \to \mathbb{R}$.

As we proceed in Branch-and-Bound we create a set of sub-problems $\mathcal{P}$. A sub-problem $P \in \mathcal{P}$ is defined by *the description of* a subset $S_P \subseteq S_0$. We also keep a *lower bound* b_P where

$$b_P \leq \min \{f(x) : \ x \in S_P\}.$$

At all times we act as if we have $x^* \in S_0$, some known feasible solution to P_0 and $v^* = f(x^*)$. If we do not actually have a solution x^* then we let $v^* = \infty$. We will have a procedure BOUND that computes b_P for a sub-problem P. In many cases, BOUND *sometimes* produces a solution $x_P \in S_0$ and sometimes determines that $S_P = \emptyset$.

We initialize $\mathcal{P} = \{P_0\}$.

Branch and Bound:

Step 1 If $\mathcal{P} = \emptyset$ then x^* solves the problem.

Step 2 Choose $P \in \mathcal{P}$. $\mathcal{P} \leftarrow \mathcal{P} \setminus \{P\}$.

Step 3 **Bound:** Run BOUND(P) to compute b_P.

Step 4 If $S_P = \emptyset$ or $b_P \geq v^*$ then we consider P to be solved and go to Step 1.

Step 5 If BOUND generates $x_P \in S_0$ and $f(x_P) < v^*$ then we update, $x^* \leftarrow x_P, v^* \leftarrow f(x_P)$.

Step 6 **Branch:** Split P into a number of subproblems $Q_i, i = 1, 2, \ldots, \ell$, where $S_P = \bigcup_{i=1}^{\ell} S_{Q_i}$. And $S_{Q_i} \neq S_P$ is a strict subset for $i = 1, 2, \ldots, \ell$.

Step 7 $\mathcal{P} \leftarrow \mathcal{P} \cup \{Q_1, Q_2, \ldots, Q_\ell\}$.

Assuming S_0 is finite, this procedure will eventually terminate with $\mathcal{P} = \emptyset$. This is because the feasible sets S_P are getting smaller and smaller as we branch.

Most often the procedure BOUND has the following form: while it may be difficult to solve P directly, we may be able to find $T_P \supseteq S_P$ such that there is an efficient algorithm that determines whether or not $T_P = \emptyset$ and finds $\xi_P \in T_P$ that minimizes $f(\xi), \xi \in T_P$, if $T_P \neq \emptyset$. In this case, $b_P = f(\xi_P)$ and Step 5 is implemented if $\xi_P \in S_0$. We call the problem of minimizing $f(\xi), \xi \in T_P$, a *relaxed problem*.

Examples:

Ex. 1 Integer Linear Programming. Here S_P is the set of integer solutions and T_P is the set of solutions, if we ignore integrality. The procedure BOUND solves the linear program. If the solution ξ_P is not integral, we choose a variable x, whose value is $\zeta \notin \mathbb{Z}$ and form 2 sub-problems by adding $x \leq \lfloor z \rfloor$ to one and $x \geq \lceil z \rceil$ to the other.

Ex. 2 Traveling Salesperson Problem (TSP): Here S_P is the set of tours i.e. single directed cycles that cover all the vertices. We can take T_P to be the set of collections of vertex disjoint directed cycles that cover all the vertices. More precisely, to solve the TSP we must minimise $\sum_{i=1}^{n} C(i, \pi(i))$ as π ranges over all cyclic permutations. Our relaxation is to minimise $\sum_{i=1}^{n} C(i, \pi(i))$ as π ranges over all permutations, i.e. the assignment problem. We branch as follows. Suppose that the assignment solution consists of cycles $C_1, C_2, \ldots, C_k, k \geq 2$. Choose a cycle, C_1 say. Suppose that $C_1 = (v_1, v_2, \ldots, v_r)$ as a sequence of vertices. Then in Q_1 we disallow $\pi(v_1) = v_2$, in Q_2 we insist that $\pi(v_1) = v_2$, but that $\pi(v_2) \neq v_3$, in Q_3 we insist that $\pi(v_1) = v_2$, $\pi(v_2) = v_3$, but that $\pi(v_3) \neq v_4$ and so on.

Ex. 3 Implicit Enumeration: Here the problem is

$$\text{Minimize } \sum_{j=1}^{n} c_j x_j \text{ subject to } \sum_{j=1}^{n} a_{i,j} x_j \geq b_i,\ i \in [m], \quad x_j \in \{0,1\}, j \in [n].$$

A sub-problem is associated with two sets $I, O \subseteq [n]$. This is the sub-problem $P_{I,O}$ where we add the constraints $x_j = 1, j \in I, x_j = 0, j \in O$. We also check to see if $x_j = 1, j \in I, x_j = 0, j \notin I$ gives an improved feasible solution. As a bound $b_{I,O}$ we use $\sum_{j \in I} c_j + \sum_{j \notin O \cup I} \min\{c_j, 0\}$. To test feasibility we check the *necessary condition* that $\sum_{j \in I} a_{i,j} + \sum_{j \notin I \cup O} \max\{a_{i,j}, 0\} \geq b_i, i \in [m]$. To branch, we split $P_{I,O}$ into $P_{I \cup \{j\},O}$ and $P_{I,O \cup \{j\}}$ for some $j \notin I \cup O$.

8 Combinatorial Optimization

8.1 Shortest path

8.1.1 Non-negative lengths

We are given a digraph $D = ([n], E)$ with vertex set $[n]$. Let $\mathcal{P}$ denote the set of paths in D and let $\ell : \mathcal{P} \to \mathbb{R}$. Set $\ell(e) = \infty$ if e is not an edge of D. Think initially that there are edge lengths $\ell : E \to \mathbb{R}_+$ and that

$$\ell(P) = \ell_{reg}(P) = \sum_{e \in P} \ell(e).$$

Dijkstra's Algorithm:
begin
for $i = 2, \ldots, n$, $d(i) \leftarrow \ell(1, i)$, $P_i \leftarrow (1, i)$; $S_1 \leftarrow \{1\}$;
  for $k = 1, \ldots, n-1$ **do**;
  begin
    $i = \text{argmin}\{d(j) : j \notin S_k\}$;
    $S_{k+1} \leftarrow S_k \cup \{i\}$;
    for $j \notin S_{k+1}$ **do**
    if $d(j) > \ell(P_i, j)$ then $d(j) \leftarrow \ell(P_i, j)$, $P_j \leftarrow (P_i, j)$;
  end
end

Lemma 19. *On termination of Dijkstra's Algorithm, $d(i) = \ell(P_i)$ is the minimum length of a path from 1 to i, for all i*

Proof. At each stage we can verify by induction on k that for each $i \notin S_k$, $d(i)$ is the minimum length of a path from 1 to i for which all vertices but i are in S_k. If true for k then when we add vertex i we simply update the d's correctly.

Suppose that i is added at Step r. Let $P = (x_0 = 1, x_1, \ldots, x_m = i)$ be a path from 1 to i. Suppose that $x_0, x_1, \ldots, x_{l-1} \in S_{r-1}$ and $x_l \notin S_{r-1}$. Then,

$$\ell(P) \geq \ell(P_{x_{l-1}}, x_l, \ldots, x_m) \geq \ell(P_{x_{l-1}}, x_l) \geq d(x_l) \geq d(i) = \ell(P_i).$$

□

Note now that all we have assumed about ℓ is that

(i) $\ell(X, Z) \geq \ell(X)$ for all internally vertex disjoint paths X, Z such that Z starts at the end of X.

(ii) If X, Y have the same end vertices and $\ell(X) \leq \ell(Y)$ then $\ell(X, Z) \leq \ell(Y, Z)$ for all paths Z that begin at the end of X.

We used (ii) with $X = (P_{x_{l-1}}, x_l), Y = (x_0, x_1, \ldots, x_l), Z = (x_l, \ldots, x_m)$ to justify $\ell(P) \geq \ell(P_{x_{l-1}}, x_l, \ldots, x_m)$ and (i) with $X = (P_{x_{l-1}}, x_l), Z = (x_l, \ldots, x_m)$ to justify $\ell(P_{x_{l-1}}, x_l, \ldots, x_m) \geq \ell(P_{x_{l-1}}, x_l)$.

In which case, we can apply the algorithm to solve problems where path length is defined as follows:

Time dependent path lengths: Suppose edge $e = (x, y)$ has two parameters $a_e, b_e \geq 0$ and that if we start a walk at time 0 and arrive at x at time t then the edge length is $a_e + b_e t$. Suppose that $P = (e_0, e_1, \ldots, e_k)$ as a sequence of edges and that $P_i = (e_0, e_1, \ldots, e_i)$. Then we now have $\ell(P_0) = a_{e_0}$ and $\ell(P_i) = a_{e_i} + (1 + b_{e_i})\ell(P_{i-1})$.

8.1.2 No negative cycles

Our algorithms find shortest walks between vertices. When there are no negative length cycles, this amounts to finding shortest paths. When there are negative cycles, there will sometimes be no minimum length walk.

We begin by discussing the case where $\ell(P) = \sum_{e \in P} \ell(e)$ and we assume that there are no negative cycles. Assume that the edges of D are $E = \{e_i = (x_i, y_i), i = 1, 2, \ldots, m\}$. Let $P_i, i = 1, 2, \ldots, n$ be a collection of paths, where $P_1 = (1)$ and P_i goes from 1 to i.

Lemma 20. *The following is a necessary and sufficient condition for $P_1, P_2, \ldots, P_n$ to be a collection of shortest paths with start vertex 1:*

$$\ell(P_y) \leq \ell(P_x) + \ell(x, y) \text{ for all } (x, y) \in E. \tag{70}$$

Proof. It is clear that (70) is necessary. If it fails then $P_x + (x, y)$ is a shorter walk from 1 to y than P_y and we can remove a cycle to make a path if necessary, without increasing the length.

Suppose that (70) holds. Let $P = (1 = x_0, x_1, x_2, \ldots, x_k = i)$ be a path from 1 to i. We show by induction on j that

$$\ell(P_{x_j}) \leq \ell(P(1, x_1, x_2, \ldots, x_j)). \tag{71}$$

Now when $j = 0$, both sides of (71) are zero. Then if it holds for some $j \geq 0$ then (70) and the inductive assumption imply that

$$\ell(P_{x_{j+1}}) \leq \ell(P_{x_j}) + \ell(x_j, x_{j+1}) \leq \ell(P(1, x_1, \ldots, x_j)) + \ell(x_j, x_{j+1}) = \ell(P(1, x_1, \ldots, x_{j+1})).$$

Thus (70) is sufficient. □

Ford's Algorithm:
begin
for $i = 2, \ldots, n$, $d(i) \leftarrow \ell(1, i)$, $P_i \leftarrow (1, i)$;
    repeat;
    $flag \leftarrow 0$;
    for $i = 1, 2, \ldots, m$;
    begin
        if $\ell(P_{y_i}) > \ell(P_{x_i}) + \ell(x_i, y_i)$ **then**;
        begin;
            $P_{y_i} \leftarrow (P_{x_i}, y_i)$; $flag \leftarrow 1$;
        end;
    end;
    until $flag = 0$;
    end
end

Lemma 21. *Ford's algorithm terminates after at most n rounds with a collection of shortest paths.*

Proof. If the algorithm terminates then, because $flag = 0$ at this point, we have that (70) holds. Thus we have shortest paths.

We now argue that if the minimum number of arcs in a shortest path from 1 to i has ν_i edges then P_i is correct after ν_i rounds. We argue by induction. This is true for $i = 1$ and $\nu_i = 0$. Suppose that it is true for all i such that $\nu_i \leq \nu$ and that vertex j satisfies $\nu_j = \nu + 1$. Let $P = (1 = x_0, x_1, \ldots, x_{\nu+1} = j)$ be a shortest path from 1 to j. Then, by induction, after ν rounds P_{x_ν} is a shortest path from 1 to x_ν and then after one more round P_j is correct. □

Electric cars: Suppose when we drive along edge e, $\ell(e)$ the amount of energy used is $\ell(e)$. This is normally positive, but when going down hill it can be negative. In this scenario, there can be no negative cycles under ℓ_{reg}.

Assume that the edges of D are $E = \{e_i = (x_i, y_i), i = 1, 2, \ldots, m\}$. Let $P_i, i = 1, 2, \ldots, n$ be a collection of paths, where $P_1 = (1)$ and P_i goes from 1 to i.

Lemma 22. *The following is a necessary and sufficient condition for $P_1, P_2, \ldots, P_n$ to be a collection of shortest paths with start vertex 1:*

$$\ell(P_y) \leq \ell(P_x * y) \text{ for all } (x, y) \in E. \tag{72}$$

Proof. It is clear that (72) is necessary. If it fails then $P_x * y$ is "shorter" than P_y.

Suppose that (72) holds. Let $P = (1 = x_0, x_1, x_2, \ldots, x_k = i)$ be a path from 1 to i. We show by induction on j that

$$\ell(P_{x_j}) \leq \ell(P(1, x_1, x_2, \ldots, x_j)). \tag{73}$$

Now when $j = 0$, both sides of (73) are zero. Then if it holds for some $j \geq 0$ then (72) and the inductive assumption imply that

$$\ell(P_{x_{j+1}}) \leq \ell(P_{x_j} * x_{j+1}) \leq \ell(P(1, x_1, \ldots, x_{j+1})).$$

Thus (72) is sufficient. □

Ford's Algorithm:
begin
for $i = 2, \ldots, n$, $d(i) \leftarrow \ell(1, i)$, $P_i \leftarrow (1, i)$;
    repeat;
    $flag \leftarrow 0$;
    for $i = 1, 2, \ldots, m$;
    begin
        if $\ell(P_{y_i}) > \ell(P_{x_i} * y_i)$ **then**;
        begin;
            $P_{y_i} \leftarrow (P_{x_i} * y_i)$; $flag \leftarrow 1$;
        end;
    end;
    until $flag = 0$;
    end
end

Lemma 23. *Ford's algorithm terminates after at most n rounds with a collection of shortest paths.*

Proof. If the algorithm terminates then because $flag = 0$ at this point, we have that (72) holds. Thus we have shortest paths.

We now argue that if the minimum number of arcs in a shortest path from 1 to i has ν_i edges then P_i is correct after ν_i rounds. We argue by induction. This is true for $i = 1$ and $\nu_i = 0$. Suppose that it is true for all i such that $\nu_i \leq \nu$ and that vertex j satisfies $\nu_j = \nu + 1$. Let $P = (1 = x_0, x_1, \ldots, x_{\nu+1} = j)$ be a shortest path from 1 to j. Then, by induction, after ν rounds P_{x_ν} is a shortest path from 1 to x_ν and then after one more round P_j is correct. □

8.1.3 Shortest paths between all pairs of nodes

We use the two matrices $D = \|d_{ij}\|$ and $N = \|n_{ij}\|$, where d_{ij} is the length of the current best known path from i to j and n_{ij} is the second node after i on this path. The algorithm tries to improve paths as follows:

If $d_{ik} + d_{kj} < d_{ij}$, then the known path from i to j should be replaced by the known path from i to k followed by the one from k to j.

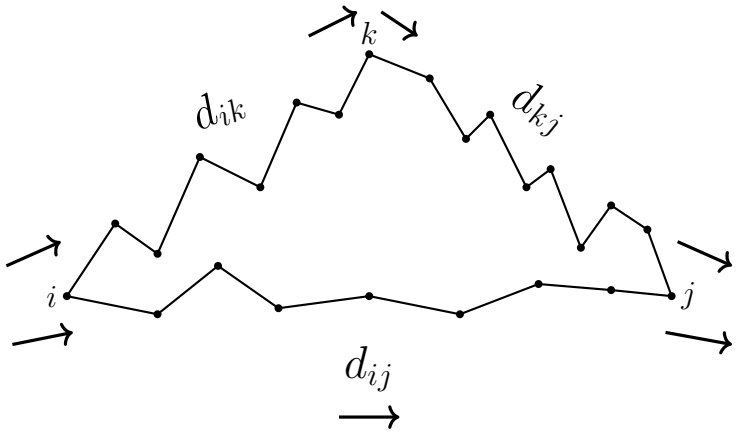
Floyd-Warshall Algorithm

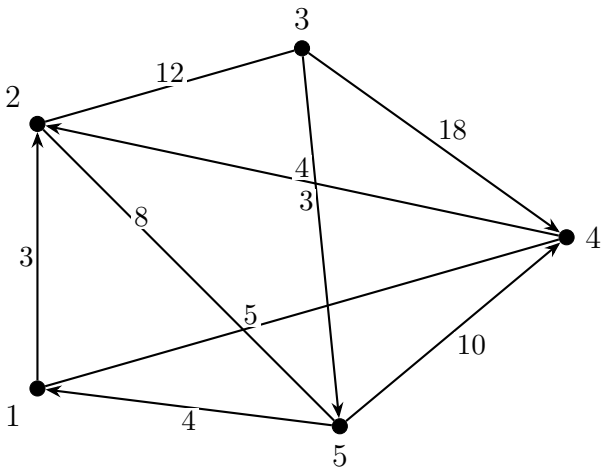
(Initialization)

```
begin
for i := 1 to n do
    for j := 1 to n do
        d(i,j) := ℓ(i,j)
        n(i,j) := j
```

(Main Loop)

```
for k := 1 to n do
    for i := 1 to n do
        for j := 1 to n do
            if d(i,k) + d(k,j) < d(i,j) then
                d(i,j) := d(i,k) + d(k,j)
                n(i,j) := n(i,k)
end
```





Matrix Iterations

Initially

$$D = \begin{pmatrix} 0 & 3 & \infty & 5 & \infty \\ \infty & 0 & 12 & \infty & 8 \\ \infty & 12 & 0 & 18 & 3 \\ 5 & 4 & \infty & 0 & \infty \\ 4 & 8 & \infty & 10 & 0 \end{pmatrix}, \quad N = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

Stage 1

$$D = \begin{pmatrix} 0 & 3 & \infty & 5 & \infty \\ \infty & 0 & 12 & \infty & 8 \\ \infty & 12 & 0 & 18 & 3 \\ 5 & 4 & \infty & 0 & \infty \\ 4 & 7 & \infty & 9 & 0 \end{pmatrix}, \quad N = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 3 & 1 & 5 \end{pmatrix}$$

Stage 2

$$D = \begin{pmatrix} 0 & 3 & 15 & 5 & 11 \\ \infty & 0 & 12 & \infty & 8 \\ \infty & 12 & 0 & 18 & 3 \\ 5 & 4 & 16 & 0 & 12 \\ 4 & 7 & 19 & 9 & 0 \end{pmatrix}, \quad N = \begin{pmatrix} 1 & 2 & 2 & 4 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 2 & 4 & 2 \\ 1 & 1 & 1 & 1 & 5 \end{pmatrix}$$

Stage 3

$$D = \begin{pmatrix} 0 & 3 & 15 & 5 & 11 \\ \infty & 0 & 12 & 30 & 8 \\ \infty & 12 & 0 & 18 & 3 \\ 5 & 4 & 16 & 0 & 12 \\ 4 & 7 & 19 & 9 & 0 \end{pmatrix}, \quad N = \begin{pmatrix} 1 & 2 & 2 & 4 & 2 \\ 1 & 2 & 3 & 3 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 2 & 4 & 2 \\ 1 & 1 & 1 & 1 & 5 \end{pmatrix}$$

Stage 4

$$D = \begin{pmatrix} 0 & 3 & 15 & 5 & 11 \\ 35 & 0 & 12 & 30 & 8 \\ 23 & 12 & 0 & 18 & 3 \\ 5 & 4 & 16 & 0 & 12 \\ 4 & 7 & 19 & 9 & 0 \end{pmatrix}, \quad N = \begin{pmatrix} 1 & 2 & 2 & 4 & 2 \\ 3 & 2 & 3 & 3 & 5 \\ 4 & 2 & 3 & 4 & 5 \\ 1 & 2 & 2 & 4 & 2 \\ 1 & 1 & 1 & 1 & 5 \end{pmatrix}$$

Stage 5

$$D = \begin{pmatrix} 0 & 3 & 15 & 5 & 11 \\ 12 & 0 & 12 & 17 & 8 \\ 7 & 10 & 0 & 12 & 3 \\ 5 & 4 & 16 & 0 & 12 \\ 4 & 7 & 19 & 9 & 0 \end{pmatrix}, \quad N = \begin{pmatrix} 1 & 2 & 2 & 4 & 2 \\ 5 & 2 & 3 & 5 & 5 \\ 5 & 5 & 3 & 5 & 5 \\ 1 & 2 & 2 & 4 & 2 \\ 1 & 1 & 1 & 1 & 5 \end{pmatrix}$$

$$d_{3,2} = 10, \quad n_{3,2} = 5, \quad n_{5,2} = 1, \quad n_{1,2} = 2.$$

Shortest path from 3 to 2 is $(3, 5, 1, 2)$.

Theorem 24. *The Floyd-Warshall algorithm finds a shortest path between any pair of nodes.*

Proof. We shall prove this by induction, the hypothesis being that at the beginning of the kth stage d_{ij} is the minimum length of a path from i to j with intermediate nodes taken from $\{1, 2, \ldots, k-1\}$. This is true for $k = 1$ and so assume it to be true for a general k.

Now paths from i to j which only use $\{1, 2, \ldots, k\}$ as intermediate nodes either use node k or they do not. The minimum length of such paths which do not use k is the value of d_{ij} at the beginning of the k'th stage. A path from i to j which uses k is the concatenation of a path from i to k and a path from k to j and both these paths use only $1, 2, \ldots, k-1$ as intermediate nodes.

Thus the minimum length of a path from i to j which only uses $1, 2, \ldots, k$ as intermediate nodes is $\min\{d_{ij}, d_{ik} + d_{kj}\}$ which is the value of d_{ij} at the beginning of stage $k+1$. □

8.1.4 Digraphs without circuits

These are important, not least because they occur in Critical Path Analysis. Their application in this area involves computing longest paths.

8.1.5 Topological Ordering

Let the vertices of a digraph $D = ([n], E)$ be ordered $v_1, v_2, \ldots, v_n$. This ordering is *topological* if $(v_i, v_j) \in E$ implies that $i < j$.

Lemma 25. *Digraph D has a topological ordering if and only if D has no directed circuits.*

Proof. Suppose first that $v_1, v_2, \ldots, v_n$ is a topological ordering and that D has a directed cycle $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$. Then we have $i_1 < i_2 < \cdots < i_k < i_1$, contradiction.

Conversely, suppose there are no directed circuits. Let $P = (x_1, x_2, \ldots, x_k)$ be a longest path in D. Then x_k is a *sink* i.e. there are no directed edges (x_k, y). (If $y \in X = \{x_1, x_2, \ldots, x_{k-1}\}$ then D contains a circuit. If $y \notin X$ then (P, y) is longer than P.)

To get a topological ordering, we let $v_n = x_k$ and inductively order the subgraph H induced by $[n] \setminus \{v_n\}$. This is a topological ordering. If $(v_i, v_j) \in E(H)$ then $i < j$ because H is topologically ordered. Any other edge must be of the form (v_i, v_n). □

To solve the longest path problem for paths starting at v_1, we take a topological ordering and then compute $d(v_1) = 0$ and then for $j \geq 2$,

$$d(v_j) = \max\{d(v_i) + \ell(v_i, v_j) : i < j \text{ and } (v_i, v_j) \in E\}. \tag{74}$$

Lemma 26. *Equation* (74) *computes the value of a longest path from* v_1 *to every other vertex.*

Proof. That $d(v_j)$ is correct follows by induction on j. It is trivially true for $j = 1$ and then for $j > 1$ we use the fact if $P = (x_1 = v_1, x_2, \ldots, x_k = v_j)$ is a longest path from v_1 to v_j then (i) $x_{k-1} = v_l$ for some $l < j$ and (ii) $(x_1, x_2, \ldots, x_{k-1})$ is a longest path from v_1 to v_l and (iii) $\ell(P) = d(v_l) + \ell(v_l, v_j)$. □

(If v_j is not reachable from v_1 then we maximise over the emptyset in (74) and take $d(v_j) = -\infty$.)

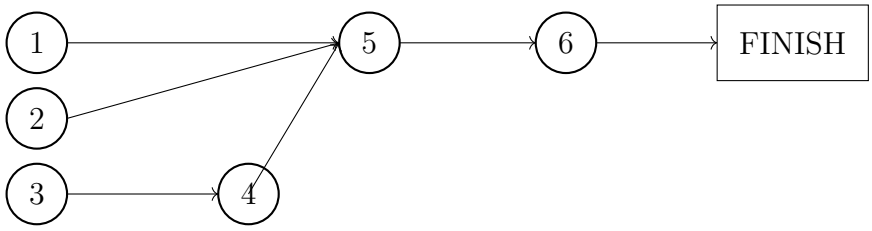
Critical Path Analysis: Imagine that a project consists of n activities.

Making a cup of tea:

1. Get a cup from the cupboard.
2. Get a tea bag.
3. Fill the kettle with water.
4. Boil the water.
5. Pour water into cup.
6. Allow to brew.

We define a digraph with n vertices, one for each activity and an edge (i, j) if (i) activity j cannot start until activity i has been completed but (ii) only include (i, j) if it is not implied by a path (i, k, j). Each edge (i, j) has a length equal to the estimated duration of the activity i.

Tea Digraph:



Associate a time t_i to start activity i. Then t_i is the length of the longest path to vertex i. The estimated completion time of the project is then the length of the longest path to FINISH.

Other parameters are: (assume that $\ell(i, j) = -\infty$ if (i, j) is not an edge of D.)

$$\text{Earliest Start Time } EST(i) = t_i.$$
$$\text{Latest Start Time } LST(i) = \min\{LST(j) - \ell(i, j) : i < j\}, \qquad LST(FINISH) = t_{FINISH}.$$
$$Float(i) = LST(i) - EST(i).$$

8.2 Assignment Problem

A *matching* M in a graph is a set of vertex disjoint edges. A vertex v is covered by M if there exists $e \in M$ such that $v \in e$. A matching M is *perfect* if every vertex of G covered by M. For the complete bipartite graph $K_{A,B}$ on vertex set $A = \{a_i : i \in [n]\}, B = \{b_i : i \in [n]\}$, perfect matchings can be represented by permutations of n i.e $M = \{(a_i, b_{\pi(i)}) : i \in [n]\}$. Given a cost matrix $(c(i,j))$, the cost of a perfect matching $M = M(\pi)$ is given by

$$c(M) = \sum_{i=1}^{n} c(i, \pi(i)).$$

The assignment problem is that of finding a perfect matching of minimum cost.

8.2.1 Alternating paths

Given a matching M, a path $P = (e_1, e_2, \ldots, e_k)$ (as a sequence of edges) is *alternating* if the edges alternate between being in M and not in M.

An alternating path is *augmenting* if it begins and ends at uncovered vertices. If P is augmenting with respect to matching M, then $M' = M \oplus P$ is also a matching and $|M'| = |M| + 1$.

8.2.2 Successive shortest path algorithm

The algorithm produces a sequence $M_1, M_2, \ldots, M_n$ where M_k is a minimum cost matching from $[k]$ to $[k]$. It begins with $M_1 = (1,1)$.

Suppose that $k > 1$ and that we have constructed $M_{k-1} = \{(a_i, b_{\pi(i)}) : i = 1, 2, \ldots, k-1\}$ and let $\pi(k) = *$. The graph Γ_k is the complete graph K_{A_k,B_k}. The digraph $\vec{\Gamma}_k$ on vertex set $A_k = \{a_i : i \in [k]\}$, $B_k = \{b_i : i \in [k]\}$ is defined as follows. The directed edges are $X = \{(b_{\pi(i)}, a_i) : i \in [k-1]\}$ and
$Y = \{(a_i, b_j) : i \in [k], j \in [k], j \neq \pi(i)\}$. The edge $(b_{\pi(i)}, a_i) \in X$ is given length $-c(i, \pi(i))$ and the edge $(a_i, b_j) \in Y$ is given length $c(i,j)$.

We observe the following:

- If M is a perfect matching of Γ_k then $M \oplus M_{k-1}$ consists of a collection $C_1, \ldots, C_p$ of vertex disjoint alternating cycles plus an augmenting path from a_k to b_k.
- $$c(M) - c(M_{k-1}) = \sum_{i=1}^{p} \ell(C_i) + \ell(P)$$
  where length ℓ is defined with respect to $\vec{\Gamma}_k$.
- $\ell(C_i) \geq 0$ for all i. Otherwise $M_{k-1} \oplus C_i$ is a matching of Γ_{k-1} with a cost $c(M_{k-1}) + \ell(C_i) < c(M_{k-1})$.

It follows from the above that to find a minimum cost matching of Γ_k, we should find a shortest path in $\vec{\Gamma}_k$ from a_k to b_k. Second, because $\vec{\Gamma}_k$ has no negative circuits, we can apply Ford's algorithm to find this path.

8.3 Network Flows

A *network* consists of a loopless digraph $D = (V, A)$ plus a function $c : A \to \mathbb{R}_+$. Here $c(x, y)$ for $(x, y) \in A$ is the *capacity* of the edge (x, y).

We use the following notation: if $\phi : A \to \mathbb{R}$ and S, T are (not necessarily disjoint) subsets of V then

$$\phi(S, T) = \sum_{\substack{x \in S, y \in T \\ (x,y) \in A}} \phi(x, y).$$

Let s, t be distinct vertices. An $s - t$ flow is a function $f : A \to \mathbb{R}$ such that

$$f(v, V \setminus \{v\}) = f(V \setminus \{v\}, v) \quad \text{for all } v \neq s, t.$$

In words: flow into v equals flow out of v.

An $s - t$ flow is *feasible* if

$$0 \leq f(x, y) \leq c(x, y) \quad \text{for all } (x, y) \in A.$$

An $s - t$ *cut* is a partition of V into two sets $S, \bar{S}$ such that $s \in S$ and $t \in \bar{S}$.

The *value* v_f of the flow f is given by

$$v_f = f(s, V \setminus \{s\}) - f(V \setminus \{s\}, s).$$

Thus v_f is the net flow leaving s.

The *capacity* of the cut $S : \bar{S}$ is equal to $c(S, \bar{S})$.

8.3.1 Max-Flow Min-Cut Theorem

Theorem 27.

$$\max v_f = \min c(S, \bar{S})$$

where the maximum is over feasible $s - t$ flows and the minimum is over $s - t$ cuts.

Proof We observe first that

$$\begin{aligned} f(S, \bar{S}) - f(\bar{S}, S) &= (f(S, V) - f(S, S)) - (f(V, S) - f(S, S)) \\ &= f(S, V) - f(V, S) \\ &= v_f + \sum_{v \in S \setminus \{s\}} (f(v, V) - f(V, v)) \\ &= v_f. \end{aligned}$$

So,

$$v_f \leq f(S, \bar{S}) \leq c(S, \bar{S}).$$

This implies that

$$\max v_f \leq \min c(S, \bar{S}). \tag{75}$$

Given a flow f we define a *flow augmenting path* P to be a sequence of distinct vertices $x_0 = s, x_1, x_2, \ldots, x_k = t$ such that for all i, either

F1 $(x_i, x_{i+1}) \in A$ and $f(x_i, x_{i+1}) < c(x_i, x_{i+1})$, or

F2 $(x_{i+1}, x_i) \in A$ and $f(x_{i+1}, x_i) > 0$.

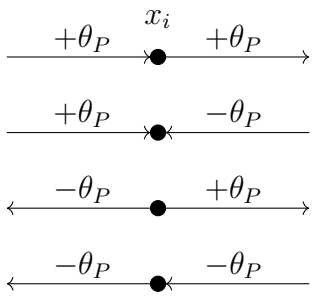
If P is such a sequence, then we define $\theta_P > 0$ to be the minimum over i of $c(x_i, x_{i+1}) - f(x_i, x_{i+1})$ (Case (F1)) and $f(x_{i+1}, x_i)$ (Case (F2)).

Claim 1: f is a maximum value flow, iff there are no flow augmenting paths.

Proof. If P is flow augmenting then define a new flow f' as follows:

1. $f'(x_i, x_{i+1}) = f(x_i, x_{i+1}) + \theta_P$ or
2. $f'(x_{i+1}, x_i) = f(x_{i+1}, x_i) - \theta_P$
3. For all other edges, (x, y), we have $f'(x, y) = f(x, y)$.

We can see that the flow stays balanced at x_i.



We can see then that if there is a flow augmenting path then the new flow satisfies

$$v_{f'} = v_f + \theta_P > v_f,$$

and so f is not a maximum flow.

Suppose now that f does not admit any flow augmenting paths. Let S_f denote the set of vertices v for which there is a sequence $x_0 = s, x_1, x_2, \ldots, x_k = v$ which satisfies F1, F2 of the definition of flow augmenting paths.

If $t \in S_f$ then the associated sequence defines a flow augmenting path. So, $t \notin S_f$. Then we have,

1. $s \in S_f$.
2. If $x \in S_f, y \in \bar{S}_f, (x, y) \in A$ then $f(x, y) = c(x, y)$, else we would have $y \in S_f$.
3. If $x \in S_f, y \in \bar{S}_f, (y, x) \in A$ then $f(y, x) = 0$, else we would have $y \in S_f$.

We therefore have

$$\begin{aligned} v_f &= f(S_f, \bar{S}_f) - f(\bar{S}_f, S_f) \\ &= c(S_f, \bar{S}_f). \end{aligned}$$

We see from this and (75) that f is a flow of maximum value and that the cut $S_f : \bar{S}_f$ is of minimum capacity.

This finishes the proof of Claim 1 and the Max-Flow Min-Cut theorem. □

Note also that we can construct S_f by beginning with $S_f = \{s\}$ and then repeatedly adding any vertex $y \notin S_f$ for which there is $x \in S_f$ such that F1 or F2 holds. (A simple inductive argument based on sequence length shows that all of S_f is constructed in this way.)

This defines an algorithm for finding a maximum flow. The construction either finishes with $t \in S_f$ and we can augment the flow.

Or, we find that $t \notin S_f$ and we have a maximum flow.

Note, that if all the capacities $c(x, y)$ are integers and we start with the all zero flow then we find that θ_P is always a positive integer (formally one can use induction to verify this).

It follows that in this case, there is always a maximum flow that only takes integer values on the edges.

8.3.2 Hall's Theorem.

Let $G = (A, B, E)$ be a bipartite graph with $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$. A matching M is a set of edges that meets each vertex at most once. A matching is perfect if it meets each vertex.

Hall's theorem:

Theorem 28. *G contains a perfect matching iff $|N(S)| \geq |S|$ for all $S \subseteq A$.*

Here $N(S) = \{b \in B : \exists a \in S \text{ s.t. } \{a, b\} \in E\}$.

Define a digraph Γ by adding vertices $s, t \notin A \cup B$. Then add edges (s, a_i) and (b_i, t) of capacity 1 for $i = 1, 2, \ldots, n$. Orient the edges E for A to B and give them capacity ∞.

G has a matching of size m iff there is an $s - t$ flow of value m.

An $s - t$ cut $X : \bar{X}$ has capacity

$$|A \setminus X| + |B \cap X| + |\{a \in X \cap A, b \in B \setminus X : \{a, b\} \in E\}| \times \infty.$$

It follows that to find a minimum cut, we need only consider X such that

$$\{a \in X \cap A, b \in B \setminus X : \{a, b\} \in E\} = \emptyset. \tag{76}$$

For such a set, we let $S = A \cap X$ and $T = X \cap B$. Condition (76) means that $T \supseteq N(S)$. The capacity of $X : \bar{X}$ is now $(n - |S|) + |T|$ and for a fixed S this is minimised for $T = N(S)$.

Thus, by the Max-Flow Min-Cut theorem

$$\max\{|M|\} = \min_X \left\{c(X : \bar{X})\right\} = \min_S \{n - |S| + |N(S)\}.$$

This implies Hall's theorem.

8.3.3 Graph orientation problem

Let $G = (V, E)$ be a graph. When is it possible to orient the edges of G to create a digraph $\Gamma = (V, A)$ so that every vertex has out-degree at least d. We say that G is d-orientable.

Theorem 29. *G is d-orientable iff*

$$|\{e \in E : e \cap S \neq \emptyset\}| \geq d|S| \text{ for all } S \subseteq V. \tag{77}$$

Proof. If G is d-orientable then

$$|\{e \in E : e \cap S \neq \emptyset\}| \geq |\{(x,y) \in A : x \in S\}| \geq d|S|.$$

Suppose now that (77) holds. Define a network D as follows; the vertices are s, t, V, E – yes, D has a vertex for each edge of G.

There is an edge of capacity d from s to each $v \in V$ and an edge of capacity one from each $e \in E$ to t. There is an edge of infinite capacity from $v \in V$ to each edge e that contains v.

Consider an integer flow f. Suppose that $e = \{v, w\} \in E$ and $f(e, t) = 1$. Then either $f(v, e) = 1$ or $f(w, e) = 1$. In the former we interpret this as orienting the edge e from v to w and in the latter from w to v.

Under this interpretation, G is d-orientable iff D has a flow of value $d|V|$.

Let $X : \bar{X}$ be an $s - t$ cut in D. Let $S = X \cap V$ and $T = X \cap E$.

To have a finite capacity, there must be no $x \in S$ and $e \in E \setminus T$ such that $x \in e$.

So, the capacity of a finite capacity cut is at least

$$d(|V| - |S|) + |\{e \in E : e \cap S \neq \emptyset\}|$$

And this is at least $d|V|$ if (77) holds. □

8.3.4 0-1 Matrices

Theorem 30. *Let $a_1, \ldots, a_m$ and $b_1, \ldots, b_n$ be two sets of non-negative integers where $b_1 \geq \cdots \geq b_n$. Then there is an $m \times n$ $0-1$ matrix $M = (M_{i,j})$ satisfying*

$$\sum_{i=1}^{m} M_{i,j} = b_j, j \in [n] \quad \text{and} \quad \sum_{j=1}^{n} M_{i,j} \leq a_i, i \in [m] \tag{78}$$

iff

$$\sum_{j=1}^{k} b_j \leq \sum_{i \in A_k} a_i + k(m - |A_k|), \quad k = 1, \ldots, n, \tag{79}$$

where $A_k = \{i : a_i < k\}$

Proof. Suppose first that the matrix M exists. Fix k and observe that the number of 1's in the first k columns is $b_1 + \cdots + b_k$.

On the other hand the number of 1's in the first k columns is at most $\sum_{i \in A_k} a_i + k(m - |A_k|)$ and so (79) holds.

Now suppose that (79) holds. Define a network N as follows; the vertices are s, t, R, C where $R = \{r_1, \ldots, r_m\}$, $C = \{c_1, \ldots, c_n\}$.

There is an edge of capacity b_i from s to $r_i, i \in [n]$; an edge of capacity a_j from c_j to $t, j \in [m]$; an edge of capacity 1 from r_i to c_j.

Then matrix M exists if there is a flow f of value $b_1 + \cdots + b_n$ from s to t. It is defined by $M_{i,j} = f(r_i, c_j)$.

Let $X : \bar{X}$ be an $s - t$ cut and let $S = X \cap R$, $T = X \cap C$ where $|S| = k$. The capacity of $X : \bar{X}$ is

$$\begin{aligned}
&\sum_{i \notin S} b_i + \sum_{j \in T} a_j + |S|(m - |T|) \\
&\geq \sum_{i=k+1}^{n} b_i + \sum_{j \in A_k} a_j + k(m - |A_k|) \\
&= \sum_{i=1}^{n} b_i + \left(\sum_{j \in A_k} a_j + k(m - |A_k|) - \sum_{i=1}^{k} b_i \right) \\
&\geq \sum_{i=1}^{n} b_i,
\end{aligned}$$

as we have assumed that (79) holds. Applying the Max-Flow Min-Cut theorem, we see that there is a flow of value $b_1 + \cdots + b_n$. □

8.3.5 Polynomial time algorithm

As described, there are instances where the capacities are irrational and the Ford-Fulkerson algorithm does not even converge to the optimum. Edmonds and Karp proved

Theorem 31. *If we always find a shortest augmenting path then the augmenting path algorithm runs in time $O(|V|\,|E|^2)$.*

We discuss this next. Given an $s - t$ flow f we let D_f denote the digraph consisting of edges (x, y) for which $f(x, y) < c(x, y)$ or $f(y, x) > 0$. For $v, w \in V$ we let $d_f(v, w)$ denote the shortest distance (in number of edges) from v to w in D_f.

Lemma 32. *Suppose that we always augment along a shortest augmenting path. Then an augmentation does not decrease $d_f(s, v)$ or $d_f(v, t)$.*

Proof. Suppose, for the sake of contradiction, that $d_f(s, x)$ decreases after an augmentation that replaces f by g and that x is the closest vertex to s whose distance from s decreases. Consider any y, a predecessor to x on a shortest $s - x$ path in D_g. Since $d_g(s, y) \geq d_f(s, y)$ (by closestness of x) but x became closer, the edge (y, x) must be $D_g \setminus D_f$. This means that the augmentation sent flow along (x, y) in D_f, which in turn means that $d_f(s, x) < d_f(s, y)$ (by our choice of augmenting path). Thus,

$$d_g(s, x) < d_f(s, x) < d_f(s, y) \leq d_g(s, y).$$

But since we chose y to be a predecessor of x in D_g, $d_g(s, x) \geq d_g(s, y)$, contradiction. The proof for $d_f(v, t)$ is similar. □

Lemma 33. *An edge can be saturated at most $(n-1)/2$ times during the entire execution of the algorithm.*

Proof. Let (v, w) be an edge saturated in one iteration. Before (v, w) can be saturated again, it must be involved in another augmentation. In the first augmentation on (v, w), v is closer to s than w. During the augmentation on (w, v), w is closer to s than v. Finally, to augment on (v, w) a second time, v must once again be closer to s than w. Lemma 32 says that the distances to s are non-decreasing, so we have that (in an obvious change of notation)

$$d_1(s, v) < d_1(s, w) \leq d_2(s, w) < d_2(s, v) \leq d_3(s, v).$$

Distances are integral, so $d(s, v)$ must have increased by at least 2 by the second augmentation of (v, w) □

Now each augmentation saturates at least one edge and so there can be at most $(n-1)|E|/2$ augmentations, each requiring $O(|E|)$ time. This proves Theorem 31.

8.4 Matroids and the Greedy Algorithm

Given a *ground set* E, an *independence system* on E is a collection of subsets $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$ such that

$$I \in \mathcal{I} \text{ and } J \subseteq I \text{ implies that } J \in \mathcal{I}. \tag{80}$$

Examples

Ex. 1 The set $\mathcal{M}$ of matchings of a graph $G = (V, E)$.

Ex. 2 The set of (edge-sets of) forests of a graph $G = (V, E)$.

Ex. 3 The set of *stable* sets of a graph G. We say that a set of vertices S is stable if it contains no edges.

Ex. 4 The set of solutions to the $\{0, 1\}$-knapsack problem. Here we are given positive integers $w_1, w_2, \ldots, w_n, W$ and $E = [n]$ and $\mathcal{I} = \left\{S \subseteq [n] : \sum_{i \in S} w_i \leq W\right\}$.

Ex. 5 Let $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n$ be the columns of an $m \times n$ matrix $\mathbf{A}$. Then $E = [n]$ and $\mathcal{I} = \{S \subseteq [n] : \{\mathbf{c}_i, i \in S\}$ are linea

An independence system is a *matroid* if

$$\text{whenever } I, J \in \mathcal{I} \text{ with } |J| = |I| + 1 \text{ there exists } e \in J \setminus I \text{ such that } I \cup \{e\} \in \mathcal{I}. \tag{81}$$

Only Ex. 2 and 5 above are matroids. To check Ex. 5, let $\mathbf{A}_I$ be the $m \times |I|$ sub-matrix of $\mathbf{A}$ consisting of the columns in I. If there is no $e \in J \setminus I$ such that $I \cup \{e\} \in \mathcal{I}$ then $\mathbf{A}_J = \mathbf{A}_I \mathbf{M}$ for some $|I| \times |J|$ matrix. But then

$$|J| = rank(\mathbf{A}_J) \leq \min\{rank(\mathbf{A}_I), rank(\mathbf{M})\} \leq |I|,$$

contradiction.

To check Ex. 2 we can argue (exercise) that $I \subseteq E$ defines a forest if and only if the columns corresponding to I in the vertex-edge incidence matrix $\mathbf{M}_G$ are linearly independent.
($\mathbf{M}_G$ has a row for each vertex of G and a column for each edge of G. The column $\mathbf{c}_e, e = \{x, y\}$ has a one in row x and a -1 in row y and a zero in all other rows. It doesn't matter which of the two endpoints is viewed as x.)

8.4.1 Greedy Algorithm

Suppose that each $e \in E$ is given a weight $w_e \geq 0$ and that the weight $w(I)$ of an independent set I is given by $w(I) = \sum_{e \in I} w_e$. The problem we discuss is

$$\text{Maximize } w(I) \text{ subject to } I \in \mathcal{I}.$$

Greedy Algorithm:
begin
    Sort $E = \{e_1, e_2, \ldots, e_m\}$ so that $w(e_i) \geq w(e_{i+1})$ for $1 \leq i < m$;
    $S \leftarrow \emptyset$;
    for $i = 1, 2, \ldots, m$;
    begin
        if $S \cup \{e_i\} \in \mathcal{I}$ **then**;
        begin;
            $S \leftarrow S \cup \{e_i\}$;
        end;
    end;
end

Theorem 34. *The greedy algorithm finds a maximum weight independent set* **for all choices of** w *if and only if it is a matroid.*

Proof. Suppose first that the Greedy Algorithm always finds a maximum weight independent set. Suppose that $\emptyset \neq I, J \in \mathcal{I}$ with $|J| = |I| + 1$. Define

$$w(e) = \begin{cases} 1 + \frac{1}{2|I|} & e \in I. \\ 1 & e \in J \setminus I. \\ 0 & e \notin I \cup J. \end{cases}$$

If there does not exist $e \in J \setminus I$ such that $I \cup \{e\} \in \mathcal{I}$ then the Greedy Algorithm will choose the elements of I and stop. But I does not have maximum weight. Its weight is $|I| + 1/2 < |J|$. So if Greedy succeeds, then (81) holds.

Conversely, suppose that our independence system is a matroid. We can assume that $w(e) > 0$ for all $e \in E$. Otherwise we can restrict ourselves to the matroid defined by $\mathcal{I}' = \{I \subseteq E^+\}$ where $E^+ = \{e \in E : w(e) > 0\}$.

Suppose now that Greedy chooses $I_G = e_{i_1}, e_{i_2}, \ldots, e_{i_k}$ where $i_t < i_{t+1}$ for $1 \leq t < k$. Let $I = e_{j_1}, e_{j_2}, \ldots, e_{j_\ell}$ be any other independent set and assume that $j_t < j_{t+1}$ for $1 \leq t < \ell$. We can assume that $\ell \geq k$, for otherwise we can add something from I_G to I to give it larger weight. We show next that $k = \ell$ and that $i_t \leq j_t$ for $1 \leq t \leq k$. This implies that $w(I_G) \geq w(I)$.

Suppose then that there exists t such that $i_t > j_t$ and let t be as small as possible for this to be true. Now consider $I = \{e_{i_s} : s = 1, 2, \ldots, t-1\}$ and $J = \{e_{j_s} : s = 1, 2, \ldots, t\}$. Now there exists $e_{j_s} \in J \setminus I$ such that $I \cup \{e_{j_s}\} \in \mathcal{I}$. But $j_s \leq j_t < i_t$ and Greedy should have chosen e_{j_s} before choosing e_{i_t}. Also, $i_k \leq j_k$ implies that $k = \ell$. Otherwise Greedy can find another element from $I \setminus I_G$ to add. □

The Matroid Polyhedron Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid on the finite ground set E For $x \in \mathbb{R}^E$ and $A \subseteq E$, write

$$x(A) = \sum_{e \in A} x_e.$$

The independent-set polytope of $\mathcal{M}$ is

$$P(\mathcal{M}) = \text{conv}\{\mathbf{1}_I : I \in \mathcal{I}\},$$

where $\mathbf{1}_I$ is the incidence vector of I.

Theorem 35 (Edmonds)**.**

$$P(\mathcal{M}) = \left\{x \in \mathbb{R}^E_{\geq 0} : x(A) \leq r(A) \quad \textit{for every } A \subseteq E\right\}.$$

Proof. Put

$$Q = \left\{x \in \mathbb{R}^E_{\geq 0} : x(A) \leq r(A) \quad \text{for every } A \subseteq E\right\}.$$

If $I \in \mathcal{I}$, then for every $A \subseteq E$,

$$\mathbf{1}_I(A) = |I \cap A| \leq r(A).$$

Thus $\mathbf{1}_I \in Q$ for every $I \in \mathcal{I}$, and hence

$$P(\mathcal{M}) \subseteq Q.$$

The reverse inclusion is more difficult to prove. Fix an arbitrary weight vector $\mathbf{c} = (c_e : e \in E) \in \mathbb{R}^E$ and consider the linear program

$$\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} c_e x_e & & (82)\\
\text{subject to} \quad & \sum_{e \in A} x_e \leq r(A), \qquad & A \subseteq E, & (83)\\
& x_e \geq 0, & e \in E. & (84)
\end{aligned}$$

The dual program is

$$\begin{aligned}
\text{minimize} \quad & \sum_{A \subseteq E} r(A) y_A & & (85)\\
\text{subject to} \quad & \sum_{\substack{A \subseteq E \\ e \in A}} y_A \geq c_e, \qquad & e \in E, & (86)\\
& y_A \geq 0, & A \subseteq E. & (87)
\end{aligned}$$

The primal is feasible because $\mathbf{x} = \mathbf{0}$ is feasible. It is bounded because

$$0 \leq x_e \leq r(\{e\}) \leq 1$$

for every $e \in E$. Thus strong LP duality applies.

Ignore the elements of nonpositive weight and order the positive-weight elements as $c_{e_1} \geq c_{e_2} \geq \cdots \geq c_{e_k} > 0$. Put $c_{e_{k+1}} = 0$ and $A_i = \{e_1, \ldots, e_i\}, 1 \leq i \leq k$.

Run the matroid greedy algorithm in this order. Starting with $I_0 = \varnothing$, define

$$I_i = \begin{cases} I_{i-1} \cup \{e_i\}, & \text{if } I_{i-1} \cup \{e_i\} \in \mathcal{I}, \\ I_{i-1}, & \text{otherwise.} \end{cases}$$

Let $I = I_k$. For every i, the set $I_i = I \cap A_i$ is a maximal independent subset of A_i. Indeed, if $e_j \in A_i \setminus I_i$, then $I_{j-1} \cup \{e_j\}$ was dependent when e_j was considered. Since $I_{j-1} \subseteq I_i$, the set $I_i \cup \{e_j\}$ is also dependent. Thus no element of $A_i \setminus I_i$ can be added to I_i. In a matroid every maximal independent subset of A_i has cardinality $r(A_i)$, so

$$|I \cap A_i| = r(A_i), \qquad i = 1, \ldots, k. \tag{88}$$

Thus the incidence vector $\mathbf{1}_I$ is feasible for the primal. Now define a dual solution $\mathbf{y}$ supported on the chain $A_1 \subset A_2 \subset \cdots \subset A_k$ by

$$y_{A_i} = c_{e_i} - c_{e_{i+1}}, \qquad i = 1, \ldots, k,$$

and set $y_A = 0$ for every other $A \subseteq E$. These variables are nonnegative. For $1 \leq j \leq k$,

$$\sum_{\substack{A \subseteq E \\ e_j \in A}} y_A = \sum_{i=j}^{k} (c_{e_i} - c_{e_{i+1}}) = c_{e_j}.$$

If e has nonpositive weight, then the left-hand side of the corresponding dual constraint is $0 \geq c_e$. Hence $\mathbf{y}$ is dual feasible.

Its objective value is

$$\sum_{A \subseteq E} r(A) y_A = \sum_{i=1}^{k} (c_{e_i} - c_{e_{i+1}}) r(A_i) = \sum_{i=1}^{k} (c_{e_i} - c_{e_{i+1}}) |I \cap A_i| = \sum_{e_j \in I} c_{e_j} = \mathbf{c}^T \mathbf{1}_I.$$

Thus the primal feasible solution $\mathbf{1}_I$ and the dual feasible solution $\mathbf{y}$ have the same objective value. By strong LP duality, both are optimal, i.e. for every $\mathbf{c} \in \mathbb{R}^E$,

$$\max_{\mathbf{x} \in Q} \mathbf{c}^T \mathbf{x} = \max_{I \in \mathcal{I}} \sum_{e \in I} c_e. \tag{89}$$

We can now prove that every vertex of Q is integral. Let $\mathbf{v}$ be a vertex of Q. Since Q is a polytope, there is a vector $\mathbf{c} \in \mathbb{R}^E$ for which $\mathbf{v}$ is the unique maximizer of $\mathbf{c}^T \mathbf{x}$ over Q. By (89), some incidence vector $\mathbf{1}_I$, with $I \in \mathcal{I}$, is also optimal. By uniqueness, $\mathbf{v} = \mathbf{1}_I$. Hence every vertex of Q is the incidence vector of an independent set. Since a polytope is the convex hull of its vertices,

$$Q \subseteq \operatorname{conv}\{\mathbf{1}_I : I \in \mathcal{I}\} = P(\mathcal{M}).$$

Together with $P(\mathcal{M}) \subseteq Q$, this proves the theorem. □

9 Dynamic Programming

Dynamic programming is an approach to solving problems rather than a technique for solving a particular problem. The approach can be applied to a wide range of problems, although in many cases it leads to impractical algorithms.

The problem to be tackled is formulated as making a sequence of decisions. Having made one decision, the problem of choosing the remaining decisions is often a similar but smaller version of the original problem. This can lead to a *functional equation* for finding the best initial decision and each subsequent decision.

9.1 Simple Production Problem

As a simple example we consider the following problem: a company estimates the demand d_j for one of its products over the next n periods. It costs the company $c(x)$ to manufacture x units in any one period. All demand must be met in the period in which it occurs but stocks may be built up to provide for demand in future periods. The maximum stock that can be held at any time is H. How much should be produced in each period to minimise the total cost of production. To make the problem self-contained we have to say something about initial and final stocks. Suppose then that there is an initial stock of i_0 and that any stock left over at the end of period n is worthless.

The problem then is to decide how much to produce in period 1, how much to produce in period 2 etc. Suppose we decide to produce an amount x_1, in period 1, then at the beginning of period 2 we will have a stock level of $i_0 + x_1 - d_1$ and the problem of minimising the production cost over the next $n-1$ periods. We can write this down mathematically. Define the quantity $f_r(i)$ to be the minimum cost of meeting demand in periods $r, r+1 \ldots, n$ given that one has i units in stock at the beginning of period r.

Focussing temporarily on period 1, we can ask the question, if we decide to produce an amount x, in period 1, what is the minimum production cost obtainable over the whole n periods? This minimum cost is clearly

$$c(x_1) + f_2(i_0 + x_1 - d_1). \tag{90}$$

The first term is the cost of period:1 and the second term is the minimum cost over periods $2, 3, \ldots, n$ given that we produced x_1.

The next question is what is the best value of x, to take. The answer must be, the value of x that minimises (90), This will give us the minimum production cost for periods $1, 2, \ldots, n$, starting with a stock i_0, i.e. $f_1(i_0)$. We have thus proved that

$$f_1(i_0) = \min_{x_1} \left\{ c(x_1) + f_2(i_0 + x_1 - d_1) \right\}. \tag{91}$$

A similar argument about the decision to be taken at the beginning of period r given that the stock level is currently i shows that in general

$$f_r(i) = \min_{x_r} \left\{ c(x_r) + f_{r+1}(i + x_r - d_r) \right\}. \tag{92}$$

The range over which the *decision variable* x_r is to be minimised depends on our assumptions about the problem. Firstly, we must have $x_r \geq 0$ and since we must produce enough to meet the demand d_r, we must have $i + x_r \geq d_r$. The maximum stock level is H and consequently we must have $i + x_r - d_r \leq H$. Thus x_r, is to be chosen in the range

$$\max\{0, d_r - i\} \leq x_r \leq H + d_r - i. \tag{93}$$

Now the argument that produced (92) only really holds true for $r \leq n-1$, basically because we have not defined $f_{n+1}(i)$. Examining our assumption about final stocks we can see that this is equivalent to

$$f_n(i) = \min_{x_n \geq (d_n - i)^+} c(x_n). \tag{94}$$

This can be put into the framework of (92) by defining $f_{n+1}(i) = 0$. Equations (92) and (94) give us a means of solving our problem. We first calculate $f_n(i)$ for $i = 0, 1, 2, \ldots, H$. We then use (92) to calculate $f_{n-1}(i)$ for $i = 0, 1, 2, \ldots, H$, and then $f_{n-2}(i)$ and so on until we reach $f_1(i)$. If the production quantities x need not be integral then we have to approximate by dividing the range [0,H] into a suitable number of points - depending on the accuracy required and computer storage and time available.

Let us solve the above problem when $n = 4, d_j = 3$ in all periods, the maximum stock level $H = 4$ and $c(x) = 18x - x^2$.

So that we can keep track of the optimal production policy we make a note of the value of x minimising the R.H.S of (92). Denote this value by $x_r(i)$.

Calculation of f_4 By definition $f_4(i) = \min\{18x - x^2 : \max\{0, 3-i\} \leq x \leq 7-i\}$.

$$\begin{array}{ll} f_4(0) = 45 & x_4(0) = 3 \\ f_4(1) = 32 & x_4(1) = 2 \\ f_4(2) = 17 & x_4(2) = 1 \\ f_4(3) = 0 & x_4(3) = 0 \\ f_4(4) = 0 & x_4(4) = 0 \end{array}$$

Calculation of f_3 By definition $f_3(i) = \min\{18x - x^2 + f_4(i+x-3) : \max\{0, 3-i\} \leq x \leq 7-i\}$.

$$f_3(0) = \min\{45 + f_4(0), 56 + f_4(1), 65 + f_4(2), 72 + f_4(3), 77 + f_4(4)\} = 72 \text{ and } x_3(0) = 6.$$

Continuing this we build up the table

i	$f_4(i)$	$x_4(i)$	$f_3(i)$	$x_3(i)$	$f_2(i)$	$x_2(i)$	$f_1(i)$	$x_1(i)$
0	45	3	72	6	109	7	142	7
1	32	2	65	5	104	2/6	137	5/6
2	17	1	56	4	89	1	126	1
3	0	0	45	0/3	72	0	109	0
4	0	0	32	0/2	65	0	104	0/2

Suppose for example that the initial stock level in period 1 is 0. We see from the table that the minimum total production cost is 142. The optimal production policy is found as follows: $x_1(0) = 7$ i.e. given a stock level of 0 at the beginning of period 1 the optimum production for period 1 is 7. Producing 7 in period 1 means we start period 2 with a stock level 4. From the table $x_2(4) = 0$ i.e. given a stock level of at the beginning of period 2 the optimum production for period 2 is 0. This means we start period 3 with stock level 1. Now $x_3(1) = 5$, so we produce 5 units in period 3 and therefore start period 4 with initial stock 3. As $x_4(3) = 0$ we produce nothing in this period. Thus the optimal policy starting period 1 with zero stock is

x_1	x_2	x_3	x_4
7	0	5	0

We may in a similar manner use the table to find the optimum policy for all possible initial stock levels. In the method above we have worked backwards from period n in calculating the optimum policy. This is called the backward formulation of the problem.

In the backward formulation model we need to be explicit on what happened to the final stock, in the forward formulation we have to fix the initial stock at some value. For simplicity assume the initial stock is zero.

Now let us define the quantity $g_r(i)$ to be the minimum cost of meeting demand in periods $1, 2, \ldots, r$ given that the stock level at the end of period r is i. Then arguing in a similar manner to the backward formulation we get

$$g_1(i) = c(i + d_1) \tag{95}$$

$$g_r(i) = \min_{x_r} \{c(x_r) + g_{r-1}(i + d_r - x_r)\} \tag{96}$$

where x_r in (96) ranges over

$$\max\{0, i + d_r - H\} \leq x_r \leq i + d_r.$$

Starting with g_1 as defined in (95) we use (96) iteratively to calculate g_n and we can thus calculate an optimum for any value of the final stock.

Holding Cost We now add a cost of holding non-zero inventory. Suppose that this cost is $h(i_1, i_2)$, where i_1 is the inventory at the start of the period and i_2 is the inventory at the end of the period. Then we replace (92) by

$$f_r(i) = \min_{x_r} \{c(x_r) + h(i, i + x_r - d_r) + f_{r+1}(i + x_r - d_r)\}.$$

Backordering Suppose we are allowed to be B behind in filling the demand, then the changes to (92) are that we allow $i \in [-B, H]$ and so (93) is replaced by

$$\max\{0, d_r - i - B\} \leq x_r \leq H + d_r - i.$$

Smoothing Production levels In the example we computed, the production changed significantly from period to period. Suppose we do not like that and that we add a cost $s(x_{r-1}, x_r)$. We replace (92) by

$$f_r(x, i) = \min_{x_r} \{c(x_r) + s(x, x_r) + f_{r+1}(x_r, i + x_r - d_r)\}.$$

Here x represents the production level for the previous period.

9.2 Replacement of a machine

A company uses a machine to manufacture a single product over the next N periods. The demand in period n is known to be d_n and the maximum amount of stock that can be held at one time is H. The cost of producing an amount x depends on the current age of the machine. It costs $c(x, t)$ to produce an amount x using a machine of age t. A machine of age T has to be scrapped. Assume that we start in period 0 with a new machine. A new machine costs A to buy. Here is how we formulate the problem: Let $f_n(t, h)$ denote the minimum cost of meeting demand in periods $n, n+1, \ldots, N$ if we start period n with a machine of age t and h units in stock. Then, assuming that $f_{N+1}(t, i) = 0$ for all t, i,

$$f_n(t, h) = \min \begin{cases} \min\limits_{\substack{0 \leq x \leq H-h+d_n \\ x \geq d_n - h}} \{c(x, t) + f_{n+1}(t + 1, x + h - d_n)\} & \text{Keep old machine} \\ \min\limits_{\substack{0 \leq x \leq H-h+d_n \\ x \geq d_n - h}} \{A + c(x, 0) + f_{n+1}(1, x + h - d_n)\} & \text{Replace machine} \end{cases}$$

The above recurrence is computed for $n = N, N-1, \ldots, 1$, $t = 0, 1, \ldots, T-1$ and $h = 0, 1, \ldots, H$. If $t = T$ then we let

$$f_n(T, h) = A + f_n(0, h).$$

9.3 Probabilistic production problem

A company needs to meet demand for its single product over the next N periods. The cost of producing an amount x is $c(x)$ in any period. The demand is a random variable and let us assume that

$$\mathbb{P}(d_n = d) = p_{n,d} \qquad d \geq 0.$$

The company can store up to amount H at any time. The company will try to meet the demand, but if it is too large then there is a penalty cost of π for any demand left unsatisfied. The company wishes to minimise the expected cost of production. Assume first that the company has to make its period n production decision *before* it knows d_n. Let $f_n(h)$ denote the minimum expected cost of production in periods $n, n+1, \ldots, N$ if we start period n with h units in stock. Then, if $\xi^+ = \max\{0, \xi\}$ and assuming that $f_{N+1}(i) = 0$ for all i,

$$f_n(h) = \min_{x \geq 0} \left(c(x) + \sum_{d \geq 0} p_{n,d}(f_{n+1}(\min\{(x+h-d)^+, H\}) + \pi \max\{0, d-(h+x)\}) \right).$$

As an alternative criterion, suppose one has to minimise expected cost subject to having at least a 90% chance of meeting demand in every period. Then we let $f_n(h)$ be the minimum cost of operating under these criteria for a given n and h.

$$f_n(h) = \min_{x \geq \alpha_h} \{c(x) + \sum_{d \geq 0} p_{n,d}(f_{n+1}(\min\{(x+h-d)^+, H\}) + \pi(d-(h+x))^+)\}$$

where $\alpha_h = \min\left\{\alpha : \sum_{d > \alpha + h} p_{n,d} \leq .1\right\}$.

If the company can make its period n production decision *after* it knows d_n then we have

$$f_n(h) = \sum_{d \geq 0} p_{n,d} \min_{\substack{x \geq (d-h)^+ \\ x \leq H+d-h}} \{c(x) + f_{n+1}(h+x-d)\}.$$

9.4 Knapsack Problem

$w_1, w_2, \ldots, w_n \leq W, c_1, c_2, \ldots, c_n$ are positive integers.
Problem

$$\begin{aligned} \text{Maximize} & \sum_{j=1}^{n} c_j x_j \\ \text{Subject to} & \sum_{j=1}^{n} w_j x_j \leq W \\ & x_j \geq 0 \text{ and integer } j = 1, 2, \ldots, n \end{aligned}$$

A scout is going on a trip and has a knapsack that can hold at most W pounds. They have to pack items of type $1, 2, \ldots, n$. Each item of type j weighs w_j pounds and has value c_j. They can take more than one of each item and they want to maximise the value of the items packed.

Let now $f_r(w)$ denote the maximum to the above integer program when W is replaced by w and n is replaced by r. Let $f_r(w) = -\infty$ if $w < 0$.
First recurrence:

$$f_r(w) = \max_{0 \leq x \leq \lfloor w/w_r \rfloor} \{c_r x + f_{r-1}(w - w_r x)\}.$$

For any choice of $x = x_r$ we gain $c_r x$ plus the optimum available from the remaining variables $x_1, x_2, \ldots, x_{r-1}$.

Second recurrence:

$$f_r(w) = \max \begin{cases} f_{r-1}(w) & x_r = 0 \text{ in optimum.} \\ c_r + f_r(w - w_r) & x_r \geq 1 \text{ in optimum.} \end{cases}$$

Example:

$$\begin{aligned} &\text{Maximize } 2x_1 + 3x_2 + 5x_3 + 7x_4 \\ &\text{Subject to } 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 12 \\ &\qquad x_1, \ldots, x_4 \geq 0 \text{ and integer} \end{aligned}$$

w	f_1	δ_1	f_2	δ_2	f_3	δ_3	f_4	δ_4
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	2	1	2	0	2	0	2	0
3	2	1	3	1	3	0	3	0
4	4	1	4	0	5	1	5	0
5	4	1	5	1	5	0/1	7	1
6	6	1	6	0/1	7	1	7	0/1
7	6	1	7	1	8	1	9	1
8	8	1	8	0/1	10	1	10	0/1
9	8	1	9	1	10	1	12	1
10	10	1	10	0/1	12	1	14	1
11	10	1	11	1	13	1	14	1
12	12	1	12	0/1	15	1	16	1

Solution Let $x_r(w)$ denote the value of x_r in the optimum solution for w. $\delta_r(w) = 1$ iff $x_r(w) \geq 1$.

$\delta_4(12) = 1$ and so $x_4(12) \geq 1$. $\delta_4(7) = 1$ and so $x_4(7) \geq 1$. $\delta_4(2) = 0$ and so $x_4(12) = 2$.
$\delta_3(2) = 0$ and so $x_3(2) = 0$.
$\delta_2(2) = 0$ and so $x_2(2) = 0$.
$\delta_1(2) = 1$ and so $x_1(2) = 1$.

Thus the solution is $x_1 = 1, x_2 = x_3 = 0, x_4 = 2$.

Execution time The above algorithm requires $O(nW)$ arithmetic operations. $O(1)$ for each choice of r, w.

Another recurrence: let $f(w) = -\infty$ if $w < 0$ and

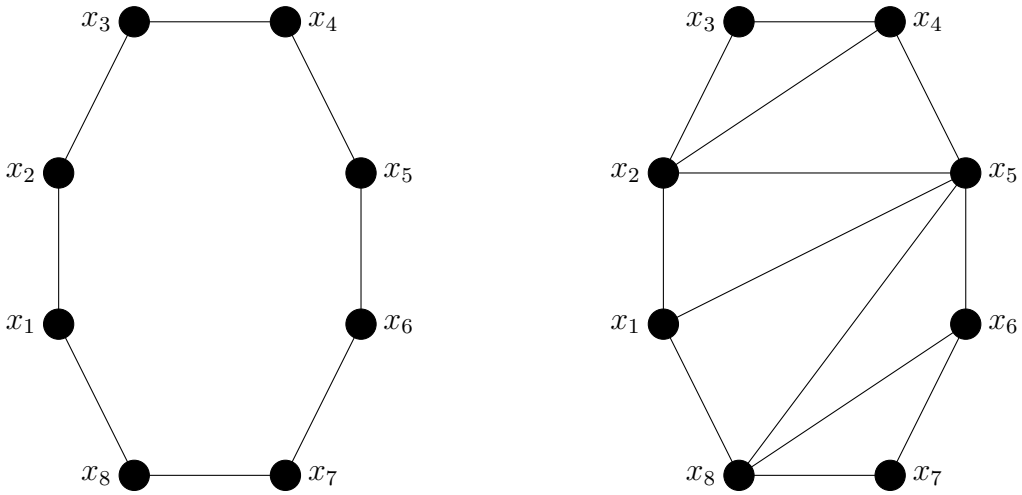
$$\begin{aligned} f(w) &= \text{max.} \sum_{j=1}^{n} c_j x_j \\ \text{Subject to} &\sum_{j=1}^{n} w_j x_j \leq w \\ &x_j \geq 0 \text{ and integer } j = 1, 2, \ldots, n \end{aligned}$$

Then

$$f(w) = \max_{j=1,2,\ldots,n} (c_j + f(w - w_j)).$$

9.5 Minimal triangulation of a convex polygon

Let P be a convex polygon with vertices $X_1, X_2, \ldots, X_n$. We want to triangulate it in such a way as to minimise the sum of the lengths of the chords used.



Let $m^*_{k,l}$ be the sum of the perimiter of the polygon plus the length of the minimum length triangulation of the polygon defined by $X_k, X_{k+1}, \ldots, X_l, X_k$. Then

$$m^*_{k,l} = \min_{k<j<l} \{m^*_{k,j} + m^*_{j,l} + |X_k - X_l|\} \tag{97}$$

where $|X_k - X_j|$ is the length of the edge X_k, X_j etc.

Here $m^*_{k,k+1} = |X_k - X_{k+1}|$ and we use the recurrence (97) to compute what we want i.e. $m^*_{1,n}$.

9.6 Breaking up a stick

A stick of length L is to be broken into pieces of integer length. Let $v_{i,j}$ be the *value* of a piece $[i, i+1, \ldots, j]$. How should the stick be broken in order to maximise the total value.

Let $f(r), r = 0, 1, \ldots, L$ be the maximum value obtainable from breaking up $[0, r]$. Then

$$f(0) = 0 \text{ and } f(r) = \max_{0 \le i < r} \{f(i) + v_{i,r}\}, \qquad 0 < r \le L.$$

So $f(L)$ can be computed in $O(L^2)$ operations.

Suppose now that the stick must be broken into k pieces. Now use $f(j, r)$ to be the maximum obtainable from breaking $[0, r]$ into j pieces. Then,

$$f(j, r) = \begin{cases} -\infty & j > r. \\ \max_{0 \le i < r} \{f(j-1, i) + v_{i,r}\} & j \le r. \end{cases}$$

So $f(k, L)$ can be computed in $O(kL^2)$ operations.

9.7 A problem with an infinite time horizon

A *system* can be in one of a set V of possible states. For each $v \in V$ one can choose any $w \in V$ and move to w at a cost of $c(v,w)$. The system is to run *forever* and it is required to minimise the *discounted cost* of running the system, assuming that the discount factor is $0 \leq \alpha < 1$. A *policy* is a function $\pi : V \to V$. So if $|V| = n$ then there are n^n distinct policies to choose from.

Example

$$\text{Costs} \begin{bmatrix} 2 & 1 & 3 \\ 4 & 3 & 2 \\ 1 & 3 & 2 \end{bmatrix} \quad \alpha = 1/2.$$

Let π be a policy and let y_v be the discounted cost of this policy, starting at $v \in V$. Then

$$y_v = c(v, \pi(v)) + \alpha y_{\pi(v)} \qquad v \in V. \tag{98}$$

Example Let $\pi(1) = \pi(2) = \pi(3) = 1$. Then

$$\begin{aligned} y_1 &= 2 + \frac{1}{2} y_1 \\ y_2 &= 4 + \frac{1}{2} y_1 \\ y_3 &= 1 + \frac{1}{2} y_1. \end{aligned}$$

So

$$y_1 = 4,\ y_2 = 6,\ y_3 = 3.$$

Problem: Find the policy π^* which minimises y_v simultaneously for all $v \in V$.

Theorem 36. Optimality Criterion
π^* is optimal iff its values y_v^* satisfy

$$y_v^* = \min_{w \in V} \{c(v,w) + \alpha y_w^*\} \qquad \forall v \in V. \tag{99}$$

Proof Suppose that (99) does not hold for some π.

$$\begin{aligned} y_u &> c(u, \lambda(u)) + \alpha y_{\lambda(u)} \qquad u \in U \\ y_u &= \min_{w \in V} \{c(u,w) + \alpha y_w\} \qquad u \notin U \end{aligned}$$

Define $\tilde{p}$ by $\tilde{p}(u) = \lambda(u)$ for $u \in U$ and $\tilde{p}(v) = \pi(v)$ for $v \notin U$. Then for $u \in U$,

$$\begin{aligned} y_u &> c(u, \lambda(u)) + \alpha y_{\lambda(u)} \\ \tilde{y}_u &= c(u, \lambda(u)) + \alpha \tilde{y}_{\lambda(u)} \end{aligned}$$

So if $\xi_v = y_v - \tilde{y}_v$ for $v \in V$ then

$$\xi_u > \alpha \xi_{\tilde{p}(u)} \qquad u \in U. \tag{100}$$

Also, for $v \notin U$

$$\begin{aligned} y_v &= c(v, \pi(v)) + \alpha y_{\pi(v)} \\ \tilde{y}_v &= c(v, \pi(v)) + \alpha \tilde{y}_{\pi(v)} \end{aligned}$$

and so

$$\xi_v = \alpha \xi_{\tilde{p}(v)} \qquad v \notin U. \tag{101}$$

It follows from (100), (101) that

$$\begin{aligned} \xi_v &\geq \alpha^t \xi_{\tilde{p}^t(v)} \qquad \forall v \notin U, t \geq 1 \\ \xi_u &> \alpha^t \xi_{\tilde{p}^t(u)} \qquad \forall u \in U, t \geq 1 \end{aligned}$$

Letting $t \to \infty$ we see that

$$\xi_v \geq 0 \ \forall v \text{ and } \xi_u > 0 \ \forall u \in U.$$

Thus $\tilde{p}$ is *strictly better* than π i.e. if (99) does not hold, then we can improve the current policy.

Conversely, if (99) holds and $\widehat{\pi}$ is any other policy and $\eta_v = \widehat{y}_v - y^*_v$ then

$$\begin{aligned} \widehat{y}_v &= c(v, \widehat{\pi}(v)) + \alpha \widehat{y}_{\widehat{\pi}(v)} \\ y^*_v &\leq c(v, \widehat{\pi}(v)) + \alpha y^*_{\widehat{\pi}(v)} \end{aligned}$$

and so

$$\eta_v \geq \alpha \eta_{\widehat{\pi}(v)} \geq \cdots \geq \alpha^t \eta_{\widehat{\pi}^t(v)} \qquad \text{for } t \geq 1$$

which implies that $\eta_v \geq 0$ for $v \in V$. □

Policy Improvement Algorithm

1. Choose arbitrary initial policy π.
2. Compute y as in (98).
3. If (99) holds – current π is optimal, stop.
4. If (99) doesn't hold then
5. compute λ by
   $c(v, \lambda(v)) + \alpha y_{\lambda(v)} = \min_w \{c(v, w) + \alpha y_w\}$.
6. $\pi \leftarrow \lambda$.
7. goto 2.

In our example with $\pi = (1, 1, 1)$. First compute $\lambda = (1, 3, 1)$ and re-compute $y = (4, \frac{7}{2}, 3)$. Now $\lambda = (2, 3, 1)$ and $y = (18/7, 22/7, 16/7)$. Now (99) holds and we are done.

9.7.1 Probabilistic version

Let us introduce some probability: Suppose now that for each $i \in V$ there is a set X_i of possible decisions. Suppose that if the system is in state i and decision $x \in X_i$ is taken then

- The expected cost of the immediate step is $c(x, i)$.
- The next state is j with probability $P(x, i, j)$

A policy π specifies a decision $\pi(i) \in X_i$ for each $i \in V$.

First let us evaluate this policy. Let y_i denote the expected discounted cost of pursuing policy π indefinitely, starting from $i \in V$. Then

$$y_i = c(\pi(i), i) + \alpha \sum_{j \in V} P(\pi(i), i, j) y_j$$

or

$$y = c_\pi + \alpha P_\pi y \text{ or } y = (I - \alpha P_\pi)^{-1} c_\pi = \sum_{t=0}^{\infty} (\alpha P_\pi)^t c_\pi$$

where $P_\pi(i, j) = P(\pi(i), i, j)$ and $c_\pi(i) = c(\pi(i), i)$.

So policy π can be evaluated.

Theorem 37. *Optimality criterion:*

$$c(\pi(i), i) + \alpha \sum_{j \in V} P(\pi(i); i, j) y_j = \min_{x \in X_i} \left\{ c(x, i) + \alpha \sum_{j \in V} P(x, i, j) y_j \right\} \tag{102}$$

π is optimal iff (102) holds.

Proof Suppose first that (102) does not hold. Define a new policy $\widehat{\pi}$ by

$$c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) y_j = \min_{x \in X_i} \left\{ c(x, i) + \alpha \sum_{j \in V} P(x, i, j) y_j \right\}$$

We have

$$\begin{aligned} y_i &\geq c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) y_j \\ \widehat{y}_i &= c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) \widehat{y}_j \end{aligned} \tag{103}$$

and so

$$(I - \alpha P_{\widehat{\pi}})(y - \widehat{y}) \geq 0$$

and then since $(I - \alpha P_{\widehat{\pi}})^{-1}$ has only non-negative entries:

$$(I - \alpha P_{\widehat{\pi}})^{-1}(I - \alpha P_{\widehat{\pi}})(y - \widehat{y}) \geq 0 \text{ or } y - \widehat{y} \geq 0$$

But $\widehat{y} \neq y$ since there is strict inequality in (103) for at least one i and $\widehat{\pi}$ is strictly better than π.

Conversely, if (102) holds and $\widehat{\pi}$ is any other policy, we get that

$$\begin{aligned} y_i &\leq c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) y_j \\ \widehat{y}_i &= c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) \widehat{y}_j \end{aligned}$$

and so

$$(I - \alpha P_{\widehat{\pi}})(y - \widehat{y}) \leq 0$$

and then since $(I - \alpha P_{\widehat{\pi}})^{-1}$ has only non-negative entries:

$$(I - \alpha P_{\widehat{\pi}})^{-1}(I - \alpha P_{\widehat{\pi}})(y - \widehat{y}) \leq 0 \text{ or } y - \widehat{y} \leq 0$$

□

A taxi driver's territory comprises 3 towns A,B,C. If he is in town A he has 3 alternatives:

1. He can cruise in the hope of picking up a passenger by being hailed.
2. He can drive to the nearest cab stand and wait in line.
3. He can pull over and wait for a radio call.

In town C he has the same 3 alternatives, but in town B he only has alternatives 1 and 2.

The transition probabilities and the rewards for being in the various states and making the various transitions are as follows:

A:

$$P = \begin{bmatrix} .5 & .25 & .25 \\ .0625 & .75 & .1875 \\ .25 & .125 & .625 \end{bmatrix} \quad R = \begin{bmatrix} 10 & 4 & 8 \\ 8 & 2 & 4 \\ 4 & 6 & 4 \end{bmatrix}$$

B:

$$P = \begin{bmatrix} .5 & 0 & .5 \\ .0625 & .875 & .0625 \end{bmatrix} \quad R = \begin{bmatrix} 14 & 0 & 18 \\ 8 & 16 & 8 \end{bmatrix}$$

C:

$$P = \begin{bmatrix} .25 & .25 & .5 \\ .125 & .75 & .125 \\ .75 & .0625 & .1875 \end{bmatrix} \quad R = \begin{bmatrix} 10 & 2 & 8 \\ 6 & 4 & 2 \\ 4 & 0 & 8 \end{bmatrix}$$

He wishes to find the policy which maximises his long run average gain per period.

9.8 Traveling Salesperson problem

We are given a matrix of costs $c(i, j), 1 \leq i, j \leq n$. The problem is to find a permutation $\pi = (1 = j_1, j_2, \ldots, j_n)$ of $[n] = \{1, 2, \ldots, n\}$ that minimises

$$TSP(\pi) = c(j_1, j_2) + c(j_2, j_3) + \cdots + c(j_n, j_1).$$

This represents the total cost of a "tour through $[n]$ in the order $j_1, j_2, \ldots, j_n$.

There are $(n-1)!$ distinct tours (each tour, as a set of directed edges of $\vec{K}_n$, arises from n distinct permutations.)

With DP we can solve the problem in $O(n^2 2^n)$ time. For $1 \in S \subseteq [n]$ and $x \in S$, let $f(x, S)$ denote the minimum cost of a path that begins at 1, ends at x and visits each vertex in S exactly once. Then, $f(1, \{1\}) = 0$ and

$$f(x, S) = \min\{f(z, S \setminus \{x\}) + c(z, x) : z \in S \setminus \{1, x\}\}.$$

There are $\binom{n-1}{k-1}$ choices for $|S| = k$ and given S there are $k-1$ choices for x and then $k-2$ choices for z. So, to compute $f(x,[n])$ for all $1 \neq x \in [n]$ takes time

$$n-1+\sum_{k=2}^{n}(k-1)(k-2)\binom{n-1}{k-1} = n-1+\sum_{k=3}^{n}(k-1)(k-2)\binom{n-1}{k-1} =$$
$$n-1+(n-1)(n-2)\sum_{k=3}^{n}\binom{n-3}{k-3} = n-1+(n-1)(n-2)2^{n-3}.$$

To finish we compute $\min\{f(x,[n])+c(x,1) : x \neq 1\}$.

10 P v NP: informally

We want some formal way of saying that a problem is efficiently solvable or otherwise. We are given an algorithm A that solves a set of problems $\mathcal{I}$. A problem will be of the form

$$\text{“does instance } I \in \mathcal{I} \text{ have property } \mathcal{P}\text{”}. \tag{104}$$

Each problem instance $I \in \mathcal{I}$ has a *size* $\sigma(I)$. (Size could be the number of bits needed to describe the problem or more loosely the number vertices in a graph problem or $m+n$ for a linear program in standard form.) Let $T(A, I)$ denote the time (in steps) that A takes to solve I. We say that A runs in *polynomial time* if $T(A, I) = O(\sigma(I)^c)$ where c is some constant independent of I. Then P is the set of problems that can be solved in polynomial time. Examples include “does G have a perfect matching?”; “can G be properly 2-colored?”; “is this set of linear equations solvable?”. An optimization problem can be put into this framework by using binary search on problems of the form “is the minimal value at most L?”. It turns out that Linear Programming is solvable in polynomial time.

P is relatively small compared to the set of problems NP that arise in optimization. A problem (104) is in NP if there is a *witness* $w(I)$ of polynomial size that can be used in polynomial time to verify $I \in \mathcal{I}$. For example, if $\mathcal{I}$ is the set of graphs and $\mathcal{P}$ is the set of graphs with a Hamilton cycle (i.e. one that goes through each vertex exactly once) then $w(I)$ would be such a cycle. Informally, a problem is in NP if when I give you a proposed solution, it is easy to check the truth of this.

Reductions A polynomial time reduction PRT of $\mathcal{I}_1$ to $\mathcal{I}_2$ is a map $\phi : \mathcal{I}_1 \to \mathcal{I}_2$ such that (i) $\sigma(f(I))$ is polynomially related to $\sigma(I)$ and $\phi(I)$ can be computed in polynomial time and (ii) $I \in \mathcal{I}_1$ iff $\phi(I) \in \mathcal{I}_2$. In other words, we can solve an instance of $\mathcal{I}_1$ in polynomial time if we can solve an instance of $\mathcal{I}_2$ in polynomial time. Notice that if there is a PRT of $\mathcal{I}_1$ to $\mathcal{I}_2$ and a PRT of $\mathcal{I}_2$ to $\mathcal{I}_3$ then there is a PRT of $\mathcal{I}_1$ to $\mathcal{I}_3$.

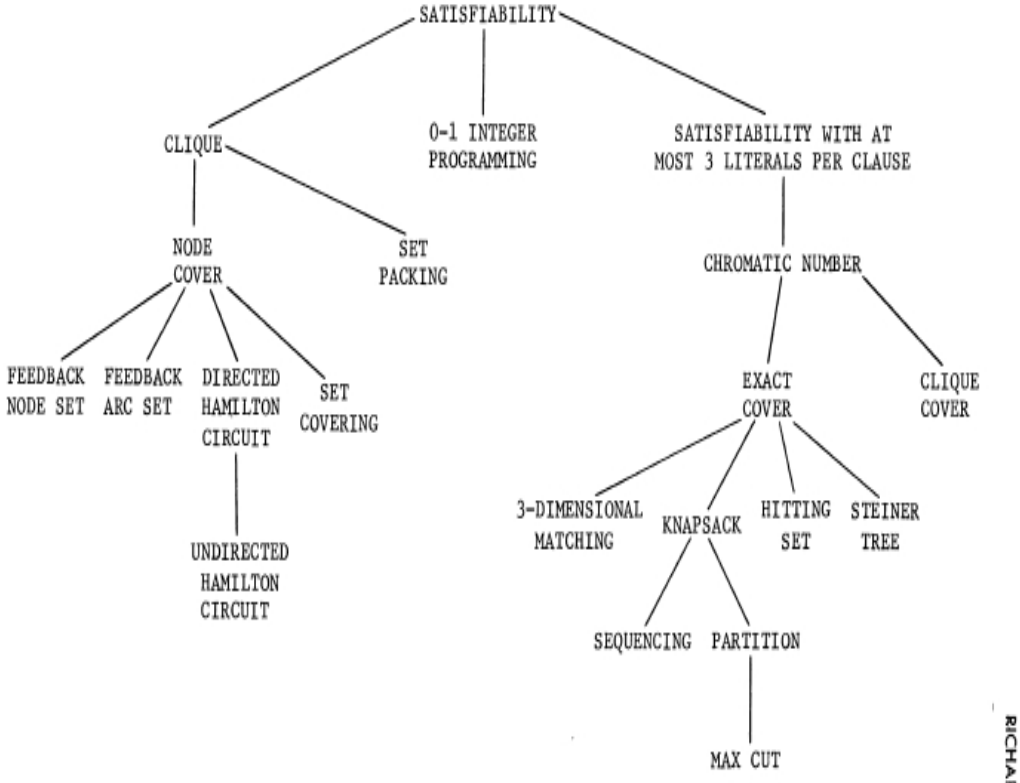


FIGURE 1 - Complete Problems

A problem $\mathcal{I} \in NP$ is said to be *NP-complete* if every problem in NP has a PRT to $\mathcal{I}$. Cook proved that SAT is NP-complete: we have a set $\mathcal{C}$ of m clauses consisting of sets of *literals* x_j or $\bar{x}_j = 1 - x_j, j = 1, 2, \ldots, n$. $\mathcal{C}$ is *satisfiable* if there is an assignment of values 0 or 1 to the x_j such that every clause contains at least one literal of value 1.
Example: $\{x_1, \bar{x}_2, x_3\}, \{\bar{x}_1, x_3, \bar{x}_4\}, \{x_2, x_4\}$ is satisfied by taking $x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 1$. On the other hand $\{x_1, x_2\}, \{x_1, \bar{x}_2\}, \{\bar{x}_1, x_2\}, \{\bar{x}_1, \bar{x}_2\}$ cannot be satisfied.

If $\mathcal{I}_1$ is NP-complete and it has a PRT to $\mathcal{I}_2$ then $\mathcal{I}_2$ is also NP-complete. If there is a polynomial time algorithm for any NP-complete problem then there is a polynomial time algorithm for solving any problem in NP. By now there are thousands of naturally defined NP-complete problems. At present, no one has found a polynomial time algorithm for any of these. It is therefore conjectured that there are no polynomial time algorithms for any NP-complete problems and that $NP \neq P$.

11 Approximation Algorithms

If solving an optimization problem is considered to be hard, then we can sometimes efficiently find approximate solutions with guarantees how far from optimum they are. An α-approximation algorithm for a minimisation problem computes a solution whose value is at most αv^*, where v^* is the minimum value for the problem. (For maximisation we have at least αv^*.)

11.1 Traveling Salesperson Problem – TSP

11.1.1 Unrestricted

There is no polynomial time M-approximation algorithm unless P=NP. Given a graph G, give a cost of 1 to each edge of G and $Mn+1$ to each non-edge. Then, the TSP has a tour of length n iff G is Hamiltonian. Otherwise the cost of the tour is at least $(M+1)n$. An M-approximation algorithm could tell if G is Hamiltonian. If G is Hamiltonian, it produces a tour of length at most $Mn < (M+1)n$, implying that G is Hamiltonian. Otherwise it produces a tour of length at least $(M+1)n$, implying that the minimum length tour is greater than n, so G is not Hamiltonian.

11.1.2 Triangle Inequality

We assume next that the costs $C(i,j), 1 \leq i,j \leq n$ satisfy the *triangular inequality* i.e. $C(i,j)+C(j,k) \geq C(i,k)$ and that $C(i,j) = C(j,i)$ for all i,j.

Tree heuristic

Step 1 Find a minimum cost spanning tree T.

Step 2 Double the edges of T to make an Eulerian multigraph K. (Eulerian because all degrees are even. Such a graph contains a closed walk that goes through each edge exactly once.)

Step 3 Construct an Euler tour $i_1, i_2, \ldots, i_{2n-2}$ through the edges of K.

Step 4 Shortcut the Euler tour until it is a Hamilton cycle H. I.e. go through the vertices $i_1, i_2, \ldots,$ in sequence and skip over any vertex that has already been visited.

Theorem 38. *The tour H found by the tree heuristic satisfies $C(H) \leq 2C^*$, where C^* is the minimum cost of a tour.*

Proof. We observe that

$$C(H) \leq C(K) \leq 2C(T) \leq 2C^*.$$

For $C(H) \leq C(K)$ we use the triangle inequality repeatedly to argue that $C(j_1,j_2) + C(j_2,j_3) + \cdots + C(j_{k-1},j_k) \geq C(j_1,j_k)$. The other inequalities are obvious. □

Christofides' heuristic A simple idea reduces the 2 to 3/2.

Step 1 Find a minimum cost spanning tree T.

Step 2 Let O be the set of vertices of T of odd degree. $|O|$ is even.

Step 3 Find a minimum cost matching M that covers O.

Step 4 Let $K = M + T$. K is Eulerian.

Step 5 Construct an Euler tour $i_1, i_2, \ldots, i_m, m = n - 1 + |O|/2$ through the edges of K.

Step 6 Shortcut the Euler tour until it is a Hamilton cycle H.

Theorem 39. *The tour H found by the Christofides' heuristic satisfies $C(H) \leq 3C^*/2$.*

Proof. This follows from the fact that $C(M) \leq C^*/2$. Start with the optimal tour. Shortcut the vertices that are not in O. What is left has cost at most C^* and is the union of two disjoint matchings. Each of them have cost at least that of M. □

11.2 Knapsack problem

We consider the problem

$$\begin{aligned} \text{Maximize} \sum_{i=1}^{n} p_i x_i & \qquad (105) \\ \text{Subject to} \sum_{i=1}^{n} a_i x_i \leq B & \\ x_i = 0/1, i = 1, 2, \ldots, n. & \end{aligned}$$

We will assume that $p_1/a_1 \geq p_2/a_2 \geq \cdots \geq p_n/a_n$ and that $a_i \leq B, i = 1, 2, \ldots, n$. To avoid the problem being trivial, we assume that $a_1 + a_2 + \cdots + a_n > B$.

11.2.1 Greedy Algorithm

Find i such that $A = a_1 + a_2 + \cdots + a_{i-1} \leq B < a_1 + a_2 + \cdots + a_i$. Then choose the better of $\{1, 2, \ldots, i-1\}$ and $\{i\}$. (We associate the set $\{j : x_j = 1\}$ with a solution.)

Let $T = p_1 + p_2 + \cdots + p_{i-1}$. We have the maximum value

$$OPT \leq T + \frac{B - A}{a_i} p_i.$$

This follows from the fact that its RHS is the optimal value allowing fractional values for the x_j.

$$T \geq \frac{OPT}{2} \text{ or } p_i \geq \frac{B - A}{a_i} p_i \geq \frac{OPT}{2}.$$

11.2.2 Profit rounding algorithm

Let $\widehat{p}_i = \lfloor Np_i/p_{\max} \rfloor$ for $i = 1, 2, \ldots, n$. Here $p_{\max} = \max_{i\in[n]} p_i$ and $N = \lceil n/\varepsilon \rceil$. We solve the knapsack problem with profits $\widehat{p}_i$. Because the profits are "small" ($\leq N$), we can solve the new problem in polynomial time via Dynamic programming.

Theorem 40. *The solution produced in this way has value at least* $(1-\varepsilon)OPT$.

Proof. We first discuss the solution of the knapsack problem. Let

$$\begin{aligned} g_r(p) = &\text{minimum } a_1x_1 + \cdots + a_rx_r \\ &\text{Subject to } \widehat{p}_1x_1 + \cdots + \widehat{p}_rx_r \geq p \\ &\qquad x_i = 0/1 \text{ for } i \in [r]. \end{aligned} \tag{106}$$

Note that

$$g_r(p) = \min \begin{cases} g_{r-1}(p) & x_r = 0. \\ a_r + g_{r-1}(p - \widehat{p}_r) & x_r \geq 1. \end{cases} \tag{107}$$

We evaluate (107) for $p = 0, 1, \ldots, Nn$ and $r = 1, 2, \ldots, n$. This takes $O(n^2N) = O(n^3/\varepsilon)$ time.

If we know $g_n(p), 0 \leq p \leq Nn$, then we recover the approximate solution to (105) as follows: compute p^* such that $g_n(p^*) \leq B$ and $g_n(p^*+1) > B$. Then $\widehat{S}$, the solution to (106) with $p = p^*$, is our approximation.

We now verify the quality of the solution. Let S^* define the solution to the actual knapsack problem. Then

$$\sum_{i\in\widehat{S}} \widehat{p}_i \geq \sum_{i\in S^*} \widehat{p}_i.$$

Therefore

$$\sum_{i\in\widehat{S}} p_i \geq \sum_{i\in\widehat{S}} \left\lfloor \frac{p_iN}{p_{\max}} \right\rfloor \frac{p_{\max}}{N} = \frac{p_{\max}}{N} \sum_{i\in\widehat{S}} \widehat{p}_i \geq \frac{p_{\max}}{N} \sum_{i\in S^*} \widehat{p}_i. \tag{108}$$

But,

$$\begin{aligned} &\frac{p_{\max}}{N} \sum_{i\in S^*} \left\lfloor \frac{Np_i}{p_{\max}} \right\rfloor \geq \frac{p_{\max}}{N} \sum_{i\in S^*} \left(\frac{p_iN}{p_{\max}} - 1 \right) \geq \sum_{i\in S^*} p_i - \frac{np_{\max}}{N} \geq \\ &\sum_{i\in S^*} p_i - \varepsilon p_{\max} \geq \sum_{i\in S^*} p_i - \varepsilon OPT = (1-\varepsilon)OPT. \end{aligned}$$

□

11.3 Set cover

11.3.1 Primal-Dual algorithm

Recall the set cover problem of Section 6.1. The dual of the LP relaxation (67) is

$$\text{Maximise } \sum_{i=1}^{m} y_i. \tag{109}$$

$$\text{Subject to } \sum_{i=1}^{m} a_{i,j} y_i \leq c_j, \qquad \text{j=1,2,...,n.} \tag{110}$$

$$y_i \geq 0, i = 1, 2, \ldots, m.$$

The constraints (110) can be re-expressed as

$$\sum_{i \in S_j} y_i \leq c_j. \tag{111}$$

Consider the following algorithm:

Step 1 $\mathbf{y} \leftarrow 0$, $J \leftarrow \emptyset$.

Step 2 **while** there exists $i \notin \bigcup_{j \in J} S_j$ **do** increase y_i until there is some ℓ with $i \in S_\ell$ such that $\sum_{k \in S_\ell} y_k = c_\ell$. $J \leftarrow J \cup \{\ell\}$.

Let J^* define the optimum cover and let $\widehat{J}$ be produced by the above algorithm.

Theorem 41. *Let f_i be the number of sets S_j that contain i and let $f = \max_i f_i$. The above algorithm produces a set-cover $\widehat{J}$ and $c(\widehat{J}) \leq f c(J^*)$.*

Proof. It is clear from Step 2 that $\widehat{J}$ defines a cover. Let $\widehat{\mathbf{y}}$ denote $\mathbf{y}$ at the end of the algorithm. It is also clear that $\widehat{\mathbf{y}}$ satisfies (111) and so it satisfies the dual constraints. Let $z_{LP} \leq c(J^*)$ denote the value of the optimum solution to (67). Then,

$$\sum_{j \in \widehat{J}} c_j = \sum_{j \in \widehat{J}} \sum_{i \in S_j} \widehat{y}_i = \sum_{i=1}^{m} \left| \left\{ j \in \widehat{J} : i \in S_j \right\} \right| \widehat{y}_i \leq \sum_{i=1}^{m} f_i \widehat{y}_i \leq f \sum_{i=1}^{m} \widehat{y}_i \leq f z_{LP} \leq f c(J^*).$$

□

11.3.2 Greedy Algorithm

In this algorithm we add sets to our cover according to the average cost of newly covered elements.

Step 1 $J \leftarrow \emptyset$, $\widehat{S}_j \leftarrow S_j$, $j = 1, 2, \ldots, n$.

Step 2 $\ell = \text{argmin}_{j: \widehat{S}_j \neq \emptyset} \frac{c_j}{|\widehat{S}_j|}$.

Step 3 $J \leftarrow J \cup \{\ell\}$, $\widehat{S}_j \leftarrow \widehat{S}_j \setminus \widehat{S}_\ell$, $j = 1, 2, \ldots, n$.

Let $H_k = \sum_{t=1}^{k} t^{-1}$ denote the *Harmonic number* k. Once again, let J^* define the optimum cover and let $\widehat{J}$ be produced by the above algorithm.

Theorem 42. *The above algorithm produces a set-cover $\widehat{J}$ and $c(\widehat{J}) \leq H_m \cdot c(J^*)$.*

Proof. Let n_k denote the number of uncovered elements at the start of the kth iteration. Now at the start of iteration k,

$$\min_{j:\widehat{S}_j \neq \emptyset} \frac{c_j}{|\widehat{S}_j|} \leq \frac{\sum_{j \in J^*} c_j}{\sum_{j \in J^*} |\widehat{S}_j|} = \frac{c(J^*)}{\sum_{j \in J^*} |\widehat{S}_j|} \leq \frac{c(J^*)}{n_k}. \tag{112}$$

Thus, if ℓ minimises the fraction on the LHS of (112), then

$$c_\ell \leq \frac{|\widehat{S}_\ell| \cdot c(J^*)}{n_k} = \frac{(n_k - n_{k+1})c(J^*)}{n_k}.$$

And then if there are K iterations altogether,

$$\sum_{\ell \in \widehat{J}} c_\ell \leq \sum_{k=1}^{K} \frac{(n_k - n_{k+1})c(J^*)}{n_k} \leq c(J^*) \sum_{k=1}^{K} \left(\frac{1}{n_k} + \frac{1}{n_k - 1} + \cdots + \frac{1}{n_{k+1} + 1} \right) = H_m \cdot c(J^*).$$

□

11.4 Submodular functions

A function $f : 2^X \to \mathbb{R}$ is *submodular* if for every $A, B \subseteq X$,

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B). \tag{113}$$

Note that if f, g satisfy (113), then so does $f + g$.

Example: Simple plant location problem Here we modify the problem to maximise profit. For every non-empty $S \subseteq [m]$,

$$f(S) = -\sum_{i \in S} f_i + \sum_{j=1}^{n} \max \{p_{i,j} : i \in S\}.$$

The constant function is clearly (sub)modular and this deals with $\sum_i f_i$. Then we observe that for reals $x_1, x_2, \ldots, x_n$,

$$\max \{x_i : i \in A \cap B\} + \max \{x_i : i \in A \cup B\} \leq \max \{x_i : i \in A\} + \max \{x_i : i \in B\}.$$

(Assume that the largest x_i is for $i \in A$. Then, $\max \{x_i : i \in A \cup B\} = \max \{x_i : i \in A\}$ and $\max \{x_i : i \in A \cap B\} \leq \max \{x_i : i \in B\}$.)

f is monotone increasing if

$$f(B) \geq f(A) \text{ whenever } B \supseteq A.$$

Greedy Algorithm:
Step 0: $S_0 = \emptyset$.
Step i**:** $S_i = S_{i-1} \cup \{x_i\}$ where $x_i \notin S_{i-1}$ maximises $f(S_{i-1} \cup \{x\})$.

Theorem 43. *If f is monotone increasing and submodular and if $f(\emptyset) = 0$ then after k steps of Greedy*

$$f(S_k) \geq (1 - e^{-1}) f(S_k^*),$$

where S_k^* maximises f over sets of size k.

Proof. Let $\Delta(v \mid T) = f(T \cup \{v\}) - f(T)$. If $S \supseteq T$ and $v \notin S$ then

$$f(S \cup \{v\}) + f(T) \leq f(T \cup \{v\}) + f(S) \qquad (A = T \cup \{v\}, B = S),$$

which implies that

$$\Delta(v \mid S) \leq \Delta(v \mid T). \tag{114}$$

(The larger the set, the smaller the gain from v.)

Note that (114) is still true if $v \in S$, from monotonicity. Let $S_k^* = \{v_1^*, \ldots, v_k^*\}$.

$$\begin{aligned}
f(S_k^*) &\leq f(S_k^* \cup S_i) \\
&= f(S_i) + \sum_{j=1}^{k} \Delta(v_j^* \mid S_i \cup \{v_1^*, \ldots, v_{j-1}^*\}) \\
&\leq f(S_i) + \sum_{j=1}^{k} \Delta(v_j^* \mid S_i) \\
&\leq f(S_i) + \sum_{j=1}^{k} (f(S_{i+1}) - f(S_i)) \\
&= f(S_i) + k(f(S_{i+1}) - f(S_i)).
\end{aligned}$$

We re-write the last line as

$$f(S_k^*) - f(S_{i+1}) \leq \left(1 - \frac{1}{k}\right)(f(S_k^*) - f(S_i)).$$

This implies that

$$f(S_k^*) - f(S_k) \leq \left(1 - \frac{1}{k}\right)^k (f(S_k^*) - f(\emptyset)) \leq e^{-1} f(S_k^*).$$

□

11.5 Local Search

This is a general approach to solving hard problems in Combinatorial Optimisation. Suppose that the problem is to

$$\text{Maximise } f(\mathbf{x}) \text{ Subject to } \mathbf{x} \in X. \tag{115}$$

One proceeds as follows: for each $\mathbf{x} \in X$ we define a *neighborhood* $N_{\mathbf{x}} \subseteq X$ containing $\mathbf{x}$. It is defined so that finding $\max\{f(\mathbf{y}) : \mathbf{y} \in N_{\mathbf{x}}\}$ can be done efficiently. We can then find a good (not necessarily optimal) solution as follows: let $\mathbf{x}_0 \in X$ be chosen in some way. Then define the sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m$ where $\mathbf{x}_i$ maximises $f(\mathbf{y}) : \mathbf{y} \in N_{\mathbf{x}_{i-1}}$. The value m will be the smallest i such that $f(\mathbf{x}_i) = f(\mathbf{x}_{i-1})$.

11.5.1 MaxCut

We are given a connected graph $G = (V, E)$ and a function $w : E \to \mathbb{Z}_+$. We let $X = 2^V$ and $f(S) = w(S, \bar{S})$ i.e. $f(S)$ is the weight of the cut $S : \bar{S}$. We then let $N_S = \{T : |T \setminus S| + |S \setminus T| = 1\}$. Let $W = \sum_{e \in E} w(e)$.

Theorem 44. *Local search finds a solution $\widehat{S}$ such that $f(\widehat{S}) \geq OPT/2$. It requires at most W iterations.*

Proof.

$$f(\widehat{S}) \geq f(\widehat{S} \cup \{v\}) \text{ for } v \notin \widehat{S} \text{ implies that } w(v, \widehat{S}) \geq w(v, \bar{\widehat{S}}).$$
$$f(\widehat{S}) \geq f(\widehat{S} \setminus \{v\}) \text{ for } v \in \widehat{S} \text{ implies that } w(v, \bar{\widehat{S}}) \geq w(v, \widehat{S}).$$

So,

$$\begin{aligned} 4f(\widehat{S}) &= 2\sum_{v \in \widehat{S}} w(v, \bar{\widehat{S}}) + 2\sum_{v \notin \widehat{S}} w(v, \widehat{S}) \\ &\geq \sum_{v \in \widehat{S}} w(v, \bar{\widehat{S}}) + \sum_{v \notin \widehat{S}} w(v, \widehat{S}) + \sum_{v \in \widehat{S}} w(v, \widehat{S}) + \sum_{v \notin \widehat{S}} w(v, \bar{\widehat{S}}) \\ &= 2W \geq 2OPT. \end{aligned}$$

□

The bound W on the number of iterations might be excessive. We can reduce this to a polynomial at a small degradation in performance. Let $w_{\max} = \max\{w(e) : e \in E\}$ and $N = |E| w_{\max}/(\varepsilon W)$. Then let $w^*(e) = \lceil N w(e)/w_{\max} \rceil$ for $e \in E$ and $W^* = \sum_{e \in E} w^*(e)$.

Suppose we run the above algorithm, using w^* in place of w. Then we have

$$\sum_{e \in \widehat{S} : \bar{\widehat{S}}} \frac{N w(e)}{w_{\max}} + |E| \geq \frac{W^*}{2} \geq \frac{NW}{2 w_{\max}}.$$

So,

$$w(\widehat{S}) \geq \frac{W}{2} - \frac{w_{\max}|E|}{N} \geq W\left(\frac{1}{2} - \varepsilon\right).$$

The running time is at most $nW^* \leq 2n|E|/\varepsilon$.

11.6 Randomised rounding

This is usually applied in the case of 0-1 integer programming. The idea is to solve the LP relaxation to obtain $0 \leq p_j \leq 1$ and then to obtain an integer solution by putting $x_j = 1$ with probability p_j.

11.6.1 A set cover problem

We are given a set L and a collection of sets $S_1, S_2, \ldots, S_m \subseteq L$. In addition we have a set of n requirements. For each requirement j we must choose a set S_i, where $i \in I_j \subseteq [m]$. The number of sets in a solution that

contain a given $e \in L$ is termed the width $w(e)$. The width of a solution equals $\max\{w(e) : e \in L\}$. The objective is to find a solution of minimum width.

As an example, L is the $n \times n$ square lattice. Each requirement R_j is a set of nodes of L that must be connected together by a tree. $S_1, S_2, \ldots, S_m$ are the set of possible trees and I_j is the set of trees that connect all the vertices in R_j.

This problem is readily formulated as a 0-1 integer program. Let $x_{i,j} = 0/1$ be an indicator variable denoting whether or not S_i is chosen for requirement j. Then as an integer program we have to solve

$$\text{IP=Minimize } W. \tag{116}$$

$$\text{Subject to} \sum_{i \in I_j} x_{i,j} = 1.$$

$$\sum_{j=1}^{n} \sum_{\substack{i \in I_j \\ e \in S_i}} x_{i,j} \le W, \quad e \in L.$$

$$x_{i,j} = 0/1,\ i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n. \tag{117}$$

We solve the LP relaxation where we replace (117) by $0 \le x_{i,j} \le 1$. Suppose the solution to LP is $(p_{i,j})$. Then, for each j we choose $i \in I_j$ with probability $p_{i,j}$ i.e. one i for each j, and let $\widehat{W}$ be the width of the solution produced by this procedure. Let W_{LP} be the optimum value in LP.

Theorem 45. *Fix $0 < \varepsilon < 1$ and assume that $W_{LP} \ge 3\log(|L|/\varepsilon)$. Then,*

$$\widehat{W} \le W_{LP} + \left(3W_{LP} \log\left(\frac{|L|}{\varepsilon}\right)\right)^{1/2}$$

with probabilty at least $1 - \varepsilon$.

Proof. The width of $e \in L$ is the sum of independent 0/1 random variables viz. $w(e) = \sum_{j=1}^{n} Z_j$ where $Z_j = \sum_{\substack{i \in I_j \\ e \in S_i}} x_{i,j}$. Note that $\mathbb{E}Z_j \le W_{LP}$. Applying (182), we get

$$\mathbb{P}(\exists e : w(e) \ge (1+\theta)W_{LP}) \le |L| e^{-\theta^2 W_{LP}/3} \le \varepsilon,$$

if $\theta^2 W_{LP} = (3\log(|L|/\varepsilon))$. □

11.6.2 Multi-commodity flow

We are given a network $D = (V, E)$ and k pairs of vertices $s_i, t_i, i = 1, 2, \ldots, k$. One unit of flow has to be sent from s_i to t_i for each i, along a single path. Let $w(e)$ denote the flow sent along e. We wish to minimize $\max_e w(e)$ i.e. the maximum *congestion*.

We first set up an LP relaxation: let $x_{i,e} \in [0, 1]$ denote the amount of commodity i sent along arc e. Then the

LP is

$$\begin{aligned}
&\text{Minimize } C \\
&\text{Subject to } \sum_{e=(s_i,v)} x_{i,e} = 1, \quad i = 1,2,\ldots,k. \\
&\sum_{e=(u,v)} x_{e,i} = \sum_{e=(v,u)} x_{e,i}, \quad i = 1,2,\ldots,k,\ v \neq s_i, t_i. \\
&\sum_{i=1}^{k} x_{e,i} \leq C, \quad e \in E. \\
&x_{e,i} \geq 0, e \in E,\ i = 1,2,\ldots,k.
\end{aligned}$$

This defines a flow of value 1 for each i. Then for each i construct a set of paths $\mathcal{P}_i = \{P_{i,1}, P_{i,2}, \ldots, P_{i,m_i}\}$ from s_i to t_i and flow of commodity $f_{i,j}$ along path P_i such that for each edge e, $\sum_{j=1}^{m_i} f_{i,j} = x_{e,i}$. (Not hard to do – exercise.)

Now randomly choose a path Q_i from $\mathcal{P}_i$ where each path is chosen with probability $f_{i,j}$. The expected congestion on edge e is

$$\sum_{i=1}^{k} \sum_{j=1}^{m_i} f_{i,j} 1_{e \in P_{i,j}} = \sum_{i=1}^{k} x_{e,i} \leq C.$$

Furthermore, the total load on each edge is the sum of k independent 0/1 random variables $1_{e \in Q_i}$. It follows from Hoeffdings inequality that

$$\mathbb{P}(\exists e : w(e) \geq (1+\varepsilon)C) \leq k|E|e^{-\varepsilon^2 C/3}.$$

This will be at most ε if $C \geq 3\varepsilon^{-2}\log(k|E|/\varepsilon)$.

12 Inventory Control

We discuss some simple models that attempt to control some of the costs of keeping inventory/stock. Our analysis involves trying to answer the question (i) when to order goods and (ii) how much to order.

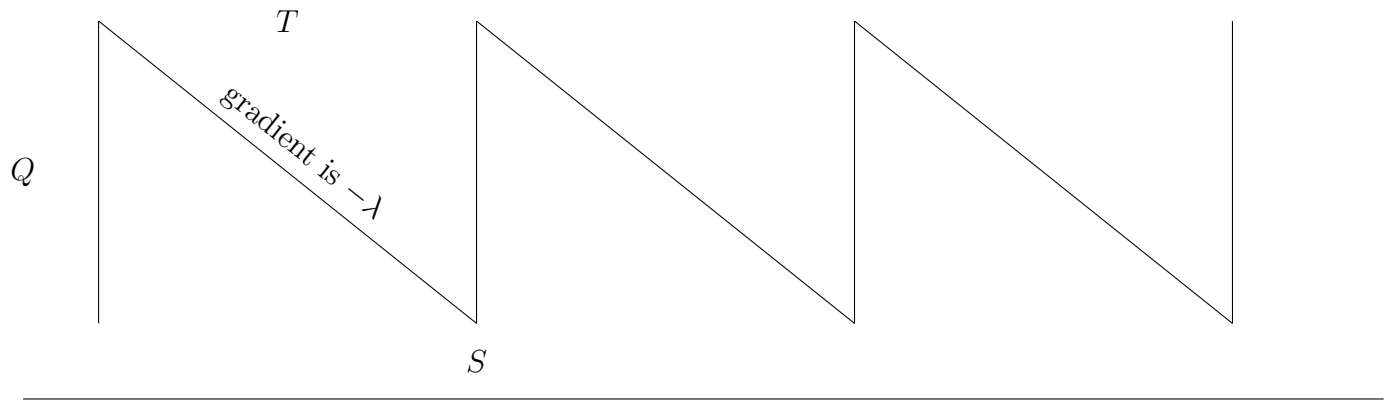
12.1 Model 1

The demand for the product is constant and λ per period there is a zero lead time and no stock outs are allowed. At constant intervals of time T we make an order for Q items of stock. The problem is to choose T and Q so as to minimize the average cost per period of running the system.

Notation

- A is the fixed cost associated with making an order.
- C is the unit cost per item.
- I is the inventory carrying charge. If one unit of stock is kept for t periods then the inventory charge is It.

Let S be the stock level when each order arrives. The stock level will have a pattern as shown in the diagram below:



From the diagram we see that $Q = \lambda T$ and so only one of these variables can be chosen independently.
Ordering cost
The actual cost of the items bought or produced is λC per period on average regardless of the inventory policy and will not be affected by the choice of Q. It is therefore ignored.

The average number of orders per period will be $1/T = \lambda/Q$. Therefore the average fixed cost of making orders is $A\lambda/Q$.

Holding cost
The average stock level is $(S + (S + Q))/2 = S + Q/2$ and therefore the average holding cost per period is $I(S + Q/2)$.

Therefore the average total variable cost per period K is given by

$$K = \frac{A\lambda}{Q} + I\left(\frac{Q}{2} + S\right). \tag{118}$$

K is to be minimized for Q and $S \geq 0$. Now for given Q we minimize K by taking $S = 0$. To find the optimal Q we differentiate the right hand side of (118). Now

$$\frac{dK}{dQ} = -\frac{A\lambda}{Q^2} + \frac{I}{2}.$$

Putting $\frac{dK}{dQ} = 0$ gives the optimal Q_W where

$$Q_W = \left(\frac{2\lambda A}{I}\right)^{1/2}. \tag{119}$$

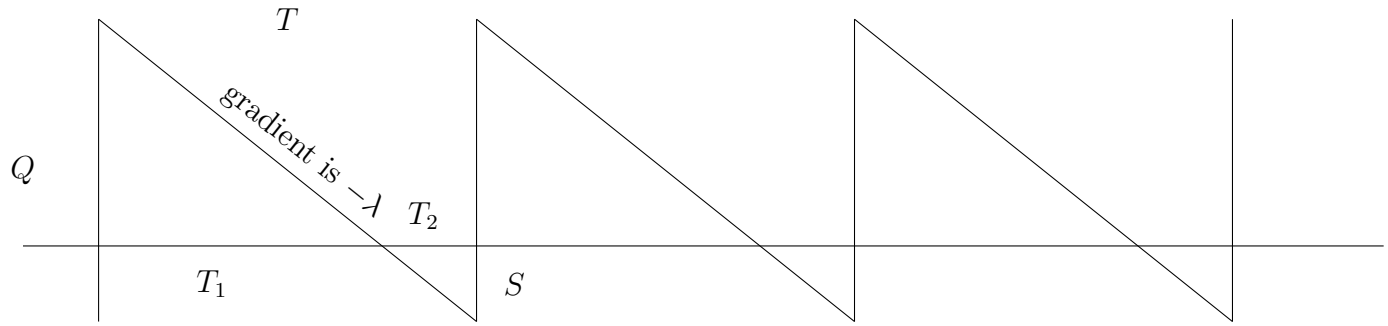
The RHS of (119) is sometimes referred to as the Wilson lot size formula. Using this we see that the optimal time interval T_W and the minimum cost K_W are given by

$$T_W = \left(\frac{2A}{\lambda I}\right)^{1/2} \text{ and } K_W = (2\lambda A I)^{1/2}.$$

Discrete case
Suppose that we can only order integer quantities Q and that Q_W is not an exact integer. In this case we simply compare the cost for Q_W rounded up and Q_W rounded down and take the smaller

12.2 Model 2

This model is the same as for Model 1 except that items out of stock can be backordered and supplied when goods come into stock. The cost of each item back ordered is πt where t is the length of time the demand remains unfilled. The Inventory level will follow the pattern below:



S is the number of back orders when the order Q arrives. The objective is to choose Q and S so that the average cost per period is minimized. The following relations are apparent from the diagram:

$$Q = \lambda T, \quad S = \lambda T_2, \quad Q - S = \lambda T_1.$$

Ordering cost:
The number of orders per period is again $1/T = \lambda/Q$ and so the variable ordering cost is $\lambda A/Q$.

Holding cost:
The proportion of time that there are goods in stock is T_1/T. During this time the average inventory level is $(Q-S)/2$ and so the average holding cost per period is $I(Q-S)T_1/2T$. But from the above we have that $T_1/T = (Q-S)/Q$ and so this cost is in fact $I(Q-S)^2/2Q$.

Backorder cost:
The proportion of time that the system is out of stock is T_2/T. Because of the given form of backorder cost we may work out the average back order cost in a similar way to that of the holding cost. During the time the system is out of stock the average amount backordered is $S/2$. Therefore the average back order cost is $\pi S T_2/2T = \pi S^2/2Q$. Thus the total variable annual cost K is then

$$K = \frac{\lambda A}{Q} + \frac{I(Q-S)^2}{2Q} + \frac{\pi S^2}{2Q}.$$

In order to minimize K we solve the equations $\frac{\partial K}{\partial Q} = \frac{\partial K}{\partial S} = 0$, yielding

$$-\frac{\lambda A}{Q^2} - \frac{I(Q-S)^2}{2Q^2} + \frac{I(Q-S)}{Q} - \frac{\pi S^2}{2Q^2} = 0. \tag{120}$$

$$-\frac{I(Q-S)}{Q} + \frac{\pi S}{Q} = 0. \tag{121}$$

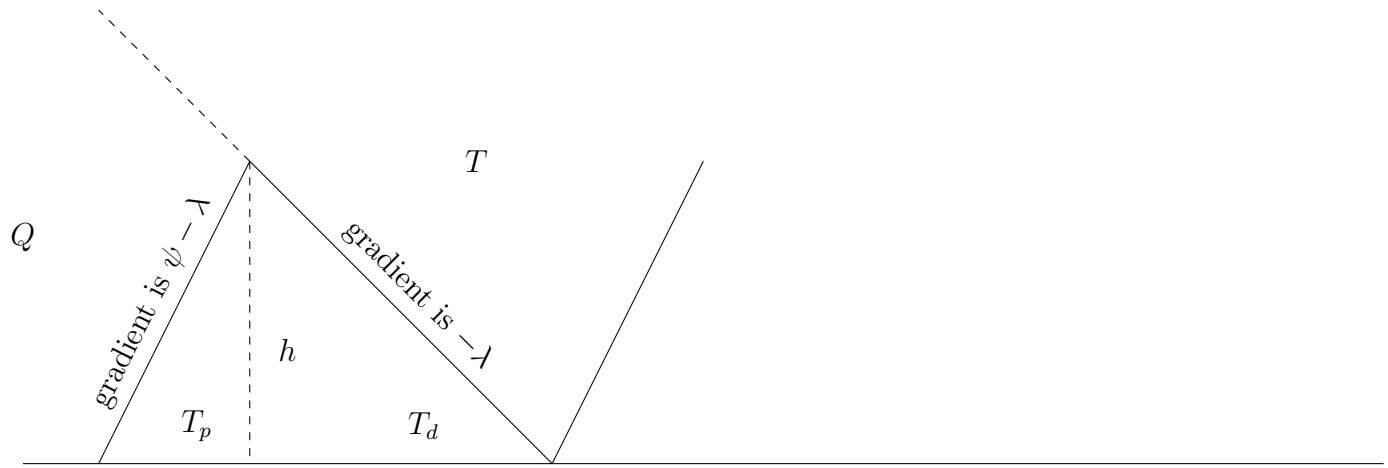
These can be solved by using (121) to express Q in terms of S, substitute in (120) to give an equation in S and then solving, giving

$$S = \left(\frac{2\lambda A I}{\pi(\pi+I)}\right)^{1/2}, \quad Q = Q_W\left(\frac{\pi+I}{\pi}\right)^{1/2}, \quad K = K_W\left(\frac{\pi}{\pi+I}\right)^{1/2}.$$

Comparing with model 1 we see that the extra freedom of allowing back orders enables us to reduce the total cost. We can obtain the results for Model 1 by putting $\pi = \infty$.

12.3 Model 3:

In the previous models we assumed that the orders arrived instantaneously in a single lot size of Q units we now consider a situation in which having made an order the items arrive in a continuous stream at a rate $\psi > \lambda$ per period. The inventory level assuming no back ordering will then have the pattern below



During period T_p the stock level increases at a rate $\psi - \lambda$. During period T_d the stock level decreases at a rate λ. If the length of the time between orders is T and the total amount Q is produced at a time then we must have $Q = \lambda T$, otherwise stock levels will not be zero again at the end of each cycle.

Ordering cost:
The average number of orders per period is again $1/T = \lambda/Q$ and so the average ordering cost $A\lambda/Q$.

Holding cost:
If h is the maximum stock level during a cycle then the average stock $h/2$. Now h is the amount of stock built up during T_p and so $h = (\psi - \lambda)T_p$. We also have that $Q = \psi T_p$ as the order is produced in time T_p. Substituting in the above gives $h = (1 - \lambda/\psi)Q$. Therefore the average inventory cost per period is $I(1 - \lambda/\psi)Q/2$ total variable cost is

$$\frac{\lambda A}{Q} + \frac{IQ(\psi - \lambda)}{2\psi}.$$

Solving the equation $\frac{dK}{dQ} = 0$ gives the optimal batch quantity as

$$Q = Q_W \left(\frac{\psi}{\psi - \lambda}\right)^{1/2} \text{ and } K = K_W \left(\frac{\psi - \lambda}{\psi}\right)^{1/2}.$$

The total cost has decreased again as against Model 1 due to decreasing inventory costs. Note that taking $\psi = \infty$ gives Model 1 again.

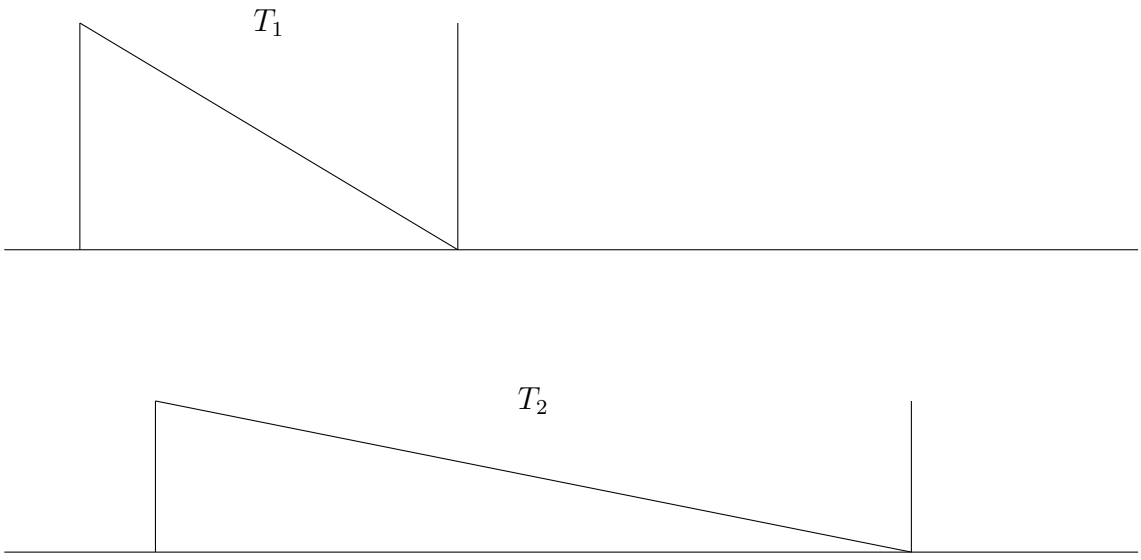
12.4 Deterministic multi-item problems

We consider here some problems where more than one type of item is being stored and when the inventory cost of the items interact. If there is no interaction we can consider each item separately.

Models 4 and 5 of this section are generalizations of Model 1 although we could equally well have generalized Model 2 or Model 3.

12.5 Model 4

Assume there are n distinct types of item whose individual characteristics are that of Model 1. When an order is made it is possible to order more than one type of item but the fixed cost of ordering A is unaffected by the number of different types of goods ordered. Suppose then that there is a demand λ_j and holding cost I_j for item j. Let the optimal policy be to order items j at intervals T_j.



We show that in an optimum schedule, $T_1 = T_2 = \cdots = T_n$. For let $T = T_k = \min_j T_j$. Then if $T_j > T$ for some j then by increasing the frequency of ordering of item j to that of item k and ensuring that item j is ordered at the same time as k we
(i) eliminate any extra ordering cost for j over those of k
(ii) decrease the average stock level of j.

This implies non-optimality for the current schedule and the claim follows. The problem is then to find the optimal value for T.

Ordering cost
The average number of orders made per period is $1/T$ and so the average ordering cost per period is A/T.

Holding cost
Let Q_j be the amount of item j ordered at a time. Then since orders are made at intervals of time T we must have $Q_j = \lambda_j T$. Now the average inventory for item j is $Q_j/2$ and so the average inventory cost per period for item j is $I_j Q_j/2$. We see therefore that the average total inven tory cost is $\sum_j I_j Q_j/2$. So the average total cost period is given by

$$K = \frac{A}{T} + T \sum_{j=1}^{n} \frac{I_j \lambda_j}{2}. \tag{122}$$

The optimal value of T is obtained by solving $\frac{dK}{dT} = 0$ which gives

$$T = \left(\frac{2A}{\sum_{j=1}^{n} \lambda_j I_j} \right)^{1/2}.$$

The optimal lot sizes $Q_j = \lambda_j T$ and the minimal cost K as in (122) can now be calculated.

12.6 Model 5

In the previous model the items were able to share facility (ordering) without an increase in cost. Thus the total cost was smaller than if all the items were ordered separately. In some situations items will make

conflicting demands on resources, for example if there is a shortage of storage space then calculating the optimal lot size for each item separately may lead to too great a demand for storage space. We again have to assume there are n items whose individual characteristics are those of Model 1 but that the lot sizes Q_j have to satisfy the constraint

$$\sum_{j=1}^{n} f_j Q_j \leq f. \tag{123}$$

For given batch sizes $Q_1, \ldots, Q_n$ we may evaluate the average total cost by summing individual costs as calculated in Model 1. This gives

$$K = \sum_{j=1}^{n} \left(\frac{\lambda_j A_j}{Q_j} + \frac{I_j Q_j}{2} \right). \tag{124}$$

The problem is therefore to minimize K subject to (123) and we can proceed as follows:

(i) We first find the minimum of K ignoring constraint (123). Now to minimize K overall we can simply minimize each term in the summation separately. This will naturally lead to the Wilson lot sizes $Q_{jW} = (2\lambda_j A_j / I_j)^{1/2}$, $j = 1, 2, \ldots, n$. If these values satisfy the constraint then we have solved the problem.

(ii) Otherwise we may assume that the constraint is active at the optimum $\mathbf{Q}^* = (Q_1^*, \ldots, Q_n^*)$. This follows because we may express

$$K(\mathbf{Q}) = K(\mathbf{Q}^*) + \sum_{j=1}^{n} \frac{\partial K}{\partial Q_j}(Q_j - Q_j^*) + \cdots$$

and if $\sum_j f_j Q_j^* < f$ then $\left(Q_1^* - \theta \frac{\partial K}{\partial Q_1}, \ldots, Q_n^* - \theta \frac{\partial K}{\partial Q_n} \right)$ is both feasible and better than $\mathbf{Q}^*$ for small enough $\theta > 0$, unless $\frac{\partial K}{\partial Q_j} = 0$ for $j = 1, 2, \ldots, n$, which we have already ruled out.

So now we tackle the problem

$$\text{Minimise } K \text{ subject to } F = \sum_{j=1}^{n} f_j Q_j - f = 0. \tag{125}$$

(We are *lucky* in that we can ignore the constraints $Q_j \geq 0, j = 1, \ldots, n$.)

Problem (125) is in a form suitable for application of the classical Lagrange Multiplier method. This states that there exists θ such that at the optimum to (125)

$$\frac{\partial K}{\partial Q_j} = \theta \frac{\partial F}{\partial Q_j} \text{ for } j = 1, \ldots, n. \tag{126}$$

On differentiation we see that this states

$$-\frac{\lambda_j A_j}{Q_j^2} + \frac{I_j}{2} = \theta f_j \text{ for } j = 1, \ldots, n.$$

or

$$Q_j = \left(\frac{2\lambda_j A_j}{I_j - 2\theta f_j} \right)^{1/2}. \tag{127}$$

The value for θ can be calculated by solving

$$\sum_{j=1}^{n} f_j \left(\frac{2\lambda_j A_j}{I_j - 2\theta f_j} \right)^{1/2} = f.$$

Given θ we use (127) to find the Q_j's.

12.7 Newsvendor problem

A newsvendor pays c per copy of a paper and sells it for price $p > c$. The demand D is a random variable with CDF $F(d) = \mathbb{P}(D \leq d)$. Assume that F is differentiable with derivative f. How many copies of the paper should the vendor purchase in order to maximise expected profit?

If the vendor buys q copies then the expected profit Φ is given by

$$\begin{aligned}\Phi(q) = \mathbb{E}(p\min\{q, D\}) - cq &= pq\mathbb{P}(D \geq q) + p\mathbb{E}(D \mid D < q)\mathbb{P}(D < q) - cq \\ &= pq(1 - F(q)) + p\int_{x=0}^{q} xf(x)dx - cq.\end{aligned}$$

So,

$$\Phi'(q) = p(1 - F(q)) - pqf(q) + pqf(q) - c = p(1 - F(q)) - c.$$

Now $\Phi''(q) = -pf(q) \leq 0$ and so we just need to put $\Phi'(q) = 0$ to find the maximum. This gives

$$q^* = F^{-1}\left(\frac{p - c}{p}\right)$$

as the optimal order quantity.

13 Job Shop Scheduling

Jobs $J_1, J_2, \ldots, J_n$ have to be processed on machines $M_1, M_2, \ldots, M_m$ in order to complete the production of some item. We study a few associated optimisation problems. Here we will only have $m = 1$ or 2.

13.1 Single machine, minimise weighted completion time

$$\begin{aligned} C_j &= \text{ completion time of job } j. \\ p_j &= \text{ processing time of job } j. \end{aligned}$$

Objective: find the order (π) in which to do the jobs that minimises $S = S(\pi) = \sum_{j=1}^{n} w_j C_j$.

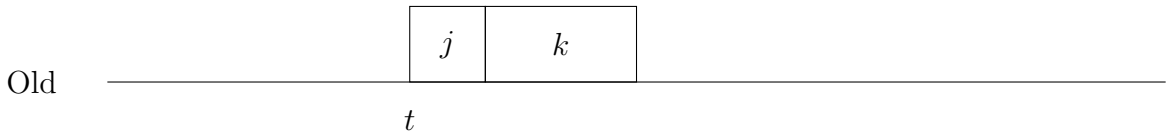
Example: $n = 4$, $p_1 = 3, p_2 = 2, p_3 = 5, p_4 = 6$ and $w_1 = 1, w_2 = 10, w_3 = 3, w_4 = 2$.

$S(1, 2, 3, 4) = 1 \times 3 + 10 \times 5 + 3 \times 10 + 2 \times 16 = 115$.
$S(2, 3, 1, 4) = 10 \times 2 + 3 \times 7 + 1 \times 10 + 2 \times 16 = 83$.

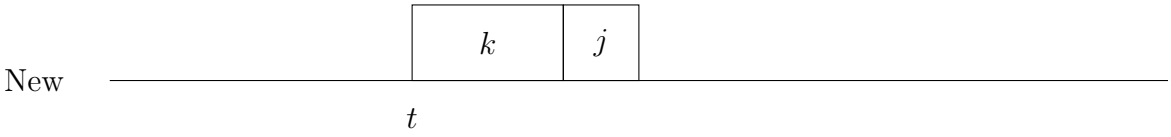
In general the optimum order satisfies

$$\frac{w_{\pi(1)}}{p_{\pi(1)}} \geq \frac{w_{\pi(2)}}{p_{\pi(2)}} \geq \cdots \geq \frac{w_{\pi(n)}}{p_{\pi(n)}}.$$

Suppose the processing order does not satisfy this: j is processed immediately before k.



Here $w_j/p_j < w_k/p_k$. Now interchange j, k in the order. Only the completion times of j, k are affected.



$$S_{new} - S_{old} = w_k(t + p_k) + w_j(t + p_k + p_j) - w_j(t + p_j) - w_k(t + p_k + p_j) = w_j p_k - w_k p_j < 0.$$

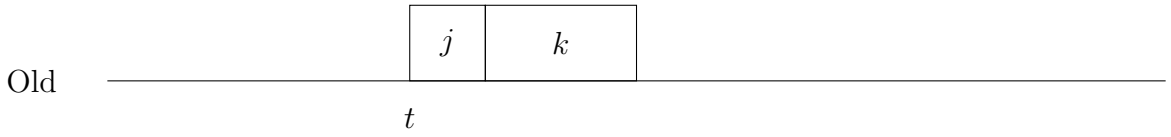
13.2 Single machine, minimise maximum lateness

$$\begin{aligned} d_j &= \text{ due date for job } j. \\ L_j &= \max\{C_j - d_j, 0\} = (C_j - d_j)^+ = \text{ lateness of job } j. \\ L_{\max} &= \max_j L_j. \end{aligned}$$

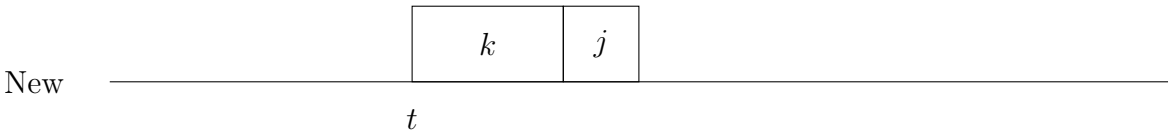
Objective: find the order (π) in which to do the jobs that minimises $L_{\max}$.

In general the optimum order satisfies $d_1 \leq d_2 \leq \cdots \leq d_n$.

Suppose the processing order does not satisfy this: suppose that $d_j > d_k$ and j is processed immediately before k.



Now interchange j, k in the order. Only the completion times of j, k are affected.



Contribution of j, k to $L_{\max}$:

$$\text{Old: } \max\left\{(t+p_j-d_j)^+, (t+p_j+p_k-d_k)^+\right\} = (t+p_j+p_k-d_k)^+.$$
$$\text{New: } \max\left\{(t+p_k-d_k)^+, (t+p_j+p_k-d_j)^+\right\}.$$

But clearly

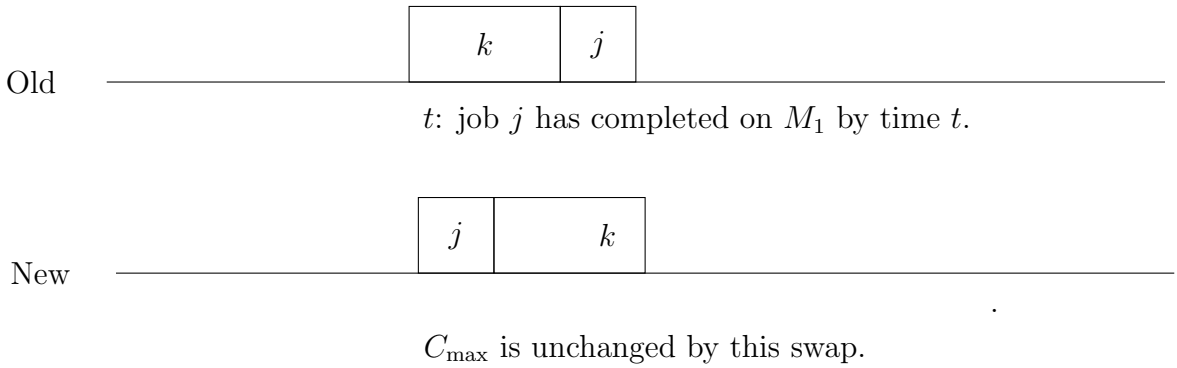
$$\max\left\{(t+p_k-d_k)^+, (t+p_j+p_k-d_j)^+\right\} \leq (t+p_j+p_k-d_k)^+.$$

13.3 Two machine flow shop: Johnson's rule

Each job has to be processed first on machine M_1 and then on machine M_2. The goal is to minimise the completion time of all the jobs. In principle the order of jobs on the two machines π_1, π_2 can be different.

13.3.1 $\pi_1 = \pi_2$: Permutation flow shop:

We first show that it is optimal to fix $\pi_1 = \pi_2$. By re-labelling if needed, we can assume that $\pi_1(i) = i$ for $i \in [n]$. Now suppose there exists $j < k$ such that $\pi_2(j) > \pi_2(k)$ and k immediately precedes j on machine 2. The diagram below refers to M_2.

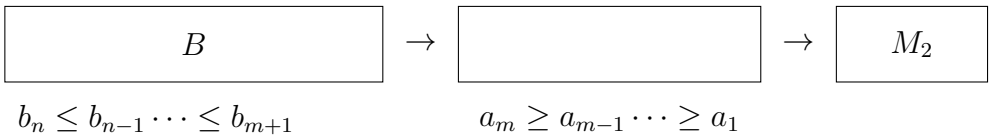


So, we will choose a single π for both machines.

13.3.2 Johnson's Rule

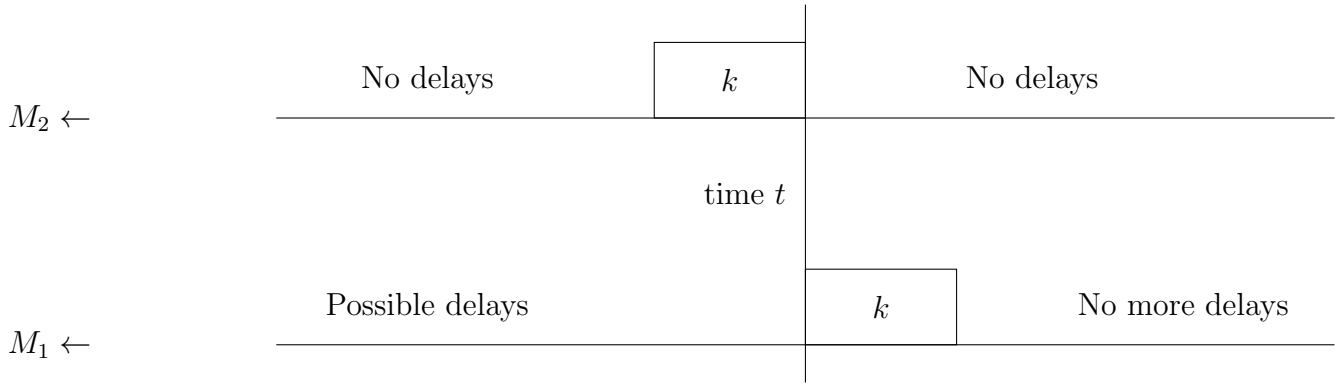
Processing times: $a_1, \ldots, a_n$ on M_1.
$b_1, \ldots, b_n$ on M_2.

Then we let $A = \{j : a_j \leq b_j\}$ and $B = [n] \setminus A$. Suppose that $|A| = m$. Renumber so that



JOHNSON'S RULE

For all permutation schedules there is a time t, after which there are no delays on machine M_2.

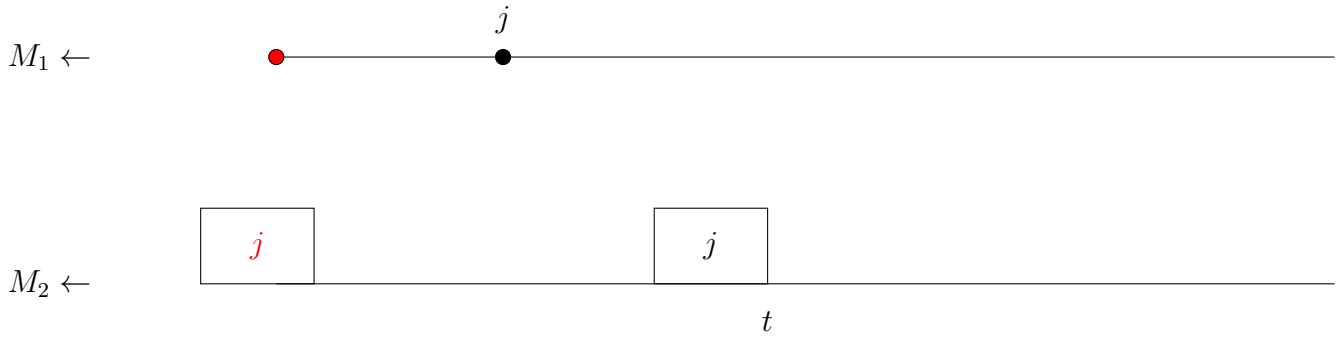


It follows that

(i) $C_{\max}$ is the sum of $n + 1$ job processing times.

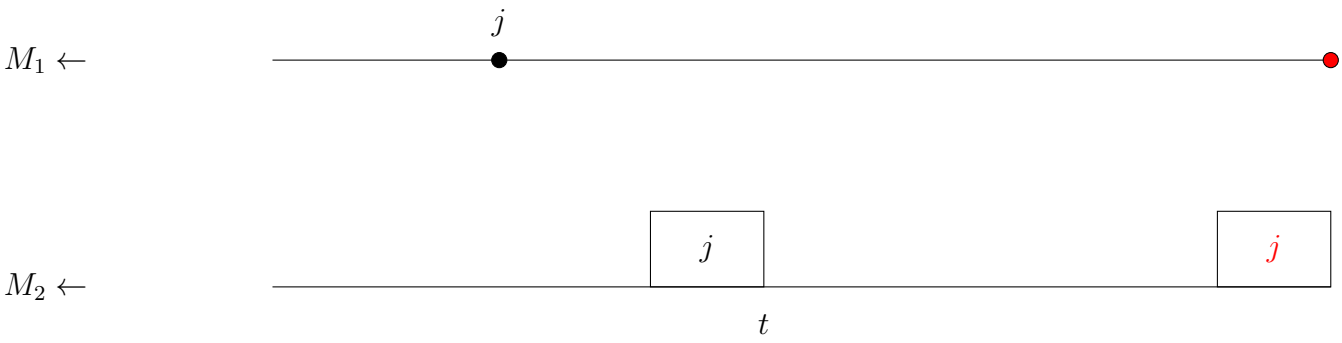
(ii) Reducing the processing time of each item by an amount ρ does not affect the optimal ordering. This is because each ordering schedule is reduced by $(n + 1)\rho$.

Now let $\rho = \rho([n]) = \min\{a_1, \ldots, a_n, b_1, \ldots, b_n\}$. There are two cases:
Case (a): $\rho = a_j$ for some j. Now let $a'_i = a_i - \rho, b'_i = b_i - \rho$ for $i = 1, \ldots, n$. Note that $a'_j = 0$. We claim that j can go first onto the machines.



Moving job j to the front does not increase $C_{\max}$. To see this, first move j to be the first job on machine M_1. This has no effect on $C_{\max}$. Now the orders on the two machines are different and we use the argument of Section 13.3.1 that moving job j closer to the beginning of the order will not increase $C_{\max}$.

Case (b): $\rho = b_j$ for some j. Now let $a'_i = a_i - \rho, b'_i = b_i - \rho$ for $i = 1, \ldots, n$. Note that $b'_j = 0$. We claim that j can go last onto the machines.



Moving job j to the end does not increase $C_{\max}$. To see this, first move j to be the last job on machine M_2. This has no effect on $C_{\max}$. Now the orders on the two machines are different and we can modify the argument of Section 13.3.1 that moving job j closer to the end of the order will not increase $C_{\max}$.

Let now S denote the jobs that have not been ordered. The general rule is

1. Suppose that the current state of the ordering is $i_1, \ldots, i_t, S, j_1, \ldots, j_s$ where $s + t = n - |S|$.
2. $\rho = r(S) = \min\{\min_{i \in S} a_i, \min_{i \in S} b_i\}$.
3. If $\rho = a_j$ then job j is placed immediately after job i_t.
4. If $\rho = b_j$ then job j is placed immediately before job j_1.

14 Decision Analysis

Company GFB owns a plot of land. It is considering drilling for oil.

Alternatives	Oil	Dry
Drill	700K	-100K
Sell Land	90K	90K
Probability	0.25	0.75.

Decision strategies:

(i) Choose alternative that maximises minimum payoff: Sell Land.

(ii) Choose alternative that maximises payoff *under the most likely alternative*: Sell land.

(iii) Choose alternative that maximises the expected payoff:
Drill: $0.25 \times 700 - 0.75 \times 100 = 100^*$.
Sell: 90.

Now introduce a third alternative: do a seismic study (SS) and then make a decision. The cost of the survey is 30K.

Outcome: FSS=Favorable, USS=Unfavorable.
Data:

$$\begin{aligned} \mathbb{P}(FSS \mid Oil) &= 0.6 \qquad \mathbb{P}(USS \mid Oil) = 0.4. \\ \mathbb{P}(FSS \mid Dry) &= 0.2 \qquad \mathbb{P}(USS \mid dry) = 0.8. \end{aligned}$$

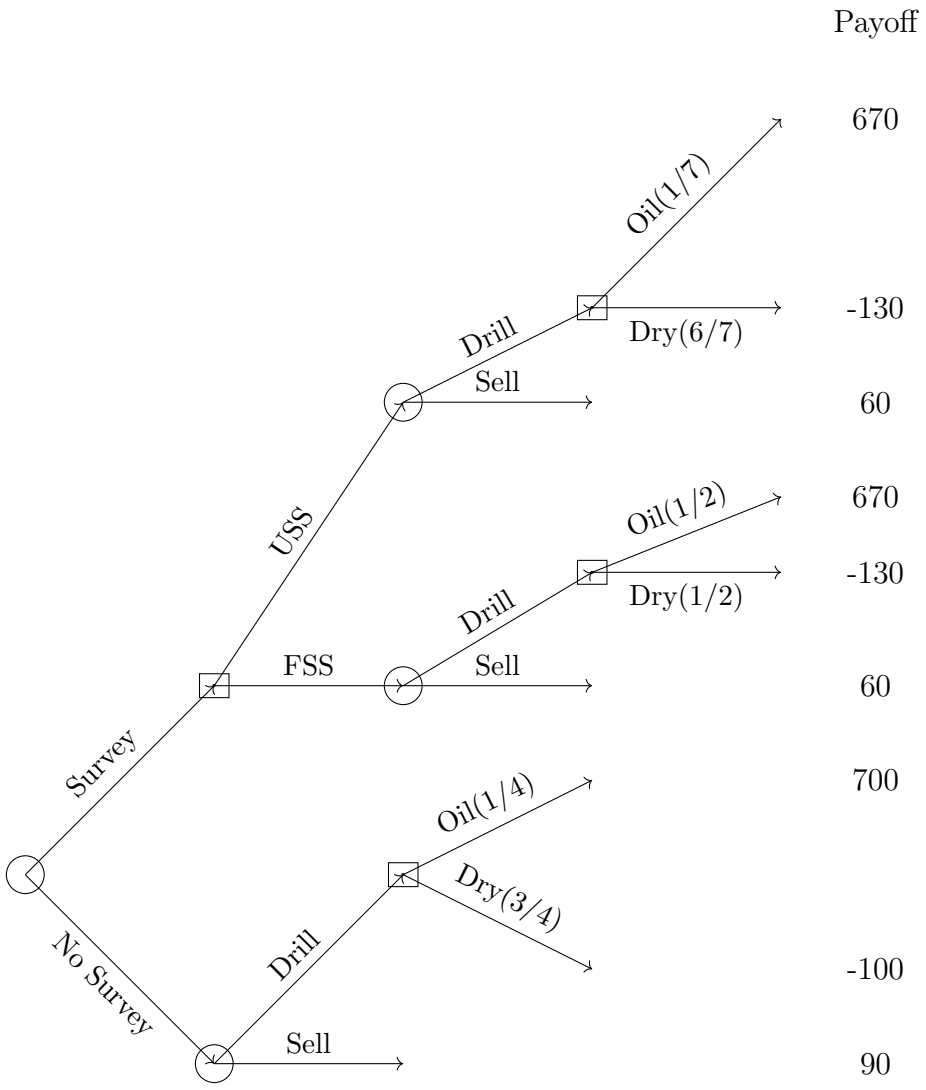
Bayes computation. We need

$$\begin{aligned} \mathbb{P}(Oil \mid FSS) &= \frac{\mathbb{P}(Oil \wedge FSS)}{\mathbb{P}(FSS)} \\ &= \frac{\mathbb{P}(FSS \mid Oil)\mathbb{P}(Oil)}{\mathbb{P}(FSS \mid Oil)\mathbb{P}(Oil) + \mathbb{P}(FSS \mid Dry)\mathbb{P}(Dry)} \\ &= \frac{0.6 \times 0.25}{0.6 \times 0.25 + 0.2 \times 0.75} \\ &= \frac{1}{2}. \end{aligned}$$

Similar calculations give

$$\mathbb{P}(Oil \mid FSS) = \frac{1}{2},\ \mathbb{P}(Dry \mid FSS) = \frac{1}{2},\ \mathbb{P}(Oil \mid USS) = \frac{1}{7},\ \mathbb{P}(Dry \mid USS) = \frac{6}{7}.$$

Decision tree.

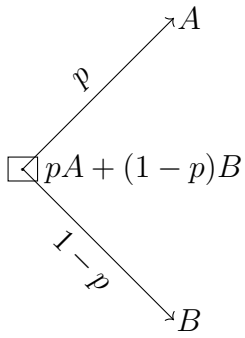


Decision node ◯

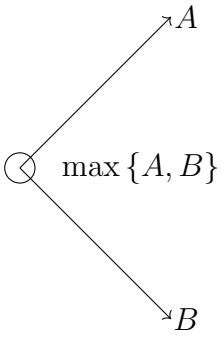
Chance node ⊡

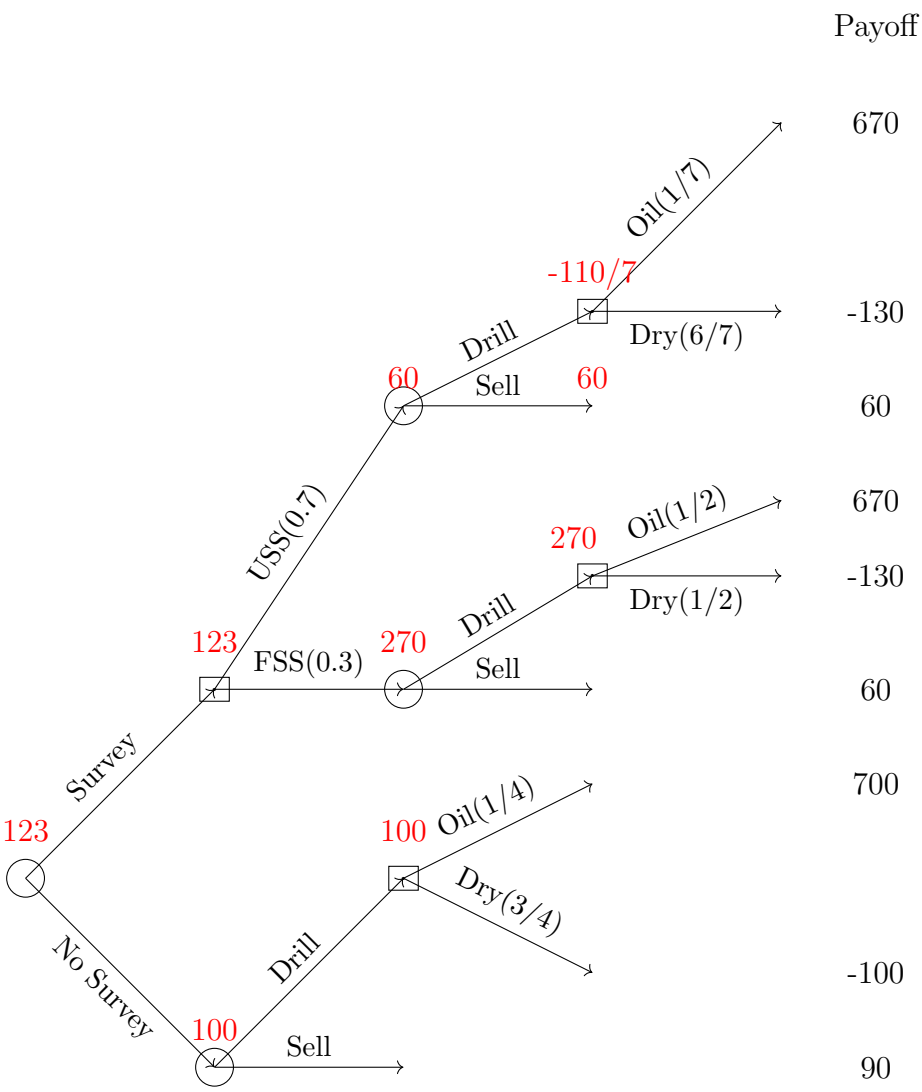
Evaluating nodes of tree. A, B are the expected values at these nodes.

Chance node:



Decision node.




Payoff
670
-130
60
670
-130
60
700
-100
90
Oil(1/7)
-110/7
Drill
Dry(6/7)
60
Sell
60
USS(0.7)
Oil(1/2)
270
Drill
Dry(1/2)
123
FSS(0.3)
270
Sell
Survey
123
Oil(1/4)
100
Dry(3/4)
No Survey
Drill
100
Sell

15 Non-linear Optimization Problems

We consider the following problem:

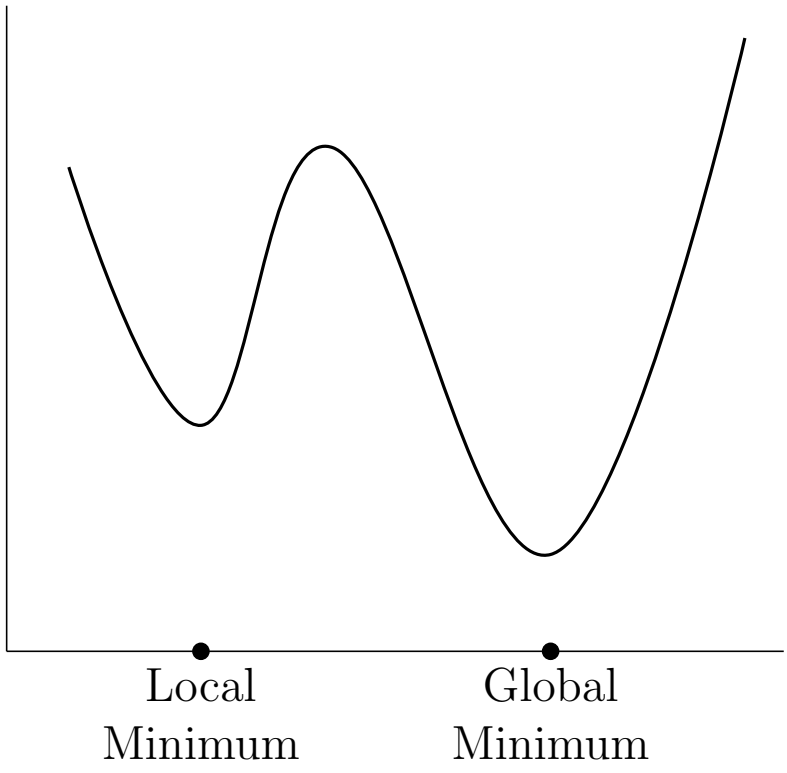
$$\text{Minimize } f(\mathbf{x}) \text{ subject to } \mathbf{x} \in S, \tag{128}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $S \subseteq \mathbb{R}^n$.

Example: $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ and $S = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$ – Linear Programming.

Local versus Global Optima: $\mathbf{x}^*$ is a *global minimum* if it is an actual minimizer in (128).

$\mathbf{x}^*$ is a *local minimum* if there exists $\delta > 0$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in B(\mathbf{x}^*, \delta) \cap S$, where $B(\mathbf{x}^*, \delta) = \{\mathbf{y} : |\mathbf{y} - \mathbf{x}^*| \leq \delta\}$ is the *ball* of radius δ, centred at $\mathbf{x}^*$.



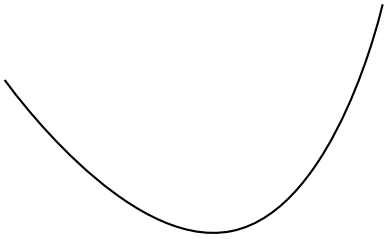
If $S = \mathbb{R}^n$ then we say that the problem is *unconstrained*, otherwise it is *constrained*.

16 Convex functions

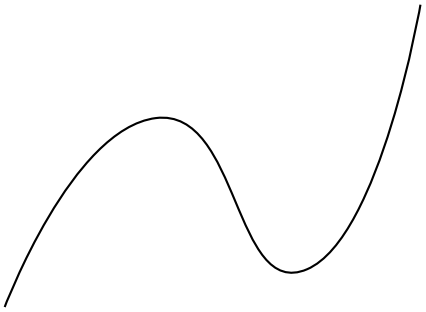
Be aware that in many cases below we have used min or max where we should use inf or sup.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *convex* if for all $0 \leq \lambda \leq 1$ and all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}).$$



Convex Function



Non-Convex Function

16.0.1 Examples of convex functions:

F1 A linear function $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x}$ is convex.

F2 If $n = 1$ then differentiable f is convex iff

$$f(y) \geq f(x) + f'(x)(y - x) \text{ for all } x, y. \tag{129}$$

Proof. Suppose first that f is convex and differentiable. Then for $0 \leq \lambda \leq 1$,

$$f(x + \lambda(y - x)) \leq (1 - \lambda)f(x) + \lambda f(y).$$

Thus, putting $h = \lambda(y - x)$ we have

$$f(y) \geq f(x) + \frac{f(x + h) - f(x)}{h}(y - x).$$

Taking the limit as $\lambda \to 0$ implies (129).

Now suppose that (129) holds. Choose $x \neq y$ and $0 \leq \lambda \leq 1$ and let $z = \lambda x + (1 - \lambda)y$. Then we have

$$f(x) \geq f(z) + f'(z)(x - z) \text{ and } f(y) \geq f(z) + f'(z)(y - z).$$

Multiplying the first inequality by λ and the second by $1 - \lambda$ and adding proves that

$$\lambda f(x) + (1 - \lambda)f(y) \geq f(z).$$

□

F3 If $n \geq 1$ then differentiable f is convex iff $f(\mathbf{y}) \geq f(\mathbf{x}) + (\nabla f(\mathbf{x}))^T(\mathbf{y} - \mathbf{x})$ for all $\mathbf{x}, \mathbf{y}$.
Apply F2 to the function $h(t) = f(t\mathbf{x} + (1-t)\mathbf{y})$.

F4 If $n = 1$ and f is twice differentiable then f is convex iff $f''(z) \geq 0$ for all $z \in \mathbb{R}$.

Proof. Taylor's theorem implies that

$$f(y) = f(x) + f'(x)(y-x) + \frac{1}{2}f''(z)(y-x)^2 \text{ where } x \leq z \leq y.$$

We now just apply (129). □

F5 It follows from F4 that e^{ax} is convex for any $a \in \mathbb{R}$.

F6 x^a is convex on $(0, \infty)$ for $a \geq 1$ or $a \leq 0$. x^a is concave for $0 \leq a \leq 1$.
Here f is *concave* iff $-f$ is convex.

F7 Suppose that A is a symmetric $n \times n$ positive semi-definite matrix. Then $Q(\mathbf{x}) = \mathbf{x}^T A\mathbf{x}$ is convex.
By positive semi-definite we mean that $Q(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$.
We have

$$\begin{aligned} & Q(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) - \lambda Q(\mathbf{x}) - (1-\lambda)Q(\mathbf{y}) \\ =& \lambda^2 Q(\mathbf{x}) + (1-\lambda)^2 Q(\mathbf{y}) + 2\lambda(1-\lambda)\mathbf{x}^T A\mathbf{y} - \lambda Q(\mathbf{x}) - (1-\lambda)Q(\mathbf{y}) \\ =& -\lambda(1-\lambda)Q(\mathbf{y} - \mathbf{x}) \leq 0. \end{aligned}$$

F8 If $n \geq 1$ then f is convex iff the *Hessian* $\nabla^2 f = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]$ is positive semi-definite for all $\mathbf{x}$, assuming of course that f is twice differentiable.
Apply F4 to the function $h(t) = f(\mathbf{x} + t\mathbf{d})$ for all $\mathbf{x}, \mathbf{d} \in \mathbb{R}^n$.

16.0.2 Operations on convex functions

E1 If f, g are convex, then $f + g$ is convex.

E2 If $\lambda > 0$ and f is convex, then λf is convex.

E3 If f, g are convex then $h = \max\{f, g\}$ is convex.

Proof.

$$\begin{aligned} h(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) &= \max\{f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}), g(\lambda\mathbf{x} + (1-\lambda)\mathbf{y})\} \\ &\leq \max\{\lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}), \lambda g(\mathbf{x}) + (1-\lambda)g(\mathbf{y})\} \\ &\leq \lambda \max\{f(\mathbf{x}), g(\mathbf{x})\} + (1-\lambda)\max\{f(\mathbf{y}), g(\mathbf{y})\} \\ &= \lambda h(\mathbf{x}) + (1-\lambda)h(\mathbf{y}). \end{aligned}$$

□

E4 If f is convex then $g(\mathbf{x}) = f(\mathbf{A}\mathbf{x} + \mathbf{b})$ is convex for a matrix $\mathbf{A}$ and vector $\mathbf{b}$.

Proof.

$$g(\lambda\mathbf{x}+(1-\lambda)\mathbf{y}) = f(\lambda(\mathbf{A}\mathbf{x}+\mathbf{b})+(1-\lambda)(\mathbf{A}\mathbf{y}+\mathbf{b})) \leq \lambda f(\mathbf{A}\mathbf{x}+\mathbf{b})+(1-\lambda)f(\mathbf{A}\mathbf{y}+\mathbf{b}) = \lambda g(\mathbf{x})+(1-\lambda)g(\mathbf{y}).$$

□

Jensen's Inequality
If f is convex and $\mathbf{a}_i \in \mathbb{R}^n, \lambda_i \in \mathbb{R}_+, 1 \leq i \leq m$ and $\lambda_1 + \lambda_2 + \cdots + \lambda_m = 1$ then

$$f\left(\sum_{i=1}^{m} \lambda_i \mathbf{a}_i\right) \leq \sum_{i=1}^{m} \lambda_i f(\mathbf{a}_i).$$

The proof is by induction on m. $m = 2$ is from the definition of convexity and then we use

$$\sum_{i=1}^{m} \lambda_i \mathbf{a}_i = \lambda_m \mathbf{a}_m + (1-\lambda_m) \sum_{i=1}^{m-1} \frac{\lambda_i}{1-\lambda_m} \mathbf{a}_i.$$

Application: Arithmetic versus geometric mean.
Suppose that $a_1, a_2, \ldots, a_m \in \mathbb{R}_+$. Then

$$\frac{a_1 + a_2 + \cdots + a_m}{m} \geq (a_1 a_2 \cdots a_m)^{1/m}. \tag{130}$$

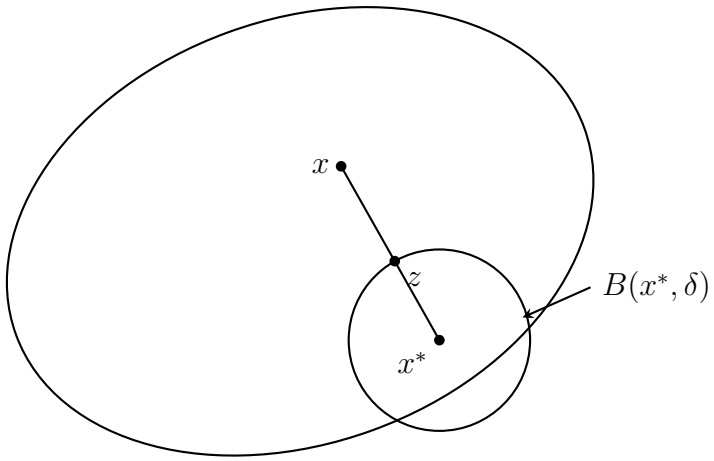
$-\log(x)$ is a convex function for $x > 0$. So, applying Jensen's inequality,

$$-\log\left(\sum_{i=1}^{m} \lambda_i a_i\right) \leq \sum_{i=1}^{m} -\lambda_i \log(a_i).$$

Now let $\lambda_i = 1/m$ for $i = 1, 2, \ldots, m$.

We now prove something that implies the importance of the above notions. Most optimization algorithms can only find local minima. We do however have the following theorem:

Theorem 46. *Let f, S both be convex in* (128)*. Then if $\mathbf{x}^*$ is a local minimum, it is also a global minimum.*



Proof. Let δ be such that $\mathbf{x}^*$ minimises f in $B(\mathbf{x}^*, \delta) \cap S$ and suppose that $\mathbf{x} \in S \setminus B(\mathbf{x}^*, \delta)$. Let $\mathbf{z} = \lambda \mathbf{x}^* + (1-\lambda)\mathbf{x}$ be the point on $L(\mathbf{x}^*, \mathbf{x})$ at distance δ from $\mathbf{x}^*$. Note that $\mathbf{z} \in S$ by convexity of S. Then by the convexity of f we have

$$f(\mathbf{x}^*) \leq f(\mathbf{z}) \leq \lambda f(\mathbf{x}^*) + (1-\lambda) f(\mathbf{x})$$

and this implies that $f(\mathbf{x}^*) \leq f(\mathbf{x})$. □

The following shows the relationship between convex sets and functions.

Lemma 47. *Let $f_1, f_2, \ldots, f_m$ be convex functions on $\mathbb{R}^n$. Let $\mathbf{b} \in \mathbb{R}^m$ and let*

$$S = \{\mathbf{x} \in \mathbb{R}^n : f_i(\mathbf{x}) \leq b_i, i = 1, 2, \ldots, m\}.$$

Then S is convex.

Proof. It follows from O3 that we can consider the case $m = 1$ only and drop the subscript. Suppose now that $\mathbf{x}, \mathbf{y} \in S$ i.e. $f(\mathbf{x}), f(\mathbf{y}) \leq b$. Then for $0 \leq \lambda \leq 1$

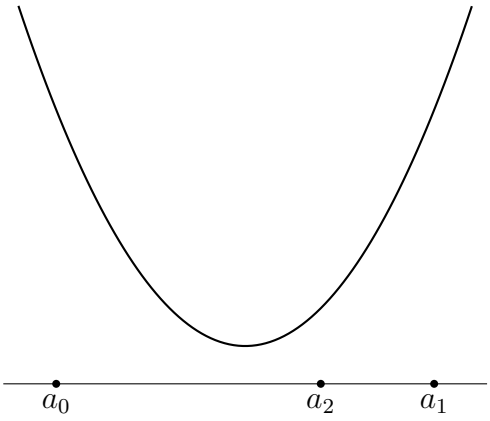
$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \leq \lambda b + (1-\lambda) b = b.$$

So, $\lambda \mathbf{x} + (1-\lambda)\mathbf{y} \in S$. □

17 Algorithms

17.1 Line search − $n = 1$

Here we consider the simpler problem of minimising a convex (more generally *unimodal*) function $f : \mathbb{R} \to \mathbb{R}$.



We assume that we are given a_0, a_1 such that $a_0 \leq x^* \leq a_1$ where x^* minimises f. This is not a significant assumption. We can start with $a_0 = 0$ and then consider the sequences $\zeta_i = f(2^i - 1), \xi_i = f(1 - 2^i)$ until we find $\zeta_{i-1} \leq \min\{\zeta_0, \zeta_i\}$ (resp. $\xi_{i-1} \leq \min\{\xi_0, \xi_i\}$). Then we know that $x^* \in [0, 2^i - 1]$ (resp. $x^* \in [1 - 2^i, 0]$).

Assume then that we have an interval $[a_0, a_1]$ of uncertainty for x^*. Furthermore, we will have evaluated f at two points in this interval, two points inside the interval at $a_2 = a_0 + \alpha^2(a_1 - a_0)$ and $a_3 = a_0 + \alpha(a_1 - a_0)$ respectively. We will determine α shortly. And at each iteration we make one new function evaluation and decrease the interval of uncertainty by a factor α. There are two possibilities:

(i) $f(a_2) \leq f(a_3)$. This implies that $x^* \in [a_0, a_3]$. So, we evaluate $f(a_0 + \alpha^2(a_3 - a_0))$ and make the changes $a_i \to a'_i$:

$$a'_0 \leftarrow a_0,\ a'_1 \leftarrow a_3,\ a'_2 \leftarrow a_0 + \alpha^2(a_3 - a_0),\ a'_3 \leftarrow a_2.$$

(ii) $f(a_2) > f(a_3)$. This implies that $x^* \in [a_2, a_1]$. So, we evaluate $f(a_2 + \alpha(a_1 - a_2))$ and make the changes $a_i \to a'_i$:

$$a'_0 \leftarrow a_2,\ a'_1 \leftarrow a_1,\ a'_2 \leftarrow a_3,\ a'_3 \leftarrow a_2 + \alpha(a_1 - a_2).$$

In case (i) we see that $a'_1 - a'_0 = a_3 - a_0 = \alpha(a_1 - a_0)$ and so the interval has shrunk by the required amount. Next we see that $a'_2 - a'_0 = \alpha^2(a_3 - a_0) = \alpha^2(a'_1 - a_0)$. Furthermore, $a'_3 - a'_0 = a_2 - a_0 = \alpha^2(a_1 - a_0) = \alpha(a'_1 - a'_0)$.

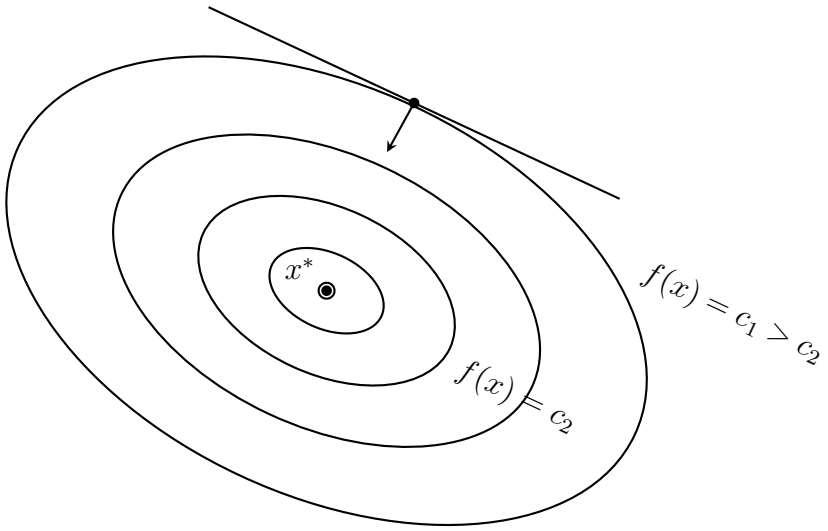
In case (ii) we see that $a'_1 - a'_0 = a_1 - a_2 = a_1 - (a_0 + \alpha^2(a_1 - a_0)) = (1 - \alpha^2)(a_1 - a_0)$. So, to shrink by α in this case we choose α to satisfy $1 - \alpha^2 = \alpha$. This gives us

$$\alpha = \frac{\sqrt{5} - 1}{2} \text{ – } \textit{the inverse of the golden ratio.}$$

Next we see that $a'_2 - a'_0 = a_3 - a_2 = (\alpha - \alpha^2)(a_1 - a_0) = \frac{\alpha - \alpha^2}{\alpha}(a'_1 - a'_0) = (1 - \alpha)(a'_1 - a'_0) = \alpha^2(a'_1 - a'_0)$. Finally, we have $a'_3 - a'_0 = a_2 + \alpha(a_1 - a_2) - a_2 = \alpha(a_1 - a_2) = \alpha(a'_1 - a'_0)$.

Thus to achieve an accuracy within δ of x^* we need to take t steps, where $\alpha^t D \leq \delta$ where D is our initial uncertainty.

17.2 Gradient Descent



Here we consider the unconstrained problem. At a point $\mathbf{x} \in \mathbb{R}^n$, if we move a small distance h in direction $\mathbf{d}$ then we have

$$f(\mathbf{x} + h\mathbf{d}/|\mathbf{d}|) = f(\mathbf{x}) + h(\nabla f)^T \frac{\mathbf{d}}{|\mathbf{d}|} + O(h^2) \geq f(\mathbf{x}) - h|\nabla f| + O(h^2).$$

Thus, at least infinitessimally, the best direction is $-\nabla f$. So, for us, the steepest descent algorithm will follow a sequence of points $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots$, where

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k \qquad \text{where } \mathbf{g}_k = \nabla f(\mathbf{x}_k).$$

Then we have

$$|\mathbf{x}_{k+1} - \mathbf{x}^*|^2 = |\mathbf{x}_k - \mathbf{x}^*|^2 - 2\alpha_k \mathbf{g}_k^T(\mathbf{x}_k - \mathbf{x}^*) + \alpha_k^2 |\mathbf{g}_k|^2 \tag{131}$$

$$\leq |\mathbf{x}_k - \mathbf{x}^*|^2 - 2\alpha_k (f(\mathbf{x}_k) - f(\mathbf{x}^*)) + \alpha_k^2 |\mathbf{g}_k|^2. \tag{132}$$

The inequality comes from F3.

Applying (132) repeatedly we get

$$|\mathbf{x}_k - \mathbf{x}^*|^2 \leq |\mathbf{x}_0 - \mathbf{x}^*|^2 - 2\sum_{i=0}^{k-1} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) + \sum_{i=0}^{k-1} \alpha_i^2 |\mathbf{g}_i|^2. \tag{133}$$

Putting $R = |\mathbf{x}_0 - \mathbf{x}^*|$, we see from (133) that

$$2\sum_{i=0}^{k-1} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) \leq R^2 + \sum_{i=0}^{k-1} \alpha_i^2 |\mathbf{g}_i|^2. \tag{134}$$

On the other hand,

$$\sum_{i=0}^{k-1} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) \geq \left(\sum_{i=0}^{k-1} \alpha_i \right) \min \{ f(\mathbf{x}_i) - f(\mathbf{x}^*) : 0 \leq i < k \} = \left(\sum_{i=0}^{k-1} \alpha_i \right) (f(\mathbf{x}_{min}) - f(\mathbf{x}^*)), \tag{135}$$

where $f(\mathbf{x}_{min}) = \min \{ f(\mathbf{x}_i) : i \in [k] \}$.

Combining (134) and (135) we get

$$f(\mathbf{x}_{min}) - f(\mathbf{x}^*) \leq \frac{R^2 + G^2 \sum_{i=0}^{k-1} \alpha_i^2}{2 \sum_{i=0}^{k-1} \alpha_i}, \tag{136}$$

where G is an assumed uniform upper bound on $|\nabla f(\mathbf{x})|$ for $\mathbf{x} \in \mathbb{R}^n$.

So, if we choose α_k so that $\sum_{i=1}^{\infty} \alpha_i = \infty$ and $\sum_{i=1}^{\infty} \alpha_i^2 = O(1)$ then

$$|f(\mathbf{x}_{min}) - f(\mathbf{x}^*)| \to 0 \text{ as } k \to \infty. \tag{137}$$

As an example, we could let $\alpha_i = 1/i$.

Sub-gradient optimization Note that the only property of $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ used is that $\mathbf{g}_k^T(\mathbf{x}_k - \mathbf{x}^*) \geq f(\mathbf{x}_k) - f(\mathbf{x}^*)$. A vector $\mathbf{g}$ is said to be a *sub-gradient* of the function f at the point $\mathbf{x}$ if $\mathbf{g}^T(\mathbf{x} - \mathbf{y}) \geq f(\mathbf{x}) - f(\mathbf{y})$ for all $\mathbf{y}$. So, sub-gradients can be used to minimise non-differentiable convex functions.

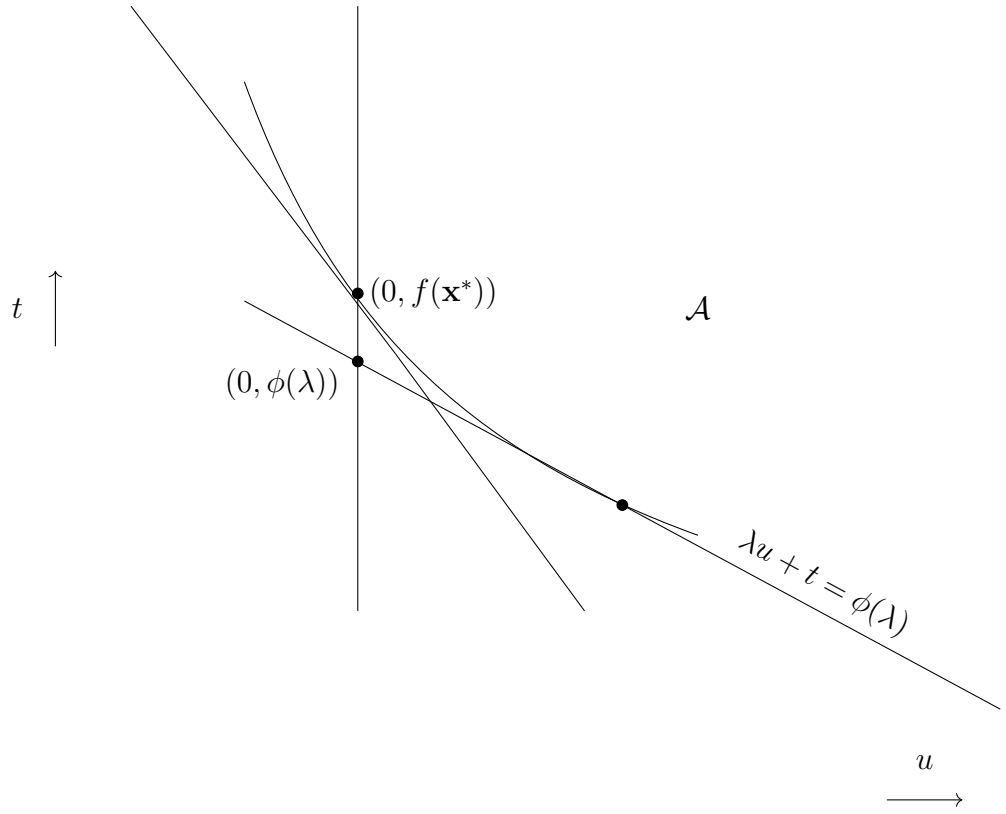
The following is sometimes useful.

Lemma 48. *Suppose that S is a closed bounded convex set and that f is a convex function. Then f achieves its* maximum *at an extreme point.*

Proof. Suppose the maximum occurs at $\mathbf{x} = \lambda_1 \mathbf{x}_1 + \cdots + \lambda_k \mathbf{x}_k$ where $0 \leq \lambda_1, \ldots, \lambda_k \leq 1$ and $\lambda_1 + \cdots + \lambda_k = 1$ and $\mathbf{x}_1, \ldots, \mathbf{x}_k \in ext(S)$. Then by Jensen's inequality we have $f(\mathbf{x}) \leq \lambda_1 f(\mathbf{x}_1) + \cdots + \lambda_k f(\mathbf{x}_k) \leq \max \{ f(\mathbf{x}_i) : 1 \leq i \leq k \}$. □

This explains why the solutions to linear programs occur at extreme points.

18 Lagrangean Duality



Here we consider the *primal problem*

$$\text{Minimize } f(\mathbf{x}) \text{ subject to } g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, m, \tag{138}$$

where $f, g_1, g_2, \ldots, g_m$ are convex functions on $\mathbb{R}^n$.

The Lagrangean

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}).$$

The *dual problem* is

$$\text{Maximize } \phi(\boldsymbol{\lambda}) \text{ subject to } \boldsymbol{\lambda} \geq 0 \text{ where } \phi(\boldsymbol{\lambda}) = \min_{\mathbf{x} \in \mathbb{R}^n} L(\mathbf{x}, \boldsymbol{\lambda}). \tag{139}$$

We note that ϕ is a concave function. It is the minimum of a collection of linear functions of $\boldsymbol{\lambda}$.

D1 Linear programming. Let $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ and $g_i(\mathbf{x}) = -\mathbf{a}_i^T\mathbf{x} + b_i$ for $i = 1, 2, \ldots, m$. Then

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \left(\mathbf{c}^T - \boldsymbol{\lambda}^T A\right)\mathbf{x} + \mathbf{b}^T\boldsymbol{\lambda} \text{ where } A \text{ has rows } \mathbf{a}_1, \ldots, \mathbf{a}_m.$$

It follows that $\mathbf{A}^T\boldsymbol{\lambda} \neq \mathbf{c}$ implies that $\phi(\boldsymbol{\lambda}) = -\infty$. So the dual problem is

$$\text{Maximize } \mathbf{b}^T\boldsymbol{\lambda} \text{ subject to } \mathbf{A}^T\boldsymbol{\lambda} = \mathbf{c},\ \boldsymbol{\lambda} \geq 0.$$

Weak Duality: If $\boldsymbol{\lambda}$ is feasible for (139) and $\mathbf{x}$ is feasible for (138) then $f(\mathbf{x}) \geq \phi(\boldsymbol{\lambda})$.

$$\phi(\boldsymbol{\lambda}) \leq L(\mathbf{x}, \boldsymbol{\lambda}) \leq f(\mathbf{x}) \text{ since } \lambda_i \geq 0, g_i(\mathbf{x}) \leq 0,\ i = 1, 2, \ldots, m. \tag{140}$$

Now note that $\phi(\boldsymbol{\lambda}) = -\infty$, unless $\mathbf{c}^T = \boldsymbol{\lambda}^T A$, since $\mathbf{x}$ is unconstrained in the definition of ϕ. And if $\mathbf{c}^T = \boldsymbol{\lambda}^T A$ then $\phi(\boldsymbol{\lambda}) = \mathbf{b}^T \boldsymbol{\lambda}$. So, the dual problem is to
Maximize $\mathbf{b}^T \boldsymbol{\lambda}$ subject to $\mathbf{c}^T = \boldsymbol{\lambda}^T A$ and $\boldsymbol{\lambda} \geq 0$, i.e. the LP dual.

Strong Duality: We give a sufficient condition *Slater's Constraint Condition* for tightness in (140).

Theorem 49. *Suppose that there exists a point $\mathbf{x}^*$ such that $g_i(\mathbf{x}^*) < 0, i = 1, 2, \ldots, m$. Then*

$$\max_{\boldsymbol{\lambda} \geq \mathbf{0}} \phi(\boldsymbol{\lambda}) = \min_{\mathbf{x}: g_i(\mathbf{x}) \leq 0, i \in [m]} f(\mathbf{x}).$$

Proof. Let

$$\mathcal{A} = \{(\mathbf{u}, t) : \exists \mathbf{x} \in \mathbb{R}^n, g_i(\mathbf{x}) \leq u_i, i = 1, 2, \ldots, m \text{ and } f(\mathbf{x}) \leq t\}.$$
$$\mathcal{B} = \left\{(0, s) \in \mathbb{R}^{m+1} : s < f^*\right\} \text{ where } f^* = \min_{\mathbf{x}: g_i(\mathbf{x}) \leq 0, i \in [m]} f(\mathbf{x}).$$

Now $\mathcal{A} \cap \mathcal{B} = \emptyset$ and so from Corollary 7 there exists $\boldsymbol{\lambda}, \gamma, b$ such that $(\boldsymbol{\lambda}, \gamma) \neq \mathbf{0}$ and

$$b \leq \min\left\{\boldsymbol{\lambda}^T \mathbf{u} + \gamma t : (\mathbf{u}, t) \in \mathcal{A}\right\}. \tag{141}$$
$$b \geq \max\left\{\boldsymbol{\lambda}^T \mathbf{u} + \gamma t : (\mathbf{u}, t) \in \mathcal{B}\right\}. \tag{142}$$

We deduce from (141) that $\boldsymbol{\lambda} \geq 0$ and $\gamma \geq 0$. If $\gamma < 0$ or $\lambda_i < 0$ for some i then the minimum in (141) is $-\infty$. We deduce from (142) that $\gamma t < b$ for all $t < f^*$ and so $\gamma f^* \leq b$. And from (141) that

$$\gamma f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \geq b \geq \gamma f^* \qquad \text{for all } \mathbf{x} \in \mathbb{R}^n. \tag{143}$$

If $\gamma > 0$ then we can divide (143) by γ and see that $L(\mathbf{x}, \boldsymbol{\lambda}/\gamma) \geq f^*$, and together with weak duality, we see that $L(\mathbf{x}, \boldsymbol{\lambda}/\gamma) = f^*$.

If $\gamma = 0$ then substituting $\mathbf{x}^*$ into (143) we see that $\sum_{i=1}^m \lambda_i g_i(\mathbf{x}^*) \geq 0$ which then implies that $\boldsymbol{\lambda} = 0$, contradiction. □

19 Conditions for a minimum: First Order Condition

19.1 Unconstrained problem

We discuss necessary conditions for $\mathbf{a}$ to be a (local) minimum. (We are not assuming that f is convex.) We will assume that our functions are differentiable. Then Taylor's Theorem implies that

$$f(\mathbf{a} + \mathbf{h}) = f(\mathbf{a}) + (\nabla f(\mathbf{a}))^T \mathbf{h} + o(|\mathbf{h}|)$$

implies that

$$\nabla f(\mathbf{a}) = 0 \tag{144}$$

is a necessary condition for $\mathbf{a}$ to be a local minimum. Otherwise,

$$f\left(\mathbf{a} - t\nabla f(\mathbf{a})\right) \leq f(\mathbf{a}) - t|\nabla f(\mathbf{a})|^2/2$$

for small $t > 0$.

Of course (144) is not sufficient in general, $\mathbf{a}$ could be a local maximum. Generally speaking, one has to look at second order conditions to distinguish between local minima and local maxima.

However,

Lemma 50. *If f is convex then* (144) *is also a sufficient condition.*

Proof. This follows directly from F3. □

19.2 Constrained problem

We will consider Problem (138), but we will not assume convexity, only differentiability. The condition corresponding to (144) is the *Karush-Kuhn-Tucker* or KKT condition. Assume that $f, g_1, g_2, \ldots, g_m$ are differentiable. Then (subject to some *regularity conditions*, a necessary condition for $\mathbf{a}$ to be a local minimum to Problem (138) is that there exists $\boldsymbol{\lambda}$ such that

$$g_i(\mathbf{a}) \leq 0, \qquad 1 \leq i \leq m. \tag{145}$$

$$\lambda_i \geq 0 \qquad 1 \leq i \leq m. \tag{146}$$

$$\nabla f(\mathbf{a}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{a}) = 0. \tag{147}$$

$$\lambda_i g_i(\mathbf{a}) = 0, \qquad 1 \leq i \leq m. \qquad \text{Complementary Slackness} \tag{148}$$

Equation (148) says that only *active* constraints ($g_i(\mathbf{a}) = 0$) are involved in (147).

One deals with $g_i(\mathbf{x}) \geq 0$ via $-g_i(\mathbf{x}) \leq 0$ and $g_i(\mathbf{x}) = 0$ by $-g_i(\mathbf{x}) \leq 0$ and $g_i(\mathbf{x}) \leq 0$.

Consider the problem

$$\begin{aligned} &\text{Minimise } (x_1 - 5)^2 + (x_2 - 4)^2 \\ &\text{Subject to } x_1 + x_2 \leq 1 \\ &\qquad\qquad 2x_1 + 3x_2 \leq 2. \\ &\qquad\qquad x_1, x_2 \geq 0. \end{aligned}$$

The KKT conditions for this problem are:

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ 2x_1 + 3x_2 &\leq 2 \\ 2(x_1 - 5) + \lambda_1 + 2\lambda_2 - \mu_1 &= 0 \\ 2(x_2 - 4) + \lambda_1 + 3\lambda_2 - \mu_2 &= 0 \\ \lambda_1(x_1 + x_2 - 1) = \lambda_2(2x_1 + 3x_2 - 2) &= 0 \\ \mu_1 x_1 = \mu_2 x_2 &= 0 \\ \lambda_1, \lambda_2, \mu_1, \mu_2 &\geq 0. \end{aligned}$$

There are 16 possibilities to check: $\lambda_i, \mu_i = 0, > 0, i = 1, 2$.

Case 1: $\lambda_1 = 0, \lambda_2 = 0, \mu_1 = 0, \mu_2 = 0$**:** This gives $x_1 = 5, x_2 = 4$ which is infeasible.

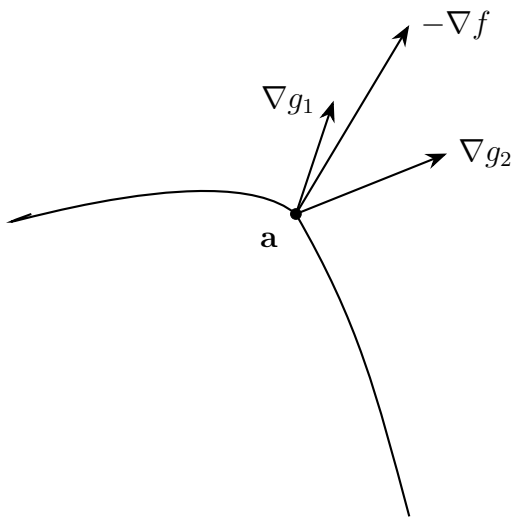
Case 2: $\lambda_1 = 0, \lambda_2 > 0, \mu_1 = 0, \mu_2 = 0$**:** This gives $2x_1 + 3x_2 = 2, x_1 - 5 = -\lambda_2, x_2 - 4 = -3\lambda_2/2$ which implies $x_1 = 25/13, x_2 = -8/13, \lambda_2 = 40/13 > 0$ which is infeasible.

Case 3: $\lambda_1 > 0, \lambda_2 = 0, \mu_1 = 0, \mu_2 = 0$ This gives $x_1 + x_2 = 1, x_1 - 5 = x_2 - 4 = -\lambda_1/2$ which implies that $x_1 = 1, x_2 = 0, \lambda_1 = 8$ which satisfies the KKT conditions.

Thus the solution is $x_1 = 1, x_2 = 0$.

In the convex case, we will see that (147), (146) and (148) are sufficient for a global minimum. Any solution to (145) – (148) serves as a global optimum. Our example is a convex problem and that is why we skipped the cases 4–16.

19.2.1 Heuristic Justification of KKT conditions



Suppose that **a** is a local minimum and assume w.l.o.g. that $g_i(\mathbf{a}) = 0$ for $i = 1, 2, \ldots, m$. Suppose that we cannot find $\lambda_i \geq 0, i = 1, 2, \ldots, m$ such that $\nabla f = -\lambda_1 \nabla g_1(\mathbf{a}) - \cdots - \lambda_m \nabla g_m(\mathbf{a})$. Then the Farkas lemma implies that we can find $\mathbf{h}$ such that $\mathbf{h}^T(-\nabla g_i(\mathbf{a})) \geq 0, i = 1, 2, \ldots, m$ and $\mathbf{h}^T \nabla f(\mathbf{a}) < 0$. But then (heuristically) for small $\varepsilon > 0$, Taylor's theorem implies that we should have $f(\mathbf{a}+\varepsilon\mathbf{h}) = f(\mathbf{a})+\varepsilon\mathbf{h}^T\nabla f(\mathbf{a})+... < f(\mathbf{a})$ and similarly $g_i(\mathbf{a} + \varepsilon\mathbf{h}) \leq g_i(\mathbf{a}) \leq 0$ for $i = 1, 2, \ldots, m$, which contradicts the assumption that **a** is a local minimum.

Convex case: Suppose now that $f, g_1, \ldots, g_m$ are all convex functions and that $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ satisfies the KKT conditions. Now $\boldsymbol{\lambda}^* \geq 0$ implies that $\phi(\mathbf{x}) = L(\mathbf{x}, \boldsymbol{\lambda}^*)$ is a convex function of $\mathbf{x}$. Equation (147) and Lemma 50 implies that $\mathbf{x}^*$ minimises ϕ. But then for any feasible $\mathbf{x}$ we have

$$f(\mathbf{x}^*) = \phi(\mathbf{x}^*) \leq \phi(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i^* g_i(\mathbf{x}) \leq f(\mathbf{x}).$$

19.3 Special cases of non-linear optimization

19.3.1 Fractional Linear Programming

This is the name given to the following problem:

$$\begin{aligned} \text{Maximize } & \frac{\mathbf{c}^T\mathbf{x}+\alpha}{\mathbf{d}^T\mathbf{x}+\beta} \\ \text{Subject to } & \mathbf{A}\mathbf{x} \le \mathbf{b}. \end{aligned} \tag{149}$$

Let $S = \{\mathbf{A}\mathbf{x} \le \mathbf{b}\}$. Here we assume that S is bounded and that $\mathbf{x} \in S$ implies that $\mathbf{d}^T\mathbf{x}+\beta > 0$.

(a) **Bank Portfolio Management**: Maximize the ratio of average return on investment to total risk or total capital allocated, subject to budget limits, risk ceilings, and regulatory requirements.

(b) **Hospital or Production Efficiency**: Maximize the ratio of total patient recovery output (or total units produced) to operational staffing cost, constrained by available nurse hours, facility capacity, and supply budgets.

(c) **Cutting Stock / Paper Industry**: Maximize the ratio of valuable usable trim-loss material to total raw material consumed, constrained by pattern demands and machine tolerances.

It turns out that we can reduce (149) to Linear Programming via the Charnes-Cooper transformation. Consider the LP

$$\begin{aligned} \text{Maximize } & \mathbf{c}^T\mathbf{y}+\alpha t \\ \text{Subject to } & \mathbf{A}\mathbf{y} \le t\mathbf{b}. \\ & \mathbf{d}^T\mathbf{y}+\beta t = 1. \\ & t \ge 0. \end{aligned} \tag{150}$$

Lemma 51. *Suppose that* $\mathbf{y}, t$ *solves* (150). *Then* $\mathbf{x} = \mathbf{y}/t$ *solves* (149).

Proof. We first show that $t > 0$. If $t = 0$ then there exists $\mathbf{y}$ such that $\mathbf{A}\mathbf{y} \le 0$. Then if $\mathbf{x} \in S$ then $\mathbf{x}+\lambda\mathbf{y} \in S$ for all $\lambda \ge 0$, contradicting the assumption that S is bounded.

Suppose that $\mathbf{y}, t > 0$ solves (150) and $\mathbf{x} = \mathbf{y}/t$ and $\mathbf{A}\mathbf{z} \le \mathbf{b}$. Let $\widehat{t} = 1/(\mathbf{d}^T\mathbf{z}+\beta)$ so that $\mathbf{d}^T\mathbf{w}+\beta\widehat{t} = 1$ where $\mathbf{w} = \widehat{t}\mathbf{z}$. Then because $\mathbf{w}$ is feasible for (150), we have

$$\frac{\mathbf{c}^T\mathbf{x}+\alpha}{\mathbf{d}^T\mathbf{x}+\beta} = \frac{\mathbf{c}^T\mathbf{y}+\alpha t}{\mathbf{d}^T\mathbf{y}+\beta t} = \mathbf{c}^T\mathbf{y}+\alpha t \ge \mathbf{c}^T\mathbf{w}+\alpha\widehat{t} = \frac{\mathbf{c}^T\mathbf{w}+\alpha\widehat{t}}{\mathbf{d}^T\mathbf{w}+\beta\widehat{t}} = \frac{\mathbf{c}^T\mathbf{z}+\alpha}{\mathbf{d}^T\mathbf{z}+\beta}.$$

□

19.3.2 Geometric Programming

A *monomial* is a product $x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}$ and a function $f : \mathbb{R}_+^n \to \mathbb{R}$ is a *posynomial* if $f = \sum_{k=1}^m c_k f_k$ where $c_k > 0$ and f_k is a monomial for $k = 1, 2, \ldots, m$. A *Geometric Program* is an optimization problem of the form

$$\begin{aligned} \text{Minimize } & f_0(\mathbf{x}) \\ \text{Subject to } & f_i(\mathbf{x}) \le 1, i = 1, 2, \ldots, p. \\ & g_j(\mathbf{x}) = 1, j = 1, 2, \ldots, q. \end{aligned} \tag{151}$$

Here $f_i, i = 0, 1, \ldots, p$ are posynomials and $g_j, j = 1, 2, \ldots, q$ are monomials.

Simple application: optimize the shape of a box-shaped structure with height h, width w, and depth d. We have a limit on the total wall area $2(hw + hd)$, and the floor area wd, as well as lower and upper bounds on the aspect ratios h/w and w/d. Subject to these constraints, we wish to maximize the volume of the structure, hwd. This leads to the problem

$$\begin{array}{ll} \text{maximize} & hwd \\ \text{subject to} & 2(hw + hd) \leq A_{\text{wall}}, \quad wd \leq A_{\text{flr}}, \\ & \alpha \leq h/w \leq \beta, \quad \gamma \leq d/w \leq \delta. \end{array} \tag{152}$$

Here d, h, and w are the optimization variables, and the problem parameters are A_{wall} (the limit on wall area), A_{flr} (the limit on floor area), and $\alpha, \beta, \gamma, \delta$ (the lower and upper limits on the wall and floor aspect ratios). This problem is a GP (in the extended sense, using the simple transformations described above). It can be transformed to the standard form GP

$$\begin{array}{ll} \text{minimize} & h^{-1}w^{-1}d^{-1} \\ \text{subject to} & (2/A_{\text{wall}})hw + (2/A_{\text{wall}})hd \leq 1, \quad (1/A_{\text{flr}})wd \leq 1, \\ & \alpha h^{-1}w \leq 1, \quad (1/\beta)hw^{-1} \leq 1, \\ & \gamma wd^{-1} \leq 1, \quad (1/\delta)w^{-1}d \leq 1. \end{array} \tag{153}$$

We can convert (151) into a convex program as follows: let $y_i = \log x_i, i = 1, 2, \ldots, n$ and replace (151) by

$$\begin{array}{rl} \text{Minimize} & \log f_0(\mathbf{y}) \\ \text{Subject to} & \log f_i(\mathbf{y}) \leq 0, i = 1, 2, \ldots, p. \\ & \log g_j(\mathbf{y}) = 0, j = 1, 2, \ldots, q. \end{array} \tag{154}$$

Suppose that if $\phi(\mathbf{x}) = \sum_{k=1}^{\ell} c_k x_1^{a_{k,1}} x_2^{a_{k,2}} \cdots x_n^{a_{k,n}}$ and we put $x_i = e^{y_i}$ then

$$\log \phi = \psi(\mathbf{y}) = \log \left(\sum_{k=1}^{m} \exp \left\{ \mathbf{a}_k^T \mathbf{y} + b_k \right\} \right)$$

where $\mathbf{a}_k = (a_{k,1}, \ldots, a_{k,n})$ and $b_k = \log c_k$. Now putting $z_k = \mathbf{a}^T \mathbf{y} + \mathbf{b}$, to show convexity of ϕ, we need only show convexity of the LogSumExp function $\psi(z_1, z_2, \ldots, z_m) = \log(e^{z_1} + \cdots e^{z_m})$ and then apply E4 of Section 16.0.2. We leave it as an execise to show that ψ satisfies F8 of Section 16.0.1.

19.3.3 Semi-definite Programming

A semi-definite program has the form

$$\begin{array}{rl} \text{Minimise} & \displaystyle\sum_{i,j=1}^{n} c_{i,j} x_{i,j} \\ \text{Subject to} & \displaystyle\sum_{i,j=1}^{n} a_{i,j,k} x_{i,j} = b_k, k = 1, 2, \ldots, m. \\ & x_{i,j} = x_{j,i}, i, j = 1, 2, \ldots, n. \\ & X = (x_{i,j}) \text{ is positive semi-definite i.e. } \mathbf{y}^T X \mathbf{y} \geq 0, \forall \mathbf{y} \in \mathbb{R}^n. \end{array} \tag{155}$$

The set $S = \left\{X \in \mathbb{R}^{n^2} : X \text{ is positive semi-definite}\right\}$ is convex. If $\mathbf{A}, \mathbf{B} \in S$ and $0 \leq \lambda \leq 1$ then

$$\mathbf{x}^T(\lambda\mathbf{A} + (1-\lambda)\mathbf{B})\mathbf{x} = \lambda\mathbf{x}^T\mathbf{A}\mathbf{x} + (1-\lambda)\mathbf{x}^T\mathbf{B}\mathbf{x} \geq 0.$$

A positive semi-definite matrix $X = (x_{i,j})$ can be expressed as $X = (\mathbf{v}_i^T\mathbf{v}_j)$ where $\mathbf{v}_i \in \mathbb{R}^n, i = 1, 2, \ldots, n$. So, we can replace (155) by

$$\begin{aligned} &\text{Minimise } \sum_{i,j=1}^{n} c_{i,j}\mathbf{v}_i^T\mathbf{v}_j \\ &\text{Subject to } \sum_{i,j=1}^{n} a_{i,j,k}\mathbf{v}_i^T\mathbf{v}_j = b_k, k = 1, 2, \ldots, m. \end{aligned} \tag{156}$$

We show to use semi-definite programming in a heuristic for the Max-Cut problem. Given a graph $G = ([n], E)$ with non-negative edge weights $w : E \to \mathbb{R}$, the Max-Cut problem is to

$$\text{Maximise } w(S) = \sum_{i \in S, j \in \bar{S}} w_{i,j} \text{ over } S \subseteq V.$$

This is NP-hard (as opposed to Min-Cut) and the following algorithm of Goemans and Williamson illustrates some nice ideas. As an integer program we can express Max-Cut as

$$\begin{aligned} &\text{Maximise } \frac{1}{2}\sum_{\{i,j\} \in E}^{n} w_{i,j}(1 - y_i y_j) \\ &\text{Subject to } y_i = \pm 1, i = 1, 2, \ldots, n. \end{aligned} \tag{157}$$

We relax this to

$$\begin{aligned} &\text{Maximise } \frac{1}{2}\sum_{\{i,j\} \in E}^{n} w_{i,j}(1 - \mathbf{v}_i^T\mathbf{v}_j) \\ &\text{Subject to } \mathbf{v}_i \in \mathbb{R}^n, |\mathbf{v}_i| = 1, i = 1, 2, \ldots, n. \end{aligned} \tag{158}$$

This is a relaxation because given a feasible solution to (157) we get a solution to (158) by putting $\mathbf{v}_i = (y_i, 0, \ldots, 0)$, $i = 1, 2, \ldots, n$. Let Z_{SDP} denote the optimum value for (158).

Algorithm:

Step 1 Solve (158).

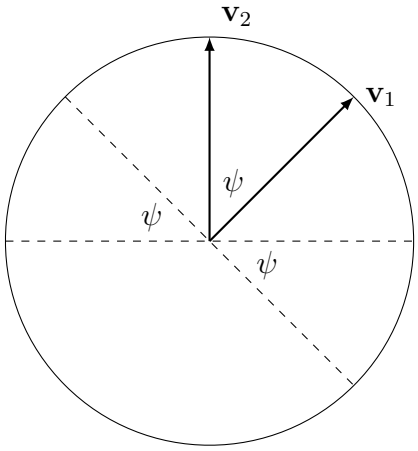
Step 2 Choose a random vector $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ where each $r_i = N(0, 1)$, independently. Then construct a solution to (157) by putting $y_i = 1$ if $\mathbf{r}^T\mathbf{v}_i \geq 0$.

Theorem 52. *If $\widehat{C}$ is the cut produced by the algorithm and C^* is the maximum cut then $\mathbb{E}w(\widehat{C}) \geq 0.878w(C^*)$.*

Proof. We begin by proving that if $\psi = \arccos(\mathbf{v}_i^T\mathbf{v}_j)$ then

Claim 2. $\mathbb{P}(\{i, j\} \in \widehat{C}) = \psi/\pi$.

Proof of Claim 2
Let $\mathbf{r}'$ denote the projection of $\mathbf{r}$ onto the plane Π generated by $\mathbf{v}_1, \mathbf{v}_2$. Then $\mathbf{u} = \mathbf{r}'/|\mathbf{r}'|$ is uniformly chosen from unit circle $\mathcal{C}$ of Π and $\mathbf{r}^T\mathbf{v}_i = \mathbf{u}^T\mathbf{v}_i, i = 1, 2$. Let U, V_1, V_2 be the points on $\mathcal{C}$ corresponding to $\mathbf{u}, \mathbf{v}_1, \mathbf{v}_2$ and let D_i be the half-disc of points making a negative inner product with $\mathbf{v}_i, i = 1, 2$. Then $\{i, j\} \in \widehat{C}$ iff $\mathbf{u} \in (D_1 \setminus D_2) \cup D_2 \setminus D_1)$, which subtends 2ψ of the circumference of $\mathcal{C}$.



End of proof of Claim 2
We next observe that

$$\frac{\arccos(x)}{\pi} \geq 0.439(1-x) \text{ for } -1 \leq x \leq 1. \tag{159}$$

So now let $X_{i,j}$ be the indicator for edge $\{i, j\} \in \widehat{C}$. Then,

$$\mathbb{E}(w(\widehat{C})) = \sum_{\{i,j\} \in E} \frac{w_{i,j} \arccos(\mathbf{v}_i^T \mathbf{v}_j)}{\pi} \geq 0.439 \sum_{\{i,j\} \in E} w_{i,j}(1 - \mathbf{v}_i^T \mathbf{v}_j) \geq 0.878 Z_{SDP} \geq 0.878 w(C^*).$$

□

20 Lagrangian Relaxation

Lagrangian Relaxation is a fundamental technique in Combinatorial Optimization used to compute tight bounds by relaxing "hard" constraints into the objective function via non-negative penalty multipliers. These bounds can then be used in a branch and bound algorithm.

In many structured IP problems, constraints naturally partition into two sets:

1. *Easy Constraints ($\mathbf{x} \in X$):* Constraints preserving a tractable structure (e.g., network flow, minimum spanning tree.).
2. *Hard Constraints ($\mathbf{Ax} \leq \mathbf{b}$):* Complicating constraints that destroy the special structure and render the problem NP-hard.

By removing $\mathbf{Ax} \leq \mathbf{b}$ from the constraint set and penalizing their violation in the objective function using Lagrange multipliers $\lambda \geq 0$, the problem decomposes into manageable subproblems.

20.1 Mathematical Formulation

Consider a problem:

$$z^* = \max_{\mathbf{x}} \mathbf{c}^T\mathbf{x} \tag{160}$$

$$\text{s.t. } \mathbf{x} \in X \quad \text{(easy constraints)} \tag{161}$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \quad \text{(hard constraints)} \tag{162}$$

The Lagrangian Dual Problem Define $L^* = \min_{\boldsymbol{\lambda}\geq 0} \phi(\boldsymbol{\lambda})$ where

$$\phi(\boldsymbol{\lambda}) = \max_{\mathbf{x}\in X} \left\{\mathbf{c}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{b} - \mathbf{A}\mathbf{x})\right\} = \max_{\mathbf{x}\in X} \left\{(\mathbf{c}^T - \boldsymbol{\lambda}^T\mathbf{A})\mathbf{x}\right\} + \boldsymbol{\lambda}^T\mathbf{b}. \tag{163}$$

We denote the maximization problem on the RHS of (163) by $LR(\boldsymbol{\lambda})$.

Note that $\phi(\boldsymbol{\lambda}) \geq z^*$ for any $\boldsymbol{\lambda} \geq 0$ and so L^* provides a valid upper bound on the optimal value z^*. We note in fact that

$$L^* = \max\left\{\mathbf{c}^T\mathbf{x} :\ \mathbf{x} \in \mathit{conv}(X),\ \mathbf{A}\mathbf{x} \leq \mathbf{b}\right\}. \tag{164}$$

Here is a proof of (164) for the case where X is finite. It uses (48), so that $\mathit{conv}(X) = \{\mathbf{x} : \mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1\}$.

$$\begin{aligned}
L^* &= \min_{\boldsymbol{\lambda}\geq 0} \max_{\mathbf{x}\in X} \left\{\mathbf{c}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{b} - \mathbf{A}\mathbf{x})\right\} \\
&= \min_{\boldsymbol{\lambda}\geq 0} \max_{\mathbf{x}\in \mathit{conv}(X)} \left\{\mathbf{c}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{b} - \mathbf{A}\mathbf{x})\right\} \\
&= \min_{\boldsymbol{\lambda}\geq 0} \max_{\mathbf{x}} \left\{\mathbf{c}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{b} - \mathbf{A}\mathbf{x}) : \mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1\right\} \\
&= \min_{\boldsymbol{\lambda}\geq 0} \min_{\boldsymbol{\mu}\geq 0} \left\{\mathbf{b}_1^T\boldsymbol{\mu} + \boldsymbol{\lambda}^T\mathbf{b} : \mathbf{A}_1^T\boldsymbol{\mu} = \mathbf{c} - \mathbf{A}^T\boldsymbol{\lambda}\right\} \\
&= \max_{\mathbf{x}} \left\{\mathbf{c}^T\mathbf{x} : \mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1, \mathbf{A}\mathbf{x} \leq \mathbf{b}\right\}.
\end{aligned}$$

20.2 Solving the Dual: Subgradient Optimization

The dual function $\phi(\boldsymbol{\lambda})$ is convex, continuous, but not necessarily differentiable. The standard method for finding L^* is *Subgradient Optimization*, a generalisation of steepest descent:

1. Initialize multipliers $\boldsymbol{\lambda}_0 \geq 0$.
2. At iteration k, solve $LR(\boldsymbol{\lambda}_k)$ to obtain an optimal point $\mathbf{x}_k \in X$.
3. Compute the subgradient vector:
$$\mathbf{g}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k \tag{165}$$
4. Update the multiplier vector:
$$\boldsymbol{\lambda}_{k+1} = (\boldsymbol{\lambda}_k - \theta_k\mathbf{g}_k)^+. \tag{166}$$
where the step size θ_k is typically computed using the Held–Karp rule:
$$\theta_k = \frac{\gamma_k\left(\phi(\boldsymbol{\lambda}_k) - z_{LB}\right)}{\|\mathbf{g}_k\|^2}, \quad \gamma_k \in (0, 2] \tag{167}$$
with z_{LB} being a known lower bound on z^*.

20.2.1 Convergence

We can prove convergence as we did in Section 17.2 if we prove that $\mathbf{g}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ is a sub-gradient. So, let

$$\phi(\boldsymbol{\lambda}) = \begin{cases} \max_{\mathbf{x}\in X} \left\{(\mathbf{c}^T - \boldsymbol{\lambda}^T\mathbf{A})\mathbf{x}\right\} + \boldsymbol{\lambda}^T\mathbf{b} & \boldsymbol{\lambda} \geq 0. \\ \infty & \text{otherwise.} \end{cases}$$

and let $\mathbf{x}_{\boldsymbol{\theta}}$ maximise $(\mathbf{c}^T - \boldsymbol{\theta}^T\mathbf{A})\mathbf{x}$. Then we have to show that for $\boldsymbol{\lambda}, \boldsymbol{\mu} \geq 0$,

$(\mathbf{b} - \mathbf{A}\mathbf{x}_{\boldsymbol{\lambda}})^T(\boldsymbol{\lambda} - \boldsymbol{\mu}) \geq \phi(\boldsymbol{\lambda}) - \phi(\boldsymbol{\mu})$ or

$(\mathbf{b} - \mathbf{A}\mathbf{x}_{\boldsymbol{\lambda}})^T(\boldsymbol{\lambda} - \boldsymbol{\mu}) \geq ((\mathbf{c}^T - \boldsymbol{\lambda}^T\mathbf{A})\mathbf{x}_{\boldsymbol{\lambda}} + \boldsymbol{\lambda}^T\mathbf{b}) - ((\mathbf{c}^T - \boldsymbol{\mu}^T\mathbf{A})\mathbf{x}_{\boldsymbol{\mu}} + \boldsymbol{\mu}^T\mathbf{b})$ or after cancelling terms

$\mathbf{c}^T\mathbf{x}_{\boldsymbol{\lambda}} - \boldsymbol{\mu}^T\mathbf{A}\mathbf{x}_{\boldsymbol{\lambda}} \leq \mathbf{c}^T\mathbf{x}_{\boldsymbol{\mu}} - \boldsymbol{\mu}^T\mathbf{A}\mathbf{x}_{\boldsymbol{\mu}}$, which is clear.

20.3 Examples

Our examples are minimization problems, that can easily be made into maximisation problems.

Traveling Salesperson Problem (TSP) on a graph $G = (V = [n], E)$:

- X is the set of 1-trees of G. A 1-tree consists of a spanning tree of $[2, n]$ plus two edges joining vertex 1 to this tree.
- In place of (162) we have the node degree constraints, $\sum_{e\ni i} x_e = 2$ for all $i \in V$.

So, an integer solution would be a 1-tree where every vertex has degree 2, i.e. a Hamilton cycle. Computing the dual then becomes finding a minimum cost 1-tree where the cost of an edge $e = \{i, j\}$ is $c(e) - \lambda_i - \lambda_j$. (Here we do not impose $\boldsymbol{\lambda} \geq 0$, why?)

Depot Location Recall that as an integer program this is

$$\begin{aligned} \text{Minimise } & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} + \sum_{i=1}^{m} f_i y_i \\ \text{Subject to } & \sum_{i=1}^{m} x_{i,j} = 1, \qquad j = 1, 2, \ldots, n \\ & x_{i,j} \leq y_i, \quad i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n \\ & x_{i,j} \geq 0,\ 0 \leq y_i \leq 1, y_i \text{ integer}, \qquad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n. \end{aligned}$$

- In place of (162) we have the constraints, $\sum_{i=1}^{m} x_{i,j} = 1$ for all $j = 1, 2, \ldots, n$.
- The dual problem, for a fixed $\boldsymbol{\lambda}$, then becomes to minimize over $\mathbf{x}, \mathbf{y}$

$$\sum_{i=1}^{m}\left(f_i y_i + \sum_{j=1}^{n}(c_{i,j} - \lambda_j)x_{i,j}\right) + \sum_{j=1}^{n}\lambda_j \text{ subject to } x_{i,j} \leq y_i, \forall i, j.$$

  To do this, we compute $z_i = f_i + \sum_{j=1}^{n} \min\{0, c_{i,j} - \lambda_j\}$ for each i. If $z_i < 0$ then we take $y_i = 1$ and put $x_{i,j} = 1$ whenever $c_{i,j} < \lambda_j$. If $z_i \geq 0$ then we put $y_i = 0 = x_{i,j}$ for all j.

For much more on this subject see Convex Optimization, by Boyd and Vandenberghe.

21 Average case of optimization problems

21.1 TSP in the unit square

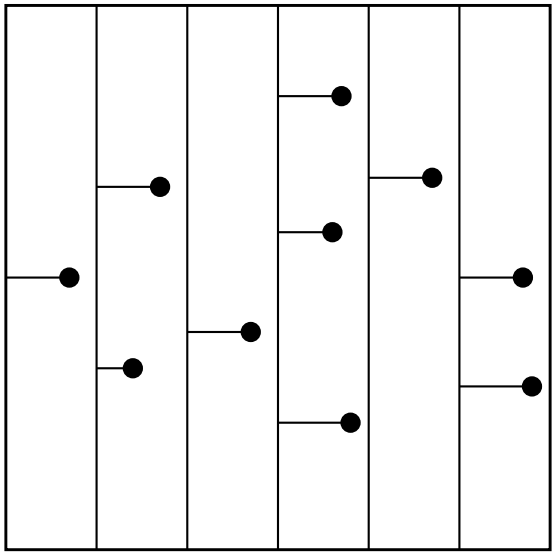
Here we consider the length of the shortest polygon through n points $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ independently chosen uniformly at random in the unit square $[0,1]^2$. We denote this value by $L(\mathcal{X})$.

21.1.1 Simple deterministic upper bound

We first give a simple proof of

Theorem 53. *$L(\mathcal{X}) \leq 3n^{1/2} + 2 + 2^{1/2}$ for any choice of $\mathcal{X}$.*

Proof



In this diagram the small black discs represent the points of $\mathcal{X}$. There are $m = \lceil n^{1/2} \rceil + 1$ vertical lines and they are $1/m$ apart. We visit the points, starting in the bottom left hand corner. We traverse the odd numbered vertical lines from bottom to top and the even numbered lines from top to bottom. We use the upper and lower boundaries to go from vertical line to vertical line. In this way we travel distance $m + 2$ in order to traverse all vertical lines.

We add the points as follows: as we traverse a vertical line and become on the same level as an unvisited point, we make a horizontal excursion to the point and back. This adds at most $2n/m \leq 2n^{1/2}$ to the length of the journey.

We now have a journey that visits the points in some order $X_{i_1}, X_{i_2}, \ldots, X_{i_n}$. We make our tour in this order, going directly from point to point and then adding the edge $\{X_{i_n}, X_{i_1}\}$. □

The following greedy algorithm also finds an $O(n^{1/2})$ tour: start with $T = \emptyset$; then, until the very end, always choose the shortest edge e that (i) does not create a cycle or (ii) a vertex of degree 3.

Theorem 54. *Greedy builds a tour of length at most* $3n^{1/2}$.

Proof. After we have added k edges, T will consist of $n-k$ paths, some of which may be single vertices. Choose one end vertex from each such path to create a set S_k of size $n-k$. Greedy will choose an edge that has length at most the minimum length edge with both ends in S_k. Now, as we will see:

$$\text{Any set } S \text{ of } m \text{ points in } [0,1]^2 \text{contains a pair of points at distance at most } 2^{1/2}/\left\lfloor (m-1)^{1/2} \right\rfloor. \tag{168}$$

Thus the length of the Greedy tour is at most

$$2^{1/2}\left(1+\sum_{k=1}^{n-1}\frac{1}{\lfloor (n-k)^{1/2}\rfloor}\right) \le 3n^{1/2}$$

for large n.

Proof of (168) Let $m_1 = \left\lfloor (m-1)^{1/2} \right\rfloor$ and divide $[0,1]^2$ into $m_1^2 \le m-1$ squares of side $1/m_1$. The pigeon hole principle implies that one of the squares contains 2 points of S. The diameter of a square is $2^{1/2}/m_1$. □

21.1.2 Random points: lower bound on tour length

Now let $\mathcal{X}$ be randomly chosen. We divide $[0,1]^2$ into n subsquares of side $n^{-1/2}$. Then we divide each subsquare into 9 cells of side $n^{-1/2}/3$. We say that a point of $\mathcal{X}$ is *isolated* if it (i) lies in the middle cell of some subsquare and (ii) there are no other points of $\mathcal{X}$ in the square. Let I denote the set of isolated points. We begin by proving

Lemma 55. $|I| \ge n/(20e)$ *w.h.p.*[1]

Proof. We will use the Chebyshev inequality. If n is large then

$$\mathbb{E}|I| = n\mathbb{P}(X_1 \in I) = n \times \frac{1}{9}\left(1-\frac{1}{n}\right)^{n-1} \ge \frac{n}{10e}.$$

For the variance

$$\begin{aligned}
\mathbb{E}(|I|(|I|-1) &= \sum_{1\le i\ne j\le n}\mathbb{P}(X_i, X_j \in I) = n(n-1)\mathbb{P}(X_1, X_2 \in I)\\
&= n(n-1)\mathbb{P}(X_2 \in I \mid X_1 \in I)\mathbb{P}(X_1 \in I) = n(n-1)\frac{\left(1-\frac{1}{n}\right)^{n-1}}{9}\mathbb{P}(X_2 \in I \mid X_1 \in I)\\
&= \frac{n(n-1)\left(1-\frac{1}{n}\right)^{n-1}}{9} \times \frac{1}{9} \times \left(\frac{1-\frac{2}{n}}{1-\frac{1}{n}}\right)^{n-2} = \frac{(n-1)^2\left(1-\frac{2}{n}\right)^{n-2}}{9^2}\\
&= \left(\frac{n(n-2)}{(n-1)^2}\right)^{n-2}(\mathbb{E}|I|)^2 \le (\mathbb{E}|I|)^2.
\end{aligned}$$

[1] A sequnce of events $\mathcal{E}_n, n \ge 1$ is said to occur *with high probability* (w.h.p.) if $\mathbb{P}(\mathcal{E}_n) \to 1$ as $n \to \infty$.

So,

$$\mathbb{V}ar(|I|) = \mathbb{E}((|I|(|I|-1)) + \mathbb{E}(|I|) - (\mathbb{E}|I|)^2 \le \mathbb{E}(|I|).$$

The Chebyshev inequality then implies that

$$\mathbb{P}(|I| \le \tfrac{1}{2}\mathbb{E}|I|) \le \frac{\mathbb{V}ar|I|}{(\mathbb{E}|I|)^2/4} \le \frac{4}{\mathbb{E}|I|} \to 0.$$

□

Now if $X_i \in I$ then the distance to its closest other point in $\mathcal{X}$ is at least $\frac{1}{3}n^{-1/2}$. Thus w.h.p.

$$L(\mathcal{X}) \ge \frac{|I|}{3n^{1/2}} \ge \frac{n^{1/2}}{60e}. \tag{169}$$

It has been known for many years that w.h.p. $L(\mathcal{X})/n^{1/2} \to \beta$ where β is an (unknown) constant.

21.1.3 Karp's partitioning heuristic

We begin by partitioning $[0,1]^2$ into m^2 squares $S_1, S_2, \ldots, S_{m^2}$ of side $1/m$ where $m = \lceil (n/\log n)^{1/2} \rceil$.

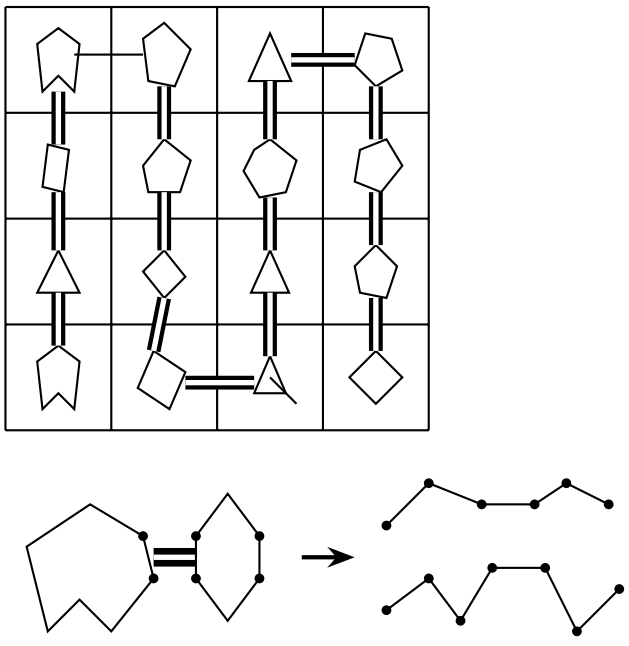
Lemma 56. *W.h.p. no subsquare contains more than $4\log n$ points of $\mathcal{X}$.*

Proof. Fix a subsquare S. Then, using (184),

$$\mathbb{P}(|S \cap \mathcal{X}| \ge 4\log n) = \mathbb{P}\left(Bin\left(n, \frac{\log n}{n}\right) \ge 4\log n\right) \le \left(\frac{e}{4}\right)^{4\log n} \le n^{-1.2}.$$

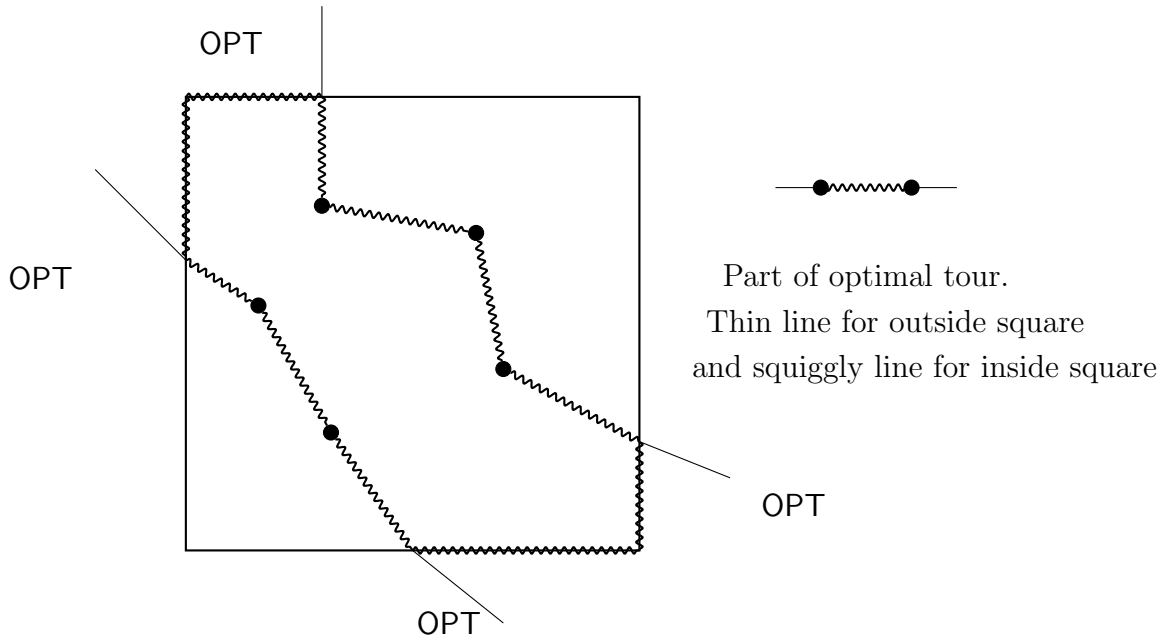
So the probability there is a subsquare with more than $4\log n$ points is at most $n \times n^{-1.2} \to 0$. □

Karp's heuristic goes as follows: solve the n TSP's where the points are restricted to the points in a single square. Using dynamic programming, we see from Lemma 56 that w.h.p. this takes $O((n/\log n) \times \log^2 n \times 2^{4\log n}) = O(n^4)$ time (with care one can improve on the 5.) After this we connect these n small tours $T_1, T_2, \ldots, T_m$ into a single tour through all of $\mathcal{X}$ as shown:



All we need now is to show that if $\widehat{L}$ is the length of the tour produced by the heuristic then

Lemma 57. *W.h.p* $\widehat{L} \le L(\mathcal{X}) + o(n^{1/2})$.



Proof

The diagram shows the optimal tour entering and leaving a fixed square in several places. (Note that the optimal tour does not cross itself. Any tour that crosses itself can be shortened via an uncrossing. Basically because for any quadrilateral the sum of the lengths of the diagonals is less than the sum of the lengths of two opposite sides.) The length of the squiggly polygon in square i can be written as the sum $\ell_i + p_i$ where ℓ_i is the length of the parts of the optimal tour that go through square S_i and p_i is the length of the parts on the perimeter of the square. Now we have $\ell_i + p_i \ge \ell(T_i)$ for $i = 1, 2, \ldots, m^2$, since T_i is an optimal tour of S_i. So,

$$\widehat{L} \le \sum_{i=1}^{m^2} \ell_i + \sum_{i=1}^{m^2} p_i + m^2 \times \frac{5^{1/2}}{m} \le L(\mathcal{X}) + (4m+1) + 5^{1/2}m.$$

Now $m = o(n^{1/2})$ and the lemma follows.

21.2 Shortest path problem

Let the edges of the complete digraph $\vec{K}_n$ on $[n]$ be given independent lengths $X_e,\ e \in [n]^2$. Here X_e is exponentially distributed with mean 1 i.e. $\mathbb{P}(X_e \ge t) = e^{-t}$.

Theorem 58. *Let $X_{i,j}$ be the distance from vertex i to vertex j in the complete graph with edge weights independent $EXP(1)$ random variables. Then, for every $\epsilon > 0$, as $n \to \infty$,*

(i) For any fixed i, j,

$$\mathbb{E}X_{i,j} = \frac{1}{n}\sum_{k=1}^{n-1}\frac{1}{k}.$$

$$\mathbb{P}\left(\left|X_{i,j} - \frac{\log n}{n}\right| \ge \frac{\epsilon \log n}{n}\right) \to 0, \quad \textit{for any fixed } \varepsilon > 0.$$

(ii) For any fixed i,

$$\mathbb{E}\max_j X_{i,j} = \frac{2}{n}\sum_{k=1}^{n-1}\frac{1}{k}.$$
$$\mathbb{P}\left(\left|\max_j X_{i,j} - \frac{2\log n}{n}\right| \geq \frac{\epsilon\log n}{n}\right) \to 0, \quad \textit{for any fixed } \varepsilon > 0.$$

Proof. First, recall the following two properties of the exponential X:

(P1) $\mathbb{P}(X > \alpha+\beta | X > \alpha) = \mathbb{P}(X > \beta)$.

(P2) If $X_1, X_2, \ldots, X_m$ are independent $EXP(1)$. exponential random variables then $\min\{X_1, X_2, \ldots, X_m\}$ is an exponential with mean $1/m$.

Suppose that we want to find shortest paths from a vertex s to all other vertices in a digraph with non-negative arc-lengths. Recall Dijkstra's algorithm. After several iterations there is a rooted tree T such that if v is a vertex of T then the tree path from s to v is a shortest path. Let $d(v)$ be its length. For $x \notin T$ let $d(x)$ be the minimum length of a path P that goes from s to v to x where $v \in T$ and the sub-path of P that goes to v is the tree path from s to v. If $d(y) = \min\{d(x) : x \notin T\}$ then $d(y)$ is the length of a shortest path from s to y and y can be added to the tree.

Suppose that vertices are added to the tree in the order $v_1, v_2, \ldots, v_n$ and that $Y_j = dist(v_1, v_j)$ for $j = 1, 2, \ldots, n$. It follows from property P1 that

$$Y_{k+1} = \min_{\substack{i=1,2,\ldots,k \\ v \neq v_1,\ldots,v_k}} [Y_i + X_{v_i,v}] = Y_k + E_k$$

where E_k is exponential with mean $\frac{1}{k(n-k)}$ and is independent of Y_k. This is because X_{v_i,v_j} is distributed as an independent exponential X conditioned on $X \geq Y_k - Y_i$. Hence

$$\mathbb{E}Y_n = \sum_{k=1}^{n-1}\frac{1}{k(n-k)} = \frac{1}{n}\sum_{k=1}^{n-1}\left(\frac{1}{k}+\frac{1}{n-k}\right) = \frac{2}{n}\sum_{k=1}^{n-1}\frac{1}{k} = \frac{2\log n}{n} + O(n^{-1}).$$

Also, from the independence of E_k, Y_k,

$$\mathbb{V}arY_n = \sum_{k=1}^{n-1}\mathbb{V}arE_k = \sum_{k=1}^{n-1}\left(\frac{1}{k(n-k)}\right)^2 \leq 2\sum_{k=1}^{n/2}\left(\frac{1}{k(n-k)}\right)^2 \leq \frac{8}{n^2}\sum_{k=1}^{n/2}\frac{1}{k^2} = O(n^{-2})$$

and we can use the Chebyshev inequality to prove (ii). Indeed, $Y_n = \max_j\{X_{i,j}\}$ and so

$$\mathbb{P}\left(\left|\max_j X_{i,j} - \frac{2\log n}{n}\right| \geq \frac{\epsilon\log n}{n}\right) \leq \frac{\mathbb{V}arY_n}{(\varepsilon\log n/n)^2} = O\left(\frac{1}{\log^2 n}\right).$$

Now fix $j = 2$. Then if i is defined by $v_i = 2$, we see that i is uniform over $\{2, 3, \ldots, n\}$. So

$$\mathbb{E}X_{1,2} = \frac{1}{n-1}\sum_{i=2}^{n}\sum_{k=1}^{i-1}\frac{1}{k(n-k)} = \frac{1}{n-1}\sum_{k=1}^{n-1}\frac{n-k}{k(n-k)} = \frac{1}{n-1}\sum_{k=1}^{n-1}\frac{1}{k} = \frac{\log n}{n} + O(n^{-1}).$$

For the variance of $X_{1,2}$ we have

$$X_{1,2} = \delta_2 Y_2 + \delta_3 Y_3 + \cdots + \delta_n Y_n,$$

where

$$\delta_i \in \{0,1\}; \quad \delta_2 + \delta_3 + \cdots + \delta_n = 1; \quad \mathbb{P}(\delta_i = 1) = \frac{1}{n-1}.$$

$$\mathbb{V}arX_{1,2} = \sum_{i=2}^{n} \mathbb{V}ar(\delta_i Y_i) + \sum_{i \neq j} Cov(\delta_i Y_i, \delta_j Y_j) \leq \sum_{i=2}^{n} \mathbb{V}ar(\delta_i Y_i).$$

The last inequality holds since

$$Cov(\delta_i Y_i, \delta_j Y_j) = \mathbb{E}(\delta_i Y_i \delta_j Y_j) - \mathbb{E}(\delta_i Y_i)\mathbb{E}(\delta_j Y_j) = -\mathbb{E}(\delta_i Y_i)\mathbb{E}(\delta_j Y_j) \leq 0.$$

So

$$\mathbb{V}arX_{1,2} \leq \sum_{i=2}^{n} \mathbb{V}ar(\delta_i Y_i) \leq \sum_{i=2}^{n} \frac{1}{n-1} \sum_{k=1}^{i-1} \left(\frac{1}{k(n-k)} \right)^2 = O(n^{-2}).$$

We can now use the Chebyshev inequality to finish the proof of (i) as we did for (ii). □

21.3 Set cover

Let $\mathbf{A}$ be an $m \times n$ matrix of 0's and 1's where $\mathbf{A}(i,j) = 1$ independently with probability p, where $0 < p < 1$ independent of m, n.

Lemma 59. *If $n \leq -(1-\varepsilon)\frac{\log m}{\log(1-p)}$ then w.h.p. there is no solution to $\mathbf{Ax} \geq \mathbf{1}_m$ where $x_j \in \{0,1\}$ for $j = 1, 2, \ldots, n$.*

Proof. Let Z denote the number of zero rows of $\mathbf{A}$. A given row is zero with probability q^n, and the rows are independent, so

$$Z \sim \mathrm{Bin}(m, q^n), \qquad \mathbb{E}Z = mq^n, \qquad \mathrm{Var}(Z) = mq^n(1-q^n) \leq \mathbb{E}Z.$$

If $n \leq (1-\epsilon)L$, then

$$\mathbb{E}Z = mq^n \geq mq^{(1-\epsilon)L} = m \exp\bigl\{-(1-\epsilon)\log m\bigr\} = m^{\epsilon}.$$

Consequently, by Chebyshev's inequality,

$$\mathbb{P}(Z = 0) \leq \mathbb{P}\bigl(|Z - \mathbb{E}Z| \geq \mathbb{E}Z\bigr) \leq \frac{\mathrm{Var}(Z)}{(\mathbb{E}Z)^2} \leq \frac{1}{\mathbb{E}Z} \leq m^{-\epsilon} = o(1).$$

Thus $\mathbf{A}$ has a zero row w.h.p. For such a row, the corresponding coordinate of $\mathbf{Ax}$ is zero for every $\mathbf{x} \in \{0,1\}^n$. Hence w.h.p. there is no solution of

$$\mathbf{Ax} \geq \mathbf{1}_m.$$

□

Lemma 60. *If $n \geq -(1+\varepsilon)\frac{\log m}{\log(1-p)}$ then putting $\mathbf{x} = \mathbf{1}^n$ does satisfy $\mathbf{Ax} \geq \mathbf{1}^m$ w.h.p.*

Proof. Again let Z be the number of zero rows. If $n \geq (1+\epsilon)L$, then

$$\mathbb{E}Z = mq^n \leq mq^{(1+\epsilon)L} = m \exp\bigl\{-(1+\epsilon)\log m\bigr\} = m^{-\epsilon} = o(1).$$

By Markov's inequality,

$$\mathbb{P}(Z \geq 1) \leq \mathbb{E}Z = o(1).$$

Therefore w.h.p. every row contains at least one 1. On taking $\mathbf{x} = \mathbf{1}_n$, the ith coordinate of $\mathbf{Ax}$ is the number of ones in row i, and hence

$$\mathbf{A}\mathbf{1}_n \geq \mathbf{1}_m$$

w.h.p. □

Lemma 61. *Assume now that the costs in the set cover problem $c_1, c_2, \ldots, c_n$ are independent uniform $[0,1]$ random variables. Then w.h.p. the optimal cost OPT satisfies*

$$OPT = (1 + o(1))\frac{L^2}{2n}, \text{ where } L = \frac{\log m}{\log(1/(1-p))}.$$

Proof. Let

$$C_{(1)} \leq C_{(2)} \leq \cdots \leq C_{(n)}$$

be the ordered costs, and put

$$S_k = \sum_{i=1}^{k} C_{(i)}.$$

We first record the following order-statistic estimate. For uniform $[0,1]$ order statistics,

$$\mathbb{E}C_{(i)} = \frac{i}{n+1}$$

and, for $i \leq j$,

$$\mathrm{Cov}(C_{(i)}, C_{(j)}) = \frac{i(n-j+1)}{(n+1)^2(n+2)}.$$

It follows that, whenever $k \to \infty$ and $k \leq n$,

$$\mathbb{E}S_k = \frac{k(k+1)}{2(n+1)}, \qquad \mathrm{Var}(S_k) = O\left(\frac{k^3}{n^2}\right),$$

and therefore, by Chebyshev's inequality,

$$S_k = (1 + o(1))\frac{k^2}{2n}, \qquad w.h.p.. \tag{1}$$

Upper bound. Let

$$k_+ = \left\lceil L + \sqrt{L} \right\rceil.$$

Because $n \geq (1+\epsilon)L$, we have $k_+ \leq n$ for all sufficiently large m. Select the k_+ columns with the smallest costs. Their indices are determined by the costs and are independent of $\mathbf{A}$, so the corresponding $m \times k_+$ submatrix is again an independent Bernoulli-p matrix. The probability that some row is uncovered is at most

$$mq^{k_+} \leq mq^{L+\sqrt{L}} = \exp\{-a\sqrt{L}\} = o(1).$$

Thus these columns form a cover w.h.p. By (1), their total cost w.h.p. satisfies

$$S_{k_+} = (1 + o(1))\frac{k_+^2}{2n} = (1 + o(1))\frac{L^2}{2n}.$$

Hence w.h.p.

$$OPT \le (1+o(1))\frac{L^2}{2n}. \tag{2}$$

Lower bound. Fix $\eta > 0$ and set

$$B = (1-\eta)\frac{L^2}{2n}.$$

Every column used by a solution of cost at most B must itself have cost at most B. Let

$$N_B = \big|\{j : C_j \le B\}\big|.$$

If $B < 1$, then

$$N_B \sim \mathrm{Bin}(n, B), \qquad \mathbb{E}N_B = nB = (1-\eta)\frac{L^2}{2},$$

so a Chernoff bound gives $N_B \le 2L^2$ w.h.p. If $B \ge 1$, then all costs are at most B, but in this case

$$n \le (1-\eta)\frac{L^2}{2},$$

so again $N_B = n \le 2L^2$. Consequently,

$$N_B \le 2L^2 \tag{3}$$

w.h.p.

Let

$$h = \left\lceil \frac{4}{a}\log L \right\rceil, \qquad k_- = \lfloor L - h \rfloor.$$

Condition on the costs and on the event (3). For any fixed set of k of these columns, the probability that it covers all m rows is

$$(1-q^k)^m \le \exp\{-mq^k\}.$$

For $k \le k_-$,

$$mq^k \ge mq^{k_-} \ge mq^{L-h} = e^{ah} \ge L^4.$$

There are at most

$$\sum_{k=0}^{k_-} \binom{N_B}{k} \le (k_-+1)(2L^2)^{k_-} = \exp\big\{O(L\log L)\big\}$$

such subsets. Hence, by the union bound, the probability that any subset of at most k_- columns of cost at most B covers all rows is at most

$$\exp\big\{O(L\log L) - L^4\big\} = o(1).$$

Thus a cover of total cost at most B, if one existed, would have to use more than k_- columns. Its cost would therefore be at least S_{k_-}. Since $k_- = L - O(\log L)$, (1) gives that w.h.p.

$$S_{k_-} = (1+o(1))\frac{L^2}{2n} > B$$

w.h.p., a contradiction. Therefore

$$\mathbb{P}\left(OPT < (1-\eta)\frac{L^2}{2n}\right) = o(1).$$

Since $\eta > 0$ was arbitrary, this and (2) yield that w.h.p.

$$OPT = (1+o(1))\frac{L^2}{2n} = (1+o(1))\frac{(\log m)^2}{2n\,(\log(1/(1-p)))^2}.$$

□

21.4 Quadratic assignment problem

The quadratic assignment problem is to

$$\text{Minimize} F(X) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{p=1}^{n}\sum_{q=1}^{n} a_{i,j,p,q}x_{i,p}x_{j,q}$$

Subject to

$$\sum_{j=1}^{n} x_{i,j} = 1 \text{ for } i = 1, 2, \ldots, n.$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \text{ for } j = 1, 2, \ldots, n.$$

$$x_{i,j} = 0 \text{ or } 1 \text{ for } i, j = 1, 2, \ldots, n.$$

Here $a_{i,j,p,q}$ is the cost of assigning entity i to position p and entity j to position q. For example, we could be designing a hospital, entities could be types of room and positions could be placements in the building and then "cost equals expected traffic".

Now suppose that the costs are independent uniform $[0, 1]$. Then for a *fixed* choice of $X = \{x_{i,j} : 1 \le i, j \le n\}$,

$$\mathbb{E}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{p=1}^{n}\sum_{q=1}^{n} a_{i,j,p,q}x_{i,p}x_{j,q}\right) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{p=1}^{n}\sum_{q=1}^{n} x_{i,p}x_{j,q} = \frac{n^2}{2}.$$

Applying (182) we see that for any $\varepsilon > 0$,

$$\mathbb{P}\left(\exists X : \left|F(X) - \frac{n^2}{2}\right| \ge \varepsilon n^2\right) \le 2n!e^{-\varepsilon^2 n^2/6} = o(1).$$

Thus, in this random model, any choice of X will w.h.p. be close to the optimum, in ratio.

22 Smoothed Analysis

Smoothed analysis, introduced by Daniel Spielman and Shang-Hua Teng in 2001, measures the performance of an algorithm under a worst-case input that is subsequently perturbed by a small amount of random noise. This framework bridges the gap between pessimistic worst-case analysis and overly optimistic average-case analysis.

22.1 0-1 Knapsack problem

In the knapsack problem, there is a set U of n items, where every item i has positive integer weight w_i and positive profit p_i. In addition there is a knapsack of capacity W. The goal is to select a set S of items maximizing the total profit $P = \sum_{i\in S} p_i$, subject to not violating the capacity constraint $\sum_{i\in S} w_i \le W$. If 2 sets have the same profit, we prefer the set of lower weight.

The problem is NP-hard and we have seen that it can be efficiently approximated. In addition it can be solved optimally in time $poly(n, 1/\delta)$ with probability at least $1 - \delta$ under the following assumptions: Consider a

smoothed version of the knapsack problem. An adversary first selects an arbitrary instance and then weights of items are perturbed at random. For simplicity and concreteness, we consider the following perturbation, parametrized by $\delta > 0$. For each adversarial weight $\tilde{w}_i$, we select r_i uniformly at random in the range $[0, \delta W]$, and the perturbed weight becomes $w_i = \tilde{w}_i + r_i$. Our claim rests on the following two lemmas.

Lemma 62. *Let $W^* \leq W$ denote the weight of an optimal solution to the knapsack problem and let P^* denote its profit. Suppose that for a given input instance it is known that for some explicit $\varepsilon > 0$, no solution of weight in the range $(W^*, W^* + \varepsilon W]$ has profit at least P^*. Then the input instance can be solved in time polynomial in $(n, 1/\varepsilon)$.*

Lemma 63. *With probability at least $1 - \varepsilon n/\delta$, there is no solution of weight in the range $(W^*, W^* + \varepsilon W]$ that has profit at least P^*.*

So, given knowledge of δ, we select $\varepsilon = \eta\delta/n$ and then in time $poly(n, 1/\eta)$ we can with probability at least $1 - \eta$, solve the instance exactly.

22.1.1 Proof of Lemma 62

We run two algorithms, at least one of which will produce the optimal solution. The first algorithm assumes that $W^* > (1 - \varepsilon/2))W$. Modify the weights as follows. Let $k = \varepsilon W/2n$. Round down each weight w_i to the nearest multiple of k and round down W the nearest multiple of k, giving new weights $\widehat{w}_i$, $\widehat{W}$. Observe that with the new weights, the original optimal solution S^* remains both feasible and optimal. Indeed, let $\widehat{w}_i = \lfloor w_i/k \rfloor k$ and let $\widehat{W} = \lfloor W/k \rfloor k$. Then

$$\widehat{w}(S^*) \leq w(S^*) \leq W^* \leq W. \tag{170}$$

Because the LHS of (170) is a multiple of k, we see that in fact that it is at most $\widehat{W}$. Thus S^* is feasible for the modified problem and its profit it at least as good any S for which $w(S) \leq W^*$. On the other hand, if $w(S) > W^* + \varepsilon W$ then

$$\widehat{w}(S) > w(S) - kn = w(S) - \frac{\varepsilon W}{2} > W^* + \frac{\varepsilon W}{2} > W \geq \widehat{W},$$

and S is not feasible for the modified problem. So, to solve the original problem, we only have to solve the modified problem. Now the dynamic programming table has size $n\lfloor W/k \rfloor = O(n^2/\varepsilon)$, and so the problem instance can be solved in time polynomial in $n, 1/\varepsilon)$.

The second algorithm assumes that $W^* \leq (1 - \varepsilon/2)W$. In this case each weight w_i is rounded up to the nearest multiple of k and W is still rounded down. The analysis proceeds as for the first algorithm: $\widehat{w}_i = \lceil w_i/k \rceil k$ and let $\widehat{W} = \lfloor W/k \rfloor k$. So now we have

$$\widehat{w}(S^*) < w(S^*) + kn = W^* + kn \leq W^* + \frac{\varepsilon W}{2} \leq W. \tag{171}$$

Again because the LHS of (171) is a multiple of k, we see that in fact that it is at most $\widehat{W}$ and so S^* is feasible for the modified problem. And if $\widehat{w}(S) \leq \widehat{W}$ then

$$w(S) \leq \widehat{w}(S) \leq \widehat{W} \leq W, \tag{172}$$

so S is feasible for the original problem. Consequently $p(S) \leq p(S^*)$, and hence S^* solves the modified problem.

Let S^- be the solution returned by the first algorithm and S^+ the solution returned by the second algorithm. The set S^+ is always feasible for the original problem, see (172). At least one of S^-, ,S^+ is an optimal original solution. Thus discard S^- if $w(S^-) > W$, and return the feasible candidate having the larger profit.

22.1.2 Proof of Lemma 63

If $w([n]) \leq W^*$ then there is nothing to prove. Hence we can assume that $w([n]) > W^*$ and so $S^* \neq [n]$. Note also that with probability one, different sets have different weights.

For $i \in [n]$ let $U_i = 2^{[n]\setminus\{i\}}$ and let P_i denote the maximum profit from a feasible set in U_i and let S_i be a minimum weight corresponding solution. Next let $W_i \leq W$ denote the minimum weight of a feasible set in U_i that has profit at least $P_i - p_i$ and let T_i denote the corresponding solution. Observe that if $i \notin S^*$ then $S^* \in U_i$ and so $S_i = S^*$ and

$$W_i + w_i \geq W^*. \tag{173}$$

Inequality (173) follows from tie breaking rule. Assume that $i \notin S^*$.

Suppose now that there exists a set S with $w(S) \in (W^*, W^* + \varepsilon W]$, that has profit at least P^*. And let $\widehat{S}$ be the set of least weight amongst such. There must be an $i \in \widehat{S} \setminus S^*$ else $w(S^*) \geq w(\widehat{S}) > W^*$. Note that

$$W_i + w_i \leq w(\widehat{S}). \tag{174}$$

If $\widehat{S} \setminus \{i\}$ is feasible then this follows from the choice of T_i, since $\widehat{S} \setminus \{i\}$ is a candidate for T_i ($p(\widehat{S} \setminus \{i\}) \geq P^* - p_i = P_i - p_i$). If $\widehat{S} \setminus \{i\}$ is infeasible then $w(\widehat{S} \setminus \{i\}) > W \geq W_i$.

Let $V_i = w(S_i)$ and note that $V_i = W^*$, assuming that $i \notin S^*$. Thus, inequalities (173) and (174) imply that

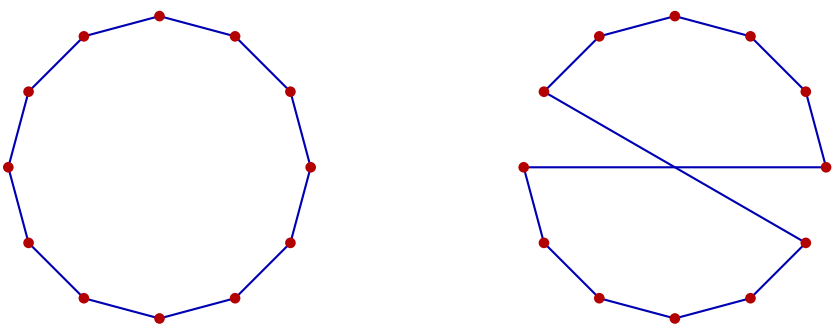
$$V_i \leq W_i + w_i \leq w(\widehat{S}) \leq V_i + \varepsilon W. \tag{175}$$

Fix all weights and perturbations except for r_i and let $\mathcal{E}_i$ be the event in (175). Then from (175), $r_i \in [V_i - (W_i + \tilde{w}_i), V_i + \varepsilon W^* - (W_i + \tilde{w}_i)]$ and this interval is independent of r_i and so $\mathbb{P}((175)) \leq \varepsilon W/\delta W$ and the lemma follows from a union bound over $\mathcal{E}_i$ for $1 \leq i \leq n$.

22.2 2-OPT for TSP

The 2-OPT heuristic for the TSP can be described as follows: we are given costs $C(i,j)$ for the edge $\{i,j\}$. A *tour* corresponds to a permutation π of the set $[n]$. The cost of a tour $C(\pi) = \sum_{i=1}^{n} C(\pi(i), \pi(i+1))$ where $\pi(n+1) = \pi(1)$. The *2-swap(i,j)*(i,j,π) where $i < j$ replaces π by

$$\pi(1), \ldots, \pi(i), \pi(j), \pi(j-1), \ldots, \pi(i+1), \pi(j+1), \ldots, \pi(j+1), \ldots, \pi(n).$$



The 2-OPT algorithm performs 2-swaps so long as there is an i, j such that $C(2swap(i,j,\pi)) < C(\pi)$, in which case it will replace π by $2swap(i,j,\pi)$. The costs are defined as follows: an adversary chooses an

initial $C, C(i,j) \le 1$ and then nature adds a random value $\varepsilon(i,j)$ to $C(i,j)$ where $\varepsilon(i,j)$ is uniform in $[-\varepsilon, \varepsilon]$. Starting at an arbitrary π, bound the expected number of steps until the algorithm terminates, in terms of n and ε.

Q1 Let the perturbed cost of an edge e be

$$c_e = C_e + \xi_e, \qquad \xi_e \sim \mathrm{Unif}[-\varepsilon, \varepsilon],$$

where the perturbations are independent. We use the standard normalization $0 \le C_e \le 1$ and assume $0 < \varepsilon \le 1$. With only the stated upper bound $C_e \le 1$ and no lower bound, no bound depending only on n and ε follows from the objective-range argument below.

A proper 2-OPT move deletes two disjoint tour edges, say $\{a, b\}$ and $\{c, d\}$, and inserts one of the alternative pairings, say $\{a, c\}$ and $\{b, d\}$. The decrease in tour cost is

$$D = c_{ab} + c_{cd} - c_{ac} - c_{bd}.$$

The move is improving when $D > 0$.

There are at most n^4 distinct expressions of this form. Fix one of them and condition on all perturbations except, for example, ξ_{ab}. The random variable D is then a translate of a uniform random variable on an interval of length 2ε. Hence, for every $\delta > 0$,

$$\mathbb{P}(|D| \le \delta) \le \frac{\delta}{\varepsilon}.$$

Let

$$\Delta = \min\{|D| : D \neq 0\},$$

where the minimum is taken over all possible 2-OPT exchanges. Since the perturbations have continuous distributions, $\Delta > 0$ with probability one. By the union bound,

$$\mathbb{P}(\Delta \le \delta) \le \frac{n^4 \delta}{\varepsilon}. \tag{176}$$

Every perturbed edge cost lies in $[-\varepsilon, 1 + \varepsilon]$. Therefore all tour costs lie in an interval of length at most

$$B = n(1 + 2\varepsilon).$$

Let T be the number of 2-OPT steps. If $T \ge t$, then among the first t improvements at least one has size at most B/t; otherwise the total decrease would exceed B. Thus

$$\{T \ge t\} \subseteq \left\{\Delta \le \frac{B}{t}\right\},$$

and (176) gives

$$\mathbb{P}(T \ge t) \le \min\left\{1, \frac{n^4 B}{\varepsilon t}\right\}. \tag{177}$$

A strict improvement never revisits a tour, so deterministically $T \le n!$. Put

$$K = \frac{n^4 B}{\varepsilon}.$$

Using the tail-sum formula and (177),

$$\mathbb{E}T = \sum_{t=1}^{n!} \mathbb{P}(T \geq t) \leq 1 + K + K \sum_{t=\lceil K \rceil + 1}^{n!} \frac{1}{t} \leq 1 + K\big(2 + \log(n!)\big).$$

Since $B = n(1+2\varepsilon)$ and $\log(n!) = O(n \log n)$,

$$\mathbb{E}T = O\left(\frac{n^6(1+\varepsilon)\log n}{\varepsilon}\right).$$

In particular, when $0 < \varepsilon \leq 1$,

$$\mathbb{E}T = O\left(\frac{n^6 \log n}{\varepsilon}\right).$$

The estimate holds for any rule that chooses an improving 2-OPT move.

Q2 Use a constraint-generation, or row-generation, version of the simplex algorithm.

Start with a manageable family $\mathcal{C}$ of cut constraints and solve

$$\begin{aligned} \text{minimize} \quad & \sum_{1 \leq i < j \leq n} C(i,j) x_{ij}, & \\ \text{subject to} \quad & \sum_{i \in S,\, j \notin S} x_{ij} \geq 1, & S \in \mathcal{C}, \\ & 0 \leq x_{ij} \leq 1, & 1 \leq i < j \leq n, \end{aligned}$$

by the simplex algorithm. Let x be the resulting optimal solution. Regard x_{ij} as the capacity of edge $\{i, j\}$ in the complete undirected graph and compute

$$\lambda(x) = \min_{\emptyset \neq S \subsetneq [n]} \sum_{i \in S,\, j \notin S} x_{ij}.$$

This is a global minimum-cut problem and can be solved in polynomial time.

If $\lambda(x) < 1$, let S be a minimum cut attaining this value. Then

$$\sum_{i \in S,\, j \notin S} x_{ij} < 1,$$

so the corresponding constraint is violated. Add this row to $\mathcal{C}$ and reoptimize. Since the previous basis is available, the reoptimization can conveniently be carried out by the dual simplex method.

If $\lambda(x) \geq 1$, then every cut constraint is satisfied, so x is feasible for the full LP. It is also optimal for the full LP.

23 Probabilistic Inequalities

23.1 Markov and Chebyshev inequalities

Let X be a non-negative random variable with a finite mean. Then,

Lemma 64 (Markov Inequality)**.** *For any $t > 0$, we have $\mathbb{P}(X \geq t) \leq \mathbb{E}X/t$.*

Proof.
$$\mathbb{E}X = \mathbb{E}(X \mid X < t)\mathbb{P}(X < t) + \mathbb{E}(X \mid X \geq t)\mathbb{P}(X \geq t) \geq t\mathbb{P}(X \geq t).$$

□

Corollary 65 (First moment method)**.** *Let X be a non-negative integer random variable. Then $\mathbb{P}(X \neq 0) \leq \mathbb{E}X$.*

Proof. Use the Markov inequality with $t = 1$. □

Theorem 66 (Chebyshev inequality)**.** *Let X be a random variable with variance σ^2. Then for any $t > 0$,*

$$\mathbb{P}(|X - \mathbb{E}X| \geq t) \leq \frac{\sigma^2}{t^2}.$$

Proof. $\mathbb{P}(|X - \mathbb{E}X| \geq t) = \mathbb{P}((X - \mathbb{E}X)^2 \geq t^2)$. Now apply the Markov inequality to $Z = (X - \mathbb{E}X)^2$. □

23.2 Sums of independent bounded random variables

Suppose that $S_n = X_1 + X_2 + \cdots + X_n$ where $X_i, i = 1, \ldots, n$ are independent random variables. We will be concerned here with bounds on the *upper and lower tail* of the distribution of S_n. Assume that $0 \leq X_i \leq 1$ and $\mathbb{E}X_i = \mu_i$ for $i = 1, 2, \ldots, n$. Let $\mu = (\mu_1 + \mu_2 + \cdots + \mu_n)/n$.

Let $\lambda \geq 0$, then by the Markov inequality,

$$\begin{aligned}\mathbb{P}(S_n \geq \mu n + t) &= \mathbb{P}(e^{\lambda S_n} \geq e^{\lambda(\mu n + t)}) \leq e^{-\lambda(\mu n + t)}\mathbb{E}(e^{\lambda S_n}) \\ &= e^{-\lambda(\mu n + t)} \prod_{i=1}^{n} \mathbb{E}(e^{\lambda X_i}). \end{aligned} \tag{178}$$

Now the convexity of e^x and $0 \leq X_i \leq 1$ implies that

$$e^{\lambda X_i} \leq 1 - X_i + X_i e^{\lambda}.$$

Taking expectations we get

$$\mathbb{E}(e^{\lambda X_i}) \leq 1 - \mu_i + \mu_i e^{\lambda}.$$

Equation (178) becomes, for $\lambda \geq 0$,

$$\mathbb{P}(S_n \geq \mu n + t) \leq e^{-\lambda(\mu n + t)} \prod_{i=1}^{n} (1 - \mu_i + \mu_i e^{\lambda}) \leq e^{-\lambda(\mu n + t)}(1 - \mu + \mu e^{\lambda})^n. \tag{179}$$

The second inequality follows from the fact that the geometric mean is at most the arithmetic mean i.e. $(x_1x_2\cdots x_n)^{1/n} \le (x_1+x_2+\cdots+x_n)/n$ for non-negative $x_1, x_2, \ldots, x_n$.

The right hand side of (179) attains its minimum, as a function of λ, at

$$e^{\lambda} = \frac{(\mu n + t)(1-\mu)}{(n(1-\mu)-t)\mu}. \tag{180}$$

Hence, if $t \le n(1-\mu)$, then

$$\mathbb{P}(S_n \ge \mu n + t) \le \left(\frac{\mu n}{\mu n + t}\right)^{\mu n + t} \left(\frac{n(1-\mu)}{n(1-\mu)-t}\right)^{n(1-\mu)-t}, \tag{181}$$

while for $t > n(1-\mu)$ this probability is zero.

Putting $t = \varepsilon\mu n$, $0 \le \varepsilon \le 1$ we obtain

$$\begin{aligned}
\mathbb{P}(S_n \ge \mu n(1+\varepsilon)) &\le \left(\frac{1}{1+\varepsilon}\right)^{(1+\varepsilon)\mu n} \left(\frac{1-\mu}{1-\mu(1+\varepsilon)}\right)^{n(1-\mu(1+\varepsilon))} \\
&= \left(\frac{1}{1+\varepsilon}\right)^{(1+\varepsilon)\mu n} \left(1 + \frac{\varepsilon\mu}{1-\mu(1+\varepsilon)}\right)^{n(1-\mu(1+\varepsilon))} \\
&\le \left(\frac{1}{1+\varepsilon}\right)^{(1+\varepsilon)\mu n} e^{\varepsilon\mu n} \\
&\le e^{-\varepsilon^2\mu n/3}.
\end{aligned} \tag{182}$$

Similarly for $0 \le \varepsilon \le 1$,

$$\mathbb{P}(S_n \le \mu n(1-\varepsilon)) \le e^{-\varepsilon^2\mu n/2}. \tag{183}$$

We follow that same strategy, but begin with

$$\mathbb{P}(S_n \le \mu n - t) = \mathbb{P}(e^{\lambda(\mu n - S_n)} \ge e^{\lambda t}) \le e^{\lambda t}\mathbb{E}(e^{\lambda(\mu n - S_n)}).$$

If $\alpha > 1$ then we get from (181) that

$$\mathbb{P}(S_n \ge \alpha\mu n) \le \left(\frac{\mu}{\alpha}\right)^{\alpha\mu n} \left(1 + \frac{(\alpha-1)\mu}{1-\alpha\mu}\right)^{(1-\alpha\mu)n} \le \left(\frac{e}{\alpha}\right)^{\alpha\mu n}. \tag{184}$$

23.3 McDiarmid's Inequality

In this section we will prove the following:

Theorem 67. *Let $Z = Z(Y_1, Y_2, \ldots, Y_n)$ be a random variable where $Y_1, Y_2, \ldots, Y_n$ are independent. Assume also that for any $i, Y_i, \widehat{Y}_i$ that*

$$|Z = Z(Y_1, Y_2, \ldots, Y_i, \ldots, Y_n) - Z(Y_1, Y_2, \ldots, \widehat{Y}_i, \ldots, Y_n)| \le c_i. \tag{185}$$

(I.e. changing component i, changes Z by at most c_i.)
Then,

$$\mathbb{P}(|Z - \mathbb{E}Z| \ge t) \le 2\exp\left\{-\frac{t^2}{2\sum_{i=1}^n c_i^2}\right\}.$$

(With some more work one can replace $t^2/2$ by $2t^2$ in the RHS of (185).)

Proof. We begin by proving the following claim:

Claim 3. *Let X be a random variable such that $|X| \le c$ and $\mathbb{E}X = 0$. Suppose that $\lambda \ge 0$. Then*

$$\mathbb{E}(e^{\lambda X}) \le e^{\lambda^2 c^2/2}.$$

Proof of Claim 3
We have by convexity that

$$e^{\lambda X} \le \frac{X+c}{2c} e^{\lambda c} + \frac{c-X}{2c} e^{-\lambda c}.$$

Taking expectations, and using $\mathbb{E}X = 0$,

$$\mathbb{E}(e^{\lambda X}) \le \frac{e^{\lambda c} + e^{-\lambda c}}{2} = \sum_{k=0}^{\infty} \frac{(\lambda c)^{2k}}{(2k)!} \le e^{\lambda^2 c^2/2}.$$

End of proof of Claim 3
Next let $\mathcal{Y}_i = (Y_1, \ldots, Y_i)$ and $Z_i = Z_i(\mathcal{Y}_i) = \mathbb{E}(Z \mid \mathcal{Y}_i)$ so that $Z_n = Z$ and $Z_0 = \mathbb{E}Z$. Now, for any $\lambda \ge 0$,

$$\begin{aligned}\mathbb{P}(Z - \mathbb{E}Z \ge t) &= \mathbb{P}(e^{\lambda(Z-\mathbb{E}Z)} \ge e^{\lambda t}) \le e^{-\lambda t}\, \mathbb{E}(e^{\lambda(Z-\mathbb{E}Z)}) = e^{-\lambda t}\, \mathbb{E}_{\mathcal{Y}_n}\left(\prod_{i=1}^{n} e^{\lambda(Z_i - Z_{i-1})}\right) \\ &= e^{-\lambda t}\, \mathbb{E}_{\mathcal{Y}_{n-1}}\left(\prod_{i=1}^{n} e^{\lambda(Z_i - Z_{i-1})}\, \mathbb{E}(e^{\lambda(Z_n - Z_{n-1})} \mid \mathcal{Y}_{n-1})\right).\end{aligned}$$

Now Y_n is independent of $\mathcal{Y}_{n-1}$ and given $\mathcal{Y}_{n-1}$, $X = Z_n - Z_{n-1} = Z_n - \mathbb{E}Z_n$ satisfies the conditions of Claim 3 with $c = c_n$. (If there exists $\mathcal{Y}_n$ such that $|X| > c_n$ then we would have $\mathbb{E}X \ne 0$, because of (185).) So,

$$\mathbb{P}(Z - \mathbb{E}Z \ge t) \le e^{-\lambda t + \lambda^2 c_n^2/2} \mathbb{E}_{\mathcal{Y}_{n-1}}\left(\prod_{i=1}^{n} e^{\lambda(Z_i - Z_{i-1})}\right) \le \exp\left\{-\lambda t + \lambda^2 \sum_{i=1}^{n} c_i^2/2\right\},$$

where the second inequality follows by induction on n. Putting $\lambda = t/\sum_{i=1}^{n} c_i^2$ proves the the upper bound on $\mathbb{P}(Z - \mathbb{E}Z \ge t)$. The proof of the bound on $\mathbb{P}(Z - \mathbb{E}Z \le -t)$ is similar and begins with

$$\mathbb{P}(Z - \mathbb{E}Z \le -t) = \mathbb{P}(e^{\lambda(\mathbb{E}Z - Z)} \ge e^{\lambda t}) \le e^{-\lambda t}\mathbb{E}(e^{\lambda(\mathbb{E}Z - Z)}).$$

□