Notes for OR1

1 Basic 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: $x = 1.4, y = 0.3, z = 7.6$. Cost is 3.62.

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 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 and have y_i tons in storage at the beginning of period i. Let $x_0 = y_1 = 0$.

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

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

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

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.

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

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

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

Transforming to standard form:

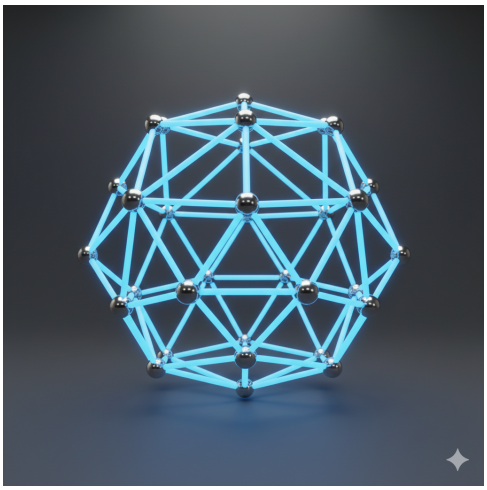
$$\begin{aligned}
\mathbf{a}^T\mathbf{x} \leq b &\longrightarrow \mathbf{a}^T\mathbf{x} + s = b.\\
\mathbf{a}^T\mathbf{x} \geq b &\longrightarrow \mathbf{a}^T\mathbf{x} - s = b.\\
x_j \leq 0 &\quad \text{Replace } x_j \text{ by } -x'_j,\ x'_j \geq 0.\\
x_j \text{ free} &\quad \text{Replace } x_j \text{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

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.)

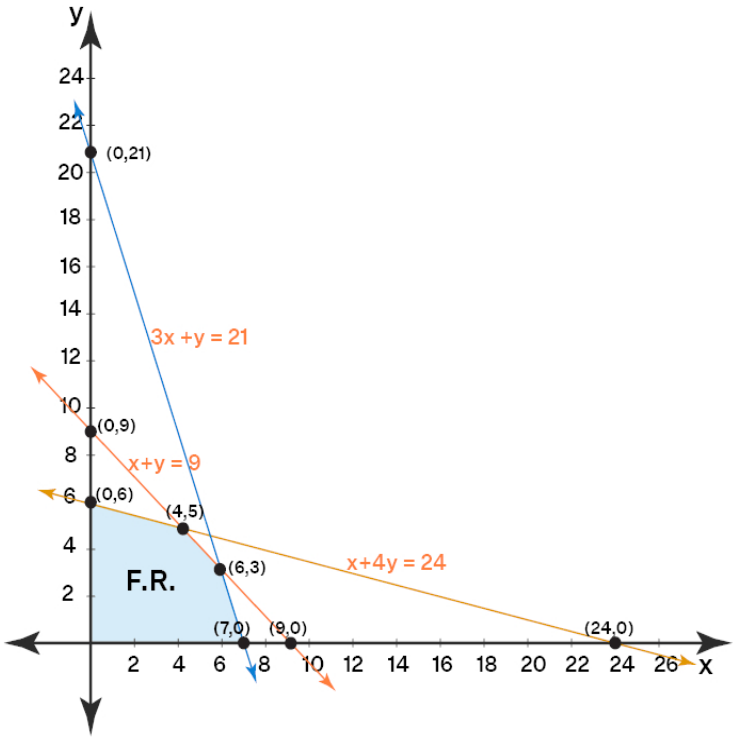


The "corners" are called *extreme points*. If there is an optimal solution, then there is an extreme point with optimal value.

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:

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

$B.V.$	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:

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

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

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

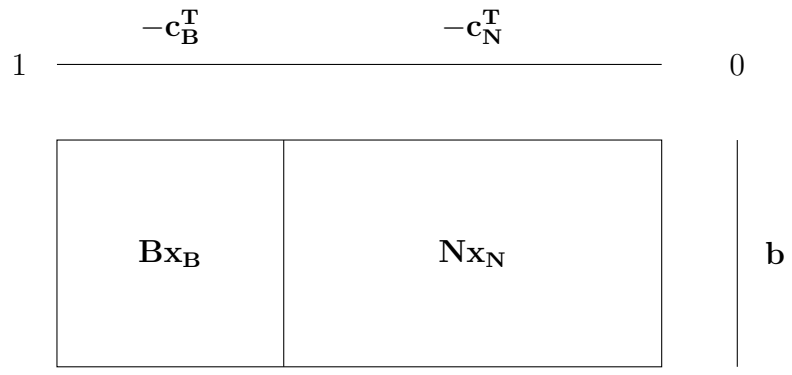
$$B = \begin{bmatrix} 0 & 1 & 4 \\ 1 & 3 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad B^{-1} = \begin{bmatrix} 2/3 & 1 & -11/3 \\ -1/3 & 0 & 4/3 \\ 1/3 & 0 & -1/3 \end{bmatrix}$$

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

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 $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2\, \cdots \mathbf{a}_n]$.

$$x_0 - \mathbf{c_B^T x_B} - \mathbf{c_N^T x_N} = 0.$$
$$\mathbf{Bx_B} + \mathbf{Nx_N} = \mathbf{b}.$$

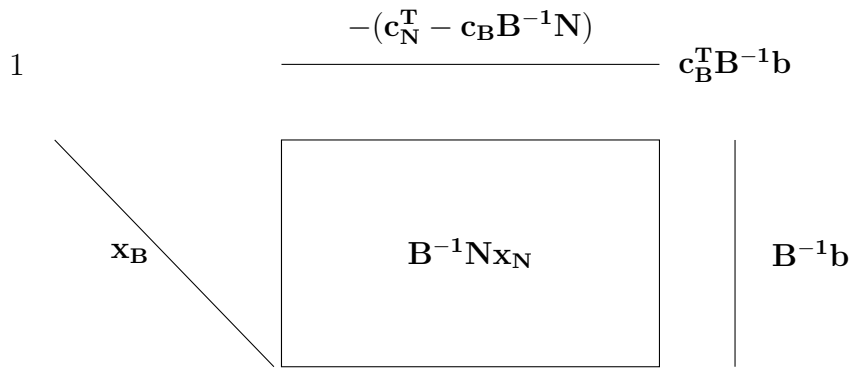


Multiply matrix $\mathbf{A}$ on the left by $\mathbf{B^{-1}}$ and substitute $\mathbf{x_B} = \mathbf{B^{-1}}(\mathbf{b} - \mathbf{Nx_N})$. To get an equivalent set of linear equations. Solutions to $\mathbf{Ax} = \mathbf{b}$ satisfy the following and vice-versa:

$$x_0 - (\mathbf{c_N} - \mathbf{c_B B^{-1} N})\mathbf{x_N} = \mathbf{c_B^T B^{-1} b}.$$
$$\mathbf{x_B} + \mathbf{B^{-1} N x_N} = \mathbf{B^{-1} b}.$$

The values $c_j - \mathbf{c_B B^{-1} a_j}$ are called the *reduced costs*.

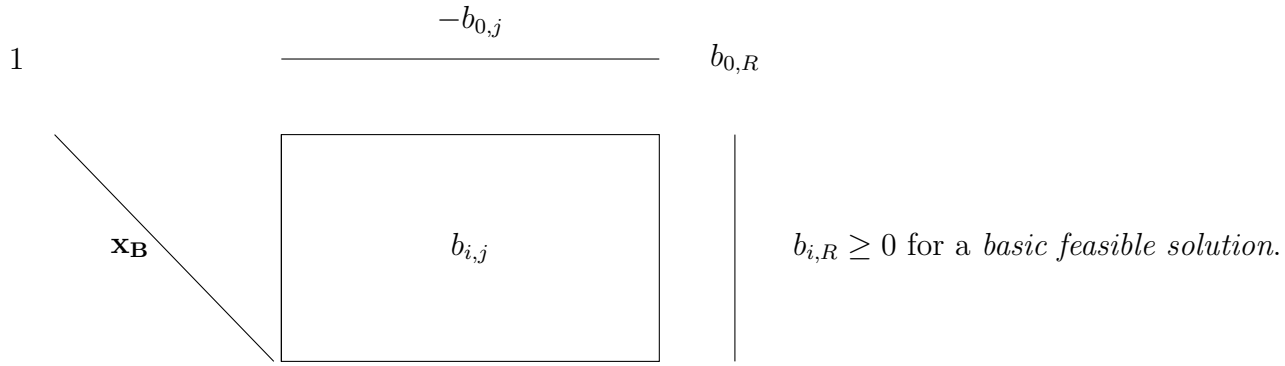
Simplex tableau



Basic Solution: I denotes the index set of the *basic* columns $\mathbf{B}$, (note that $0 \in I$). 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

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

The associated *basic solution* is obtained by putting put $\mathbf{x_N} = 0$ and $\mathbf{x_B} = \mathbf{B^{-1} b}$. This is *feasible* if $\mathbf{B^{-1} b} \geq \mathbf{0}$ 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 $\mathbf{x_N}$ and then putting $\mathbf{x_B} = \mathbf{B^{-1} b} - \mathbf{B^{-1} N x_N}$.



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 $\mathbf{y} \geq \mathbf{0}$ satisfies

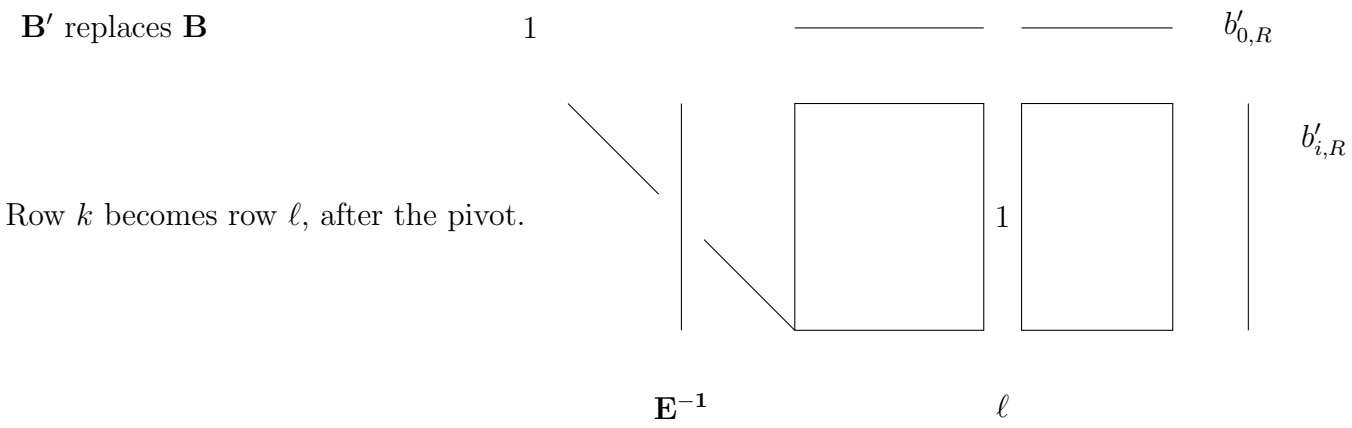
$$y_0 = b_{0,R} - \sum_{j \in J} b_{0,j} y_j \leq b_{0,R}.$$

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 basic variable from (4). Then for large t

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

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



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

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

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

Then

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

and

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

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 impotant 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: $b'_{\ell,R} = b_{k,R}/b_{\ell,k}$ so we need $b_{\ell,k} > 0$, assuming that $b_{k,R} > 0$.
We need $b'_{i,R} = b_{i,R} - b_{i,\ell} \times \frac{b_{k,R}}{b_{k,\ell}} \geq 0$. This is automatically true if $b_{i,\ell} \leq 0$.
Otherwise we need $\frac{b_{k,R}}{b_{k,\ell}} \leq \frac{b_{i,R}}{b_{i,\ell}}$ – *ratio test.*

Objective improvement – maximization: $b'_{0,R} = b_{0,R} - b_{0,\ell} \times \frac{b_{k,R}}{b_{k,\ell}}$ and so we choose $b_{0,\ell} < 0$. Note that $-b_{0,\ell}$ 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_ℓ, keeping $x_j = 0, j \in J \setminus \ell$ and satisfying (4). Then $x_i \leftarrow b_{i,R} - b_{i,\ell} x_\ell$ and so the maximum we can increase x_ℓ to is $\min\{b_{i,R}/b_{i,\ell} : b_{i,\ell} > 0\}$. The solution we get is the same as the basic solution after the pivot. We now have $x_j = 0$ for $j \in (J \setminus \{\ell\}) \cup \{k\}$ and $\mathbf{a}_i, i \in (I \cup \{\ell\}) \setminus \{k\}$ forms our new basis.

Degeneracy: A BFS is degenerate if $b_{k,R} = 0$ for some $k \in I$. If you pivot on $b_{k,\ell}$ in such a row, then the RHS $\mathbf{b}_R$ does not change. After the pivot we still $x_k = x_\ell = 0$ 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.
First solve

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

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.

Termination: If an LP problem is *non-degenerate* then for every vertex, we have $b_{i,R} > 0, i \in R$ for all $i \in I$. This means that $b'_{0,R} = b_{0,R} - b_{0,\ell} \times \frac{b_{k,R}}{b_{k,\ell}} < b_{0,R}$ 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 $K = \{i_1, i_2, \ldots, i_p\} = \{j_1, j_2, \ldots, j_p\}$. 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 K. Applying the Simplex algorithms to these tableaus will result in the same endless sequence of pivots. Let $t = \max K$ 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 $\mathbf{Ax} = \mathbf{b}$ 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 $i \in K$, 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

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

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, $\sum_{i \in K} b^2_{0,i} b^1_{i,s} > 0$ and there exists $k \in K$ such that $b^2_{0,k} b^1_{k,s} > 0$. Now $k \neq t$ 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 $b^2_{0,j} \geq 0, j \in K \setminus \{t\}$ (Bland's rule) and so we have $b^1_{0,s} < 0, b^1_{k,s} > 0$ implying that we should have selected x_k 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 a choiice of pivot column that gives an algorithm that runs in time poly$(m + n)$.

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 is

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

As an exercise confirm that the dual of

$$\begin{aligned} \text{Minimise} \quad & \mathbf{c_1^T x_1} + \mathbf{c_2^T x_2} + \mathbf{c_3^T x_3} \\ \text{Subject to} \quad & \mathbf{A}_{1,1}\mathbf{x}_1 + \mathbf{A}_{1,2}\mathbf{x}_2 + \mathbf{A}_{1,3}\mathbf{x}_3 = \mathbf{b}_1 \\ & \mathbf{A}_{2,1}\mathbf{x}_1 + \mathbf{A}_{2,2}\mathbf{x}_2 + \mathbf{A}_{2,3}\mathbf{x}_3 \geq \mathbf{b}_2 \\ & \mathbf{A}_{3,1}\mathbf{x}_1 + \mathbf{A}_{3,2}\mathbf{x}_2 + \mathbf{A}_{3,3}\mathbf{x}_3 \geq \mathbf{b}_3 \\ & \mathbf{x}_1 \geq 0,\ \mathbf{x}_2 \leq 0. \end{aligned}$$

is

$$\begin{aligned} \text{Maximise} \quad & \mathbf{b}_1^T\mathbf{y_1} + \mathbf{b_2^T y_2} + \mathbf{c_3^T y_3} \\ \text{Subject to} \quad & \mathbf{A}_{1,1}^T\mathbf{y_1} + \mathbf{A_{1,2}^T y_2} + \mathbf{A_{1,3}^T y_3} \leq \mathbf{c_1} \\ & \mathbf{A}_{2,1}^T\mathbf{y_1} + \mathbf{A_{2,2}^T y_2} + \mathbf{A_{2,3}^T y_3} \geq \mathbf{c_2} \\ & \mathbf{A}_{3,1}^T\mathbf{y_1} + \mathbf{A_{3,2}^T y_2} + \mathbf{A_{3,3}^T y_3} = \mathbf{c_3} \\ & \mathbf{y_2} \geq \mathbf{0},\ \mathbf{y_3} \leq \mathbf{0}. \end{aligned}$$

The relationship can be summarised: primal variables gives rise to dual constraint and vice-versa.
The primal constraint i is associated with dual variable $\mathbf{y_i}$ and the dual constraint i is associated with primal variable $\mathbf{x}_i$.
Equations give rise to free variables; in the context of primal minimisation, $\geq$constraints give rise to non-negative dual variables and $\leq$constraints give rise to non-positive dual variables.

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

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

So, if $\mathbf{x}_0$ is primal feasible and $\mathbf{y_0}$ is dual feasible and $\mathbf{c^T x_0} = \mathbf{b^T y_0}$ 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 $\mathbf{c^T x_0} = \mathbf{b^T y_0}$.
Let $\mathbf{B}$ be an optimal basis and let $\mathbf{y_0} = \mathbf{c_B^T B^{-1}}$. Then (i) $\mathbf{y_0}$ is dual feasible since the reduced costs $c_j - \mathbf{y_0^T a_j}$ are all non-positive; (ii) the objective value is $\mathbf{c_B^T B^{-1} b}$ for both primal and dual.

Shadow prices: The vector $\boldsymbol{\pi} = \mathbf{y_0} = \mathbf{c_B^T B^{-1}}$ is often called the vector of *shadow prices*. (It is also the solution to the dual problem.) Suppose that we perturb the RHS $\mathbf{b}$ to $\mathbf{b'} = \mathbf{b} + \boldsymbol{\delta}\mathbf{b}$. Suppose that $\mathbf{B}$ is the optimal basis. If $\mathbf{B^{-1}b' \geq 0}$ then the reduced costs have not changed and so $\mathbf{B}$ remains as the optimal basis. The new optimal value will be $\mathbf{c_B^T B^{-1} b'} = \mathbf{c_B^T B^{-1} b} + \mathbf{c_B^T B^{-1}}\boldsymbol{\delta} = \mathbf{b_{0,R}} + \sum_{\mathbf{i \in I}} \pi_\mathbf{i}\delta_\mathbf{i}$.

In the context of maximising the profit $\mathbf{c^T x}$ subject to $\mathbf{Ax} \leq \mathbf{b}$ 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 $\mathbf{c^T x}$ subject to $\mathbf{Ax} \geq \mathbf{b}$ 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.

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 artifical 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 $-(0 - \boldsymbol{\pi}^T\mathbf{e_i}) = \boldsymbol{\pi_i}$.

Sensitivity Analysis: This can refer to computing the maximum by which we can change b_k without changing the optimal basis. If $\mathbf{e_k} = [0, 0, \ldots, 1, \ldots, 0]^T$ is the vector which all 0's except in the kth position where it equals 1, then this amounts to finding the the interval for λ such that $\mathbf{B^{-1}}(\mathbf{b} + \lambda\mathbf{e_k}) \geq \mathbf{0}$ i.e. $b_{i,R} + \lambda\boldsymbol{\beta}_{i,k} \geq 0$ where $\boldsymbol{\beta}_{i,k}$ is the element in row i, column k of $\mathbf{B^{-1}}$.
It can also refer computing the maximum by which we can change c_k without changing the optimal basis. This amounts to finding the interval for λ such that $c_k + \lambda - \boldsymbol{\pi}^T\mathbf{a}_k \leq 0$ i.e. $\lambda \leq \boldsymbol{\pi}^T\mathbf{a}_k - c_k$.

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 (slack) then the corresponding 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,

$$\mathbf{c^T x_0} - \mathbf{b^T y_0} = \mathbf{x_0^T}(\mathbf{c} - \mathbf{A^T y_0}) = \mathbf{0}.$$

Thus strong duality implies complementary slackness holds for $\mathbf{x}_0, \mathbf{y_0}$ optimal.

For an LP in the form: Maximise $\mathbf{c^T x}$ subject to $\mathbf{Ax} \leq \mathbf{b}$, which has a dual Minimise $\mathbf{b}^T\mathbf{y}$ subject to $\mathbf{A}^T\mathbf{y} = \mathbf{c},\ \mathbf{y} \geq \mathbf{0}$, complementary slackness becomes $(\mathbf{b} - \mathbf{Ax}_0)\mathbf{y_0} = \mathbf{0}$ or $y_i > 0$ implies $\mathbf{r_i^T x} = b_i$ 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 then $\mathbf{c}$ is a non-negative linear combination of the normals $\mathbf{r_i}$ of the tight contstraints i such that $\mathbf{r_i^T x_0} = b_i$ i.e. $\mathbf{c}$ is in the *cone* generated by these normals.

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

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

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

Suppose (i) fails and so the LP in standard form with $\mathbf{c} = -\mathbf{1}$ is infeasible. Now $\mathbf{u} = \mathbf{0}$ 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: 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 $0 \leq b'_{0,k} = b_{0,k} - b_{0,\ell} \times \frac{b_{k,k}}{b_{k,\ell}} = -\frac{b_{0,\ell}}{b_{k,\ell}}$ or $b_{k,\ell} < 0$. Also need, for $j \in J \setminus \{\ell\}$, that $0 \leq b'_{0,j} = b_{0,j} - b_{0,\ell} \times \frac{b_{k,j}}{b_{k,\ell}}$. This is automatically true if $b_{k,j} \geq 0$.
Otherwise we need $\frac{b_{0,j}}{b_{k,j}} \geq \frac{b_{0,\ell}}{b_{k,\ell}}$ – dual ratio test.
Improve the dual objective: this means reduce $b_{0,R}$. Now $b'_{0,R} = b_{0,R} - b_{0,\ell} \times \frac{b_{k,R}}{b_{k,\ell}}$ and so we want $b_{0,\ell} < 0$.

3 Convex Sets

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

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

See Diagram 3 at the end of these notes.

Examples of convex sets:

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

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

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

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

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

Operations on convex sets:

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

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

O3 Using induction we see that if $S_i, 1 \le i \le k$ 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 $\alpha S + \beta T = \{\alpha \mathbf{x} + \beta \mathbf{y}\}$ is convex.
If $\mathbf{z}_i = \alpha \mathbf{x}_i + \beta \mathbf{y_i} \in \mathbf{T}, \mathbf{i} = \mathbf{1}, \mathbf{2}$ then

$$\lambda \mathbf{z}_1 + (1-\lambda)\mathbf{z}_2 = \alpha(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) + \beta(\lambda \mathbf{y_1} + (\mathbf{1}-\lambda)\mathbf{y_2}) \in \mathbf{T}.$$

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

3.1 Extreme Points

A point $\mathbf{x}$ of a convex set S is said to be an *extreme point* if **THERE DO NOT EXIST** $\mathbf{y}, \mathbf{z} \in \mathbf{S}$ 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, then $ext(S) = \emptyset$.

Extreme points and BFS's Let $C = \{\mathbf{x} : \ \mathbf{Ax} = \mathbf{b}, \mathbf{x} \ge 0\}$ 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 $\lambda, 1-\lambda > 0, x_{1,j}, x_{2,j} \ge 0$). 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 $\mathbf{r_i}$ denote row i of $\mathbf{B^{-1}N}$. Suppose first that there exists $\boldsymbol{\xi} \in \mathbb{R}^N$ such that $\boldsymbol{\xi}^T \mathbf{r_i} = \mathbf{0}$ for $i \in I_0$. Then clearly $\mathbf{x} = (\mathbf{x}_1 + \mathbf{x}_2)/2$ where $\mathbf{x}_1 = \mathbf{x} - \varepsilon[0_\mathbf{B}, \boldsymbol{\xi}]$ and $\mathbf{x}_2 = \mathbf{x} + \varepsilon[0_\mathbf{B}, \boldsymbol{\xi}]$ 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 $|I_0| \ge n_1 = n - m$, 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 $\mathbf{r_i} \in \mathbf{I_1} \subseteq \mathbf{I_0}$. 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 Also, $\mathbf{z}$ is an extreme point of C iff there exists $\mathbf{c}$ such that $\mathbf{z}$ is the unioue maximiser of $\mathbf{c^T x}$ 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 $\mathbf{c^T z} \le \max\left\{\mathbf{c^T x_1}, \mathbf{c^T x_2}\right\}$. 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, $\mathbf{c^T z} = \mathbf{0}$ and $\mathbf{c^T x} = \sum_{j \in J} x_j < 0$ if $\mathbf{x} \neq \mathbf{z}$.

4 Primal-Dual Algorithms

Here is the general idea. Suppose we have the primal (3) and a solution $\mathbf{y}$ to the dual (5). Let $\mathbf{A_y}$ 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 $\mathbf{z} \ge 0$ such that $\mathbf{A_y z} = \mathbf{b}$ 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 $\mathbf{A}_{\mathbf{y}}^T\mathbf{u} \geq \mathbf{0}$ and $\mathbf{b}^T\mathbf{u} < \mathbf{0}$. We replace $\mathbf{y}$ by $\mathbf{y} + \varepsilon\mathbf{u}$. If $c_j = \mathbf{a}_j^T\mathbf{y}$ then $\mathbf{a}^T\mathbf{u} \geq \mathbf{0}$ and so $c_j \leq \mathbf{a}_j^T(\mathbf{y} + \varepsilon\mathbf{u})$ and if we ensure that ε is small enough that 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, $\mathbf{b}^T((\mathbf{y} + \varepsilon\mathbf{u}) < \mathbf{b}^{\mathbf{T}}\mathbf{y}$ and so the dual value has improved. In the following, we describe a specific LP for which this idea leads to a fast algorithm.

4.1 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 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 $M = M(\pi)$ be 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:

$$\begin{aligned}
&\text{Minimize} && \textstyle\sum_{i=1}^n \sum_{j=1}^n c_{i,j}x_{i,j} && && (6)\\
&\text{Subject to} && && &\\
& && \textstyle\sum_{j=1}^n x_{i,j} = 1 && \text{for } i = 1, 2, \ldots, n. && (7)\\
& && \textstyle\sum_{i=1}^n x_{i,j} = 1 && \text{for } j = 1, 2, \ldots, n. && (8)\\
& && x_{i,j} \geq 0 && \text{for } i, j = 1, 2, \ldots, n. && (9)
\end{aligned}$$

The assignment problem is the solution to ALP where we replace (9) by

$$x_{i,j} = 0 \text{ or } 1 \text{ for } i, j = 1, 2, \ldots, n. \tag{10}$$

This is because (7), (8) force the set $\{(i,j) : x_{i,j} = 1\}$ to be a perfect matching and (6) is then the cost of this matching.

In general replacing non-negativity constraints (9) by *integer* contraints (10) makes an LP hard to solve. Not however in this case.

The dual of ALP is the linear program DLP:

$$\begin{aligned}
&\text{Maximize} && \textstyle\sum_{i=1}^n u_i + \sum_{j=1}^n v_j && && (11)\\
&\text{Subject to} && && &\\
& && u_i + v_j \leq c(i,j) && \text{for } i, j = 1, 2, \ldots, n. && (12)
\end{aligned}$$

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

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

$$x_{i,j} > 0 \text{ implies that } u_i + v_j = c(i,j). \tag{13}$$

then **x** solves ALP and **u,v**, solves DLP. For then

$$0 = \sum_{i=1}^{n}\sum_{j=1}^{n}(c(i,j) - u_i - v_j)x_{i,j} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}x_{i,j} - \left(\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j\right), \tag{14}$$

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 (7) 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, **u,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 (13) holds and M provides a solution to the assignment problem.

Step 5 If M is not perfect, update **u,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 v. 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 **u,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. $u_i^* + v_j^* \leq u_i + v_j$ except for the case where $a_i \in A_M, b_j \notin B_M$ and θ is chosen so that the increase maintains feasiblity.

2. If $b \in B_M$ for the pair **u,v** then it will stay in B_M when we replace **u,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 **u,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 **u,v** in Step 5, the size of M increases by at least one. This is because updating **u,v** increases B_M by at least one. Thus the algorithm finishes in $O(n^4)$ time. ($O(n^3)$ time if done carefully.)

Total Unimodularity Notice that the above alogrithm solves the LP (6) and satisfies the integrality constraints (10). The reason for this is that all of the extreme points of the feasible region defined by (7), (8) and (9) are integral. The reason for this is that the constraint matrix of (6) 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 $\mathbf{B^{-1}b}$ is integer for every integer $\mathbf{b}$. Recall that $\mathbf{B^{-1}} = (adjoint\ \mathbf{B})/\mathbf{detB}$ 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 (6) has the following property 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 2 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 deteminant 0.

5 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.

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 of 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 midle. 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 knpowing 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

$$PAY(u, v_0) \leq PAY(u_0, v_0) \leq PAY(u_0, v) \tag{15}$$

holds for all u, v.

If (15) 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 indiceses 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 1.

(a) $P_A \leq P_B$.

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

Proof. (a)

$$P_A = ROWMIN(u_0) \leq PAY(u_0, v_0) \leq COLMAX(v_0) = P_B. \tag{16}$$

(b) Suppose first that (u_0, v_0) is stable. Then, from (15), 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 (16) we deduce that

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

which implies (15). □

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 (15).

For a game who's 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 34 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 $\boldsymbol{\pi} = (p_1, \ldots, p_m)$ 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

$$\begin{aligned} S_A &= \{p \in \Re^m : \boldsymbol{\pi} \geq 0 \text{ and } p_1 + \cdots + p_m = 1\}. \\ S_B &= \{q \in \Re^n : \mathbf{q} \geq 0 \text{ and } q_1 + \cdots + q_n = 1\}. \end{aligned} \tag{17}$$

We now show using the duality theory of linear programming that $S_A \times S_B$ as defined in (17) contains a stable solution.

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

$$P_A = \max_{\boldsymbol{\pi} \in S_A} \left(\min_{\mathbf{q} \in S_B} \sum_{j=1}^{n} c_j(\boldsymbol{\pi}) q_j \right) \tag{18}$$

Lemma 2.

$$\min_{\mathbf{q} \in S_B} \sum_{j=1}^{n} \xi_j q_j = \min \{\xi_1, \ldots, \xi_n\}. \tag{19}$$

Proof. Let $\xi_t = \min \{\xi_1, \ldots, \xi_n\}$ and let L be the LHS of (19). 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 (18) that

$$\begin{aligned} P_A &= \max_{\boldsymbol{\pi} \in S_A} \min \{c_1(\boldsymbol{\pi}), \ldots, c_n(\boldsymbol{\pi})\} \\ &= \max \min \{c_1(\boldsymbol{\pi}), \ldots, c_n(\boldsymbol{\pi})\} \\ &\text{Subject to} \\ &\quad p_1 + \cdots + p_m = 1 \\ &\quad p_1, \ldots, p_m \geq 0 \\ &= \max \ \xi \\ &\text{Subject to} \\ &\quad \xi \leq \sum_{i=1}^{m} a_{i,j} p_i, \qquad j = 1, \ldots, n \\ &\quad p_1 + \cdots + p_m = 1 \\ &\quad p_1, \ldots, p_m \geq 0 \end{aligned} \tag{20}$$

Using similar arguments we can show that

$$\begin{aligned} P_B = &\quad \min \ \eta \\ &\text{Subject to} \\ &\quad \eta \geq \sum_{j=1}^{n} a_{i,j} q_j, \qquad i = 1, \ldots, m \\ &\quad q_1 + \cdots + q_n = 1 \\ &\quad q_1, \ldots, q_n \geq 0 \end{aligned} \tag{21}$$

We note next that (20), (21) are a pair of dual linear programs. They are both feasible and hence $P_A = P_B$ and stable solutions exist. In fact if $\boldsymbol{\pi}^0$ solves (20) and $\mathbf{q}^0$ solves (21) $(\boldsymbol{\pi}^0, \mathbf{q}^0)$ is stable as $PAY(\boldsymbol{\pi}^0, \mathbf{q}^0) = P_A = P_B$.

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 who's 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 (20) is positive. We can therefore replace p_i by $x_i = p_i/\xi$ and then we find

$$P_A^{-1} = \text{Minimum } \xi^{-1} = \text{Minimum } \sum_{i=1}^{m} x_i \text{ subject to } \sum_{i=1}^{m} a_{i,j}x_i \geq 1 \text{ for all } j, \sum_{i=1}^{m} x_i = 1. \tag{22}$$

$$P_B^{-1} = \text{Maximum } \eta^{-1} = \text{Maximum } \sum_{j=1}^{n} y_j \text{ subject to } \sum_{j=1}^{n} a_{i,j}j_j \leq 1 \text{ for all } i, \sum_{j=1}^{n} y_j = 1. \tag{23}$$

5.1 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.

5.2 Latin Square Game

Suupose 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 (22), (23) that solve the game: putting $x_i = 1/C$ and $y_j = 1/R$ gives two feasible solutions with the same objective value.

5.3 Non-singular games

Suppose that A is non-singular and that $\mathbf{1^T A^{-1} 1} > \mathbf{0}$. Then the value of the game is $V = \frac{1}{\mathbf{1^T A^{-1} 1}}$. Then, $x^T = \frac{\mathbf{1^T A^{-1}}}{V}$ and $y = \frac{A^{-1}\mathbf{1}}{V}$ solve (22), (23) respectively.

5.4 Symmetric games

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

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$i = 1, , ...m$. 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 $S = \bigcup_{j \in I} S_i$. 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 specifiifed 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

$$\begin{aligned} \text{Minimize} & \sum_{j=1}^{n} c_j x_j \\ \text{Subject to} & \sum_{j=1}^{n} a_{i,j} x_j \geq 1, \qquad i = 1, 2, \ldots, m. \\ & x_j = 0 \text{ or } 1, j = 1, 2, \ldots, n. \end{aligned} \tag{24}$$

The m inequality constraints have the following significance: since $x_j = 0$ or 1 and the coefficients $_{i,j}$y 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 preblem. An airline has m scheduled flight-iegs 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 coverirg 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 (24) an equality – the set partitioning problem.

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

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

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 $c_1w_1 + e_2w_2 + c_3w_3$ where $c_1$in the cdost 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

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

Hardness Integer programming problems generally take much longer to solve then 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 $x1/2$ 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

$$\sum_{j=1}^{n} f_j x_j - f = \lfloor b \rfloor - \sum_{j=1}^{n} \lfloor a_j \rfloor x_j. \tag{25}$$

Now for $x \in S$, the RHS of (25) 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 $f_i = b_{i,j} - \lfloor b_{i,j} \rfloor$ for $j \notin I$ and $f = b_{i,0} - \lfloor b_{i,0} \rfloor$ we see that

$$\sum_{j \notin I} f_j x_j > f \tag{26}$$

for all integer solutions to our problem.

Now $f > 0$ since $b_{i,0}$ is not an integer and so (26) is not satisfied by the current L.P. solution since $x_j = 0$ for $j \notin I$ and so (26) is a cut.

The initial continuous problem solved by the algorithm is the L.P. problem obtained by ingoring 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 (26) 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:

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

$B.V.$	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

valve. It is interesting that finitness of the algorithm has not been proved for this strategy although finitness 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 $f : S_0 \to \Re$.

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 Person 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 assciated with two sets $I, O \subseteq [n]$. This 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 $\sum_{j \notin O} \max\{c_j, 0\}$. To test feasibility we check that $\sum_{j \notin O} \max\{a_{i,j}, 0\} \geq b_i, i \in [m]$. 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 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

$$\text{“does instance } I \in \mathcal{I} \text{ have property } \mathcal{P}\text{”}. \tag{27}$$

Each problem instance $I \in \mathcal{P}$ 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 (27) is in NP if there is a *witness* $w(I)$ 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 $\mathcal{I}_2$ if there 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 $\mathcal{I}_3$.

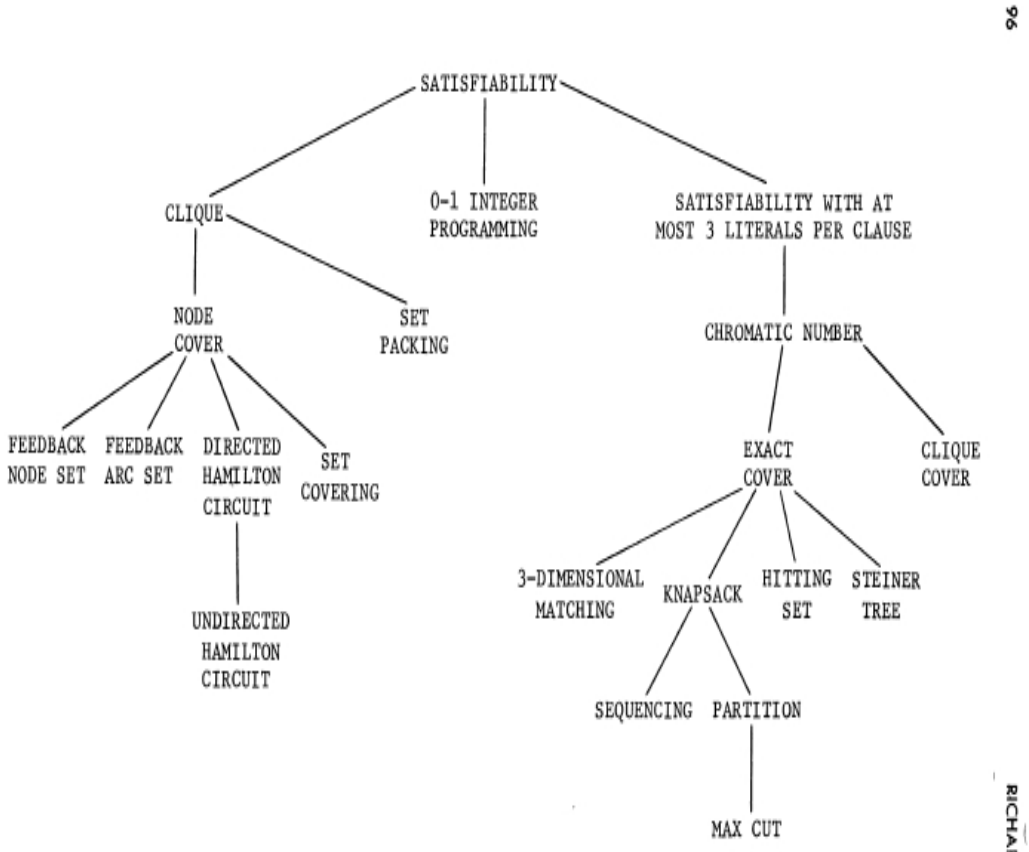


FIGURE 1 - Complete Problems

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 poynomial time algorithms for any NP-complete problems and that $NP \neq P$.

9 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^*.)

9.1 Traveling Salesperson Problem – TSP

9.1.1 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.

9.1.2 Triangle Inequality

We assume next that the costs $C(i,j), 1 \leq i,j \leq n$ satisfy the *triangular inequality* i.e. $C(i,j) + C(J,k) \geq C(i,k)$.

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 3. *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 $C(j_1, j_2) + C(j_j, j_3) + \cdots + C(j_{k-1}, j_k) \geq C(j_1, j_k)$. 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 $i_1, i_2, \ldots, i_{2n-2}$ through the edges of K.

Step 6 Shortcut the Euler tour until it is a Hamilton cycle H.

Theorem 4. *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. □

9.2 Knapsack problem

We consider the problem

$$\begin{aligned} \text{Maximize} \sum_{i=1}^{n} p_i x_i \\ \text{Subject to} \sum_{i=1}^{n} a_i x_i \leq B \\ x_i = 0/1, i = 1, 2, \ldots, n \end{aligned}$$

We will assume that $p_1/a_1 \geq p_2/a_2 \geq \cdots \geq p_n/a_n$.

9.2.1 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.

$$T \geq \frac{OPT}{2} \text{ or } p_i \geq \frac{B-A}{a_i} p_i \geq \frac{OPT}{2}.$$

9.2.2 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 5. *The solution produced in this way has value at least* $(1-\varepsilon)OPT$.

Proof. We first discuss the solution of the knapsack problem. Let

$$\begin{aligned} g_r(p) = &\text{minimum } a_1x_1 + \ldots + a_rx_r \\ &\text{Subject to } \widehat{p}_1x_1 + \cdots + \widehat{p}_rx_r \geq p \\ &\qquad x_i = 0/1 \text{ for } i \in [r]. \end{aligned} \tag{28}$$

Note that

$$g_r(p) = \min \begin{cases} g_{r-1}(p) & x_r = 0. \\ a_r + g_r(p - \widehat{p}_r) & x_r \geq 1. \end{cases} \tag{29}$$

We evaluate (29) for $p = 0, 1, \ldots, Nn$ and $r = 1, 2, \ldots, n$. This takes $O(N^2) = O(n^3/\varepsilon)$ time.

If we know $g_n(p), 0 \leq p \leq Nn$, then

$$OPT = g_n(p^*) \text{ where } g_n(p^*) \leq B \text{ and } g_n(p^*+1) > B.$$

We now verify the quality of the solution. Let $\widehat{S}$ define the solution to (28) and 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

$$\sum_{i \in \widehat{S}} p_i \geq \sum_{i \in \widehat{S}} \left\lfloor \frac{p_i N}{p_{\max}} \right\rfloor \frac{p_{\max}}{N} = \frac{p_{\max}}{N} \sum_{i \in \widehat{S}} \widehat{p}_i \geq \frac{p_{\max}}{N} \sum_{i \in S^*} \widehat{p}_i. \tag{30}$$

But,

$$\frac{p_{\max}}{N} \sum_{i \in S^*} \left\lfloor \frac{Np_i}{p_{\max}} \right\rfloor \geq \frac{p_{\max}}{N} \sum_{i \in S^*} \left(\frac{p_i N}{p_{\max}} - 1 \right) \geq \sum_{i \in S^*} p_i - \frac{np_{\max}}{N} \geq$$

$$\sum_{i \in S^*} p_i - \varepsilon p_{\max} \geq \sum_{i \in S^*} p_i - \varepsilon OPT = (1-\varepsilon)OPT.$$

□

9.3 Set cover

9.3.1 Primal-Dual algorithm

Recall the set cover problem of Section 6.1. The dual of the LP relaxation (24) is

$$\text{Maximise } \sum_{i=1}^{m} y_i. \tag{31}$$

$$\text{Subject to } \sum_{i=1}^{m} a_{i,j} y_i \le c_j, \qquad \text{j=1,2,...,n.} \tag{32}$$

$$y_j \ge 0, j = 1, 2, \ldots, n.$$

The constraints (32) can be re-expressed as

$$\sum_{i: j \in S_i} y_i \le c_j. \tag{33}$$

Consider the following algorithm:

Step 1 $\mathbf{y} \leftarrow \mathbf{0}$, $J \leftarrow \emptyset$.

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 6. *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 $c(\widehat{J}) \le f c(J^*)$.*

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 (33) and so it satisfies the dual constraints. Let $z_{LP} \le c(J^*)$ denote the value of the optimum solution to (24). Then,

$$\sum_{j \in \widehat{J}} c_j = \sum_{j \in \widehat{J}} \sum_{i \in S_j} \widehat{y}_j = \sum_{i=1}^{m} |\left\{ j \in \widehat{J} : i \in S_j \right\}| = \sum_{i=1}^{m} f_i \widehat{y}_i \le f \sum_{i=1}^{m} \widehat{y}_i \le f z_{LP} \le f c(J^*).$$

□

9.3.2 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 $H_k = \sum_{t=1}^{k} k^{-1}$ denote the *Harmonic number* k. Once again, let J^* define the optimum cover and let $\widehat{J}$ be produced by the above algorithm.

Theorem 7. *The above algorithm produces a set-cover* $\widehat{J}$ *and* $c(\widehat{J}) \leq H_n \cdot c(J^*)$.

Proof. Let n_k denote the number of uncovered elements at the start of the kth iteration. Now at the start of iteration k,

$$\min_{j:\widehat{S}_j \neq \emptyset} \frac{c_j}{|\widehat{S}_j|} \leq \frac{\sum_{j \in J^*} c_j}{\sum_{j \in J^*} |\widehat{S}_j|} = \frac{c(J^*)}{\sum_{j \in J^*} |\widehat{S}_j|} \leq \frac{c(J^*)}{n_k}. \tag{34}$$

Thus, if ℓ minimises the fraction on the LHS of eqrefg1, 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,

$$\sum_{\ell \in \widehat{J}} c_\ell \leq \sum_{k=1}^{K} \frac{(n_k - n_{k+1})c(J^*)}{n_k} \leq c(J^*) \sum_{k=1}^{K} \left(\frac{1}{n_k} + \frac{1}{n_k - 1} + \cdots + \frac{1}{n_{k+1} + 1} \right) = H_n \cdot c(J^*).$$

□

9.4 Submodular functions

A function $f : 2^X \to \mathbb{R}$ is *submodular* if for every $A, B \subseteq X$,

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B). \tag{35}$$

Note that if f, g satisfy (35), then so does $f + g$.

Example: Simple plant location problem Here f refers to profit:

$$f(S) = -\sum_{i \in S} f_i + \sum_{j=1}^{n} \max\{p_{i,j} : i \in S\}.$$

The constant function is clearly submodular 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 8. *If f is monotone increasing and submdular 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 $\Delta(v \mid T) = f(T \cup \{v\}) - f(v)$. 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

$$\Delta(v \mid S) \leq \Delta(v \mid T). \tag{36}$$

(The larger the set, the smaller the gain from v.)

Note that (36) is still true if $v \in S$, from monotonicity. Let $S_k^* = \{v_1^*, \ldots, v_k^*\}$.

$$\begin{aligned} f(S_k^*) &\leq f(S_k^* \cup S_i) \\ &= f(S_i) + \sum_{j=1}^{k} \Delta(v_j^* \mid S_i \cup \left\{v_1^*, \ldots, v_{j-1}^*\right\} \\ &\leq f(S_i) + \sum_{j=1}^{k} \Delta(v_j^* \mid S_i) \\ &\leq f(S_i) + \sum_{j=1}^{k} (f(S_{i+1}) - f(S_i)) \\ &= f(S_i) + k(f(S_{i+1}) - f(S_i)). \end{aligned}$$

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

$$f(S_k^*) - f(S_k) \leq \left(1 - \frac{1}{k}\right)^k (f(S_k^*) - f(\emptyset)) \leq e^{-1} f(S_k^*).$$

□

9.5 Local Search

This is a general approach to solving hard problems in Combinatorial Optimisation. Suppose that the problem is to

$$\text{Maximise } f(\mathbf{x}) \text{ Subject to } \mathbf{x} \in X. \tag{37}$$

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 $\max\{f(\mathbf{y}) : \mathbf{y} \in \mathbf{N_x}\}$ can be done efficiently. We can then find a good (not necesarily 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 $f(\mathbf{y}) : \mathbf{y} \in \mathbf{N_{x_{i-1}}}$. The value m will the smallest i such that $\mathbf{x}_i = \mathbf{x}_{i-1}$.

9.5.1 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 9. *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

$$\sum_{e \in \widehat{S}} \frac{N w(e)}{w_{\max}} + |E| \geq \frac{W^*}{2} \geq \frac{NW}{2 w_{\max}}.$$

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 $nW^* \leq 2n|E|/\varepsilon$.

10 Non-linear Optimization Problems

We consider the following problem:

$$\text{Minimize } f(\mathbf{x}) \text{ subject to } \mathbf{x} \in S, \tag{38}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $S \subseteq \Re^n$.

Example: $f(\mathbf{x}) = \mathbf{c^T x}$ and $S = \{\mathbf{x} \in \Re^n : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$ – Linear Programming.

Local versus Global Optima: $\mathbf{x}^*$ is a *global minimum* if it is an actual minimizer in (38).

$\mathbf{x}^*$ is a *local minimum* if there exists $\delta > 0$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in B(\mathbf{x}^*) \cap S$, where $B(\mathbf{x}, \delta) = \{\mathbf{y} : |\mathbf{y} - \mathbf{x}| \leq \delta\}$ is the *ball* of radius δ, centred at $\mathbf{x}$.

See Diagram 1 at the end of these notes.

If $S = \emptyset$ then we say that the problem is *unconstrained*, otherwise it is *constrained*.

11 Convex sets and functions

11.1 Convex Functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *convex* if

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda\mathbf{f}(\mathbf{x}) + (\mathbf{1}-\lambda)\mathbf{f}(\mathbf{y}).$$

See Diagram 2 at the end of these notes.

Examples of convex functions:

F1 A linear function $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x}$ is convex.

F2 If $n = 1$ then f is convex iff

$$f(y) \geq f(x) + f'(x)(y-x) \text{ for all } x, y. \tag{39}$$

Proof. Suppose first that f is convex. Then for $0 < \lambda \leq 1$,

$$f(x + \lambda(y-x)) \leq (1-\lambda)f(x) + \lambda f(y).$$

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 (39).

Now suppose that (39) 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 f is convex iff $f(\mathbf{y}) \geq \mathbf{f}(\mathbf{x}) + (\nabla \mathbf{f}(\mathbf{x}))^{\mathbf{T}}(\mathbf{y} - \mathbf{x})$ for all $\mathbf{x}, \mathbf{y}$.
Apply F2 to the function $h(t) = f(t\mathbf{x} + (1-t)\mathbf{y})$.

F4 A $n = 1$ 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

$$f(y) = f(x) + f'(x)(y-x) + \frac{1}{2}f''(z)(y-x)^2 \text{ where } z \in [x, y].$$

We now just apply (39). □

F5 It follows from F4 that e^{ax} is convex for any $a \in \mathbb{R}$.

F6 x^a is convex on $\mathbb{R}_+$ 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

$$\begin{aligned} & Q(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) - \lambda\mathbf{Q}(\mathbf{x}) - (\mathbf{1}-\lambda)\mathbf{Q}(\mathbf{y}) && (40)\\ =& \lambda^2 Q(\mathbf{x}) + (1-\lambda)^2 Q(\mathbf{y}) + \mathbf{2}\lambda(\mathbf{1}-\lambda)\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{y} - \lambda\mathbf{Q}(\mathbf{x}) - (\mathbf{1}-\lambda)\mathbf{Q}(\mathbf{y}) && (41)\\ =& -\lambda(1-\lambda)Q(\mathbf{y}-\mathbf{x}) \leq \mathbf{0}. && (42) \end{aligned}$$

F8 If $n \geq 1$ then f is convex iff $\nabla^2 F = \left[\frac{\partial f^2}{dx_i dx_j}\right]$ is positive semi-definite for all $\mathbf{x}$.
Apply F7 to the function $h(t) = f(\mathbf{x} + t\mathbf{d})$ for all $\mathbf{x}, \mathbf{d} \in \mathbb{R}^n$.

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.

$$\begin{aligned} h(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) &= \max\{f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}), \mathbf{g}(\lambda\mathbf{x} + (\mathbf{1}-\lambda)\mathbf{y})\} && (43)\\ &\leq \max\{\lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}), \lambda\mathbf{g}(\mathbf{x}) + (\mathbf{1}-\lambda)\mathbf{g}(\mathbf{y})\} && (44)\\ &\leq \lambda \max\{f(\mathbf{x}), g(\mathbf{x})\} + (1-\lambda)\max\{f(\mathbf{y}), \mathbf{g}(\mathbf{y})\} && (45)\\ &= \lambda h(\mathbf{x}) + (1-\lambda)h(\mathbf{y}). && (46) \end{aligned}$$

□

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

$$f\left(\sum_{i=1}^{m} \lambda_i \mathbf{a}_i\right) \leq \sum_{i=1}^{m} f(\lambda_i \mathbf{a}_i).$$

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

$$\frac{a_1 + a_2 + \cdots + a_m}{m} \geq (a_1 a_2 \cdots a_m)^{1/m}. \tag{47}$$

$-\log(x)$ is a convex function for $x \geq 0$. So, applying (47),

$$-\log\left(\sum_{i=1}^{m} \lambda_i \mathbf{a}_i\right) \leq \sum_{i=1}^{m} -\log(\lambda_i \mathbf{a}_i).$$

Now let $\lambda_i = 1/m$ for $i = 1, 2, \ldots, m$.

11.2 Convex Sets

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

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

See Diagram 3 at the end of these notes.

Examples of convex sets:

C1 $S = \left\{\mathbf{x} : \mathbf{a}^T \mathbf{x} = 1\right\}$. $\mathbf{x}, \mathbf{y} \in S$ 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 $S = \left\{\mathbf{x} : \mathbf{a}^T \mathbf{x} \leq 1\right\}$. 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 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 $\alpha S + \beta T = \{\alpha \mathbf{x} + \beta \mathbf{y}\}$ is convex.
If $\mathbf{z}_i = \alpha \mathbf{x}_i + \beta \mathbf{y}_i \in T, i = 1, 2$ then

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

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 $\mathbf{b}$.

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 10. *Let f, S both be convex in* (38)*. Then if $\mathbf{x}^*$ is a local minimum, it also a global minimum.*

Proof.

See Diagram 4 at the end of these notes.

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 11. *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, $\lambda \mathbf{x} + (1-\lambda)\mathbf{y} \in S$. □

12 Algorithms

12.1 Line search − $n = 1$

Here we consider the simpler problem of minimising a convex (more generally *unimodal*) function $f : \mathbb{R} \to \mathbb{R}$.

See Diagram 5 at the end of these notes.

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 $\zeta_i = f(2^i), \xi_i = f(-2^i)$ 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 $x^* \in [\zeta_0, \zeta_i]$ (resp. $x^* \in [\xi_0, \xi_i]$).

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 $f(a_0+)$ and make the changes $a_i \to a_i'$:

$$a_0' \leftarrow a_2,\ a_1' \leftarrow a_1,\ a_2' \leftarrow a_3,\ a_3' \leftarrow a_2 + \alpha^2(a_1 - a_0).$$

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, shrink by α in this case we choose α to satisfy $1 - \alpha^2 = \alpha$. This gives us

$$\alpha = \frac{\sqrt{5} - 1}{2} \text{ – } \textit{the golden ratio.}$$

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 $a_3' - a_0' = a_2 + \alpha^2(a_1 - a_0) - a_2 = \alpha^2(a_1 - a_0) = \alpha(a_1' - a_0')$.

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.

12.2 Gradient Descent

See Diagram 6 at the end of these notes.

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 algorithm will follow a sequence of points $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots$, where

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k).$$

Then we have

$$|\mathbf{x}_{k+1} - \mathbf{x}^*|^2 = |\mathbf{x}_k - \mathbf{x}^*|^2 - 2\alpha_k \nabla f(\mathbf{x}_k)^T(\mathbf{x}_k - \mathbf{x}^*) + \alpha_k^2 |\nabla f(\mathbf{x}_k)^2| \tag{48}$$

$$\leq |\mathbf{x}_k - \mathbf{x}^*|^2 - 2\alpha_k(f(\mathbf{x}_k) - f(\mathbf{x}^*)) + \alpha_k^2 |\nabla f(\mathbf{x}_k)|^2. \tag{49}$$

The inequality comes from F3.

Applying (49) repeatedly we get

$$|\mathbf{x}_k - \mathbf{x}^*|^2 \leq |\mathbf{x}_0 - \mathbf{x}^*|^2 - 2\sum_{i=1}^{k} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) + \sum_{i=1}^{K} \alpha_i^2 |\nabla f(\mathbf{x}_k)|^2. \tag{50}$$

Putting $R = |\mathbf{x}_0 - \mathbf{x}^*|$, we see from (50) that

$$2\sum_{i=1}^{k} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) \leq R^2 + \sum_{i=1}^{K} \alpha_i^2 |\nabla f(\mathbf{x}_k)|^2. \tag{51}$$

On the other hand,

$$\sum_{i=1}^{k} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) \geq \left(\sum_{i=1}^{k} \alpha_i\right) \min\{f(\mathbf{x}_k) - f(\mathbf{x}^*) : i \in [k]\} = \left(\sum_{i=1}^{k} \alpha_i\right) (f(\mathbf{x}_{min} - f(\mathbf{x}^*)), \tag{52}$$

where $f(\mathbf{x}_{min}) = \min\{f(\mathbf{x}_i) : i \in [k]\}$.

Combining (51) and (52) we get

$$f(\mathbf{x}_{min}) - f(\mathbf{x}^*) \leq \frac{R^2 + G^2 \sum_{i=1}^{k} \alpha_i^2}{2\sum_{i=1}^{k} \alpha_i}, \tag{53}$$

where $G = \max\{|\nabla f(\mathbf{x}_i)| : i \in [\kappa]\}$.

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

$$|f(\mathbf{x}_{min}) - f(\mathbf{x}^*)| \to 0 \text{ as } k \to \infty. \tag{54}$$

As an example, we could let $\alpha_i = 1/i$.

13 Separating Hyperplane

See Diagram 7 at the end of these notes.

Theorem 12. *Let C 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.

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 $\mathbf{x}_n, \notin \bar{C}, n \geq 1$ that tends to $\mathbf{x}$. For each $\mathbf{x}_n$, let $\mathbf{a}_n, b_n = \mathbf{a}_n^T \mathbf{z}_n$ define a hyperplane that separates $\mathbf{x}_n$ from $\bar{C}$, as in Case 2. We can assume that $|\mathbf{a}_n| = 1$ (scaling) and that b_n is in some bounded set and so there must be a convergent subsequence of $(\mathbf{a}_n, b_n), n \geq 1$ that converges to $(\mathbf{a}, b), |\mathbf{a}| = 1$. Assume that we re-label so that this subsequence is $(\mathbf{a}_n), n \geq 1$. Then for $\mathbf{y} \in \bar{C}$ we have $\mathbf{a}_n^T \mathbf{y} \leq b_n$ for all n. Taking limits we see that $\mathbf{a}^T \mathbf{y} \leq b$. Furthermore, for $\mathbf{y} \notin \bar{C}$ we see that for large enough n, $\mathbf{a}_n^T \mathbf{y} > b_n$. taking limits we see that $\mathbf{a}^T \mathbf{y} \leq b$. □

Corollary 13. *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 12 we see that there exists $\mathbf{a}$ such that $\mathbf{a}^T \mathbf{z} \leq 0$ for all $\mathbf{z} \in W$. Now put

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

□

Corollary 14 (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 $\mathbf{x} \geq \mathbf{0}, A\mathbf{x} = \mathbf{b}$.

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

Proof. We cannot have both (i), (ii) holding. For then we have

$$0 \leq \mathbf{u^T A x} = \mathbf{u^T b} < \mathbf{0}.$$

Suppose then that (i) fails to hold. Let $S = \{\mathbf{y} : \mathbf{y} = A\mathbf{x} \text{ for some } \mathbf{x} \geq \mathbf{0}\}$. Then $\mathbf{b} \notin S$ and since S is closed there exists $\boldsymbol{\alpha}, \beta$ such that (a) $\boldsymbol{\alpha}^T \mathbf{b} \leq \beta$ and (b) $\boldsymbol{\alpha}^T A\mathbf{x} \geq \beta$ for all $\mathbf{x} \geq 0$. This implies that $\boldsymbol{\alpha}^T(\mathbf{b} - A\mathbf{x}) \leq 0$ for all $\mathbf{x} \geq 0$. This then implies that $\mathbf{u} = \boldsymbol{\alpha}$ satisfies (ii). □

13.1 Convex Hulls

See Diagram 8 at the end of these notes.

Given a set $S \subseteq \mathbb{R}^n$, we let

$$conv(S) = \left\{\sum_{i \in I} \lambda_i \mathbf{x}_i : (i)\ |I| < \infty,\ (ii)\ \sum_{i \in I} \lambda_i = 1,\ (iii)\ \lambda_i > 0, i \in I,\ (iv)\ \mathbf{x}_i \in S, i \in I\right\}.$$

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

Lemma 15. *$conv(S)$ 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

$$\alpha\mathbf{x} + (1-\alpha)\mathbf{y} = \sum_{i \in K} (\alpha\lambda_1 + (1-\alpha)\mu_i)\mathbf{x}_i \text{ and } \sum_{i \in K} (\alpha\lambda_1 + (1-\alpha)\mu_i) = 1$$

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

Lemma 16. *If S is convex, then $S = conv(S)$.*

Proof. Exercise. □

Corollary 17. *$conv(conv(S)) = conv(S)$ for all $S \subseteq \mathbb{R}^n$.*

Proof. Exercise. □

13.1.1 Extreme Points

A point $\mathbf{x}$ of a convex set S is said to be an *extreme point* if **THERE DO NOT EXIST** $\mathbf{y}, \mathbf{z} \in S$ 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, then $ext(S) = \emptyset$.

Theorem 18. *Let S be a closed, bounded convex set. 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, $S \supseteq T = conv(ext(S))$ and suppose there exists $\mathbf{x} \in S \setminus T$. Let $\mathbf{z}$ be the closest point of T to $\mathbf{x}$ and let $H = \left\{\mathbf{y} : \mathbf{a}^T\mathbf{y} = b\right\}$ be the hyperplane defined in Theorem 12. Let $b^* = \max\left\{\mathbf{a}^T\mathbf{y} : \mathbf{y} \in S\right\}$. We have $b^* < \infty$ since S is bounded. Let $H^* = \left\{\mathbf{y} : \mathbf{a}^T\mathbf{y} = b^*\right\}$ and let $S^* = S \cap H^*$.

We observe that if $\mathbf{w}$ is a vertex of S^* 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.

Now consider the point $\mathbf{w}$ on the half-line from $\mathbf{z}$ through $\mathbf{x}$ that lies in S^* i.e

$$\mathbf{w} = \mathbf{z} + \frac{b^* - b}{\mathbf{a}^T\mathbf{x} - b}(\mathbf{x} - \mathbf{z}).$$

Now by induction, we can write $\mathbf{w} = \sum_{i=1}^{k} \lambda_i \mathbf{w}_i$ where $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k$ are extreme points of S^* and hence of S. Also, $\mathbf{x} = \mu\mathbf{w} + (1-\mu)\mathbf{z}$ for some $0 < \mu \leq 1$ and so $\mathbf{x} \in ext(S)$. □

The following is sometimes useful.

Lemma 19. *Suppose that S is a closed bounded convex set and that f is a convex function. The 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.

14 Lagrangean Duality

See Diagram 9 at the end of these notes.

Here we consider the *primal problem*

$$\text{Minimize } f(\mathbf{x}) \text{ subject to } g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, m, \tag{55}$$

where $f, g_1, g_2, \ldots, g_m$ are convex functions on $\mathbb{R}^n$.

The Lagrangean

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g(\mathbf{x}).$$

The *dual problem* is

$$\text{Maximize } \phi(\boldsymbol{\lambda}) \text{ subject to } \boldsymbol{\lambda} \geq 0 \text{ where } \phi(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in\mathbb{R}^n} L(\mathbf{x}, \boldsymbol{\lambda}). \tag{56}$$

We note that ϕ is a concave function. It is the minimum of a collection of convex (actually linear) functions of $\boldsymbol{\lambda}$ – see E3.

D1 :Linear programming. Let $f(\mathbf{x}) = \mathbf{c^T x}$ 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 $A\boldsymbol{\lambda} \neq \mathbf{c}$ implies that $\phi(\boldsymbol{\lambda}) = -\infty$. So the dual problem is

$$\text{Minimize } \mathbf{b}^T\boldsymbol{\lambda} \text{ subject to } A^T\boldsymbol{\lambda} = \mathbf{c}.$$

Weak Duality: If $\boldsymbol{\lambda}$ is feasible for (56) and $\mathbf{x}$ is feasible for (55) then $f(\mathbf{x}) \geq \phi(\boldsymbol{\lambda})$.

$$\phi(\boldsymbol{\lambda}) \leq L(\mathbf{x}, \boldsymbol{\lambda}) \leq f(\mathbf{x}) \text{ since } \lambda_i \geq 0, g_i(\mathbf{x}) \leq 0, \ i = 1, 2, \ldots, m. \tag{57}$$

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 (57).

Theorem 20. *Suppose that there exists a point* $\mathbf{x}^*$ *such that* $g_i(\mathbf{x}^*) < 0, i = 1, 2, \ldots, m$. *Then*

$$\max_{\boldsymbol{\lambda}\geq\mathbf{0}} \phi(\lambda) = \min_{\mathbf{x}: g_i(\mathbf{x})\leq 0, i\in[m]} f(\mathbf{x}).$$

Proof. Let

$$\mathcal{A} = \{\mathbf{u}, t) : \exists \mathbf{x} \in \mathbb{R}^n, g_i(\mathbf{x}) \le u_i, i = 1, 2, \dots, m \text{ and } f(\mathbf{x}) \le t\}.$$
$$\mathcal{B} = \left\{(0, s) \in \mathbb{R}^{m+1} : s < f^*\right\} \text{ where } f^* = \min_{\mathbf{x}: g_i(\mathbf{x}) \le 0, i \in [m]} f(\mathbf{x}).$$

Now $\mathcal{A} \cap \mathcal{B} = \emptyset$ and so from Corollary 13 there exists $\boldsymbol{\lambda}, \gamma, b$ such that $(\boldsymbol{\lambda}, \gamma) \neq \mathbf{0}$ and

$$b \le \min\left\{\boldsymbol{\lambda}^T \mathbf{u} + \gamma \mathbf{t} : (\mathbf{u}, \mathbf{t}) \in \mathcal{A}\right\}. \tag{58}$$
$$b \ge \max\left\{\boldsymbol{\lambda}^T \mathbf{u} + \gamma \mathbf{t} : (\mathbf{u}, \mathbf{t}) \in \mathcal{B}\right\}. \tag{59}$$

We deduce from (58) that $\boldsymbol{\lambda} \ge 0$ and $\bar{g} \ge 0$. If $\gamma < 0$ or $\lambda_i < 0$ for some i then the minimum in (58) is $-\infty$. We deduce from (59) that $\gamma t < b$ for all $t < f^*$ and so $\gamma f^* \le b$. And from (58) that

$$\gamma f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x}) \ge b \ge \gamma f^* \qquad \text{for all } \mathbf{x} \in \mathbb{R}^n. \tag{60}$$

If $\gamma > 0$ then we can divide (60) by γ and see that $L(\mathbf{x}, \boldsymbol{\lambda}) \ge f^*$, and together with weak duality, we see that $L(\mathbf{x}, \boldsymbol{\lambda}) = f^*$.

If $\gamma = 0$ then substituting $\mathbf{x}^*$ into (60) we see that $\sum_{i=1}^m \lambda_i g_i(\mathbf{x}^*) \ge 0$ which then implies that $\boldsymbol{\lambda} = 0$, contradiction. □

15 Conditions for a minimum: First Order Condition

15.1 Unconstrained problem

We discuss necessary conditons 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

$$f(\mathbf{a} + \mathbf{h}) = f(a) + (\nabla f(\mathbf{a}))^T \mathbf{h} + o(|\mathbf{h}|)$$

implies that

$$\nabla f(\mathbf{a}) = 0 \tag{61}$$

is a necessary condition for $\mathbf{a}$ to be a local minimum. Otherwise,

$$f\left(\mathbf{a} - t\nabla f(\mathbf{a})\right) \le f(\mathbf{a}) - t|\nabla f(\mathbf{a})|^2/2$$

for small $t > 0$.

Of course (61) is not sufficient in general, $\mathbf{a}$ could be a local maximum. Generally spealking, one has to look at second order conditions to distinguish between local minima and local maxima.

However,

Lemma 21. *If f is convex then* (61) *is also a sufficient condition.*

Proof. This follows directly from F3. □

15.2 Constrained problem

We will consider Problem (55), but we will not assume convexity, only differentiability. The condition corresponding to (61) 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 (or maximum) to Problem (55) is that there exists $\boldsymbol{\lambda}$ such that

$$g_i(\mathbf{a}) \leq 0, \qquad 1 \leq i \leq m. \tag{62}$$

$$\lambda_i \geq 0 \qquad 1 \leq i \leq m. \tag{63}$$

$$\nabla f(\mathbf{a}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{a}) = 0. \tag{64}$$

$$\lambda_i g_i(\mathbf{a}) = 0, \qquad 1 \leq i \leq m. \qquad \text{Complementary Slackness} \tag{65}$$

The second condition says that only *active* constraints ($g_i(\mathbf{a}) = 0$) are involved in the first condition.

One deals with $g_i(\mathbf{x}) \geq 0$ via $-g_i(\mathbf{x}) \leq 0$ (and $\lambda_i \leq 0$) and $g_i(\mathbf{x}) = 0$ by $g_i(\mathbf{x}) \geq 0$ and $-g_i(\mathbf{x}) \leq 0$ (and λ_i not constrined to be non-negative or non-positive).

In the convex case, we will see that (64), (63) and (65) are sufficient for a global minimum.

15.2.1 Heuristic Justification of KKT conditions

See Diagram 10 at the end of these notes.

Suppose that $\mathbf{a}$ is a local minimum and assume w.l.o.g. that $g_i(\mathbf{a}) = 0$ for $i = 1, 2, \ldots, m$. Then (heuristically) Taylor's theorem implies that if (i) $\mathbf{h}^T \nabla g_i(\mathbf{a}) \leq 0, i = 1, 2, \ldots, m$ then (ii) we should have $\mathbf{h}^T \nabla f(\mathbf{a}) \geq 0$. (The heuristic argument is that (i) holds then we should have (iii) $\mathbf{a} + \mathbf{h}$ feasible for small $\mathbf{h}$ and then we should have (ii) since we are at a local minimum. You need a regularity condition to ensure that (ii) implies (iii).)

Applying Corollary 14 we see that the KKT conditions hold. We let A have *columns* $\nabla g_i(\mathbf{a}), i = 1, 2, \ldots, m$. Then the KKT conditions are $A\boldsymbol{\lambda} = -\nabla f(\mathbf{a})$.

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 (64) and Lemma 21 implies that $\mathbf{x}^*$ minimises ϕ. But then for any feasible $\mathbf{x}$ we have

$$f(\mathbf{x}^*) = \phi(\mathbf{x}^*) \leq \phi(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i^* g_i(\mathbf{x}) \leq f(\mathbf{x}).$$

For much more on this subject see Convex Optimization, by Boyd and Vendenberghe.

Diagram 1

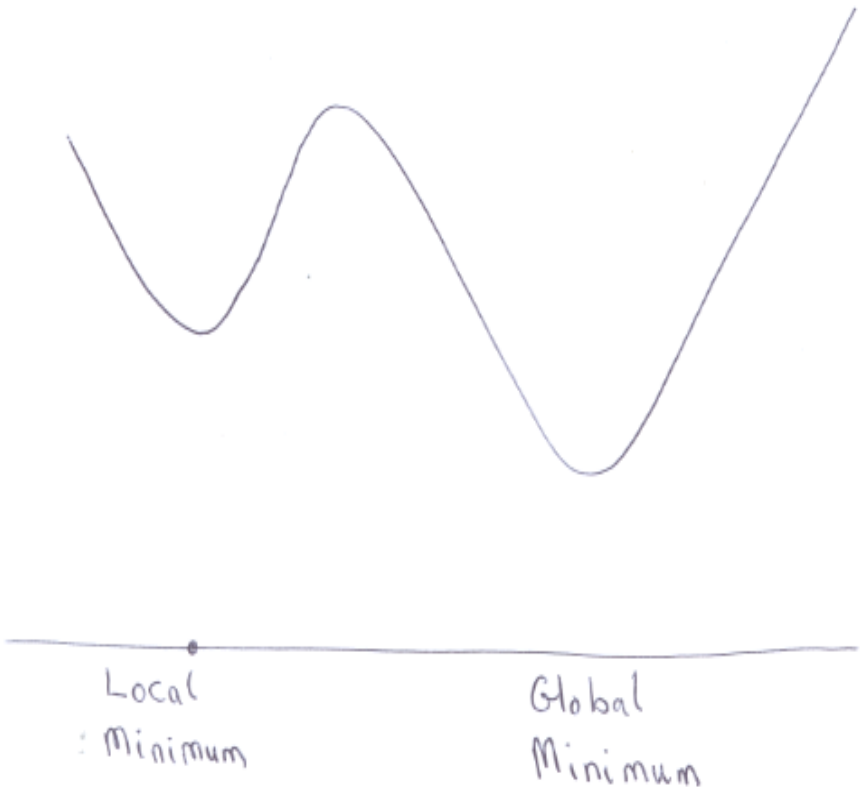
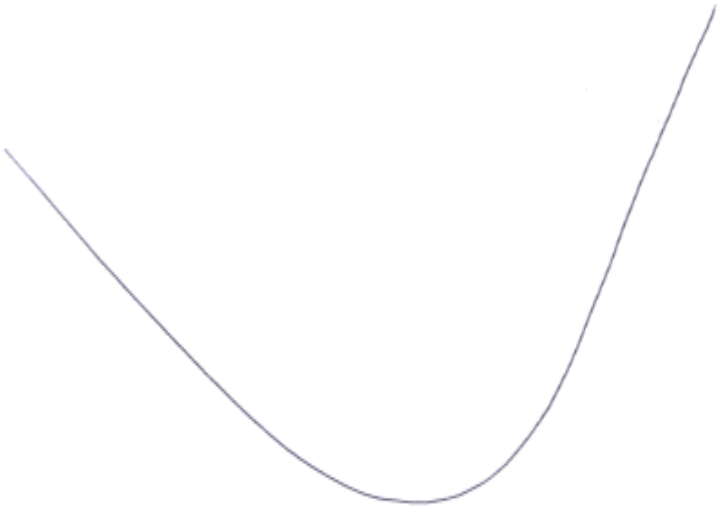
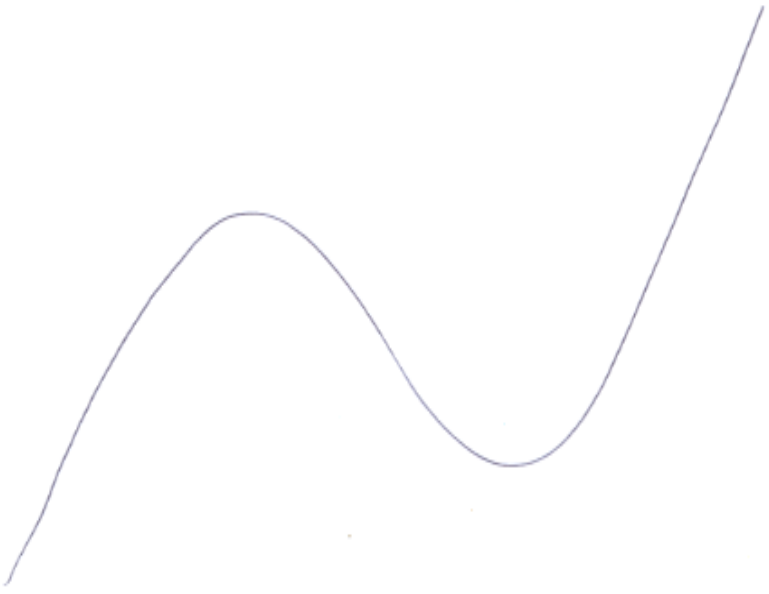


Diagram 2

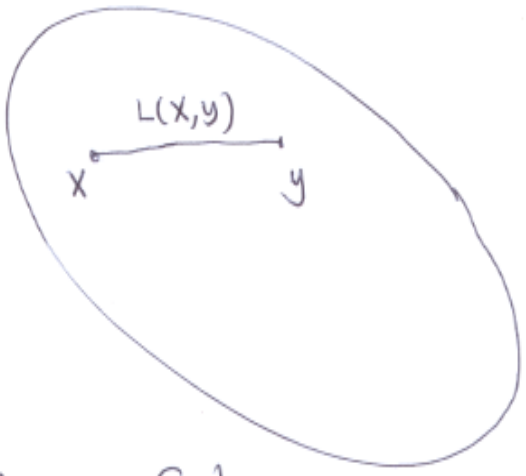


Convex Function

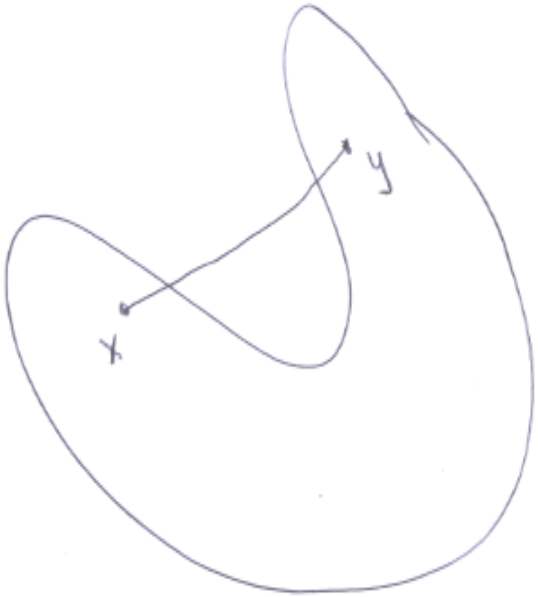


Non-Convex Function

Diagram 3



Convex Set



Non-Convex Set

Diagram 4

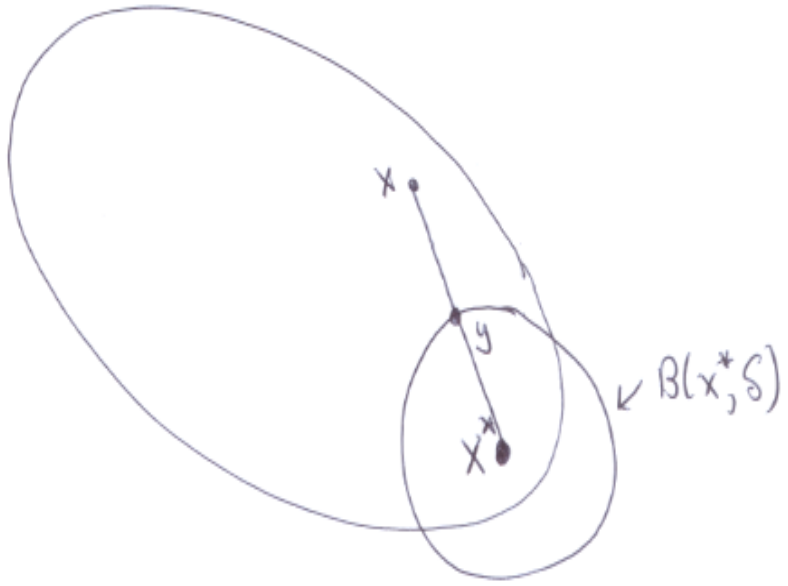


Diagram 5

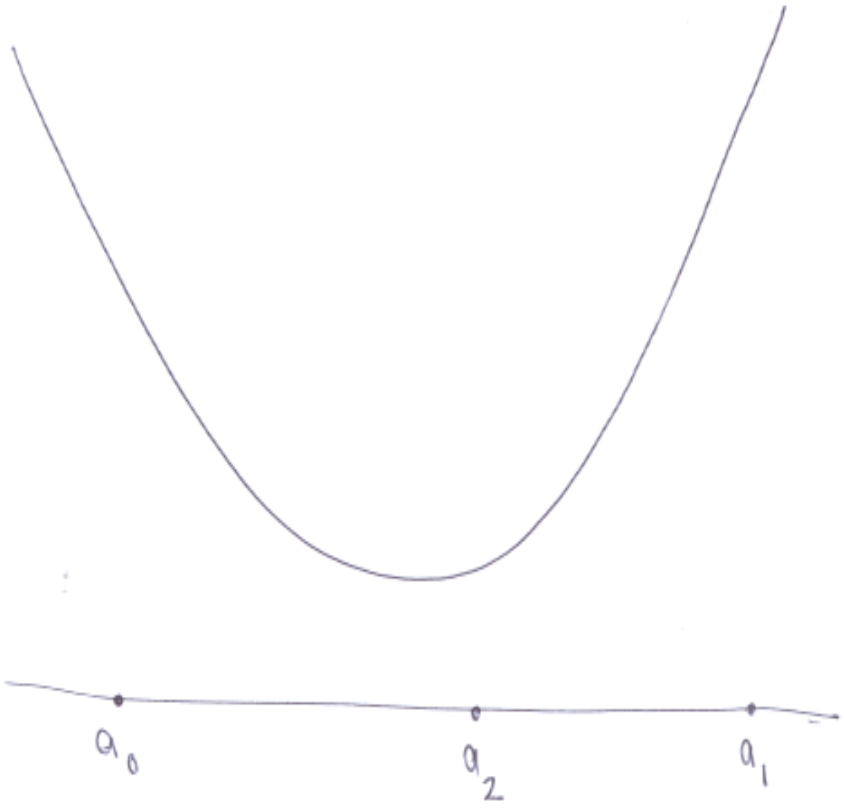


Diagram 6

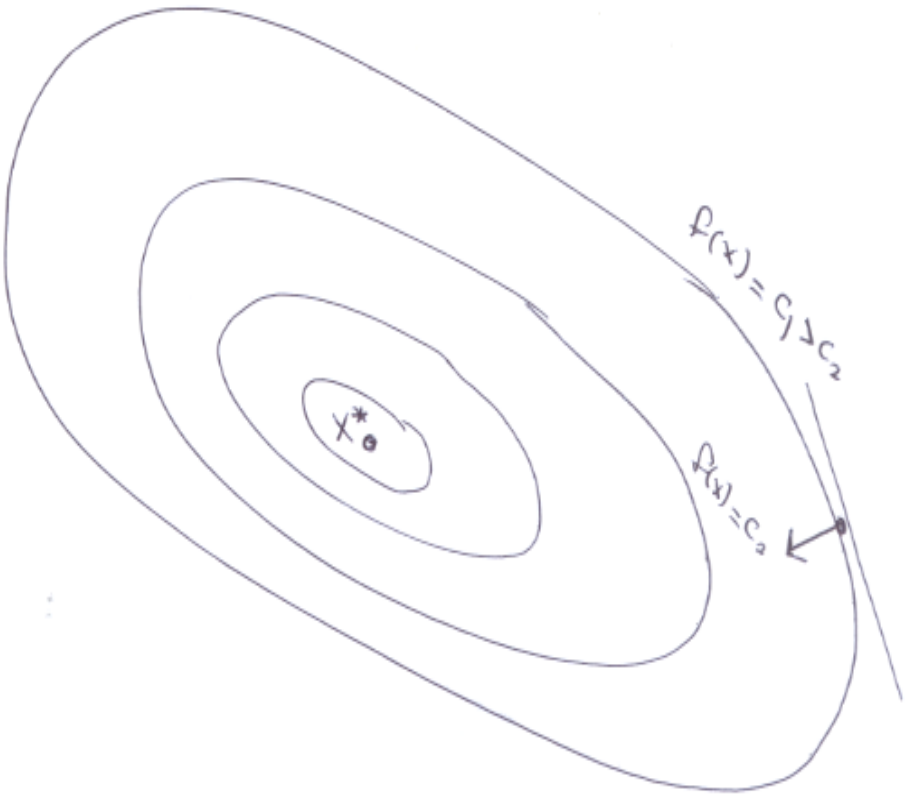


Diagram 7

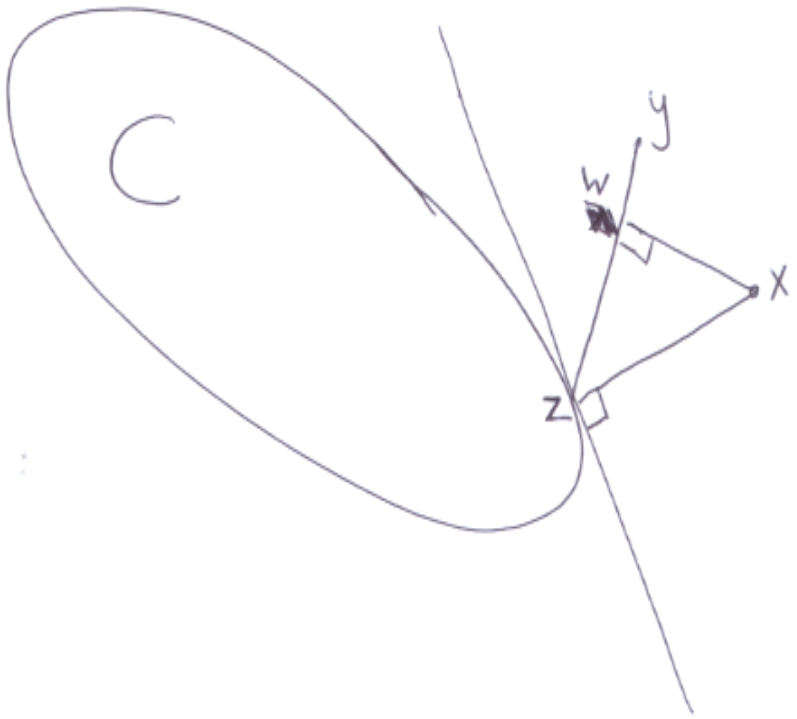


Diagram 8

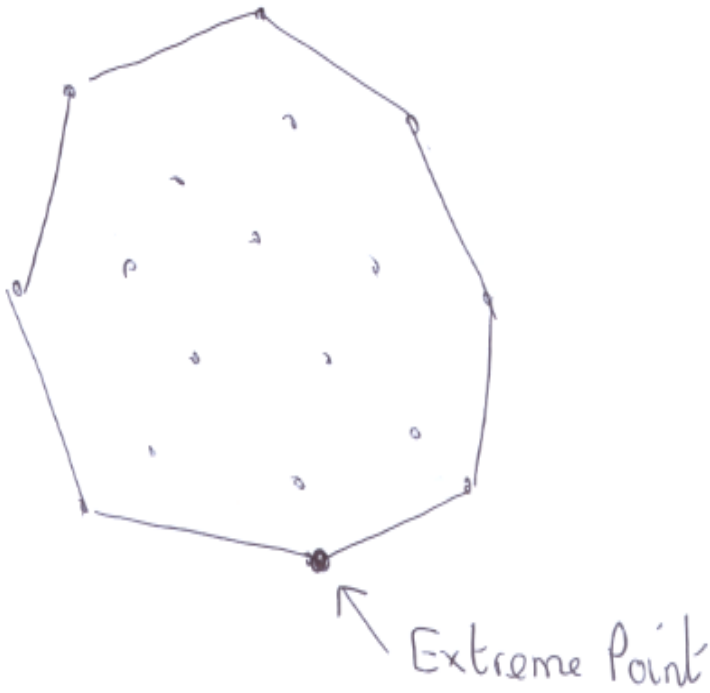


Diagram 9

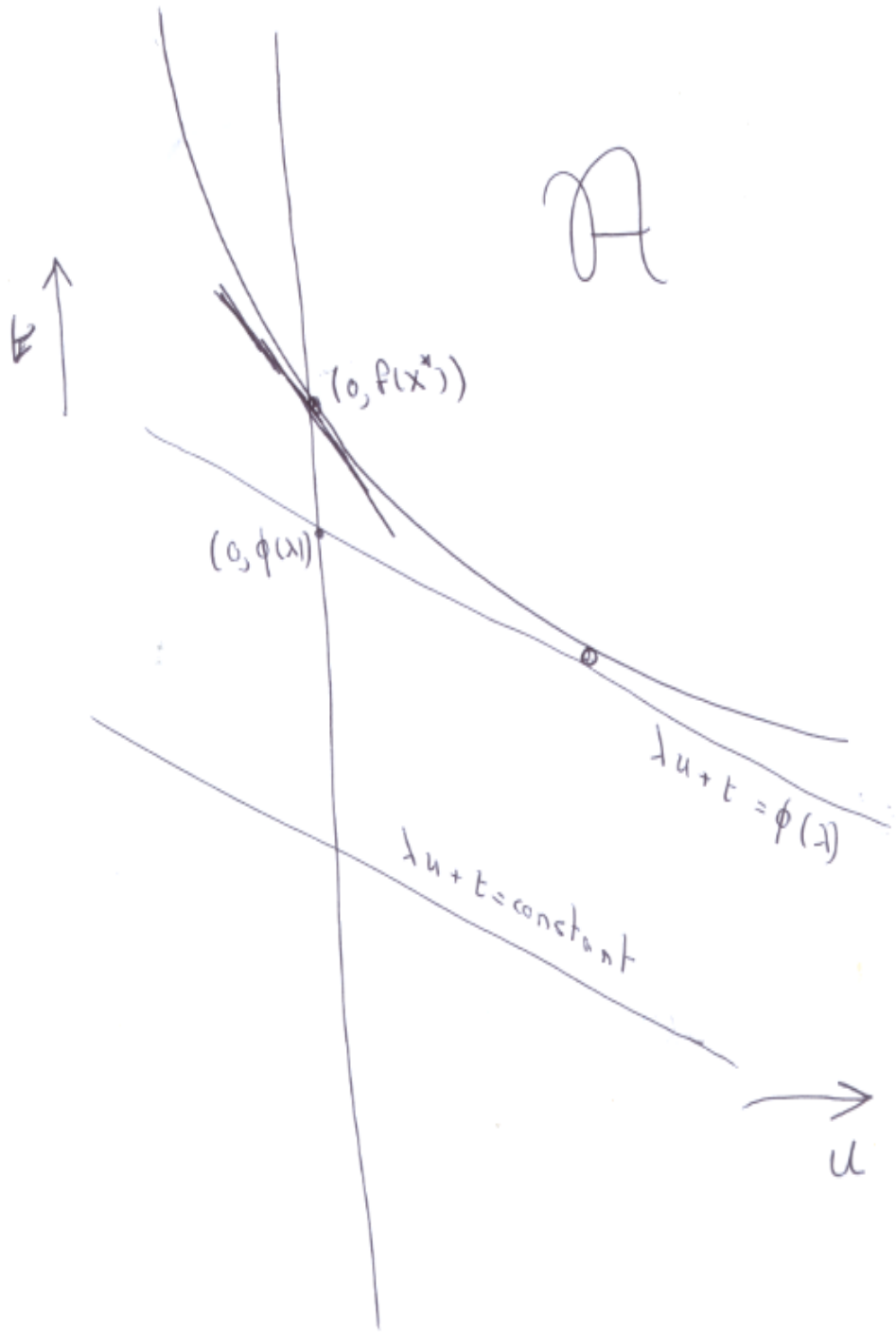

A
$(0, f(x^*))$
$(0, \phi(\lambda))$
$\lambda u + t = \phi(\lambda)$
$\lambda u + t = \text{constant}$
u

Diagram 10