Operations Research
Semester 5

Ahmad Abu Zainab

Contents

Chapter 1

Linear Programming

A linear programming problem is a problem in the form

$$\max Z = \sum_{i=0}^{n} (c_i) x_i.$$

Subject to

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &\leqslant b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n &\leqslant b_2 \\ &\vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n &\leqslant b_1 \end{aligned}.$$

Where x_i are the decision variables
In matrix form

$$\max Z = \mathbf{c}^T \mathbf{x}.$$

Subject to

$$\mathbf{Ax} \leqslant \mathbf{b}.$$

$$\mathbf{x} \geqslant \mathbf{0}.$$

1.1 Simplex Method

1.1.1 Augmented Form

First we need to convert the problem to the standard form

$$\max Z = \mathbf{c}^T \mathbf{x}.$$

Subject to

$$\mathbf{Ax} = \mathbf{b}.$$

$$\mathbf{x} \geqslant \mathbf{0}.$$

Then we transform the problem to the augmented form by adding slack variables. We add m slack variables to the problem, where m is the number of constraints.

$$\max Z = \mathbf{c}^T \mathbf{x}.$$

Subject to

$$\mathbf{Ax} + \mathbf{Is} = \mathbf{b}.$$

$$\mathbf{x}, \mathbf{s}, \mathbf{b} \geqslant \mathbf{0}.$$

Where **I** is the identity matrix and **s** is the vector of slack variables.

1.1.2 Basic Feasible Solution

A basic solution is a solution where n of the variables are set to zero and the rest are set to the m values of the corresponding entries in **b**.
A basic *feasible* solution is a basic solution where all the variables are non-negative, and it corresponds to a vertex of the feasible region. Two adjacent vertices share all but one basic variable.
Variables set to zero are called non-basic variables, and the rest are called basic variables. The choice of basic variables is called the basis.
A basic feasible solution is called degenerate if one of the basic variables is zero.

1.1.3 Initialising the Simplex Method

We set up the initial tableau by adding the slack variables and the objective function to the augmented form, as follows

BV	x_1	x_2	$\cdots$	x_n	s_1	s_2	$\cdots$	s_m	RHS
Z	$-c_1$	$-c_2$	$\cdots$	$-c_n$	0	0	$\cdots$	0	0
s_1	a_{11}	a_{12}	$\cdots$	a_{1n}	1	0	$\cdots$	0	b_1
s_2	a_{21}	a_{22}	$\cdots$	a_{2n}	0	1	$\cdots$	0	b_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
s_m	a_{m1}	a_{m2}	$\cdots$	a_{mn}	0	0	$\cdots$	1	b_m

.

Let's take an example

$$\max Z = 3x_1 + 5x_2.$$

Subject to

$$\begin{aligned} x_1 \quad\quad &\leqslant 4 \\ 2x_2 &\leqslant 12 . \\ 3x_1 + 2x_2 &\leqslant 18 \end{aligned}$$

$$x_1, x_2 \geqslant 0.$$

First we convert the problem to the augmented form by adding slack variables

$$\boxed{\begin{array}{l} \max Z = 3x_1 + 5x_2 \\ \\ \text{Subject to} \\ \begin{array}{rcl} x_1 \quad\quad + s_1 \quad\quad &=& 4 \\ 2x_2 + \quad s_2 \quad &=& 12 \\ 3x_1 + 2x_2 + \quad\quad s_3 &=& 18 \end{array} \end{array}}$$

Then we set up the initial tableau

BV	x_1	x_2	s_1	s_2	s_3	RHS
Z	-3	-5	0	0	0	0
s_1	1	0	1	0	0	4
s_2	0	2	0	1	0	12
s_3	3	2	0	0	1	18

.

First, we identify the entering variable. The entering variable is the variable with the most negative coefficient in the objective function. In this case, the entering variable is x_2.

BV	x_1	x_2	s_1	s_2	s_3	RHS
Z	-3	-5	0	0	0	0
s_1	1	0	1	0	0	4
s_2	0	2	0	1	0	12
s_3	3	2	0	0	1	18

.

Then, we identify the leaving variable. The leaving variable is the variable with the smallest ratio of the RHS to the coefficient of the entering variable in the objective function. In this case, the leaving variable is s_2.

BV	x_1	x_2	s_1	s_2	s_3	RHS
Z	-3	-5	0	0	0	0
s_1	1	0	1	0	0	4
s_2	0	2	0	1	0	12
s_3	3	2	0	0	1	18

.

Then we perform the pivot operation on the leaving variable and the entering variable. The pivot operation is performed by dividing the row of the leaving variable by the coefficient of the entering variable in that row, and then subtracting the resulting row from the other rows, multiplied by the coefficient of the entering variable in that row.
After the pivot operation, the tableau becomes

BV	x_1	x_2	s_1	s_2	s_3	RHS
Z	-3	0	0	2.5	0	30
s_1	1	0	1	0	0	4
x_2	0	1	0	0.5	0	6
s_3	3	0	0	-1	1	6

.

Finally, we repeat the process until the objective function has no negative coefficients.

Note:-

The *optimality test* is as follows

- If the objective function has no negative coefficients, then the current solution is optimal.
- If the objective function has negative coefficients, then the current solution is not optimal, and we repeat the process.

In our case, after iterating another time (x_1 entering and s_3 leaving), we get

BV	x_1	x_2	s_1	s_2	s_3	RHS
Z	0	0	0	1.5	1	36
s_1	0	0	1	1/3	$-1/3$	2
x_2	0	1	0	0.5	0	6
x_1	1	0	0	$-1/3$	1/3	2

.

Since the objective function has no negative coefficients, the current solution is optimal. The optimal solution is $x_1 = 2, x_2 = 6, s_1 = 2, s_2 = 0, s_3 = 0$ and $Z = 36$.

1.1.4 Unbounded Solution

In a given tableau, if the entering variable has all zero entries in its column, then the solution is unbounded. Which means that the objective function can be increased indefinitely.

1.1.5 Alternative Optimal Solutions

Alternative optimal solutions occur when one of the non-basic variables has a zero coefficient in the objective function in the final tableau. In this case, the solution is degenerate.

1.1.6 Non-Standard Form

$$0.4x_1 - 0.3x_2 \geqslant -10 \overset{\times -1}{\Longrightarrow} -0.4x_1 + 0.3x_2 \leqslant 10$$
$$\min Z = 0.4x_1 + 0.3x_2 \overset{\times -1}{\Longrightarrow} \max Z = -0.4x_1 - 0.3x_2$$

1.1.7 Artificial Variables

Consider the following linear system

$$\mathbf{Ax} = \mathbf{b}.$$

and

$$\mathbf{x} \geqslant \mathbf{0}.$$

We can use the simplex method to solve this system by adding artificial variables to the system, as follows

$$\mathbf{Ax} + \mathbf{Ia} = \mathbf{b}.$$

$$\mathbf{x}, \mathbf{a} \geqslant \mathbf{0}.$$

Where **I** is the identity matrix and **a** is the vector of artificial variables.
Take the following example

$$\begin{array}{rcrcrcl} x_1 & + & 2x_2 & + & x_3 & = & 4 \\ x_1 & + & x_2 & & & = & 3 \end{array}.$$

The augmented form is

$$\begin{array}{rcrcrcrcrcl} x_1 & + & 2x_2 & + & x_3 & + & a_1 & & & = & 4 \\ x_1 & + & x_2 & & & + & & & a_2 & = & 3 \end{array}.$$

And we can solve it using the simplex method.

Two-Phase Method

In the case where the basic feasible solution isn't very obvious, we can use the two-phase method to find the basic feasible solution by adding artificial variables to the system, as follows

$$\mathbf{Ax} + \mathbf{Is} + \mathbf{Ia} = \mathbf{b}.$$

$$\mathbf{x}, \mathbf{s}, \mathbf{a} \geqslant \mathbf{0}.$$

Where **I** is the identity matrix and **s** is the vector of slack variables and **a** is the vector of artificial variables.
In the first phase we solve the problem starting from the BFS where all the artificial variables are basic variables. If the problem is feasible, we will be able to find a BFS where all artificial variables are 0. This automatically gives us a BFS for the original problem.
For example, consider the following problem

$$\min Z = 2x_1 + 3x_2.$$

Subject to

$$\begin{array}{rcrcl} x_1 & + & 2x_2 & = & 4 \\ 2x_1 & - & x_2 & = & 3 \end{array}.$$

We can convert it to the following augmented form

$$\begin{array}{rcrcrcrcl} x_1 & + & 2x_2 & & & + & a_1 & = & 4 \\ 2x_1 & - & x_2 & + & a_2 & & & = & 3 \end{array}.$$

We define a new objective function

$$\max W = -a_1 - a_2.$$

The initial tableau is set up as follows

BV	x_1	x_2	a_1	a_2	RHS
W	-3	-1	0	0	7
Z	-2	-3	0	0	0
a_1	1	2	1	0	4
a_2	2	-1	0	1	3

.

The resulting solution is a BFS for the original problem. In the second phase we solve the original problem starting from the BFS we found in the first phase.

Big-M Method

The Big-M method is a variation of the two-phase method where we add a large number M to the objective function for each artificial variable. This ensures that the artificial variables will be set to zero in the optimal solution.
The objective function looks like this

$$\max Z = \mathbf{c}^T\mathbf{x} - M\mathbf{a}.$$

The initial simplex tableau is set up as follows

BV	x_1	x_2	$\cdots$	x_n	s_1	s_2	$\cdots$	s_m	RHS
Z	$-c_1$	$-c_2$	$\cdots$	$-c_n$	M	M	$\cdots$	M	0
s_1	a_{11}	a_{12}	$\cdots$	a_{1n}	1	0	$\cdots$	0	b_1
s_2	a_{21}	a_{22}	$\cdots$	a_{2n}	0	1	$\cdots$	0	b_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
s_m	a_{m1}	a_{m2}	$\cdots$	a_{mn}	0	0	$\cdots$	1	b_m

.

1.1.8 Infeasible LPs

An LP is infeasible if in the final tablea there are still artificial variables that are basic variables.

1.1.9 Constraints with $\geqslant$

If we have a contraint with $\geqslant$ and a negative RHS we can multiply the constraint by -1 and proceed as usual. However, if we have a constraint with $\geqslant$ and a positive RHS we turn the constraint into an equality by adding a slack variable as seen below

$$x_1 - 2x_2 + x_3 \geqslant 20 \quad \Longrightarrow \quad x_1 - 2x_2 + x_3 - s_1 = 20.$$

Where $s_1 > 0$.

But for initial BFS we cannot simplex s_1 in to the base so we need to add an artificial variable a_1 to the contraint as if it were an equality contraint

$$x_1 - 2x_2 + x_3 - s_1 + a_1 = 20.$$

Where $a_1 > 0$.

1.1.10 Variables Unconstrained in Sign

If we have a variable that is unconstrained in sign we can split it into two variables ${x_1}^+$ and ${x_1}^-$ where $x_1 = {x_1}^+ - {x_1}^-$.

$${x_1}^+ = \max(0, x_1) \quad {x_1}^- = \max(0, -x_1).$$

1.1.11 Maximize the Minimum

Some real life problems consist of maximizing the lowest of many economic functions.

$$\max Z = \min(3x_1 + 2x_2, x_1 - 2x_3).$$

We can solve this problem by adding a new variable u and adding the following constraints

$$\begin{array}{r} 3x_1 + 2x_2 - u \geqslant 0 \\ x_1 - 2x_3 - u \geqslant 0 \end{array}.$$

where u is unconstrained in sign.

Chapter 2

Network Optimization Models

2.1 Max Flow Problem

Consider a directed graph $G = (V, E)$ with a source node s and a sink node t. Each edge $(i, j) \in E$ has a capacity u_{ij}. The max flow problem is to find the maximum flow from s to t.

LP Model

$$\max Z = \sum_{i \in \delta^+(s)} x_{si}.$$

or

$$\max Z = \sum_{i \in \delta^-(t)} x_{it}.$$

where $\delta^+(s)$ is the set of nodes that are flowing out of to s and $\delta^-(t)$ is the set of nodes that are flowing in to t.

Subject to

$$-u_{ij} \leqslant x_{ij} \leqslant u_{ij}.$$

and

$$\sum_{i \in \delta^+(j)} x_{ij} = \sum_{i \in \delta^-(j)} x_{ji} \quad \forall j \in V \setminus \{s, t\}.$$

2.2 Min Cost Flow Problem

Consider a directed graph $G = (V, E)$. Each edge $(i, j) \in E$ has a capacity u_{ij}, a cost c_{ij}, and a supply b_i. The min cost flow problem is to find the minimum cost flow within this network.

LP Model

$$\min Z = \sum_{(i,j) \in E} c_{ij} x_{ij}.$$

Subject to

$$\sum_{i \in \delta^+(j)} x_{ji} - \sum_{i \in \delta^-(j)} x_{ij} = b_j \quad \forall j \in V.$$

and

$$x_{ij} \leqslant u_{ij} \quad \forall (i, j) \in E.$$

Note:-

The supply b_i is positive if the node is a source, negative if the node is a sink, and zero if the node is a transshipment node.

Note:-

If the capacity u_{ij} is not specified, we can assume it is ∞. (The edge is unconstrained in capacity)

2.2.1 Special Cases

Transportation Problem

Sources have $b_i > 0$, sinks have $b_i < 0$, and transshipment nodes have $b_i = 0$. All nodes have unlimited capacity, and edges all have a cost c_{ij}.

Shortest Path Problem

Sources have $b_i = 1$, sinks have $b_i = -1$, and transshipment nodes have $b_i = 0$. All nodes have unlimited capacity, and edges all have a cost c_{ij}.

Sink Tree Problem

Sources have b_i = number of nodes in the network, and at every other node $b_i = -1$. All nodes have unlimited capacity, and edges all have a cost c_{ij}.

Max Flow Problem

At the source we set $b = M$, where M is a very large number, and we set $b = -M$ for target nodes. We then create a dummy node d and connect it to the source and target nodes with unlimited capacity and cost 1. The simplex algorithm will route flow through the non-dummy links since they have 0 cost.

Theorem 2.2.1 Integrality Theorem

If all b_i and u_{ij} are integers, then there exists an optimal solution where all x_{ij} are integers.

2.3 Shortest Path Problem

Given a directed graph $G = (V, E)$ with a source node s and a sink node t. Each edge $(i, j) \in E$ has a cost c_{ij}. The shortest path problem is to find the shortest path (minimum cost) from s to t.

LP Model

$$\min Z = \sum_{(i,j)\in E} c_{ij}x_{ij}.$$

Subject to

$$\sum_{i\in\delta^+(j)} x_{ji} - \sum_{i\in\delta^-(j)} x_{ij} = \begin{cases} 1 & \text{if } j = s \\ -1 & \text{if } j = t \\ 0 & \text{otherwise} \end{cases}.$$

where x_{ij} is a binary value (0 or 1).

2.3.1 Dijkstra's Algorithm

Given the same setup as the shortest path problem, we can solve it using Dijkstra's algorithm. The algorithm is as follows

1. Set $d(s) = 0$ and $d(i) = \infty$ for all other nodes.
2. Set $S = \emptyset$ (the set of visited nodes).
3. Set $i = s$.
4. Find the node j with the smallest $d(j)$ such that $j \notin S$.
5. Add j to S.
6. For all $k \notin S$, set $d(k) = \min(d(k), d(j) + c_{jk})$.
7. If $j \neq t$, set $i = j$ and go to step 4.
8. Repeat step 4 until $S = V$.

Algorithm 1: Dijkstra's Algorithm

```
Result: Shortest path from $s$ to $t$
Function Dijkstra($G, s, t$):
    $S \leftarrow \emptyset$;
    $d(s) \leftarrow 0$;
    while $t \notin S$ do
        $j \leftarrow \text{argmin}_{j \notin S}\, d(j)$;
        $S \leftarrow S \cup \{j\}$;
        for $k \notin S$ do
            $d(k) \leftarrow \min(d(k), d(j) + c_{jk})$;        /* $c_{ij}$ being the cost of the edge */
        end
    end
    return $d(t)$;
```

2.4 Minimum Spanning Tree Problem

Given a connected undirected graph $G = (V, E)$ with a cost c_{ij} for each edge $(i, j) \in E$. The minimum spanning tree problem is to find the minimum cost tree that connects all nodes.

We can solve this using an algorithm called Prim's algorithm. The algorithm is as follows

1. Set $S = \{s\}$ (the set of visited nodes).
2. Set $i = s$.
3. Find the edge (i, j) with the smallest c_{ij} such that $j \notin S$.
4. Add j to S.
5. If $S = V$, stop.
6. Set $i = j$ and go to step 3.

Algorithm 2: Prim's Algorithm

```
Result: Minimum spanning tree
Function Prim($G$):
    $S \leftarrow \{s\}$;
    while $S \neq V$ do
        $(i, j) \leftarrow \mathrm{argmin}_{j \notin S}\, c_{ij}$;
        $S \leftarrow S \cup \{j\}$;
    end
    return $S$;
```

Chapter 3

Transportation and Assignment Problems

3.1 Transportation Problem

The transportation problem is an LP problem where we have a set of m supply nodes and a set of n demand nodes. Each supply node has a supply (capacity) s_i and each demand node has a demand $-d_j$. The cost of transporting one unit $x_{ij} = 1$ from supply node i to demand node j is c_{ij} . The transportation problem is to find the minimum cost of transporting the supply to the demand. The total cost of shipping is

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

In a balanced transportation problem, the total supply is equal to the total demand.

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

3.1.1 LP Model

$$\min Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}.$$

Subject to

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} &= s_i \quad \forall i \\ \sum_{i=1}^{m} x_{ij} &= d_j \quad \forall j \\ x_{ij} &\geqslant 0 \quad \forall i, j \end{aligned}$$

The LP has

- mn variables
- $m + n$ equality constraints
- One redundant constraint
- The effective number of constraints is $m + n - 1$

- The rank of the constraint matrix is $m + n - 1$

Due to the shape of the problem the simplex method is not efficient for solving the transportation problem. Instead we use the transportation algorithm

1. For the non-redundant constraints, we select the next basic variable according to a rule (northwest corner rule or the Vogel rule).
2. Set that basic varaible to the largest possible value. This is either the remaining supply or the remaining demand, whichever is smaller.
3. Update the supply and demand and remove the constraint.

If both the supply and demand are zero, then we eliminate the row and assume that the demand is 0. We will have a degenerate basic variable in the final solution.

Theorem 3.1.1 Northwest rule

We start at the northwest corner of the cost matrix and allocate as much as possible to the basic variable regardless of the cost.

Theorem 3.1.2 Vogel's rule

We calculate the difference between the two smallest costs for each row and each column. We then select the row or column with the largest difference and allocate as much as possible to the basic variable.

3.1.2 Special Cases

Prohibited Routes If a route is prohibited, we can set the cost to M (∞).

Excess Supply If there is excess supply, we can add a dummy demand node with a demand of the excess supply and a cost of 0. The quantity shipped to the dummy node is "throw away".

Excess Demand(shortage) If there is excess demand, we can add a dummy supply node with a supply of the excess demand and a cost of 0. The quantity shipped from the dummy node is "how much the destination will be missing out".

- Distribute the available production with minimal shipping cost, ignoring fairness.
- Guarantee some "minimum" to some (or all) destinations. Split destination D_j in 2:
  - D_j^1 with demand d_j^1 = minimum demand and cost $c_{ij}^1 = \infty$.
  - D_j^2 with the remaining demand ($d_j^2 = d_j - d_j^1$) and cost $c_{ij}^2 = 0$.
- **Proportional shortage** No dummy source. Adjust the demands proportionally to match the available supply. All destinations receive the same "proportion" of their demand. For example, if the supply is 85% of the total demand, each destination receives 85% of its need. For each destination D_j modify the demand $d_j = d_j \times$ total supply/total demand

3.2 Assignment Problem

The assignment problem is a special case of the transportation problem where the supply and demand are both equal to 1. The assignment problem is to find the minimum cost of assigning n tasks to n workers. The decision variables are binary, $x_{ij} = 1$ if task i is assigned to worker j, and $x_{ij} = 0$ otherwise. The total cost of the assignment is

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

In its standard form, AP is balanced (the total number of workers is equal to the total number of tasks). When not balanced, we can add dummy workers or tasks to balance the problem. Its dedicated algorithm is the Hungarian algorithm.

3.2.1 LP Model

$$\min Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}.$$

Subject to (with on redundant constraint)

$$\begin{aligned}
\sum_{j=1}^{n} x_{ij} &= 1 \quad \forall i \\
\sum_{i=1}^{n} x_{ij} &= 1 \quad \forall j \\
x_{ij} &\geqslant 0 \quad \forall i, j
\end{aligned}$$

3.2.2 The Hungarian Algorithm

Theorem 3.2.1

The ordering by cost of feasible solutions does not change if in the cost table

- we add or subtract a constant from all the elements of a row
- we add or subtract a constant from all the elements of a column
- we add or subtract a constant from all the elements of the table

The Hungarian algorithm is a method for solving the assignment problem. The algorithm is as follows

1. Subtract the smallest element in each row from all the elements in that row.
2. Subtract the smallest element in each column from all the elements in that column.
3. Find the smallest number of lines that can cover all the zeros in the matrix. If the number of lines is equal to n, stop.
4. Find the smallest number not covered by the lines and subtract it from all the uncovered elements and add it to the elements that are covered by two lines.
5. Go to step 3.

We then allocate the tasks to the workers according to the zeros in the matrix. If there are multiple zeros in a row or column, we allocate the task to the worker who has the least amount of zeros in his row and column.

3.2.3 Special Cases

Prohibited Assignments If an assignment is prohibited, we can set the cost to M (∞).

Unbalanced Assignment If the number of workers is not equal to the number of tasks, we can add dummy workers or tasks to balance the problem.

Maximization When we have a profit table (p_{ij}) instead of a cost table, we can convert the problem to a minimization problem by:

1. Find a number k that is greater or equal all elements: $k \geqslant p_{ij}$ for all i, j (we can choose k as the highest profit in the matrix)
2. Replace each profit element p_{ij} by a cost $c_{ij} = k - p_{ij}$
3. Solve the minimization problem

Chapter 4

Nonlinear Programming

Note:-

- If f concave and has a local maximum, it is a global maximum.
- If f convex and has a local minimum, it is a global minimum.

4.1 Univariate Unconstrained Optimization

4.1.1 Bisection Method

Given a function $f(x)$, we can find the local maximum by using the bisection method given 2 numbers $a < b$ such that $f'(a) > 0 > f'(b)$. The algorithm is as follows

1. Given 2 numbers a and b.
2. Set $x = \frac{a+b}{2}$.
3. If $\frac{b-a}{2} \leqslant \varepsilon$, stop.
4. If $f'(x) > 0$, set $a = x$.
5. Else, set $b = x$.
6. Repeat steps 2-5.

Algorithm 3: Bisection Method

```
Result: Minimum of $f(x)$
Function Bisection($f(x), a, b, \varepsilon$):
    $x \leftarrow \frac{a+b}{2}$;
    while $\frac{b-a}{2} > \varepsilon$ do
        if $f'(x) > 0$ then
            $a \leftarrow x$;
        else
            $b \leftarrow x$;
        end
        $x \leftarrow \frac{a+b}{2}$;
    end
    return $x$;
```

4.1.2 Newton's Method

Given a function $f(x)$, we can find the local maximum by using Newton's method given a number x_0. This function garantees a local maximum if f is concave (minimum if f is convex). The algorithm is as follows

1. Given a number x_0.
2. Set $x = x_0 - \frac{f'(x_0)}{f''(x_0)}$.
3. If $|x - x_0| \leqslant \varepsilon$, stop.
4. Set $x_0 = x$.
5. Repeat steps 2-4.

Algorithm 4: Newton's Method

```
Result: Minimum of $f(x)$
Function Newton($f(x), x_0, \varepsilon$):
    $x \leftarrow x_0 - \frac{f'(x_0)}{f''(x_0)}$;
    while $|x - x_0| > \varepsilon$ do
        $x_0 \leftarrow x$;
        $x \leftarrow x_0 - \frac{f'(x_0)}{f''(x_0)}$;
    end
    return $x$;
```

4.2 Multivariate Unconstrained Optimization

4.2.1 Gradient Descent

Given a function $f(x)$, we can find the local maximum by using the gradient descent method given a number x_0. The function doesn't garantee a local maximum except when f is convex/concave. We consider an initial guess x_0 and a step size t, then we update the guess $x_1 = x_0 + t\nabla f(x_0)$ and repeat the process. We find the step size t by using the exact line search method. The algorithm is as follows

1. Given a number x_0 and calculate $\nabla f(x_0)$.
2. Maximize the univariate function $g(t) = f(x_0 + t\nabla f(x_0))$. (Exact line search)
3. Set $x_1 = x_0 + t\nabla f(x_0)$.
4. If $|x_1 - x_0| \leqslant \varepsilon$, stop.

Algorithm 5: Gradient Descent Method

```
Result: Minimum of $f(x)$
Function GradientDescent($f(x), x_0, \varepsilon$):
    while $|x_1 - x_0| > \varepsilon$ do
        $t \leftarrow \text{argmin}_t\, f(x_0 + t\nabla f(x_0))$;
        $x_1 \leftarrow x_0 + t\nabla f(x_0)$;
        $x_0 \leftarrow x_1$;
    end
    return $x_1$;
```

4.2.2 Newton's Method

Given a function f and an initial guess x_0

$$f(x_0) + \nabla f(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0).$$

$$x_{i+1} = x_i - \nabla^2 f(x_i)^{-1} \nabla f(x_i).$$

Iterate until some convergence criteria is met $\|x_{i+1} - x_i\| \leqslant \varepsilon$ or $\max_j \left\|{x_j}^{i+1} - {x_j}^{i}\right\| \leqslant \varepsilon$.

The algorithm is as follows

Algorithm 6: Newton's Method

```
  Result: Minimum of $f(x)$
1 Function Newton($f(x), x_0, \varepsilon$):
2     while $\|x_{i+1} - x_i\| > \varepsilon$ do
3         $x_{i+1} \leftarrow x_i - \nabla^2 f(x_i)^{-1} \nabla f(x_i)$;
4         $x_i \leftarrow x_{i+1}$;
5     end
6     return $x_{i+1}$;
```

4.3 Convex Sets

A set S is convex if for any $x, y \in S$ and any $\lambda \in [0, 1]$, we have $\lambda x + (1 - \lambda)y \in S$.

4.4 Hessian Matrix and Definiteness

The Hessian matrix is the matrix of second derivatives of a function. It is defined as

$$\nabla^2 f(x, y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}.$$

$$f \text{ strictly convex } \Leftrightarrow \nabla^2 f(x) \text{ positive definite }.$$

$$H^{n \times n} \text{ positive definite } \Leftrightarrow \text{ eigenvalues} > 0 \Leftrightarrow \forall \mathbf{v} \in \mathbb{R}^n, \mathbf{v}^T H \mathbf{v} > 0 \Leftrightarrow \text{ all leading principal minors are positive.}$$

$$f \text{ convex } \Leftrightarrow \nabla^2 f(x) \text{ positive semi-definite }.$$

$$H^{n \times n} \text{ positive semi-definite } \Leftrightarrow \text{ eigenvalues} \geqslant 0 \Leftrightarrow \forall \mathbf{v} \in \mathbb{R}^n, \mathbf{v}^T H \mathbf{v} \geqslant 0 \Leftrightarrow \text{ all leading principal minors } \geqslant 0.$$

4.5 Langrange Multipliers

Given a function $f(x)$ and a set of constraints $g_i(x) = 0$, we can find the local maximum by using the Langrange multipliers method. At x^* the gradient of f is orthogonal to the curve of the constraints.

The optimality conditions are

$$\nabla f(x^*) = \lambda \nabla g(x^*).$$

Points that verify these conditions are called stationary points.

Langrange unified the constraints and the objective function into a single function $L(x, \lambda) = f(x) - \lambda g(x)$. We find the stationary points of L and verify that the constraints are satisfied.

$$\frac{\partial L}{\partial x} = 0 \implies \nabla f - \lambda \nabla g = 0 \text{Optimality condition} \quad .$$

$$\frac{\partial L}{\partial \lambda} = 0 \implies g(x) = 0 \text{Constraint} \quad .$$

When there are multiple constraints L is defined as

$$L(x, \lambda) = f(x) - \sum_i \lambda_i g_i(x) = f(x) - \boldsymbol{\lambda}^T \mathbf{g}(x).$$

When the constraints are inequalities $g_i(x) \leqslant 0$, we define the Lagrangian as

$$L(x, \lambda) = f(x) - \sum_i u_i g_i(x) = f(x) - \boldsymbol{u}^T \mathbf{g}(x).$$

The KKT conditions are

1. For $i = 1, \ldots, m$
   (a) $g_i(x) \leqslant 0$
   (b) $u_i \geqslant 0$
   (c) $u_i g_i(x) = 0$
2. For $j = 1, \ldots, n$
$$\frac{\partial L}{\partial x_j} = 0.$$

Note:-
For equality constraints we drop the $u_i \geqslant 0$ and the $u_i g_i(x) = 0$ conditions.

If we have non-negativity constraints $x_i \geqslant 0$, we define the Lagrangian as

$$L(x, \lambda) = f(x) - \sum_i u_i x_i + \sum_j a_j x_j.$$

Where we substitute

$$a_j = -\frac{\partial f}{\partial x_j} + \sum_i u_i \frac{\partial g_i}{\partial x_j}.$$

The KKT conditions are

1. For $i = 1, \ldots, n$
   (a) $g_i \leqslant 0$
   (b) $u_i \geqslant 0$
   (c) $u_i g_i = 0$
2. For $j = 1, \ldots, n$
   (a) $\frac{\partial f}{\partial x_j} - \sum_i u_i \frac{\partial g_i}{\partial x_j} \leqslant 0$
   (b) $x_j(\frac{\partial f}{\partial x_j} - \sum_i u_i \frac{\partial g_i}{\partial x_j}) = 0$
   (c) $x_j \geqslant 0$

Problem Statement

$$\max Z = f(\mathbf{x}).$$

Subject to

$$g_i(\mathbf{x}) \leqslant b_i \quad i = 1, \dots, m$$
$$x_j \leqslant 0 \quad j = 1, \dots, n$$

We define the Lagrangian by adding m dual variables u_i

$$L(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) - \sum_{i=1} u_i(g_i(\mathbf{x}) - b_i).$$

The KKT conditions are

1. For $j = 1, \dots, m$
   (a) $\frac{\partial f}{\partial x_j} - \sum_i u_i \frac{\partial g_i}{\partial x_j} \leqslant 0$
   (b) $x_j(\frac{\partial f}{\partial x_j} - \sum_i u_i \frac{\partial g_i}{\partial x_j}) = 0$
   (c) $x_j \leqslant 0$
2. For $i = 1, \dots, m$
   (a) $g_i(\mathbf{x}) - b_i \leqslant 0$
   (b) $u_i \geqslant 0$
   (c) $u_i(g_i(\mathbf{x}) - b_i) = 0$

Note:-

The KKT conditions are necessary conditions for optimality. They are not sufficient conditions. They are only valid for local optima.
However, if f is concave and the constraints are convex, then the KKT conditions are also sufficient.

4.6 Convex Programming

A convex programming problem is a problem where the objective function and the constraints are convex. The KKT conditions are necessary and sufficient for optimality.

Note:-

Important properties of convex programming problems:

- Any local extremum is a global extremum.
- The KKT conditions are sufficient as well as necessary.

4.6.1 Quadratic Programming

A quadratic programming problem is a convex programming problem where the objective function is quadratic and the constraints are linear. The problem has the following shape

$$\mathbf{Ax} \leqslant \mathbf{b} \quad \mathbf{x} \geqslant 0.$$

Quadratic polynomials can be written as

$$\mathbf{cx} - \frac{1}{2}\mathbf{x}^T\mathbf{Qx}.$$

Where $\mathbf{Q}$ is the Hessian matrix of the quadratic function times -1.

$$\mathbf{Q} = -\nabla^2 f(\mathbf{x}).$$

Example 4.6.1

Consider the following quadratic programming problem

$$\max Z == 15x_1 + 30x_2 + 4x_1x_2 - 2{x_1}^2 - 4{x_2}^2.$$

Subject to

$$x_1 + 2x_2 \leqslant 30.$$

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \end{bmatrix}$$
$$\mathbf{b} = \begin{bmatrix} 30 \end{bmatrix}$$
$$\mathbf{c} = \begin{bmatrix} 15 \\ 30 \end{bmatrix}$$
$$\mathbf{Q} = \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix}$$

If Q is PSD (positive semi-definite), then the problem is convex. Any method would solve the problem however there exists a modified simplex method that is more efficient.

4.6.2 Modified Simplex Method

- Begin by writing down the KKT conditions.
- Add slack variables y_i, v_i for each inequality constraint.
- Add artificial variables z_i for feasibility.
- Solve the problem using the simplex method with the Restricted-Entry Rule which states that we should avoid having complimentary variables in the BFS. Always select an entering basic variable that its complementary variable is not in the base (not necessarily the one that has the highest increase rate).

Example 4.6.2

Consider the following quadratic programming problem

$$\max Z == 15x_1 + 30x_2 + 4x_1x_2 - 2{x_1}^2 - 4{x_2}^2.$$

Subject to

$$x_1 + 2x_2 \leqslant 30.$$

The KKT conditions are

$$L = 15x_1 + 30x_2 + 4x_1x_2 - 2{x_1}^2 - 4{x_2}^2 - \lambda_1(x_1 + 2x_2 - 30).$$

$$\begin{aligned} 15 + 4x_2 - 4x_1 - \lambda_1 &\leqslant 0 \\ 30 + 4x_1 - 8x_2 - 2\lambda_1 &\leqslant 0 \\ x_1 + 2x_2 - 30 &\leqslant 0 \\ \\ x_1(15 - 4x_1 + 4x_2 - \lambda_1) &= 0 \\ x_2(30 + 4x_1 - 8x_2 - 2\lambda_1) &= 0 \\ \lambda_1(x_1 + 2x_2 - 30) &= 0 \end{aligned}$$

Now we add slack variables y_1, v_1 for the inequality constraints

$$\begin{aligned} 4x_1 - 4x_2 + \lambda_1 - y_1 &= 15 \quad x_1 y_1 = 0 \text{ complementarity constraint} \\ -4x_1 + 8x_2 + 2\lambda_1 - y_2 &= 30 \quad x_2 y_2 = 0 \text{ complementarity constraint} \\ x_1 + 2x_2 + v_1 &= 30 \quad \lambda_1 v_1 = 0 \text{ complementarity constraint} \end{aligned}$$

One constraint summarizes all complementarity constraints

$$x_1 y_1 + x_2 y_2 + \lambda_1 v_1 = 0.$$

We add artificial variables z_1, z_2, z_3 for feasibility

$$\min z_1 + z_2.$$

$$\begin{aligned} 4x_1 - 4x_2 + \lambda_1 - y_1 + z_1 &= 15 \\ -4x_1 + 8x_2 + 2\lambda_1 - y_2 + z_2 &= 30 \\ x_1 + 2x_2 + v_1 &= 30 \end{aligned}$$