CHAPTER 2

Basic Concepts and Preliminaries

This chapter is organized as follows: Section 2.1 presents the basic concepts and properties of the optimization problem. Section 2.2 presents important methods for solving the optimization problem. Section 2.3 presents a classification of the location problems and provides a brief history of those location problems.

2.1 Basic Concepts

Theorem 2.1.1. ***(Unimodular matrix)*** *[19] Let A be a square integer matrix. A matrix A is unimodular if the determinant of matrix A has value -1, 0 or 1.*

Theorem 2.1.2. ***(Totally unimodular matrix)*** *[19] Let A be an integer matrix. A matrix A is totally unimodular if the determinant of every square submatrix of A (also called minor of matrix A) has value -1, 0 or 1.*

Theorem 2.1.3. ***(Convex set)*** *[19] A set $S \in \mathbb{R}^n$ is said to be covex if the closed line segment joining any points x_1 and x_2 of the set S, that is, $(1-\lambda)x_1 + \lambda x_2$, belongs to the set S for each $\lambda \in [0,1]$.*

Lemma 2.1.4. ***(Properties of convex sets)*** *[19] Let S_1 and S_2 be convex sets in $\mathbb{R}^n$. Then,*

1) The intersection $S_1 \cap S_2$ is a convex set.

2) The sum $S_1 + S_2$ of two convex sets is a convex set.

3) The product θS_1 of the real number θ and the set S_1 is a convex set.

Definition 2.1.1. (Convex combination) [19] Let $\{x_1, x_2, \ldots, x_r\}$ be any finite set of points in $\mathbb{R}^n$. A convex combination of this set is a point of the form:

$$\lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_r x_r,$$

$$\lambda_1 + \lambda_2 + \ldots + \lambda_r = 1,$$

$$\lambda_1, \lambda_2, \ldots, \lambda_r \geq 0.$$

Definition 2.1.2. (Convex hull) [19] Let S be a set (convex or nonconvex) in $\mathbb{R}^n$. The convex hull, $H(S)$, of S is defined as the intersection of all convex sets in $\mathbb{R}^n$ which contain S as a subset.

Definition 2.1.3. (Concave function) [19] Let S be a convex subset of $\mathbb{R}^n$, and $f(x)$ be a real valued function defined on S. The function $f(x)$ is said to be concave if for any $x_1, x_2 \in S$, and $0 \leq \lambda \leq 1$, we have

$$f[(1-\lambda)x_1 + \lambda x_2] \geq (1-\lambda)f(x_1) + \lambda f(x_2).$$

Definition 2.1.4. (Strictly concave function) [19] Let S be a convex subset of $\mathbb{R}^n$, and $f(x)$ be a real valued function defined on S. The function $f(x)$ is said to be strictly concave if for any $x_1, x_2 \in S$, and $0 < \lambda < 1$, we have

$$f[(1-\lambda)x_1 + \lambda x_2] > (1-\lambda)f(x_1) + \lambda f(x_2).$$

Definition 2.1.5. (Convex function) [19] Let S be a convex subset of $\mathbb{R}^n$, and $f(x)$ be a real valued function defined on S. The function $f(x)$ is said to be convex if for any $x_1, x_2 \in S$, and $0 \leq \lambda \leq 1$, we have

$$f[(1-\lambda)x_1 + \lambda x_2] \leq (1-\lambda)f(x_1) + \lambda f(x_2).$$

Definition 2.1.6. (Strictly convex function) [19] Let S be a convex subset of $\mathbb{R}^n$, and $f(x)$ be a real valued function defined on S. The function $f(x)$ is said to be strictly convex if for any $x_1, x_2 \in S$, and $0 < \lambda < 1$, we have

$$f[(1-\lambda)x_1 + \lambda x_2] < (1-\lambda)f(x_1) + \lambda f(x_2).$$

Definition 2.1.7. (Discrete convex function) [56] Let S be a subspace of discrete $n-$dimensional space. A function $f(x)$ is a discrete convex function if all $x_1, x_2 \in S$, and $0 \leq \lambda \leq 1$, we have

$$f[(1-\lambda)x_1 + \lambda x_2] \leq (1-\lambda)f(x_1) + \lambda f(x_2).$$

2.1.1 Properties of Convex and Concave Functions

Convex functions can be combined in a number of ways to produce new convex functions as illustrated by the following:

1) Let $f_1(x), f_2(x), \ldots, f_n(x)$ be convex functions on a convex subset S of $\mathbb{R}^n$. Then, their summation

$$f_1(x) + f_2(x) + \ldots + f_n(x)$$

is a convex. Furthermore, if at least one $f_i(x)$ is strictly convex on S, then their summation is strictly convex.

2) Let $f(x)$ be convex (strictly convex) on a convex subset S of $\mathbb{R}^n$, and λ is a positive real number. Then, $\lambda f(x)$ is convex (strictly convex).

3) Let $f(x)$ be convex (strictly convex) on a convex subset S of $\mathbb{R}^n$, and $g(x)$ be an convex function defined in the range of $f(x)$ in $\mathbb{R}$. Then, the composite function $g[f(x)]$ is convex (strictly convex) on S.

4) Let $f_1(x), f_2(x), \ldots, f_n(x)$ be convex functions and bounded above on a convex subset S of $\mathbb{R}^n$. Then, the pointwise supremum function

$$f(x) = \max\{f_1(x), f_2(x), \ldots, f_n(x)\}$$

is a convex function on S.

5) Let $f_1(x), f_2(x), \ldots, f_n(x)$ be concave functions and bounded below on a convex subset S of $\mathbb{R}^n$. Then, the pointwise infimum function

$$f(x) = \min\{f_1(x), f_2(x), \ldots, f_n(x)\}$$

is a concave function on S.

2.1.2 Feasible Solution, Local and Global Minimum

Consider the minimizing problem (P): $\min f(x)$
subject to $x \in S$.

Definition 2.1.8. (Feasible solution) [19] A point $x \in S$ is a feasible solution of problem (P).

Definition 2.1.9. (Local minimum) [19] Suppose that $x^* \in S$ and that there exists an $\epsilon > 0$ such that

$$f(x) \geq f(x^*) \qquad \forall x \in S : \|x - x^*\| < \epsilon$$

then x^* is a local minimum.

Definition 2.1.10. (Global minimum) [19] Suppose that $x^* \in S$ and

$$f(x) \geq f(x^*) \qquad \forall x \in S$$

then x^* is a global minimum.

Definition 2.1.11. (Unique global minimum) [19] Suppose that $x^* \in S$ and

$$f(x) > f(x^*) \qquad \forall x \in S$$

then x^* is the unique global minimum.

Theorem 2.1.5. *[19] Let S be a nonempty convex set in $\mathbb{R}^n$ and $x^* \in S$ be a local minimum.*

1) If $f(x)$ is convex, then x^* is a global minimum.

2) If $f(x)$ is strictly convex, then x^* is a unique global minimum.

2.1.3 Network Theory

Let $G = (V, E, W)$ be a weighted network with a node (vertex) set V, an edge (arc) set E, and the weight set W which specifies weights $w_{i,j}$ for the edges $(i, j) \in E$. If i and j are nodes of V, then an edge of the form (i, j) or (j, i) is said to join i and j.

Definition 2.1.12. (Simple graph) Two or more edges joining the same pair of nodes are called multiple edges, and an edge joining a node to it self is call a loop. A graph with no loops and no multiple edges is called a simple graph.

Definition 2.1.13. (Degree of vertex) Let $v \in V$. The number of all nodes which are adjacent to the node v is called the degree of v, denote by $\deg(v)$.

Definition 2.1.14. (Complete graph) A graph $G = (V, E, W)$ is said to be complete if every two distinct nodes of G are adjacent. A complete graph of order n is denoted by K_n. Therefore, K_n has the maximum possible size for a graph with n nodes.

Definition 2.1.15. (Connected graph) A graph $G = (V, E, W)$ is said to be connected if every two distinct nodes of G, there exist a sequence $a_1, a_2, \ldots, a_n$ in V such that $a_1 = u, a_2 = v$ and $(a_i, a_{i+1}) \in E$ for all $i = 1, 2, \ldots, n-1$.

Definition 2.1.16. (Path and trail) A walk consists of an alternating sequence of nodes and edges, the consecutive elements of which are incidental and begins and ends with a node. A trail is a walk without repeated edges. A path is a walk without repeated nodes.

Theorem 2.1.6. ***(Tree)*** *[10] A graph $G = (V, E, W)$ is a tree if and only if every two nodes of G are connected by a unique path.*

Shortest Path Problem

The shortest path problem is the problem of determining the shortest path from the origin node $o \in V$ to the destination node $d \in V$. The shortest path problem model can be formulated as follows:

$$\min \sum_{i \in V} \sum_{j \in V} w_{i,j} x_{i,j} \tag{2.1}$$

$$s.t. \quad \sum_{j \in V - \{o\}} x_{o,j} = 1, \qquad \text{origin node } o \in V, \tag{2.2}$$

$$\sum_{j \in V} x_{j,i} - \sum_{j \in V} x_{i,j} = 0, \qquad \forall i \in V - \{o, d\}, \text{ intermediate nodes}, \tag{2.3}$$

$$\sum_{j \in V - \{d\}} x_{j,d} = 1, \qquad \text{destination node } d \in V, \tag{2.4}$$

$$x_{i,j} \in \{0, 1\}, \qquad \forall i, j \in V, \tag{2.5}$$

where $x_{i,j} = 1$ if the edge (i, j) is on the shortest path, $x_{i,j} = 0$ otherwise. The objective (2.1) is to find the shortest path between the origin and the destination nodes. The first constraint (2.2) is to guarantee that the origin node has only 1 edge out. The second constraint (2.3) is to ensure that the edge in and edge out of the intermediate node $i \in V - \{o, d\}$ must be equal. The last constraint (2.4) is to guarantee that the destination node has only 1 edge out. The important algorithm for solving this solution is Dijkstra's algorithm.

Maximum Flow Problem

The maximum flow problem is a problem of the network with flow capacities on the edges. Addressing this problem involves finding the maximum capacity flow from the origin node (source) to the destination node (sink). Each edge $(i, j) \in E$ has the capacity $c_{i,j}$ that is the maximum flow that can traverse (in either direction). If the edge $(i, j) \in E$ has unlimited capacity, $c_{i,j} = +\infty$. The maximum flow problem model can be formulated

as follows:

$$\max \quad f \tag{2.6}$$

$$s.t. \qquad f - \sum_{j \in V - \{o\}} x_{o,j} = 0, \qquad \text{origin node } o \in V, \tag{2.7}$$

$$\sum_{j \in V} x_{j,i} - \sum_{j \in V} x_{i,j} = 0, \qquad \forall i \in V - \{o, d\}, \text{ intermediate nodes}, \tag{2.8}$$

$$-f + \sum_{j \in V - \{d\}} x_{j,d} = 0, \qquad \text{destination node } d \in V, \tag{2.9}$$

$$0 \leq x_{i,j} \leq c_{i,j}, \qquad \forall i, j \in V, \tag{2.10}$$

where $x_{i,j} \in R$ is amount of flow from node i to node j. The objective (2.6) is to find the maximum capacity f flow from the origin node (source) $o \in V$ to the destination node (sink) $d \in V$. The first constraint (2.7) is to guarantee that the total flow f out of o must use the edges of the form (o, j). The second constraints (2.8) are to ensure that the flow in and flow out of the intermediate node $i \in V - \{o, d\}$ must be equal. The last constraint (2.9) is to guarantee that the total flow f in d must use the edges of the form (j, d).

2.1.4 Integer Linear Programming

Integer linear programming (ILP) deals with the class of the optimization problems in which some or all variables are integers. In this thesis, we focus on integer linear programing in which all variables are integers. The standard form of integer linear programing can be formulated as follows:

$$\text{(ILP):} \quad \min c^T x \tag{2.11}$$

$$s.t. \qquad Ax \leq b, \tag{2.12}$$

$$x \in (\mathbb{Z}^*)^n, \tag{2.13}$$

where $\mathbb{Z}^* = \mathbb{Z}^+ \cup 0$, $x \in (\mathbb{Z}^*)^n$ is a vector of the n variables, $c \in \mathbb{R}^n, b \in \mathbb{R}^m$ are vectors of the coefficients, and $A \in \mathbb{R}^{m \times n}$ is a matrix of the coefficients.

Integer linear programing is proved to be an NP-hard problem but it has also been proved that some special structures of problems can be solved in polynomial time. One of the most interesting matrix structures is a totally unimodular matrix. For problems with a totally unimodular constrained matrix, we can find the integer optimal solution by solving its LP relaxation. For example, when the constrained matrix of the assignment problem is totally unimodular, the problem can be solved using a linear programming technique.

The assignment problem (AP) consists of n jobs and n agents which are available for carrying out the jobs. Each agent can carry out exactly one job, and each agent has different levels of efficiency in carrying out the jobs. How should we assign the agents to the jobs in order to maximize overall efficiency?

A model of assignment problem can be formulated as follows:

$$\text{(AP):} \quad \max \sum_{i\in I}\sum_{j\in J} p_{i,j}x_{i,j} \tag{2.14}$$

$$s.t. \quad \sum_{j\in J} x_{i,j} = 1, \qquad \forall i \in I, \tag{2.15}$$

$$\sum_{i\in I} x_{i,j} = 1, \qquad \forall j \in J, \tag{2.16}$$

$$x_{i,j} \in \{0,1\}, \qquad \forall i \in I, j \in J, \tag{2.17}$$

where $I = \{1, 2, 3, \ldots, n\}$ is a set of agents, $J = \{1, 2, 3, \ldots, n\}$ is a set of jobs, $p_{i,j}$ is the efficiency of the agent, i does job j, and $x_{i,j}$ is the decision variable, taking value 1 if the agent i assigned to job j, and 0 otherwise.

The objective of function (2.14) is to maximize the total efficiency. The constraints (2.15) ensure all agents must be assigned to certain job. The constraints (2.16) guarantee that all jobs must be assigned by some agent. The matrix form of the assignment problem can be considered as follows:

$$\text{(AP):} \quad \max PX \tag{2.18}$$

$$s.t. \quad AX = b, \tag{2.19}$$

$$X \in \{0,1\}^{n^2} \tag{2.20}$$

where

$$A = \left[\begin{array}{cccc|cccc|c|cccc} 1 & 1 & \cdots & 1 & & & & & & & & & \\ & & & & 1 & 1 & \cdots & 1 & & & & & \\ & & & & & & & & \cdots & & & & \\ & & & & & & & & & 1 & 1 & \cdots & 1 \\ \hline 1 & & & & 1 & & & & & 1 & & & \\ & 1 & & & & 1 & & & & & 1 & & \\ & & \cdots & & & & \cdots & & \cdots & & & \cdots & \\ & & & 1 & & & & 1 & & & & & 1 \end{array}\right]_{n+n,n^2}, b = \left[\begin{array}{c} 1 \\ 1 \\ \cdots \\ 1 \end{array}\right]_{n+n,1},$$

$$P = [p_{1,1}, p_{1,2}, \ldots, p_{1,n}, p_{2,1}, p_{2,2}, \ldots, p_{2,n}, \ldots, p_{n,1}, p_{n,2}, \ldots, p_{n,n}],$$

$$X = [x_{1,1}, x_{1,2}, \ldots, x_{1,n}, x_{2,1}, x_{2,2}, \ldots, x_{2,n}, \ldots, x_{n,1}, x_{n,2}, \ldots, x_{n,n}]^T.$$

The matrix A is a totally unimodular matrix.

Another interesting integer linear problem is the knapsack problem. The objective of the knapsack problem is to choose items such that the total size of the chosen items is smaller than or equal to the knapsack capacity and the profit of the chosen items are maximized. The following notations have been introduced to formulate the mathematical model. Let

$N = \{1, 2, 3, \ldots, n\}$ be a set of items,

s_i be the size of item i where s_i is a positive constant number,

p_i be the profit of item i where p_i is a positive constant number,

C be the knapsack capacity,

$$x_i = \begin{cases} 1 & \text{if item } i \text{ should be included in the knapsack, and} \\ 0 & \text{otherwise.} \end{cases}$$

The knapsack problem model can be formulated as follows:

$$\max \quad \sum_{i=1}^{n} p_i x_i \tag{2.21}$$

$$s.t. \quad \sum_{i=1}^{n} s_i x_i \leq C, \tag{2.22}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in N. \tag{2.23}$$

The objective of function (2.21) is to maximize the total profit. The constraint (2.22) is a capacity restriction.

2.2 Methods

This section presents important methods for solving the optimization problem.

2.2.1 Simplex Method

The simplex method is a procedure that is used to solve linear programming problems. Developed by George Dantzig in 1947, it has proven to be a remarkably efficient method that is used routinely to solve huge problems on today's computers. Before starting, we need to highlight the point that every linear program can be converted into

standard form

$$\begin{aligned}
\max \quad & z = c_1x_1 + c_2x_2 + \dots c_nx_n \\
s.t. \quad & a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n = b_1, \\
& a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,n}x_n = b_2, \\
& \dots \\
& a_{m,1}x_1 + a_{m,2}x_2 + \dots + a_{m,n}x_n = b_m, \\
& x_1, x_2, \dots, x_n \geq 0.
\end{aligned}$$

If the problem is $\min z$, convert it to $\max -z$.

If a constraint is in less than or equal form $(a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n \leq b_1)$, convert it into standard form by adding a nonnegative slack variable s_i to the left hand side of inequality $(a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n + s_i = b_1)$.

If a constraint is in greater than or equal form $(a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n \geq b_1)$, convert it into standard form by subtracting a nonnegative surplus variable s_i to the left hand side of inequality $(a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n - s_i = b_1)$.

If some variable x_i is unrestricted in sign, replace it everywhere in the formulation by $x_i^* - x_i^{**}$, where $x_i^*, x_i^{**} \geq 0$.

Any linear programming problem can have a feasible solutions and a bounded feasible region. The problem must posses corner point feasible solutions (a corner point feasible solution is a solution that lies at a corner of the feasible region) and at least one optimal solution. Furthermore, the best corner point feasible solution must be an optimal solution. Thus, if a problem has exactly one optimal solution, it must be a corner point feasible solution. If the problem has multiple optimal solutions, at least two must be corner point feasible solutions. The simplex method focuses solely on corner feasible solutions. This method is an iterative algorithm with the following structure.

Step 1: Find the initial corner point feasible solution.

Step 2: If the current corner point feasible solution is an optimal solution, stop. Else, go to Step 3.

Step 3: Find a better corner point feasible solution.

2.2.2 Linear Programing Relaxation

Linear programming relaxation (LP relaxation) is use to relax the integer programing problem to the Linear programing problem by considering the integer variable to real number. For the example, the binary variable x_i of AP will be relax to $0 \leq x_i \leq 1$. The

resulting of this relaxation is a linear program. Note that the optimal solution to the LP relaxation is not necessarily integer. However, the feasible region of the LP relaxation is larger than the feasible region of the integer programing problem. This implies that the optimal value of the LP relaxation is a lower bound of the optimal value of original integer programing problem.

2.2.3 Lagrangian Relaxation

Consider the following general optimization problem, which we call the primal problem.

$$\begin{aligned} \max \quad & f(x) \\ s.t. \quad & g_i(x) \geq b_i, \qquad i = 1, 2, \ldots, m, \\ & x \in X, \end{aligned}$$

where the functions $f(x)$, and $g_i(x),\ i = 1, 2, .., m$ can be arbitrary functions. The feasible region of the problem consists of explicit constraints (including nonnegativity and integrality restrictions) which are represented by a set X. We assume that the problem would be easy to solve in the absence of constraints. Lagrangian relaxation relaxes constraints $g_i(x) \geq b_i,\ i = 1, 2, ..., m$ by moving that constraints to the objective function with associated Lagrange multipliers $u_i \geq 0,\ i = 1, ..., m$ which results in the following:

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

2.2.4 Branch and Bound

The basic concept of the branch and bound method for solving the integer problem is to divide (branching) and conquer (fathoming). Since the original problem is too difficult to be solved directly, it is divided into smaller subproblems so that these subproblems can be conquered. A general branch and bound method for solving the integer problem can be stated as follows:

Step 1: List of candidate subproblems that consist of the integer problem alone, and set $z^* = \infty$.

Step 2: If the list of candidate subproblems is empty, then the optimal solution is the current incumbent. If an incumbent does not exist, then the integer problem is infeasible.

Step 3: Select one of the subproblems in the candidate list to become the current candidate subproblem.

Step 4: Solve a relaxed current candidate subproblem (e.g. LP relaxation) and denote its solution by z_{RCS}.

Step 5: Apply the three fathoming criteria:

1) If the relaxed current candidate subproblem has no feasible solution; go to Step 2.

2) If $z_{RCS} \geq z^*$, then the current candidate subproblem has no feasible solution better than the incumbent; go to Step 2.

3) If the optimal solution of the relaxed current candidate subproblem is feasible for the current candidate subproblem (e.g. integral), then it is an optimal of the current candidate subproblem. If $z_{RCS} \leq z^*$, then record this solution as the new incumbent; that is, $z^* = z_{RCS}$. Go to Step 2.

Step 6: Separate the current candidate subproblem and add its children nodes to the list of candidate subproblems. Go to Step 2.

An illustration of the search space of the branch and bound algorithm is shown in Figure 2.1. For more information about this method can read in [19], [30].

2.2.5 Cutting Plane

The cutting plane method is a technique used to solve the integer linear programing by modifying the linear programming solution until the integer solution is obtained. It does not partition the feasible region into subdivisions, as in branch and bound approaches, but instead works with a single linear program, which it refined by adding new constraints. The new constraints successively reduce the feasible region until an integer optimal solution is found. This method was proposed by Gomory in the 1950's. The constraints of the LP problem in basic canonical form are written as:

$$\sum_{j \in N} a_{i,j} x_j + x_i^* = b_i, \quad \forall i = 1, 2, \ldots, m, \ a_{i,j} \in R$$

where N is a set of nonbasic variables, and x_i^* is a basic variable.

We can obtain another constraint, which preserves all the integer feasible solutions, through two steps:

1) The constraint rounding down its coefficients

$$\sum_{j \in N} \lfloor a_{i,j} \rfloor x_j + x_i^* = b_i, \quad \forall i = 1, 2, \ldots, m.$$

2) The constraint rounding down the right-hand-side

$$\sum_{j \in N} a_{i,j} x_j + x_i^* = \lfloor b_i, \rfloor \quad \forall i = 1, 2, \ldots, m.$$

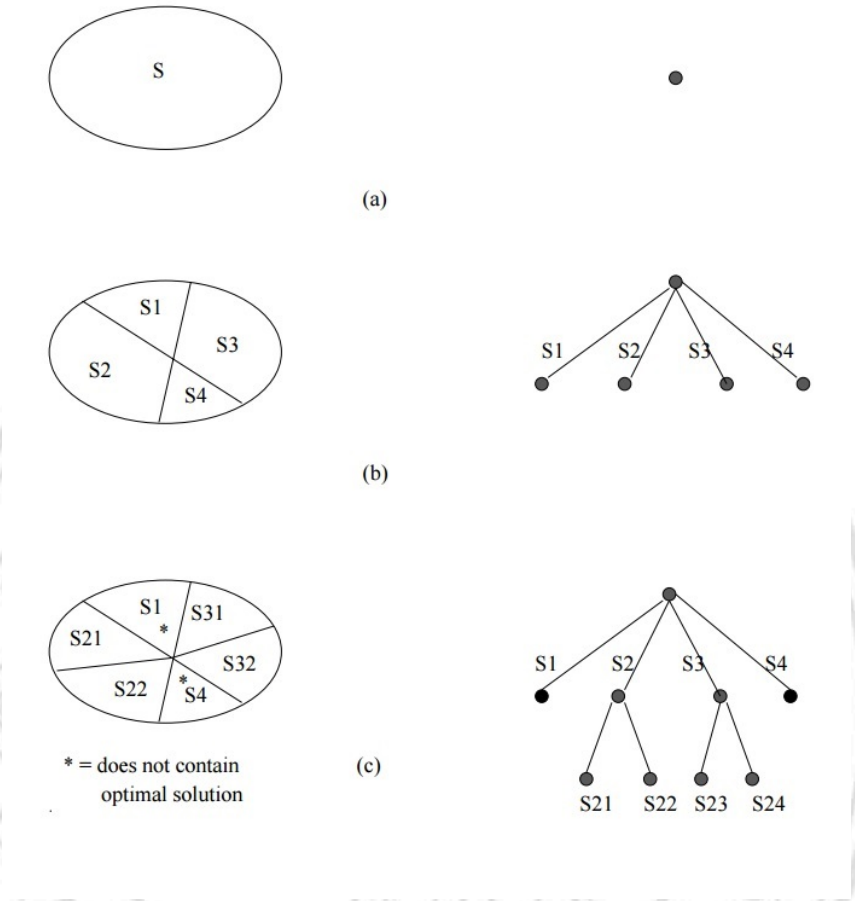

Figure 2.1: Illustration of the search space of the branch and bound algorithm.

2.2.6 Branch and Cut

A branch and cut algorithm is a combination of a branch and bound algorithm and a cutting plane method. A general branch and cut algorithm method for solving an integer problem can be stated as follows:

Step 1: Define a list of candidate subproblems to consist of the integer problem alone, and set $z^* = \infty$.

Step 2: If the list of candidate subproblems is empty, then the optimal solution is the current incumbent. If an incumbent does not exist, then the integer problem is infeasible.

Step 3: Select one of the subproblems in the candidate list to become the current candidate subproblem.

Step 4: Solve the relaxed current candidate subproblem (e.g. LP relaxation) and

denote its solution by z_{RCS}. If the relaxed current candidate subproblem has no feasible solution; go to Step 2.

Step 5: If the optimal solution of the relax current candidate subproblem is infeasible for the current candidate subproblem, add new constraints to the relax current candidate subproblem and go to Step 4.

Step 6: If $z_{RCS} \leq z^*$, then record this solution as the new incumbent; that is, $z^* = z_{RCS}$. Go to Step 2.

2.2.7 Dijkstras Algorithm

Dijkstras algorithm is applied to find the length of all the shortest paths in a directed graph with non-negative weights on the edges, from a source vertex s to every other vertex v_i in the graph. We define the distance d_i to be the length of the shortest path from s to vertex v_i. Dijkstras algorithm maintains a set of vertices $S \subseteq V$, with two properties. Firstly, S is a set of vertices in the graph nearest to s; that is:

$$\forall v_i \in S,\ \forall v_j \in V - S,\ d_i \leq d_j.$$

And secondly, for all vertices $v_j \in S$, there is a shortest path from s to v_j using only vertices of S as intermediates. There might be several different choices for S; Dijkstras algorithm chooses one arbitrarily.
For each outside vertex $d_j \in V - S_k$, we define an estimated distance:

$$d_j^{est} = \min_{v_i \in S} d_i + w_{ij}.$$

Since the estimated distance is the length of some path to d_j, it is upper bound on the length of the true shortest path from s.

2.3 Location Problems

In this section, we classify location problems related to the objective functions as follows: 1) covering problems, 2) p-median problems, 3) p-center problems and 4) facility location problems. We define the notation to be used in this thesis as follows:

Let $I = \{1, 2, 3, \ldots, n\}$ be a set of clients or customers,

$J = \{1, 2, 3, \ldots, m\}$ be a set of potential facility sites,

f_j be facility setup cost for facility $j \in J$,

h_i be supply of client $i \in I$,

s_j be capacity of facility $j \in J$,

$d_{i,j}$ be the distance from client $i \in I$ to facility $j \in J$. We assume $I \cup J$ is a node set of a complete graph, and $d_{i,j}$ is the minimum distances (or shortest path) between facility j and client i,

$c_{i,j}$ be the transportation cost from client $i \in I$ to facility $j \in J$. We assume $I \cup J$ is a node set of a complete graph, and $c_{i,j}$ is the minimum transportation cost between facility j and client i,

β be the first feasible or a minimum number of opened facilities,

γ be a maximum number of opened facilities,

$$x_{i,j} = \begin{cases} 1 & \text{if client } i \text{ is assigned to facility } j, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{if facility } j \text{ is opened, and} \\ 0 & \text{otherwise.} \end{cases}$$

2.3.1 Covering Problem

The covering problem is one of the most popular forms of location problem. This problem is used to find the optimum place to locate a facility in relation to the customers served, based on the idea that the distance from customer to facility is equal or less than the coverage distance or coverage radius. This problem can be classified into two major types according to the objective. If the objective of the problem tries to minimize the location cost by satisfying a specified coverage distance, this problem is called the set covering problem (SCP). Another type is a maximal covering location problem (MCLP). The objective of this type is to maximize the amount of demand covered within the coverage distances by locating a given fixed number of opened facilities.

The mathematical model of the set covering problem can be formulated as follows:

$$\text{(SCP):} \quad \min \sum_{j \in J} f_j y_j \tag{2.24}$$

$$s.t. \quad \sum_{j \in J} x_{i,j} = 1, \qquad \forall i \in I, \tag{2.25}$$

$$d_{i,j} x_{i,j} \leq \delta y_j, \qquad \forall i \in I, j \in J, \tag{2.26}$$

$$x_{i,j} \in \{0,1\}, \qquad \forall i \in I, j \in J, \tag{2.27}$$

$$y_j \in \{0,1\}, \qquad \forall j \in J, \tag{2.28}$$

where δ is a given coverage radius.

The objective function (2.24) is to minimize the total cost of locating the facilities. The constraints (2.25) ensure that all clients must be assigned to some facility. The constraints (2.26) guarantee that a client must be assigned to an opened facility such that the distance from client i to facility j is less than or equal to the coverage radius δ.

The mathematical model above considers that all customers have the same demand size and that all facilities have no limitation of capacity. However, most real-world applications consider facilities to have limitations of capacity. Current and Storbeck [14] proposed a capacitated version of a set covering problem. They added capacities restriction to the SCP model by adding (2.29) to the SCP model.

$$\sum_{i \in I} h_i x_{i,j} \leq s_j y_j, \qquad \forall\ j \in J. \tag{2.29}$$

Moreover, they also proposed a capacitated maximal covering location problem. The mathematical model of the capacitated maximal covering problem can be formulated as follows:

$$\text{(MCLP):} \quad \max \sum_{i \in I} h_i z_i \tag{2.30}$$

$$s.t. \quad z_i \leq \sum_{j \in J} x_{i,j}, \qquad \forall i \in I, \tag{2.31}$$

$$\sum_{j \in J} y_j = p, \tag{2.32}$$

$$d_{i,j} x_{i,j} \leq \delta y_j, \qquad \forall i \in I, j \in J, \tag{2.33}$$

$$x_{i,j} \in \{0,1\}, \qquad \forall i \in I, j \in J, \tag{2.34}$$

$$y_j \in \{0,1\}, \qquad \forall j \in J, \tag{2.35}$$

$$z_i \in \{0,1\}, \qquad \forall i \in I, \tag{2.36}$$

where z_i is a binary decision variable; it is equal to 1 if the client i is covered by some opened facility within the coverage radius, it is equal to 0 otherwise, p is the given number of facilities that can be open and δ is a given coverage radius.

The objective of function (2.30) is to maximize the coverage demands of the clients. Constraint (2.31) guarantees that z_i will be equal to 0 if a node i is not covered. Constraint (2.32) requires that the exact p facilities must be opened. The constraints (2.33) ensure that a client must be assigned to a facility such that the distance from client i to facility j is less than or equal to the coverage radius δ.

Since a covering problem is an NP hard problem [21], many methods have been developed in order to solve this problem. Next we will summarize certain methods to solve a covering problem.

In 1974, Church and Revelle [13] described an uncapacitated maximal covering problem. They solved the problem by using a branch and bound method. For this, they proposed both greedy adding and greedy adding with substitution methods. After that, Bazaraa and Goode [4] proposed a quadratic set covering problem. They change the locating facility cost $f_j y_j$ in objective function (2.24) to the cost of the relationship between each pair of the facilities $y_i f_j y_j$ but they did not explain the application of their model. This problem is a non-linear binary model. They solved the problem by using a cutting plane method.

A three stage method to solve a set covering problem was proposed by Beasley [5]. This method consists of a dual ascent procedure, sub-gradient and solve the dual problem of the LP relaxation problem. This method can be used to solve problems that include up to 400 candidate nodes and 4,000 covered nodes. After that, Beasley [6] developed a heuristic method for a set covering problem by using the Lagrangian relaxation and sub-gradient optimization method. This method can solve the problem for up to 1,000 candidate nodes and 10,000 covered nodes. Later, Lee and Lee [39] proposed a hierarchical covering location model. This problem is a partial coverage problem. They proposed a two phase method for a hierarchical covering location problem. The first phase is used to find a good initial solution through the use of a hierarchical allocation method. The second phase is used to improve the solution by using a tabu search and an allocation heuristic method.

Some researchers have studied set covering problems by transforming them to another problem. For example, Afif et al. [1] proposed a transformation approach to convert a set covering problem to a maximum flow problem. They used the Ford and Fulkerson algorithm to solve the maximum flow problem. This algorithm is a polynomial time algorithm.

2.3.2 P-Center Problem

The p-center problem (PC) is a class of the location problem that has a specific objective function of minimizing the maximum distance between each client and the facility it is assigned to, providing that the number of opened facilities are not greater than p. This problem can be partitioned into the uncapacitated and capacitated cases. The uncapacitated case is a basic p-center problem that does not include the demands of the clients and the capacities of the facilities. In the capacitated case, each client has a certain demand to meet and the facilities have certain capacity restrictions i.e. the total demands

of the clients assigned to a facility cannot exceed the facility's capacity. The p-center problem has been studied together with the covering and p-median problems.

The model of the uncapacitated p-center problem (UPC) can be formulated as follows:

$$\text{(UPC):} \quad \min \ w \tag{2.37}$$

$$s.t. \quad d_{i,j}x_{i,j} \leq w, \qquad \forall i \in I, \forall j \in J \tag{2.38}$$

$$\sum_{j \in J} y_j \leq p, \tag{2.39}$$

$$\sum_{j \in J} x_{i,j} = 1, \qquad \forall i \in I, \tag{2.40}$$

$$x_{i,j} \leq y_j, \qquad \forall i \in I, \ j \in J \tag{2.41}$$

$$x_{i,j} \in \{0,1\}, \qquad \forall i \in I, \ j \in J, \tag{2.42}$$

$$y_j \in \{0,1\}, \qquad \forall j \in J, \tag{2.43}$$

where p is the limited number of facilities that can be opened, w in the objective function (2.37) and the first constraints (2.38) are the maximum distance between a client and the facility it is assigned to.

The objective of function (2.37) is to minimize the maximum distance between each client and the facility it is assigned to. The first constraints (2.38) are to ensure that w is the maximum distance between the client and the facility it is assigned to. The second constraint (2.39) is to guarantee that at most p facilities can be open. The third constraints are to ensure that each client is assigned to some facility. The last constraints (2.41) are to guarantee that the client is assigned to the opened facility.

The model of the capacitated p-center problem (CPC) can be formulated as the UPC model with the addition of the capacity restriction to the UPC model by changing constraints (2.41) into

$$\sum_{i \in I} h_i x_{i,j} \leq s_j y_j, \qquad \forall \ j \in J. \tag{2.44}$$

The uncapacitated p-center problem has been proposed by Hakimi [27]. He wanted to find the optimum location to build the switching center between a communication network and police station in a highway system. He formulated the problem by considering a finite graph G with V being the set of nodes in graph G. He found the location for the police station P such that P is a subset of V, $|P| = p$ and the maximum distance between all nodes in P and V is at a minimum.

Since this problem is an NP hard problem [21, 36] many researchers have proposed methods to solve this problem. Next we will summarize the methods to solve the p-center problem. An $O(n)$ time algorithm for the 1-center problem was presented in [45]. He considered this problem as a tree graph and found the node which was nearest the center of the graph. After that, Hochbaum and Shmoys [31] proposed a heuristic for the uncapacitated p-center problem as follows. Initially, all distances in a graph are sorted in nondecreasing order. An edge with the minimum distances in a graph after removing all edges with higher distances would be fewer than p.

In 1970, Minieka [41] first used a series of set covering problems to solve uncapacitated p-center problems. The set covering problem is solved by giving the coverage radius. If a set covering problem can be the coverage by the p opened facilities, the minimum coverage radius is the solution of the p-center problem. Later, Mark and Daskin [40] proposed a modification using the maximum cover version of this approach. After that, Ilhan and Pinar [33] applied this idea to a method, under which at each step, a cover distance was chosen and increased until all clients can be covered within this distance by using it at most p facilities.

A polynomial exact algorithm for the capacitated p-center problem in tree networks has been developed by Jaeger and Goldberg [34]. The method described in the article solves set-covering subproblems on trees with a polynomial algorithm. Blaser [7] designed an exact algorithm for the p-partial vertex cover problem and showed that this problem can be solved in polynomial time for a certain p.

In 2006, Ozsoy and Pinar [46] used [33] idea to solve the capacitated case by labeling the quantity of demand to the clients and the capacity of the facility. Then, Maria Albareda [2] improved their result by considering only clients in a given radius and improved the result using Lagrangian relaxation. Later, Dantrakul and Likasiri [15] develop a maximal client coverage algorithm to solve the p-center problem by combining the set covering problem, greedy algorithm and bisection method. The algorithms are given for both uncapacitated and capacitated cases.

2.3.3 P-Median Problem

The p-median problem is a problem involved with locating p facilities such that the sum of the distance or transportation cost from each client to its assigned facility is minimized. This problem has also been proposed by Hakimi [26, 27]. The model of

p-median problem can be formulated as follows:

$$\text{(UPM):}\ \min \sum_{i\in I}\sum_{j\in J} c_{i,j}x_{i,j} \tag{2.45}$$

$$s.t. \quad \sum_{j\in J} y_j \le p, \qquad \forall j \in J, \tag{2.46}$$

$$\sum_{j\in J} x_{i,j} = 1, \qquad \forall i \in I, \tag{2.47}$$

$$x_{i,j} \le y_j, \qquad \forall i \in I,\ j \in J, \tag{2.48}$$

$$x_{i,j} \in \{0,1\}, \qquad \forall i \in I, j \in J, \tag{2.49}$$

$$y_j \in \{0,1\}, \qquad \forall j \in J. \tag{2.50}$$

The objective of function (2.45) is to minimize the total transportation cost. Constraints (2.46) ensure that most p facilities can be opened. The second (2.47) constraints are to ensure that each client is assigned to some facility. The last constraints (2.48) are to guarantee that the client is assigned to the opened facility.

The model of the capacitated p-median problem (CPM) can be formulated as the UPM model, with the addition of capacities restriction to the UPM model by changing constraints (2.48) into

$$\sum_{i\in I} h_i x_{i,j} \le s_j y_j, \qquad \forall j \in J. \tag{2.51}$$

Many heuristic and exact methods have been proposed to solve this problem. Next we will summarize the methods developed to solve the p-median problem.

Kuehn and Hamburger [37] proposed a greedy heuristic to solve the p-median problem. This method starts with an empty set of open facilities. Next, the facilities with the most reduced costs are added one by one until the number of opened facilities is equal to p.

The stingy heuristic has been proposed by Feldman et al. [18]. This method is similar to the greedy heuristic. It starts with all facilities opened, and then the facilities with the least increase costs are removed one by one until the number of opened facilities is equal to p. Later, Salhi and Atkinson [50] modified the stingy heuristic by starting with a subset of the candidate facilities.

A Lagrangian heuristic for solving the p-median problem has been proposed by Galvao [20]. He relaxed the constraints (2.47) of UPM by using Lagrangian multipliers. The problem after relaxing can be formulated as follows:

$$\min \sum_{i\in I}\sum_{j\in J} (c_{i,j} - u_i)x_{i,j} + \sum_{i\in I} u_i, \tag{2.52}$$

s.t. (2.46), (2.48), (2.49) and (2.50), where u_i are called Lagrangian multipliers. Next, he gave the value of Lagrangian multipliers u_i and solved the problem to find $x_{i,j},\ y_j$. However, this method cannot guarantee a feasible solution $x_{i,j},\ y_j$.

A tabu search for solving the p-median problem has been proposed by Glover [22, 23]. The tabu search is an improved version of the local search. The main idea is similar to that of the local search but with the addition of some rules that exclude certain structures (the so-called "forbidden solution"), known as the tabu list.

Typically, a local search heuristic starts with any feasible solution, and improves the solution at each iteration. If the solution cannot be improved, that solution is called the local minimum solution. At each step, it considers only local operations that can improve the cost of the solution. The local search heuristic for facility location was proposed by Kuehn and Hamburger in 1963 [37].

To escape local optima, Hansen and Mladenovic [29] present a variable neighborhood search algorithm for solving the p-median problem. The algorithm performs an intensive local search (similar to the interchange algorithm outlined above) on the current solution until it settles in a local optimum. It then searches by randomly selecting a solution from a neighborhood at a distance of k from the current best solution. The process continues, incrementing k, until some exogenously specified maximum value of k is attained. The algorithm compares very well with conventional heuristics as well as the enhancements provided by the tabu search.

2.3.4 Facility Location Problem

The facility location problem is a basis model that has been used in supply chain design. The objective of this problem is to find the optimal place to locate the facilities and assign clients to facilities, with the aim of minimizing the total cost. This total cost consists of the transportation cost that is incurred in traveling between the facilities and the clients and the cost of locating the facilities. This problem is similar to the p-median problem but does not limit the number of locating facilities, and does consider the cost

of locating facilities. The facility location problem can be formulated as follows:

$$\text{(UFP):}\ \min \sum_{j\in J} f_j y_j + \sum_{i\in I}\sum_{j\in J} c_{i,j}x_{i,j} \tag{2.53}$$

$$s.t. \qquad \sum_{j\in J} x_{i,j} = 1, \qquad \forall i \in I, \tag{2.54}$$

$$x_{i,j} \le y_j, \qquad \forall i \in I,\ j \in J, \tag{2.55}$$

$$x_{i,j} \in \{0,1\}, \qquad \forall i \in I, j \in J, \tag{2.56}$$

$$y_j \in \{0,1\}, \qquad \forall j \in J. \tag{2.57}$$

The objective function (2.53) is to minimize the total cost. Constraints (2.54) ensure that each client is assigned to some facility. Constraints (2.55) are also utilized to guarantee that the client is assigned to the opened facility.

The model of the capacitated facility location problem (CFP) can be formulated as the UFP model with the addition of capacity restriction to the UFP model by changing the constraints (2.55) into

$$\sum_{i\in I} h_i x_{i,j} \le s_j y_j, \qquad \forall j \in J. \tag{2.58}$$

Note that some studies have modified the uncapacitated case by using CFP by considering $h_i = 1,\ \forall i \in I$ and $s_j = n,\ \forall j \in J$.

There are many ways to solve facility location problems including the dual-based algorithm proposed by Erlenkotter [17]. The main idea of this algorithm is to use the dual problem of the LP relaxation to find a bound for the primal objective function.

Other popular methods include local search, greedy heuristics, tabu search, neighborhood search, etc. The main concept of the above methods is similar to the methods for covering, p-median and p-center problems.

In addition to the above-mentioned scenarios, some researchers have studied facility location problems by making some specific assumptions. For example, a general setup cost function for the uncapacitated facility location problem has been described [25] with the conclusion that if this function is concave, it can be solved in polynomial time. The general setup cost function for the capacitated facility location problem has also been studied [55].

Next, we will describe the method applied to solve the location problem in Table 2.1.

Table 2.1: List of literature on covering, p-center, p-median and facility location problems.

Author name	year	study area	algorithm
Hakimi [27]	1964	p-center, p-median	-
Hakimi [26]	1965	p-center, p-median	-
Minieka [41]	1970	p-center	series of sets covering problem
Toregas et al. [53]	1971	p-center	-
Church and Revelle [13]	1974	maximal covering	greedy
Bazaraa and Goode [4]	1975	quadratic set covering	cutting plane
Erlenkotter [17]	1978	facility location	dual-based
Kariv and Hakimi [36]	1979	p-center	-
Gonzalez [24]	1985	p-center	greedy
Hochbaum and Shmoys [31]	1985	p-center	graph removed
Beasley [5]	1987	set covering	tree search
Garey and Johnson [21]	1990	p-median	-
Olariu [45]	1990	1-center	tree networks
Jaeger and Goldberg [34]	1994	p-center	tree networks
Afif et al. [1]	1995	set covering	convert to maximum flow
Daskin [16]	2000	p-center	series of maximum covering problem
Ilhan snd Pinar [33]	2001	p-center	series of sets covering problem
Hansen et al.[29]	2001	p-median	neighborhood search
Blaser [7]	2003	partial covering	improve bounded of subgraph
Hajiaghayi et al. [25]	2003	general cost function facility location	greedy
Mladenovic et al. [43]	2003	p-center	tabu search and neighborhood search
Pallottino et al. [47]	2004	p-center	local search
Scaparra et al. [51]	2004	p-center	local search
Wu et al. [55]	2006	general cost function facility location	Lagrangian heuristic
Huang et al. [32]	2006	set covering	LFCP-ant
Mladenovic et al. [42]	2006	facility location	Erlenkotter-Korkel
Ozsoy and Pinar [46]	2006	p-center	series of sets covering problem
Janacek [35]	2008	facility location	Erlenkotter-Korkel
Hansen et al. [28]	2009	p-median	neighborhood search
Albareda-Sambola et al. [2]	2010	p-center	Lagrangian relaxation and series of sets covering problem
Almeida and Ararujo [3]	2010	p-median	metaheuristic search
Chen and Yuan [11]	2010	set covering	tabu search
Lee and Lee [39]	2010	hierarchical covering	two phase

Lai et al. [38]	2010	facility location	hybrid Benders and genetic
Rainwater et al. [49]	2011	facility location	neighborhood search
Zarandi et al. [57]	2011	facility location with fuzzy travel times	Simulate Annealing
Dantrakul and Likasiri [15]	2012	p-center	maximal client coverage
Sun [52]	2012	facility location	tabu search
Murali et al. [44]	2012	facility location under demand uncertainty	locate-allocate
Rahmaniani et al. [48]	2013	facility location under uncertainty	neighborhood search
Brimberg and Danzer [8]	2013	p-median in the plane	local search
Zare Mehrjerdi and Nadizadeh [58]	2013	facility location	greedy
Brimberg et al. [9]	2014	p-median	local search