Chapter 3

Modelling and optimization techniques

This chapter aims to familiarize the reader with background on modelling and optimization techniques relevant to the research, including optimization concepts, linear, non-linear, integer and Mixed Integer Linear Programming (MILP) models as well as the general Multifacility Location-Allocation Problem (MLAP). Solving techniques based on exact, heuristic and meta-heuristic methods will be given as well as relevant previous work on PON planning.

3.1 Models

With the increase in computing power, the modelling and simulation of increasingly complex real-world systems becomes possible, whether to visualize the underlying principles, test hypotheses before practical experiments are done or to optimize. This trend is made possible through the use of models, which can be defined as a *representation* of the inherent *structure* of a real-world system or object.

When a model is written down as a set of mathematical equations, it is known as a mathematical model, which is a mathematical realization of a specific problem or system. This model is composed of a number of main elements including variables, an objective function and zero or more constraints. Even though most models are tailored to their respective problems, defined problems exist which describe a general problem type encountered in a variety of areas. These predefined problems are typically encountered in some derived form in optimization.

3.1.1 Optimization

Given a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ and a set $\mathbf{S} \subseteq \mathbb{R}^n$, optimization can be defined as the goal to find the best possible elements $x^* \in \mathbb{R}^n$ that solves [27]:

$$\min_x \quad f(x) \tag{3.1}$$

$$\text{subject to} \quad x \in \mathbf{S} \tag{3.2}$$

Stated generally, it is a way of altering model inputs to arrive at the best outcome. As systems become increasingly autonomous, the opportunity for optimization increase, allowing for more productive factories, more effective system management and more economical deployment.

Mathematical models as defined above can be formulated in a number of ways, including a variety of linear, non-linear, continuous and integer objective functions and constraints.

Linear and non-linear programming

Linear programming is a subset of optimization to include only linear elements in its objective and criteria, i.e. that $f(x)$ is a linear function. They are usually expressed in standard form:

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned} \tag{3.3}$$

where **c** and **b** are given vectors, A is a matrix of coefficients and **x** is the variable vector to be determined.

In contrast, if the objective function is quadratic in nature, i.e. if it is defined as $f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$, the formulation is said to be a quadratic program. Similarly, if one or more constraints are quadratic, the formulation is known as a quadratically constrained program.

Stated more generally, when a non-linear objective function is included in a model, it is termed a non-linear program. Non-linear formulations are not preferred due to problems that can be encountered such as objective function non-convexity, for which no efficient optimization methods are known. Even though methods exist to solve certain convex non-linear as well as general quadratic programs, more efficient methods are known to solve and optimize linear formulations, including the simplex algorithm. Therefore, linear models are used if possible.

Integer and mixed integer linear programming

When all of the elements of x in equation 3.3 are integers, the formulation becomes an Integer Linear Program (ILP). This adds a new dimension to the problem that constrains the search space of the linear program to integer points inside the solution space as seen in the difference between figures 3.1a and 3.1b.

Mixed Integer Linear Programming (MILP) in turn is a hybrid of a normal continuous Linear Program (LP) and an ILP, in which some of the decision variables are constrained to be integers while some remain continuous.

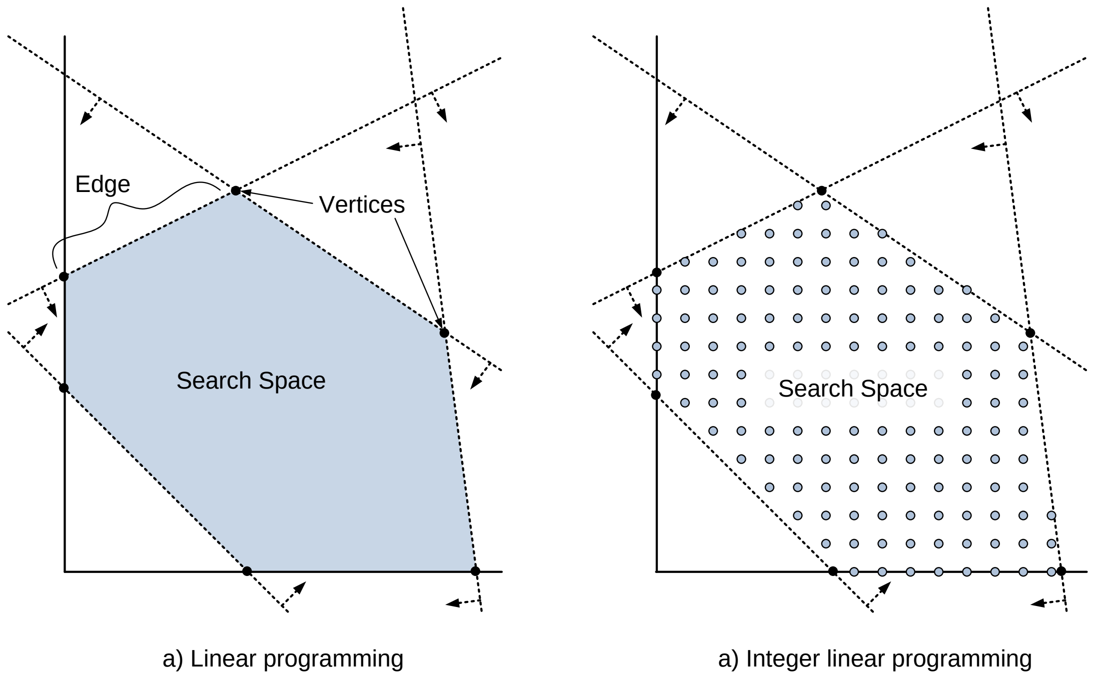

Figure 3.1: Linear programming (LP) vs integer linear programming (ILP)

3.1.2 Complexity

Determining the relative complexity of a problem is of great importance in modelling as it largely determines the solution method used. For simple problems, the computational complexity may be immediately obvious, such as determining the sign of a real number, which has an asymptotic complexity of $\mathcal{O}(1)$. This means that regardless of the input of the $sgn(\cdot)$ function, it will always take a constant amount of time to determine the output. A number of well known algorithms for which the asymptotic complexity can be easily determined is given in table 3.1.

When examining problems for which the complexity is not immediately obvious, computational complexity theory is used, which classifies problems according to the theoretical methods known to solve them. Particularly, the P, NP, NP-complete and NP-hard classifications are used, which, along with Turing machines, will now be very broadly defined. For a more in-depth introduction to computational complexity, Turing machines and the $P = NP$ problem, refer to [28].

A Turing machine is a theoretical machine, used in computational thought experi-

Table 3.1: Time complexity of some well known problems

$\mathcal{O}(\cdot)$	Time complexity	Problem
$\mathcal{O}(1)$	Constant	Signum function ($sgn(\cdot)$)
$\mathcal{O}(log(n))$	Logarithmic	Searching a balanced binary tree with n nodes
$\mathcal{O}(n)$	Linear	Finding the smallest element in an unsorted vector
$\mathcal{O}(n \cdot log(n))$	Linearithmic	Balanced binary tree sort
$\mathcal{O}(n^2)$	Quadratic	Multiplying an $n \times n$ matrix with a vector of size n
$\mathcal{O}(n^3)$	Cubic	Gaussian elimination for an $n \times n$ matrix
$\mathcal{O}(n^k)$, fixed k	Polynomial	Interior point methods for solving LPs
$\mathcal{O}(k^n)$, fixed k	Exponential	Simplex method for solving LPs

ments, that can be defined as a mathematical model of a machine that can shift, read or write a length of tape and that has a table of actions to execute depending on the current state of the machine. Thus, a Turing machine can be seen as a generic computer algorithm. Two types of Turing machines are relevant to the research: deterministic and non-deterministic.

Definition 3.1 (Deterministic Turing machine). *For each step in the computation of a deterministic Turing machine or algorithm, a unique successor state exists for each non-final state. Thus, at each stage of computation, the next step is uniquely defined.*

Definition 3.2 (Non-deterministic Turing machine). *For each step in the computation of a non-deterministic Turing machine or algorithm, a number of successor states exists for each non-final state. Thus, at each stage of computation, there exists a number of steps that should be followed simultaneously, branching into a number of different machines with different states.*

The sequence of computation in a non-deterministic Turing machine can be seen as an initial configuration, branching out over a potentially infinite number of paths before stopping at some final state or returning to some previous state in that leaf. Now that deterministic and non-deterministic Turing machines are defined, the computational classes can be defined (assume in this case $P \neq NP$):

Definition 3.3 (P). *Let $DTIME(t)$ denote the class of problems solvable by a deterministic Turing machine in time t.*

$$P = \bigcup_{k \geq 0} DTIME(n^k) \tag{3.4}$$

is a class of problems that can be solved by a deterministic Turing machine in polynomial time, also known as efficiently solvable or tractable problems.

Definition 3.4 (NP). *Let $NTIME(t)$ denote the class of problems solvable by a non-deterministic Turing machine in time t.*

$$NP = \bigcup_{k \geq 0} NTIME(n^k) \tag{3.5}$$

is a class of problems for which solutions can be guessed and verified in polynomial time, i.e. solved by a non-deterministic Turing machine in polynomial time.

To define NP-hard and NP-complete, the lower complexity bound needs to be analyzed using the concept of *reducibility*. Reducibility is used to compare the complexity of two problems: if a problem A can be reduced to a problem B, then A is at most as *hard* as B. Furthermore, polynomial time reducibility is the ability to compute a function f in polynomial time to reduce one problem to another. Informally, we can now define [28]:

Definition 3.5 (NP-complete and NP-hard). *Let A and B be two problems. If A can be (polynomial time) reduced to B for all $A \in NP$ (written $A \leq_m^p B$), B is said to NP-hard. If B is NP-hard and in NP, B is said to be NP-complete.*

Therefore, if a problem is NP-hard, it is at least as hard as the hardest NP problems, but due the fact that NP-hard problems are not necessarily in NP, they can be harder than the hardest NP problems, i.e. harder than any NP-complete problem. The relationships between complexity classes are illustrated in figure 3.2.

3.2 Methods

Due to its real-world correlation, the practical usefulness of a mathematical model is only as good as its solution, i.e. models can be formulated to include all aspects of a system in absolute detail, but no conclusions can be drawn unless the model is solved to some degree of optimality.

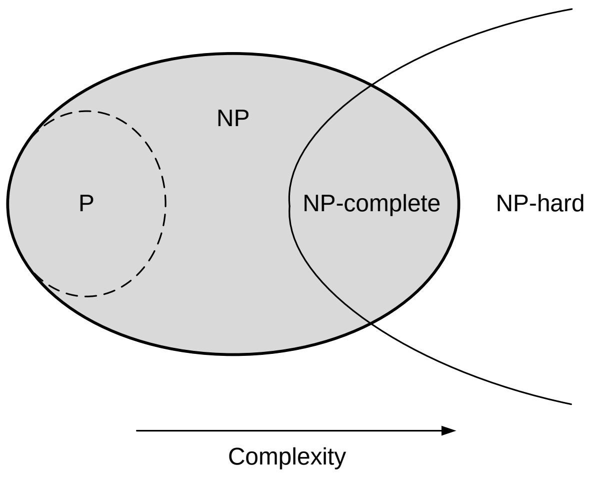

Figure 3.2: Euler diagram of the NP type of complexity classes

3.2.1 Optimality

To define optimality, we must first look at local optima as defined by Weise in [27]:

Definition 3.6 (Local Minimum). *Given an objective function $f : \mathbf{X} \mapsto \mathbb{R}$, $x_\ell^{MIN} \in \mathbf{X}$ is a local minimum if $f(x_\ell^{MIN}) \leq f(x)$ for all x neighbouring x_ℓ^{MIN}.*

If $\mathbf{X} \subseteq \mathbb{R}^n$, there exists an $\varepsilon > 0$, so that:

$$f(x_\ell^{MIN}) \leq f(x) \quad \forall x \in \mathbf{X}, |x - x_\ell^{MIN}| < \varepsilon \tag{3.6}$$

Definition 3.7 (Local Maximum). *Given an objective function $f : \mathbf{X} \mapsto \mathbb{R}$, $x_\ell^{MAX} \in \mathbf{X}$ is a local maximum if $f(x_\ell^{MAX}) \geq f(x)$ for all x neighbouring x_ℓ^{MAX}.*

If $\mathbf{X} \subseteq \mathbb{R}^n$, there exists an $\varepsilon > 0$, so that:

$$f(x_\ell^{MAX}) \geq f(x) \quad \forall x \in \mathbf{X}, |x - x_\ell^{MAX}| < \varepsilon \tag{3.7}$$

Definition 3.8 (Local Optimum). *Given an objective function $f : \mathbf{X} \mapsto \mathbb{R}$, $x_\ell^* \in \mathbf{X}$ is a local optimum if it is a local minimum or local maximum.*

Thus, a local optimum is an optimum point for some subset of $\mathbf{X}$. By generalizing definitions 3.6 through 3.8, we can now define the global optimum [27]:

Definition 3.9 (Global Minimum). *Given an objective function $f : \mathbf{X} \mapsto \mathbb{R}$, $x^{MIN} \in \mathbf{X}$ is a global minimum if $f(x^{MIN}) \leq f(x) \quad \forall x \in \mathbf{X}$.*

Definition 3.10 (Global Maximum). *Given an objective function $f : \mathbf{X} \mapsto \mathbb{R}$, $x^{MAX} \in \mathbf{X}$ is a global maximum if $f(x^{MAX}) \geq f(x) \quad \forall x \in \mathbf{X}$.*

Definition 3.11 (Global Optimum). *Given an objective function $f : \mathbf{X} \mapsto \mathbb{R}$, $x^* \in \mathbf{X}$ is a global optimum if it is a global minimum or global maximum.*

Therefore, the global optimum is an optimum point for the whole domain $\mathbf{X}$ and is illustrated in figure 3.3. In the LP domain, the global optimum is the vector $\mathbf{x}^*$ that produces the optimal objective function value if f is defined as $\mathbf{c}^T\mathbf{x}^*$.

Three solution approaches are used to solve mathematical models depending on the problem type, including exact methods, heuristics and meta-heuristics. These methods involve an iterative process of refining a *feasible solution*, or a solution that satisfies all model constraints, until an optimal solution is reached. A measure of optimality calculable without knowledge of the global optimum, known as the *optimality gap*, can now be defined as follows [29]:

Definition 3.12 (Feasible Solution). *For a general LP with objective function $f : \mathbf{X} \mapsto \mathbb{R}$:*

$$\begin{aligned} \textit{minimize} \quad & f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ \textit{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned} \tag{3.8}$$

a solution vector $\bar{\mathbf{x}} \in \mathbf{X}$ is a feasible solution if it satisfies the LP constraints, i.e. $A\bar{\mathbf{x}} \leq \mathbf{b}$ and $\bar{\mathbf{x}} \geq 0$ holds.

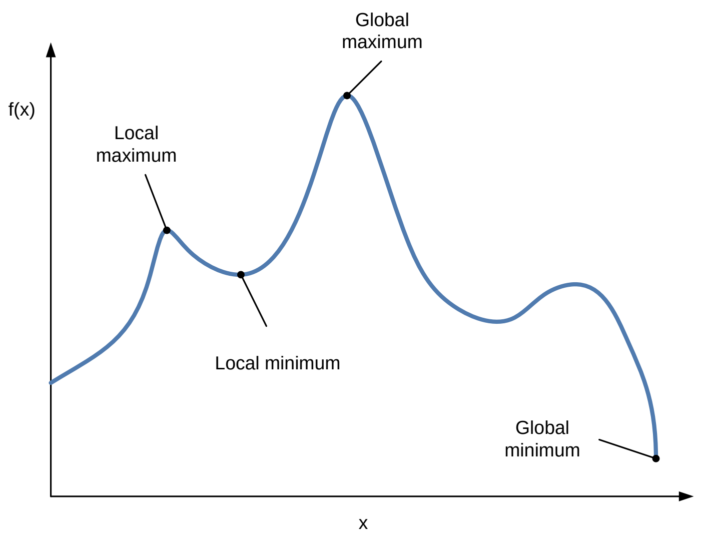

Figure 3.3: Local and global optima for a function $f(x)$

3.2.2 Exact methods

Exact methods aim to produce a globally optimal solution for the model in question, using algorithms such as simplex or interior point to solve LPs and branch and bound for MILPs. Dedicated solutions using exact methods exists, including IBM ILOG CPLEX Optimization Studio [13], Gurobi Optimizer [30] and CLP [31]. These solutions use advanced techniques including problem-specific cut generation and heuristics to decrease execution times while preserving global optimality. When looking at solving MILPs however, all these solutions use the same method to solve sub-problems i.e. the simplex method [*].

Simplex method

The simplex method or algorithm is a popular basis-exchange pivot algorithm used in solving LPs, first published by George B. Dantzig in 1947. It builds on the notion that if an LP is in standard form the minimum objective value must be at an extreme point in the feasible region. Furthermore, with **c** and **b** parameter vectors, A a parameter matrix

[*]Alternatives to the simplex method exist in some of these solutions, including the interior point or barrier methods

and $\mathbf{x}$ the variable vector, the feasible region $A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0$ is a convex polytype.

Referring to figure 3.1, the algorithm can then be described as a method of moving from an initial vertex along those edges of the polytype where the objective function strictly decreases, until it reaches the optimum solution vertex. Due to the fact that vertices are always visited from a single direction i.e. originating from a point of higher objective value, the overall number of vertices visited is small, speeding up execution considerably [32].

Branch and bound

Developed by A.H. Land and A.G. Doig [33] and published in 1960, branch and bound is a general algorithm used to solve discrete optimization problems such as MILPs. It uses an approach where possible candidate or feasible solution subsets are enumerated, compared with calculated bounds and discarded if proven to be worse than the best known solution. Therefore the approach can be seen as *divide and conquer*, whereby whole sets of feasible solutions can be discarded *en mass*, allowing faster convergence to the optimal integer solution. Figure 3.4 illustrates how branch and bound divides the search space and discards whole sections through pruning and the algorithm can be found in appendix D.1.

For ILP or MILP models, the branch and bound algorithm uses the objective value of the relaxed LP (with missing integer constraints) as a lower bound, the objective value of the current feasible solution as an upper bound and branching is done on unfixed integer variables. This algorithm then gives an indication of the computational progress by calculating the optimality gap of the current best solution as seen in figure 3.5. Unfortunately for some models, due to their complexity, computing a globally optimal solution is infeasible with regards to resource usage, be it time, memory or storage.

Definition 3.13 (Optimality Gap). *Given an integer linear program, the optimality gap of a feasible solution $\bar{\mathbf{x}} \in \mathbf{X}$ is defined as:*

$$\textit{Optimality gap (\%)} = \frac{|f(\bar{\mathbf{x}}) - f(\check{\mathbf{x}})|}{f(\bar{\mathbf{x}})} \times 100\% \qquad (3.9)$$

where $f(\check{\mathbf{x}})$ is the best known lower bound of f.

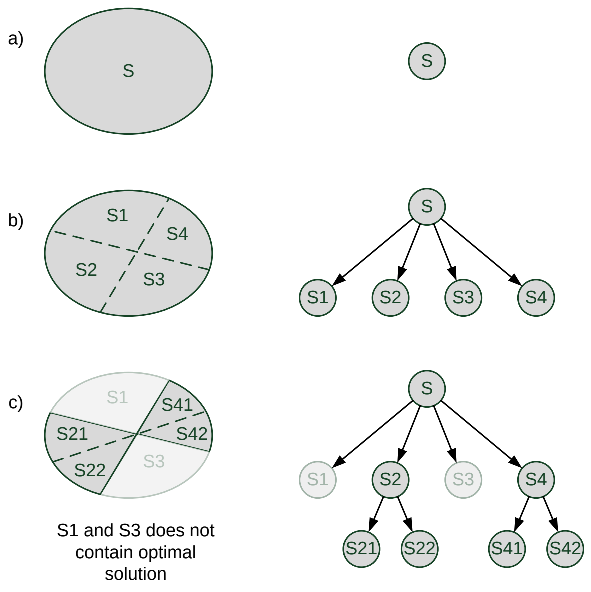

Figure 3.4: Search space traversal of the branch and bound algorithm

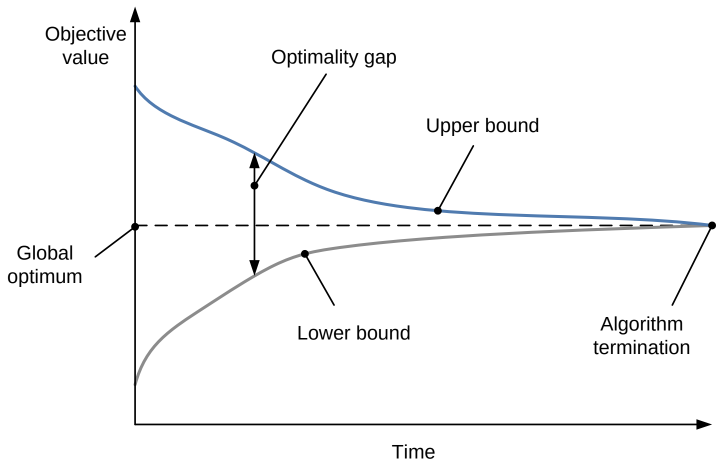

Figure 3.5: Upper and lower bounds of discrete minimization problem solved using branch and bound

3.2.3 Heuristics

Due to the complexity of some problems, solvers struggle to obtain a globally optimal solution for large data sets in a reasonable amount of time. To increase the speed at which a solution can be computed, heuristics are often used, resulting in a near-optimal solution at a fraction of the computational cost.

According to Weise [27], a heuristic can be defined as a step in an optimization algorithm that uses current available data gathered by the algorithm thus far to determine which solution from the search space should be investigated in the next iteration. This is then a way to *guide* the algorithm to a solution without exhaustive enumeration.

Although heuristics depend on the problem class, typical examples include disintegration of the search space into smaller segments through random or guided cuts [34, 35] and random sampling of the search space to get representative samples [2].

3.2.4 Meta-heuristics

A meta-heuristic can be defined as a method of solving very general problems by treating objective functions and constraints in a *generic* way - without using insight into their *structure* or form i.e. as black boxes [27].

Meta-heuristics are often based on natural phenomena such as genetics (Genetic Algorithms), solidifying metallic elements (Simulated Annealing) and creature behaviour (Ant-colony or Swarm Optimization). However, techniques exist without direct correlation to real-world models such as Tabu search and Random optimization.

Due to the fact that these heuristics are abstract from the underlying structure of the problem, they can also be used to solve non-linear and polynomial problems, which is of great importance in a number of research papers [36–42]. Unfortunately, meta-heuristic methods can sometimes converge to local optimum solutions and in contrast with exact methods, there are no generated upper or lower bounds to determine optimality or current computational progress.

3.3 Network planning optimization

In accordance with all types of network deployment, planning plays a fundamental role with regard to overall cost as well as network efficiency. Planning of fiber networks is traditionally done manually by planners trying to find a subjectively optimal solution. Even though these plans are close to optimal for smaller networks, large network plans are usually sub-optimal, time consuming and require experienced planners. With the invention of planning optimization, this process can be automated to produce optimal results in minutes, even with the inclusion of obstacles [37].

Network planning is usually done by means of a two step process; formulation of the problem as a mathematical model and solving the model to optimality or near-optimality if optimality is not feasible. One particular problem that is of great importance in modelling network planning is the MLAP.

3.3.1 Multifacility Location-Allocation Problem (MLAP)

The MLAP (also known as the Facility location problem) can be conceptually defined as a problem where a subset of a number of potential facilities must serve a fixed number of demand points as illustrated in figure 3.6. Each demand point must be assigned to a facility and the usage of each facility has a fixed cost, while travel between a demand point and a facility incurs a cost proportional to the distance between them. The problem then aims to allocate demand points to facilities so as to minimize the overall cost of the system, i.e. the lowest cost to serve all demand points [43].

The problem can be formulated mathematically as follows [44]:

$$\text{minimize} \quad \sum_{i\in\mathbf{I}} f_i y_i + \sum_{i\in\mathbf{I}} \sum_{j\in\mathbf{J}} c_{ij} x_{ij} \tag{3.10}$$

$$\text{s.t.} \quad \sum_{i\in\mathbf{I}} x_{ij} = 1, \quad \forall j \in \mathbf{J} \tag{3.11}$$

$$\sum_{i\in\mathbf{I}} y_i \leq p \tag{3.12}$$

$$x_{ij} \leq y_i, \quad \forall i \in \mathbf{I}, j \in \mathbf{J} \tag{3.13}$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in \mathbf{I}, j \in \mathbf{J} \tag{3.14}$$

$$y_i \in \{0,1\}, \quad \forall i \in \mathbf{I} \tag{3.15}$$

where $\mathbf{I}$ is the set of candidate facility locations, $\mathbf{J}$ is the set of demand points and f_i is the fixed cost of setting up a facility at $i \in \mathbf{I}$. c_{ij} denotes the cost of supplying demand point j from facility i while p is the maximum number of facilities allowed. The decision variable x_{ij} takes on the fractional value of the demand of $j \in \mathbf{J}$ supplied by facility $i \in \mathbf{I}$ while the binary decision variable y_i takes on a value of 1 if the i-th facility is used and 0 otherwise.

The facility location problem is very well known in operations research due to the fact that it is difficult to find a solution and that the complexity of the problem scales up considerably as the number of demand points increase. This is due to it being classified as NP-hard as proven in [45]. With the inclusion of a variable amount of facilities, this problem needs to effectively enumerate all possibilities of facility locations and allocations.

3.4 Passive Optical Network (PON) planning problem

Using automated methods for PON planning requires a mathematical model of the network topology and a solver to arrive at an optimal solution. In essence, the PON planning problem is very similar to the facility location problem, consisting of three sub-problems, all of which must be solved simultaneously for an optimal solution [2]:

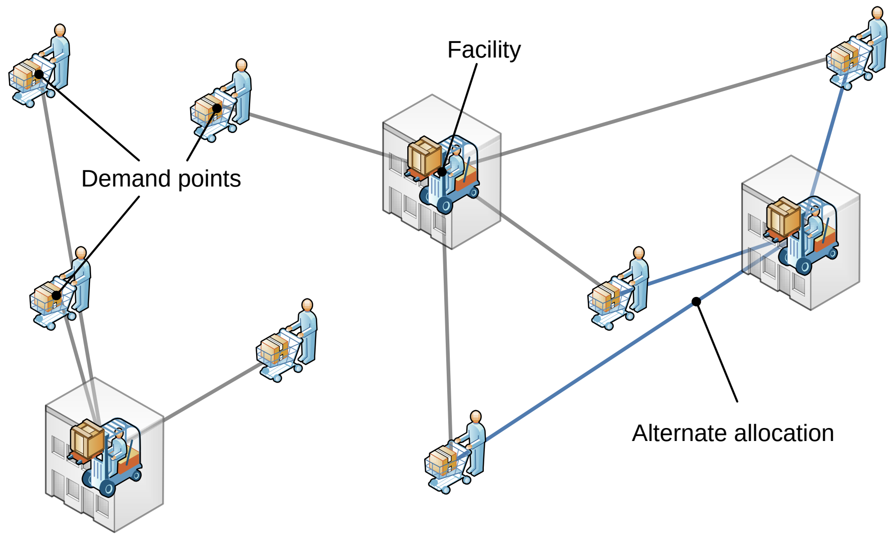

Figure 3.6: The multi-facility location-allocation problem (MLAP)

1. Determine the optimal number of splitters used.
2. Optimal allocation of ONUs to splitters.
3. Relocation and reallocation of splitters for optimal cost.

For the first sub-problem, it is evident that the number of splitters cannot be determined in advance. This ensures that for optimality, all possible patterns should be exhaustively compared to find the optimal number. Due to this brute-force search, the complexity increases factorially with the number of ONUs.

The second sub-problem combines with the first to form a MILP model, provided the location of splitters are given. Even though this model is solvable, it is still NP-complete, ensuring complexity becomes prohibitive for large-scale networks.

Lastly, the third sub-problem locates splitters so as to minimize the cost of laying fiber and connecting splitters to ONUs. This problem is known as the Fermat-Weber point problem [46]. Due to the problem's non-linearity, it is difficult to solve using traditional techniques, although algorithms exist to solve it specifically, one of which is the Weiszfeld algorithm [2,46].

Due to the nature of the problem, a large number of binary decision variables is required, including variables modelling whether a splitter is used at a specific location or whether a specific splitter is connected to a specific ONU. Therefore, the PON problem is usually modelled as a MILP.

3.5 Previous work on PON planning

Since the model in question is usually either intractable due to it being non-linear or NP-complete and overly complex, heuristics are implemented to give a close to optimal solution in a feasible amount of time. Previous studies on optimized PON planning can be divided into two sections: those based on heuristics and exact techniques and those based on meta-heuristics such as genetic algorithms [27].

3.5.1 Exact methods and heuristics

In [2], the authors provide a new heuristic called Random Allocation and Reallocation Algorithm (RARA) to speed up the optimization process of greenfield PON network planning through disintegration. The heuristic is based on randomly sampling a solution from the search space and incrementally optimizing it through a Simulated Annealing (SA)-like step in their algorithm. The network model provided takes into account different types of splitters for deployment and includes GPON and EPON standard constraints such as differential distance and maximal transmission distance. The authors also introduce maximal fiber duct sharing to further decrease the overall cost of deployment. Finally, the algorithm is compared to a basic random-cut heuristic to show its efficiency.

The authors of [47] propose a purely exact model, also based on the MILP technique, to solve a multi-hierarchy PON network. The model includes splitters as well as Cable Distribution (CD) points which link to specific tenancies. Furthermore, the authors include planning constraints such as the maximum number of tenancies per CD and maximum number of CDs per splitter. Since the model is solved exactly, the placement of splitters and CDs lead to global optimal results when applied to the *Medium Net* and *Big Net* data they examine.

In [34], the author provides an algorithm based on graph theory to optimize the planning of a grid-like area such as Manhattan (New York City). The greedy algorithm transforms population density on the original grid to a new graph which Jacobian is proportional to some population density function. The vertices on the new graph are then compared for minimum distance allocation. This is then repeated iteratively to find minimum cost locations for OLTs, splitters and ONUs. Finally, the algorithm is compared to another population density type algorithm, PNLA.

In [48], the authors investigate the same optimization problem of PON planning, but implements a complete Capital Expenditure (CAPEX) model to account for the various subdivisions of expenditure streams by minimizing for each. The paper specifically examines very large problem sizes and computation times inherent to each approach. They propose a heuristic called the Branch Contracting Algorithm (BCA) which builds a tree and then iteratively refines it through customer segmentation and splitter placement. For the final step, the algorithm builds a Steiner-tree to connect CDs with splitters. The algorithm is compared with a lower bound ILP to determine its efficiency. The same authors also provide a methodology in a more recent work [12] for the general planning of broadband access networks and provide four different heuristics based on the network type; including PON, AE and Very-high-bit-rate Digital Subscriber Line (VDSL). Finally, the heuristics are shown to provide solutions in the range of 10-15 % to optimal.

The authors of [49] propose the same planning problem solution for PONs, but incorporates survivability into their model. Three models based on 99.99 % and 99.999 % service availability are given and solved with exact methods. In contrast to other works, the models are formulated to optimize for network reach, maximizing the covered area. The same authors then propose a planning heuristic in [35] to provide a survivable long-reach PON design.

In [50], the authors formulate the planning problem as a multi-level capacitated facility location problem on a tree topology but with non-linear cost for links. Due to this non-linearity, MILP models can only give upper and lower bounds for the solution through objective function relaxation. The paper goes on to propose both valid inequalities and a heuristic to improve the upper bound of the solution through two phases called upper improvement and downward refinement.

3.5.2 Meta-heuristics

The authors of [36] present a hybrid approach to solving the multi-hierarchy, single split PON problem, incorporating an algorithm utilizing a Minimum Spanning Tree (MST) step to form trees, a heuristic method to cluster PONs and finally a genetic algorithm to identify distribution point locations. The algorithm minimizes the total number of splitters and distribution points and the total length of fiber cable connections.

In [37], the authors present an evolutionary computing-based algorithm that includes non-traversable obstacle avoidance through convex hull mapping. The problem is subdivided through the use of a k-means clustering algorithm and compared to a handmade connection plan.

While most works address the single-hierarchy PON problem, the authors of [38] propose a genetic algorithm with a multi-hierarchy planning heuristic incorporating a min-max distance algorithm to cluster potential splitters to ONUs. The authors of [41] also address the multi-hierarchy problem, with placement of primary and secondary nodes optimized through a genetic algorithm. In addition, they include the split ratios of each splitter into their model as well as optic attenuation and apply the algorithm to a residential area in Turkey.

In [39], the authors apply Genetic Algorithm (GA) techniques combined with graph theory to modified versions of the PON problem, including bus, tree and ring topologies. The authors of [40] use the same meta-heuristic GA techniques, but propose an algorithm to optimize for minimum power consumption, comparing optimal MILP, heuristic and meta-heuristic solutions based on performance.

Finally, the authors of [42] introduce an interesting algorithm based on a combination of clustering techniques and ant colony optimization. It produces better global solutions than comparable heuristic algorithms since it is not as vulnerable to local optima.

3.6 Conclusion

In this chapter, the basic concepts surrounding modelling and solution techniques were discussed. Linear, integer and mixed integer models were defined, along with the respective techniques of classifying their complexity. After detailing degrees of optimality, the differences between exact methods, heuristics and meta-heuristics were illustrated.

The ability of exact methods to quantify the quality of an intermediate solution through the use of the optimality gap and their ability to always terminate with the proven optimal makes them superior to heuristics given computation time remains feasible. Conversely, heuristics are not capable of quantifying optimality, but computes a feasible solution in a short period of time. Therefore, the exact approach will be followed as far as possible before implementing heuristics to reduce computation time.

Finally, after focussing on how the PON planning problem is typically defined, an outline of previous work on PON planning was given. In the next chapter, a basic mathematical model of the PON planning problem will be described and tested with theoretical data.