1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Gustavo Pereira Gama

Automatic Design of Networks

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Professor João Correia and Professor Nuno Lourenço and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2022

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Ð
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Gustavo Pereira Gama

Automatic Design of Networks

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Professor João Correia and Professor Nuno Lourenço and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2022

Acknowledgements

Firstly, I would like to thank Professor João Correia and Professor Nuno Lourenço from the Department of Informatics Engineering, Faculty of Sciences and Technology, University of Coimbra, who helped me throughout this thesis with their valuable suggestions. I am grateful for their availability and time dedicated in the different meetings.

Special thanks for my family for their consistent support and patience during all those academic years. Also, I'd like to mention my friends for being there during challenging times and for all the good experiences shared.

This work is funded by the project POWER (grant number POCI-01-0247-FEDER-070365), co-financed by the European Regional Development Fund (FEDER), through Portugal 2020 (PT2020), and by the Competitiveness and Internationalization Operational Programme (COMPETE 2020).

Lastly, I would like to thank Altice team (our collaborator) for their time, for their feedback on the work done and for the datasets.

Abstract

With the increasing demand of high quality internet services, the deployment of "GPON/Fiber-to-the-Home" networks is one of the biggest challenges that internet providers have to deal with due to the large investments involved. The usage of automated network designs becomes more and more important in order to aid with the task of planning the network by minimizing the costs of planning and deployment.

The main objective of this thesis is to tackle this problem of optimization of networks that requires to take into account multiple factors such as the equipment placement and their configuration, the optimization of the cables routes, the optimization of the clients allocation and other constraints involved in the minimization problem. An AI-based solution is proposed to automate the process of design of networks which is often done manually. It is a complicated task that involves a great amount of time to complete by hand, whereas the proposed system manages to find new design solutions in a few seconds/minutes, depending on the size of the network.

This document proposes a discussion around the topic of Automatic Design of Networks and about the work performed during this thesis. The system developed is described and for a fixed experimental setup with real datasets, multiple experiments are analyzed. A comparison with handmade solutions (whenever available) is also proposed to evaluate the quality of the solutions generated by the AI system which is based on Genetic Algorithms. The quality of the solutions is based on their cost compared to the handmade solution. But also based on the fact that the multiple constraints are respected and based on the computational time required to obtain such solution.

Keywords

Fiber-to-the-Home, GPON, Graph Theory, Optimization, Nature-inspired Algorithms, Genetic Algorithms, Local Search.

Resumo

Com o aumento dos pedidos de serviços internet de alta qualidade, a implantação de redes "GPON/Fiber-to-the-Home" é um dos maiores desafios que os provedores de internet enfrentam devidos aos grandes investimentos involvidos. O uso de projetos de rede automatizados torna-se cada vez mais relevante para ajudar na tarefa de planejamento de redes, minimizando os custos.

O principal objetivo desta tese é abordar este problema de otimização de redes que requer ter em conta múltiplos fatores como a localização e configuração dos equipamentos, a otimização dos percursos dos cabos, a otimização da alocação de clientes e outras condicionantes envolvidas no problema de minimização. Uma solução baseada em IA é proposta para automatizar o processo de projeto de redes, que muitas vezes é feito manualmente. É uma tarefa complicada que envolve muito tempo para ser executada manualmente, enquanto que o sistema proposto consegue encontrar novas soluções de projeto de redes GPON em poucos segundos/minutos, dependendo do tamanho da rede.

Este documento propõe uma discussão em torno do tema de projeto de redes e sobre o trabalho realizado durante esta tese.O sistema desenvolvido é descrito e para uma configuração experimental fixa com conjuntos de dados reais e pelos quais vários testes são analisados. Uma comparação com soluções feitas a mão (quando disponíveis) também é proposta para avaliar a qualidade das soluções geradas pelo sistema de IA baseado em Algoritmos Genéticos. A qualidade das soluções baseia-se no seu custo em relação à solução manual. Mas também com base no fato de que as múltiplas restrições são respeitadas e com base no tempo computacional necessário para obter tais soluções.

Palavras-Chave

Fiber-to-the-Home, GPON, Teoria dos Grafos, Otimização, Algoritmos inspirados na Natureza, Algoritmo Genético.

Contents

Acronyms

ACO Ant Colony Optimization.

CAPEX Capital Expenditure.

CO Central Office.

EA Evolutionary Algorithms.

FTTB Fiber-to-the-Building.

FTTH Fiber-to-the-Home.

FTTX Fiber-to-the-X.

GA Genetic Algorithms.

GPON Gigabyte Passive Optical Network.

ILP Integer Linear Programming.

ISP Internet Service Provider.

JSO Joint Optical Splitter.

LP Linear Programming.

MDU Multi Dwelling units.

MILP Mixed Integer Linear Programming.

MST Minimum Spanning Tree.

ONU Optical Network Units.

PDO Optical Distribution Point.

SDU Single Dwelling units.

SI Swarm Intelligence.

List of Figures

List of Tables

Chapter 1

Introduction

The demand for high quality internet services has been increasing at rapid rate for the past years, due to the fact that consumers require more and more bandwidth intensive applications (like cloud services, video streaming, audio, increase of the number of connected devices for example). This forces operators to consider faster technologies to answer the increasing demand, like the usage optical fiber, i.e. Fiber-to-the-X (FTTX). FTTX refers to a wide range of deployment configurations for different scenarios in the last mile stage of telecommunications networks. The most common are Fiber-to-the-Home (FTTH) which aims to connect directly to the residence of the client and Fiber-to-the-Building (FTTB) which reaches the buildings directly.

One way of providing this type of service (like FTTH) is through a Gigabyte Passive Optical Network (GPON) based solution. GPON is a point-to-multipoint (P2MP) network which enables a single fiber system to serve multiple customer premises with a fiber splitting process.

This technology has many advantages in comparison to other type of networks, e.g., copper-based networks (DSL), which has been the standard solution for many years:

- Faster data transmission and increased range with optical fiber cables being able to cover distances up to 20km;
- Boosted security;
- Future-proof technology, if there is the need to upgrade the network, we can do so by changing only the endpoints and leaving the fiber infrastructure;
- Cost effective.

To deploy a GPON/FTTH network, there are many challenges that the Internet Service Provider (ISP) has to consider, involving large investments. The nature of the problem of networks means that there is a lot of hand work involved to maintain the network or even build new infrastructure (e.g., duct digging, laying new cables to connect to the homes), so it is important to have a way of optimize the planning/design of the network to avoid excessive costs. The origin of most of the difficulties during the process of deploying a network are from engineering or technical aspects. A proper planning often results in savings in the range of 30% Council [2016]. However, to create a cost effective GPON/FTTH, engineers have to consider several factors:

- Optical splitter position and its maximum ratio;
- The headend position;
- Optical distribution point position;
- The routes to consider, the maximum distance;
- The number of homes that require the services;
- Optical budget.

Taking into account all of these factors it is a complex task that requires time. By automating the design, we can reduce the time required for the design process from days to a few hours or minutes.

1.1 Objectives

The objective of this work, in collaboration with Altice Labs, is to design and develop with an AI based solution for the problem of Automatic Design of Networks. That solution has to be flexible enough to allow the input of different parameters (e.g. the position of certain equipment in the configuration, the costs of the operations and materials) and adjust the solution output based on those inputs.

Another objective for this work is to produce a scalable solution for the problem at hand, capable of dealing with multiple sizes of networks. To give an idea, the data sets used during this work are mostly medium sized networks (around 200 nodes for example), but we can also have networks containing around 7000 nodes, which requires much more computational power to solve.

For that purpose, a certain number of approaches are considered and studied, more specifically, the usage of Evolutionary Algorithms (EA).

1.2 Document Structure

The remainder of this work is structured as follows:

Chapter 2 discusses the problem to be solved in this work.

Chapter 3 aims to provide some background context in optimization approaches used in the topic of Automatic design of networks. It also presents the State-of-the-Art in the topic of AI for the problem in question.

Chapter 4 introduces the approach considered to build the automatic design system.

Chapter 5 describes the results obtained, its purpose is to define a experimental setup and perform the tests on that setup. The results obtained for the different maps (datasets) are also discussed.

Finally, *Chapter 6* concludes on the work done during this thesis.

Chapter 2

Problem Statement

The main goal of this chapter is to describe the problem related to the design of networks with a focus on the GPON/FTTH configuration. We start by describing the principal components to consider for the design of GPON. Then, we summarize and formulate the minimization problem of Automatic design of networks.

2.1 Description

When dealing with the problem of planning a network, it is important to consider different elements. This section aims to introduce them such as the architecture considered, the different elements of the network and the multiple constraints involved.

2.1.1 GPON Overview

There are multiple architectures that can be considered to plan a network, but the tree-based topology is the most common for GPON networks. A basic GPON network (see Figure 2.1) is composed by the following components:

- *Central Office (CO)*: This is where the shared network equipment is located, such as the Optical Line Terminals (OLT) which connect the optical fiber and transfer signals towards the rest of the network (Feeder).

- *Feeder Network*: the optical fiber cable that connects the CO to the first splitting point.

- *Joint Optical Splitter (JSO)*: a splitter or a set of splitters. Those splitters are the devices that split the input fiber and its signal into multiple. The ratio associated to a splitter corresponds to the splitting of the fiber, for example a ratio of $1 : N$ means that the single input signal can be divided by N outputs.

- *Distribution network*: Secondary set of cables that link the JSO to the final distribution points (PDOs).

- *Optical distribution point* (*ODP* or *PDO* [1]): Those are the installation points placed on the street poles or other network equipment that connect to the ONUs.

- *Optical Network Units (ONU)*: Represents the equipment on the client side that converts and process the signals.

- *Drop cable*: fiber cable that connects a Optical Distribution Point (PDO) to the nearby clients premises (there is one for each client premise).

A GPON network can reach up to 20 km of distance between the internet provider and the users (usually 64 for one fiber). In a GPON, the streams of data are bidirectional, with a downstream transmission (bit rate up to 2.488 Gbits/s from the OLT to the user) and the upstream transmission (bit rate up to 1.244 Gbit/s from the user to the OLT). [2]

Figure 2.2 illustrates how the downstream transmission works, the OLT broadcasts the data continuously and the splitter sends this data to every user. It is the ONU on the user side that filters the correct information that is destined for each user. On the other side, the upstream transmission is not continuous, it is made of bursts of data where each user is given a time slots where they can transmit data, the information is regrouped in the splitter before reaching the OLT (the provider). Figure 2.3 illustrates the upstream transmission.

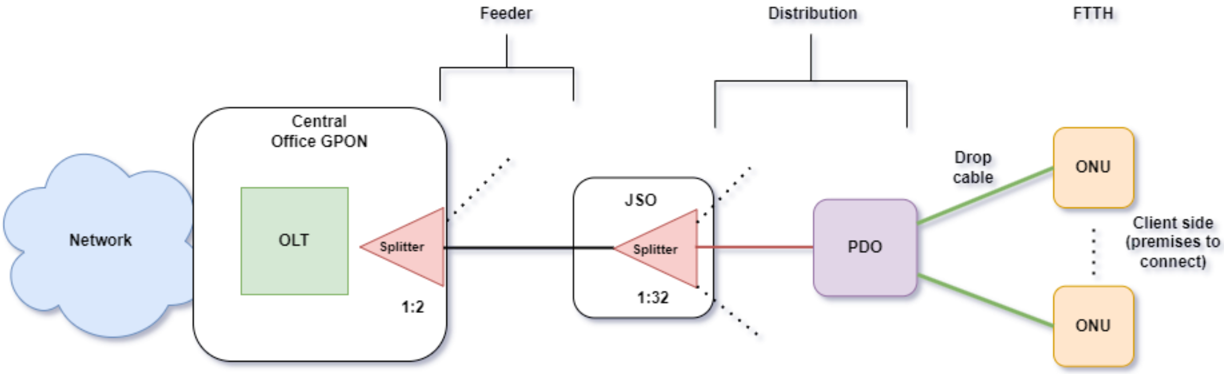


Figure 2.1: Basic GPON/FTTH architecture (centralized approach).

2.1.2 Splitters Configuration

The splitters are at the core of the architecture and their configuration has to be considered to design the network, specifically their ratios and levels.

[1] PDO is more common, specially in the Portuguese literature (from our collaborator). This will be the preferred notation for the rest of the work.

[2] http://www.gpon.com/how-gpon-works

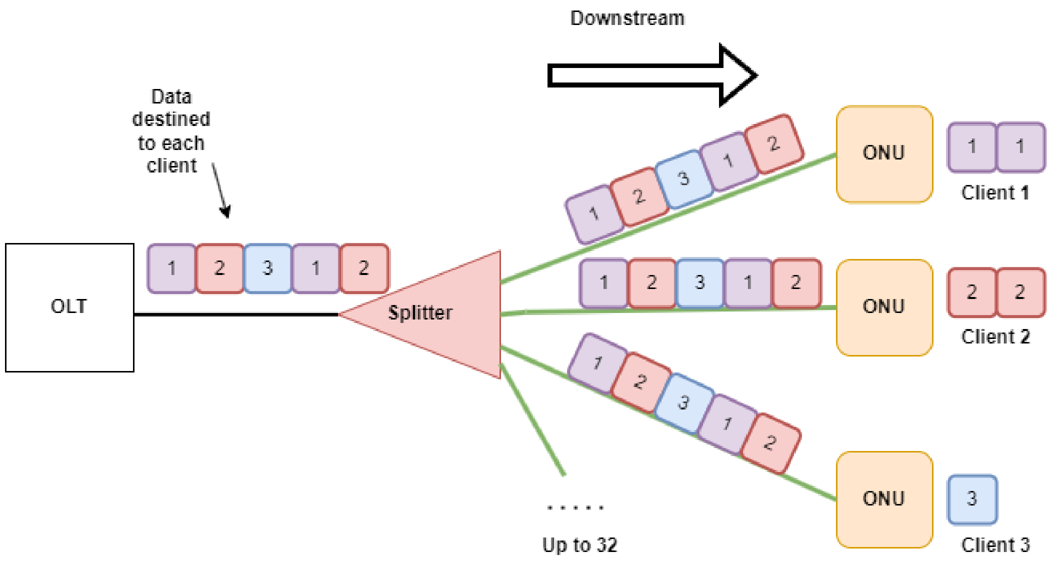


Figure 2.2: Illustration of the downstream transmission in GPON.

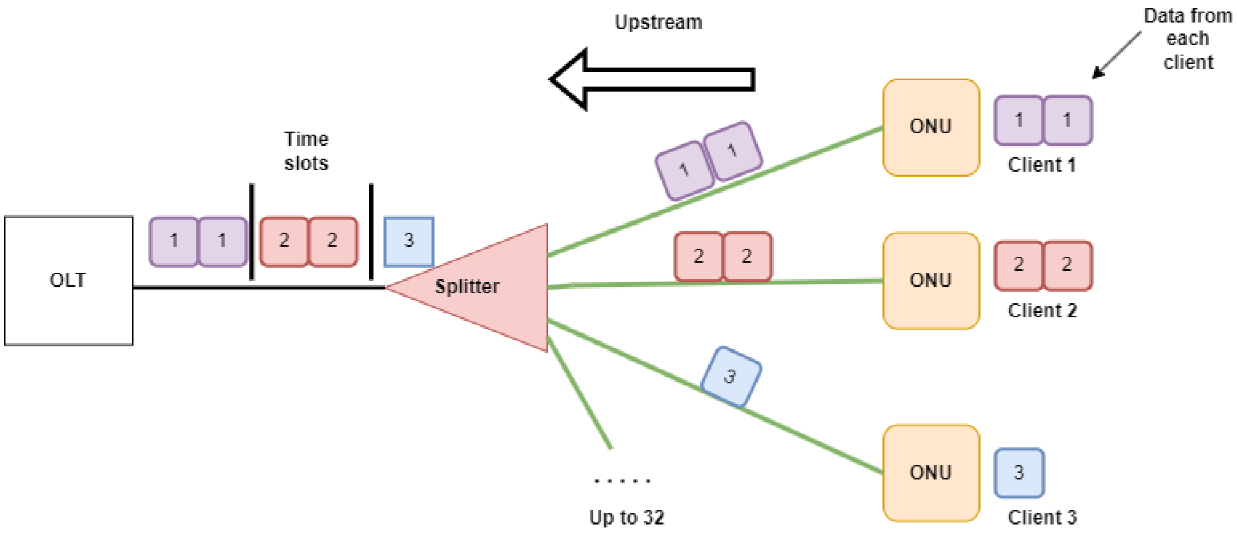


Figure 2.3: Illustration of the upstream transmission in GPON.

The architecture mentioned previously is the one chosen for the current work, it corresponds to a centralized approach due to the fact that there is only one level of splitting outside of the CO. A splitter can be directly installed in the CO to allow extra flexibility to the network, it allows to connect a client home directly to the CO if it is within in range. When the CO has a splitter, the typical ratio for the JSO splitter is 1:32. There are also other possible ratio configurations. For example when the CO has no splitter, the JSO usually has a splitter of ratio 1:64 that serves the 64 users of the network (for a single fiber cable coming from the OLT).

More complex architectures are also possible, with multiple levels of splitting, called the cascaded splitting approach (see Figure 2.4 for an illustration of the cascade architecture), in this case there are different splitters connecting each other before reaching the client side [Papaefthimiou, 2013]. For example, for the equipment outside of the CO, a ratio of 1:4 followed by a 1:8 splitters is also possible. This last splitter (ratio 1:8) could eventually be placed in a PDO directly. These alternate configurations are applied in more specific scenarios, where we might have buildings with more demand. Or even in scenarios where it is required to cover an extended area, for example in Long-Reach Passive Optical Networks [Lin, 2012].

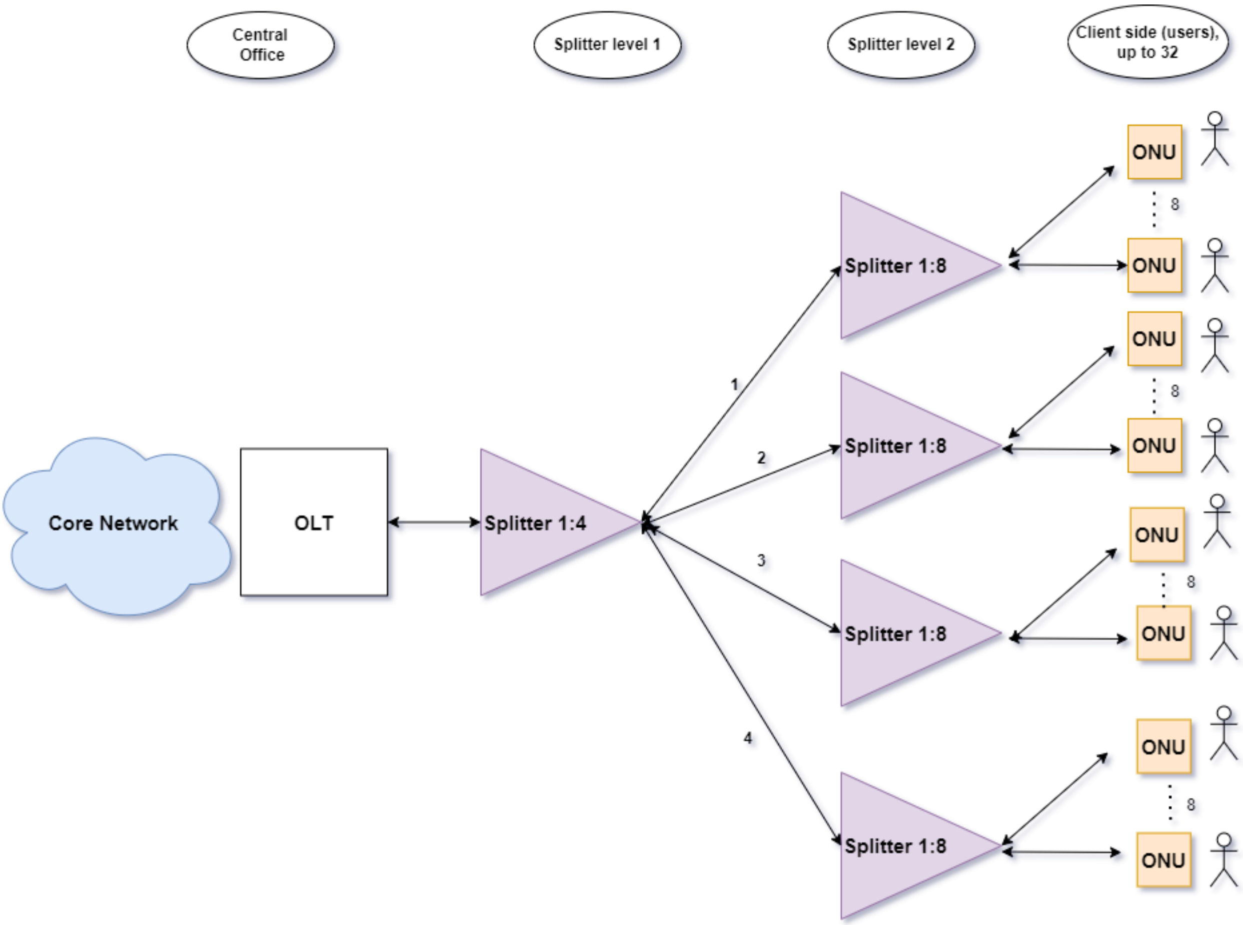


Figure 2.4: Illustration of the GPON/FTTH multi-level (2 levels) architecture (cascaded approach).

2.1.3 Equipment Location

The locations of the network equipment are crucial in the design of networks, due to the high costs of the materials and the manual labour involved. It is important to consider the different equipment factors such as the splitters location, the location of the different ONUs (i.e. the clients), the optical distribution point (PDO) position, the cheapest routes for the cables.

To plan a cost-effective solution, all equipment has to be placed optimally to minimize the amount of cable used in the network and the overall cost.

In the case we are dealing with buildings as the homes (a Fiber-to-the-Building scenario), we might have to consider the fiber demand of that location, it might require a demand of 50 ports for all the homes in the building, while in a simple house we got typically one. This forces the installation of the components like the PDO to be placed directly on the location of the building.

2.1.4 Link Power Budget

The performance of the network is measured by the link power budget, it has to satisfy the GPON class B standard specifications [3] [Ubaidillah et al., 2018].

In a GPON, most of the components can cause loss of signal and for the network to function properly, the loss budget has to be respected. Table 2.1 shows the minimum and maximal loss values for the GPON class B. The values depend on the equipment transmitting power and receiving sensitivity, for our purposes we will limit ourselves to these classes. [4]

In case the network does not respect this loss budget, it might cause loss of information, delays and overall loss of quality of service.

Sources of attenuation

After the loss budget is defined (depends on the equipment used), it is important to consider the different sources of attenuation that can occur in the network, such as:

- Fiber cable attenuation A_{Cable}. This value is fixed to 0.4 dB per km of fiber cable (distribution cable, feeder, drop cable).
- Splitter attenuation $A_{Splitter}$. The splitter attenuation depends on the number of ports, a higher number of ports means a higher attenuation. Table 2.2 shows the loss in dB for each type of splitter.
- Splice attenuation A_{Splice}. This value is fixed to 0.1 dB per splice. Splicing is simply the process of joining two fiber cables together, the fiber cables do

[3] Usage recommendations from ITU-T G984.

[4] https://community.fs.com/blog/overview-of-gpon-technology.html

	Class B	***Class B+***
Minimum loss	10	13
Maximum loss	25	28

Table 2.1: Loss budget for GPON (in dB).

Number of ports in the splitter	**Splitter loss (in dB)**
2	3
4	6
8	9
16	12
32	15
64	18

Table 2.2: Splitter loss depending on the number of ports. Based on [EXFO, 2012].

not come in larger lengths such as 20 km, so multiple splicing is required in a network. Typically in a 20 km network, 4 splices are required (each one in a range of 6 km). [5]

- Connectors attenuation $A_{Connector}$. For the equipment connectors, which are the hardware at the end of a fiber that allows the connection to another component like the transmitter or receiver, we have a loss of 0.35 dB per connector.

- Additional attenuation *Extras*. This corresponds to attenuation from error measurements for example, safety margin or even from future repairs (in splicing) that can introduce loss. We can fix this value to 2.5 dB.

Power budget

With those sources of attenuation taken into consideration, it is now possible to compute the power budget P as follows:

$$P = A_{Cable} * L_{Cable} + A_{Splitter} + A_{Splice} * N_{Splice} + A_{Connector} * N_{Connector} + Extras \tag{2.1}$$

With L_{Cable} the total amount of fiber cable used. N_{Splice} and $N_{Connector}$ the amount of splices and connectors required respectively.

As an example, the Table 2.3 shows the computation for a very simple scenario in the 20km range where there is only one splitter 1 : 32. The result is 26.6 dB (including the extras), it is inferior to the maximum of 28 budget mentioned earlier, so a GPON network in this scenario is valid.

[5] https://www.perle.com/supportfiles/optical-power-budgets.shtml

Equipment	*Attenuation (dB)*	*Amount*	*Total loss (dB)*
Splitter 1:32	15	1	15
Cable loss	0.4	20 km	8
Splices	0.1	4	0.4
Connectors	0.35	2	0.7
Extras	2.5	1	2.5
			Sum: 26.6

Table 2.3: Example of power budget computation.

2.2 Summary of the Problem

Automatically configuring networks can be modelled as a minimization problem where the objective is to reduce the cost required to create the network. In the context of this work and based on the requirements, the focus is to solve and design the last part of the GPON network, more specifically the distribution network and the clients connections.

The main goal is to find the optimal placement of the PDOs to connect the client houses to the network. Those clients have to be assigned properly to reduce costs and to maximize the coverage of the network (i.e., serve all the homes). Additionally, it is important to consider the best possible routes for the fiber cable to avoid excessive costs.

This problem depends on the input information provided and on multiple constraints. This section summarizes the inputs, constraints involved and the mathematical formulation of the minimization problem.

2.2.1 Inputs

The inputs for our problem can be separated into two categories: the infrastructure information and the business rules. The infrastructure information corresponds to the following:

- Locations of the clients;
- Locations of the network equipments and potential installation points;
- Location of the JSO (starting point);
- Possible routes to connect the equipment.

As for the business rules, they correspond to the following:

- Financial costs to build the different parts of the network: distribution fiber cable, drop cable, PDOs;
- Clients demand for fiber, a house has a demand of 1 fiber whereas a building has a demand that can vary.

2.2.2 Constraints

There are a few constraints that have to be respected to design the network such as:

- Limit ports on the PDOs;
- The maximum distance for the network equipment (limit 20km);
- The maximum drop cable distance that links the homes to the a network equipment (a PDO in a pole for example);
- Types of routes to consider in the solution: there is mainly the aerial type and the buried type of route. The latter should be avoided due to the excessive costs and complications related to it (more resources required for the digging of the cable tunnels etc.).
- The optical budget.

2.2.3 Formulation

With the inputs and constraints defined, we can formulate the problem as a minimization problem. Given that we want to find the location of the PDOs equipment (i.e the distribution and drop part of the network based on the architecture mentioned earlier), we are assuming the same cost between the distribution cable and drop cable, and that the attenuation is limited to the fiber cable only.

Table 2.4 summarizes the notation for the variables. Additionally, we need to define a few binary variables to aid in the formulation of the problem:

- $e_{i,j}$: if an edge exists from node i to j, it takes the value 1, else it is 0.
- $drop_{i,j}$: if an edge is a drop cable link, it takes the value 1, else 0 (i.e it is a fiber cable from the distribution part of the network).
- p_k: if a node k is selected to be a PDO, it takes the value 1, else it is 0.

The minimization problem can be expressed as follows:

Minimize

$$C_{Fiber} \sum_{i,j} (e_{i,j} * d_{i,j} * B_{i,j}) + C_{Pdo} \sum_{k} p_k \tag{2.2}$$

$$\forall i \in JSO \cup S_{Pdo}, \forall j \in S_{Pdo} \cup S_{Onu},$$
$$\forall k \in S_{Pdo}$$

Subject to

$$(e_{i,j} * d_{i,j}) \leq MaxDrop \quad \text{if} \quad drop_{i,j} = 1, \tag{2.3}$$

Notation	***Meaning***
S_{Onu}	Set of clients (ONUs)
S_{Pdo}	Set of potential PDO spots
JSO	JSO node (the starting point)
$d_{i,j}$	Distance from node i to node j
C_{Fiber}	Cost of the fiber cable per length unit
C_{Pdo}	Cost of a PDO
$B_{i,j}$	Penalty route factor for node i to j (relevant in case of buried routes, else it is 1)
$MaxDrop$	Maximum drop cable range allowed to connect a client to the network
A_{Fiber}	Attenuation (signal loss) from fiber cable
$MaxAttenuation$	Maximum attenuation

Table 2.4: Notation for the minimization problem

$$\sum_{i,j}(e_{i,j} * d_{i,j}) \leq 20km, \tag{2.4}$$

$$A_{Fiber}\sum_{i,j}(e_{i,j} * d_{i,j}) \leq MaxAttenuation. \tag{2.5}$$

Where Equation 2.2 aims to minimize the cost of the resources to build the network, i.e the cost of fiber cable and the installation points (PDOs). The fiber cable cost is computed by multiplying the length needed by its cost. Additionally, the fiber cable has an additional factor that is only relevant in case of buried routes.

As for the constraints that the problem is subject to, the equation 2.3 limits the drop cable range whereas the Equation 2.4 limits the overall range of the network to 20km. Concerning the optical budget, it is limited in the formulation to the fiber cable, which is represented by the Equation 2.5.

Chapter 3

Background and State-of-the-Art

This chapter aims to discuss a few concepts and provide some theoretical background for the rest of the work. A review of the State-of-the-Art on the topic of Automatic design of networks is proposed as well. The first section covers a few notions from the vast field of Graph theory useful for this thesis. The second section gives an overview about some Nature-inspired metaheuristic approaches: Genetic Algorithms and Swarm Intelligence. The third section briefly discusses the optimization method known as Local Search. Finally, the last section focuses on the State-of-the-Art.

3.1 Graph Theory

As we are dealing with networks, it is crucial to grasp a few concepts from Graph theory in order to proceed. For that purpose, a few key definitions are presented in a first part. Then, some specific algorithms and problems tackled during this work are studied.

3.1.1 Basic Definitions

In this work we consider **undirected graphs** defined by:

Definition 3.1.1 (Graph)**.** a graph $G(V, E)$ is defined by a set of vertices (the nodes) V and a set of edges E.

For each edge connecting two vertices, we can represent them by (v_1, v_2), we say that they are adjacent. Also, we say that the edge is incident to both vertices. In an undirected graph the edges are bidirectional.

Definition 3.1.2 (Degree)**.** The degree of a vertex is defined by the amount of edges incident to it.

Definition 3.1.3 (Path)**.** A path is a sequence of vertices connected to each other, without repeating edges.

Definition 3.1.4 (Cycle)**.** A cycle is a path that starts and ends with the same vertex.

Definition 3.1.5 (Acyclic graph)**.** An acyclic graph is a graph without cycles.

Definition 3.1.6 (Complete graph)**.** A complete graph is a graph in which every pair of distinct vertices is connected by an edge.

Definition 3.1.7 (Subgraph)**.** A graph H is a subgraph of a graph G if all vertices and edges in H are also in G, i.e H is a subset of G.

Definition 3.1.8 (Connected graph)**.** A graph is connected if there is a path for each distinct pair of vertices.

Definition 3.1.9 (Weighted graph)**.** A weighted graph is a graph that contains values associated to the edges.

In this work, those values (the weights of the edges) are positive, they represent distances.

3.1.2 Trees and Minimum Spanning Trees

With the help of the previous definitions we can start defining some more concrete concepts applied during this thesis:

Definition 3.1.10 (Tree)**.** A tree is a connected, acyclic graph.

Definition 3.1.11 (Forest)**.** A forest is an acyclic graph, i.e a disjoint set of trees.

Definition 3.1.12 (Spanning tree)**.** A spanning tree for G is defined by a subgraph which is a tree and contains all the vertices of the graph G.

Since we are dealing with networks represented by weighted undirected graphs, we can then consider the problem of the **Minimum Spanning Tree (MST)**, which is basically a spanning tree of minimal weight/cost. It is worth noting that this MST is not unique, there might be others (with same cost) even though it is unlikely, specially in our context of telecommunication networks.

There are a few different algorithms to obtain the MST, such as the Prim's algorithm.

Prim's algorithm

Prim's algorithm is a greedy algorithm that gives the MST, the main steps are described in Algorithm 1. Basically it starts with a chosen point (which can be a random one) and builds the MST step by step by making locally optimum choices. Local optimum solution corresponds to the solution which is optimal within a small range of nearby possible solution sets.

For that we keep track of two sets, one for the result (MST) and one to keep track of the vertices yet to be added in the MST. The algorithm adds an edge at every step by considering the edges that connect both sets, the edge added is of minimum weight by making sure it does not create a cycle in the MST.

In terms of complexity, it depends of the data structure used. A simple choice is a matrix of adjacency which makes the algorithm complexity of $O(V^2)$ running time. Other choices of representation can be used, for example by using Fibonacci heaps, which in turn makes the algorithm complexity $O(E + VlogV)$ [Cormen].

```
Result: $MST$
Initialize a set $MST$, the result minimal spanning tree;
Initialize a set $S$ to keep track of vertices that are not yet in the tree;
Select a starting vertex (can be random), add it to $MST$ and update $S$;
while $MST$ doesn't contain all the vertices do
    Find the edges that connect $S$ to $MST$ ;
    Pick the minimum among those edges;
    Add it to the $MST$ (as long as it doesn't create a cycle) and update $S$;
end
```

Algorithm 1: Prim algorithm pseudo-code

Other common options to obtain the MST are also available, like the Kruskal's Algorithm which is also a greedy algorithm using heaps. One key difference between those two algorithms is based on the graphs density, Kruskal's performs better in sparse graph whereas Prim's performs better in dense graphs (i.e with many edges).

3.1.3 Classic Graph Theory Problems

There are a few classic problems from graph theory that are particularly relevant to solve the problem of designing networks automatically, more specifically the Shortest path problem and the Steiner tree problem.

Shortest paths

One thing to note is that the Prim's algorithm gives the MST, i.e a tree that connects all nodes in G of minimum total weight. But if we are interested in the shortest path between two distinct nodes, the MST might not be the solution of shortest path.

In order to find the shortest path from a source node to the other nodes in a weighted graph, one of the most common algorithms used is the Dijkstra's algorithm (it builds shortest path tree).

As a concrete example, the Dijkstra's is useful if we are interested in going from a city to another with minimal time. Whereas the Prim's is more interesting in case we want to get a physical connection between all nodes (like cables) of minimum cost.

The Dijkstra's algorithm pseudo-code is given in Algorithm 2. Similarly to Prim's, Dijkstra's algorithm also belongs to the family of greedy algorithms, it builds a solution by considering the most promising choice at a certain moment, in this case the shortest edge.

The algorithm initially begins by assigning a distance of zero to the source node, while the rest are assigned with infinity. It keeps track of the currently known shortest path from the source to each node, updates are made in case a better path is found (all of the nodes have to be visited, this is done by using a queue). A visual example (step by step) is provided in Figure 3.1.

```
Data: Graph G, source s
Result: Shortest paths
foreach vertex v in G do
    -$dist[v]$ ← Infinity # dist array that keeps track of current distances
      from s to other nodes ;
    -$prev[v]$ ← Null # array that keeps track of previous-hops on the
      shortest path from s to a given node;
    -add $v$ to queue;
end
-$dist[s] \leftarrow 0$ ;
while queue is not empty do
    -$u$ ← extract min. from queue ;
    foreach neighbor v of u still in queue do
        -$tmp \leftarrow dist[u] + lengthEdge(u, v)$ # contains the length of the path
          from $s$ to $v$ if passing by $u$;
        if tmp < dist[v] then
            -$dist[v] \leftarrow tmp$ ;
            -$prev[v] \leftarrow u$ ;
        end
    end
end
```

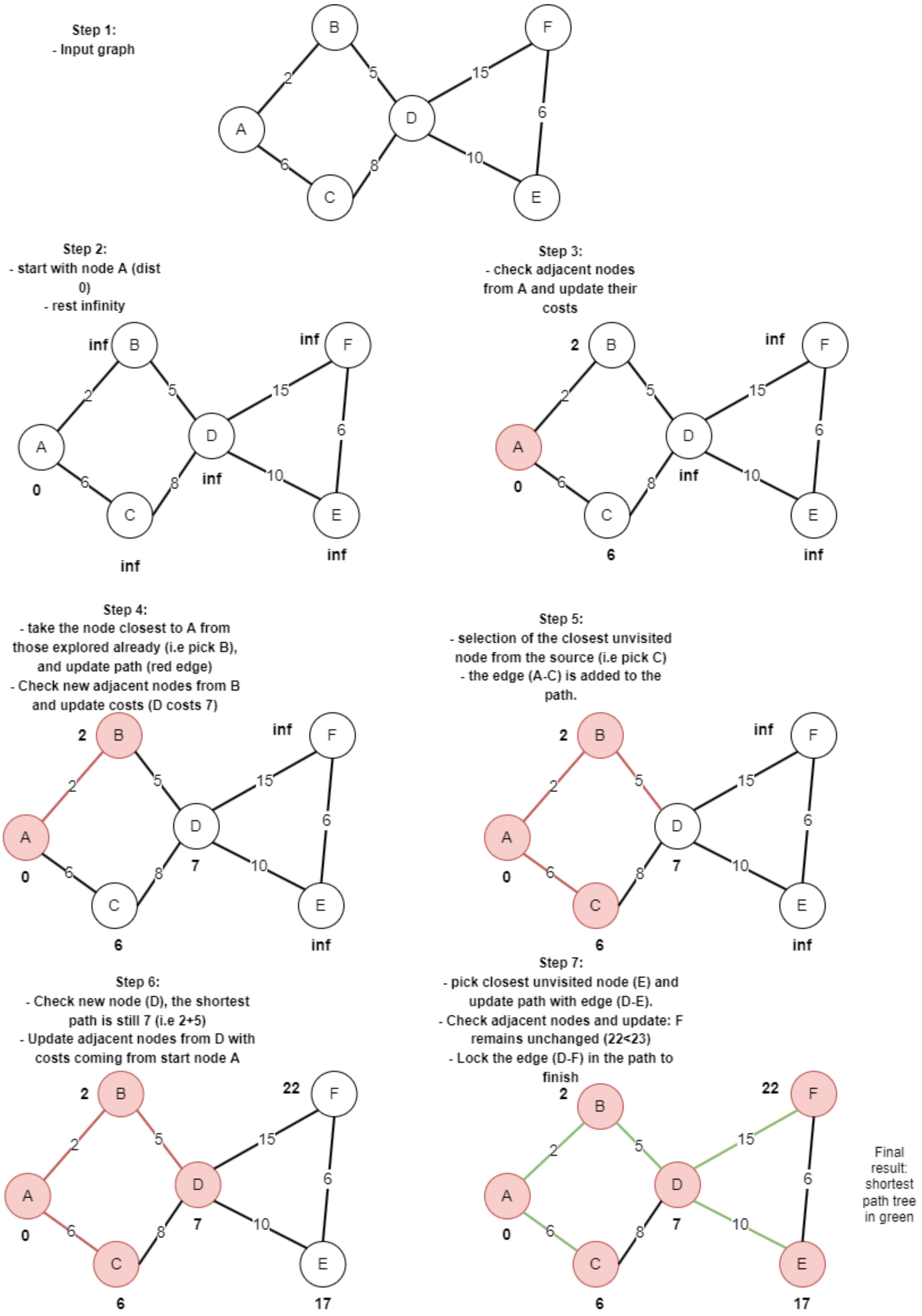
Algorithm 2: Dijkstra's pseudo-code.

Steiner Tree Problem

Another classic problem is the one of the Steiner Tree problem, which also relates to the work at hand, a Steiner tree is described by: given a graph *G* and a subset of nodes, the Steiner tree spans through all the given nodes in that subset, i.e., determine the MST that connects the subset.

There are a few special cases: if the subset is in reality all the nodes, we are back to the MST problem. Also, if the subset is only two nodes, it becomes a shortest path problem.

The vertices/nodes are split into two categories, the terminals and the non terminals. The former is the given subset of nodes, the latter corresponds to the nodes that are not in the subset (but connected to it) but that are included in the solution, i.e they can help build the optimal solution that connects all the terminal nodes.



Step 1:
- Input graph

Step 2:
- start with node A (dist 0)
- rest infinity

Step 3:
- check adjacent nodes from A and update their costs

Step 4:
- take the node closest to A from those explored already (i.e pick B), and update path (red edge)
- Check new adjacent nodes from B and update costs (D costs 7)

Step 5:
- selection of the closest unvisited node from the source (i.e pick C)
- the edge (A-C) is added to the path.

Step 6:
- Check new node (D), the shortest path is still 7 (i.e 2+5)
- Update adjacent nodes from D with costs coming from start node A

Step 7:
- pick closest unvisited node (E) and update path with edge (D-E).
- Check adjacent nodes and update: F remains unchanged (22<23)
- Lock the edge (D-F) in the path to finish

Final result: shortest path tree in green

Figure 3.1: Example of the application of the Dijkstra's Algorithm.

To obtain a MST, a polynomial-time algorithm is sufficient, but for the Steiner Tree problem, it is more complex since we only want a specific set of nodes. It becomes an NP-hard problem, NP-hard means that the problem is at least as hard as any NP-problem (i.e., if it is solvable in polynomial time by a nondeterministic Turing machine).

To solve this kind of problem, approximation algorithms are often considered. To approximate the Steiner Tree, a common approach is to compute the minimum spanning tree of the sub-graph of the metric closure of the graph induced by the terminal vertices. The metric closure of a graph G corresponds to the complete graph in which each edge is weighted by the shortest path distance between the nodes in the original graph G (with the original costs).

Basically once the metric closure is computed (can be done by finding all the pairs shortest paths), we can apply a simple algorithm to find the MST. This kind of algorithm has around a 2-approximation factor (i.e it could be twice as costly as the optimal solution), which is why it is an approximation, there is no guarantee of optimal solution due to the complexity of the problem. A more in depth mathematical analysis of this approach can be found in the literature [Dinitz, 2019].

There is also a variant of this problem named the *Euclidean Steiner minimal trees*: which considers not only a fixed set of vertices, but the whole Euclidean space (infinite) [Wu and Chao].

To conclude on the application of this Steiner Tree problem: it can be useful in problems where we have important locations to connect, some applications in telecommunication networks have been done for example. This is something that can be considered in the current work too, where we can have the terminal set being the clients premises, so we would obtain the MST that contains all the clients nodes.

3.2 Genetic Algorithms

This section describes a few notions about Genetic Algorithms (GA), with an emphasis on the basic design considerations of a GA. This summary is inspired by the works of Costa [2022], and Eiben and Smith [2015].

3.2.1 Overview

Genetic algorithms (GA) are inspired by the process of natural selection, they are population-based stochastic search algorithms used to solve optimization problems. They belong to a larger part of computation known as Evolutionary Computing. Those algorithms are guided by a fitness function based on the problem to solve. Each individual (chromosome) from the population corresponds to a solution in the search space (i.e., the set through which an algorithm searches).

The pseudo algorithm of a basic GA is shown in Algorithm 3, the population is initialized randomly and then the evolutionary steps begins, for each generation, the population is modified with the help of natural mechanisms like mutation of an individual and cross-over (recombination between two selected parents). Then the new population is evaluated based on a fitness function and usually the best individuals (solutions) carry on to the next generation. This is what allows the algorithm to eventually converge after multiple generations. In the end, the individual with the best fitness score represents the best solution (it can be the optimal or almost optimal, there is no guarantee of optimal solution).

```
Result: Best candidate solution
Initialize population with random candidate solutions;
Evaluate each candidate;
while termination condition is not satisfied do
    Select parents;
    Recombine pairs of parents;
    Mutate the resulting offsprings;
    Evaluate new candidates;
    Select individuals to survive to next generation;
end
```

Algorithm 3: GA pseudo-code

In order to work with GA, a few design issues have to be addressed, the solution representation is certainly the most crucial step along with the fitness function choice which is related to the problem. Once that is completed, the other design issues can be solved like the choice of the variation operators (i.e. mutation and cross-over), population initialization, stop criteria etc.

3.2.2 Representation

The representation (genotype) is dependent on the problem at hand and it is often the first design issue to solve in GAs. Usually a good representation leads to better results, and for complex problems it might often introduce a gap when it differs from the real world object (phenotype), in that case we require an encoding/decoding step. Here are a few common representations:

- Binary (string of bits).
- Number-based (integers, floats).
- Permutations, usually adapted for problems with order requirements and constraints.

As an example, we can consider the Travelling salesman problem (TSP) which is a NP-hard problem in combinatorial optimization and that can be solved by the usage of GA. The objective of the TSP is to find the shortest path between N different cities by visiting them once where the starting point is equal to the ending point. For this minimization problem, a classic representation is permutations where each city corresponds to an unique identifier (integer) and the order in the vector corresponds to the order of visit of the cities (Figure 3.2).

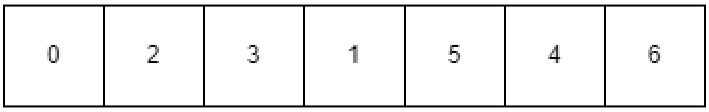


Figure 3.2: Example of genotype for the TSP with $N = 7$ (permutation).

3.2.3 Variation Operators

The main variation operators are the cross-over (recombination) and the mutation operator, the choice of those operators highly depends on the representation previously chosen for the problem to be solved. There are multiple choices of techniques for each variation operator in the literature.

Crossover is a genetic step used to combine the selected parents chromosomes into offsprings (child solution) to be carried on the next generation, the idea is to mix the parents genetic material to create a new solution (with better fitness ideally).

Mutation consists in perturbing parts of one solution randomly, this leads to higher diversity of the population (to differ from the initial pool of individuals) and is useful in order to avoid local optimum problem (which corresponds to the fact that we might have the best solution for a small possible set of solutions, but there might be a better one in the overall search space).

In the continuity of the previous example, here are a few applications of those variation operators for the TSP (as a permutation):

- Swap mutation: consists in selecting randomly two genes and swapping them (Figure 3.3).

- Order crossover (Figure 3.4) which consists in taking a subset of one parent $P1$ and put it into child C in the exact same location index. Then we put the rest of the genes from $P2$ in order of appearance and by avoiding repetitions (by starting at the right of the introduced part from $P1$).

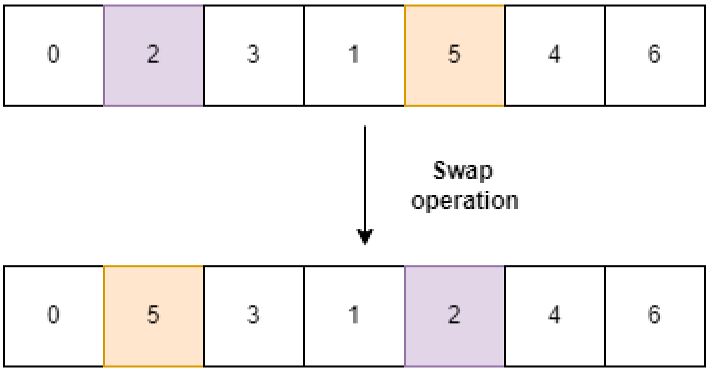


Figure 3.3: Example of swap operation (mutation) for the TSP with $N = 7$.

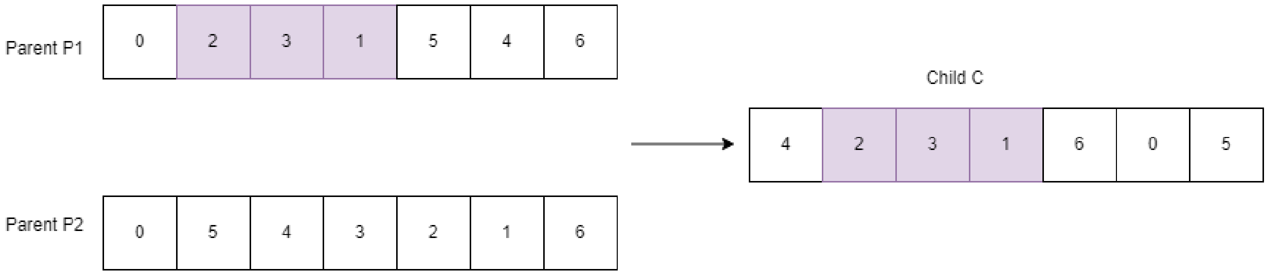


Figure 3.4: Example of ordered crossover for the TSP with $N = 7$.

3.2.4 Selection Steps

There are two main selection steps in GAs, the selection of parents which then leads to the changes created by the variation operators described previously and the selection of survivors to move on the next generation.

The selection of the parents is done in order to perform the reproduction step (crossover). There are many methods to perform this selection, usually fitness proportional. For example the most common one, the Roulette-wheel selection, which gives a probability to each individual of being selected to be a parent based on its fitness. With this method, the most fitted individuals have higher odds to be chosen to be the parents, it is the selective pressure on the best individuals of the population. This can also be a drawback sometimes, the selective pressure can lead to premature convergence to a local optimum which should be avoided.

Concerning the selection of survivals, it refers to the individuals from the new population (*current population + new offsprings*) that get the opportunity to move on the next generation. Once again this step can be done based on the fitness of each individual, but there are also other options, for example elitism which will keep a constant number of best individuals, age-based i.e an individual can only live up to a certain amount of generations, then he will be replaced etc.

3.2.5 Other Design Issues

Some other important design aspect to consider is related to the termination criteria for the GA, it can be related to the number of generations, the population diversity, CPU-time or even fitness related (stop after a certain threshold or after stagnation during multiple generations).

Also, there is the step concerned with the initial population, it is usually generated randomly, but there are other options, like the heuristic method which requires additional knowledge about the problem in order to initialize the population. This is called *hybridization*, which combines other techniques in order to improve the algorithm.

3.2.6 Improvements

In the case of the initial population, the hybridization consisted on providing an heuristic, but it can also be applied on other steps of the GA, for example using Local Search after the variation operators which can guide the evolutionary process to better solutions.

Also it is important to keep in mind that the usage of GA highly depends on the problem, for example in the case of a constrained optimization problem, we may have a restricted feasible space of solutions (in the whole search space) due to constraints that have to be satisfied by each candidate solution. One way to deal with this kind of problem is by using penalty functions (i.e., we reduce the fitness of the individuals that do not satisfy the constraints). Another common method consists in fixing/repairing the solution that is infeasible by modifying it until it fits in the feasible region of the solution space.

3.3 Swarm Intelligence

Swarm Intelligence (SI) is a type of AI algorithms defined by the collective behavior of multiple agents (nature inspired, like animals) interacting with each other with simple mechanics. Those algorithms have the following properties:

- Agents perform simple tasks (typically).
- Decentralized system, there is no central control.
- Agents can communicate with each other, directly or indirectly with the help of a few mechanisms.

So, to design such algorithms, we have to consider the behavior of a single agent and their interactions as well. There are multiple sources of inspiration from the nature for those algorithms: such as the ant foraging, bird flocking, bees etc. They can be applied on multiple optimization problems like planning, routing (vehicle routing problem for example), TSP and more.

Ant colony optimization

One of the most common SI algorithms is the Ant Colony Optimization (ACO), inspired on the foraging behavior of ants whose task is to look for food. It does so by exploring the environment and depositing pheromone in the ground, they communicate through the pheromone trails and base their decisions on the pheromone levels.

In the modeled system (Algorithm 4) their decisions can be based on heuristics like the highest pheromone level, or more typically, in a stochastic manner, i.e., the higher the pheromone level the higher the chance to pick that path.

The update of the pheromone levels can be made in different ways: directly after each movement of an ant, or after every ant has made its move. Also, other considerations can be taken into account, like the choice of which ants contribute to the update of the pheromones, it can follow an elitist approach based on the quality of solution for example.

```
Result: Solution
Initialize the pheromone levels;
while termination criteria is not satisfied do
    foreach ant going from a point $P$ to another do
        Consider the options to build a solution based on the pheromone
          trail (which can be done stochastically or heuristic based). ;
        Check quality of solution. ;
    end
    Update pheromone levels based on the traversal of the ants (deposit);
    Update pheromone levels by evaporation. ;
end
```

Algorithm 4: Foraging simulation

In the end we obtain a system where the best solution will have the highest pheromone level, and eventually all the agents will converge towards that solution (i.e., taking the shortest path towards the food location).

3.4 Local Search Optimization

Local search is a well known heuristic method (i.e., it can provide a good enough solution faster than classic methods) used to solve optimization problems, it relies on the concept of neighborhood. The neighborhood of a solution s corresponds to the set of solutions that are "close" to s.

Basically, this optimization method explores the search space (possible solutions) sequentially, going from a current candidate solution to another in the neighborhood. There are a few techniques based on this such as the Gradient methods (descent) where we look for a neighbor whose cost is the lowest and we replace the current solution. It is adapted for when the objective function is continuous where you can compute the gradient given a state. But there is also techniques such as Hill Climbing that are adapted for discrete neighborhoods. In this work we take more interest on the latter due to the nature of our problem (the usage of this approach to deal with an allocation sub-problem is described on the next Chapter).

Hill Climbing

The Hill Climbing pseudo-code is shown in Algorithm 5. As can be seen, the basic algorithm has a greedy approach, the search goes towards the region where the function is optimized (an heuristic function to rank the solutions and states).

With this approach there are a few issues, the algorithm might not reach the global optimum if it enters in certain states like a local optimum. The Figure 3.5 illustrates a few different scenarios where we might not reach the best solution (in case we want to maximize an objective function for example).

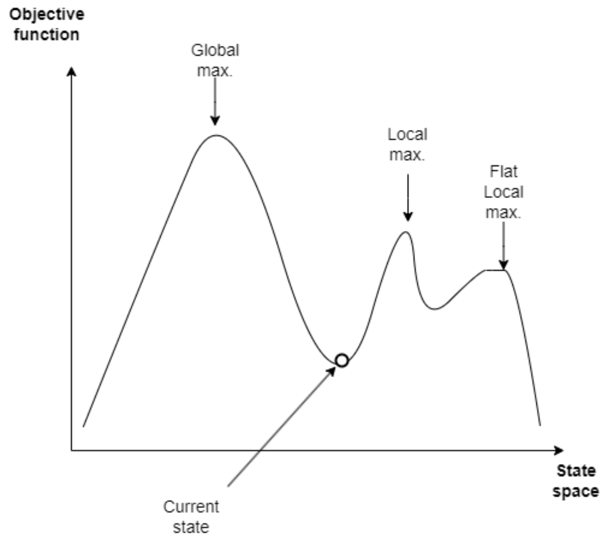


Figure 3.5: State space - Hill Climbing.

To deal with local maximum problem, a few modifications to the Hill Climbing algorithm can be considered to improve the quality of the solution, like a random restart that consists in starting at a random point of the search space until there is no more progress (this might require more iterations of the algorithm).

There are other options to deal with local optimum like Simulated Annealing (a variant of Hill Climbing) which consists in allowing the choice of steps that decrease the optimality of the current solution in order to avoid paths that can lead to local optimum.

As for the flat local problem, a way to deal with it consists in taking big steps in the search space far away from the current state.

```
Result: Solution
Evaluate the initial state, if it is the goal state, return, else continue with
  this current state ;
while goal is not found or state does not change do
    -Select a new state from neighborhood of the current state ;
    -Evaluate the new state ;
    if new state is the goal then
        -Return;
    end
    else if new state is better than current one then
        -Replace current state with the new one and proceed ;
    else
        -Continue until a solution is found;
    end
end
```

Algorithm 5: Hill Climbing pseudo-code

3.5 State-of-the-Art

The aim of this Section is to review the works in the topic of Automatic Design of Networks available in the literature. These works are often divided in two main approaches:

- Nature-inspired Meta-heuristic approaches;
- Math-oriented solutions such as Linear Programming (LP).

The first approach is a nature inspired approach used to find solutions (or approximations) to optimization and search problems, such as Genetic Algorithms (GA) or Ant-Colony optimization. The second approach is an optimization technique for a system dealing with linear constraints and objective function, in the literature the works usually make use of variants of this approach like Integer Linear Programming (ILP) or Mixed Integer Linear Programming (MILP). Table 3.1 summarizes the works reviewed in this section.

3.5.1 Design of Networks based on Meta-heuristics

The problem of designing fiber networks is often tackled with meta-heuristics techniques.

In Tany [2009], an application of a Genetic Algorithm combined with Graph theory to design PONs for different network topologies (bus, tree and ring) is proposed for a multiple level network (i.e., with multiple splitters levels). The main objective was to minimize the cost of implementation of the network with a focus on optimizing the material needed. Like the number of splitters, their geographical location, their split ratio and the cable length. For that purpose the authors solution for the GA was based on the paths between the OLT (the shared equipment in the central office) to each ONU (the homes) obtained with the help of graph theory algorithms.

Another similar work was performed in Kokungal and Ari [2011], where a GA is applied to solve the PON optimization problem with multiple constraints, once again a multiple level splitter design was considered. An interesting representation was used in this approach, a single individual is represented by a double string (where the order of the elements matter), one string for the splitters equipment location (primary and secondary level) and ratio and another for the ONUs locations. The size of those strings depends on the number of ONUs to satisfy and the available material/splitters. This kind of representation allows to automatically satisfy a certain amount of constraints. The author also did a comparison work between the GA approach and an ILP approach, both seem to find exact solutions for smaller networks. But as the instance of the problem increases, ILP solution falls behind in terms of execution time for an acceptable solution where the GA prevails, even though it is not certain that the global optimal solution is found.

A more recent work proposed by Dias et al. [2022], also detailed an application of the GA in combination with Graph theory to solve the problem of design of PONs. Four different topologies were considered, using different configurations of splitters, including both centralized and cascaded approaches. The representation choice for the GA consisted in a binary representation to find the locations to place the equipment amongst a set of potential candidate spots. Their system was tested in real networks from Brazil, where the authors compare the application of the different topologies for different maps. Overall it depends on the type of map, dense and non dense maps were considered, for example in scenarios where there is less clients, a cascaded approach might be more adapted to reduce cable usage. The optical budget is also taken into account in the solutions to check the validity of the networks. A Capital Expenditure (CAPEX) is also studied in this work.

Other types of meta-heuristics approaches have also been used to address this problem of designing networks, for example in Andrej et al. [2012] a solution using Ant Colony Optimization (ACO) is used to design GPON/FTTH for a greenfield (i.e., deployment of a network to a fresh new site) with the input requirements of the potential locations for the materials (splitters and cable distribution locations). Their method allows to approach near optimal solution, they compared with exact solutions solvers (simplex algorithms) and the difference in terms of solution quality is less than 1%, with the advantage of a much better execution time for larger instances ($50 - 90\%$ faster). Multiple additional steps were required to achieve this result, like softening the constraints and post-optimization. In Andrej et al. [2013] the same authors propose a similar approach with ACO with extra meta-heuristics, but this time for a problem evolving additional equipment (aggregating equipment/drop closures).

In addition, in Ali et al., a Local Search approach was introduced. A Local Search procedure will modify locally the candidate solutions until no improvement is detected (or other stop criteria). In this work those modifications can be simply assignment of a client to new network equipment, or swapping the connection (link) between two clients etc. As for the guided aspect, it is basically a penalty function for when the algorithm is stuck in a local optimum, the objective function will be changed (i.e., the weights of the features present in the current solution will be modified). This approach is compared to Simulated-annealing one and seems to outperform the latter significantly.

Finally, there are also applications of AI in other sub problems of telecommunications networks. In Mata [2018], the author proposes a survey of the vast applications of AI for telecommunication problems. Some of the referenced works are worth to mention, for example Morais et al. [2011] which proposes a GA algorithm to deal with topology design with the consideration of network survability. Network survavibility refers to the capacity of the network to operate under presence of equipment failures. Once more, ILP approaches are also used in order to benchmark their approach.

3.5.2 Design of Networks based on LP

Most solutions proposed until now made use of meta-heuristics to guide the search of a solution. Those solutions can be optimal or near optimal, but other types of approaches can be considered, more math oriented, like LP and its variants. Those solutions can often result in exact solutions. But this comes at a cost, often related to execution time for larger instances of the problem.

A decent amount of works can be found in the literature that make use of LP. For example in Chardy et al. [2012], the author studies the PON/FTTH problem for an existing infrastructure using mixed integer linear programming. The algorithm is applied to large (thousands of nodes) but low density urban environments in France with real practice datasets. The author describes the model used and the different approaches considered to reduce the complexity of the problem, such as graph reduction techniques to reduce the number of nodes (low density scenario). Those reductions are made to increase the performance of the ILP approach (and to make it easier).

Some approaches making usage of hybrid methods are also available in the litterature, such as Eira et al. [2012] where the authors tackle the problem of designing a single-stage and multiple-stage splitting PON (focused on tree topology) with ILP. However, to deal with problems that are too complex for optimal methods, a two-stage heuristic approach is considered. The concept consists on applying ILP in a first step to obtain an initial network configuration. Then to attempt to reduce the overall cost of the network by applying local modifications (heuristic approach). For the multiple-stage splitting network, the approach proposed by the authors allowed to reach a solution within 7% of the optimum solution, but overall the data sets were relatively small sized (the number of candidate locations for the splitter equipment was limited to 50).

Similarly, in Hoang [2014], another formulation of an ILP problem to deal with FTTH network optimization for exact solutions is proposed. According to the author, this approach is only relevant for small instances of the problem. So, the author also deals with the problem for larger instances by proposing an approximation solution, a variant of its own solution.

We can start to distinguish a pattern in those related works, the LP variants are used to deal with small instances of the problem. However, in real problems, bigger instances of the problem are required to be solved, in that case the authors of the works tend to use approximate solutions to generate a satisfactory solution. The exact methods are still relevant, usually as a benchmark whenever it is possible (and to solve small problems), or are eventually used in combination with other approaches that reduce the complexity of the problem at hand.

In conclusion, for our purposes, the approximation approaches are more adapted as we aim to obtain a scalable solution. Moreover, an approach based on Genetic Algorithms, that will be explained in the next chapter, should be sufficient, due to the fact that the system developed is supposed to be a tool to aid the designers/engineers (users) in the process of planning the fiber network so a perfectly exact solution is not mandatory.

Table 3.1: Related works recap.

Approach	Approach	Problem solved	Author and title
Meta-Heuristic	- GA - Graph theory	-PON/FTTH planning - multi-level problem	Tany et al, 2009 - *Design of PON using genetic Algorithms*
	- GA		Kokungal and Ari, 2011 - *Optimization of passive optical network planning*
	- ACO	- GPON/FTTH design - multi-level problem - applied to a greenfield	Andrej Chu et al, 2012/2013 - *An enhanced ant colony optimization for ftth access network design*
	- ACO and extra meta-heuristics		
	- Guided Local search - also compares with Simmulated Annealing	- Optimize networks, minimize CAPEX - Tree based solution - Detailed cost / bill (quantity of material...)	Ali Rais et al. - *Guided Local Search for Optimal GPON/FTTP Network Design*
	- GA - Graph theory	- PON optimization - Multiple topologies and configurations - Real networks	Dias et al, 2022, - *Evolutionary strategy for practical designof passive optical networks*
Math oriented approaches (LP)	- ILP - Local search	- FTTH design. - single stage and multi-stage	Eira et al, 2012, - *Optimized design of multistage passive optical networks*
	ILP	-FTTH opt. multilevel problem	Chardy, 2011, - *Optimizing splitter and fiber location in a multilevel optical FTTH network*
	Approximation algorithm and ILP	- FTTH opt. problem	Hoang Nghia, 2014 , -*FTTH Network Optimization*

Chapter 4

Approach

This chapter presents the proposed approach to solve the problem of automatically design of networks. It details how each sub-problem is solved in the computational system proposed.

Section 4.1 presents the process related to the exploration of the data provided by our collaborator. Then, Section 4.2 mentions the preliminary approach used to attempt to tackle the problem of design of networks. Section 4.3 presents the Genetic Algorithm (GA) used, which is the core of the proposed system. Finally, Section 4.4 proposes a memetic algorithm, where the GA is complemented with a Local Search, to address the capacitated sub-problem related to the links between the homes and the installation points (PDOs).

4.1 Data Exploration

The process of exploring and analyzing the datasets provided by our collaborator was crucial to create a proper model to build the automating design system. This section describes briefly the input data involved and the assumptions considered.

4.1.1 Description of the Data

The input data consists mainly of one file in the JSON format that contains the network elements. Namely:

- The JSO, i.e., the starting point which represents a set of splitters.
- The geographical coordinates (latitude, longitude) of the different network equipment and the clients homes.
- The possible existing routes that link the different elements of the network (possible routes to place the distribution cable). These routes represent existing infrastructure on site that is not yet fiber cable (it could be for example copper links).

Some additional information is given such as the demand of each client. In the case of a Multi Dwelling units (MDU), which represents a building, we might have a higher demand for fiber, whereas for a Single Dwelling units (SDU) which represents a house, the demand is often one. The type of each route is also provided, which is an important aspect to take into account when creating a cost efficient design for a GPON network. Making choices that avoid extra costs is crucial. The type of route "aerial" should be preferred over the type "buried", since are higher the costs related to the creation of the underground infrastructure.

This input data is loaded into the computational system alongside the business rules, which consist of the different constraints and material costs to build the network. The costs are of great importance due to their impact on the solution.

A visual example of a dataset can be seen in the Figure 4.1. We can see the initial state of a network, where there are only the existing routes (black edges) from the data set that connect the network equipment (in blue) and the starting point (in red). Additionally, there are the homes (in orange or purple) that are not yet connected to the network.

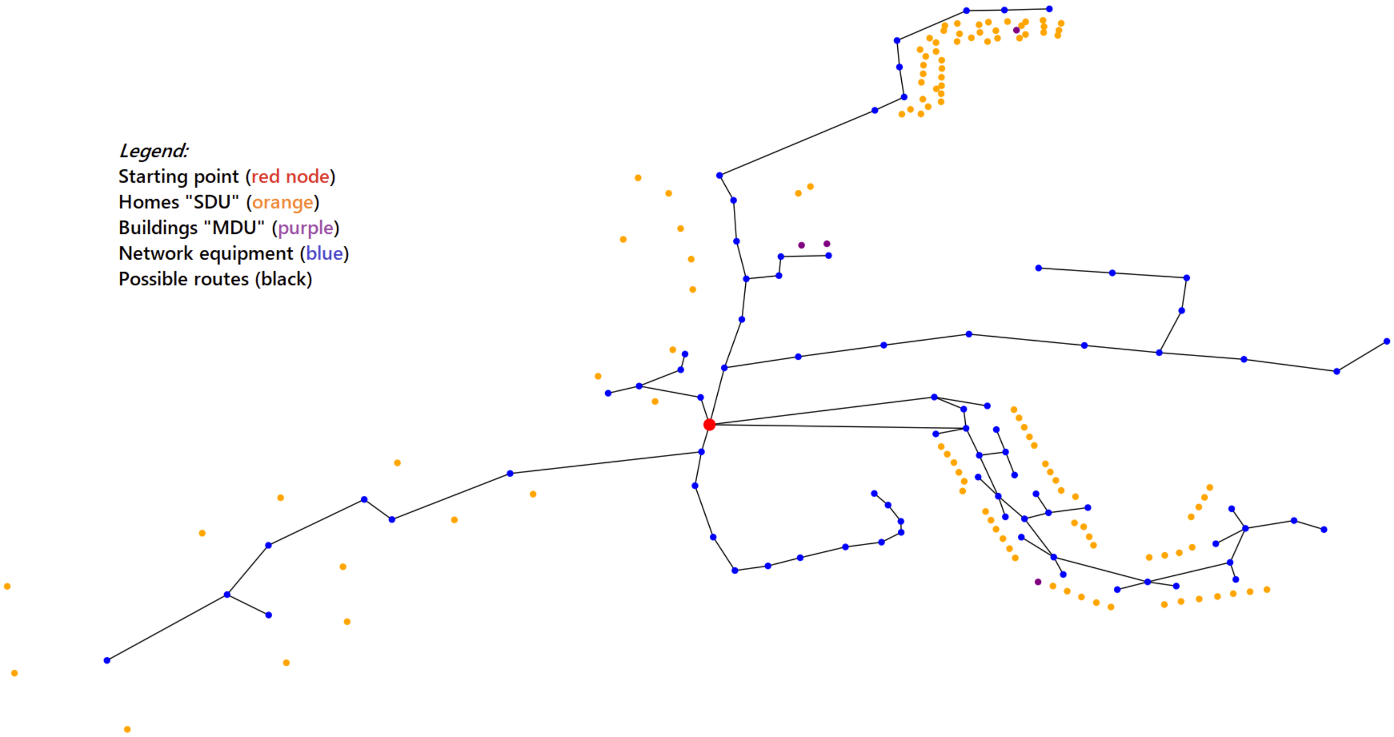


Figure 4.1: Example of a network - initial state.

4.1.2 Assumptions and pre-processing of the Data

These datasets represent real networks, or portions of them, which naturally can introduce inconsistencies. For example the presence of isolated network equipment nodes (like poles, pedestals) that are not linked to the main component (i.e., the routes for those isolated elements are non existent). As such, some assumptions have to be considered during the initial phase of the system:

- Discard the isolated network equipment due to the missing routes and infrastructure. This is done by extracting the main connected component of the network, which corresponds to the sub-graph containing the starting point.
- The only infrastructure that can be added to the network (i.e., nodes or edges) is the drop cables that connect the homes to the network equipment.
- Due to the presence of geographical coordinates, the distance metric used was the Haversine metric. It computes the angular distance between two points on a sphere.

Figure 4.2 summarizes the steps of the initial phase. It is also important to compute the possible distances for the drop cables and store them into memory for later usage, to avoid repetitive computations in the midst of the optimization phase (which will be discussed on the following sections of this chapter).

Finally, if necessary (choice of the user or eventually by missing data), it is possible to compute the starting point (JSO) which is done by taking into account the influence of the homes to find the center of the graph with respect to the clients homes only. For that, the Dijkstra's algorithm is used to compute the shortest paths between all network equipment nodes to all the homes, and the one that returns the best score in terms of distance (minimization) is the new starting point.

4.2 Preliminary Approach

The first optimization approach considered is based on Graph Theory and heuristics. Concretely, it is an application of the Prim's algorithm, which is a greedy approach, to compute the Minimum Spanning Tree. This allow us to obtain a network of minimal cost (with the objective to reduce the fiber cable used). In combination with simple heuristics (e.g., closest point) to connect the homes to the network equipment, we can obtain a solution to the problem of design of networks. The overview of the process is described in Algorithm 6.

```
Data: Network data and geographical coordinates
Result: Solution/plot
- Load data set: network information and geographical coordinates;
- Apply pre-processing steps;
- Obtain the Minimal spanning tree (application of the Prim's algorithm);
- Locate ONUs (homes) and use heuristic: link to the closest network
  equipment that is part of the main connected component of the network,
  i.e. we avoid linking to a pole that is isolated for example.
```

Algorithm 6: Pseudo-code of the process for the first approach/experiment

This simple approach is not the most optimal for a few reasons:

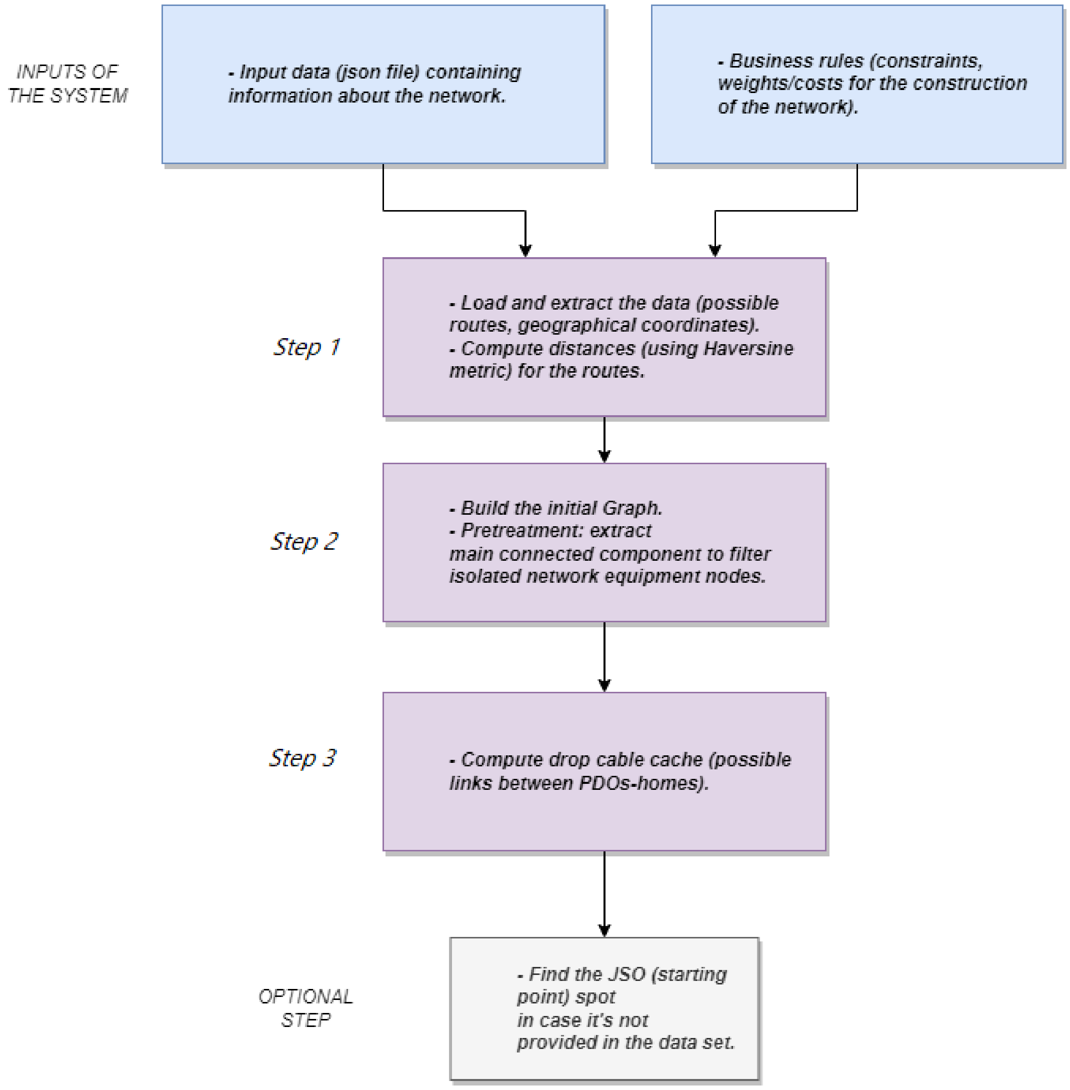

INPUTS OF THE SYSTEM
- Input data (json file) containing information about the network.
- Business rules (constraints, weights/costs for the construction of the network).
Step 1
- Load and extract the data (possible routes, geographical coordinates).
- Compute distances (using Haversine metric) for the routes.
Step 2
- Build the initial Graph.
- Pretreatment: extract main connected component to filter isolated network equipment nodes.
Step 3
- Compute drop cable cache (possible links between PDOs-homes).
OPTIONAL STEP
- Find the JSO (starting point) spot in case it's not provided in the data set.

Figure 4.2: Initial phase of the system.

- All the nodes (network equipment) are being covered, which might not be necessary due to the fact that certain regions of the network do not have houses nearby;

- Depending on the costs provided for the materials (cable costs, PDO cost), the heuristic of the closest point might not be the most optimal due to the fact that it places too many PDOs, which results in increased costs.

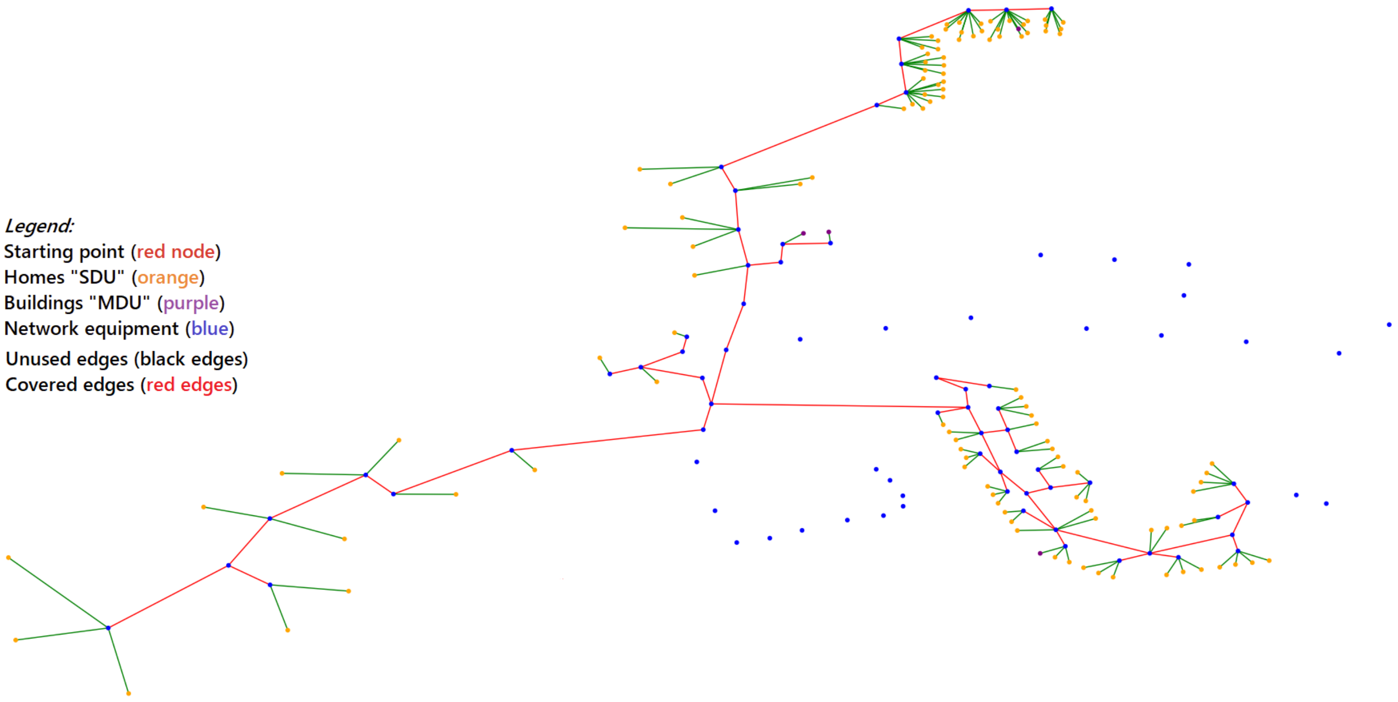

Legend:
Starting point (red node)
Homes "SDU" (orange)
Buildings "MDU" (purple)
Network equipment (blue)
Unused edges (black edges)
Covered edges (red edges)

Figure 4.3: Result of the application of the Steiner Tree to one of the data sets.

One way to overcome this first shortcoming is to consider a Steiner Tree, in which case we can obtain a tree of minimal cost that covers only a specific set of terminal nodes (in this case it is the clients homes). By doing this we reduce the graph, keeping only the relevant zones of the network where optical fiber cable could be installed. An example of the application of the Steiner Tree can be seen in the Figure 4.3 (it also showcases the issue where we have too many PDOs installed and that sometimes we have a single PDO for a single home, which is not optimal).

However, this approach has other limitations, more specifically in terms of scalability when executed in larger datasets. Additionally, the potential to explore new configurations to the network design is also limited in comparison to other approaches like Genetic Algorithms, which will be described in more detail in the next section.

4.3 Genetic Algorithm

As for the core optimization approach, we developed a system based on Genetic Algorithms to search for a good solution to the problem of design of networks. As seen in the Chapter 3, some design considerations have to be made in order to obtain a robust GA. This section will provide an overview of the system developed and it will also outline the principal algorithmic design choices considered.

4.3.1 Overview of the Evolutionary System

As mentioned earlier, the system proposed aims to deal with the multiple constraints imposed by the problem of design of networks. For that, it is important to have a flexible and robust algorithm which should also allow the design of multiple distinct solutions that the user can select from.

For those reasons a GA was selected to solve this problem, since its optimizing process relies in exploring a large amount of solutions that will evolve during multiple generations and where the most fit solutions thrive.

As a remainder, we can summarize the main features and constraints that are managed by the GA step of the automating design process, which begins after the initial phase described in the Section 4.1:

- Find the placement of the network equipment, more specifically the optical distribution point positions (PDOs), to connect the clients homes to the network. The PDOs are capacitated, i.e they possess a limited amount of ports available.
- The routes/edges to consider in order to minimize the amount of resources needed to cover the totality of clients. This also means that the routes that are "buried" (more expensive) should be avoided whenever it's possible.
- Demand of each client is considered (1 unit of demand for fiber corresponds to 1 port allocated to the client). Moreover, in case that we have buildings, the placement of the PDO is expected to be on site (closest network equipment node) for "on field" practicality purposes, it also reduces the amount of drop cable in case of a building to do so, since the demand is much higher.
- Find the optimal drop cable links for the capacitated problem related to the PDOs ports.
- Constraints related to maximum distance for the drop cables (limit the amount of cable that can be used to link a home to a PDO) and for the distribution cables (in order to restrict the size of the network and to respect the optical budget).
- Design impacted by weights (material costs) that can provide multiple configurations.

- Constraints related to intersections (a drop cable cannot intersect with a distribution cable). This is an optional constraint, but it can be specially relevant in case of only aerial cables.

The workflow of the evolutionary process is summarized in the Figure 4.4.

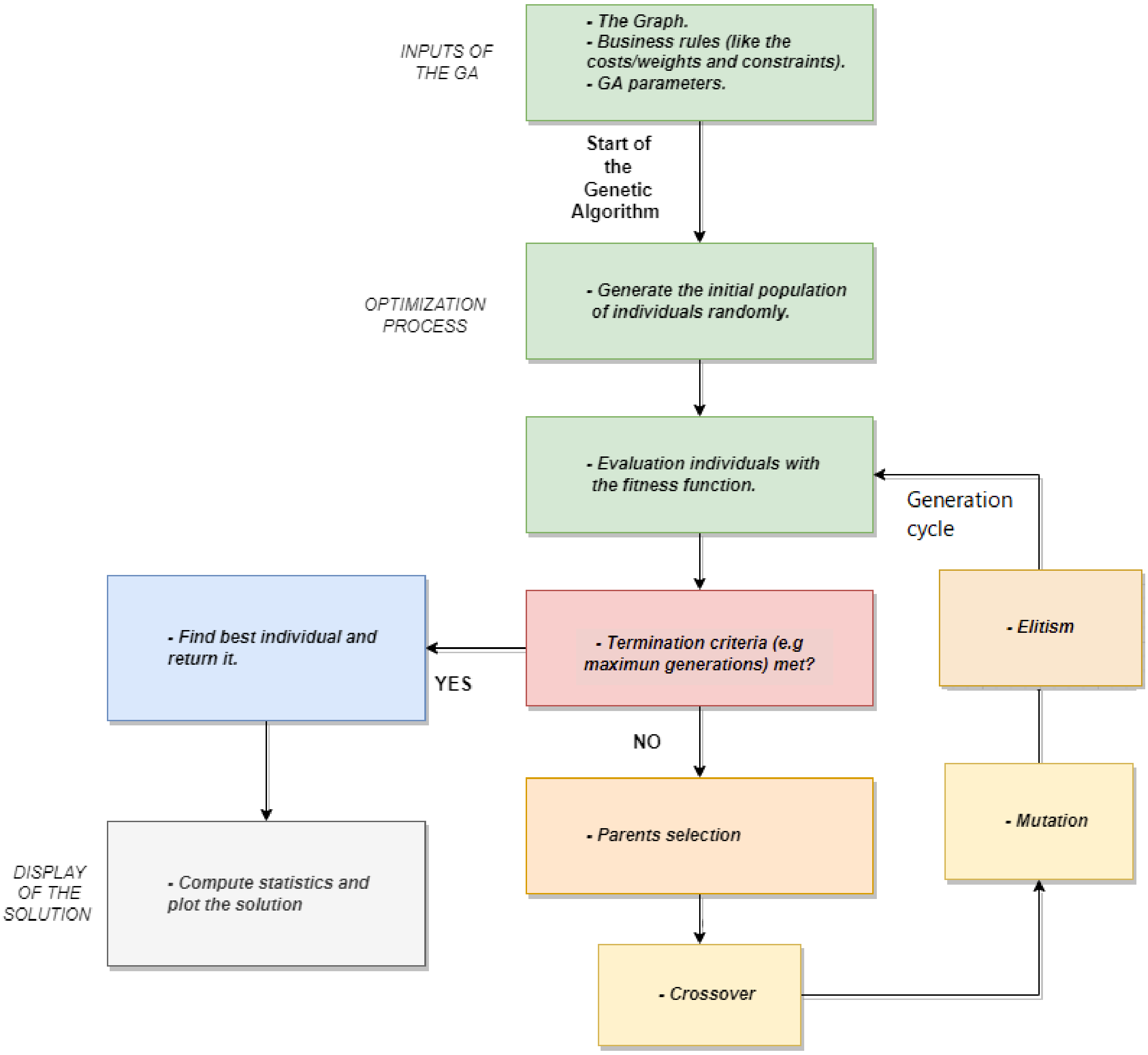

INPUTS OF THE GA
- The Graph.
- Business rules (like the costs/weights and constraints).
- GA parameters.
Start of the Genetic Algorithm
OPTIMIZATION PROCESS
- Generate the initial population of individuals randomly.
- Evaluation individuals with the fitness function.
Generation cycle
- Termination criteria (e.g maximun generations) met?
YES
NO
- Find best individual and return it.
DISPLAY OF THE SOLUTION
- Compute statistics and plot the solution
- Parents selection
- Crossover
- Mutation
- Elitism

Figure 4.4: Workflow of the optimization phase with the Genetic Algorithm.

4.3.2 Design Choices

As explained in Chapter 3, it is fundamental to design the GA properly in order to obtain reliable results and a scalable system. The design choices are the following:

Representation

The representation is one of the most important design considerations for the GA. In our case, each individual of the population is composed by two arrays:

- Array 1 - Binary representation where each index corresponds to a network equipment candidate to the placement of a PDO. If the value is one, it means that the PDO is placed on that equipment.

- Array 2 - Integer based representation used to manage the drop cables links for the capacitated sub-problem related to the limited ports of the PDOs. Each index corresponds to a different client and the values within correspond to the associated PDO (node id) for that specific client home. If a house is not assigned to a PDO the value is -1. This array is dependent on the first binary representation, where a modification of the first might impact the second (it will be discussed in more detail in the next section).

Variation operators

Once the representation is defined, we need to define the variation operators: mutation and crossover. Those operators are applied to the binary array. The main purpose of the mutation is to modify the population individuals randomly by changing some of the genetic material. This allows the population to remain diverse and to avoid the problem of premature convergence towards local optimums. Three design options were considered:

- Bit Flip mutation, which consists in simply modifying a bit from the first representation array, i.e., a 1 can become a 0 with a fixed probability and vice-versa.

- Swap mutation, which consists in selecting two indexes in the binary representation and for each one, we assign the value of the other (swap the values), this option might not modify the genetic material if the values are the same.

- Double Bit flip, which is similar to the first option, but it modifies two bits from the first representation array.

Figure 4.5 illustrates the application of the mutation. At this initial state of the network (that corresponds to the phenotype), there are no PDOs in place yet and no drop cables. The representation (genotype) that corresponds to this scenario is the following:

- Array 1 (binary): [0, 0, 0, 0, 0]. An array of size 5 (blue dots corresponding to the network equipment like poles for example) containing only zeros due to the fact that there is no PDOs installed.

- Array 2 (integer): [-1, -1, -1, -1, -1, -1]. An array of size 6 (orange dots corresponding to the homes of the clients). It is all -1, since there is no drop cables (no connections homes-PDOs based on the array 1).

We can notice that it is a bad solution, but over the course of multiple generations, it can be improved, for example by applying the mutation on the representation 1. Let us assume that the indexes 1 and 4 are mutated (flipped, i.e the zeros become ones), the solution now becomes the Figure 4.6, which is the optimal for this scenario. The genotype is modified like so:

- Array 1 (binary): [1, 0, 0, 1, 0]. The network equipment 1 and 4 now have a PDO (grey nodes) installed and can provide fiber to the homes nearby.
- Array 2 (integer): [4, 4, 1, 1, 1, 1]. The homes *a* and *b* are now linked with drop cable (green edges) to the PDO on the node 4 and the homes *c*, *d*, *e*, *f* are linked to the PDO on node 1.

Similarly, the crossover operator also allows to vary the population at each generation by combining the genetic material of two distinct individuals (the parents) and obtaining a new solution. The function for the crossover operator is the uniform crossover.

It is important to remind that in the evolutionary system proposed, the variation operators modify the representation 1 which can require an update of the representation 2 since we are potentially modifying the placement of a PDO and some homes might need to be reinspected to check their links. Both those operators are applied for a selected pool of individuals (the parents) based on fitness scores.

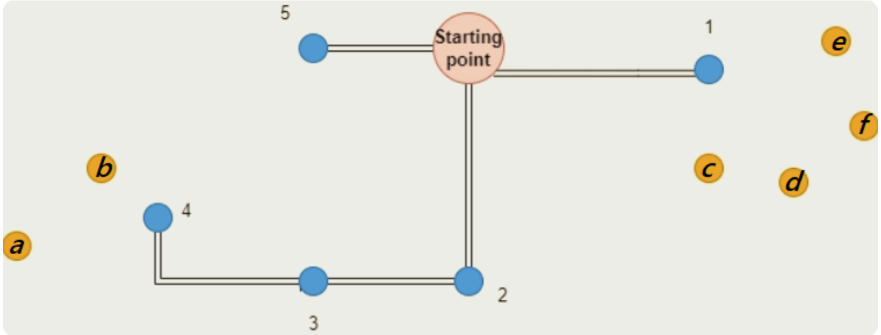


Figure 4.5: Example of a network - before.

Fitness function

Another key aspect of the GA is the fitness function which is used to evaluate the performance of each solution. In the evolutionary system proposed it evaluates the population initially and at every generation cycle. It represents the financial cost to build the network, where the objective is to minimize this cost. Figure 4.7 describes the steps needed to evaluate a solution.

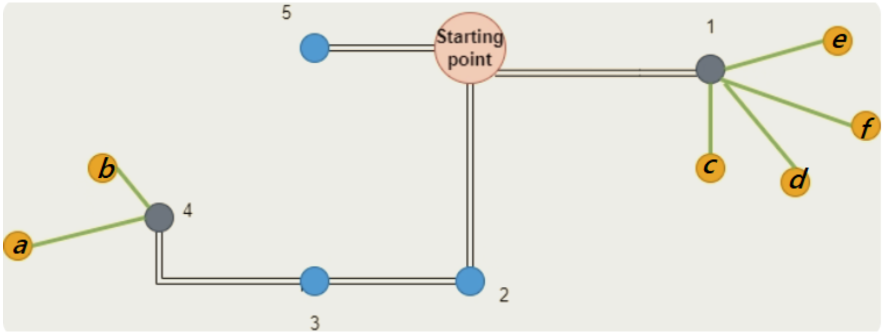


Figure 4.6: Example of a network - after.

The fitness function requires the graph, the individual to be evaluated, the precalculated costs for the drop cable and the different input weights (business rules) provided by the user. There are 3 main materials that greatly impact the cost of solution returned:

- Drop cable cost C_{Drop};
- Distribution cable cost C_{Dist};
- PDO costs C_{Pdo} which should depend on the maximum ports available.

The first step consists in computing the amount of drop cable needed, N_{Drop}, for the current solution (this can done be by accessing the cache containing the drop cable distance costs computed in the initial phase of the system). A simple iteration through the second representation (integer one) is sufficient to perform this step. It is important to note that a factor 10 is applied to the financial cost of the drop cable in case of the buildings (MDUs), to guide the GA to place the PDOs near the buildings (if there is any on the map).

The second step is focused on the distribution cable needed, N_{Dist}, for the solution. This is the cable between the starting point and the different PDOs (total of N_{Pdo}) of the network. This step needs more computational power to perform due to the fact that it is required to compute the shortest path between the PDO to the starting point. The Dijkstra's algorithm is applied in this scenario and the results are stored in a dynamic cache so they can be eventually reused by other calls to this fitness function. This storage step is specially important when dealing with larger networks and more dense graphs to speed up the process.

The third step multiplies the input weights to the resources needed (cables and PDOs) with the following formula for the cost of the materials C_{Mat} :

$$C_{Mat} = C_{Drop} * N_{Drop} + C_{Dist} * N_{Dist} + C_{Pdo} * N_{Pdo} \tag{4.1}$$

Finally, extra penalties can be applied to this cost C_{Mat} in case of broken constraints, in this case the system attempts to cover all the clients if possible. So, every solution that does not have all the clients connected to the network suffers a constant penalty P, the constant P is fixed to the cost of a PDO (chosen by trial and error). This penalty is based on the number of homes missing to be allocated

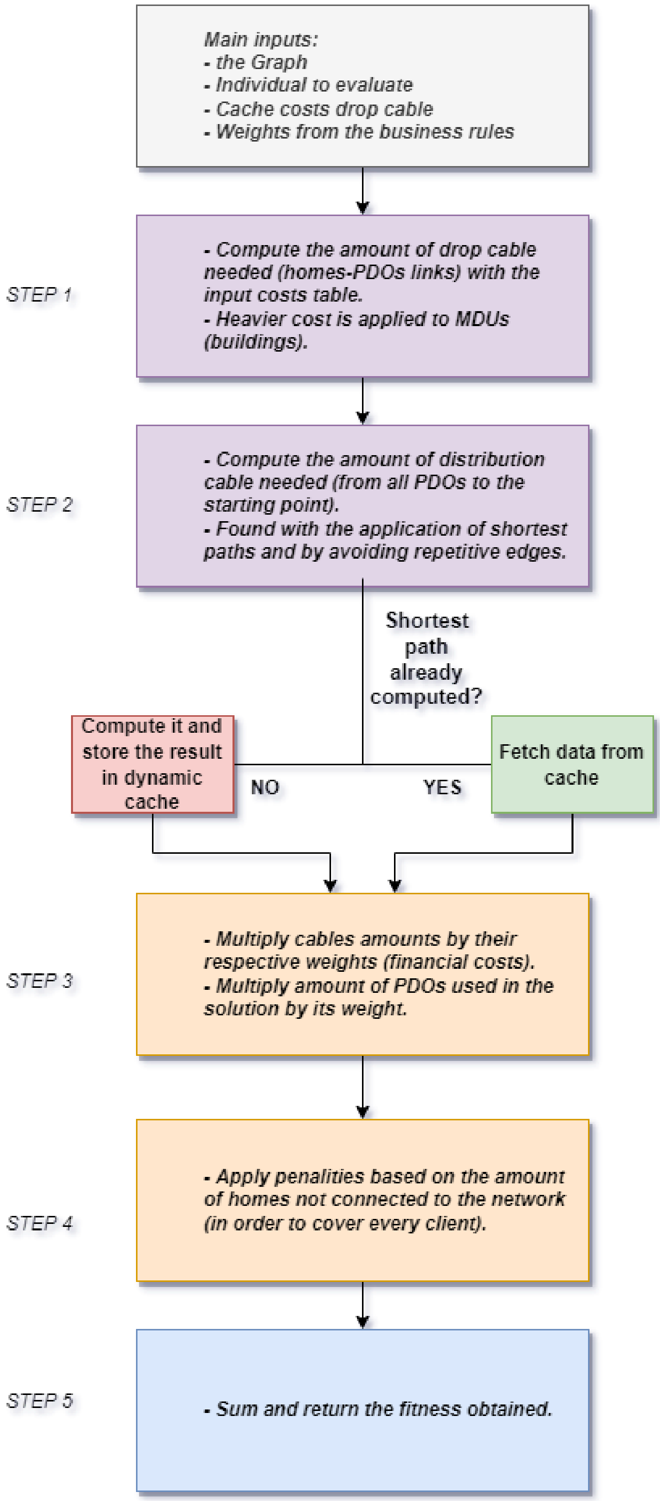

Main inputs:
- the Graph
- Individual to evaluate
- Cache costs drop cable
- Weights from the business rules
STEP 1
- Compute the amount of drop cable needed (homes-PDOs links) with the input costs table.
- Heavier cost is applied to MDUs (buildings).
STEP 2
- Compute the amount of distribution cable needed (from all PDOs to the starting point).
- Found with the application of shortest paths and by avoiding repetitive edges.
Shortest path already computed?
Compute it and store the result in dynamic cache
NO
YES
Fetch data from cache
STEP 3
- Multiply cables amounts by their respective weights (financial costs).
- Multiply amount of PDOs used in the solution by its weight.
STEP 4
- Apply penalities based on the amount of homes not connected to the network (in order to cover every client).
STEP 5
- Sum and return the fitness obtained.

Figure 4.7: Fitness function overview.

$H_{Missing}$, of course if all houses are covered this value will be zero and no penalty is added (minimization). The final result returned is the following:

$$Fitness = C_{Mat} + H_{Missing} * P \tag{4.2}$$

4.4 Local Search

Concerning the sub-problem related to the capacitated problem of the PDOs mentioned earlier, there are two options for the step of connecting each home to a PDO:

Option 1: Straightforward method

Simple iteration through the homes connecting them to the next available PDO. Next available PDO signifies that the equipment node is within the limit drop cable range allowed and it also has ports left to serve the demand of the home currently inspected. This is a straightforward methodology relying on the closest point heuristic to work.

However it can introduce new flaws in certain scenarios, more specifically in zones of the map where we might have a higher demand for optical fiber. The fact that we are linking every home to the closest PDO might not result in the most efficient solution. The Figure 4.8 showcases this problem.

Option 2: Local Search method

The second option available in the system to deal with this problem is slightly more complex, since it requires a Local Search step, through the application of an Hill Climbing optimization technique. As mentioned in Chapter 3, this technique relies on iterating through a solution to attempt to improve it by performing small local changes. If those changes are better we keep the modified solution. The pseudo-code of this technique applied to the problem at hand is described in Algorithm 7.

For our specific case, the algorithm starts by performing the straightforward approach and if we end up with homes not allocated to their closest PDO, we store them for further inspection. A computation of the possible set of PDO nodes for each home allocated to their second (or more) closest PDO is required and cache can be used for this to avoid repetitive computations. Then, the swap procedure can start, by swaping drop cables with clients homes from that set of PDOs. If the cost is reduced, we keep the modification.

It is important to note that every swap made has to respect the constraints in terms of limit drop cable range and ports for the PDOs (for both the current home inspected and the other client home swapped).

Another point worth to mention is that we do not enter in this Local Search every time, which would be computationally costly. The algorithm only considers the Local Search step if there are homes not allocated to their closest PDO which in theory only happens in more dense zones of the graph.

The Figure 4.9 shows the result of the Local Search application for the same scenario as the Figure 4.8. More experiments and comparisons of the multiple variants of the computational system proposed are analyzed in the next chapter.

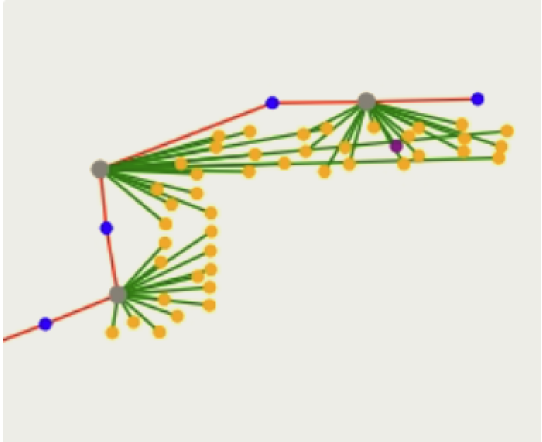


Figure 4.8: Example (portion of larger network) of a drop cable allocation with the straightforward method.

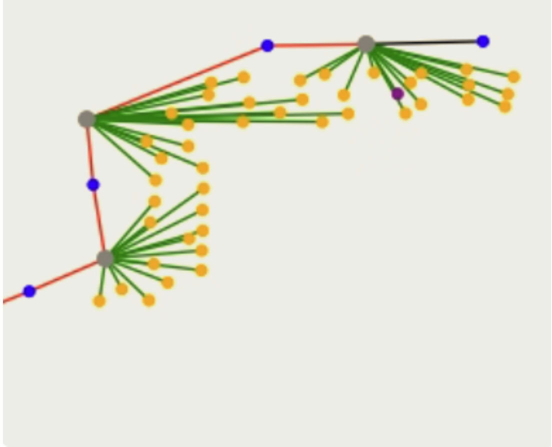


Figure 4.9: Example (portion of larger network) of a drop cable allocation optimized with the Local Search method.

```
Data: List of Homes nodes homes, individual indiv
Result: Updated indiv (second representation concerning the drop cables)
-$indiv_{bin}$ ← Binary representation of the individual (PDO nodes);
-$indiv_{int}$ ← Integer representation of the individual (drop links);
-homesNotInClosest ← [] # To store homes not allocated to their closest
  PDO. ;
foreach h in homes do
    -tmpDist ← Fetch the distances from cache of h to every PDO active in
      $indiv_{bin}$ ;
    - Sort tmpDist in ascending order ;
    for d in tmpDist do
        $pdo_d$ ← PDO node corresponding to the distance d;
        if ($pdo_d$ has ports available) and (d <= limit drop cable range) then
            - Allocate h to $pdo_d$;
            if d is not the first element of tmpDist then
                - Append h to homesNotInClosest ;
            end
            - Break Loop;
        end
    end
end
-queue ← homesNotInClosest #Queue to track the homes to inspect;
while queue is not empty do
    -currentHouse ← pop from queue ;
    -setPossiblePDO ← Find the possible set (with cache help) of PDOS
      for the currentHouse that respects the constraints (range, ports);
    for p in setPossiblePDO do
        - Swap the drop cable links from currentHouse with other clients
          connected to p as long as they also respect the constraints;
        - If a swap reduces the cost in terms of distance, keep the
          modification in the second representation $indiv_{int}$ ;
        - Append the other client (that was modified) to the queue so he
          can also be inspected ;
    end
end
```

Algorithm 7: Allocation of the clients homes to the PDOs - Version with Local Search.

Chapter 5

Experiments and Results

This chapter details and discusses the experimental scenario used to evaluate the proposed approach to solve the optimization problem of Automatic Design of Networks. The first section defines the setup used for the experiments. The second section concerns the validation of the Genetic Algorithm parameters, as it is important to find a good combination of parameters to increase the overall performance of the system. The third section presents and discusses the design results obtained automatically for the different data sets where an handmade solution exists (a comparison is made). The fourth section proposes some extra visual design results for a few maps where there is no handmade solution. Finally, the last section summarizes the experiments.

5.1 Experimental Setup

This section defines the experimental setup to execute the multiple tests, i.e., it mentions the multiple maps available for testing, the different business rules, the scenarios considered and the data collected for each one of the maps.

5.1.1 Maps Details

As mentioned in the Chapter 4, our collaborator provided multiple maps (data sets in the JSON format) to test the system. The network information contained in those data sets is from New York and New Jersey. Table 5.1 details the available maps for which we have a corresponding handmade solution.

It is worth to mention that in this table, the number of location nodes (column 2) might be sometimes superior to the sum of the number of valid network equipment nodes (column 4) and the total clients nodes (column 5). This is due to the fact that for the equipment nodes, some of them got filtered, since they were not part of the main connected component that contains the starting point, i.e., they were isolated nodes.

The objective of the experiments is to propose a solution for each dataset and to

Data set (map)	Number of locations (nodes)	Number of routes (edges)	Number of valid network equipment nodes	Total clients: Houses(SDUs) & Buildings (MDUs)	Sum of the distance of all routes
Map 1	188	81	80	103 & 5	2.11 km
Map 2	197	85	82	110 & 5	2.72 km
Map 3	222	166	133	76 & 11	4.76 km
Map 4	289	192	189	83 & 12	6.11 km

Table 5.1: Details of the different maps initially (for which we have a corresponding handmade solution).

compare it to a handmade solution (also provided by our collaborator). The GA will be used to generate a solution close to the handmade and also a second one with the purpose to generate a different design.

5.1.2 Scenarios and Business Rules

The business rules comprise of the materials costs and the constraints related to the problem. They have a direct impact on the result returned by the algorithm. To limit the amount of experiments (due to the number of possible configurations), two main scenarios have been selected to design the network:

- Scenario 1: obtain a configuration with less PDOs and less distribution cable, but more drop cable.
- Scenario 2: obtain a configuration with more PDOs and more distribution cable, but less drop cable.

Material costs

Each of those scenarios is dependent on a fixed set of weights that represent the material costs (input provided by the user). Table 5.2 displays the weights for each scenario. The set 1 corresponds to the scenario 1 and the set 2 corresponds to the scenario 2. Each of those sets is arbitrary (could vary slightly) even though they are inspired from real material costs.

It is worth to note that these are the weights that are plugged in the fitness function (described in the Chapter 4.3) of the Genetic Algorithm.

Constraints parameters

As for the constraints, they refer to everything that might limit the search space to find a solution. Each of the scenarios is also limited by the set of constraints defined in the Table 5.3. There are two types of constraints to consider:

Material	**Set 1**	**Set 2**
Cost PDO	300$ unit	400$ unit
Cost drop cable per meter	2$	9$
Cost distribution cable per meter	5$	5$

Table 5.2: Sets of experimental weights for the materials.

Constraint parameter	**Value**
Limit ports per PDO	12
Margin ports open for each PDO (in percentage)	10
Limit drop cable (range in meters)	85 (range: 50-100)
Max possible range of the network (km)	20

Table 5.3: Experimental constraints.

- The ones concerning the PDOs: maximum number of ports per PDOs and also the margin to leave for each PDO for futures expansion of the network. For example if a PDO with a limit of 12 ports is chosen with a margin of 10% to leave open, in practice only 11 ports are operational.
- And the ones that concern the cables: restriction on the maximum distance allowed for the cables, like the maximum overall distance of the network (from the OLT to a client) and the limits imposed on the drop cables. For the drop cables, the designs are proposed for a 85 meters limit, but it can vary depending on the data set (range is from 50 to 100 meters maximum but exceptions for more could happen in case a home is way too far from a network equipment node).

In the system proposed, all those inputs (costs and constraints) are adjustable and are chosen by the user.

5.1.3 Data Collected

With those baselines (scenarios and constraints) the data collected for each experiment (for each map) consists of the following:

- Costs and materials used for the best solution found. This includes the types of cables, number of PDOs and splitters in the JSO;
- Client demand served and penetration rate (i.e., the clients coverage, if it is a rate of 100%, it means that all clients are connected to the network);

- Amount of ports available in the PDOs for future expansion of the network.
- Plot of the fitness over the generations for the multiple runs (30).
- Optical budget check for each PDO.
- Execution time required to find a solution.

To avoid redundancy (similar plots that show same patterns among the two sets of weights), the fitness plot and the optical budget are displayed only for the set of weights 1, which theoretically corresponds to the scenario where we need to install less PDOs. This corresponds to a design scenario of preference from our collaborator where the aim is to reduce the amount of PDOs installed.

On a side note, as we are dealing with the optimization of the last part of the network (distribution network and clients connections), we can fix some types of equipment that are computed in post-optimization process. More specifically the types of cables from the distribution network (red edges in the plots) and the splitters types. The cables can take the following values $\{8, 16, 32\}$ fibers type. As for the splitters we can fix the ratio to $1 : 64$. Once again this is all input parameters that can be adjusted by the user, the point here in the post-process is to find the quantity of each extra equipment is required to serve the demand of the network.

Also, in terms of hardware configuration used for the experiments, we had the following specs:

- Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz.
- GeForce GTX 1080 Ti.
- RAM: 16Go.

Now that the experimental setup is fixed, before proceeding with the design tests, it is important to validate the optimization algorithm and the GA parameters, this task is discussed in more detail in the next section.

5.2 Validation of the Genetic Algorithm

The amount of possible configurations for the Genetic Algorithm is high due to the multiple rates and inner functions involved. As such, the focus will be on testing a few combinations. More specifically the mutation operator and the percentage rates.

5.2.1 Parameters for the Genetic Algorithm

Table 5.4 presents the configuration used, we can notice that the parameters related to the mutation can vary, the best configuration for this variation operator is

Parameter	Value
Population size	100
Number of generations	100
Mutation	- Bitflip - Swap -Double bitflip
Mutation rate *(in percentage)*	- 5 -10 - x/len
Crossover	Uniform (with 50% per gene)
Crossover rate *(in percentage)*	85
Tournament selection	5
Survivors selection *(elitism percentage)*	10

Table 5.4: Genetic Algorithm parameters.

tested next and used for the experiments. In the case of the mutation rate x/len, the x can be 1 or 2 or 3, that means that we are aiming to vary only x gene(s) per individual of size len.

5.2.2 Validation

The validation process to find the best configuration consists of the following steps:

- Step 1: check the box plot that compares the different mutation functions, applied to different rates. Initially, this is done for the version of the optimization algorithm without Local Search.

- Step 2: pick the best mutation parameters from step 1 and check the version with Local Search and compare.

- Step 3: keep the best parameters as the final configuration and run the tests.

Those steps are only applied to a single data set (the "Map 2") in the case of the scenario 1, due to the fact that the Local Search is only useful for dense zones, where the PDOs ports are more likely of being full and where the drop cables coming from each client might intersect. Also, to obtain statistically valid results, 30 runs were performed per configuration and the solutions with the most performance (in terms of fitness) were retained for the plots.

Step 1

Figure 5.1 shows the box plot from the Step 1 of the validation process, based on this box plot, we can already discard a few configurations of parameters:

- The mutation Swap and Double-Bitflip seems to under perform in comparison to the Bitflip every time. This might be due to the sensibility of the problem at hand, most likely it is required to only perform small changes, which is not the case with the Swap and Double-Bitflip, they modify 2 genes instead of 1.
- This seems to be reflected with the percentage rates, a higher percentage rate seems to impact too much the solutions whereas in the smaller percentages 1.2% (corresponds to $1/len$) and the 5% seems to perform better.

A Kruskal-Wallis test was performed to ensure that the differences in the experiments are statistically significant: the result of this test returned a p-value less than 0.05 we can conclude that one or more samples are statistically different. It is also important to perform a multiple pair-wise test (Wilcoxon) to verify the differences amongst pairs. Table 5.5 displays the pairs where the p-value is superior to 0.05. In red we have the Bitflip functions highlighted. We can notice that for all pairs including the Bitflip function, where only the mutation rate changes, there is no statistically significant differences, except when the 10% rate is involved.

Also, graphically, the Bitflip function is the best one, which can be verified with the statistical test: due to their significant differences with the Bitflip function (which provides graphically better results, i.e., lower fitness), we can discard the Swap and Double-Bitflip functions. Finally, as the mutation rate of 10 is the only one that is significant different from the other Bit-flip configurations, we can discard this higher mutation rate.

So, for the rest of the experiments, the Bitflip function and lower mutation rates are kept.

Step 2

Based on the results of the step 1 in terms of configuration for the mutation operator, we can now compare the performance of the algorithm with Local Search. The box plot in Figure 5.2 displays the results for the previously chosen mutation parameters applied to the version of the GA with Local Search.

We can see that the configurations that include the Local Search step in the Genetic Algorithm perform better than the version without. It might seem that in terms of fitness the gains are very small (after all it is a local optimization for some drop cables links), but visually the improvement is clear. The Figure 5.3 and Figure 5.4 are the automatic design solutions returned by the system for the version without and with Local Search respectively. On the second design, the drop cables (green links) are more optimized in comparison to the first design. The difference in fitness of the individuals of those design solutions (with and without Local search) is equal to 275, which is the amount saved by using the approach with Local Search.

Additionally, we can see graphically (in the box plot) that the configuration with the mutation rate $1/len$ seems to slightly under perform in comparison to the others, so we can discard this mutation rate and pick one of the rates left (they are not statistically different).

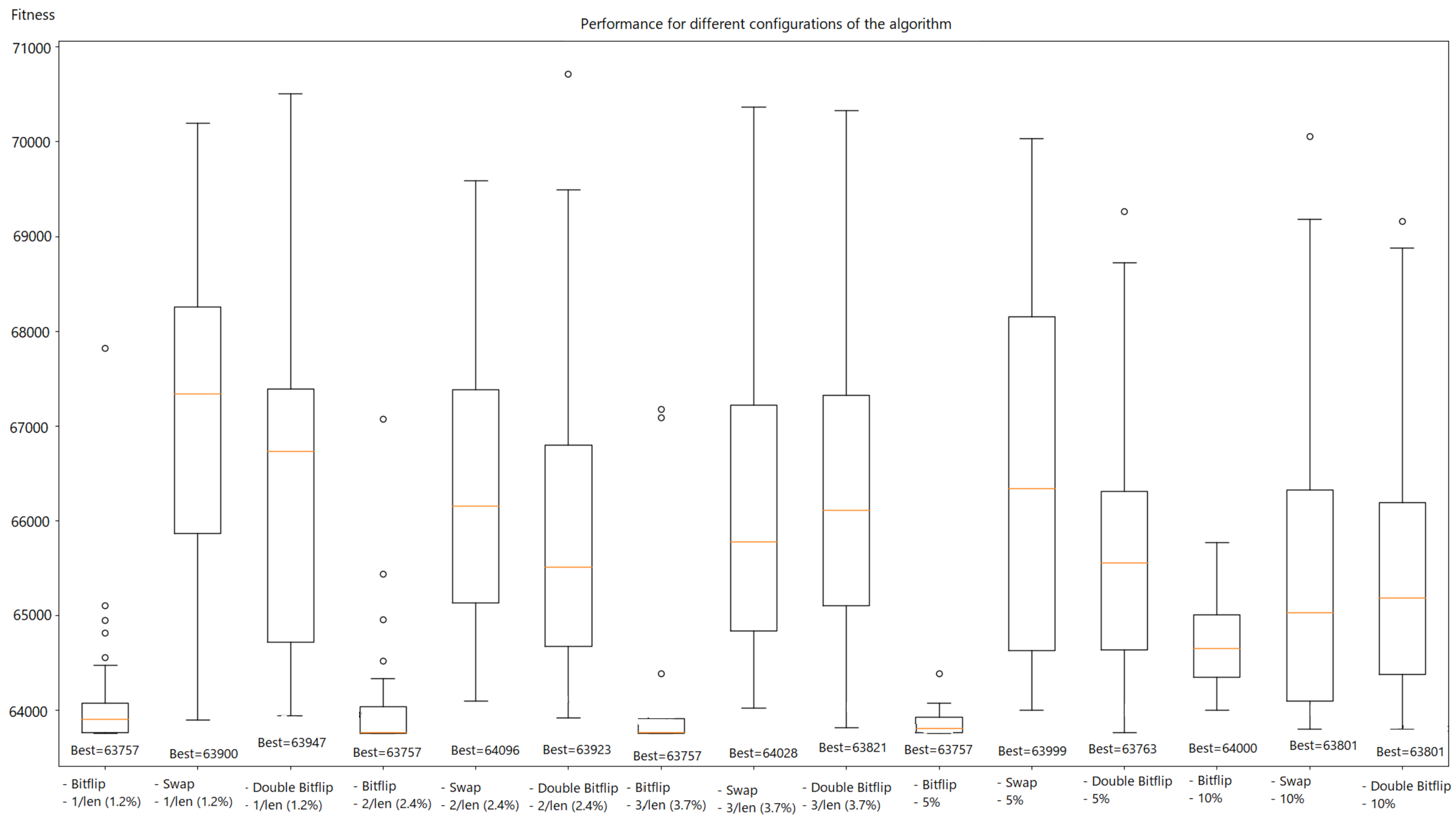


Figure 5.1: Box plot to compare the different options for the mutation parameters (step 1).

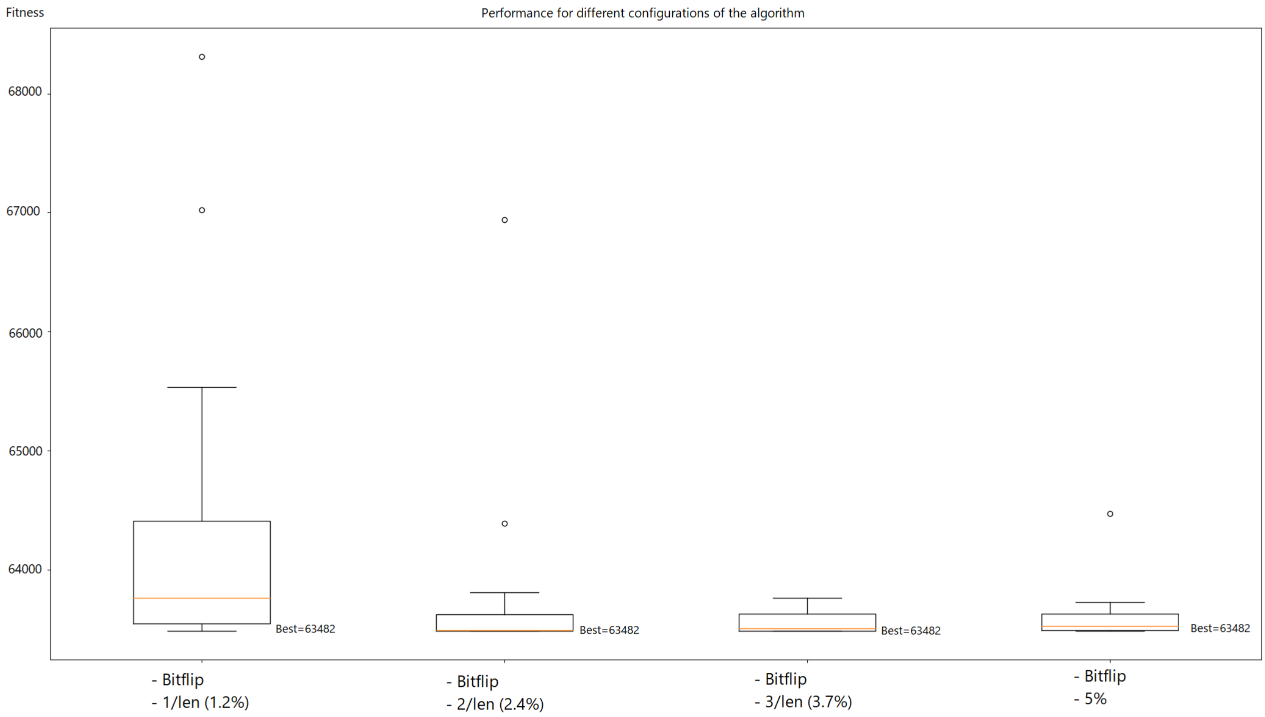


Figure 5.2: Box plot to compare the different options for the mutation parameters and for the Local Search option (step 2).

Step 3

Finally, we can lock the choice of parameters based on the previous tests:

- Mutation function: Bitflip.
- Mutation rate: $2/len$ (again, with the len being the size of the binary representation of an individual, which vary from map to map).
- Local search active.

5.3 Design Results

This section focuses on the design results obtained for all the datasets (maps) described in the Table 5.1. For each one of the maps, two solutions are generated based on the experimental setup defined on the first Section, the solutions are generated with different weights:

- Solution with the first set of weights (scenario 1).
- Solution with the second set of weights (scenario 2).

Configuration 1	Configuration 2	p-value
BF - 0.012	BF - 0.024	0.31
BF - 0.012	BF - 0.037	0.09
BF - 0.012	BF - 0.05	0.07
SW - 0.012	DB - 0.037	0.13
SW - 0.012	SW - 0.05	0.19
DB - 0.012	SW - 0.024	0.99
DB - 0.012	DB - 0.024	0.13
DB - 0.012	SW - 0.037	0.55
DB - 0.012	DB - 0.037	0.95
DB - 0.012	SW - 0.05	0.89
DB - 0.012	DB - 0.05	0.13
DB - 0.012	DB - 0.1	0.06
BF - 0.024	BF - 0.037	0.43
BF - 0.024	BF - 0.05	0.63
SW - 0.024	DB - 0.024	0.13
SW - 0.024	SW - 0.037	0.4
SW - 0.024	DB - 0.037	0.68
SW - 0.024	SW - 0.05	0.97
SW - 0.024	DB - 0.05	0.22
SW - 0.024	SW - 0.1	0.06
SW - 0.024	DB - 0.1	0.07
DB - 0.024	SW - 0.037	0.55
DB - 0.024	DB - 0.037	0.14
DB - 0.024	SW - 0.05	0.17
DB - 0.024	DB - 0.05	0.70
DB - 0.024	SW - 0.1	0.5
DB - 0.024	DB - 0.1	0.42
BF - 0.037	BF - 0.05	0.52
SW - 0.037	DB - 0.037	0.46
SW - 0.037	SW - 0.05	0.53
SW - 0.037	DB - 0.05	0.46
SW - 0.037	SW - 0.1	0.08
SW - 0.037	DB - 0.1	0.21
DB - 0.037	SW - 0.05	0.81
DB - 0.037	DB - 0.1	0.06
SW - 0.05	DB -0.05	0.21
SW - 0.05	SW - 0.1	0.09
SW - 0.05	DB - 0.1	0.06
DB - 0.05	SW - 0.1	0.76
DB - 0.05	DB - 0.1	0.64
SW - 0.1	DB - 0.1	0.89

Table 5.5: Multiple pair-wise results (only the ones with p-value >0.05, i.e., no significant differences). In red we have the ones that we are interested in: the ones that use Bitflip function for the mutation and a lower mutation rate. Note about the functions acronyms: BF stands for Bitflip; SW stands for Swap; DB stands for DoubleBit-flip.

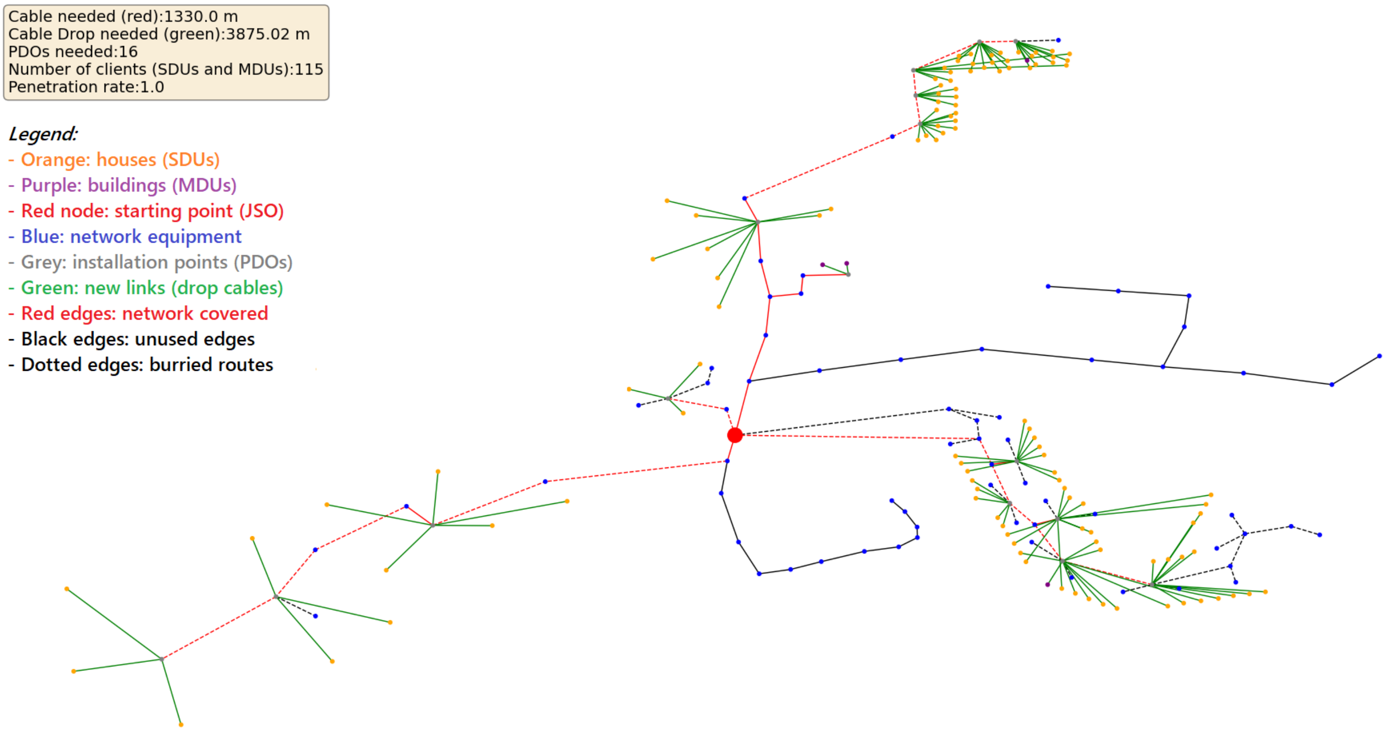

Cable needed (red):1330.0 m
Cable Drop needed (green):3875.02 m
PDOs needed:16
Number of clients (SDUs and MDUs):115
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes

Figure 5.3: Design solution for the dataset named "Map 2" - example without Local Search (set of weights 1).

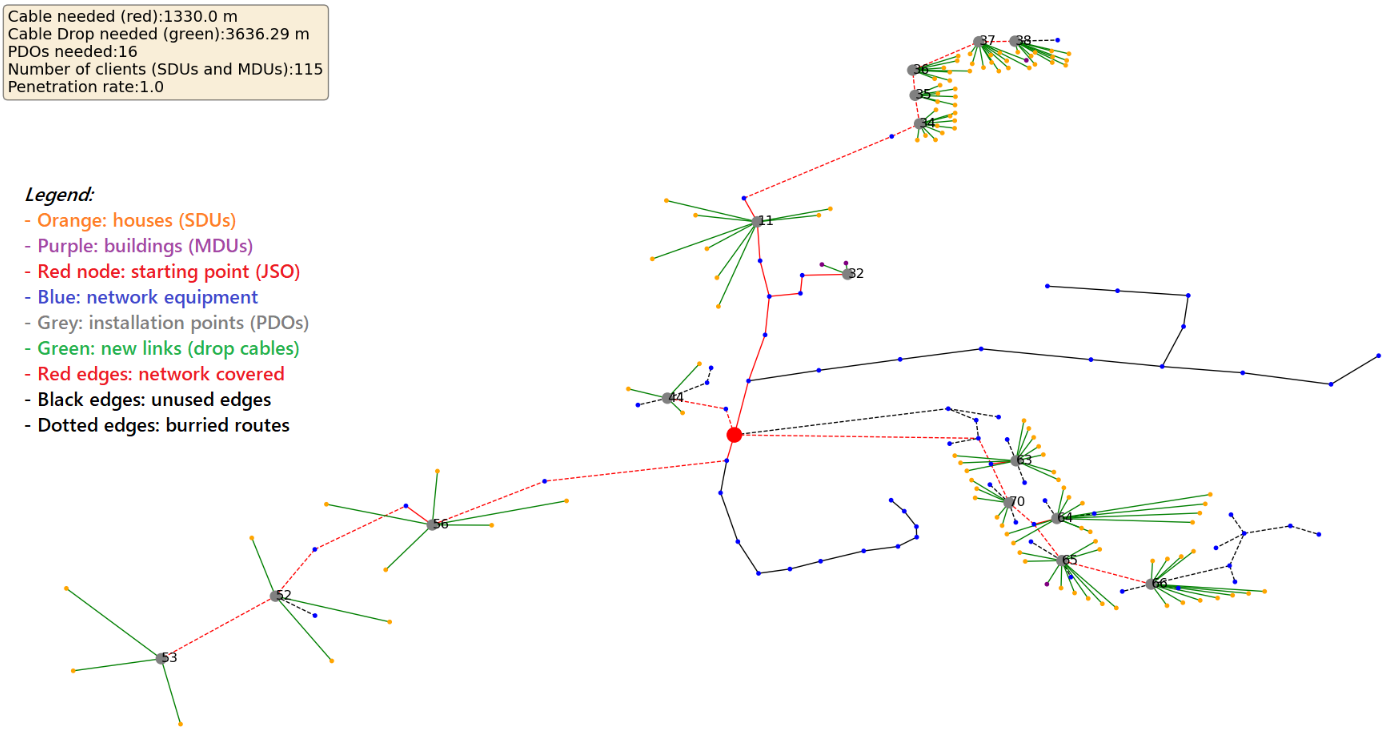

Cable needed (red):1330.0 m
Cable Drop needed (green):3636.29 m
PDOs needed:16
Number of clients (SDUs and MDUs):115
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes
37
38
36
35
34
11
32
44
63
70
64
65
66
56
52
53

Figure 5.4: Design solution for the dataset named "Map 2" - example with Local Search (set of weights 1).

An attempt to find a similar design to the handmade solution provided by our collaborator (whenever available) is also performed and compared in tables containing the metrics for each map, namely the materials required to build the network and the fitness score (cost) associated. For the GA, due to the variability of the solutions obtained, the table presents the results for the best solution, the worst, the median one and the mean from population.

5.3.1 Results for "Map 1"

The first map analyzed is a map from New York called "Map 1". Figure 5.5 corresponds to the automatic solution generated by the GA for the set 1 of weights, and Figure 5.6 corresponds to the set 2.

Fitness performance results

The sets of weights are of great importance and they have a direct impact in the output of the GA through the fitness function. As can be seen in this first map, we have two different designs. This can also be seen in the Tables 5.6 and 5.7, which summarize the data collected during this first experiment. When comparing the best GA solutions for each set of weights, we can see that the set 1 (Table 5.6) gives priority in reducing the number of PDOs (16) but allowing larger distances for the drop cables (as long as they respect the distance limits). Whereas the set 2 (Table 5.7) proposes a solution where the PDOs are much more present (27). The installation of multiple PDOs allows to reduce the usage of longer drop cables, but it also forces the usage of more distribution cable, meaning that more edges from the network are covered.

We can also notice that for the worst solutions (for both sets of weights), the fitness score increases considerably, sometimes by a factor 10 in comparison to the best solution (e.g., for the set 1, the best has a score of 23908 and the worst solution has a score of 204545). This high score is due to some penalties in the fitness function of the GA, more specifically in the case where some homes are not allocated to the network. A high constant penalty based on the homes left to be allocated is applied to guide the evolutionary algorithm towards full coverage of the homes (which corresponds to a penetration rate of 100%).

The second column of the table (Handmade) contains the statistics for the handmade solution. When comparing it to the solutions generated by the proposed system, for both scenarios in terms of fitness score, the GA outperforms the handmade design when considering the best solution (and the median). This is due to the fact that there is way too many PDOs installed in the handmade solution (visual map in the Figure 5.7). In every network equipment (like a pole or pedestal), there is an installation point (PDO), which is not optimal. With this in consideration, the closest scenario to the handmade design corresponds to the second scenario (set of weights 2 - Figure 5.6). This design reduces the number of PDOs from 62 to 27. However it also requires more drop cable overall, but the fitness (cost) is reduced considerably, with a cost difference of 11045, which means we can save this amount when using the design proposed by the GA instead of the handmade. This difference in fitness score also comes from reduced distribution cable since the GA solution does not require the full coverage of the edges, where some edges are unused (black edges in the plots).

The Figure 5.8 refers to the fitness function over the generations for this map. It shows the fitness for the best individual overall and the average of the bests (for each run of the 30 runs). We can notice that the fitness score decreases over the generations, which is expected, since the objective is to minimize the cost of the network. We can also notice that after a certain point (70 generations), the gains in terms of fitness are considerably reduced and it starts to stabilize (which could mean that it reach optimal or near optimal point).

Other results

One of the requirements for the PDOs was the amount of ports available and a margin of ports to leave open, this margin is set to 10 percent in the current experiments. Table 5.8 and 5.9 shows the results for the PDO nodes where there is a building link. Those buildings (MDUs) can have a higher demand (i.e., more ports required) than a simple home (SDU) which is usually 1. We can see that even when multiple homes and buildings are connected to the same PDO, there is always a margin left of ports (it never reaches the maximum limit), this allows for future expansion of the network in case that new homes are built in those areas for example or another existing home changes its demand. Also, as expected, in the scenario 2 there are much more PDOs which leaves more ports open for each PDO in comparison to the scenario 1, this might not be very optimal due to the fact that each PDO is not fully optimized/used.

As for the types of splitters and cables, the result can be found in the Table 5.10. The demand for this network is of 121 fibers, to serve that we need at least 2 splitters of type 1 : 64. As for the cables, there are different types needed based on the branches that come out of the starting point node. For example a branch that requires 47 fibers, it requires a distribution cable 32FO and another of 16FO (which results in 48 fibers, with 1 additional that is unused, but still can be useful for future expansion).

The Figure 5.9 shows the optical budget for this map, it is computed for each PDO node, we can see that they never reach the limit (as mentioned in the Chapter 2), so the network is valid. The small variations on the power between each PDO node is mainly due to the distance difference.

Finally, in terms of execution time, it takes only 30.5 seconds to do a single run to design this first network. This is pretty advantageous in comparison to the handmade solution, which in theory could take several hours (around 3h) for a network of this size. It is important to note that this execution time is dependent on the hardware setup and on the experimental setup (more generations or bigger population size can have a great impact on the time required to design the network).

Metrics \ Approach	**Handmade**	**GA ǀ best (set 1)**	**GA ǀ worst (set 1)**	**GA ǀ median (set 1)**	**GA ǀ mean SD (set 1)**
Number of PDOs	62	17	22	20	18.8 SD: 0.92
Drop cable used (km)	2.75	4.18	3.76	4.07	4.15 SD: 0.12
Distribution cable used (km)	2.08	1.74	1.77	1.74	1.7 SD: 0.03
Fitness score	34505	23908	204545	24561	38893 SD: 5832
Difference fitness (GA - Handmade)	——-	-10597	170040	-9944	-4388

Table 5.6: Results for the dataset "Map 1" with set of weights 1. Results obtained with 30 runs.

Metrics \ Approach	**Handmade**	**GA ǀ best (set 2)**	**GA ǀ worst (set 2)**	**GA ǀ median (set 2)**	**GA ǀ mean & SD(set 2)**
Number of PDOs	62	27	26	29	28.6 SD: 0.78
Drop cable used (km)	2.75	3.14	3.29	3.13	3.16 SD: 0.04
Distribution cable used (km)	2.08	1.94	1.94	1.97	1.95 SD: 0.02
Fitness score	66846	55801	97442	56636	58903 SD: 1852
Difference fitness (GA - Handmade)	——-	-11045	30596	10210	-7943

Table 5.7: Results for the dataset "Map 1" with set of weights 2. Results obtained with 30 runs.

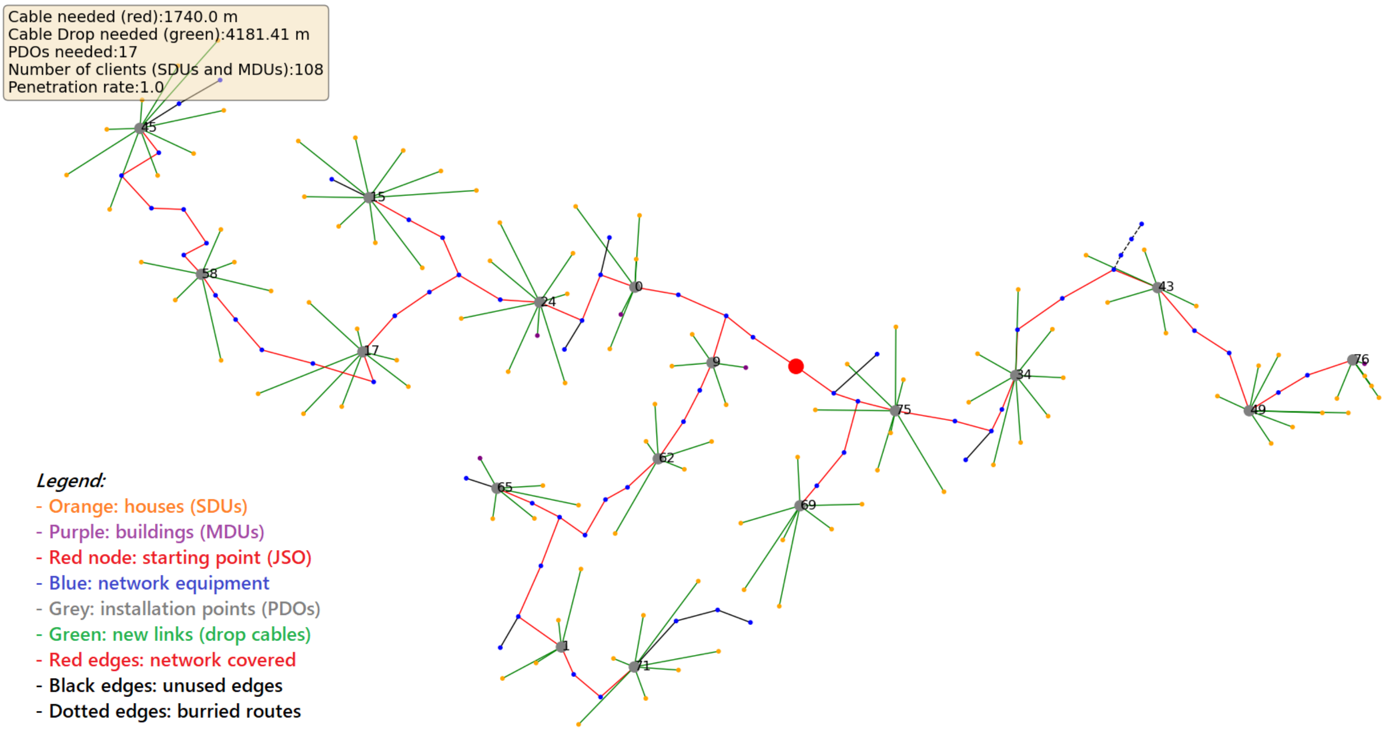

Cable needed (red):1740.0 m
Cable Drop needed (green):4181.41 m
PDOs needed:17
Number of clients (SDUs and MDUs):108
Penetration rate:1.0
45
15
58
17
24
0
9
62
65
69
1
71
75
34
43
49
76
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes

Figure 5.5: Design generated by the GA for the dataset "Map 1 " - set of weights 1. The visualized result corresponds to the best solution.

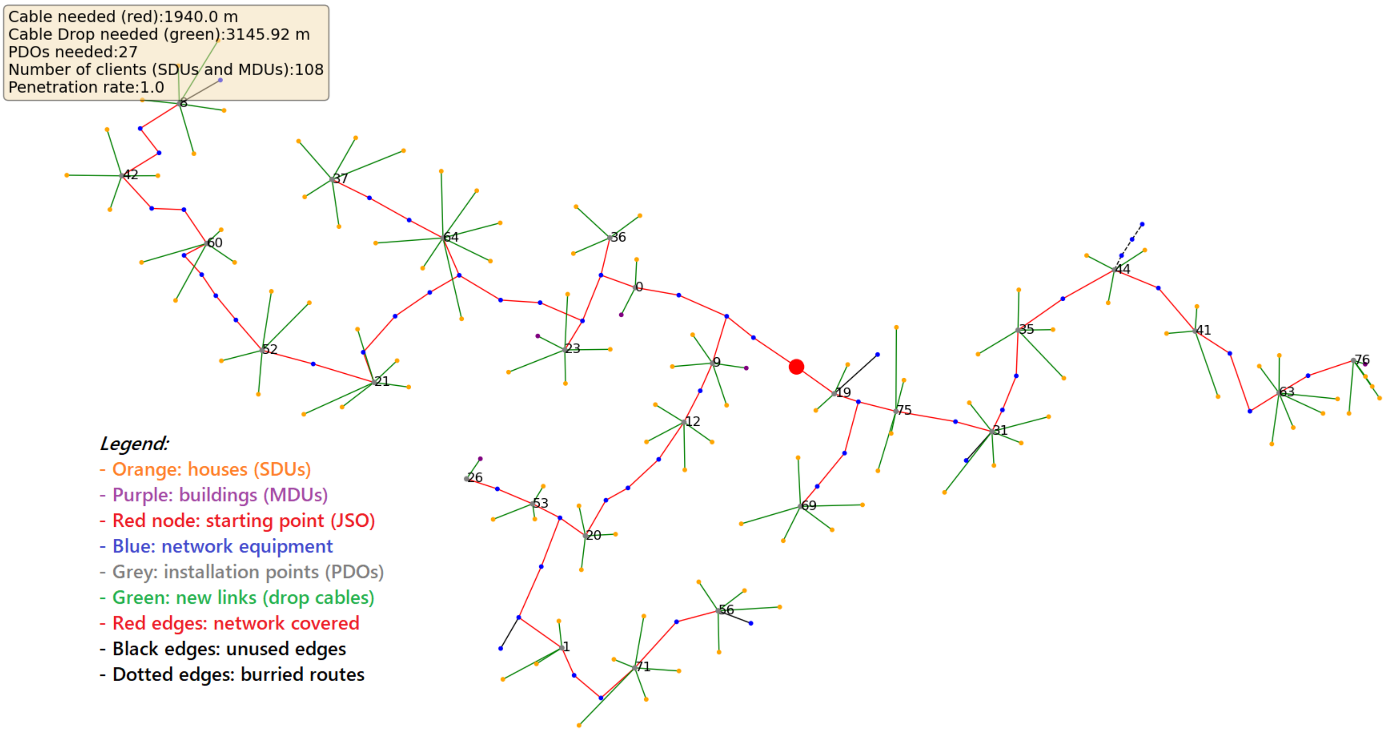

Cable needed (red):1940.0 m
Cable Drop needed (green):3145.92 m
PDOs needed:27
Number of clients (SDUs and MDUs):108
Penetration rate:1.0
6
42
37
64
36
60
0
23
52
21
9
19
75
12
35
44
41
63
76
31
26
53
20
69
56
1
71
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes

Figure 5.6: Design generated by the GA for the dataset "Map 1 " - set of weights 2. The visualized result corresponds to the best solution.

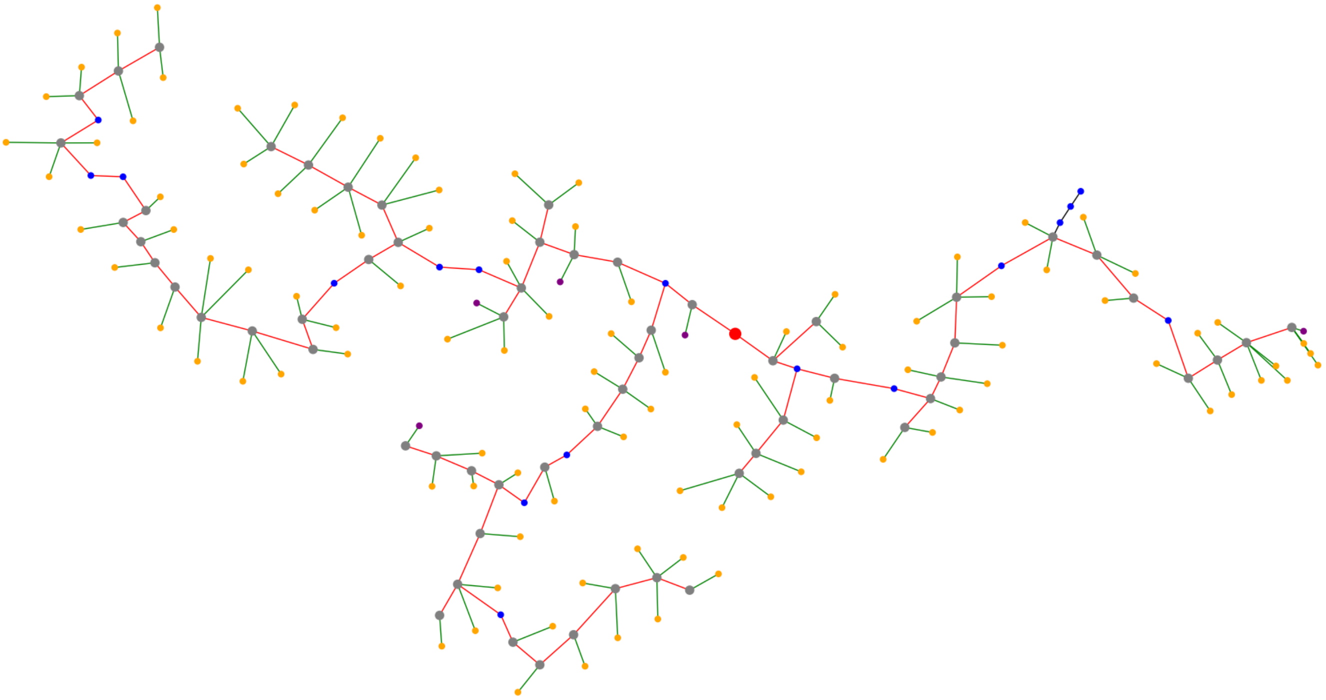


Figure 5.7: Handmade solution for the dataset "Map 1".

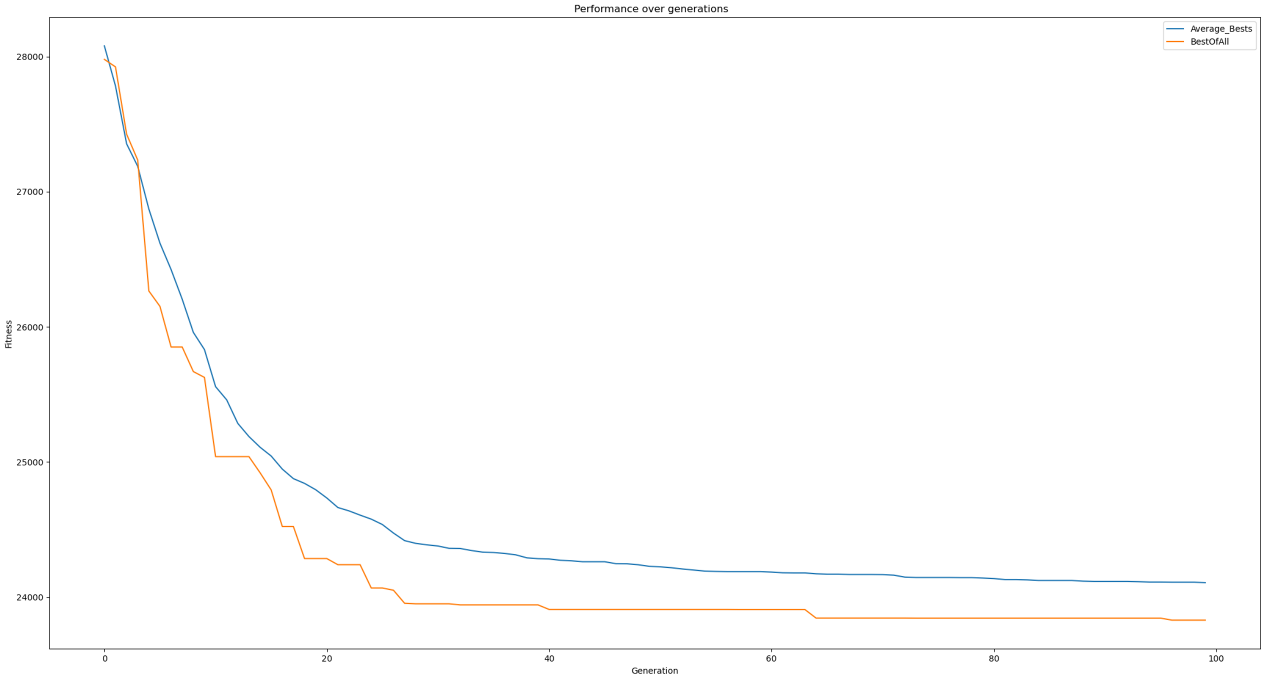


Figure 5.8: Fitness over the generations for the dataset "Map 1" - set of weights 1.

PDO node id (that contains a MDU link)	**Ports used (out of 12 max.)**	**Amount of unique clients served (distinct SDUs/MDUs nodes)**
0	6	5
9	9	4
24	9	8
65	7	5
76	11	5

Table 5.8: Amount of ports used in the PDOs that contain a connection to a building (MDU). Dataset "Map 1 " with set 1.

PDO node id (that contains a MDU link)	**Ports used (out of 12)**	**Amount of unique clients served (distinct SDUs/MDUs nodes)**
0	3	2
9	5	4
23	6	5
26	3	1
76	11	5

Table 5.9: Amount of ports used in the PDOs that contains a connection to a building (MDU). Dataset "Map 1" with set 2.

Extra equipment needed	**Quantity**
Splitters Type 1x64 (in the JSO/starting point)	2
Cables 16FO	2
Cables 32FO	3

Table 5.10: Extra equipment needed for the dataset "Map 1" to serve a fiber demand of 121 from 108 distinct homes locations.

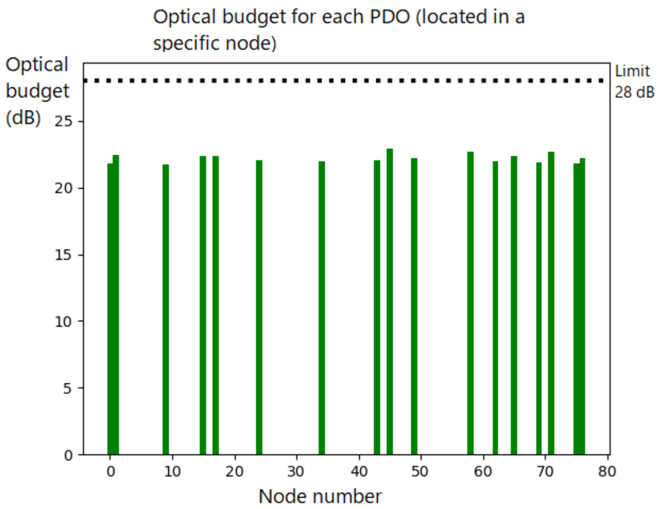


Figure 5.9: Optical power budget for the dataset "Map 1" - set of weights 1. Each bar corresponds to a PDO located in a network equipment node (which has a number id).

5.3.2 Results for "Map 2"

This second dataset from New York is the same as the one used previously used to illustrate a few concepts and to validate the GA in the previous section (e.g., Figure 5.4 which uses the set 1 of weights). The main difference in comparison to the first map relates to a big portion of the network that is not really relevant, where there are no homes around.

Figure 5.10 corresponds to the scenario 2 (i.e., more PDOs but more drop cable overall) and the Figure 5.11 shows the handmade solution. In this map we notice that the GA seems to be robust enough to avoid those less relevant routes (those without homes nearby) without any pre-processing (e.g., like computing the Minimal Spanning Tree or the Steiner Tree prior to executing the GA phase). In both designs the GA completely ignores those less relevant zones (black edges on the plots).

Fitness performance results

The Tables 5.11 and 5.12 summarize the statistics collected for this map. Once again the evolutionary system proposed seems to be able to generate new and better solutions in comparison to the handmade solution for the best and median, using less materials to build the network. It can be seen in the tables by checking the difference of fitness between the GA and the handmade solution. For example, for the best solution with the set 1 we save 14361 dollars and we save 11042 dollars for the set 2. The median solution is useful to analyze the solutions that are in the middle of the population without taking too much into account outliers (due to the fact that some worst solutions can have a very high score due to the penalties of the fitness function). We can see that the median solution is actually very close to the best one for both sets, so actually most solutions of the population are very similar.

Overall, in terms of resources to build this map in comparison to the previous one, it is pretty similar (as the network is close in size). But, in this map, we have a lot of buried edges (dotted edges in the design plots) that can not be avoided which shows in the fitness results (higher than the previous map). Those buried paths have an increased cost factor of 10, this is why it is better to use aerial paths whenever it is possible).

An example of the fitness function score over the fixed 100 generations can be seen in the Figure 5.12 for the scenario 1. Same as previously, when a certain amount of generations is reached, the best individual fitness does not improve anymore, but this time around, a big part of the population seem (average of bests) to converge towards that same solution (fitness score of 63482) which confirms why our median solution is pretty close in performance to the best one (as mentioned in the previously).

Other results

Similarly to the previous map result, the PDOs have a margin of ports to leave open, the Tables 5.13 and 5.14 show the ports open for the PDOs that contain a building linked to it (the rest of the PDOs ports can be seen visually). Also, the Figure 5.13 corresponds to the power budget, every PDO respects the limit.

The extra equipment types required to build this network can be found in Table 5.15. The total fiber demand of this network is 126 fibers, which requires 3 splitters of type 64. This can be a bit of overkill, in practice 2 of type 1:64 and a third smaller one should suffice to satisfy the demand.

Concerning the execution time for this map, a single run takes up to 35 seconds. This is a bit more than the previous map due to the fact that it is slightly bigger and also the Local Search step is more relevant in this map due to its density in certain regions.

5.3.3 Results for "Map 3"

This dataset is from New Jersey and it is a slightly bigger map than the previous ones, with more nodes and edges. In reality this is probably the most interesting map amongst the datasets provided by our collaborator due to the fact that we actually can test the performance of the system when dealing with multiple possible paths. On the datasets mentioned earlier, usually the path choice was very limited to reach a node on a certain branch, i.e., there was only one single path. This current map actually allowed us to notice the shortcomings of the approach with the GA alone (which was used alone until now due to the limitations of the previous datasets, any additional approach/pre-processing was not really relevant).

To present this subsection, a different set of steps is considered (in comparison to

Metrics \ Approach	Handmade	GA I best (set 1)	GA I worst (set 1)	GA I median (set 1)	GA I mean SD (set 1)
Number of PDOs	31 (used out of 42)	16	16	17	17.1 SD: 0.42
Drop cable used (km)	2.80	3.63	2.89	3.58	3.57 SD: 0.05
Distribution cable used (km)	1.84	1.33	1.41	1.35	1.35 SD: 0.03
Fitness score	77843	63482	288 730	64694	78895 SD: 5826
Difference fitness (GA - Handmade)	——-	-14361	210887	-13149	-1052

Table 5.11: Results for the map "Map 2" with set 1. Results obtained with 30 runs.

Metrics \ Approach	Handmade	GA I best (set 2)	GA I worst (set 2)	GA I median (set 2)	GA I mean SD (set 2)
Number of PDOs	31 (used out of 42)	19	20	20	19.5 SD: 0.2
Drop cable used (km)	2.80	3.18	2.92	3.14	3.10 SD: 0.03
Distribution cable used (km)	1.84	1.38	1.35	1.4	1.40 SD: 0.08
Fitness score	105154	94112	290488	94985	110285 SD: 5526
Difference fitness (GA - Handmade)	——-	-11042	185334	-10169	5131

Table 5.12: Results for the map "Map 2" with set 2. Results obtained with 30 runs.

PDO node id (that contains a MDU link)	Ports used (out of 12)	Amount of unique clients served (distinct SDUs/MDUs nodes)
32	11	10
37	11	10
65	11	8

Table 5.13: Amount of ports used in the PDOs that contain a connection to a building (MDU). Dataset "Map 2" set of weights 1.

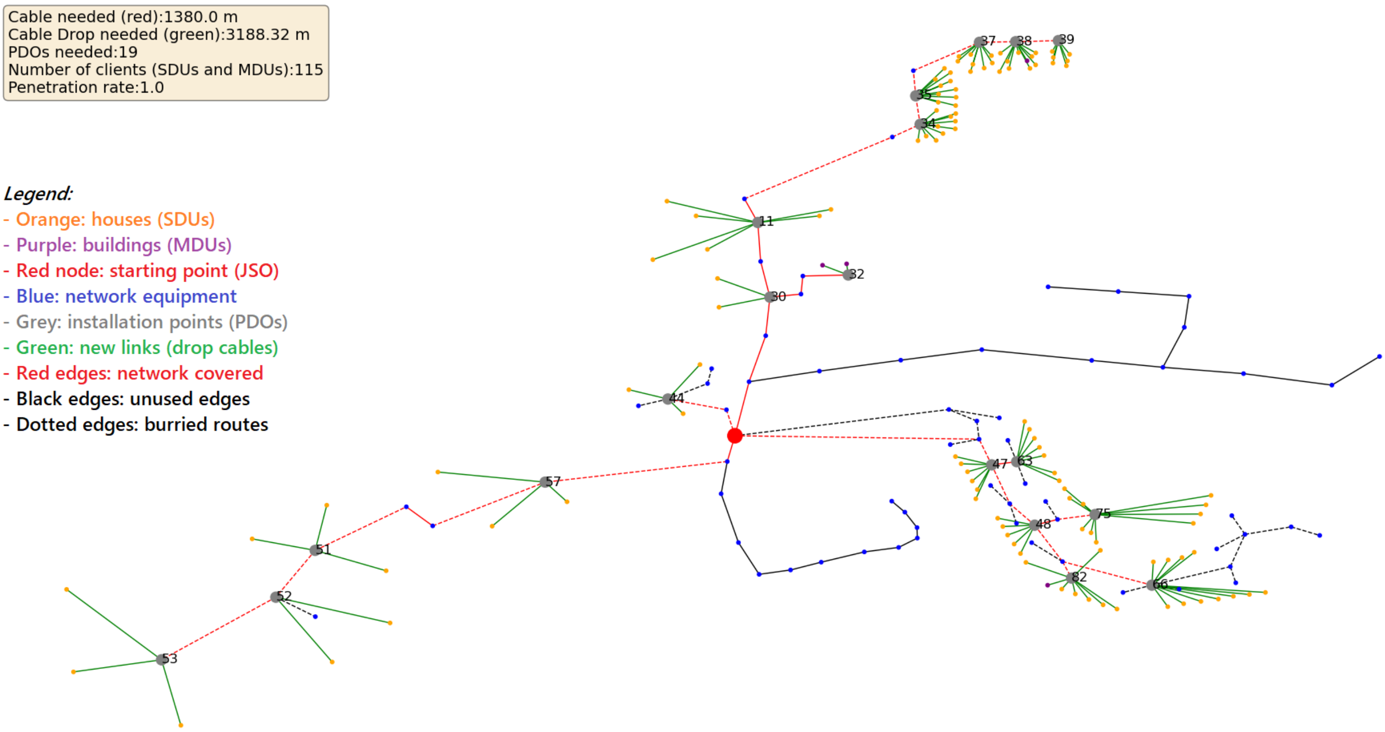

Cable needed (red):1380.0 m
Cable Drop needed (green):3188.32 m
PDOs needed:19
Number of clients (SDUs and MDUs):115
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes
37
38
39
35
34
11
32
30
44
57
47
63
75
48
82
66
51
52
53

Figure 5.10: Design generated by the GA for the dataset "Map 2" - set of weights 2. The visualized result corresponds to the best solution.

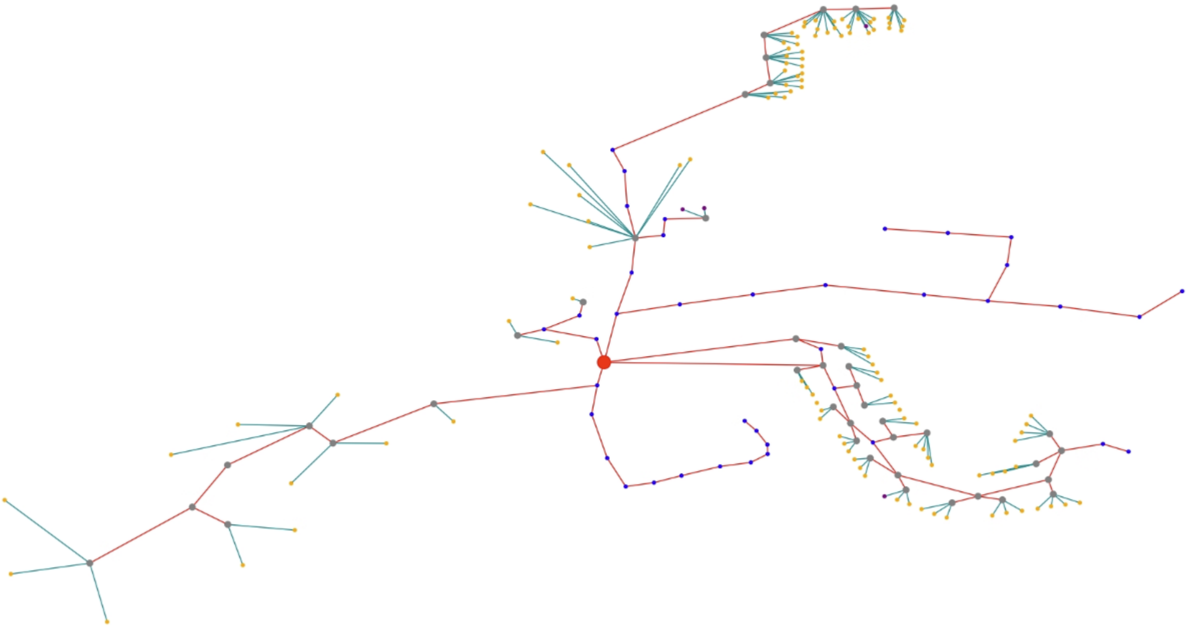


Figure 5.11: Handmade solution for the dataset "Map 2".

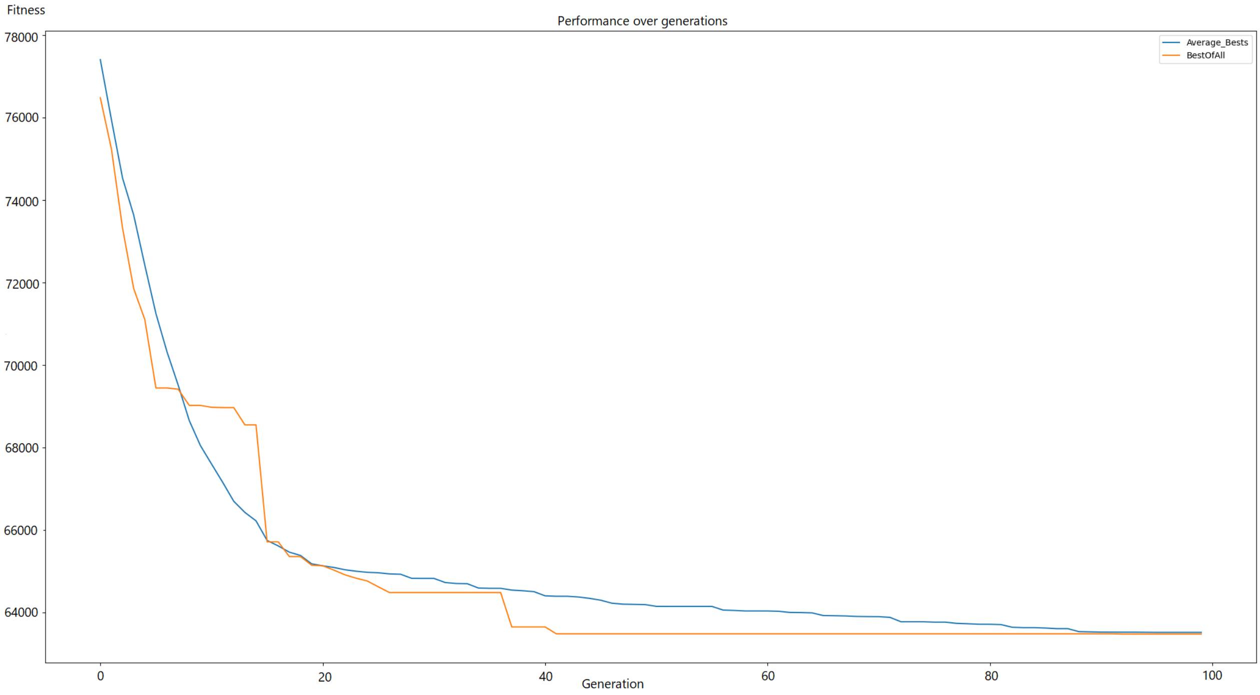


Figure 5.12: Fitness over the generations for the dataset "Map 2" - set of weights 1.

PDO node id (that contains a MDU link)	Ports used (out of 12)	Amount of unique clients served (distinct SDUs/MDUs nodes)
32	11	2
38	11	10
82	9	8

Table 5.14: Amount of ports used in the PDOs that contain a connection to a building (MDU). Dataset "Map 2" set 2.

Extra equipment needed	Quantity
Splitters Type 1x64 (in the JSO/starting point)	3
Cables 8FO	1
Cables 16FO	2
Cables 32FO	3

Table 5.15: Extra equipment needed for the dataset "Map 2" to serve a fiber demand of 126 from 115 distinct homes locations.

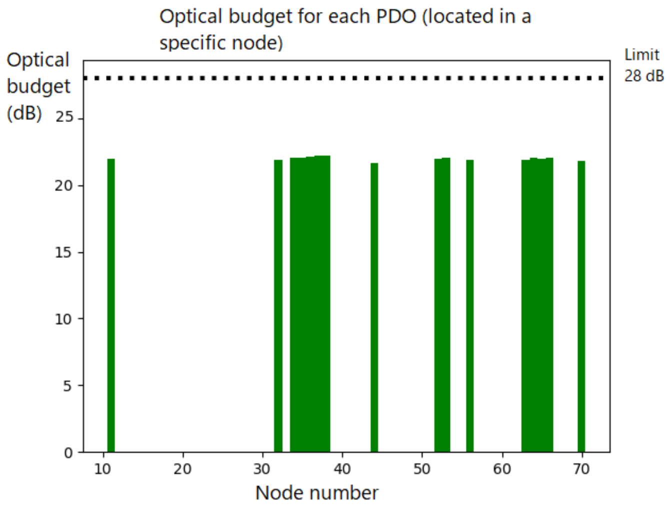


Figure 5.13: Optical power budget for the dataset "Map 2" - set of weights 1. Each bar corresponds to a PDO located in a network equipment node (which has a number id).

the previous map results subsections): first we analyze the GA approach alone for the first scenario, then we analyze the result of the GA in combination with the Steiner Tree pre-processing, a comparison of approaches is also provided between those two approaches (also by checking the handmade one).

Results for the GA (without pre-processing)

For this specific map, we have initially the design result for the approach without pre-processing (i.e., "raw data" provided to the GA) on the Figure 5.14 (set 1 of weights). As for the handmade, it is displayed on Figure 5.15. The metrics for those designs are displayed on the Table 5.16.

Looking at the design plots of the network, we can notice that for the handmade solution, there seems to be a bit of redundancy, where multiple drop cables are connected to the same building (MDU) in the bottom left of the network. This could be related to an error in the dataset provided. Still, it is costly to have such design, which can be seen in the recap table about the fitness score. We can see that the fitness score of the handmade solution is more than the double of the best solutions found by the GA. There are too many unnecessary resources allocated to build the network. Even the average solutions outperform the handmade solution in this scenario in terms of fitness (it is not the case for the worst ones due to the fact that those solutions probably are missing the allocation of a home to the network).

What is the most interesting aspect is the choice of some of the distribution cables routes, which are clearly different between those designs (Figure 5.14 and 5.15), specially considering the upper side of the network (i.e., by ignoring the weird pattern on the bottom left of the handmade solution). We can compare this GA solution with a GA solution with Steiner Tree pre-processing to see the relevancy of the choice of the paths (which one is the best).

Results for the GA with Steiner Tree pre-processing

Figure 5.16 refers to the design plot of the GA with Steiner Tree pre-processing (scenario 1). As can be seen the distribution cables (red edges) are different from the case with GA only. The choice of routes is actually visually close to the handmade solution. The Table 5.17 shows the metrics collected for this scenario for the GA with Steiner Tree. If we compare to the previous Table of the GA without Steiner Tree (Table 5.16), we can see significant improvements in terms of costs and materials required for the best and median solutions. For the best solution, the approach that makes use of the Steiner Tree has a fitness of 23990 whereas the one that uses the GA alone has a fitness of 24828 (a difference of 838 in terms of cost). This goes to show that the Steiner Tree pre-processing to the dataset, has actually a lot of value in practice for our problem (even if the GA by itself managed to find a satisfactory solution). But only in the case that we are dealing with maps where multiple path possibilities are available.

As for the fitness over generations, the plot can be found in Figure 5.18.

Other results

Similarly to the previous results, there is also the scenario 2 (of the GA with Steiner Tree) that is shown in the Figure 5.17 and the corresponding metrics on the Table 5.18. Amongst the solutions found by the GA, the solution that comes closest in terms of design to the handmade seems to be the set 2 where we have more PDOs installed, if we were to remove the redundancy of the handmade solution, the number of PDOs would be similar to the solution of the set of weights 2.

The Table 5.19 and 5.20 respectively shows the amount of ports used where a building is connected for the scenario 1 and 2. We can notice that this map has actually a building with a higher demand than usual, a demand of 22 (signaled by a * in the Tables), whereas our input limit is 12 for the PDOs capacity. This requires a flexibility of the ports constraint in the system to allow this connection (an exception to the rule), otherwise the connection of the node 133 would be impossible. So a placement of a PDO of higher capacity is needed here (for example 24 ports).

As for the time required to execute the code on this map is on average 37 seconds (a single run), which again, is much less than the time required to design such a network manually.

5.3.4 Results for "Map 4"

This last dataset (for which we have a corresponding handmade solution) is also a map from New York, similar to the previous maps subsections, two designs are proposed. The Figure 5.20 and 5.21 respectively show the designs for the set of weights 1 and 2. The Figure 5.22 refers to the handmade solution. This

Metrics \ Approach	**Handmade**	**GA | best (set 1)**	**GA | worst (set 1)**	**GA | median (set 1)**	**GA |mean SD (set 1)**
Number of PDOs	27	14	14	15	16 SD: 0.5
Drop cable used (km)	3.53	3.09	3.29	2.90	3.01 SD: 0.1
Distribution cable used (km)	2.76	1.65	1.57	1.76	1.71 SD: 0.05
Fitness score	50216	24828	115013	25274	26635 SD: 1247
Difference fitness (GA - Handmade)	——-	-25388	64797	-24942	-23581

Table 5.16: Results for the dataset "Map 3" with set 1. Results obtained with 30 runs of GA without pre-processing of the dataset.

Metrics \ Approach	**Handmade**	**GA | best (set 1)**	**GA | worst (set 1)**	**GA | median (set 1)**	**GA | mean SD (set 1)**
Number of PDOs	27	15	14	15	15.38 SD: 0.55
Drop cable used (km)	3.53	2.87	3.01	2.99	3.02 SD: 0.1
Distribution cable used (km)	2.76	1.55	1.51	1.57	1.72 SD: 0.06
Fitness score	50216	23990	144533	24396	27007 SD: 1577
Difference fitness (GA - Handmade)	——-	-26226	-7530	-25820	-23209

Table 5.17: Results for the dataset "Map 3" with set 1. Results obtained with 30 runs of GA with Steiner Tree pre-processing.

Metrics \ Approach	**Handmade**	**GA | best (set 2)**	**GA | worst (set 2)**	**GA | median (set 2)**	**GA |mean SD (set 2)**
Number of PDOs	27	22	21	23	22.75 SD: 0.6
Drop cable used (km)	3.53	2.20	2.17	2.20	2.2 SD: 0.03
Distribution cable used (km)	2.76	1.83	1.70	1.81	2.06 SD: 0.03
Fitness score	152 063	65 541	103 204	64 881	66147 SD: 148.2
Difference fitness (GA - Handmade)	——-	-86 522	-48 859	-87 182	-85916

Table 5.18: Results for the dataset "Map 3" with set 2. Results obtained with 30 runs of GA with Steiner Tree pre-processing.

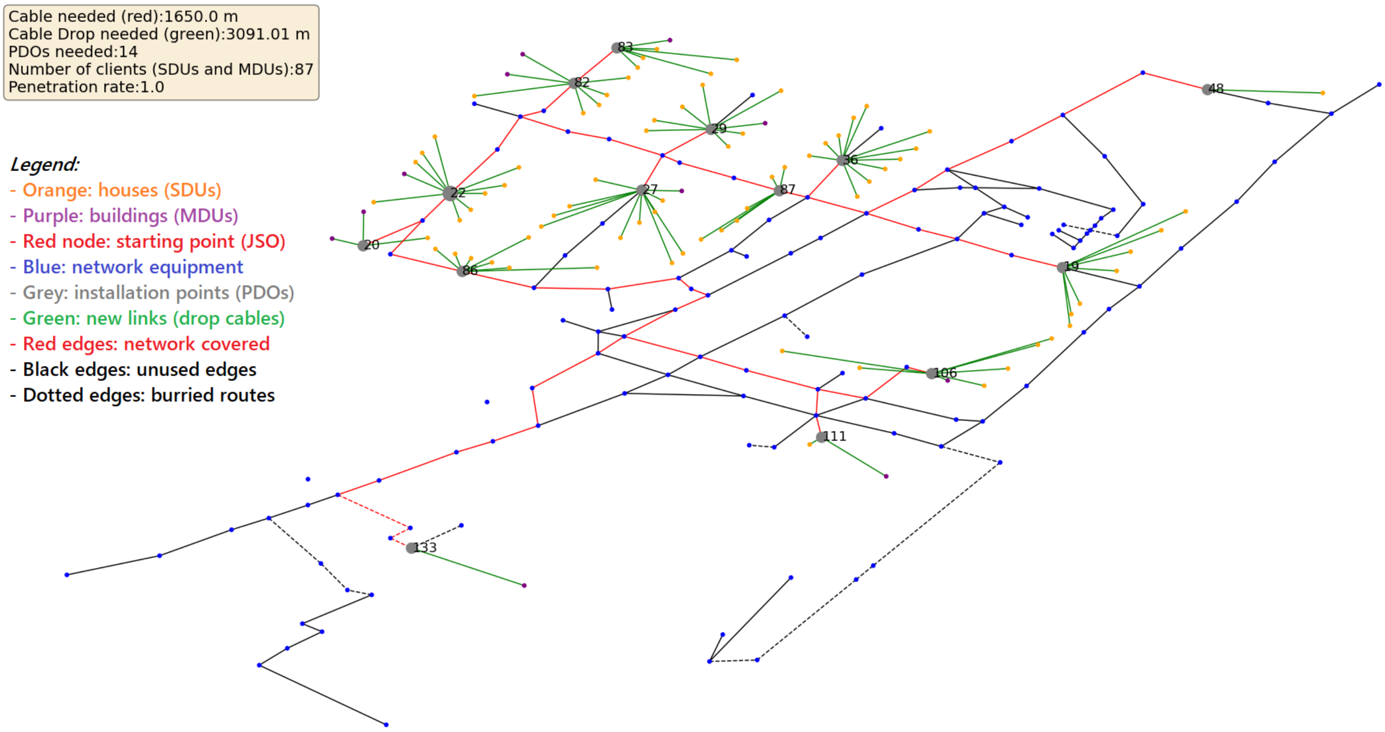

Cable needed (red):1650.0 m
Cable Drop needed (green):3091.01 m
PDOs needed:14
Number of clients (SDUs and MDUs):87
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes
83
82
29
36
48
22
27
87
20
86
19
106
111
133

Figure 5.14: Design generated by the GA (without pre-processing) for the dataset "Map 3" - set of weights 1. The visualized result corresponds to the best solution. Starting point/JSO is on node 22 (it also contains a PDO in this solution).

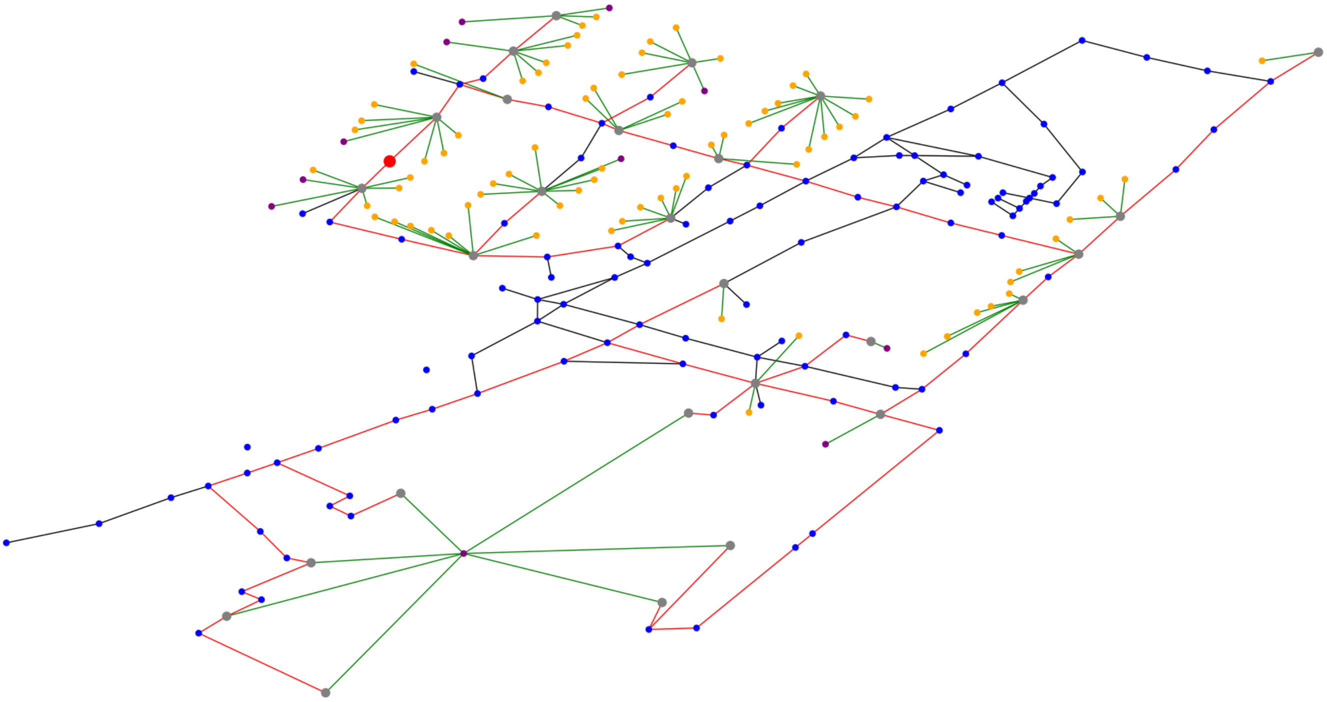


Figure 5.15: Handmade solution for the dataset "Map 3".

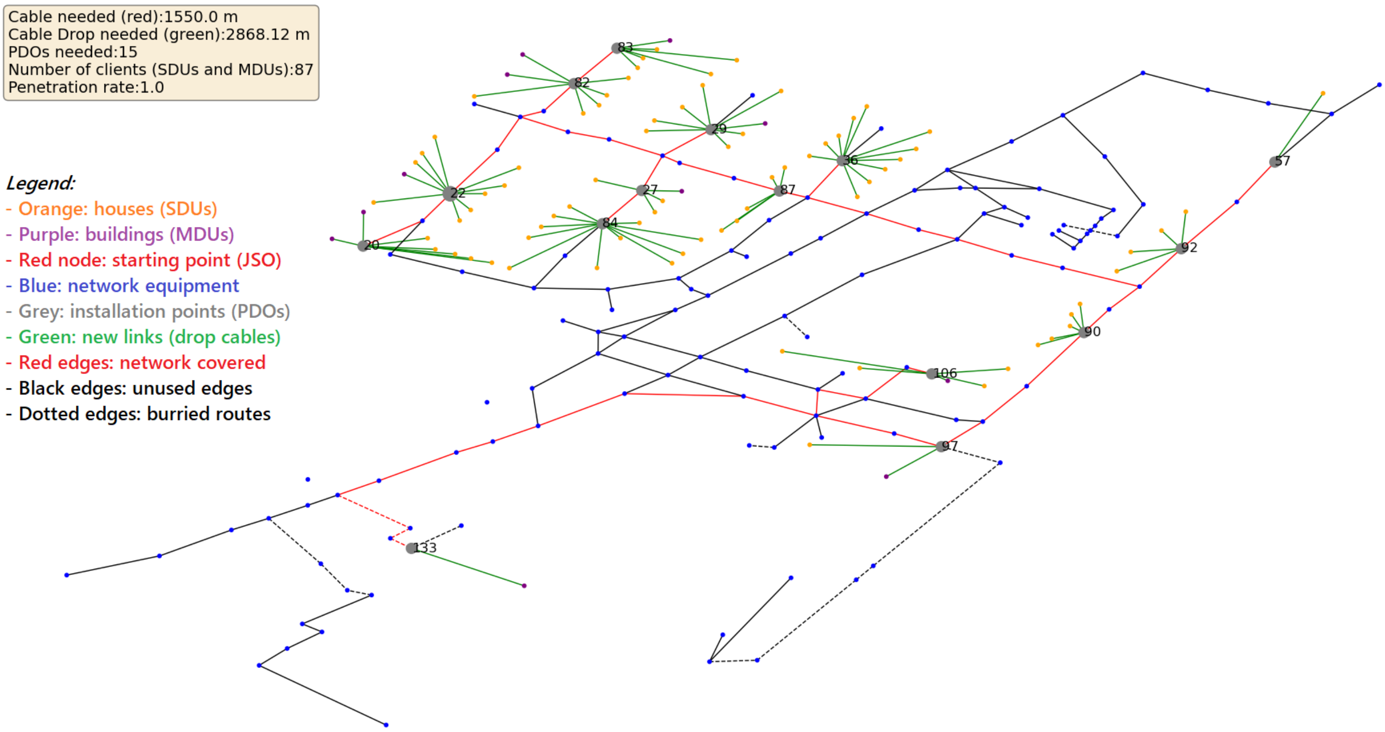

Cable needed (red):1550.0 m
Cable Drop needed (green):2868.12 m
PDOs needed:15
Number of clients (SDUs and MDUs):87
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes
83
82
29
36
22
27
87
20
84
57
92
90
106
97
133

Figure 5.16: Design generated by the GA with Steiner Tree pre-processing for the dataset "Map 3" - set of weights 1. The visualized result corresponds to the best solution. Starting point/JSO is on node 22 (it also contains a PDO in this solution)

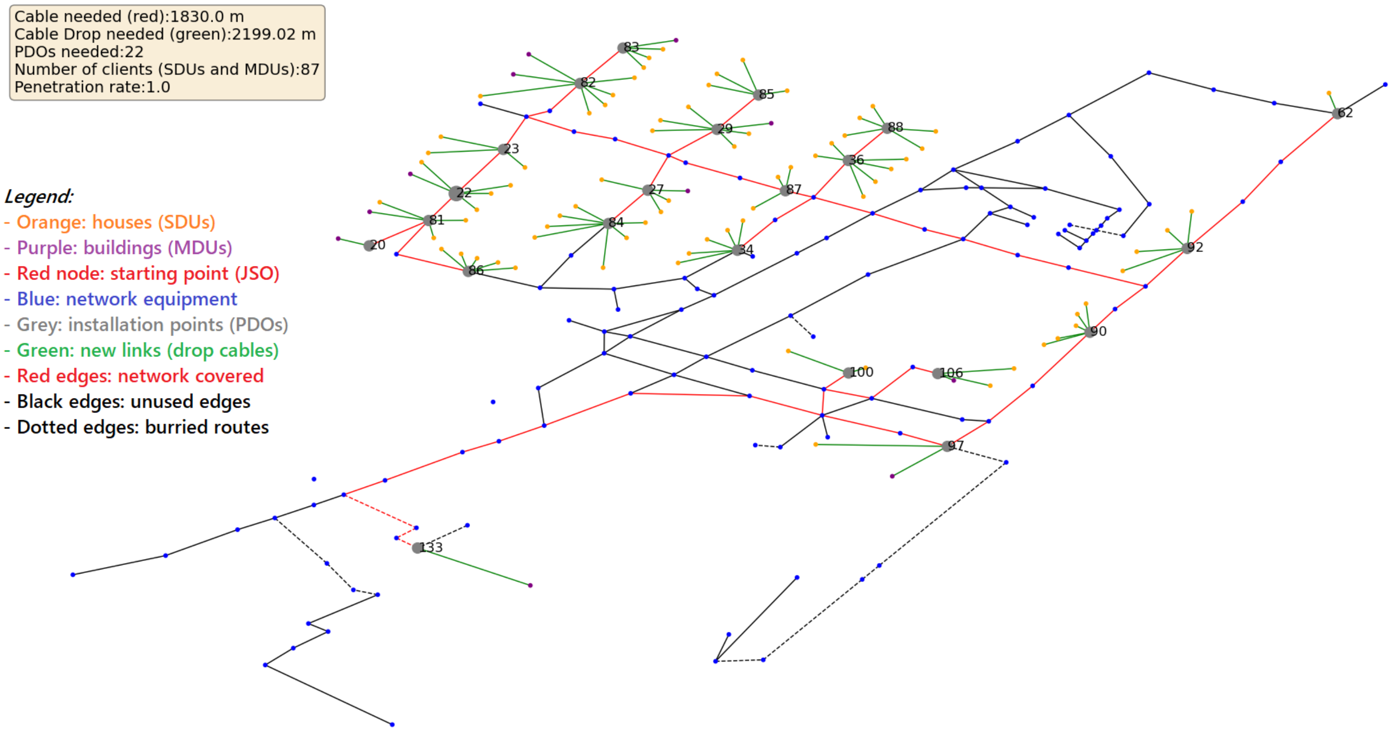

Cable needed (red):1830.0 m
Cable Drop needed (green):2199.02 m
PDOs needed:22
Number of clients (SDUs and MDUs):87
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes
83
82
85
29
23
88
36
22
27
87
81
84
20
34
86
62
92
90
100
106
97
133

Figure 5.17: Design generated by the GA with Steiner Tree pre-processing for the dataset "Map 3" - set of weights 2. The visualized result corresponds to the best solution. Starting point/JSO is on node 22 (it also contains a PDO in this solution).

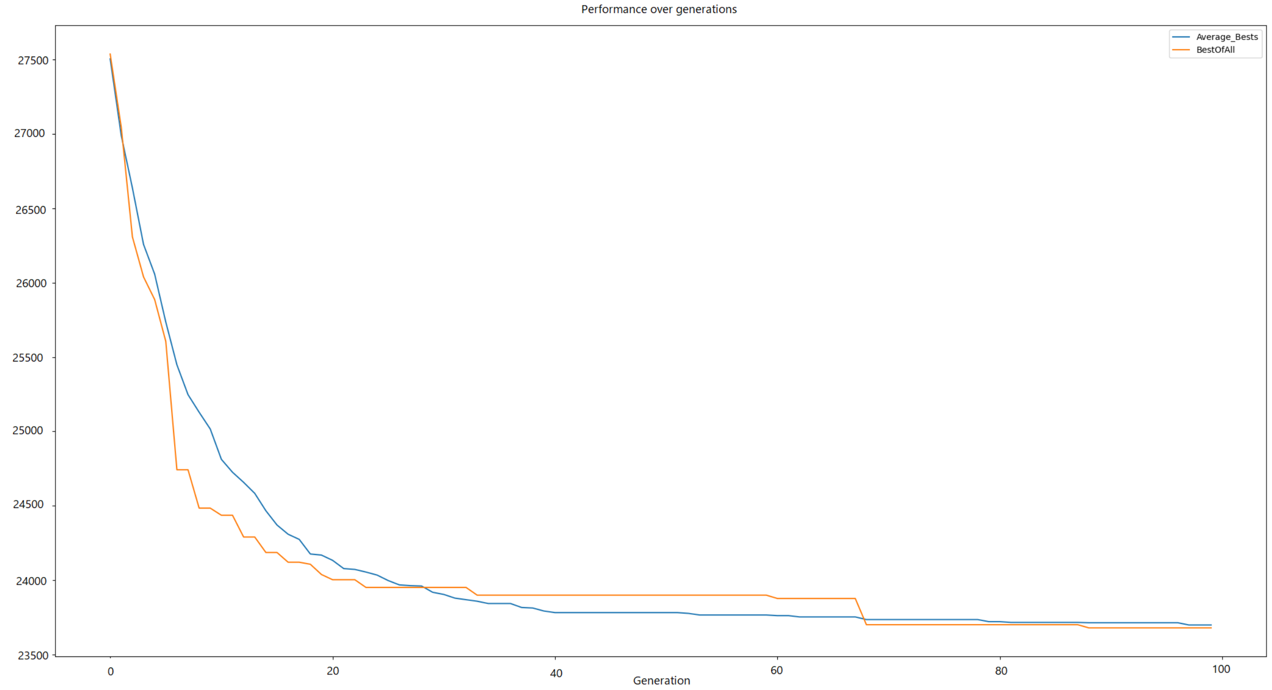


Figure 5.18: Fitness over the generations for the dataset "Map 3" - set of weights 1 - GA with Steiner Tree pre-processing.

PDO node id (that contains a MDU link)	Ports used (out of 12)	Amount of unique clients served (distinct SDUs/MDUs nodes)
20	10	7
22	11	10
82	9	7
83	8	6
97	2	2
106	6	5
133	22*	1

Table 5.19: Amount of ports used in the PDOs that contain a connection to a building (MDU). Dataset "Map 3" with set 1.

PDO node id (that contains a MDU link)	Ports used (out of 12)	Amount of unique clients served (distinct SDUs/MDUs nodes)
20	2	1
22	6	5
81	6	4
82	9	7
83	5	4
85	5	4
106	4	3
133	22*	1

Table 5.20: Amount of ports used in the PDOs that contain a connection to a building (MDU). Dataset "Map 3" with set 2.

Extra equipment needed	**Quantity**
Splitters Type 1x64 (in the JSO/starting point)	2
Cables 16FO	1
Cables 32FO	3

Table 5.21: Extra equipment needed for the dataset "Map 3" to serve a fiber demand of 106 from 87 distinct homes locations.

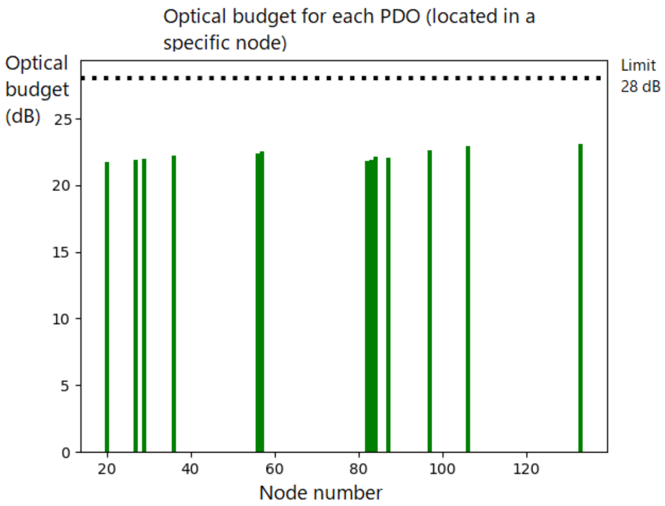


Figure 5.19: Optical power budget for the dataset "Map 3" - set of weights 1. Each bar corresponds to a PDO located in a network equipment node (which has a number id).

dataset seems standard, there is no particular challenge for the algorithm like in the previous map where multiple paths were possible, and where the Steiner Tree showed good results. So for this map it is pretty straight forward (we will focus only on the fitness results and design plots), except for one detail: it was required to allow a bit more drop cable range (115meters) in order for the algorithm to find a solution with a penetration rate of 100 percent (i.e., that covers all the homes), otherwise one of the homes would not be allocated.

Fitness performance results

The Tables 5.22 and 5.23 show the different metrics collected. Once again we can notice the same pattern as for the other maps, overall the system proposed (GA) seems to propose better solutions with much less installation points (PDOs) required and overall materials. The solution that comes close to the handmade is the one for the scenario 2, where the number of PDOs is the closest (54 for the handmade and 42 for the best solution). The fitness plot can be found in Figure 5.23.

Metrics \ Approach	**Handmade**	**GA | best (set 1)**	**GA | worst (set 1)**	**GA | median (set 1)**	**GA | mean SD (set 1)**
Number of PDOs	54	26	28	28	31.14 SD: 1.30
Drop cable used (km)	4.58	5.60	5.25	5.27	5.28 SD: 0.12
Distribution cable used (km)	4.68	3.90	3.85	3.95	3.99 SD: 0.06
Fitness score	58529	45959	135686	46695	49035 SD: 2396
Difference fitness (GA - Handmade)	——	-12570	77157	-11834	-9494

Table 5.22: Results for the dataset "Map 4" with set 1. Results obtained with 30 runs.

Metrics \ Approach	**Handmade**	**GA | best (set 2)**	**GA | worst (set 2)**	**GA | median (set 2)**	**GA | mean SD (set 2)**
Number of PDOs	54	42	42	38	42 SD: 1.63
Drop cable used (km)	4.58	4.18	4.23	4.29	4.18 SD: 0.10
Distribution cable used (km)	4.68	4.23	4.13	4.46	4.40 SD: 0.10
Fitness score	130151	108796	148741	109359	109826 SD: 398
Difference fitness (GA - Handmade)	——	-21355	18590	-20506	-20325

Table 5.23: Results for the dataset "Map 4" with set 2. Results obtained with 30 runs.

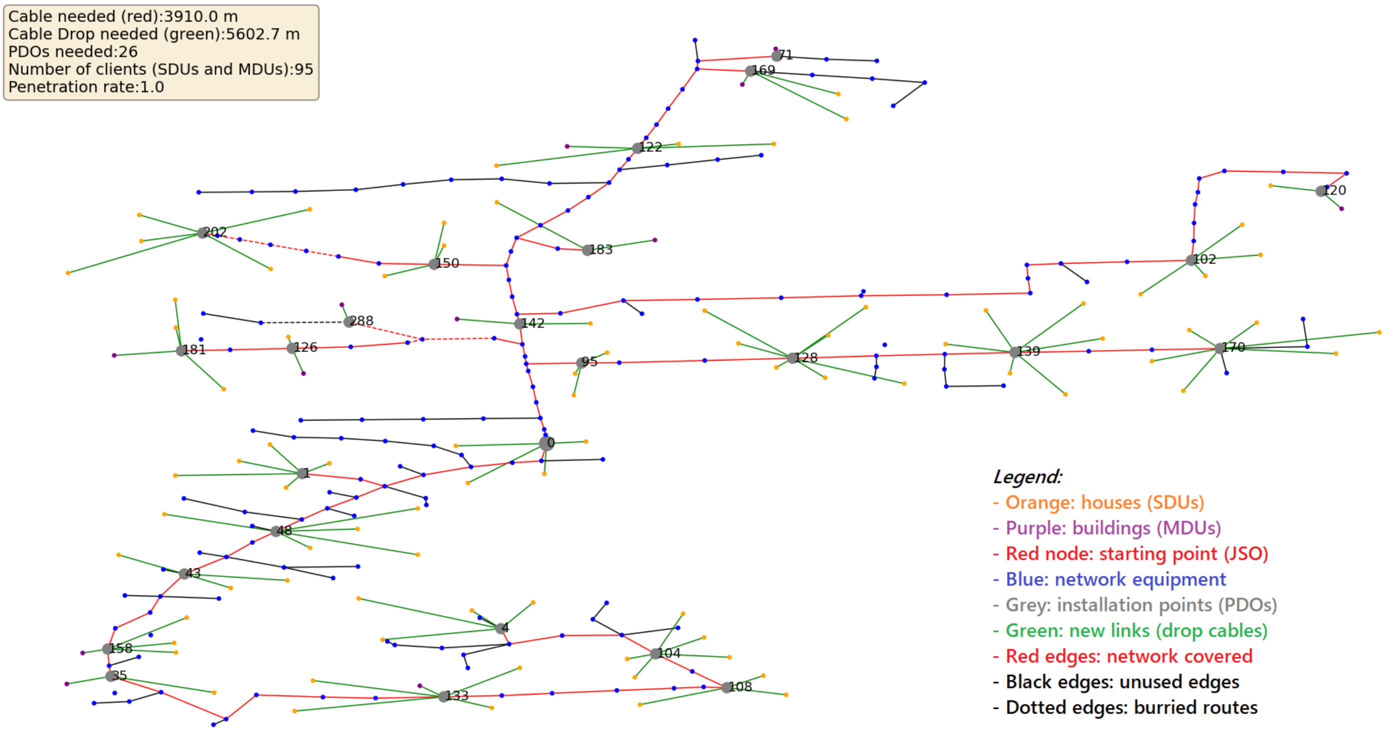

Cable needed (red):3910.0 m
Cable Drop needed (green):5602.7 m
PDOs needed:26
Number of clients (SDUs and MDUs):95
Penetration rate:1.0
71
169
122
120
202
150
183
102
288
142
181
126
95
128
139
170
0
1
48
43
4
104
158
35
133
108
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes

Figure 5.20: Design generated by the GA for the dataset "Map 4" - set of weights 1. The visualized result corresponds to the best solution. Starting point/JSO is on node 0 (it also contains a PDO in this solution).

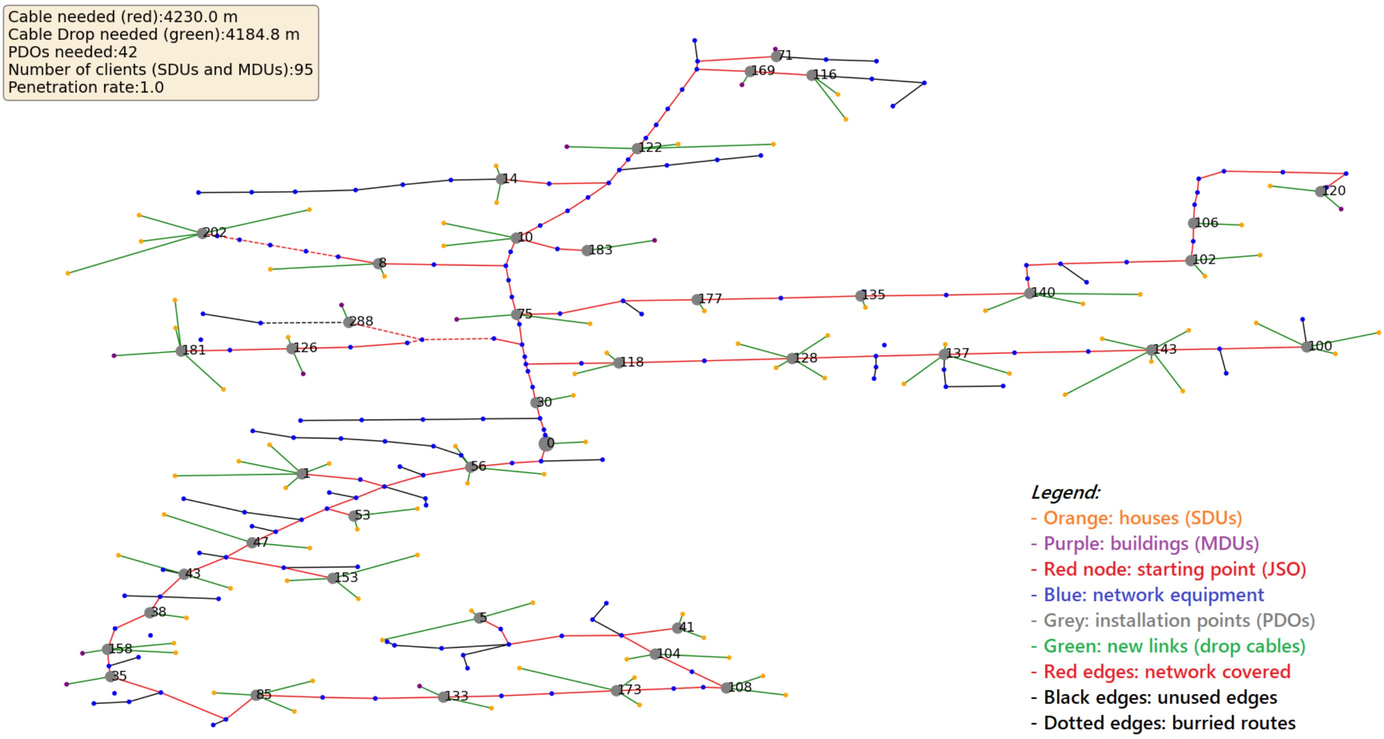

Cable needed (red):4230.0 m
Cable Drop needed (green):4184.8 m
PDOs needed:42
Number of clients (SDUs and MDUs):95
Penetration rate:1.0
71
169
116
122
14
120
202
10
183
106
8
102
177
135
140
75
288
181
126
118
128
137
143
100
30
0
56
1
53
47
43
153
38
5
41
104
158
35
85
133
173
108
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes

Figure 5.21: Design generated by the GA for the dataset "Map 4" - set of weights 2. The visualized result corresponds to the best solution. Starting point/JSO is on node 0 (it also contains a PDO in this solution).

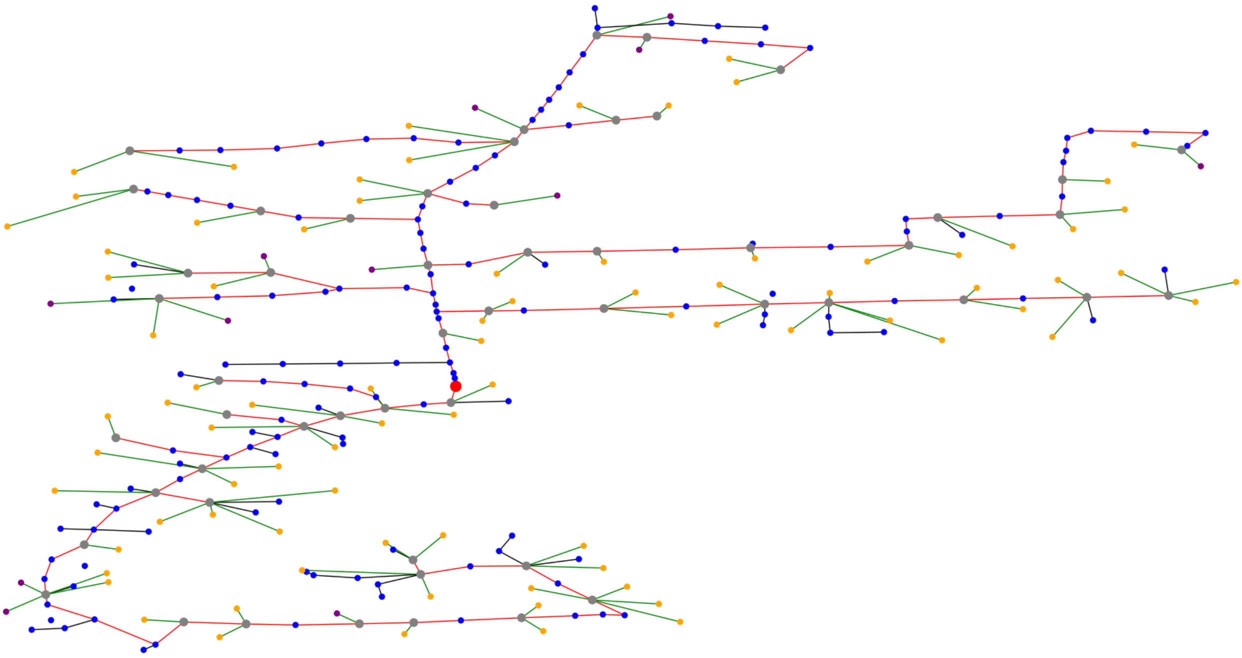


Figure 5.22: Handmade solution for the dataset "Map 4".

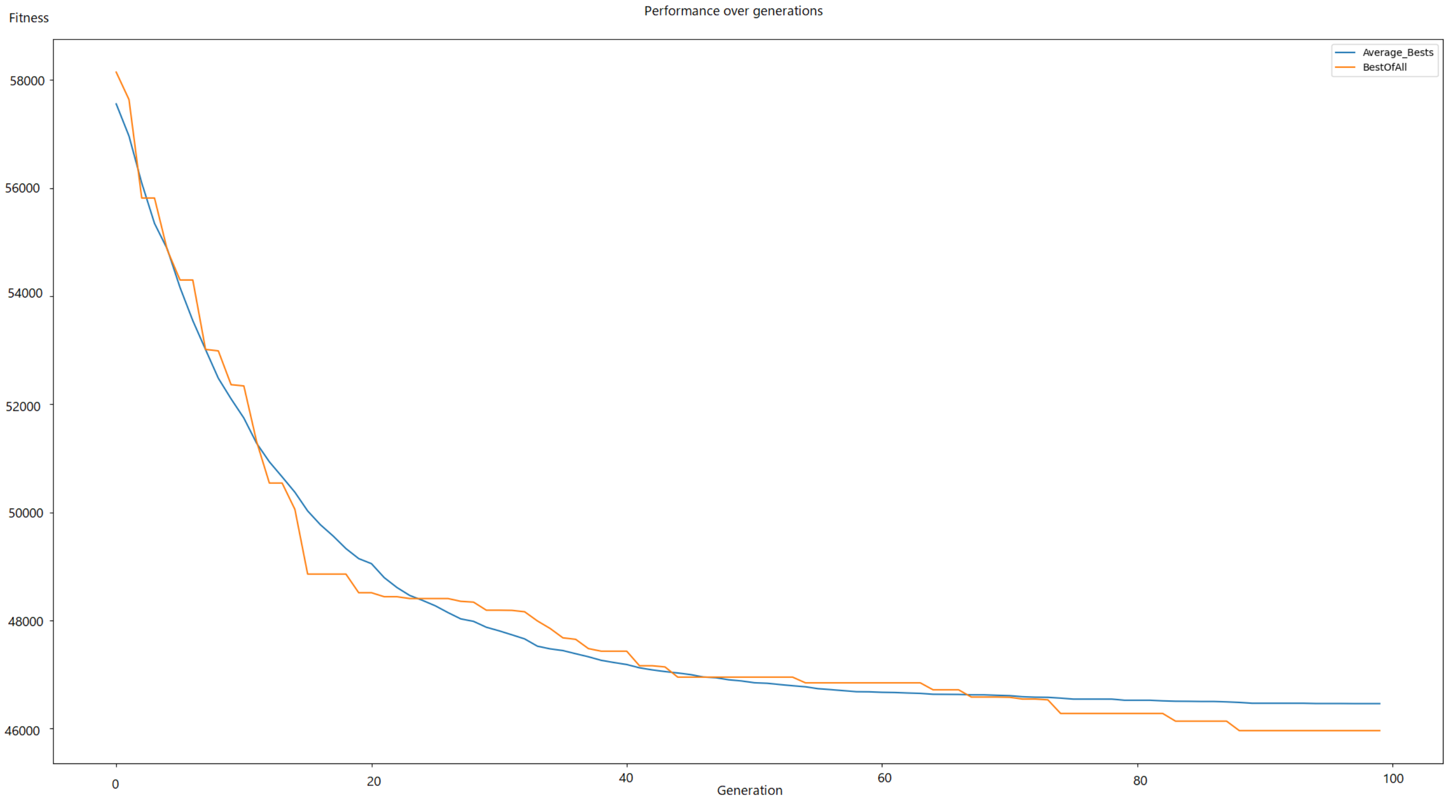


Figure 5.23: Fitness over the generations for the dataset "Map 4" - set of weights 1.

Data set name (map)	Number of locations (nodes)	Number of routes (edges)	Number of valid network equipment nodes	Total clients: Houses(SDUs) Buildings (MDUs)	Sum of the distance of all routes
Map 5	273	192	188	49 & 36	8.28 km
Map 6	305	205	205	88 & 12	10.65 km
Map 7	7244	4570	4442	2107 & 292	187 km

Table 5.24: Details of the different maps initially (for which we do not have a corresponding handmade solution).

5.4 Extra Design Results

This section showcases some extra datasets for which there is no corresponding handmade solution. The goal is simply to propose the visual results (design plots) for those maps due to the fact that we can not compare with an existing solution. Additionally, we will take interest on the scalability of the system by running it on bigger datasets.

Table 5.24 details these extra maps.

Design for "Map 5"

Figures 5.24 and 5.25 respectively show the design proposal for the map "Map 5" for the scenario 1 and 2. The solutions correspond to the best individual of the multiple runs (30). A solution can be found in 60 seconds for this map.

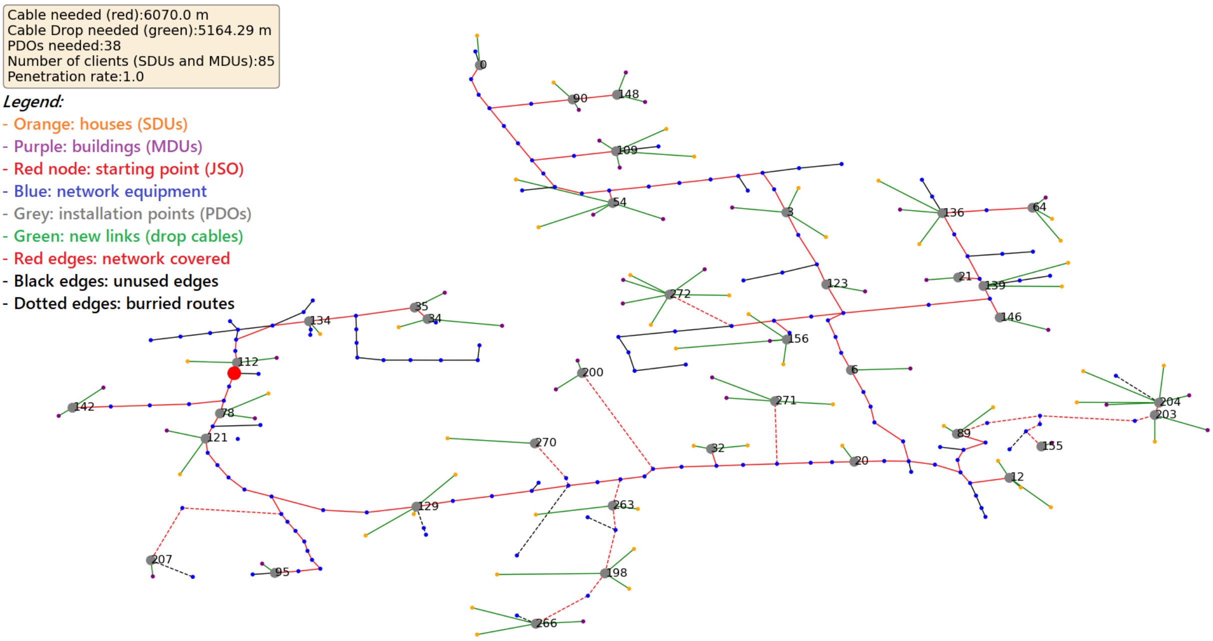

Cable needed (red):6070.0 m
Cable Drop needed (green):5164.29 m
PDOs needed:38
Number of clients (SDUs and MDUs):85
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes
0
90
148
109
54
3
136
64
21
139
123
272
146
35
34
134
112
156
6
200
142
78
271
204
203
121
270
32
89
155
20
12
129
263
207
95
198
266

Figure 5.24: Design generated by the GA for the dataset "Map 5" - set of weights 1. The visualized result corresponds to the best solution.

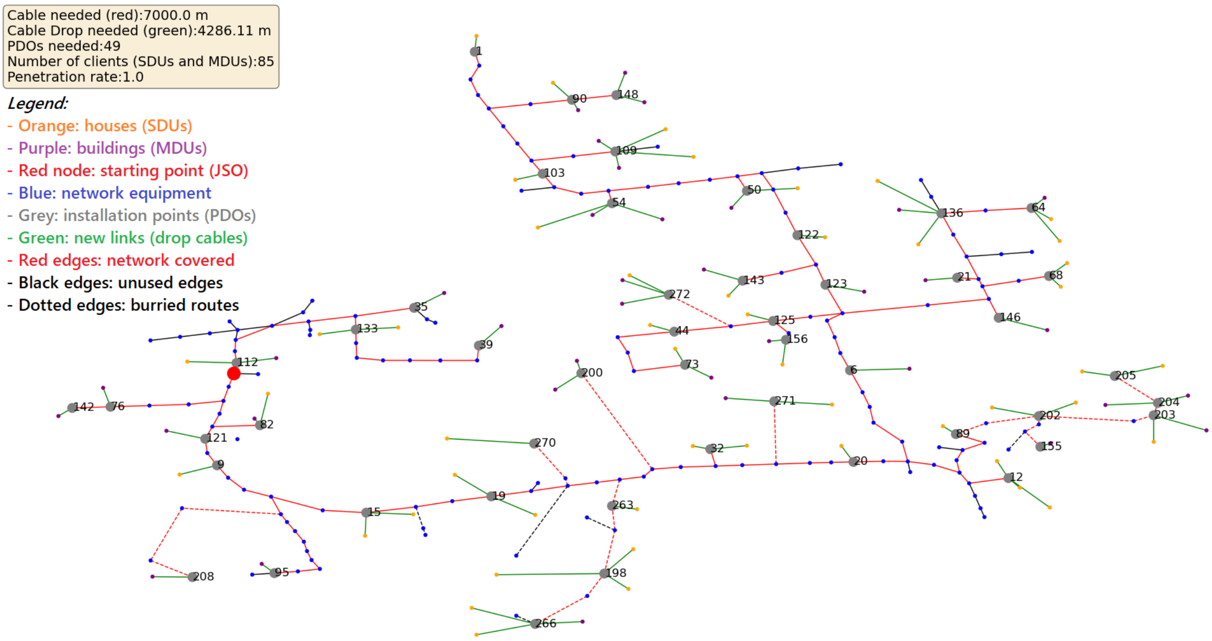

Cable needed (red):7000.0 m
Cable Drop needed (green):4286.11 m
PDOs needed:49
Number of clients (SDUs and MDUs):85
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes
1
90
148
109
103
54
50
136
64
122
143
123
21
68
272
125
146
156
44
35
133
39
73
200
6
112
142
76
82
121
9
271
270
202
204
205
203
89
155
32
20
12
19
15
263
198
208
95
266

Figure 5.25: Design generated by the GA for the dataset "Map 5" - set of weights 2. The visualized result corresponds to the best solution.

Design for "Map 6"

Similarly, Figures 5.26 and 5.27 respectively show the design proposal for "Map 6" for the scenario 1 and 2. The execution time for this dataset is around 68 seconds (for a single run).

Design for "Map 7"

Finally, the last extra dataset is the biggest one, which contains 7244 nodes and 4540 edges. The design proposed is only for the first scenario (set of weights 1) due to the time required to perform a run on networks of this size. Also, a reduction of the GA parameters was considered, instead of 100 generations and 100 population size, we used 50 generations and 50 population size to speed up the process.

The design for this map is shown in Figure 5.28. It can be noted that for this map, the penetration rate is not 100%, this is due to the fact that some homes are out of drop cable range and can not be assigned to the network unless an exception for those homes is considered.

In terms of execution time, the system managed to obtain a solution for this dataset in around 5 hours for those parameters, which is not that long if considering the amount of work that it would require to perform this task manually. It is worth to mention that manually, this kind of design is not performed in one go, it is often split in smaller datasets and combined later.

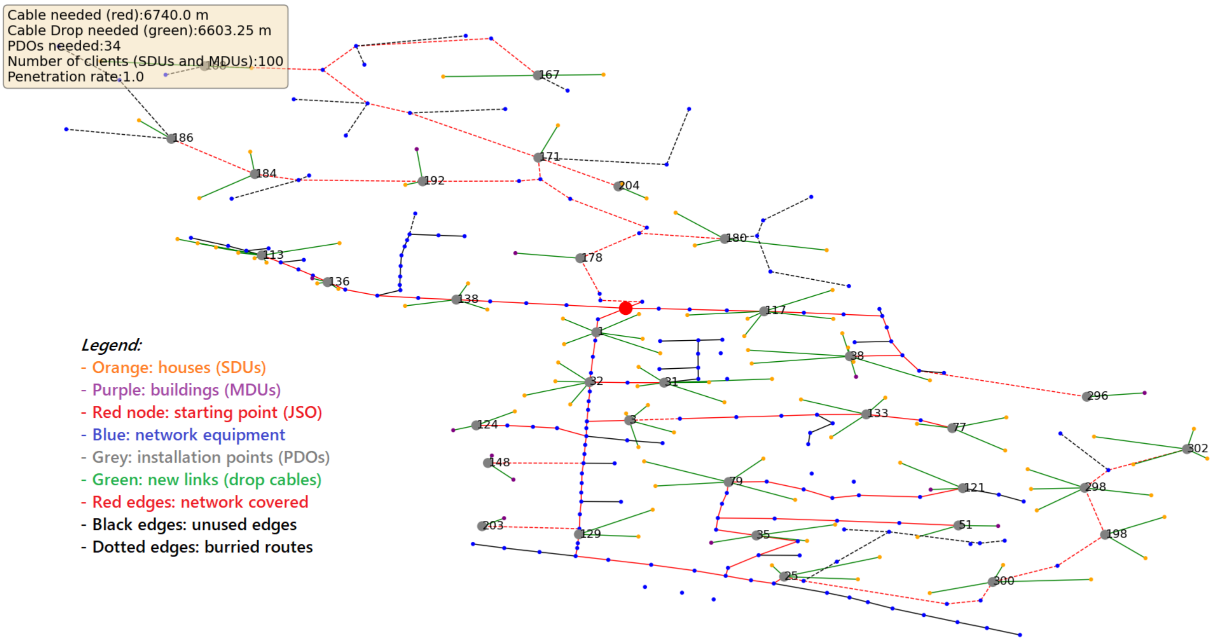

Cable needed (red):6740.0 m
Cable Drop needed (green):6603.25 m
PDOs needed:34
Number of clients (SDUs and MDUs):100
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes
206
167
186
171
184
192
204
180
113
178
136
138
117
1
38
32
31
296
124
3
133
77
302
148
79
121
298
203
129
35
51
198
25
300

Figure 5.26: Design generated by the GA for the dataset "Map 6" - set of weights 1. The visualized result corresponds to the best solution.

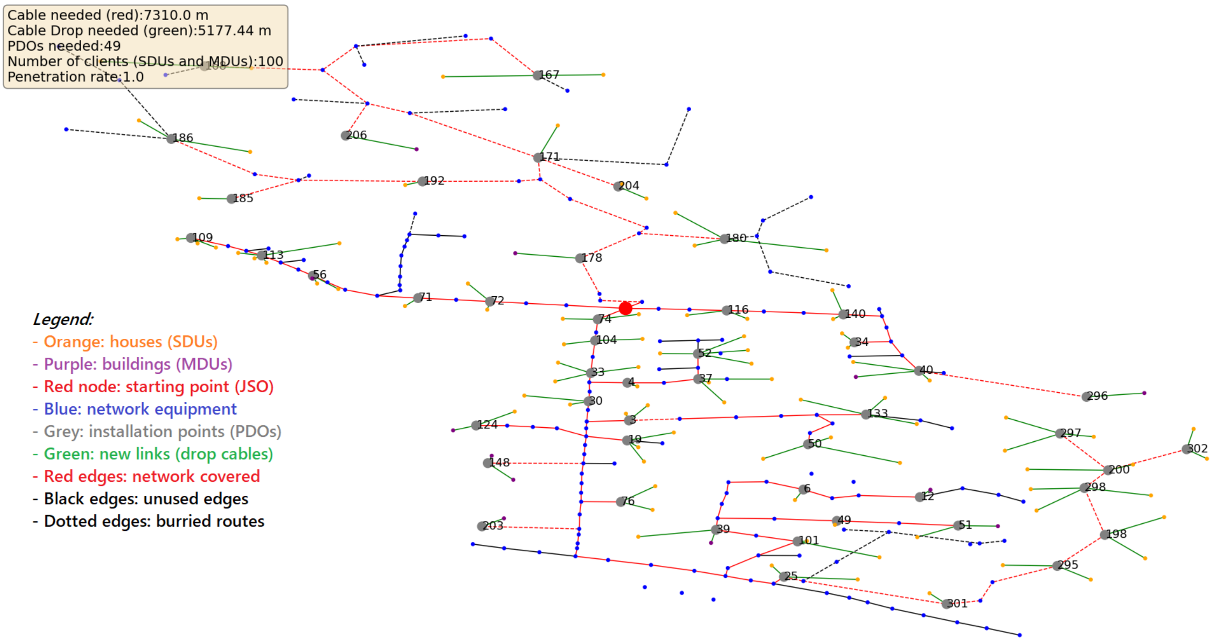

Cable needed (red):7310.0 m
Cable Drop needed (green):5177.44 m
PDOs needed:49
Number of clients (SDUs and MDUs):100
Penetration rate:1.0
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes
167
186
206
171
192
204
185
180
109
113
178
56
71
72
116
74
140
104
34
52
33
4
37
40
30
296
124
133
3
297
19
50
302
148
200
6
12
298
76
49
203
51
39
101
198
295
25
301

Figure 5.27: Design generated by the GA for the dataset "Map 6" - set of weights 2. The visualized result corresponds to the best solution.

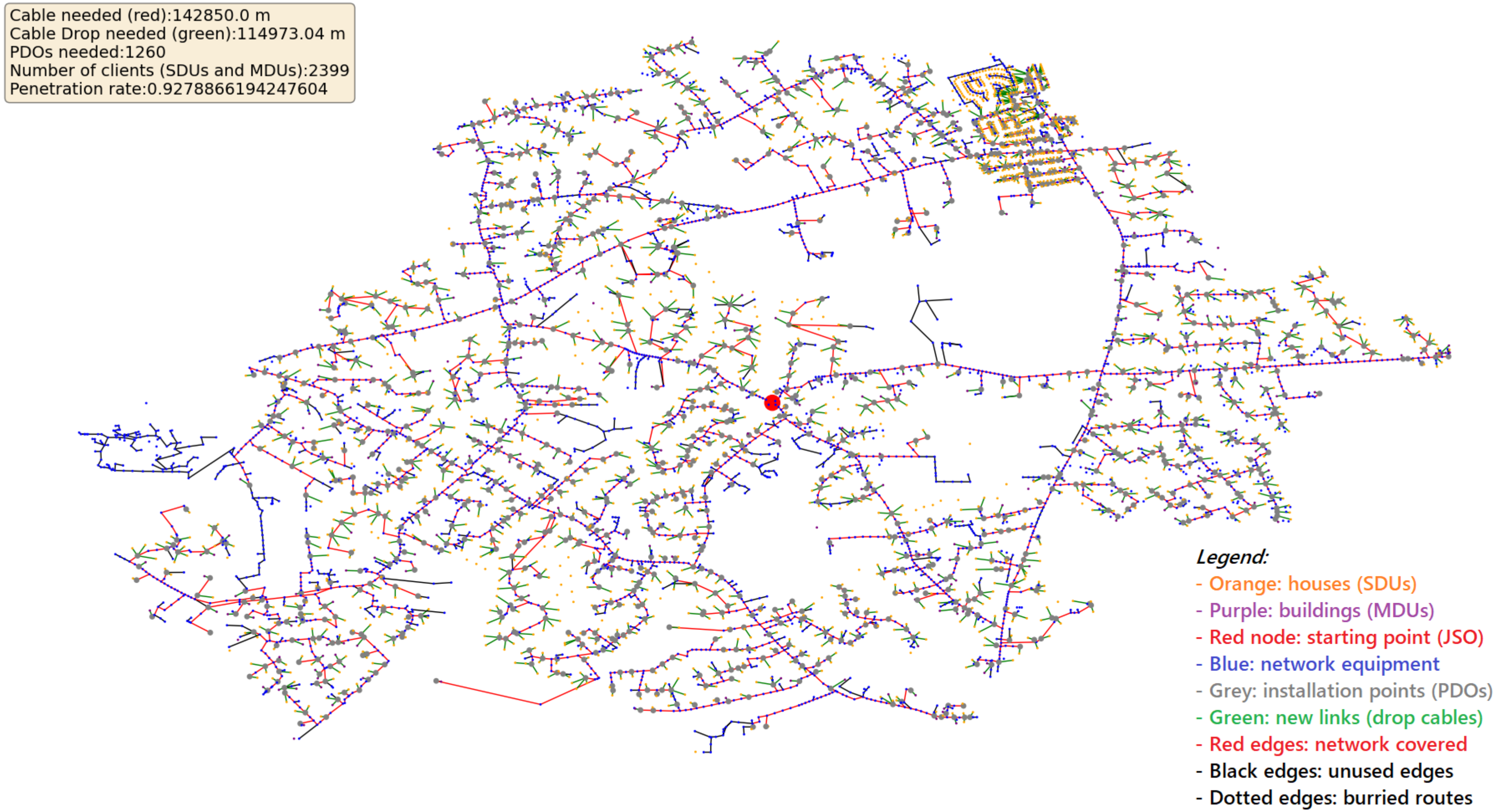

Cable needed (red):142850.0 m
Cable Drop needed (green):114973.04 m
PDOs needed:1260
Number of clients (SDUs and MDUs):2399
Penetration rate:0.9278866194247604
Legend:
- Orange: houses (SDUs)
- Purple: buildings (MDUs)
- Red node: starting point (JSO)
- Blue: network equipment
- Grey: installation points (PDOs)
- Green: new links (drop cables)
- Red edges: network covered
- Black edges: unused edges
- Dotted edges: burried routes

Figure 5.28: Design generated by the GA for the dataset "Map 7" - set of weights 1.

5.5 Summary of the Experiments

To conclude on this chapter, we first defined the experimental setup where two scenarios with different weights (costs for the materials) were considered to generate diverse design results. Additionally, the problem constraints and the GA parameters were also fixed for the rest of the experiments.

For the GA parameters, a validation step was required to test different GA parameters, it allowed us to fix the GA parameters for the rest of the experiments and eventually for future tests. It was concluded that a lower mutation rate (fixed to $2/len$, with *len* the size of the individual) and the Bitflip function were more adapted due to the sensibility of the problem, i.e., the performance of the algorithm is better when smaller changes are made. Also, a comparison with and without Local Search was also performed to deal with the sub-problem of the capacitated PDOs, and it was decided that the Local Search version was able to generate better solutions, visually speaking (cleaner drop cables links) and in terms of fitness.

As for the design results generated automatically, a first set of maps with a corresponding handmade solutions was analyzed. The improvements in terms of fitness (cost) in comparison to the handmade solutions for the different maps and for the different scenarios were the following (taking into account the best individual of the multiple runs):

- "Map 1" - Scenario 1: saved amount of 10597 (31% saved).
- "Map 1" - Scenario 2: saved amount of 11045 (12% saved).
- "Map 2" - Scenario 1: saved amount of 14364 (18.5% saved).
- "Map 2" - Scenario 2: saved amount of 11045 (10.5% saved).
- "Map 3" - Scenario 1: saved amount of 26226 (52.2% saved).
- "Map 3" - Scenario 2: saved amount of 11045 (56.9% saved).
- "Map 4" - Scenario 1: saved amount of 12570 (21.5% saved).
- "Map 4" - Scenario 2: saved amount of 21355 (16.4% saved).

We can notice significant improvements in terms of fitness, those are related to the materials used, the GA managed to find better solutions that require less materials such as cables and installation points. The solutions also respect the defined constraints and they maximize the coverage of the network (i.e., a penetration rate of 100%).

Finally, a second set of maps was tested, even if there was no corresponding handmade solution to perform a comparison, we can see that the design results seem very promising. It is also worth to mention that the proposed system is able to generate such solutions in less than a minute for the chosen parameters and for those medium sized networks (typically around 200-300 nodes), which is much faster than doing this task manually.

Chapter 6

Conclusion

The main goal of this thesis was to study GPON networks and to develop an AI system to design such networks in an optimal way.

The system proposed is based on Graph Theory combined with Nature-inspired meta-heuristic approaches, more specifically Genetic Algorithms. The choice of such approach is justified by the fact that in this system we aimed to deal with larger instances of the problem in reasonably times and such approximation algorithms are known to provide a good solution in lesser time than an exact approach. Plus, it provided a good flexibility to deal with the different constraints of the problem at hand.

The proposed system only requires as input the different problem variables, such as the business rules, the constraints and the geographical map of the network. From there it automatically computes the best locations for equipment placement by respecting the multiple constraints involved such as the limit drop cable ranges, equipment capacity and budget. To aid the user of such system, a basic graphical user interface (GUI) with mouse-hovering options was also developed to help in the analysis of the generated networks. A description of this GUI is provided in the Appendix A.

To validate the approach, multiple experiments were performed, the first set of the experiments concerned the validation of the GA parameters. The aim of this set was to find the best configuration of parameters which is crucial to obtain a good solution when dealing with such type of algorithms. Second set was focused on performing tests on different data sets and scenarios with different sets of material costs to find multiple designs for the maps provided by our collaborator. For each map, a comparison with an handmade solution was studied to find the best design between the solutions generated by the system and the handmade one. An attempt to match the handmade design was performed and a proposal of new design was also considered.

Overall, the system seems to be able to generate better solutions than the handmade ones. Moreover it optimizes the solution by minimizing the costs required to build this type of fiber network. In the end, for each map we were more interested on the *best* individual of the multiple GA runs as it corresponds to the network with the lowest cost. Another advantage of this kind of system is that it allows to find satisfactory solutions in short times. Usually the design of a handmade solution can take up to multiple hours or eventually days, depending on the complexity of the network, whereas the AI system proposed can generate solutions in seconds or a few minutes, depending on the size of the map and on the GA parameters.

Finally, we can highlight some potential future work:

- Reduce the execution time even more.
- Development of new variants of the algorithm such as Swarm Intelligence or LP and a comparison with the current approach (GA).
- Potential API integration or eventually the development of a website to aid in the design of networks with the current approach. With additional features such as improved GUI, currently it is simply a mouse-hovering graphical interface that contains the data of each node and edge of the network. The improved GUI could for example be dynamic, allowing the user of the system to modify the parameters at will and automatically updating the solution.

References

Rais Ali, Tim Glover, Michael Kampouridis, and Edward Tsang. Guided local search for optimal gpon/fttp network design. In *Computer Networks Communications (NetCom)*, UK.

Chu Andrej, Poon Kin, and Ouali Anis. An enhanced ant colony optimization for ftth access network design. In *2012 17th European Conference on Networks and Optical Communications*, Abu Dhabi,UAE, 2012.

Chu Andrej, Poon Kin, and Ouali Anis. Using ant colony optimization to design gpon-ftth networks with aggregating equipment. In *2013 IEEE Symposium on Computational Intelligence for Communication Systems and Networks (CIComms)*, 2013.

Chardy, Costa, Faya, and Trampont. Optimizing splitter and fiber location in a multilevel optical ftth network. In *European Journal of Operational Research*, France, 2012.

Thomas Cormen. Prim algorithm. In *Introduction to algorithms*.

Ernesto Costa. In *Nature-Inspired Artificial Intelligence (primer, EC course)*, Coimbra, Portugal, 2022.

FTTH Council. Fiber-to-the-home handbook edition 7. Europe, 2016.

Leonardo Dias, Alex Santos, Helder Pereira, Raul Almeida, William Giozza, Rafael Sousa, and Karcius Assis. Evolutionary strategy for practical design of passive optical networks. In *Photonics 2022, 9, 278*, 2022.

Michael Dinitz. 601.435/635 approximation algorithms. 2019.

Eiben and Smith. Introduction to evolutionary computing second edition. In *Introduction to Evolutionary Computing Second Edition*, 2015.

Antonio Eira, João Pedro, and João Pires. Optimized design of multistage passive optical networks. In *J. Opt. Commun, vol. 4 num. 5*, 2012.

EXFO. Ftth pon guide. 2012.

Nghia Hoang. Ftth network optimization. In *Journal of telecommunications and information technology*, 2014.

Kokungal and Ari. Optimization of passive optical network planning. In *Applied Mathematical Modelling, Volume 35, Issue 7*, Turkey, 2011.

Bin Lin. Cascaded splitter topology optimization in lrpons. 2012.

Javier Mata. Artificial intelligence (ai) methods in optical networks: A comprehensive survey. UK, 2018.

R. Morais, C. Pavan, A. Pinto, and C. Requejo. Genetic algorithm for the topological design of survivable optical transport networks. In *Journal of Optical Communications and Networking, vol. 3*, Portugal, 2011.

Konstantinos Papaefthimiou. Algorithmic pon/p2p ftth access network design for capex minimization. Serbia, Belgrade, 2013.

Villalba Tany. Design of passive optical networks using genetic algorithm. In *SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*, Brazil, 2009.

Ubaidillah, Alfita, and Toyyibah. Link power budget and traffict qos performance analysis of gygabit passive optical network. In *J. Phys.: Conf. Ser. 953 012129*, Madura, Indonesia, 2018.

Bang Ye Wu and Kun-Mao Chao. Steiner minimal trees.

Appendices

Appendix A

Overview of the system's GUI

The system developed to optimize the design of fiber networks in this thesis comes with a basic UI that allows to user of the system to check the different information about the network. The objective of this appendix is to showcase the different information available on the GUI and how to visualize it. By running the system to automatically design the fiber network, the user can obtain a certain amount of data about the network after the optimization process finishes. A basic graphical user interface (GUI) using Matplotlib (Python) opens up and the user can interact with it with mouse-hovering features to check the routes and locations data (i.e., edges and nodes of the graph). It is worth to mention that all the network data can be saved in a JSON output file and reloaded later.

A.1 Locations Information

The locations (nodes) can be either network equipment nodes (i.e poles, pedestals, JSO etc.) where equipment can be installed or they can be homes (houses or buildings). The data depends on the type of node, but every node has the same basis:

- Unique identifier.
- Node name and number.
- Geographical coordinates.
- Type of node.

The type of node is what defines if it is a home (SDU or MDU) or a network equipment node (pedestal, pole, junction, lockbox etc.).

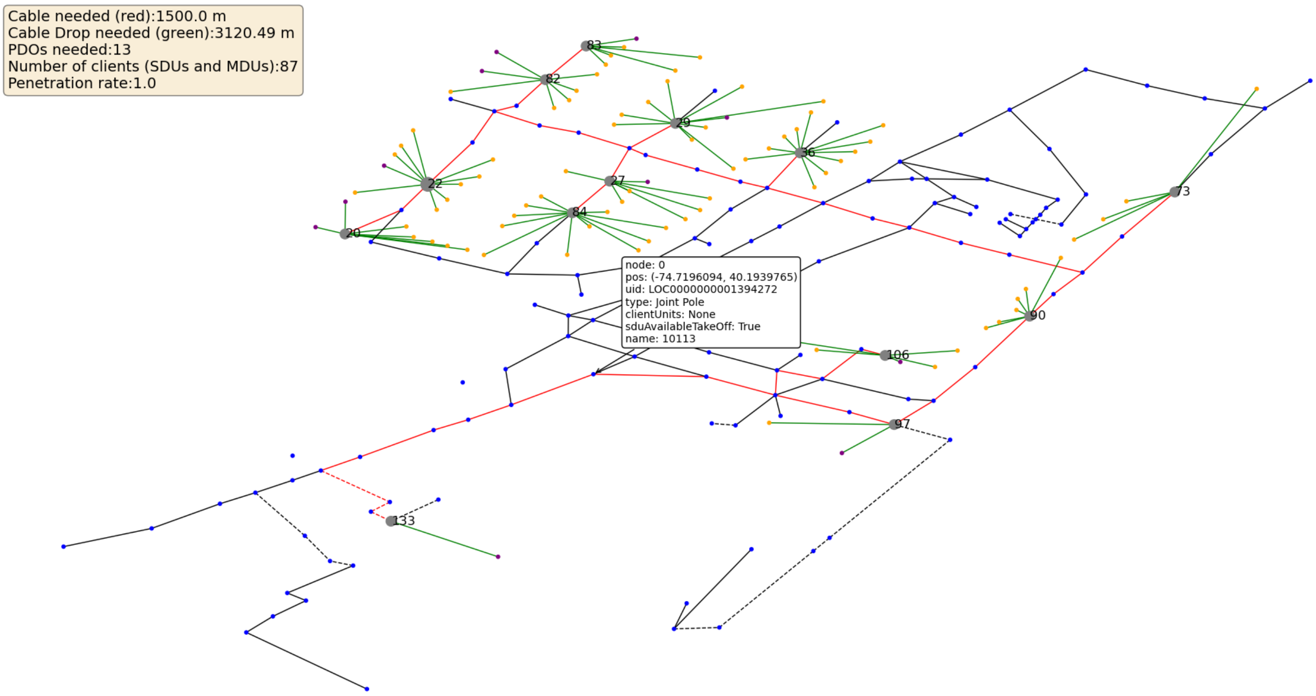
The legend for the graph is the following (same as for the core of this document):

- Orange: houses (SDUs).
- Purple: buildings (MDUs).

- Red node: starting point (JSO).
- Blue: network equipment node.
- Grey: installation points (PDOs).
- Green: new links (drop cables).
- Red edges: network covered (distribution cables).
- Black: unused edges.
- Dotted edges: buried routes.

A.1.1 Equipment Nodes

An unused equipment node (i.e there is no PDO installed that links to a home) can be seen in Figure A.1 with mouse-hovering, in the Figure, "sduAvailableTakeOff" means that this node can connect to a home basically.



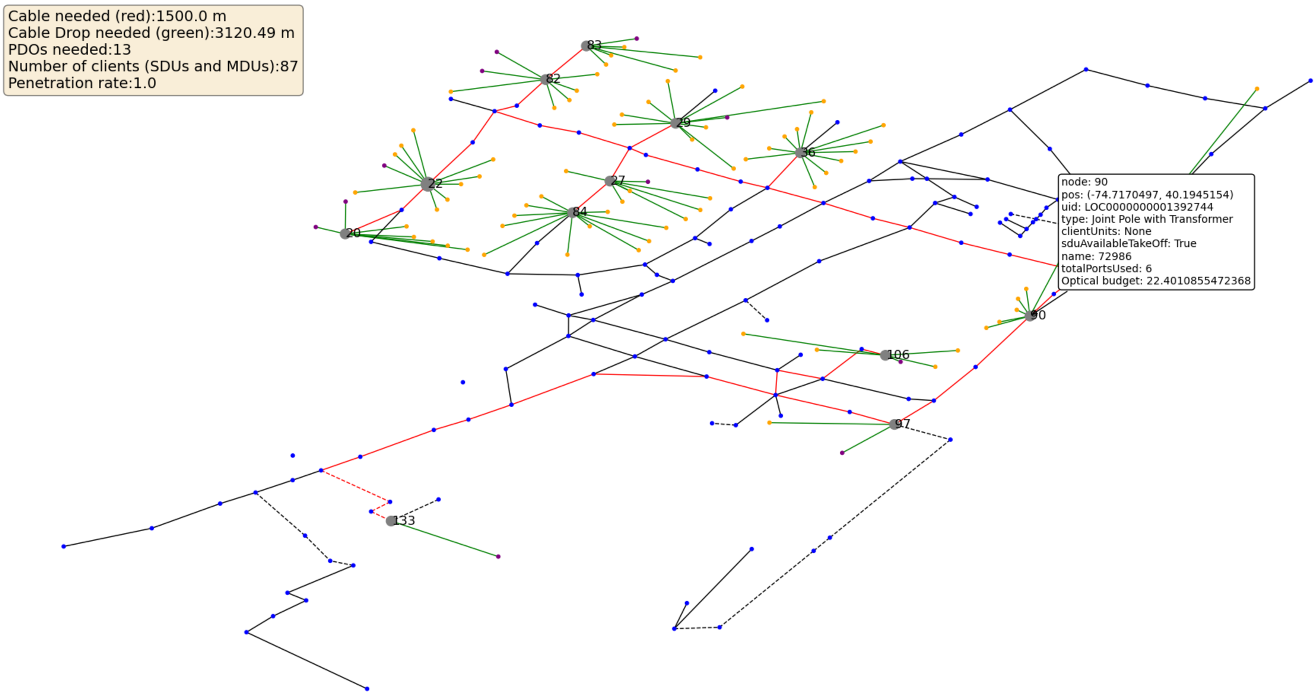
Cable needed (red):1500.0 m
Cable Drop needed (green):3120.49 m
PDOs needed:13
Number of clients (SDUs and MDUs):87
Penetration rate:1.0

node: 0
pos: (-74.7196094, 40.1939765)
uid: LOC0000000001394272
type: Joint Pole
clientUnits: None
sduAvailableTakeOff: True
name: 10113

Figure A.1: Mouse-hovering feature - unused node.

As for a used node (that contains a PDO), the mouse-hovering feature shows additional information: the number of ports used ("totalPortsUsed") for the PDO and the optical budget computed for the selected node. The Figure A.2 shows the mouse-hovering box with the data for this kind of node.

Concerning the JSO (starting point node), it can also contain a PDO installed, it shows the data for the whole graph such as the amount of resources needed to build it (cables amounts, demand, penetration rate, the number of splitter of type *x* that it is required etc.). The Figure A.3 shows an example of the mouse-hovering data for the JSO node. It is in grey (slightly bigger size than the rest) due to the fact that the JSO is also a PDO (node number 22).



Cable needed (red):1500.0 m
Cable Drop needed (green):3120.49 m
PDOs needed:13
Number of clients (SDUs and MDUs):87
Penetration rate:1.0

node: 90
pos: (-74.7170497, 40.1945154)
uid: LOC0000000001392744
type: Joint Pole with Transformer
clientUnits: None
sduAvailableTakeOff: True
name: 72986
totalPortsUsed: 6
Optical budget: 22.4010855472368

Figure A.2: Mouse-hovering feature - PDO node.

A.1.2 Clients Nodes

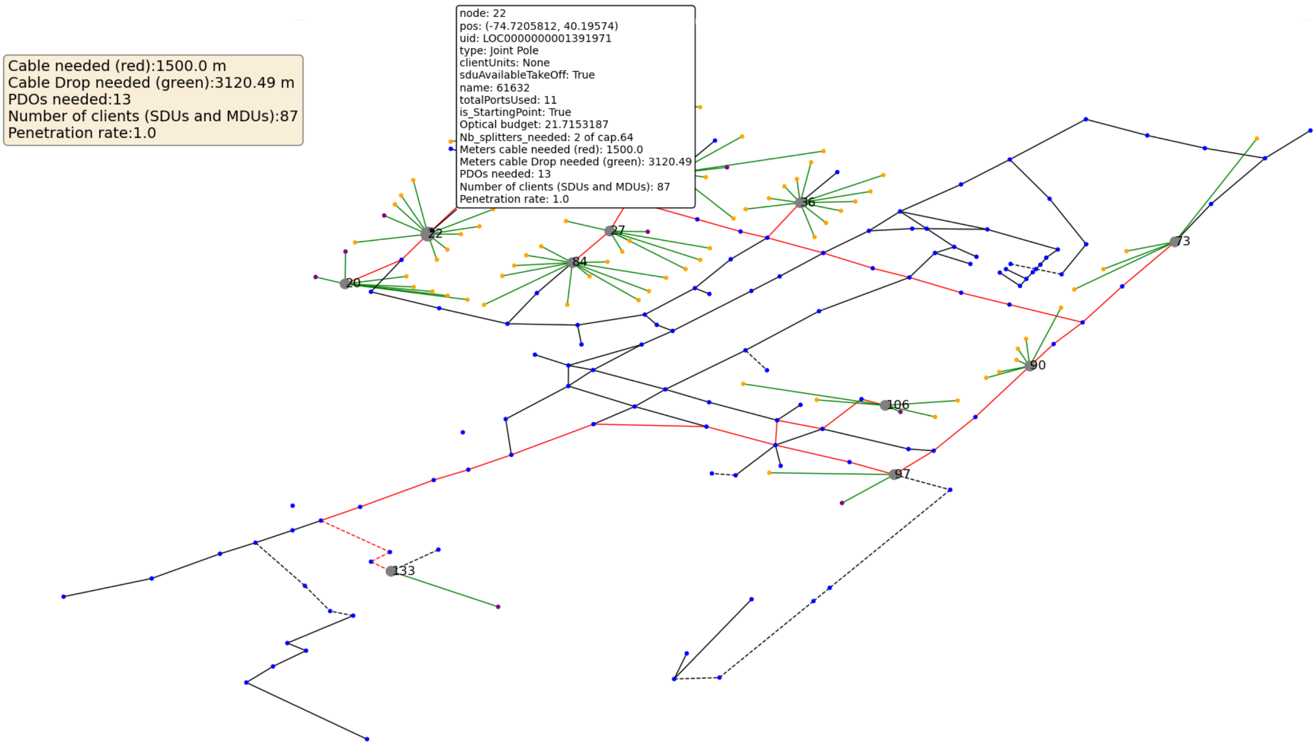
The user of the system can also check the clients homes to see the fiber demand of each node (if it is a building it can be more than one). It also shows some additional data like the distance of the link and the distance of the potential closest link (this is just for reference, the client might not be linked by using the displayed edge on the mouse-hovering box). An example of a client node is shown in Figure A.4.

A.2 Routes Information

The routes can also be analyzed, each route (edge) can be part of two groups: used edge or unused edge (i.e the algorithm does not consider this edge relevant for the final solution where the cable should pass).

An example of a simple unused buried edge is displayed in Figure A.5. Again, the buried edges are shown by dotted edges. We can see that in the mouse-hovering box, the weight of an edge is displayed which corresponds to the cost related to it (distance).

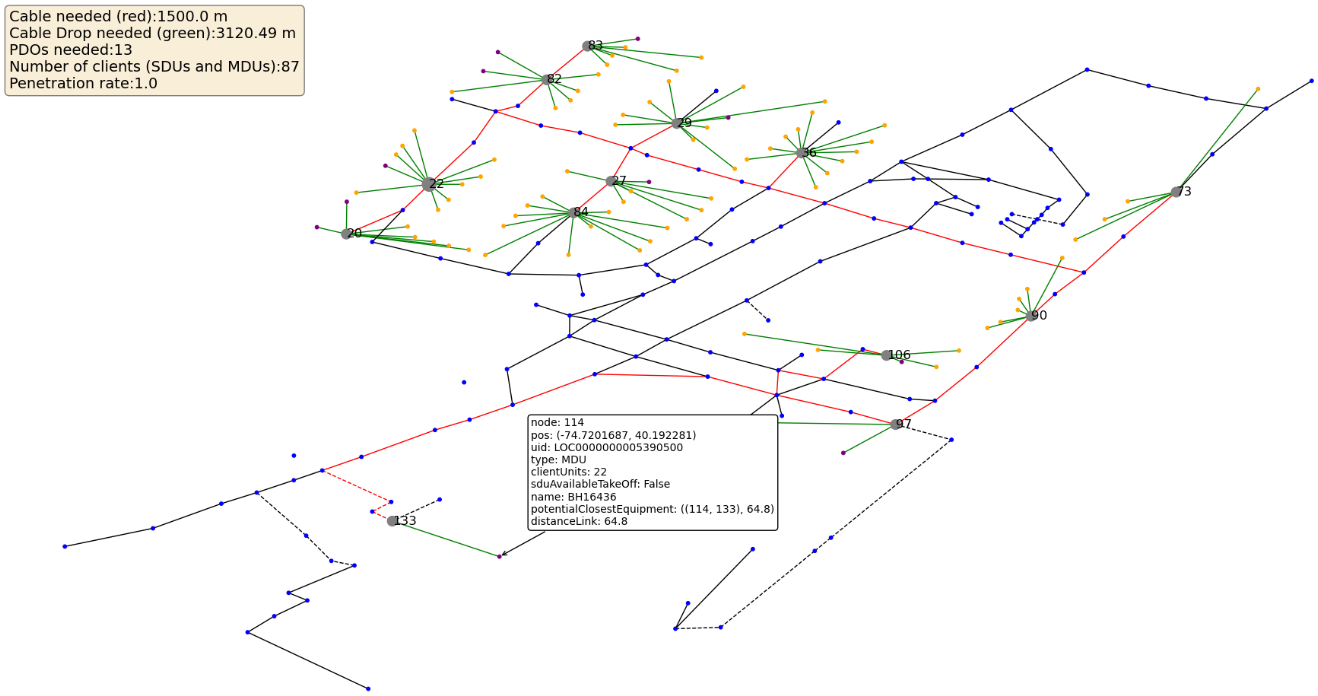
Finally, concerning the edges that are part of the final design of the network generated by the system, there are the new links (drop cables in green) that simply



```
Cable needed (red):1500.0 m
Cable Drop needed (green):3120.49 m
PDOs needed:13
Number of clients (SDUs and MDUs):87
Penetration rate:1.0
```

```
node: 22
pos: (-74.7205812, 40.19574)
uid: LOC0000000001391971
type: Joint Pole
clientUnits: None
sduAvailableTakeOff: True
name: 61632
totalPortsUsed: 11
is_StartingPoint: True
Optical budget: 21.7153187
Nb_splitters_needed: 2 of cap.64
Meters cable needed (red): 1500.0
Meters cable Drop needed (green): 3120.49
PDOs needed: 13
Number of clients (SDUs and MDUs): 87
Penetration rate: 1.0
```

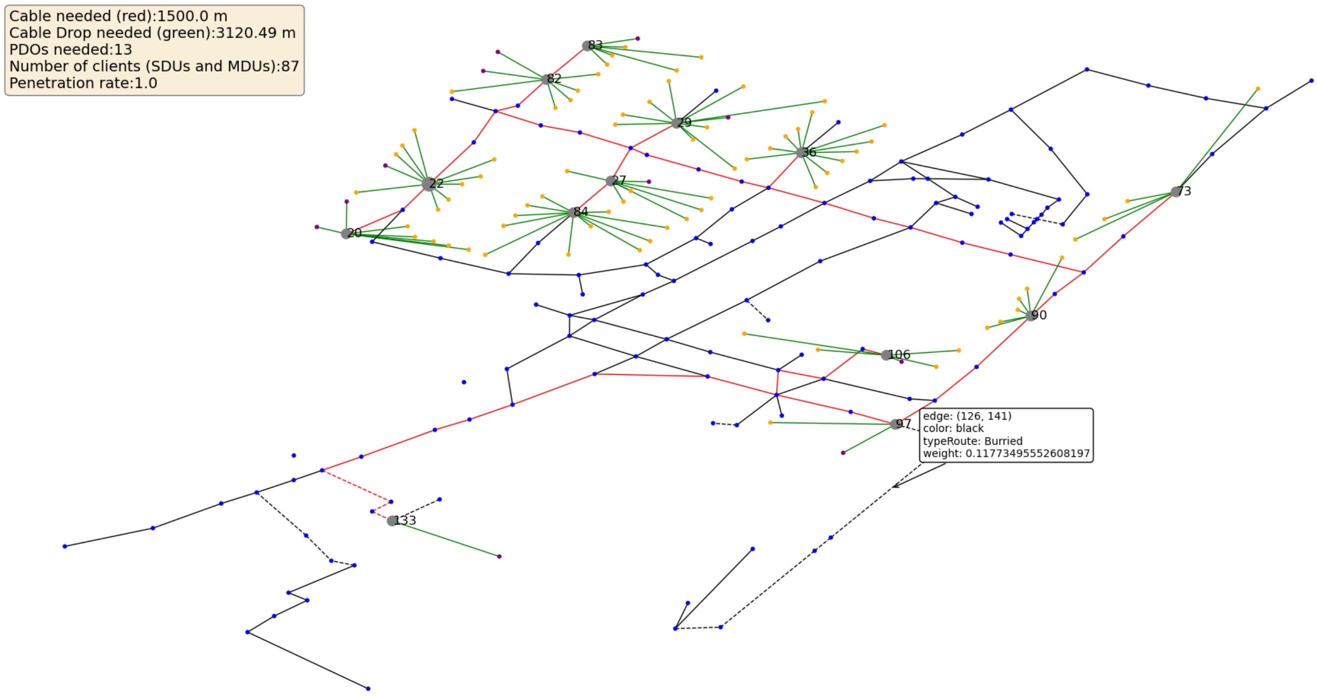
Figure A.3: Mouse-hovering feature - JSO (starting point) node.



```
Cable needed (red):1500.0 m
Cable Drop needed (green):3120.49 m
PDOs needed:13
Number of clients (SDUs and MDUs):87
Penetration rate:1.0
```

```
node: 114
pos: (-74.7201687, 40.192281)
uid: LOC0000000005390500
type: MDU
clientUnits: 22
sduAvailableTakeOff: False
name: BH16436
potentialClosestEquipment: ((114, 133), 64.8)
distanceLink: 64.8
```

Figure A.4: Mouse-hovering feature - Client node.

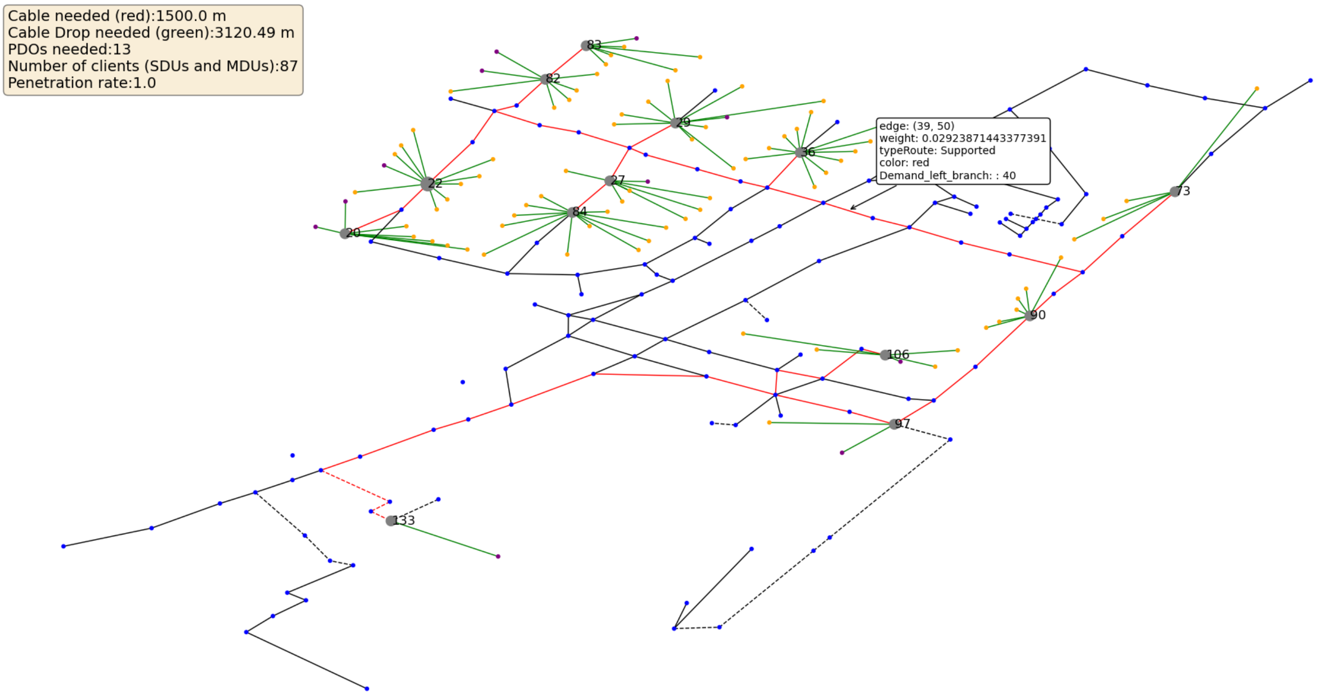


Cable needed (red):1500.0 m
Cable Drop needed (green):3120.49 m
PDOs needed:13
Number of clients (SDUs and MDUs):87
Penetration rate:1.0

edge: (126, 141)
color: black
typeRoute: Burried
weight: 0.11773495552608197

Figure A.5: Mouse-hovering feature - Unused buried edge.

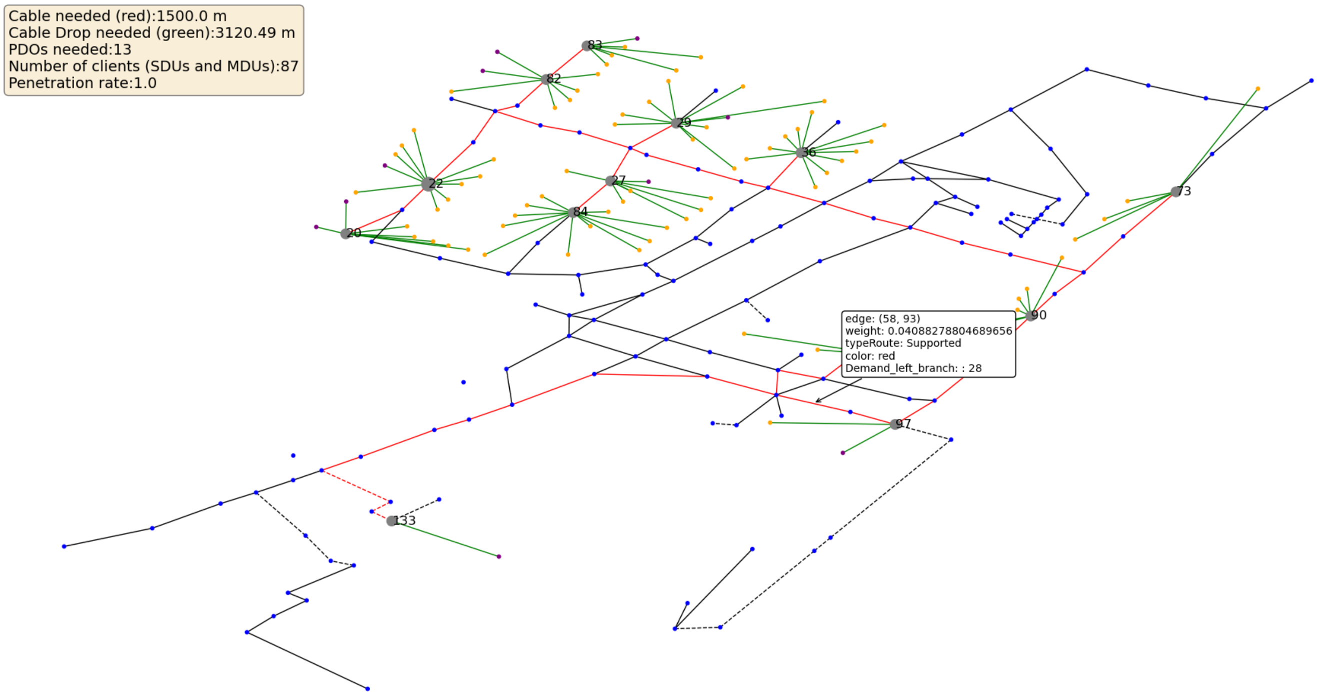
display the weight (distance of the edge) and there is the distribution cables (red ones) that show an additional feature that is the demand left on that cable starting from the JSO (the closest it is to the JSO, the more demand it shows, and as it crosses the homes demands on its way, it reduces). This is simple feature made with a recursive algorithm in order to guide the user of the system in case of larger networks, it shows the demand left on a branch of the network. An example of this feature is provided in Figure A.6, it shows in the mouse-hovering box that the demand of the red cable left is of 40, and as we continue in that cable (in that branch), and as the houses are being covered, this value is reduced to 28. We subtracted the demand required from nodes 73, 90 and 97 (Figure A.7).



Cable needed (red):1500.0 m
Cable Drop needed (green):3120.49 m
PDOs needed:13
Number of clients (SDUs and MDUs):87
Penetration rate:1.0

edge: (39, 50)
weight: 0.02923871443377391
typeRoute: Supported
color: red
Demand_left_branch: : 40

Figure A.6: Mouse-hovering feature - Example of the demand in the distribution cable - before.



Cable needed (red):1500.0 m
Cable Drop needed (green):3120.49 m
PDOs needed:13
Number of clients (SDUs and MDUs):87
Penetration rate:1.0

edge: (58, 93)
weight: 0.04088278804689656
typeRoute: Supported
color: red
Demand_left_branch: : 28

Figure A.7: Mouse-hovering feature - Example of the demand in the distribution cable - after.