Operations Research Society of South Africa

Submitted for publication in ORiON

Operasionele Navorsings-vereniging van Suid-Afrika

Heuristic Approach to the Passive Optical Network with Fibre Duct Sharing Planning Problem

Authors' identities suppressed: Blind refereeing copy

Abstract

Similar to the constrained facility location problem, the passive optical network (PON) planning problem necessitates the search for a subset of deployed facilities (*splitters*) and their allocated demand points (*optical network units*) to minimise the overall deployment cost. In this paper we use a mixed integer linear programming formulation stemming from network flow optimisation to construct a heuristic based on limiting the total number of interconnecting paths when implementing fibre duct sharing. A disintegration heuristic is proposed based on the output of a centroid, density-based and a hybrid clustering algorithm to reduce the time complexity while ensuring close to optimal results. The proposed heuristics are then evaluated using a large real-world dataset, showing favourable performance.

Key words: Integer programming, network flow optimisation, passive optical network, telecommunication network design

1 Introduction

In accordance with the current exponential growth in telecommunication network bandwidth requirements, service providers are opting for optical fibre-based solutions for *last-mile* deployment. With fibre interconnects moving from the backbone to the access networks and the accompanying large capital expenditure, optimisation of these networks have become paramount. The main contender for these fibre-based access networks is the Passive Optical Network (PON) [5]. A single optic fibre cable runs from a Central Office (CO) to a cabinet housing a passive distribution unit called an optic splitter. In the case of Fibre to the Home (FTTH), the optic signal is then distributed to a number of smaller fibres running directly to termination points known as Optical Network Units (ONUs) at customer premises. The PON topology is illustrated in Figure 1. Since these fibres are installed in subterranean ducts, expensive trenches need to be dug all the way from the CO to the customer. The problem is then to minimise the cost of connecting customer premises to the CO by choosing appropriate locations for these splitters and deciding which customers to connect to them.

A number of papers address this problem using both discrete optimisation and metaheuristic techniques. Although Li and Shen [17] incorporated *fibre duct sharing* into their

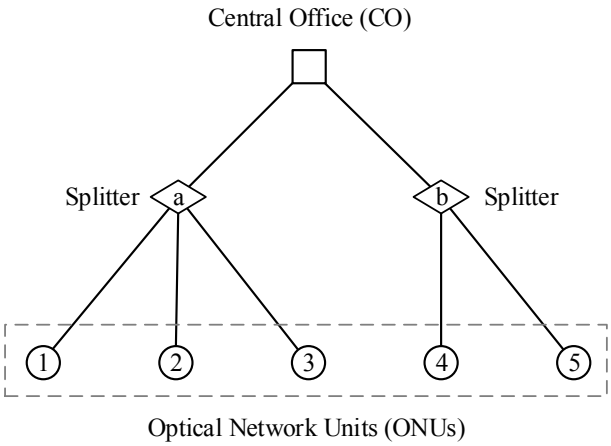

Figure 1: *Schematic layout of the PON topology*

Random Allocation and Reallocation Algorithm (RARA) algorithm through the use of constrained minimum spanning trees, most ignore this fundamental part in the deployment phase of real-world networks. Since it would be impractical and expensive to dig a trench for each individual fibre, a single duct usually contains a number of fibres that share a part of their route to customer premises.

This paper introduces a way of using concepts from network flow optimisation to incorporate fibre duct sharing into a mixed integer formulation of the Passive Optical Network Planning Problem (PONPP).

The rest of the paper is organised as follows: Work related to PONPP are discussed in section 2, before PONPP is defined more formally in section 3. A mixed integer model is given in section 4. The path and disintegration heuristics are given in sections 5 and 6. Results from solving the model are discussed in section 7 before concluding the paper in section 8.

2 Related work

PONPP has been studied for a number of years and authors typically take one of two approaches; one based on exact models combined with valid inequalities or heuristics and another based on meta-heuristics.

Khan [12] developed a greedy algorithm, which is based on the transformation of a population density graph to one proportional to some Population Density Function. Minimum distance allocation is then done on the transformed graph iteratively. Mitcsenkov *et al.* [22] provides a Capital Expenditure (CAPEX) model for PONPP, which they solve for very large problem instances using a tree-based heuristic algorithm known as the Branch Contracting Algorithm (BCA). The same authors provide a general methodology to design broadband infrastructure in [21]. Li and Shen did a comprehensive study on greenfield PON planning in [17], introducing a heuristic called RARA which sequentially refines an Mixed Integer Linear Programming (MILP) model solution through the use of simulated

annealing. The model in question also takes into account the PON specifications in terms of network reach and differential distance. The survivable PONPP has been studied thoroughly by Kantarci *et al.* [10], introducing three MILP models based on different service availability solved using standard branch-and-bound before providing a greedy planning heuristic in [11]. The multi-level PONPP is studied by Kim *et al.* [13], providing bounds through linear relaxation as well as a two stage incremental improvement heuristic. Ouali and Poon [23] developed a basic PONPP model and solved small test instances using branch-and-bound.

A wide range of meta-heuristics have also been employed to solve PONPP, with genetic algorithms (GA) being the most popular. Poon *et al.* [24] used a GA to identify splitter locations, with a clustering heuristic to form PONs. Lv and Chen [20] and Kokangul and Ari [14] address the multi-level PONPP using a GA, while Villalba *et al.* [29] studied modified versions of PONPP with ring and bus topologies. Ahmad *et al.* [1] uses a GA to solve PONPP, but optimises for minimum power consumption instead of minimum deployment cost. Lakic and Hajduczenia [15] studied PONPP with the inclusion of non-traversable obstacles through the use of convex hull mapping, which is then again solved using evolutionary computing techniques. Finally, Xiong *et al.* [28] provides an algorithm that is less vulnerable to local optima using ant colony optimisation.

PONPP is conceptually similar to the *Connected Facility Location Problem* (ConFL), first introduced by Gupta *et al.* [9] and studied by a number of authors since. Starting from ConFL, including trenching cost as an additional fixed cost assigned to every edge and substituting facilities and demand points with splitters and ONUs respectively, we arrive at the PONPP. Swamy *et al.* [25] provided a primal-dual 8.55-approximation algorithm for ConFL while Gollowitzer and Ljubić [8] did a polyhedral and computational study on a large number of formulations. Arulselvan *et al.* [2] introduced a multi-period incremental formulation of the ConFL along with cover and cut-set inequalities solved using branch-and-cut. The hop constrained ConFL was illustrated by Ljubić and Gollowitzer [18] using a cut formulation on layered graph approach while Leitner *et al.* [16] studied the two-architecture ConFL and provided a cut formulation Integer Linear Program (ILP). Finally, Bley *et al.* [4] studied the survivable constrained ConFL problem and solved a number of small instances using Bender's decomposition in a branch-and-cut framework.

3 Problem definition

Assume an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is given with edge costs $c_e \geq 0, e \in \mathcal{E}$, a disjoint partition $\mathcal{J} = \{\mathrm{S}, \mathrm{U}\}$ with $\mathrm{S} \subset \mathcal{V}$ the possible splitter locations and $\mathrm{U} \subset \mathcal{V}$ the set of fixed ONU locations. The CO location is denoted by $c \in \mathcal{V} \backslash \mathcal{J}$. Assume fixed deployment costs $c_{\mathrm{co}} \geq 0$, $c_{\mathrm{s}} \geq 0$ and $c_{\mathrm{onu}} \geq 0$ for the CO, splitters and ONUs respectively and a splitter capacity of κ. Furthermore, assume c_t and c_f is a cost per unit length of trenching and fibre respectively.

In the case of PONPP, the objective is to find a subset of open facilities F, with every element in U assigned to a single facility $f \in \mathrm{F}$ by distributing into $\mathrm{U}_f \subseteq \mathrm{U}$, all vertices in U_f connected via a Steiner tree T_f rooted in f and all vertices in F connected via a

Steiner tree T rooted in c so as to *minimise* the overall deployment cost

$$|\mathrm{F}|c_\mathrm{s} + \sum_{f\in\mathrm{F}} c_f \ell_f^c + \sum_{f\in\mathrm{F}} \sum_{j\in\mathrm{U}_f} c_f \ell_j^f + \sum_{e\in\mathrm{E_U}} c_t \ell_e \tag{1}$$

where ℓ_u^v is the length of the shortest path between vertices u and v in the Steiner tree with v as root and the set $\mathrm{E_U}$ contains every edge used by a Steiner tree. ℓ_e is the length of the edge e. The first term describes the splitter cost while the second and third terms describe the fibre cost between CO and splitter, and splitter and Optical Network Unit (ONU) respectively. The final term is the total trench cost.

Typically, the problem is formulated using directed arc based flows. This approach is tractable for small data sets and for medium sized data sets if combined with strong valid inequalities. These valid inequalities are the focus of many papers [2, 4, 16, 18] and usually allows fast convergence. When moving to larger data sets however, the arc based formulation suffers from memory problems due to the large number of constraints and becomes intractable.

The above formulation can be transformed into a path-based formulation. Using constructed paths, this formulation automatically ensures that flow conservation holds and therefore most of the constraints can be removed, resulting in a compact formulation. PONPP can then be redefined using paths.

Define a commodity pair $k \in \mathrm{K}$. The set K consists of all possible pairs of the CO and splitters as well as all possible pairs of splitters and ONUs. For each commodity pair $k = \{i, j\} \in \mathrm{K}$, define a set $\mathrm{P}(k) \subseteq \mathrm{P}$ of all non-cyclic paths between $i \in \mathcal{V}$ and $j \in \mathcal{V}$. Next, define a set $\mathrm{E} \subseteq \mathcal{E}$ of all edges traversed in paths $p \in \mathrm{P}$ and a set $\mathrm{P}(e) \subseteq \mathrm{P}$ containing all paths that traverses edge $e \in \mathrm{E}$. Conversely, $\mathrm{E}(p) \subseteq \mathrm{E}$ is the set of all edges contained within path $p \in \mathrm{P}$.

Two additional constraints are applicable to PONs: maximum and differential network reach. The total length of fibre connecting the CO with an ONU $j \in \mathrm{U}$ through a splitter $i \in \mathrm{S}$, *i.e.* the network reach, may not exceed a threshold $\ell_{\max}^{\mathrm{total}}$ due to optic loss. To avoid synchronisation issues between ONUs, the difference between the maximum and minimum network reach for a splitter $i \in \mathrm{S}$ may not exceed $\ell_{\max}^{\mathrm{diff}}$ [5].

With paths representing fibre cables and edges representing trenches, PONPP then becomes the search for a subset of deployed splitters such that

- each ONU connects to one and only one splitter via a single path,
- each splitter connects to the CO via a single path,
- a maximum of κ ONUs can connect to a single splitter,
- the maximum and differential network reach constraints are satisfied and
- the sum of the deployment, path and edge costs are minimised.

The non-cyclic paths in set P would typically be calculated by forming Steiner trees rooted at c and $i \in \mathrm{S}$ with Steiner nodes $i \in \mathrm{S}$ and $j \in \mathrm{U}$ respectively. Another approach would be to start with a reduced subset of paths and use column generation to generate additional columns (paths) until no columns exist with negative reduced costs. In our approach, the paths will be generated once at the onset using a heuristic specific to PONPP. Hence, the set P will be treated as a parameter in the formulation.

4 Mixed Integer Formulation

From the above problem, a MILP model can now be formulated. Let y_p indicate the usage of the path $p \in \mathrm{P}$ and x_e the usage of edge $e \in \mathrm{E}$. Let ψ_i indicate the deployment of splitter $i \in \mathrm{S}$. $k_{ij}^{\mathrm{so}} \in \mathrm{K}$ denotes the commodity pair of splitter $i \in \mathrm{S}$ and ONU $j \in \mathrm{U}$ while $k_i^{\mathrm{cs}} \in \mathrm{K}$ denotes the commodity pair of the CO and splitter $i \in \mathrm{S}$. Parameter ℓ_p denotes the total length of path $p \in \mathrm{P}$ while the variables $\ell_i^{\min}$ and $\ell_i^{\max}$ denote the minimum and maximum network reach for splitter $i \in \mathrm{S}$ respectively. Let ℓ_e be the length of edge $e \in \mathrm{E}$. As done by Li and Shen [17], the introduction of a binary *if-then* variable $d_{ij}, i \in \mathrm{S}, j \in \mathrm{U}$ and a large value, Δ, allows the formulation of PONPP as follows:

$$(PONPP) \quad \min \quad c_{\mathrm{co}} + \sum_{i \in \mathrm{S}} \psi_i c_{\mathrm{s}} + |\mathrm{U}| c_{\mathrm{o}} + \quad \sum_{e \in \mathrm{E}} c_{\mathrm{t}} \ell_e x_e + \sum_{p \in \mathrm{P}} c_{\mathrm{f}} \ell_p y_p \tag{2}$$

$$\text{s.t.} \quad \sum_{i \in \mathrm{S}} \sum_{p \in \mathrm{P}(k_{ij}^{\mathrm{so}})} y_p = 1, \qquad \forall j \in \mathrm{U} \tag{3}$$

$$\sum_{j \in \mathrm{U}} \sum_{p \in \mathrm{P}(k_{ij}^{\mathrm{so}})} y_p \leq \kappa \psi_i, \qquad \forall i \in \mathrm{S} \tag{4}$$

$$\sum_{p \in \mathrm{P}(e)} y_p \leq \Delta x_e, \qquad \forall e \in \mathrm{E} \tag{5}$$

$$\ell_i^{\min} - \Big(\sum_{p \in \mathrm{P}(k_i^{\mathrm{cs}})} y_p \ell_p + \sum_{p \in \mathrm{P}(k_{ij}^{\mathrm{so}})} y_p \ell_p \Big) \leq \Delta d_{ij}, \qquad \forall i \in \mathrm{S}, \forall j \in \mathrm{U} \tag{6}$$

$$\Big(\sum_{p \in \mathrm{P}(k_i^{\mathrm{cs}})} y_p \ell_p + \sum_{p \in \mathrm{P}(k_{ij}^{\mathrm{so}})} y_p \ell_p \Big) - \ell_i^{\max} \leq \Delta d_{ij}, \qquad \forall i \in \mathrm{S}, \forall j \in \mathrm{U} \tag{7}$$

$$\sum_{p \in \mathrm{P}(k_{ij}^{\mathrm{so}})} y_p \leq \Delta(1 - d_{ij}), \quad \forall i \in \mathrm{S}, \qquad \forall j \in \mathrm{U} \tag{8}$$

$$\ell_i^{\max} \leq \ell_{\max}^{\mathrm{total}}, \qquad \forall i \in \mathrm{S} \tag{9}$$

$$\ell_i^{\max} - \ell_i^{\min} \leq \ell_{\max}^{\mathrm{diff}}, \qquad \forall i \in \mathrm{S} \tag{10}$$

$$y_p \in \{0, 1\}, \qquad \forall p \in \mathrm{P} \tag{11}$$

$$x_e \in \{0, 1\}, \qquad \forall e \in \mathrm{E} \tag{12}$$

$$\psi_i \in \{0, 1\}, \qquad \forall i \in \mathrm{S} \tag{13}$$

$$d_{ij} \in \{0, 1\}, \qquad \forall i \in \mathrm{S}, j \in \mathrm{U} \tag{14}$$

Contraint set (3) ensures that each ONU connects to a splitter via a path while (4) limits

the maximum number of ONUs per splitter as well as sets the splitter usage variable ψ_i. The inequalities in (5) ensures that all edges of used paths are marked used as well. To ensure numerical stability, the values of Δ are set as small as possible, in this case $\Delta = |\mathrm{P}(e)|$. Constraints (6)) and (7) sets the minimum and maximum network reach parameters for each splitter while equation (8) activates the previous two constraints only when paths between commodities are used. Finally, the inequalities (9) and (10) implement the global PON fibre distance constraints.

The constraint set (5) can be substituted with $0 \leq y_p \leq x_e, \forall p \in \mathrm{P}, \forall e \in \mathrm{E}(p)$ to strengthen the model relaxation, but this increases the number of constraints of the model and therefore the memory required to solve dramatically. For this paper, (5) will be left as is to ensure model tractability for large instances.

Due to space limitations, the model in question incorporates only the fundamental constraints inherent to PONPP. For a more complete model that includes refinements such as economies of scale, network coverage and non-symmetrical fibre costs, refer to [26].

5 Paths heuristic

It is evident that the set P will include an infeasible number of paths for large graphs. Consider now how the paths are generated as a preprocessing step for each commodity $k \in \mathrm{K}$. As a first step, $\mathrm{P}(k)$ contains the shortest path between commodity pair k. In this formulation of PONPP, the aim of generating additional longer paths to include in $\mathrm{P}(k)$ is to increase the possibility of edges being shared between paths of different commodities. However, it should be evident that during the generation procedure, a point exists where generating a longer path will not result in a decrease in the objective function value. This point is reached when the total additional fibre cost exceeds the cost saved if an additional trench segment is shared. As this path generation process is independent of the fixed costs $c_{\mathrm{s}}, c_{\mathrm{onu}}$ and c_{co}, the only relevant costs involved are c_f and c_t. This reasoning is stated formally in proposition 1.

Proposition 1 *Let $p^* \in \mathrm{P}(k)$ be the shortest non-cyclic path between commodity pair $k \in \mathrm{K}$ with length ℓ_{p^*}. For PONPP as defined by (2)–(14), the set of paths $\mathrm{Q}(k) \subset \mathrm{P}(k)$ will not be found in the minimal solution, where $\ell_q > (1 + c_t/c_f)\ell_{p^*}, \forall q \in \mathrm{Q}(k)$.*

Proof: The cost of a shortest path $p^* \in \mathrm{P}(k)$ with no fibre duct sharing is given by its trenching and fibre components, *i.e.* $c_{p^*} = \sum_{e \in \mathrm{E}(p^*)} c_t \ell_e + c_f \ell_{p^*}$. It follows that any fibre duct sharing will result in a longer path p^+ with length ℓ_{p^+}. Let $\mathrm{E_s} \subseteq \mathrm{E}(p^*)$ be the edges path p^+ shares with other paths. Therefore, the total cost of path p^+ can be given by $c_{p^+} = \sum_{e \in \mathrm{E}(p^*)} c_t \ell_e - \sum_{e \in \mathrm{E_s}} c_t \ell_e + c_f \ell_{p^+}$. It is evident that if $c_{p^+} > c_{p^*}$, path p^+ will not be used in the minimal solution. Substituting and simplifying:

$$c_f \ell_{p^+} - \sum_{e \in \mathrm{E_s}} c_t \ell_e > c_f \ell_{p^*} \tag{15}$$

The maximum sharing that can occur is if all edges are shared, *i.e.* $\mathrm{E_s} = \mathrm{E}(p^*)$. Furthermore, from the definition of a path, it is evident that $\ell_{p^*} = \sum_{e \in \mathrm{E}(p^*)} \ell_e$. Substituting into equation (15), we arrive at $c_f \ell_{p^+} - c_t \ell_{p^*} > c_f \ell_{p^*}$, which can be rearranged to $\ell_{p^+} > (1 + c_t/c_f)\ell_{p^*}$. □

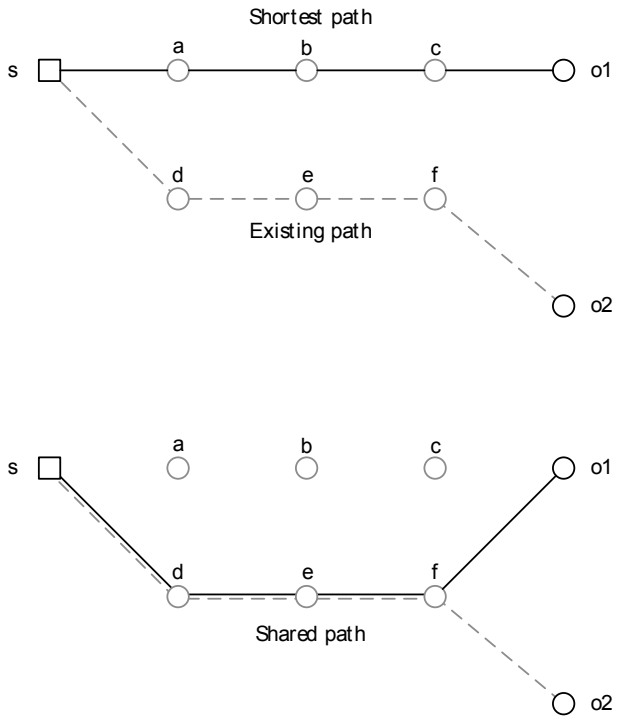

Figure 2: *Fibre duct sharing opportunity when allowing for longer paths.*

Figure 2 illustrates the significance of Preposition 1 by means of an example. The most edges that can be shared by a path between s and o1 is to share edges (s,d), (d,e) and (e,f) with the existing path between s and o2. If however the cost saved by sharing those edges exceed the extra cost incurred to use a longer path, the path will not be selected in the minimal solution. In fact, since no two commodities can have both the same source and destination nodes, proposition 1 tends to be conservative.

Using this proposition, all paths that are c_t/c_f times longer than the shortest path p^* will not be calculated since they will not be used in the minimal solution. In practice however, civil restrictions ensure even greater diminishing returns when deviating from the shortest path. This is due to the fact that trenches are made alongside roads which are usually the only access to customer premises. In these cases, the only opportunities for fibre duct sharing exist at road junctions where fibres can be routed together on one side of the road if possible.

For smaller data sets, the number of paths generated for $\mathrm{P}(k)$ can be adjusted through the use of a k shortest simple path algorithm such as Yen's algorithm [6,30]. These algorithms typically start from a shortest path and sequentially add additional edge segments with least weight to form longer paths. Unfortunately, they usually have time complexities

that increase linearly with both k and $|\mathcal{V}|$, adding substantial preprocessing time as more paths are generated. It is therefore necessary to minimise the paths generated to increase computational performance.

In the case of larger data sets, we will also determine the performance when including only the shortest path ($k = 1$) in the model, effectively using opportunistic fibre duct sharing when shortest paths take a similar route. With $k = 1$, the much faster Dijkstra's algorithm can be used, increasing preprocessing performance. This special case, where $k = 1$, will henceforth be called the *shortest path heuristic* (SPATH).

6 Disintegration heuristic

Since real-world PON data are usually grouped into interconnected neighbourhoods, a disintegration of the input data into clusters should give good computational performance while staying close to the global minimal solution. As the central office is global to all clusters only splitter and ONU nodes are clustered, or the set $\mathrm{D} = \mathrm{U} \cup \mathrm{S}$. A number of methods exist to cluster the PON data sets, including centroid, density and hybrid clustering. Each method is implemented and compared with respect to efficacy of computational effort distribution as well as numerical performance.

6.1 Centroid clustering

Firstly, to test centroid clustering, the common k means algorithm [19] is used. This simple algorithm minimises intra-cluster distances by minimising the sum of Euclidian distances between each point assigned to cluster i, and the cluster mean μ_i. Generically, the technique can be stated as follows:

$$\min_{\mathrm{D}} \sum_{i=1}^{k} \sum_{x_j \in \mathrm{L}_i} ||x_j - \mu_i||^2 \tag{16}$$

The k means algorithm provides the k output sets $\mathrm{L}_1, \mathrm{L}_2, \ldots, \mathrm{L}_k$, where $\mathrm{D} = \mathrm{L}_1 \cup \mathrm{L}_2 \cup \cdots \cup \mathrm{L}_k$. The algorithm is promising since it provides roughly equi-sized clusters which is useful for effective division of computational complexity.

6.2 Density clustering

The Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm was introduced in 1996 by Ester *et al.* [7] and is by far the most commonly cited density-based clustering algorithm in literature. The algorithm incrementally creates clusters by adding points within a distance ϵ from any point already in the cluster. Another parameter MinPts determines the minimum number of points a cluster should consist of, discarding all isolated points as noise. In our implementation, this parameter is set to zero to avoid noise classification. The algorithm provides K output sets for which the following holds:

$$\{\mathrm{L}_i : \ |\mathrm{L}_i| \geq \mathrm{MinPts} \ | \ \exists \ (x_j, x_k)_{x \neq j} \in \mathrm{L}_i : \sqrt{||x_j - x_k||^2} \leq \epsilon\} \qquad \forall i = 1, \ldots, K \quad (17)$$

Since density estimates are used to cluster the data, these techniques are good at capturing neighbourhoods of roughly equal density and should therefore ensure that equi-dense clusters are not bifurcated, ensuring improved numerical performance.

6.3 Hybrid clustering

In 1982, Wong *et al.* [27] introduced a hybrid clustering approach that combines the computational performance of k means with density-based clustering advantages. Given a required number of clusters K, the algorithm executes two steps; a preliminary k means clustering with $k \gg K$, followed by an iterative step that analyses and combines the output clusters at each iteration according to a density measure function φ until K clusters remain. k is usually proportional to the number of observations N and can be approximated as $k \approx 7(\frac{N}{log(N)})^{\frac{1}{3}}$.

In the algorithm, two preliminary clusters, L_u and L_v, are considered adjacent if the midpoint between the centroids μ_u and μ_v is closer to either μ_u or μ_v than any other cluster mean using Euclidian distance. Define the intra-cluster sum of squares d_i:

$$d_i = \sum_{j \in \mathrm{L}_i} ||x_j - \mu_i||^2 \quad (18)$$

Then the density measure function φ in two dimensions is given as:

$$\varphi(\mathrm{L}_u, \mathrm{L}_v) = \begin{cases} \dfrac{d_u + d_v + \dfrac{1}{4}(|\mathrm{L}_u| + |\mathrm{L}_v|)||\mu_u - \mu_v||^2}{(|\mathrm{L}_u| + |\mathrm{L}_v|)^2} & \text{if } \mathrm{L}_u \text{ is adjacent to } \mathrm{L}_v \\ \infty & \text{otherwise} \end{cases} \quad (19)$$

At each iteration of the algorithm, the cluster pair for which (19) is a minimum is combined until the required number of clusters remain.

6.4 Cluster post-processing

To ensure a feasible solution exists for each cluster, *valid clusters* are built from the output of each of the clustering algorithms. These clusters, defined in definition 2, contain both splitters and ONUs and have enough splitters to serve all ONUs contained within the cluster. The minimum split ratio is defined as the capacity of a splitter, κ.

Definition 2 (Valid cluster) *Given a set of splitters* $\mathrm{S} \neq \varnothing$*, a set of ONUs* $\mathrm{U} \neq \varnothing$ *and a minimum split ratio* κ*, a set* $\mathrm{L} \subseteq \mathrm{U} \cup \mathrm{S}, \mathrm{L} \neq \varnothing$ *is said to be a* valid cluster *iif*

$$\kappa|\mathrm{L} \cap \mathrm{S}| \geq |\mathrm{L} \cap \mathrm{U}|. \quad (20)$$

A number of possibilities exist when encountering *invalid* clusters, the most basic of which is to simply combine the invalid cluster with another cluster until all clusters are valid. A more sophisticated approach is to combine an invalid cluster with its nearest neighbour in terms of inter-cluster centroid distance, *i.e.* at every iteration, an invalid cluster L_i is combined with its nearest neighbour L^i_{NN} as defined in (21).

$$\mathrm{L}^i_{NN} = \min_{\mathrm{L}_j} ||\mu_i - \mu_j||^2 \tag{21}$$

This process continues until all clusters are valid or only a single cluster remain. This will ensure that if the original set D is valid, which needs to be true for a feasible solution to be found to the original problem, the resulting clusters will also be valid.

It should be mentioned that the efficacy of the clustering process depends not only on the underlying structure of the data, but also on the splitter capacity κ. When moving to cluster sizes smaller than the splitter capacity, the cluster bounds will intersect the splitter reach, resulting in potentially inflated deployment costs. Therefore the cluster sizes should ideally be some factor $M > 1$ larger than the splitter capacity κ. The clustering process is also influenced by the amount of *excess* capacity available. If the original dataset contains *just* enough splitter capacity to connect all ONUs, the probability of a cluster being invalid is high, decreasing the efficacy of the process. Therefore, ideally, $\kappa|\mathrm{S}| \gg |\mathrm{U}|$ should hold for the dataset to ensure efficient operation.

7 Computational results

The model given in (2)–(14) is implemented using `C++` and IBM ILOG CPLEX Concert extensions. It is then solved on a quad core Intel Core i7 at 2.67 GHz with 16 GiB main memory running Windows. Deterministic parallel processing is enabled to increase computational performance.

7.1 Path heuristic

Firstly, the path heuristic efficiency can be tested by varying the number of shortest paths between 1 (opportunistic fibre duct sharing) and 100. This is done using three small data sets containing less than 160 ONUs each, with parameters as in Tablet 1. These data sets were constructed using small subsets of real-world data sets generously provided by *atesio GmbH* [3]. Table 2 contains the results, with P^k the number of shortest paths, t_s the time to solve and MEM^p the peak memory consumption in MiB during the test. The GAP_b % is calculated according to the best known integer bound for each of the data sets, which were calculated using a standard arc flow formulation running for 3 hours each. The best bound optimality gaps are given in the GAP_ℓ % column of Tablet 1, with $\mathrm{SR}_{\mathrm{avg}}$ showing the average split ratio available. In this case, the relative differences is important to determine the typical influence fibre duct sharing can have on the objective function value, and how the addition of extra paths affect both the computational effort and the deployment cost.

Table 1: *Small dataset parameters.*

Instance	|U|	|S|	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	SR_{avg}	LB_b	GAP_ℓ (%)
city_154	154	11	640	736	14	3,870.14	3.12
sub_112	112	9	449	490	12.44	2,957.58	**0.00**
med_136	136	8	415	447	17	4,273.30	0.82

Table 2: *Numerical and computational results for path heuristic test.*

Instance	P^k	OBJ (1000 R)	GAP_b (%)	t_s (s)	MEM_{peak}
city_154	1	3,979.43	2.75	0.87	93
	10	3,898.78	0.73	12.94	252
	20	3,884.49	0.37	67.97	676
	40	3,880.83	0.28	291.28	1,721
	100	3,872.05	0.05	1,991.12	3,780
sub_112	1	3,016.02	1.94	0.93	85
	10	2,957.58	**0.00**	7.36	178
	20	2,957.58	**0.00**	19.31	303
	40	2,957.58	**0.00**	44.60	522
	100	2,957.58	**0.00**	65.56	1,158
med_136	1	4,316.12	0.99	0.96	95
	10	4,277.89	0.11	13.60	360
	20	4,273.30	**0.00**	30.92	486
	40	4,273.30	**0.00**	83.27	1,039
	100	4,273.30	**0.00**	193.26	1,509

The results support the notion of diminishing returns as more shortest paths are introduced into the model, with deployment cost savings only improving by 2 % on average when moving from $k = 1$ to $k = 100$ and with most of the savings attained when moving from the shortest path to $k = 10$. While the typical memory requirements increase linearly with k, the computational effort required increases exponentially. Therefore, it might be deemed worthwhile to use a lower number of paths that will result in good bounds without requiring an infeasible time to solve, especially when solving large instances. It should be noted that almost all of the instances were solved in a fraction of the time with comparable or equivalent numerical performance to the best calculated bound. In the case of sub_112, optimality was attained in just a few seconds compared to the 1.5 hour computation time of the arc flow formulation.

7.2 Disintegration heuristic

The three clustering methods are compared using a real-world GIS-mapped dataset containing 6,698 nodes and 7,660 edges, called *CityNet* [3]. This dataset contains a total of 2,190 splitters and ONUs to be clustered. Cluster metrics are then calculated, including the average intra-cluster distance d^i_{avg}, average cluster size $|\text{L}|_{\text{avg}}$, cluster size standard deviation $|\text{L}|_{\text{dev}}$, maximum cluster size $|\text{L}|_{\text{dev}}$ and number of valid clusters $|L_i|_{\text{valid}}$. Table 3 shows the clustering results, as well as the clustering processing time t_c.

Table 3: *Clustering algorithm comparison.*

Algorithm		t_c (ms)	d^i_{avg}	$\lvert\text{L}\rvert_{\text{avg}}$	$\lvert\text{L}\rvert_{\text{dev}}$	$\lvert\text{L}\rvert_{\text{max}}$	$\lvert\text{L}_i\rvert_{\text{valid}}$
k means	$k = 10$	7	174.05	219.00	60.39	358	10
	$k = 20$	14	121.87	109.50	59.59	257	20
	$k = 30$	17	98.31	73.00	50.33	240	30
	$k = 50$	25	75.37	43.80	42.85	237	50
DBSCAN	$\epsilon = 80$	9	112.05	182.50	412.33	1,395	12
	$\epsilon = 50$	37	72.78	54.75	196.07	1,150	40
	$\epsilon = 30$	93	43.06	19.38	56.70	404	113
	$\epsilon = 20$	251	31.41	10.27	9.55	81	216
Hybrid	$K = 10$	8	165.57	219.00	487.32	1,593	10
	$K = 20$	14	122.92	109.50	256.28	1,175	20
	$K = 30$	15	103.35	73.00	129.32	697	30
	$K = 50$	22	77.55	47.61	44.43	240	46

Looking at the clustering results, the strengths of each method is apparent. Firstly, as expected, the k means algorithm provides equi-sized clusters, with the lowest maximum cluster size and low cluster size deviation. DBSCAN delivers very low average intra-cluster distances, illustrating its efficacy in correctly clustering dense regions. Unfortunately, the maximum cluster sizes and cluster size deviations of DBSCAN is large, which will negatively impact computational effort distribution. Wong's hybrid clustering has slightly better average intra-cluster distances in the $K = 10$ scenario in comparison to k means with the same number of clusters, showing some improvement with the introduction of density-based clustering. Like DBSCAN however, the hybrid method suffers from high cluster size deviations and large maximum cluster sizes. Finally, k means provided no invalid clusters in any of the cases, whereas the hybrid method started deviating at $K = 50$.

Even with this relatively large dataset, all clustering methods performed amicably, completing the clustering in a few milliseconds. Overall, it seems as though the standard k means algorithm will provide the best computational effort distribution and the lowest overall computation time due to its low maximum cluster size value.

Next, *CityNet* is solved through the use of a modified version of the *Branch Contracting Algorithm* (BCA) heuristic proposed in [21]. BCA is chosen since this heuristic explicitly includes elements of fibre duct sharing through a tree-based clustering method. The authors claim performance of 10–15 % deviation from optimal with very fast computational speed. Since the original article does not specify values for Q, a grouping factor, the algorithm is run with all practical values for Q, *i.e.* $0 < Q \leq \kappa$, taking the minimum objective value. Next, since BCA is randomly initialised, it is run 20 times for each Q value, again noting the minimum objective value. This ensures a fair comparison with the proposed path and disintegration heuristics.

It should be noted that the last step of BCA requires a heuristic Steiner tree implementation to connect all splitters to the central office. Since the details of this heuristic are not clear from the original article, the algorithm is modified to connect splitters through shortest path routes to the central office, sharing fibres as possible. This produces an upper

bound on the true BCA objective value. BCA without *any* connecting fibres between the central office and splitters (BCANoF) is also tested to provide a lower bound and ensure the modification does not produce biased results.

The model was solved incorporating the shortest path heuristic (SPATH) with no disintegration and a 1 hour time limit. Then, the clustering scenarios defined in Tablet 3 above were solved using the shortest path heuristic. k means results are denoted with k10 to k50, DBSCAN with DB20 to DB80 and the hybrid clustering results with H10 to H50. The optimality gap (GAP_b) is specified in terms of the best upper bound found among all the instances. For the instance that produced the best bound the normal branch and bound optimality gap is given. Peak memory usage in MiB during the tests is given by MEM_{peak}. If the total time to solve, t_{solve}, exceeds the time limit, a $>$ sign is placed next to the value. Finally, the number of splitters deployed is given in the SPs column.

Table 4: *Numerical and computational results for the* CityNet *dataset.*

Instance	t_{solve} (s)	OBJ (mil R)	MEM_{peak}	GAP_b (%)	SPs
SPATH	>3,600	**53.430**	12,837	10.32*	78
BCA (UB)	2.11	76.676	5,627	43.51	68
BCANoF (LB)	1.99	60.869	5,164	13.92	131
k10	212.77	55.111	508	3.15	79
k20	70.45	56.249	321	5.28	81
k30	50.68	56.751	321	6.21	79
k50	31.76	58.819	**319**	10.09	89
DB80	>3,600	54.526	8,520	**2.05**	83
DB50	>3,600	58.111	4,962	8.76	100
DB30	1,250	66.856	420	25.13	149
DB20	8.06	83.442	445	56.17	219
H10	>3,600	55.217	12,613	3.34	82
H20	>3,600	56.905	7,549	6.50	82
H30	>3,600	57.935	3,656	8.43	86
H50	37.23	59.081	298	10.58	88

* Optimality gap between best upper integer and best lower relaxation bound

From the numerical results in Tablet 4 it is clear that the BCA heuristic is much faster than the clustering methods, but produces an increase of up to 44 % in objective value. It should however be noted that the time to solve for BCA is specified as the time to complete one iteration. To get the minimum values for BCA, the algorithm was run for approximately 40 minutes with the various parameters. Another observation is the variance in BCA due to its random initialisation. Given a fixed Q value, the heuristic gave solutions with objective values varying by up to 12 %. Therefore it is critical that the algorithm is run a number of times to ensure a good solution is produced.

Computation times for the clustering methods are consistent with the maximum cluster size values obtained in Tablet 3, with maximum values of over ± 700 resulting in a sub-optimal run at the end of the 1 hour time limit. In this regard, the standard k means algorithm produces the best results by far considering the number of valid clusters, showing its efficacy in computational effort distribution for PONPP.

Peak memory usage values are once again consistent with the maximum cluster size values, as this determines the largest sub-problem. k means produced excellent numbers across the board, with H50 just managing to produce the lowest memory usage of 298 MiB. Most of the instances compared favourably to the memory usage of BCA, which peaked at around 5 GiB due to the memory required to build and maintain the initial tree, although there are a number of optimisations than can be implemented to reduce this value.

As for objective function values, SPATH produced the best upper bound with R 53.43 mil, with DB80, k10 and H10 all producing good values within approximately 3 %. Due to the low average intra-cluster distance of DBSCAN, it produces the best bound of the clustering methods in 1 hour. Comparing the standard k means algorithm with the hybrid algorithm, the k means algorithm gave slightly better bounds under the time constraint. Similar objective values for SPATH and the clustering indicates that the best clustering instances does not introduce errors of more than 14 %, although the actual error margin may be much lower.

8 Conclusion

In this paper two heuristic techniques were incorporated into a MILP model of PONPP, allowing for the optimisation with the inclusion of fibre duct sharing. The numerical and computational results of the path heuristic in small scale tests showed promising performance, with drastically reduced computation times and less than a 3 % gap in comparison with the best calculated bounds across all data sets. This could indicate that in practice, paths that are much longer than the shortest path rarely result in lower deployment cost, indicating that fibre duct sharing opportunities may be limited in real-world deployments with civil restrictions.

Given the numerical results, both SPATH and the clustering methods outperforms BCA by quite a margin, even when BCA is given the best possible chance, producing up to 44 % lower objective values. Time complexity wise, the heuristics dramatically reduce computation time, although BCA is still faster by an order of magnitude. However, in practice, this discrepancy is reduced since BCA needs to be run a number of times to produce a good solution.

Overall, the heuristics proved to be very capable at solving PONPP with high accuracy and with fast computation times. The results suggest that the standard k means algorithm is best suited for clustering PONPP, providing very good bounds at a fraction of both the computational effort and memory required. Unfortunately, worse than claimed performance for BCA suggests that it may be unsuitable for practical and inherently clustered data sets such as *CityNet*.

Following this research, a more connectivity-aware clustering method can be investigated to take advantage of the nature of PONPP. Also, the estimation of the true distance from optimum for *CityNet* would be interesting to determine the effectiveness of the SPATH heuristic when applied to large instances. Also, the data sets can be preprocessed to reduce its complexity through edge substitution, as is done in a large number of other studies.

References

[1] Ahmad A, Bianco A, Bonetto E, Cuda D, Castillo G & Neri F, 2011, *Power-aware logical topology design heuristics in wavelength-routing networks*, Proceedings of the 15th International Conference on Optical Network Design and Modeling (ONDM), pp. 1–6

[2] Arulselvan A, Bley A, Gollowitzer S, Ljubić I & Maurer O, 2011, *MIP modeling of incremental connected facility location*, Network Optimization, pp. 490–502

[3] Atesio GmbH, Bundesallee 89, Berlin, Germany. http://www.atesio.de

[4] Bley A, Hashemi S & Rezapour M, 2013, *IP modeling of the survivable hop constrained connected facility location problem*, Electronic Notes in Discrete Mathematics, **41**, pp. 463–470

[5] Effenberger F, Clearly D, Haran O, Kramer G, Li RD, Oron M & Pfeiffer T, 2007, *An introduction to PON technologies, Topics in optical communications*, IEEE Communications Magazine, **45(3)**, March, pp. 17–25

[6] Eppstein D, 1994, *Finding the k shortest paths*, Proceedings of 35th Annual Symposium on Foundation of Computer Science, pp. 154–165

[7] Ester M, Kriegel H, Jörg S & Xu X, 1996, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231

[8] Gollowitzer S & Ljubić I, 2011, *MIP models for connected facility location: A theoretical and computational study*, Computers & Operations Research, **38**, pp. 435–449

[9] Gupta A, Kleinberg J, Kumar A, Rastogi R & Yener B, 2001, *Provisioning a virtual private network: a network design problem for multicommodity flow*, Proceedings of the 33rd Anual ACM Symposium on Theory of Computing, New York, pp. 389–398

[10] Kantarci B & Mouftah H, 2011, *Optimization models for reliable long-reach PON deployment*, Proceedings of the IEEE Symposium on Computers and Communications, pp. 506–511

[11] Kantarci B & Mouftah H, 2012, *Availability and cost-constrained long-reach passive optical network planning*, IEEE Transactions on Reliability, **61(1)**, pp. 113–124

[12] Khan S, 2005, *Heuristics-based PON deployment*, IEEE Communications Letters, **9(9)**, pp. 847–849

[13] Kim Y, Lee Y & Han J, 2011, *A splitter location-allocation problem in designing fiber optic access networks*, European Journal of Operational Research, **210(2)**, pp. 425–435

[14] Kokangul A & Ari A, 2011, *Optization of passive optical network planning*, Applied Mathematical Modelling, **35(7)**, pp. 3345–3354

[15] Lakic B & Hajduczenia M, 2007, *On optimized passive optical network (PON) deployment*, Proceedings of the 2nd International Conference on Access Networks Workshops, pp. 1–8

[16] Leitner M, Ljubić I, Sinnl M & Werner A, 2013, *On the two-architecture connected facility location problem*, (Unpublished) Technical Report 13–29, ZIB, Takustr. 7, 14195 Berlin

[17] Li J & Shen G, 2009, *Cost minimization planning for greenfield passive optical networks*, IEEE/OSA Journal of Optical Communications and Networking, **1(1)**, pp. 17–29

[18] Ljubić I & Gollowitzer S, 2013, *Layered graph approaches to the hop constrained connected facility location problem*, INFORMS Journal on Computing, **25(2)**, pp. 256–270

[19] Lloyd S, 1982, *Least squares quantization in PCM*, IEEE Transactions on Information Theory, **28(2)**, pp. 129–137

[20] Lv M & Chen X, 2009, *Heuristic based multi-hierarchy passive optical network planning*, Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1–4

[21] Mitcsenkov A, Paksy G & Cinkler T, 2011, *Geography- and infrastructure-aware topology design methodology for broadband access networks (FTTx)*, Photonic Network Communications, **21(3)**, pp. 253–266

[22] Mitcsenkov A, Paksy G & Cinkler T, 2009, *Topology design and CAPEX estimation for passive optical networks*, Proceedings of the 6th International Conference on Broadband Communications, Networks and Systems, pp. 1–8

[23] Ouali A & Poon KF, 2011, *Optimal design of GPON/FTTH networks using mixed integer linear programming*, Proceedings of the 16th European Conference on Networks and Optical Communications, pp. 137–140

[24] Poon K, Mortimore D & Mellis J, 2006, *Designing optimal FTTH and PON networks using new automatic methods*, Proceedings of the 2nd Institution of Engineering and Technology International Conference on Access Technologies, pp. 49–52

[25] Swamy C & Kumar A, 2004, *Primal-dual algorithms for connected facility location problems*, Algorithmica, **40(4)**, pp. 245–269

[26] van Loggerenberg S, 2013, *Optimization of passive optical network planning for fiber-to-the-home applications*, MEng Thesis, North-West University, Potchefstroom Campus

[27] Wong MA, 1982, *A hybrid clustering method for identifying high-density clusters*, Journal of the American Statistical Association, **77(380)**, pp. 841–847

[28] Xiong W, Wu C, Wu L, Guo X, Chen Y & Xie M, 2011, *Ant colony optimization for PON network design*, Proceedings of the 3rd IEEE International Conference on Communication Software and Networks, pp. 380–383

[29] y Villalba TV, Rossi S, Mokarzel M, Salvador M, Neto H, Cesar A, Romero M & de L. Rocha, 2009, *Design of passive optical networks using genetic algorithm*, Proceedings of the SBMO/IEEE MTT-SS International Microwave and Optoelectronics Conference, pp. 682–686

[30] Yen JY, 1971, *Finding the k shortest looless paths in a network*, Management Science, **17(11)**, pp. 712–716