This is a post-peer-review, pre-copyedit version
CP 2015: Principles and Practice of Constraint Programming
doi: 10.1007/978-3-319-23219-5_35

Constraint-based Local Search for Finding Node-Disjoint Bounded-Paths in Optical Access Networks

Alejandro Arbelaez, Deepak Mehta, and Barry O'Sullivan

Insight Centre for Data Analytics, University College Cork, Ireland
{alejandro.arbelaez|deepak.mehta|barry.osullivan}@insight-centre.org

Abstract. Mass deployment of fibre access networks is without doubt one of the goals of many network operators around the globe. The Passive Optical Network has been proposed as a solution to help deliver mass deployment of fibre-to-the-home (FTTH), by reducing the cost per customer compared to current commercially available technologies. A major failure in the access network (e.g., fibre cut, amplifier failure, or other equipment can fail) might affect tens of thousands of customers. Therefore, protecting the network from such failures is one of the most important requirements in the deployment of FTTH solutions. In this paper we use a constraint-based local search approach to design reliable passive optical networks via node-disjoint paths whereby each customer is protected against any one node or link failure. We experiment with a set of very large size real-world networks corresponding to Ireland, Italy and the UK and demonstrate the tradeoff between cost and resiliency.

1 Introduction

Continuous growth in the amount of data transferred within national and global networks in the last decade necessitates new infrastructures and data transfer technologies [1]. In line with this, the goal of the DISCUS project [2] is to develop an end-to-end network design that can provide high-speed broadband capability of at least three orders-of-magnitude greater than today's networks to all users, reduce energy consumption by 95%, and remain economically viable.[1] The architecture is based on maximising the sharing of network infrastructure between customers by deploying a Long-Reach Passive Optical Network (LR-PON) [3] in the access part. All network points have equal bandwidth and service capability including core bandwidth (10Gbs to 100Gbs) delivered to the access network.

LR-PONs provide an economically viable solution for fibre-to-the-home network architectures [4]. In a LR-PON fibres are distributed from metro nodes (MNs) to exchange sites (ESs) through cables that form a tree distribution network. Typically due to signal loss the maximum distance from a MN to an ES is up to 90 km, and the maximum distance from an ES to the customers is up to 10 km. A major fault occurrence in a LR-PON would be a complete failure of a

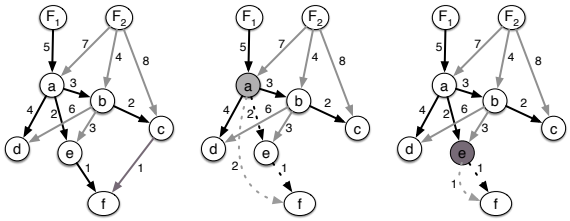
[1] http://www.discus-fp7.eu

MN, which could affect tens of thousands of customers. A dual homing protection mechanism is recommended for LR-PONs whereby customers are connected to two MNs, so that whenever a single MN fails all customers are still connected to a back-up. Simply connecting two MNs to an ES is not sufficient to guarantee the connectivity because if a link or a node is used is common in the routes of fibre going from ES to its 2 MNs then both MNs would be disconnected.

Broadly speaking, there are two protection strategies for dual homing. One approach is protecting links in the tree distribution network, e.g., a fibre cut, amplifier failure, or other equipment that can fail. This strategy, known as the edge-disjoint solution, allows switching to an alternative path whenever a link in the distribution network fails. Alternatively, protection can be provided at the node level with node-disjoint paths between MNs and ESs, when an entire ES fails and all adjacent links to the node will be affected. This strategy, known as node-disjoint solution, provides a stronger protection mechanism and allows switching to an alternative path whenever a node fails. The selection of one protection mechanism over another is a matter of choice for the broadband provider, generally higher protection means higher deployment cost.

Figure 1 shows an example with two MNs F_1 and F_2 and the set of ESs $\{a, b, c, d, e, f\}$. Black and gray coloured edges are used to show the trees corresponding to the facilities F_1 and F_2 respectively, and maximum allowed distance λ=12. This example shows three scenarios satisfying the length constraint, however, each scenario shows a different level of resiliency. Dashed lines indicate edges leading to a conflict, i.e., violating node or edge disjointness, and gray nodes indicate the conflicting node when violating node-disjointness. The total solution cost for Figure 1(a) is 46. The indicated solution satisfies the length constraint (i.e., the distance from the MNs to any ES is at most λ=12) and it also satisfies node-disjoint constraints and consequently edge-disjoint constraints. Here we observe that failure of any single link or node would not affect the remaining ESs. Figure 1(b) does not satisfy node-disjoint constraints but edge-disjoint constraints. Here we observe that a single node failure could disconnect one or more ESs from the MNs F_1 and F_2, e.g., f would be disconnected from F_1 and F_2 whenever a fails. Nevertheless, the solution is resilient to any single facility or single link failure. Figure 1(c) shows a solution that violates both node and edge disjoint constraints. In this example the link $\langle e,\ f\rangle$ appears in both trees and the path from f is neither edge-disjoint nor node-disjoint but it is resilient to a failure of a single MN.

Network operators tend to evaluate multiple options. Their choices of providing resiliency depends on many factors, e.g., captial and operational costs, rural or sparse regions, business or residential customers etc. Ideally they would prefer a tool where one can try to impose different configurations and get feedback quickly. In particular we focus on node-disjointness as the other options can be seen as relaxation of the node-disjoint constraints. The problem is to determine the routes of cable fibres such that there are two node-disjoint paths from each ES to its two MNs, the length of each path must be below a given threshold and the total cable length required for connecting each ES to two MNs



(a) Node and Edge disjoint (b) Not Node but Edge Disjoint (c) Neither Node nor Edge Disjoint

Fig. 1. Passive Optical Networks with and without node and edge disjoint protection.

should be minimised. The bottleneck is that the instances of the network sizes can be very huge e.g., UK has approximately 5000 ESs while Italy has more than 10,000. Solving very large instances of this problem in limited time is very challenging. We therefore apply a constraint-based local search and develop an efficient algorithm to handle node-disjoint constraints for selecting best move during LS.

2 Constraint-based Local Search

Constraint-based local search has been recently introduced as an alternative solution to efficiently tackle many network design problems [5, 6] ranging from telecommunications to transportations, and VLSI circuit design.

The local search (LS) algorithm starts with an initial solution and iteratively improves the solution by performing small changes. A move operator guides the algorithm towards better solutions. [5] defines the *node* and *subtree* operator, the algorithm moves a given node (resp. a subtree and the nodes emanating from it) from one location to another in the tree by improving the objective. These operators have been used to find distance-constrained and edge-disjoint paths. Empirical results suggest that the subtree operator outperforms the node operator. [6] defines the *arc* operator: the algorithm moves a given arc from one location to another in the tree, this operator has been used in the context of the routing and wavelength assignment problem [7] and to find edge disjoint paths in a given graph [8]. We remark that `node` and `subtree` move-operators are specifically designed for tree structures whereas the arc operator is more general and it can be applied to more general graphs. Nevertheless, the former is more efficient when we are dealing with trees, while the latter is more expensive. In this paper we limit our attention to using the subtree move-operator as it provides a better time complexity than that of the arc operator, i.e., $O(n)$ vs. $O(n^3)$ when considering the distance constraint. We refer the reader to [5] for a detailed analysis of the time complexities of the local search operators.

Other related work includes [9] where the authors proposed a dedicated algorithm to tackle the Distance Constrainted Minimum Spanning Tree problem.

The authors studied two move-operators to tackle the distance constraint, informally speaking nodes are sequentially added in the tree using Prim's algorithm, and a backup route is used whenever the move cannot satisfy the length constraint. This framework is difficult to extend with side constraints, and therefore the algorithms cannot be extended to solve our problem with node and edge disjointed paths. For instance, the operators rely on a pre-computed alternative route from the root to any node. However that route might violate disjointedness.

2.1 Node-Disjointness

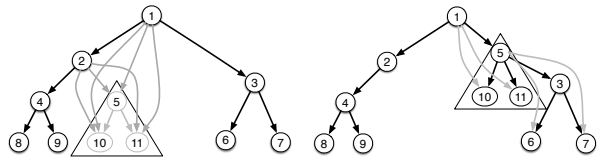
The LS algorithm uses the move-operators in two different phases. In the intensification phase a subtree is selected and it is moved to another location that provides the maximal improvement in the objective function. In the perturbation phase both a subtree and its new location are selected randomly in order diversify the search by adding noise to the current incumbent solution. In order to complete the intensification and diversification phases, the sub-tree operator requires four main functions: *removing* a subtree from the current solution; checking whether a given solution is *feasible* or not; *inserting* a subtree in the solution; finding the *best location* of a subtree in the current solution.

Additionally, the LS algorithm is equipped with an incremental way of maintaining information related to checking constraints and computing the cost of the assignment. For the length constraint it is necessary to maintain for each node e_j the length of the path from MN i to ES (or node in the tree) j, and the length of the path emanating from e_j down to the farthest leaf from e_j. These two lengths are updated after removing and inserting a subtree in the tree. For enforcing the edge-disjoint constraint in the two paths of a client, we must maintain the predecessors of each client in the two paths associated to the MNs, and require that the two predecessors must be different [5]. For enforcing the node-disjoint constraint between the paths of two clients it is necessary to maintain all the nodes occurring in each path except the source and target. This is done by maintaining the transitive graph.

Let M be the set of facilities or MCs. Let E be the set of clients or ESs. Let $E_i \subseteq E$ be the set of clients associated with MN $m_i \in M$. We use N to denote the set of nodes, which is equal to $M \cup E$. We use T_i to denote the tree network associated with MN i. We also use $N_i \subseteq N = E_i \cup \{m_i\}$ to denote the set of nodes in T_i. Let λ be the maximum path-length from a facility to any of its clients.

Let e_j be a node in the tree T_i associated with facility m_i. Let e_{p_j} be the predecessor of e_j, and let L_{e_j} be a list of all nodes in the subtree starting at e_j of tree T_i. We now describe the complexities of remove, insert, feasibility check and best location operations performed during search with respect to node-disjoint constraint:

Remove. To remove a subtree it is necessary to update the transitive graph. This is done by removing all the transitive links between any node in the path from the MN m_i down to e_j and all the elements in L_{e_j}. In Figure 2(a) gray lines show the links that must be removed from the transitivity graph, i.e., links that



(a) Remove the subtree rooted at node 5

(b) Insert the subtree rooted at 5 between node 1 and node 3

Fig. 2. Removing/Inserting a subtree while maintaining transitive links.

reach any node in the subtree emanating from node 5. Thus, the time-complexity of removing a subtree rooted at e_j from the tree T_i is quadratic with respect to the number of clients of facility m_i.

Insert. The insertion involves adding the transitive links between each node in the path from the MN down to e_j and each element of L_{e_j}. Figure 2(b) shows the links that must be added in the transitivity graph; black links denote existing links in the tree and gray links denote the new links added by transitivity in the graph. Therefore, a subtree rooted at e_j can be inserted in T_i in quadratic time complexity with respect to the number of clients of facility m_i.

Feasibility. To verify the consistency of the node-disjoint constraint we need to check that any node occurring in the subtree starting from the node e_j is not transitively connected to any node in the path from the MN down to the potential predecessor of e_j in any other tree corresponding to other MNs. This is done by checking the occurrence of the links corresponding to the pairs of nodes occurring in the transitive graph. As the number of links can be quadratic, the time complexity is quadratic. Additionally, if the node e_j is breaking an existing arc $\langle e_p, e_q \rangle$, it is necessary to check that e_j is not transitively connected to any node in the subtree emanating from e_q.

Best Location. Selecting the best location for a given subtree involves traversing all the nodes of the tree associated with the MN and selecting the one that maximises the reduction in the cost. As the complexity of feasibility is quadratic, the time complexity of finding the best location is cubic in the number of nodes of a given tree.

The complexities for feasibility checking and best location described above do not make any assumption on the order in which different locations are explored. In the following we will show that the amortized complexities can be reduced if the locations in the tree are traversed in a particular order.

For a given subtree rooted at e_j there are two types of feasible locations. The first is a node-type location, say e_k, of the tree and the feasibility check is whether the parent of e_j can be e_k or not. This type of location is denoted by $(e_k, \emptyset)$. The second is a link-type location, say $\langle e_{p_l}, e_l \rangle$, of the tree and the feasibility check is whether we can break the link $\langle e_{p_l}, e_l \rangle$ by making e_{p_l} and e_j the parents of e_j and e_l respectively. This type of location is denoted by

Algorithm 1 BestLocation(e_j, T_i, G)

```
1: Remove subtree rooted at e_j from T_i, update T_i and G
2: Nlocations ← ∅, Llocations ← ∅
3: Let s be a stack, let q be a queue
4: s.push(root(T_i))
5: while s ≠ ∅ do
6:     e_k ← s.pop()
7:     trans ← {⟨e_k, e_l⟩) | e_l ∈ L_{e_j} ∧ ⟨e_k, e_l⟩ ∈ G}
8:     if trans = ∅ then
9:         Nlocations ← Nlocations ∪ {(e_k, ∅)}
10:        for all ⟨e_k, e_l⟩ ∈ T_i do
11:            Llocations ← Llocations ∪ (e_k, {e_l})
12:            s.push(e_l)
13:        end for
14:        if out(e_k, T_i) = 0 then
15:            q.append((e_k, check))
16: while q ≠ ∅ do
17:    (e_l, status) ← q.remove()
18:    if ⟨e_{p_l}, e_l⟩ ∈ Llocations then
19:        if ⟨e_j, e_l⟩ ∈ G||status =rem then
20:            Llocations ← Llocations \{⟨e_{p_l}, e_l⟩}
21:            q.append((e_{p_l}, rem))
22:        else
23:            q.append((e_{p_l}, check))
24: Best ← SelectBest(e_j, NLocations ∪ Llocations)
25: return Best
```

$(e_{p_l}, \{e_l\})$. For node-type locations we explore the tree in a depth-first manner, whereas for link-type locations we explore the tree in a bottom-up manner.

Algorithm 1 shows the pseudo-code to compute the feasible set of locations for a given subtree rooted at e_j and then select the best location in a tree T_i with respect to a transitive graph G. The algorithm maintains the set of feasible node-type locations with respect to the node disjoint constraint in the set *Nlocations*, and employs a depth-first search algorithm with a stack s (Lines 5-18). It explores the successors of a node e_k only when there is no existing link in the transitive graph between the node e_k and any node in L_{e_j}. Notice that the time complexity of the first while loop is quadratic in the number of the nodes of the tree. Also notice that in the first while loop, the algorithm updates the set *Llocations* by adding all the candidate links starting from the node e_k which will be filtered in the next while loop. Moreover, for each leaf node, when the outgoing degree of node e_k is 0 (i.e. $out(e_k, T_i) = 0$), the node is appended to the queue with the status `check` as the feasible set of link locations follows a bottom-up approach.

In the second while loop, the algorithm filters the set of candidate links to determine the feasible set by employing a bottom-up approach with a queue q. It explores a node e_l only if its predecessor is a valid parent candidate for the node e_j of a given subtree. If there is a link from e_j to e_l in the transitive

graph or its status is marked as `rem` then (e_{p_l}, e_l) is not a feasible link-location and subsequently all the links involved in the path from the facility to e_l would not be feasible link-locations. Therefore, the queue is appended with $(e_{p_l}, \texttt{rem})$ and the set *Llocations* is filtered. Otherwise, the status of the predecessor of e_l is set to `check` and appended to the q. Notice that the time complexity of second while loop is linear in the number of links of the tree. Thus, the amortized time complexity of feasibility checking is linear and, consequently, the amortized complexity of *BestLocation* is quadratic.

We remark that computing f_j^i for checking the length constraint can be merged with the verification of the transitive graph with the same worst-case time complexity i.e., linear. However we still incrementally maintain f_j^i and b_j^i for each node in the graph in order to amortize the complexity of the operation to depend only in the number of nodes in the subtree enumerating from e_j.

3 Empirical Evaluation

All experiments were performed on a 4-node cluster, each node features 2 Intel Xeon E5430 processors at 2.66Ghz and 12 GB of RAM memory. We report the median cost solution of 11 independent executions of the LS algorithm with a 1-hour time limit for each experiment. We would like to point out that CPLEX ran out of memory when we tried to create and solve the MIP model for nearly all instances. Hence, we limit our attention to the CBLS algorithm.

To evaluate the performance of the proposed LS we use 3 real networks from three EU countries: Ireland with 1121 ESs, the UK with 5393 ESs, and Italy with 10708 ESs. For each dataset we use [10] to identify the position of the MNs and computed four instances for each country. Ireland with 18, 20, 22, and 24 MNs; the UK with 75, 80, 85, and 90 MNs; and Italy with 140, 150, 160, and 170 MNs. We set λ=90 km. Cable fibre cost is a dominant factor in the deployment of LR-PONs, therefore we report the estimated total cost in euro to deploy edge and node disjoint distribution networks. To this end, we use the subtree operator to optimise the total cable link distance and then we extract the actual fibre cost of the solutions. [11] suggests that on average four fibres will be needed to provide broadband services to all customers for each ES, and the cost of the fibres are defined as follows: {12, €2430}, {24, €2430}, {48, €3145}, {96, €4145}, {144, €5145}, {192, €6145}, {240, €7145}, {276, €7859}, the left part indicates the number of fibres per cable and the respective cost per km.

In Table 1 we report results. As expected the cost of node-disjointness is higher than that of edge-disjointness as the former provides stronger protection. However, it is worth pointing out that the total cable link distance of deployment of a fully node disjoint solution is only up to 13.7% more expensive for Ireland, 16.3% for the UK, and 19.7% for Italy, with respect the best known edge-disjoint solutions. And with a gap between 22% and 31% with respect to the LBs,[2]

[2] We recall that the lower bounds are valid for both edge and node disjoint paths. Therefore we expect the node disjoint solution to be much closer to the actual optimal solution.

Table 1. Results for Ireland, UK and Italy to provide Node and Edge disjoint paths.

Country	$\|M\|$	Cable Link distance					Fibre Cost in Mill €	
		LB	Protection		GAP (%)			
			Node	Edge	Node	Edge	Node	Edge
Ireland $\|E\|$=1121	18	14809	19824	17107	25.78	13.43	55	49
	20	14845	19508	16819	24.27	11.73	54	48
	22	14990	19076	16711	22.18	10.29	53	48
	24	14570	18452	16163	22.23	9.85	51	46
UK $\|E\|$=5393	75	54720	78131	65377	30.31	16.30	223	197
	80	54975	77269	64565	29.20	14.85	220	194
	85	55035	75799	63517	28.00	13.35	216	191
	90	55087	74376	62163	25.93	11.38	212	188
Italy $\|E\|$=10708	140	76457	111434	89418	31.43	14.49	335	293
	150	76479	109954	88255	30.45	13.34	329	288
	160	76794	109708	88336	30.05	13.06	326	286
	170	77013	108288	87405	29.09	11.88	321	282

which is used as the reference baseline for comparison. Finally, we would like to highlight the cost in euro of deployment both the edge and node disjoint alternatives. We foresee that the cost of deployment node disjoint protection in LR-PON in Ireland, the UK, and Italy is respectively 9.9%, 11%, and 12% more expensive (on average) than the corresponding edge disjoint alternative.

4 Conclusions and Future Work

We have extended an existing constraint-based local search framework for finding the Distance-Constrained Node-Disjoint paths between a given pairs of nodes. The efficiency of our approach is demonstrated by experimenting with a set of problem instances provided by network operators in Ireland, the UK, and Italy. We have seen that providing node-disjointness increases the cable link distance cost up to 13% for Ireland, 16% for the UK, and 19% for Italy with respect to the best known solutions with a weaker protection mechanism. Additionally, we provided a cost estimation in euro to deploy both protection solutions. Our experiments indicate that implementing node disjointness is between 9% and 11% more expensive with respect to the best known solutions with weaker protection.

Acknowledgments. We would like to thank both *eircom* and *Telecom Italia* for providing their optical network data. We would also like to thank *atesio* for providing the Italian reference network on which the Italian experiments have been carried out. This work is supported by the EU Seventh Framework Programme (FP7/ 2007-2013) under grant agreement no. 318137 (DISCUS) and by Science Foundation Ireland (SFI) under grant 08/CE/I1423. The Insight Centre for Data Analytics is also supported by SFI under Grant Number SFI/12/RC/2289.

References

1. Benhamiche, A., M.A.R., Perror, N.: On the design of optical ofdm-based networks. Lecture Notes in Computer Science **6701** (2011) 1–6
2. Ruffini, M., Wosinska, L., Achouche, M., Chen, J., Doran, N.J., Farjady, F., Montalvo, J., Ossieur, P., O'Sullivan, B., Parsons, N., Pfeiffer, T., Qiu, X.Z., Raack, C., Rohde, H., Schiano, M., Townsend, P., R., W., Yin, X., Payne, D.B.: Discus: An end-to-end solution for ubiquitous broadband optical access. IEEE Communications Magazine **52**(2) (2014) 24–32
3. Davey, R., Grossman, D., Rasztovits-Wiech, M., Payne, D., Nesset, D., Kelly, A., Rafel, A., Appathurai, S., Yang, S.H.: Long-reach passive optical networks. Journal of Lightwave Technology **27**(3) (Feb 2009) 273–291
4. Payne, D.B.: FTTP deployment options and economic challenges. In: Proceedings of the 36th European Conference and Exhibition on Optical Communication (ECOC 2009). (2009)
5. Arbelaez, A., Mehta, D., O'Sullivan, B., Quesada, L.: Constraint-based local search for edge disjoint rooted distance-constrainted minimum spanning tree problem. In: CPAIOR'15. (2015) 31–46
6. Pham, Q.D., Deville, Y., Van Hentenryck, P.: LS(Graph): a constraint-based local search for constraint optimization on trees and paths. Constraints **17**(4) (2012) 357–408
7. Mukherjee, B.: Optical WDM Networks (Optical Networks). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
8. Blesa, M., Blum, C.: Finding edge-disjoint paths in networks: An ant colony optimization algorithm. Journal of Mathematical Modelling and Algorithms **6**(3) (2007) 361–391
9. Ruthmair, M., Raidl, G.R.: Variable neighborhood search and ant colony optimization for the rooted delay-constrained minimum spanning tree problem. In Schaefer, R., Cotta, C., Kolodziej, J., Rudolph, G., eds.: PPSN (2). Volume 6239 of Lecture Notes in Computer Science., Springer (2010) 391–400
10. Ruffini, M., Mehta, D., O'Sullivan, B., Quesada, L., Doyle, L., Payne, D.B.: Deployment strategies for protected Long-Reach PON. Journal of Optical Communications and Networking **4** (2012) 118–129
11. DISCUS: Project Deliverable 2.6, Architectural optimization for different geotypes