Noname manuscript No.
(will be inserted by the editor)

An Adaptive Large Neighbourhood Search Algorithm for Diameter Bounded Network Design Problems

Michele Garraffa · Deepak Mehta ·
Barry O'Sullivan · Cemalettin Ozturk ·
Luis Quesada

July 7, 2021

Abstract This paper focuses on designing a diameter - constrained network where the maximum distance between any pair of nodes is bounded. The objective considered is to minimise a weighted sum of the total length of the links followed by the total length of the paths between the pairs of nodes. First, the problem is formulated in terms of Mixed Integer Linear Programming (MILP) and Constraint Programming (CP) to provide two alternative exact approaches. Then, an adaptive large neighbourhood search (LNS) to overcome memory and runtime limitations of the exact methods in large size instances is proposed. Such approach is based on computing an initial solution and repeatedly improve it by solving relatively small subproblems. We investigate various alternatives for finding an initial solution and propose two different heuristics for selecting subproblems. We have introduced a tighter lower bound, which demonstrates the quality of the solution obtained by the proposed approach. The performance of the proposed approach is assessed using three real-world network topologies from Ireland, UK and Italy, which are taken from national telecommunication operators and are used to design a transparent optical core network. Our results demonstrate that the LNS

First Author
School of Computer Science & IT, University College Cork, Cork, Ireland
E-mail: michele.garraffa@ucc.ie

Second, Third and Fifth Authors
Insight Centre for Data Analytics
School of Computer Science & IT, University College Cork, Cork, Ireland
E-mail: deepakmehta79@gmail.com, barry.osullivan@insight-centre.org, luis.quesada@insight-centre.org

Fourth Author
Munster Technological University, Process, Energy and Transport Engineering, Bishopstown Cork, Ireland. T12 P928
E-mail: cemalettin.ozturk@cit.ie

approach is scalable to large networks and it can compute very high quality solutions that are close to being optimal.

Keywords Diameter Bounded Network Design · Mixed Integer Linear Programming · Constraint Programming · Large Neighbourhood Search · Optical Networks.

1 Introduction

Many network design problems arising in areas as diverse as Quality of Service (QoS) routing, traffic engineering, and computational sustainability, require clients to be connected to a facility under path-length and budget constraints. In general, the length of the path can be interpreted as distance, delay, signal loss, etc. For example, in a multicast communication setting where a single node broadcasts to a set other of nodes, it is important to restrict the path delays between them. In Long-Reach Passive Optical Networks (LR-PON), a metro-core node is connected to a set of local-exchange sites via optical fibres and the length of the fibre between a local exchange site and its metro-core node is bounded due to signal loss. The goal in this case is to minimise the cost resulting from the total length of fibres [31, 19]. In Transparent Optical Core Networks (TOCN), all pairs of core nodes are connected via optical fibres and the length of the fibre between any pair of core nodes is bounded due to signal attenuation. The objective in this case is to minimise the cost resulting from the length of the links chosen to connect the core nodes. In Very Large Scale Integration (VLSI) circuit design, the path delay is a function of the maximum interconnection path length, while the power consumption is a function of the total interconnection length [29]. In package shipment, service guarantee constraints are expressed as restrictions on total travel time from an origin to a destination, and the organisation wants to minimise the transportation cost [36]. In wildlife conservation, which is an application from computational sustainability [15], the landscape connectivity is vital in supporting resilient wildlife populations in an increasingly fragmented habitat matrix. In this setting, landscape connectivity is a function of the length of the path in terms of landscape resistance to animal movement.

Many of these network design problems can be modelled as a Diameter Constrained Network Design (DCND) problem. The objective is to find a minimum cost network subject to the additional constraint that the length of the path between any pair of nodes must not exceed a given threshold. The DCND problem is NP-hard which can be shown using a reduction from the constrained-shortest path problem [37]. This problem can be formulated both as a Mixed Integer Linear Programming (MILP) and a Constraint Programming (CP) model as exact solution approaches. While MILP formulation takes advantage of network flow representation, CP formulation exploits dedicated global constraints on networks [34] to formulate and solve the problem.

Since the size of real-world instances can be prohibitively large for exact approaches both in terms of time and memory, a large neighbourhood search

(LNS) technique [32] is proposed and its utility is demonstrated by solving instances of the TOCN problem close to optimality. LNS attempts to combine the power of systematic search with the scalability of local search. LNS scales significantly beyond commercial optimisation tools, such as CPLEX, with the cost of not guaranteeing optimality. The key idea behind LNS is to first compute an initial solution, then repeatedly consider a sub-problem and optimise it until the stopping condition is met. In order to find an initial solution, we use a simple heuristic-based MILP decomposition scheme. Although this heuristic allows us to start with a good quality solution, it consumes a lot of time. Therefore, we propose an alternative approach that first enforces the connectivity constraint by computing a minimum spanning tree. Then, a MILP-based decomposition scheme is applied to repair the pairs for which the length constraint is violated. This significantly reduces the time required to solve the problem without sacrificing the quality of the initial solution. In the context of LNS, and improving an initial solution, one of the important aspect is how to select subproblems. If subproblems are selected in an ad-hoc manner, then LNS might not help. For the diameter-bounded network design problem, we propose two different ways of selecting subproblems. The first one is based on the notion of *support*, which is computed as the number of pairs of nodes that are affected when the current solution is destroyed. The second criterion is *link graph*, which is based on the size of the subnetwork with respect to the current solution that is going to be affected for a given subproblem.

We also investigate the impact of different ways of computing initial solutions, subproblem selection, and size of the subproblem on the time and solution quality for the instances of the TOCN problem. The adaptive LNS method presented in this paper is general. It can also be easily extended to solve other variants of the network design problems such as the Steiner tree problem [5, 27], the buy-at-bulk network design, transportation network problems [20] and placement of warehouse problems [26].

The remainder of the paper is organised as follows. Section 2 gives a formal specification of the problem and discuss the related literature. Section 3 provides a MILP formulation and a CP formulation for the DCND. Section 4 proposes a LNS approach and describes alternative methods to find an initial solution and two ways of selecting subproblems. Section 5 presents a computational assessment of the exact approaches and some empirical results obtained by running the LNS approach on national telecommunication networks of three countries. Section 6 introduces a new lower bound, which is defined in terms of the rooted distance-constrained minimum spanning tree problem (RDMSTP) [16]. Finally, Section 7 includes some conclusions and future research directions.

2 Problem Specification and Related Work

The Diameter Constrained Network Design (DCND) problem consists of computing a subgraph $G^* = \langle N, L^* \rangle$ from a given graph $G = \langle N, L \rangle$ such that there exists a path between each pair of nodes $\{i, j\} \subseteq N$ in G^*, whose length is bounded by a value λ. We also say that the diameter of G^* is bounded by λ, since the diameter of a graph is defined as the length of the "longest shortest path" between any two nodes. The primary objective is to minimise the sum of the cost of the selected links. Additionally, we also seek to minimise the sum of the lengths of the paths.

As mentioned before, DCND is NP-hard [5]. The difference between DCND and related approaches relies on the fact that we ensure that there is a bounded path for every pair of nodes, and the distance is expressed in terms of the sum of the lengths of the links involved in the paths and not the number of hops. In fact, the following related problems can be considered specific cases of DCND:

1. The constrained shortest path problem consists of finding a minimum cost path from a source node to a destination node in a given graph subject to the condition that the total length of the path must be less than or equal to a specified value. This problem has been well studied [37]. However, the difference with respect to our case is that they look for one distance-bounded path for a given pair of nodes, while we are interested in finding bounded paths for a given set of pairs of nodes and the primary goal is to minimise the sum of the length of the links.
2. The Multiple Origin Multiple Destination (MOMD) problem is to find a sub-network with the minimum total weighted length that connects each origin-destination pair with a path. In this problem, the length of the path is not bounded. Nevertheless, the problem is still NP-hard, as shown in [2]. The MOMD problem can be seen as a simplified version of the logistics planning problem in which packages are required to be transported from their origins to their destinations by unlimited number of trucks for transporting unlimited number of packets. DCND is a more general problem than MOMD as it also considers the bounds on the lengths of the paths. In terms of logistics planning problem, DCND includes the capacities of the trucks. The optimal solution of DCND would be a tighter lower bound than that of MOMD for the logistics planning problem.
3. The Rooted Distance-Constrained Minimum Spanning Tree Problem (RD-CMST) is to find a minimum cost tree such that the length of the path from the root node to any other node in the tree does not exceed a given threshold [29]. Even though the distance between the root node to any other node is bounded, the distance between the remaining pair of nodes is not bounded. In relation to this problem, an approximation algorithm with an approximation ratio of $\mathcal{O}(n \log d)$ for poly-bounded lengths of links has been proposed [11]. In [5] the authors propose a new algorithm to improve the approximation introduced in [11], but they ignore the minimisation of the length of the paths.
4. The hop-constrained network design problem is the closest problem to DCND. The difference is that in this problem the lengths of all the links are equal, so the bound is on the number of links included in a path connecting a pair of nodes [21]. An instance of DCND can be transformed into

an instance of the hop-constrained network design problem by introducing additional nodes and splitting the existing links such that all of them have the same length. This process may require adding a number of dummy nodes equivalent to the size of the total length of the links in the input graph in the worst case, which makes this transformation unsuitable even for the smallest instances considered in this paper. The hop-constrained network design problem can be modelled in Constraint Programming using the graph variables and the global constraints available in Choco 3.0, Choco-Graph module [33]. The diameter constraint provided by this library assumes links of equal length and it can handle graphs of up to 500 nodes. However, the transformation described before can lead to graphs of several thousands of nodes, which would compromise the viability of the constraint programming approach. A way to overcome the limitation of these global constraints is to use *PathCumulative* constraint of Google-Or-Tools [30], which considers graphs with non-unitary weight and can impose that the diameter of the network is bounded by a certain value (see Section 3.3).

In this paper, we develop an adaptive large neighbourhood search algorithm to solve the DCND problem. Preliminary results of this approach were presented in [24], where the approach was used as a preprocessing step for generating the input for the routing and spectrum allocation problem discussed in [25]. With respect to what has been already presented in other publications, here we provide: (1) an improved MILP model, (2) a novel CP formulation, (3) a computational evaluation of both exact formulations, and (4) a novel lower bounding procedure. Furthermore, we enhance the results obtained on the LNS approach by providing: (5) alternative methods for neighbourhood search, (6) a detailed analysis of subproblem selection methods and (7) a correlation between the size of the subproblems and search effort / quality of the solution. We also present some empirical results for several real size networks of Ireland, UK and Italy. Our computational results demonstrate the effectiveness of our approach both in terms of scalability and quality.

3 Problem Formulations

In this section, a MILP and a CP model of the DCND are presented. In both formulations, the objective is a weighted sum of these terms:

- the total length of the links (TLL)
- the total weighted sum of the paths between all pair of nodes (TLP)

Some notation used to formulate the problem is introduced in Section 3.1. The MILP model and the CP model are presented in Section 3.2 and Section 3.3, respectively.

3.1 Notation

Here follows some notation used to formulate the problem:

- E is a set of directed edges. Each undirected link $\{i, j\} \subseteq L$ is associated with two directed edges $\langle i, j\rangle$ and $\langle j, i\rangle$ in E. The length of edge $\langle i, j\rangle$ is denoted as d_{ij} and the symmetry property ($d_{ij} = d_{ji}$) holds. We say that an undirected link is used if any corresponding directed edge is used.
- D is an $|N| \times |N|$ matrix whose components are equal to $d_{i,j}$ if $< i, j >\in E$, 0 otherwise.
- $\kappa \in N$ is the source node, which corresponds to the first node in the ordering of the nodes.
- $\mathcal{P}$ is the set of all ordered pairs of nodes. Each pair of nodes, $\rho \in \mathcal{P}$, is a tuple $\langle s(\rho), t(\rho)\rangle$ where $s(\rho)$ and $t(\rho)$ refer to the corresponding source and target nodes of that pair, respectively.
- $\texttt{In}(i)$ (resp. $\texttt{Out}(i)$) represents the set of all edges entering (resp. leaving) node i.
- λ is the upper bound on the length of the shortest path between any pair of nodes.
- β_ρ denotes the amount of commodity flowing from the source node to the target node of a pair of nodes $\rho \in \mathcal{P}$.

3.2 The MILP model

We now present a MILP model for DCND. In this model we opt for a node-link formulation because we do not have mandatory nodes. We remark that a link-path formulation would lead to consider all possible paths whose length is below the threshold as no further pruning is possible due to the absence of mandatory nodes and links [5]. The reader is referred to [23] for a detailed comparison of the two formulations.

As mentioned before, a solution of DCND is a connected undirected graph. Although the diameter constraint implies connectivity, we add redundant constraints to address this requirement in our model. We do so by computing a directed graph out of the undirected one, which corresponds to an acyclic orientation of the undirected graph. An acyclic orientation of an undirected graph is an orientation of the links of the undirected graph that leads to no cycles in the obtained directed graph. This orientation can be obtained by ordering the nodes and then orienting each edge from the smaller node to the greater node. The length of the directed edge is equivalent to its corresponding undirected link so the total length of the acyclic orientation of an undirected graph is the same of the undirected graph. We take advantage of this fact when enforcing connectivity as we will show later.

Once we have an acyclic orientation we enforce connectivity by selecting the node that is first in the order and making the incoming degree of any non-source node equal to one, and the incoming degree of the source equal to zero. Table 1 shows the complete MILP model. We now introduce the variables, constraints and objective function.

Variables In our model, we have the following variables:

Table 1 The MILP model proposed

$$
\begin{array}{llll}
\text{Minimise} & \alpha \times TLL + TLP & & \\
\text{Subject to} & \sum_{i\in(N-\{\kappa\})} y_{i\kappa} = 0 & & (1.1)\\
 & \sum_{j\in N} y_{ji} \geq 1 & \forall_{i\in N\setminus\{\kappa\}} & (1.2)\\
 & f_i + d_{ij} \leq f_j + 2\times\lambda\times(1-y_{ij}) & \forall_{\langle i,j\rangle\in E} & (1.3)\\
 & y_{ij} + y_{ji} \leq 1 & \forall_{\{i,j\}\in L} & (1.4)\\
 & \sum_{\langle i,j\rangle\in E} y_{ij} \geq |N|-1 & & (1.5)\\
 & \sum_{\langle i,j\rangle\in E, i=n\vee j=n} y_{ij} \geq 1 & \forall_{n\in N} & (1.6)\\
 & \sum_{e\in\mathtt{In}(s(\rho))} x_{\rho e} = 0, \sum_{e\in\mathtt{Out}(s(\rho))} x_{\rho e} = 1 & \forall_{\rho\in\mathcal{P}} & (1.7)\\
 & \sum_{e\in\mathtt{Out}(t(\rho))} x_{\rho e} = 0, \sum_{e\in\mathtt{In}(t(\rho))} x_{\rho e} = 1 & \forall_{\rho\in\mathcal{P}} & (1.8)\\
 & \sum_{e\in\mathtt{In}(i)} x_{\rho e} = \sum_{e\in\mathtt{Out}(i)} x_{\rho e} & \forall_{\rho\in\mathcal{P}}\forall_{i\in N\setminus\{s(\rho),t(\rho)\}} & (1.9)\\
 & \sum_{e\equiv\langle i,j\rangle\in E} d_{ij}\times x_{\rho e} \leq \lambda & \forall_{\rho\in\mathcal{P}} & (1.10)\\
 & y_{ij} + y_{ji} \geq x_{\rho e} & \forall_{\rho\in\mathcal{P}}\forall_{e\equiv\langle i,j\rangle\in E} & (1.11)
\end{array}
$$

– y_{ij}: a binary variable that if equal to 1 implies that the link $\{i,j\} \in L$ is included in the solution. Please note that there are two y variables associated with the link $\{i,j\}$: y_{ij} and y_{ji}. Such link is included in the solution if and only if one of them is set to 1.
– $x_{\rho e}$: a binary variable that is equal to 1 if and only if an edge e is used for connecting pair $\rho \in \mathcal{P}$.
– f_i: an auxiliary real variable that is used for each node i to represent the distance from κ to node i. f_κ is zero since κ is the source node. In the other cases, the domain is $\{d_{\kappa i}, \ldots, \lambda\}$ because the distance cannot be smaller than the one obtained when we connect node i directly to the source node, under the assumption that it is always possible to connect any node directly to the source. Here we are also assuming that triangular inequality holds.

Constraints The set of constraints is divided into two groups. The first group of constraints ensure that the computed graph is connected (Constraints (1.1)–(1.5)). The second group of constraints enforce that the paths connecting the pairs of nodes are well formed (Constraints (1.6)–(1.11)).

– The incoming degree of the source node is zero (Constraint 1.1).
– Any non-source node must have at least one incoming edge (Constraint 1.2).
– If a directed edge $\langle i,j\rangle \in E$ is selected, then the length of the path from κ to node j, f_j, must be greater than or equal to the length of the path from κ to i, f_i, plus d_{ij} (Constraint 1.3). If $\langle i,j\rangle \in E$ is not selected then we do not want to enforce any constraint between f_i and f_j. As f_i, f_j and all d_{ij} are bounded by λ, a valid upper bound for $f_i + d_{ij}$ is $2\times\lambda$. Therefore, we use $2\times\lambda$ as the big-M value in this constraint. This constraint is providing a valid inequality to tighter the formulation.
– An acyclic graph cannot have cycles, so only one of the two directed edges associated with a link is selected (Constraint 1.4).

- The number of selected edges in a consistent graph is greater than or equal to the number of nodes minus one (Constraint 1.5).
- For any node, the total number of incoming and outgoing edges must be at least 1 to ensure connectivity (Constraint 1.6).
- For each pair of nodes in $\mathcal{P}$, there exists a path connecting the nodes. No path reaches the source or leaves the target of the pair and only one edge leaves (resp. reaches) the source (resp. the target) (Constraints 1.7 and 1.8).
- The incoming and outgoing degrees of an intermediate node are equal (Constraint 1.9).
- The length of the path connecting a pair of nodes cannot exceed the threshold (Constraint 1.10).
- A link between any pair of nodes i and j is selected if any of the corresponding directed edges is used by a path connecting a pair of nodes in $\mathcal{P}$ (Constraint 1.11).

Objective As shown in Table 1, the objective function considered is given by $\alpha \times TLL + TLP$. In this definition, α is a constant that is greater than the weighted total length of the paths connecting the pairs of nodes, such that TLL is minimised first. In the MILP model, TLL and TLP are expressed as $TLL = \sum_{\langle i,j\rangle \in E} d_{ij} \times y_{ij}$ and $TLP = \sum_{\rho \in \mathcal{P}} \beta_\rho \left(\sum_{e \equiv \langle i,j\rangle \in E} d_{ij} \times x_{\rho e} \right)$. Recall that β_ρ denotes the amount of commodity flowing from the source node to the target node of ρ, which leads to reduce the cost of deploying required infrastructure such as fibre cables and amplifiers in optical networks. If the amount of commodity between a pair of nodes is not given or negligible with respect to the total length of the links, then we can set β_ρ to one for all the pairs of nodes. We can also chose to minimise the sum of the lengths of the selected links and the average path length by setting the values of α to 1 and β_ρ to $1/|N|^2$.

3.2.1 Size of the model

In our mathematical model we have $2 \times (|N| \times (|N| - 1)) \times |L|$ $x_{\rho e}$ variables, $2 \times |L|$ y_{ij} variables, and $|N|$ f_i variables. However, the size of the mathematical model is actually dominated by the number of linear equations, which depends on the number of nodes ($|N|$) and links ($|L|$) only since we are assuming that we have to connect all pair of nodes (i.e., $|\mathcal{P}| = |N| \times (|N| - 1)$) and $|E| = 2 \times |L|$.

It is clear from the universal quantification of the constraints that Constraints (9) and (11) are the ones contributing the most to the total number of linear equations. In fact, for those cases where the graph is dense (i.e., $|E| \simeq |N| \times (|N| - 1)$), Constraint (11) is the greatest contributor since there are $\mathcal{O}(|N|^4)$ linear equations coming from that constraint alone, so the number of linear equations in this mathematical model is $\mathcal{O}(|N|^4)$.

3.3 The CP model

In the last decades, the constraint programming community put a lot of effort in the definition of global constraints that are specifically focused on graphs [12]. The idea is to model network problems by using global constraints that exploit powerful graph-based filtering algorithms. In the case of the DCND, a path from any node $k \in N$ to all other nodes in N has to be enforced in order to guarantee connectivity among couples of nodes. This is equivalent to enforcing that, for each $k \in N$, there is a directed tree $tree_k$, rooted in k and connecting k to all other nodes. Furthermore, we need to impose the diameter constraint, meaning that the length of each path from the root to a leaf has to be bounded by λ in each of the above mentioned trees. These properties can be imposed on these trees by means of a *PathCumulative* constraint [7]. Given a graph G a node k of G, a tree $tree_k$ and an integer value λ, it imposes that the condition above is verified. Finally, a solution of the problem is represented by an undirected graph where a link $\{i, j\} \in L$ is taken if at least one of the edges $< i, j >$ and $< j, i >$ belongs to one of the trees $tree_k$.

In the following, we describe variables, constraints and the objective function required for formulating the problem. Please refer to Table 2, for the complete CP model.

Variables that are considered in the CP model:

- Z is an $|N| \times |N|$ symmetric matrix whose components are the binary variables $z_{i,j}$ indicating if the link $\{i, j\} \in L$ is taken in the solution or not.
- $x_{\rho e}$ follows the same definition used in the MILP model.
- $tree_i$ is a λ bounded directed spanning tree of G routed in $i \in N$.
- Γ_i is an $|N| \times |N|$ matrix representing the adjacency matrix of the directed tree $tree_i$. Please note that this matrix is not symmetric because $tree_i$ is a directed graph. The components of Γ_i are binary variables indicated with $\gamma^i_{j,k}$.
- $treecost_i$ is the sum of the length of the edges in $tree_i$.
- $tree_{i,j}$ is the predecessor of the node $j \in N$ in $tree_i$, with $tree_{i,i}$ set to i.

Constraints The constraints indicated in (2.1) are ensuring both connectivity and the diameter-bounded constraint, while all the other constraints are used to express the objective functions and to link the variables.

- Any node is connected to any other node by means of a λ-bounded path. This is done with $|N|$ *PathCumulative* constraints, generating a tree $tree_i$ rooted in each node $i \in N$, whose longest path between i and the other nodes is bounded by λ (Constraint 2.1).
- The edges taken in the solution are the union of all the edges considered in each tree $tree_i$. This is done by means of an element constraint assigning 1 to all the variables associated with the edges considered in each tree (Constraint 2.2), since $tree_{i,j}$ is the predecessor of j in the tree $tree_i$.

Table 2 The CP model proposed

$$
\begin{array}{llll}
\text{Minimise} & \alpha \times TLL + TLP & & \\
\text{Subject to} & PathCumulative(G, i, tree_i, \lambda) & \forall_{i \in N} & (2.1) \\
 & z_{j,tree_{i,j}} = 1 & \forall_{i \neq j \in N} & (2.2) \\
 & \gamma^i_{j,tree_{i,j}} = 1 & \forall_{i \neq j \in N} & (2.3) \\
 & \gamma^i_{j,k} = 1 \text{ iff } tree_{i,j} = k & \forall_{i \neq j \neq k \in N} & (2.4) \\
 & Sum(Z) \geq |N| - 1 & & (2.5) \\
 & Sum(\Gamma_i) = |N| - 1 & \forall_{i \in N} & (2.6) \\
 & Sum(z_{i,j}\ (i,j) \in E) \geq 1 & \forall_{i \in N} & (2.7) \\
 & Sum(z_{i,j}\ (i,j) \in E) \geq 1 & \forall_{j \in N} & (2.8) \\
 & treecost_i = Scalar(D, \Gamma_i) & \forall_{i \in N} & (2.9) \\
 & treecost_i \leq TLL & \forall_{i \in N} & (2.10) \\
 & x_{\rho e} = 1 \text{ iff } e \text{ in the path from } s(\rho) \text{ to } t(\rho) \text{ in } tree_{s(\rho)} & \forall_{e \in E} \forall_{\rho \in \mathcal{P}} & (2.11)
\end{array}
$$

– Constraint 2.3 and Constraint 2.4 are linking constraints between the components of the adjacency matrices Γ_i and the directed trees $tree_i$.
– The minimum number of edges to ensure connectivity is $|N|-1$ (Constraint 2.5), while the number of edges in each directed tree $tree_i$ is exactly $|N|-1$ (Constraint 2.6). These constraints are imposed by means of the Sum global constraints, computing the summation of the components of the matrix indicated as an argument.
– Each node of the graph has at least one in-going edge and one out-going edge (Constraint 2.7 and Constraint 2.8). Here Sum indicates the sum of the variables included as arguments.
– The cost of each tree $treecost_i$ is formulated by means of the $Scalar$ global constraints, which computes the Frobenius inner product between the distance matrix D and the adjacency matrix Γ_i (Constraint 2.9).
– The total length of links TLL is lower bounded by the sum of the lengths of the edges considered in a each tree, indicated with $treecost_i$ (Constraint 2.10).
– An edge e is included in $tree_{s(\rho)}$ if and only if a certain edge e is included in the path from $s(\rho)$ to $t(\rho)$ (Constraint 2.11).

Objective The objective considered is the same of the MILP model. As for the cost of the trees $treecost_i$, TLL and TLP are expressed via the $Scalar$ global constraint. TLL can be expressed as $TLL = \frac{1}{2} Scalar(D, Z)$. In order to define TLP, we indicate with X_ρ an $|N| \times |N|$ matrix whose $< i, j > -th$ components are the variables $x_{\rho,<i,j>}$ if $< i, j > \in E$, 0 otherwise. The equality $TLP = Sum(\beta_\rho \times Scalar(D, X_\rho) : \rho \in \mathcal{P})$ holds.

3.3.1 Size of the model

Similarly to the MILP model, we have $(|N| \times (|N| - 1)) \times |L|$ $x_{\rho e}$ variables, $2 \times |L|$ z_{ij} variables, plus $|N|$ trees $tree_i$ (and their adjacency matrices) for each $i \in N$. Regarding the constraints, Constraints (2.11) clearly include $\mathcal{O}(|N|^4)$, similarly to Constraints (1.11) of the MILP model. However, the highest effort in terms of propagation is given by the $|N|$ constraints $PathCumulative$ (G, i, $tree_i$, λ), which ensures both connectivity and the fact that the diameter is bounded by λ.

4 Adaptive Large Neighbourhood Search Method

The models defined in the previous section do not scale well and it is challenging to solve even for small size instances with $|N| \leq 20$ as they are intractable [37]. The reader is kindly referred to Section 5.1 for a computational assessment of the exact approaches previously described. Although there are branch-and-cut, and/or column generation methods for considerably small instances of network design problems [23], the problem becomes highly challenging when the distance bound exists between pairs of nodes. As pointed out in a recent study [5], the current literature provides only approximation schemes with a reasonable error level. Hence, we propose an adaptive Large Neighbourhood Search (LNS) [32] approach for solving the DCND problem. The presented approach is scalable for large size instances and effective in terms of the solution quality.

LNS is a meta-heuristic method which attempts to combine the power of systematic search with the scalability of local search. LNS is applied to many kind of combinatorial optimisation problems such as vehicle routing [3, 1], scheduling [18], machine reassignment [22], lot-sizing [28] and network design [13, 4]. The overall LNS approach is shown in Figure 1. We first find an initial solution. We maintain the current solution, which is first set to the initial solution. At each iteration of LNS, we select and create a subproblem by selecting a subset of the pairs of MC nodes for which we want to recompute the routes to connect them. We solve the resulting sub-problem, and keep the best solution found during search.

Note that changing the size of the subproblems during the neighbourhood search helps to escape from local optimal solutions. Such an adaptive search is applied in many problems like the number of processes selected to be re-assigned [22] or the number of customers to be re-located [1]. The search stops when the total elapsed time is greater than the given time threshold or the desired optimality gap is achieved. The lower bound for the total length of the links is the minimum spanning tree since it is independent from the amount of commodity transferred between pair of nodes and hence valid for all instances of the DCND problem. The lower bound for the total length of the paths is computed as the sum of the shortest paths between all pairs of MC nodes.

In the following, we describe all these steps in detail.

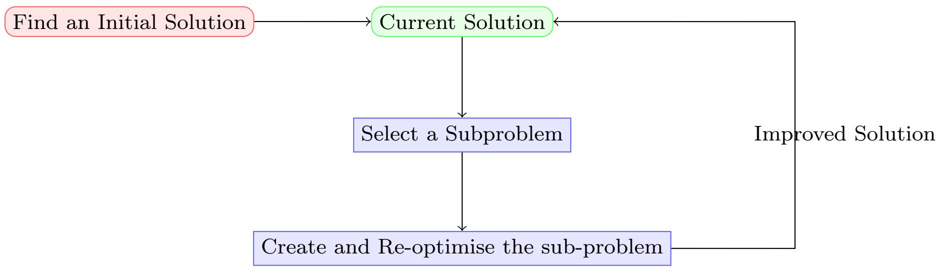


Fig. 1 Principle of the LNS approach.

4.1 Initial solution

As shown in 5.1, since the presented MILP model is performing better than CP a decomposition approach exploiting the MILP model is used to find an initial solution. However, other heuristic approaches are also evaluated as presented in section 5.2.5 The problem of constructing a diameter constrained network design is decomposed into a sequence of subproblems which are solved iteratively. In each iteration of the decomposition scheme, the objective is to find a bounded path consisting of the cheapest (in terms of the total length of the links and the path) set of links to connect a given pair of nodes.

Algorithm 1 FINDSOLUTION($\langle N, L\rangle,\ S,\ L^b,\ \lambda,\ k,\ B$)

```
1:  $L^o \leftarrow L^b$
2:  $S \leftarrow S \setminus \{\rho | \rho \in S, \textsc{BoundedShortestPath}(\rho, L^o, \lambda) \neq \emptyset\}$
3:  while $S \neq \emptyset$ do
4:      Select and remove a subset of pairs $K \subseteq S$ such that $|K| = k$
5:      $L^t \leftarrow \textsc{BoundedCheapestPath}(K, \langle N, L\rangle,\ L^o,\ \lambda)$
6:      if $cost(L^t) < B$ then
7:          $L^o \leftarrow L^t$
8:          $S \leftarrow S \setminus \{\rho | \rho \in S, \textsc{BoundedShortestPath}(\rho, L^o, \lambda) \neq \emptyset\}$
9:      else
10:         $L^o \leftarrow L$
11:         break
12:     end if
13: end while
14: return $L^o$
```

The heuristic-based decomposition approach for finding a solution is presented in Algorithm 1. Given an input graph $\langle N, L\rangle$, a subset of links $L^b \subseteq L$, a set of pairs of nodes S, the upper bound on the diameter λ, an integer constant k and and a real constant B, the algorithm returns a set of links L^o such that $L^o \supseteq L^b$. It decomposes the set of pairs of nodes S in to subsets of size up to k and tries to find a path for each pair of nodes in the set S using the links L^o such that its length is bounded by λ, and the sum of the lengths of the links in L^o is at most B. To find an initial solution the algorithm is invoked as follows: FindSolution($\langle N, L\rangle$, $\mathcal{P}$, $\emptyset$, λ, k, ∞). Here S is initialised to the set $\mathcal{P}$ which denotes the set of all pairs of nodes and L^b is initialised to an empty set for finding an initial solution.

First L^o is initialised to L^b, and then for each pair of MC nodes, ρ, a polynomial check is performed to verify whether the bounded path can be found in L^o without adding any link. This is done by invoking BoundedShortestPath which returns the shortest path between $S(\rho)$ and $T(\rho)$ if the length of the path is not greater than λ, otherwise it returns an empty set. The path is denoted by a set of links. If the polynomial check fails then the MILP model given in the previous section is solved for the k pairs of nodes by invoking BoundedCheapestPath. The MILP model is solved by enforcing an additional constraint that the links in L^o must be included. The objective function is reformulated as:

$$\min \alpha \times \sum_{\{i,j\}\in L\setminus L^o} d_{ij} \times (y_{ij} + y_{ji}) + \sum_{\rho\in K} \beta_\rho \left(\sum_{e\equiv\langle i,j\rangle\in E} d_{ij} \times x_{\rho e} \right) \quad (1)$$

When it comes to selecting ρ from the set S we consider three criterion: random order (`random`), non-decreasing (`nondec`) and non-increasing order (`noninc`) based on the lengths of the shortest paths between pairs of nodes. The impact of these criteria is studied later in the paper. Although this procedure reduces the number of links substantially, the resulting number of links and the total length of the links will be sub-optimal because of the myopic nature of the algorithm. In other words, a link added in the previous iterations may become redundant when new links added in the next iterations. Therefore, neighbourhood search is used to improve the initial solution further by repeatedly selecting and solving subproblems as described in the next sections.

The heuristic-based decomposition presented in Algorithm 1 can be parameterised in many ways. For example, to find an initial solution instead of setting L^b to an empty set of links, it can be initialised with the links in the minimum spanning tree of the graph $\langle N, L\rangle$. The advantage is that the connectivity constraint would be entailed. Consequently, when Algorithm 1 is called, it would be possible to have already a bounded path for many pairs of nodes. Therefore, BoundedCheapestPath might be invoked for fewer pairs and the solution time can be extensively reduced.

Furthermore, instead of solving a modified MILP model in every call of BoundedCheapestPath, a MILP-free approach can also be used. More precisely, a greedy algorithm can be used to compute a set of links for each pair of nodes in the set K. In every invocation of BoundedCheapestPath, (i) order the set of pairs of nodes of the set K based on the lengths of their shortest-paths between them in the input graph, (ii) select and remove the first pair of nodes from the ordered set K, (iii) add one or more links involved in the shortest path to the set L^b, (iv) remove any pairs of nodes from the set K which have now bounded paths, (v) repeat steps (i) to (iv) until the set K is empty. We have evaluated these alternative ways for finding an initial solution and the results are shown in the experimental section.

4.2 Subproblem Selection

In the context of designing a diameter constrained network, a subproblem can be seen as a subset of the pairs of nodes for which we want to destroy and repair their corresponding bounded paths in the hope of improving overall cost of the solution.

To select a subproblem, we first select one or more links of the current solution and then select those pairs of nodes whose corresponding bounded paths rely on any of these selected links. The goal is to select a subproblem such that it improves the cost of the current solution. The hope is that one or more selected pairs of nodes might find alternate paths to improve the overall cost. We remark that the number of pairs of nodes relying on a single link can vary. Therefore, the size of the subproblem can vary during the search, too. Moreover, we also increase the number of links that are selected as search progresses. Hence, these two aspects of the algorithm make it adaptive in the sense of dynamically changing the subproblem sizes.

We propose two different ways of selecting links:

– **Minimum Support.** The first selection approach is called *minimum support* (`minsup`). Let $path_\rho$ be the set of links used in the current bounded path for connecting the end nodes of ρ. Let $support(l)$ be the set of pairs of nodes whose paths are using link l:

$$support(l) = \{\rho \in \mathcal{P} | l \in path_\rho\}.$$

The intuition here is that a link supporting fewer pairs of nodes might be easier to remove or replace. The worst-case complexity of computing support values for all links is $\mathcal{O}(n^3)$. The reason is that the number of links is bounded by n^2 and any path between any pair of nodes cannot contain more than $n - 1$ links.

– **Minimum Link Graph.** The second selection approach is called *minimum link graph* (`minlgraph`). It is recalled that each pair of nodes is associated with a bounded path consisting of a set of links that are used to connect the end nodes. Let $linkgraph(l)$ denotes the graph consisting of the links associated with the bounded paths of the pairs of nodes supported by the link l:

$$linkgraph(l) = \bigcup_{\rho \in support(l)} path_\rho$$

When we select and remove a link for a subproblem selection, the entire path associated with each pair of nodes supported by the removed link is affected. If the graph associated with all the affected paths has many links then it might be more difficult to find alternative paths while improving the cost of the solution. Therefore, it makes sense to select a set of links of a given cardinality such that the size of the graph obtained by projecting the routes of all the affected pairs of nodes is minimum. The worst-case complexity of computing the size of the link-graph is also $\mathcal{O}(n^3)$. Here instead of counting the number of paths where a link occurs, the union of the sets of all links associated with these path is computed.

In the following, we provide a numerical example to demonstrate how to compute `minsup` and `minlgraph`. Figure 2 shows a network with 5 nodes and 5 links to connect 10 pairs.

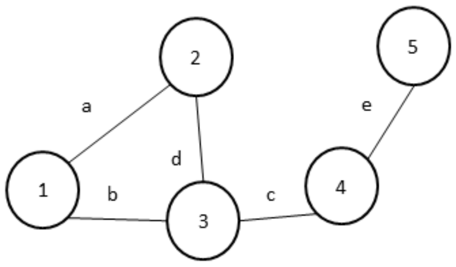


Fig. 2 A network to demonstrate `minsup` and `minlgraph` operators

The paths between every pair of nodes are listed in Table 3. The *support*(l) for each link, l, is shown in Table 4. The best link based on `minsup` is a since it has the minimum number of paths relying on it.

Table 3 Paths between all pairs of nodes in the sample network given in Figure 2

ρ	$\langle 1,2\rangle$	$\langle 1,3\rangle$	$\langle 1,4\rangle$	$\langle 1,5\rangle$	$\langle 2,3\rangle$	$\langle 2,4\rangle$	$\langle 2,5\rangle$	$\langle 3,4\rangle$	$\langle 3,5\rangle$	$\langle 4,5\rangle$
$path_\rho$	{a}	{b}	{b,c}	{b,c,e}	{d}	{d,c}	{d,c,e}	{c}	{c,e}	{e}

Table 4 *support*(l) computation for the network given in Figure 2

Link (l)	*support*(l)	$\lvert support(l)\rvert$
a	$\{\langle 1,2\rangle\}$	1
b	$\{\langle 1,3\rangle,\langle 1,4\rangle,\langle 1,5\rangle\}$	3
d	$\{\langle 2,3\rangle,\langle 2,4\rangle,\langle 2,5\rangle\}$	3
e	$\{\langle 1,5\rangle,\langle 2,5\rangle,\langle 3,5\rangle,\langle 4,5\rangle\}$	4
c	$\{\langle 1,4\rangle,\langle 1,5\rangle,\langle 2,4\rangle,\langle 2,5\rangle,\langle 3,4\rangle,\langle 3,5\rangle\}$	6

In Table 5 presents the *linkgraph*(l) for each link, l. For example, link b occurs in the paths of pairs $\langle 1,3\rangle$, $\langle 1,4\rangle$ and $\langle 1,5\rangle$. The union of the sets of all the links occurring in these paths constitute *linkgraph*(l) which is {b,c,e}.

Based on the `minlgraph` approach, the link with the smallest size of graph associated with projected routes of all the affected pairs is link a.

Table 5 *linkgraph*(l) computation for the network given in Figure 2

Link (l)	*linhgraph*(l)	$\|linkgraph(l)\|$
a	$path_{\langle 1,2\rangle} = \{a\}$	1
b	$path_{\langle 1,3\rangle} \cup path_{\langle 1,4\rangle} \cup path_{\langle 1,5\rangle} = \{b, c, e\}$	3
d	$path_{\langle 2,3\rangle} \cup path_{\langle 2,4\rangle} \cup path_{\langle 2,5\rangle} = \{d, c, e\}$	3
c	$path_{\langle 1,4\rangle} \cup path_{\langle 1,5\rangle} \cup path_{\langle 2,4\rangle} \cup path_{\langle 2,5\rangle} \cup path_{\langle 3,4\rangle} \cup path_{\langle 3,5\rangle} = \{b, c, e, d\}$	4
e	$path_{\langle 1,5\rangle} \cup path_{\langle 2,5\rangle} \cup path_{\langle 3,5\rangle} \cup path_{\langle 4,5\rangle} = \{b, c, e, d\}$	4

4.3 Subproblem Solving

A solution of DCND is represented by a set of links. An initial solution might not be minimal in the sense that if one or more links are removed from the current solution then all the pairs of nodes can still find bounded paths using the remaining links of the current solution. For some subproblems especially in the beginning of the LNS search, it might be possible that when a set of pairs of nodes is selected one or more links are removed from the current solution without adding any links to it. In this case, there is no need to create and solve a MILP model. We instead use all-pairs shortest path [9] for checking bounded connectivity, which is a polynomial algorithm and it speeds up the solving time. Thus, in the context of LNS when a solution is destroyed by removing one or more links from the current solution, we apply all-pairs shortest path and create the MILP model only if the output of all-pairs shortest path fails to satisfy the diameter constraint.

Algorithm 2 $\textsc{SolveDCND}(\langle N, L\rangle,\ \mathcal{P},\ \lambda,\ k)$

```
1: $L^b \leftarrow \textsc{FindSolution}(\langle N, L\rangle,\ \mathcal{P}, \emptyset, \lambda,\ k,\ \infty)$
2: $L^b \leftarrow \{l | l \in L^b, \exists \rho \in \mathcal{P}, \textsc{shortestpath}(\rho, L^b \setminus \{l\}) > \lambda\}$
3: $\mathcal{L} \leftarrow 2^{L^b} - \{\emptyset\}$
4: while stopping condition is not true do
5:     select and remove $R$ from $\mathcal{L}$
6:     $S \leftarrow \bigcup_{l \in R} \textsc{support}(l)$
7:     $L^c \leftarrow \textsc{FindSolution}(\langle N, L\rangle, S, L^b - R, \lambda, k, cost(L^b))$
8:     if $cost(L^c) < cost(L^b)$ then
9:         $L^b \leftarrow L^c$
10:        $\mathcal{L} \leftarrow 2^{L^b} - \{\emptyset\}$
11:    end if
12: end while
13: Return $L^b$
```

The pseudo-code of the adaptive large neighbourhood search algorithm is given in Algorithm 2. In this algorithm, we start by finding an initial solution (Line 1). We then compute a minimal solution by discarding from L^b a subset of links that are redundant (Line 2).

Afterwards, the neighbourhood search is performed (Lines 3–13). First the power set of the non-redundant links ($\mathcal{L}$) is computed. An element of the power set is selected based on one of the criteria introduced (e.g., `minsup` or `minlgraph`) (Line 5). Furthermore, in each iteration of LNS when the current solution is destroyed by removing even a single link, more than one pair of nodes might be affected. Thus Algorithm 2 adapts the size of the subproblem, i.e., $|S|$ by varying this dynamically during the search (Line 6). The subproblem where one needs to find bounded paths even for just 100 pairs of nodes could be very large to be handled by the MILP based systematic search in a reasonable time. We therefore again use a heuristic-based decomposition approach similar to the one used for finding an initial solution (Line 7) by invoking $\textsc{FindSolution}(\langle N, L\rangle,\ L^b - R,\ S,\ \lambda,\ k,\ cost(L^b))$. If the cost of the newly found solution L^c is better than the current best solution then L^b is updated, and the set of subproblems that can be explored is reset (Lines 8–10). We remark that the power-set of the subproblems is not generated in practice as subproblems are created on demand.

5 Computational experiments

This section provides a computational assessment of the models and algorithms described previously. It is divided in two subsections. Section 5.1 is devoted to evaluate the performances of the exact approaches (the MILP the CP models presented in Section 3.2 and Section 3.3) on small size randomly generated instances. Section 5.2 focuses on analyzing the performances of the LNS approach on larger instances arising in a telecommunication network design application.

5.1 Computational assessment of the exact approaches

In this section, some preliminary experiments related to the CP and the MILP formulations are presented. The aim of this analysis is twofold. On the one hand, we want to show that both methods are not scalable enough to solve real world instances. On the other hand, we want to determine the most efficient formulation to be used in repairing phase of the LNS algorithm. Since here the focus is on the size of the instances that can be solved through exact methods, we generate challenging instances with a simple procedure whose outcome is a complete graph, given the input number of nodes $|N|$. The first step of the procedure is to generate $|N|$ random points, by randomly assigning their coordinates such that they are uniformly distributed integers between 1 and 100. A node is associated with each of the points generated. The length of each edge $< i, j > \in E$ is then computed by considering the distances between the points associated to node i and node j. Finally, the diameter λ is set to the largest of the shortest paths between any couple of nodes in the graph. In this way, we guarantee that the complete graph is always a feasible solution.

For the sake of simplicity, both formulations are simplified by considering only the first objective that is minimizing the total length of the links in the resulting network. The search procedure used in the CP model is based on branching on the components of the matrices Γ_i with $i = 1, ..., |N|$. The value ordering heuristic chosen is branching on the shortest edge available. Both models were tested on an AMD A9-9410 by using Google-Or-Tools version 7.5.7466 and IBM ILOG CPLEX Optimisation Studio 12.6.0 as CP and MILP solvers, respectively. A time limit of 1200 seconds was set in both solvers.

The results obtained are shown in Table 6. The first three columns of such table describe the instance considered by including the ID of the instance, the number of nodes $|N|$, and the diameter considered. Please note that 3 different instances are generated for each value of $|N|$. The next two columns provide the best TLL found by the CP solver (indicated with Obj) and the computational time in seconds (indicated with Time(s)). The last three columns are related to the results obtained with the MILP model. They are related to the best objective found, the computational time in seconds and the percentage optimality gap. Each different row is referring to a different instance and the results reported in bold are identifying the cases where the optimality is proven. The results reported show that the instances generated are pretty challenging for both models. The CP model is able to solve only one instance ($ID = 2$) to optimality within the time limit. When $|N| \geq 13$, the CP model does not converge to a feasible solution. The MILP model proposed outperforms the CP model, by solving to optimality all the instances with $|N| \leq 13$ and 2 out of 3 instance with $|N| = 14$. However, the MILP model was not able to solve to optimality any of the instances with $|N| \geq 14$ within the time limit.This is partially due to the very high time required to generate the model with its variables. Since the MILP model outperforms the CP model, we considered MILP-based subroutines in the proposed LNS approach while repairing solutions.

Table 6 The results obtained by running the MILP and the CP solver

Instance			**CP**		**MILP**		
ID	$\lvert N \rvert$	λ	**Obj**	**Time(s)**	**Obj**	**Time(s)**	**Gap(%)**
1	8	101.98	325.53	1200.00	**325.53**	**4.80**	0.00
2	8	96.66	**458.83**	**22.23**	**458.83**	**0.77**	0.00
3	8	81.04	388.60	1200.00	**331.28**	**2.13**	0.00
4	9	103.77	614.38	1200.00	**532.62**	**5.74**	0.00
5	9	92.41	684.66	1200.00	**604.90**	**1.02**	0.00
6	9	82.73	397.40	1200.00	**386.81**	**12.42**	0.00
7	10	105.41	798.45	1200.00	**547.11**	**8.34**	0.00
8	10	98.18	886.11	1200.00	**543.91**	**15.27**	0.00
9	10	89.28	968.83	1200.00	**602.43**	**9.48**	0.00
10	11	105.41	924.88	1200.00	**547.52**	**122.49**	0.00
11	11	98.18	753.79	1200.00	**546.52**	**36.04**	0.00
12	11	113.14	1006.05	1200.00	**519.13**	**6.38**	0.00
13	12	105.41	846.18	1200.00	**522.60**	**234.73**	0.00
14	12	98.18	1275.49	1200.00	**634.87**	**32.43**	0.00
15	12	113.14	1257.77	1200.00	**641.16**	**85.83**	0.00
16	13	105.41	No sol. found	1200.00	**555.35**	**262.87**	0.00
17	13	98.18	1044.58	1200.00	**493.43**	**452.98**	0.00
18	13	113.14	No sol. found	1200.00	**659.93**	**661.83**	0.00
19	14	105.41	No sol. found	1200.00	**576.44**	**936.39**	0.00
20	14	98.18	1187.59	1200.00	413.60	1200.00	6.36
21	14	113.14	No sol. found	1200.00	**719.69**	**618.24**	0.00
22	15	105.41	No sol. found	1200.00	611.29	1200.00	9.87
23	15	109.01	1134.99	1200.00	423.66	1200.00	11.87
24	15	113.14	No sol. found	1200.00	643.39	1200.00	21.91
25	16	109.20	No sol. found	1200.00	543.91	1200.00	17.01
26	16	89.28	No sol. found	1200.00	558.35	1200.00	8.48
27	16	113.89	1524.69	1200.00	631.60	1200.00	24.91
28	17	109.20	No sol. found	1200.00	530.13	1200.00	16.79
29	17	105.11	No sol. found	1200.00	497.23	1200.00	27.97
30	17	113.89	No sol. found	1200.00	569.90	1200.00	26.29
31	18	109.20	No sol. found	1200.00	547.17	1200.00	19.51
32	18	113.14	No sol. found	1200.00	685.98	1200.00	25.69
33	18	94.64	No sol. found	1200.00	665.36	1200.00	29.26
34	19	109.20	No sol. found	1200.00	536.21	1200.00	18.75
35	19	113.14	No sol. found	1200.00	660.21	1200.00	26.81
36	19	112.60	2113.70	1200.00	504.39	1200.00	22.23
37	20	109.20	No sol. found	1200.00	577.56	1200.00	18.20
38	20	113.14	No sol. found	1200.00	9853.18	1200.00	100.00
39	20	112.60	No sol. found	1200.00	8533.35	1200.00	95.45

5.2 Computational assessment of the LNS approach

In this section, we present experimental results obtained using adaptive large neighborhood search (LNS) on different national network topologies. All the algorithms are implemented using IBM ILOG CPLEX Optimisation Studio 12.6.0, which provides the OPL Script language for integrated algorithm development and communicating with CPLEX solver. All test instances were run on a PC with an Intel(R) Core (TM) i7-4600U 2.10 GHz CPU, 8 GB RAM and 64-bit Microsoft Windows 7 operating system.

As test instances, national network topologies of Ireland, UK and Italy are derived from DISCUS project [35] which aims to develop a transparent optical core network, TOCN (also referred as optical island and defined as a set of Metro-Core (MC)) that can provide high-speed broadband capability while reducing energy consumption and remains economically viable.

The advantage of a TOCN is the absence of Optical-Electrical-Optical conversions and the reduction in switching, routing and packet processing in the MC nodes. The disadvantage is that as the number of MC nodes, denoted by $|N|$, grows, the number of physical links can grow quickly and in the worst-case it could be $|N|(|N|-1)/2$. Therefore, it is important to design a network by minimising the total length of links, while guaranteeing that the distance between any pair of MC nodes is within a given threshold in order to ensure transparency. Another important cost factor in the design of a TOCN is the amount of fibers deployed, which is mostly related with the optical signals used to carry lightpaths. Optical signals with higher capacity allow to carry the same amount of traffic with less number of lightpaths and hence consume less total slots in fibers. Therefore, using higher capacity optical signals reduces the length and the cost of fibers deployed [25]. However, the reach of these signals is limited [10], so minimising the total length between each pair of metro-core nodes is considered as a secondary objective in the design of a TOCN. Overall, the design of a TOCN is an optimisation problem where the cost of a nation-wide core network is typically dominated by the connection cost (i.e., fibre deployment, placement of optical amplifiers at regular intervals etc.).

In fact, the design of a TOCN is an instance of the DCND problem, where the diameter of the network consisting of MC nodes must be less than a given threshold. A part of TOCN for Italy with 132 MC nodes is illustrated in Figure 3 [24].

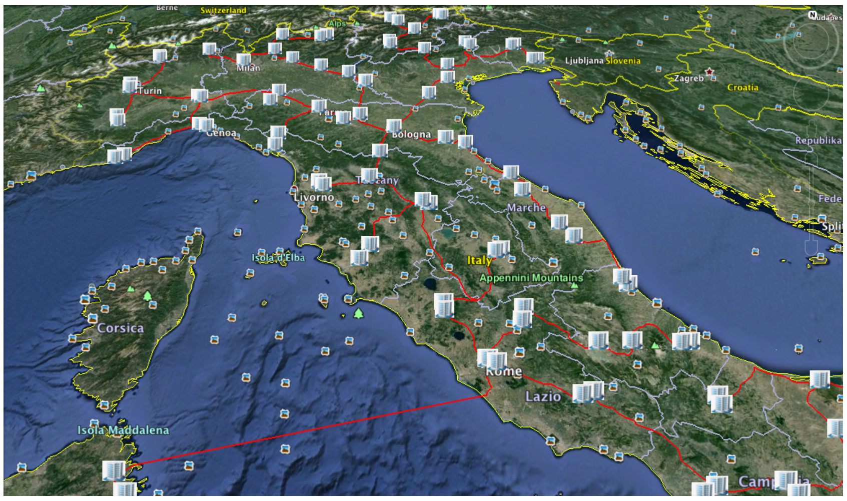


Fig. 3 Resulting Italy network with 132 metro-nodes

The instances of Ireland are small as they only contain from 18 to 24 MC nodes. However, the instances of UK ranges from 74 to 99, and the ones of Italy from 132 to 189 MC nodes. These instances are available in CSPLib [6], Problem 71. The LNS algorithm terminates when the upper bound on the size of the network is less than 1% away from the lower bound, considering a timeout of 5 hours. The results are presented in terms of Total Length of Links (TLLs), Total Length of Paths (TLPs), solution time in seconds (time) and the diameter of the network. The value of λ, which is the maximum diameter, is set to 2430KMs [25].

In order to design a transparent optical core network one option is to connect all pairs of MC nodes using shortest paths between them. This solution is referred as **Initial Design Network** (IDN). The advantage of IDN is that the total length of the paths between all pairs of MC nodes would be minimum but the disadvantage is that the total length of links is going to be very high. In the worst-case, each pair of MC nodes is connected directly. Another possibility is to design a network by selecting only a subset of the links L such that the total length of the links is minimised and the distance between any pair of MC nodes does not exceed the maximum optical signal reach which is obtained by running Algorithm 2.

In the following subsections, we first provide lower bounds for both the size of the networks and the total length of the paths. Then, we discuss heuristics for finding an initial solution. Next, different subproblem selection heuristics are discussed. We continue with the trade-off between the size of the subproblems and the solution effort to find the best subproblem size. Finally, we investigate different ways of finding initial solution and solving subproblems to improve the performance of developed LNS algorithm .

5.2.1 Lower Bounds

In this section we present the lower bounds so that the quality of the solutions obtained by the proposed approach can be evaluated. The lower bounds for the total length of links are obtained by simply computing Minimum Spanning Tree (MST) solutions. The lower bounds for the total length of paths between all pairs of nodes are obtained by summing the shortest paths between all pairs of nodes based on the initial design networks. The values of the lower bounds for the Total Length of Links (TLL) and the Total Length of Paths (TLP) are shown in Table 7.

The diameter of the initial networks and minimum spanning tree networks are also presented in the table. Note that if the diameter of IDN is greater than λ then the problem instance is unsatisfiable. It is recalled that when the diameter of the MST of the input network is not greater than λ then the total length of the links of the MST is optimal.

As the diameter of the MST of the Irish network is less than 2430kms, there is no need to apply the LNS algorithm.The UK and Italy network results are also summarised in Table 7. Because of the size of these countries, it is not possible to obtain an optical island with the MST algorithm as the diameter

Table 7 Lower bounds for the total length of the links and the total length of the paths with diameters of the networks

			Diameter		Lower Bounds (KMs)	
Network	#MC	#MC Pairs	IDN	MST	TLP	TLL
Ireland	18	306	509	783	71272	1314
Ireland	20	380	558	918	91584	1400
Ireland	22	462	549	897	107450	1416
Ireland	24	552	576	1135	134040	1645
UK	74	5402	1747	3443	3154070	5506
UK	79	6162	1751	3091	3599974	6010
UK	84	6972	1761	3176	3981740	5943
UK	89	7832	1751	3129	4434340	6198
UK	94	8742	1754	2740	4802718	5949
UK	99	9702	1748	2918	5554962	6480
Italy	132	17292	1779	2974	11799056	7124
Italy	189	35532	1844	3367	24157057	11300

of the MST of the input network is more than the value of λ. Furthermore, because of the number of variables and constraints, solving the UK and Italy instances are out of the scope of systematic search (see Section 5.1 for details). Therefore, in the following sections we will present the results obtained using LNS for those instances. We omit the lower bound on the total length of the paths as this is a secondary objective.

5.2.2 Heuristics for Finding An Initial Solution

In this section, we compare the impact of three different heuristics for ordering the pairs of MC nodes on the quality of finding initial solutions using Algorithm 1. The heuristics are based on random order (`random`), non-decreasing order (`nondec`) and non-increasing order (`noninc`) of the lengths of the shortest paths between the pairs of MC nodes in the IDN. For random order, the presented results are the median values of solving each instance ten times. We compare these heuristics by showing the gaps with respect to the lower bounds on the total length of the links.

Table 8 Comparison of heuristics for finding initial solution

	`nondec`		`noninc`		`random`	
Network	Gap(%)	Time (sec.)	Gap(%)	Time (sec.)	Gap(%)	Time (sec.)
UK 74	14.89	547.31	44.49	544.26	35.45	537.41
UK 79	7.48	651.16	47.28	669.41	37.62	647.12
UK 84	11.63	818.69	44.91	834.88	37.72	823.72
UK 89	10.55	973.82	46.90	1002.10	34.24	959.92
UK 94	6.93	1171.91	48.35	1207.68	35.39	1163.80
UK 99	6.63	1415.95	44.77	1499.96	35.76	1444.84
Italy132	7.30	4311.58	63.72	4462.44	40.02	4267.11
Italy189	19.95	17293.60	68.15	17700.80	43.27	17514.30

The results show that ordering the pairs of metro-core nodes has a clear effect on the quality of the initial solution. Furthermore, the results also indicate that the non-decreasing heuristic has an absolute advantage over the other heuristics while non-increasing order is the worst performing heuristic. However, the reported gaps show the need of a neighborhood search algorithm to improve the result of the initial solution. Table 8 also indicates that the time to find an initial solution is primarily proportional to the size of the network.

5.2.3 Impact of Neighbourhood Selection

Table 9 Comparison of Different Neighbourhood Selection

	`minsup`		`maxsup`		`minlgraph`		`maxlgraph`	
N	Gap(%)	Time (sec.)	Gap(%)	Time (sec.)	Gap(%)	Time (sec.)	Gap(%)	Time (sec.)
UK 74	**1.18**	**8348.02**	19.43	2913.29	1.18	8777.81	19.43	2877.39
UK 79	10.27	38.53	9.88	1845.78	**0.91**	**7623.83**	9.88	647.80
UK 84	1.41	6171.11	0.91	998.92	0.97	377.70	**0.80**	**491.86**
UK 89	12.31	715.68	12.31	453.01	**12.31**	**254.15**	12.31	428.88
UK 94	9.01	418.05	9.59	500.75	9.01	421.13	**1.92**	**4102.62**
UK 99	12.31	880.10	21.57	3401.27	**10.92**	**6438.70**	38.05	2706.45
IT 132	30.23	1695.95	**0.63**	**3758.85**	17.28	3904.84	32.32	3131.64
IT 189	67.33	183.40	67.47	318.90	**65.19**	**278.50**	67.91	338.60

To evaluate the subproblem selection heuristics given in Section 4.2, we use the worst initial solutions presented in Table 8 since it gives a larger room for neighbourhood search and allows evaluating different heuristics fairly. In Table 9, *minimum support* (`minsup`), *maximum support* (`maxsup`), *minimum link graph* (`minlgraph`), and *maximum link graph* (`maxlgraph`) based subproblem selection heuristics are compared in the neighbourhood search algorithm. The links are ordered in the increasing order of `support` and `linkgraph` using `minsup` and `minlgraph` heuristics respectively and they are ordered in the decreasing order using the maximum versions of the heuristics. Note that the time column indicates the time between the initial solution found and the time at which the corresponding gap is found. All instances are run until the optimality gap in terms of the size of the network falls below 1% or the runtime exceeds 5 hours including the time for finding the initial solution. The best solutions in terms of gap are shown in bold. We break the ties based on the solution time.

Overall, the results in Table 9 suggest that the `minlgraph` heuristic performs better in terms of solution quality. The average gaps for `minsup` and `maxsup` are 18% and 17.72% respectively whereas the average gaps for `minlgraph` and `maxlgraph` are 14.72% and 22.82% respectively. Furthermore, although `minlgraph` is more complex, in some cases it can find the same solution faster depending on the topology of the network as in the UK 89 instance.

Based on these findings, we conclude that the `minlgraph` heuristic is better than the `minsup` and we continue to use it throughout the next experiments. Note that as a consequence of being adaptive, we were not limiting the size of the subproblems. However, it may not be worthwhile to explore some large subproblems since they are not improving at the end. Hence, in the next subsection, we analyse the best upper bound for the size of the subproblems to be considered during the search.

5.2.4 Evaluation of different subproblem sizes

As shown in Figure 4, which shows the change in the size of the subproblems for the UK 94 instance, the size of the subproblems can be limited without harming the search quality. Hence, in this section, we will investigate the effect of restricting the size of the subproblems on the quality and search effort to find the minimum upper bound for the size of the subproblems.

The impact of bounding the subproblem size can be seen in Table 10. For example, when the subproblem size is bounded to 20% in the UK network with 94 metro-nodes, the same quality solution of unbounded subproblem size is obtained in shorter time. Note that the time column in Table 10 indicates the time elapsed between finding the initial solution and finding the best solution in the neighbourhood search with the given subproblem size.

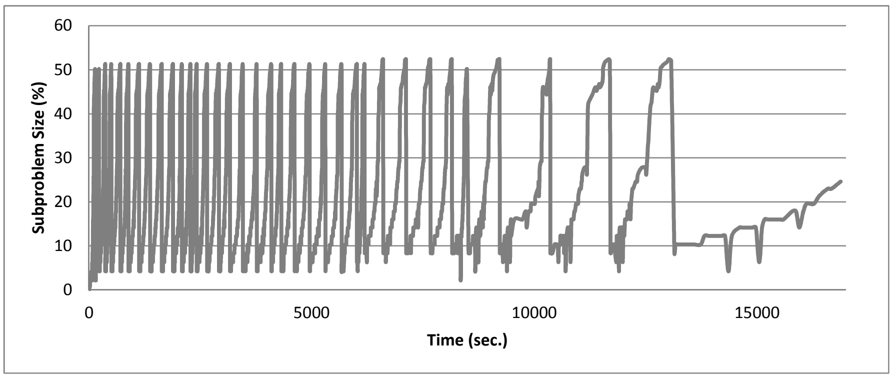


Fig. 4 Changes in the size of the subproblem during the search progress for the UK 94 instance

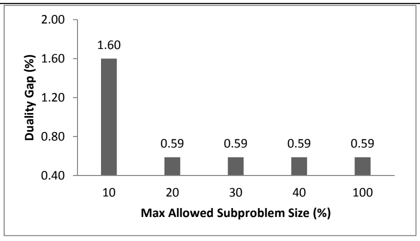
The convergence of the duality gap for each instance is depicted in Table 11. Based on these plots, the subproblem size of 20% seems enough for networks of all sizes. However, we set the upper bound on the size of the subproblem to 40% considering the instance of the UK network with 89 metro-nodes.

Max Subproblem Size	10%		20%		30%		40%		100%	
Network	Gap (%)	Time (sec.)	Gap (%)	Time (sec.)	Gap (%)	Time (sec.)	Gap (%)	Time (sec.)	Gap (%)	Time (sec.)
UK 74	1.60	57.44	0.59	106.30	0.59	108.53	0.59	112.01	0.59	114.22
UK 79	1.84	60.09	1.00	14.35	1.00	14.37	1.00	14.46	1.00	14.38
UK 84	1.03	108.40	0.93	128.58	0.93	128.22	0.93	128.01	0.93	125.05
UK 89	3.04	122.69	1.50	144.44	1.13	179.87	0.70	199.81	0.70	194.72
UK 94	1.59	70.23	1.52	4845.05	1.52	5882.49	1.52	5882.64	1.52	8316.53
UK 99	0.99	88.07	0.99	87.81	0.99	87.66	0.99	88.58	0.99	84.87

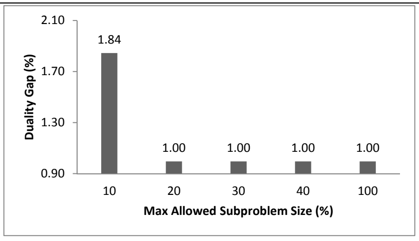
Table 10 Effect of limiting the subproblem size on the neighborhood search

Table 11 Performance using different subproblem-size bounds

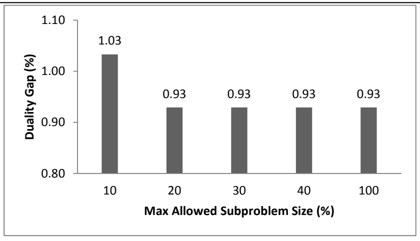
UK 74



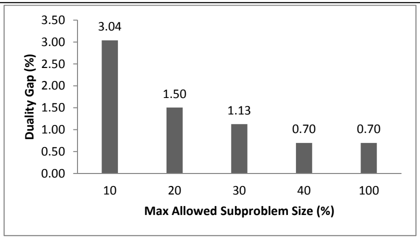
UK 79



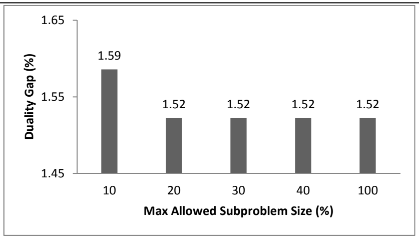
UK 84



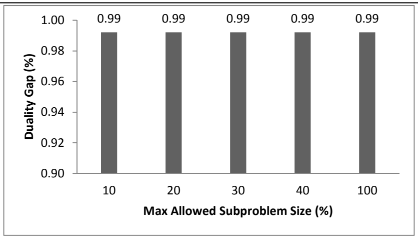
UK 89



UK 94



UK 99



5.2.5 Alternative Methods for Finding an Initial Solution

The results of all the networks using the best initial solution heuristic for subproblem selection (`minlgraph`), and bounding the subproblem size to 40% is shown in Table 12. The results indicate that the proposed neighbourhood search algorithm succeeds at reducing the duality gap of a given solution. However, the overall adaptive large neighbourhood search algorithm is hampered by the time spent in finding an initial solution. For example, in the biggest instance, IT 189, almost all the solving time is spent finding the initial solution. Therefore, in this subsection, we investigate two alternative ways of finding an initial solution faster.

Network	Adaptive Large Neighbourhood Search Steps				Termination Time (sec.)
	Initial Solution		Neighbourhood Search		
	Duality Gap (%)	Time (sec.)	Duality Gap (%)	Time (sec.)	
UK 74	14.89	547.31	0.59	112.01	659.32
UK 79	7.48	651.16	1.00	14.46	665.62
UK 84	11.63	818.69	0.93	128.01	946.70
UK 89	10.55	973.82	0.70	199.81	1173.63
UK 94	6.93	1171.91	1.52	5882.64	18000.00
UK 99	6.63	1415.95	0.99	88.58	1504.53
IT 132	7.30	4311.58	0.94	1466.72	5778.30
IT 189	19.95	17293.60	9.88	773.40	18000.00

Table 12 Time consumed by steps of the adaptive large neighbourhood search

1. In the first alternative approach, we invoke FINDSOLUTION by setting L^b to the set of links in the Minimum Spanning Tree (MST) instead of setting it to the empty set. The result would be that many pairs of nodes would already be bounded and therefore BOUNDEDCHEAPESTPATH would be invoked fewer times as there would be relatively fewer pairs of nodes that would need to be repaired. Therefore, the set S is initialised with the set of pairs of nodes whose path lengths are violating the bound when considering the links of the MST. Furthermore, we order the set S in Algorithm 1 using the non-increasing order of the lengths in the spanning tree. Here the intuition is that repairing the longest path first might repair other pairs of relatively shorter paths and consequently this would further reduce the number of calls to MILP-based BOUNDEDCHEAPESTPATH.
2. The second approach is to find an initial set of links with the MST but instead of using a MILP model to repair unbounded paths, we use a greedy algorithm. Here also FINDSOLUTION would be invoked by initialising L^b with the set of links in the MST, S with the set of pairs of nodes whose paths are not bounded, and ordering the set S based on the lengths of the paths in the MST in a non-increasing manner. The MILP-free BOUNDED-CHEAPESTPATH would compute the shortest path for a given pair of nodes in the input graph and add the corresponding links which are not included in the current solution. Subsequently the paths of all the pairs of nodes of the set S would be re-computed based on the current solution. The reason is that adding more links to the current solution might solve the infeasi-

bility of some of the other pairs of nodes of the set S. The greedy search would continue until all the paths are repaired.

The results obtained using different methods for computing initial solutions are shown Table 13 where the best result of each instance is shown in bold. The results show that finding an initial solution by computing minimum spanning tree and repairing infeasible paths using MILP is the faster method to find an initial feasible solution compared to the MILP based decomposition. As an example, consider the Italy instance with 189 metro-nodes. While an initial solution is found in almost five hours with the MILP decomposition, it takes only four minutes with MST and repairing infeasible paths with the MILP based method. Moreover, the quality of the initial solutions obtained by computing MST and then repairing with MILP is also better than the other methods.

	MILP Based Initial Solution		**MST repaired with MILP based Initial Solution**		**MST repaired with Greedy Heuristic based Initial Solution**	
	Duality Gap (%)	**Time (sec.)**	**Duality Gap (%)**	**Time (sec.)**	**Duality Gap (%)**	**Time (sec.)**
UK 74	14.89	547.31	**3.93**	**8.58**	20.83	24.43
UK 79	7.48	651.16	**4.82**	**9.44**	13.85	23.36
UK 84	11.63	818.69	**6.72**	**11.93**	19.36	35.53
UK 89	10.55	973.82	**6.44**	**15.94**	26.78	60.5
UK 94	6.93	1171.91	**5.77**	**16.72**	18.76	37.14
UK 99	6.63	1415.95	**4.08**	**19.27**	16.76	49.14
IT 132	7.3	4311.58	**5.43**	**54.62**	45.56	499.2
IT 189	19.95	17293.6	**12.1**	**236.19**	40.51	2829.99

Table 13 Performance comparison of all alternative methods to find an initial solution

The impact of different initial solutions on LNS is summarised in Table 14 where the best result of each instance is shown in bold. We recall that the algorithm stops when the gap with respect to the lower bound is less than or equal to 1% or when it runs out of time. Note that time column shows the total elapsed time until the corresponding solution is found. Although there is no clear winner, the results suggest that the combination of the neighbourhood search and the initial solutions obtained using MST repaired with MILP is better than the other methods. It is worth mentioning that it terminates for 5 instances and computed the best results for 4 instances including the biggest Italy instance with 189 MC nodes. Figure 5 illustrates how the upper bound on the total length of the links (TLL) converges during the adaptive large neighbourhood search for the Italy instance with 189 MC nodes. In this experiment we obtain a initial solution by computing the minimum spanning tree and repairing the infeasible paths using the MILP approach.

	MILP Based Initial Solution		**MST repaired with MILP based Initial Solution**		**MST repaired with Greedy Heuristic based Initial Solution**	
	Duality Gap (%)	**Time (sec.)**	**Duality Gap (%)**	**Time (sec.)**	**Duality Gap (%)**	**Time (sec.)**
UK 74	0.59	659.32	**0.5**	**24.23**	0.59	197.28
UK 79	1.00	665.62	1.67	61.43	**0.98**	**76.76**
UK 84	**0.93**	**946.7**	2.45	43.45	0.96	211.13
UK 89	0.7	1173.63	**0.7**	**128.26**	2.47	13998.3
UK 94	1.52	7054.55	**1.01**	**101.59**	1.39	322.35
UK 99	0.99	1504.53	**0.99**	**110.71**	0.94	364.71
IT 132	**0.94**	**5778.3**	1.36	1494.78	2.42	5343.48
IT 189	9.88	18000	**3.22**	**5094.36**	15.48	17346.6

Table 14 Performance comparison of neighbourhood search algorithm with all alternative methods to find an initial solution

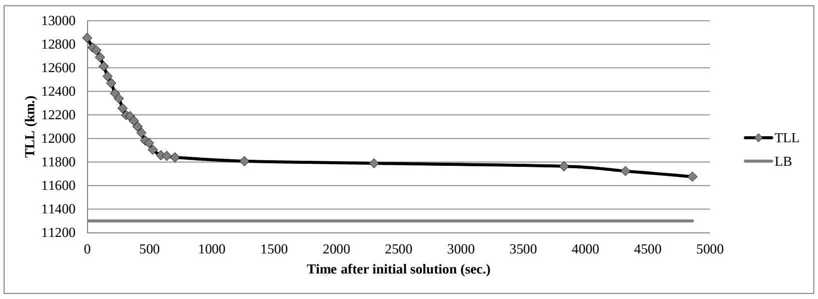


Fig. 5 Toal length of the links vs. the lower bound during the search for Italy instance with 189 metro-nodes

6 Exploring alternative approaches based on RDMSTP techniques

In this section we look closely at two related approaches and discuss how they can be adapted to complement our approach to DCND.

6.1 Computing a tighter lower bound for DCND

We introduce a tighter lower bound implemented in terms of the rooted distance-constrained minimum spanning tree problem (RDMSTP) [16].

In Table 7 we presented a lower bound for DCND based on the computation of the minimum spanning tree connecting the nodes of the network. In this section we consider a tighter lower bound, which is defined in terms of the rooted distance-constrained minimum spanning tree problem (RDMSTP) [16].

Given a graph $G = (V, E)$ with set of nodes N and set of edges E, a cost c_e and a delay δ_e associated with each edge e of E, and an upper bound λ, RDMSTP is to find a spanning tree T of the graph with minimum total cost and such that every path from the root node to any other node has total delay not greater than λ.

DCND can be easily expressed in terms of RDMSTP. In order to see this let us refer to the decision version of RDMSTP. That is, instead of looking for a tree of minimum cost, we want to decide whether there is a tree whose cost is less than or equal to a given bound.

Let us simplify the definition of RDMSTP by unifying the two constants associated with every edge. That is, we assume that the delay and the cost of each edge ere equivalent. Let us also abuse the notation by saying that we have a predicate with the same name:

$$RDMSTP(G^*, s, \lambda, cost, T)$$

that holds iff T is a tree, subgraph of G^*, that connects s with all the other nodes using paths bounded by λ and the cost of the tree is less than or equal to $cost$. Then $DCND(G^*, \lambda, cost, G)$ can be expressed as follows:

– $RDMSTP(G^*, v, \lambda, cost_v, T_v)$, for all $v \in nodes(G^*)$
– $G = \bigcup_{v \in nodes(G^*)} T_v$
– $\sum_{v \in nodes(G^*)} cost_v \leq cost$

where $nodes(G^*)$ denotes the set of nodes of G^*. This model leads us to two bounds for DCND:

– An upper bound can be obtained by composing the optimal solutions of the corresponding RDMSTPs. That is, if T_v is the optimal tree for node v, a suboptimal solution for DCND is $G = \bigcup_{v \in nodes(G^*)} T_v$ and the upper would therefor be the cost of G.
– A lower bound can be obtained by choosing the most expensive T_v. Indeed the cost of the optimal solution of DCND is bound to be at least as expensive as the cost of the most expensive T_v since all the nodes need to be reached from v respecting the bound on the length of the paths.

As mentioned before, if the length of the paths are not constrained, the optimal solution to DCND is a minimum spanning tree. What makes the problem hard is the constraint on the length of the paths. A drawback of the lower bound proposed before is that it does not take into account the upper bound on the length of the path.

Figure 6 shows a simple scenario where we can appreciate the difference between the two bounds. In this cases we have assumed that λ is 15, which rules out the possibility of associating all nodes with the same tree. Notice that if we did so for node 2, the path from node 2 to node 3 would violate the upper bound on the length of the path. Therefor, (b), (c) and (d) are the best bounded trees for nodes 1, 2, and 3 respectively. Here we can observe the gap between the two lower bounds since the lower bound based on the minimum spanning tree would be 20, and the one based on the maximum cost over the bounded spanning trees would be 25.

We can observe a remarkable difference between the two bounds in Table 15. In this table we are focusing on the Irish instances since the MILP approach used to compute the tighter bounds would not scale up to the sizes of the large instances. For each case we are reporting the number of nodes

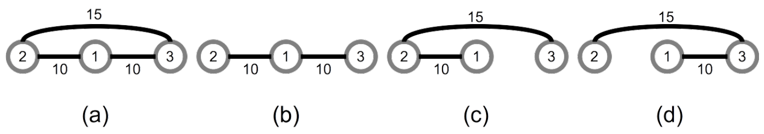


Fig. 6 (a) is n example of graph with bounded trees of different costs. λ is assumed to be 15, thus ruling out the tree associated with node 1 for the other two nodes. (b), (c) and (d) are the bounded trees for nodes 1, 2, and 3 respectively.

Table 15 Comparing the two lower bounds proposed for DCND on the Irish instances

Network	#MC	λ	MST	BT	LNS	MST Gap (%)	BT Gap (%)
Ireland	18	510	1314	1914	2485	47.12	22.98
Ireland	20	559	1400	1996	2194	36.19	9.02
Ireland	22	550	1416	1796	2502	43.41	28.22
Ireland	24	577	1645	2297	2616	37.12	12.19

(Column #MC), the upper bound on the length of the paths considered (Column λ), the lower bound obtained by computing the minimum spanning tree (Column MST), the lower bound obtained by considering the maximum on the optimal costs of the bounded spanning trees (Column BT), and the cost obtained by our large neighbourhood search approach (Column LNS). In the last two columns (Column MST Gap and Column BT Gap) we are also showing the gap obtained with the corresponding lower bounds.

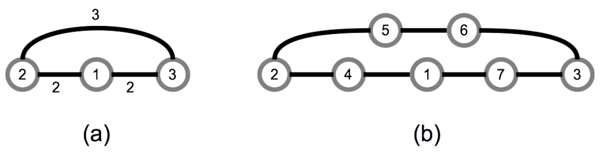


Fig. 7 Computation of a unit distance graph. (a) is the original graph and (b) is its corresponding unit distance graph.

6.2 Modelling DCND in terms of the Steiner tree problem

We consider the layered graph approach to RDMSTP proposed in [17]. In this approach it is assumed that the edges of the graph are associated with unit delays. We remind the reader that in our case we have unified the delays and the costs of the edges. So, this means that we need to transform our networks so that the resulting edges have unit costs. In Figure 7 we show an example of this transformation. Figure 7(a) is the original graph and Figure 7(b) is its corresponding unit cost graph.

The next step in the approach consist is computing the layered graph. As explained in [17], the layered graph $G_L = (N_L, E_L)$ of a given directed graph graph $G = (N, E)$ with source node 0 can be defined as follows:

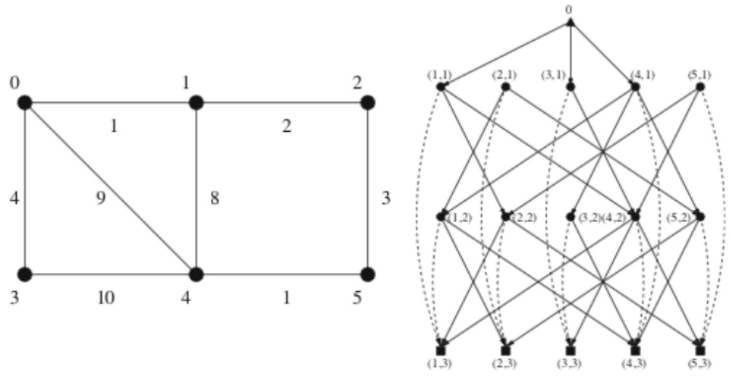


Fig. 8 Layered graph transformation: original graph of an instance with $\lambda = 3$ on the left-hand side and the corresponding layered graph on the right-hand side (taken from [17])

$$\begin{aligned}
N_L &= \{0\} \cup \{(i,h) : 1 \leq h \leq \lambda, i \in N \setminus \{0\}\} \\
E_L &= E_0 \cup E_1 \cup E_2 \\
E_0 &= \{(0,(j,1)) : (0,j) \in E\} \\
E_1 &= \{((i,h),(j,h+1)) : (i,j) \in E, i \neq 0, 1 \leq h \leq \lambda - 1\} \\
E_2 &= \{(i,h),(i,\lambda)) : i \in N \setminus \{0\}, 1 \leq h \leq \lambda - 1\}
\end{aligned} \tag{2}$$

Basically, in the layered graph, each node in the original graph is replicated λ times. In the transformation the nodes $\{(i,\lambda) : i \in N \setminus \{0\}\}$ play the role of terminal nodes. Any path from the source to a terminal node corresponds to a path in the original graph whose length is less than or equal to λ. Therefor, a minimum Steiner tree containing the source and the terminal nodes corresponds to an optimal solution of RDMSTP. In Figure 8 we show an example of a transformation, which has been taken from [17].

Something that we can infer from the definition of the transformation is that there are $(|N| - 1) * \lambda + 1$ nodes in G_L. This together with the fact that the size of the graph is remarkably increased in the transformation to its corresponding unit cost graph prevents us from exploring this approach further. In Table 16 we consider again the Irish instances. In Column G we present the sizes of the original graphs. In Column G_U we show the sizes of their corresponding unit graphs. In Column $|N_L|$ we report the number of nodes in their corresponding layered graphs. As it can be observed in the table, even though the unit cost graphs and the layered graphs are sparse, their set of nodes are very large. In fact we could only compute the layered graph of the Irish network of 18 nodes, which has a layered graph of 36202351 nodes and 72401195 edges. For the other cases we run out of memory.

Table 16 Comparing the size of the original network with its unit length transformation and its multi-layer transformation

Network	#MC	λ	G		G_U		$\lvert N_L\rvert$
			$\lvert N\rvert$	$\lvert E\rvert$	$\lvert N\rvert$	$\lvert E\rvert$	
Ireland	18	510	18	306	70986	71274	36202351
Ireland	20	559	20	380	91222	91582	50992540
Ireland	22	550	22	462	107014	107454	58857151
Ireland	24	577	24	552	133508	134036	77033540

7 Conclusions and Future Work

In this paper we proposed a MILP model, a CP model and an adaptive large neighbourhood search (LNS) method for diameter constrained network design (DCND) problems. We showed that both exact approaches can not solve even small size instances within a time limit of 1200 seconds. Hence, we investigated performances of the LNS approach to design an optical core network. Our empirical results with nation-wide large telecommunication networks of various countries show the scalability and quality of the proposed LNS, which are very close to optimal. Furthermore, we investigated various methods to speed-up the search process and proposed alternative approaches for finding an initial solution, selecting subproblems with bounded sizes, and computing tighter lower bounds. We showed that the methods proposed have a big impact on the performance of our LNS approach and the quality of the solutions obtained.

The proposed approach is general and therefore it can be applied in a number of settings without changing the methodology, e.g., in many network design problem one might enforce degree constraints that is a node should be connected to at least k other nodes. Another possible extension of this study is to consider node/edge disjointness for resilient network design problems. We also consider to develop Benders decomposition [8] and Lagrangian relaxation [14] methods to obtain better lower bounds for these extensions.

When it comes to the current implementation, we believe that one way of improving the overall performance of our LNS approach is by implementing an incremental version of the Minimum Support and Minimum Link Graph subproblem selection approaches.

Finally, developing multi-objective methods for investigating trade-off between minimizing the total length of links and paths is a promising future research direction.

Acknowledgement

The authors acknowledge both Eircom and Telecom Italia for providing network data. This work was supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318137 (DISCUS), and Science Foundation Ireland (SFI) under grant numbers 12/RC/2289

(Insight P2), 16/SP/3804 (ENABLE), and 16/RC/3918 (CONFIRM).

References

1. Azi, N., Gendreau, M., Potvin, J.: An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Computers & OR **41**, 167–173 (2014)
2. Barták, R., Zhou, N., Dovier, A.: Multiple-origin-multiple-destination path finding with minimal arc usage: Complexity and models. In: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (2016)
3. Bent, R., Van Hentenryck, P.: A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Comput. Oper. Res. **33**(4), 875–893 (2006). URL http://dx.doi.org/10.1016/j.cor.2004.08.001
4. Chabrier, A., Danna, E., Le Pape, C., Perron, L.: Solving a network design problem. Annals OR **130**(1-4), 217–239 (2004)
5. Chimani, M., Spoerhase, J.: Network design problems with bounded distances via shallow-light steiner trees. CoRR **abs/1409.6551** (2014). URL http://arxiv.org/abs/1409.6551
6. CSPLib: A problem library for constraints. http://www.csplib.org (1999)
7. De Backer, B., Furnon, V.: Meta-heuristics in constraint programming experiments with tabu search on the vehicle routing problem. In: 2nd International Conference on Metaheuristics (1997)
8. de Camargo, R., de Miranda, G., Løkketangen, A.: A new formulation and an exact approach for the many-to-many hub location-routing problem. Applied Mathematical Modelling **37**(12-13), 7465 – 7480 (2013). DOI http://dx.doi.org/10.1016/j.apm.2013.02.035. URL http://www.sciencedirect.com/science/article/pii/S0307904X1300142X
9. Dijkstra, E.: A Note on Two Problems in Connection with Graphs. Numerische Mathematik **1**(1), 269–271 (1959)
10. Discus: Deliverable 7.2, Preliminary quantitative results for flat optical network. Tech. rep., The DISCUS Project (FP7 Grant 318137) (2014)
11. Dodis, Y., Khanna, S.: Designing networks with bounded pairwise distance. In: Proc. 21st Ann. ACM Symposium on Theory of Computing (STOC'99), pp. 750–759 (1999)
12. Dooms, G., Deville, Y., Dupont, P.: Cp(graph): Introducing a graph computation domain in constraint programming. In: P. van Beek (ed.) Principles and Practice of Constraint Programming - CP 2005, pp. 211–225. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
13. Elias, J., Martignon, F., Carello, G.: Very large-scale neighborhood search algorithms for the design of service overlay networks. Telecommunication Systems **49**(4), 391–408 (2012). DOI 10.1007/s11235-010-9381-4. URL http://dx.doi.org/10.1007/s11235-010-9381-4
14. Gelareh, S., Maculan, N., Mahey, P., Monemi, R.: Hub-and-spoke network design and fleet deployment for string planning of liner

shipping. Applied Mathematical Modelling **37**(5), 3307 – 3321 (2013). DOI http://dx.doi.org/10.1016/j.apm.2012.07.017. URL http://www.sciencedirect.com/science/article/pii/S0307904X12004295

15. Gomes, C.: Computational sustainability: Computational methods for a sustainableenvironment, economy, and society. The Bridge **39**(4), 5–13 (2009)
16. Gouveia, L., Paias, A., Sharma, D.: Modeling and solving the rooted distance-constrained minimum spanning tree problem. Computers & OR **35**(2), 600–613 (2008). DOI 10.1016/j.cor.2006.03.022. URL https://doi.org/10.1016/j.cor.2006.03.022
17. Gouveia, L., Simonetti, L., Uchoa, E.: Modeling hop-constrained and diameter-constrained minimum spanning tree problems as steiner tree problems over layered graphs. Math. Program. **128**(1-2), 123–148 (2011). DOI 10.1007/s10107-009-0297-2. URL https://doi.org/10.1007/s10107-009-0297-2
18. Hurink, J.: An exponential neighborhood for a one-machine batching problem. OR Spectrum **21**(4), 461–476 (1999). URL http://dx.doi.org/10.1007/s002910050098
19. Kokangul, A., Ari, A.: Optimization of passive optical network planning. Applied Mathematical Modelling **35**(7), 3345 – 3354 (2011). DOI dx.doi.org/10.1016/j.apm.2011.01.017. URL http://www.sciencedirect.com/science/article/pii/S0307904X11000308
20. Kowalski, D., Nutov, Z., Segal, M.: Scheduling of vehicles in transportation networks. In: A. Vinel, R. Mehmood, M. Berbineau, C. Garcia, C.M. Huang, N. Chilamkurti (eds.) Communication Technologies for Vehicles, pp. 124–136. Springer Berlin Heidelberg (2012)
21. Mahjoub, R., Simonetti, L., Uchoa, E.: Hop-level flow formulation for the survivable network design with hop constraints problem. Networks **61**(2), 171–179 (2011)
22. Malitsky, Y., Mehta, D., O'Sullivan, B., Simonis, H.: Tuning parameters of large neighborhood search for the machine reassignment problem. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pp. 176–192. Springer (2013)
23. Medhi, D.: Network Routing: Algorithms, Protocols, and Architectures. Morgan Kaufmann (2010)
24. Mehta, D., O'Sullivan, B., Ozturk, C., Quesada, L.: An adaptive large neighbourhood search for designing transparent optical core network. In: Telecommunications (ConTEL), 2015 13th International Conference on, pp. 1–8 (2015). DOI 10.1109/ConTEL.2015.7231187
25. Mehta, D., O'Sullivan, B., Ozturk, C., Quesada, L., Simonis, H.: Designing an optical island in the core network: From routing to spectrum allocation. In: 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, pp. 560–567 (2014). DOI 10.1109/ICTAI.2014.90
26. Meyerson, A.: Online algorithms for network design. In: IN PROCEEDINGS OF THE 16TH ACM SYMPOSIUM ON PARALLELISM IN ALGORITHMS AND ARCHITECTURES, pp. 275–280. ACM Press (2003)

27. Miranda, G., Luna, H., de Camargo, R., Pinto, L.: Tree network design avoiding congestion. Applied Mathematical Modelling **35**(9), 4175 – 4188 (2011). DOI http://dx.doi.org/10.1016/j.apm.2011.02.046. URL http://www.sciencedirect.com/science/article/pii/S0307904X1100117X
28. Muller, L., Spoorendonk, S.: A hybrid adaptive large neighborhood search algorithm applied to a lot-sizing problem. DTU Management 2010. DTU Management (2010)
29. Oh, J., Pyo, I., Pedram, M.: Constructing minimal spanning/steiner trees with bounded path length. Integration **22**(1-2), 137–163 (1997)
30. van Omme, N., Perron, L., Furnon, V.: or-tools user's manual. Tech. rep., Google (2014)
31. Payne, D.: FTTP deployment options and economic challenges. In: Proceedings of the 36th European Conference and Exhibition on Optical Communication (ECOC 2009) (2009)
32. Pisinger, D., Ropke, S.: Large neighborhood search. In: Handbook of metaheuristics, pp. 399–419. Springer US (2010)
33. Prud'homme, C., Fages, J., Lorca, X.: Choco3 Documentation. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S. (2014). URL http://www.choco-solver.org
34. Régin, J.C.: Global constraints: A survey. In: P. van Hentenryck, M. Milano (eds.) Hybrid Optimization: The Ten Years of CPAIOR, pp. 63–134. Springer New York, New York, NY (2011)
35. Ruffini, M., Wosinska, L., Achouche, M., Chen, J., Doran, N., Farjady, F., Montalvo, J., Ossieur, P., O'Sullivan, B., Parsons, N., Pfeiffer, T., Qiu, X., Raack, C., Rohde, H., Schiano, M., Townsend, P., Wessaly, R., Yin, X., Payne, D.: Discus: an end-to-end solution for ubiquitous broadband optical access. Communications Magazine, IEEE **52**(2), S24–S32 (2014). DOI 10.1109/MCOM.2014.6736741
36. Ruthmair, M., Raidl, G.: A kruskal-based heuristic for the rooted delay-constrained minimum spanning tree problem. In: Computer Aided Systems Theory-EUROCAST 2009, pp. 713–720. Springer (2009)
37. Ziegelmann, M.: Constrained Shortest Paths and Related Problems - Constrained Network Optimization. VDM Verlag, Saarbrücken, Germany, Germany (2007)