Joint Optimization of Primary and Backup Controller Placement and Availability Link Upgrade in SDN Networks

Joint Optimization of Primary and Backup Controller Placement and Availability Link Upgrade in SDN Networks⋆

Dorabella Santos[a,*], Teresa Gomes[b,a]

[a] INESC Coimbra, DEEC, Rua Sílvio Lima, Polo 2, 3030-290 Coimbra, Portugal
[b] University of Coimbra, Department of Electrical and Computer Engineering, Rua Sílvio Lima, Polo 2, 3030-290 Coimbra, Portugal

ARTICLE INFO

Keywords:
SDN
availability
controller placement
spine
integer linear programming
bi-objective

ABSTRACT

In Software-Defined Networking (SDN), the control and data planes are decoupled, leading to a more programmable and efficient network management. In this paper, the controller placement problem in SDN is addressed, jointly with the problem of exploring a high-availability tree subgraph, in order to support delay and availability requirements between the switches and the controllers. We consider that each switch connects to a primary and to a backup controller. We formulate the joint optimization model as an integer linear programming model (ILP), and propose a heuristic method when the exact model becomes impractical. Furthermore, we compare two ILP formulations, and we also compare the controller redundancy solutions with those considering path redundancy alone.

1. Introduction

Software-Defined Networking (SDN) simplifies network management and allows for rapid network innovations. Traditionally, the forwarding and control planes were integrated into the network switches. However, as the complexity of networks increased, the paradigm of decoupling the control and data (forwarding) planes became more and more significant. The network management and control decisions are centralized in the control plane which consists in one or more SDN controllers, while the data plane switches basically become forwarding devices managed by the controllers. This approach circumvents the high cost of black box technology, providing higher network programmability and more efficient network management.

The benefits of SDN has led to real deployments especially in datacenters, such as Google's B4 [1], and in campus and enterprise networks. There is also a significant effort to deploy SDN in transport networks [2]. Especially with the advent of 5G, SDN is a promising approach for transport layers to meet the demands of very high availability and bandwidth [3, 4]. Several works address SDN optimization problems for transport networks related to bandwidth [4], protection [5], service provisioning and restoration [6, 7] and even availability [8].

In this paper, we address the SDN controller placement problem (CPP) under delay and availability constraints. The CPP addresses the question of how many controllers and where to deploy them in the network. It was introduced in [9] and shown to be NP-hard.

The controller placement is strongly influenced by the delays between the switches and the controllers, called switch-controller (SC) delays [10]. For practical reasons, multiple controllers are deployed in the network that function as a logically centralized unit. Synchronization between controllers can result in important delays between the controllers themselves, called controller-controller (CC) delays [11]. Several works have studied the CPP under different delay and load balancing criteria. In [12], the authors aim at selecting several controller placement solutions, under multiple objectives: number of controllers, maximum SC delay, maximum CC delay and controller load balance. However, minimizing the average SC and CC delays are conflicting objectives [11] since considering more controllers will, in general, decrease the average SC delay but increase the average CC delay.

In our work, we consider maximum SC and CC delay guarantees, as in [13]. Moreover, we consider that each switch connects to a primary and backup controller via a node-disjoint pair of paths. We intend to optimize the controller placement guaranteeing the maximum delay requirements, and guaranteeing a minimum end-to-end availability between each switch and its controllers. However, the desired end-to-end availability cannot always be achieved by path redundancy alone [14].

In [15], the concept of a spine is proposed, where a higher availability subgraph exists in the network. In this work, we also consider such a subgraph, where its links can be upgraded to have increased availability at a given cost [16]. This can be done by reducing the average time to repair of the link and/or by reducing the average time between failures (for example, by installing more robust equipment on the link or by burying a link). The existence of this subgraph in the network allows high resilience routing and also resiliency differentiation, in a more effective manner than just increasing availability through path redundancy [16].

Our previous works [17] and [18], have also addressed the CPP problem in the context of availability link upgrade. In [17], we considered maximum delay and availability guarantees for the paths connecting each switch to its controller (control paths). Path redundancy was not considered. In [18], we extended the work to include path redundancy. Each switch connects to its primary controller via a pair of node-

*Corresponding author
dorabella.santos@gmail.com (D. Santos); teresa@deec.uc.pt (T. Gomes)
ORCID(s): 0000-0003-0368-2222 (D. Santos); 0000-0002-3084-5608 (T. Gomes)

disjoint paths. The end-to-end availability was guaranteed for the pair of paths, by considering a spanning tree subgraph whose links could be upgraded to have high availability. For both works, we considered integer linear programming (ILP) models to solve the problems.

In our recent paper [19], we have also addressed the CPP problem in the context of availability link upgrade, considering geodiversity. The ILP model in [19] can be simplified to consider controller redundancy (without geodiversity). However, we present here a clean version of the ILP model for our optimization problem with controller redundancy.

The contribution of this paper is as follows: (i) we relate the ILP formulations used in [17] and [18] (for the simplest form of our problem), and show that the latter is more efficient than the former; (ii) we present the clean ILP model considering controller redundancy via node-disjoint paths, which is a derivative from the ILP model in [19]; (iii) we present a more thorough analysis between path and controller redundancy than in [19], where the focus was on the performance of different geodiverse solutions; (iv) we propose a heuristic for solving the addressed joint optimization problem, when the exact method becomes impractical.

The paper is organized as follows. In Section 2, a brief summary of the most relevant related work is presented. In Section 3, the addressed CPP problem is described and the availability link upgrade model is presented. In Section 4, the joint optimization of primary and backup controller placement and spine upgrade problem is formulated as an ILP model. In Section 5, the relation between the formulations used in [17] and [18] is presented. In Section 6, a heuristic for the addressed CPP problem is proposed, for instances where the exact ILP model becomes impractical. In Section 7, the computational results comparing the two formulations and assessing the exact ILP model and the heuristic method are presented. Finally, Section 8 presents the conclusions.

2. Related Work

In [20], the CPP is addressed for single link failures, considering path redundancy and controller redundancy. The authors aim at minimizing the average SC delays. A more recent work addressing the CPP against multiple link failures is [21], where the switches can reconnect to a surviving controller in case of disconnection to its primary controller. The authors aim at minimizing the worst-case SC delay.

In [13], the CPP is addressed considering controller redundancy, to make the network robust against multiple controller failures. The authors assume that the switches can reconnect to the closest surviving controller when they lose connectivity with their primary controller. Robustness against multiple controller failures is also addressed in [22], for the capacitated CPP. The authors aim to minimize the number of controllers, while guaranteeing that each switch is assigned a given number of backup controllers.

The CPP has also been addressed against targeted attacks. In [23] and [24], the authors aim at optimizing the controller placement, against a set of targeted attacks leading to multiple node failures.

There have also been works that addressed the CPP in the context of availability. In [25], the CPP is addressed for a multiple failure scenario to assess the network availability. In this work, the controller placement is based on a failure correlation assessment of network nodes and links. In this context, [26] has also addressed the CPP considering failure probability of each network component. The authors propose several heuristics for the CPP to maximize the availability of the control paths.

In [27], the CPP is addressed in order to guarantee availability requirements for the control paths. The authors consider that each switch connects to a primary and to one or more backup controllers. This work is extended in [28], where an enhanced algorithm is presented. In this work, they show that the number of controllers placed in the network is strongly correlated to the number of nodes with degree 1 (also known as leaves or spokes).

These papers do not address the CPP under delay and availability guarantees simultaneously. We introduced the joint optimization of the controller placement under such guarantees in [17]. We further extended the work in [18] to include path redundancy and considered the spine to be a spanning tree. In [19], we considered this framework where geodiversity guarantees were further imposed.

3. Primary and Backup Controller Placement and Spine Design Problem

The problem addressed in this paper consists in selecting the controller placement and the set of links to be upgraded, such that the upgrade cost is minimized. However, another important objective is to minimize the number of controllers, in order to minimize intercontroller communication overhead.

Due to the discrete nature of C, we can solve a single objective optimization problem for each value of C, aiming to minimize the upgrade cost. Therefore, we obtain a set of solutions representing the trade-off between the number of controllers and the upgrade cost. By selecting the nondominated solutions (solutions for which no other can be better in both the number of controllers and the upgrade cost simultaneously), we obtain the Pareto front for our biobjective optimization problem of minimizing the number of controllers and minimizing the upgrade cost.

In the framework of this paper, we consider that each switch connects to a primary and a backup controller via a pair of node-disjoint paths. This ensures protection against single link, node or controller failures. We certify that CC delay between any two controllers is at most D_{cc}. Likewise, the SC delay between each switch and its primary controller is at most a stipulated maximum value D_{sc}. These maximum values ensure that the control plane has reasonable performance, as a logically centralized control plane in the failure-free case. Considering that the frequency of SC communications exceeds that of CC communications, we postulate that $D_{sc} < D_{cc}$. However, we do not guarantee a maximum value for the delay between each switch and its backup controller. The minimum value C for the number of controllers can be obtained by solving the CPP problem with respect to the delay requirements alone [23].

The node-disjoint pair of paths between each switch and its primary and backup controllers, is required to have a minimum end-to-end availability of at least $0 < \lambda \lesssim 1$. Consider a node-disjoint path pair, such that the availability of primary path and of the backup path of the pair, is given by $\mathcal{A}p$ and $\mathcal{A}_b$ respectively. Then the availability of the path pair is given by $\mathcal{A} = 1 - (1 - \mathcal{A}_p)(1 - \mathcal{A}_b)$. The target path pair availability, λ, will be ensured by guaranteeing that the primary path availability of at least λ_p and the backup path availability of at least λ_b are such that $1-(1-\lambda_p)(1-\lambda_b) \geq \lambda$ [16]. We assume $\lambda_p > \lambda_b$, given that the maximum delay for the primary control paths and the availability of the paths depend on their length.

We further assume that the links of the network can be upgraded to have enhanced availability, in order to achieve the required target availabilities. We impose that the upgraded links belong to a tree subgraph. However, neither the primary nor the backup paths are imposed to be routed over the tree subgraph. If needed, the paths will use the links belonging to the tree subgraph in order to achieve the necessary target availability.

The SDN plane is characterized by an undirected graph $G = (N, E)$, with node set N and link set E. Each link is defined the set of its end nodes, $\{i, j\}$, with $i \neq j$ and $i, j \in N$. The default availability of the links is distance-based and given by [29, pages 185-186]:

$$\alpha^0_{ij} = 1 - \frac{MTTR}{MTBF_{ij}}, \quad \{i,j\} \in E \tag{1}$$

where $MTTR = 24$ h designates the mean time to repair, and $MTBF_{ij}$ represents the mean time between failures (in hours) of link $\{i, j\}$ which is given by $MTBF_{ij} = CC \times 365 \times 24/\ell_{ij}$, where $CC = 450$ km corresponds to the cable cut rate and ℓ_{ij} is the link length.

Each link of the selected tree subgraph can be upgraded, so that the unavailability is decreased by a given value $\varepsilon \in (0, 1)$. We assume that there are κ levels of link upgrade. The unavailability of $\{i, j\}$, upgraded to level $k = 1, ..., \kappa$, is denoted as μ^k_{ij} and given by $\mu^k_{ij} = (1 - \varepsilon)\mu^{k-1}_{ij}$ [16]. Differently from [16], we do not consider any level of availability link downgrade. Since the default unavailability is given by $\mu^0_{ij} = 1 - \alpha^0_{ij}$, we have that $\alpha^k_{ij} = 1 - \mu^k_{ij}$ and so

$$\alpha^k_{ij} = \alpha^{k-1}_{ij} + \varepsilon - \varepsilon\alpha^{k-1}_{ij}, \quad k = 1, ..., \kappa, \quad \{i,j\} \in E \tag{2}$$

The availability of a link can be upgraded to a level k, incurring in a cost given by [16, 30]:

$$c^k_{ij} = -\ell_{ij} \cdot \ln\left(\frac{1 - \alpha^k_{ij}}{1 - \alpha^0_{ij}}\right), \quad k = 1, .., \kappa, \quad \{i,j\} \in E \tag{3}$$

To better illustrate these ideas, consider Fig 1. In the example, we have considered the nobel_germany network from SNDlib [31] with $C = 4$ controllers. The controller nodes are shown as large red circles. The tree subgraph is shown in solid lines, where the thickness of the links is proportional to the level of upgrade: the thinnest link is not upgraded ($k = 0$), while thickest links are upgraded to level $k = 2$.

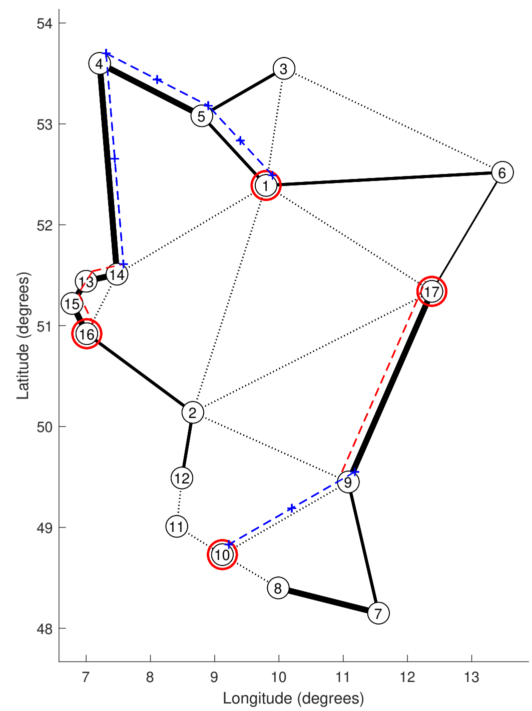


Figure 1: Nobel_germany network with $C = 4$ controllers. The tree subgraph is shown in solid lines. The controller nodes are indicated by large red circles. The primary paths and backup paths for nodes 9 and 14 to their primary and backup controllers are shown by red dashed lines and by blue dashed lines with '+' markers, respectively.

Note that the thickest links are those that serve the paths of nodes further away from the controllers. To show this, the primary paths of nodes 9 and 14 to their primary controllers are depicted as red dashed lines, while the backup paths to their backup controllers are depicted as dashed blue lines with '+' markers. The required availabilities are $\lambda_p = 0.999$ and $\lambda_b = 0.99$.

Note that node 14 connects to the controller in node 16 as its primary and to the controller in node 1 as its backup. The shortest path to the backup controller is using the direct link $\{1, 14\}$. However this link does not belong to the tree subgraph and its length is too long to allow for λ_b to be achieved. Therefore, the backup path is routed using the upgraded links.

Also note that node 9 connects to the controller in node 17 as its primary and to the controller in node 10 as its backup. Although the controller in node 10 is closest, the controller in node 17 still satisfies the maximum delay guarantee and allows for λ_p to be achieved, since the direct link belongs to the tree subgraph and is upgraded to level $k = 2$. Although the link connecting node 9 to the controller in node 10 is not available enough for the primary path, it does satisfy the smaller backup availability without needing to be upgraded, and therefore node 10 serves as the backup controller.

4. Joint Optimization of Controller Placement and Spine Upgrade

The optimization problem we address is the controller placement problem in SDN, jointly with the problem of minimizing the cost of the upgraded links on a high-availability tree subgraph (the spine). Each node is guaranteed to connect to two controllers, one as primary and one as backup, via a pair of node-disjoint paths. Maximum delay guarantees are imposed between the controllers and between the nodes and their primary controllers. Minimum availability requirements are also imposed for the primary and backup paths of each node to their controllers, in order to achieve a target end-to-end availability.

The optimization problem is formulated as an ILP model. The spine is modelled as a spanning tree. After the ILP has been solved, the non-upgraded links connecting to leaves are pruned. The resulting subgraph is a tree connecting the upgraded links and such that all the links connecting to leaves are upgraded. This makes clear which links constitute the smallest connected improved subgraph contained in the spanning tree.

The following additional definitions are needed to formulate the ILP model. Let A designate the the set of arcs or directed links, such that for each link $\{i, j\} \in E$ there is a pair of symmetrical arcs $(i, j), (j, i) \in A$. The set of nodes adjacent to each node $i \in N$ is denoted by $V(i)$, formally $V(i) = \{j \in N : \{i, j\} \in E\}, \forall i \in N$. The delay between two nodes i and j is represented by d_{ij}, and defined, as in [9, 13], by the shortest distance between those nodes.

Consider the following parameters:

- C number of controllers
- D_{sc} maximum delay between a node and its primary controller
- D_{cc} maximum delay between any pair of controllers
- λ_p target availability for the primary path
- λ_b target availability for the backup path
- λ target end-to-end availability: $1-(1-\lambda_p)(1-\lambda_b) \geq \lambda$
- α_{ij}^k availability of link $\{i, j\} \in E$ for default level ($k = 0$) or for upgraded level $k > 0$
- ρ arbitrary node referred as the root node to model the spanning tree for the spine
- t_i^s binary parameter that is 1 if $i = s$, and 0 otherwise

and the following decision variables:

- y_i binary variable that is 1 if a controller is placed in node i, and 0 otherwise
- a_i^s binary variable that is 1 if the primary controller of node s is placed in node i, and 0 otherwise
- b_i^s binary variable that is 1 if the backup controller of node s is placed in node i, and 0 otherwise
- z_{ij}^k binary variable that is 1 if link $\{i, j\} \in E$ is upgraded to level k, and 0 otherwise ($k = 1, ..., \kappa$)
- x_{ij}^{s0} binary variable that is 1 if arc $(i, j) \in A$ belongs to the primary path of node s to its primary controller when link $\{i, j\} \in E$ is not upgraded, and 0 otherwise
- x_{ij}^{sk} binary variable that is 1 if arc $(i, j) \in A$ belongs to the primary path of node s to its primary controller when link $\{i, j\} \in E$ is upgraded to level k, and 0 otherwise ($k = 1, ..., \kappa$)
- u_{ij}^{s0} binary variable that is 1 if arc $(i, j) \in A$ belongs to the backup path of node s to its backup controller when link $\{i, j\} \in E$ is not upgraded, and 0 otherwise
- u_{ij}^{sk} binary variable that is 1 if arc $(i, j) \in A$ belongs to the backup path of node s to its backup controller when link $\{i, j\} \in E$ is upgraded to level k, and 0 otherwise ($k = 1, ..., \kappa$)
- β_{ij}^{σ} binary variable that is 1 if arc $(i, j) \in A$ is in the path from node σ to root node ρ, and 0 otherwise
- θ_{ij} binary variable that is 1 if arc $(i, j) \in A$ belongs to the spanning tree, and 0 otherwise

Note that the definition of decision variables x_{ij}^{sk} is separated into x_{ij}^{s0} and x_{ij}^{sk}, with $k \geq 1$; similarly with decision variables u_{ij}^{s0} and u_{ij}^{sk}. This is done to make the ILP formulation more clear, namely when introducing the link upgrade constraints.

The availability guarantee constraints are nonlinear in nature. Previously, the constraints were linearized by using approximate methods, but in [17] we introduced an exact linearization for such constraints. Since, it is not possible to linearize the end-to-end availability guarantees for a pair of paths, we employ λ_p and λ_b target availabilities (as explained in the previous section) to achieve linearized expressions for such constraints.We now present, in Proposition 1, the linearized expressions for the availability constraints to be used in the ILP model (as used in [18]). To make the paper self-contained the proof of Proposition 1, derived in [18] is given in the Appendix.

Proposition 1. *The linearized expression related to the availability of the primary path of node s, assuming links can be upgraded up to κ levels, can be expressed as*

$$\mathcal{L}_s = \sum_{(i,j)\in A} \sum_{k=0}^{\kappa} x_{ij}^{sk} \log(\alpha_{ij}^k) \tag{4}$$

The ILP model for our joint optimization of controller placement and spine upgrade is given by:

$$\min \sum_{k=1}^{\kappa} \sum_{\{i,j\}\in E} c_{ij}^k z_{ij}^k \quad (5)$$

s.t.

Controller placement constraints:

$$\sum_{i\in N} y_i = C \quad (6)$$

$$\sum_{\substack{j\in N:\\ d_{ij}\le D_{sc}}} y_j \ge 1 \qquad i \in N \quad (7)$$

$$y_i + y_j \le 1 \qquad i\in N, j\in N\backslash\{i\} : d_{ij} > D_{cc} \quad (8)$$

Controller redundancy routing via node-disjoint paths:

$$\sum_{k=0}^{\kappa} \sum_{j\in V(i)} \left(x_{ij}^{sk} - x_{ji}^{sk}\right) = t_i^s - a_i^s \quad s\in N,\ i\in N \quad (9)$$

$$\sum_{k=0}^{\kappa} \sum_{j\in V(i)} \left(u_{ij}^{sk} - u_{ji}^{sk}\right) = t_i^s - b_i^s \quad s\in N,\ i\in N \quad (10)$$

$$\sum_{k=0}^{\kappa} \sum_{j\in V(i)} \left(x_{ji}^{sk} + u_{ji}^{sk}\right) \le 1 \qquad s\in N,\ i\in N\backslash\{s\} \quad (11)$$

$$\sum_{k=0}^{\kappa} \sum_{(i,j)\in A} d_{ij} x_{ij}^{sk} \le D_{sc} \qquad s\in N \quad (12)$$

$$a_i^s + b_i^s \le y_i \qquad s\in N,\ i\in N\backslash\{s\} \quad (13)$$

$$a_s^s + b_s^s = 2y_s \qquad s\in N \quad (14)$$

$$\sum_{i\in N} \left(a_i^s + b_i^s\right) = 2 \qquad s\in N \quad (15)$$

Link upgrade constraints:

$$x_{ij}^{s0} + x_{ji}^{s0} \le 1 - z_{ij}^k \quad s\in N,\ \{i,j\}\in E,\ k=1,...,\kappa \quad (16)$$

$$x_{ij}^{sk} + x_{ji}^{sk} \le z_{ij}^k \quad s\in N,\ \{i,j\}\in E,\ k=1,...,\kappa \quad (17)$$

$$u_{ij}^{s0} + u_{ji}^{s0} \le 1 - z_{ij}^k \quad s\in N,\ \{i,j\}\in E,\ k=1,...,\kappa \quad (18)$$

$$u_{ij}^{sk} + u_{ji}^{sk} \le z_{ij}^k \quad s\in N,\ \{i,j\}\in E,\ k=1,...,\kappa \quad (19)$$

$$\sum_{k=0}^{\kappa} z_{ij}^k \le 1 \qquad \{i,j\}\in E \quad (20)$$

Path availability guarantees:

$$\sum_{k=0}^{\kappa} \sum_{\{i,j\}\in E} \left(x_{ij}^{sk} + x_{ji}^{sk}\right) \log(\alpha_{ij}^k) \ge \log(\lambda_p) \quad s\in N \quad (21)$$

$$\sum_{k=0}^{\kappa} \sum_{\{i,j\}\in E} \left(u_{ij}^{sk} + u_{ji}^{sk}\right) \log(\alpha_{ij}^k) \ge \log(\lambda_b) \quad s\in N \quad (22)$$

Spanning tree constraints:

$$\sum_{j\in V(i)} (\beta_{ij}^\sigma - \beta_{ji}^\sigma) = \begin{cases} 1 & i=\sigma \\ 0 & i\neq\sigma \end{cases} \qquad i\in N\backslash\{\rho\},\ \sigma\in N\backslash\{\rho\} \quad (23)$$

$$\theta_{ij} \ge \beta_{ij}^\sigma \qquad (i,j)\in A,\ \sigma\in N\backslash\{\rho\} \quad (24)$$

$$\sum_{j\in V(i)} \theta_{ij} = \begin{cases} 1 & i\neq\rho \\ 0 & i=\rho \end{cases} \qquad i\in N \quad (25)$$

$$z_{ij}^k \le \theta_{ij} + \theta_{ji} \qquad \{i,j\}\in E,\ k=1,..,\kappa \quad (26)$$

Variable domain constraints:

$$y_i \in \{0,1\} \qquad i\in N \quad (27)$$

$$x_{ij}^{sk} \in \{0,1\} \qquad s\in N,\ (i,j)\in A,\ k=0,...,\kappa \quad (28)$$

$$u_{ij}^{sk} \in \{0,1\} \qquad s\in N,\ (i,j)\in A,\ k=0,...,\kappa \quad (29)$$

$$a_i^s \in \{0,1\} \qquad s\in N,\ i\in N \quad (30)$$

$$b_i^s \in \{0,1\} \qquad s\in N,\ i\in N \quad (31)$$

$$z_{ij}^k \in \{0,1\} \qquad \{i,j\}\in E,\ k=1,...,\kappa \quad (32)$$

$$\theta_{ij} \in \{0,1\} \qquad (i,j)\in A \quad (33)$$

$$\beta_{ij}^\sigma \in \{0,1\} \qquad (i,j)\in A,\ \sigma\in N\backslash\{\rho\} \quad (34)$$

The objective function given by (5) aims to minimize the cost of upgrading the links in the spanning tree.

Constraint (6) guarantees that a given number C of nodes host controllers. Constraints (7) guarantee that for each node s, there is a controller distanced at most D_{sc} (maximum SC delay) from it. Constraints (8) guarantee that any two controllers are distanced at most Dcc (maximum CC delay) from each other.

Constraints (9) are the flow conservation constraints for the primary path of node s to its primary controller, located at node i such that $a_i^s = 1$. Constraints (10) are the flow conservation constraints for the backup path of node s to its backup controller, located at node i such that $b_i^s = 1$. Constraints (11) guarantee that the primary and backup paths are node-disjoint. Constraints (12) guarantee that the primary path length does not exceed D_{sc}.

Constraints (13) guarantee that any primary or backup controller is placed in a controller node. Constraints (14) guarantee that if node s is a controller node, then it is managed by the controller deployed there. Constraints (15) guarantee that for each node s, there is a primary controller and a backup controller. If node s is not a controller node, due to constraints (13), the primary and backup controller nodes must be distinct.

Constraints (16) guarantee that variables x_{ij}^{s0} are assigned to the non-upgraded arcs of the primary path from node s to its primary controller. Constraints (17) guarantee that variables x_{ij}^{sk} are assigned to the arcs of the primary path for node s to its primary controller, that are upgraded to level $k = 1,...,\kappa$. Constraints (18) guarantee that variables u_{ij}^{s0} are assigned to the non-upgraded arcs of the backup path from node s to its backup controller. Constraints (19) guarantee that variables u_{ij}^{sk} are assigned to the arcs of the backup path for node s to its backup controller, that are upgraded to level $k = 1,...,\kappa$. Constraints (20) guarantee that each link is upgraded to one and only one specific level k, or is not upgraded at all.

Constraints (21) guarantee that the primary path of each node to its primary controller has a minimum availability of λ_p. Constraints (22) guarantee that the backup path of each node to its backup controller has a minimum availability of λ_b. Recall that in this work, we consider $\lambda_p > \lambda_b$ and that $1-(1-\lambda_p)(1-\lambda_b) \ge \lambda$, i.e., to achieve an end-to-end availability of at least λ.

Constraints (23) guarantee a routing path from any node σ to the root node ρ. Constraints (24) account for the spanning tree links given by the routing paths from σ to ρ. Constraints (25) guarantee a directed spanning tree towards the root node ρ. Constraints (26) guarantee that the upgraded links belong to the spanning tree.

Finally, constraints (27)-(34) are the variable domain constraints.

We would like to point out that, the primary paths do not have to be on the spanning tree; otherwise we might not be able to ensure the SC delay is at most D_{sc}. The backup paths, each node-disjoint with the corresponding primary path, may use links of the upgrade subgraph if that is decisive to achieve the required path pair availability.

Finally, we prune the spanning tree, in order to obtain the smallest connected improved subgraph contained in the spanning tree. After the ILP has been solved, we know which links have been upgraded in the spanning tree. We check to see if any links connecting the leaves have not been upgraded. If so, we prune these links from the spanning tree and we repeat the pruning process on the new tree.

5. Comparing ILP Formulations

There are two ILP formulations that were used to define the joint optimization problem of controller placement and upgrading the link availability [17]-[18]. In [17], only one level of upgrade was considered, i.e. $\kappa = 1$. It was assumed that the upgrade was given by the installation of a parallel link (alternative path) with the same availability as the original link. Therefore, in [17], the upgraded availability was given by $a_{ij}^1 = a_{ij}^0(2 - a_{ij}^0)$. Moreover, neither path nor controller redundancy were considered. Also, the upgraded links did not have to belong to any connected subgraph. These simplifications led to a first and more naive formulation of the model. Besides the decision variables described in Section 4, consider the additional decision variables:

- χ_{ij}^s binary variable that is 1 if arc $(i,j) \in A$ belongs to the path of node s to its controller, and 0 otherwise
- w_{ij}^s binary variable that is 1 if arc $(i,j) \in A$ belongs to the path of node s to its controller and is upgraded (to level $k = 1$), i.e., $w_{ij}^s = x_{ij}^s \cdot z_{ij}^1$

In [18], several levels of upgrade were considered and the upgrade model follows the one described in Section 3, which is more realistic than the one used in [17]. This led to a more general and compact formulation for the problem, where variables χ_{ij}^s and w_{ij}^s were aggregated into the variables x_{ij}^{sk}. This variable aggregation allowed for the availability constraints to be expressed in the more general and compact form (4). In general, variable aggregation leads to a more efficient model, in terms of runtime. Moreover, in [18], path redundancy was considered: each node was guaranteed to connect to its controller via a pair of node-disjoint paths. Moreover, the upgraded links were imposed to belong to a spanning tree subgraph.

To compare the two formulations, consider the availability link upgrade model described in Section 3, with only one level of upgrade, i.e. $\kappa = 1$. Neither path nor controller redundancy is considered and the upgraded links do not need to belong to any tree subgraph. Therefore, the problem is to jointly optimize the controller placement and the availability link upgrade, under delay and availability guarantees. Since, there is only one path from each node to its controller, the path availability guarantee is given by λ.

The disaggregated ILP model, equivalent to the one used in [17], is given by:

$$\min \sum_{\{i,j\}\in E} c_{ij}^1 z_{ij}^1$$

s.t.

(6) − (8), (27), (30)

$$\sum_{j\in V(i)} \left(\chi_{ij}^s - \chi_{ji}^s\right) = t_i^s - a_i^s \quad s \in N,\ i \in N \tag{35}$$

$$\sum_{(i,j)\in A} d_{ij}\chi_{ij}^s \le D_{sc} \quad s \in N \tag{36}$$

$$a_s^s = y_s \quad s \in N \tag{37}$$

$$a_i^s \le y_i \quad s \in N,\ i \in N \tag{38}$$

$$z_{ij}^1 \le \sum_{s\in N} \left(\chi_{ij}^s + \chi_{ji}^s\right) \quad \{i,j\} \in E \tag{39}$$

$$z_{ji}^1 = z_{ij}^1 \quad \{i,j\} \in E \tag{40}$$

$$w_{ij}^s \le \chi_{ij}^s \quad s \in N,\ (i,j) \in A \tag{41}$$

$$w_{ij}^s \le z_{ij}^1 \quad s \in N,\ (i,j) \in A \tag{42}$$

$$w_{ij}^s \ge \chi_{ij}^s + z_{ij}^1 - 1 \quad s \in N,\ (i,j) \in A \tag{43}$$

$$\sum_{\{i,j\}\in A} \left[\chi_{ij}^s \log(\alpha_{ij}^0) + w_{ij}^s \left(\log(\alpha_{ij}^1) - \log(\alpha_{ij}^0)\right)\right] \ge \log(\lambda) \quad s \in N \tag{44}$$

$$\chi_{ij}^s,\ w_{ij}^s \in \{0,1\} \quad s \in N,\ (i,j) \in A \tag{45}$$

$$z_{ij}^1 \in \{0,1\} \quad (i,j) \in A \tag{46}$$

Recall that there is only one level of upgrade, i.e., $\kappa = 1$. As in (5), the objective function aims to minimize the cost of the upgraded links.

Constraints (35) are the flow conservation constraints of each node to its controller. Constraints (36) guarantee that the control paths do not exceed D_{sc}. Constraints (38) guarantee that any destination of a control path is placed in a controller node. Constraints (37) guarantee that if node s is a controller node, it is managed by the controller deployed there. These constraints are the equivalent to (9), (12)-(14), without controller redundancy.

Constraints (39) guarantee that the upgraded arcs serve at least one control path. Constraints (40) account for both arcs of a link, meaning that if one arc is upgraded, then the arc in the opposite direction is upgraded too. This results in the corresponding link being upgraded.

Constraints (41)-(43) are the linearization constraints for $w_{ij}^s = \chi_{ij}^s \cdot z_{ij}^1$. The variables w_{ij}^s are auxiliary variables

# var	Aggreg	$\lvert N\rvert^2 + 4\lvert E\rvert \cdot \lvert N\rvert + \lvert N\rvert + \lvert E\rvert$
	Disagg	$\lvert N\rvert^2 + 4\lvert E\rvert \cdot \lvert N\rvert + \lvert N\rvert + 2\lvert E\rvert$
# cons	Aggreg	$1 + 3\lvert N\rvert^2 + 3\lvert N\rvert + 2\lvert E\rvert \cdot \lvert N\rvert$
	Disagg	$1 + 3\lvert N\rvert^3 + 3\lvert N\rvert + 2\lvert E\rvert + 6\lvert E\rvert \cdot \lvert N\rvert$

Table 1
Number of variables and constraints for the disaggregated and aggregated models

used to define constraints (44). These constraints result from the linearization of the availability constraints (see details in [17]), which guarantee that the control paths have availability values of at least λ.

The equivalent aggregated ILP model is given by:

$$\sum_{\{i,j\}\in E} c_{ij}^1 z_{ij}^1$$

s.t.

$$(6) - (9),\ (12),\ (37) - (38),\ (16) - (17)$$

$$(27) - (28),\ (30),\ (32)$$

$$\sum_{k=0}^{1} \sum_{\{i,j\}\in E} \left(x_{ij}^{sk} + x_{ji}^{sk} \right) \log(\alpha_{ij}^k) \geq \log(\lambda) \quad s \in N \quad (47)$$

In Table 1, the total number of variables (# var) and total number of constraints (# cons) are shown for both the disaggregated model (Disagg) and aggregated model (Aggreg). In both models, variables y_i and a_i^s are present which account for N and $|N|^2$ variables, respectively. Additionally, the disaggregated model makes use of variables χ_{ij}^s and auxiliary variables w_{ij}^s which together account for $4|E| \cdot |N|$ variables. In turn, the aggregated model makes use of variables x_{ij}^{sk} with $k = 0, 1$, also accounting for $4|E| \cdot |N|$ variables. However, for the aggregated model variables z_{ij}^1 are defined for $\{i, j\} \in E$, while in the disaggregated model they are defined for $(i, j) \in A$. Therefore, the disaggregated model has $2|E|$ more variables than its aggregated counterpart.

Concerning the number of constraints, note that both models use the CPP constraints (6)-(8) which accounts for *at most* $1+|N|^2+|N|\cdot(|N|-1)$ constraints. They both also use the primary controller allocation constraints (37)-(38) which account for $|N|^2 + |N|$ constraints. The flow conservation (9) or (35), and primary path delay constraints (12) or (36), account for a total of $|N|^2+|N|$ constraints. The target availability constraints (44) or (47) account for $|N|$ constraints. The remaining constraints linking variables z_{ij}^1 to the flow variables χ_{ij}^s or x_{ij}^{sk}, account for a total of $2|E| + 6|E| \cdot |N|$ constraints in the disaggregated model and only $2|E| \cdot |N|$ constraints in the aggregated one (for the total number of constraints, please refer to Table 1).

The computational results presented in Section 7.1 show that the aggregated model is more efficient to find the optimal solutions, in terms of runtime as expected from Table 1.

6. Heuristic Method

In [17], a heuristic was proposed to solve the CPP and link upgrade problem. The heuristic consisted in two steps. The first step was to solve an ILP problem only for the controller placement under delay requirements. The second step, was solving the joint ILP problem, but fixing the controller placements given by the first step.

In this work, the problem considers that each node connects also to a backup controller via a node-disjoint path to the primary control path. Moreover, the upgraded links are required to belong to a tree subgraph. Given these particularities, the above mentioned heuristic performs poorly for our problem. Therefore, we propose a heuristic that also consists of two steps. In the first step, the optimization model (5)-(34) is solved, but considering that either the links are not upgraded (level 0) or the links are upgraded all the way to level $k = \kappa$. The intermediate levels $k = 1, ..., \kappa - 1$ are not considered at this stage. The first step provides us with a 'good enough' controller placement solution.

Hence, in the second step, the optimization model is solved for $k = 0, 1, ..., \kappa$, but with fixed controller placements given by the first step. This phase optimizes the spanning tree and upgrade cost for that particular CPP solution. As for the exact method, the spanning tree is then pruned to discard unnecessary non-upgraded links.

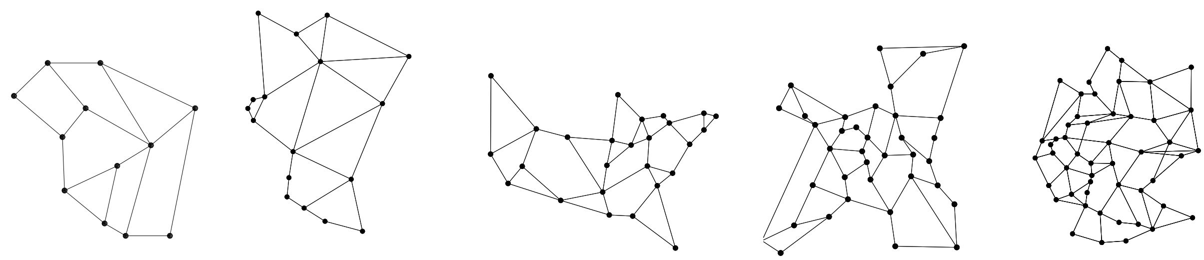
The computational results presented in Section 7.3 show that the heuristic is a good compromise, when the exact method becomes computationally impractical.

7. Computational Results

In this section, several computational results are presented. The first computational results that are put forward and discussed refer to Section 5, where the two ILP formulations are compared. Then, the computational results for the joint optimization model proposed in Section 4, are presented and discussed. These results are compared with those in [18], to compare the gains in terms of upgrade costs of having either path redundancy to the primary controller, or having path redundancy to a primary and a backup controller (controller redundancy). A brief analysis between path versus controller redundancy is also present in [19], where we focused mainly on the performance of the geodiverse solutions. Finally, comparison between the solutions and runtimes of the joint optimization ILP model with the heuristic method described in Section 6, are discussed. The computational results show that the heuristic is a good compromise when the joint optimization ILP model becomes impractical to obtain the optimal solutions.

We have considered five networks from SNDlib [31] for the test instances, whose topologies are shown in Fig. 2. The characteristics for these networks are summarized in Table 2, which shows the number of nodes, the number of links, the average node degree and the graph diameter (in km) for each network. The graph diameter of a network, D_g, is the longest shortest path between any two nodes of the network.

All the exact and heuristic methods were implemented in C/C++, using the Callable libraries of CPLEX 12.8 to solve the ILP models. All instances were run in an 8 core Intel Core i7 PC with 64 GB of RAM, running at 3.6 GHz.



(a) (b) (c) (d) (e)

Figure 2: Graphs of the networks (a) polska, (b) nobel_germany, (c) janos_us, (d) cost266, (e) germany50

Network	#nodes	#links	avg deg	diameter [km]
polska	12	18	3.00	811
nobel_germany	17	26	3.06	790
janos_us	26	42	3.23	4690
cost266	37	57	3.08	4032
germany50	50	88	3.52	934

Table 2
Characteristics of the networks

7.1. Comparing ILP Formulations

To compare the disaggregated and aggregated models described in Section 5, we have considered the three larger SNDlib networks of Table 2: janos_us, cost266 and germany50. The polska and nobel_germany networks are too small to render significant differences in the runtimes of the models.

Recall that to compare both formulations, we only consider one level of upgrade $\kappa = 1$ and neither path nor controller redundancy is considered. The computational results are shown in Table 3. For janos_us and cost266 the required availability was $\lambda = 0.9965$ due to their large graph diameter, while for germany50 the required availability was $\lambda = 0.999$. The tables show for each instance the D_{sc} and D_{cc} values considered which are given as percentages of the graph diameter D_g [13, 17, 18]. We assumed D_{sc} to be 30%, 35% and 40%, while D_{cc} was chosen as 60%, 65% and 70% of D_g, for all networks. The number of controllers C was chosen to be the minimum possible for each pair of (D_{sc}, D_{cc}) values.

The tables show the optimal value of the cost upgrade (column 'cost') and the number of upgraded links (column '#upg'). These values are the same for both models. The runtimes (in seconds) for the aggregated and disaggregated models are shown in columns 't_a(s)' and 't_d(s)', respectively. The total runtime for each method is shown in the last row for each network.

It is clear for all instances, that the aggregated model is faster in obtaining the optimal solutions when compared to the disaggregated one. Although for janos_us, the times are small for both models, the total runtime results in a reduction of a tenth for the aggregated model. The differences become more significant for cost266, where the aggregated model obtains the optimal solution under 10 seconds for each instance. For germany50, the disaggregated model becomes computationally too expensive, while the aggregated model can still find the optimal solution in reasonable runtime.

network	D_{sc}	D_{cc}	C	cost	#upg	t_a(s)	t_d(s)
janos_us $\lambda = 0.9965$	30%	60%	4	6624.41	16	1.11	0.23
		65%	4	6065.73	14	1.19	0.32
		70%	4	6065.73	14	1.59	0.35
	35%	60%	4	6624.41	16	1.33	0.23
		65%	4	6065.73	14	2.45	0.28
		70%	4	6065.73	14	3.88	0.31
	40%	60%	4	6624.41	16	1.89	0.23
		65%	4	6065.73	14	2.77	0.28
		70%	4	6065.73	14	7.31	0.30
						23.52	2.53
cost266 $\lambda = 0.9965$	30%	60%	4	5100.87	16	4.08	1.11
		65%	4	4845.10	16	11.37	2.01
		70%	4	4845.10	16	16.17	2.62
	35%	60%	4	5100.87	16	14.54	1.84
		65%	4	4845.10	16	21.77	3.51
		70%	4	4845.10	16	43.28	5.41
	40%	60%	4	5100.87	16	25.45	2.35
		65%	4	4845.10	16	59.08	6.20
		70%	4	4845.10	16	64.35	8.99
						260.08	34.04
germany50 $\lambda = 0.999$	30%	60%	4	1347.48	21	519.55	18.38
		65%	4	1238.65	20	141.10	19.36
		70%	4	1131.22	18	229.26	30.65
	35%	60%	4	1110.42	19	2986.46	65.46
		65%	4	987.04	16	2714.66	92.89
		70%	3	2224.31	37	408.82	18.42
	40%	60%	4	1110.42	19	4647.76	727.78
		65%	4	987.04	16	8668.96	913.30
		70%	3	2193.12	36	14534.52	72.12
						34851.09	1958.36

Table 3
Comparison of runtimes for the disaggregated and aggregated models.

7.2. Joint Optimization of Controller Placement and Spine Upgrade

The joint optimization ILP model (5)-(34) was tested for polska, nobel _germany and janos_us. Since path redundancy without backup controllers was addressed in [18], and the computational results were reported for polska and nobel_germany, we compare these results with the ones we obtain with controller redundancy. Therefore, we have chosen (D_{sc}, D_{cc}) to be (35%,70%) and (40%,75%) for polska, (35%,65%) and (40%,70%) for nobel_germany, and (35%, 60%) and (40%,65%) for janos_us.

The number of controllers is incremented from the minimum possible number to the maximum possible. The minimum and maximum numbers are determined by D_{sc} and D_{cc}, respectively. While a minimum number of controllers must exist so that the SC delays do not exceed D_{sc}, a maximum number of controllers is dictated by the fact that the CC delays cannot exceed D_{cc}. Minimizing the upgrade cost for each possible value of C, provides us with the Pareto front for the problem of minimizing the number of controllers and minimizing the upgrade cost. We show the trade-off between the number of controllers and the upgrade cost, for the network operator to weigh the benefits of each solution.

A Pareto or nondominated solution is a solution such that any other solution to the problem which has a smaller number of controllers must have a greater upgrade cost, or such that any other solution with a smaller upgrade cost must have a greater number of controllers. In other words, a nondominated solution is such that it is not possible to improve both objectives simultaneously even further.

For all the networks, we have considered minimum availability values for the primary paths given by $\lambda_p = 0.999$, and for the backup paths given by $\lambda_b = 0.99$, to achieve end-to-end availabilities of at least $\lambda = 0.99999$. To obtain the required availabilities, we have considered $\kappa = 4$ levels of link upgrade. In each level $k = 1, ..., \kappa$, the link unavailability is reduced by a factor of $\epsilon = 0.5$.

The computational results are shown in Tables 4, 5 and 6 for polska, nobel_germany and janos_us, respectively. For polska and nobel_germany, the controller redundancy solution is compared with the path redundancy solution reported in [18]. The tables show the D_{sc}, D_{cc} and C values for each instance. The following columns show the results for the solution of our optimization problem with controller redundancy: column 'cost' shows the upgrade cost, column 't(s)' shows the runtime in seconds, and the following columns show the total number of upgraded links (column 'total'), and the number of links upgraded to each level $k = 1, 2, 3, 4$. The final columns in Tables 4 and 5, show the results for the solution with path redundancy.

In Table 4, for polska, we can see that the upgrade cost of having controller redundancy is smaller than the cost of having only path redundancy. Since controller redundancy also protects against node failures, not just link failures, it is desirable to choose controller redundancy. Even more so, since the link availability upgrade is cheaper for controller redundancy. Surprisingly, the controller redundancy problem is easier to solve than path redundancy alone, as shown by the runtimes.

The values marked with an asterisk (*), indicate the nondominated solutions for each case. Recall that the nondominated solutions are such that no other solution can have a lower number of controllers and upgrade cost simultaneously. In general, increasing the number of controllers will allow the upgrade cost to decrease. Indeed, note that in Table 4 that the upgrade cost decreases as the number of controllers increases. However, after a certain number of controllers are deployed, the upgrade cost does not decrease further for most of the instances. Also the most significant cost reductions occur when the number of controllers is still small. As the number of controllers increases, the cost reduction becomes less significant or stalls altogether. Since we want to keep the number of controllers small, this confirms that we do not have to deploy many controllers to have a good trade-off between the upgrade cost and the number of controllers.

Note that in general, when the cost decreases, the total number of upgraded links also decreases. However, this is not always the case. For $D_{sc} = 35\%$ and $D_{cc} = 70\%$, the path redundancy solutions show a decrease in cost from 4 to 5 controllers, while the total number of upgraded links is 9 for both. We can see that the more costly solution has two links upgraded to level $k = 4$, while for the cheaper solution none of the links are upgraded to level $k = 4$. This is consistent with the cost function given by (3), which grows exponentially with the level upgrade. Also note that from 6 to 7 controllers, there is a cost reduction but the number of upgraded links actually increases from 8 to 9. This is due to the greater link lengths involved in the more costly solution.

Similar observations can be drawn from Table 5, for nobel_germany. In this network, it is more noticeable the difference in the runtimes for the controller redundancy problem, and for the path redundancy alone. Once again, including controller redundancy makes the problem easier to solve.

Note that for path redundancy with $D_{sc} = 35\%$ and $D_{cc} = 65\%$, deploying more than 9 controllers actually incurs in an increase in the cost upgrade (values shown in bold). This is due to the fact that the 9 controller placements cannot serve 10 controllers, because of the D_{cc} requirement. Consequently, the controllers are repositioned leading to higher distances for a few switches and, thus, forcing the link upgrade cost to increase.

Another interesting observation is that for $C = 2$, the controller redundancy solution is more expensive. Since only 2 controllers are deployed in the network, each switch must connect to both, one as primary and one as backup. This leads to very large backup paths, forcing the upgrade cost to increase.

For this network, we also see some cases where the decrease in cost is not accompanied by a decrease in the total number of upgraded links. For example, when $D_{sc} = 35\%$ and $D_{cc} = 65\%$, we can see that the number of upgraded links actually increased from 14 to 16, when going from 2 to 3 controllers, although the cost reduced significantly. Note that the more costly solution has 6 links upgraded to level $k = 4$, while the cheaper solution has 2 more upgraded links but all upgraded to levels $k < 4$.

For janos_us we only show the solutions for our controller redundant problem in Table 6. Note that the observations made for the trade-off between the number of controllers and the cost also hold for janos_us. The cost reduction is more significant when the number of controllers is

			Controller redundancy							Path redundancy						
					no. upg links							no. upg links				
D_{sc}	D_{cc}	C	cost	t(s)	total	1	2	3	4	cost	t(s)	total	1	2	3	4
35%	70%	3	2263.82*	0.97	10	5	3	2	0	5054.43*	43.69	11	1	0	3	7
		4	1484.03*	0.96	9	7	2	0	0	2995.09*	59.39	9	2	1	4	2
		5	1097.25*	0.56	6	5	1	0	0	2469.68*	43.28	9	2	5	2	0
		6	936.44*	0.30	5	4	1	0	0	2299.17*	6.66	8	2	4	2	0
		7	832.47*	0.20	4	3	1	0	0	2266.59*	3.60	9	3	3	3	0
		8	832.47	0.17	4	3	1	0	0	2266.59	2.08	9	3	3	3	0
		9	–	0.02	–					–	0.05	–	–	–	–	–
				3.18							158.75					
40%	75%	3	1977.55*	4.81	11	7	4	0	0	3695.86*	252.18	11	2	2	7	0
		4	1384.91*	2.49	8	7	1	0	0	2635.35*	106.75	9	3	1	5	0
		5	1035.56*	0.81	5	4	1	0	0	2299.17*	40.43	8	2	4	2	0
		6	845.64*	0.39	4	3	1	0	0	2143.90*	30.69	8	3	3	2	0
		7	727.80*	0.35	3	2	1	0	0	1958.83*	5.40	6	2	0	4	0
		8	727.80	0.30	3	2	1	0	0	1829.91*	3.16	5	1	0	4	0
		9	727.80	0.19	3	2	1	0	0	1795.25*	0.69	5	0	2	3	0
		10	–	0.02	–					–	0.03	–	–	–	–	–
				9.35							439.30					

Table 4
Computational results for polska network with availability requirements given by $\lambda_p = 0.999$ and $\lambda_b = 0.99$

still small and from a certain of number of controllers, the cost does not decrease anymore. For this network, the runtimes become very large, when $D_{sc} = 40\%$ and $D_{cc} = 65\%$, rendering the joint optimization ILP model impractical for larger networks. In fact, for $D_{sc} = 40\%$ and $D_{cc} = 70\%$, the ILP model could not retrieve the optimal solution for $C = 2$ after 2 days.

7.3. Heuristic Method

The heuristic method was also implemented and tested. The computational results are shown in Tables 7 and 8 for nobel_germany and janos_us, respectively. The values in bold indicate that the heuristic found the optimal cost and/or the optimal number of upgraded links. The values in italic indicate that the heuristic found a solution with higher cost than the optimal one, but with a total number of upgraded links less than the optimal.

For nobel_germany, we can see in Table 7, that the heuristic obtained solutions much faster than the exact method. Note that the total runtime for the heuristic is 25 seconds compared to 423 seconds when $D_{sc} = 35\%$ and $D_{cc} = 65\%$, and 47 seconds compared to 1884 seconds when $D_{sc} = 40\%$ and $D_{cc} = 70\%$.

We can also see that the heuristic found the optimal solution for 9, 10 and 11 controllers with $D_{sc} = 35\%$ and $D_{cc} = 65\%$, and for 12 controllers with $D_{sc} = 40\%$ and $D_{cc} = 70\%$. However, although it did not find the optimal solution, it did find a solution with the same total number of upgraded links for 2 controllers with $D_{sc} = 35\%$ and $D_{cc} = 65\%$, and for 7 controllers with $D_{sc} = 40\%$ and $D_{cc} = 70\%$. In the former case, we can see that for the heuristic there are 8 links upgraded to level $k = 4$ and only 2 links upgraded to level $k = 2$, while in the optimal solution there are only 6 links upgraded to level $k = 4$ and 4 links upgraded to level $k = 2$.

Moreover, the heuristic also found more costly solutions, but with a smaller number of upgraded links for 3, 4 an 5 controllers with $D_{sc} = 35\%$ and $D_{cc} = 65\%$, and for 2 and 3 controllers with $D_{sc} = 40\%$ and $D_{cc} = 70\%$.

For janos_us, we can see in Table 8, that the heuristic is able to find solutions in reasonable runtimes, while the exact method struggles for $D_{sc} = 40\%$ and $D_{cc} = 65\%$. Note that for all instances with $D_{sc} = 35\%$ and $D_{cc} = 60\%$, the heuristic is able to find either the optimal solution, or a more costly solution but with the optimal total number of upgraded links. This is also true for some cases of $D_{sc} = 40\%$ and $D_{cc} = 65\%$. For all the other cases, the heuristic found a more costly solution, but where the total number of upgraded links is indeed smaller. These observations show that the heuristic is a good compromise between optimality and runtime.

8. Conclusions

In this paper, we have addressed the controller placement and spine design problem, considering delay and availability guarantees while imposing that each switch connects to a primary and to a backup controller, via a pair of node-disjoint paths. This framework offers resiliency against single link or node failures, as well as resiliency against controller failures, while guaranteeing the required control plane performance and availability guarantees.

An ILP model was proposed in [19] for the more complex variant of this work considering geodiversity. It is possible to obtain a simplified version for controller redundancy when assuming zero geodiversity. The clean version of the ILP model considering controller redundancy specifically, was presented in this paper.

			Controller redundancy							Path redundancy						
D_{sc}	D_{cc}	C	cost	t(s)	no. upg links					cost	t(s)	no. upg links				
					total	1	2	3	4			total	1	2	3	4
35%	65%	2	4187.30*	21.434	14	2	4	2	6	3076.88*	170.983	15	5	6	3	1
		3	2116.87*	126.34	16	5	9	2	0	2682.48*	14994.96	16	8	6	2	0
		4	1583.15*	136.19	15	11	3	1	0	2068.35*	696.95	14	7	6	1	0
		5	1215.09*	96.94	13	10	2	1	0	1477.79*	131.70	10	5	4	1	0
		6	986.35*	28.50	8	4	3	1	0	1337.08*	12.30	9	5	3	1	0
		7	894.85*	3.74	6	4	1	1	0	1337.08	11.97	9	5	3	1	0
		8	894.85	3.65	6	4	1	1	0	1337.08	23.16	9	5	3	1	0
		9	894.85	4.23	6	4	1	1	0	1337.08	16.06	9	5	3	1	0
		10	894.85	0.83	6	4	1	1	0	**1556.12**	4.36	9	6	2	1	0
		11	894.85	1.21	6	4	1	1	0	**1556.12**	4.06	9	6	2	1	0
		12	–	0.05	–					–	0.12	–	–	–	–	–
				423.104							16066.613					
40%	70%	2	3419.29*	227.192	16	3	3	8	2	3076.88*	1093.335	15	5	6	3	1
		3	2116.87*	379.94	16	5	9	2	0	2672.08*	2508.29	12	5	3	4	0
		4	1518.69*	389.14	10	5	4	1	0	2068.35*	3379.28	14	7	6	1	0
		5	1110.42*	607.85	12	9	3	0	0	1477.79*	627.54	10	5	4	1	0
		6	817.22*	235.29	8	6	2	0	0	1165.87*	141.77	7	1	5	1	0
		7	585.71*	36.20	6	4	2	0	0	955.85*	51.20	7	2	5	0	0
		8	438.76*	5.44	5	3	2	0	0	886.54*	8.55	7	3	4	0	0
		9	347.27*	0.75	3	3	0	0	0	817.22*	8.95	6	2	4	0	0
		10	347.27	0.79	3	3	0	0	0	817.22	9.96	6	2	4	0	0
		11	347.27	0.83	3	3	0	0	0	817.22	7.50	6	2	4	0	0
		12	347.27	0.58	3	3	0	0	0	817.22	2.24	6	2	4	0	0
		13	–	0.04	–					–	0.14	–	–	–	–	–
				1884.026							7838.744					

Table 5
Computational results for nobel_germany network with availability requirements given by $\lambda_p = 0.999$ and $\lambda_b = 0.99$

The ILP model allows the network operators to obtain a set of solutions representing the trade-off between the number of controllers and the upgrade cost. In general, as the number of controllers increases, the upgrade cost decreases. However, it is desirable from the control plane perspective to have a small number of controllers to minimize intercontroller communication overhead.

We compared two ILP formulations for the simplest form of the controller placement and availability link upgrade problem, and showed that the formulation used in our model is more efficient. Even so, for medium-sized networks, our joint optimization model considering controller redundancy begins to struggle. Therefore, we proposed a heuristic method that has proven to be a good compromise, when the exact ILP becomes impractical. We have seen that when the heuristic is not able to find the optimal solution, it often finds a slightly more costly solution, with a number of upgraded links identical to or lower than that of the optimal solution.

Appendix

Proposition 1. *The linearized expression related to the availability of the primary path of node s, assuming links can be upgraded up to κ levels, can be expressed as*

$$\mathcal{L}_s = \sum_{(i,j)\in A} \sum_{k=0}^{\kappa} x_{ij}^{sk} \log(\alpha_{ij}^{k}) \tag{4}$$

Proof. The availability of the primary path of node s, i.e., of the routing path from s to its primary controller, assuming links can be upgraded up to κ levels, is given by

$$\mathcal{A}_s = \prod_{\substack{(i,j)\in A:\\ x_{ij}^{s0}=1}} \alpha_{ij}^{0} \prod_{\substack{(i,j)\in A:\\ x_{ij}^{s1}=1}} \alpha_{ij}^{1} \cdots \prod_{\substack{(i,j)\in A:\\ x_{ij}^{s\kappa}=1}} \alpha_{ij}^{\kappa}$$

By the binary nature of variables x_{ij}^{sk}, it is possible to show that

$$\begin{aligned}\mathcal{A}_s &= \prod_{(i,j)\in A} \prod_{k=0}^{\kappa} \left[x_{ij}^{sk}\alpha_{ij}^{k} + \left(1 - x_{ij}^{sk}\right)\right] \\ &= \prod_{(i,j)\in A} \prod_{k=0}^{\kappa} \left[1 - x_{ij}^{sk}\left(1 - \alpha_{ij}^{k}\right)\right]\end{aligned}$$

Applying logarithms to linearize the expressions, results in

$$\log(\mathcal{A}_s) = \sum_{(i,j)\in A} \sum_{k=0}^{\kappa} \log\left[1 - x_{ij}^{sk}\left(1 - \alpha_{ij}^{k}\right)\right]$$

Due to the binary nature of variables x_{ij}^{sk}, it is possible to show that $\log(\mathcal{A}_s) = \mathcal{L}_s$. In fact, by definition of variables x_{ij}^{sk}, for each $\{i,j\} \in E$, there is one and only one $k_{ij} \in$

	$D_{sc}=35\%$ $D_{cc}=60\%$							$D_{sc}=40\%$ $D_{cc}=65\%$						
C	cost	t(s)	no. upg links					cost	t(s)	no. upg links				
			total	1	2	3	4			total	1	2	3	4
3	25451.67*	226.28	23	0	9	11	3	25211.15*	23384.93	24	0	7	13	4
4	22812.86*	639.76	22	0	10	11	1	21081.38*	3875.96	22	0	8	10	4
5	20391.00*	288.22	21	1	8	11	1	18789.14*	8021.01	21	1	8	10	2
6	18736.46*	222.36	20	1	11	7	1	16907.25*	1466.03	20	1	8	9	2
7	17659.31*	45.60	19	2	10	6	1	15687.31*	790.97	19	2	9	7	1
8	16809.51*	106.79	18	2	9	6	1	14787.60*	498.79	18	2	8	7	1
9	16015.16*	27.20	17	2	8	6	1	13937.80*	275.26	17	2	7	7	1
10	15302.61*	9.64	16	2	7	6	1	13225.25*	43.81	16	2	6	7	1
11	14793.84*	8.62	15	2	6	6	1	12716.48*	67.65	15	2	5	7	1
12	14292.00*	8.22	14	2	5	6	1	12214.64*	18.74	14	2	4	7	1
13	13806.80*	5.56	13	2	4	6	1	11729.44*	17.30	13	2	3	7	1
14	13603.71*	3.61	12	1	4	6	1	11505.55*	9.09	11	1	5	2	3
15	13603.71	3.00	12	1	4	6	1	11020.35*	7.82	10	1	4	2	3
16	–	0.10	–					10544.85*	2.87	9	1	3	2	3
17								10341.76*	2.04	8	0	3	2	3
18								10341.76	2.01	8	0	3	2	3
19								–	1.38	–				
		1594.93							38485.66					

Table 6
Computational results for janos_us network with availability requirements given by $\lambda_p = 0.999$ and $\lambda_b = 0.99$

	$D_{sc}=35\%$ $D_{cc}=65\%$							$D_{sc}=40\%$ $D_{cc}=70\%$						
C	cost	t(s)	no. upg links					cost	t(s)	no. upg links				
			total	1	2	3	4			total	1	2	3	4
2	4338.41	2.81	**14**	2	2	2	8	4338.41	5.71	*14*	2	2	2	8
3	2360.86	4.56	*13*	3	5	5	0	2447.50	8.24	*14*	4	8	2	0
4	1678.11	5.08	*13*	6	7	0	0	1652.46	9.88	11	6	5	0	0
5	1269.85	3.39	*12*	7	5	0	0	1208.85	13.64	13	11	1	1	0
6	1127.06	2.57	10	5	5	0	0	1078.54	4.16	10	8	1	1	0
7	969.71	1.54	8	6	2	0	0	663.34	1.05	**6**	6	0	0	0
8	1037.64	1.72	8	6	1	1	0	935.06	1.87	6	5	1	0	0
9	**894.85**	1.32	**6**	4	1	1	0	494.21	0.61	4	4	0	0	0
10	**894.85**	0.93	**6**	4	1	1	0	562.84	0.89	5	5	0	0	0
11	**894.85**	1.22	**6**	4	1	1	0	520.55	0.64	4	4	0	0	0
12	–	0.03	–					**347.27**	0.59	**3**	3	0	0	0
13								–	0.02	–				
		25.16							47.29					

Table 7
Heuristic method results for nobel_germany network

$\{1, ..., \kappa\}$ such that $x_{ij}^{sk_{ij}} = 1$. So,

$$\log(\mathcal{A}_s) = \sum_{(i,j)\in A} \log\left[1 - \left(1 - \alpha_{ij}^{sk_{ij}}\right)\right] = \sum_{(i,j)\in A} \log\left(\alpha_{ij}^{sk_{ij}}\right) = \mathcal{L}_s$$

□

Acknowledgements

This work was partially supported by Fundação para a Ciência e a Tecnologia (FCT) under the project grant UIDB/00308/2020 and was financially supported by FEDER Funds and National Funds through FCT under the project CENTRO-01-0145-FEDER-029312. This work was also partially supported by the COST Action CA15127 ("Resilient communication services protecting end user applications from disaster-based failures – RECODIS").

References

[1] S. Jain, et al., B4: Experience with a globally-deployed software defined WAN, ACM SIGCOMM Computer Communication Review 43 (4) (2013) 3–14 (2013).

[2] R. Alvizu, G. Maier, N. Kukreja, A. Pattavina, R. Morro, A. Capello, C. Cavazzoni, Comprehensive survey on T-SDN: Software-defined networking for transport networks, IEEE Communications Surveys & Tutorials 19 (4) (2017) 2232–2283 (2017).

	$D_{sc} = 35\%\ D_{cc} = 60\%$							$D_{sc} = 40\%\ D_{cc} = 65\%$						
C	cost	t(s)	no. upg links					cost	t(s)	no. upg links				
			total	1	2	3	4			total	1	2	3	4
3	**25451.67**	60.97	**23**	0	9	11	3	25452.37	147.66	*23*	0	3	13	7
4	23029.82	49.79	**22**	1	7	11	3	24419.58	441.63	**22**	0	1	15	6
5	21334.38	104.93	**21**	1	7	11	2	20424.97	114.51	**21**	0	8	8	5
6	20484.58	65.93	**20**	1	6	11	2	19525.26	200.26	**20**	0	7	8	5
7	19121.85	43.05	**19**	1	6	10	2	18167.39	123.96	*18*	0	7	6	5
8	17524.15	13.99	**18**	1	8	8	1	16613.35	88.14	*17*	0	7	7	3
9	16084.48	9.71	**17**	2	7	7	1	15446.09	70.57	*16*	1	7	4	4
10	**15302.61**	7.98	**16**	2	7	6	1	14147.13	69.03	*15*	1	7	4	3
11	**14793.84**	6.62	**15**	2	6	6	1	13395.76	23.58	*14*	0	9	2	3
12	**14292.00**	4.71	**14**	2	5	6	1	12740.74	11.65	*13*	1	7	2	3
13	**13806.80**	5.00	**13**	2	4	6	1	12007.39	8.34	*12*	1	6	2	3
14	**13603.71**	4.58	**12**	1	4	6	1	**11505.55**	6.43	**11**	1	5	2	3
15	**13603.71**	3.91	**12**	1	4	6	1	**11020.35**	4.04	**10**	1	4	2	3
16	–	0.05	–					**10544.85**	2.28	**9**	1	3	2	3
17								**10341.76**	1.98	**8**	0	3	2	3
18								**10341.76**	2.14	**8**	0	3	2	3
19								–	0.09	–				
		381.22							1316.29					

Table 8
Heuristic method results for janos_us network

[3] G. Maier, M. Reisslein, Transport SDN at the dawn of the 5G era, Optical Switching and Networking 33 (2019) 34–40 (2019).

[4] S. Chandna, N. Naas, H. Mouftah, Software defined survivable optical interconnect for data centers, Optical Switching and Networking 31 (2019) 86–99 (2019).

[5] Y. Xiong, Z. Li, B. Zhou, X. Dong, Cross-layer shared protection strategy towards data plane in software defined optical networks, Optic Communications 412 (2018) 66–73 (2018).

[6] M. Savi, D. Siracusa, Application-aware service provisioning and restoration in SDN-based multi-layer transport networks, Optical Switching and Networking 30 (2018) 71–84 (2018).

[7] X. Cao, I. Popescu, G. Chen, H. Guo, N. Yoshikane, T. Tsuritani, J. Wu, I. Morita, Optimal and dynamic virtual datacenter provisioning over metro-embedded datacenters with holistic SDN orchestration, Optical Switching and Networking 24 (2017) 1–11 (2017).

[8] Y. Zheng, Z. Ge, X. Zhang, J. Xu, X. Sun, High-reliability optical process level network in smart substation, Optical Switching and Networking 36 (2020) 100552 (2020).

[9] B. Heller, R. Sherwood, N. McKeown, The controller placement problem, in: ACM HotSDN, New York, USA, 2012, pp. 7–12 (2012).

[10] Y. Jiménez, C. Cervelló-Pastor, A. J. García, On the controller placement for designing a distributed SDN control layer, in: IFIP, Trondheim, Norway, 2014, pp. 1–9 (2014).

[11] T. Zhang, A. Bianco, P. Giaccone, The role of inter-controller traffic in SDN controllers placement, in: NFV-SDN, Palo Alto, USA, 2016, pp. 87–92 (2016).

[12] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, P. Tran-Gia, Pareto-optimal resilient controller placement in sdn-based core networks, in: ITC, Shanghai, China, 2013, pp. 1–9 (2013).

[13] N. Perrot, T. Reynaud, Optimal placement of controllers in a resilient SDN architecture, in: DRCN, Paris, France, 2016, pp. 145–151 (2016).

[14] S. Verbrugge, D. Colle, P. Demeester, R. Huelsermann, M. Jaeger, General availability model for multilayer transport networks, in: DRCN, Ischia, Italy, 2005, pp. 1–8 (2005).

[15] A. Alashaikh, T. Gomes, D. Tipper, The spine concept for improving network availability, Computer Networks 82 (2015) 4 – 19 (2015).

[16] A. Alashaikh, D. Tipper, T. Gomes, Embedded network design to support availability differentiation, Annals of Telecommunications 74 (9-10) (2019) 605–623 (2019).

[17] D. Santos, T. Gomes, Controller placement and availability link upgrade problem in SDN networks, in: RNDM, Nicosia, Cyprus, 2019, pp. 1–8 (2019).

[18] D. Santos, T. Gomes, D. Tipper, Software-defined network design driven by availability requirements, in: DRCN, Milan, Italy, 2020, pp. 1–7 (2020).

[19] D. Santos, T. Gomes, D. Tipper, Sdn controller placement with availability upgrade under delay and geodiversity constraints, IEEE Transactions on Network and Service Management 18 (1) (2021) 301–314 (2021).

[20] P. Vizarreta, C. M. Machuca, W. Kellerer, Controller placement strategies for a resilient SDN control plane, in: RNDM, Halmstad, Sweden, 2016, pp. 253–259 (2016).

[21] S. Petale, J. Thangaraj, Failure based controller placement in software defined networks, IEEE Transactions on Network and Service Management Early Access (2019).

[22] M. Tanha, D. Sajjadi, R. Ruby, , J. Pan, Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs, IEEE Transactions on Network and Service Management 15 (3) (2018) 991–1005 (2018).

[23] D. Santos, A. de Sousa, C. M. Machuca, Combined control and data plane robustness of SDN networks against malicious node attacks, in: CNSM 2018, Rome, Italy, 2018, pp. 54–62 (2018).

[24] D. Santos, A. de Sousa, C. M. Machuca, The controller placement problem for robust sdns against malicious node attacks considering the control plane with and without split-brain, Annals of Telecommunications 74 (9) (2019) 575–591 (2019).

[25] G. Nencioni, B. E. Helvik, P. E. Heegaard, Including failure correlation in availability modeling of a software-defined backbone network, IEEE Transactions on Network and Service Management 14 (4) (2017) 1032–1045 (2017).

[26] Y. Hu, W. Wang, X. Gong, X. Que, S. Cheng, On reliability-optimized controller placement for software-defined networks, China Communications 11 (2) (2014) 38–54 (2014).

[27] F. J. Ros, P. M. Ruiz, Five nines of southbound reliability in software-defined networks, in: ACM HotSDN, New York, USA, 2014, pp. 31–36 (2014).

[28] F. J. Ros, P. M. Ruiz, On reliable controller placements in software-defined networks, Computer Communications 77 (2016) 41–51 (2016).

[29] J.-P. Vasseur, M. Pickavet, P. Demeester, Network Recovery – Protection and Restoration of Optical, SONET-SDH, IP, and MPLS, Elsevier, 2004 (2004).

[30] U. Franke, Optimal IT service availability: Shorter outages, or fewer?, IEEE Transactions on Network and Service Management 9 (1) (2012) 22–33 (2012).

[31] S. Orlowski, R. Wessäly, M. Pióro, A. Tomaszewski, SNDlib 1.0–Survivable Network Design library, Networks 55 (3) (2010) 276–286, http://sndlib.zib.de (2010).