

Chapter 2

Connection-Oriented Networks

This chapter focuses on the optimization of connection-oriented networks (CONs). Following a brief introduction to the basics of technologies and protocols used in CONs, we formulate several optimization problems that arise in the context of connection-oriented networking applied to cloud computing and content-oriented services. To address specific attributes of cloud computing and content-oriented services, the optimization problems presented—besides including classical unicast flows—also embrace anycast and multicast network flows. All presented optimization problems are formulated as ILP models. Moreover, for selected optimization problems, we propose and analyze solution algorithms and report results of numerical experiments.

2.1 Introduction

In a connection-oriented network, information (packets, frames, bits) is forwarded along a predefined connection from the source to the destination. More precisely, all data belonging to a particular demand is transmitted along one routing path determined and established before the transmission begins. The telephone network is notable as probably the first CON developed for communication. Following the deployment of telephone networks, the concept of connection has been extended and employed in many network technologies, including MPLS (MultiProtocol Label Switching), ATM (Asynchronous Transfer Mode), WDM (Wavelength Division Multiplexing), and Connection-Oriented Ethernet [1]. We outline these technologies below.

The MPLS architecture was proposed by the Internet Engineering Task Force (IETF) in the standard RFC3013 [2]. MPLS is designed to carry IP packets, but with more efficient traffic engineering and QoS guarantees than classical IP networks. MPLS networks consist of two types of devices: LER (Label Edge Router) located

at the entry and exit points of the MPLS network, and LSR (Label Switch Router) located inside the MPLS network. In the MPLS network, packets are transmitted along a LSP (Label Switch Path) between LERs and LSRs. Each packet header includes a piece of information known as the *label* which indicates a particular LSP. When an MPLS packet enters the LSR, the label is used to find the outgoing port of the LSR where the packet should be forwarded according to the packet's FEC (Forwarding Equivalence Class). It should be noted that in MPLS the labels are local, i.e., each LSR can change the label of a particular LSP for each subsequent link along the path, which is explicitly included in the switching tables of each LSR. This procedure ensures that MPLS is a connection-oriented technology, i.e., all packets assigned to the same FEC follow exactly the same routing path. However, it should be noted that MPLS can also be used in a connectionless mode, if LDP (Label Distribution Protocol) is used for signaling. To classify a packet to a particular FEC class, an IP address or other element of the IP packet header can be used. FEC classes are characterized with various QoS parameters, therefore it is possible to provision different types of traffic in a different ways. Thus, packets included in different FEC classes but with the same source and destination nodes can be assigned to different routing paths. This is the key attribute of MPLS that enables efficient traffic engineering provisioning. For more information on MPLS refer to [1–5].

The ATM architecture was standardized by the ITU-T (International Telecommunication Union—Telecommunication Standardization Sector) in 1987 as the preferred solution for implementing the B-ISDN (Broadband Integrated Services Digital Network) concept. In the late 1980s, ATM was one of the most promising proposals for high-speed networking in backbone networks supported by demand-switched, semipermanent, and permanent broadband connections for both point-to-point and point-to-multipoint applications. The general idea of ATM is very similar to MPLS, which in fact can be regarded as a successor of ATM. More specifically, ATM—like MPLS—is based on the concept of label-switching. In ATM, fixed-size packets known as *cells* arrive in one port of the ATM switch and the label included in the cell header is used to indicate the outgoing port of the switch, while a new label number is assigned to the cell. However, in spite of the fact that ATM offered many sophisticated, mature and robust solutions for high speed networking, its popularity has been declining. According to experts, the main reasons for this trend are ATM's relatively high complexity, costly equipment and poor compatibility with legacy IP networks. For further information on ATM, see [1, 3, 4, 6]

WDM is an optical technology which enables the multiplexing of multiple optical signals on a single optical fiber by using different wavelengths (colors) of laser light to carry different signals. The concept of WDM was first published in the late 1970s. Since late 1990s, WDM has been the most popular technology implemented in backbone transport networks. A WDM network is formed by OXCs (Optical Cross Connects) interconnected by fibers. Note that WDM is a connection-oriented technique, since the entire signal is transmitted along a single connection (routing path) known as a *lightpath*. The frequency grid of WDM is divided with 50 GHz channel spacing. Optical devices generally cannot convert the wavelength, therefore the whole WDM connection must use the same color on every link along the path. This

additional requirement, known as the *wavelength continuity* constraint, complicates the demand provisioning in WDM networks compared to aforementioned technologies such as MPLS or ATM. More precisely, an optimization problem known as the Routing and Wavelength Assignment (RWA) appears in WDM networks, where both the routing path and wavelength must be selected for each demand. In contrast, optimization of flows in MPLS and ATM networks involves selecting the routing paths only, since there are no additional constraints on the link capacity resources. However, it is possible to use *opaque* OXC in WDM networks. In particular, the OXC is equipped with converters that transform the optical signal transmitted over a wavelength to another wavelength. This is achieved by first demultiplexing the optical signals and then converting them into electronic signals. Next, electronic signals are switched using electronic switching and then converted back into optical signals. Finally, these signals are multiplexed again into the output optical fiber. With this capability, optimization of connections in WDM networks is exactly the same as in MPLS and ATM networks. Note that the optimization problems presented in this chapter can be applied in the context of WDM networks with full wavelength conversion capabilities. For problems assuming the wavelength continuity constraint, when the wavelength conversion is not available, the reader is referred to Chap. 3. For a good survey on the topic of the WDM networks see [1, 5–9]

The supremacy of Ethernet in enterprise networking and in local area networks is a direct consequence of well understood operational practices, low-cost solutions and plug-and-play features. As a result, applications of Ethernet appear in larger environments, including backbone transport networks. However, the reduced use of Ethernet networks has brought the need to enhance Ethernet to improve network scalability and answer the challenge of using Ethernet in larger networks. The key additions have been PBB (Provider Backbone Bridging) proposed in 2008 [10] and PBB-TE (Provider Backbone Bridging—Traffic Engineering) defined in 2009 [11]. Both standards provide traffic-engineered, resource-managed transport using special tunnels for transmitting information (frames) belonging to the same VLAN (Virtual Local Area Network). Moreover, using the PBB-TE standard means that customer VLAN traffic is encapsulated in a PBB header, which includes Mac-in-Mac encapsulation. Next, the traffic is forwarded through the network using static database entries. As a result, the Ethernet transmission is connection-oriented, since all frames included in the same VLAN are transported by one Ethernet switched path that follows the same routing path [3, 12, 13].

The remainder of this chapter is devoted to various aspects of modeling and optimization of connection-oriented networks in the context of cloud computing and content-oriented services. The key assumption following from the connection-oriented nature of the considered networks is that non-bifurcated multicommodity flows are applied to model network traffic.

It should be stressed that all optimization problems presented in this section are $\mathcal{NP}$ -complete. This follows directly from the fact that the problems use non-bifurcated flows, and in the case of network design problems modular link capacity is applied. Therefore, since the authors of [4] show that the unicast flow allocation

problem with non-bifurcated flows and the network design problem with unicast flows and modular link capacity are $\mathcal{NP}$ -complete, the anycast and multicast versions of these problems are also $\mathcal{NP}$ -complete.

2.2 Allocation of Anycast Flow

This section starts with a relatively simple problem related to the allocation of anycast flows. Flow allocation problems assume that only network flows need be optimized, since an existing network is studied and the capacity of the network links is fixed. This assumption stems from by the fact that the considered network is in an operational phase and augmenting its physical resources, such as link capacity, is not achievable in a short time perspective. However, at the same time, there is a need to improve network performance, which can only be achieved by an improved optimization of network flows [4].

2.2.1 Formulation

The general notation is the same as in Sect. 1.2. To recall briefly, the network is modeled as a directed graph $G = (V, E)$, where V is a set of nodes (vertices) and E is a set of edges (directed links) with limited capacity c_e . A number of data centers (DCs) are located in the network. For simplicity, the local connection between the data center and the backbone network node is not included in the model, i.e., it is assumed that the network node is equivalent with the data center connected to that node. This is because local access links used to connect data centers are usually of a high bandwidth and thus capacity of this link is not incorporated in the model.

A set of anycast demands denoted by D is given. A standard model of anycasting is employed [14]; a single anycast request is realized by two associated demands: upstream (from the client to the data center) and downstream (in the opposite direction). Let $\tau(d)$ be the index of a demand associated with demand d . The demand volume is described by the constant h_d . Link-path modeling is assumed, and therefore for each demand $d \in D$, a set of candidate paths $P(d)$ is given. If d is an upstream demand, set $P(d)$ contains candidate paths that originate at one of the DC nodes and terminate at the client node. In turn, if d is a downstream demand, candidate paths in set $P(d)$ connect the client node and one of the DC nodes. Paths are defined using constant δ_{edp} , i.e., δ_{edp} equals 1 if path p of demand d includes link e and 0 otherwise. There is one decision variable x_{dp} that denotes, which path $p \in P(d)$ is selected to realized demand d . The objective of the flow allocation problem is to find a routing path for each demand that minimizes a selected performance metric and at the same time provides a feasible solution in terms of the link capacity constraint. Moreover, the anycast constraint must be satisfied to ensure that both associated anycast demands are assigned to the same DC node.

CON/A/FA/Cost**sets**

- E links
 D anycast demands
 D^{DS} anycast downstream demands
 $P(d)$ candidate paths for flows realizing demand d . If d is an anycast upstream demand, candidate path connects client node and DC node. If d is a downstream demand, candidate path connects the DC node and the client node

constants

- δ_{edp} =1, if link e belongs to path p realizing demand d ; 0, otherwise
 h_d volume (bit-rate) of demand d
 c_e capacity of link e
 ζ_e unit routing cost on link e
 $\tau(d)$ index of a demand associated with demand d . If d is a downstream demand, then $\tau(d)$ must be an upstream demand and vice versa
 $s(p)$ origin node of path p
 $t(p)$ destination node of path p

variables

- x_{dp} =1, if path p is used to realize demand d ; 0, otherwise (binary)
 f_e flow on link e (continuous non-negative)

objective

$$\text{minimize } F = \sum_{e \in E} \zeta_e f_e \quad (2.2.1a)$$

constraints

$$f_e = \sum_{d \in D} \sum_{p \in P(d)} \delta_{edp} x_{dp} h_d, \quad e \in E \quad (2.2.1b)$$

$$\sum_{p \in P(d)} x_{dp} = 1, \quad d \in D \quad (2.2.1c)$$

$$f_e \leq c_e, \quad e \in E \quad (2.2.1d)$$

$$\sum_{p \in P(d)} x_{dp} s(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} t(p), \quad d \in D^{DS}. \quad (2.2.1e)$$

Objective (2.2.1a) is to minimize the overall routing cost. However, other performance metrics can be considered as the optimization goal, e.g., delay or network congestion. Moreover, the allocation problem can be formulated without the objective function, i.e., the challenge is to find a feasible routing configuration that satisfies

the link capacity constraint (see [4] for more details). Equality (2.2.1b) defines the value of variable f_e . Constraint (2.2.1c) ensures that exactly one routing path is selected to realize demand d . Condition (2.2.1d) is a link capacity constraint to meet the requirement that the flow of each link given as a sum of all demands that use this link cannot exceed the link capacity. Finally, equality (2.2.1e) defines the anycast constraint guaranteeing that two associated demands are assigned to the same DC node.

2.2.2 Algorithms

This section is devoted to heuristic and metaheuristic algorithms applicable to the optimization problem of anycast flow allocation formulated as (2.2.1a)–(2.2.1e). Note that it can be modified easily to enable joint optimization of anycast flows and other types of flows such as unicast flows using the link-path modeling and multicast flows applying the candidate tree modeling.

Greedy Algorithm

The first proposed algorithm named GR/A/FA (Greedy Algorithm for Anycast Flows), is based on a very simple greedy approach. More precisely, the general idea is to allocate anycast demands one by one according to a selected ordering of demands. However, due to the anycast constraint (2.2.1e), both associated anycast demands must be processed jointly. Algorithm 2.1 shows the scheme of the GR/A/FA algorithm.

Algorithm 2.1 GR/A/FA (Greedy Algorithm for Anycast Flow Allocation)

Require: set of edges E , set of anycast demands D , sets $P(d)$ including candidate paths for each demand $d \in D$

Ensure: path selection (routing) for each demand $d \in D$ included in set X , value of objective function

```

1: procedure GR/A/FA( $D, P(d)$ )
2: for  $d \in D$  do
3:   if  $Is\_Not\_Allocated(\tau(d))$  then
4:      $p := Find\_Best\_Path(d)$ 
5:   else
6:      $p := Find\_Best\_Path\_DC(d, DC\_Node(\tau(d)))$ 
7:   end if
8:    $X := X \cup \{x_{dp}\}$ 
9:    $D := D \setminus \{d\}$ 
10: end for
11: end procedure

```

Let set X (known as the solution) include all variables x_{dp} that are equal to 1. Solution X determines the unique set of selected routing paths for each demand $d \in D$. The algorithm performs a single loop that analyzes all demands (lines 2–10). Demands are processed according to a selected ordering. For instance, the demands can be sorted in decreasing order of the bit-rate denoted as h_d . The rationale of this approach is that larger demands should be processed first, since they need more capacity resources. Next, when the network becomes more congested, it is easier to allocate smaller demands than larger ones. Another possible metric for ordering is a multiplication of the bit-rate and the hop count of the shortest path available for a particular demand. Note that the construct of the greedy method makes it possible to use a wide range of ordering strategies.

As mentioned above, due to the anycast constraint, both associated demands d and $\tau(d)$ must be connected to the same DC node, therefore there are two possible ways to process a demand. In particular, if the associated demand $\tau(d)$ is yet to be allocated, the current demand d can be assigned to any DC node and function *Find_Best_Path*(d) can use all candidate paths available for demand d (line 4). On the other hand, if the associated demand $\tau(d)$ is already processed by the algorithm, demand d can only be assigned to the path connected to the same DC node as demand $\tau(d)$, i.e., function *DC_Node*($\tau(d)$) returns the DC node selected for demand $\tau(d)$ (line 6). Note that the functions *Find_Best_Path* and *Find_Best_Path_DC* are generic and can implement various strategies to find a routing path. For instance, the shortest path in terms of kilometer distance can be selected in a residual graph that only includes links with a residual capacity greater than the requested bit-rate of the considered demand. The maximum complexity of algorithm GR/A/FA is $O(|D| |P|)$, where $|D|$ denotes the number of demands and $|P|$ is the number of candidate paths for each demand.

Note that the main advantages of the greedy algorithm are its adaptability to the means of ordering and routing strategies and its relatively low complexity, which should result in a short execution time. On the other hand, the algorithm's main drawback is that in a congested network, it may have trouble finding a feasible solution due to a shortage of capacity resources. Therefore, below we introduce more advanced methods of solving the problem of anycast flow allocation.

Flow Deviation

The Flow Deviation (FD) algorithm was proposed for optimization of bifurcated and non-bifurcated flows in [15], and has been widely applied to various optimization problems [4, 14, 16–27].

The FD method was developed following the expansion of store-and-forward communications networks together with the ARPANET [19]. The FD algorithm is based on the concept of a *shortest route* flow applied to successive flow deviations that lead to local minima. More precisely, for each demand in the network, the shortest route is determined according to a selected objective function and based on the current routing (flow allocation) in the network. Next, we attempt to deviate the flow of the demand to the shortest route. In the context of bifurcated flows, the flow deviation can

be applied to a part of the demand flow, since multipath routing is allowed. However, for non-bifurcate flows with single path routing, the flow deviation must affect the whole demand [15].

Since this chapter focuses on connection-oriented networks, the non-bifurcated version of the FD method is applied. An FD algorithm for optimization of anycast flows known as FD/A/FA (Flow Deviation for Anycast Flow Allocation) is formulated and discussed. Note that the first time the FD method was applied in the context of anycast flows was in [26], while in [14, 27] the same author proposed an FD method for joint optimization of anycast and unicast flows. The key innovation of the FD/A/FA algorithm, in comparison to the unicast version proposed in [15], is that the new method processes associated anycast demands together, which satisfies the anycast constraint (2.2.1e). Algorithm 2.2 reports the pseudocode of the FD/A/FA heuristic.

Algorithm 2.2 FD/A/FA (Flow Deviation for Allocation of Anycast Flows)

Require: set of edges E , set of anycast demands D , sets $P(d)$ including candidate paths for each demand $d \in D$

Ensure: path selection (routing) for each demand $d \in D$ included in set X_i , value of objective function

```

1: procedure FD/A/FA( $D, P(d)$ )
2:  $X_1 := \text{Find\_Initial\_Solution}(D, P(d))$ ,  $i := 1$ 
3:  $test := 0$ 
4: repeat
5:    $SR(X_i) := \text{Find\_Shortest\_Paths}(D, P(d), X_i)$ 
6:    $H := X_i$ 
7:   for  $d \in D$  do
8:      $K := (H - \{x_{dp}\}) \cup \{x_{dq}\}$ , where  $x_{dp} \in H$  and  $x_{dq} \in SR(X_i)$ 
9:      $K := (K - \{x_{\tau(d)p'}\}) \cup \{x_{\tau(d)q'}\}$ , where  $x_{\tau(d)p'} \in H$  and  $x_{\tau(d)q'} \in SR(X_i)$ 
10:     $F(H) = \text{Find\_Objective}(H)$ 
11:     $F(K) = \text{Find\_Objective}(K)$ 
12:    if  $F(K) < F(H)$  then  $H := K$ 
13:     $D := D \setminus \{d, \tau(d)\}$ 
14:  end for
15:  if  $X_i \neq H$  then
16:     $i := i + 1$ 
17:     $X_i := H$ 
18:  else
19:     $test := 1$ 
20:  end if
21: until  $test < 1$ 
22: end procedure

```

Let X_i denote a solution of the problem obtained in iteration i . In turn, sets H and K are temporary solutions used in the algorithm. A special flag $test$ denotes the stopping condition of the algorithm. The algorithm starts with a feasible initial solution X_1 (line 2). To find a such a solution, an algorithm based on Phase 1 of the original FD method [15] can be applied with the modifications required to process

anycast flows; for more details, see [14]. Next, the flag *test* is initialized. The main loop of the algorithm (lines 4–21) is repeated until a new solution H generated in the current iteration provides a modification of the solution X_i created in the previous iteration. Function *Find_Shortest_Paths*($D, P(d), X_i$) calculates set $SR(X_i)$ including the shortest routes for all demands (line 5). This function ensures that the anycast constraint (2.2.1e) is always satisfied. More specifically, for each pair of associated demands d and $\tau(d)$ a demand with a large value of volume h_d is selected first. Without losing generality, let us assume that $h_d \geq h_{\tau(d)}$. In the case of demand d , the *Find_Shortest_Paths* function finds the shortest route under a selected metric taking into account all available routing paths included in $P(d)$. To ensure that both associated demands use the same DC node, in the case of demand $\tau(d)$ the function *Find_Shortest_Paths* considers only the paths from $P(\tau(d))$ that are connected to the DC node selected for demand d . Note that usually a partial derivative of the objective function is used as a link metric applied in the selection of the shortest path in function *Find_Shortest_Paths*.

Next, the current selection X_i is saved as H (line 6). The loop defined in lines 7–14 processes all demands included in set D according to a selected ordering as in the case of Algorithm 2.1. However, to ensure the anycast constraint, associated demands d and $\tau(d)$ are processed together. A new selection K is obtained by using the flow deviation operation, i.e., demands d and $\tau(d)$ are switched to shortest paths included in set $SR(X_i)$ (lines 8 and 9). Solutions H and K are compared in terms of the objective function value; the new solution K is saved as the current solution (lines 10–12) only if it decreases the objective function value. When all demands are processed, we check whether solution H obtained in the current iteration brings about a change relative to the previous solution X_i (lines 15–20). If the solutions differ, the algorithm is continued, otherwise the algorithm stops.

Note, that the algorithm converges in a finite number of steps, since there is a finite number of non-bifurcated flows. Repetitions of the same solution are impossible due to the stopping condition. The maximum number of the FD/A/FA algorithm iterations can be estimated as the number of all possible routing path combinations. In particular, let k denote the number of candidate paths defined between a pair of nodes and let r denote the number of DC nodes in the network. In consequence, kr defines the number of candidate paths for each anycast demand. However, since associated demands d and $\tau(d)$ must be processed jointly due to the anycast constraint (2.2.1e), for each kr routing paths available for d , only k routing paths are available for $\tau(d)$, since both d and $\tau(d)$ must use the same DC node. Thus, the possible number of path combinations for a pair of associated anycast demands is defined as rk^2 . In consequence, the number of all possible path combinations for a pair of associated anycast demands is $(rk^2)^{\frac{|D|}{2}}$.

Lagrangian Relaxation

The next algorithm proposed for optimization of anycast flows in connection-oriented networks is the Lagrangian relaxation (LR) method combined with a subgradient optimization approach. The main aim of this approach is to iteratively solve the

optimization problem using a heuristic algorithm FD/A/FA that uses results given by solving dual problems as the initial solutions. The LR technique with the subgradient optimization has been successfully used for solving various network optimization problems, e.g., [4, 6, 14, 28–36].

To the best of our knowledge, the Lagrangian relaxation technique for anycast flows in connection-oriented flows was introduced for the first time in [35]. In turn, the authors of [14] report how to apply Lagrangian relaxation for joint optimization of anycast and unicast flows.

The key aim of the LR decomposition algorithm is to consider the dual problem of the original optimization problem by relaxing constraints in order to obtain a simpler subproblem. This procedure makes it possible to move iteratively towards the optimal solution of the original problem. Consequently, selecting a suitable constraint to be relaxed is key.

In order to formulate a dual problem to model (2.2.1), the same approach as in [14, 29, 35] is applicable. More specifically, the problem (2.2.1) is first transformed into an equivalent formulation, which is better suited for the LR procedure. The key observation is that the objective function (2.2.1a) does not decrease with f_e , therefore the equality (2.2.1b) one be replaced with the following inequality:

$$f_e \leq \sum_{d \in D} \sum_{p \in P(d)} \delta_{edp} x_{dp} h_d, \quad e \in E. \quad (2.2.1f)$$

For ease of reference, the modified optimization problem defined as (2.2.1a), (2.2.1c)–(2.2.1f) will be referred to as CON/A/FA/Cost/2. It should be noted that problems CON/A/FA/Cost (2.2.1a)–(2.2.1e) and CON/A/FA/Cost/LR (2.2.1a), (2.2.1c)–(2.2.1f) are equivalent, i.e., the optimal solution obtained to one of these problems guarantees the optimal solution to the other problem.

A popular approach of Lagrangian relaxation in the context of network optimization is to relax the capacity constraint [4, 6, 32, 33]. The method shown below is based on the concept proposed in [29] and constraint (2.2.1f) is relaxed using vector $\lambda = (\lambda_1, \lambda_2, \lambda_{|E|})$ of positive Lagrangian multipliers λ_e for each link $e \in E$. Accordingly, the following Lagrangian relaxation of problem CON/A/FA/Cost/2 is formulated as follows.

CON/A/FA/Cost/LR

objective

$$\text{minimize } \varphi(\lambda) = \sum_{e \in E} \zeta_e f_e + \sum_{e \in E} \lambda_e \left(\sum_{d \in D} \sum_{p \in P(d)} \delta_{edp} x_{dp} h_d - f_e \right) \quad (2.2.2a)$$

constraints

$$\sum_{p \in P(d)} x_{dp} = 1, \quad d \in D \quad (2.2.2b)$$

$$f_e \leq c_e, \quad e \in E \quad (2.2.2c)$$

$$\sum_{p \in P(d)} x_{dp} s(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} t(p), \quad d \in D^{DS} \quad (2.2.2d)$$

$$f_e \geq 0, \quad e \in E. \quad (2.2.2e)$$

The objective function (2.2.2a) follows directly from the relaxation of condition (2.2.1f). Constraints (2.2.2b)–(2.2.2d) are copied from the original problem. A new constraint (2.2.2e) is added to the problem. Note that condition (2.2.2e) in model CON/A/FA/Cost/2 is guaranteed by constraint (2.2.1f). However, since (2.2.1f) is relaxed and incorporated to the dual function (2.2.2a), this new constraint (2.2.2e) is required to guarantee variables f_e to be positive.

The objective function (2.2.2a) of the Lagrangian problem can be rewritten as

$$\text{minimize } \varphi(\lambda) = \left(\sum_{e \in E} (\zeta_e f_e - \lambda_e f_e) \right) + \left(\sum_{e \in E} \sum_{d \in D} \sum_{p \in P(d)} \lambda_e \delta_{edp} x_{dp} h_d \right). \quad (2.2.3)$$

Since there are no coupling constraints between variables f_e and d_{ap} , the problem (2.2.2) can be divided into two independent subproblems CON/A/FA/Cost/LR/1 and CON/A/FA/Cost/LR/2, defined below.

CON/A/FA/Cost/LR/1**objective**

$$\text{minimize } \varphi_1(\lambda) = \sum_{e \in E} (\zeta_e f_e - \lambda_e f_e) \quad (2.2.4a)$$

constraints

$$f_e \leq c_e, \quad e \in E \quad (2.2.4b)$$

$$f_e \geq 0, \quad e \in E. \quad (2.2.4c)$$

It should be noted that problem CON/A/FA/Cost/LR/1 (2.2.4) can be separated into $|E|$ subproblems, each of which is solved by the following formula:

$$f_e = \begin{cases} c_e & \text{if } \zeta_e - \lambda_e < 0 \\ 0 & \text{if } \zeta_e - \lambda_e \geq 0 \end{cases} \quad e \in E. \quad (2.2.5)$$

CON/A/FA/Cost/LR/2
objective

$$\text{minimize } \varphi_2(\lambda) = \sum_{e \in E} \sum_{d \in D} \sum_{p \in P(d)} \lambda_e \delta_{edp} x_{dp} h_d \quad (2.2.6a)$$

constraints

$$\sum_{p \in P(d)} x_{dp} = 1, \quad d \in D \quad (2.2.6b)$$

$$\sum_{p \in P(d)} x_{dp} s(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} t(p), \quad d \in D^{DS}. \quad (2.2.6c)$$

In turn, problem CON/A/FA/Cost/LR/2 defined in (2.2.6) can be separated into $|D^{DS}|$ subproblems, i.e., one problem for a pair of associated demands d and $\tau(d)$. To solve such a problem, we need to find the shortest pair of paths under metric $\lambda_e h_d$ for demands d and $\tau(d)$ considering each DC node individually. Next, the pair of paths with the lowest value of path lengths is selected as the final solution. This ensures the optimal solution of problem CON/A/FA/Cost/LR/2.

The above method solves the dual problem; the next step is the subgradient search [4, 37]. Let $x_{dp}(\lambda)$ be the optimal solution of the Lagrangian relaxation for a fixed vector of multipliers λ . Let X denote the set of all variables x_{dp} equal to one. The corresponding subgradient of the dual function (2.2.6) at λ is defined as:

$$\gamma_e(\lambda) = \left(\sum_{d \in D} \sum_{p \in P(d)} \delta_{edp} x_{dp}(\lambda) h_d \right) - f_e(\lambda) \quad e \in E. \quad (2.2.7)$$

The Lagrangian multipliers in subsequent iterations of the subgradient procedure (denoted as i) are updated as follows:

$$\lambda_e^{i+1} = \max(0, \lambda_e^i + t_i \gamma_e^i) \quad e \in E. \quad (2.2.8)$$

The step-size t_i can be defined as proposed in [4, 29]:

$$t_i = \frac{\rho(\bar{\varphi} - \varphi(\lambda^i))}{\|\gamma^i\|^2}. \quad (2.2.9)$$

Note that $\bar{\varphi}$ denotes the upper bound of the dual function (2.2.2a), which can be calculated using a heuristic algorithm that yields a feasible solution of the primal problem. Moreover, ρ is commonly used in the range $0 \leq \rho \leq 2$ [4]. Algorithm 2.3 reports the pseudocode of the subgradient optimization procedure

applied to Lagrangian relaxation of the anycast flow allocation problem defined in (2.2.1a)–(2.2.1e).

First, tuning parameters used in algorithm LR/A/FA are initialized (line 2). The values selected for the initialization are as proposed in [14, 35]. The maximum number of iterations is set to $i_{\max} := 100$ as the stopping condition for dual iterations. Additionally, it is possible to select another value depending on specific problem characteristics. Parameter ρ_{iter} counts the number of subsequent iterations that do not improve the dual function value, while parameter ρ_{maxiter} defines the limit of such iterations. If ρ_{iter} reaches ρ_{maxiter} , then parameter ρ used to calculate the step size according to (2.2.9) decreases. The vector of Lagrangian multipliers λ^1 is initialized with all values equal to 1 (line 3), but again—depending on the specific problem—different assignment can be made. The upper bound of the dual function $\bar{\varphi}$ is obtained as the solution of the FD/A/FA method shown in Algorithm 2.2 (line 4). The main loop of the algorithm including subsequent iterations of the subgradient search is defined in lines 5–26. In summary, first the dual problem is solved according to the

Algorithm 2.3 LR/A/FA (Lagrangian Relaxation for Anycast Flows)

Require: set of edges E , set of anycast demands D , sets $P(d)$ including candidate paths for each demand $d \in D$

Ensure: path selection (routing) for each demand $d \in D$ included in set X^{best} , value of objective function

```

1: procedure LR/A/FA( $D, P(d)$ )
2:  $\rho := 2, \rho_{\min} := 0.005, \rho_{\text{iter}} := 0, \rho_{\text{maxiter}} := 3, i_{\max} := 100, F^{\text{best}} = \infty, \varphi^{\text{best}} = -\infty$ 
3:  $\lambda^1 = \mathbf{1}$ 
4:  $\bar{\varphi} := \text{FD/A/FA}(D, P(d))$ 
5: for  $i := 1$  to  $i_{\max}$  do
6:    $\rho_{\text{iter}} := \rho_{\text{iter}} + 1$ 
7:    $\varphi(\lambda^i) = \text{Solve\_Dual\_Problems}(\lambda^i)$ 
8:   if  $\varphi(\lambda^i) > \varphi^{\text{best}}$  then
9:      $\varphi^{\text{best}} := \varphi(\lambda^i)$ 
10:     $\rho_{\text{iter}} := 0$ 
11:   end if
12:    $X := \text{FD/A/Init\_Sol}(D, P(d), X(\lambda^i))$ 
13:    $F = \text{Find\_Objective}(X)$ 
14:   if  $F < F^{\text{best}}$  then
15:      $F^{\text{best}} := F$ 
16:      $X := X^{\text{best}}$ 
17:      $\bar{\varphi} := F$ 
18:   end if
19:   if  $\rho_{\text{iter}} > \rho_{\text{maxiter}}$  then
20:      $\rho := \max(\rho/2, \rho_{\min})$ 
21:      $\rho_{\text{iter}} := 0$ 
22:   end if
23:    $\gamma^i := \text{Subgradient}(\lambda^i)$   $\triangleright$  refer to (2.2.7)
24:    $t_i := \text{Step\_Size}(\lambda^i, \gamma^i)$   $\triangleright$  refer to (2.2.9)
25:    $\lambda^{i+1} := \text{Update\_Lambda}(\lambda^i, \gamma^i)$   $\triangleright$  refer to (2.2.8)
26: end for
27: end procedure

```

methodology described above (line 7). If the value of the dual function $\varphi(\lambda^i)$ is greater than the previous best result, the algorithm saves this value and resets counter ρ_{iter} (lines 8–11). Solution $X(\lambda^i)$ obtained by solving the dual problem is used to initialize the FD/A/FA algorithm, which then finds a primal feasible solution of the problem denoted as X with the value of the objective function denoted as F (lines 12–13). If the new solution outperforms the previous best one, it is saved (lines 14–18). Next, it is checked whether ρ_{iter} reaches the limit defined by $\rho_{maxiter}$ and parameter ρ is updated accordingly (lines 19–22). Finally, the vector of Lagrangian multipliers λ^i is updated using the subgradient optimization approach and formulas defined in (2.2.7)–(2.2.9). Note that the complexity and the execution time of the LR/A/FA method mainly depends on the number of iterations given by parameter i_{max} , since the most time complex part of each iteration is the execution of the FD/A/FA algorithm.

The optimization problem (2.2.1a)–(2.2.1e) uses the link-path notation to model multicommodity flows. Some papers that apply Lagrangian relaxation to routing problems use the node-link formulation [4, 28, 32–34]. However, in both cases the dual is constructed analogously, i.e., we obtain an optimization problem that can be decoupled into two independent subproblems, where one is the minimum cost routing problem with link metrics given by Lagrangian multipliers.

Evolutionary Algorithm

The evolutionary algorithm (EA) is a stochastic heuristic method widely applied in the context of various optimization problems related to computer and communication networks. For an extensive survey of EAs and their application to network problems refer to [4, 38–42].

The key issue in designing the EA is to develop a method of encoding the optimization problem into chromosomes. To recall, in the anycast flow allocation problem (2.2.1a)–(2.2.1e), the decision is to select routing paths for each demand. A solution of the problem is denoted as a set that includes all variables x_{dp} equal to 1, which directly indicates the selected routing paths. Regarding connection-oriented networks, a common approach of EA encoding is to assume that each allele in the chromosome represents one demand. The value of each allele is an index of a path selected for a particular demand (denoted as p_d) [4, 43–49]. Therefore, the chromosome is defined as follows:

$$X = [p_1, p_2, \dots, p_{|D|}]. \quad (2.2.10)$$

For example, chromosome $X = [3, 1, 2]$ means that demand $d = 1$ uses path $p = 3$ ($x_{13} = 1$), demand $d = 2$ is allocated to path $p = 1$ ($x_{21} = 1$) and demand $d = 3$ is realized on path $p = 2$ ($x_{32} = 1$). Using encoding (2.2.10), it is straightforward to calculate flow on each link (2.2.1b) and next to calculate the value of the objective function (2.2.1a). It should be noted that other ways of encoding the multicommodity flow optimization problems in EAs have been proposed in literature, e.g., see [50–54].

Originally, EA was designed to solve optimization problems without constraints and with the aim of maximizing the objective (fitness) function. Therefore, to address

the fact that the problem includes constraints and the objective function is to minimize the routing cost, the following modifications are required. There are three ways of incorporating processing of constraints in EA. Firstly, the encoding scheme can ensure that particular constraints are directly satisfied. Secondly, a *penalty function* can be employed, i.e., the fitness function contains not only the problem objective function, but also a special term including a measure of violation of the constraints scaled by a penalty parameter. It is assumed that the measure of violation is nonzero if the constraint is violated, and zero in the region where the constraint is not broken. Thirdly, a *repair function* can be used when the current solution (chromosome) is not feasible.

Note that in this problem (2.2.1), condition (2.2.1c) is included directly in the encoding scheme (2.2.10). More specifically, since a single allele is assigned to exactly one demand in the chromosome, the single path routing is imposed.

To address the link capacity constraint (2.2.1d), the penalty function approach is applied. More specifically, let $F(X)$ return the value the objective function (2.2.1c) of the solution encoded in chromosome X . Next, let $F^{PEN}(X)$ denote the value of the objective function with an additional penalty function for chromosome X defined as follows:

$$F^{PEN}(X) = F(X) + Pn \sum_{e \in E} CapCon(X, e). \quad (2.2.11)$$

Function $CapCon(X, e)$ represents the violation of capacity constraint (2.2.1c) according to network flows defined in X . If the capacity constraint for link e is not violated (i.e., $f_e \leq c_e$), then $CapCon(X, e) = 0$, otherwise, $CapCon(X, e) = f_e - c_e$. In turn, Pn denotes the penalty scaling parameter that tunes the penalty function.

Finally, to tackle the anycast constraint (2.2.1e), a simple repair function is developed (Algorithm 2.4). For an input demand d it is checked whether the associated demand $\tau(d)$ uses the same DC node as the DC node of demand d in the current solution X (line 2). If this condition is not fulfilled, a new path for demand $\tau(d)$ needs to be selected, but using only candidate paths from set $P(\tau(d))$ connected to the DC node of demand d (lines 3–5). To repair the whole solution, a function described as Algorithm 2.4 must be executed for all associated demand pairs. In order to improve the performance of the repair procedure for the whole solution including all demands, the demands can be analyzed in a particular order, e.g., by decreasing value of the requested volume (bitrate).

To address the fact that EA requires the objective function to be maximized, the fitness function is defined as follows

$$Fitness(X) = M(F^{MAX} - F^{PEN}(X)) \quad (2.2.12)$$

where F^{MAX} denotes the maximum possible value function $F^{PEN}(X)$ taking into account all chromosomes X in a given population, while M is a scaling parameter that conducts additional tuning of the algorithm during the optimization process.

Algorithm 2.4 RFAD (Repair Function for Anycast Demand)

Require: set of edges E , associated anycast demands d and $\tau(d)$, set $P(\tau(d))$ including candidate paths for demands d and $\tau(d)$, current solution X

Ensure: feasible path selection (routing) for demand $\tau(d)$

```

1: procedure RFAD( $X, d, P(d), P(\tau(d))$ )
2: if  $DC\_Node(d) \neq DC\_Node(\tau(d))$  then
3:    $X := X \setminus \{x_{\tau(d)p}\}$ 
4:    $q := Find\_Best\_Path\_DC(\tau(d), DC\_Node(d))$ 
5:    $X := X \cup \{x_{\tau(d)q}\}$ 
6: end if
7: end procedure

```

It is worth noting that the proposed encoding scheme and fitness function formulation apply the classical crossover and mutation operators. For instance, to make the one-point crossover of two parent solutions $X^1 = [p_1^1, p_2^1, \dots, p_{|D|}^1]$ and $X^2 = [p_1^2, p_2^2, \dots, p_{|D|}^2]$, it is necessary to select at random an integer d between 1 and $|D|$. In consequence, the following two children are obtained $X^3 = [p_1^1, p_2^1, \dots, p_d^1, p_{d+1}^2, \dots, p_{|D|}^2]$ and $X^4 = [p_1^2, p_2^2, \dots, p_d^2, p_{d+1}^1, \dots, p_{|D|}^1]$. Next, the repair function defined in Algorithm 2.4 must be applied to both new chromosomes to ensure the anycast constraint. The mutation operation assumes that for a randomly selected demand d , the routing path is changed by a random selection of an integer between 1 and $|P(d)|$. Again, the repair function is required to fix the potential problem with the anycast constraint. Additionally, for the described encoding scheme and fitness function formulation, more complex approaches of crossover and mutation operators can be used.

Finally, we discuss the issue of the initial solution of EA. More specifically, in many cases the size of a feasible solution space in the flow allocation problem is very small compared to the total solution space. Consequently, it is likely that EA can encounter significant difficulties in finding a feasible solution, even using mechanisms such as the penalty function and repair function. In such a case, the performance of EA can be improved by using a feasible solution provided by another method, e.g., the FD algorithm. Thereafter, using the *elite* member concept (i.e., the best feasible solution(s) are copied directly to the next generation) [38, 40] ensures that EA method yields a feasible solution.

Neighborhood Search Methods

There is a wide family of stochastic heuristics based on the concept of *neighborhood search*, including Local Search (LS), Tabu Search (TS), Simulated Annealing (SA), and Greedy Randomized Adaptive Search Procedure (GRASP) [4, 39, 40, 42, 55–58]. The neighborhood $N(X)$ of a particular solution X is a subset of the total solution space including solutions $Y \in N(X)$ that can be generated from X by a simple modification of the solution. The most common approach to generating the neighborhood solution for unicast flow allocation problems in connection-oriented networks is to simply change a routing path for one demand. The number of neighborhood

solutions can then be estimated as $\sum_{d \in D} (|P(d)| - 1)$, since for each demand $d \in D$, the currently selected path can be changed to any of the remaining paths included in set $P(d)$ [4].

However, to address the additional anycast constraint (2.2.1e) that arises in anycast flow allocation problems, the default procedure must be modified. More precisely, if the created neighborhood solution $Y \in N(X)$ does not satisfy the anycast constraint, i.e., a new path selected for demand d uses a different DC node than the DC node of demand $\tau(d)$, the repair function shown in Algorithm 2.4 must be executed for demand d .

In addition, the neighborhood solution $Y \in N(X)$ may be not feasible in terms of the link capacity constraint (2.2.1d). In such a case, there are two options. First, the penalty function approach can be used similarly to (2.2.11). As a result, the algorithm based on the neighborhood search is able to examine unfeasible solutions as well. Secondly, all solutions that are not feasible cannot be analyzed, i.e., they are excluded from set $N(X)$.

Using this definition of the neighborhood solution, it is easy to develop range of neighborhood search methods such as LS, TS, SA and GRASP for the anycast flow allocation problem. As in the context of EA, the algorithm performance can be improved by using an initial solution yielded by another algorithm.

2.3 Network Design Problems for Anycast, Multicast and Unicast Flows

This section focuses on a network design problem with joint optimization of link capacity assignment and anycast, multicast and unicast flow allocation. Note that the network design problem is also referred to as the capacity and flow allocation (CFA) problem. Network design problems are among the most common optimization problems in networks. They need to be resolved when a new network is being designed from scratch or an existing network needs to be updated. The main goal of the optimization process is to select the capacity of network links and the routing configuration in order to realize all demands. The most common objective function that occurs in network design problems is the CAPEX/OPEX cost, although other network performance metrics (e.g., delay, survivability, throughput) can be applied. The link capacity constraint which ensures that the total flow on each link cannot exceed the assigned link capacity appears to be the key element of all network design problems [4, 49].

The majority of earlier network design problems have focused on the optimization of unicast flows only. Relatively few papers address joint optimization of two types of flows, i.e., anycast and unicast or multicast and unicast. The only paper that focuses on joint optimization of anycast, multicast and unicast flows in connection-oriented networks is [59], where a static RWA problem with unicast, anycast and multicast connections is examined and some heuristics are proposed and evaluated. In the case

of multicast flows, the routing is fixed since only one tree is created using a minimal spanning tree algorithm, which is then iteratively pruned to remove all unnecessary leaves.

2.3.1 Formulation

The routing is modeled in a similar way to that presented in Sect. 2.2. However, since anycast, multicast and unicast network flows are to be provisioned in the network, some modifications are required. In particular, for each demand $d \in D$, the set $P(d)$ includes candidate structures. When d is a unicast demand, the candidate structure is simply a path connecting the source and destination nodes of the demand. When d is an anycast upstream demand, the candidate structure is a path connecting the client node and one of the DC nodes. In turn, when d is a downstream demand, the candidate structure is a path connecting one of the DC nodes and the client node. Finally, when d is a multicast demand, the candidate structure is a tree that includes a root node and all receivers of the demand.

The link capacity is given in modules, and the capacity assigned to each link is expressed as a multiple of one of the modules. This assumption follows from the fact that in most network technologies such as Ethernet, SDH/SONET and WDM the network uses some predefined values of the potential capacity available on each link. Constant M denotes the size of the link capacity module given in the same unit as the demand volume h_d , e.g., in bits per seconds. Moreover, constant ξ_e denotes the cost of using one capacity module on link e . Note that constant ξ_e can represent various types of costs, such as CAPEX cost, OPEX cost, power consumption, etc. Integer variable y_e denotes the number of capacity modules allocated to link e [4, 49].

CON/AMU/ND/Cost/Link-path

sets

E	links
D	demands (anycast, multicast, unicast)
D^{DS}	anycast downstream demands
$P(d)$	candidate structures for flows realizing demand d . If d is a unicast demand, candidate structure is a path connecting end nodes of the demand. If d is an anycast upstream demand, candidate structure is a path connecting the client node and the DC node. If d is a downstream demand, candidate structure is a path connecting the DC node and the client node. If d is a multicast demand, candidate structure is a tree that includes the root node and all receivers of the demand

constants

δ_{edp}	=1, if link e belongs to structure p realizing demand d ; 0, otherwise
h_d	volume of demand d
ξ_e	unit cost of link e
M	size of the link capacity module
$\tau(d)$	index of a demand associated with demand d . If d is a downstream demand, then $\tau(d)$ must be an upstream demand and vice versa
$s(p)$	origin node of path p
$t(p)$	destination node of path p

variables

x_{dp}	=1, if structure p is used to realize demand d ; 0, otherwise (binary)
y_e	capacity of link e as the number of capacity modules (integer)

objective

$$\text{minimize } F = \sum_{e \in E} \xi_e y_e \quad (2.3.1a)$$

constraints

$$\sum_{p \in P(d)} x_{dp} = 1, \quad d \in D \quad (2.3.1b)$$

$$\sum_{d \in D} \sum_{p \in P(d)} \delta_{edp} x_{dp} h_d \leq M y_e, \quad e \in E \quad (2.3.1c)$$

$$\sum_{p \in P(d)} x_{dp} s(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} t(p), \quad d \in D^{DS}. \quad (2.3.1d)$$

The objective function (2.3.1a) is to minimize the total network cost defined as the overall cost of all capacity allocated to network links. Equality (2.3.1b) ensures that for each demand $d \in D$ exactly one routing structure is selected. Condition (2.3.1c) is the link capacity constraint, but since a network design problem is considered, the right-hand side of (2.3.1c) is not fixed and follows from the selection of link variable y_e . Finally, constraint (2.3.1d) imposes the anycast constraint that holds for anycast demands only.

Model (2.3.1) is a generic formulation which can be modified to address various additional constraints that can occur in different types of network design problems. For instance, the network cost given by formula $\sum_{e \in E} \xi_e y_e$ can be incorporated as a budget constraint that limits the maximum cost to be spent on network deployment, and another function can be used as the objective, e.g., network delay, proportion of the realized demand volumes. For more details see [4].

2.3.2 Algorithms

The network design problem formulated as (2.3.1) is generally comparable to the flow allocation problem addressed in Sect. 2.2 with a key additional element of link capacity optimization. Therefore, the general concepts of heuristic algorithms outlined in Sect. 2.2.2 can be adapted to the new constraints.

Firstly, we show how to use the FD method proposed in [15] for the network design problem with anycast, multicast and unicast flows. Algorithm 2.5 presents the FD/AMU/ND method. The key assumption behind this heuristic is that the algorithm directly decides on the routing variables x_{dp} , while the values of link capacity variables y_e are calculated indirectly according to a particular flow allocation. More specifically, having selected a routing structure for each demand $d \in D$ (represented in solution X that includes all variables x_{dp} equal to 1) the flow on each link f_e can be calculated as described in Sect. 2.2.1 and formulated in (2.2.1b). Next, the link capacity variable y_e for each link $e \in E$ is simply selected as the minimum value that satisfies constraint $My_e \geq f_e$. In order to evaluate the quality of the routing assignment given in solution X , the network cost is calculated as $\sum_{e \in E} \xi_e y_e$ using the obtained values of y_e . The main advantage of this concept is that any flow allocation algorithm such as FD/A/FA shown in Algorithm 2.2 can be applied in this context almost directly.

The FD/AMU/ND algorithm starts with a calculation of an initial solution using any flow allocation algorithm (line 2). A flag *test* used in the stopping condition of the algorithm is initialized next (line 3). The main loop of the algorithm (lines 4–30) is repeated until the new solution improves the network cost function. First, metric l_e is calculated for each link $e \in E$ according to the routing solution given in the current solution X_i (line 5); using this metric, the shortest structure for each demand $d \in D$ is calculated. In the case of an anycast demand, the anycast constraint is satisfied in this function. Next, all demands are examined according to a predefined order (lines 8–23). In particular, each demand attempts to be rerouted to the shortest structure included in set $SR(X_i)$. In the case of anycast demands, both associated demands must be processed jointly (lines 9–11), while multicast and unicast demands are processed individually (line 13). The new solution K is evaluated against the previous solution H , i.e., the link capacity cost is calculated accordingly to the flow allocation (lines 15–17). If the obtained solution is the same as the solution from the previous iteration after processing all demands, the algorithm stops.

A Lagrangian relaxation method is also used in solving the network design problem with anycast, multicast and unicast flows (2.3.1). A general framework of solving network design problems using this method is described in [30]. As in the case of the flow allocation problem, the key issue is the formulation of a dual problem of the original optimization problem by relaxing constraints in order to obtain a simpler subproblem. For more details see [30].

Regarding the EA method described in Sect. 2.2.2, the following modifications are necessary to solve the network design problem with anycast, multicast and unicast flows. First, since various types of flows are considered, the chromosome must be

Algorithm 2.5 FD/AMU/ND (Flow Deviation for Network Design with Anycast, Multicast and Unicast Flows)

Require: set of edges E , set of anycast, multicast and unicast demands D , sets $P(d)$ including candidate structures for each demand $d \in D$, capacity module size M , link cost

Ensure: routing structure selection (routing) for each demand $d \in D$ included in set X_i , value of objective function

```

1: procedure FD/AMU/ND( $D, P(d)$ )
2:  $X_1 := \text{Find\_Initial\_Solution}(D, P(d)), i := 1$ 
3:  $test := 0$ 
4: repeat
5:   for  $e \in E$  do  $l_e := \xi_e \min \{y_e : My_e \geq f_e(X_i)\}$ 
6:    $SR(X_i) := \text{Find\_Shortest\_Structure}(D, P(d), X_i, L)$ 
7:    $H := X_i$ 
8:   for  $d \in D$  do
9:     if  $Type(d) = ANYCAST$  then
10:        $K := (H - \{x_{dp}\}) \cup \{x_{dq}\}$ , where  $x_{dp} \in H$  and  $x_{dq} \in SR(X_i)$ 
11:        $K := (K - \{x_{\tau(d)p'}\}) \cup \{x_{\tau(d)q'}\}$ , where  $x_{\tau(d)p'} \in H$  and  $x_{\tau(d)q'} \in SR(X_i)$ 
12:     else
13:        $K := (H - \{x_{dp}\}) \cup \{x_{dq}\}$ , where  $x_{dp} \in H$  and  $x_{dq} \in SR(X_i)$ 
14:     end if
15:      $F(H) = \text{Find\_Objective}(H)$ 
16:      $F(K) = \text{Find\_Objective}(K)$ 
17:     if  $F(K) < F(H)$  then  $H := K$ 
18:     if  $Type(d) = ANYCAST$  then
19:        $D := D \setminus \{d, \tau(d)\}$ 
20:     else
21:        $D := D \setminus \{d\}$ 
22:     end if
23:   end for
24:   if  $X_i \neq H$  then
25:      $i := i + 1$ 
26:      $X_i := H$ 
27:   else
28:      $test := 1$ 
29:   end if
30: until  $test < 1$ 
31: end procedure

```

divided into three parts, with each part encoding a selection of routing structures for a particular type of flow. This approach involves a straightforward implementation of the crossover operator, which selects the crossover point(s) independently for each part (demand type) of the chromosome. In consequence, the obtained child solution is feasible in terms of constraint (2.3.1b). Note that after the crossover operation, the repair function reported in Algorithm 2.4 is applied only for the chromosome part that refers to anycast demands. In turn, the mutation operator does not require any changes. The second adjustment of the EA algorithm refers to chromosome evaluation. More specifically, since the goal of the optimization is to minimize network cost in terms of capacity cost, to evaluate a particular solution X , link flow f_e for each $e \in E$ is first calculated according to (2.2.1b). Next, the link capacity variable y_e

is calculated for each link $e \in E$ as the minimum value that satisfies constraint $My_e \geq f_e$ and the network cost is obtained. Note that because link capacities are determined according to the flow allocation encoded in the chromosome, the link capacity constraint (2.3.1c) is always satisfied and thus there is no need to use the penalty function mechanism. This approach means that link capacity variables y_e are not directly encoded in the chromosome, but they are selected indirectly according to the formulation of the optimization problem.

Concerning the metaheuristics based on the neighborhood search, e.g., LS, TS, SA, and GRASP, the approach to tackling the network design problem is similar to the EA method. In particular, the solution encodes the routing decision variable x_{dp} only. The link capacity variables y_e are selected according to the flow allocated on each link, which yields a value of the objective function (network cost) that is used to evaluate a particular solution. Consequently, it is not necessary to use the penalty function associated with the link capacity constraint. Again, in the case of anycast demands, the repair function shown in Algorithm 2.4 is required to cope with the anycast constraint. Note that another approach to encoding the solution in the context of neighborhood search methods is proposed in [60] where the authors use two vectors with variables x_{dp} and y_e to represent the solution. To control the feasibility of the solution, a penalty function is used.

2.4 Location Problems for Anycast and Unicast Flows

In this section, we address optimization problems related to the location of DC nodes for a network realizing anycast and unicast flows. Two types of location problems are formulated in this section. Firstly, it is assumed that each installed DC provides the same content/service and in consequence each DC can serve every anycast demand. The goal is to decide on the location of DC nodes, link capacity and routing in order to minimize the cost related to both DC and link capacity resources. In the second problem, DCs offer various types of content divided into content groups (CGs). The optimization process involves locating particular content groups and selecting routing paths.

2.4.1 Data Center Location and Network Design

Location problems are presented in depth in earlier studies, mainly in the context of CDNs including web proxy placement, cache location, replica location, e.g., [49, 61–73]. However, the majority of papers on location problems with anycast flows do not tackle the routing problem, i.e., it is assumed that each client is simply assigned to the closest DC node using the shortest path. This assumption considerably reduces the complexity of the optimization problem, since link capacity is not required in the model. Therefore, the location and network design (NDL) optimization problem

presented below, involving joint optimization of location, link capacity and flow allocation, results in significantly more detailed modeling and optimization of real networks.

The NDL problem for anycast and unicast flows is formulated for both link-path and node-link notations. In order to write the models, some new notation must be introduced. First, let R denote a set of network nodes that can host a data center. The decision variable that determines the location of DC nodes in the network is defined as u_v , which equals 1 if node v hosts a DC node and 0 otherwise. The cost of locating a DC at node $v \in R$ is given by constant η_v .

Link-Path Formulation

To recall, in the link-path notation of anycast flows, the DC node is selected for a demand directly by choosing the routing path. To ensure that the path selected to realize an anycast demand a proper (installed) DC as a DC node, constant π_{vdp} denotes whether node $v \in R$ is a DC node for path p realizing anycast demand d .

CON/AU/NDL/Cost/Link-path

sets

E	links
R	candidate nodes for data center location
D	demands (anycast, unicast)
D^{AN}	anycast demands
D^{DS}	anycast downstream demands
$P(d)$	candidate paths for flows realizing demand d . If d is a unicast demand, the candidate path connects end nodes of the demand. If d is an anycast upstream demand, the candidate path connects the client node and a node included in set R (DC candidate node). If d is a downstream demand, the candidate path connects a node from set R (DC candidate node) and the client node

constants

δ_{edp}	=1, if link e belongs to path p realizing demand d ; 0, otherwise
π_{vdp}	=1, if node $v \in R$ is a DC node for path p realizing anycast demand d ; 0, otherwise
h_d	volume of demand d
ξ_e	unit cost of link e
M	size of the link capacity module
$\tau(d)$	index of a demand associated with demand d . If d is a downstream demand, then $\tau(d)$ must be an upstream demand and vice versa
$s(p)$	origin node of path p

- $t(p)$ destination node of path p
 r number of data centers to be located in network
 η_v cost of location of a data center at node v , if opened

variables

- x_{dp} =1, if path p is used to realize demand d ; 0, otherwise (binary)
 y_e capacity of link e as the number of capacity modules (integer)
 u_v =1, if node v is selected to host a data center; 0, otherwise (binary)

objective

$$\text{minimize } F = \sum_{e \in E} \xi_e y_e + \sum_{v \in R} \eta_v u_v \quad (2.4.1a)$$

constraints

$$\sum_{p \in P(d)} x_{dp} = 1, \quad d \in D \quad (2.4.1b)$$

$$\sum_{d \in D} \sum_{p \in P(d)} \delta_{edp} x_{dp} h_d \leq M y_e, \quad e \in E \quad (2.4.1c)$$

$$\sum_{p \in P(d)} x_{dp} s(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} t(p), \quad d \in D^{DS} \quad (2.4.1d)$$

$$\sum_{p \in P(d)} \pi_{vdp} x_{dp} \leq u_v, \quad d \in D^{AN}, v \in R \quad (2.4.1e)$$

$$\sum_{v \in R} u_v = r. \quad (2.4.1f)$$

The goal of optimization (2.4.1a) is to minimize the sum of link capacity cost and data center location cost. Equality (2.4.1b) ensures that exactly one routing path is selected to realize demand d . Condition (2.4.1c) defines the link capacity constraint to meet the requirement that the flow of each link cannot exceed the allocated link capacity. Equality (2.4.1d) denotes the anycast constraint. Condition (2.4.1e) ensures that an anycast demand can be assigned to a routing path p that includes a DC node installed in the network. More precisely, if a data center is not located at node v (i.e., $u_v = 0$), then the right-hand side of (2.4.1e) must be zero, which guarantees that path p selected to realize demand d cannot use node v as the DC node. The final constraint (2.4.1f) is in the model to ensure that exactly r data centers are located in the network.

Note that heuristic algorithms described in Sects. 2.2.2 and 2.3.2 can be adapted to solve the CON/AU/NDL problem defined as (2.4.1a)–(2.4.1f). To achieve this, the problem must be divided to two separate subproblems: a DC location problem (without capacity and flow allocation) and a network design problem. The first subproblem can be solved using methods developed in the context of cache/replica location problems; see [61–64, 68, 69, 71–73]. When the DCs are located in the network, the pure network design problem with anycast and unicast flows can be solved.

Node-Link Formulation

New notation is required to write the node-link formulation. Let x_{ed} denote whether the routing path selected for demand d uses link e . For anycast demands, variable z_{vd} denotes whether node v is selected as a DC for demand d .

CON/AU/NDL/Cost/Node-link

sets (additional)

V	nodes
$\delta^+(v)$	links leaving node v
$\delta^-(v)$	links entering node v
D^{UN}	unicast demands
D^{US}	anycast upstream demands

constants (additional)

s_d	source node of demand d
t_d	destination node of demand d

variables

x_{ed}	=1, if demand d uses link e ; 0, otherwise (binary)
y_e	capacity of link e as the number of capacity modules (integer)
u_v	=1, if node v is selected to host a data center; 0, otherwise (binary)
z_{vd}	=1, if DC node v is selected as a DC for demand d ; 0, otherwise (binary)

objective

$$\text{minimize } F = \sum_{e \in E} \xi_e y_e + \sum_{v \in R} \eta_v u_v \quad (2.4.2a)$$

constraints

$$\sum_{e \in \delta^+(v)} x_{ed} - \sum_{e \in \delta^-(v)} x_{ed} = \begin{cases} +1 & \text{if } v = s_d \\ -1 & \text{if } v = t_d \\ 0 & \text{otherwise} \end{cases} \quad v \in V, d \in D^{UN} \quad (2.4.2b)$$

$$\sum_{e \in \delta^+(v)} x_{ed} - \sum_{e \in \delta^-(v)} x_{ed} = \begin{cases} +z_{vd} & \text{if } v \in R \\ -1 & \text{if } v = t_d \\ 0 & \text{otherwise} \end{cases} \quad v \in V, d \in D^{DS} \quad (2.4.2c)$$

$$\sum_{e \in \delta^+(v)} x_{ed} - \sum_{e \in \delta^-(v)} x_{ed} = \begin{cases} +1 & \text{if } v = s_d \\ -z_{vd} & \text{if } v \in R \\ 0 & \text{otherwise} \end{cases} \quad v \in V, d \in D^{US} \quad (2.4.2d)$$

$$\sum_{d \in D} h_d x_{ed} \leq M y_e, \quad e \in E \quad (2.4.2e)$$

$$z_{vd} = z_{v\tau(d)}, \quad d \in D^{DS}, v \in R \quad (2.4.2f)$$

$$\sum_{v \in R} z_{vd} = 1, \quad d \in D^{AN} \quad (2.4.2g)$$

$$z_{vd} \leq u_v, \quad d \in D^{AN}, v \in V \quad (2.4.2h)$$

$$\sum_{v \in R} u_v = r. \quad (2.4.2i)$$

The objective function (2.4.2a) is formulated in the same way as in model (2.4.1). Condition (2.4.2b) is a node-link formulation of unicast demands. Equalities (2.4.2c)–(2.4.2d) define the flow conservation constraints for downstream and upstream anycast demands, respectively. More specifically, if node v is included in set R (i.e., node v can be selected as a DC node), the left-hand side of (2.4.2c) must be z_{vd} . Note that if v is not selected as the DC node of downstream demand d ($z_{vd} = 0$), the left-hand side of (2.4.2c) is 0 and v is a transit node. In turn, if node v is selected as the DC node of demand d ($z_{vd} = 1$), the left-hand side of (2.4.2c) is 1 and v is the source node of the demand. If the considered node v is the destination (client) node ($v = t_d$) of downstream demand d , the left-hand side of (2.4.2c) is -1 . Finally, in all other cases, v is a transit node and the left-hand side of (2.4.2c) is 0. Constraint (2.4.2d) defines the flow conservation for upstream demands. Inequality (2.4.2e) denotes the link capacity constraint. Equality (2.4.2f) ensures that two associated demands d and $\tau(d)$ use the same DC node. Constraint (2.4.2g) is in the model to guarantee that every anycast demand uses exactly one DC node. Constraint (2.4.2h) imposes the requirement that an anycast demand can be assigned to node v that hosts a DC. The last inequality (2.4.2i) limits the number of DCs to be installed in the network.

2.4.2 Content Location and Flow Allocation

This section presents the problem of content location and flow allocation in a network with anycast and unicast flows. The content location problem is similar to other location problems, reviewed in depth in literature, such as the data placement problem [74, 75] and object placement problem [61, 62, 76–81]. However, the majority of previous research in this area considers location problems assuming that the optimization of routing (flow allocation) is not included in the model, which significantly simplifies the formulation and the optimization process.

The optimization model addressed in this section uses a concept of a *content group*, which follows directly from the observation that online content can be divided into groups according to popularity [61, 76, 77, 82–87]. More specifically, set B includes content groups and constant ψ_b denotes the size of data related to content group b given in GB. Due to the fact that popularity of a particular content group can depend on the geographical location of users and on the population of users located at node v , constant h_{vb} defines the volume (bit-rate) of content group b requested by a client located at node v . This means that network flow associated with a particular content group b can vary for different network nodes. Content groups are located at DCs node that are already placed in the network. Binary variable u_{vb} determines whether the DC located at node v stores content group b . However, DCs have a limited storage capacity, therefore the whole content cannot be placed in every DC node. To make the problem more realistic, it is assumed that some background unicast traffic is transmitted in the network and h_{vw} denotes the volume (bit-rate) of unicast traffic from node v to node w .

Since anycast demands are defined for each node, route modeling is a little different than in the previous models. In particular, set $P(v, w)$ includes candidate paths for flows realizing demand from node v to node w . The assignment of clients to a DC providing the requested content is controlled by a binary variable z_{vwb} that denotes whether node v downloads content group b from a DC located at node w . Due to traffic asymmetry, the upstream traffic from clients to DCs is not included directly in the model. However, the bit-rate associated with the upstream traffic is built-in unicast flows between particular nodes (constant h_{vw}). To reduce the number of connections to be established in the network, traffic grooming is assumed. More specifically, the whole traffic between a particular pair of nodes (both anycast traffic serving downloads of content groups and unicast traffic) is provisioned using a single connection. Without this assumption, due to a potential large number of content groups, the number of connections serving individual requests to single content groups may be high. In consequence, binary variable x_{vwp} denotes whether path p is used to realize the demand from node v to node w . Similarly, variable f_{vwp} represents the volume of flow from node v to node w allocated to path p .

CON/AU/FAL/Content Groups/Cost

sets

E	links
V	nodes (clients) that request content
R	nodes with a data center
$P(v, w)$	candidate paths for flows realizing demand from node v to node w
B	content groups

constants

δ_{evwp}	=1, if link e belongs to path p realizing demand between from node v to node w ; 0, otherwise
h_{vw}	volume (bit-rate) of unicast traffic from node v to node w
g_{vb}	volume (bit-rate) of content group b requested by node v
s_v	storage capacity of a DC located at node v (GBytes)
ψ_b	size of data related to content group b (GBytes)
c_e	capacity of link e
ζ_e	unit routing cost on link e
M	large number

variables

x_{vwp}	=1, if path p is used to realize demand from node v to node w ; 0, otherwise (binary)
z_{vwb}	=1, if node v downloads content group b from a DC located at node w ; 0, otherwise (binary)
f_{vwp}	volume of flow from node v to node w allocated to path p (continuous non-negative)
u_{vb}	=1, if DC located at node v stores content group b ; 0, otherwise (binary)

objective

$$\text{minimize } F = \sum_{e \in E} \sum_{v, w \in V} \sum_{p \in P(v, w)} \delta_{evwp} \zeta_e f_{vwp} \quad (2.4.3a)$$

constraints

$$\sum_{p \in P(v, w)} x_{vwp} = 1, \quad v, w \in V \quad (2.4.3b)$$

$$f_{vwp} \leq M x_{vwp}, \quad v, w \in V, p \in P(v, w) \quad (2.4.3c)$$

$$\sum_{p \in P(v, w)} f_{vwp} \geq h_{vw} + \sum_{b \in B} z_{vwb} g_{wb}, \quad v, w \in V \quad (2.4.3d)$$

$$\sum_{v,w \in V} \sum_{p \in P(d)} \delta_{evwp} f_{vwp} \leq c_e, \quad e \in E \quad (2.4.3e)$$

$$z_{vwb} \leq u_{vb}, \quad v \in R, w \in V, b \in B. \quad (2.4.3f)$$

$$\sum_{b \in B} \psi_b u_{vb} \leq s_v, \quad v \in R \quad (2.4.3g)$$

$$\sum_{v \in R} u_{vb} \geq 1 \quad b \in B. \quad (2.4.3h)$$

The goal of optimization (2.4.3a) is to minimize the routing cost of network flows required to transmit content from DCs and the background unicast traffic. Equality (2.4.3b) imposes a single path routing. Constraint (2.4.3c) binds variables x_{vwp} and f_{vwp} to ensure that flow on path $p \in P(v, w)$ is zero if path p is not selected to realize traffic from node v to node w . Condition (2.4.3d) implements traffic grooming, i.e., the volume allocated to a path selected for node pair v and w must be enough to serve both unicast traffic (h_{vw}) and content traffic ($\sum_{b \in B} z_{vwb} g_{wb}$). Inequality (2.4.3e) is the link capacity constraint. Condition (2.4.3f) ensures that anycast client at node w can download a content group b from DC located at node v , only if DC v stores content group b . The next inequality (2.4.3g) controls the storage capacity limit of each DC. Finally, constraint (2.4.3h) ensures that every content group is allocated to at least one DC node. Note that to introduce additional survivability constraints, the right-hand side of constraint (2.4.3h) could be changed to 2 or a larger number to enable content replication.

The most straightforward way of solving the above problem using heuristic algorithms is to consider two separate subproblems. The decision on content group placement needs to be determined first, which makes the problem a pure flow allocation problem and methods proposed in Sect. 2.2.2 can be applied.

2.5 Survivable Allocation of Anycast and Unicast Flows

This section focuses on the flow allocation problem with additional survivability constraints. Two types of network flows are addressed: anycast and unicast. We start with a short discussion on survivability of connection-oriented networks.

The key idea of providing survivability in a connection-oriented network is to establish two failure-disjoint paths for each demand, i.e., *working* path used in a normal, failure-free state of the network and *backup* path used when the working path is not available due to a network failure. Two popular protection methods based on this approach and considered in the context of connection-oriented networks are Dedicated Path Protection (DPP) and Shared Backup Path Protection (SBPP). The key difference between DPP and SBPP is the fact that SBPP makes it possible to share capacity between backup paths belonging to different demands under the condition that these resources can be used by a single demand in a given failure scenario. In

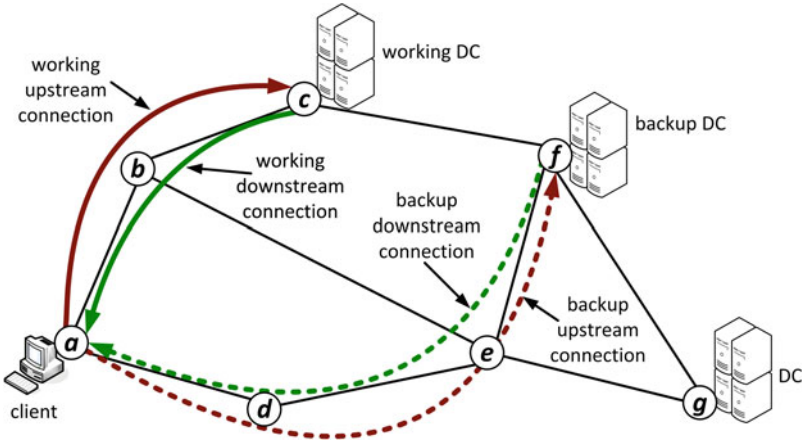


Fig. 2.1 An example of a survivable anycast connection with different working and backup DCs

contrast, backup paths in DPP have their own dedicated capacity. This means that SBPP brings capacity savings compared to DPP.

Additionally, there are two categories of DPP schemes: $1+1$ and $1:1$. In the former, traffic is permanently duplicated on both the working path and the backup path. The receiving node selects the signal with the best quality. The $1+1$ approach is very efficient in terms of recovery time, since the reaction to a network failure is almost immediate. However, the $1+1$ method is relatively costly in terms of capacity usage. On the other hand, the $1:1$ method assumes that in failure-free conditions traffic is transmitted over the working path, which means extra traffic can be transported along the backup path in failure-free conditions. When a failure occurs along the working path, the extra traffic must be prevented from entering the backup path to enable switching the traffic affected by the failure to the backup path. Consequently, in comparison with $1+1$, the $1:1$ approach requires higher recovery times [5].

In the context of anycast flows, path protection methods such as DPP and SBPP can be used differently to the protection of unicast flows. More specifically, anycasting assumes that the same content/service can be provisioned by several DCs located in the network. In consequence, working and backup paths of the same demand can use different DC nodes. For ease of reference, the DC node used by the working path is denoted as *working DC* and the DC node used by the backup path is referred to as *backup DC*. There are two main reason for this relocation approach using different DC nodes for working and backup paths. Firstly, this concept protects the network against a failure of a single DC node, improving network survivability. Secondly, in some cases a backup path connected to another DC node can provide better performance according to the considered metric compared to a backup path that uses the same DC node as the working path. It should be noted that both associated anycast demands (upstream and downstream) must use the same DC node for working paths and backup paths, respectively [88]. The relocation approach is presented in [70, 89–99].

To illustrate the concept of a backup DC, a simple example is shown in Fig. 2.1. An anycast client is located at node 1 and three DCs are placed in the network at nodes c, f and g . The anycast client uses node c as the working DC and node f as the backup DC. Both downstream and upstream connections are shown. Note that if the backup DC is located in the same node as the working DC (node c), the shortest backup paths will include at least 4 links (for instance path (a, d, e, b, c)), when the backup DC located at node f is used, the backup path is shorter and includes only 3 links (path (a, d, e, f)).

2.5.1 Formulations

In this section, two optimization models using DPP and SBPP approaches to provide network survivability are formulated and discussed. Both models use the node-link formulation of anycast and unicast flows with some enhancements for addressing additional survivability requirements. The network is protected against a single link failure. The following models can be easily adapted to tackle other types of failures, such as a single node failure or a region failure [70, 98, 99].

Dedicated Path Protection

In DPP, two types of flow variables are used for each demand: x_{ed} denoting whether working path of demand d uses link e , and b_{ed} denoting whether the backup path of demand d uses link e . In anycast demands, two types of DC node selection variables are also required, i.e., z_{vd} defining whether DC node v is selected as a working DC for demand d and w_{vd} defining whether DC node v is selected as a backup DC for demand d .

CON/AU/FA/DPP/Cost

sets

V	nodes
R	data center nodes
E	links
D	anycast demands
$\delta^+(v)$	links leaving node v
$\delta^-(v)$	links entering node v
D	demands (anycast and unicast)
D^{UN}	unicast demands
D^{AN}	anycast demands
D^{DS}	anycast downstream demands
D^{US}	anycast upstream demands

constants

- h_d volume of demand d
 c_e capacity of link e
 ζ_e unit routing cost on link e
 s_d source node of demand d
 t_d destination node of demand d
 $\tau(d)$ index of a demand associated with demand d . If d is a downstream demand, then $\tau(d)$ must be an upstream demand and vice versa

variables

- x_{ed} =1, if demand d uses link e on working path; 0, otherwise (binary)
 b_{ed} =1, if demand d uses link e on backup path; 0, otherwise (binary)
 z_{vd} =1, if DC node v is selected as a working DC for demand d ; 0, otherwise (binary)
 w_{vd} =1, if DC node v is selected as a backup DC node for demand d ; 0, otherwise (binary)

objective

$$\text{minimize } F = \sum_{e \in E} \sum_{d \in D} \zeta_e h_d (x_{ed} + b_{ed}) \quad (2.5.1a)$$

constraints

$$\sum_{e \in \delta^+(v)} x_{ed} - \sum_{e \in \delta^-(v)} x_{ed} = \begin{cases} +1 & \text{if } v = s_d \\ -1 & \text{if } v = t_d, \\ 0 & \text{otherwise} \end{cases} \quad v \in V, d \in D^{UN} \quad (2.5.1b)$$

$$\sum_{e \in \delta^+(v)} b_{ed} - \sum_{e \in \delta^-(v)} b_{ed} = \begin{cases} +1 & \text{if } v = s_d \\ -1 & \text{if } v = t_d, \\ 0 & \text{otherwise} \end{cases} \quad v \in V, d \in D^{UN} \quad (2.5.1c)$$

$$\sum_{e \in \delta^+(v)} x_{ed} - \sum_{e \in \delta^-(v)} x_{ed} = \begin{cases} +z_{vd} & \text{if } v \in R \\ -1 & \text{if } v = t_d \\ 0 & \text{otherwise} \end{cases} \quad v \in V, d \in D^{DS} \quad (2.5.1d)$$

$$\sum_{e \in \delta^+(v)} b_{ed} - \sum_{e \in \delta^-(v)} b_{ed} = \begin{cases} +w_{vd} & \text{if } v \in R \\ -1 & \text{if } v = t_d \\ 0 & \text{otherwise} \end{cases} \quad v \in V, d \in D^{DS} \quad (2.5.1e)$$

$$\sum_{e \in \delta^+(v)} x_{ed} - \sum_{e \in \delta^-(v)} x_{ed} = \begin{cases} +1 & \text{if } v = s_d \\ -z_{vd} & \text{if } v \in R \\ 0 & \text{otherwise} \end{cases} \quad v \in V, d \in D^{US} \quad (2.5.1f)$$

$$\sum_{e \in \delta^+(v)} b_{ed} - \sum_{e \in \delta^-(v)} b_{ed} = \begin{cases} +1 & \text{if } v = s_d \\ -w_{vd} & \text{if } v \in R \\ 0 & \text{otherwise} \end{cases} \quad v \in V, d \in D^{US} \quad (2.5.1g)$$

$$z_{vd} = z_{v\tau(d)}, \quad d \in D^{DS}, v \in R \quad (2.5.1h)$$

$$w_{vd} = w_{v\tau(d)}, \quad d \in D^{DS}, v \in R \quad (2.5.1i)$$

$$\sum_{v \in R} z_{vd} = 1, \quad d \in D^{AN} \quad (2.5.1j)$$

$$\sum_{v \in R} w_{vd} = 1, \quad d \in D^{AN} \quad (2.5.1k)$$

$$x_{ed} + b_{ed} \leq 1, \quad e \in E, d \in D \quad (2.5.1l)$$

$$\sum_{d \in D} h_d(x_{ed} + b_{ed}) \leq c_e, \quad e \in E. \quad (2.5.1m)$$

The objective (2.5.1a) is to find the routing of both working and backup paths for all anycast and unicast demands using the DPP approach and minimizing routing cost. Equations (2.5.1b) and (2.5.1c) define the unicast flow conservation constraints for working and backup paths, respectively. Equations (2.5.1d)–(2.5.1g) define the corresponding flow conservation constraints for working and backup paths of downstream and upstream anycast demands. Constraints (2.5.1h) and (2.5.1i) are in the model to ensure that working paths of two associated anycast demands d and $\tau(d)$ use the same working and backup DC node, respectively. Equations (2.5.1j) and (2.5.1k) ensure that each anycast demand is assigned to exactly one working and backup DC node, respectively. Constraint (2.5.1l) expresses the DPP, i.e., it ensures that working and backup path are link disjoint for each demand. Finally, the inequality (2.5.1m) ensures the link capacity constraint taking into account flows of both working and backup paths.

The model (2.5.1a)–(2.5.1m) does not include a coupling between the working and backup DC nodes. Two cases are possible for each anycast demand: (i) working and backup DCs are located in different network nodes and (ii) working and backup DC nodes are located in the same network node. For ease of reference, the above model is referred to as the ADN (Any DC node) model. Some additional constraints on the selection of working and backup DC nodes are introduced and discussed below.

The disjoint DC node (DDN) scenario assumes that the working and backup DC nodes are disjoint for each anycast demand:

$$\sum_{v \in R} (z_{vd} + w_{vd}) \leq 1, \quad d \in D^{DS}. \quad (2.5.1n)$$

DDN model given by (2.5.1a)–(2.5.1n) provides protection against a single DC node failure, since it protects each anycast demand against any single DC node failure including the working DC node.

Let κ_{vd} denote a binary constant which is 1 if v is the nearest DC for anycast demand d and 0 otherwise. To find the value of κ_{vd} , the shortest paths between the client node of anycast demand d and each DC node $v \in R$ are calculated and the DC node with the lowest value is selected. The next model, known as the nearest

DC node (NDN), ensures that both working and backup DC nodes are located in the same network node, which is the closest to the client node of a particular anycast demand:

$$z_{vd} = w_{vd} = \kappa_{vd}, \quad d \in D^{DS}, v \in R. \quad (2.5.1o)$$

The main advantage of the NDN model given by (2.5.1a)–(2.5.1m) and (2.5.1o) is the fact that in many anycasting systems the client is assigned to the nearest DC node by default. A more comprehensive treatment of DPP of anycast flows and various working and backup DC node scenarios is given in [70, 98].

Shared Backup Path Protection

The next model assumes the SBPP approach applied to protect the network against a single link failure [99]. Since the SBPP model is similar to the DPP model formulated as (2.5.1), we only present additional elements here. To control the sharing of capacity among backup paths three new variables are introduced. First, let binary variable y_{deg} denote whether link e is used for a backup path of demand d in the event of link g failure. Using variable y_{deg} , we are able to provide sharing of the backup capacity among demands for which working paths are failure disjoint. Next, variable y_{eg} defines the spare capacity on link e required in the event of link g failure. Finally, variable y_e denotes the maximum amount of spare capacity on link e required in the event of a single link failure, and is calculated simply as the maximum value y_{eg} considering all single link failures one by one.

CON/AU/FA/SBPP/Cost

variables (additional)

- y_{deg} =1, if in the event of link g failure link e is used as a backup path of demand d ;
0, otherwise
- y_{eg} spare capacity on link e required in the event of a link g failure (integer)
- y_e maximum spare capacity on link e required in the event of a single link failure
(integer)

objective

$$\text{minimize } F = \sum_{e \in E} \sum_{d \in D} \zeta_e h_d x_{ed} + \sum_{e \in E} \zeta_e y_e \quad (2.5.2a)$$

constraints (2.5.1a)–(2.5.1k) and

$$x_{ed} + b_{gd} \leq 1 + y_{deg}, \quad e, g \in E : e \neq g, d \in D \quad (2.5.2b)$$

$$2y_{deg} \leq x_{ed} + b_{gd}, \quad e, g \in E : e \neq g, d \in D \quad (2.5.2c)$$

$$y_{eg} = \sum_{d \in D} h_d y_{deg}, \quad e, g \in E : e \neq g \quad (2.5.2d)$$

$$y_{eg} \leq y_e, \quad e, g \in E : e \neq g \quad (2.5.2e)$$

$$y_e + \sum_{d \in D} h_d x_{ed} \leq c_e, \quad e \in E. \quad (2.5.2f)$$

The objective function (2.5.2a) aims to minimize the network similarly to function (2.5.1a); however, the cost related to working paths is included directly in the objective function (first term), while the cost related to backup paths is limited to the cost of spare capacity (second term). When compared with the DPP, the SBPP model (2.5.2) includes the same constraints regarding the definition of flows and survivability requirements, and thus these constraints are not repeated in the formulation. There are five new constraints following from the SBPP approach. Constraint (2.5.2b) and (2.5.2c) are used to define variable y_{deg} . Constraint (2.5.2b) ensures that if both variables of the left-hand side are 1 (i.e., demand d is affected by the failure of link g and link e is used by the backup path of demand d), variable y_{deg} must be 1. In turn, constraint (2.5.2c) guarantees that if at least one of the variables x_{ed} and b_{gd} is 0, then y_{deg} must also be set to 0. Equality (2.5.2d) directly defines variable y_{eg} that denotes the spare capacity required for on link e when link g is broken. Constraint (2.5.2e) is used to find value of y_e defined as the maximum amount of spare capacity on link e taking into account all failure scenarios. Finally, inequality (2.5.2f) imposes the link capacity constraint, i.e., for each link $e \in E$ the flow allocated to working paths and the capacity left for backup paths cannot exceed the link capacity.

2.5.2 Numerical Results

This section presents and discusses results of numerical experiments reported in [99]. The main goal of the experiments was to compare the SBPP approach assuming sharing of the backup capacity against the DPP approach, where the backup capacity is not shared in the context of two working and backup DC node scenarios, namely, ADN and NDN. Numerical experiments were performed with the CPLEX solver [100] using optimization models (2.5.1) and (2.5.2) formulated in the previous section.

All simulations were run on the NSF network (Fig. A.4, Table A.3). All links were assigned a capacity equal to 40 Gb/s. Several scenarios referring to 2 and 3 DC nodes available in the network were evaluated. DCs were located at nodes with a relatively high value of the average node degree. To examine the influence of anycast traffic on the analyzed protection mechanisms, we define the *anycast*

ratio (AR) parameter. Let h^{Any} and h^{Uni} denote the overall volume of all anycast and unicast demands, respectively. Next, let $h^{All} = h^{Any} + h^{Uni}$ be the overall demand in the network. The AR parameter is defined as the volume (capacity) of all anycast demands divided by the volume of all demands in the network, i.e., $AR = h^{Any} / h^{All}$. Eight scenarios of network load were examined in terms of the AR parameter, namely, 0, 10, 20, ..., 80 %. In each case, three sets of demands were generated at random, giving 24 different demand sets in total. The number of unicast demands in each set was selected in the range 7–44, while the corresponding number of anycast demands was in the range 8–28. The demand volume was selected from the range 1–9 Gb/s in order to obtain the particular anycast proportion parameter value. For each set of demands, experiments considered two scenarios (2 and 3 DC nodes) and four models (SBPP-ADN, SBPP-NDN, DPP-ADN, DPP-NDN). This yields the overall number of 192 distinct experiments.

To report the results showing the comparison between the SBPP and DPP approaches, we use a ratio calculated as F^{SBPP} / F^{DPP} , where F^{SBPP} and F^{DPP} denote the value of a given performance metric obtained for SBPP and DPP models, respectively. It should be noted that if the value of this ratio is lower than 1, then SBPP provides a lower value of a particular performance metric compared with DPP.

Figure 2.2 shows the $cost^{SBPP} / cost^{DPP}$ ratio as a function of the anycast ratio parameter. In turn, Table 2.1, presents the average values of the ratio between SBPP and DPP (averaged over all cases of the analyzed anycast traffic proportion) for the following performance metrics: cost (objective function of both models), capacity utilization (ratio of network capacity allocated to demands), working and backup path length in km, and hop count for both unicast and anycast demands.

The average value of the $cost^{SBPP} / cost^{DPP}$ ratio (taking into account all experiments) is 0.64, which means that SBPP outperforms DPP by 36 %. The detailed analysis of the results presented in Fig. 2.2 indicates that the ratio between both models becomes less evident with the increase of the anycast ratio. This can be explained by the fact that anycast demands must have one of the end nodes located at the DC node. Let us recall that the capacity sharing used in the SBPP approach assumes that the backup capacity can be shared by demands that have failure-disjoint working paths. However, the majority of anycast traffic concentrates on links adja-

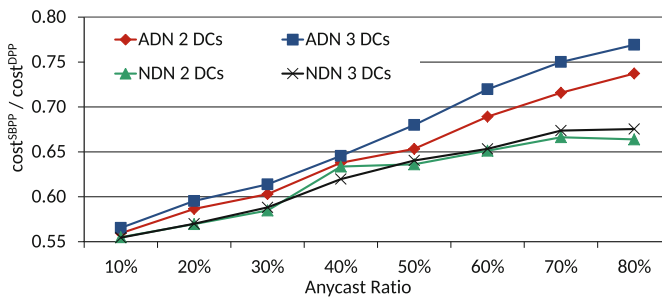


Fig. 2.2 Average cost ratio between SBPP and DPP models as a function of the anycast ratio

Table 2.1 Average ratio between SBPP and DPP approaches for various performance metrics

Number of DC nodes	2	3	2	3
DC node scenarios	ADN	ADN	NDN	NDN
Cost	0.65	0.67	0.62	0.62
Capacity utilization	0.60	0.61	0.56	0.57
Unicast working path length	1.01	1.01	1.01	1.05
Unicast backup path length	1.71	1.68	1.78	1.86
Unicast working path hops	1.01	1.01	0.99	1.03
Unicast backup path hops	1.49	1.43	1.53	1.55
Anycast working path length	1.01	1.06	1.01	1.03
Anycast backup path length	2.00	2.09	1.79	1.91
Anycast working path hops	1.00	0.98	1.01	1.02
Anycast backup path hops	1.54	1.60	1.43	1.56

cent to DC nodes. Accordingly, there are fewer opportunities to share the backup capacity in the event of a failure of a link adjacent to a DC node.

Moreover—as shown in Fig. 2.2—the increase of the number of DC nodes (here from 2 to 3) also reduces the gap between SBPP and DPP. As DC nodes are spread over the network, anycast demands use shorter backup paths than unicast demands in terms of the hop count. Accordingly, when the number of DC nodes increases, the average path hop count decreases, which means that once again there are fewer possibilities to share the backup capacity.

As shown in Table 2.1, the DC node location scenario (ADN vs. NDN) does not have a significant impact on the SBPP/DPP ratio. Nevertheless, the ADN approach provides lower values of the cost percentage difference (equivalent to higher values of the SBPP/DPP ratio) compared to the NDN scenario. This is because the ADN scenario is flexible and benefits more from switching anycast demands to another DC node after a network failure. However, in the NDN approach, both working and backup paths of anycast demands are assigned to the same, closest DC node. Therefore, in this case anycast traffic performs similarly to unicast traffic; in consequence—as explained above in the context of cost values (see Fig. 2.2)—the difference between the SBPP and DPP approaches increases.

Another interesting observation is that the capacity utilization metric returns a performance similar to the cost objective, i.e., as the anycast ratio parameter and the number of DC nodes increase, the difference between SBPP and DPP decreases. Note that on average, the SBPP model requires 42 % less capacity than DPP. Moreover, the results show that the SBPP model yields significantly longer backup paths compared to the DPP model. This is because the objective function (2.5.2a) includes the cost of the shared backup capacity, not the cost of every backup path as in the DPP model. Hence, in the SBPP model backup paths are selected in order to share the backup capacity, even if they become relatively long. More information on SBPP protection of anycast and unicast flows and additional results can be found in [99].

2.6 Protection Design with Anycast and Unicast Flows

This section continues the discussion on the protection of connection-oriented networks with anycast flows and presents a network design problem. In contrast to the previous section where the node-link notation was used to model network flows, here we use the link-path approach. Moreover, a general concept of *failure state* (situation) is applied to model network failures [4]. Simply put, a failure state models any failure that can occur in the network and it is specified by the availability status of the network elements (links and nodes). The DPP approach is as the protection method; the optimization and heuristic can be modified easily to also address the SBPP approach.

2.6.1 Formulation

The optimization model involves the joint optimization of anycast and unicast flows protected by the DPP method [101–103]. The objective is to minimize the cost of the network essential to fully protect all flows (unicast and anycast). As in [4], the notion of failure states is used. Let set S contain all failure states considered in the network including the special state s_0 denoting the normal state of the network when all network elements are available. Each failure state $s \in S$ is defined by a vector of binary link availability coefficients $\alpha_s = (\alpha_{1s}, \alpha_{2s}, \dots, \alpha_{|E|s})$, i.e., in a given failure state a particular link e is either fully available ($\alpha_{es} = 1$) or it is completely broken ($\alpha_{es} = 0$). This approach, makes it easy to model various failure scenarios. For instance, if a single link failure scenario is studied, then set S includes $|E| + 1$ failure states, s_0 denotes the normal state with all links available ($\alpha_{es_0} = 1$ for each $e \in E$) and s_e denotes the situation when link e is broken ($\alpha_{es_e} = 0$).

To provide path protection for each demand $d \in D$, set $P(d)$ including candidate pairs of the failure situation disjoint paths is given. Each pair of paths is denoted as (w_{dp}, b_{dp}) , where w_{dp} refers to the working path and b_{dp} refers to the backup path. Constant δ_{edp} defines working path w_{dp} and is 1 if link e belongs to w_{dp} . In turn, constant β_{edp} describes the backup path b_{dp} and is 1 if link e belongs to b_{dp} . The working path w_{dp} (used in the normal, failure-free network state) is protected by the dedicated backup path b_{dp} , which is failure-situation disjoint with the working path w_{dp} . This approach ensures that when a particular working path is broken, its backup path must be available for each failure state $s \in S$. The binary availability coefficient θ_{dps} indicates whether the working path w_{dp} is affected by a failure s . In particular, $\theta_{dps} = \prod_{e \in w_{dp}} \alpha_{es}$ since if at least one link e of path w_{dp} is broken in state s ($\alpha_{es} = 0$), then path w_{dp} is not available ($\theta_{dps} = 0$) and must be restored using backup path b_{dp} .

Because associated anycast connections d and $\tau(d)$ must be connected to the same DC, there are two possible cases when a failure affects demand d and/or $\tau(d)$. Firstly, in failure state s only one demand of the pair $(d, \tau(d))$ is broken. Let d denote

the broken connection, i.e., working path w_{dp} selected for demand d containing a link broken in state s . Thus, backup path b_{dp} must use the same DC node as the DC included in working path $w_{\tau(d)q}$ of demand $\tau(d)$. In the second case, both associated anycast demands $(d, \tau(d))$ fail due to failure s (i.e., both working paths w_{dp} and $w_{\tau(d)q}$ are broken in situation s). Then, to restore demands d and $\tau(d)$, backup paths b_{dp} and $b_{\tau(d)q}$ can use a different DC to the node included in corresponding working paths w_{dp} and $w_{\tau(d)q}$, respectively. This approach following directly from the anycast paradigm should make it possible to reduce network cost, and it is analogous to the backup DC concept described in Sect. 2.5.

CON/AU/ND/DPP/Cost/Link-path

sets

E	links
D	demands (anycast, unicast)
D^{DS}	anycast downstream demands
$P(d)$	pairs of failure disjoint candidate paths for flows realizing demand d . If d is a unicast demand, the candidate path connects end nodes of the demand. If d is an anycast upstream demand, the candidate path connects client node and DC node. If d is a downstream demand, the candidate path connects the DC node and the client node
S	failure states (situations)

constants

δ_{edp}	$=1$, if link e belongs to working path w_{dp} realizing demand d ; 0 , otherwise
β_{edp}	$=1$, if link e belongs to backup path b_{dp} realizing demand d ; 0 , otherwise
h_d	volume of demand d
ξ_e	unit cost of link e
M	size of the link capacity module
$\tau(d)$	index of a demand associated with demand d . If d is a downstream demand, then $\tau(d)$ must be an upstream demand and vice versa
θ_{dps}	binary availability coefficient of working path w_{dp} in state s
$o(p)$	origin node of working path p
$t(p)$	destination node of working path p
$\bar{o}(p)$	origin node of backup path p
$\bar{t}(p)$	destination node of backup path p

variables

x_{dp}	$=1$, if pair of paths (w_{dp}, b_{dp}) is used to realize demand d ; 0 , otherwise (binary)
y_e	capacity of link e as the number of capacity modules (integer)

objective

$$\text{minimize } F = \sum_{e \in E} \xi_e y_e \quad (2.6.1a)$$

constraints

$$\sum_{p \in P(d)} x_{dp} = 1, \quad d \in D \quad (2.6.1b)$$

$$\sum_{d \in D} \sum_{p \in P(d)} x_{dp} h_d (\delta_{edp} \theta_{dps} + \beta_{edp} (1 - \theta_{dps})) \leq M y_e, \quad e \in E, s \in S \quad (2.6.1c)$$

$$\sum_{p \in P(d)} x_{dp} o(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} t(p), \quad d \in D^{DS} \quad (2.6.1d)$$

$$\sum_{p \in P(d)} x_{dp} \bar{o}(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} \bar{t}(p), \quad d \in D^{DS}. \quad (2.6.1e)$$

The objective (2.6.1a) is to minimize the cost of network capacity required for working flows and for protection against all failure scenarios included in set S . Condition (2.6.1b) is in the model to guarantee that a single pair of paths (working and backup) is selected to realize demand d . Inequality (2.6.1c) controls the link capacity constraint. More precisely, for each link $e \in E$ and each possible failure state $s \in S$, the allocated link capacity (right-hand side of (2.6.1c)) must exceed the flow on the link (left-hand side of (2.6.1c)). Note that if working path w_{dp} is not available (i.e., at least one link belonging to w_{dp} is broken in state s and $\theta_{dps} = 0$), then backup path b_{dp} is activated. Therefore, the left-hand side of the link capacity constraint (2.6.1c), includes both working flows transmitted in the event of failure s (term $\sum_{d \in D} \sum_{p \in P(d)} x_{dp} h_d \delta_{edp} \theta_{dps}$) and backup flows activated after failure s (term $\sum_{d \in D} \sum_{p \in P(d)} x_{dp} h_d \beta_{edp} (1 - \theta_{dps})$). Constraint (2.6.1c) assumes the stub-release scenario, i.e., the flow of the broken working path is released in the network and this free capacity can be used for restoration [4]. Constraints (2.6.1d) and (2.6.1e) ensure that working and backup paths of two associated anycast demands connect the same pair of nodes, respectively.

Note that in the problem (2.6.1a)–(2.6.1e) the DC node of an anycast demand can be changed due to the restoration process (backup DC concept). To remove this option, the following constraint must be added to the model:

$$\sum_{p \in P(d)} x_{dp} o(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} \bar{o}(p), \quad d \in D^{DS}. \quad (2.6.1f)$$

More discussion on joint optimization of anycast and unicast flows with additional survivability constraints can be found in [101–103].

2.6.2 Cut Inequalities

Cut inequalities are used to facilitate the optimization process. More specifically, cut inequalities enable the branching phase of the branch-and-cut algorithm to use additional information included in the cuts in calculations of more effective bounds. A cut-and-branch variant of the branch-and-cut algorithm assumes that cut inequalities are added in the root node of the solution tree only. In consequence, all generated cuts are valid throughout the whole solution tree [104]. The main advantage of this approach is that the cut inequalities are calculated once only, and more time can be spent generating relatively tight bounds compared to the classical scenario, when cuts are generated in each node of the solution tree. For more general information on applications of branch-and-cut algorithms with cut inequalities to various network optimization problems see [4, 104–115].

The first cut inequality proposed for problem (2.6.1) is a version of a partition inequality obtained by separating network nodes into two subsets and analyzing the flow and capacity of links included in the cut between these two sets [108]. The key modifications of the below approach—compared to the classical partition inequality—follow from two specific features of the problem (2.6.1), i.e., anycasting and survivability.

Set V contains all nodes in the considered network. Let R denote the set embracing all nodes that host a data center and let $C = V \setminus R$ denote the set of all other network nodes. For ease of notation, let $o(e)$ and $t(e)$ be the origin node and destination node of link e , respectively. Analogously, let $o(d)$ and $t(d)$ denote the origin node and destination node of demand d , respectively. Let us recall that in the context of anycast demands, the origin (destination) node of upstream (downstream) connection is the client node. The destination (origin) nodes of upstream (downstream) demand are selected from DC nodes included in set R .

Let $\eta(W)$ denote a graph cut induced by $W \subseteq V$, i.e., $\eta(W)$ includes all links with the origin node in set W and the destination node in its complement set $(V \setminus W)$. Note that links are directed, i.e., in the case of $\eta(W)$ links originate in W . Moreover, $h(W, W')$ is the overall flow related to all demands that have the origin node included in W and the destination node included in W' :

$$h(W, W') = \sum_{d: o(d) \in W, t(d) \in W'} h_d. \quad (2.6.2)$$

For instance, $h(R, C)$ defines the overall flow from client nodes to DC nodes taking into account both unicast and anycast demands. To formulate the DC partition inequality, the following inequalities are defined:

$$\sum_{e \in \eta(R)} My_e \geq h(R, C) \quad (2.6.3a)$$

$$\sum_{e \in \eta(C)} My_e \geq h(C, R). \quad (2.6.3b)$$

Inequality (2.6.3a) is explained as follows. The left-hand side of (2.6.3a) is the overall link capacity necessary to be installed on links included in $\eta(R)$ to satisfy traffic coming from nodes included in R to nodes included in C (right-hand side), i.e., cut $\eta(R)$ must carry the whole anycast downstream traffic as well as unicast traffic related to demands that originate in nodes from R . Similarly, inequality (2.6.3b) is formulated for traffic in the opposite direction. Using the mixed-integer rounding (MIR) approach [108, 111], it is possible to make (2.6.3a) and (2.6.3b) stronger as follows:

$$\sum_{e \in \eta(R)} y_e \geq \lceil \frac{h(R, C)}{M} \rceil \quad (2.6.4a)$$

$$\sum_{e \in \eta(C)} y_e \geq \lceil \frac{h(C, R)}{M} \rceil. \quad (2.6.4b)$$

Finally, network survivability can be taken into account to provide more effective cut inequalities. In a nutshell, assuming a single link failure scenario and using a single backup path to protect the network in 100 %, the left-hand side of (2.6.4a) and (2.6.4b) can be modified by removing from a particular cut one by one a single link. At the same time, all other remaining links must provide enough capacity to carry the whole traffic between sets R and C in both directions:

$$\sum_{e \in \eta(R) \setminus e'} y_e \geq \lceil \frac{h(R, C)}{M} \rceil, \quad e' \in \eta(R) \quad (2.6.5a)$$

$$\sum_{e \in \eta(C) \setminus e'} y_e \geq \lceil \frac{h(C, R)}{M} \rceil, \quad e' \in \eta(C). \quad (2.6.5b)$$

Constraints (2.6.5a) and (2.6.5b) are the final versions of the partition inequality.

To formulate the next inequality, the following notation is introduced. First recall that $\delta^+(v)$ is defined as a set of links that originate at node v and $\delta^-(v)$ is a set of all links that terminate in node v . Moreover, let $h^+(v) = \sum_{d: o(d)=v} h_d$ and let $h^-(v) = \sum_{d: t(d)=v} h_d$ denote the overall demand flow leaving and entering node v , respectively. This cut inequality is again based on the concept of partition inequality and the MIR approach. In fact, the outgoing and incoming demands are analyzed for each node $v \in V$. Since a single backup path protection method is used, similarly to (2.6.5a) and (2.6.5b), the capacity of links in the outgoing (incoming) node v excluding any single link must be sufficient to satisfy the outgoing (incoming) overall demand flow of node v . Therefore, the node partition inequality can be formulated as follows:

$$\sum_{e \in \delta^+(v) \setminus e'} y_e \geq \lceil \frac{h^+(v)}{M} \rceil, \quad v \in V, e' \in \delta^+(v) \quad (2.6.6a)$$

$$\sum_{e \in \delta^-(v) \setminus e'} y_e \geq \lceil \frac{h^-(v)}{M} \rceil, \quad v \in V, e' \in \delta^-(v). \quad (2.6.6b)$$

Finally, a rather obvious cut inequality involves using a solution of problem (2.6.1) provided by the heuristic algorithm as an upper bound of the objective function (2.6.1a).

Results showing the effectiveness of cut inequalities proposed above are presented and discussed in [102]. In turn, heuristic algorithms solving problem (2.6.1) can be found in [102, 103].

2.7 p-Cycle Protection of Anycast Flows

Sections 2.5 and 2.6 focused on path protection methods. Further interesting concepts that can be used to provide network survivability are methods based on ring topology, e.g., bidirectional line switched rings (BLSRs) or unidirectional path-switched rings (UPSRs). The main advantage of ring-based survivability is the fact that rings use a simple switching mechanism which permits very fast restoration. However, the key disadvantage of ring-based methods is the high consumption of spare capacity required to provide survivability. i.e., a redundancy of at least 100% is needed. Moreover, in recent years there has been a trend to shift from ring-based to mesh-based networks, therefore the use of protection rings is on the decline [5, 6, 116, 117].

An interesting survivability approach that combines the relatively short restoration time offered in ring-based protection with efficient usage of network capacity provided in mesh protection is the concept of *p-cycles* proposed in the late 1990s. In particular, a p-cycle can be defined as a preconfigured ring created in a mesh network. The main advantage of the p-cycle concept is that as well as providing protection for all on-cycle links (as in traditional ring-based protection), a p-cycle can also protect *straddling* links that are not on the p-cycle but both their end nodes are included in the p-cycle. This additional protection improves the capacity efficiency of p-cycles over traditional ring-based schemes [6, 117, 118].

The majority of earlier research into p-cycles focused on unicast flows, e.g., [6, 118–129]. Moreover, some papers also considered p-cycle protection of multicast flows, e.g., [130–141]. This section presents a novel p-cycle known as the Anycast-Protecting p-Cycle (APpC) proposed to protect anycast flows in connection-oriented networks [142–145].

The APpC approach is used to protect anycast flows related to streaming services provided in backbone networks. More precisely, a set of streaming servers (data centers) is located in some nodes of the network and each server provides the same signal with the same rate, e.g., IPTV service. End users requesting the streaming service are connected to backbone network nodes by access networks. However, we focus on backbone network optimization only, therefore all users connected to the

same backbone network are aggregated and represented as a single anycast client that must receive the streaming service. In other words, if at least one user connected to a particular backbone network node requests the streaming service, this node must be provided with a connection to a streaming server. Transmission in a failure-state of the network is provided by anycast connections, while protection against single link failures is provided by p-cycles. We start by describing the concept of Anycast-Protecting p-Cycles, followed by formulating and discussing the optimization model and presenting and analyzing numerical results.

2.7.1 Anycast-Protecting p-Cycles

The concept of Anycast-Protecting p-Cycles (APpC) was first proposed in [142] and applied to various optimization problems with anycast flows [143–146].

A classical p-cycle supports the protection of physical links that are on-cycle links or straddling links. Therefore, if all network links are to be protected, the p-cycles must be configured such that each network link is an on-cycle link or a straddling link of at least one p-cycle established in the network [6]. An illustrative example is shown in Fig. 2.3. The network consists of 10 nodes. There are two streaming servers r_1 and r_2 located in nodes a and c , respectively. The anycast client c_1 is placed at node j . A working path (a, e, h, j) (solid bold line) is established to provide a streaming service for client c_1 , which means that client c_1 uses server r_1 . To provide 100% protection of the working path $p_1 = (a, e, h, j)$, two classical p-cycles are established (dotted lines): $q_1 = (a, d, e, b, a)$ and $q_2 = (e, f, i, j, h, g, e)$. More precisely, link (a, e) is a straddling link of p-cycle q_1 , link (e, h) is a straddling link of p-cycle q_2 and link (h, j) is an on-cycle link of p-cycle q_2 .

The authors of [124] propose a new idea of *flow* p-cycles. The key innovation is that protection is provided on the level of flow paths instead of physical links as in the case of classical p-cycles. Accordingly, instead of a single physical link, a

Fig. 2.3 Example of classical p-cycles

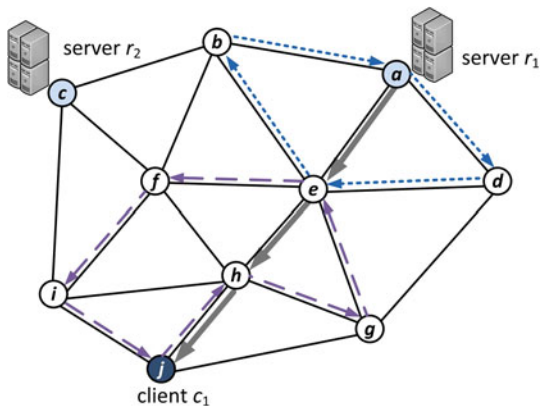
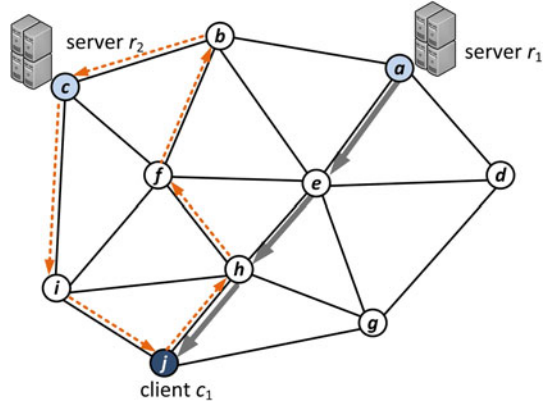


Fig. 2.4 Example of an anycast-protection p-cycles



path segment can be considered to straddle the cycle, i.e., two end nodes of the path segment are included in the cycle to enable protection of the whole path segment. This results in a more efficient allocation of spare capacity and leads to a decrease in the capacity usage, while still providing 100% protection. Moreover, [147] introduces an extension of flow p-cycles known as *Failure-Independent Path-Protection* (FIPP) p-cycles. The main assumption behind FIPP p-cycles is that failure-independent paths are grouped and only one p-cycle is needed to protect each group. This approach results in an even more efficient allocation of spare capacity in comparison to flow p-cycles.

The anycast-protecting p-cycle approach proposed in this section combines the concept of classical and flow p-cycles with additional anycasting properties. We use the fact that each data center (streaming server) provides the same signal. Therefore, if a particular p-cycle includes one streaming server, it can protect a working path of a client connected to another server. This is described using a simple example shown in Fig. 2.4. The general assumptions are the same as in Fig. 2.3. To protect the working path established for client c_1 , only one anycast-protecting p-cycle is required $q_3 = (b, c, i, j, h, f, b)$. It should be stressed that p-cycle q_3 includes streaming server r_2 located at node c . Note that only link (h, j) is protected in a classical way, i.e., as an on-cycle link of p-cycle q_3 . Two other links of the working path p_1 , (a, e) and (e, h) , are protected due to the fact that p-cycle q_3 includes server r_2 and thus if any of these links fail (meaning that client c_1 is disconnected from server r_1 located at node a), client c_1 can still receive the streaming signal using p-cycle q_3 and server r_2 .

2.7.2 Formulation

Since only streaming servers provide transmission to anycast clients, only downstream anycast demands are considered. Moreover, because all streaming servers provide the same signal with the same bit-rate, each demand has the same volume

equal to 1 normalized capacity unit. To model anycast flows in a failure-free state of the network, link-path modeling is applied, and for each anycast demand d there is a set of candidate paths $P(d)$ connecting one of the streaming servers and client node of demand d . Moreover, set Q including candidate p-cycles is given. Each p-cycle $q \in Q$ is described with a set of constants. Firstly, let binary constant β_{eq} denote whether link e belongs to (creates) p-cycle q . Next, binary constant γ_{eq} defines whether link e is protected in a classical way (as an on-cycle link or a straddling link) by a p-cycle q . Finally, constant ω_{eqdp} describes the anycast-protection properties of p-cycles. More specifically, ω_{eqdp} is 1 only if link e belongs to path p realizing demand d and is protected by a p-cycle q in the anycast-protection mode. It should be stressed that constant ω_{eqdp} is activated (set to 1) only if additional protection resulting from anycast-protection p-cycles is used. Consequently, ω_{eqdp} does not duplicate the protection provided in a classical way and represented by constant γ_{eq} .

The routing of working paths is controlled by the binary variable x_{dp} . Link capacity expressed in normalized units is denoted by the integer variable y_e . The selection of p-cycles required to fully protect the network is described by four types of variables. Firstly, binary variable z_{eqdp} denotes whether p-cycle q is used to protect path p realizing demand d in the event of link e failure. Secondly, the binary variable z_{eqd} indicates whether p-cycle q is applied to protect demand d in the event of link e failure. Thirdly, the integer variable z_{eq} represents the number of required copies of p-cycle q in the event of link e failure. Finally, the integer variable z_q is the maximum value over z_{eq} for each network failure, and thus it denotes the total number of required copies of p-cycle q .

CON/A/ND/PCycle/Cost

sets

E	links
D	anycast demands (downstream)
$P(d)$	candidate paths for flows realizing demand d , the candidate path originates at the DC node and terminates at the client node
Q	candidate p-cycles

constants

δ_{edp}	=1, if link e belongs to path p realizing demand d ; 0, otherwise
β_{eq}	=1, if link e belongs to p-cycle q ; 0, otherwise
γ_{eq}	=1, if link e is protected in a classical way (as an on-cycle link or a straddling link) by a p-cycle q ; 0, otherwise
ω_{eqdp}	=1, if link e belongs to path p realizing demand d and it is protected by p-cycle q in the anycast-protection mode, but link e is not protected in the classical mode by p-cycle q ($\gamma_{eq} = 0$); 0, otherwise
ξ_e	unit cost of link e

variables

x_{dp}	=1, if path p is used to realize demand d ; 0, otherwise (binary)
y_e	capacity of link e as the number of normalized units M (integer)
z_{eqdp}	=1, if p-cycle q is used to protect path p realizing demand d in the event of link e failure; 0, otherwise (binary)
z_{eqd}	=1, if p-cycle q is used to protect of demand d in the event of link e failure; 0, otherwise (binary)
z_{eq}	number of required copies of p-cycle q in the event of link e failure (integer)
z_q	total number of required copies of p-cycle q (integer)

objective

$$\text{minimize } F = \sum_{e \in E} \xi_e y_e \quad (2.7.1a)$$

constraints

$$\sum_{p \in P(d)} x_{dp} = 1, \quad d \in D \quad (2.7.1b)$$

$$\sum_{q \in Q} (\gamma_{eq} + \omega_{eqdp}) z_{eqdp} \geq \delta_{edp} x_{dp}, \quad d \in D, p \in P(d), e \in E \quad (2.7.1c)$$

$$z_{eqd} \geq z_{eqdp}, \quad d \in D, p \in P(d), q \in Q, e \in E \quad (2.7.1d)$$

$$z_{eq} = \sum_{d \in D} z_{eqd}, \quad q \in Q, e \in E \quad (2.7.1e)$$

$$z_q \geq z_{eq}, \quad q \in Q, e \in E \quad (2.7.1f)$$

$$\sum_{d \in D} \sum_{p \in P(d)} \delta_{edp} x_{dp} + \sum_{q \in Q} \beta_{eq} y_q \leq y_e, \quad e \in E. \quad (2.7.1g)$$

The goal of the optimization described in (2.7.1a) is to minimize the cost of the capacity required in the network to establish working paths for every anycast demand and to provide 100 % protection using of p-cycles. Equation (2.7.1b) ensures that there is exactly one routing path chosen for each demand. Constraint (2.7.1c) states that if link e is used to realize demand d ($\delta_{edp} = 1$ and $x_{dp} = 1$), then there must be protection against link e failure by at least one p-cycle q , either in a classical way ($\gamma_{eq} = 1$) or in the anycast-protecting mode ($\omega_{eqdp} = 1$). Conditions (2.7.1d) and (2.7.1e) define variables z_{eqd} and z_{eq} , respectively. Inequality (2.7.1f) states that variable z_q is the maximum of z_{eq} variables taking into consideration every single link failure. The last inequality (2.7.1g) is the link capacity constraint. Note that the first term on the left-hand side of (2.7.1g) denotes the working capacity required on link e to realize all anycast demands, while the second term represents the spare capacity needed for p-cycles.

Model (2.7.1) can be modified so that only classical p-cycles are used by removing constant ω_{eqdp} from the left-hand side of constraint (2.7.1c). Furthermore, if routing of anycast demands (working paths) is fixed and values of variables x_{dp} are given in advance as constants, model (2.7.1) can be applied to optimize spare capacity usage only. For more details see [142–146].

Moreover, note that an idea similar to anycast-protection of p-cycles can be applied to protect multicast flows as shown in [132, 133, 146]. Let us recall, that in anycast-protection of p-cycles, alternative data (signal) sources were accessible by using another DC included in a particular p-cycle. Considering multicast flows, alternative signal sources are found within the current multicast tree used for streaming. More specifically, if a p-cycle includes any network node connected to the tree and not affected by the network failure (i.e., a node not disconnected from the root node), such a node can provide the signal to nodes of the multicast tree affected by the failure and disconnected from the root node.

2.7.3 Numerical Results

To obtain optimal results for the CON/A/ND/PCycle/Cost problem (2.7.1), the Gurobi solver [148] was applied. In [142–146] two heuristic methods of solving several optimization problems with anycast-protecting p-cycles were proposed and analyzed: a greedy algorithm Feedback Functional Efficiency Ratio Algorithm (F²ERH) and the Simulated Annealing (SA) approach. However, since the Gurobi solver provided optimal results in a relatively short execution time, here we focus only on the optimal results. The main goal of the numerical result experiments was to compare the performance of anycast-protecting p-cycles and classical p-cycles.

The following four network topologies included in the SNDlib library [149] were tested: cost266 (37 nodes and 114 links), germany50 (50 nodes and 176 links), giul39 (39 nodes and 172 links), and pioro40 (40 nodes and 178 links). Test scenarios (problem instances) were created at random according to several unique parameters: number and location of streaming servers (data centers), number and location of anycast clients, candidate p-cycles. For working flows three candidate paths using the k-shortest path algorithm were generated. We chose this number of candidate paths to find a good trade-off between the overall problem size resulting directly from the number of candidate paths and precision of the model in terms of routing diversity.

In order to generate candidate p-cycles, two algorithms (p-cycle generators) were used: Straddling Link Algorithm (SLA) known as the Straddling Span Algorithm (SSA) [6, 150] and Expand [151]. We used values 5, 10, 15, 20 and 25 of the p-cycle maximum hop limit to generate candidate p-cycles. The reason for limiting the maximum hop limit for p-cycles is as follows. The length of a p-cycle estimates the transmission delay experienced in a particular p-cycle used to protect the working flows. Minimizing the transmission delay is an important Quality of Service (QoS) requirement expected in networks, especially when multimedia or real-time data

is transmitted. Instead of adding an extra constraint to the optimization model, the p-cycle length (number of hops)—and in consequence—transmission delay can be addressed during the phase of generating candidate p-cycles. This approach facilitates solving the optimization model using branch-and-bound algorithms. Moreover, without limiting the p-cycle hop count, some generators yield very high numbers of candidate p-cycles, significantly increasing the time needed to solve the model in an optimal way.

Table 2.2 shows the average percentage gain in the network cost of using anycast-protecting p-cycles (APpC) instead of classical p-cycles (CpC). The results are presented for all tested networks and two cases: spare capacity optimization and joint working and backup capacity optimization. The results are averaged over 75 different tests assuming 20 clients, 4 DCs and the Expand generator with 10 hops limit. It is clear that using classical p-cycles is between 15.2 and 19.3 % more expensive on average than using anycast-protecting p-cycles.

Tables 2.3 and 2.4 show more detailed results using the cost266 network for spare capacity and joint working and spare capacity optimization problems, respectively. Anycast-protecting p-cycles and classical p-cycles are compared in terms of the following performance metrics: average network cost (columns 3–5), average p-cycle length (in hops) selected in the optimization (columns 6–7) and average number of selected p-cycles (columns 8–9). The results are presented separately for seven cases regarding p-cycle generator scenarios (rows): the SLA generator with hop limit 5

Table 2.2 Anycast-protecting p-cycles versus classical p-cycles—average percentage gap of the network cost for 20 clients, 4 DCs and expand generator with 10 hops path limit

Network	Spare capacity (%)	Joint capacity (%)
cost266	18.1	18.0
germany50	18.7	17.1
giul39	16.8	15.2
pioro40	19.3	18.6

Table 2.3 Anycast-protecting p-cycles versus classical p-cycles for spare capacity optimization—various parameters obtained for cost266 network

p-cycles		Av. network cost			Av. p-cycle length		Av. number of p-cycles	
Generator	Hop limit	CpC	APpC	Gap (%)	CpC	APpC	CpC	APpC
SLA	5	606	573	7.4	3.4	3.4	23.2	21.7
SLA	10	492	450	10.8	7.3	7.5	13.7	12.5
Expand	5	591	559	7.9	3.9	4.0	22.7	21.2
Expand	10	536	473	14.8	8.4	8.6	11.6	10.1
Expand	15	501	446	13.2	12.4	12.5	9.5	8.5
Expand	20	493	438	13.5	17.9	18.0	9.0	8.1
Expand	25	493	442	11.8	20.3	20.6	8.9	8.1

Table 2.4 Anycast-protecting p-cycles versus classical p-cycles for joint working and spare capacity optimization—different parameters obtained for cost266 network

p-cycles		Av. network cost			Av. p-cycle length		Av. number of p-cycles	
Generator	Hop limit	CpC	APpC	Gap (%)	CpC	APpC	CpC	APpC
SLA	5	761	689	11.7	3.7	3.4	18.0	22.5
SLA	10	641	576	12.1	8.0	7.5	10.0	12.6
Expand	5	748	674	12.2	4.0	3.7	17.3	21.9
Expand	10	663	585	13.4	9.0	8.7	8.0	10.1
Expand	15	630	565	11.6	13.0	12.9	6.9	8.2
Expand	20	631	568	11.1	18.6	18.4	6.7	7.7
Expand	25	558	505	10.6	21.7	21.5	6.0	6.7

and 10, and the Expand generator with hop limit 5, 10, 15, 20 and 25. Since the SLA algorithm generates relatively short p-cycles, the maximum considered hop count was limited to 10. The results are averaged over 4875 distinct cases different in terms of the anycast client number (2–26), anycast client location, DC number (2–8) and DC location.

The first observation is that these results confirm that using anycast-protecting p-cycles reduces the network cost related to spare and working capacity when compared to classical p-cycles. Tables 2.3 and 2.4 show that the gain of using anycast-protecting p-cycles is lower than that in the results shown in Table 2.2. This is due to the number of anycast client assumed in both cases, i.e., the results in Tables 2.3 and 2.4 are averaged over various number of anycast clients (2–26), while Table 2.2 presents results yielded for 20 anycast clients. Comprehensive analysis of the results clearly reveals that as anycast clients increase, the gap between both types of p-cycles grows. For more details on this issue see [144–146].

The second interesting observation refers to the examination of the p-cycle type and hop limit influence on the results. It is clear that, in general, increasing the hop count limit of p-cycles reduces the network cost for both anycast-protecting p-cycles and classical p-cycles. This is because longer p-cycles protect a large number of working flows. Moreover, for a higher hop count limit, a lower number of p-cycles is required to provide protection for all working flows, making management of the network simpler. However, as mentioned above, longer p-cycles can significantly increase the transmission delay experienced in the event of a network failure.

The average execution time required to solve optimization models using the Gurobi solver was counted in tens of seconds. The key difficulty in solving the models was the memory requirement resulting from the high number of variables and constraints. Accordingly, for larger problem instances the Gurobi solver returned frequently the ‘out of memory’ error and was unable to provide a feasible solution of the problem. For more results and discussions on the application of anycast-protection p-cycles, see [142–146].

2.8 Multi-layer Optimization

All optimization problems presented so far in this book are related to single-layer networks. The concept of *multi-layer* network modeling has been gaining a lot of attention in recent years due to growing demands to provide more precise models that enable efficient optimization and in consequence cost savings in design and operation of computer and communication networks. Many issues related to multi-layer modeling have been addressed in numerous books and papers, e.g., [4, 6, 152–163]. However, multi-layer optimization with anycast flows has only been considered in [49, 164].

It should be stressed that the multi-layer network approach makes it possible to optimize the whole network much more efficiently compared to the single-layer method where each layer is optimized separately, which cannot guarantee the global optimality of the solution. Nevertheless, optimization of multi-layer networks brings additional challenges. In particular, because more network layers are considered, the optimization problem increases; in consequence this triggers the need to develop new heuristic algorithms, since exact solutions given by branch-and-bound and branch-and-cut methods can be obtained for relatively small networks only. A more comprehensive treatment of modeling and optimization of multi-layer networks is given in [4].

2.8.1 Formulation

In this section, we present a two-layer network design problem with simultaneous unicast, multicast and anycast flows realized in the upper layer [49, 164]. The network consists of two layers; upper layer links are denoted by set E and lower layer links are represented as set G . Anycast, multicast and unicast traffic demands included in set D are defined in the upper layer, since it is assumed that DCs are accessed by protocols or technologies implemented in the upper layer of the network. For each demand $d \in D$, a set of candidate structures $P(d)$ is given. Each candidate structure $p \in P(d)$ is defined as a set of upper layer links. The concept of two-layer network modeling assumes that each link $e \in E$ of the upper layer can be realized using paths $q \in Q(e)$ established in the lower layer, i.e., each path $q \in Q(e)$ is defined as a set of lower layer links. In consequence, links $e \in E$ of the upper layer create a virtual topology used to realize traffic demands. Going down the network hierarchy, upper layer links included in set E represent the demand pattern of the lower layer. Note that to formulate a multi-layer model, this approach is repeated going down the network hierarchy for each subsequent layer.

The connection-oriented approach is applied in both layers, i.e., a single path routing is ensured. The link capacity of both layers is allocated in modules; the upper layer capacity uses modules of size M , while the lower layer capacity is allocated with modules of size N . Integer variables y_e and u_g denote the capacity (in modules)

assigned to upper and lower layers, respectively. Binary variable x_{dp} denotes the routing structure selected to realize demand d , while integer variable z_{eq} indicates how many copies of path q are used to realize upper layer link e using lower layer resources. Variable z_{eq} is a non-binary integer, in order to allow capacity y_e assigned to link e to be greater than the size of one capacity module M .

To illustrate this idea, the following example of MPLS over a WDM network is presented. The upper layer uses the MPLS protocol, and upper layer demands included in set D and the upper layer capacity are expressed in Mb/s. In turn, the lower layer is based on the WDM technology with 40 Gb/s per one wavelength. Therefore, the upper layer capacity module is $M = 40$ Gb/s. Continuing the example, the lower layer links $g \in G$ are represented as fibers and it is assumed that the lower layer capacity module is $N = 80$, which means that each fiber can support at most 80 wavelengths [4, 49, 164].

CON/AMU/MLN/Cost

sets

E	links of upper layer
D	demands (anycast, multicast, unicast)
D^{DS}	anycast downstream demands
$P(d)$	candidate structures for flows realizing demand d . If d is a unicast demand, the candidate structure is a path connecting end nodes of the demand. If d is an anycast upstream demand, the candidate structure is a path connecting the client node and DC node. If d is a downstream demand, candidate structure is a path connecting the DC node and the client node. If d is a multicast demand, the candidate structure is a tree that includes the root node and all receivers of the demand
$Q(e)$	candidate paths in the lower layer for flows realizing the upper layer link e
G	links of lower layer

constants

δ_{edp}	=1, if link e belongs to structure p realizing demand d ; 0, otherwise
h_d	volume of demand d
γ_{geq}	=1, if link g of lower layer belongs to path q realizing link e of upper layer; 0, otherwise
ξ_e	unit cost of link e
κ_g	unit cost of link g
M	size of the upper layer link capacity module
N	size of the lower layer link capacity module
$\tau(d)$	index of a demand associated with demand d . If d is a downstream demand, then $\tau(d)$ must be an upstream demand and vice versa
$s(p)$	origin node of path p
$t(p)$	destination node of path p

variables

- x_{dp} = 1, if structure p is used to realize demand d ; 0, otherwise (binary)
 y_e capacity of upper layer link e as the number of capacity modules (integer)
 z_{eq} number of path q instances used to realize the capacity of link e (integer)
 u_g capacity of lower layer link g as the number of capacity modules (integer)

objective

$$\text{minimize } F = \sum_{e \in E} \xi_e y_e + \sum_{g \in G} \kappa_g u_g \quad (2.8.1a)$$

constraints

$$\sum_{p \in P(d)} x_{dp} = 1, \quad d \in D \quad (2.8.1b)$$

$$\sum_{d \in D} \sum_{p \in P(d)} \delta_{edp} x_{dp} h_d \leq M y_e, \quad e \in E \quad (2.8.1c)$$

$$\sum_{p \in P(d)} x_{dp} s(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} t(p), \quad d \in D^{DS} \quad (2.8.1d)$$

$$\sum_{q \in Q(e)} z_{eq} = y_e, \quad e \in E \quad (2.8.1e)$$

$$\sum_{e \in E} \sum_{q \in Q(e)} \gamma_{geq} z_{eq} h_d \leq N u_g, \quad g \in G. \quad (2.8.1f)$$

The objective function (2.8.1a) aims to minimize the cost of capacity assigned in both network layers. Constraints (2.8.1b)–(2.8.1d) are the same as in the single layer network design problem formulated in Sect. 2.3.1. Equality (2.8.1e) ensures that each upper layer link is realized by a set of lower layer paths. Condition (2.8.1f) states that flow in each lower layer link cannot exceed its capacity.

Due to the complexity of multi-layer models, heuristic algorithms are required to solve larger problem instances. One approach is to tackle the optimization in all network layers jointly. However, since routing and capacity decision variables of both layers are bound to each other, this strategy may not be efficient. Another approach is to optimize each network layer in a separate phase, i.e., the problem in the upper layer is solved first, and the lower layer is optimized next. Algorithms developed for single layer problems can be used, i.e., methods proposed in Sects. 2.2.2 and 2.3.2. If the algorithms used to optimize each layer are relatively fast, the procedure can be repeated many times and in each subsequent iteration, information obtained in the previous iteration can be used to improve the performance of the overall optimization process.

References

1. Perros, H.: *Connection-Oriented Networks: SONET/SDH, ATM, MPLS and Optical Networks*. Wiley, New York (2005)
2. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol label switching architecture, RFC3013, Internet Engineering Task Force (2001)
3. Kasim, A.: *Delivering Carrier Ethernet: Extending Ethernet Beyond the LAN*, 1st edn. McGraw-Hill Inc, New York (2008)
4. Pioro, M., Medhi, D.: *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann, San Francisco (2004)
5. Vasseur, J., Pickavet, M., Demeester, P.: *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Morgan Kaufmann Publishers Inc., San Francisco (2004)
6. Grover, Wayne D.: *Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*. Prentice Hall PTR, Upper Saddle River (2003)
7. Habib, M.F., Tornatore, M., De Leenheer, M., Dikbiyik, F., Mukherjee, B.: Design of disaster-resilient optical datacenter networks. *J. Lightwave Technol.* **30**(16), 2563–2573 (2012)
8. Ramaswami, R., Sivarajan, K., Sasaki, G.: *Optical Networks: A Practical Perspective*, 3rd edn. Morgan Kaufmann Publishers Inc, San Francisco (2009)
9. Simmons, J.: *Optical Network Design and Planning*, 2nd edn. Springer (2014)
10. IEEE. 802.1ah, Provider backbone bridging (2008)
11. IEEE. 802.1ay, Provider backbone bridging traffic engineering (2009)
12. Allan, D., Bragg, N., McGuire, A., Reid, A.: Ethernet as carrier transport infrastructure. *IEEE Commun. Mag.* **44**(2), 95–101 (2006)
13. Takacs, A., Green, H., Tremblay, B.: GMPLS controlled ethernet: an emerging packet-oriented transport technology. *IEEE Commun. Mag.* **46**(9), 118–124 (2008)
14. Walkowiak, K.: Anycasting in connection-oriented computer networks: models, algorithms and results. *Int. J. Appl. Math. Comput. Sci.* **20**(1), 207–220 (2010)
15. Fratta, L., Gerla, M., Kleinrock, L.: The flow deviation method: an approach to store-and-forward communication network design. *Networks* **3**(2), 97–133 (1973)
16. Bienstock, D., Raskina, O.: Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem. *Math. Program.* **91**(3), 479–492 (2002)
17. Burns, J.E., Ott, T., Krzesinski, A.E., Muller, K.: Path selection and bandwidth allocation in MPLS networks. *Perform. Eval.* **52**(2–3), 133–152 (2003)
18. Duhamel, Ch., Mahey, P.: Multicommodity flow problems with a bounded number of paths: a flow deviation approach. *Networks* **49**(1), 80–89 (2007)
19. Fratta, L., Gerla, M., Kleinrock, L.: Flow deviation: 40 years of incremental flows for packets, waves, cars and tunnels. *Comput. Netw.* **66**(0), 18–31 (2014). Leonard Kleinrock Tribute Issue: A Collection of Papers by his Students
20. Gerla, M., Kleinrock, L.: On the topological design of distributed computer networks. *IEEE Trans. Commun.* **25**(1), 48–60 (1977)
21. LeBlanc, L., Chifflet, J., Mahey, P.: Packet routing in telecommunication networks with path and flow restrictions. *INFORMS J. Comput.* **11**(2), 188–197 (1999)
22. Murakami, K., Kim, H.S.: Virtual path routing for survivable ATM networks. *IEEE/ACM Trans. Netw.* **4**(1), 22–39 (1996)
23. Ng, T., Hoang, D.B.: Joint optimization of capacity and flow assignment in a packet-switched communications network. *IEEE Trans. Commun.* **35**(2), 202–209 (1987)
24. Ouorou, A., Mahey, P.: Vial, J-Ph: A survey of algorithms for convex multicommodity flow problems. *Manage. Sci.* **46**(1), 126–147 (2000)
25. Walkowiak, K.: A new method of primary routes selection for local restoration. In: Mitrou, N., Kontovasilis, K., Rouskas, G.N., Iliadis, I., Merakos, L. (eds.) *Networking 2004. Lecture Notes in Computer Science*, vol. 3042, pp. 1024–1035. Springer, Berlin (2004)

26. Walkowiak, K.: Heuristic algorithm for anycast flow assignment in connection-oriented networks. In: Sunderam, V.S., van Albada, G., Sloot, P.M.A., Dongarra, J. (eds.) *Computational Science (ICCS 2005)*. Lecture Notes in Computer Science, vol. 3516, pp. 1092–1095. Springer, Berlin (2005)
27. Walkowiak, K.: A flow deviation algorithm for joint optimization of unicast and anycast flows in connection-oriented networks. *Computational Science and Its Applications (ICCSA 2008)*. Lecture Notes in Computer Science, vol. 5073, pp. 797–807. Springer, Berlin (2008)
28. Dias, R.A., Camponogara, E., Farines, J., Willrich, R., Campestrini, A.: Implementing traffic engineering in MPLS-based IP networks with Lagrangean relaxation. In: *Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC 2003)*, pp. 373–378, June 2003
29. Gavish, B., Hantler, S.: An algorithm for optimal route selection in SNA networks. *IEEE Trans. Commun.* **31**(10), 1154–1161 (1983)
30. Gavish, B., Neuman, I.: A system for routing and capacity assignment in computer communication networks. *IEEE Trans. Commun.* **37**(4), 360–366 (1989)
31. Gavish, B., Neuman, I.: Routing in a network with unreliable components. *IEEE Trans. Commun.* **40**(7), 1248–1258 (1992)
32. Holmberg, K., Yuan, D.: A Lagrangean approach to network design problems. *Int. Trans. Oper. Res.* **5**(6), 529–539 (1998)
33. Holmberg, K., Yuan, D.: A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Oper. Res.* **48**(3), 461–481 (2000)
34. Retvdri, G., Biro, J.J., Cinkler, T.: A novel lagrangian-relaxation to the minimum cost multi-commodity flow problem and its application to OSPF traffic engineering. In: *Proceedings of the Ninth International Symposium on Computers and Communications (ISCC 2004)*, vol. 2, pp. 957–962, June 2004
35. Walkowiak, K.: Lagrangean heuristic for anycast flow assignment in connection-oriented networks. *Computational Science (ICCS 2006)*. Lecture Notes in Computer Science, vol. 3991, pp. 626–633. Springer, Berlin (2006)
36. Walkowiak, K.: Lagrangean heuristic for primary routes assignment in survivable connection-oriented networks. *Comput. Opt. Appl.* **40**(2), 119–141 (2008)
37. Held, M., Wolfe, P., Crowder, H.: Validation of subgradient optimization. *Math. Program.* **6**(1), 62–88 (1974)
38. Coley, D.: *An Introduction to Genetic Algorithms for Scientists and Engineers*. World scientific, Singapore (1999)
39. Corne, D., Oates, M., Smith, G. (eds.): *Telecommunications Optimization: Heuristic and Adaptive Techniques*. Wiley, Chichester (2000)
40. Donoso, Y., Fabregat, R.: *Multi-Objective Optimization in Computer Networks Using Meta-heuristics*. Auerbach Publications, Boston (2007)
41. Goldberg, David E.: *Genetic Algorithms in Search Optimization and Machine Learning*, 1st edn. Addison-Wesley Longman Publishing Co., Inc, Boston (1989)
42. Yang, X.: *Nature-Inspired Optimization Algorithms*. Elsevier (2014)
43. Arabas, J., Kozdrowski, S.: Applying an evolutionary algorithm to telecommunication network design. *IEEE Trans. Evol. Comput.* **5**(4), 309–322 (2001)
44. Banerjee, N., Mehta, V., Pandey, S.: A genetic algorithm approach for solving the routing and wavelength assignment problem in WDM networks. In: *Proceedings of the 3rd IEEE/IEE International Conference on Networking (ICN 2004)*, pp.70–78 (2004)
45. Koppen, M., Yoshida, K., Tsuru, M., Oie, Y.: Evolutionary routing-path selection in congested communication networks. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC 2009)*, pp. 2155–2160, October 2009
46. Pitsillides, A., Stylianou, G., Pattichis, C.S., Sekercioglu, A., Vasilakos, A.: Aggregated bandwidth allocation: investigation of performance of classical constrained and genetic algorithm based optimisation techniques. *Comput. Commun.* **25**(16), 1443–1453 (2002)
47. Przewozniczek, M., Walkowiak, K.: Evolutionary algorithm for congestion problem in connection-oriented networks. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee,

- H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) Computational Science and Its Applications ICCSA. Lecture Notes in Computer Science, vol. 3483, pp. 802–811. Springer, Berlin (2005)
48. Rubio-Largo, Á., Vega-Rodríguez, M.: Applying MOEAs to solve the static routing and wavelength assignment problem in optical WDM networks. *Eng. Appl. Artif. Intell.* **26**(5U6), 1602–1619 (2013)
 49. Walkowiak, K.: Modeling and Optimization of Computer Networks. Wroclaw University of Technology, Wroclaw (2011)
 50. Bhanja, U., Mahapatra, S., Roy, R.: An evolutionary programming algorithm for survivable routing and wavelength assignment in transparent optical networks. *Inf. Sci.* **222**, 634–647 (2013). Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems
 51. Bhanja, U., Mahapatra, S.: A metaheuristic approach for optical network optimization problems. *Appl. Soft Comput.* **13**(2), 981–997 (2013)
 52. El-Alfy, E., Mujahid, S., Selim, S.: A pareto-based hybrid multiobjective evolutionary approach for constrained multipath traffic engineering optimization in MPLS/GMPLS networks. *J. Netw. Comput. Appl.* **36**(4), 1196–1207 (2013)
 53. Li, K., Zhou, X., Zhang, W.: An efficient anycast routing algorithm for load-balancing based on evolutionary algorithm. In: Proceedings of the 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT 2010), pp. 48–54, October 2010
 54. Przewozniczek, M., Walkowiak, K.: uasi-hierarchical evolutionary algorithm for flow optimization in survivable MPLS networks. Computational Science and Its Applications (ICCSA 2007). Lecture Notes in Computer Science, vol. 4707, pp. 330–342. Springer, Berlin (2007)
 55. Glover, F.: Tabu search fundamentals and uses. Technical report, University of Colorado (1995)
 56. Kirkpatrick Jr., S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science* **220**, 671–680 (1983)
 57. Resende, M.: Greedy randomized adaptive search procedures (grasp). *Encycl. optim.* **2**, 373–382 (2001)
 58. Talbi, El.G.: Metaheuristics - From Design to Implementation. Wiley, New York (2009)
 59. Hyytia, E.: Heuristic algorithms for the generalized routing and wavelength assignment problem. In: Proceedings of the 17th Nordic Teletraffic Seminar(NTS-17), pp. 373–386 (2004)
 60. Shen, Jian, Fuyong, Xu, Zheng, Peng: A tabu search algorithm for the routing and capacity assignment problem in computer networks. *Comput. Oper. Res.* **32**(11), 2785–2800 (2005)
 61. Bakiras, S., Loukopoulous, T.: Combining replica placement and caching techniques in content distribution networks. *Comput. Commun.* **28**(9), 1062–1073 (2005)
 62. Bektas, T., Oguz, O., Ouveysi, I.: Designing cost-effective content distribution networks. *Comput. Oper. Res.* **34**(8), 2436–2449 (2007)
 63. Krishnan, P., Raz, D., Shavitt, Y.: The cache location problem. *IEEE/ACM Trans. Netw.* **8**(5), 568–582 (2000)
 64. Li, B., Golin, M., Italiano, G., Deng, X., Sohrawy, K.: On the optimal placement of web proxies in the internet. In: Proceedings of Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 1999), pp. 1282–1290 (1999)
 65. Markowski, M., Kasprzak, A.: The web replica allocation and topology assignment problem in wide area networks: algorithms and computational results. Computational Science and Its Applications (ICCSA 2005). Lecture Notes in Computer Science, vol. 3483, pp. 772–781. Springer, Berlin (2005)
 66. Markowski, M., Kasprzak, A.: The three-criteria servers replication and topology assignment problem in wide area networks. Computational Science and Its Applications (ICCSA 2006). Lecture Notes in Computer Science, vol. 3982, pp. 1119–1128. Springer, Berlin (2006)
 67. Markowski, M., Kasprzak, A.: An exact algorithm for the servers allocation, capacity and flow assignment problem with cost criterion and delay constraint in wide area networks. Computational Science (ICCS 2007). Lecture Notes in Computer Science, vol. 4487, pp. 442–445. Springer, Berlin (2007)

68. Qiu, L., Padmanabhan, V., Voelker, G.: On the placement of web server replicas. In: Proceedings of Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 2001), pp. 1587–1596 (2001)
69. Thouin, F., Coates, M.: Video-on-demand server selection and placement. In: Mason, L., Drwiega, T., Yan, J. (eds.) *Managing Traffic Performance in Converged Networks*. Lecture Notes in Computer Science, vol. 4516, pp. 18–29. Springer, Berlin (2007)
70. Walkowiak, K., Rak, J.: Simultaneous optimization of unicast and anycast flows and replica location in survivable optical networks. *Telecommun. Syst.* **52**(2), 1043–1055 (2013)
71. Wang, Jia: A survey of web caching schemes for the internet. *CM SIGCOMM Comput. Commun. Rev.* **29**(5), 36–46 (1999)
72. Wang, y., Li, Z., Tyson, G., Uhlig, S., Xie, G.: Optimal cache allocation for content-centric networking. In: Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP 2013), pp. 1–10, October 2013
73. Woeginger, G.: Monge strikes again: optimal placement of web proxies in the internet. *Oper. Res. Lett.* **27**(3), 93–96 (2000)
74. Baev, I., Rajaraman, R.: Approximation algorithms for data placement in arbitrary networks. In: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '01. Society for Industrial and Applied Mathematics pp. 661–670, Philadelphia (2001)
75. Fang, W., Wang, Z., Lloret, J., Zhang, D., Yang, Z.: Optimising data placement and traffic routing for energy saving in backbone networks. *Trans. Emerg. Telecommun.* **25**(9), 914–925 (2014)
76. Araldo, A., Mangili, M., Martignon, F., Rossi, D.: Cost-aware caching: optimizing cache provisioning and object placement in ICN. In: Proceedings of the IEEE Global Communications Conference (GLOBECOM 2014), pp. 1108–1113, December 2014
77. Bektas, T., Cordeau, J., Erkut, E., Laporte, G.: Exact algorithms for the joint object placement and request routing problem in content distribution networks. *Comput. Oper. Res.* **35**(12), 3860–3884 (2008). Part Special Issue: Telecommunications Network Engineering
78. Cidon, I., Kuten, S., Soffer, R.: Optimal allocation of electronic content. In: Proceedings of Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 2001), vol. 3, pp. 1773–1780 (2001)
79. Kangasharju, J., Roberts, J., Ross, K.: Object replication strategies in content distribution networks. *Comput. Commun.* **25**(4), 376–383 (2002)
80. Laoutaris, N., Zissimopoulos, V., Stavrakakis, I.: Joint object placement and node dimensioning for internet content distribution. *Inf. Process. Lett.* **89**(6), 273–279 (2004)
81. Laoutaris, N., Zissimopoulos, V., Stavrakakis, I.: On the optimization of storage capacity allocation for content distribution. *Comput. Netw.* **47**(3), 409–428 (2005)
82. Abhari, A., Soraya, M.: Workload generation for youtube. *Multimed. Tools. Appl.* **46**(1), 91–118 (2010)
83. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web caching and zipf-like distributions: evidence and implications. In: Proceedings of Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 1999), vol. 1, pp. 126–134, March 1999
84. Cho, K., Lee, M., Park, K., Kwon, T.T., Choi, Y., Pack, S.: Wave: popularity-based and collaborative in-network caching for content-oriented networks. In: Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS 2012), pp. 316–321, March 2012
85. Gill, P., Arlitt, M., Li, Z., Mahanti, A.: Youtube traffic characterization: a view from the edge. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC '07, New York, pp. 15–28. ACM (2007)
86. Hasslinger, G., Hohlfeld, O.: Efficiency of caches for content distribution on the internet. In: Proceedings of the 22nd International Teletraffic Congress (ITC 2010), pp. 1–8, September 2010
87. Janaszka, T., Bursztynowski, D., Dzida, M.: On popularity-based load balancing in content networks. In: Proceedings of the 24th International Teletraffic Congress (ITC 2012), pp. 1–8, September 2012

88. Walkowiak, K.: A unified approach to survivability of connection-oriented networks. *Computer and Information Sciences (ISCIS 2005)*. Lecture Notes in Computer Science, vol. 3733, pp. 3–12. Springer, Berlin (2005)
89. Bui, M., Jaumard, B., Develder, C.: Anycast end-to-end resilience for cloud services over virtual optical networks. In: *Proceedings of the 15th International Conference on Transparent Optical Networks (ICTON 2013)*, pp. 1–7, June 2013
90. Buysse, J., De Leenheer, M., Develder, C., Dhoedt, B.: Exploiting relocation to reduce network dimensions of resilient optical grids. In: *Proceedings of the 7th International Workshop on Design of Reliable Communication Networks (DRCN 2009)*, pp. 100–106, October 2009
91. Buysse, J., De Leenheer, M., Dhoedt, B., Develder, C.: On the impact of relocation on network dimensions in resilient optical grids. In: *Proceedings of the 14th Conference on Optical Network Design and Modeling (ONDM 2010)*, pp. 1–6, February 2010
92. Develder, C., Buysse, J., De Leenheer, M., Jaumard, B., Dhoedt, B.: Resilient network dimensioning for optical grid/clouds using relocation. In: *Proceedings of the IEEE International Conference on Communications (ICC 2012)*, pp. 6262–6267, June 2012
93. Develder, C., Buysse, J., Shaikh, A., Jaumard, B., De Leenheer, M., Dhoedt, B.: Survivable optical grid dimensioning: Anycast routing with server and network failure protection. In: *Proceedings of the IEEE International Conference on Communications (ICC 2011)*, pp. 1–5, June 2011
94. Develder, C., Buysse, J., Dhoedt, B., Jaumard, B.: Joint dimensioning of server and network infrastructure for resilient optical grids/clouds. *IEEE/ACM Trans. Netw.* **22**(5), 1591–1606 (2014)
95. Goscien, R., Walkowiak, K., Klinkowski, M.: Gains of anycast demand relocation in survivable elastic optical networks. In: *Proceedings of the 6th International Workshop on Reliable Networks Design and Modeling (RNDM 2014)*, pp. 109–115, November 2014
96. Jaumard, B., Buysse, J., Shaikh, A., De Leenheer, M., Develder, C.: Column generation for dimensioning resilient optical grid networks with relocation. In: *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2010)*, pp. 1–6, December 2010
97. Shaikh, A., Buysse, J., Jaumard, B., Develder, C.: Anycast routing for survivable optical grids: Scalable solution methods and the impact of relocation. *IEEE/OSA J. Opt. Commun. Netw.* **3**(9), 767–779 (2011)
98. Walkowiak, K., Rak, J.: Joint optimization of anycast and unicast flows in survivable optical networks. In: *Proceedings of the 14th International Telecommunications Network Strategy and Planning Symposium (NETWORKS 2010)*, pp. 1–6, September 2010
99. Walkowiak, K., Rak, J.: Shared backup path protection for anycast and unicast flows using the node-link notation. In: *Proceedings of the 2011 IEEE International Conference on Communications (ICC 2011)*, pp. 1–6, June 2011
100. IBM. ILOG CPLEX optimizer. <http://www.ibm.com>
101. Gladysz, J., Walkowiak, K.: Modeling of survivable network design problems with simultaneous unicast and anycast flows. In: *Proceedings of the 2nd International Logistics and Industrial Informatics (LINDI 2009)*, pp. 1–6, September 2009
102. Gladysz, J., Walkowiak, K.: Optimization of survivable networks with simultaneous unicast and anycast flows. In: *Proceedings of the International Conference on Ultra Modern Telecommunications Workshops (ICUMT 2009)*, pp. 1–6, October 2009
103. Gladysz, J., Walkowiak, K.: Tabu search algorithm for survivable network design problem with simultaneous unicast and anycast flows. *Int. J. Electr. Telecommun.* **56**(1), 39–45 (2010)
104. Mitchell, J.: Branch-and-cut methods for combinatorial optimization problems. In: Resende, M., Pardalos, P. (eds.) *Handbook of Applied Optimization*. Oxford University Press, Oxford (2002)
105. Barnhart, C., Hane, Ch., Vance, P.: Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.* **48**(2), 318–326 (2000)
106. Bienstock, D., Muratore, G.: Strong inequalities for capacitated survivable network design problems. *Math. Program.* **89**(1), 127–147 (2000)

107. Caccetta, L., Hill, S.P.: A branch and cut method for the degree-constrained minimum spanning tree problem. *Networks* **37**(2), 74–83 (2001)
108. Gunluk, O.: A branch-and-cut algorithm for capacitated network design problems. *Math. Program.* **86**, 17–39 (1998)
109. Klinkowski, M., Pioro, M., Zotkiewicz, M., Ruiz, M., Velasco, L.: Valid inequalities for the routing and spectrum allocation problem in elastic optical networks. In: *Proceedings of the 16th International Conference on Transparent Optical Networks (ICTON 2014)*, pp. 1–5, July 2014
110. Koster, A., Kutschka, M., Raack, Ch.: Robust network design: formulations, valid inequalities, and computations. *Networks* **61**(2), 128–149 (2013)
111. Marchand, H., Wolsey, L.: Aggregation and mixed integer rounding to solve MIPs. *Oper. Res.* **49**(3), 363–371 (2001)
112. Minoux, M.: Discrete cost multicommodity network optimization problems and exact solution methods. *Ann. Oper. Res.* **106**(1–4), 19–46 (2001)
113. Ortega, F., Wolsey, L.: A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks* **41**(3), 143–158 (2003)
114. Raack, Ch., Koster, A., Orlowski, S., Wessäly, R.: On cut-based inequalities for capacitated network design polyhedra. *Networks* **57**(2), 141–156 (2011)
115. Tomaszewski, A., Pioro, M., Dzida, M., Mycek, M., Zagodzón, M.: Valid inequalities for a shortest-path routing optimization problem. In: *Proceedings of the 3rd International Network Optimization Conference (INOC 2007)* (2007)
116. Chow, T.Y., Chudak, F., Ffrench, A.M.: Fast optical layer mesh protection using pre-cross-connected trails. *IEEE/ACM Trans. Netw.* **12**(3), 539–548 (2004)
117. Stamatelakis, D., Grover, W.D.: IP layer restoration and network planning based on virtual protection cycles. *IEEE J. Sel. Areas Commun.* **18**(10), 1938–1949 (2000)
118. Stamatelakis, D., Grover, W.D.: Theoretical underpinnings for the efficiency of restorable networks using preconfigured cycles (‘p-cycles’). *IEEE Trans. Commun.* **48**(8), 1262–1265 (2000)
119. Gruber, C.G.: Resilient networks with non-simple p-cycles. In: *Proceedings of the 10th International Conference on Telecommunications, ICT 2003*, vol. 2, pp. 1027–1032, February 2003
120. Li, W., Doucette, J., Zuo, M.: p-cycle network design for specified minimum dual-failure restorability. In: *Proceedings of the IEEE International Conference on Communications, ICC '07*, pp. 2204–2210, June 2007
121. Mukherjee, D.S., Assi, C., Agarwal, A.: Alternate strategies for dual failure restoration using p-cycles. In: *IEEE International Conference on Communications, ICC '06*, vol. 6, pp. 2477–2482, June 2006
122. Oliveira, H.M.N.S., da Fonseca, N.L.S.: Algorithm for FIPP p-cycle path protection in flexgrid networks. In: *Proceedings of the Global Communications Conference (GLOBECOM)*, pp. 1278–1283. IEEE, December 2014
123. Rocha, C., Jaumard, B., Bougue, P.: Directed vs. undirected p-cycles and FIPP p-cycles. In: *Proceedings of the International Network Optimization Conference, INOC 2009* (2009)
124. Shen, G., Grover, W.D.: Extending the p-cycle concept to path segment protection for span and node failure recovery. *IEEE J. Sel. Areas Commun.* **21**(8), 1306–1319 (2003)
125. Szigeti, J., Cinkler, T.: Evaluation and estimation of the availability of p-cycle protected connections. *Telecommun. Syst.* **52**(2), 767–782 (2013)
126. Wei, Y., Xu, K., Jiang, Y., Zhao, H., Shen, G.: Optimal design for p-cycle-protected elastic optical networks. *Photonic Netw. Commun.* **29**, 1–12 (2015)
127. Y. Wei, K. Xu, H. Zhao, and G. Shen. Applying p-cycle technique to elastic optical networks. In *Proceedings of the 2014 International Conference on Optical Network Design and Modeling (ONDM 2014)*, pages 1–6, May 2014
128. Wu, J., Liu, Y., Yu, C., Wu, Y.: Survivable routing and spectrum allocation algorithm based on p-cycle protection in elastic optical networks. *Optik - Int. J. Light. Electron Opt.* **125**(16), 4446–4451 (2014)

129. Zhang, F., Zhong, W.: A novel path-protecting p-cycle heuristic algorithm. In: Proceedings of the International Conference on Transparent Optical Networks (ICTON 2006), vol. 3, pp. 203–206, June 2006
130. Feng, T., Ruan, L., Zhang, W.: Intelligent p-cycle protection for dynamic multicast sessions in WDM networks. *IEEE/OSA J. Opt. Commun. Netw.* **2**(7), 389–399 (2010)
131. Frikha, A., Cousin, B., Lahoud, S.: Extending node protection concept of p-cycles for an efficient resource utilization in multicast traffic. In: Proceedings of the 2011 IEEE 36th Conference on Local Computer Networks (LCN), pp. 175–178, October 2011
132. Smutnicki, A., Walkowiak, K.: p-cycle based multicast protection - a new ILP formulation. In: Proceedings of the 2nd Baltic Congress on Future Internet Communications (BCFIC 2012), pp. 268–274, April 2012
133. Smutnicki, A., Walkowiak, K.: A new approach to optimization of p-cycle protected multicast optical networks. In: Proceedings of the 5th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT 2013), pp. 74–81, September 2013
134. Zhang, F., Zhong, W.: Applying p-cycles in dynamic provisioning of survivable multicast sessions in optical WDM networks. In: Proceedings of the 2007 Conference on Optical Fiber Communication and the National Fiber Optic Engineers Conference (OFC/NFOEC 2007), pp. 1–3, March 2007
135. Zhang, F., Zhong, W.: Optimized design of node-and-link protecting p-cycle with restorability constraints for optical multicast traffic protection. In: Proceedings of the 14th OptoElectronics and Communications Conference (OECC 2009), pp. 1–2, July 2009
136. Zhang, F., Zhong, W.: p-cycle based tree protection of optical multicast traffic for combined link and node failure recovery in WDM mesh networks. *IEEE Commun. Lett.* **13**(1), 40–42 (2009)
137. Zhang, F., Zhong, W.: Performance evaluation of optical multicast protection approaches for combined node and link failure recovery. *J. Lightwave Technol.* **27**(18), 4017–4025 (2009)
138. Zhang, F., Zhong, W.: Extending p-cycles to source failure recovery for optical multicast media traffic. *IEEE/OSA J. Opt. Commun. Netw.* **2**(10), 831–840 (2010)
139. Zhong, W., Zhang, F.: An overview of p-cycle based optical multicast protection approaches in mesh wdm networks. *Opt. Switch. Netw.* **8**(4), 259–274 (2011). Optical network architectures and management
140. Zhong, W., Zhang, F.: p-cycle based optical multicast protection approaches for combined node and link failure recovery. In: Proceedings of the Asia Communications and Photonics Conference and Exhibition (ACP 2010), pp. 367–368, December 2010
141. Zhong, W., Zhang, F.: Source failure recovery for optical multicast traffic in WDM networks. In: Proceedings of the 13th International Conference on Transparent Optical Networks (ICTON 2011), pp. 1–4, June 2011
142. Smutnicki, A., Walkowiak, K.: Optimization of p-cycles for survivable anycasting streaming. In: Proceedings of the 7th International Workshop on Design of Reliable Communication Networks (DRCN 2009), pp. 227–234, October 2009
143. Smutnicki, A., Walkowiak, K.: Optimal results for anycast-protecting p-cycles problem. In: Proceedings of the 2010 International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT 2010), pp. 511–517, October 2010
144. Smutnicki, A., Walkowiak, K.: Joint working and spare capacity assignment for anycast streaming in survivable networks protected by p-cycles. In: Proceedings of the 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT 2011), pp. 1–6, October 2011
145. Smutnicki, A., Walkowiak, K.: A heuristic approach to working and spare capacity optimization for survivable anycast streaming protected by p-cycles. *Telecommun. Syst.* **56**(1), 141–156 (2014)
146. Smutnicki, A.: Algorithms for flow and capacity assignment in p-cycle protected computer networks (in Polish). Ph.D. thesis, Wrocław University of Technology, Department of Systems and Computer Networks, Report PRE008, 2012. Supervisor: K. Walkowiak

147. Kodian, A., Grover, W.D., Doucette, J.: A disjoint route-sets approach to design of path-protecting p-cycle networks. In: Proceedings. 5th International Workshop on Design of Reliable Communication Networks, (DRCN 2005), p. 8, October 2005
148. Inc., Gurobi Optimization. Gurobi optimizer reference manual
149. Orłowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0 – Survivable network design library. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, April 2007. <http://sndlib.zib.de>
150. Zhang, H., Yang, O.: Finding protection cycles in DWDM networks. In: Proceedings of the IEEE International Conference on Communications (ICC 2002), vol. 5, pp. 2756–2760 (2002)
151. Doucette, J., He, D., Grover, W.D., Yang, O.: Algorithmic approaches for efficient enumeration of candidate p-cycles and capacitated p-cycle network design. In: Proceedings of the Fourth International Workshop on Design of Reliable Communication Networks, DRCN 2003), pp. 212–220, October 2003
152. Belotti, P., Capone, A., Carello, G., Malucelli, F.: Multi-layer MPLS network design: the impact of statistical multiplexing. *Comput. Netw.* **52**(6), 1291–1307 (2008)
153. Capone, A., Carello, G., Matera, R.: Multi-layer network design with multicast traffic and statistical multiplexing. In: Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2007), pp. 2565–2570, November 2007
154. Gerstel, O., Filsfils, C., Telkamp, T., Gunkel, M., Horneffer, M., Lopez, V., Mayoral, A.: Multi-layer capacity planning for IP-optical networks. *IEEE Commun. Mag.* **52**(1), 44–51 (2014)
155. Katib, I., Medhi, D.: A network optimization model for multi-layer IP/MPLS over OTN/DWDM networks. In: Nunzi, Giorgio, Scoglio, Caterina, Li, Xing (eds.) *IP Operations and Management*. Lecture Notes in Computer Science, vol. 5843, pp. 180–185. Springer, Berlin (2009)
156. Katib, I., Medhi, D.: Optimizing node capacity in multilayer networks. *IEEE Commun. Lett.* **15**(5), 581–583 (2011)
157. Katib, I., Medhi, D.: IP/MPLS-over-OTN-over-DWDM multilayer networks: an integrated three-layer capacity optimization model, a heuristic, and a study. *IEEE Trans. Netw. Serv. Manage.* **9**(3), 240–253 (2012)
158. Liu, Y., Tipper, D., Vajanapoom, K.: Spare capacity allocation in two-layer networks. *IEEE J. Sel. Areas Commun.* **25**(5), 974–986 (2007)
159. Mikoshi, T., Takenaka, T., Sugiyama, R., Masuda, A., Shiimoto, K., Hiramatsu, A.: High-speed calculation method for large-scale multi-layer network design problem. In: Proceedings of the XVth International Telecommunications Network Strategy and Planning Symposium (NETWORKS 2012), pp. 1–6, October 2012
160. Pedrola, O., Ruiz, M., Velasco, L., Careglio, D., de Dios, O.G., Comellas, J.: A grasp with path-relinking heuristic for the survivable IP/MPLS-over-WSN multi-layer network optimization problem. *Comput. Oper. Res.* **40**(12), 3174–3187 (2013)
161. Ruiz, M., Pedrola, O., Velasco, L., Careglio, D., Fernal-Palacios, J., Junyent, G.: Survivable IP/MPLS-over-WSN multilayer network optimization. *IEEE/OSA J. Opt. Commun. Netw.* **3**(8), 629–640 (2011)
162. Schnitter, S., Barth, A., Schnitter, O., Horneffer, M.: Benefits from 2-layer traffic engineering for IP/MPLS networks. In: Proceedings of the the 13th International Telecommunications Network Strategy and Planning Symposium (Networks 2008), volume Supplement, pp. 1–6, September 2008
163. Zhang, Xiaoning; Shen, Feng, Wang, Li, Wang, Sheng, Li, Lemin, Luo, Hongbin: Two-layer mesh network optimization based on inter-layer decomposition. *Photonic Netw. Commun.* **21**(3), 310–320 (2011)
164. Gladysz, J., Walkowiak, K.: Design of MPLS over DWDM architecture for unicast and anycast flows. In: Balandin, S., Roman, D., Yevgeni, K. (eds.) *Smart Spaces and Next Generation Wired/Wireless Networking*. Lecture Notes in Computer Science, pp. 172–183. Springer, Berlin (2010)

Modeling and Optimization of Cloud-Ready and
Content-Oriented Networks

Walkowiak, K.

2016, XVIII, 279 p. 50 illus., 7 illus. in color., Hardcover

ISBN: 978-3-319-30308-6