J Netw Syst Manage
DOI 10.1007/s10922-013-9273-5

A Hybrid End-to-End QoS Path Computation Algorithm for PCE-Based Multi-Domain Networks

Ahmed Frikha · Samer Lahoud · Bernard Cousin

Received: 30 June 2011 / Revised: 28 February 2013 / Accepted: 11 March 2013

Abstract Inter-domain quality of service (QoS) routing is a challenging problem for today's Internet. This problem requires the computation of paths that cross multiple domains and meet different QoS constraints. In addition, the used computation methods must meet the constraints of confidentiality and autonomy imposed by the domains of different operators. Path computation element (PCE)-based architecture offers a promising solution for inter-domain QoS routing. It ensures the computation of end-to-end QoS paths while preserving the confidentiality and the autonomy of the domains. In this paper, we propose a novel hybrid end-to-end QoS path computation algorithm, named HID-MCP, for PCE-based networks. HID-MCP is a hybrid algorithm that combines the advantages of pre-computation and on-demand computation to obtain end-to-end QoS paths. Moreover, it integrates a crankback mechanism for improving path computation results in a single domain or in multiple domains based on the PCE architecture. Detailed analyses are provided to assess the performance of our algorithm in terms of success rate and computational complexity. The simulation results show that our algorithm has an acceptance rate of the requests very close to the optimal solution; precisely, the difference is lower than 1 % in a realistic network. Moreover, HID-MCP has a low computational complexity. Besides, our solution relies on the PCE architecture to overcome the limitations related to inter-domain routing such as domain autonomy and confidentiality.

A. Frikha (✉) · S. Lahoud · B. Cousin
University of Rennes 1, IRISA, 35042 Rennes Cedex, France
e-mail: ahmed.frikha@irisa.fr
URL: www.ahmedfrikha.com

S. Lahoud
e-mail: samer.lahoud@irisa.fr

B. Cousin
e-mail: bernard.cousin@irisa.fr

Published online: 24 March 2013

Keywords Inter-domain QoS routing · Path computation element (PCE) · Crankback mechanism · Pre-computation · On-demand computation · *Look-ahead* information

1 Introduction

Nowadays, diverse advanced applications are provided over IP-based networks (e.g., IPTV, video-on-demand, and VoIP). Guaranteeing the quality of service (QoS) to such applications is a difficult problem, especially when service delivery requires crossing heterogeneous domains under the responsibility of different operators. In such a case, the problem becomes more complex. Moreover, operators adopt different policies that consolidate their economic interests and confidentiality clauses. Thus, cooperation between operators for providing QoS is possible only if it satisfies their policies. Routing is one of the primary mechanisms for providing QoS. It consists in the computation of an end-to-end path that ensures the delivery of the service while meeting the QoS constraints. Several recent works studied multi-domain QoS routing. Yannuzzi et al. [1] presented the challenges of finding disjoint QoS paths in multi-domain networks. One of the most challenging problems is domain visibility. In fact, information about domain topology and QoS metrics are confidential and cannot be exchanged between domains. This makes finding an inter-domain QoS path a hard task. Some other studies proposed to aggregate QoS information and exchange them between domains. Uludag et al. [2] provided an analysis of the different techniques of topology aggregation for QoS routing.

Path computation element (PCE)-based architecture [3] offers a promising solution for inter-domain QoS routing. It enables computing end-to-end QoS paths in multi-domain networks while preserving the confidentiality and autonomy of the domains by distributing the computations over the domains. This architecture supposes that each domain has one or more PCEs responsible for the intra-domain computation and can cooperate with other PCEs using the PCEP protocol in order to compute an end-to-end path [4, 5]. The backward-recursive procedure (BRPC) is introduced by Vasseur et al. [6] for the multi-domain path computation. This procedure relies on the PCE architecture to compute an end-to-end QoS path that meets the QoS requirement. This procedure allows PCE to exchange aggregated paths with other PCEs using a compact structure, called the virtual shortest path tree (VSPT). The drawback of this procedure is that it does not allow domains to exchange all aggregated paths but only one single path per entry border node. Geleji et al. [7] provided a comparison of different schemes for the end-to-end path computation based on the PCE architecture.

In this paper, we propose a novel inter-domain QoS routing algorithm relying on the PCE-based architecture, named HID-MCP. HID-MCP is based on a hybrid computation scheme. The hybrid computation scheme combines the advantages of the pre-computation scheme [8, 9] such as the low computational time and the advantages of the on-demand computation such as the high acceptance rate of the requests [10]. Our algorithm consists of two phases: an offline phase and an online

phase. In the offline phase, HID-MCP pre-computes a set of QoS paths, as well as look-ahead information for each domain. Look-ahead information gives a measure of the best QoS performance that can be provided by the domain. In the online phase, HID-MCP combines the pre-computed paths to obtain an end-to-end path that fulfills the QoS constraints. Combining the pre-computed paths does not always lead to an end-to-end path. In such a case, a crankback mechanism is executed to perform on-demand computations. Combining pre-computation and on-demand computation using a crankback mechanism improves the computation results and allows computational complexity to be reduced.

We distinguish two different crankback mechanisms: intra-domain crankback and inter-domain crankback. The intra-domain crankback is solicited to locally improve the computation results by executing an on-demand computation in the failing domain, while the inter-domain crankback performs a global improvement by executing an on-demand computation starting from one of the downstream domains in the crossed domain set. Note that this solution extends our recently published work in [11] by extending the algorithm and enhancing the simulations. We extend the algorithm by adding a new parameter that represents the distance in terms of domains between the failing domain and the domain from where the on-demand computation should start. Precisely, in [11], the inter-domain crankback signaling is sent to the destination domain. While in this paper, the inter-domain crankback signaling can be sent to one of the downstream domains in order to execute the on-demand computation starting from this domain and not only starting from the destination domain. This parameter allows us to tune the performance of the algorithm and to ensure a trade-off between the rapidity and the success rate of the algorithm. We detail the operations performed by HID-MCP using some examples. We improve the simulation side by considering larger and more realistic networks. We also compare the average computational time of the path combination and the on-demand computation in order to show the advantage of using pre-computation. Finally, we study the performance of our algorithm by comparing it with two algorithms. The first algorithm is an exact one, and it represents the optimal solution. The second algorithm is based on the Border Gateway Protocol (BGP), and it represents the inter-domain routing protocol employed in today's Internet. Detailed analyses are provided to assess the performance of our algorithm in terms of success rate and computational complexity. The simulation results show that our algorithm has an acceptance rate of the requests very close to the optimal solution, i.e., the difference is lower than 1 % in a realistic network. Moreover, HID-MCP has a low computational complexity. Besides, our solution relies on the PCE architecture to overcome the limitations related to inter-domain routing such as domain autonomy and confidentiality.

This paper is organized as follows. In Sect. 2, we formally define the inter-domain QoS routing problem and present the existing computation schemes for QoS routing. Section 3 presents the HID-MCP algorithm and its operations. Simulation results are presented and analyzed in Sect. 4 and conclusions are given in Sect. 5.

2 The Inter-Domain QoS Routing Problem and Computation Schemes

2.1 The Inter-Domain QoS Routing Problem

Inter-domain QoS routing, also known as the Inter-Domain Multi-Constraint Path (ID-MCP) computation problem, consists in computing a path subject to multiple QoS or cost constraints between a source and a destination node of a multi-domain network. Computing this path requires knowledge of QoS metrics on the network links. QoS metrics can be classified into three types: bottleneck metrics such as bandwidth, additive metrics such as delay, and multiplicative metrics such as loss rate [12]. The multiplicative metrics can be translated into additive metrics using the logarithm function. The bottleneck metrics can be resolved by omitting all that which violate the constraints and then computing the path on the residual graph. Therefore, we consider only additive metrics, in the following.

Let us introduce some notations to formally define the ID-MCP problem. Let $G(N, L, D)$ denote a graph of a multi-domain network, where N is the set of nodes, L is the set of links, and D is the set of domains. The graph G is composed of $|D|$ domains. Let m be the number of QoS constraints. An m-dimensional weight vector is associated with each link $e \in L$. This vector consists of m non-negative QoS weights $w_i(e)$, $i = 1 \ldots m$. Let p be a path in the graph $G(N, L, D)$ and $w_i(p)$ be the weight of p corresponding to the metric i. As metrics are additive, $w_i(p)$ is given by the sum of the weights of the ith metric of the links of the path p : $w_i(p) = \sum_{e_j \in p}(w_i(e_j))$. Let $\vec{W}(p) = (w_1(p), w_2(p), \ldots, w_m(p))$ denote the weight vector of the path p.

Definition 1 Given a source node s, a destination node d and a set of constraints given by the constraint vector $\vec{C} = (c_1, c_2, \ldots, c_m)$, the ID-MCP computation problem consists in finding a path p that satisfies $w_i(p) \leq c_i$, $\forall i \in 1 \ldots m$. Such a path p is called a feasible path.

The ID-MCP problem belongs to the $\mathcal{NP}$-complete problems [12] and may have zero, one, or multiple solutions (feasible paths). Besides, information about the internal topology or the QoS metrics on the links is confidential since the operators can be in competition. Therefore, computing such a path while meeting all of these constraints is a challenging task.

Currently, the universal inter-domain routing protocol is BGP. Although BGP has proved its efficiency to enable inter-domain routing while taking into account confidentiality and autonomy concerns, this protocol cannot solve the ID-MCP problem since it does not take the QoS constraints into account. One major limitation of BGP is the unique path propagation. Infact, BGP cannot propagate multiple paths for the same destination. This prevents the algorithm from being QoS-aware. Another problem is that BGP performs the inter-domain route selection without any coordination between the domains. To overcome these limitations, many extensions for BGP are proposed to support QoS routing [13, 14]. However,

the QoS capabilities of these propositions remain limited, due to the fact that only one path per destination can be propagated from a domain to another neighbor domain. Therefore, it is not possible to find an end-to-end path that meets the QoS constraints using these propositions.

The research community has recently been exploring the use of distributed architecture to solve the ID-MCP problem. The IETF defines in [3] an alternative architecture based on Path Computation Elements (PCE) that enables the cooperation between domains to find a path subject to multiple constraints. In this architecture, each domain has one or more PCEs that are able to perform advanced routing computations within the domain. The PCE can communicate with external PCEs in order to offer the inter-domain cooperation needed for QoS end-to-end routing. Moreover, the PCE-based architecture preserves confidentiality and the autonomy of each domain by distributing the computation over the domains. Each domain locally computes the segment of the end-to-end path that traverses it.

To our knowledge, few works have been proposed to solve the ID-MCP problem using the PCE architecture. The algorithm proposed in [15] extends the exact algorithm SAMCRA [16] to an inter-domain level to solve the ID-MCP problem. The drawback of this algorithm is its high complexity. The ID-PPPA algorithm [17] and the ID-MEFPA algorithm [18] rely on the PCE architecture and attempt to solve the ID-MCP problem using a pre-computation scheme. These solutions are efficient in terms of rapidity and have a good success rate. The pre-computation scheme ensures a very low computational time but requires a periodical update of the network link state information to maintain a high success rate. Work in [19] also proposes a promising distributed solution with crankback mechanisms for inter-domain routing. However, this solution cannot take into account several QoS metrics.

2.2 Computation Schemes for Inter-Domain QoS Routing

Different computation schemes have been proposed in the literature to solve the QoS routing problem [10]. The on-demand computation scheme attempts to find a feasible path for each request using network state information. The computation is triggered upon the reception of a QoS request. This computation scheme provides a high probability of finding a feasible path since the network state information is up-to-date [10]. This scheme is widely used in existing networks but presents some serious limitations with the emerging applications in the Internet. In fact, the ID-MCP problem belongs to the $\mathcal{NP}$-Complete problems, and consequently, the performance of on-demand routing (in terms of response time) is severely affected. In contrast with the on-demand computation scheme, the pre-computation scheme allows the QoS routing problem to be solved while speeding up the response time [8, 9]. The pre-computation proceeds in two phases: in advance, it prepares a set of paths satisfying predetermined QoS requests. Then, at the reception of a QoS request, it attempts to rapidly provide a feasible path among the pre-computed

paths. However, the deployment of a pre-computation scheme in networks can run into problems. One problem is that the stored paths are computed based on a snapshot of the network state taken before the reception of the QoS request [10]. Consequently, the pre-computed paths do not necessarily satisfy the QoS requests after a change in the network state [10]. In addition, a pre-computation scheme cannot predict all possible QoS requests even if the snapshot remains valid. This decreases the success rate of the pre-computation scheme. For these reasons, a hybrid computation scheme is proposed. This scheme combines the two aforementioned computation schemes and is performed in two phases. As with the pre-computation scheme, the first phase consists in preparing a set of paths or segment of paths in advance. The second phase is more sophisticated. When the pre-computed paths do not lead to a path that satisfies the request, on-demand computation is triggered. Therefore, the hybrid computation scheme ensures a high acceptance rate for requests while reducing the computation time. In this paper, we rely on a hybrid computation scheme to solve the ID-MCP problem.

3 The HID-MCP Algorithm

In this paper, we propose a novel inter-domain QoS routing algorithm based on a hybrid computation scheme and named HID-MCP (Hybrid ID-MCP). The present section details the operations performed by this algorithm. Figure 1 illustrates the building block architecture of the algorithm. This architecture is implemented in each PCE of each domain. Based on this architecture, each PCE performs autonomous computations and cooperates with other PCEs to compute the end-to-end path. This reinforces domain autonomy and confidentiality. As shown in Fig. 1, the HID-MCP algorithm consists of two phases. In the first phase, named the offline path computation phase, the algorithm executes a path segment computation procedure and computes *look-ahead* information in each domain. The pre-computed path segments and *look-ahead* information are stored in a database for later use. The second phase, named the online path computation phase, is triggered upon the reception of a QoS request. In this phase, HID-MCP computes an end-to-end path

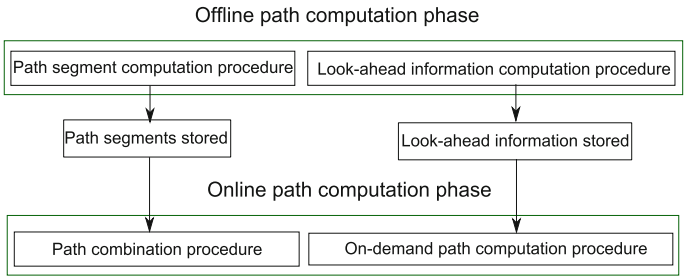

Fig. 1 The building block architecture of the HID-MCP algorithm

Table 1 Notations for the offline phase

Offline phase	
D_q	A given domain
N_q	The set of nodes in the domain D_q
L_q	The set of links in the domain D_q
B_{i_q}	The set of ingress border nodes of the domain D_q
B_{e_q}	The set of egress border nodes of the domain D_q
n_j	An ingress border node in B_{i_q}
n_k	A node within the domain D_q or an egress border node in B_{e_q}
m	The number of QoS metrics

that spans multiple domains and fulfills the QoS constraints. The algorithm attempts to find such a path, first by combining the stored pre-computed intra-domain path segments and second, by executing an on-demand path computation procedure that takes benefits from the stored *look-ahead* information to speed up the computational time of the algorithm.

3.1 The Offline Path Computation Phase

The offline computation phase consists in computing a set of intra-domain paths subject to multiple predetermined QoS constraints in advance. It also computes *look-ahead* information at the level of each entry border node of the corresponding domain. In this section, we detail the operations involved in these two computations. For now, let us introduce the necessary notations. Table 1 presents the notations used for describing the offline path computation phase.

3.1.1 The Path Segment Computation Procedure

This procedure pre-computes a set of paths from each entry border node of the domain toward the other nodes of this domain as well as the entry border nodes of the neighbor domains. These paths satisfy a set of predetermined additive QoS constraints. In practice, some QoS metrics are more critical for certain applications, such as the delay for VoIP-based applications. Therefore, our procedure pre-computes for each single QoS metric the path that minimizes the weight corresponding to this metric. For example, it pre-computes the path that minimizes the delay; this path can be useful for VoIP-based applications.

Let D_q be the considered domain, n_j be an entry border node of domain D_q (i.e., $n_j \in B_{i_q}$), n_k be a node of D_q or an entry border node of a neighbor domain of D_q (i.e., $n_k \in N_q \cup B_{e_q}$), and m be the number of QoS metrics. Our procedure computes m shortest paths from n_j to n_k. Each shortest path minimizes a single QoS metric. Hence, from each entry border node n_j of D_q, this procedure computes m shortest

path trees. Each shortest path tree is computed using the Dijkstra algorithm and considering a single metric. Therefore, our procedure executes Dijkstra m times per border node. The complexity of the path segment computation procedure is given by (1).

$$\mathcal{O}(|B_{i_q}| * m((|N_q| + |B_{e_q}|)\log(|N_q| + |B_{e_q}|) + |L_q|)) \tag{1}$$

The complexity of this procedure depends on the number of constraints m. For one border node, the complexity of this procedure corresponds to m times the complexity of Dijkstra with the fibonacci heap, which is $\mathcal{O}((|N_q| + |B_{e_q}|)\log(|N_q| + |B_{e_q}|) + |L_q|)$. Considering the B_{i_q} entry border nodes of the domain, the global complexity is then in (1).

3.1.2 Look-Ahead Information Computation Procedure

During the offline phase of HID-MCP, we propose the computation of *look-ahead* information in each domain. This information gives a measure of the best QoS performance that can be provided by the domain. Particularly, it allows the computation search space of a potential on-demand path computation procedure to be reduced. For instance, this information allows infeasible paths to be discarded from the search space of the procedure before exploring these paths. Therefore, *look-ahead* information reduces the computational complexity of the online phase and contributes to maintaining a reasonable response time. *Look-ahead* information is inferred from the result of the pre-computation algorithm. Let n_j be an entry border node of the domain, and n_k be a node of the domain or an entry border node of a neighbor domain, and $p_{n_j \mapsto n_k;i}*$ denotes the pre-computed shortest path between node n_j and node n_k considering the ith metric. The weight $w_i(p_{n_j \mapsto n_k;i}*)$ is the lowest possible path weight between n_j and n_k. Similarly, let us denote by $\overrightarrow{W^*}_{n_j \mapsto n_k} = (w_1^*, \ldots, w_m^*)$ the vector where $w_i^* = w_i(p^*_{n_j \mapsto n_k;i})$. Thus, $\overrightarrow{W^*}_{n_j \mapsto n_k}$ represents the lowest weights to reach n_k from n_j for each single metric. We note that a path does not necessarily exist with these lowest weights for all the metrics simultaneously. However, this vector can be used in the online path computation phase to discard infeasible paths from the search space.

The complexity of the look-ahead information computation procedure is given by (2).

$$\mathcal{O}(m * (|N_q| + |B_{e_q}|) * |B_{i_q}|) \tag{2}$$

Look-ahead information is inferred from the result of the path segment computation. At each entry border node of the domain, there are at most $m * (|N_q| + |B_{e_q}|)$ stored pre-computed paths. Hence, at the level of an entry border node n_j the complexity of computing the $(|N_q| + |B_{e_q}|)$ vectors $\overrightarrow{W^*}_{n_j \mapsto n_k}$ is in $\mathcal{O}(m * (|N_q| + |B_{e_q}|))$. Therefore,

Table 2 Notations for the online computation algorithms

Algorithm 1	
s	The source node
d	The destination node
Seq	The selected domain sequence $Seq = \{D_1, D_2, .., D_r\}$
r	The number of domains in Seq
D_1	The destination domain
D_r	The source domain
H_q	The VSPH received in the domain D_q
$hops$	The number of backward hops in terms of domains for the crankback
Algorithm 2	
P_q	The set of pre-computed paths in domain D_q
E_q	The set of egress nodes in domain D_q
I_q	The set of ingress nodes in domain D_q
m	The number of constraints
Algorithm 3	
l	The maximum number of shortest paths selected from a VSPH
k	The parameter of TAMCRA
Ψ_q	Set of look-ahead information in domain D_q
L_{a_q}	Set of look-ahead information in domain D_q from I_q to E_q

the complexity of computing the *look-ahead* information for all the entry border nodes of the domain is in (2).

3.2 The Online Path Computation Phase

The online path computation consists in finding a feasible end-to-end path using the pre-computed paths and taking advantage of the *look-ahead* information. Upon the reception of a QoS request, the source and the destination domains are determined. The PCE of the source domain forwards the QoS request between PCEs toward the PCE of the destination domain using a PCReq message defined in the PCEP protocol [4]. The best domain sequence that links the source and the destination domains is then selected according to the cooperation policy of the operators. Operators can also base their selection for the best domain sequence according to the number of traversed domains, i.e., by selecting the shortest domain sequence. Furthermore, selecting the best domain sequence will reduce the complexity of the problem by limiting the path computation in the domain sequence and not in all domains. After that, the path computation is triggered in the destination domain toward the source domain following the selected domain sequence. Note that, without loss of generality, we rely on the BRPC, introduced by Vasseur et al. [6], for the end-to-end path computation.

Algorithm 1 Online Phase of HID-MCP (Seq,s,d,$hops$)

```
1: q ← 1; H_1 ← φ; reject_request ← false;
2: while (q ≤ r) and not(reject_request) do
3:    H_{q+1} ← Path_combination_procedure(D_q, H_q, Seq, s, d);
4:    if H_{q+1} ≠ φ then
5:       q ← q + 1;
6:    else if hops == 0 then
7:       H_{q+1} ← On_demand_computation(D_q, H_q, Seq, s, d);
8:       if H_{q+1} ≠ φ then
9:          q ← q + 1;
10:      else
11:         reject_request ← true;
12:      end if
13:   else
14:      q ← max{1, q − hops};
15:      while (q ≤ r) and not (reject_request) do
16:         H_{q+1} ← On_demand_computation(D_q, H_q, Seq, s, d);
17:         if H_{q+1} ≠ φ then
18:            q ← q + 1;
19:         else
20:            reject_request ← true;
21:         end if
22:      end while
23:   end if
24: end while
25: Return reject_request == false
```

Table 2 illustrates the notations used in the online computation algorithms. As shown in this table, $Seq = \{D_1, D_2, \ldots, D_r\}$ denotes the selected domain sequence, where D_1 is the destination domain and D_r is the source domain. d is the destination node and s is the source node. Algorithm 1 illustrates the operations performed in the online phase of HID-MCP. First, our algorithm attempts to compute an inter-domain path by combining the pre-computed paths in each domain D_q in Seq starting from the destination domain D_1. The path combination procedure is called (line 3). Operations performed by this procedure are detailed in Sect. 3.2.1. The result of the combination procedure in each domain D_q is a set of sub-paths linking the destination node to the entry border nodes of the up-stream domain D_{q+1}. These sub-paths are sent to domain D_{q+1} to combine them with the pre-computed segments in domain D_{q+1}. To preserve domain confidentiality, sub-paths are communicated between domains under a novel compact structure named VSPH (Virtual Shortest Path Hierarchy[1]). This structure contains only the end nodes of the paths (the destination node and the entry border nodes of the up-stream domain) as well as the weight vector of each path. The protocol PCEP allows domains to transfer a VSPT [6]. Therefore, we can also transfer a VSPH as it can be defined as a set of VSPTs. Hence, the VSPH structure is consistent with the PCEP protocol. The VSPH received in domain D_q is denoted by H_q in algorithm 1. A virtual path $p_{d \to n_k}$ is represented in the VSPH by $\left[d, n_k, \vec{W}\left(p_{d \to n_k}\right)\right]$, where d is the destination node,

[1] The hierarchy is a structure that enables the storage of multiple paths between any two nodes [20].

n_k is an entry border node of the upstream domain D_{q+1}, and $\vec{W}(p_{d\to n_k})$ is the weight vector of $p_{d\to n_k}$.

Combining the pre-computed paths in each domain can lead to an end-to-end path, as detailed in Sect. 3.2.1. However, in some cases, no feasible path is found; i.e., the returned VSPH is empty. In the following, we introduce two novel approaches that use athe crankback mechanism in order to overcome this limitation. The first approach executes an intra-domain crankback while the second approach executes an inter-domain crankback. Both of these approaches perform an on-demand path computation. The aim of the on-demand path computation procedure is to provide better results than the pre-computed ones. Operations performed by this procedure are detailed in Sect. 3.2.2.

The parameter *hops* in the algorithm represents the distance in terms of domains between the failing domain and the domain from where the on-demand computation should start. When $hops = 1$ the on-demand computation should start from the downstream domain $D_{q\text{-}1}$, and when $hops = q - 1$ the on-demand computation starts from the destination domain D_1, etc. For the intra-domain crankback approach $hops = 0$. The intra-domain crankback approach (lines 6–12) executes the on-demand path computation procedure in the current domain D_q, i.e., where the combination has failed. Then, if a feasible path is found in the current domain, this path is sent to the up-stream domain, which will resume the path combination procedure. Otherwise, if the algorithm does not find a solution in the current domain, i.e., $H_{q+1} = \phi$, the request is rejected. The look-ahead information allows operators to know if the inter-domain crankback may lead to a feasible path or not. When the combination fails in D_q, we combine H_q with the look-ahead information computed in D_q, if all obtained paths are infeasible. The intra-domain crankback cannot lead to a feasible path. In this case, operators have the choice between executing an inter-domain crankback or rejecting the request. Combining look-ahead information with the VSPH will be detailed in Sect. 3.2.2.

The inter-domain crankback approach (lines 13–23) executes the on-demand path computation procedure starting from the domain $D_{max\{1,q\text{-}hops\}}$. When $q - hops < 1$, the computation starts from the destination domain D_1. Each domain executes the on-demand path computation procedure and sends the computed VSPH to the up-stream domain. The computation stops when an end-to-end path is found or when the on-demand path computation procedure does not find a solution, i.e., the returned VSPH is empty. In the latter case, the request is rejected.

Figure 2 illustrates the crankback signaling when the combination fails in the domain D_q. In this example, the number of constraints $m = 2$, and the constraint vector equals (10,10). Domain D_q receives the VSPH H_q from domain $D_{q\text{-}1}$. It combines the pre-computed paths in D_q with the aggregated paths in H_q. The combination procedure is detailed in the next section with an example. The result of the combination procedure is the VSPH H_{q+1}. However, H_{q+1} does not contain any feasible paths. Therefore, a crankback mechanism must be executed. According to the value of *hops*, the on-demand computation is executed in either the current domain D_q ($hops = 0$) or the downstream domain in $\{D_{q\text{-}1}, D_{q\text{-}2}, \ldots, D_1\}$ ($hops > 0$).

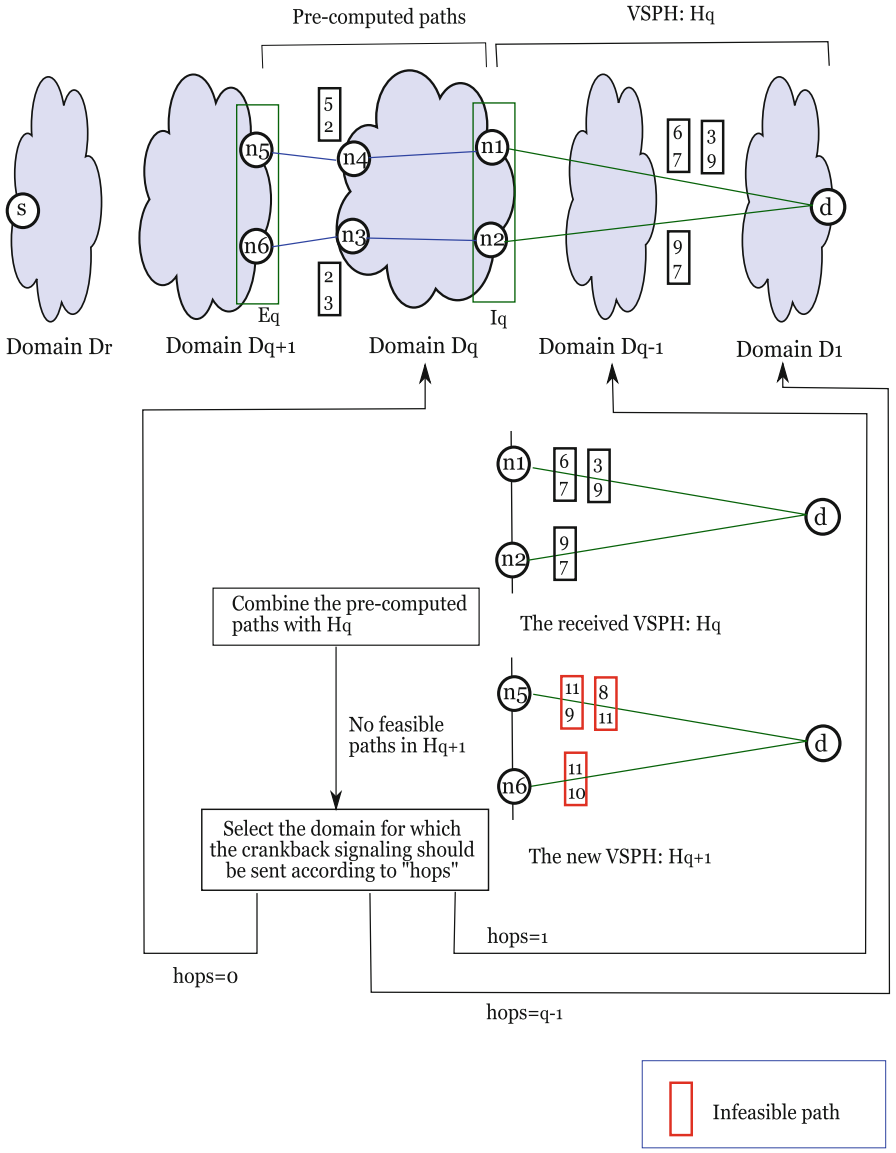

Fig. 2 Crankback signaling when the combination fails in domain D_q

3.2.1 The Path Combination Procedure

The aim of this procedure is to combine the paths in the received VSPH with the internally pre-computed one. Algorithm 2 illustrates the operations performed by the path combination procedure. First, the combination procedure selects the pre-computed paths linking nodes in the set I_q to nodes in the set E_q, where I_q is the

ingress node set and E_q the egress node set (lines 1-13). Then, these paths are combined with the aggregated paths received in the VSPH (line 17). Finally, feasible paths are aggregated and added to the new VSPH that will be sent to the upstream domain. Note that, at the level of the destination domain D_1 there is no received VSPH ($H_1 = \phi$), the procedure selects the feasible pre-computed paths linking the destination d to the entry border nodes of domain D_2, and aggregates them in a VSPH to be sent to domain D_2.

Algorithm 2 Path combination procedure (D_q,H_q,Seq,s,d)

1: $P_q \leftarrow \{p/p$ pre-computed path in domain $D_q\}$;
2: $H_{q+1} \leftarrow \phi$;
3: **if** $D_q == D_1$ **then**
4: $\quad I_q \leftarrow \{d\}$;
5: **else**
6: $\quad I_q \leftarrow \{n_j/n_j$ leaf node in $H_q\}$;
7: **end if**
8: **if** $D_q == D_r$ **then**
9: $\quad E_q \leftarrow \{s\}$;
10: **else**
11: $\quad E_q \leftarrow \{n_k/n_k$ entry border node of domain $D_{q+1}\}$;
12: **end if**
13: $Selected\_paths \leftarrow \left\{p_{n_j \rightarrow n_k}/p_{n_j \rightarrow n_k} \in P_q, n_j \in I_q, n_k \in E_q\right\}$;
14: **if** $D_q \neq D_1$ **then**
15: $\quad$ **for** $p_{d \rightarrow n_j} \in H_q$ **do**
16: $\quad\quad$ **for** $p_{n_j \rightarrow n_k} \in Selected\_paths$ **do**
17: $\quad\quad\quad \overrightarrow{W}(p_{d \rightarrow n_k}) \leftarrow \overrightarrow{W}(p_{d \rightarrow n_j}) + \overrightarrow{W}(p_{n_j \rightarrow n_k})$;
18: $\quad\quad\quad$ **if** $p_{d \rightarrow n_k}$ is feasible **then**
19: $\quad\quad\quad\quad$ Add $\left[d, n_k, \overrightarrow{W}(p_{d \rightarrow n_k})\right]$ to H_{q+1};
20: $\quad\quad\quad$ **end if**
21: $\quad\quad$ **end for**
22: $\quad$ **end for**
23: **else**
24: $\quad$ **for** $p_{d \rightarrow n_k} \in Selected\_paths$ **do**
25: $\quad\quad$ **if** $p_{d \rightarrow n_k}$ is feasible **then**
26: $\quad\quad\quad$ Add $\left[d, n_k, \overrightarrow{W}(p_{d \rightarrow n_k})\right]$ to H_{q+1};
27: $\quad\quad$ **end if**
28: $\quad$ **end for**
29: **end if**
30: Return H_{q+1};

Figure 3 illustrates an example of path combination with two constraints ($m = 2$) in an intermediate domain D_q. In this example the constraint vector equals $\overrightarrow{C} = (10, 10)$. Domain D_q pre-computes two paths corresponding to the weight vectors (5, 2) and (2,3). The VSPH H_q contains two aggregated paths from the destination d to the domain D_q. The weight vectors of these aggregated paths are respectively (3,7) and (5,4). We combine the aggregated paths with the pre-computed ones and we obtain two paths corresponding to the weight vectors (8,9) and (7,7). These two paths are aggregated in a new VSPH H_{q+1} and then sent to domain D_{q+1}.

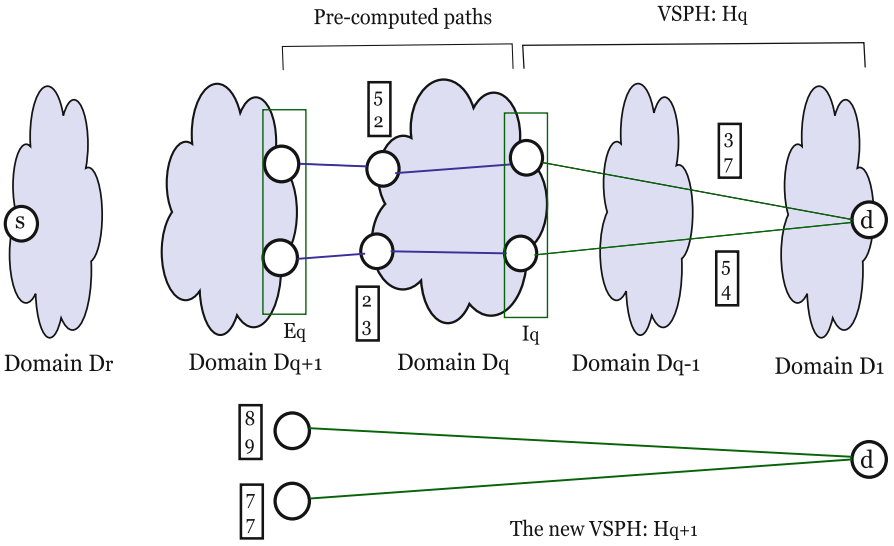

Fig. 3 Combining the pre-computed paths in domain D_q with the VSPH

The complexity of the pre-computed path combination procedure at the level of an intermediate domain $D_q \in \{D_2, \ldots, D_{r-1}\}$ is given by (3).

$$\mathcal{O}(m^q |E_q| * |I_q|) \tag{3}$$

There are at most m^{q-1} paths from the destination to each entry border node of the domain D_q. In addition, at each entry border node, there are at most $m^*|E_q|$ stored pre-computed paths to reach the upstream domain D_{q+1}. Hence, the complexity of combining the pre-computed paths and the received paths at the level of an entry border node is in $\mathcal{O}(m^q|E_q|)$. This operation is performed at each entry border node between the domain D_q and the downstream domain D_{q-1}. Therefore, the global complexity of this procedure at each domain is in $\mathcal{O}(m^q|E_q| * |I_q|)$.

3.2.2 The On-demand Path Computation Procedure

When the pre-computed path combination procedure does not lead to a feasible path, the on-demand path computation procedure is called in the current domain or starting from the domain $D_{max\{1,q\text{-}hops\}}$ according to the two aforementioned approaches. We propose a modified version of the TAMCRA algorithm to perform the on-demand computation. Work in [21] shows that TAMCRA is an efficient tunable heuristic for the MCP problem. TAMCRA introduces a new parameter k that limits the maximum number of stored paths at each intermediate node when searching for a feasible path. This parameter allows TAMCRA's performance to be

tuned: the success rate can be improved by increasing k at the expense of increased computational complexity.

Algorithm 3 On demand computation procedure (D_q,H_q,Seq,s,d)

1: $temp\_paths \leftarrow \phi$; $feasible\_paths \leftarrow \phi$; $Selected\_paths \leftarrow \phi$
2: $\Psi_q \leftarrow \left\{ \overrightarrow{W^*} \,/\, \overrightarrow{W^*} \text{ look-ahead information in domain } D_q \right\}$;
3: **if** $D_q \neq D_1$ **then**
4: $I_q \leftarrow \{n_j / n_j \text{ leaf node in } H_q\}$;
5: **if** $D_q == D_r$ **then**
6: $E_q \leftarrow \{s\}$;
7: **else**
8: $E_q \leftarrow \{n_k / n_k \text{ entry border node of domain } D_{q+1}\}$;
9: **end if**
10: $L_{a_q} \leftarrow \left\{ \overrightarrow{W^*}_{n_j \rightarrow n_k} \in \Psi, n_j \in I_q, n_k \in E_q \right\}$;
11: **for** $\left[d, n_j, \overrightarrow{W}\,(p_{d \rightarrow n_j}) \right] \in H_q$ **do**
12: **for** $\overrightarrow{W^*}_{n_j \rightarrow n_k} \in L_{a_q}$ **do**
13: $\overrightarrow{W^*}\,(d \rightarrow n_j \rightarrow n_k) \leftarrow \overrightarrow{W}\,(p_{d \rightarrow n_j}) + \overrightarrow{W^*}_{n_j \rightarrow n_k}$;
14: **end for**
15: $S(p_{d \rightarrow n_j}) \leftarrow \min_{n_k \in E_q} \left\{ \max_{i \in 1..m} \left(\frac{w_i^*(d \mapsto n_j \mapsto n_k)}{c_i} \right) \right\}$;
16: **if** $S(p_{d \rightarrow n_j}) \leq 1$ **then**
17: Add $[n_j, \overrightarrow{W}\,(p_{d \rightarrow n_j}), S(p_{d \rightarrow n_j})]$ to $temp\_paths$;
18: **end if**
19: **end for**
20: **if** $temp\_paths \neq \phi$ **then**
21: $Selected\_paths \leftarrow l$ shortest paths having the lowest score S in $temp\_paths$
22: **else**
23: $H_{q+1} \leftarrow \phi$;
24: Return H_{q+1}
25: **end if**
26: **for** $\overrightarrow{W}\,(p_{d \rightarrow n_j}) \in Selected\_paths$ **do**
27: Initialize n_j with the weight vector $\overrightarrow{W}\,(p_{d \rightarrow n_j})$ in H_q
28: Execute TAMCRA in D_q starting from n_j toward every node in E_q
29: Add the obtained feasible paths to $feasible\_paths$
30: **end for**
31: **else**
32: Execute TAMCRA in D_1 starting from d
33: Add the obtained feasible paths to $feasible\_paths$
34: **end if**
35: Extract H_{q+1} from $feasible\_paths$
36: Return H_{q+1}

Algorithm 3 illustrates the operations performed by the on-demand path computation procedure. First of all, our proposed procedure computes a prediction for the lowest weight vector to reach domain D_{q+1} through each path in the received VSPH (lines 11–19). We define for each aggregated path $\left[d, n_j, \overrightarrow{W}\,(p_{d \rightarrow n_j}) \right]$ in the VSPH and for each node n_k in E_q, a weight vector $\overrightarrow{W^*}\,(d \mapsto n_j \mapsto n_k)$ that represents

the sum of the weight vector of the computed segment $p_{d\mapsto n_j}$ and the lowest weight vector to reach n_k from n_j (line 13). Therefore, $\overrightarrow{W^*}(d\mapsto n_j\mapsto n_k) = \overrightarrow{W}(p_{d\mapsto n_j}) + \underset{n_j\mapsto n_k}{\overrightarrow{W^*}}$, where $\underset{n_j\mapsto n_k}{\overrightarrow{W^*}}$ is given by the *look-ahead* information. Note that the weight vector $\overrightarrow{W^*}(d\mapsto n_j\mapsto n_k)$ is not necessarily associated with an existing path. Next, we discard infeasible paths from the VSPH. For that, we define a new score for each path p given by: $S(p_{d\mapsto n_j}) = \min_{n_k\in E_q}\left\{\max_{i\in 1..m}\left(\frac{w_i^*(d\mapsto n_j\mapsto n_k)}{c_i}\right)\right\}$. This score represents the lowest score to reach d through $p_{d\mapsto n_j}$. A path p, that has a score $S(p) > 1$, is infeasible since it cannot lead to any node in E_q while meeting the QoS constraints. Then, we classify the remaining paths in the VSPH according to the score S. We select the l shortest paths having the l lowest scores, where l is a parameter of HID-MCP. The parameter l of HID-MCP is very important to reduce the computational complexity of the on-demand computation procedure and to decrease the number of paths exchanged between domains. After that, for each selected shortest path $p_{d\mapsto n_j}$, we initialize the node n_j by the corresponding weight vectors $\overrightarrow{W}(p_{d\mapsto n_j})$ in H_q and we execute the TAMCRA algorithm starting from node n_j to reach the nodes in E_q. The parameter l represents the maximum number of executing the TAMCRA algorithm. We note that at the destination domain, i.e., where the computations start, there is no received VSPH. Hence, the on-demand procedure executes TAMCRA starting from the destination node. Finally, we aggregate the feasible paths computed by TAMCRA in a new VSPH.

Figure 4 illustrates an example of on-demand computation in the intermediate domain D_q with $m = 2$, $k = 2$ and $l = 2$. In this example, the constraint vector equals $\overrightarrow{C} = (10, 10)$. The VSPH H_q contains four aggregated paths from the destination d to D_q. Two aggregated paths from d to n_1 corresponding to the weight vectors (5,3) and (4,4) and two aggregated paths from d to n_2 with the weight vectors (5,3) and (3,5). We combine the aggregated paths with the look-ahead information in order to discard infeasible paths from H_q and select the l shortest paths from H_q. The two shortest aggregated paths in H_q are $p_{d\mapsto n_1}$ and $p_{d\mapsto n_2}$ corresponding to the weight vectors (4,4) and (5, 3), respectively. Finally, we initialize node n_1 with (4,4) and node n_2 with (5, 3), and we execute the TAMCRA algorithm twice. One time starting from n_1 and another time starting from n_2. We obtain three feasible paths from d to n_5 with the weight vectors: (8, 8), (7, 8) and (10, 8). (8, 8) and (10,8) are dominated by (7, 8), so we discard them from the VSPH H_{q+1}. We also obtain three feasible paths from d to n_6 with the weight vectors: (7, 10), (7, 8), and (10, 7). (7, 10) is dominated by (7, 8), so we discard it from H_{q+1}.

The complexity of the on-demand path computation procedure at the level of an intermediate domain D_q is given by (4).

$$\mathcal{O}(l(k * (|N_q| + |E_q|)\log(k * (|N_q| + |E_q|)) + k^2 * m * |L_q|)) \tag{4}$$

The most significant point that determines the complexity of the on-demand path computation procedure is the number of executions of the TAMCRA algorithm. Knowing that the number of initialized nodes (i.e., nodes from where TAMCRA is

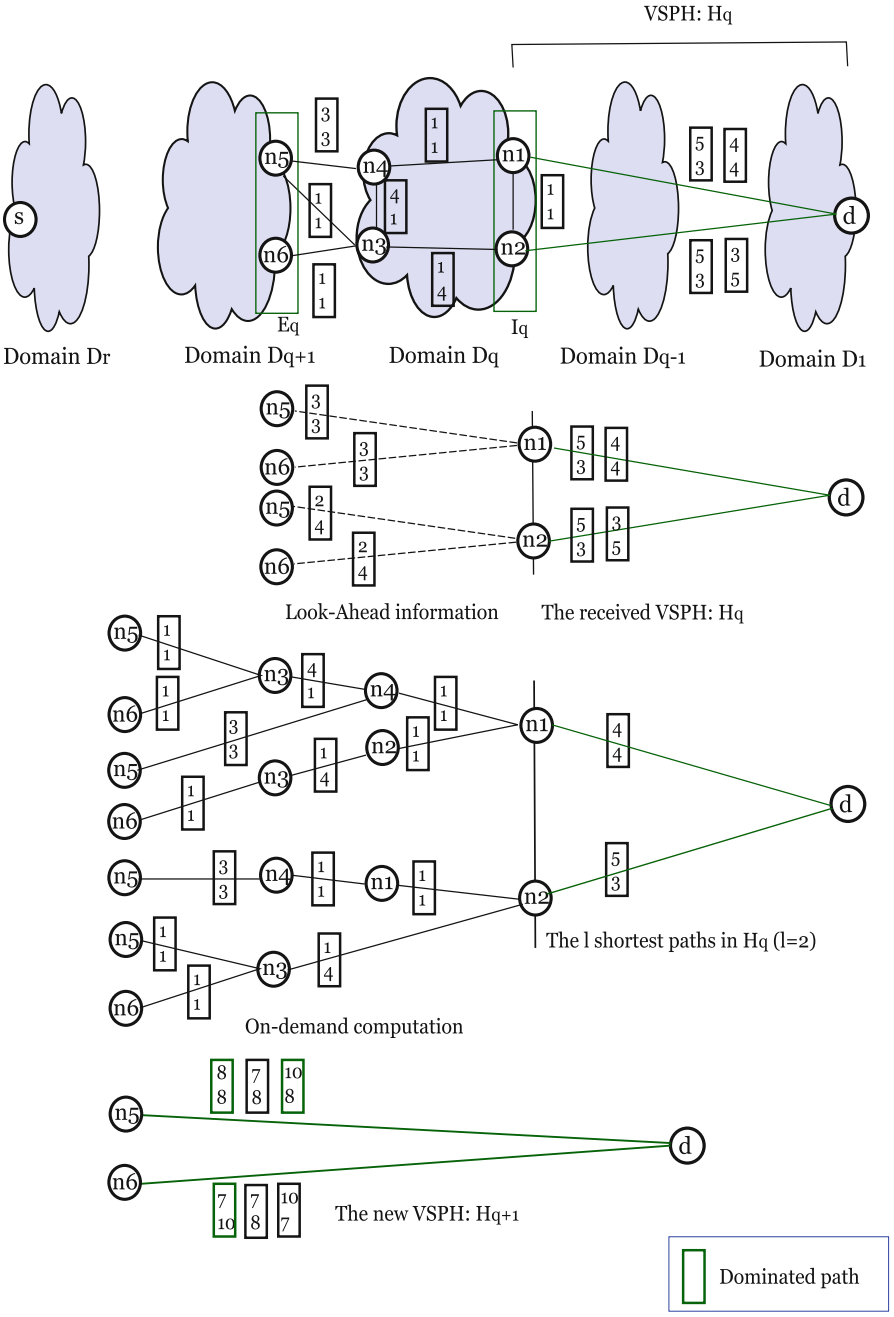

Fig. 4 The on-demand computation procedure in domain D_q with $m = 2$, $k = 2$ and $l = 2$

executed) is less than or equal to l, the complexity of this operation is in $\mathcal{O}(l(k * (|N_q| + |E_q|) \log(k * (|N_q| + |E_q|)) + k^2 * m * |L_q|))$, corresponding to l times the complexity of TAMCRA.

3.3 Security Considerations

In the proposed scheme, we can identify two security strategies: protecting the computation resources and protecting the network resources from malicious attacks. For the first strategy we propose to authenticate each QoS path computation request arriving in the PCE. Moreover, this authentication can be coupled with billing strategies. For the second one, we propose to encrypt path segments exchanged between domains, in order to preserve the confidentiality of the network topology. This can be done according to the work by Bradford et al. [22].

4 Simulation and Analysis

In this section, we evaluate the performance of the HID-MCP algorithm in two steps. First, we perform an experimental study of the complexity of the two proposed approaches: HID-MCP with an intra-domain crankback mechanism, and HID-MCP with an inter-domain crankback mechanism. We assess the performance of the algorithm in terms of computational time, and percentage of requests requiring either an intra-domain or an inter-domain crankback mechanism. Second, we compare HID-MCP with some existing well-known inter-domain QoS routing algorithms. We run simulations using two different topologies:

- The first topology is a network of 50 domains. Each domain is built based on Waxman's model with 50 nodes in each one. The probability that two nodes of the network are connected by an edge is expressed in [23]. We use this topology to assess the scalability of the algorithms in large networks.
- The second topology has a symmetrical backbone and is called SYM-CORE. SYM-CORE contains five interconnected domains and is taken from the work in [24]. This topology is used to assess the algorithms in a realistic case.

For these two topologies, we associate two additive weights generated independently following a uniform distribution [10, 1023] with each link. The QoS constraints are randomly generated according to the following: Let p_1 and p_2 denote the two shortest paths that minimize the first and the second metrics, respectively. Let $Z = [w_1(p_1), w_1(p_2)] \times [w_2(p_2), w_2(p_1)]$ be the constraint generation space, where $[w_1(p_1),w_1(p_2)]$ and $[w_2(p_2),w_2(p_1)]$ are two intervals. In other words, Z is the set of points (x, y), where $x \in [w_1(p_1), w_1(p_2)]$ and $y \in [w_2(p_2), w_2(p_1)]$. The problem does not belong to the $\mathcal{NP}$-complete problems outside Z, i.e., either infeasible or trivial. As shown in Fig. 5, we select from this space ten zones Z_i, $i = 1..10$ and we browse them from the strictest constraint zone Z_1 to the loosest constraint zone Z_{10}. Then, we assess the performance of the algorithms according to these zones. Without loss of generality, we do not consider

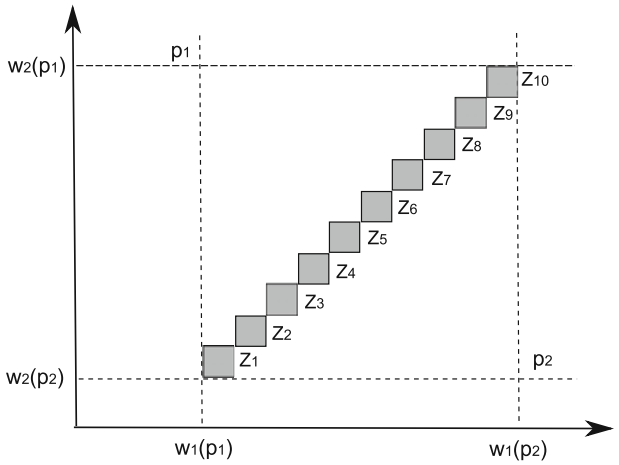

Fig. 5 Constraint generation zones for $m = 2$

Table 3 Comparison of the average computational times of the path combination procedure and the on-demand computation procedure in the SYM-CORE topology

Constraint zones	Z_3	Z_5	Z_8	Z_{10}
Combination Procedure	4.26×10^{-3}s	4.37×10^{-3}s	4.7×10^{-3} s	5.81×10^{-3}s
On-demand Procedure	1.73×10^{-1}s	2.09×10^{-1}s	2.34×10^{-1}s	2.97×10^{-1}s

the cases where one of the constraints is strict and the second one is loose, as the metrics are non-correlated and uniformly distributed in the same interval. In the following, each figure measures the variation of a performance metric according to the constraint generation zones Z_i, $i \in 1...10$, with a 95 % confidence interval.

4.1 Performance evaluation of HID-MCP

First, we compare the average computational time of the on-demand computation procedure and the combination procedure. This comparison will show us the importance of the pre-computation phase in reducing the computational time. In Table 3, we see the average computational times of the combination procedure and the on-demand computation procedure per domain in the constraint generation zones Z_3, Z_5, Z_8 and Z_{10}, in the SYM-CORE topology. The combination procedure has a computational time lower than 5.81 ms, while the on-demand computation has a computational time higher than 173 ms.

For the Waxman's model-based topology, Table 4 illustrates the average computational time of the combination procedure and the on-demand computation procedure in the constraint generation zones: Z_3, Z_5, Z_8 and Z_{10}. As shown in this table, the average computation time of the combination procedure is very low (lower than 2.52 ms) compared with that of the on-demand computation procedure (higher than 1,170 ms).

Table 4 Comparison of the average computational times of the path combination procedure and the on-demand computation procedure in the Waxman's model-based topology

Constraint zones	Z_3	Z_5	Z_8	Z_{10}
Combination Procedure	2.43×10^{-3} s	2.6×10^{-3} s	2.55×10^{-3} s	2.52×10^{-3} s
On-demand Procedure	1.27 s	1.33 s	1.33 s	1.17 s

It is clear that the combination procedure is very fast compared with the on-demand computation. This is due to the pre-computation phase that computes in advance multiple QoS paths and considerably reduces the computational time of the algorithm.

Secondly, we evaluate the success rate of the combination procedure and the need to execute either an intra-domain crankback or an inter-domain crankback in each domain. This study can help operators to make the best decision to execute either an intra-domain crankback or an inter-domain crankback, whenever the combination procedure fails to find an end-to-end feasible path. It also allows us to deduce the percentage of executions of the on-demand computation procedure in each domain for intra-domain crankback based HID-MCP. To determine if the intra-domain crankback may lead to a feasible path or not, we use the look-ahead information stored in the domain where the combination procedure has failed. Precisely, we combine the received VSPH with the look-ahead information and if all the obtained paths are infeasible, the intra-domain crankback cannot lead to a feasible path. In such a case the inter-domain crankback is required. Otherwise, an intra-domain crankback may lead to a feasible path in the failed domain.

We first consider the Waxman's model-based topology. The source and the destination domains are selected randomly among the 50 domains of the topology. The shortest domain sequence linking the source and the destination domains is selected. We denote the domain sequence by $Seq = \{D_1, D_2, \ldots, D_r\}$, where D_1 is the destination domain and D_r is the source domain. The combination procedure is always successful in the domains $\{D_1, D_2, \ldots, D_{r-1}\}$. Its success rate equals 100 % in these domains. The percentage of requests requiring either inter-domain crankback or intra-domain crankback equals zero. Therefore, the crankback mechanisms are never called from these domains.

For the source domain (D_r), Fig. 6 shows the success rate of the combination procedure as well as the percentages of requests requiring an inter-domain or an intra-domain crankback. In this figure, we see that the success rate of the combination is low when the constraints are generated in Z_1 or Z_2. The percentage of requests requiring the inter-domain crankback is around 90 %, and the success rate of the path combination procedure is lower than 5 % in these two zones. Nonetheless, the path combination procedure performs very well when the constraints are less strict. From Z_5 to Z_{10} more than 60 % of the requests do not require crankback mechanisms.

Now, we consider the SYM-CORE topology. We note that this topology is composed of five domains. The length of the domain sequence is three as SYM-CORE has a symmetrical backbone, (i.e., all the domains are interconnected through one central domain). In the destination and intermediate domains of the domain sequence, the success rate of the combination procedure equals 100 % in all

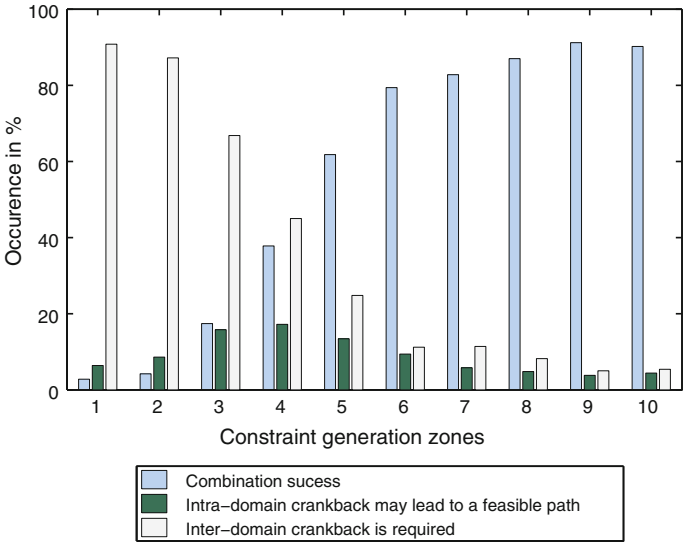

Fig. 6 Source domain (D_r): success rate of the combination procedure, percentage of requests requiring intra-domain crankback, and percentage of requests requiring inter-domain crankback

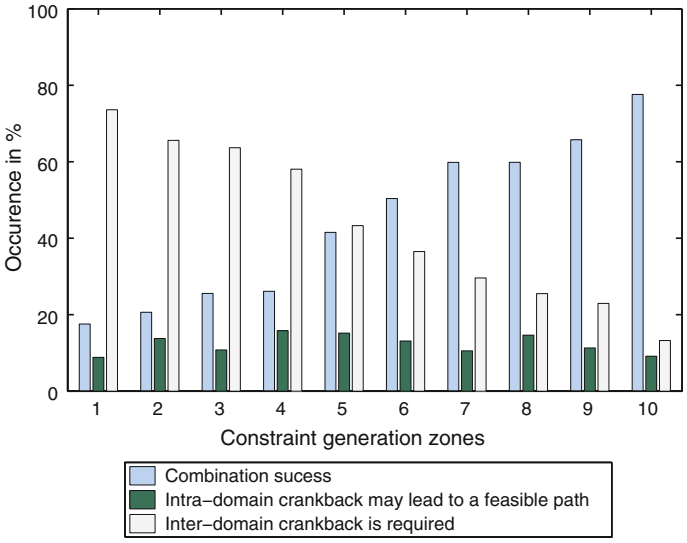

Fig. 7 Source domain: success rate of the combination procedure, and percentage of requests requiring intra-domain, and percentage of requests requiring inter-domain crankback

constraint generation zones. For the source domain, Fig. 7 shows the success rate of this procedure as well as the percentage of requests requiring the inter-domain crankback, and percentage of requests for which the intra-domain crankback may lead to a feasible path. The success rate of the combination is good and increases when

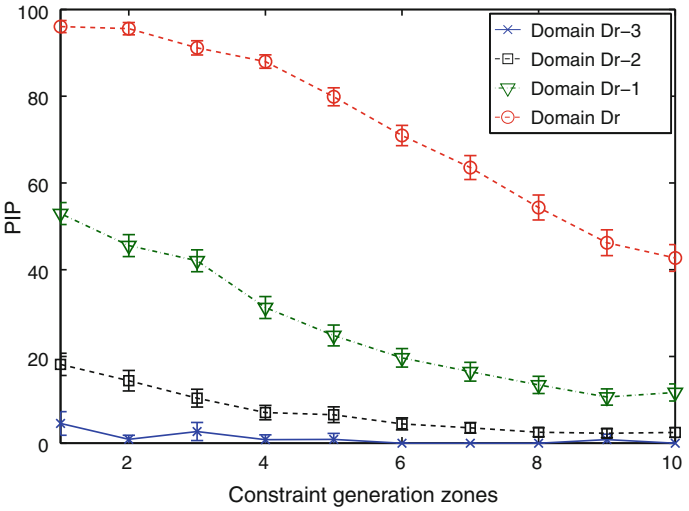

Fig. 8 Percentage of infeasible paths in the received VSPH in the Waxman's model-based topology

constraints are less strict. The inter-domain crankback is required with a percentage lower than 43 % from zone Z_5 to Z_{10}. The presented results show that the inter-domain crankback is infrequently required when the constraints are not very strict. This proves that the global empirical complexity of HID-MCP remains reasonable.

Finally, we focus on the importance of the look-ahead information to discard infeasible paths before executing the on-demand computation procedure. We define the percentage of infeasible paths (*PIP*) as the number of infeasible paths in the VSPH received by a given domain over the total number of paths in this VSPH. Figure 8 illustrates the *PIP* in the VSPH received in the source domain (i.e., D_r), and in the intermediate domains $\{D_{r\text{-}1}, D_{r\text{-}2}, D_{r\text{-}3}\}$, respectively. In the other domains, the PIP equals zero. In Fig. 8, we see that the *PIP* is very low in $D_{r\text{-}3}$ and $D_{r\text{-}2}$. In fact, the weights of the paths in $D_{r\text{-}3}$ and $D_{r\text{-}2}$ are still low compared to the constraint vector. Therefore, detecting infeasible paths using the look-ahead information is unlikely to happen. However, in the source domain, (i.e., D_r, the last domain in the computation process) the percentage of the discarded infeasible paths is very high, and equals 95 % in the constraint generation zones Z_1 and Z_2. This means that only 5 % of the paths in the received VSPH may lead to a feasible path. The *PIP* decreases when the constraints are less strict and it equals 45 % in Z_{10}. In $D_{r\text{-}1}$, the *PIP* is also high and equals 54 % in the constraint generation zone Z_1. Thus, 54 % of the paths will be discarded from the VSPH, and this will reduce the computational time of the on-demand computation procedure. These results prove the importance of look-ahead information in discarding infeasible paths, especially in the last two domains of the computation process (D_r and $D_{r\text{-}1}$). Nonetheless, the look-ahead information remains important in all crossed domains in order to select the l shortest paths. As explained previously, the parameter l is required to limit the number of executions of the TAMCRA algorithm when calling the on-demand computation procedure. Therefore, the lower l is, the faster the on-demand computation is.

4.2 Comparaison of HID-MCP with Existing Approaches

In this section, we evaluate the performance of the HID-MCP algorithm by comparing it with two different kinds of inter-domain routing algorithms: practical algorithms employed in today's Internet and exact QoS algorithms based on the PCE architecture. For the first type of algorithms, we select the BGP, as it is the current inter-domain routing protocol employed in the Internet. This protocol has a low computational complexity. However, it does not take into consideration the QoS metrics when routing. Thus, finding an end-to-end path that meets the QoS constraints using BGP is not always possible, even if such a path exists. Furthermore, with BGP only one single path per destination can be propagated between two neighbor domains. These two limitations prevent BGP from dealing with QoS routing.

The second type of algorithms are the exact ones. An algorithm is called exact if it can find a feasible end-to-end path, when such a path exists. As an exact algorithm, we choose the ID-MCP algorithm introduced by Bertrand et al. [15]. This algorithm has the best success rate as it is exact. However, the complexity of executing this algorithm in a given domain D_q corresponds to the complexity of the SAMCRA algorithm. This complexity is given by (5), where $K_{max} = \min\left(\exp((|N_q| + |E_q|) - 2)!, \frac{\prod_{i=1}^{m} c_i}{\max_j c_j}\right)$ [21]. Hence, the complexity of the ID-MCP algorithm is very high compared with that of BGP and HID-MCP algorithms.

$$\mathcal{O}((K_{max} * (|N_q| + |E_q|) \log(K_{max} * (|N_q| + |E_q|)) + K_{max}^2 m * |L_q|)) \tag{5}$$

We define the global success rate (*GSR*) of the algorithms by the ratio of the number of requests for which a feasible path is found to the total number of requests. For the inter-domain crankback based HID-MCP, we choose the parameter $hops = q - 1$. Thus, the crankback signaling will be sent to the destination domain D_1. We select this value for *hops* in order to study the performance of HID-MCP in the two extremity cases: intra-domain crankback and inter-domain crankback starting from the destination domain. For the other values of *hops*, the *GSR* of HID-MCP will be lower than that of HID-MCP with $hops = q - 1$ and higher than that of HID-MCP with $hops = 0$.

Figure 9 illustrates the variation of the *GSR* of HID-MCP with intra-domain crankback mechanism, HID-MCP with inter-domain crankback mechanism, the exact algorithm ID-MCP, and the BGP algorithm, in Waxman's model-based topology. The BGP algorithm has the lowest success rate. This is due to the fact that only one path can be propagated from a domain to another neighbor domain using BGP, while with the other algorithms multiple paths can be propagated using the VSPH structure. The ID-MCP algorithm is an exact algorithm, while the others are heuristic ones. This is why the success rate of ID-MCP is the highest. We remark that the success rate of HID-MCP with inter-domain crankback when $l = 2$ and $k = 2$ is very close to the success rate of ID-MCP. As explained in Sect. 3, k is a parameter of TAMCRA and l is the maximum number of paths selected from the VSPH for executing the TAMCRA computation. Of course, the higher l and k, the

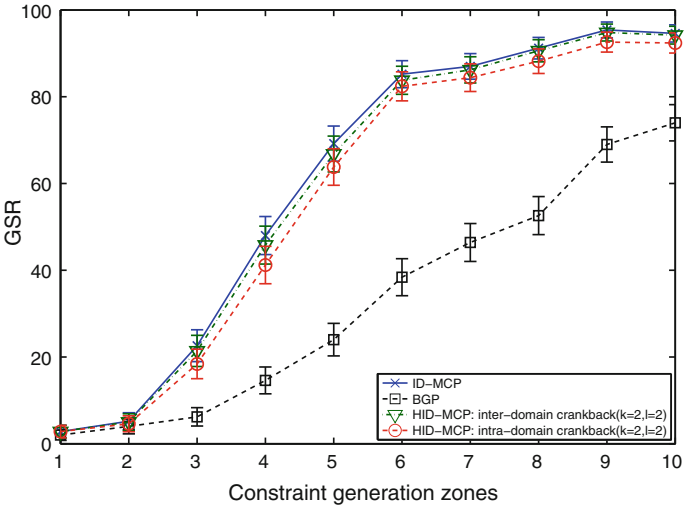

Fig. 9 Comparison of the global success rate of the algorithms in the Waxman's model-based topology

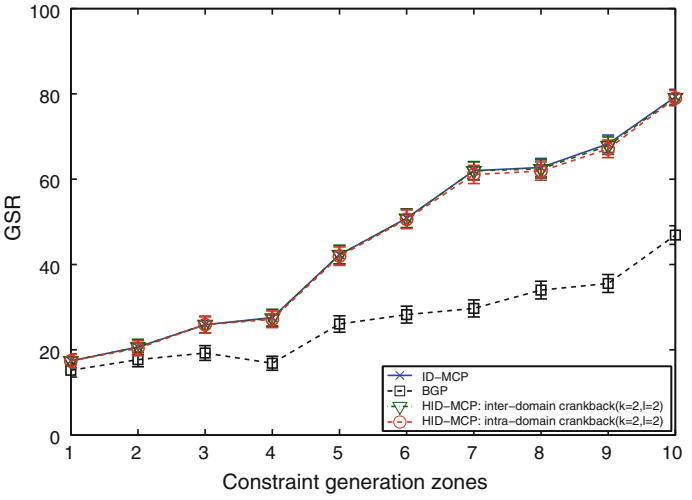

Fig. 10 Comparison of the global success rate of the algorithms in the SYM-CORE topology

closer the success rate to the optimal solution given by ID-MCP. Even so, with low values of l and k ($l = 2$ and $k = 2$), the success rate of the HID-MCP with inter-domain crankback is very close to that of ID-MCP. As expected, the success rate of HID-MCP with intra-domain crankback when $l = 2$ and $k = 2$ is slightly lower than that of HID-MCP with inter-domain crankback with the same parameters. In fact, when the combination procedure fails, HID-MCP with inter-domain crankback executes the on-demand computation procedure starting from the destination domain, while HID-MCP with intra-domain crankback executes it only in the

current domain. Consequently, the quality of the paths computed by the inter-domain crankback approach in each domain is better than the ones computed by the intra-domain crankback approach. Thus, the probability that a feasible path is found using HID-MCP with inter-domain crankback is higher. However, its computational complexity is high compared to HID-MCP with intra-domain crankback, but remains acceptable compared to ID-MCP.

Finally, we evaluate the success rate of algorithms in the SYM-CORE topology. In Fig. 10, we see that the inter-domain crankback based HID-MCP, the intra-domain crankback based HID-MCP, and the ID-MCP algorithm have almost the same success rate, while BGP has the lowest success rate. The low success rate of BGP is explainable as this algorithm does not provide QoS guarantees when routing the requests. The fundamental result deduced from the simulations is that HID-MCP has a GSR very close to that of the exact algorithm while having a low computational complexity.

5 Conclusion

In this paper, we studied the inter-domain QoS routing problem. We proposed a novel inter-domain QoS routing algorithm based on a hybrid computation scheme, named HID-MCP. Our algorithm relies on the PCE architecture. This architecture allows end-to-end path computation subject to multiple QoS constraints, while preserving the confidentiality and the autonomy of the domains. We introduced two different mechanisms for improving the success rate of the HID-MCP algorithm. The first mechanism performs local improvement using an intra-domain crankback, while the second one executes a global improvement using an inter-domain crankback.

The simulation results showed that HID-MCP, with both of the aforementioned approaches, has a success rate very close to the optimal solution while having a low computational complexity. We also compared the average computational time of the path combination and the on-demand computation. This comparison study showed us the advantage of using pre-computation in reducing the computational time of our proposed algorithm.

As future prospects of this work, the stability of the pre-computed paths should be analyzed when they deal with dynamic changes of link state. This allows inferring the validity of the pre-computed paths after an eventual change of the network conditions. Another interesting extension for this work takes into account operators that do not participate in our scheme. In fact, when End-to-End services have to be set over several networks managed by different providers, an agreement has to be established between these providers. Thus we made the assumption that all providers of the alliance have accepted to participate in the proposed inter-domain QoS routing algorithm. In other cases, if one operator does not accept to participate in the computation scheme, we include this as a constraint in the domain sequence selection, in order to avoid this operator's domain.

References

1. Yannuzzi, M., Masip-Bruin, X., Sanchez, S., Domingo-Pascual, J., Orda, A., Sprintson, A.: On the challenges of establishing disjoint QoS IP/MPLS paths across multiple domains. IEEE Commun. Mag. **44**(12), 60–66 (2006)
2. Uludag, S., Lui, K., Nahrstedt, K., Brewster, G.: Analysis of topology aggregation techniques for QoS routing. ACM Comput. Surv (CSUR), **39**(3), paper 7 (2007)
3. Farrel, A., Vasseur, J.P., Ash, J.A.: Path Computation element (PCE)-based architecture. In: IETF RFC 4655 (2006)
4. Vasseur, J.P., Le Roux, J.: Path computation element (PCE) communication protocol (PCEP). In: IETF RFC 5440 (2009)
5. Torab, P., Jabbari, B., Xu, Q., Gong, S., Yang, X., Lehman, T., Tracy, C., Sobieski, J.: On cooperative inter-domain path computation. In: The Eleventh IEEE Symposium on Computers and Communications (IEEE ISCC), Sardinia, Italy (2006)
6. Vasseur, J.P., Zhang, R., Bitar, N., Le Roux, J.: A backward-recursive PCE-based computation (BRPC) Procedure to compute shortest constrained inter-domain traffic engineering label switched paths. In: IETF RFC 5441 (2009)
7. Geleji, G., Perros, H., Xin, Y., Beyenne, T.: A performance analysis of inter-domain QoS routing schemes based on path computation elements. In: High-Capacity Optical Networks and Enabling Technologies (HONET), pp. 146–152 (2008)
8. Orda, A., Sprintson, A.: QoS routing: the precomputation perspective. IEEE Int. Conf. Comput. Commun. **1**, 128–136 (2000)
9. Orda, A., Sprintson, A.: Precomputation schemes for QoS routing. IEEE/ACM Trans. Netw. **11**(4), 578–591 (2003)
10. Ma, Z., Zhang, P., Kantola, R.: Influence of link state updating on the performance and cost of QoS routing in an intranet. In: IEEE Workshop on High Performance Switching and Routing (IEEE HPSR). Dallas, TX, USA (2001)
11. Frikha, A., Lahoud, S., Cousin, B.: Hybrid inter-domain QoS routing with crankback mechanisms. In: The 11th International Conference on Next Generation Wired/Wireless Advanced Networking (NEW2AN), Lecture Notes in Computer Science, vol. 6869, pp. 450–462, St. Petersburg, Russia (2011)
12. Zheng, W., Crowcroft, J.: Quality-of-service routing for supporting multimedia applications. IEEE J. Sel. Areas Commun. **14**(7), 1228–1234 (1996)
13. Knoll, T.: BGP extended community for QoS marking. In: draft-knoll-idr-qos-attribute-11, Work in Progress, IETF (2013)
14. Griffin, D., Spencer, J., Griem, J., Boucadair, M., Morand, P., Howarth, M., Wang, N., Pavlou, G., Asgari, A., Georgatsos, P.: Inter-domain routing through QoS-class planes. IEEE Commun. Mag. **45**(2), 88–95 (2007)
15. Bertrand, G., Lahoud, S., Texier, G., Molnar, M.: A distributed exact solution to compute inter-domain multi-constrained paths. Int. Futur. Lect. Notes Comput. Sci. **5733**, 21–30 (2009)
16. Mieghem, P.V., Kuipers, F.A.: Concepts of exact QoS routing algorithms. IEEE/ACM Trans. Netw. **12**(5), 851–864 (2004)
17. Frikha, A., Lahoud, S.: Pre-computation based Heuristic for Inter-Domain QoS routing. In: The Fourth IEEE International Conference on Advanced Networks and Telecommunication Systems (IEEE ANTS), pp. 61–63. Mumbai, India (2010)
18. Frikha, A., Lahoud, S.: Performance evaluation of pre-computation algorithms for inter-domain QoS routing. In: The 18th International Conference on Telecommunications (ICT), pp. 327–332. Ayia Napa, Cyprus (2011)
19. Esmaeili, M., Xu, F., Peng, M., Ghani, N., Gumaste, A., Finochietto, J.: Enhanced crankback signaling for multi-domain IP/MPLS networks. J. Comput. Commun. **33**(18), 2215–2223 (2010)
20. Molnar, M.: Hierarchies for constrained partial spanning problems in graphs. In: IRISA Technical Report 1900 (2008)
21. Mieghem, P.V., De Neve, H., Kuipers, F.A.: Hop-by-hop quality of service routing. J. Comput. Netw. **37**, 407–423 (2001)
22. Bradford, R., Vasseur, J.P., Farrel, A.: Preserving topology confidentiality in inter-domain path computation using a path-key-based mechanism. In: IETF RFC 5520 (2009)
23. Calvert, K.I., Doar, M.B., Zegura, E.W.: Modelling internet topology. IEEE Commun. Mag. **35**(6), 160–163 (1997)

24. Guichard, J., Le Faucheur, F., Vasseur, J.P.: Definitive MPLS Network Designs. Cisco Press, Indiana (2005)
25. Frikha, A., Lahoud, S.: Hybrid inter-domain QoS routing based on look-ahead information. In: IRISA Technical Report 1946 (2010). http://hal.inria.fr/inria-00463460
26. Frikha, A., Cousin, B., Lahoud, S.: Extending node protection concept of P-cycles for an efficient resource utilization in multicast traffic. In: The 36th IEEE Conference on Local Computer Networks (IEEE LCN), pp. 175–178, Bonn, Germany (2011)
27. Frikha, A., Lahoud, S., Cousin, B.: Candidate-cycle-based Heuristic algorithm for node-and-link protection of dynamic multicast traffic in optical DWDM networks. In: The 26th International Conference on Information Networking (ICOIN), pp. 47–52, Bali, Indonesia (2012)
28. Frikha, A., Lahoud, S., Cousin, B., Molnar, M.: Reliable multicast sessions provisioning in sparse light-splitting DWDM networks using P-cycles. In: The 2012 Annual IEEE Communications Quality and Reliability International Workshop (IEEE CQR), San Diego, USA (2012)

Author Biographies

Ahmed Frikha received his Engineering Diploma in Computer Science from the National School of Computer Science (Tunisia) in 2009. He also received his Master Diploma in Networking from the same school with collaboration of the University of Rennes I (France). In 2012, he obtained the Ph.D. degree in Computer Science from the University of Rennes I, France. He is currently working with France Telecom R&D in Lannion as a research engineer. His main reasearch activity is in network design and survivability, QoS routing, inter-domain routing, traffic engineering and optical transport networks.

Samer Lahoud received the Ph.D. degree in Computer Science from Telecom Bretagne, Rennes. After his Ph.D. he spent 1 year with Alcatel-Lucent Bell Labs Europe working as a research engineer. Since 2007, he has been with the University of Rennes 1, where he is working as assistant professor, and with IRISA of Rennes, where he is taking part in the research activities. His main research activity is in network design, combinatorial optimization and engineering algorithms for communication networks.

Bernard Cousin since 1992, is a Professor of Computer Science at the University of Rennes 1, in France. Bernard Cousin received, in 1987, his PhD degree in Computer Science from the University of Paris 6. He is leading a research group on Advanced Networking. His research interests include next generation Internet, green networking, all-optical networks, dependable networking, high speed networks, traffic engineering, multicast routing, network QoS management, network security and multimedia distributed applications.