Hybrid Inter-Domain QoS Routing with Crankback Mechanisms

Ahmed Frikha, Samer Lahoud, Bernard Cousin

Hybrid Inter-Domain QoS Routing with Crankback Mechanisms

Ahmed Frikha, Samer Lahoud, and Bernard Cousin

University of Rennes 1, IRISA,
35042 Rennes Cedex, France
{ahmed.frikha,samer.lahoud,bernard.cousin}@irisa.fr

Abstract. In this paper we tackle the challenging problem of Quality of Service (QoS) routing in multiple domains. We propose a novel inter-domain QoS routing algorithm named HID-MCP. HID-MCP benefits from two major concepts that ensure high performance in terms of success rate and computational complexity. First, HID-MCP is a hybrid algorithm that combines the advantages of pre-computation and on-demand computation to obtain end-to-end QoS paths. Second, HID-MCP integrates crankback mechanisms for improving the path computation results in a single domain or in multiple domains. Extensive simulations confirm the efficiency of our algorithm on randomly generated topologies.

Keywords: QoS routing, inter-domain routing, crankback mechanisms, pre-computation, on-demand computation

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. Inter-domain QoS routing, also known as Inter-Domain Multi-Constraint Path (ID-MCP) computation problem is one of the primary mechanisms for providing QoS. It consists of computing a path subject to multiple QoS constraints between a source and a destination node of a multi-domain network. Let us introduce some notations to formally define the ID-MCP problem. Let $G(N, E, D)$ denote a network of D domains, N is the set of nodes and E the set of links. Let m be the number of QoS constraints. In our study, we consider only additive metrics, such as cost and delay, without loss of generality [1]. An m-dimensional weight vector is associated with each link $e \in E$. This vector consists of m non-negative QoS weights $w_i(e)$, $i = 1..m$. Let p be a path in the graph $G(N, E, D)$ 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 i^{th} metric of the links of the path p: $w_i(p) = \sum_{e_j \in p}(w_i(e_j))$. Let $\vec{W}(p) = (w_1(p), w_2(p), ..., w_m(p))$ 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 $\vec{C}= (c_1, c_2, .., c_m)$, the Inter-Domain Multi-Constraint Path (ID-MCP) computation problem consists in finding a path p which satisfies $w_i(p) \leq c_i$, $\forall i \in 1..m$. Such a path p is called a feasible path.

The ID-MCP problem is $\mathcal{NP}$-hard [2] and may have zero, one or multiple solutions (feasible paths). Computing such a path requires knowledge of the topology of each domain in the network, as well as the QoS metrics on network links. As the operators can be in competition, information about the internal topology or the available resources in the network is confidential. Hence, computing such a path using a centralized method is a hard task. Currently, the inter-domain routing protocol is BGP. This protocol cannot solve the ID-MCP problem since it does not take into account QoS constraints. Many extensions for BGP are proposed to support QoS routing [4]-[5]. However, the QoS capabilities of these propositions remain limited. Furthermore, solving the ID-MCP problem using a centralized method is a very complex problem. Therefore, the research community has recently been exploring the use of distributed architectures to solve this problem, such as the PCE (Path Computation Element) architecture [6]. Distributing the computation over domains preserves confidentiality of each domain and solves the scaling problem. To our knowledge, few works have been proposed to solve the ID-MCP problem using distributed methods. The algorithm proposed in [7] extends the exact algorithm SAMCRA [3] to an inter-domain level to solve the ID-MCP problem. The drawback of this algorithm is its high complexity. Work in [8] proposes also a promising distributed solution with crankback mechanisms for inter-domain routing. However this solution cannot take into account several QoS metrics.

In this paper, we propose a novel inter-domain QoS routing algorithm, named HID-MCP. HID-MCP is based on a hybrid computation scheme that combines path pre-computation and on-demand path computation. HID-MCP consists of two phases: An offline phase and an online phase. In the offline phase, HID-MCP pre-computes a set of QoS paths. 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 lead always 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 crankback mechanisms improves the computation results and allows computational complexity to be reduced. Besides, our solution relies on a distributed architecture to overcome the limitations related to inter-domain routing. Extensive simulations confirm the efficiency of our algorithm in terms of success rate and computational complexity.

The rest of this paper is organized as follows. In Section 2, we present the concept of the HID-MCP algorithm and its operations. Simulation results are presented in detail in Section 3 and a conclusion is given in Section 4.

2 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 HID-MCP algorithm consists of two phases. In the first phase, named the offline path computation phase, the algorithm executes an intra-domain pre-computation algorithm and computes *look-ahead* information for each domain. The pre-computed intra-domain paths and the *look-ahead* information are stored in a database for later use. The second phase, named online path computation phase, is triggered upon the reception of a QoS request. In this phase, HID-MCP computes an end-to-end path that spans multiple domains and fulfills the QoS constraints. The end-to-end path computation benefits from the stored pre-computed intra-domain paths and the *look-ahead* information to speed up the computational time of the algorithm.

2.1 The Offline Path Computation Phase

The offline computation phase consists of computing in advance a set of intra-domain paths subject to multiple predetermined QoS constraints. It also computes *look-ahead* information at the level of each entry border node of the corresponding domain. In the following, we detail the operations involved in these two computations.

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 the VoIP-based applications. Therefore, our procedure pre-computes for each single QoS metric the path which minimizes the weight corresponding to this metric. For example, it pre-computes the path which minimizes the delay; this path can be useful for the VoIP-based applications.

Let D_q be the considered domain, n_1 be a border node of D_q, n_2 be a node of D_q or an entry border node of a neighbor domain, and m be the number of the QoS metrics, our procedure computes m shortest paths from n_1 to n_2. Each shortest path minimizes a single QoS metric. Hence, from each entry border node n_1 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.

Theorem 1
The complexity of the path segment computation procedure is in $\mathcal{O}(B * m(N \log(N) + E))$, *where* B *is the number of the entry border nodes of the domain.*

Proof: The complexity of this procedure depends on the number of constraints m. For one border node, this procedure is in $\mathcal{O}\left(m(N \log(N) + E)\right)$ corresponding to m times the complexity of Dijkstra, which is $\mathcal{O}((N \log(N) + E))$. Considering the B entry border nodes of the domain, the global complexity is then given by: $\mathcal{O}(B * m(N \log(N) + E))$.

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 maintain a reasonable response time. *Look-ahead* information is inferred from the result of the pre-computation algorithm. Let n_1 be a border node of the domain, and n_2 be a node of the domain or an entry border node of a neighbor domain, and $p^*_{n_1 \mapsto n_2;i}$ denotes the pre-computed shortest path between node n_1 and node n_2 considering the metric i. The weight $w_i(p^*_{n_1 \mapsto n_2;i})$ is the lowest possible path weight between n_1 and n_2. Similarly, let us denote by $\overrightarrow{W^*}_{n_1 \mapsto n_2} = (w^*_1, .., w^*_m)$ the vector where $w^*_i = w_i(p^*_{n_1 \mapsto n_2;i})$. Then, $\overrightarrow{W^*}_{n_1 \mapsto n_2}$ represents the lowest weights to reach n_2 from n_1 for each single metric. We note that a path does not necessarily exist with this 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.

Theorem 2
The complexity of the look-ahead information computation procedure is in $\mathcal{O}(m * N * B)$.

Proof: 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 $m * N$ stored pre-computed paths. Hence, at the level of an entry border node n the complexity of computing the N vectors $\overrightarrow{W^*}_{n \mapsto n_j}$, where $n_j \in N$, is in $\mathcal{O}(m * N)$. Therefore, the complexity of computing the *look-ahead* information for all the entry border nodes of the domain is in $\mathcal{O}(m * N * B)$.

2.2 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. According to the cooperation policy, the service provider computes the best domain sequence that links the source and the destination domain [6]. 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 backward computation according to the PCE architecture. Let $Seq = \{D_1, D_2, .., D_r\}$ denote the selected domain sequence, where D_1 is the destination domain and D_r the source domain. Let d be the destination node and s be 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

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

```
1: q ← 1; H ← φ; reject_request ← false;
2: while (q ≤ r) and not(reject_request) do
3:    H ← Path_combination_procedure(D_q, H, s, d);
4:    if H ≠ φ then
5:       q ← q + 1;
6:    else if intra_domain_crankback then
7:       H ← On_demand_computation(D_q, H, s, d);
8:       if H ≠ φ then
9:          q ← q + 1;
10:      else
11:         reject_request ← true;
12:      end if
13:   else
14:      H ← φ; q ← 1;
15:      while (q ≤ r) and not (reject_request) do
16:         H ← On_demand_computation(D_q, H, s, d);
17:         if H ≠ 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
```

is called (line 3). Operations performed by this procedure are detailed in section 2.2. 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 domain under a novel compact structure named VSPH (Virtual Shortest Path Hierarchy[1]) [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 VSPH is denoted by H in algorithm 1. A virtual path $p_{d\to n}$ is represented in the VSPH by $\left[d, n, \vec{W}\left(p_{d\to n}\right)\right]$, where d is the destination node, n is an entry border node of the upstream domain D_{q+1}, and $\vec{W}\left(p_{d\to n}\right)$ is the weight vector of $p_{d\to n}$.

Combining the pre-computed paths in each domain can lead to an end-to-end path, as detailed in section 2.2. However in some cases, no feasible path is found, i.e. the returned VSPH is empty. We introduce in the following two novel

[1] The hierarchy is a structure which enables the storage of multiple paths between any two nodes [9].

approaches using crankback mechanisms 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 section 2.2.

The intra-domain crankback approach (lines 6-12) executes the on-demand path computation procedure in the current domain, 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. $H = \phi$, the request is rejected.

The inter-domain crankback approach (lines 13-23) executes the on-demand path computation procedure starting from the destination node d. 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.

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 to nodes in the set E, where I is the ingress node set and E 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 which will be sent to the upstream domain. Notes that at the level of the destination domain D_1 there is no received VSPH ($H = \phi$), 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. Figure 1 illustrates an example of path combination with two constraints ($m = 2$) in an intermediate domain D_q.

Theorem 3
The complexity of the pre-computed path combination procedure at the level of an intermediate domain $D_q \in \{D_2, .., D_{r-1}\}$ is in $\mathcal{O}(m^q * B_{max}^2)$, where B_{max} denotes the maximum number of border nodes between two domains.

Proof: 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 $m*B_{max}$ 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 $\mathcal{O}(m^q*B_{max})$. 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 $\mathcal{O}(m^q * B_{max}^2)$.

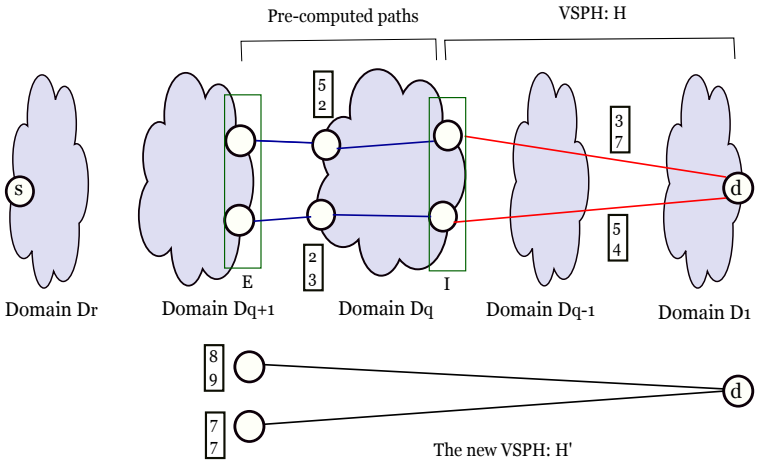

Fig. 1. Combining the pre-computed paths in domain D_q with the VSPH

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 destination domain according to the two aforementioned approaches. We propose a modified version of the TAMCRA algorithm to perform the on-demand computation. Work in [10] 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 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 $\left[d, n, \overrightarrow{W}\left(p_{d\rightarrow n}\right)\right]$ in the VSPH and for each node n_k in E, a weight vector $\overrightarrow{W^*}\left(d \mapsto n_j \mapsto n_k\right)$ 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, $\overrightarrow{W^*}\left(d \mapsto n_j \mapsto n_k\right) = \overrightarrow{W}\left(p_{d\mapsto n_j}\right) + \overrightarrow{W^*}_{n_j\mapsto n_k}$, where $\overrightarrow{W^*}_{n_j\mapsto n_k}$ is given by the *look-ahead* information. Note that the weight vector $\overrightarrow{W^*}\left(d \mapsto n_j \mapsto n_k\right)$ is not necessarily associated to an existing path. Next, we discard infeasible paths from the VSPH. For that, we define a new score for each path p given by: $S(p_{d\mapsto n_j}) = \min_{n_k\in E}\left\{\max_{i\in 1..m}\left(\frac{w_i^*(d\mapsto n_j\mapsto n_k)}{c_i}\right)\right\}$. 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 while meeting the QoS constraints. Then, we classify the remaining paths in 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

Algorithm 2 Path combination procedure (D_q,H,s,d)

$P \leftarrow \{p/p$ pre-computed path in domain $D_q\}$;
$H' \leftarrow \phi$;
if $D_q == D_1$ **then**
  $I \leftarrow \{d\}$;
else
  $I \leftarrow \{n_j/n_j$ leaf node in $H\}$;
end if
if $D_q == D_r$ **then**
  $E \leftarrow \{s\}$;
else
  $E \leftarrow \{n_k/n_k$ entry border node of domain $D_{q+1}\}$;
end if
$Selected\_paths \leftarrow \left\{p_{n_j \rightarrow n_k}/p_{n_j \rightarrow n_k} \in P, n_j \in I, n_k \in E\right\}$;
if $D_q \neq D_1$ **then**
  for $p_{d \rightarrow n_j} \in H$ **do**
    for $p_{n_j \rightarrow n_k} \in Selected\_paths$ **do**
      $\overrightarrow{W}(p_{d \rightarrow n_k}) \leftarrow \overrightarrow{W}(p_{d \rightarrow n_j}) + \overrightarrow{W}(p_{n_j \rightarrow n_k})$;
      if $p_{d \rightarrow n_k}$ is feasible **then**
        Add $\left[d, n_k, \overrightarrow{W}(p_{d \rightarrow n_k})\right]$ to H';
      end if
    end for
  end for
else
  for $p_{d \rightarrow n_k} \in Selected\_paths$ **do**
    if $p_{d \rightarrow n_k}$ is feasible **then**
      Add $\left[d, n_k, \overrightarrow{W}(p_{d \rightarrow n_k})\right]$ to H';
    end if
  end for
end if
Return H';

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 $\overrightarrow{W}(p_{d \mapsto n_j})$ and we execute the TAMCRA algorithm starting from node n_j to reach the nodes in E. 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.

Theorem 4
The complexity of the on-demand path computation procedure at the level of an intermediate domain is in $\mathcal{O}(l(k * N \log(k * N) + k^3 * m * E))$.

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

$temp\_paths \leftarrow \phi$; $feasible\_paths \leftarrow \phi$;
$\Psi \leftarrow \left\{ \vec{W^*} \,/\, \vec{W^*} \text{ look-ahead information in domain } D_q \right\}$;
if $D_q \neq D_1$ **then**
  $I \leftarrow \{n_j / n_j \text{ leaf node in } H\}$;
  if $D_q == D_r$ **then**
    $E \leftarrow \{s\}$;
  else
    $E \leftarrow \{n_k / n_k \text{ entry border node of domain } D_{q+1}\}$;
  end if
  $L \leftarrow \left\{ \vec{W^*}_{n_j \rightarrow n_k} \in \Psi, n_j \in I, n_k \in E \right\}$;
  for $\left[d, n_j, \vec{W}\,(p_{d \rightarrow n_j}) \right] \in H$ **do**
    for $\vec{W^*}_{n_j \rightarrow n_k} \in L$ **do**
      $\vec{W^*}\,(d \rightarrow n_j \rightarrow n_k) \leftarrow \vec{W}\,(p_{d \rightarrow n_j}) + \vec{W^*}_{n_j \rightarrow n_k}$;
    end for
    $S(p_{d \rightarrow n_j}) \leftarrow \min_{n_k \in E} \left\{ \max_{i \in 1..m} \left(\frac{w_i^*(d \mapsto n_j \mapsto n_k)}{c_i} \right) \right\}$;
    if $S(p_{d \rightarrow n_j}) \leq 1$ **then**
      Add $[n_j, \vec{W}\,(p_{d \rightarrow n_j}), S(p_{d \rightarrow n_j})]$ to $temp\_paths$;
    end if
  end for
  if $temp\_paths \neq \phi$ **then**
    $Selected\_paths \leftarrow l$ shortest paths having the lowest S in $temp\_paths$
  else
    $H' \leftarrow \phi$;
    Return H'
  end if
  for $\vec{W}\,(p_{d \rightarrow n_j}) \in Selected\_paths$ **do**
    Initialize n_j with the weight vector $\vec{W}\,(p_{d \rightarrow n_j})$
    Execute TAMCRA in D_q starting from n_j toward evry border node of domain D_{q+1}
    Add the obtained feasible paths to $feasible\_paths$
  end for
else
  Execute TAMCRA in D_1 starting from d
  Add the obtained feasible paths to $feasible\_paths$
end if
Extract H' from $feasible\_paths$
Return H'

Proof 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 node is less or equal to l, the complexity of this operation is in $\mathcal{O}(l(k * N \log(k * N) + k^3 m * E))$, corresponding to l times the complexity of TAMCRA.

3 Simulation and analysis

In this section, we evaluate the performance of our novel algorithm HID-MCP by comparasion with the exact on-demand algorithm ID-MCP introduced in [7]. This algorithm has the best success rate, i.e. no other algorithm can have a success rate higher than that of ID-MCP, because ID-MCP finds a feasible path whenever a such path exists. However, the complexity of executing ID-MCP in each domain corresponds to the complexity of the SAMCRA algorithm given by: $\mathcal{O}((K_{max} * N \log(K_{max} * N) + K_{max}^3 m * E))$, where $K_{max} = \min(\exp(N-2)!, \frac{\prod_{i=1}^{m} c_i}{\max_j c_j})$ [10]. This complexity is very high comparing with that of our proposed algorithm. The simulations are performed using a network of three domains where each domain is built based on Waxman's model with 50 nodes in each domain. The probability that two nodes of the network are connected by an edge is expressed in [11]. We associate with each link two additive weights generated independently following a uniform distribution [10, 1023]. The QoS constraints are also randomly generated according to the following: Let p_1 and p_2 denote the two shortest paths which minimize the first and the second metric, 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. The problem is not $\mathcal{NP}$-Hard outside Z, i.e. either infeasible or trivial. As shown in figure 2, we divide this space into ten zones Z_i, $i = 1..10$ and we browse the space 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. We evaluate the algorithms based on the following performance

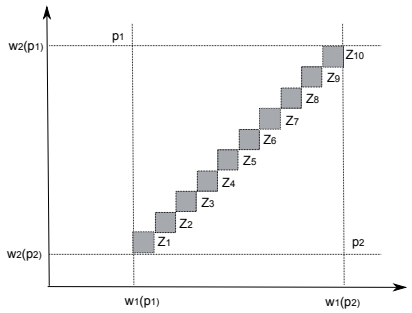

Fig. 2. Constraint generation zones for $m = 2$

criteria:

- *GSR*: the global success rate given by the ratio of the number of the requests for which a feasible path is found and the total number of QoS requests.
- *CSR*: the efficiency of the combination procedure at a specific domain given by the ratio of the number of requests where the combination procedure is successful (e.g. leading to at least one feasible segment between the destination and the border nodes of the upstream domain) and the total number of received requests.

In the following, each figure measures the variation of one performance metric according to the constraint generation zones Z_i, $i \in 1..10$, with a 95% confidence interval. We focus on the success rate of the combination procedure (CSR) in a

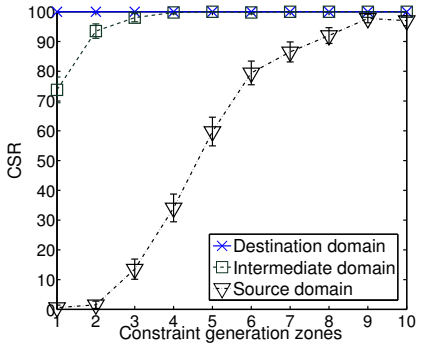

Fig. 3. Success rate of the combination procedure in each domain.

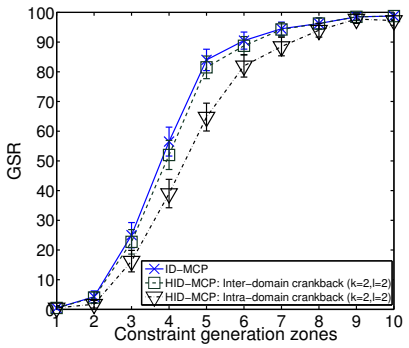

Fig. 4. Comparison of the global success rate of the algorithms.

given domain. The complement of CSR corresponds to the percentage of executions of the on-demand computation procedure. Figure 3 illustrates the variation of the CSR in each domain according to strictness of the QoS constraints. In the destination domain, the combination procedure is always successful. In the intermediate domain the success rate of this procedure is high and equals 100% when constraints are not very strict. However, we note that the CSR in the source domain is low, specifically when the constraints are strict. Nonetheless, this procedure performs well when the constraints are less strict. From this figure, we deduce that the probability of executing the on-demand computation is high only in the source domain when the constraints are very strict. This proves that the global empirical complexity of HID-MCP remains reasonable.

Figure 4 illustrates the variation of the global success rate (GSR) of HID-MCP with intra-domain crankback mechanism, HID-MCP with inter-domain crankback mechanism, and the exact algorithm ID-MCP, according to the strictness of the QoS constraints. 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 section 2, k is a parameter of TAMCRA and l is the maximum number of paths selected from the VSPH. As expected, the success rate of HID-MCP with intra-domain crankback when $l = 2$ and $k = 2$ is lower than that of HID-MCP with inter-domain crankback with the same parameters, especially in the middle of the constraint generation space. 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. The fundamental result deduced from this figure is that HID-MCP has a slightly lower global success rate compared to the exact algorithm, while having a very low complexity.

4 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. We introduced two different mechanisms for improving the success rate. The first mechanism performs local improvement using an intra-domain crankback, while the second mechanism executes a global improvement using an inter-domain crankback. Extensive simulations showed that HID-MCP with both of the aforementioned approaches provides a high success rate and maintains a low complexity time. In particular, the inter-domain crankback improvement-based HID-MCP has a success rate very close to that of the exact solution, while the intra-domain crankback improvement-based HID-MCP provides lower computational complexity. This gives operators the choice to execute either of the approaches, depending on the computation policy and the priority of the request.

References

1. A. Frikha, and S. Lahoud, Hybrid Inter-Domain QoS Routing based on Look-Ahead Information, *IRISA*, Tech. Rep. 1946, 2010.
2. Z. Wang and J. Crowcroft, Quality-of-Service Routing for Supporting Multimedia Applications, *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228-1234, 1996.
3. P. Van Mieghem and F.A Kuipers, Concepts of exact QoS routing algorithms, *IEEE/ACM Transactions on Networking*, vol. 12, no. 5, pp. 851-864, October 2004.
4. T. Knoll, BGP Extended Community Attribute for QoS Marking, *draft-knoll-idr-qos-attribute-02, work in progress, IETF*, 2009.
5. D. Griffin, J. Spencer, J. Griem, M. Boucadair, P. Morand, M. Howarth, N. Wang, G. Pavlou, A. Asgari, and P. Georgatsos, Interdomain routing through QoS-class planes, *IEEE Commun. Mag.*, vol. 45, no. 2, pp. 88-95, February 2007.
6. A. Farrel, J.P. Vasseur, J.A Ash, Path Computation Element (PCE)-Based Architecture, *IETF RFC 4655*, August 2006.
7. G. Bertrand, S. Lahoud, G. Texier, and M. Molnar, A Distributed Exact Solution to Compute Inter-domain Multi-constrained Paths, *The Internet of the Future, Lecture Notes in Computer Science*, vol. 5733, pp. 21-30, 2009.
8. M. Esmaeili, F. Xu, M. Peng, N. Ghani, A. Gumaste and J. Finochietto, Enhanced Crankback Signaling for Multi-Domain IP/MPLS Networks, *Computer Communications*, vol. 33, no. 18, pp. 2215-2223, 2010.
9. M. Molnar, Hierarchies for Constrained Partial Spanning Problems in Graphs, *IRISA*, Tech. Rep. 1900, 2008.
10. P. Van Mieghem, H. De Neve, and F. A. Kuipers, Hop-by-hop quality of service routing, Computer Networks, 37, 407–423, 2001.
11. K. I. Calvert, M.B. Doar, and E.W. Zegura, Modelling Internet Topology. IEEE Communications Magazine, vol. 35, no. 6, pp. 160–163, 1997.