Inter-Domain Path Computation with Multiple Constraints

Gilles Bertrand, Samer Lahoud, Miklos Molnar, Géraldine Texier

I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1902

INTER-DOMAIN PATH COMPUTATION WITH MULTIPLE CONSTRAINTS

GILLES BERTRAND , SAMER LAHOUD , MIKLÓS MOLNÁR , GÉRALDINE TEXIER

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Inter-Domain Path Computation with Multiple Constraints

Gilles Bertrand[*] , Samer Lahoud[**] , Miklós Molnár[***] , Géraldine Texier[****]

Systèmes communicants
Projet At-Ren

Publication interne n1902 — August 2008 — 40 pages

Abstract: The interest for providing services with performance guarantees across domain boundaries has driven recent technical solutions allowing the computation of constrained inter-domain paths. The computation of optimal paths subject to multiple constraints is an NP-complete problem for which efficient exact solutions exist in the intra-domain case. However, these solutions cannot be used for inter-domain path computations, because of confidentiality and scalability constraints. Thus, the present paper investigates the problem of computing inter-domain paths subject to multiple constraints. We describe the information exchanges required between the domains for optimal computations. We extend existing algorithms for inter-domain computations, and describe new heuristics approximating exact solutions. We propose an exact solution, named pID-MCP, allowing the precomputation of path segments in the domains. After proving the correctness and the complexity of exact solutions, we evaluate by simulation the performance of the algorithms and the heuristics proposed. Our solutions allow the computation of inter-domain paths subject to multiple constraints without breaking the confidentiality constraints of the domains. Moreover, the heuristics can be used in large-scale networks.

Key-words: Inter-domain routing, multi-constrained path computation

(Résumé : tsvp)

[*] Institut TELECOM ; TELECOM Bretagne ; RSM
[**] IRISA - Université de Rennes 1
[***] IRISA - INSA de Rennes
[****] Institut TELECOM ; TELECOM Bretagne ; RSM

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes
Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Calcul de chemins inter-domaines soumis à plusieurs contraintes

Résumé : La volonté de permettre la fourniture de service avec des garanties de performances au delà des frontières des domaines à engendré des solutions techniques récentes qui permettent le calcul de chemins inter-domaines contraints. Le calcul de chemins optimaux soumis à de multiples contraintes est un problème NP-complet pour lequel des solutions exactes existent dans le cas intra-domaine. Cependant, ces solutions ne sont pas applicables pour le calcul de chemins inter-domaines, en raison de contraintes de confidentialité et d'extensibilité. C'est pourquoi le présent article étudie le problème du calcul de chemins inter-domaines soumis à de multiples contraintes. Nous décrivons les échanges d'information nécessaires entre les domaines afin de permettre des calculs optimaux. Nous étendons les algorithmes intra-domaines existant au calcul de chemins inter-domaines et nous décrivons des heuristiques approchant des solutions exactes. Nous proposons une solution exacte, appelée pID-MCP, qui permet le précalcul de segments dans les domaines. Après avoir prouvé la correction et la complexité des solutions exactes, nous évaluons par simulation les performances des algorithmes et des heuristiques proposés. Nos solutions permettent le calcul de chemins inter-domaines soumis à plusieurs contraintes, tout en respectant les contraintes de confidentialité des domaines. De surcroît, les heuristiques peuvent être utilisées dans des réseaux de grande taille.

Mots clés : Routage inter-domaine, calcul de chemin soumis à plusieurs contraintes

1 Introduction

Computing inter-domain paths subject to multiple constraints has long been considered as impracticable because of scalability and confidentiality constraints. Thus, in the Internet, inter-domain routing is mainly based on connectivity and domain policies. The computation of constrained inter-domain paths has become a highly regarded topic in the last decade, to provide more control on routing to the operators and to enable the delivery of services with quality of service (QoS) across domain boundaries. There are several propositions for extending the inter-domain routing protocol of the Internet, the Border Gateway Protocol (BGP), with QoS capabilities (see for instance [3, 10, 22, 34, 35]). However, modifying BGP is difficult, because this protocol is widely deployed.

Recent initiatives propose solutions based on Multiprotocol Label Switching (MPLS) for allowing the computation of constrained inter-domain paths without breaking the confidentiality constraints of the domains [11, 14]. The present paper pertains to this recent stream of work. Our goal is to investigate the problem of computing inter-domain paths following a predetermined domain sequence and subject to multiple constraints. We call this problem Inter-MCP. We show that the present solution to the problem Inter-MCP [5, 32] sometimes forbids to solve it exactly, because the tree structure used for exchanging information between the domains does not include enough data. Thus, we describe the information that the domains must exchange for solving the problem Inter-MCP exactly.

The computation of a path subject to multiple constraints related to independent additive link-weights is an important, but difficult, problem. This problem arises for example when computing a path satisfying the constraints of an application (*e.g.,* short mouth-to-ear delay for telephony) and the constraints of the network (*e.g.,* traffic engineering objectives). It is called the multi-constrained path (MCP) problem. The problem MCP is NP-complete [33]. Nevertheless, recent work shows that most of its instances in telecommunication networks can be solved exactly [17, 29]. Several exact solutions are described in the literature (*e.g.,* SAMCRA [27], H_MCOP [16]). These algorithms solve the problem MCP inside a domain.

This work explains why the solutions of the intra-domain MCP problem cannot be used to solve the problem Inter-MCP. Consequently, we adapt these solutions. We prove the resulting algorithms and compute their worst-case complexity. In particular, the present paper describes an exact solution allowing the *precomputation* of path segments by the domains crossed by a path. This solution has two desirable properties, considering that the problem Inter-MCP is NP-complete. First, it divides the path computation operations into per-domain computations. Second, it reuses the path segments computed, independently of the sequence of domains crossed. We derive heuristics approaching with a tunable accuracy the exact algorithms proposed. We evaluate the performance of the exact algorithms and the heuristics in a realistic scenario, as well as in extreme cases, which allows us to illustrate the trade-offs among the solutions proposed and improves our understanding of the problem considered.

The remainder of the paper is organized as follows. In Section 2, we introduce the technological context of this study. Section 3 describes the MCP problem and related intra-domain solutions. In Section 4, we discuss the limitations of present solutions for inter-

domain path computations. Section 5 introduces the extension of exact intra-domain MCP algorithms for problem Inter-MCP, proves the exact algorithms proposed and derives efficient heuristics approaching them. Finally, we evaluate our solutions to the problem Inter-MCP in Section 6.

2 Inter-Domain Routing and Path Computation

2.1 General Scenario and Assumptions

In this work, we consider the path computation problem across multiple domains and subject to multiple constraints. For the purpose of this document a domain consists of an Autonomous System (AS) that resides under a single path computation responsibility. In each domain, the nodes that interconnect to other domains are called Border Nodes (BN). Path computation across multiple domains typically occurs when the source and the destination nodes of the path belong to different domains. The entity that is responsible for the path computation can be the source node, the intermediate BNs, or dedicated computation elements such as the Path Computation Element (PCE) [8]. Several computation models exist based on the different computation entities. A path computation entity uses its own vision of the state of the domains in order to output a path that connects the source and the destination nodes and satisfies the given constraints. An objective function may also be optimized during the computation procedure. Thus, the efficiency of one computation model depends on the amount of information that is available on the computation entities. This is a key issue for the inter-domain path computation due to the limited information that is spread outside the domain borders [9]. Particularly, resource and topology information is considered to be confidential for each domain and may be hidden from other domains.

In this work, we assume that the path computation problem follows a preliminary step that determines the sequence of traversed domains. Precisely, we consider computing a path between source and destination nodes, subject to multiple constraints, and traversing a pre-computed sequence of domains. This assumption follows a compromise between the quality and the complexity of the global solution. On the one hand, the computation of the sequence of domains and the computation of the path itself can be thought as a joint optimization problem. This approach should capture both the constraints and the objectives of the two components. For instance, computing the sequence of domains is mainly subject to policy and business constraints whereas computing the path is subject to performance constraints. This increases the complexity of the joint optimization and makes the search for efficient solutions a big challenge. On the other hand, the computation of the sequence of domains can be performed in advance of the path computation. This approach enables to capture the specificity of the two computation tasks, though it does not guarantee to reach global optimal solutions. In the following, we consider that the sequence of domains is pre-computed outside of the scope of our study and focus on the subsequent path computation problem. Thus, in all the following occurrences, an optimal path is implicitly defined for a pre-computed sequence of domains.

In order to illustrate the general path computation problem across multiple domains, let us consider the scenario in Figure 1. In this figure, node A in domain X is trying to communicate with node B in domain Z. A path, subject to a set of constraints, should be computed to enable nodes A and B to communicate. According to the assumptions presented above, a preliminary sequence of domains is computed, i.e. domains X,Y, and Z, fulfilling some policy or business constraints. Hence, the path computation problem consists of computing a path from A to B traversing successively domains X, Y, and Z. This computation can be performed by the source node A if it has sufficient topology or resource information on the three domains. Otherwise, the path computation can be distributed on the intermediate border nodes, or offloaded onto dedicated per-domain entities such as PCEs. For instance, when the domains X, Y, and Z belong to a single management organization, one computation entity may have sufficient information to perform end-to-end efficient computations. Otherwise, when these domains belong to concurrent organizations, topology and resource information may not be spread outside the domain border. Thus, the computation result is less efficient unless some collaboration is introduced between per-domain computation entities.

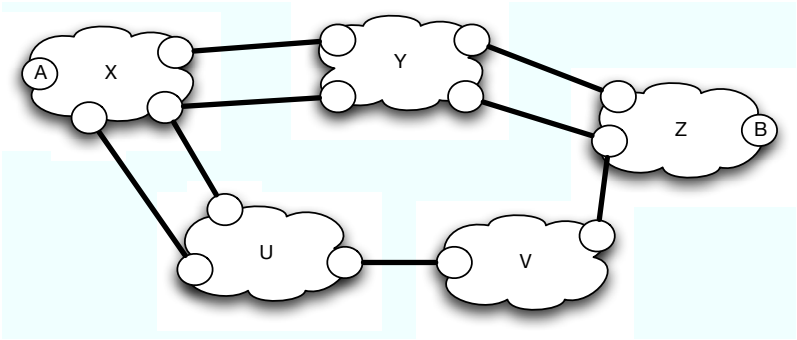

Figure 1: General scenario for path computation across multiple domains

2.2 Domain Visibility

In order to compute a constrained path crossing multiple domains, a computation entity needs to gather information about each domain crossed. The optimality of the computation depends on the amount of gathered information. Thus, three scenarios are defined depending on the domain visibility level: multi-domain visibility, partial visibility and local domain visibility [9].

First, in a multi-Domain visibility scenario, the computation entity may have sufficient visibility of the topology for all of the domains crossed by the constrained path. Thus, the computation entity may compute and provide the entire path. However, achieving this information scenario may not be practical given the requirement that a domain should never

advertise resources and topological information outside its boundaries.

Second, in a partial visibility scenario, the computation entity has full information about its own domain, and has information about the connectivity between domains as well as resource availability across other domains. However, it does not have full visibility of the topologies inside other domains. Consequently, the computation entity is not able to provide, a fully specified strict explicit path from source to destination. It can still supply some useful information such as the path to reach its domain boundary, abstract nodes to replace intermediate domains or a loose hop for the destination.

Third, in a local domain visibility scenario, the computation entity has full visibility of its own domain and connectivity information only as far the BN of the local domain. In this case the computation entity computes an explicit path that comprises explicit hops to reach the local BN and a loose hop identifying the destination.

In this work, we consider a local domain visibility scenario, where each computation entity has full visibility of its own domain. This choice reflects the limitations imposed by network operators on topology and resource information related to their domain. This information is considered confidential, thus is not spread outside the domain borders. However, the local domain visibility scenario compromises the optimality of the computation due to lack of information. One solution to remedy this problem consists of implementing a collaboration between computation entities in each domain. Such collaboration should not violate the aforementioned inter-domain limitations, though it may restore the optimality of the computation. The level of collaboration enables to introduce different computation models as defined in the following section.

2.3 Path Computation Models

Depending on the localization of the computation entities, two basic computation models may be defined: per-domain path computation and PCE-based computation.

2.3.1 Per-domain Path Computation

Per-domain computation [30] applies where the inter-domain path cannot be or is not determined at the source node. This is most likely to arise owing to visibility limitations. The path through each domain, possibly including the choice of exit point from the domain, must be determined within the domain.

The per-domain path computation technique involves the computation of individual path segments in every intermediate domain without the sharing of any path information from other domains. The complete path is obtained by concatenating the path segments that are computed for every domain. Figure 2 illustrates a simple scenario of per-domain path computation. The source node computes the first path segment to a selected exit BN. The second path segment is then computed for the second domain by this BN to the next nearest exit BN.

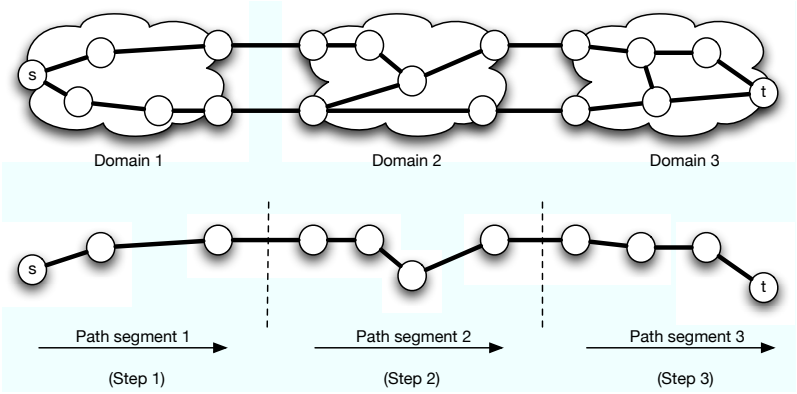

Figure 2: Per-domain path computation steps

2.3.2 PCE-Based Computation

As standardized in [8], the PCE is an entity responsible for computing an inter-domain path upon receiving a request from a path computation client (PCC). In most cases, the source node serves as the PCC. When a new path computation request arrives, the PCC forwards the path computation request to a selected PCE using PCC-to-PCE communication [31]. A PCE may compute the end-to-end path itself if enough topology and resource information is available to it. Furthermore, the PCE architecture provides mechanisms for the resolution of path computation requests when an individual PCE does not have sufficient visibility. For example, a PCE may cooperate with other PCEs to determine intermediate loose hops, or a full explicit path.

Backward Recursive PCE-based Computation Path computation across domains is particularly difficult when the visibility of a source node is restricted only to the local domain. As introduced in [32], the Backward Recursive PCE-Based Computation (BRPC) utilizes multiple PCEs to compute the shortest inter-domain constrained path along a determined sequence of domains. This technique also preserves confidentiality across domains, an important requirement when domains are managed by different service providers [9]. The BRPC procedure is backward, as the path is computed starting from the destination domain to the source domain. It is recursive since the same basic sequence of steps is repeated for every intermediate domain lying between the source and destination.

Consider a pre-computed sequence of D domains denoted by V_1 to V_D where V_1 represents the source domain and V_D the destination domain. BRPC uses the concept of a virtual shortest path tree (VSPT). VSPT(i) with $1 \leq i \leq D$ denotes the multipoint-to-point (MP2P) tree formed by the set of constrained shortest paths from the entry BNs of domain i to the destination node. Each link of tree VSPT(i) represents a shortest path. The BRPC procedure is as follows. For the destination domain V_D, VSPT(D) is computed from the

list of shortest paths from all the entry BNs to the destination node. This is illustrated in Figure 3. VSPT(D) is then sent to the PCEs in the previous domain $D-1$ where it is concatenated to their topology and resource information database. The updated database is then used by the PCEs in domain $D-1$ to compute VSPT($D-1$). This sequence of operations is repeated until the source domain and, subsequently, the source node of the path are reached, as shown in Figure 3.

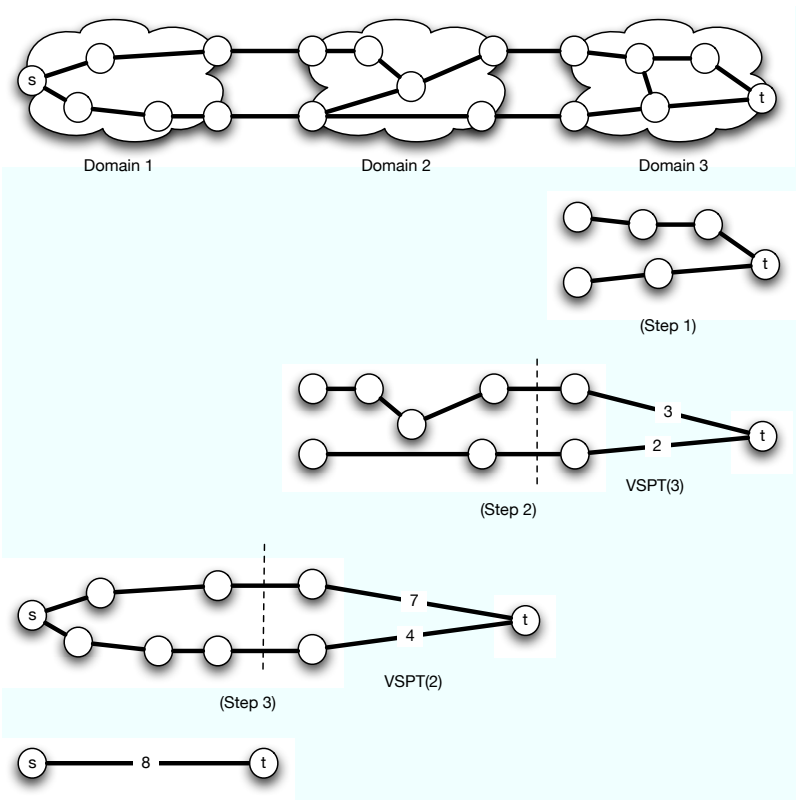

Figure 3: BRPC steps for computing a path between s and t with the minimum number of hops

3 Path Computation with Multiple Constraints

3.1 The Multi-Constrained Path Problem

The general problem of computing paths subject to several additive constraints (*e.g.,* a maximum propagation delay D and a maximum number of hops H) is called the *multi-*

constrained path (MCP) problem. The problem of finding a feasible multi-constrained path that is optimal with respect to a length function is called the *multi-constrained optimal path* (MCOP) problem [18]. MCP and MCOP problems consider additive constraints only, because, first, constraints related to bottleneck metrics (*e.g.,* bandwidth) can be easily treated by edge pruning and, second, positively-valued multiplicative constraints can be transformed into additive constraints by using a logarithm function. The MCP and MCOP problems are NP-complete if the additive metrics are considered to be independent [33].

The MCP and MCOP problems can be defined more formally, as follows. We consider a network represented by a directed graph $G = (V, E)$ where V is the set of vertices or nodes and E the set of edges or links. Each edge $l \in E$ is associated with K additive weights[1] $w_k(l) \in \mathbb{R}^+,\ k = 1..K$. We assume that at least one of the K weights associated with an edge l is different from zero. We define a path as a finite sequence of adjacent edges denoted by $\mathbf{p} = \big((s,i),(i,j),\ldots,(k,t)\big)$. The set of paths between two nodes $s \in V$ and $t \in V$ is denoted as $P_{s\to t}$. Each path $\mathbf{p}$ is associated with K weights denoted as $w_k(\mathbf{p}) \in \mathbb{R}^+$ and defined by $w_k(\mathbf{p}) \equiv \sum_{l\in\mathbf{p}} w_k(l),\ k = 1..K$. Moreover, a length function $c(\mathbf{p})$ is associated with the path $\mathbf{p}$ subject to K constraints related to the weights considered and denoted as $W_k,\ k = 1..K$. The most appropriate path length function, defined in Reference [27], expresses the critical value of the path with respect to the constraints:

$$c(\mathbf{p}) = \max_{i=1..K} \left(\frac{w_i(\mathbf{p})}{W_i} \right) \tag{1}$$

Intuitively, the expression for $c(\mathbf{p})$ means that the path that is the furthest from violating the constraints is selected. With this path length measure, a path $\mathbf{p}$ is feasible if and only if $c(\mathbf{p}) \leq 1$. Hereafter, we formally define the two fundamental problems that are used in this work, namely MCP and MCOP.

Problem. *MCP – Multi-Constrained Path Problem [16, 18] Given a source s and a destination t, K constraints $W_k,\ k = 1..K$, find a path $\boldsymbol{p} \in P_{s\to t}$ such that $w_k(\boldsymbol{p}) \leq W_k$, for $k = 1..K$.*

Any solution of the problem MCP is called a *feasible path.*

Problem. *MCOP – Multi-Constrained Optimal Path Problem [16,18] Given a source s and a destination t, K constraints W_k, find a feasible path $\boldsymbol{p}^* \in P_{s\to t}$ such that for every other feasible path $\boldsymbol{p} \in P_{s\to t}$, $c(\boldsymbol{p}^*) \leq c(\boldsymbol{p})$.*

3.2 Exact and Approximated Intra-Domain Solutions

An overview of multi-constrained path computation methods is provided in Reference [18]. Exact solutions for the MCP problem include brute-force solutions (*e.g.,* depth first search with backtracking). Several brute-force algorithms including search-space reduction mechanisms have been described in the literature (*e.g.,* SAMCRA [27], H_MCOP [16]). Due to

[1] In this document, the set of the strictly positively-valued real numbers and the set of the positively-valued real numbers are denoted as $\mathbb{R}^{+*}$ and $\mathbb{R}^+$, respectively.

the computational intractability of the MCP and MCOP problems, the time complexity of brute-force algorithms grows exponentially, in the worst-case. Thus, several approximation algorithms have been described. For example, Jaffe's algorithm [13] is an approximation algorithm addressing the MCP problem with two constraints. This algorithm determines two positive multipliers d_1 and d_2, and optimizes the length function $d_1 w_1(\mathbf{p}) + d_2 w_2(\mathbf{p})$. According to [18], Jaffe's algorithm can be extended to an arbitrary number of constraints. In Reference [17, 29], Kuipers and Van Mieghem assert that the NP-complete behavior of the MCP problem emerges only in specially constructed graphs with edge weights carefully chosen. Thus, exact QoS routing algorithms seem feasible in practice.

A standard strategy for computing a path subject to several additive constraints involves optimizing a single measure of the path length. This measure of the path length can be defined as a composition function of the original weights considered. If the weights considered are positively-correlated, a linear composition function $\mathbf{p} \rightarrow \sum_{k=1..K} d_k w_k(\mathbf{p})$, with $d_k \in \mathbb{R}^+$ may be used (*e.g.,* Lagrangean-based linear composition, Jaffe's algorithm [13]). However, in the general case, non-linear composition functions like the one defined in Equation 1 (*e.g.,* TAMCRA [6], SAMCRA [27], H_MCOP [16]) provide better results according to an evaluation performed by Korkmaz and Krunz in Reference [16].

The difficulty of computing a shortest path with respect to a non-linear length function resides in the property that segments of an optimal path are not necessarily optimal. Thus, it is not sufficient to remember a single shortest path segment to each intermediate node, as is done when considering a single linear path length function (see the counter-example in Figure 4 explained in Section 4). A guaranteed optimal solution for the MCP problem remembers all intermediate segments. The best existing algorithms try to keep track of as few intermediate path segments as possible. This implies memorizing only k-shortest paths to intermediate nodes (*e.g.,* TAMCRA [6, 7]), memorizing non-dominated feasible paths only (*e.g.,* SAMCRAv1 [27] and H_MCOP [16]) or using predictions of end-to-end cost to avoid memorizing path segments of infeasible paths (*e.g.,* H_MCOP, MPMP [23], SAMCRAv2 [29]).

4 Problems of the Current Approaches

The purpose of this work is to devise solutions for the computation of inter-domain paths subject to several constraints. In this section, we show, first, that MCP and MCOP algorithms cannot be applied for inter-domain path computations and, second, that current inter-domain approaches are not applicable for exact MCP computations.

4.1 Limitations of MCP Algorithms

In communication networks, domains (*e.g.,* ASes) usually are independent, preserve the confidentiality of their topology and compute paths autonomously. Due to these constraints, multi-constrained path computation solutions which do not consider the segmentation of the network into domains are not acceptable for inter-domain path computations. For

example, SAMCRA relies on a queue containing shortest path segments to all intermediate nodes. Thus, all domains involved in the computation of a path have access to information about path segments in other domains, which is usually unacceptable for confidentiality and scalability reasons. Consequently, we investigate methods for multi-constrained inter-domain path computation that fulfill autonomy, scalability and confidentiality constraints.

4.2 Limitations of Inter-Domain Path Computation Methods

With the per-domain method, the end-to-end path is a concatenation of segments computed independently by the domains. Thus, it cannot guarantee to compute an optimal (shortest) constrained inter-domain path. Furthermore, it cannot be efficiently used to compute a set of inter-domain diversely routed TE LSPs [32].

The current version of the Internet draft describing the BRPC procedure states that with this procedure, *optimal* end-to-end paths across a predetermined sequence of domains can be computed [32]. In this assertion, optimal means that the path selected is the same as if the computations were performed in a single domain. However, when several additive constraints are considered, the VSPT structure described in Reference [32] sometimes forbids to find a feasible path when it exists, because it contains at most a *single* shortest path from each entry border node to the destination (tree structure). Figure 4 illustrates this problem. In this figure, the weight vectors are indicated close to the edge to which they apply. We compute a path between s and t, considering the constraints $(W_1, W_2)^T = (4, 4)^T$, which means that both weights $w_1(\mathbf{p})$ and $w_2(\mathbf{p})$ of the path $\mathbf{p}$ selected must be lower or equal to 4. The shortest path computation procedure inside each domain considers the path length measure used by SAMCRA [27] and H_MCOP [16] (Equation 1). A single shortest path (solid line) from the domain border router a to the destination t is included in the VSPT. Its length is given by $\max\left(\frac{2}{4}, \frac{2}{4}\right) = \frac{1}{2}$. The shortest path from s to t using this segment has a length equal to $\frac{5}{4} > 1$. Thus, the BRPC procedure is not able to find a feasible path, whereas, the optimal path from s to t has a length equal to $\frac{4}{4} = 1$ and thus, is feasible. But this path cannot be found using the BRPC procedure because it includes a non-shortest path segment (dotted line, length $\frac{3}{4}$) between a and t going through b and this path segment is typically not included into the VSPT.

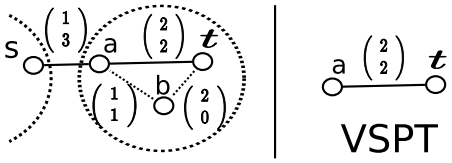

Figure 4: An example of non optimal multi-constrained path computation with the BRPC procedure

Neither the per-domain method, nor the BRPC method computes inter-domain paths subject to multiple constraints exactly. Thus, we investigate a more accurate methods

approximating the optimal solution. Due to the aforementioned limitations of the per-domain method, we focus on extending the BRPC method.

5 Extensions for Inter-Domain Path Computation

5.1 Information Required for Exact Inter-Domain MCP Computations

We focus on describing algorithms solving the MCP problem for inter-domain path computations. With this purpose, we extend the BRPC procedure [32] for enabling the computation of optimal inter-domain paths subject to multiple constraints. The counter-example in Figure 4 demonstrates that, for exact computations, the VSPT structure described in Section 2.3.2 must sometimes include more than a single shortest path per entry border node[2]. Despite its name, such extended VSPT is not necessarily a tree. We evaluate the performance gain brought by including into the VSPT several paths from each ingress to the destination. The extension of the VSPT considered does not require any further change of the PCE protocol than the ones introduced in Reference [32] and is fully backward-compatible with the BRPC procedure. We assume that the PCE protocol supports the additive constraints considered in this work.

As explained in Section 3.2, segments of shortest paths with respect to a non-linear length function are not necessarily optimal. Thus, the VSPT must be able to store more than a single shortest path between each ingress and the destination. However, during the computation of a multi-constrained path not every path must be memorized at intermediate nodes. First, as positive weights are considered, infeasible paths segments do not have to be memorized [7].

Lemma. If a path segment $\mathbf{p}_1$ is infeasible with weights $w_k(\mathbf{p}_1), k = 1..K$, then the weights $w_k(\mathbf{p})$ of any path $\mathbf{p}$ using $\mathbf{p}_1$ are such that $w(\mathbf{p}) \geq w(\mathbf{p}_1)$, thus $\mathbf{p}$ is infeasible.

Second, *dominated* paths do not have to be memorized [7, 29], which means that every path $\mathbf{p}$ such that there is a path $\mathbf{p}'$ verifying $w_k(\mathbf{p}') \leq w_k(\mathbf{p}), \forall k = 1..K$ and such that there is a $i \in [1..K]$ for which $w_i(\mathbf{p}') < w_i(\mathbf{p})$ does not need to be considered.

Lemma. If a path segment $\mathbf{p}_1$ with weights $w_k(\mathbf{p}_1), k = 1..K$ is dominated by a path $\mathbf{p}_2$ with weights $w_k(\mathbf{p}_2)$, then any path $\mathbf{p}$ using $\mathbf{p}_1$ is dominated by the path $\mathbf{p}'$ obtained by replacing $\mathbf{p}_1$ by $\mathbf{p}_2$ in $\mathbf{p}$, thus $\mathbf{p}$ is non-optimal.

Figure 5 illustrates the concept of *dominance*. In this figure the paths represented by circles are non-dominated, whereas the one represented by a square ($\mathbf{p}_4$) is dominated by $\mathbf{p}_2$, because $\mathbf{p}_2$ is better than $\mathbf{p}_4$ for all weights considered ($w_k(\mathbf{p}_2) \leq w_k(\mathbf{p}_4)$, for $k = 1..K$) and strictly better with respect to at least one weight ($w_1(\mathbf{p}_2) < w_1(\mathbf{p}_4)$).

Figure 6 extends the example presented in Figure 4 assuming that all feasible non-dominated paths are included in the VSPT. Thus, the source s learns both paths to the

[2]Henceforth, we call the entry border nodes of a domain the *ingresses* of the domain.

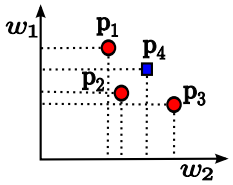

Figure 5: Non-dominance with K=2

destination t and is able to select the shortest end-to-end path. Considering the same constraint vector as before: $(4,4)^T$, s selects the path segment with cost $(3,1)^T$ leading to a feasible path.

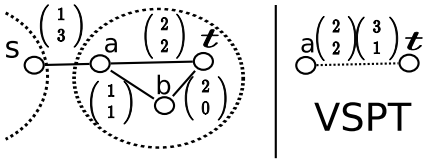

Figure 6: The BRPC procedure with an extended VSPT

5.2 Extended MCP Algorithms

In this section, we focus on the extension of MCP algorithms for inter-domain path computation. Let us define the problem Inter-MCP as follows:

Problem. *Inter-MCP — Given a sequence $S = V_1, \ldots, V_D$ of domains, a source node $s \in V_1$, a destination node $t \in V_D$ and a set of K constraints W_k, $k = 1..K$, find a feasible path from s to t following the sequence of domains S.*

Any solution to the problem Inter-MCP must compute a multi-constrained path from the source $s \in V_1$ to at least one of the nodes of V_1 connected to a node in V_2. This means that to solve the problem Inter-MCP, a NP-complete problem must be solved. Thus, the problem Inter-MCP is NP-complete.

In the following, we start by detailing the adaptation steps of SAMCRA and H_MCOP for Inter-MCP and introduce a new algorithm called ID-MCP. Then, we detail the algorithmic steps of ID-MCP and illustrate it by a step-by-step example.

5.2.1 Adaptation of the Principles of SAMCRA and H_MCOP

ID-MCP is a direct adaptation of existing algorithms to solve the Inter-MCP problem. ID-MCP uses the same principles as SAMCRA [27] and H_MCP [16] for reducing the search-space. We have adapted these principles to the specificities of inter-domain path

computations. We have not implemented the use of cost predictions (*look-ahead* [29]) for excluding segments of non-feasible path in the first version of our algorithm. ID-MCP performs shortest path computations with respect to the cost function of SAMCRA given by $c(\mathbf{p}) = \max_{i=1..K} (\frac{w_i(\mathbf{p})}{W_i})$ for a path $\mathbf{p}$ and returns feasible paths ($c(\mathbf{p}) \leq 1$).

Unlike SAMCRA or H_MCOP, ID-MCP is able to compute *inter-domain* paths while preserving the confidentiality of topology and resource information, as detailed in Section 4. ID-MCP adopts the principle of the BRPC method described in Reference [32] for segmenting end-to-end path calculations into per-domain segment computations. This segmentation implies that, in each domain, shortest paths from several nodes (the border nodes connected to the previous domain) to the destination must be computed. However, SAMCRA and H_MCOP do not compute a multi-constrained path from multiple sources to one destination. Particularly, H_MCOP is designed to compute paths between two nodes only, namely the source and the destination [16], whereas, SAMCRA can be used to compute paths from one source to several destinations [29]. Thus, we adapt SAMCRA and reverse the direction of its computations. Hence, ID-MCP computes the paths backward from the single destination considered to multiple sources, namely the border nodes connected to the upstream domain.

In the algorithmic steps, we adopt the terminology of SAMCRA: the path computation is based on a queue. In this queue, a path is marked *gray* if it is selected and *black* if it is discarded. The paths that are not marked are said to be *white*. SAMCRA terminates when either a path attached to the destination has been marked gray, or the queue contains no more white elements. ID-MCP must provide all feasible non-dominated path segments to the previous domain. Thus, ID-MCP continues its computations even if all source nodes of an intermediate domain have been associated with a path marked gray. Consequently, in each domain, ID-MCP terminates only when the queue contains no more white elements. If a single path is required, the path computation procedure in the source domain can be stopped as soon as a path from the source has been marked gray.

5.2.2 ID-MCP: Algorithm and Example

A solution to the problem Inter-MCP, called Inter-Domain-MCP (ID-MCP), is presented below, using the notations of Figure 7. In this algorithm, we assume that at most one link exists between any two nodes. The input of the algorithm is a request (source node, destination node, source domain, destination domain, vector of constraints) and a sequence of domains. The sequence of domains crossed by the path is denoted as $S = V_1, \ldots, V_D$. During the operations of ID-MCP in each domain, the topology and the state of the links of the domain considered, as well as the state of the links connected to this domain are supposed to be known. The path segment computation procedure in each domain relies on a queue containing the paths memorized. Queue elements are of the following form: $(node, predecessor, weights, color)$. The output of the computations in each domain is a VSPT containing one or several non-dominated virtual paths from the ingresses of the domain to the destination. In the general case each domain may return only a subset of

the paths computed, according to its policies (*e.g.,* a maximum financial cost), but this problem is outside of the scope of this paper.

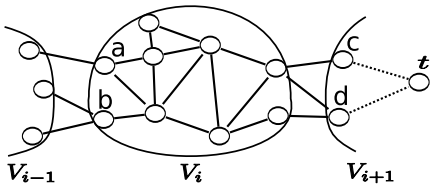

Figure 7: Problem Inter-MCP for an intermediate domain

ID-MCP is described in Algorithm 5.1. Computations are performed from the destination domain V_D to the source domain V_1 as is performed by the BRPC procedure. In the destination domain, the queue is initialized with a white element corresponding to a path to the destination t with a cost equal to zero (line 3). In any other domain V_i, the queue is initialized with the paths extracted from the leaves of the VSPT transmitted by the downstream domain V_{i+1} and a virtual topology is constructed. This virtual topology includes the virtual nodes and virtual links of the VSPT, aggregated to the graph of V_i (line 5 of Algorithm 5.1, step 1 of the example in Figure 8). In each domain, the algorithm loops while there is at least one white element in the queue (line 7). During each execution of this loop, a white path with minimum cost is extracted from the queue (line 8) and marked gray (line 9). Then, this path is relaxed (line 10), which means that the neighbors of the node to which this path is attached are visited, and the paths from the neighbors to the destination t are memorized if and only if they are feasible and non-dominated by the ones already in the queue. If some paths in the queue are dominated by a path newly discovered, then, these paths are marked black. Finally, the VSPT is extracted from the queue and transmitted to V_{i-1}, which means that the paths from the ingresses of V_i are transmitted to the upstream domain V_{i-1} in accordance with the policy of domain V_i. In the source domain, extracting the VSPT means that one or several end-to-end paths are retrieved from the final VSPT.

An example appears in Figure 8 and helps to understand the principle of ID-MCP. For more clarity, in this example, only two additive metrics are considered (K=2) and they take integer values. An intermediate domain V_i receives a VSPT containing three feasible, non-dominated virtual paths to the destination t. V_i concatenates this VSPT with its topology (step ①) and initializes the computation queue. The paths extracted from the VSPT are initially white. Then, the algorithm enters the loop (steps ② to ⑨). The path computation operations end when no more white elements are present in the queue. Finally, the VSPT to be transmitted to the upstream domain is extracted and forwarded.

ID-MCP considers the *end-to-end* constraints for the path segment calculations in each domain to guarantee the optimality of the paths computed. Splitting the constraints between the domains would make more intermediate paths be detected as non-feasible and thus, decrease the complexity of the path computation operations. However, the optimality could not be guaranteed anymore.

Algorithm 5.1 ID-MCP

```
1: for all domain i from D to 1 do
2:   if V_i = V_D then
3:     queue ← {node:dest, predecessor:∅, w:0, color:white}
4:   else
5:     {queue, virtual_topology} ← concatenate(vspt_{i+1}, topology_i)
6:   end if
7:   while ∃ white element ∈ queue do
8:     element_min = extract_min(queue)
9:     element_min.color ← gray
10:    queue ← relax(element_min)
11:  end while
12:  vspt_i ← extract_vspt(queue)
13: end for
```

5.3 Proof and Complexity of ID-MCP

Theorem 1. *Termination:* the algorithm ID-MCP terminates.

Proof. The algorithm ID-MCP marks at least one white path[3] during each iteration of its main loop. As the number of paths in each domain is finite and the algorithm stops when all paths have been marked or before, the algorithm terminates. □

Theorem 2. *Correctness:* the algorithm ID-MCP computes all feasible non-dominated inter-domain paths along the sequence of domains $V_1, \ldots, V_D$ considered.

Proof. The algorithm ID-MCP is essentially a brute-force algorithm with search-space reduction mechanisms. Thus, we only have to prove that no solution is excluded from the search-space. The proof of SAMCRAv1 in Reference [27] can be adapted for ID-MCP. Thus, we know that in the destination domain V_D all non-dominated feasible paths from the ingresses to the destination t are computed. The same proof also implies that in each intermediate domain, all non-dominated feasible paths from the ingresses to the destination are computed *in the virtual graph* resulting from the aggregation of the VSPT with the graph of the domain. Thus, we only have to show that the paths computed in this virtual graph correspond to the non-dominated feasible paths from the ingresses to the destination t in the real graph.

A finite recursion shows that ID-MCP computes all non-dominated feasible paths from the source s to the destination t. We assume that all feasible non-dominated paths from every ingress of the next domain $V_{i+1}, i < D$ to the destination t are included in VSPT_{i+1}. In domain V_i, ID-MCP concatenates the topology of V_i with VSPT_{i+1}. We know that all feasible non-dominated paths between the ingresses of V_i and the destination t follow

[3]The path that is selected and the paths that are dominated by the paths explored during the relaxation stage.

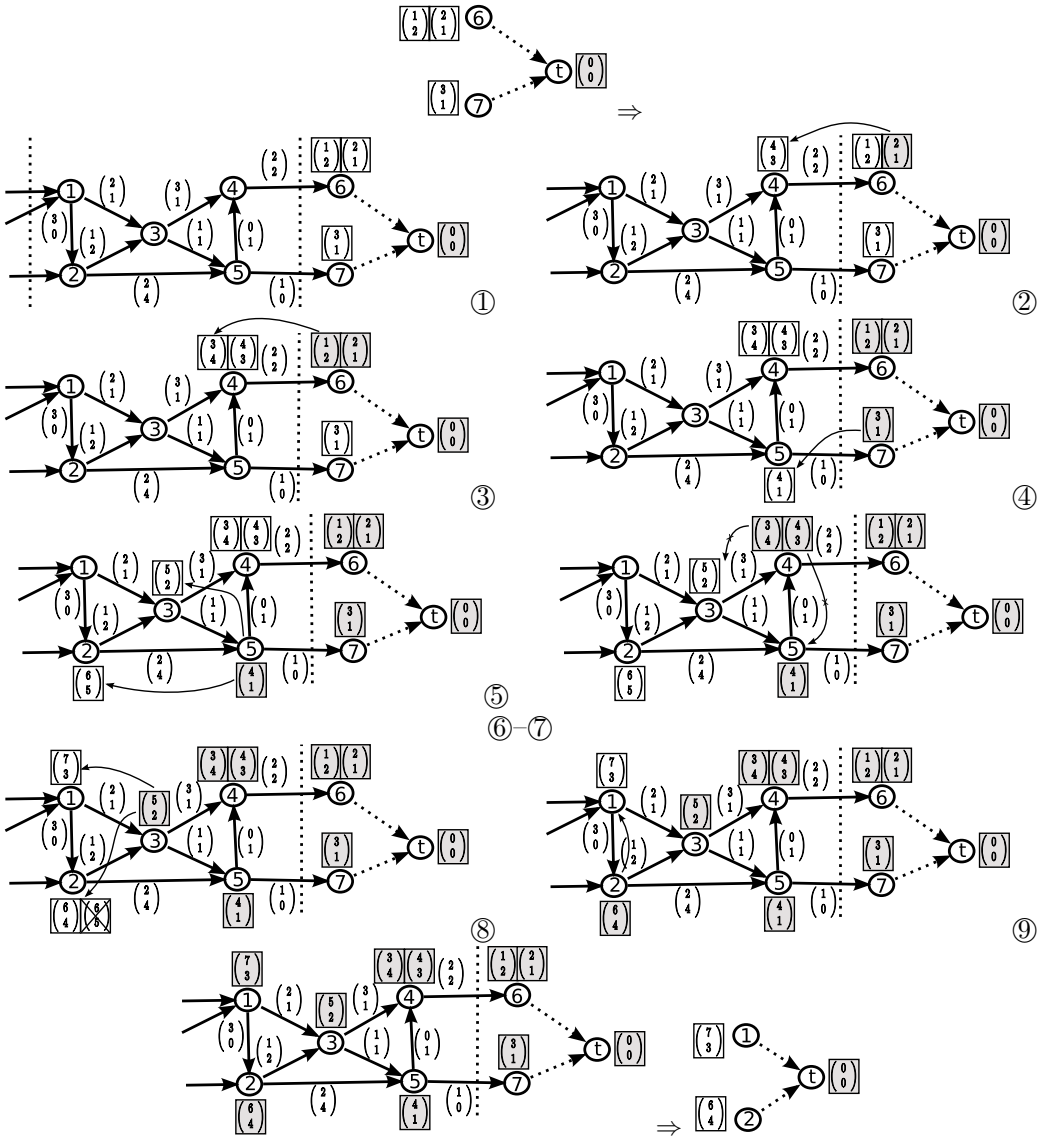

Figure 8: Example of inter-domain multi-constrained path computation with $W_1 = 15$ and $W_2 = 15$

non-dominated feasible path segments from the ingresses of V_{i+1} to t[4]. Thus, no feasible

[4]see the lemmas in Section 5.1

non-dominated path segment between an ingress of V_i and t is excluded by considering only the path segments in VSPT_{i+1} instead of all the paths from domain V_{i+1} to t, hence, ID-MCP computes all feasible, non-dominated paths from the ingresses of V_i (or the source of the request if it belongs to V_i) to the final destination of the paths. □

Theorem 3. *Worst-case complexity:* the worst-case time complexity of the algorithm ID-MCP is in $O\big(D\alpha^2 K(\alpha N + E + N^2)\big)$, where α is the maximum number of paths memorized for a node[5], K is the number of additive link metrics considered, D is the number of domains in the sequence considered, N and E are the maximum number of nodes and the maximum number of edges in a domain of the sequence of domains considered.

Proof. The worst-case complexity of the operations of ID-MCP inside a domain can be shown to follow the same expression as the one of SAMCRAv1 which is in $O\big(\alpha N \log(\alpha N) + \alpha^2 KE\big)$ according to [27], where α is the maximum number of paths memorized for a node in the queue, N is the number of nodes, K the number of additive link metrics considered and E the number of edges of the single domain considered. However, with ID-MCP the domains are extended by the contents of the VSPT. Thus, the number of nodes to consider is the number N_i of nodes of the domain plus the number of ingresses Ingr_{i+1} of the next domain V_{i+1} that are represented in the VSPT, plus the destination. Moreover, the number of links to consider is the number E_i of edges of the domain, plus the number of inter-domain links between V_i and V_{i+1}, which is bounded by $N_i \cdot \text{Ingr}_{i+1}$ and, in addition, the number of virtual edges extracted from the VSPT, which is bounded by $\alpha \cdot \text{Ingr}_{i+1}$.

Hence, the worst-case complexity of the operations of ID-MCP inside a domain is in $O\big(\alpha(N_i + \text{Ingr}_{i+1} + 1) \cdot \log(\alpha(N_i + \text{Ingr}_{i+1} + 1)) + \alpha^2 K(E_i + (\alpha + N_i) \cdot \text{Ingr}_{i+1})\big)$. This expression can be simplified into $O\big(\alpha N \cdot \log(\alpha N) + \alpha^2 K(\alpha N + E + N^2)\big)$ by introducing the notations of the theorem and noting that $\text{Ingr}_{i+1} \leq N$ and $N_i \leq N$. It can be further simplified into $O\big(\alpha^2 K(\alpha N + E + N^2)\big)$ by noticing that $\alpha N \log(\alpha N)$ is in $O(\alpha^2 N^2)$. The operations of SAMCRA are repeated in every domain along the sequence of domains, thus the complexity is multiplied by the number D of domains crossed. In addition, the operations for the aggregation of the VSPT with the graph of the domain (at most $\alpha \cdot \text{Ingr}_{i+1}$ edges are added), the initialization of the queue (at most $\alpha \cdot \text{Ingr}_{i+1} + 1$ elements are added) and the extraction of the final VSPT (at most $\alpha \cdot N_i$ elements of the queue are considered) must be taken into account. However, all these terms are in $O(\alpha N)$, which leads to the complexity of the theorem. □

5.4 Enhancement of ID-MCP

5.4.1 Parallelization and Complexity Reduction (pID-MCP)

ID-MCP does not allow the computation of path segments in parallel in the domains crossed or the precomputation of path segments. Moreover, ID-MCP performs several times similar computations if several non-dominated paths from the same ingress are learned from the

[5] A bound for α is derived in Reference [27].

VSPT. However, we show that it would be sufficient to compute the non-dominated path segments from each ingress in V_i to each ingress of V_{i+1} mentioned in the VSPT and to add the costs of the virtual edges from the ingresses of V_{i+1} to t in a final stage. Splitting these two stages allows the domains crossed to perform some path computations operations in parallel or in advance (precomputation). The segments computed can then be combined in a later stage. We have implemented an algorithm implementing these operations. We name this algorithm pID-MCP.

pID-MCP computes the non-dominated path segments from each ingress of a domain to the ingress of the next domain. Thus, the non-dominance comparisons are applied to the paths depending on their destination. For example, a path with weights $(4,4)^T$ and destination ingress A, does not dominate a path with weights $(5,5)^T$ and destination ingress B, attached to the same node. Consequently, the elements of the queue must include their destination. Their form evolves to: $(node, predecessor, destination, weights, color)$. In addition, initial weights equal to zero are considered for the elements retrieved from the VSPT received from the downstream domain. Thus, the operations performed by pID-MCP for extracting the VSPT from the queue are different from the operations performed by ID-MCP. The algorithm pID-MCP needs to combine the segments computed in the domain with the virtual edges of the previous VSPT, and to compute the end-to-end weights of the paths computed.

Consider the example in Figure 9 and Figure 10. In this example, ID-MCP performs an approximately two times higher number of calls to its sub-functions compared to pID-MCP. However, the VSPT transmitted by both algorithms is the same.

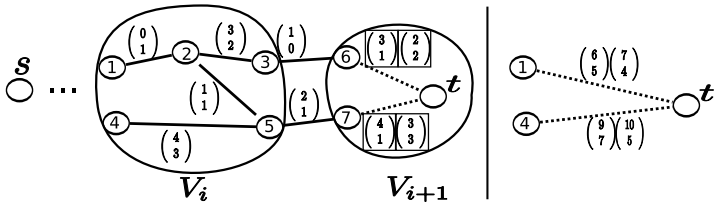

Figure 9: Initial (left) and last (right) state of ID-MCP

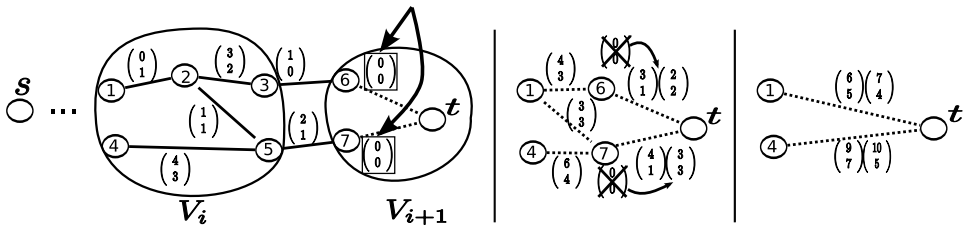

Figure 10: Initial (left), last (middle) state and VSPT (right) of pID-MCP

pID-MCP terminates for the same reason as ID-MCP.

Theorem 4. pID-MCP computes all feasible non-dominated inter-domain paths along the sequence of domains $V_1, \ldots, V_D$ considered

Proof. We know that any feasible non-dominated path is made up of feasible non-dominated segments. Thus, the feasible non-dominated paths from the ingresses of V_i to t are made up of one feasible non-dominated segment from an ingress of V_i to an ingress of V_{i+1} and one virtual path of VSPT_{i+1}. pID-MCP computes all feasible non-dominated segments from the ingresses of V_i to the ingresses of V_{i+1} represented in the VSPT. Moreover, the VSPT includes all feasible non-dominated segments from the ingresses of V_{i+1} to t. Consequently, in each domain V_i, pID-MCP computes all feasible non-dominated paths from the ingresses of V_i to the destination t. □

Theorem 5. *Worst-case complexity:* the worst-case time complexity of the algorithm pID-MCP is in $O\big(D\alpha^2 K(E+N^2)\big)$

Proof. The worst-case time complexity of pID-MCP can be computed as in the proof of Theorem 3. The only differences with this proof are, first, that the number of edges to consider for the operations of SAMCRA is the number E_i of edges of the domain, plus the number of inter-domain links between V_i and V_{i+1}, which is bounded by $N_i \cdot \text{Ingr}_{i+1}$ and, in addition, the number of virtual edges extracted from the VSPT, which is now bounded by Ingr_{i+1}, instead of $\alpha \cdot \text{Ingr}_{i+1}$. Second, the operations for combining the segments computed in domain V_i with the edges of VSPT_{i+1} have to be taken into account. The time complexity of these operations is in $O(\alpha^2 \cdot \text{Ingr}_i \cdot \text{Ingr}_{i+1})$ which is in $O(\alpha^2 N^2)$. □

The complexity of pID-MCP can be significantly higher than the one of ID-MCP in two specific situations. First, pID-MCP performs the non-dominance comparisons depending on the destination of each element, thus, in the worst-case, at least one feasible non-dominated path per destination is memorized for each intermediate node. Consequently, if one domain is connected to the next domain through many ingress nodes, a large number of paths may be memorized by pID-MCP. However, in realistic network configurations, the number of nodes connected in a domain connected the another domain is rather limited. Second, pID-MCP considers initial weights equal to zero, thus, less paths may be detected as non-feasible considering the end-to-end constraints.

We expect pID-MCP to decrease the maximum number α of paths memorized for a node compared to ID-MCP and thus to bring a reduction of the computational complexity. α is usually lower with pID-MCP than with ID-MCP, because fewer elements are considered in the initial queue for each domain. Thus, compared to ID-MCP, pID-MCP improves the time and space complexity in certain network configurations.

pID-MCP can easily be adapted to compute segments in each domain *in parallel*, then combine the elements computed and eventually determine the best path. If path segments are computed in parallel in the domains, the worst-case complexity is in $O\big(K\alpha^2(E+N^2)\big)$ with the same notations as in Theorem 5. Alternatively, each domain could compute and

advertise[6] its elements (the feasible non-dominated paths from its ingresses to the ingresses of its neighboring domains in an alliance of domains) for a few constraint-vectors corresponding to common classes of service. If we assume, that the link weights are relatively static (so that the elements advertised remain valid for a reasonable time). Then, the elements advertised could later be used by the source domain for computing an end-to-end multi-constrained path. The intra-domain paths advertised should not depend on the sequence of domains crossed, and, thus, not depend on the weights of the segments from the next domain to the destination, which explains why we consider initial weights equal to zero. This way, a single execution of pID-MCP in a domains provides all feasible non-dominated path segments for any request inside a class of service (same constraint vector W), independently of the domain sequence crossed.

The proof of Theorem 5 can easily be extended to show that, if pID-MCP is used inside a domain A to precompute paths from the entry BNs of A to the entry BNs of M neighboring domains, then the worst-case complexity of the precomputation operations inside A is in $O\big(D\alpha^2K(E+M\cdot N^2)\big)$.

If the domains compute the path segments sequentially with pID-MCP[7], then the initial weights considered for each ingress a in the VSPT of the next domain (VSPT(i+1)) should be the ones in Equation 2, where E_a is the set of elements associated with the ingress a in VSPT(i+1).

$$\big(\min_{x\in E_a} w_1(x),\ldots,\min_{x\in E_a} w_K(x)\big)^T \tag{2}$$

This higher initial cost has a positive effect on the computational complexity, because it decreases the number of feasible paths. Exact algorithms like pID-MCP compute all feasible non-dominated paths, which is interesting for balancing the traffic load, especially when the elements advertised by every domain remain valid for relatively long periods and the paths computed use different resources.

5.4.2 Heuristics

In Reference [27], Van Mieghem, De Neve and Kuipers study the problem of connectionless QoS routing inside a single domain and simulate a MCP computation algorithm on fictitious random topologies. They note that, in their simulations, in about 90% of the cases, a multi-constrained optimal path can be found by a distributed algorithm memorizing a single shortest path only and called *hop-by-hop distance based only* (HbHDBO) routing[8]. Moreover, they note that the cost of the paths computed with HbHDBO routing is close to the optimal. Hence, we describe several heuristics that are expected to provide both excellent performance with respect to the quality of the path computed and a significant reduction of the computational complexity.

[6] or memorize

[7] As mentioned before, the complexity of pID-MCP is expected to be significantly lower than the one of ID-MCP in certain network configurations.

[8] More precisely, they find that the exact solution is found by HbHDBO routing in 89.4% of the cases.

We introduce three simplified algorithms. The first is called kID-MCP, k=1. It resembles ID-MCP except for the relaxation of the paths (line 10 of Algorithm 5.1). kID-MCP, k=1 explores the paths of the neighboring nodes and keeps at most one path per node: a single shortest path (SSP) with respect to c. Consequently, kID-MCP, k=1 cannot guarantee exact MCP computations. In each domain, kID-MCP, k=1 performs similar operations as the reverse Dijkstra's algorithm. kID-MCP, k=1 is equivalent to ID-MCP with $\alpha = 1$, thus kID-MCP, k=1 terminates and the worst-case complexity of kID-MCP, k=1 is in $O\big(DK(E + N^2)\big)$, with the same notations as in Theorem 3. The expressions for the complexity of ID-MCP and pID-MCP include a dependency on α^2, which indicates that reducing the maximum number α of elements memorized for a node may improve the computational complexity of these algorithms[9], significantly.

In fact, as its name indicates, kID-MCP, k=1 is a special case of the second algorithm named kID-MCP. kID-MCP resembles ID-MCP except that, for each node, at most k elements attached to this node can be memorized in the computation queue. Thus, kID-MCP is an adaptation of TAMCRA [7]. The heuristic kID-MCP represent and intermediate cases between the algorithm memorizing every non-dominated feasible path (ID-MCP) and kID-MCP, k=1 (at most one path is memorized for each node). Similarly, we define the algorithm kpID-MCP, which is similar to pID-MCP except that at most k paths are memorized for each node. For kID-MCP and kpID-MCP, the parameter limiting the value of α must be carefully selected. Contrary to the first intuition, kpID-MCP with $k > 1$ can provide worst results than kID-MCP, k=1 in certain situations, which does not reduce its interest as an approximation of pID-MCP. Consider the example in Figure 11. In this example, kpID-MCP does not find any feasible solution considering the constraints $W = (5, 5)^T$, whereas kID-MCP, k=1 finds a feasible solution with cost $(4, 1)^T + (1, 1)^T = (5, 2)^T$. This type of situation can be partially avoided by considering the initial weights in Equation 2.

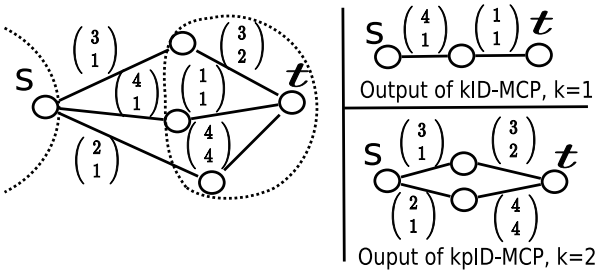

Figure 11: Example where kID-MCP, k=2 provides no feasible results whereas kID-MCP, k=1 provides a feasible solution. $W = (5, 5)^T$

The third algorithm is called oID-MCP. It resembles ID-MCP except that it uses the original VSPT structure. This means that with oID-MCP each domain computes all feasible non-dominated segments in the virtual graph, but includes at most a single path from each

[9] And, typically, also their spatial complexity.

ingress to the destination into the VSPT. Thus, oID-MCP cannot guarantee exact MCP computations. oID-MCP terminates for the same reason as ID-MCP and it can be proved that its worst-case complexity is in $O\big(D\alpha^2K(E+N^2)\big)$. Both pID-MCP and oID-MCP consider at most a single path per ingress of the downstream domain and their worst-case complexity is similar. However, compared to pID-MCP, oID-MCP degrades the quality of the paths computed. Thus, we have not studied oID-MCP any further.

6 Simulation and Results

6.1 Methodology

6.1.1 Topologies

We have compared the performance of the algorithms presented in previous sections through simulations. The performance of most routing algorithms is closely related to the properties (*e.g.,* topology, metrics) of the network to which they are applied. Here, intuitively, the size (number of domains, number of nodes in each domain) and the connectivity (domain degrees, node degrees) of the topologies considered have a strong effect on the performance of the routing algorithms compared. Thus, the topologies considered in the simulations have been carefully selected. A difficulty arising in this selection is that the PCE architecture is not expected to be deployed in the whole Internet but only in small sets of ASes, and the structure of these sets of ASes is not publicly available. Therefore, we have evaluated the performance of our algorithms on two types of topologies. First, we have used the following *lattice topologies* to assess the performances of the algorithms in extreme configurations. Lattice topologies represent an extreme case for QoS routing algorithms, as described in Reference [17]. These topologies are square grids in which the top-left node is the source and the bottom-right node is the destination of a request. We consider two topologies based on lattices.

- LatticeFM(N,D) (Full-Mesh) represents a worst-case for the complexity of the algorithms[10]. This topology is a chain of D identical domains, every domain is a grid made up of N nodes and $E = 2\sqrt{N}(\sqrt{N}-1)$ undirected links. Every node of every domain is connected to every node of the previous domain, as well as to every node of the next domain in the chain.

- LatticeSL(N,D) (Single Link) is similar to LatticeFM(N,D) except that only the top-left node of each domain is connected to the bottom-right node of the previous domain.

Second, we have used more realistic topologies to assess the applicability of our algorithms to real networks.

[10]The absolute worst-case topology considering the number of paths is a topology in which any node is connected to any other node inside or outside its domain.

- A realistic backbone topology including measurements of the networks of eight major Internet service providers in the USA. We refer to this topology as REAL(8). This topology appears in Reference [19] and is based on topologies measured by Rocket-Fuel [24], essentially.
- A fictitious inter-area topology named SYM-CORE, which was used in Reference [5].

6.1.2 Link Weights

Random weights have been added to the edges of the topologies considered, except for the topology REAL(8), which includes delay estimations. Most simulations have been realized with two weights (K=2). We have not taken any assumption about the kind of additive weights considered: for example, the weights may be related to the number of hops ($w_l = 1,\ \forall l \in E$), to the link propagation delay, to the inverse of the capacity of the links, or to a measure of the reliability of the links. In addition, we have considered additive constraints only: for instance, we have not considered the effect of the capacity of the links, because bandwidth is a bottleneck metric that can easily be treated by edge pruning. We consider that every edge has enough bandwidth to serve the requests considered. This assumption seems reasonable if the largest requests are rejected by an admission control mechanism or if the network considered is overdimensioned.

Link weight information is usually advertised through a routing protocol that allocates a fixed number of bits for this information. Thus, link weights can be considered to be bounded by a function of the size in bits of the corresponding field of the routing messages. The bounds on the link weights have been chosen arbitrarily, without any loss of generality, as weights can be considered to be scaled. The number of possible values of the weights has an effect on the number of non-dominated paths and, thus, on the complexity, consequently, we have considered weights with a relatively fine granularity.

The work in Reference [28] suggests that, in the Internet, the link weights can be modeled using a uniform distribution. In addition, several important papers in the area adopt uniformly distributed weights (*e.g.,* [7, 16, 27]). Hence, we have generated uniformly distributed link weights. We have assumed that the weights of inter-domain links follow the same distribution as the ones of intra-domain links. Constant weights (*e.g.,* number of hops) represent a special case of uniformly distributed weights.

The correlation of the weights $w_k,\ k = 1..K$ is known to have an effect on the complexity of MCP algorithms [17]. Thus, we have performed both simulations with independent link weights and simulations with positively or negatively-correlated link weights. When we consider correlated weights, we assume that the correlation of the weights is the same in all domains crossed, so that the effect of correlation is visible on end-to-end path computations. The method used for generating correlated weights is inspired by Reference [16]: to generate K positively-correlated weights in the interval $]A, B]$, we partition the interval $]A, B]$ into $\left\{I_0 \equiv \left]A, A + \frac{B-A}{2}\right], I_1 \equiv \left]A + \frac{B-A}{2}, B\right]\right\}$. We generate a random weight w_1. If w_1 is in I_0 then we generate $w_k,\ k = 2..K$ in the interval I_0, else we generate $w_k,\ k = 2..K$ inside the interval I_1.

6.1.3 Constraints

The value of the end-to-end constraints considered has an effect both on the complexity of ID-MCP and of pID-MCP and on their performance compared to kID-MCP (k=1). For example, if $\mathbf{p}_1$ is a shortest path with respect to w_1 and $\mathbf{p}_2$ a shortest path with respect to w_2, then the constraints W_1 and W_2 should be selected in the interval $w_1(\mathbf{p}_1) < W_1 < w_1(\mathbf{p}_2)$ and $w_2(\mathbf{p}_2) < W_2 < w_2(\mathbf{p}_1)$. If the constraints are chosen outside these intervals, then, either there is no solution, which can be verified with a polynomial complexity, or, a shortest path with respect to a single metric is an evident solution, which can be computed with a polynomial complexity. Reference [17] investigates the effect of the choice of the constraints.

We know that the exact algorithms will find a solution if one exists. Consequently, we have chosen the constraints to obtain a high rate of success of the path computation procedure, which means that we focus on the cases where a solution exists. We have performed both simulations where the constraints are equally strict in average or where one constraint is stricter than the other. We consider the proportion of the requests for which the exact algorithms find a solution, as well as the value of the cost metric of SAMCRA (Equation 1) for the lowest-cost solutions computed, to determine *a posteriori* if the constraints are strict or loose. More precisely, loose means that exact algorithms find a solution for every request simulated and at least one solution has a low cost ($c(\mathbf{p}) \leq 0.2$). Strict means that in average the lowest-cost solutions found by exact algorithms have a high cost ($c(\mathbf{p}) \geq 0.8$) or a feasible solution exists for less than 60% of the requests simulated.

Weights	uniform distribution with $10 \leq w_k \leq 1023$, $k = 1..K$ or $k = 1..K-1$; correlated or not; delay estimations and number of hops for REAL(8)
Constraints	constant during a simulation, fixed depending on *a posteriori* observation of the rate of success of the computation and the cost of the paths computed

Table 1: Parameters used for the generation of the weights and of the requests in the main simulations

6.1.4 Selection of Domain Sequences

We focus on the problem of computing inter-domain paths along predefined domain sequences. In each simulation run, we have selected a source and a destination node in different domains. In the Internet, the sequence of domains followed by traffic from a source to a destination domain is not necessarily the shortest, because this sequence depends on the policies of the domains. However, modeling the policies of each domain explicitly is difficult. Thus, we have rather tried to reproduce the average length of the AS-paths in the Internet. We have considered sequences of three domains, which matches the average domain sequence lengths in the Internet, according to [4]. In the realistic topology, we have computed a shortest domain-sequence between the source and the destination domain and

considered this sequence for the node-level path computation between the source node and the destination node.

6.1.5 Performance Metrics

We denote as M(A) the average value of the metric M for the algorithm A for a batch of simulations. For example, α(pID-MCP) denotes the value of α measured for the algorithm pID-MCP in a set of simulations. Confidence intervals are computed for all performance metrics. When the number of requests simulated is not provided explicitly, this means that it is large enough to provide statistically significant results.

We define the absolute success rate (ASR) as the percentage of success of the algorithms to find a feasible path when a solution exists. The success rate (SR) is the percentage of success of the algorithms to find a feasible path for the requests considered. As ID-MCP and pID-MCP are exact, their ASR is 100%, whereas the ASR of kID-MCP, k=1 or kpID-MCP is not necessarily 100%. The number of paths returned by the algorithms is denoted as NP. Both exact algorithms return all end-to-end feasible non-dominated paths, thus NP(ID-MCP)=NP(pID-MCP). The number of paths returned by kpID-MCP can exceed k, because the segments computed by a domain are combined with the segments in the previous VSPT. Our implementation of kID-MCP, k=1 returns a single path per request.

The cost (C) is the lowest value of the path length function of SAMCRA among the paths computed, thus, it takes the same value for all exact algorithms. We define an additional path length function c' as $\mu_{i=1..K}\left(\frac{w_i}{W}\right)$, where μ denotes the arithmetic mean operator. We call multi-dimensional cost (MC) the value of c' for the end-to-end path computed with the lowest value of c'. MC helps to evaluate the quality of the paths returned considering all metrics, whereas C indicates their quality with respect to the most restrictive metric. The costs (C and MC) of the paths are taken into account only for the requests for which both heuristics and exact algorithms succeed to find a feasible path, so that the comparison of the algorithms is meaningful.

In accordance with the results in Theorem 3 and Theorem 5, we derive the relative time complexity of the algorithms from the measurement of the maximum number (α) of paths attached to a node and memorized in the computation queue. The value of α provides also an indication of the spatial complexity of the algorithms. By definition, α(kID-MCP, k=1) = 1.

We evaluate the signaling overhead (SO) induced by the algorithms considered. This quantity is evaluated as the number of elements that are carried in the VSPTs exchanged, for a single request. An element represents a single virtual path from an ingress to the destination. With this definition, the overhead is equal for ID-MCP and pID-MCP. Note, however, that the segments precomputed by pID-MCP are expected to be used for several requests.

6.2 Evaluation

We have simulated the algorithms ID-MCP, pID-MCP, kID-MCP, k=1 and kpID-MCP (k>1) on various network configurations to assess their performance and to illustrate their

strengths and weaknesses. The scenarios simulated allow us to outline some of the trade-offs of the inter-domain multi-constraint path computation methods described in Section 5.

For more clarity, we define a reference simulation scenario to which all other scenarios are compared. The reference simulation scenario involves positively-correlated weights and its constraints are identical for all weights ($W_i = W_j,\ 1 \leq i \leq K,\ 1 \leq j \leq K$). Simulations with independent or negatively-correlated weights use the same constraints as in the reference scenario. We refer to the constraints as strict or loose constraints *with respect to the reference scenario.*

6.2.1 Effect of Inter-Domain Connectivity

First, we compare the performance of the algorithms on ID-MCP and of the algorithms on pID-MCP on the topologies LatticeFM(25,3) and LatticeSL(25,3) presented in Section 6.1.1. This comparison illustrates the effect of the inter-domain connectivity of the domains on the complexity of these algorithms.

On the Lattice topologies considered, the exact algorithms are penalized by a large path diversity that induces an increased time complexity. In particular, the topology LatticeFM(25,3) is designed to illustrate a drawback of the algorithms based on pID-MCP. In this topology, the number of inter-domain links is *extremely* large: each domain is connected to the next domain through 625 inter-domain links. As the algorithms based on pID-MCP perform non-dominance comparisons depending on the destination of each element, they memorize many paths if one domain is connected to the next domain through a large number of ingress nodes. The algorithm kpID-MCP limits the number of paths memorized per node, which solves the problem about the complexity. However, the paths memorized must be selected carefully among the many paths available so that the end-to-end path is close to the optimal. The version of kpID-MCP implemented selects k shortest paths considering the length measure defined in Equation 1. Considering larger initial cost (Equation 2) is probably a better solution to this problem, however it is not desirable for precomputing of path segments independently of the sequence of domains considered.

Table 2 presents the results of the simulations for the topologies LatticeSL(25,3) and LatticeFM(25,3) with loose constraints in the reference scenario. These results illustrate the aforementioned drawback of the methods based on pID-MCP when the inter-domain connectivity is large. As expected, in the LatticeSL topology α(ID-MCP) is greater than α(ID-MCP), whereas in the LatticeFM topology α(ID-MCP) is much lower than α(ID-MCP). This underlines the need for limiting α in the algorithm pID-MCP, which justifies the heuristic kpID-MCP proposed in Section 5.4.2. In the LatticeSL topology kpID-MCP, k=3 provides better results (lower value of C and MC) than kID-MCP, k=1. However, in the LatticeFM topology kpID-MCP, k=3 provides worse results (lower value of C and MC) than kID-MCP, k=1. This problem is solved by allowing larger values of α in kpID-MCP and, if the segments are computed sequentially in the domains, by considering the initial costs in Equation 2.

Table 3 presents the results of the simulations in the same scenario but this time with strict constraints. In this scenario kpID-MCP, k=3 provides better results (lower value of

C and MC, higher value of SR and NP) than kID-MCP, k=1 in the LatticeSL topology. In the LatticeFM topology, kpID-MCP, k=3 and kID-MCP, k=1 provide similar results.

	SR [%]		C [%]		MC [%]		α		NP	
Lattice	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*
ID-MCP	100	100	19.2	13.9	18.7	11.4	10	7	7	3
pID-MCP	100	100	19.2	13.9	18.7	11.4	5	52	7	3
kpID-MCP, k=3	100	100	19.3	26.8	18.8	23.8	3	3	6	1
kID-MCP, k=1	100	100	19.5	13.9	19	11.9	1	1	1	1

Table 2: Results of simulations in the reference scenario (positive correlation) with loose constraints $((49100, 49100)^T$ for LatticeSL and $(3000, 3000)^T$ for LatticeFM)

	SR [%]		C [%]		MC [%]		α		NP	
Lattice	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*
ID-MCP	66	56	89.3	72	86.5	60.1	8	2	5	1
pID-MCP	66	56	89.3	72	86.5	60.1	5	8	5	1
kpID-MCP, k=3	66	50	89.4	72	86.5	60.1	3	3	4	1
kID-MCP, k=1	64	56	89.8	72	87.9	60.1	1	1	1	1

Table 3: Results of simulations in the reference scenario (positive correlation) with strict constraints $((9800, 9800)^T$ for LatticeSL and $(400, 400)^T$ for LatticeFM)

6.2.2 Effect of the Strictness of the Constraints

Second, we investigate the effect of the non-feasibility check in the exact algorithms ID-MCP and pID-MCP by simulating either strict constraints or loose constraints in the topologies LatticeSL(25,3) and LatticeFM(25,3). ID-MCP considers non-zero initial weights, which allows it to discard several non-feasible paths when the constraints are strict, whereas pID-MCP considers initial weights equal to zero for the virtual nodes. Thus the strictness of the constraints has an effect on the complexity of exact algorithms.

We compare the results in Table 2 and in Table 3. In the LatticeFM topology: the values of α are several times smaller with strict constraints than with loose constraints (3.5 times smaller with ID-MCP and even 6.5 times smaller with pID-MCP). These results are explained by the large path diversity in the LatticeFM, which can be drastically reduced when strict constraints are used. With strict constraints, pID-MCP is less penalized by the large inter-domain connectivity of the LatticeFM topology. Typically, the success rate of the path computation procedure, the value of α, as well as the number of paths returned decrease and the cost of the paths computed increases, when the constraints become stricter.

6.2.3 Effect of Asymmetric Constraints

Third, we investigate the performance of the heuristics in the topologies LatticeSL(25,3) and LatticeFM(25,3) when the constraints are asymmetric (W_1 is large and thus, easily fulfilled, whereas W_2 is more restrictive) and the link-weights are negatively-correlated. The heuristics studied memorize shortest paths with respect to the cost c and thus, select the paths depending on the most restrictive metric. Intuitively, this can lead to non-optimal values of the other metrics for the end-to-end path selected.

Table 4 presents the results of the simulations, which we compare to the results in Table 5 for simulations with symmetric constraints. We have selected the constraints so that the cost C and the success rate SR take similar values in both tables. As expected, the difference between C and MC is larger with asymetric constraints than with symmetric constraints for all algorithms. However, somewhat surprisingly, the difference between the value of MC for the heuristics and for the exact algorithms is smaller with asymetric constraints than with symmetric constraints.

6.2.4 Effect of the Correlation of the Weights

Forth, we investigate the effect of the correlation of the weights on the performance of the algorithms considered. We keep the same constraints as in the reference scenario and simulate requests in the LatticeSL and LatticeFM topologies, but, this time, with negatively-correlated weights. These simulations are performed because the correlation of the weights is known to affect the number of non-dominated paths. When the weights are positively-correlated, a path with a low value for a metric is likely to take a low value for the other metric too: considering two paths $\mathbf{p}_1$ and $\mathbf{p}_2$, if $w_1(\mathbf{p}_1) < w_1(\mathbf{p}_2)$, then, it is likely that $w_2(\mathbf{p}_1) < w_2(\mathbf{p}_2)$. Thus, the number of non-dominated paths is usually low. However, when the weights are negatively-correlated, a path taking a low value for a weight is likely to take a large value for the other constraint. Thus, there are usually more non-dominated paths with negatively-correlated weights than with positively-correlated weights. A higher number of non-dominated paths has a negative effect on the complexity of the path computation algorithms, because more paths need to be memorized.

We compare the results in Table 5 and in Table 2. In the LatticeSL topology, the difference of cost between the results of the exact methods and the results of the heuristics is significantly larger with negatively-correlated weights than with positively-correlated weights. This increased difference certainly comes from the larger number of non-dominated paths with negatively-correlated weights. In addition, the value α and NP for exact methods are much larger with negatively-correlated weights than with positively-correlated weights. In fact, α(ID-MCP) is multiplied by seven and α(pID-MCP) by three, which increases the difference of complexity between these algorithms. Thus, pID-MCP is significantly faster than IP-MCP in this configuration. The difference on α has an effect on the cost of the paths returned by the heuristics too. First, the relative difference of cost between kpID-MCP, k=3 and ID-MCP, k=1 rises from about 1% with positively-correlated weights to 3% with negatively-correlated weights, for the requests considered. Second, the relative

difference of cost between ID-MCP, k=1 and exact algorithms rises from about 2% with positively-correlated weights to 5% with negatively-correlated weights, for the requests considered.

With pID-MCP and kpID-MCP, the segments computed in a domain are combined with the virtual segments of the preceding VSPT. Thus, in the initial queue considered by a domain, the number of paths attached to a virtual node may exceed the maximum value of α allowed. Thus, NP can exceed the number of maximum number of segments attached to the source node (*e.g.,* 3) multiplied by the maximum number of segments attached to the ingresses (*e.g.,* 3) multiplied by the number of ingresses (*e.g.,* 1). This explains why the number of paths (NP=11) returned by kpID-MCP, k=3 in LatticeSL with negatively-correlated weights and loose constraints exceeds the square of the maximum value of α allowed ($3^2 = 9$) multiplied by the number of ingresses (1) in the second domain crossed.

The same value of the constraints is usually much stricter with negatively-correlated weights than with positively-correlated weights. For instance, with the loose constraints of the reference scenario, C is approximately 25% larger with negatively-correlated weights than with positively-correlated weights in the LatticeSL topology. This difference is even larger in the LatticeFM topology: C is multiplied by approximately two when negatively-correlated weights are simulated. Moreover, with the strict constraints of the reference scenario no solution exists in the simulation runs performed with negatively-correlated weights.

	SR [%]		C [%]		MC [%]		α		NP	
Lattice	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*
ID-MCP	100	100	23.3	27.9	17.4	22.6	67	8	64	5
pID-MCP	100	100	23.3	27.9	17.4	22.6	14	57	64	5
kpID-MCP, k=3	100	100	23.3	33.2	17.4	26.1	3	3	11	2
kID-MCP, k=1	100	100	23.3	28	17.5	23.5	1	1	1	1

Table 4: Results of simulations with negatively-correlated weights and asymmetric constraints ($(147000, 37000)^T$ for LatticeSL and $(9000, 1000)^T$ for LatticeFM)

	SR [%]		C [%]		MC [%]		α		NP	
Lattice	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*	*SL*	*FM*
ID-MCP	100	100	24.3	28.6	24	24.6	69	13	65	8
pID-MCP	100	100	24.3	28.6	24	24.6	15	74	65	8
kpID-MCP, k=3	100	100	24.9	48.1	24.4	41.3	3	3	11	2
kID-MCP, k=1	100	100	25.6	28.6	25.1	26.8	1	1	1	1

Table 5: Results of simulations with negatively-correlated weights and loose constraints ($(48100, 48100)^T$ for LatticeSL and $(3000, 3000)^T$ for LatticeFM)

6.2.5 Simulations on a Realistic Inter-Domain Topology

Last, we simulate a scenario in the topology REAL(8), which represents the topology in the USA of some of the largest operators in the world [19]. These simulations are used as a demonstration of the applicability of our heuristics on a realistic example. The requests simulated are constrained by a maximum value of the end-to-end one-way speed-of-light propagation delay equal to 100 ms and a maximum number of hops equal to 30. The constraint of 100 ms is feasible according to the service level agreements advertised by Sprint for the USA [25] and seems reasonable for voice traffic, for example (see Reference [12]). The value of the constraint on the number of hops is set arbitrarily, so that this constraint is loose. The number of hops represents a traffic engineering metric equal to one for every link, which used by operators to minimize the amount of resources used by each request. The two link-weights considered are relatively static[11] and thus, segments can be precomputed for the class of service considered.

We consider a source-destination domain-pair and we generate twenty source-destination node-pairs. The domain pair considered in the simulations considered leads to a sequence of two domain. In average, the domains are made up of approximately 1000 nodes and 3500 undirected edges. They are connected through eleven undirected links. We compare the quality of the paths computed with kID-MCP, k=1 and kpID-MCP, k=3. In another simulation we compare the results provided by kID-MCP, k=1 and kpID-MCP, k=8.

	SR [%]	C [%]	MC [%]	α	NP	SO
kpID-MCP, k=3	100	14	13	3	8	×3
kID-MCP, k=1	100	10	10	1	1	–

Table 6: Results of simulations on a realistic topology with a limit on α equal to three

	SR [%]	C [%]	MC [%]	α	NP	SO
kpID-MCP, k=8	100	14	13	8	49	×10
kID-MCP, k=1	100	12	12	1	1	–

Table 7: Results of simulations on a realistic topology with a limit on α equal to eight

Table 6 and Table 7 describe the results of the simulations. The slight difference between the cost of kID-MCP, k=1 in Table 6 and in Table 7 comes from the relatively low number of random requests simulated. This number is not problem as we are mainly intersted into comparing both heuristics for a common set of requests. In both simulations, the paths computed by kpID-MCP have a slightly higher cost (C and MC) than the paths computed by kID-MCP, k=1. This comes certainly from the relatively large inter-domain connectivity for the sequence of domains considered. The difference of cost between the solutions computed by kpID-MCP and kID-MCP, k=1 decreases slightly when kpID-MCP considers a limit on

[11] They can change if the topology is modified.

α equal to eight instead of three. More precisely, for the requests considered, the difference of C between kpID-MCP and kID-MCP, k=1 is divided by two when kpID-MCP considers a limit on α equal to eight instead of three. A larger limit on α is required to approach the optimal solutions more accurately. On the one hand, in the simulations, the signaling overhead for the precomputation of the segments with kpID-MCP, k=3 and kpID-MCP, k=8 represent the signaling overhead for three and ten requests with kID-MCP, k=1. On the other hand, in average, kpID-MCP, k=3 and kpID-MCP, k=8 compute eight and forty-nine feasible paths whereas kID-MCP, k=1 returns a single feasible path.

6.2.6 Inter-Area Scenario

We have simulated hundred random requests in the SYM-CORE topology with strict constraints. The results of these simulations are presented in Table 8. They confirm the conclusions presented in previous sections. The path diversity is smaller in this inter-area topology compared to larger inter-domain topologies. Thus, α and NP take low values. In addition, ASR is quite low whereas C is lower than 80%.

	SR [%]		C [%]		MC [%]		α		NP	
Correlation	+	−	+	−	+	−	+	−	+	−
ID-MCP	60	41	69.2	76.2	64.8	70.3	5	3	2	1
pID-MCP	60	41	69.2	76.2	64.8	70.3	9	5	2	1
kpID-MCP, k=3	60	41	69.9	76.2	65.5	70.4	3	3	2	1
kID-MCP, k=1	58	41	69.6	76.5	65.2	70.4	1	1	1	1

Table 8: Results of simulations with strict constraints in the SYM-CORE topology

7 Discussion and Related Works

7.1 Trade-Offs Among the Solutions

The simulation scenarios simulated illustrate important trade-offs between the algorithms and heuristics of Section 5. First, the exact algorithms ID-MCP and pID-MCP are adapted to different topologies (Section 6.2.1): pID-MCP usually has a larger complexity (*alpha* is significantly larger) than ID-MCP when the number of ingresses is large. However, in more usual topologies, the complexity of pID-MCP is usually lower (α is lower) than the one of ID-MCP. Both algorithms are exact, thus, they return the same non-dominated feasible paths. However, pID-MCP can be used to precompute the path segments inside each domain for each class of service and combine these segments in a later step. The algorithm pID-MCP has two desirable properties, considering that the problem Inter-MCP is NP-complete. First, it divides the path computation operations into per-domain computations. Second, it reuses the path segments computed, independently of the sequence of domains crossed.

The networks of the largest operators contain several thousands of nodes. In so large networks, the exact computation of inter-domain multi-constrained paths is impracticable. In this context, heuristics provide a useful alternative to exact algorithms. The on-demand shortest-path based heuristic ID-MCP, with k=1 provides good results in most situations and has the lowest complexity among the solutions tested. However it provides at most a single solution and is not adapted for precomputing paths. As explained in Section 5.4.2, kpID-MCP can provide worst solutions than kID-MCP, k=1 in certain situations, when the number of paths to memorizes exceeded the limit on α. Nevertheless, the heuristic kpID-MCP, k>1 approaches optimal computations, depending on the limit on α and its complexity increases with this limit. The limit on α should be adapted to the level of accuracy required for the computations and to the maximum time allowed for the computations.

We have evaluated kpID-MCP with a low value of the limit on α ($\alpha \leq 3$). With this limit, kpID-MCP performs better than kID-MCP, k=1 in all simulations on LatticeSL but worse in all simulations on LatticeFM. Even with a low limit on α, kpID-MCP returns several times more solutions than kID-MCP, in most simulations, which is an important feature for balancing the load on several paths. As the precomputation of the segments is not triggered by the arrival of every request, the signaling overhead per request is expected to be lower when segments are precomputed than otherwise. In our opinion, kpID-MCP is a good choice for the precomputation of inter-domain multi-constrained paths approaching optimal solutions.

7.2 Related Works

Many recent works have studied the problems of QoS routing and path computation in the inter-domain context.

Some approaches are based on extensions of BGP, the *de facto* inter-domain routing protocol, in order to deliver QoS services. The work in [36] investigates the extension of BGP to support multiple metrics. This necessitates advertising multiple routes for each destination. The authors propose to reduce the number of advertised paths using the dominance properties. For this purpose, they introduce one optimal approach and one heuristic approach for reducing the algorithm complexity. Similarly, the authors of the work in [2] investigate the enhancements to the BGP protocol in order to support the discovery of multiple paths per destination with associated QoS attributes. They present a dominant path selection algorithm that allows nodes to discover the minimum set of paths needed to make QoS routing decisions. In [1] an extension for BGP, called EQ-BGP, is introduced. EQ-BGP enables to establish end-to-end paths that offer the most suitable QoS guarantees taking into account both the QoS capabilities of particular domains as well as inter-domain links. EQ-BGP uses QoS components such as a path attribute for QoS information, a QoS aggregation function for combining path segments, a QoS aware decision algorithm for selecting the best path, and multiple routing tables that allow border routers to keep separate paths that are optimised for different QoS objectives.

Other approaches take advantage of the emerging PCE architecture for computing inter-domain paths. In [26] the authors study the cooperation between domains in order to com-

pute end-to-end paths. They demonstrate that by distributing information about blocking in each domain, end-to-end PCE-based path computation can be made more efficient in terms of the overall path computation effort. The work in [21] study the establishment of constrained inter-domain MPLS LSPs subject to end-to-end delay constraint. The authors present and evaluate path computation techniques using the default BGP route or relying on PCEs with global or local visibility. When local visibility is used, the selection of the exit BN relies on two heuristics: the selection of the nearest BN based on the routing protocol metrics or based on an estimation of the end-to-end delay.

Alternative approaches introduce new architectures for enabling the inter-domain path computation. [20] presents an overlay architecture that enables QoS routing with no required changes to BGP. The authors introduce two algorithms: the first uses a shortest-path framework, using a cost metric that accounts for both the intrinsic cost of each link and the amount of bandwidth that the link has available for use. The second dynamically probes several paths in parallel, in order to find a path capable of handling the flow. In [15], the authors introduce a method of aggregating networks considering two QoS parameters: bandwidth and delay. They use line segments in the delay-bandwidth plane instead of points to represent the QoS parameters of logical links. They also present a corresponding QoS routing protocol combining an inter-domain and an intra-domain routing step.

8 Conclusion

The present paper analyzes the problem of computing inter-domain paths subject to multiple constraints. We describe the information required by the domains for solving this problem exactly. In particular, our work shows that the virtual shortest-path tree structure for exchanging information between the domains does not guarantee optimal solutions. We explain why the solutions to the intra-domain problem are not applicable to the inter-domain problem. Consequently, we extend these solutions: we describe two exact algorithms, called ID-MCP and pID-MCP, which respect the confidentiality constraints of the domains. Remarkably, pID-MCP allows the precomputation of segments inside the domains crossed. As the problem considered is NP-complete, we propose several heuristics approaching optimal solutions. A simulation study considering both extreme and realistic topologies reveals that the simplest heuristic, a shortest path computation with respect to a non-linear path length, provides excellent results in most configurations. However, this heuristic provides at most a single feasible path. A second heuristic, named kpID-MCP, approximates with tunable accuracy the exact algorithm pID-MCP. We describe the particular situations in which it performs worse than the simplest heuristic. The heuristic kpID-MCP returns several paths in most configurations, which is a desirable feature for balancing the load on the paths precomputed.

There are different ways to extend the results we have obtained. First, shortest paths could be computed with respect to each link-weight considered and their weight could be used as a minimum-weight prediction to exclude segments on infeasible paths. It would be interesting to study how to extend the algorithms described in the present paper with this

mechanism, without breaking confidentiality constraints. Second, kpID-MCP memorizes the paths with the lowest cost, however, other criteria could be used. For example, kpID-MCP could memorize preferably a path improving significantly one of the weights, compared to the paths already memorized, even if the cost of this new path is higher. Finally, path computations with pID-MCP use only a subset of the segments available in each domain for each class of service. It would be of interest to characterize this subset more precisely.

Contents

1 Introduction **3**

2 Inter-Domain Routing and Path Computation **4**
2.1 General Scenario and Assumptions 4
2.2 Domain Visibility 5
2.3 Path Computation Models 6
2.3.1 Per-domain Path Computation 6
2.3.2 PCE-Based Computation 7

3 Path Computation with Multiple Constraints **8**
3.1 The Multi-Constrained Path Problem 8
3.2 Exact and Approximated Intra-Domain Solutions 9

4 Problems of the Current Approaches **10**
4.1 Limitations of MCP Algorithms 10
4.2 Limitations of Inter-Domain Path Computation Methods 11

5 Extensions for Inter-Domain Path Computation **12**
5.1 Information Required for Exact Inter-Domain MCP Computations 12
5.2 Extended MCP Algorithms 13
5.2.1 Adaptation of the Principles of SAMCRA and H_MCOP 13
5.2.2 ID-MCP: Algorithm and Example 14
5.3 Proof and Complexity of ID-MCP 16
5.4 Enhancement of ID-MCP 18
5.4.1 Parallelization and Complexity Reduction (pID-MCP) 18
5.4.2 Heuristics 21

6 Simulation and Results **23**
6.1 Methodology 23
6.1.1 Topologies 23
6.1.2 Link Weights 24
6.1.3 Constraints 25
6.1.4 Selection of Domain Sequences 25
6.1.5 Performance Metrics 26
6.2 Evaluation 26
6.2.1 Effect of Inter-Domain Connectivity 27
6.2.2 Effect of the Strictness of the Constraints 28
6.2.3 Effect of Asymmetric Constraints 29
6.2.4 Effect of the Correlation of the Weights 29
6.2.5 Simulations on a Realistic Inter-Domain Topology 31
6.2.6 Inter-Area Scenario 32

7 Discussion and Related Works **32**
7.1 Trade-Offs Among the Solutions 32
7.2 Related Works 33

8 Conclusion **34**

References

[1] A. Beben. EQ-BGP: an efficient inter-domain QoS routing protocol. In *20th International Conference on Advanced Information Networking and Applications, AINA*, April 2006.

[2] L. Benmohamed, C.-J. Liang, E. Naber, and A. Terzis. QoS Enhancements to BGP in Support of Multiple Classes of Service. In *Military Communications Conference, MILCOM*, October 2006.

[3] T. Bressoud, R. Rastogi, and M. Smith. Optimal configuration for BGP route selection. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. (INFOCOM)*, volume 2, pages 916–926, 2003.

[4] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 638–647, 2002.

[5] S. Dasgupta, J. de Oliveira, and J. P. Vasseur. Performance Analysis of Inter-Domain Path Computation Methodologies. draft-dasgupta-ccamp-path-comp-analysis-02, work in progress, IETF, July 2008.

[6] H. De Neve and P. Van Mieghem. A multiple quality of service routing algorithm for PNNI. In *IEEE ATM Workshop Proceedings*, pages 324–328, 1998.

[7] H. De Neve and P. Van Mieghem. TAMCRA: a tunable accuracy multiple constraints routing algorithm. *Computer Communications*, 23(7):667–679, 2000.

[8] A. Farrel, J. P. Vasseur, and G. Ash. RFC 4655: A Path Computation Element (PCE)-Based Architecture. IETF, August 2006.

[9] A. Farrel, J. P. Vasseur, and A. Ayyangar. RFC 4726: A Framework for Inter-Domain Multiprotocol Label Switching Traffic Engineering. IETF, November 2006.

[10] M. P. Howarth, M. Boucadair, P. Flegkas, N. Wang, G. Pavlou, P. Morand, T. Coadic, D. Griffin, A. Asgari, and P. Georgatsos. End-to-end quality of service provisioning through inter-provider traffic engineering. *Computer Communications*, 29(6):683–702, march 2006.

[11] IPSF. IPSphere Framework Technical Specification, R1, June 2007.

[12] ITU-T. Recommendation G.114. `http://www.itu.int/rec/T-REC-G.114-200305-I`, May 2003.

[13] J. M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14:95–116, 1984.

[14] D. King, Y. Lee, H. Xu, and A. Farrel. Path Computation Architectures Overview in Multi-Domain Optical Networks Based on ITU-T ASON and IETF PCE. In *IEEE Network Operations and Management Symposium (NOMS) Workshops*, pages 219–226, April 2008.

[15] L. King-Shan and K. Nahrstedt. Topology aggregation and routing in bandwidth-delay sensitive networks. In *IEEE Global Telecommunications Conference, GLOBECOM*, 2000.

[16] T. Korkmaz and M. Krunz. Multi-constrained optimal path selection. In *Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 834–843, 2001.

[17] F. Kuipers and P. Van Mieghem. The impact of correlated link weights on QoS routing. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1425–1434, 2003.

[18] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz. An overview of constraint-based path selection algorithms for QoS routing. *IEEE Communications Magazine*, 40(12):50–55, Dec. 2002.

[19] M. Liljenstam, J. Liu, and D. Nicol. Development of an Internet backbone topology for large-scale network simulations. In *Winter Simulation Conference*, volume 1, pages 694–702, Dec. 2003.

[20] S. Norden. Inter-domain routing: Algorithms for QoS guarantees. *Computer Networks*, 49(4):593–619, November 2005.

[21] C. Pelsser and O. Bonaventure. Path Selection Techniques to Establish Constrained Interdomain MPLS LSPs. In *Proceedings of Networking 2006*, Coimbra, Portugal, May 15-19th 2006.

[22] B. Quoitin, C. Pelsser, O. Bonaventure, and S. Uhlig. A performance evaluation of BGP-based traffic engineering. *International journal of network management*, 15:177–191, 2005.

[23] D. Shin, E. K. P. Chong, and H. J. Siegel. A multiconstraint QoS routing scheme using a modified Dijkstra's algorithm. In *Joint International Conference on Wireless LANs and Home Networks (ICWLHN 2002) and Networking (ICN 2002)*, pages 65–76, 2002.

[24] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, Feb. 2004.

[25] Sprint. Service Level Agreements. `http://www.sprintworldwide.com`. Last visited: 8th August 2008.

[26] P. Torab, B. Jabbari, Q. Xu, S. Gong, X. Yang, T. Lehman, C. Tracy, and J. Sobieski. On cooperative inter-domain path computation. In *11th IEEE Symposium on Computers and Communications, ISCC*, June 2006.

[27] P. Van Mieghem, H. De Neve, and F. Kuipers. Hop-by-hop quality of service routing. *Computer Networks*, 37(3-4):407–423, 2001.

[28] P. Van Mieghem, G. Hooghiemstra, and R. van der Hofstad. A scaling law for the hopcount in internet. Technical report, Delft University of Technology, 2000.

[29] P. Van Mieghem and F. A. Kuipers. Concepts of exact QoS routing algorithms. *IEEE/ACM Transactions on Networking*, 12(5):851–864, 2004.

[30] J. P. Vasseur, A. Ayyangar, and R. Zhang. RFC 5152: A Per-Domain Path Computation Method for Establishing Inter-Domain Traffic Engineering (TE) Label Switched Paths (LSPs). IETF, February 2008.

[31] J. P. Vasseur and J. L. Le Roux. Path Computation Element (PCE) Communication Protocol (PCEP). draft-ietf-pce-pcep-12, work in progress, IETF, March 2008.

[32] J. P. Vasseur, R. Zhang, N. Bitar, and J. L. Le Roux. A Backward Recursive PCE-based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-domain Traffic Engineering Label Switched Paths. draft-ietf-pce-brpc-09, work in progress, IETF, 2008.

[33] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, 1996.

[34] L. Xiao, K. Lui, J. Wang, and K. Nahrstedt. QoS extension to BGP. In *10th IEEE International Conference on Network Protocols, Paris, France*, page 100, Nov. 2002.

[35] W. Xu and J. Rexford. MIRO: Multi-path Interdomain ROuting. In *ACM SIGCOMM*, 2006.

[36] T. Zhang, Y. Cui, Y. Zhao, L. Fu, and T. Korkmaz. Scalable BGP QoS Extension with Multiple Metrics. In *International conference on Networking and Services*, 2006.