Received: 30 December 2016 | Revised: 22 November 2017 | Accepted: 22 November 2017

DOI: 10.1002/net.21800

RESEARCH ARTICLE

WILEY

Domain clustering for inter-domain path computation speed-up

Lorenzo Maggi[1] | **Jérémie Leguay**[1] | **Johanne Cohen**[2] | **Paolo Medagliani**[1]

[1]Mathematical and Algorithmic Sciences Lab, France Research Center, Huawei Technologies Co. Ltd, Shenzhen, China

[2]LRI-CNRS, Université Paris-SUD, Centrale Supelec, Université Paris-Saclay, Orsay, France

Correspondence
Lorenzo Maggi, Mathematical and Algorithmic Sciences Lab, France Research Center, Huawei Technologies Co. Ltd, Shenzhen, China.
Email: lorenzo.maggi84@gmail.com

Abstract

We consider a multi-domain network scenario and we study the Inter-Domain Path Computation problem under the Domain Uniqueness constraint (IDPC-DU), that is, a path cannot visit a domain twice. It is known that hierarchical Path Computation Element (h-PCE) architecture, that is commonly used to solve IDPC-DU, shows poor scalability with respect to the number of domains. For this reason, we devise a new domain clustering concept allowing one to artificially reduce the number of domains in an offline phase, in order to solve IDPC-DU with lower complexity at run-time. More specifically, we first prove the NP-completeness of the feasibility problem associated with IDPC-DU and the inapproximability of IDPC-DU itself. Yet, we show that the number of domains is the real computational bottleneck for the solution of IDPC-DU. Then we provide a necessary and sufficient condition for a domain clustering to be *proper*, that is, without loss of optimality. Such a condition can be verified *offline* on the inter-domain graph. We finally show via numerical experiments the impact of the inter-domain treewidth on the computational speed-up brought by proper clustering.

KEYWORDS

domain clustering, domain re-entry, domain uniqueness, hierarchical PCE, inter-domain path computation, semi-hierarchical PCE

1 | INTRODUCTION

Large networks are often partitioned into several domains. Generally speaking, a network domain can be defined as "any collection of network elements within a common sphere of address management or path computational responsibility" [11]. Prominent examples of multi-domain networks are Autonomous Systems (AS) and Interior Gateway Protocol areas, which partition a single AS. Historically, multi-domain networks have been introduced to tackle issues related to *scalability* (as they allow one to reduce routing signaling) and *privacy* (domains may belong to competing operators, that may not wish to disclose confidential intra-domain information to neighboring domains) [20].

In multi-domain networks, the path computation inside each domain (ie, *intra*-domain) is managed by a Path Computation Element (PCE). A path is classically defined as a multi hop sequence of links connecting nodes of the network. Mainly two architectures, distributed and hierarchical, have been envisioned for handling communication among different PCEs. In distributed architectures there is no central PCE able to coordinate all PCEs at once. Hence, each PCE is only allowed to exchange information with its neighboring PCEs, commonly via the Backward Recursive PCE-Based Computation (BRPC) protocol [24]. The main drawback of the BRPC protocol lies in the fact that the sequence of traversed domains has to be pre-computed in an offline manner by an external process, for example, by the Border Gateway Protocol (BGP). Such dissociation between domain sequence and path computation weakens the responsiveness of such an architecture to path metric fluctuations and, eventually,

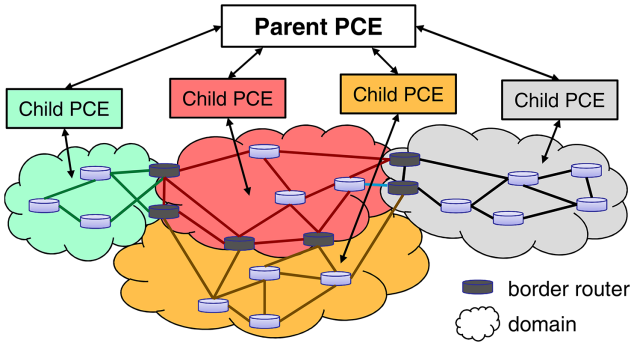

FIGURE 1 h-PCE: hierarchical Path Computation Elements architecture [Color figure can be viewed at wileyonlinelibrary.com]

path computation optimality is hardly achieved [20]. Thus, in order to allow for a joint computation of the end-to-end path and of the optimal sequence of visited domains, the hierarchical PCE (h-PCE) architecture has been promoted in the last few years [9]. The Parent PCE sits on top of the hierarchy, gathering *intra*-domain routing information from all Children PCEs to perform *inter*-domain path computation (see Figure 1). Normally, for obvious scalability and intra-domain privacy reasons, only aggregate routing information such as border-to-border path metrics are disclosed by each Child PCE to the Parent PCE.

On the other hand, h-PCE is affected by serious scalability issues when the number of domains increases [10]. As mentioned in the reference IETF document [9], "*applying the h-PCE model to large groups of domains [...] is not considered feasible*". The main reason behind the poor scalability of h-PCE is the presence of two bottlenecks, namely (1) the Parent PCE's computational capabilities and (2) the capacity of the Children-to-Parent PCE communication channel as the number of domains increases. For this reason, in this article we will mainly address issue (1) by proposing a novel domain clustering concept that allows us to artificially reduce the number of domains in the eyes of the Parent PCE, without altering the quality of computed paths.

Typically, the path computation objectives that are sought to be optimized are various and specific to the application context. Classic routing objectives are path cost, delay (to be minimized) or residual bandwidth (to be maximized). A less well-known class of objectives aims at reducing the delays introduced by the packet processing in routers' look-up tables. In MPLS networks, it is common practice to jointly target such objectives by reducing the number of Label Switch Paths (LSPs). In fact, the number of LSPs installed in a Label Switch Router (LSR) severely impacts the processing speed of packets [10]. As a positive side-effect, limiting the number of installed LSPs also allows one to reduce the signaling overhead across the network. In fact, classic connection-oriented signaling protocols as RSVP-TE require constant monitoring of each LSP for traffic engineering purposes, which generates extra-traffic between LSRs and a central controller.

In order to reduce the packet processing delay, a natural solution would be to compute the routing for a new connection request by taking into account the number of LSPs instantiated on each LSR, so as to avoid congestion on the already overloaded LSRs. Unfortunately, this approach would entail a non-negligible overhead burden, as the Parent PCE would need to be aware of the LSR load at any time. In the practice of MPLS networks, the joint reduction in the packet processing delay and signaling overhead are tackled in two different manners, either by (1) capping the number of domains traversed by a path or by (2) preventing a path from crossing a domain more than once [9]. The former approach (1) essentially attempts to cap the overall number of LSPs instantiated over the whole network, but it may produce paths that cross the same domain several times. Hence, this does not allow one to fairly distribute the processing load among different domains. In fact, every time a path re-enters a domain, a new LSP needs to be installed on the LSRs belonging to the corresponding domain. In order to overcome this problem it has been advocated that in each domain only one LSP should be installed for each connection request [9, 15]. This intuitive objective can be simply achieved by condition (2), that is, by ensuring that, once a path has left a domain, it does not re-visit it later on. In this article we focus on such a constraint, that we call the *Domain Uniqueness* (DU) constraint (In the official reference IETF document RFC 6805 [9], the DU constraint is referred to as "disallow domain re-entry".). We will show that the DU constraint makes path computation hard, and for this reason we will develop a novel domain clustering concept that shows some interesting relations with the topology of the inter-domain graph.

1.1 | Main contributions

In this article, we address hierarchical PCE (h-PCE) architectures and we study the computation of inter-domain paths with minimum cost (or, equivalently, minimum delay), under the *Domain Uniqueness* (DU) constraint. We denote this problem as

Inter-Domain Path Computation under Domain Uniqueness constraint (IDPC-DU). We are primarily interested in the reduction of the Parent PCE's computational burden, which we tackle by introducing a novel domain clustering concept. We find that, under some conditions, the number of domains can be artificially reduced in the eyes of the Parent PCE *without loss of optimality*, thanks to computations performed in an offline phase. This allows us to speed up the IDPC-DU computations at run-time. More specifically, we first formally define IDPC-DU in Section 3. We then prove in Section 4 the NP-completeness of the feasibility problem associated to IDPC-DU (Theorem 1), implying that IDPC-DU is not even approximable in polynomial time (Proposition 1). We point out that such complexity results have been independently proved in [18] (see Section 3, Theorem 2), but via a different polynomial reduction technique. On the other hand, the dynamic programming algorithm presented in Section 5 is Fixed Parameter Tractable in the number of domains $|\mathcal{D}|$ (Lemma 2), suggesting that the computational bottleneck for IDPC-DU is indeed $|\mathcal{D}|$. Therefore, in Section 6 we develop the concept of *proper domain clustering* (Definition 7), allowing us to reduce the effective number of domains to be considered for the solution of IDPC-DU. In order to verify whether a domain clustering is a proper one we provide a condition, called Intra/Extra-Cluster (IEC), which is proven to be sufficient (Theorem 2) and, under a weak assumption, necessary (Theorem 3). Remarkably, the IEC condition can be checked on the inter-domain graph, which is quasi-static with respect to all input parameters. Thus, clustering can be performed *offline*. In Section 7 we discuss how in practice clustering can be boosted via a pre-filtering technique. Then, we validate via numerical experiments the speed-up that clustering brings to IDPC-DU solutions at run-time. We finally discuss in Section 8 the potential impact of our results on the existing PCE architectures.

2 | RELATED WORK

Inter-domain domain routing has sparked the interest of several researchers and engineers throughout the last decade. In particular, an interested reader can find a good survey on recent advances on path computation for distributed and hierarchical PCE (h-PCE) architectures in [20].

On the other hand, to the best of our knowledge, this article is the first formally studying the problem of IDPC-DU, that is, a minimum cost (or, equivalently, minimum delay) path that crosses a domain at most once. Although the DU constraint has been used by MPLS practitioners for a few years, only recently has it been officially advocated in IETF documents such as [9, 15]. However, such documents do not discuss how to take the DU constraint into account for path computation. In [3] the authors introduce a solution based on a combination of hierarchical routing and path computation procedures. The former element identifies the domain sequence to cross while the latter computes the end-to-end path. In [21] the authors propose a Domain Sequence Protocol allowing us to compute the sequence of traversed domains in hierarchical PCE architectures. However, none of the mentioned approaches ensures that the DU constraint is respected.

A few works have already considered clustering ideas for path computation speed-up purposes. A seminal paper on centralized path computation discusses the need for a clustering mechanism to accelerate computation of inter-domain paths [8]. Two other seminal works study shortest path computation speed-up in hierarchically clustered networks, but without considering the DU constraint [1, 25]. However, these ideas have had little follow-up, likely because in the last decades the community has focused on purely distributed computing architectures. Other domain clustering techniques have already been proposed for different purposes, for example, to characterize the spectral features of multi-domain networks [12]. Regarding the PCE architecture for inter-domain path computation, the work in [14] is the first to advocate a hybrid hierarchical/BRPC protocol that would allow one to reduce the network control overhead induced by a pure h-PCE architecture. Nevertheless, [14] does not investigate how to distribute the path computation via BRPC while maintaining the solution quality provided by h-PCE.

From a graph theoretical perspective, IDPC-DU is similar to the computation of a rainbow path, that is, a path whose edges are all of different colors, on a colored graph. Instead, we do allow for paths having several edges belonging to the same domain, as long as such edges are adjacent. However, the literature does not provide algorithmic solutions for rainbow path computation, while it focuses on the characterization of the rainbow connection number (being the minimum number of colors needed to rainbow color the graph itself, that is, such that there exists a rainbow path between each pair of nodes; see, e.g., [17] for a survey). The complexity properties of IDPC-DU have already been studied in [18] (see Section 3, Theorem 2). In this article we independently prove the NP-hardness and the inapproximability of IDPC-DU by using different techniques than those in [18].

IDPC-DU is also related to the clustered shortest path tree problem (CluSPT), studied in [6]. There, nodes are partitioned into clusters (or domains) and the goal is, given a source node s, to compute a spanning tree over the overall graph that is still connected when restricted to any cluster. The metric to be minimized is the total length of paths from the source s to each node. We remark that if CluSPT is feasible, then it is able to find a feasible solution for (the node version of) IDPC-DU, for source

s and for all destination nodes. On the other hand, the converse is not true: in fact, if for instance the subgraph restricted to clusters is not connected then CluSPT will never find a solution, while IDPC-DU can be still feasible. To provide a simple example, consider a graph with three nodes s, i, j and two undirected edges $(s, i), (s, j)$. Nodes i, j belong to the same cluster. Then, CluSPT is infeasible, while IDPC-DU returns the paths $\{s, i\}$ and $\{s, j\}$ for destinations i and j, respectively.

3 | IDPC-DU: PROBLEM STATEMENT AND DEFINITIONS

We model a generic multi-domain network as a weighted colored directed multi-graph $\mathcal{G} = (V, E)$, where V is the set of nodes and E is the set of edges. Let $(i,j)^k \in E$ denote the k-th parallel edge between nodes i and j. Each edge $(i,j)^k$ has an associated positive weight (equivalently denoted as cost or delay) $w^k_{i,j}$ and a color $d \in \mathcal{D}$, where $\mathcal{D}$ is the finite set of colors. We interpret the color of an edge as the network *domain* [11], which the edge belongs to. Let E^d be the set of edges having color d. We assume that $\left\{E^d\right\}_{d\in\mathcal{D}}$ is a partition of E, that is, each edge belongs exactly to one domain.

Under the Domain Uniqueness (DU) constraint, a path on $\mathcal{G}$ traverses every domain at most once. The DU constraint determines the feasibility of a path on graph $\mathcal{G}$, as formally defined below.

Definition 1 (DU: Domain Uniqueness)**.** A path $\mathbf{p} = \{p_0, p_1, \dots\}$ on the directed colored graph $\mathcal{G}$, meant as a succession of edges ($p_i \in E,\ \forall\, i$), is feasible if and only if it fulfills the Domain Uniqueness (DU) constraint, that is, once $\mathbf{p}$ has left a domain it does not revisit it later on. More formally,

$$\forall\, i, d:\ p_i \in E^d\ \wedge\ p_{i+1} \notin E^d\ \Rightarrow\ p_{i+k} \notin E^d,\ \forall\, k \geq 2.$$

We denote $\mathcal{F}_{s,t}(\mathcal{G})$ as the set of feasible paths on graph $\mathcal{G}$ from source node s to destination t.
We now introduce IDPC-DU, that is investigated in this article.

Definition 2 (IDPC-DU: Inter-Domain Path Computation under Domain Uniqueness)**.** Let $s, t \in V$. Compute the min-cost feasible path $\mathbf{p}^*$ on $\mathcal{G}$ from s to t, that is, solve the following optimization problem:

$$\mathbf{p}^* = \underset{\mathbf{p}\in\mathcal{F}_{s,t}(\mathcal{G})}{\arg\min} \sum_{(i,j)^k\in\mathbf{p}} w^k_{i,j} := w(\mathbf{p}).$$

We remark that in order to take into account the size of a demand in IDPC-DU, it trivially suffices to prune the edges with capacity smaller than the demand size. We refer to Figure 2 for a simple example of the IDPC-DU problem. An interested reader may find the Integer Linear Programming (ILP) formulation for IDPC-DU in the Appendix.

We conclude this section by describing the crucial concept of inter-domain graph, describing the connections among different domains. To this aim, we introduce the set of nodes V^d_{in} through which a path can enter domain d:

$$V^d_{\text{in}} = \left\{j \in V : \exists\, (i,j)^{k_1} \in E^q \wedge (j,m)^{k_2} \in E^d, q \neq d\right\}.$$

Similarly, V^d_{out} are the nodes through which a path exits domain d:

$$V^d_{\text{out}} = \left\{j \in V : \exists\, (i,j)^{k_1} \in E^d \wedge (j,m)^{k_2} \in E^q, q \neq d\right\}.$$

The inter-domain graph $\mathcal{G}_{\mathcal{D}}$ is then defined as follows.

Definition 3 (Inter-domain graph $\mathcal{G}_{\mathcal{D}}$)**.** *Let* $\mathcal{G}_{\mathcal{D}} = (\mathcal{D}, E_{\mathcal{D}})$ be a directed graph, where $(d, q) \in E_D$ if and only if $V^d_{\text{out}} \cap V^q_{\text{in}} \neq \emptyset$. We denote $\mathcal{G}_{\mathcal{D}}$ as the inter-domain graph.

In other words, there is an edge (d, q) in the inter-domain graph $\mathcal{G}_{\mathcal{D}}$ whenever there exists a path on $\mathcal{G}$ that crosses domains d and q in this order. Several examples of inter-domain graphs $\mathcal{G}_{\mathcal{D}}$ are disseminated throughout the text, for example, in Figures 4–7.

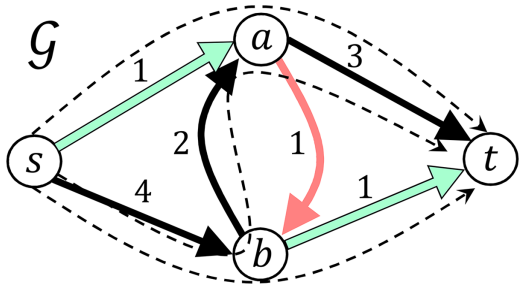

FIGURE 2 Instance of IDPC-DU problem on colored graph $\mathcal{G}$. Edges are labeled with their corresponding costs. Path $\{s, a, b, t\}$ is min-cost but infeasible as it enters the green domain twice, thus violating the Domain Uniqueness (DU) constraint. All the other paths from s to t, in dashed black lines, are feasible. Thus, $\mathbf{p}^* = \{s, a, t\}$ is the IDPC-DU optimal path [Color figure can be viewed at wileyonlinelibrary.com]

3.1 | Properties of $\mathcal{G}_\mathcal{D}$

We remark that $\mathcal{G}_\mathcal{D}$ gives a succinct description of the network $\mathcal{G}$ as, typically, $|\mathcal{D}| \ll |V|$. Moreover, $\mathcal{G}_\mathcal{D}$ is fairly *stable with respect to node/edge failures*, as $(d, q) \in E_D$ whenever there is *at least one* node in $V^d_{\text{out}} \cap V^q_{\text{in}}$. In other words, an edge (d, q) of the inter-domain graph fails only when all possible connections between domains d and q fail, which is typically unlikely. Notably, $\mathcal{G}_\mathcal{D}$ is also *independent of edge costs*. Thus, $\mathcal{G}_\mathcal{D}$ can be considered as *quasi-static* with respect to input parameters. We will leverage these properties later on, in Section 7, to claim that clustering can be performed offline during a pre-processing phase.

3.2 | System architecture

From a system architecture viewpoint, this article addresses the hierarchical PCE (h-PCE) architecture [9], depicted in Figure 1 for inter-domain path computation. The network links belong to different domains, which are interconnected via specific routers commonly called *border routers*. Each domain is managed by a so-called Child PCE, being in charge of intra-domain path computation. Children PCEs are all connected to a single Parent PCE, being responsible for the computation of the optimal inter-domain path under the DU constraint. Once the Parent PCE has solved IDPC-DU, it will contact the Children PCEs of the domains traversed by the path, requiring them to instantiate the optimal IDPC-DU path inside the corresponding domains via the Path Computation Element Communication Protocol (PCEP).

Drawing a parallel between this system model and the formal problem statement of the previous section, we can say that $\mathcal{G}$ incorporates all the network information that the Parent PCE gathers from Children PCEs. The nodes V represent routers; the colored edges E describe their connectivity, that is, the parallel edge $(i, j)^k$ of color d denotes the presence of a path from router i to router j within domain d; the weight $w^k_{i,j}$ associated the k-th parallel edge $(i, j)^k$ of color d equals the minimum cost (or minimum *delay*, depending on the application) to reach node j in graph $\mathcal{G}$ by only using edges inside domain d. The weight $w^k_{i,j}$ is computed by the Child PCE managing domain d and reported to the Parent PCE. We remark that the existence of parallel edges of different colors between two nodes describes the situation where two routers are connected via two or more domains. Finally, the DU constraint in Definition 1 claims that, for each connection request, at most one LSP can be installed in the LSRs belonging to one domain, which helps to reduce the packet processing delay and the overhead signaling traffic [9, 15] (please see the Introduction for further details).

In the next sections we will mainly deal with IDPC-DU computational aspects, ranging from complexity analysis to an offline clustering method allowing us to reduce the problem and alleviate the complexity of its solution at run-time. In Section 8 we discuss how our findings apply to h-PCE architecture and may pave the way for the development of a semi-hierarchical PCE architecture.

4 | COMPLEXITY OF IDPC-DU

In this section we start analyzing IDPC-DU, introduced in Section 3, by proving its hardness. Next, in Section 5 we will discuss a dynamic programming approach to solve IDPC-DU, which motivates our domain clustering approach studied in Section 6. This will allow us to reduce the complexity of the IDPC-DU solution at run-time.

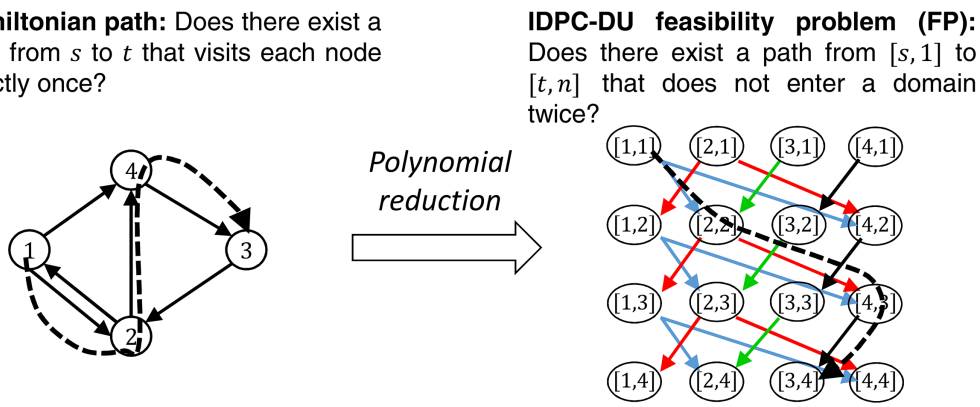

FIGURE 3 Polynomial reduction of Hamiltonian path to our IDPC-DU problem [Color figure can be viewed at wileyonlinelibrary.com]

We now prove the NP-completeness of the *feasibility* problem associated with the IDPC-DU *optimization* problem. We call it FP, standing for "Feasibility Problem".

Definition 4 (FP: Feasibility Problem)**.** Given a source node s and a destination node t, answer whether there exists a path $\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G})$ from s to t and fulfilling the Domain Unicity constraint.

Our NP-completeness proof hinges on a polynomial reduction of the Hamiltonian path problem to our FP problem. We recall that the Hamiltonian path problem answers the following question: "Given a directed graph and two nodes s and t, does there exist a path from s to t that visits each node exactly once?".

Theorem 1 (Complexity)**.** *The feasibility problem FP is* NP*-complete.*

Proof. It is easy to see that FP is in NP. In fact, given a colored directed graph $G'' = (V'', E'')$, we can then check in polynomial time that a path $\mathbf{p}$ from s to t of nodes fulfills the DU constraint.

Now, we will find a polynomial reduction of the Hamiltonian path problem, which is known to be NP-complete [13], to our IDPC-DU problem. We refer to Figure 3 for a graphical example.

Given an arbitrary graph $G' = (V', E')$ with two distinct nodes s and t, on which we will compute a Hamiltonian path, we construct a colored directed graph $G'' = (V'', E'')$ as follows. Let n be the number of nodes in G'. Let us label the nodes of G' as $\{1, \ldots, n\}$. We associate to each node x of graph G' exactly n nodes in G'' denoted by $[x, 1]$, $[x, 2]$, ..., $[x, n]$. Moreover, for each directed edge (x, y) in G' we construct $n - 1$ directed edges of color d_x in the graph G'', of the kind $([x, i], [y, i + 1])$, for $1 \leq i < n$. Hence, all outgoing edges have the same domain. Such transformation from G' into G'' can be computed in time polynomial in $|E'|$.

We claim that G' contains a Hamiltonian path from s to t if and only if G'' contains a feasible path $\mathbf{p}$ from $[s, 1]$ to $[t, n]$, that is, fulfilling the DU constraint. First, suppose that there is a Hamiltonian path $\mathcal{H} = \{s, x_2, \ldots, x_{n-1}, t\}$ in G'. We will build a feasible path $\mathbf{p}$ from node $[s, 1]$ to node $[t, n]$ as follows: $\mathbf{p} = [s, 1], [x_2, 2], \ldots, [x_{n-1}, n-1], [t, n]$. By the construction of graph G'', we deduce that $\mathbf{p}$ fulfills the DU constraint.

Conversely, suppose that $\mathbf{p}$ is a path from $[s', 1]$ to $[t', n]$ that fulfills the DU constraint. First, we can notice that by construction graph G'' satisfies the following properties: (1) For each $i = 1, 2, \ldots, n$, the set $C_i = \{[x, i] : x \in V''\}$ of nodes in G'' disconnects graph G'' into two connected components each containing only one of the nodes $[s, 1]$ and $[t, n]$; (2) For each node x in V', for each $i = 1, \ldots, n$, node $[x, i]$ has outgoing edges only in domain d_x.

Property (1) implies that $\mathbf{p}$ contains at least one node in C_i, for each $i = 1, 2, \ldots, n$, hence at least n nodes in $C = \bigcup_{1 \leq i \leq n} C_i$. Moreover, once $\mathbf{p}$ has left a domain, it cannot enter it later on; thus property (2) implies that $\mathbf{p}$ exactly contains n nodes in C and one node in $\{[x, i] : i = 1, 2, \ldots, n\}$ for each $x \in V$. We can then write $\mathbf{p} = [s, 1], [x_2, 2], \ldots, [x_{n-1}, n-1], [t, n]$. By construction of graph G'', the sequence of nodes $\mathcal{H} = \{s, x_2, \ldots, x_{n-1}, t\}$ is a path on G'. $\mathcal{H}$ contains n nodes; moreover, by the DU property of $\mathbf{p}$, $\mathcal{H}$ contains each node exactly once. Thus, $\mathcal{H}$ is a Hamiltonian path from s to t. This completes the proof. ∎

We now show that, unfortunately, IDPC-DU cannot even be approximated in polynomial time. Our proof is based on the classic *gap-introducing reduction* technique (see [23], Chapter 29).

Proposition 1 (Inapproximability)**.** *Unless* P = NP*, there is no polynomial-time approximation algorithm with approximation ratio ρ for IDPC-DU, for any constant $\rho \geq 1$.*

Proof. Assume by contradiction that there is a polynomial-time approximation algorithm $\mathcal{A}$ with approximation ratio ρ for IDPC-DU, with $\rho \geq 1$. We shall then show that $\mathcal{A}$ can be used for solving the FP problem (which is NP-complete) in polynomial time, thus implying P = NP.

We first reduce the FP problem to IDPC-DU. FP accepts as input the directed colored graph $\mathcal{G}$, while IDPC-DU works on the edge-weighted directed colored graph $\mathcal{G}'$ that we construct as follows. $\mathcal{G}'$ has the same set of nodes as $\mathcal{G}$. Each edge in $\mathcal{G}$ also belongs to $\mathcal{G}'$ and a unitary cost is assigned to it. If there exists no direct edge between source and destination nodes s and t in $\mathcal{G}$, then the edge (s,t) is added to $\mathcal{G}'$, with arbitrary color and weight $\rho(n-1)+1$.

Note that IDPC-DU is always feasible on $\mathcal{G}'$, and we can construct $\mathcal{G}'$ in polynomial time. Moreover, if $\mathcal{G}$ contains a feasible path $\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G})$ then $\mathbf{p}$ is also feasible on $\mathcal{G}'$ (ie, $\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G}')$) and its cost is at most $n-1$, since $\mathbf{p}$ has at most $n-1$ edges. Conversely, if $\mathcal{G}$ does not contain any feasible path, then the only feasible path in $\mathcal{G}'$ is the direct path from s to t, with cost $\rho(n-1)+1$.

It follows that algorithm $\mathcal{A}$ can be used for solving FP on graph $\mathcal{G}$. In fact, when run on $\mathcal{G}'$, algorithm $\mathcal{A}$ must return a solution of cost $\leq n-1$ if $\mathcal{G}$ contains a feasible path; else, it returns a solution of cost $\rho(n-1)+1$. This would imply P = NP, thus the thesis is proved. ∎

On the bright side, in Sections 5–7 we will present some positive results that will help simplify the solution of IDPC-DU. We will first observe that the real computational bottleneck is represented by the number of domains $|\mathcal{D}|$, and we will propose a domain clustering technique to alleviate this problem.

4.1 | Domains on nodes: a special case

In this section we deal with a variant of the model described above, where domains are defined on nodes rather than on edges. In other words, nodes are partitioned into domains, while edges are no longer colored. We then call IDPC-DU (nodes) the corresponding problem of computing a min-cost path from a source s to a destination t under the constraint that the sequence of visited *nodes* does not enter twice a domain.

We now show that IDPC-DU (nodes) can be seen as a special case of our IDPC-DU problem, as it is computationally equivalent to a simplified version of IDPC-DU, where the outgoing edges of each node all belong to a single domain.

Lemma 1. *Consider the simplified version of IDPC-DU where the outgoing edges of each node all belong to a single domain, that is,*

$$\forall\, i \in V,\ \exists\, d \in \mathcal{D} :\ (i,j)^k \in E^d\ \forall\, j,k. \tag{1}$$

Then, the problems IDPC-DU and IDPC-DU(nodes) are polynomially reducible one to the other.

Proof. We start by proving that there exists a polynomial reduction from IDPC-DU (nodes) to IDPC-DU. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ represent an instance of IDPC-DU (nodes). Let $\left\{\tilde{V}^d\right\}_{d\in\mathcal{D}}$ be the set of domains, partitioning the set of nodes $\tilde{V}$. Let $s,t \in \tilde{V}$ be the source and the destination node, respectively. Then, we can construct an instance of IDPC-DU in the following way. We first add a new node $t' \notin \tilde{V}$ and we define the graph $G = \left(\tilde{V} \cup \{t'\}, \tilde{E} \cup \left\{(\tilde{t}, t')\right\}\right)$, where the new edge $(\tilde{t}, t')$ has zero cost. The costs on all the other edges remain unchanged from $\tilde{G}$. Moreover, domains in G are defined on edges in the following way: if node v in $\tilde{G}$ belongs to domain V^d, then then all outgoing edges of v in G belong to the same domain E^d. In such a way, path $\tilde{\mathbf{p}} = [s, \ldots, t]$ is feasible in $\tilde{G}$ if and only if path $\mathbf{p} = [\tilde{\mathbf{p}}, t']$ is feasible in G. Moreover, $w(\tilde{\mathbf{p}}) = w(\mathbf{p})$. Therefore, a path $\tilde{\mathbf{p}}$ is optimal in $\tilde{G}$ if and only if its respective path $\mathbf{p}$ is optimal in G.

Conversely, let $G = (V, E)$ be an instance of IDPC-DU, with s,t as source and destination node, respectively. Then, we define an instance of IDPC-DU (nodes) as follows. Let $\tilde{G} = (V, E)$ be such that node $v \neq t$ in $\tilde{G}$ belongs

to domain V^d if all its outgoing edges belong to domain E^d. Moreover, destination node t is assigned to a new domain $V^{|\mathcal{D}|+1}$. All costs remain unchanged. Thus, any path $\mathbf{p}$ is feasible in G if and only it is feasible in $\tilde{G}$, and moreover $\mathbf{p}$ has the same cost in the two graphs. The thesis is proven. ∎

We notice that the NP-completeness proof of the IDPC-DU problem has been precisely carried out under the assumption that condition (1) holds. Thus, the following result directly stems from Lemma 1.

Corollary 1. *The feasibility problem associated to IDPC-DU(nodes) is* NP*-complete.*

Therefore, similar to Proposition 1, we can conclude that there is no polynomial-time approximation algorithm with approximation ratio ρ for IDPC-DU (nodes).

5 | NUMBER OF DOMAINS: A COMPUTATIONAL BOTTLENECK

The negative results of Section 4 claim that, as the network size grows in terms of number of nodes $|V|$ and domains $|\mathcal{D}|$, there is no hope to find the optimal IDPC-DU solution in reasonable time. On the other hand, we now show that the real computational bottleneck for IDPC-DU is really the number of domains $|\mathcal{D}|$. Therefore, our main idea will hinge on the fact that if we are able to artificially reduce $|\mathcal{D}|$ via some offline computations, then we can tame the inherent hardness of IDPC-DU.

5.1 | A dynamic programming (*DP*) approach

In order to demonstrate that the number of domains $|\mathcal{D}|$ is the real computational bottleneck for our problem, we now present a natural approach to solve IDPC-DU along the lines of the classic Bellman–Ford Dynamic Programming (DP) algorithm [2]. To this aim, we start by finding a convenient mapping between feasible paths, that is, fulfilling the DU constraint, and the so-called DP *states*. We map any path from source s to $v \in V$ to the state $\mathcal{S} = [v, \mathcal{H}, d]$, where $\mathcal{H} \subset \mathcal{D}$ is the set of domains visited by the path and $d \in \mathcal{H}$ is the last visited domain. We can conveniently interpret state $\mathcal{S}$ as the *history* of the path. The reader shall notice that several feasible paths may map to the same state $\mathcal{S}$.

Let us now delve into the details of our IDPC-DU dynamic programming algorithm DP. We denote $w^*(\mathcal{S})$ as the minimum cost of feasible paths mapping to state $\mathcal{S}$. We first set the cost of the initial state $S_s = [s, \emptyset, \emptyset]$ to $w^*(S_s) = 0$, since trivially the shortest path from s to s has null cost. Let us now consider the generic state $\mathcal{S} = [v, \mathcal{H}, d]$. Via a classic Bellman–Ford approach, we now express $w^*(\mathcal{S})$ using smaller subproblems. By the induction hypothesis we assume that we have already computed the min-cost $w^*([u, \mathcal{H}, d])$ and $w^*([u, \mathcal{H} \setminus \{d\}, d'])$, for all incoming nodes u and for all domains $d' \in \mathcal{D}$. By convention, an infinite cost is assigned to non-reachable states. Then, the classic Bellman–Ford DP principle allows us to characterize the min-cost $w^*[v, \mathcal{H}, d]$ via the following recursive equation:

$$w^*([v, \mathcal{H}, d]) = \min \begin{pmatrix} w^*([u, \mathcal{H}, d]) + w^k_{u,v},\ \forall\, u, k : (u, v)^k \in E^d \\ w^*([u, \mathcal{H} \setminus \{d\}, d']) + w^k_{u,v},\ \forall\, u, k : (u, v)^k \in E^d, \forall\, d' \in \mathcal{D} \end{pmatrix}. \tag{2}$$

Next, we observe that the set of feasible paths $\mathcal{F}_{s,t}(\mathcal{G})$ maps to a subset of $\cup_{\mathcal{H} \in 2^{\mathcal{D}}, d \in \mathcal{D}}[t, \mathcal{H}, d]$, called $\mathcal{S}_t$ and being the set of final states having destination node t as current node. It follows that we can solve IDPC-DU by computing the min-cost of the set of final states $\mathcal{S}_t$, that is,

$$\min_{\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G})} w(\mathbf{p}) = \min_{\mathcal{S} \in \mathcal{S}_t} w^*(\mathcal{S}) := w(\mathbf{p}^*). \tag{3}$$

In detail, in order to calculate $\min_{\mathcal{S} \in \mathcal{S}_t} w^*(\mathcal{S})$ one needs to solve the recursive Equation (2) via standard DP techniques. Finally, the optimal path $\mathbf{p}^*$ can be read backwards, as the succession of selected edges.

We remark that the solution of (2) can be sped up by pruning the states which are *dominated* by some other state. In this case, we say that state $\mathcal{S} = [u, \mathcal{H}, d]$ *dominates* state $\mathcal{S}' = [u', \mathcal{H}', d']$ whenever $u = u'$, $\mathcal{H} \subseteq \mathcal{H}'$ and $w^*(\mathcal{S}) < w^*(\mathcal{S}')$. In fact, if a state $\mathcal{S}$ is dominated by some other state, then $\mathcal{S}$ cannot lie on the optimal state sequence.

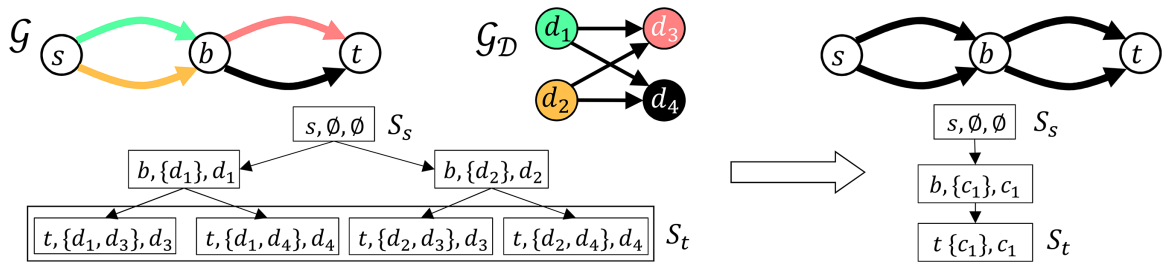

FIGURE 4 DP state size reduction via clustering: Since the inter-domain graph $\mathcal{G}_\mathcal{D}$ is acyclic, then all domains can be consider as one single (black) domain, also called a *cluster*, without loss of optimality (see Fact 1). Thus, IDPC-DU boils down to a simple min-cost path problem and the complexity of DP solution is reduced. (Left): Original topology $\mathcal{G}$ and the associated dynamic programming state tree. Each box represents a dynamic programming state $\mathcal{S}$. (Right): DP state transition space when all domains are aggregated into a single cluster [Color figure can be viewed at wileyonlinelibrary.com]

5.2 | Complexity and Number of Domains

Let us now assess the computational complexity to solve Equation (3) by calling Equation (2) recursively. To compute $w^*([v, \mathcal{H}, d])$ via Equation (2) we need $O(|V||\mathcal{D}|)$ operations. Since the total number of states is $O(|V|2^{|\mathcal{D}|}|\mathcal{D}|)$, the next result easily follows.

> **Lemma 2** (DP complexity)**.** *The computational complexity of the dynamic programming algorithm DP described in Section 5.1 is* $O(|V|^2 2^{|\mathcal{D}|}|\mathcal{D}|^2)$. *Hence, DP is Fixed Parameter Tractable (FPT) in the number of domains* $|\mathcal{D}|$.

Lemma 2 highlights the impact of the number of domains $|\mathcal{D}|$ on the computational complexity of our classic DP approach. It directly follows that DP has polynomial complexity if the number of domains is constant, that is, it is FPT with respect to $|\mathcal{D}|$. Such considerations call for a method to reduce the effective number of domains to be considered when solving IDPC-DU. In the next section we show that the inter-domain graph $\mathcal{G}_\mathcal{D}$, introduced in Section 3, contains precious information that can help us in this task. To catch a first glimpse of the reasons behind this, we now point out that in some (fortunate) cases we can reduce IDPC-DU to a simple min-cost problem.

> **Fact 1.** *If the inter-domain graph* $\mathcal{G}_\mathcal{D}$ *is acyclic, then* IDPC-DU *boils down to a min-cost path computation problem, solvable in polynomial time* [4].

In fact, if $\mathcal{G}_\mathcal{D}$ is acyclic then there is no path on $\mathcal{G}$ that visits the same domain twice, trivially. In this case, all domains can be in a sense "clustered" in a single domain without loss of optimality. To get an idea of the resulting state space reduction we refer to the simple example in Figure 4.

We observe that, informally speaking, if $\mathcal{G}_\mathcal{D}$ is acyclic then the IDPC-DU problem at hand is already *inherently simple*; in the next section we will introduce a new domain clustering approach that can, under some conditions, effectively *reduce* the hardness of IDPC-DU.

6 | PROPER DOMAIN CLUSTERING: A PROBLEM REDUCTION WITHOUT LOSS OF OPTIMALITY

The results of Section 4 highlight the hardness of IDPC-DU: the associated feasibility problem FP is NP-complete (Theorem 1), which makes the optimization problem IDPC-DU even inapproximable (Proposition 1). On the other hand, Lemma 2 confirms that the real bottleneck for IDPC-DU solution is the number of domains $|\mathcal{D}|$. Thus, if we figure out how to artificially reduce the effective number of domains to be considered for computing the optimal path, then IDPC-DU would be substantially easier to solve.

Motivated by this, in the current section we introduce the concept of *proper domain clustering*, allowing us to reduce the effective number of domains without losing the optimality of IDPC-DU solution. We start off in Section 6.1 by formally defining the concept of proper domain clustering. Next, in Section 6.2 we give some intuitions shedding light on the technical results presented in Section 6.3, where we provide a necessary and sufficient condition for a domain clustering to be proper.

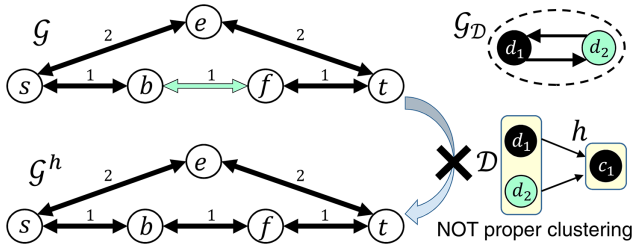

FIGURE 5 Instance of NOT proper clustering function h. In fact, $\mathbf{p} = [s, e, t]$ is the IDPC-DU optimal solution on $\mathcal{G}$, while $\mathbf{p}' = [s, b, f, t] \neq \mathbf{p}$ is optimal on the clustered graph $\mathcal{G}^h$. Note: edges are labeled with their costs [Color figure can be viewed at wileyonlinelibrary.com]

6.1 | Definitions

In this section we illustrate our concept of *proper* domain clustering. First, we provide some auxiliary definitions. A *cluster* is a subset of the set of domains $\mathcal{D}$, and we call $\mathcal{C}$ a partition of $\mathcal{D}$ into clusters. A clustering function is defined as follows.

Definition 5 (Clustering function h). *Let $\mathcal{C}$ be a partition of the set of domains into clusters. Let $h : \mathcal{D} \to \mathcal{C}$ be defined as follow: $h(d) = c$ whenever $d \in c$. Then, h is the* clustering function *associated to $\mathcal{C}$.*

We will say that domains in $h^{-1}(c) \subseteq \mathcal{D}$ are *aggregated* in the cluster $c \in \mathcal{C}$. We call h *trivial* if $|h(\mathcal{D})| = |\mathcal{D}|$. Next, we define $\mathcal{G}^h$ as the colored weighted graph associated to the clustering function h. $\mathcal{G}^h$ has the same nodes, edges and costs as $\mathcal{G}$, except for the color of edges which are clustered according to clustering function h.

Definition 6 (Clustered graph $\mathcal{G}^h$). Let $\mathcal{G}^h$ be a weighted colored multi-graph defined as follows. If $(i,j)^k \in E^d$ on graph $\mathcal{G}$, then $(i,j)^k \in E^{h(d)}$ on graph $\mathcal{G}^h$. Nodes and edge costs of $\mathcal{G}^h$ are as in $\mathcal{G}$. We call $\mathcal{G}^h$ the clustered graph associated to the clustering function h.

Importantly, we can now define a clustering function h as *proper* whenever IDPC-DU can be equivalently solved on the original graph $\mathcal{G}$ or on the (reduced) clustered graph $\mathcal{G}^h$.

Definition 7 (Proper clustering). The clustering function h is proper whenever

$$\arg\min_{\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G})} w(\mathbf{p}) = \arg\min_{\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G}^h)} w(\mathbf{p}), \quad \forall\, s, t \in V. \tag{4}$$

In the next sections we will shed light on the properties that an inter-domain graph $\mathcal{G}_\mathcal{D}$ must possess in order to admit a proper (and non trivial) domain clustering h.

6.2 | Insights on proper domain clustering

We preliminarily observed in Fact 1 (Section 5.2) that if the inter-domain graph $\mathcal{G}_\mathcal{D}$ is acyclic, then all domains can be safely aggregated into a single cluster; namely, $h : h(d) = h(d'), \forall\, d, d' \in \mathcal{D}$ is a *proper* clustering. Unfortunately, acyclic inter-domain graphs only arise in special cases. Therefore, we need a more powerful framework to guarantee proper aggregation. In order to provide to the reader some intuitions behind the technical conditions for proper clustering provided in Section 6.3, we now present three examples of clustering functions.

Let us begin with the instance shown in Figure 5. It is trivial to see that, in the original graph $\mathcal{G}$, path $\mathbf{p} = [s, e, t]$ is the IDPC-DU optimal (and the only feasible) solution, and its cost is 4. On the other hand, if the black domain d_1 and green domain d_2 are aggregated into a single black cluster, then *the set of feasible solutions expands*. This permits path $\mathbf{p}' = [s, b, f, t]$ to become optimal, as its cost (= 3) is minimized while fulfilling the DU constraint on the clustered graph $\mathcal{G}^h$. However, $w(\mathbf{p}') \neq w(\mathbf{p})$, thus the clustering function h is *not proper*. This first example seems to suggest that (1) *there cannot be cycles in the inter-domain graph $\mathcal{G}_\mathcal{D}$ restricted to each cluster*.

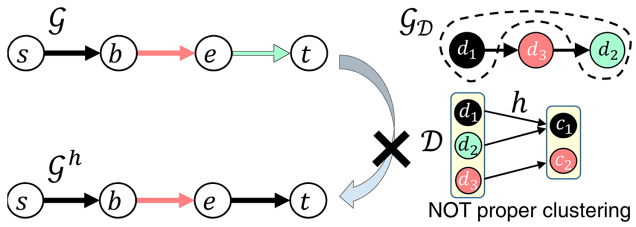

FIGURE 6 Instance of NOT proper clustering function h. In fact, $\mathbf{p} = [s, b, e, t]$ is the optimal solution on $\mathcal{G}$, but IDPC-DU becomes infeasible on the clustered graph $\mathcal{G}^h$ [Color figure can be viewed at wileyonlinelibrary.com]

We next move on to the instance shown in Figure 6. In this case, aggregating the black domain d_1 and green domain d_2 into a single black cluster c_1 leads the IDPC-DU feasibility set to shrink. In fact, path $\mathbf{p} = [s, b, e, t]$ is the only feasible solution in the original graph $\mathcal{G}$, while the feasibility set is empty on the clustered graph $\mathcal{G}^h$. Hence, also in this case, h is *not proper*. We notice that this second example advises that, for a clustering function h to be proper, (2) *there cannot exist a path in* $\mathcal{G}_\mathcal{D}$ *that exits and re-enters the same cluster via two different domains*.

Finally, let us now consider the more involved example in Figure 7. Here, paths may visit the black domain d_1 first and then the green domain d_2; conversely, no path can possibly visit domains d_2 and d_1 in this order. In fact, (1) there are no cycles in the subgraph of $\mathcal{G}_\mathcal{D}$ restricted to domains d_1 and d_2. For this reason, if a path is feasible on the clustered graph $\mathcal{G}^h$, where domains d_1 and d_2 cannot be distinguished, then it is also feasible on the original graph $\mathcal{G}$.

Moreover, let us assume that a path violates the DU constraint on the clustered graph $\mathcal{G}^h$. Then, it necessarily visits domains in the order d_2, d_3, d_2 or else d_3, d_2, d_3 on the original graph $\mathcal{G}$. We can verify that this is equivalent to the claim that (2) there is no path in $\mathcal{G}_\mathcal{D}$ from d_1 to d_2 that visits domains outside the cluster c_1. We further notice that a path crossing domains d_2, d_3, d_2 or else d_3, d_2, d_3 in this order on $\mathcal{G}$ also violates the DU constraint on $\mathcal{G}^h$, as it would cross cluster c_1 twice. Therefore, a path not in the feasibility set $\mathcal{F}_{s,t}(\mathcal{G}^h)$ does not belong to $\mathcal{F}_{s,t}(\mathcal{G})$ either, for all s, t. We conclude that $\mathcal{F}_{s,t}(\mathcal{G}^h) = \mathcal{F}_{s,t}(\mathcal{G})$ for all $s, t \in V$ and the clustering function h is *proper*.

In the next section we will formalize these intuitions and present a necessary and sufficient condition for a clustering function to be proper.

6.3 | Necessary and sufficient condition for proper clustering

The examples provided in Section 6.2 suggest that an aggregation function h is proper whenever the conditions (i), (ii), previously introduced only informally, are jointly fulfilled. We now formalize them as the IEC condition.

Condition 1 (IEC)**.** Let h be a clustering function and let $c \in \mathcal{C}$ be a cluster. We denote by $\mathcal{G}_\mathcal{D}(h^{-1}(c))$ the induced subgraph of $\mathcal{G}_\mathcal{D}$ restricted to the aggregated domains $h^{-1}(c) \subseteq \mathcal{D}$. Then, for every cluster $c \in \mathcal{C}$ the following two conditions are jointly satisfied:

1. *(Intra-Cluster)* The subgraph $\mathcal{G}_\mathcal{D}(h^{-1}(c))$ is acyclic;
2. *(Extra-Cluster)* After removing all edges in $\mathcal{G}_\mathcal{D}(h^{-1}(c))$ from $\mathcal{G}_\mathcal{D}$, all pairs of different domains belonging to $h^{-1}(c)$ are disconnected.

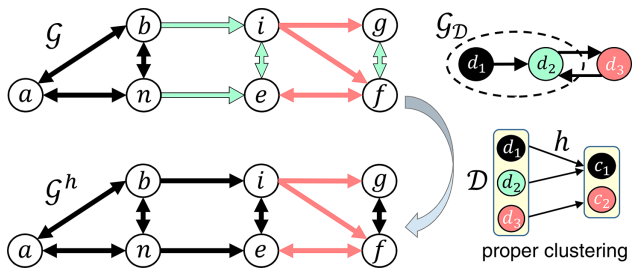

FIGURE 7 Instance of proper clustering function h. Solving IDPC-DU on the original graph $\mathcal{G}$ is equivalent to solving it on the clustered graph $\mathcal{G}^h$ [Color figure can be viewed at wileyonlinelibrary.com]

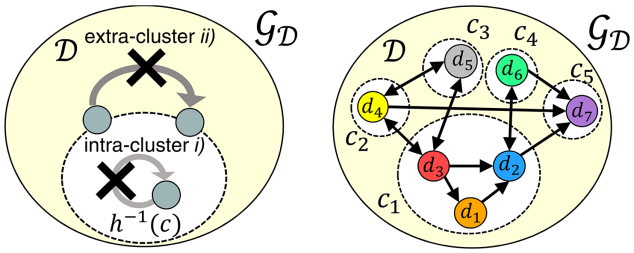

FIGURE 8 IEC condition: (left) A pictorial representation of the IEC condition, being sufficient (Theorem 2) and necessary (Theorem 3) for a domain clustering h to be proper. (Right) Example of proper domain aggregation. Clusters are drawn in dashed circles [Color figure can be viewed at wileyonlinelibrary.com]

We refer to Figure 8 for a pictorial illustration of the IEC condition. Please note that the extra-cluster condition (2) is equivalent to the following, more informal one: Let d, d' be two different domains belonging to the same cluster c. Then, there exists no path in $\mathcal{G}_\mathcal{D}$ connecting d and d' only through domains belonging to different clusters.

We now claim that IEC is a sufficient condition for an aggregation function h to be proper.

Theorem 2 (IEC is sufficient)**.** *If a clustering function h fulfills the IEC condition, then h is proper.*

Proof. We will prove that, under the IEC condition, $\mathcal{F}_{s,t}(\mathcal{G}) = \mathcal{F}_{s,t}(\mathcal{G}^h)$ holds for all $s, t \in V$. Then, Equation (4) is also verified. Before that, we observe that the extra-cluster IEC condition (ii) claims that there does not exist a path in the inter-domain graph $\mathcal{G}_\mathcal{D}$ of the kind $d_1, d_2, \ldots, d_j$ where $d_1 \in h^{-1}(c)$, $d_2 \notin h^{-1}(c)$, $d_j \in h^{-1}(c)$ and $d_1 \neq d_j$. Then, we notice that if node d_2 is not reachable from node d_1 on graph $\mathcal{G}_\mathcal{D}$, then there is no path on the original graph $\mathcal{G}$ that visits domain d_1 first and d_2 later on.

Let $\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G})$. If $\mathbf{p}$ only visits domains in $h^{-1}(c)$, then clearly $p \in \mathcal{F}_{s,t}(\mathcal{G}^h)$ as well. Else, let d be the last domain visited by $\mathbf{p}$ before $\mathbf{p}$ leaves the set of domains $h^{-1}(c)$. Then, $\mathbf{p}$ can revisit $h^{-1}(c)$ later on only via the same domain d, thanks to the extra-cluster IEC condition (2). Nevertheless, this is impossible since the DU constraint would be violated. Thus, $\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G}^h)$ since p does visit $h^{-1}(c)$ twice, for all c.

Conversely, let $\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G}^h)$. Then $\mathbf{p}$ visits each cluster $c \in h(\mathcal{D})$ at most once. Let $\mathbf{p}'$ be the subpath of $\mathbf{p}$ that only crosses cluster c. If $\mathbf{p}' = \emptyset$ then there is nothing to prove. Else, let us now translate $\mathbf{p}'$ onto the original graph $\mathcal{G}$ (and this is possible since $\mathcal{G}$ and $\mathcal{G}^h$ share the same set of nodes and edges). By intra-cluster IEC condition (1), $\mathbf{p}'$ does not visit any domain in $h^{-1}(c)$ more than once. Since the same reasoning holds for all clusters $c \in h(\mathcal{D})$, then $\mathbf{p}$ cannot visit any domain twice. Thus, $\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G})$. This proves the thesis. ∎

In order to prove the necessity of the IEC condition we need to make the technical assumption that for any feasible sequence of domains in the inter-domain graph $\mathcal{G}_\mathcal{D}$ there exists a path in the original graph that visits the domains in the same order.

Assumption 1. *Let $\{d_1, d_2, \ldots, d_k\}$ be a walk in $\mathcal{G}_\mathcal{D}$, where $d_i \neq d_j$ for all $i \neq j$, except possibly for $i = 1$ and $j = k$. Then there exists a simple path on $\mathcal{G}$ that visits domains $\{d_1, d_2, \ldots, d_k\}$ in this order.*

Assumption 1 is satisfied if, for instance, the intra-domain subgraph (V, E^d) is strongly connected for each domain $d \in \mathcal{D}$. We now show that, under Assumption 1, the IEC condition is also *necessary* for a clustering function to be proper.

Theorem 3 (IEC is necessary)**.** *Let $\mathcal{G}$ be a directed and colored graph and let $\mathcal{G}_\mathcal{D}$ be its inter-domain graph. Suppose that Assumption 1 holds. If a clustering function h does not fulfill the IEC condition, then there exists an instance of edge costs $\left\{w_{i,j}^k\right\}_{i,j,k}$ under which h is not proper.*

Proof. Assume first that intra-cluster IEC condition (1) does not hold. Then, for some $c \in h(\mathcal{D})$, there exists a simple cycle $\{d_1, \ldots, d_k\}$, with $d_1 = d_k$, in the subgraph $\mathcal{G}_\mathcal{D}$ restricted to the nodes $h^{-1}(c)$. By Assumption 1 there exists an acyclic path $\mathbf{p}$ constituted of n edges on $\mathcal{G}$ that visits domains $\{d_1, \ldots, d_k\}$, with $d_i \in \mathcal{D}$, $\forall\, i$, in this order. Let s and t be the initial and final nodes of path $\mathbf{p}$. Clearly $\mathbf{p} \notin \mathcal{F}_{s,t}(\mathcal{G})$, as domain $d_1 \in \mathcal{D}$ is visited twice, while $\mathbf{p} \in \mathcal{F}_{s,t}(\mathcal{G}^h)$ since it visits only cluster $c \in h(\mathcal{D})$. Assume that each edge belonging to path $\mathbf{p}$ has cost equal

to $\underline{w}/2n$, where $\underline{w} = \min_{(i,j)^k \notin \mathbf{p}} w^k_{i,j}$. We assign any real cost to all the remaining edges. Then $\mathbf{p}$ is the path with minimum cost in the set $\mathcal{F}_{s,t}(\mathcal{G}^h)$, thus in this case the domain clustering h is not proper.

Let us now assume that the extra-cluster IEC condition (2) does not hold, that is, there is a cluster $c \in h(\mathcal{D})$ such that it is possible to find a path $\mathbf{p}$ in graph $\mathcal{G}_\mathcal{D}$, say $\{d_1, d_2, \ldots, d_k\}$, where $d_1, d_k \in h^{-1}(c)$, $d_2 \notin h^{-1}(c)$, and $d_1 \neq d_k$. Then, by Assumption 1 there also exists an acyclic path $\mathbf{p}'$ with n edges on graph $\mathcal{G}$ that visits domains $\{d_1, d_2, \ldots, d_k\}$ in this order. Let s and t the initial and final nodes of $\mathbf{p}$, respectively. Clearly, $\mathbf{p} \notin \mathcal{F}_{s,t}(\mathcal{G}^h)$ since it visits cluster $c \in h(\mathcal{D})$ twice. Similar to before, we assume that each edge belonging to path $\mathbf{p}'$ has cost equal to $\underline{w}'/2n$, where $\underline{w}' = \min_{(i,j)^k \notin \mathbf{p}'} w^k_{i,j}$. We assign any real cost to all the remaining edges. Then $\mathbf{p}'$ is the min-cost path in the set $\mathcal{F}_{s,t}(\mathcal{G})$. Hence, h is not proper and this completes the proof. ∎

In conclusion, Theorems 2 and 3 jointly state that the IEC condition is necessary and sufficient for a clustering function h to be proper.

Remarkably, the IEC condition can be verified on the inter-domain graph $\mathcal{G}_\mathcal{D}$, which can be considered static with respect to the frequency at which new connection requests arrive (see also Section 3). We then recommend that clustering be performed *offline*, during a pre-processing phase. We will examine this point in the next section.

7 | PRE-FILTERING AND CLUSTERING

After presenting some structural properties of proper clustering functions, in this section we assess via numerical evaluations the computational speed-up that the Parent PCE benefits from when using our approach at run-time. Before this, we propose in Section 7.1 a method to pre-filter the inter-domain graph $\mathcal{G}_\mathcal{D}$ to boost the ability to find proper aggregations. Both pre-filtering and clustering procedures are to be performed *offline*, during a pre-processing phase.

7.1 | Pre-filtering the inter-domain graph

As we showed in Section 6, the existence of non-trivial proper clustering functions depends solely on the connectivity properties of the inter-domain graph $\mathcal{G}_\mathcal{D}$. Informally speaking, as the inter-domain graph $\mathcal{G}_\mathcal{D}$ becomes more sparsely connected, proper clustering manages to reduce the IDPC-DU problem more efficiently. We then argue that the Parent PCE can optionally pre-process the inter-domain $\mathcal{G}_\mathcal{D}$ to *eliminate unnecessary edges* and effectively boost its ability to find a proper clustering. We denominate this procedure *pre-filtering*.

Let us now describe our pre-filtering idea. Consider connection requests whose source and destination nodes sit at the interior of two (not necessarily distinct) domains d and q, respectively. In other words, all outgoing edges of source node s belong to the same domain d and all incoming edges of destination node t are in the same domain q. Hence, any feasible path from s to t necessarily visits domain d first and domain q last. We then claim the following simple fact.

Fact 2. *If edge (d', d'') does not lie on any acyclic path from d to q on graph $\mathcal{G}_\mathcal{D}$ then no feasible path on $\mathcal{G}$ from node s to t will visit domains d' and d'' successively.*

Fact 2 suggests to eliminate preemptively such edges (d', d'') from $\mathcal{G}_\mathcal{D}$, and then to construct a database of pre-filtered versions of $\mathcal{G}_\mathcal{D}$ based on the source/destination domain pair (d, q), called $\mathcal{G}_\mathcal{D}(d, q)$. In such a way, graph $\mathcal{G}_\mathcal{D}(d, q)$ contains the minimal set of edges able to describe all the feasible (ie, acyclic) domain transitions. Domain clustering can be then performed on $\mathcal{G}_\mathcal{D}(d, q)$, after selecting the appropriate (d, q) pair for each connection request. These steps are summarized in Figure 9 and the pre-filtering procedure is summarized below.

Input: Inter-domain graph $\mathcal{G}_\mathcal{D}$.

Output: Filtered inter-domain graph $\mathcal{G}_\mathcal{D}(d, q)$, for all $d, q \in \mathcal{D}$.

Inter-domain graph pre-filtering: For all pairs of domains $d, q \in \mathcal{D}$, construct the pre-filtered inter-domain graph $\mathcal{G}_\mathcal{D}(d, q) = (\mathcal{D}, E_\mathcal{D}(d, q))$ as follows:

$$(d', d'') \in E_\mathcal{D}(d, q) \;\Leftrightarrow\; (d', d'') \in E_\mathcal{D} \text{ lies on a simple path from } d \text{ to } q \text{ on } \mathcal{G}_\mathcal{D}. \tag{5}$$

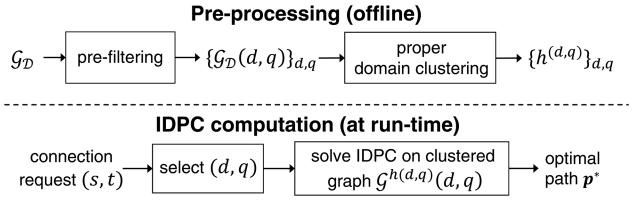

FIGURE 9 Pre-processing (offline): pre-filtering and domain clustering. **IDPC-DU** computation (at run-time): when a new connection request (s,t) appears, IDPC-DU is solved on the pre-filtered and clustered graph $G^{h(d,q)}(d,q)$

We remark that the graph pre-filtering problem can be solved in polynomial time. In fact, a possible approach to decide whether an edge lies on a simple path between two nodes, required in (5), is as follows (see, eg, [22]). First, one solves a max-flow problem between d and q on the inter-domain graph $\mathcal{G}_\mathcal{D}$, by assigning a positive lower bound to the value of the flow on edge (d', d''). A unitary capacity is appended to each edge. Then, the edge (d', d'') is added to the pre-filtered set of edges $E_\mathcal{D}(d,q)$ whenever it is possible to extract a simple path from d to q out of the edges with strictly positive max-flow.

Storing the pre-filtered inter-domain graph $\mathcal{G}_\mathcal{D}(d,q)$ for each pair of domains (d,q) requires at most $|\mathcal{D}|^4$ bits. In practice though, inter-domain graphs are sparse [5]; moreover, some domain source–destination pairs are more likely than others, which suggests that pre-filtered graphs should be stored only for those pairs. These two factors allow us to limit the memory overhead required by the pre-processing step.

We highlight that both the pre-filtering algorithm and the computation of proper domain clustering can be performed *offline*, at a much coarser time granularity than the run-time IDPC-DU calculations. In fact, both pre-processing procedures only require the inter-domain graph $\mathcal{G}_\mathcal{D}$ as input, which evolves at a much slower time scale than the underlying topology $\mathcal{G}$ and link costs w. Hence, we argue that in practice neither pre-filtering nor domain clustering must respect stringent requirements on computation speed.

7.2 | Pre-filtering and clustering

Previous intuitions suggest that pre-filtering can indeed boost the ability of clustering. More specifically, it is interesting to observe that such a boosting ability depends on the treewidth (TW) of the inter-domain graph $\mathcal{G}_\mathcal{D}$. TW is a classic graph metric, defined as the largest node set size minus 1 in a tree decomposition of the graph [7] and it intuitively measures the similarity of the graph to a tree graph (trees have TW$=1$).

To grasp the idea behind this phenomenon, we refer to Figure 10: as the TW of the inter-domain graph decreases from the left instance to the right one, the pre-filtered inter-domain graph becomes more sparse and the possibility of finding a proper clustering is enhanced.

Our observations are supported by the numerical results shown in Figure 11A. There we outline the average number of clusters obtained after (1) pre-filtering the inter-domain graph $\mathcal{G}_\mathcal{D}$ and (2) clustering domains by verifying the IEC condition

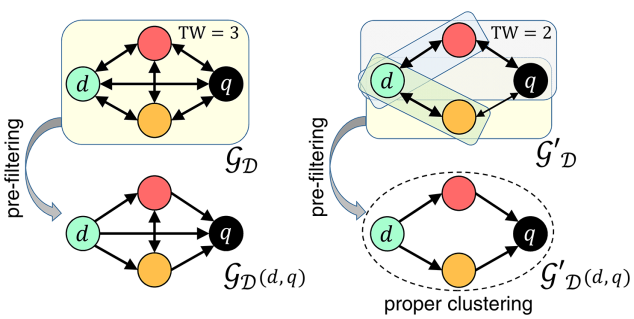

FIGURE 10 How pre-filtering boosts clustering, and impact of the TW. (Left): $\mathcal{G}_\mathcal{D}$ is a clique, its TW$= 3(= |\mathcal{D}| - 1)$ is maximal and no proper clustering is possible. Via pre-filtering some edges are removed from $\mathcal{G}_\mathcal{D}$, thus producing the filtered graph $\mathcal{G}_\mathcal{D}(d,q)$; yet, no proper clustering exists in $\mathcal{G}_\mathcal{D}(d,q)$ either. (Right): $\mathcal{G}'_\mathcal{D}$ still exhibits no proper aggregation, but its TW decreases to 2. Then, in this case the pre-filtering is beneficial as *all* domains can be properly aggregated into one cluster in $\mathcal{G}'_\mathcal{D}(d,q)$. Note: rounded boxes are nodes of the tree decomposition of $\mathcal{G}_\mathcal{D}$ and $\mathcal{G}'_\mathcal{D}$ [Color figure can be viewed at wileyonlinelibrary.com]

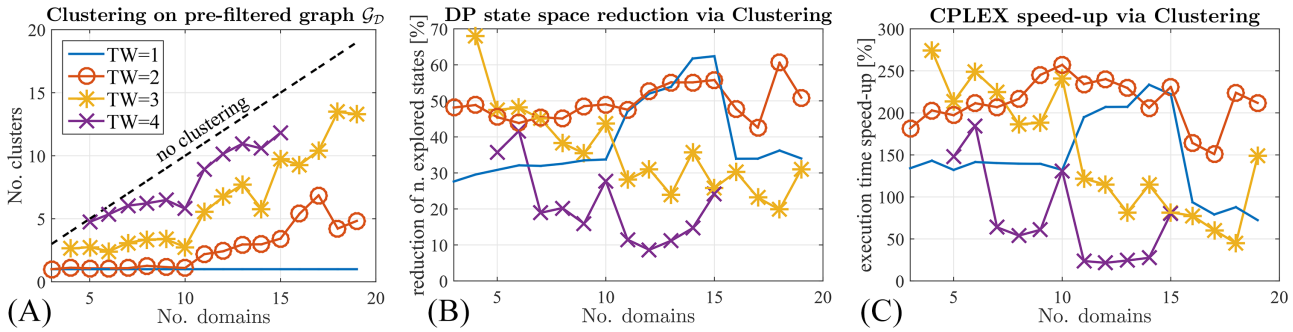

FIGURE 11 (A) Clustering on pre-filtered graphs: No. of clusters vs. No. of domains, for different values of the TW of the inter-domain graph $\mathcal{G}_\mathcal{D}$. We first pre-filtered $\mathcal{G}_\mathcal{D}$ and then we computed a proper clustering. We considered undirected and connected inter-domain graphs $\mathcal{G}_\mathcal{D}$. (B, C) Run-time speed-up via Clustering: Reduction in the state space/speed-up brought by clustering to DP (B) and CPLEX (C) solutions, on top of pre-filtering [Color figure can be viewed at wileyonlinelibrary.com]

on the pre-filtered graph $\mathcal{G}_\mathcal{D}(d, q)$, for different values of the TW. Let us now describe the simulation settings. We consider undirected inter-domain graphs, with at most 20 domains and TW of up to 4. To generate inter-domain graphs, we first produce Erdös-Rényi graphs with edge probability varying between 0.1 and 0.6.

Then, if necessary, a minimum number of inter-domain edges are added to $\mathcal{G}_\mathcal{D}$ in order to make it connected (This can be simply done as follows: identify the connected components, for each of them pick one node at random and connect them via a tree.). In order to compute proper clustering functions we use the following simple greedy algorithm. We initialize the set of clustered domains to be the empty set. We check whether the set formed by all remaining (ie, not clustered) domains satisfies the IEC condition. If yes, a new cluster is formed, otherwise we decrement the size of the set by one unit and we resume the search, until a smaller cluster is formed. The algorithm iterates until all domains are assigned to a cluster. We remark that, in the worst case, this simple algorithm ends up performing an exhaustive search over all possible subsets of the set of domains $\mathcal{D}$. Moreover, by its greedy nature, it tends to form clusters with unbalanced size. For the record, in all the experiments the greedy algorithm found a non-trivial clustering function in less than 2 minutes on an Intel(R) Core(TM) i7-5600U CPU @2.60GHz with 8GB RAM. Yet, the design of a provably good clustering algorithm is beyond the scope of this article, and it is part of future work.

We notice that in the extreme case where $\mathcal{G}_\mathcal{D}$ is a tree (TW$=1$), then $\mathcal{G}_\mathcal{D}(d, q)$ becomes a directed line from d to q and all domains can be properly aggregated. Then, IDPC-DU can be solved as a min-cost problem and it thus becomes *polynomial*. We finally remark that, in order to highlight the boosting effect of pre-filtering on clustering, we have considered the most challenging situation for clustering, that is, when the inter-domain graph is undirected. Moreover, in order to avoid trivial clustering solutions, we assume that the inter-domain graph is connected. In fact, if multiple connected components are present in $\mathcal{G}_\mathcal{D}$, then one can trivially select one domain out of each component and cluster them together, hence forming a proper clustering.

7.3 | Run-time computation speed-up via clustering

According to Figure 9, when a new connection request from s to t appears, the initial and final domains d and q are first selected. Then, the edges E on the original topology $\mathcal{G} = (V, E)$ are filtered according to the pre-filtered inter-domain graph $\mathcal{G}_\mathcal{D}(d, q)$. The modified graph $\mathcal{G}(d, q) = (V, E(d, q))$ is so produced. Then, the optimal IDPC-DU path can be computed on the domain-clustered graph $\mathcal{G}^{h(d,q)}(d, q)$, where $h(d, q)$ is the pre-computed proper clustering function.

Before illustrating the potential of our domain clustering concept via numerical simulations we provide a methodological remark. As we proved in Theorems 2 and 3, the ability of clustering only depends on some topological property of the inter-domain graph $\mathcal{G}_\mathcal{D}$ described by IEC, and not, for instance, on the size of the original graph $\mathcal{G}$. On the other hand, we already noticed in Figure 11A that TW is the graph concept that intuitively captures the property we seek. For this reason, we classify graph instances with respect to the TW of the corresponding inter-domain graph.

In Figure 11B, C we show the benefits brought by clustering to IDPC-DU computations at run-time, for classes of graphs with different TW. The inter-domain graph $\mathcal{G}_\mathcal{D}$ has already been processed via pre-filtering; hence, the complexity reduction we show is *purely* due to clustering, on top of pre-filtering that is performed by default.

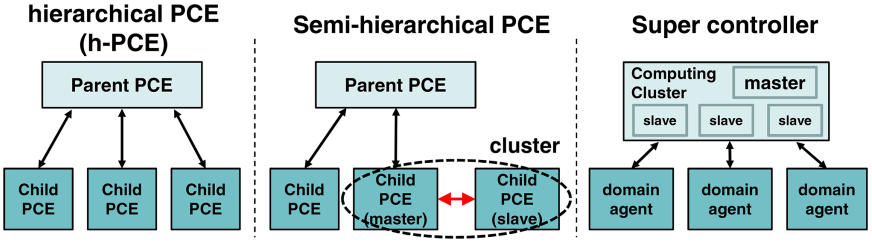

FIGURE 12 Architectures our domain clustering method may apply to [Color figure can be viewed at wileyonlinelibrary.com]

We assess the reduction in the number of explored states produced by clustering for the Dynamic Programming (DP) algorithm described in Section 5, as well as the speed-up (computed as $(t - t^h)/t^h \times 10^2$, where t (t^h) is the execution time to solve IDPC-DU on $\mathcal{G}$ ($\mathcal{G}^h$); DP state reduction is defined analogously) to CPLEX (IBM ILOG CPLEX Optimization Studio) solution of the ILP formulation described in the Appendix. Please note that we choose the state space reduction metric for the DP algorithm since it is independent of the implementation details and it is roughly proportional to the execution time.

Let us provide the details of the simulation settings. Given an inter-domain graph $\mathcal{G}_\mathcal{D}$, produced as explained in Section 7.2, the corresponding graph $\mathcal{G}$ was generated as follows. For each domain d we created a set V_d containing 50 nodes, being fully connected among them via edges of the same domain d. In order to form inter-domain connections, for each edge (d, d') in the inter-domain graph $\mathcal{G}_\mathcal{D}$ and for each pair of nodes $(i,j) : i \in V_d, j \in V_{d'}$, a directed edge of color d was added to $\mathcal{G}$ with a certain probability, ranging between 0.2 and 0.5. Each point in Figure 11A–C results from the average over 150 different network topologies generated as above, with the same number of domains and TW.

We observe that the improvement brought by our clustering approach is considerable both for CPLEX—with time execution speed-up ranging from 50% to 250%—and for the DP algorithm—with average state space reduction between 10% and 70% and standard deviation between 4% and 40%. To provide a rough order of magnitude, we report that the CPLEX execution time (on the original graph, without clustering) lies in the range 0.5–5 seconds for the medium-small network experiments considered in this article. We notice from Figure 11B, C that the improvement produced by clustering decreases with TW, as one would expect from the previous results in Figure 11A. The only exception is TW$= 1$, at which clustering often brings an improvement smaller than at TW$= 2, 3$. This is mainly because at TW$= 1$ the problem at hand is itself simpler: $\mathcal{G}_\mathcal{D}$ is by definition a tree and pre-filtering turns $\mathcal{G}_\mathcal{D}$ into a line. We finally observe that the *average* speed-up (or, similarly, state space reduction) does not show a monotonic behavior with respect to the number of domains, as it naturally reflects the non-monotonic relationship between number of clusters vs. number of domains shown in Figure 11A.

8 | DEPLOYMENT MODELS: A SHORT DISCUSSION

Our article mostly focuses on the theory of IDPC-DU computation, ranging from its complexity to a domain clustering concept allowing us to reduce the complexity of the problem at hand. From a system viewpoint, we already mentioned in the Introduction that proper clustering can help overcome the computational bottleneck at the Parent PCE's side in h-PCE. Moreover, we envision that our article will spur further research on what we call "semi-hierarchical" PCE architecture that allows us to tackle the signaling bottleneck of h-PCE. To summarize, we now briefly discuss three possible application scenarios of our work (see also Figure 12).

8.1 | Hierarchical architectures (h-PCE)

As shown in this article, in the case of h-PCE architecture our clustering approach can reduce the computational burden at the Parent PCE's side while preserving optimality of the IDPC-DU solution. More generally, we can envision that our solution may apply to controller hierarchies with several tiers, recently called *recursive* Software Define Network (SDN) architectures [19], where a controller computes an inter-domain path at the lower tier.

8.2 | Semi-hierarchical architectures

One limitation of h-PCE is that the Parent PCE has to maintain an up-to-date knowledge of all border-to-border paths in each domain. This causes congestion on the Children-to-Parent PCE communication channel as the number of domains grows. To

reduce this overhead, a promising direction is to aggregate domains into clusters, within which a *distributed* path computation procedure such as BRPC [24] operates. This engenders a hybrid, semi-hierarchical architecture, standing in between a pure hierarchical and a distributed one. In this approach, the Parent PCE offloads the intra-cluster path computation to the concerned Children PCEs, which in return send back abstracted topological information. This semi-hierarchical architecture can considerably reduce the signaling overhead, as first suggested in [14]. In this scenario, our notion of proper clustering allows us to pinpoint the clusters of domains that can function in fully distributed mode, without the solution optimality being affected.

8.3 | Super controller architectures

SDN controllers of upcoming generations are expected to adopt cluster computing architectures. In this context, each domain can be logically allocated to a pool of processors acting as a slave (cfr. Child PCE) and being subordinated to a master entity (cfr. Parent PCE). Processors are deployed in mini data centers and use modern distributed computing technologies (eg, loosely consistent databases, parallel computing, etc.) to fully exploit the computational power at hand. As examples, the open source ONOS and ODL controllers [16] are respectively using Hazelcast and Akka, two cluster computing management software systems dealing with concurrency, scalability and fault-tolerance. Then, similar to h-PCE, we foresee that our clustering procedure can help reduce the IDPC-DU solution complexity at the master's side.

9 | CONCLUSIONS

This article introduces the concept of proper domain clustering for the Inter-Domain Path Computation problem under the Domain Unicity constraint (IDPC-DU). Proper domain clustering allows us to reduce the complexity of IDPC-DU at run time, without loss of optimality, at the expense of some offline additional computations. In fact, IDPC-DU is provably hard, but its actual computational bottleneck is represented by the number of domains. In fact, we present a dynamic programming algorithm DP being FPT in the number of domains. Therefore, we focus on a clustering concept allowing us to artificially reduce the number of domains. We provide a necessary and sufficient condition (IEC) under which clustering is proper, that is, without loss of optimality. The IEC condition can be verified on the quasi-static inter-domain graph; thus, domain clustering can be performed during a pre-processing phase. We also devise a pre-filtering procedure to boost clustering performance. Finally, we validate numerically the reduction in the state space brought by proper domain clustering and we show its relation with the TW of the inter-domain graph.

From a system viewpoint, our work helps to alleviate Parent PCE's IDPC-DU computational burden at run-time in a hierarchical PCE (h-PCE) structure. This allows for an improvement of the scalability of h-PCE with respect to the number of domains, in terms of computational abilities.

Our work leaves some interesting theoretical and practical questions open for future research. In its current form, our approach is well suited to the IGP areas scenario, where some tens of areas (ie, domains) coexist. In this case, we tested via numerical simulations that a simple greedy algorithm suffices to find a proper clustering function. On the other hand, we still need fast clustering algorithms to let our approach scale up and be applied over large scale multi-domain networks. We conjecture that finding proper clustering functions is NP-complete; if so, then heuristics should be designed to relax the IEC condition, at the expenses of (rare, hopefully) DU violation occurrences.

Finally, our work prompts further research on semi-hierarchical PCE architectures that outperforms h-PCE in terms of scalability with respect to the number of domains, as briefly discussed in Section 8. In this setting, properly clustered domains communicate in a fully distributed manner, thus allowing one to alleviate signaling traffic between the Parent and the Children PCEs.

ORCID

Lorenzo Maggi http://orcid.org/0000-0003-3643-8349

REFERENCES

[1] J.K. Antonio, G.M. Huang, and W.K. Tsai, *A fast distributed shortest path algorithm for a class of hierarchically clustered data networks*, IEEE Trans. Comput. **41** (1992), 710–724.

[2] R. Bellman, *Dynamic programming*, Princeton University Press, Princeton, NJ, 1957.

[3] L. Buzzi et al., *Hierarchical border gateway protocol (HBGP) for PCE-Based multi-domain traffic engineering*, IEEE ICC, 2010.

[4] T.H. Cormen et al., *Introduction to algorithms*, McGraw-Hill Higher Education, Boston, 2001.

[5] F.D. Montgolfier, M. Soto, and L. Viennot, *Treewidth and hyperbolicity of the internet*, 2011 10th IEEE Int. Symp. Netw. Comput. Appl. (NCA), IEEE, 2011, pp. 25–32.

[6] M. D'Emidio et al., *On the clustered shortest-path tree problem*, ICTCS, 2016, p. 263.

[7] R. Diestel, *Graph theory*, Springer, New York, 2005.

[8] D. Estrin, Y. Rekhter, and S. Hotz, *Scalable inter-domain routing architecture*, Proc. ACM SIGCOM, 1992, pp. 40–52.

[9] A. Farrel and D. King, *The application of the path computation element architecture to the determination of a sequence of domains in MPLS and GMPLS*, IETF RFC6805, 2012.

[10] A. Farrel, O. Komolafe, and S. Yasukawa, *An analysis of scaling issues in MPLS-TE core networks*, IETF RFC5439, 2009.

[11] A. Farrel, J. Vasseur, and A. Ayyangar, *A framework for inter-domain multiprotocol label switching traffic engineering*, IETF RFC4726, 2006.

[12] D. Fay et al., *Weighted spectral distribution for internet topology analysis: Theory and applications*, IEEE/ACM Trans. Netw. **18** (2010), 164–176.

[13] M.R. Gary and D.S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, WH Freeman and Company, New York, 1979.

[14] G. Hernández-Sola et al.,*Scalable hybrid path computation procedure for PCE-based multi-domain WSON networks*, 2011 13th Int. Conf. Transpar. Opt. Netw., IEEE, 2011, pp. 1–4.

[15] D. King et al.,Extensions to path computation element communication protocol (PCEP) for hierarchical path computation elements (PCE), Internet Draft

[16] D. Kreutz et al., *Software-defined networking: A comprehensive survey*, Proc. IEEE, 2015, pp. 14–76.

[17] X. Li, Y. Shi, and Y. Sun, *Rainbow connections of graphs: A survey*, Graphs Combin. **29** (2013), 1–38.

[18] C.W. Lin and B.Y. Wu, *On the minimum routing cost clustered tree problem*, J. Combin. Optim. **33** (2016), 1–16.

[19] J. McCauley et al., *Recursive SDN for carrier networks*, ACM SIGCOMM Comput. Commun. Rev. **46** (2016), 1–7.

[20] F. Paolucci et al., *A survey on the path computation element (PCE) architecture*, IEEE Commun. Surv. Tutor. **15** (2013), 1819–1841.

[21] D. Siracusa et al., *Domain sequence protocol (DSP) for PCE-based multi-domain traffic engineering*, J. Opt. Commun. Netw. **4** (2012), 876–884.

[22] H. Vardhan et al., *Finding a simple path with multiple must-include nodes*, IEEE Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst., 2009. MASCOTS'09, IEEE, 2009, pp. 1–3.

[23] V.V. Vazirani, *Approximation algorithms*, Springer, Berlin, 2013.

[24] R. Zhang and N.N. Bitar, *Backward-recursive PCE-based computation (BRPC) procedure*, IETF RFC5441, 2009.

[25] S. Zhu and G.M. Huang, *A new parallel and distributed shortest path algorithm for hierarchically clustered data networks*, IEEE Trans. Parallel Distrib. Syst. **9** (1998), 841–855.

How to cite this article: Maggi L, Leguay J, Cohen J, Medagliani P. Domain clustering for inter-domain path computation speed-up. *Networks*. 2017;00:1–19. https://doi.org/10.1002/net.21800

APPENDIX

We conclude by providing an ILP formulation for the IDPC-DU problem, where the binary optimization variable $x_{i,j}^k$ equals 1 if the colored edge $(i,j)^k$ is active, otherwise $x_{i,j}^k = 0$. As we explain below, y is an auxiliary variable allowing us to enforce the DU constraint.

$$\min_{\mathbf{x},\mathbf{y}\in\{0,1\}} \quad \sum_{i,j,k:(i,j)^k\in E} w_{i,j}^k x_{i,j}^k \tag{A1}$$

$$\text{s.t.} \quad \sum_{i,k_1:(i,j)^{k_1}\in E} x_{i,j}^{k_1} - \sum_{q,k_2:(j,m)^{k_2}\in E} x_{j,m}^{k_2} = \delta(j), \quad \forall j \in V \tag{A2}$$

$$x_{i,j}^{k_1} + x_{j,m}^{k_2} - y_{i,j,m}^{k_1,k_2} \leq 1 \tag{A3}$$

$$y_{i,j,m}^{k_1,k_2} - x_{i,j}^{k_1} \leq 0 \tag{A4}$$

$$y^{k_1,k_2}_{i,j,m} - x^{k_2}_{j,m} \leq 0 \tag{A5}$$

$$y^{k_1,k_2}_{i,j,m} \leq 1, \quad \forall\, (i,j)^{k_1} \in E^d, (j,m)^{k_2} \in E^q : q \neq d \tag{A6}$$

$$\sum_{\substack{(i,j)^{k_1} \in E^q, \\ (j,m)^{k_2} \in E^d, q \neq d}} y^{k_1,k_2}_{i,j,m} + \sum_{\substack{m,k: \\ (s,m)^k \in E^d}} x^k_{s,m} \leq 1, \quad \forall\, d \in \mathcal{D} \tag{A7}$$

Equation (A2) represent the flow conservation constraints, where $\delta(j) = 1$ if $j = s$, $\delta(j) = -1$ if $j = t$ and $\delta(j) = 0$ otherwise. The variable $y^{k_1,k_2}_{i,j,m}$ expresses the logical AND between the variables $x^{k_1}_{i,j}$ and $x^{k_2}_{j,m}$, that is, $y^{k_1,k_2}_{i,j,m} = 1$ whenever edges $(i,j)^{k_1}$ and $(j,m)^{k_2}$ are visited sequentially. Thus, Equation (A7) prevents the DU constraint from being violated, that is, either there is one active edge leaving source s and belonging to domain d, or there are at most two consecutive activated edges $(i,j)^{k_1} \in E^q$ and $(j,m)^{k_2} \in E^d$ with $q \neq d$, or else none of the two aforementioned alternatives holds. We notice that the number of integer linear constraints in Equations (A2)–(A7) is $O(|V\|\mathcal{D}|^2)$.

An alternative possibility to solve the ILP above is to first compute the transitive closure of the graph $\mathcal{G}$, which assigns to each pair of nodes i,j and domain d a weight $w^d_{i,j}$ equal to the minimum cost of paths from i to j of color d. Then, the solution of IDPC-DU turns into the computation a min cost rainbow path, and the corresponding ILP becomes:

$$\begin{aligned} \min_{\mathbf{x} \in \{0,1\}} \quad & \sum_{i,j,k:(i,j)^k \in E} w^k_{i,j} x^k_{i,j} \\ \text{s.t.} \quad & \sum_{i,k_1:(i,j)^{k_1} \in E} x^{k_1}_{i,j} - \sum_{q,k_2:(j,m)^{k_2} \in E} x^{k_2}_{j,m} = \delta(j), \quad \forall\, j \in V \\ & \sum_{(i,j)^k \in E^d} x^k_{i,j} \leq 1, \quad \forall\, d \in \mathcal{D} \end{aligned} \tag{A8}$$

where (A8) ensures that at most one edge of color d is visited, for all $d \in \mathcal{D}$.