On iBGP Routing Policies

Stefano Vissicchio, Luca Cittadini, Giuseppe Di Battista

***Abstract*—Internet Service Providers (ISPs) run the internal Border Gateway Protocol (iBGP) to distribute inter-domain routing information among their BGP routers. Previous research consistently assumed that iBGP is always configured as a mere dispatcher of inter-domain routes. However, router configuration languages offer operators the flexibility of fine-tuning iBGP.**

In this paper, we study the impact of deploying routing policies in iBGP. First, we devise a provably correct inference technique to pinpoint iBGP policies from public BGP data. We show that the majority of large transit providers and many small transit providers do apply policies in iBGP. Then, we discuss how iBGP policies can help achieve traffic engineering and routing objectives. We prove that, unfortunately, the presence of iBGP policies exacerbates the iBGP convergence problem and invalidates fundamental assumptions for previous results, affecting their applicability. Hence, we propose provably correct configuration guidelines to achieve traffic engineering goals with iBGP policies, without sacrificing BGP convergence guarantees. Finally, for the cases in which our guidelines are not applicable, we propose a novel technique to verify the correctness of an iBGP configuration with iBGP policies. We implement a prototype tool and show the feasibility of off-line analyses of arbitrary policies on both real-world and in-vitro configurations.

***Index Terms*—IP networks, routing protocols, computer network management**

I. Introduction

The Internet is an interconnection of different domains, or *Autonomous Systems* (ASes), where each AS is administered by an *Internet Service Provider* (ISP). Routing among ASes is handled by the Border Gateway Protocol (BGP) [1]. External BGP (eBGP) is used to exchange routes between BGP routers in different autonomous systems (ASes), and Internal BGP (iBGP) is used to distribute routes learned from external ASes to all the BGP routers (peers) of the same autonomous system.

BGP owes its popularity to two important features: *(i)* eBGP supports expressive routing policies via router configuration languages, and *(ii)* iBGP can achieve good scalability by employing route reflection [2]. Unfortunately, these features can introduce routing and forwarding anomalies [3], [4], [5], [6], among which violations of basic correctness properties like protocol convergence.

Configuration languages also allow operators to configure policies in iBGP, by instructing routers to change attributes in BGP messages. Both theoretical [4] and practical [7], [8] previous contributions assumed that BGP messages are exclusively manipulated at the border of ISPs' networks, to express eBGP policies. The possibility to change iBGP attributes en route is taken into account only in [9]. However, the techniques presented in that work i) are still preliminary, i.e., their practical applicability has still to be proved; and ii) do not straightforwardly apply to generic IGP configurations.

In this paper, we address the following questions.

(i) Is the assumption about the absence of iBGP policies reasonable? Do ISPs actually deploy iBGP policies?

(ii) What are the pros and cons of deploying iBGP policies? Why should an ISP (not) configure its routers to modify iBGP messages en route?

(iii) How do iBGP policies relate to iBGP convergence?

(iv) Can an operator profitably apply some iBGP policies while ensuring routing convergence?

While investigating answers to these questions, we develop multiple contributions.

First, we describe an inference technique that, using public BGP data, identifies ISPs deploying iBGP policies. Despite the limitations of the dataset, we found evidence that iBGP polices are commonly deployed by the majority of large transit providers and by many small transit providers. These findings constitute a strong motivation for the rest of our work. Indeed, transit providers have been the main target of previous research work in BGP, because of the complexity of their configuration and of their routing requirements.

Second, we discuss possible advantages of configuring iBGP policies. Namely, we exemplify simple yet realistic use cases in which deploying iBGP policies is the easiest solution to achieve traffic engineering goals or to realize common business requirements of transit providers.

Third, we describe drawbacks and caveats related to iBGP policies. In particular, we focus on routing convergence guarantees, as reaching a stable state is a precondition for all the other types of BGP correctness, namely, forwarding [4] and dissemination [6]. In particular, we prove that deploying iBGP policies makes iBGP prone to convergence anomalies in otherwise correct configurations, and invalidates previously known sufficient conditions for iBGP convergence.

Fourth, we propose guidelines to take advantage of iBGP policies while ensuring routing correctness by design. Our configuration guidelines are easy to deploy, enforce realistic business objectives, and guarantee iBGP convergence independently of the eBGP announcements received and of the presence of faults.

Finally, for iBGP policies that do not fit our guidelines, e.g., because of more complex routing needs, we present a technique and a tool to statically check iBGP configurations for runtime convergence, even in the presence of iBGP policies. The tool provably avoids false positives, i.e., it never

Stefano Vissicchio is postdoctoral researcher of the Belgian Fund for Scientific Research (F.R.S.-FNRS), affiliated with ICTEAM, Université catholique de Louvain, Louvain-la-Neuve, Belgium (e-mail: stefano.vissicchio@uclouvain.be).

Luca Cittadini and Giuseppe Di Battista are with Computer Science and Automation Department, Roma Tre University, Rome, Italy (email:{ratm,gdb}@dia.uniroma3.it)

The authors thank Randy Bush, Olaf Maennel, Debbie Perouli and Olivier Bonaventure for interesting discussions on the topic and insightful comments on preliminary versions of this paper.

misreports a problematic configuration as correct. We evaluate the tool through in-depth experimentation on both in-vitro and real-world iBGP configurations. In particular, we analyze scalability properties and sensitivity to a variety of factors. The evaluation shows that our technique is suitable for off-line configuration checks and what-if analyses.

A preliminary version of this work appeared in [10]. This paper presents several new results, including: *(i)* extension of the inference technique to track the granularity of the inferred iBGP policies; *(ii)* historical analysis of ISPs deploying iBGP policies; *(iii)* discussion of additional use cases in which iBGP policies are useful; *(iv)* illustration of configurations in which known sufficient conditions [4] are invalidated by the presence of iBGP policies; *(v)* a revision and a generalization of the guidelines proposed in [10], with updated formal proofs; *(vi)* improved description of the implementation of the iBGP policy checker and of the optimizations performed by it, along with a proof of their correctness (see the Appendix); *(vii)* a complete evaluation of the policy checker, with scalability and sensitivity analyses.

The rest of the paper is organized as follows. Section II provides some background. Section III describes our inference technique and shows that Internet transit providers do deploy iBGP policies. Section IV discusses cases in which an ISP can take advantage from convenient iBGP policies. Section V shows the increased instability risks introduced by iBGP policies. Section VI presents our configuration guidelines. Section VII describes and evaluates our configuration checker. Section VIII contains the conclusions.

II. Background

BGP enables routers to distribute routing information throughout the Internet. In this paper, we take the perspective of a single AS. For this reason, we focus on iBGP which is used to distribute the information learned via eBGP to all the BGP routers of the same AS.

BGP routers exchange *routes* to inter-domain destinations using *BGP messages*. Each BGP message contains routes to one or more IP prefixes, and the association of each route to a set of *attributes*. When a BGP router receives a BGP message, it *(i)* possibly discards the BGP message or modifies some attributes according to *input filters*; *(ii)* selects, among all the routes that it received, its *best route* to that prefix; and *(iii)* sends its best route to its *BGP neighbors*, possibly after having edited some of its attributes according to its *output filters*. The best route is selected by running the deterministic *BGP decision process* summarized in Table I. Essentially, a route to a destination prefix is selected as best based on the values of the associated attributes. Different prefixes are treated separately. We refer the reader to [2] for a more detailed description of the BGP decision process.

The original design of iBGP mandated a full mesh of iBGP sessions within an AS in which each router distributes only routes learned through eBGP. However, the scaling issues of this solution spurred the proposal of alternatives. The most widespread alternative to iBGP full mesh is *route reflection* [2]. In route reflection, the iBGP neighbors of each router are split into three sets: *clients*, *peers* and *route reflectors*. In the following, we refer to both peers and route reflectors as *non-clients*. A router that has one or more clients acts as a *route reflector*, and relays routing information to its clients. In the following, we refer to the organization of iBGP sessions as iBGP topology. Intuitively, a route reflection topology consists in a hierarchy of clients and route reflectors. An iBGP full mesh can be seen as a one-layer route reflection hierarchy, where all iBGP routers act as iBGP peers. Moreover, we refer to iBGP routers that have an eBGP path to a given prefix p as *egress points* for p.

Step	Criterion
1	Prefer routes with the highest `local-preference`
2	Prefer routes with the lowest `as-path` length
3	Prefer routes with the lowest `origin`
4	Among the routes received from the same eBGP neighbor, prefer those having the lowest `MED`
5	Prefer routes learned via eBGP to those learned via iBGP
6	Prefer routes with the lowest IGP metric to the egress point
7	Prefer routes with the lowest `egress-id`
8	Prefer routes with the shortest iBGP path
9	Prefer the route from the neighbor with the lowest IP address

TABLE I: Steps in the BGP decision process [2].

Each iBGP router propagates its best route according to the following route reflection *propagation rules*. If the best route is learned from a non-client iBGP neighbor, then it is relayed only to iBGP clients, otherwise it is propagated to all iBGP neighbors. Because of these rules, not every best route is propagated by any iBGP router to any other. We define a *valid signaling path* as any sequence of iBGP routers through which a route can be disseminated according to iBGP route propagation rules. In order to ensure that routes are distributed to every router in the AS, the following conditions must hold [6]: *(i)* Each iBGP router can be univocally assigned to a level in the route reflection hierarchy, namely the level above any of its clients and below any of its route reflectors; *(ii)* iBGP sessions must not cross more than one layer; and *(iii)* all the routers in the top layer must be iBGP peers. Since invalidating those conditions leads to correctness problems even in the absence of iBGP policies [6], throughout the paper we assume that those conditions hold.

As an illustration of how iBGP works, consider the topology depicted in Figure 1, where the links represent iBGP sessions between routers. In the figure, six iBGP routers are organized in a two-layer route reflection hierarchy, with $RR1$, $RR2$, and $RR3$ acting as route reflectors of $BR1$, $BR2$, and $BR3$. For correct route dissemination, all the route reflectors are iBGP peers. $BR1$ and $BR3$ are egress points for prefix p, and receive respectively the eBGP routes $\mathcal{R}_1$ and $\mathcal{R}_3$ to p. Assume that the standard BGP decision process is used by all the iBGP routers in the figure, i.e., no input or output filters are configured. Initially, $BR1$ and $BR3$ select their respective eBGP routes as best, since they do not have other routes to p. After selecting their best routes, $BR1$ and $BR3$ respectively propagate $\mathcal{R}_1$ and $\mathcal{R}_3$ to their iBGP neighbors. Thus, $RR1$ learns $\mathcal{R}_1$, $RR3$ learns $\mathcal{R}_3$, and $RR2$ learns both $\mathcal{R}_1$ and $\mathcal{R}_3$. Since $RR1$ and $RR3$ have just one route available, they select that route as best. On the contrary, $RR2$ has to run

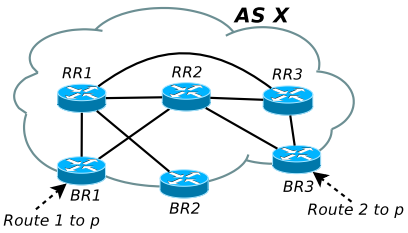


Fig. 1: A simple iBGP network.

```
ROUTE-SERVER.PHX1>SH IP BGP 189.90.12.0/24
BGP ROUTING TABLE ENTRY FOR 189.90.12.0/24
PATHS: (4 AVAILABLE, BEST #1)
  NOT ADVERTISED TO ANY PEER
13878 15180 28189
    67.17.64.89 FROM 67.17.80.210 (67.17.80.210)
       ORIGIN IGP, METRIC 0, LOCALPREF 300, BEST
       COMMUNITY: 3549:4471 3549:30840
       ORIGINATOR: 67.17.81.221,
       CLUSTER LIST: 0.0.0.92
28189 28189 28189 28189 28189 28189 28189
    67.17.64.89 FROM 67.17.82.40 (67.17.82.40)
       ORIGIN IGP, METRIC 0, LOCALPREF 300
       COMMUNITY: 3549:4950 3549:34076
       ORIGINATOR: 200.186.0.67,
       CLUSTER LIST: 0.0.2.109, 0.0.5.2
  ...
```

Entry #1

Entry #2

Fig. 2: Two routes with different `as-path` lengths simultaneously active in AS 3549.

the BGP decision process to select the best route between $\mathcal{R}_1$ and $\mathcal{R}_3$. Suppose that $RR2$ selects $\mathcal{R}_1$, e.g., because it has a shorter `as-path` (see Step 2 of Table I). Now, all three route reflectors have a best route which was received from an iBGP client. By the iBGP propagation rules, they propagate their best routes to all their neighbors. As soon as $RR3$, $BR2$ and $BR3$ learn $\mathcal{R}_1$, they run the BGP decision process and select $\mathcal{R}_1$ as best, because of its shorter `as-path`. By contrast with route reflectors, neither $BR2$ nor $BR3$ propagate their new best route further, because the best route was received from a route reflector, and they have no clients. For the same reason, $RR3$ propagates its new best route exclusively to $BR3$. Note that $BR3$ receives multiple copies of $\mathcal{R}_1$, one from $RR2$ and another from $RR3$. Again, $BR3$ compares those two copies using the BGP decision process, and chooses the one announced by $RR2$ because of the shortest iBGP path ($BR1$ $RR2$) (see Step 8 of Table I). After this step, no further messages are exchanged and all iBGP routers will steadily select their best route to p. In this case, we say that iBGP converged to a stable routing state.

In the following, we investigate the pros and cons of altering the BGP decision process by configuring input and output filters on iBGP routers. The ability to configure input and output filters provides operators with a high degree of flexibility. Indeed, BGP configuration languages allow almost arbitrary modifications to the attributes carried by BGP messages. Operators typically take advantage of this flexibility by configuring sets of conditional rules, or routing *policies*. Those rules are targeted to either influence the selection of the best route, or to control the propagation of specific routes to specific eBGP neighbors. As the router configuration language is the same for iBGP and eBGP, nothing prevents policies from being deployed in iBGP. We consider any modification of an attribute in an iBGP message (except the ones mandated by the protocol itself) as the effect of an iBGP policy. Note that iBGP attributes can be overwritten so that arbitrary steps of the BGP decision process are skipped. For example, by unconditionally modifying the `as-path` attribute of all messages so that they have the same length, an iBGP policy can force a router to disregard `as-path` length during the BGP decision process. For this reason, we treat any configuration that causes a router to skip a decision step (see, e.g., Cisco `bgp bestpath as-path ignore` command) as an iBGP policy.

III. iBGP Policies in the Internet

The vast majority of previous research assumes that iBGP is configured as a simple dispatcher of inter-domain routes inside single ISPs, while routing policies are exclusively applied to eBGP sessions. In this section, we show that this assumption is not always confirmed in reality, and ISPs do deploy policies in iBGP. We first discuss a query to a route server, whose output cannot be explained assuming the standard BGP decision process. Then, we describe an inference technique to estimate the popularity of iBGP policies. Finally, we report results of the application of our technique to public BGP data. Those results suggest a consistent (and growing) trend to deploy iBGP policies over the last years.

A. An Interesting Query

Figure 2 shows the output of a query that we performed on a publicly available route server inside the Global Crossing network (AS 3549) on 31st August 2009, at 14:36 UTC. In particular, the figure reports two BGP routes available at the queried route server for the destination prefix 189.90.12.0/24. Each BGP route, framed in a colored box in the figure, is associated with its respective attributes. The first line of each entry represents the `as-path` attribute. Other attributes e.g., `local-preference` and `origin`, follow. The presence of iBGP-only attributes like `cluster-list` (that represents the iBGP path) implies that the routes were propagated to the route server from its iBGP neighbors.

In particular, the figure shows two routes having the same `local-preference` value but different `as-path` lengths (see the highlighted text in Figure 2). The two routes are not equally good according to the first three steps of the BGP decision process (see Table I). This would not have been possible if the standard BGP decision process had been deployed. In that case, as noted in [11], only routes that are equally good up through the first three BGP decision steps can be selected as best by iBGP routers in the steady state. Hence, the route server would have received only routes with the same `as-path` length from its neighbors.

We have three possible explanations for this feature.

- the iBGP topology of the ISP was not connected at that time. However, we consider this case highly unlikely because i) we observed the presence of the routes in multiple time frames; and ii) a disconnected iBGP topology would sharply contrast with the objective of distributing inter-domain routes inside an ISP for which iBGP is used.

- the ISP deployed advanced BGP features, like BGP diverse paths [12], add-paths [13], or best-external [14]. Those features let routers propagate additional routes with respect to the one they respectively select, with the goal of increasing the route diversity.
- iBGP policies are deployed by the ISP, i.e., on the iBGP neighbors of the queried route server. This would allow some of the route server neighbors to select a route with longer `as-path` length, e.g., because of convenient modification of `local-preference` values.

In the following, we always exclude the first possibility, as we assume iBGP to be reasonably configured by ISPs.

B. An Inference Technique

The query described in the previous section raises the suspicion that ISPs deploy iBGP policies. Network operators are typically reluctant to reveal their policies, hence any attempt to quantify iBGP policies adoption in the Internet has to rely on indirect analyses rather than on the configurations themselves. We now describe an inference technique to compute a lower bound of the ISPs deploying iBGP policies in the Internet. Our technique is provably correct, i.e., it never misreports an ISP as deploying iBGP policies when it does not.

The technique is based on data stored by geographically distributed vantage points. A vantage point is a BGP route collector which has an eBGP session with at least one router inside a given ISP. To infer the usage of iBGP policies, we keep track of the routes in each AS, as reported by multiple vantage points. We then search for routes with different `as-path` lengths simultaneously active in the same AS.

The fact that vantage points collect data over eBGP sessions ensures the correctness of the inference. In fact, eBGP can only propagate the best route, hence each route collected by a vantage point is guaranteed to be **selected** by at least one BGP router in the neighboring AS. This holds even in the presence of iBGP variants that allow advertisement of multiple routes per prefix (e.g., [13], [14], [12]), meaning that the presence of routes with different `as-path` lengths can only be due to iBGP policies.

In the following, we report the results of the application of our technique to public vantage points, namely those used in the RIPE RIS project [15]. In [10], we estimated the number of ASes deploying iBGP policies on a single dataset. In this paper, we apply the same technique to several datasets to provide a historical analysis. Moreover, we extend the original technique to report intra-continent policies. Namely, we correlated the iBGP policy inference with the geographical position of the BGP collectors used in the inference. To this end, we extracted the geographical position of each monitor by relying on the information available on the Web site of the RIPE RIS project. Then, during the computation of the ISPs deploying iBGP policies, we kept track of the BGP collectors simultaneously receiving BGP announcements with different `as-path` lengths. Finally, we reported an AS X to apply intra-continent policies if at least two BGP routes with different `as-path` length were reported by at least two BGP monitors inside the same continent.

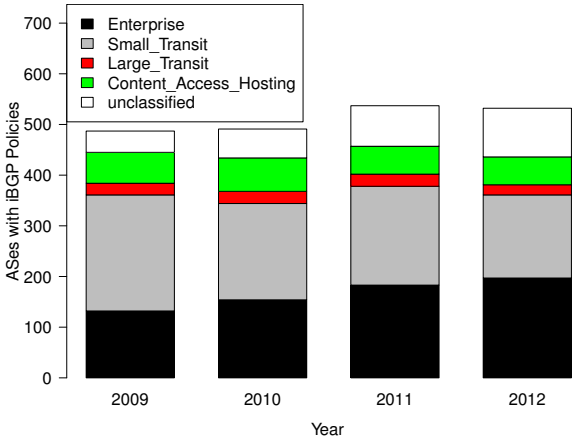


Fig. 3: Historical analysis of ASes deploying iBGP policies as detected by our inference technique.

C. Transit Providers Do Deploy iBGP Policies

In [10] we have shown that over $1,800$ ISPs (about $5,7\%$ of the ASes at that time) deployed iBGP policies already in 2009. We now study how such an estimate has evolved over the last four years. To this end, we applied the inference technique to two weeks (from September 16 to September 30) in 2009, 2010, 2011, and 2012. As we aim at understanding historical trends, we considered a smaller dataset with respect to the one used in [10]. In particular, i) we only relied on routing tables while both tables and updates were analyzed in [10]; ii) we only considered RIPE RIS collectors [15], in order to avoid synchronization problems with RouteViews collectors [16]; and iii) we restricted to the subset of collectors which were active in all the considered years, in order to avoid biases due to different sets of collectors over time. Note that the ability of our technique to detect iBGP policies increases with the number of collectors. This explains why this analysis shows smaller absolute values with respect to [10].

Figure 3 plots the absolute number of ISPs reported by our analyses. Even using our limited dataset, we found a non-negligible number, around 500, of ISPs deploying iBGP policies in the considered years. Moreover, the detected ISPs are significantly consistent over the years, e.g., an ISP that applies iBGP policies in 2009 is likely to be detected also in subsequent years. In particular, for almost half of the detected ISPs, we were able to infer that they applied iBGP policies for more than one year (typically, multiple consecutive years). Moreover, we found 137 ASes which deploy iBGP policies in all the considered years. This suggests that ISPs do not change their attitude to iBGP policies very often.

To understand which type of ISPs are more likely to deploy iBGP policies, we classified the detected ISPs according to their role in the Internet. In particular, we used the classification in [17], which distinguishes between Enterprise Customers (EC), Small Transit Providers (STP), Large Transit Providers (LTP) and Content, Access and Hosting Providers (CAHP). We found that the division of detected ISPs in classes does not substantially change across the years. More importantly, we found evidence that the majority of large

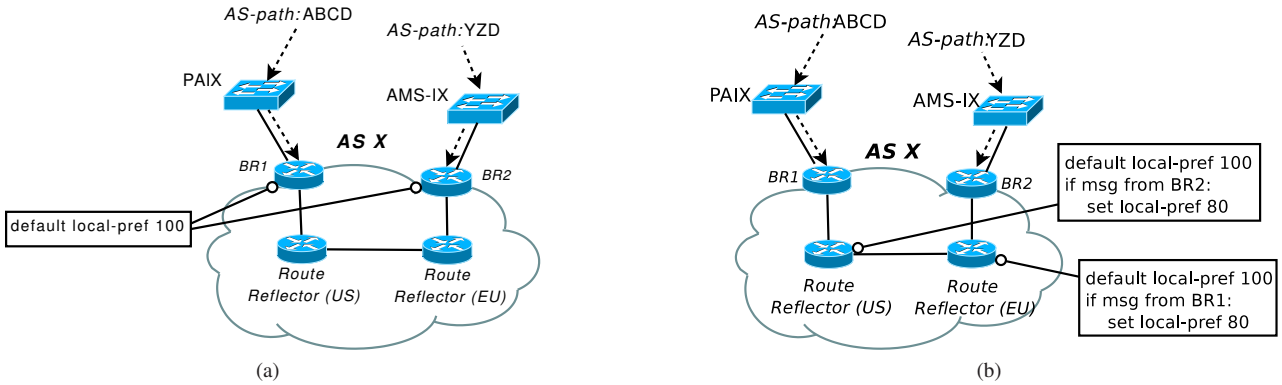


Fig. 4: (a) Default BGP configuration causes sub-optimal traffic forwarding in AS X: outbound traffic is routed through AMS-IX, due to the length of the `as-path` attribute. (b) By changing iBGP attributes, AS X is able to exploit both AMS-IX and PAIX as traffic egress points, achieving better load balancing.

transit providers and a significant percentage of small transit providers apply iBGP policies. Indeed, roughly 30% of the ISPs we detected are STPs and about 5% are LTPs throughout the four-year period. Moreover, in every year we considered, $65\% - 80\%$ of large transit providers and $5.5 - 7\%$ of small transit providers have been detected to apply iBGP policies. The relative amount of detected ISPs belonging to the remaining two categories is sensibly smaller.

Further, we detected 18 ISPs applying iBGP policies on an intra-continent level. Observe that, since the vast majority of BGP collectors in our selection are in Europe, we were only able to detect ISPs applying iBGP policies within Europe. Again, we found a disproportionate presence of transit providers, given that large transit providers only account for a tiny fraction of the ASes in the Internet. Moreover, our results testify consistency of intra-continent policy application over the years for several ISPs.

IV. iBGP Policies ⇒ More Flexibility

The analysis in Section III shows that iBGP policies are actually adopted in the Internet. In this section, we discuss possible motivations for ISPs to deploy iBGP policies.

Figure 4a provides a simple example where AS X spans over North America and Europe, and participates to Internet eXchange Points (IXPs) in Palo Alto (PAIX) and Amsterdam (AMS-IX). Configuration snippets represented in the figure are expressed in an intuitive vendor-independent pseudo-language and are trivial to translate to any vendor-specific configuration. In the example, AS X is assumed to employ a two layer route reflector hierarchy in order to scale its iBGP configuration. Among others, AS X has one route reflector in the US and another in Europe. In order to have correct route dissemination (see Section II), route reflectors are all iBGP peers. Being a large ISP, AS X is likely to exhibit high route diversity [18], that is, it is likely that multiple border routers of AS X learn different eBGP routes for the same destination prefix. Suppose that X receives two BGP routes for prefix p: *(i)* a BGP route advertising path $ABCD$ from an eBGP peer at PAIX, and *(ii)* another BGP route advertising path YZD from an eBGP peer at AMS-IX.

Assuming that X assigns `local-preference` values according to business relationships [19], [20], the received routes are assigned the same value since they both come from an eBGP peer. For this reason, the two routes are equally good according to the first step of the BGP decision process. The next step of the BGP decision process evaluates the length of the `as-path`. Since the path received at AMS-IX is shorter than the path received at PAIX, every BGP router will prefer the former. This implies that all the traffic directed to p will be forwarded to Amsterdam.

However, since AS X does not get any revenue from traffic transiting over IXPs, its best strategy would be to minimize the cost of traffic forwarding. In the configuration just described, routers in the US forward traffic to Europe, using expensive transoceanic links, instead of sending it via Palo Alto. Hence, the high-level business objective of minimizing forwarding costs is not met by the BGP configuration in Figure 4a. Indeed, this objective would be better accomplished if X was able to send traffic from US via Palo Alto and from Europe via Amsterdam, reducing the usage of transoceanic cables connecting US and Europe. Ensuring that traffic is routed via multiple egress points in diverse continents is even more useful if p is an anycast prefix (e.g., the K-root DNS server prefix). In this case, traffic segmentation can help improve network performance (e.g., latency).

Despite its simplicity, such a requirement cannot be implemented within the standard BGP decision process. One possible solution for AS X could be to split its network into multiple AS domains connected via eBGP and deploying eBGP policies between US and Europe. However, this is a rather invasive solution. An effective and simpler alternative is to deploy iBGP policies as depicted in Figure 4b. In this solution the route reflector in US is configured to prefer US routes, and the route reflector in Europe to prefer European routes by conditionally changing the value of the `local-preference` attribute (e.g., via `route-maps`). In addition to minimizing forwarding costs, this iBGP policy also ensures that the traffic balancing policy is honored regardless of what `as-paths` are announced by X's neighbors. That is, X's peers are unable to disrupt X's internal routing policies by tweaking their eBGP announcements.

In order to give a rough estimate of the extent to which external announcements can influence routing choices, we analyzed the BGP updates received from the border routers of a medium-sized Italian ISP in 2009. We found out that almost half of the Internet routing table of the ISP at that time was load-balanced across different exit points just because of equal `as-path` lengths. Should the `as-path` length vary on one of the available routes, the load balancing policy would be immediately compromised. When we talked to the ISP operators, they were surprised to know that at least 20% of their traffic was actually balanced due to eBGP routes having the same `as-path` length, which was purely accidental.

To better understand the risk of balancing traffic based on `as-path` lengths, consider again the example in Figure 4a. Suppose now that the European peer of AS X starts advertising an `as-path` of length 5 or more, e.g., because the AS that originates the prefix is performing inbound traffic engineering activities via `as-path` prepending. As soon as this new route is propagated within AS X, all the routers in X change their routing decision and start to forward traffic destined to prefix p via Palo Alto instead of Amsterdam. Transient traffic and service disruptions are very likely during the routing convergence process.

Another use case for deploying iBGP policies is the additional flexibility in the definition of internal routing policies. For example, it is known [21] that large transit providers sometimes configure local eBGP session, i.e., to provide connectivity to a customer only in a specific region or country (e.g., only between sources and destinations in Germany). This requirement translates to the need for fine-grained control of which routes are distributed to which eBGP neighbors, and can be easily fulfilled by applying filtering policies in iBGP. Again, splitting the network in multiple ASes would also implement such connectivity requirements, but it does not scale well with the number of requirements and is not easily adaptable when the requirements change.

Observe that the use cases discussed above provide an intuitive explanation of the popularity of iBGP policies among transit providers (Section III). In fact, transit providers have more complex traffic engineering and routing requirements with respect to their customers, hence they are more likely to need iBGP policies. By private conversations with transit provider operators, we were able to confirm both the deployment of iBGP policies (as inferred by our technique), and the plausibility of the above use cases.

V. More Flexibility ⇒ More Instability

After having shown the possible benefits of iBGP policies, we study their downsides and caveats.

A possible argument to leave iBGP attributes untouched is that iBGP policies have the potential to complicate the configuration, and make it hard to understand and debug. Besides the intrinsic disadvantages of a complex BGP configuration, we now show that deploying iBGP policies potentially exacerbates iBGP convergence problems.

It is well known that, in general, conflicting routing policies can prevent BGP to converge to a stable state [22]. Moreover, it has been already shown [4] that iBGP might be unable to converge when no iBGP policies are deployed. In this section, we answer two further questions. *(i)* Can iBGP policies be responsible for convergence issues that cannot occur otherwise? *(ii)* Do the known sufficient conditions for iBGP correctness hold in the presence of iBGP policies?

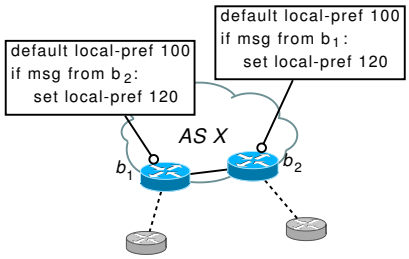


Fig. 5: A case in which iBGP policies are responsible for iBGP convergence problems that cannot be created otherwise.

A. iBGP Policies Introduce More Instability Risks

We now show that deploying iBGP policies can lead to policy conflicts that cannot exist otherwise. Indeed, the following theorem holds.

Theorem 1. *There exists at least one iBGP network such that BGP is guaranteed to converge if and only if no iBGP policies are deployed.*

Proof: Consider the iBGP network in Figure 5. We now show that this network is prone to convergence anomalies if and only if iBGP policies are deployed.

The iBGP topology in Figure 5 can be unstable in the presence of iBGP policies. Figure 5 depicts an iBGP policy setting in which iBGP is not guaranteed to converge. Indeed, with the `local-preference` settings represented in the figure, b_1 prefers the route propagated by b_2 and b_2 prefers the route propagated by b_1. This circular preference of routes between two BGP routers constitutes a DISAGREE gadget, which is known to be prone to routing instability [3].

The iBGP topology in Figure 5 is stable in the absence of iBGP policies. Assume that no iBGP policies are configured in the network. Let P_i be the best eBGP route received by b_i. We now walk through the BGP decision process at routers b_1 and b_2, examining all the possible cases.

- If P_1 and P_2 have different `local-preference` values, then the P_i with the highest value is eventually selected by both b_1 and b_2.
- Otherwise, if P_1 and P_2 have different `as-path` lengths, the route with the shortest `as-path` length is eventually selected by both b_1 and b_2.
- If there is a tie in the previous decision steps and P_1 and P_2 have different `origin` values, the route with the lowest `origin` is eventually selected by both routers.
- In case of a tie in all the previous steps, Step 5 of the BGP decision process forces each router b_i to eventually select P_i, with $i \in \{1, 2\}$.

In any case, both b_1 and b_2 steadily select a single eBGP route, and iBGP converges to a stable state. ■

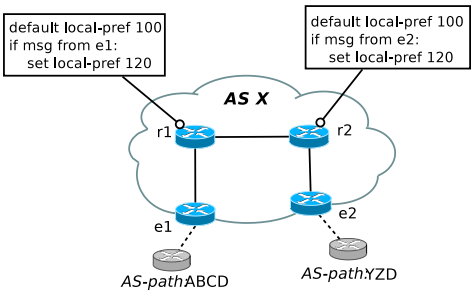


Fig. 6: A case in which iBGP policy invalidates known sufficient conditions for correctness. In this example, iBGP is not guaranteed to converge even if the *prefer-client* condition [4] is enforced by iBGP policies.

B. iBGP Policies Invalidate Previous Sufficient Conditions

The example in Figure 5 can be seen as an inconsistent iBGP policy setting, e.g., due to a configuration error. Unfortunately, iBGP may be prevented from converging even if meaningful policies are consistently configured network-wide. Even worse, known sufficient conditions for iBGP convergence may not be guarantee routing stability when iBGP policies are deployed. In other words, the presence of iBGP policies affects the validity of those sufficient conditions.

Consider, for example, the configuration in Figure 6, in which r_1 and r_2 are iBGP peers and act as route reflectors of e_1 and e_2 respectively. Both e_1 and e_2 are egress points for a given prefix originated at AS D. The configured iBGP policy dictates that each route reflector prefers routes from clients over routes from non-clients. Such a condition is known as the *prefer-client* condition and has been shown [4] to be sufficient for iBGP convergence in the absence of iBGP policies. We now show that routing convergence is not guaranteed in the presence of iBGP policies. Assume that eBGP paths of different `as-path` lengths are received at e_1 and e_2, as in the figure. The egress point e_1 prefers the route injected in iBGP by e_2, because it has shorter `as-path` than the eBGP route that e_1 itself receives. However, r_1 prefers the eBGP route received from e_1 because of the configured iBGP policy. Hence, a DISAGREE [3] exists between e_1 and r_1 because of their conflicting path preferences. It is easy to construct similar examples with different unstable structures (e.g., the permanently unstable BAD-GADGET [3]).

Note that, although implementing the same high-level routing policy as in Figure 6, the configuration in Figure 4b is not subject to the same problem. The key difference is that, since the local-preference value set by a route reflector is honored by its clients, $BR1$ does not prefer the path from $BR2$ to its eBGP route in the previous example.

VI. Profitable iBGP Policy Configuration by Design

Sections IV and V suggest that an ISP willing to deploy iBGP policies essentially faces a trade-off between flexibility and routing convergence guarantees. We now define configuration guidelines that safely exploit the flexibility of modifying iBGP attributes. The guidelines proposed in this paper generalize the ones proposed in [10] in two aspects. First, they do not rely on the svn assumption that AS relationships can be classified in customer-provider and peer-to-peer [19]. Second, they provably guarantee routing stability even when the received eBGP routes have different `as-path` lengths.

We assume that a given ISP ranks routes according to a partial order of preference over its neighboring ASes. This partial order imposes a hierarchy of N classes, such that routes learned from neighbors in the i-th class are preferred over routes learned from neighbors in the $i+1$-th class. Such a hierarchy models business relationships, i.e., routes from ASes in the i-th class are economically more convenient than routes from ASes in the $i+1$-th class. The division of eBGP neighbors into customers, eBGP peers, and providers described in [19] is a well-known example of a policy that imposes a hierarchy on three classes. However, our guidelines apply to a more general case, that captures some of the more complex business relationships between ISPs known to hold in practice [21].

To ensure iBGP convergence in the presence of iBGP policies, our guidelines enforce a stronger variant of the *prefer-client* condition formulated in [4]. Note that they solely rely on the presence of a partial order of preference over eBGP neighbors, e.g., they do not impose any constraint on the underlying IGP topology.

Guideline A (Local-preference reflects business relationships). *Every iBGP router assigns the same local-preference value LP_j to any eBGP route learned from a neighboring AS in the j-th class, so that $LP_{j-1} > LP_j > LP_{j+1}$.*

Guideline B (Higher local-preference to eBGP routes). *Among routes learned from the same class of neighbors, each iBGP router assigns higher local-preference to eBGP routes than to iBGP routes.*

Guideline C (Higher local-preference to iBGP client routes). *Among iBGP routes learned from the same class of neighbors, each iBGP router assigns higher local-preference to routes propagated by any iBGP client over those propagated by any iBGP non-client.*

Our guidelines ensure the following desirable properties, that enable the definition of profitable iBGP policies.

First of all, they guarantee *iBGP convergence* to a predictable state, as proved by Theorem 4 in the Appendix.

Second, iBGP routers *select routes according to revenues and costs*, i.e., according to hierarchy of eBGP neighbors. In particular, each iBGP router steadily selects a route propagated by an AS in class i if and only if no route is propagated by any AS in a class $j < i$ (see Theorem 3 in the Appendix).

Third, our guidelines provably minimize internal transit cost, i.e., the cost of forwarding traffic within the ISP network. Indeed, whenever possible, egress points will steadily select eBGP routes, i.e., sending traffic directly outside the ISP, and route reflectors will choose routes learned from an iBGP client (see Theorem 2 in the Appendix). Assuming that the iBGP topology is congruent with the IGP topology as suggested by current best practices [23], this leads to internal cost minimization.

Configuration Steps

(i) Tag routes

```
if msg from eBGP customer
        add community comm_cust
if msg from eBGP peer
        add community comm_peer
if msg from eBGP provider
        add community comm_prov
del community comm_ibgp
if msg from iBGP neighbor
        add community comm_ibgp
del community comm_client
if msg from iBGP client
        add community comm_client
```

(ii) Tweak route preferences

```
if comm_cust in communities
        set local-pref 200
if comm_cust and comm_client in communities
        set local-pref 180
if comm_cust and comm_ibgp in communities
        set local-pref 160
if comm_peer in communities
        set local-pref 100
if comm_peer and comm_client in communities
        set local-pref 90
if comm_peer and comm_ibgp in communities
        set local-pref 80
if comm_prov in communities
        set local-pref 50
if comm_prov and comm_client in communities
        set local-pref 40
if comm_cust and comm_ibgp in communities
        set local-pref 30
```

Fig. 7: A simple configuration complying with Guidelines A, B and C.

Fourth, configuration complexity can be kept manageable while enforcing our guidelines. As an illustration, Figure 7 shows an implementation of our guidelines for an eBGP hierarchy with 3 levels [19]. In the implementation, the `community` attribute is used to tag routes, and the `local-preference` attribute is modified accordingly. We argue that this does not add excessive management complexity, since very similar techniques are commonly used by ISPs to manage traffic from neighboring ASes [20].

Finally, routing policies imposed by our guidelines *cannot be influenced by attributes in the received eBGP messages*. Indeed, the guidelines act on the `local-preference` attribute, which is evaluated at the first step of the BGP decision process (see Table I). Hence, the iBGP policy takes the highest precedence and the selected routes are guaranteed to be compliant with the policy independently of the value of other BGP attributes. In particular, attributes like `as-path` and `origin`, which can be manipulated by external ASes for their own traffic engineering purposes, do not affect the realization of the given iBGP policy. As a side effect, the forwarding plane is no longer affected by changes to the `as-path` or `origin` attribute, which makes BGP-induced traffic shifts across the network much less likely to occur. Moreover, our guidelines are *robust to IGP and iBGP topological changes*, e.g., due to network failures. Indeed, all the theorems proving the properties of our guidelines make no assumptions on the iBGP topology nor on the combination of egress points injecting eBGP routes in iBGP (see the Appendix).

VII. A Checker for iBGP Policy Configurations

In the previous section, we provided guidelines that allow operators to configure safe and profitable iBGP policies if an arbitrary cost-revenue model, including the classification of eBGP neighbors into a hierarchy, can be applied. However, some ISPs may need to configure more complex iBGP policies that do not comply with our guidelines. In this section, we provide a practical technique to deal with such complex iBGP policies, and we discuss the implementation of a prototype tool that is able to check the correctness of the iBGP configuration with arbitrary iBGP policies.

Deciding whether a given iBGP configuration can lead to routing instabilities is computationally hard even when no iBGP policies are applied and a single prefix is considered [24]. Nevertheless, previous work [7], [8], [25] has shown that, in the absence of iBGP policies, the intrinsic complexity of the configuration check turns out to be manageable in practice through a heuristic-based approach. In the following, we show that, luckily, routing instabilities can be efficiently detected even in the presence of iBGP policies.

All the existing heuristics build on the intuition that the IGP topology is the only factor that might prevent iBGP to converge. Unfortunately, iBGP policies are fundamentally incompatible with this assumption. For this reason, we cannot reuse the techniques proposed in [7], [8]. Moreover, while more general, the technique in [25] is targeted to eBGP, and not directly applicable to iBGP, e.g., because route propagation rules are not considered. In this section, we extend the technique presented in [25] by *(i)* adapting design and implementation to check an iBGP configuration with iBGP policies; and *(ii)* adding optimizations tailored to route reflection configurations. We further describe a prototype tool that can check routing convergence of iBGP configurations with arbitrary iBGP policies. Such a tool is intended for *off-line* operation, i.e., to provide operators with useful information on convergence guarantees in the following cases:

- **configuration assessment**, to check if the running configuration guarantees iBGP convergence independently of the routes received from eBGP neighbors;
- **pre-deployment configuration check**, to verify the correctness of a new configuration before it is deployed to the production network; and
- **what-if analyses**, to highlight possible side effects of changes (e.g., topological changes or policy modifications) to a correct iBGP configuration.

Finally, we fully evaluate the performance of our prototype, showing the feasibility of our approach for off-line analyses.

A. Design and Optimizations

Figure 8 summarizes the design of a convergence checker that supports iBGP policies. The checker translates iBGP configurations to instances of a formal model which is an extended version of the SPP model [3], commonly used to study BGP stability. Our extended model is fully described in the Appendix. Then, the tool runs a stability check on the extended SPP instances using the known Greedy+ algorithm [25]. This algorithm either correctly reports the instance as stable, or

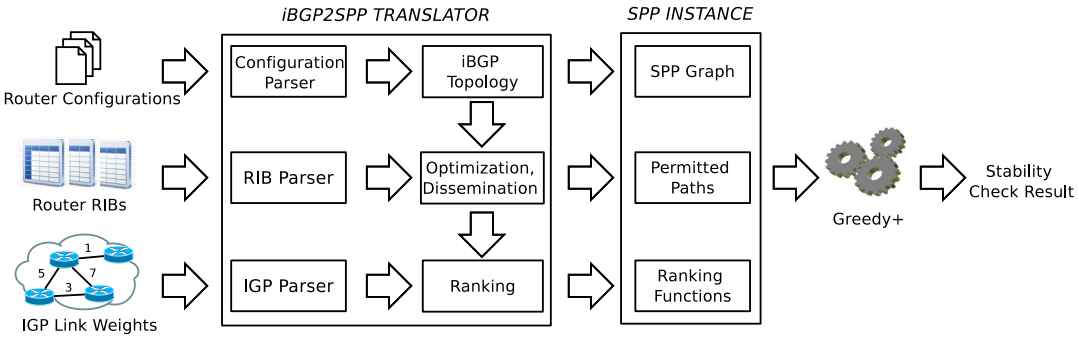


Fig. 8: Architecture of the convergence checker.

pinpoints a set of routers that may be responsible for routing instabilities.

We now provide more details of the configuration translation process. In the first step, BGP configuration files kept by each router are parsed to extract the iBGP topology. Then, the tool considers each prefix in isolation, mimicking the independent handling of different destinations in BGP. For each prefix p, the tool extracts the eBGP routes received by of its egress points from their BGP Routing Information Base (RIB). Then, it simulates the propagation of those routes through the iBGP topology, while respecting the iBGP route propagation rules. At this stage of the process, we are only interested in route propagation in order to enumerate all the valid signaling paths to p, so we do not deal with route ranking and route selection. The simulation of route propagation is performed in two distinct phases, which we call Optimization and Dissemination phases.

In the Optimization phase, routers having no clients and not being an egress point for a prefix p are excluded from the convergence analysis of iBGP for p. We refer to this pruning step as to an *optimization pass*. After an optimization pass, we might find additional routers that have no clients (e.g., because their clients have been pruned) and are not egress points for p. This means that we can iterate optimization passes to prune even further. Our tool supports a configurable number of optimization passes. A proof that the routers that we disregard in each optimization pass can never be responsible for routing instabilities is reported in the Appendix.

In the *Dissemination* phase, we simulate BGP route announcements among the remaining routers, taking into account the possible application of iBGP policies at each router. At the end of the route propagation process, for each node, we have a set of routes corresponding to the valid signaling paths from that node to any egress point to p.

As a final step, we perform the *Ranking* phase, in which we run the full BGP decision process at each node u. Note that, to perform Step 6 of the BGP decision process, we need to know the IGP topology with the corresponding metrics. After the ranking phase, we obtain a list of preference over the valid signaling paths of each node. The iBGP topology, enriched with the lists of preferences on each nodes, constitutes our SPP instance (see the Appendix).

B. Implementation and Evaluation

We implemented the architecture shown in Figure 8 by developing a prototype tool. Our tool has a core Java component that, for each prefix, performs the Optimization and Dissemination phase, computes rankings, creates the corresponding SPP instance, and runs GREEDY+ on it. Besides this component, our prototype currently features:

(i) ad-hoc scripts for configuration parsing, based on code from the `BGP2CBGP` project [26];
(ii) an MRT [27] parser for router RIBs; and
(iii) an IGP parser, which can process a set of OSPF link weights, e.g., as obtained through the SNMP protocol, to compute shortest path distances.

To evaluate the practical applicability of our approach, we ran several experiments on an entry-level server equipped with a 3 GHz CPU and 32 GB RAM. Since we never experienced problems in memory utilization, we focus on computing time.

We tested our prototype on both in-vitro and real-world iBGP configurations, performing two distinct kinds of analyses, namely performance tests and sensitivity analyses. Performance tests were designed to understand whether the tool can provide reasonable performance even on large iBGP configurations. Sensitivity analyses, instead, aimed at evaluating what factors significantly affect the performance of our tool. Except for the convergence check on a single ISP topology (i.e., a medium-sized Italian ISP), all the experiments reported in the following are extensions of [10].

As a *base topology*, we used a synthetic iBGP topology consisting of $1,100$ iBGP routers organized in a three-layer route reflection hierarchy. To mimic a realistic design of a redundant iBGP topology, each route reflector has 5 clients and each iBGP router has 2 route reflectors. Also, we injected 20 eBGP routes per prefix. Since the number of eBGP routes that need to be propagated per prefix is typically lower than 20 even for very large networks [7], our simulations reflect a conservative worst-case scenario.

Performance Analysis

To assess whether our approach is viable for large networks, we performed the convergence check of the base topology with our prototype tool. For statistical relevance, we repeated the convergence check 50 times, each time randomly selecting

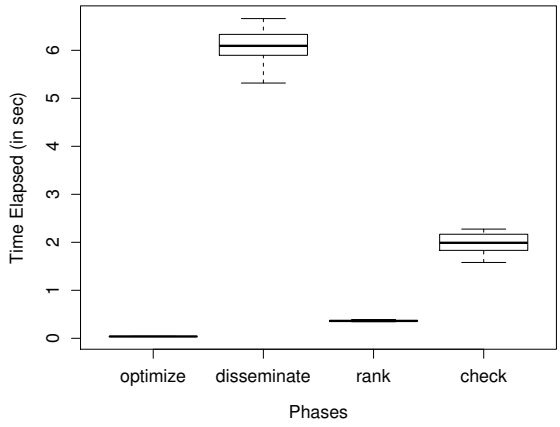


Fig. 9: Breakdown of the time taken for a single prefix check according to the different phases.

20 egress points, with uniform probability. In those experiments, the time taken by our tool to check iBGP convergence guarantees has always been lower than 10 seconds per prefix. The vast majority of the processing time was spent in the Dissemination phase, as showed by Figure 9. In the figure, phases are reported on the x-axis, and a box and a whisker are associated to each phase. For each phase, the lower and upper ends of the corresponding whisker respectively represent the minimum and maximum time elapsed for completing the phase. Moreover, the lower and upper ends of the box represent the first and the third quartile of the time taken by the corresponding phase, while the thick horizontal line within the box represents the median. The variance of the results depends on the position of the egress point in each test which, in turn, influences the effectiveness of our optimizations. Note that the apparently higher variability for the Dissemination and Checking phases is mainly an effect of higher mean values. In fact, the coefficient of variation (i.e., the ratio between the standard deviation and the mean) is similar across the different phases, ranging from approximately 8% (Checking phase) to approximately 2% (Optimization phase).

Sensitivity Analyses

To understand the impact of different factors on the processing time of the tool, we performed sensitivity analyses. Starting from the base iBGP topology, we introduced controlled variations of different parameters, namely, the number of optimization passes, the number of iBGP policies configured on each router, the number of clients per route reflector, the number of route reflectors per iBGP router, the number of egress points, and position of egress points. For each setting of those parameters we ran a different experiment. For statistical relevance, we repeated each experiment 50 times. Results are reported in Figure 10 as complementary cumulative distribution functions (CCDFs).

Firstly, we evaluated the importance of the Optimization phase, by checking the base topology using different numbers of optimization passes (as defined in Section VII-A). Results are reported in Figure 10a, where the x-axis is in logarithmic scale. In the figure, each curve represents experiments with a number of optimization passes ranging from 0 to 4. Zero indicates that the tests are performed without pruning the initial topology. The curves labeled with $x \geq 1$ relate to tests in which the topology is pruned using x optimization passes. The plot highlights the importance of our optimizations, since de-activating them translates to an increase in the computing time by an order of magnitude. However, the impact of the number of optimization passes quickly drops as the number increases. In our topology, there is basically no performance gain for a number of passes greater than 2. For this reason, all the other experiments discussed in this section have been performed using 2 optimization passes.

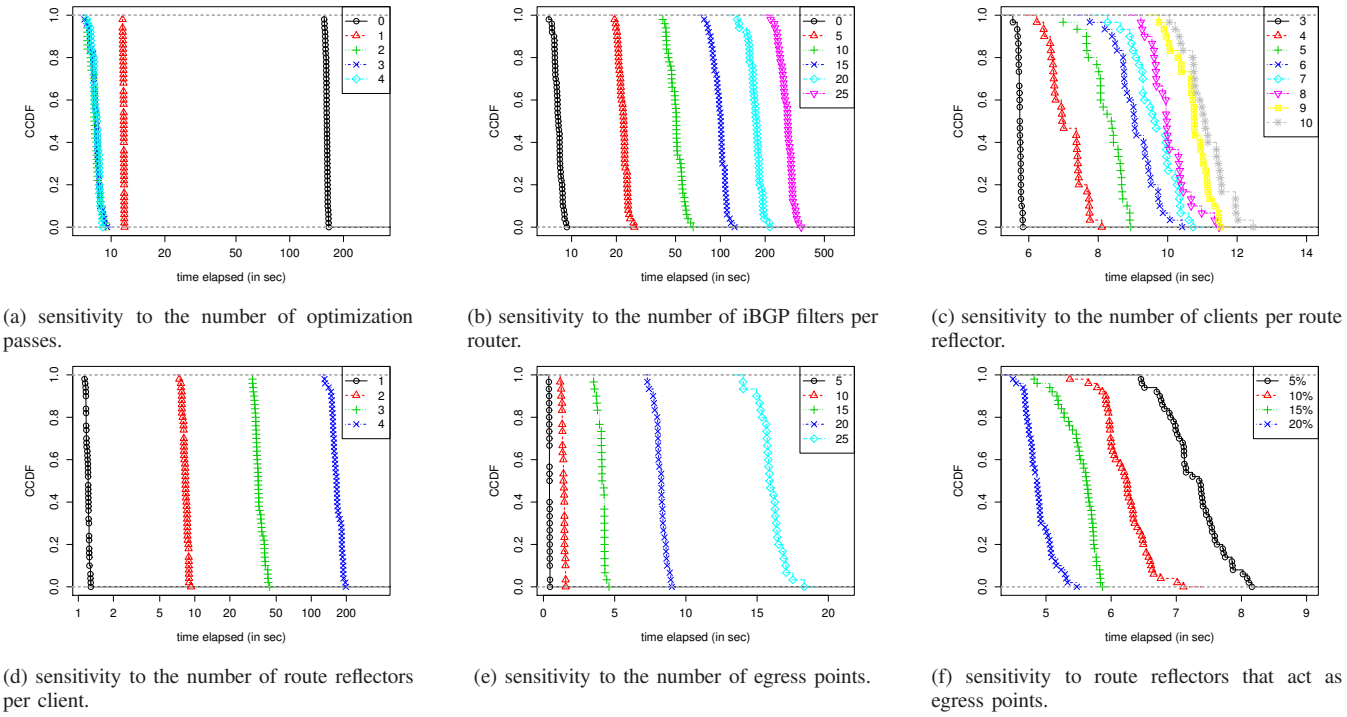
Figure 10b illustrates the impact of configuring an increasing number of BGP filters on routers. In this experiment, we used simple filters that match all BGP announcements and modify the value of the `local-preference` attribute. The different curves in the figure refer to different numbers of filters per router. Although the performance degradation is non-negligible when the number of filters increases, the total computing time is slightly greater than 1 minute in few cases, when 10 filters have been configured per iBGP router, and around 5 minutes for 25 filters. We stop our experiment at 25 filters per prefix per router as we argue that such a number of filters is enough to support complex iBGP policies.

Even better scalability properties are exhibited with respect to the number of clients per route reflector. Figure 10c shows that all our tests terminate in less 10 seconds even when each route reflector has as many as 10 clients. In this case, after a step between the case with 2 and 4 clients per route reflector, the increasing of computation time exhibits a sub-linear trend.

On the contrary, we found that the number of route reflectors per iBGP router significantly affects the performance of our checker. Figure 10d shows that the computing time tends to increase exponentially with the number of route reflectors per client (note that the x-axis of the plot is logarithmic). A possible explanation for this performance degradation is the exponential increase of valid signaling paths in the configuration. Luckily, realistic redundant topologies typically features no more than 2 route reflectors per client [23]. In this case, the computing time is less than 10 seconds per prefix.

Similarly, Figure 10e shows how the computing time tends to grow quickly with the number of egress points, but remains still affordable for the extreme cases of 20 and 25 egress points per prefix. Note that a higher number of egress points corresponds to a larger variability of our results. This is because the effectiveness of the Optimization phase is inversely proportional to the number of route reflectors having a client which acts as an egress point. Hence, the random selection of egress points heavily affects the effectiveness of our optimizations, especially for high numbers of egress points.

Figure 10f shows the results of experiments in which an increasing percentage of route reflectors act as egress points. We find that the performance of the tool improves when more egress points are placed higher in the route reflection hierarchy. This is again an effect of our optimizations. Indeed, a higher number of route reflectors that act as egress points corresponds



(a) sensitivity to the number of optimization passes.

(b) sensitivity to the number of iBGP filters per router.

(c) sensitivity to the number of clients per route reflector.

(d) sensitivity to the number of route reflectors per client.

(e) sensitivity to the number of egress points.

(f) sensitivity to route reflectors that act as egress points.

Fig. 10: Sensitivity analysis of the iBGP convergence checker performance.

to a fewer number of egress points at the bottom layer. This leads to higher effectiveness of the first optimization pass which can prune a higher number of routers.

Real Topologies

Finally, in order to test all the components of our prototype and confirm the applicability of our approach, we checked the iBGP configuration of a medium-sized Italian ISP and the one of a Tier-1 ISP.

The Italian ISP consisted of almost 40 iBGP routers and two route reflectors. We ran a test for every prefix in the full Internet routing table and found the configuration to be stable in all the cases. The full test took a few minutes.

We also checked iBGP convergence of the backbone topology of a Tier1 network. The topology consists of more than 100 routers arranged in a three layer iBGP hierarchy. Egress points were present in each of the three layers. Since we didn't have the full configuration of the routers, we considered the case in which each route reflector prefers any route forwarded by its clients over any route propagated by its non-clients, i.e., implementing the *prefer-client* condition stated in [4]. We ran the test on the full routing table. Since it is useless to test separately two prefixes having the same set of egress points and the same iBGP policies, we grouped prefixes in equivalence classes. We obtained $1,500$ equivalence classes, and the convergence check took about 40 minutes in total.

Discussion

Our evaluation highlights the viability of our approach for off-line iBGP convergence checks, both on regular synthetic topologies and on real-world configurations. Beyond showing good performance of our prototype, the evaluation identified the most critical phase from a time consumption point of view in the Dissemination phase. Furthermore, it sheds light on the factors to which the performance of the tool is more sensitive. In particular, we found that the factors affecting the number of valid signaling paths (like the number of route reflectors per client) or limiting the performance of our optimizations (like the number of egress points) are the most critical for our approach. This further stresses the importance of the Optimization phase in our architecture.

To make sure that the evaluation was not biased by the guaranteed stability of the checked configurations, we repeated all our experiments deliberately adding pairs of unstable routers at random points in the iBGP topology. Our tool always correctly reported the configurations as unsafe. Also, the checking phase has been slightly faster than in the previous experiments. This is because the checking algorithm stops as soon as it detects potentially unstable structures [25].

Of course, our tool may have worse performance on more complex configurations or larger topologies. However, our prototype can still be enriched with several performance improvements, among which parallelization of the check of different prefixes on multiple CPUs and code optimization, e.g., to avoid redundant Dissemination phases for prefixes that share many egress points.

VIII. Conclusions

BGP configuration languages offer the possibility to change iBGP messages en route and deploy iBGP policies. However, such a possibility has been commonly ignored in previous research work.

Starting from an empirical study on the popularity of iBGP policies among ISPs in the Internet, this paper discusses potential benefits and drawbacks of such a practice. In particular, we identify that the basic tradeoff posed by the application of iBGP policies consists in sacrificing correctness guarantees and configuration simplicity for improved flexibility. By extending previous iBGP formal models, we prove that iBGP policies can create convergence problems in otherwise safe configurations. Even worse, known sufficient conditions for iBGP convergence do not hold only in the presence of iBGP policies. We defined stronger sufficient conditions and leverage them to propose a systematic way to take advantage of iBGP policies while avoiding the associated risks. Finally, we describe a tool for off-line correctness checks of arbitrary iBGP configurations.

REFERENCES

[1] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, 2006.

[2] T. Bates, E. Chen, and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)," RFC 4456, 2006.

[3] T. Griffin, F. B. Shepherd, and G. Wilfong, "The Stable Paths Problem and Interdomain Routing," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 232–243, 2002.

[4] T. Griffin and G. Wilfong, "On the Correctness of IBGP Configuration," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 17–29, 2002.

[5] T. Griffin and G. Wilfong, "Analysis of the MED Oscillation Problem in BGP," in *Proc. ICNP*, 2002.

[6] S. Vissicchio, L. Cittadini, L. Vanbever, and O. Bonaventure, "iBGP deceptions: More sessions, fewer routes," in *Proc. INFOCOM*, 2012.

[7] A. Flavel, M. Roughan, N. Bean, and A. Shaikh, "Where's Waldo? Practical Searches for Stability in iBGP," in *Proc. ICNP*, 2008.

[8] A. Flavel, J. McMahon, A. Shaikh, M. Roughan, and N. Bean, "BGP Route Prediction within ISPs," *Comput. Commun.*, vol. 33, pp. 1180–1190, 2010.

[9] D. Perouli, S. Vissicchio, A. Gurney, O. Maennel, T. Griffin, I. Phillips, S. Fahmy, and C. Pelsser, "Reducing the complexity of BGP stability analysis with hybrid combinatorial-algebraic models," in *Proc. WRiPE*, 2012.

[10] L. Cittadini, S. Vissicchio, and G. Di Battista, "Doing don'ts: Modifying BGP attributes within an autonomous system," in *Proc. NOMS*, 2010.

[11] N. Feamster and J. Rexford, "Network-wide prediction of BGP routes," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 253–266, 2007.

[12] R. Raszuk, R. Fernando, K. Patel, D. McPherson, and K. Kumani, "Distribution of diverse BGP paths," RFC 6774, 2012.

[13] D. Walton, A. Retana, E. Chen, and J. Scudder, "Advertisement of Multiple Paths in BGP," Internet Draft, 2013.

[14] P. Marques, R. Fernando, E. Chen, P. Mohapatra, and H. Gredler, "Advertisement of the best external route in BGP," Internet Draft, 2012.

[15] RIPE Routing Information Service (RIS), http://www.ripe.net/ris.

[16] Oregon RouteViews Project, http://www.routeviews.org.

[17] A. Dhamdhere and C. Dovrolis, "Ten Years in the Evolution of the Internet Ecosystem," in *Proc IMC*, 2008.

[18] W. Mühlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig, "Building an AS-Topology Model that Captures Route Diversity," in *Proc. SIGCOMM*, 2006.

[19] L. Gao and J. Rexford, "Stable Internet Routing without Global Coordination," in *Proc. SIGMETRICS*, 2000.

[20] M. Caesar and J. Rexford, "BGP Routing Policies in ISP Networks," *IEEE Network*, vol. 19, no. 6, pp. 5–11, 2005.

[21] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush, "10 Lessons from 10 Years of Measuring and Modeling the Internet's Autonomous Systems," *IEEE Jour. on Sel. Areas in Comm.*, vol. 29, no. 9, pp. 1810–1821, 2011.

[22] T. Griffin, F. B. Shepherd, and G. Wilfong, "Policy Disputes in Path-Vector Protocols," in *Proc. ICNP*, 1999.

[23] R. Zhang and M. Bartell, *BGP Design and Implementation*. Cisco Press, 2003.

[24] M. Chiesa, L. Cittadini, G. Di Battista, L. Vanbever, and S. Vissicchio, "Using routers to build logic circuits: How powerful is bgp?" in *ICNP*, 2013.

[25] L. Cittadini, M. Rimondini, S. Vissicchio, M. Corea, and G. Di Battista, "From Theory to Practice: Efficiently Checking BGP Configurations for Guaranteed Convergence," *IEEE Trans. Netw. and Serv. Man.*, vol. 8, no. 4, pp. 387 – 400, 2011.

[26] S. Tandel, "BGP Converter - AS-wide conversion for C-BGP," http://alumni.info.ucl.ac.be/standel/bgp-converter/, 2006.

[27] L. Blunk, M. Karir, and C. Labovitz, "Multi-threaded routing toolkit (MRT) routing information export format," RFC 6396, 2011.

APPENDIX

A. An Extended SPP Model

To formally study iBGP convergence problems, we now extend the *Stable Paths Problem* (SPP) model [3], commonly used to study BGP stability problems. For the sake of simplicity, we exclude the `MED` attribute from our analysis. However, the technique described in [5] can be adopted to include `MED` in our model.

An SPP instance S consists of an undirected graph $G = (V, E)$, a set of permitted paths $\mathcal{P}^u$ for each node u and a ranking function λ^u on paths in $\mathcal{P}^u$. Each node in G represents a router, and the special node 0 is the destination to which every other node tries to send traffic. Moreover, the set of permitted paths $\mathcal{P}^u$ represents all the routing paths that are accepted by node u. Finally, λ^u expresses the preference that node u assigns to each permitted path.

Paths play a crucial role in this model. A *path* P in G is a sequence of nodes $P = (v_k\ v_{k-1}\ \ldots\ v_1\ v_0)$, $v_i \in V$, such that $(v_i, v_{i-1}) \in E$ for $i = 1, \ldots, k$. The empty path is denoted by ϵ and represents the unavailability of a route. The *concatenation* of two non-empty paths $P = (v_k\ v_{k-1}\ \ldots\ v_i)$, $k \geq i$, and $Q = (v_i\ v_{i-1}\ \ldots\ v_0)$, $i \geq 0$, denoted as PQ, is the path $(v_k\ v_{k-1}\ \ldots\ v_i\ v_{i-1}\ \ldots\ v_0)$. Being the destination, node 0 only has a single permitted path, namely, the path (0).

A path assignment π is a function that maps each vertex $u \in V$ to a permitted path $\pi(u) \in \mathcal{P}^u$, modeling the fact that u is using path $\pi(u)$ to reach 0. The set of available paths at a vertex is modeled by the set *choices*. More formally, $\text{choices}(u, \pi)$ is recursively defined to be all paths $P \in \mathcal{P}^u$ such that either $P = (u)$ or $P = (u\ v)\pi(v)$. For a node $u \in V$ and a set $W \subseteq \mathcal{P}^u$, define $\max(u, W) = \epsilon$ if $W = \emptyset$, otherwise $\max(u, W) = P$ where $P \in W$ is the best path in W according to the ranking defined by λ^u.

A path assignment π on an SPP instance S is *stable* if, for every $u \in V$, $\pi(u) = \max(u, \text{choices}(u, \pi))$, i.e., if every node is selecting the best possible path among those that are offered by its neighbors. In this case, the modeled configuration would converge to a stable routing state. In case no stable path assignment can be reached, iBGP never converges to a stable state.

We now show how to build an SPP instance $S(X, t, p)$ which models a given iBGP configuration for AS X at time t, with respect to a given destination prefix p, assuming that iBGP attributes can be changed within the AS. The set of vertices V consists of node 0 representing routers external to X and one node for each iBGP router in X. There is an edge (u, v) for each iBGP session between iBGP routers u and v. Moreover, there exists an edge $(u, 0)$ for each egress point u that has an eBGP path to prefix p at time t. At any node $u \neq 0$, the set of permitted paths consists of the empty

path ϵ and all paths $(u\ \dots\ v\ 0)$ where $(v,0)$ is an edge and $(u\ \dots\ v)$ is a valid signaling path (see Section II) from u to v. If an egress point u receives multiple eBGP paths to prefix p at time t, the permitted path $(u\ 0)$ represents the best among them, according to the standard BGP decision process. Permitted paths at node u are ranked according to the iBGP configuration of router u and the BGP decision process. Since Step 6 of the BGP decision process evaluates IGP metrics, we assume that these metrics are known.

Observe that our construction is more general than the one proposed in Section 5.1 of [4], where rankings are determined by only relying on IGP metrics, since the absence of iBGP policies was assumed.

With respect to routing stability, our extended model inherits the same properties of the original SPP one. In particular, *dispute wheels* represent the hallmark of BGP instabilities. Intuitively, a dispute wheel is a circular set of preferences. Formally, a dispute wheel [3] $\Pi = (\vec{\mathcal{U}}, \vec{\mathcal{Q}}, \vec{\mathcal{R}})$ is defined as a triple consisting of a sequence of nodes $\vec{\mathcal{U}} = (u_0\ u_1 \dots u_{k-1})$, called *pivot* nodes, and two sequences of nonempty paths $\vec{\mathcal{Q}} = (Q_0\ Q_1 \dots Q_{k-1})$ and $\vec{\mathcal{R}} = (R_0\ R_1 \dots R_{k-1})$, called *spoke* and *rim* paths respectively. For each $i = 0, \dots, k-1$, the following constraints hold in a dispute wheel. (i) R_i is a path from u_i to u_{i+1}, (ii) $Q_i \in \mathcal{P}^{u_i}$, (iii) $R_i Q_{i+1} \in \mathcal{P}^{u_i}$, and (iv) $\lambda^{u_i}(R_i Q_{i+1}) \leq \lambda^{u_i}(Q_i)$. Intuitively, convergence problems derive from the fact that each pivot node prefers the route from its clockwise neighbor, i.e., the rim path, over a direct path to the destination, i.e., the spoke path.

B. Correctness of our Configuration Guidelines

We now prove the correctness the statements claimed in Section VI. Consider an AS X and a prefix p. For technical reasons, we now introduce few additional definitions.

We define C_p^* as the class of the most preferred eBGP routes such that a route in C_p^* is received by at least one router in X. That is, the routes in C_p^* are the routes learned from neighboring ASes in the i-th level of the eBGP hierarchy (see Section VI), with the minimum i. Moreover, we denote the set of egress points receiving an eBGP route in C_p^* with $\mathcal{E}_p$.

Finally, we say that an iBGP path is a *descending path* if it starts from a route reflector and walks down the hierarchy, in such a way that no two routers in the path are peers. More formally, path $(r_0 \dots r_k)$, with $k \geq 0$, is descending if $\forall i = 0, \dots, k-1$ r_i is a route reflector of r_{i+1}. In the basic example in Figure 1, all the direct paths from a route reflector to a client like $(RR1\ BR1)$ is a descending path, while all the paths containing two route reflectors, like $(RR3\ RR1\ BR1)$, are not. Note that each router propagates its best route to its route reflectors if and only if it learns that route on a descending path, because of the iBGP route propagation rules (see Section II).

We now prove theorems that show the guarantees of our guidelines. We refer the reader to Section VI for a description of the relevance of each theorem.

Theorem 2. *If an iBGP configuration complies with Guidelines A, B and C, then for each prefix p every iBGP router having a descending path to an egress point in $\mathcal{E}_p$ steadily selects a route in C_p^* learned on a descending path.*

Proof: Consider any prefix p and any router r having a descending path to an egress point in $\mathcal{E}_p$. Let $P = (r_0 \dots r_k)$ be the longest descending path such that $k \geq 0$, $r_0 = r$, and $r_k \in \mathcal{E}_p$. P must exist by definition of r. We now prove the statement by induction on k.

Base case: $k = 0$. In this case, r itself is in $\mathcal{E}_p$. Guideline B ensures that r prefers its eBGP routes over any iBGP route. Moreover, r must receive at least one route in C_p^* by definition of $\mathcal{E}_p$. Hence, r steadily selects a route in C_p^* learned over the descending path (r).

Inductive case: $k > 0$. By inductive hypothesis, we assume that routers steadily select a route in C_p^* learned on a descending path if the length of all their descending paths to egress points in $\mathcal{E}_p$ is at most $k-1$. We now prove that r steadily selects a route in C_p^* learned on a descending path.

Consider the clients of r. They can be partitioned into the sets $\mathcal{D}$ and $\mathcal{N}$ of clients with and without at least one descending path to an egress point in $\mathcal{E}_p$, respectively.

Every router $n \in \mathcal{N}$ is ensured not to propagate any route of class C_p^* to r. Indeed, n cannot learn any route in C_p^* on a descending path by definition of $\mathcal{E}_p$ and $\mathcal{N}$, and it cannot propagate a route in C_p^* to r by the iBGP propagation rules.

On the contrary, r is guaranteed to steadily receive a route of class C_p^*, learned on a descending path, from every router $c \in \mathcal{D}$. Indeed, the longest descending path from c to any egress point $e_c \in \mathcal{E}_p$ must be shorter than k, otherwise r would have a descending path $(r\ c \dots e_c)$ longer than k, which is impossible by definition of r and k. Thus, by inductive hypothesis, c must steadily select a route learned on a descending path and, by the iBGP route propagation rules, it must propagate its selected route to r.

Let $\mathcal{S}$ be the set of routes received by r from all its clients in $\mathcal{D}$. Guideline C ensures that r prefers any route in $\mathcal{S}$ to all the other iBGP routes it may receive. By definition of the BGP decision process, r deterministically prefers one route R among the routes in $\mathcal{S}$. Since all the routes in $\mathcal{S}$ are steadily available to r, r is ensured to steadily select R.

The statement follows by noting that the inductive proof can be applied to any router r and any prefix p. ■

Theorem 3. *If an iBGP configuration complies with Guidelines A, B and C, then for each prefix p no iBGP router can steadily select a route which is not in C_p^*.*

Proof: Since we consider only iBGP configurations which are well-defined hierarchies (see Section II), there must exist at least one route reflector t that is at the top layer of the route reflection hierarchy and has a descending path to an egress point in $\mathcal{E}_p$. Theorem 2 ensures that t steadily selects a route $R \in C_p^*$ learned over a descending path. Moreover, the iBGP route propagation rules guarantee that t propagates R to all its iBGP peers in the top layer. Guideline A then implies that all routers in the top layer never select a route which is not in C_p^*, as they have R steadily available. By the iBGP route propagation rules, all the routers in the layer immediately below the top one also receive only routes in C_p^* from the top

layer routers, hence they are guaranteed to never select a route of a class different from C_p^*. The statement follows by noting that the latter argument can be iterated on every layer of the route reflection topology. ■

Theorem 4. *If an iBGP configuration complies with Guidelines A, B and C, then iBGP is guaranteed to converge to a unique stable state.*

Proof: Assume, by contradiction, that the given iBGP configuration may oscillate indefinitely. Then, it must contain a dispute wheel Π (see Section V). Consider any two pivot nodes u_j and u_{j+1} that are consecutive in Π. By definition of dispute wheel, u_j must prefer path $(u_j)R_j(u_{j+1} \dots e_{j+1}\ 0)$ over path $(u_j \dots e_j\ 0)$, and u_{j+1} must prefer $(u_{j+1})R_{j+1}(u_{j+2} \dots e_{j+2}\ 0)$ over $(u_{j+1} \dots e_{j+1}\ 0)$. Let $R_j = (u_j \dots r_j\ u_{j+1})$, possibly with $r_j = u_j$, and $R_{j+1} = (u_{j+1} \dots r_{j+1}\ u_{j+2})$, possibly with $r_{j+1} = u_{j+1}$. Also, for Π to oscillate, e_j, e_{j+1}, and e_{j+2} must belong to $\mathcal{E}_p$ by Theorem 3. We have the following cases.

- r_j and u_{j+1} are iBGP peers. Then, $(u_{j+1} \dots e_{j+1})$ must be a descending path, otherwise $(u_j)R_j(u_{j+1} \dots e_{j+1}\ 0)$ would not be a valid signaling path.
- r_j is a route reflector of u_{j+1}. Hence, to be a valid signaling path, $(u_j)R_j(u_{j+1} \dots e_{j+1}\ 0)$ must be a descending path.
- r_j is a client of u_{j+1}. Hence, for $(u_j)R_j(u_{j+1} \dots e_{j+1}\ 0)$ to be a valid signaling path, the reverse of R_j must be a descending path.

In the first two cases, Theorem 2 directly leads to a contradiction, as it ensures that all the routers having a descending path to an egress point in $\mathcal{E}_p$ cannot be part of a potentially oscillating structure. In the latter case, we can iterate the same argument used on u_j and u_{j+1} to all the pairs of consecutive pivot nodes in Π. Since we assumed that each router can be univocally assigned to a route reflection layer (see Section II), a cycle made only of route reflectors cannot exist. Hence, one of the first two cases eventually applies to a pair of consecutive pivots in Π, yielding a contradiction. By applying previous results [3], the absence of dispute wheels implies that the iBGP configuration has a unique stable state, hence the statement. ■

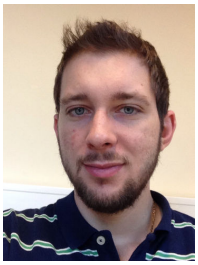
C. Correctness of iBGP Topology Pruning

We now show that the routers that we disregard in the Optimization phase of our tool (see Section VII-A) cannot be responsible for routing instabilities.

Theorem 5. *Given an iBGP topology and a prefix p, a router r cannot be part of any dispute wheel for p if r has no clients and is not an egress point for p.*

Proof: Consider a prefix p, and any router r that has no clients and is not an egress point for p. By definition of r, all its iBGP neighbors are non-clients. By definition of the route reflection propagation rules (see Section II), no iBGP neighbor of r has a valid signaling path to prefix p that traverses r. Our construction of the SPP instance then ensures that no iBGP neighbor has a permitted path through r. In turn, this implies that r cannot appear neither in a spoke path nor in a rim path in any possible dispute wheel, yielding the statement. ■

Note that the fact that router r cannot be part of a dispute wheel does not necessarily imply that r is guaranteed to select a stable route. In fact, r might indefinitely change its routing decision as a consequence of an unstable best route choice of one of its neighbors. Nevertheless, Theorem 5 ensures that r can only exhibit the symptoms of an instability, but can never be part of its root cause.



Stefano Vissicchio obtained his Master degree in 2008, and his Ph.D. degree in computer science from the Roma Tre University in April 2012. Currently, he holds a postdoctoral position at the University of Louvain (UCL) in Belgium. In 2013, two of his publications have been awarded with the IRTF Applied Network Research Prize and ICNP best paper award, respectively. Stefano Vissicchio's research interests currently span network management, routing protocols, measurements, and Software Defined Networking.

Luca Cittadini received his master degree from the Roma Tre University in 2006, and a Ph.D. degree in Computer Science and Automation from the same institution in 2010, defending the Thesis "Understanding and Detecting BGP Instabilities". During his Ph.D. he was a teaching assistant in the computer network research lab. His research activity is primarily focused on inter-domain routing, including theoretical analysis of the protocol as well as active and passive measurements.

Giuseppe Di Battista received a Ph.D. in Computer Science from the University of Rome "La Sapienza" and is currently Professor of Computer Science at the Dept. of Computer Science and Automation of the Roma Tre University. His interests include Computer networks, Graph Drawing, and Information Visualization. He published more than 100 papers in the above areas, co-authored a book on Graph Drawing, and has given several invited lectures worldwide. He is a founding member of the steering committee for the Graph Drawing Symposium.