Doing Don'ts: Modifying BGP Attributes within an Autonomous System

Luca Cittadini, Stefano Vissicchio, Giuseppe Di Battista

Dipartimento di Informatica e Automazione, Roma Tre University, Rome, Italy.
Email: {ratm,vissicch,gdb}@dia.uniroma3.it

***Abstract*—Internet Service Providers (ISPs) run the internal flavor of the Border Gateway Protocol (iBGP) for distributing routing information among border routers. While configuration languages allow routers to change iBGP attributes as a BGP message travels within the ISP's network, most prior work neglected this possibility, focusing only on the common case where iBGP attributes are left untouched.**

In this paper we aim at understanding what are the pros and cons of changing iBGP attributes.

We estimate how many ISPs change iBGP attributes, and we motivate such a practice by showing usage scenarios where modified iBGP attributes yield better traffic engineering. We also revisit a well-studied problem in iBGP, that is, routing stability. We show that changing iBGP attributes can generate routing oscillations which are not possible otherwise, and are not detectable by state-of-the-art algorithms. We present a technique to check for routing oscillations even when iBGP attributes are changed, and we give simple guidelines for changing iBGP attributes while preserving stability.

I. Introduction and Related Work

To join the Internet, an Internet Service Provider (ISP) must use the Border Gateway Protocol (BGP) [1]. More precisely, it has to exploit eBGP to exchange routes with other ISPs and iBGP for distributing routes received from neighboring ISPs among its own border routers. BGP owes its popularity to two important features: *(i)* eBGP supports expressive routing policies via simple configuration languages, and *(ii)* iBGP can achieve good scalability by employing route reflection [2].

Previous research efforts have already shown that these advantages come at the cost of potential routing oscillations: both the expressiveness of policies and the information aggregation induced by route reflectors can have a dramatic impact on routing stability [3], [4], [5].

BGP configuration languages allow border routers to change iBGP attributes that are relevant to the route selection process. However, both theoretical [4] and practical [6] research contributions neglected this peculiar feature of iBGP, assuming that those iBGP attributes which are relevant to the BGP decision process (e.g., the `local-preference` attribute) are not changed as the BGP message is passed to iBGP peers.

In this paper we investigate the possibility of changing iBGP attributes, trying to answer the following questions:

(i) What are the pros and cons of changing iBGP attributes? Why should an ISP (not) configure its routers to modify iBGP data en route?
(ii) Do ISPs actually change iBGP attributes?
(iii) How does this possibility relate to iBGP stability?
(iv) Can we profitably change some attributes in iBGP to enforce traffic engineering policies while preserving iBGP stability?

First, we discuss possible advantages of changing iBGP attributes and related caveats. Second, by analyzing BGP update traces collected at multiple vantage points in the Internet, we estimate the number of ISPs that are actually changing iBGP attributes: our data show that this practice is adopted by few ISPs. Third, we revisit a well-known theoretical model to analyze iBGP stability [4], extending it to support iBGP attributes that change within an ISP. We use this extended model to prove that changing iBGP attributes makes iBGP prone to new types of oscillations. Fourth, given that state-of-the-art algorithms to detect oscillations [6] assume that iBGP messages are left untouched, we show a technique that does not rely on this assumption. We exploit this technique to build a tool that is able to statically check an iBGP configuration for stability. Results with a prototype implementation show promising performance, hence we conclude that changing iBGP attributes does not intrinsically prevent a network operator from debugging its routing policies using advanced configuration analyses. Finally, we state configuration guidelines to change iBGP attributes in a rational and systematic way. Our guidelines are easy to configure on routers, guarantee iBGP stability even under faulty conditions, and ensure that reasonable traffic engineering policies are enforced, regardless of the behavior of other ISPs.

The rest of the paper is organized as follows. Section II covers background notions about iBGP. We outline the main pros and cons of changing iBGP attributes within an AS in Section III, and we estimate the extent to which iBGP attributes are actually changed by ISPs in Section IV. Section V analyzes the impact of changing iBGP attributes on routing stability and Section VI presents a tool to detect potential oscillations. In Section VII, we devise guidelines to modify iBGP attributes while preserving stability. Conclusions are drawn in Section VIII.

II. Background

BGP routers (also called *BGP speakers*) exchange routes to destination IP prefixes using *BGP messages*, in which each prefix is associated to a set of *attributes*. When a BGP router learns a route for a prefix, the router possibly modifies the BGP

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

TABLE I
STEPS IN THE BGP DECISION PROCESS.

Route learned from	Distribute to clients	Distribute to non-clients
eBGP neighbor	yes	yes
client	yes	yes
non-client	yes	no

TABLE II
ROUTE PROPAGATION RULES FOR AN IBGP SPEAKER.

message by editing some of its attributes, picks its *best* route for that prefix and sends it to all its *BGP peers*. The best route is selected by running the deterministic *BGP decision process*, summarized in Table I. Essentially, a route for a destination prefix is selected as best based on the values of the associated attributes. BGP configuration languages allow operators to modify the attributes carried by a message in order to influence the best route selection and therefore control outbound traffic. Some commands can even force a BGP speaker to skip some steps of the BGP decision process (see, e.g., Cisco `bgp bestpath as-path ignore` command).

Internal BGP (*iBGP*) is used by an ISP in an Autonomous System (AS) to distribute the routes that are learned from external ASes among its border routers. We refer to the modification of an attribute in an iBGP message as iBGP attribute changing (*IAC*). Observe that *IAC* implicitly takes into account the possibility to skip some BGP decision steps. As an example, skipping Step 2 has the same effect of overwriting each `as-path` with a constant string.

The original design of BGP mandated a full mesh of iBGP peerings within an AS in order to distribute the routes received from external ASes. However, the scaling issues of such a solution spurred the search for alternatives. The most common and widespread alternative to fully meshed iBGP is route reflection [2]. Route reflection organizes BGP routers within an AS in a hierarchy of *clusters*. The iBGP neighbors of each router are split into two sets: *clients* and *non-clients*. In a fully meshed iBGP network, all iBGP routers are non-clients. A router that has one or more clients acts as a *route reflector*, i.e., it relays routing information to its clients. An iBGP speaker propagates its best route according to the rules depicted in Table II: if the best route is learned from a non-client iBGP peer, then it is relayed only to clients, otherwise it is propagated to all iBGP neighbors. Each cluster has (at least) one route reflector. In order to ensure that routes are correctly distributed within the AS, there must be a full mesh of iBGP peerings at the top of the route reflection hierarchy.

We now define the concept of *valid signaling path*, which models route dissemination across the route reflection hierarchy. This concept is needed to define iBGP topology connectivity. Intuitively, the rules in Table II ensure that iBGP route distribution follows the topology of the route reflection hierarchy. Formally, let $G = (V, E)$ be the topology of the route reflection hierarchy. Namely, each node $u \in V$ represents an iBGP router, and each edge $e \in E$ represents an iBGP peering. The set of edges is partitioned into two sets **over** and **up-down**. An edge $(u, v) \in$ **over** represents the fact that v is a non-client of u and u is a non-client of v. Hence, an **over** edge indicates a vanilla iBGP peering between routers u and v. An edge $(u, v) \in$ **up-down** represents the fact that either v is a non-client of u while u is a client of v, or vice versa. Hence, an **up-down** edge (u, v) indicates an iBGP peering between routers u and v where v (u) acts as a route reflector for u (v). We say that an **up-down** edge is **up** when it is traversed from the client to its route reflector, **down** otherwise. A *valid signaling path* is any path on G that can be used to disseminate a route within the AS, according to the rules in Table II. Any valid signaling path P consists of: *(i)* a (possibly empty) sequence of **up** edges, followed by *(ii)* a (possibly missing) **over** edge, followed by *(iii)* a (possibly empty) sequence of **down** edges [4].

An iBGP topology is *connected* if there exists a valid signaling path between every pair of iBGP speakers. Intuitively, a connected iBGP topology ensures that routing information can be propagated by any iBGP speaker to any other. Throughout the paper, we only consider connected iBGP topologies.

III. WHY OR WHY NOT?

This section presents the possibilities opened by changing iBGP attributes and the drawbacks this practice can incur. We will assume the viewpoint of a single ISP managing its AS.

The main reason why a network operator might think about modifying iBGP attributes within his AS is the extended flexibility this practice allows. Operators can exploit this flexibility for implementing policies which are otherwise impossible to enforce. Figure 1a provides a simple example where AS X spans over North America and Europe, and has public peerings at Internet exchange points (IXPs) in Palo Alto (PAIX) and Amsterdam (AMS-IX). Configurations described in figures are expressed in an intuitive vendor-independent pseudo-language and are trivial to translate to any vendor-specific language. Since AS X has multiple border routers in geographically distributed locations, it employs route reflectors in order to scale its iBGP configuration. For the purpose of this example, we assume that AS X has, among others, a route reflector somewhere in the US and another one in Europe, and that route reflectors are connected in a full-mesh of iBGP peerings. Being a large ISP, X is likely to exhibit high route diversity [7], that is, multiple routes for the same destination prefix p are likely available at multiple border routers. Suppose that X receives two BGP routes for prefix p: *(i)* a BGP route advertising path $ABCD$ from a peer at PAIX, and *(ii)* another BGP route advertising path YZD from a peer at AMS-IX.

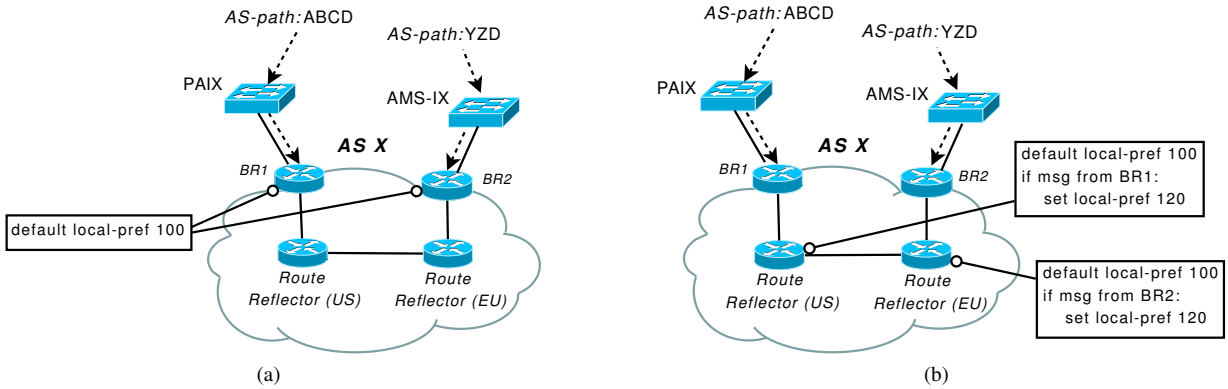

Fig. 1. (a) Default BGP configuration causes sub-optimal traffic forwarding in AS X: outbound traffic is routed through AMS-IX, due to the length of the `as-path` attribute. (b) By changing iBGP attributes, AS X is able to exploit both AMS-IX and PAIX as traffic egress points, achieving better load balancing.

Assuming that X assigns `local-preference` values according to business relationships [8], [9], the received routes are assigned the same value since they both come from a peer. For this reason, the two routes are equally ranked from the first step of the BGP decision process. The next step of the process evaluates the length of the `as-path` attribute: since the path received at AMS-IX is shorter than the path received at PAIX, every BGP router will prefer the former, which implies that all the traffic directed to p will be forwarded to Amsterdam.

Observe that AS X does not get any revenue from traffic transiting over IXPs, so its best strategy would be to minimize the cost of traffic forwarding. Since routers in the US must forward traffic towards Europe while they could simply send traffic to Palo Alto, the high-level business objective of minimizing costs seems to be not well implemented by the BGP configuration described above. Such an objective would be better accomplished if X was able to send traffic from US out of Palo Alto and from Europe out of Amsterdam, reducing the usage of cables connecting US and Europe.

Unfortunately, this simple requirement cannot be implemented (within the standard BGP decision process) unless X splits its network into multiple AS domains. On the other hand, if X performs *IAC*, it is fairly simple to force the route reflector in America to prefer American routes, and the route reflector in Europe to prefer European routes, as shown in Figure 1b. By conditionally changing the value of the `local-preference` attribute (e.g., via `route-maps`), this configuration enforces the high-level objective regardless of what `as-paths` are announced by X's neighbors.

We analyzed the BGP updates received from the border routers of a medium-sized Italian ISP and we inferred that more than 135 thousands IP prefixes (almost half routing table) were load-balanced across exit points just because of equal `as-path` lengths. Should the `as-path` length vary on one of the available routes (e.g., because of new connectivity or because the AS that originates the prefix is performing inbound traffic engineering activities via `as-path` prepending), the traffic balance would be immediately compromised. People that operate that ISP were not aware that at least 20% of their traffic is actually load balanced this way.

To better understand how a traffic shift would look like, recall the example in Figure 1a, and now suppose that the European peer of AS X started advertising an `as-path` of length 5 or more. As soon as this new route is propagated within AS X, the American route is preferred, and traffic destined to prefix p is completely forwarded via Palo Alto.

After showing that there exist benefits in manipulating iBGP attributes, we turn to study the drawbacks and caveats of *IAC*. It is a common practice not to touch iBGP attributes (see Section IV), to keep the configuration as simple and easy to understand as possible. Typically, a policy is only applied when a BGP route enters or exits the AS and iBGP is just used to distribute routes within the AS. This ensures consistent AS-wide BGP decisions, and significantly simplifies the task of translating business objectives into BGP configurations.

Another important drawback of changing iBGP attributes is that it exacerbates the iBGP stability problem, as the added flexibility can translate into the ability to create routing oscillations which would be impossible otherwise. Due to its impact, this disadvantage is discussed in depth in Section V.

IV. Changing iBGP Attributes in the Internet

Given that changing iBGP attributes provides some advantages to ISPs, as we described in the previous section, one might ask whether this practice is common in the Internet, and to what extent. Unfortunately, an exact answer to this question would require access to router configuration files, which most ISPs refuse to grant as they do not want to disclose their routing policies. However, in this section, we give a method to roughly estimate the popularity of *IAC* using public data.

In [10] it is shown that applying policies only to routes announced by eBGP peers implies that only routes that are equally good up through the first three steps of the BGP decision process (see Table I) can be selected by iBGP speakers as best routes in the steady state. The main intuition behind our measurement approach is then to search for two BGP routers in the same AS that are selecting distinct routes which are not equally good up through to the first three decision steps. In such a case, assuming a connected iBGP topology, we conclude that *IAC* is performed within the AS.

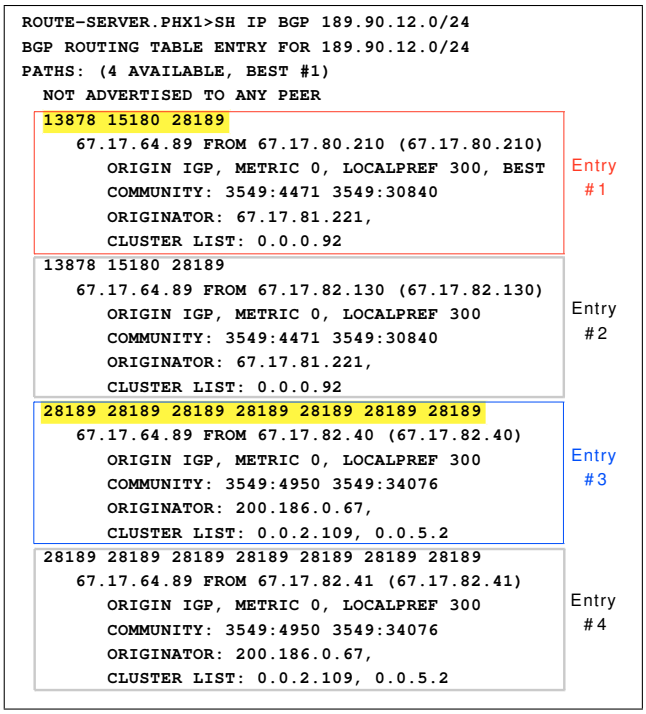

Fig. 2. A set of BGP routes that are simultaneously active within AS 3549.

Figure 2 shows a real-world example of the list of BGP routes available for destination prefix $189.90.12.0/24$ in the Global Crossing network (AS 3549), as reported by a publicly available route server on August, 31st 2009, at $14:36$ UTC. Each entry in the list (delimited by a box in the figure) represents a BGP route. The first line of each entry represents the `as-path` attribute, then other attributes follow, e.g., `local-preference`, `origin`, etc. Note that all routes were received from iBGP peers, as they include iBGP-only attributes like `cluster-list`. This implies that each route was selected as best by the corresponding iBGP peer. Observe that the first and the third entries have different `as-path` lengths (see the highlighted text in Figure 2), so they are not equally good up through Step 3 of the BGP decision process. Since the routes are simultaneously active at two distinct iBGP routers, we conclude that the ISP performs *IAC*. Of course, another possible explanation is that the iBGP topology of the ISP is not connected. However, this sharply contrasts with the objective iBGP is designed for.

For a quantitative analysis of how many ASes show this behavior in the Internet, we used the technique described in [11] for computing the sets of BGP routes for the same destination prefix which are simultaneously active in the same AS, taking as input BGP routing tables and update traces provided by RIS [12] and Routeviews [13] through May 2009. Then, when we found routes having different `as-path` length among those that are simultaneously active at AS A, we inferred that AS A was changing iBGP attributes within its network. Our analysis estimated that $1,838$ ASes out of $32,066$ (0.17%) change iBGP attributes.

Note that our estimate is actually a lower bound with respect to the real number of ASes that change iBGP attributes in the Internet. First of all, since we only have some hundreds of publicly available BGP monitors, our data do not reliably represent the full route diversity that is available in the Internet. Secondly, we only focused on the `as-path` length, disregarding other attributes that are involved in later steps of the BGP decision process. Nevertheless, our estimate confirms that the majority of ASes apply policies only to eBGP sessions and then rely on the iBGP topology just to distribute routing information within the network. However, adopting the classification of the ASes given in [14], we found that many of the $1,838$ ASes are transit providers. This could be explained by the fact that provider ASes have traffic engineering needs that are more complex to fulfill than those of customers.

V. More Flexibility ⇒ More Instability

Policy-based path vector protocols are renowned to be prone to oscillations [3] and, unfortunately, iBGP makes no exception [4]. In this section, we study how *IAC* can improve or degrade the stability of the protocol. For the sake of simplicity, we exclude from our analysis the `MED` attribute. In fact, our analysis is easy to extend to deal with `MED` adopting techniques similar to those in [5].

The *Stable Paths Problem* (SPP) framework [3] is a well-known model for studying stability properties of policy-based path vector protocols. We now briefly describe the SPP model and then we use it to prove that *IAC* can create routing oscillations which would not be possible otherwise.

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

Paths play an important 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 routing path. 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 $(v_k\ v_{k-1}\ \ldots\ v_i\ v_{i-1}\ \ldots\ v_0)$.

A path assignment π is a function that maps each vertex $u \in V$ to a permitted path $\pi(u) \in \mathcal{P}^u$. This models the fact that vertex u is using path $\pi(u)$ to reach 0. The set $\text{choices}(u, \pi)$ is recursively defined to be all paths $P \in \mathcal{P}^u$ such that either $P = (u)$ or $P = (u\ v)\pi(v)$. Given that a path assignment π represents the path chosen at each node, a set $\text{choices}(u, \pi)$ represents all the routing paths that u can learn from its neighbors. For a node $u \in V$ and a set $W \subseteq \mathcal{P}^u$, define $\max(u, W) = \epsilon$ if $W = \oslash$, 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 instance S of SPP is *stable* if, for any $u \in V$, we have $\pi(u) = \max(u, \text{choices}(u, \pi))$, that is, if every node is selecting the best possible path among those that are offered by its neighbors. In this situation, policy-based path vector protocols like BGP converge to a stable routing.

We now show how to construct an instance $S(X, t, p)$ of SPP 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 nodes consists of a special node (labeled 0) and one node for each iBGP speaker in X. Observe that some of these iBGP speakers are border routers while some others are route reflectors. There is an edge (u, v) for each iBGP peering between iBGP speakers u and v. Moreover, there exists an edge $(u, 0)$ for each border router u that has an eBGP path to prefix p at time t. At node $u \neq 0$, the set of permitted paths consists of the empty path ϵ and all paths $(u \ \ldots \ v \ 0)$ where $(v, 0)$ is an edge and $(u \ \ldots \ v)$ is a valid signaling path (see Section II) from u to v. If border router u has multiple eBGP paths to prefix p at time t, 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 much more general than the one proposed in Section 5.1 of [4], where rankings are determined by only relying on IGP metrics, since iBGP attributes are supposed to be the same at every node.

Figure 3a depicts a simple iBGP configuration, while Figure 3b shows the corresponding translation to SPP, where each node u is equipped with a list of paths representing $\mathcal{P}^u$, sorted according to λ^u (better paths are positioned higher in the list). For example, the list besides node b_1 specifies that b_1 can use paths $(b_1 \ b_2 \ 0)$ and $(b_1 \ 0)$ to reach 0, and prefers $(b_1 \ b_2 \ 0)$. The opposite happens at vertex b_2. Observe that, by modifying the `local-preference` attribute, we have been able to create a circular set of preferences which cannot be satisfied at the same time: b_1 prefers traversing b_2 rather than using the direct route to 0, and vice versa. This kind of policy conflicts can lead to routing oscillations. In fact, the SPP instance in Figure 3b is the well-known DISAGREE gadget [3], which is renowned to possibly exhibit oscillations if messages are exchanged simultaneously between routers b_1 and b_2 [15].

On the other hand, the iBGP topology in Figure 3a cannot oscillate if iBGP attributes are not allowed to be changed within the AS. 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 possible cases.

- P_1 and P_2 have different `local-preference` values. In this case, the one with the highest value is eventually selected at both routers.
- P_1 and P_2 have different `as-path` lengths. Assuming a tie in the first decision step (otherwise, we fall in the previous case), the route with the shortest length is eventually selected at both routers.

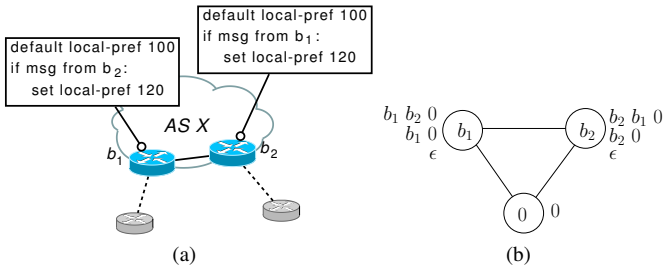

Fig. 3. (a) Configuration of two border routers that modify iBGP attributes. (b) The corresponding translation to SPP.

- P_1 and P_2 have different `origin` values. Again, assuming a tie in the previous decision steps, the route with the lowest origin is eventually selected at both routers.
- P_1 and P_2 have the same `origin` value. In this case, Step 5 of the BGP decision process implies that router b_i eventually selects P_i, $i \in \{1, 2\}$.

In every case, no oscillations can be generated.

The above discussion is an informal proof of the following theorem.

Theorem 1. *BGP configurations that allow iBGP attribute changing can generate a larger set of oscillations than BGP configurations where iBGP attributes are not modified.*

VI. A TOOL FOR DETECTING IBGP OSCILLATIONS

The problem of deciding whether a given iBGP configuration and a routing state at time t can lead to routing oscillations is NP-hard even when iBGP attributes are left untouched [4]. However, the algorithm in [6] shows that, in practice, the complexity can still be manageable. Since this algorithm only works for two levels hierarchies and under the assumption that iBGP attributes are not changed, one might ask whether *IAC* prevents an operator from using smart techniques to detect routing oscillations in his network. In this section, we show that this is not the case.

We built a prototype that can translate iBGP configurations to SPP instances in practice, enabling us to run a stability check on the SPP instance using the GREEDY+ algorithm presented in [16]. As a first step in the translation process, our prototype parses BGP configuration files to extract the iBGP peering topology and encodes this topology in a graph G (see Section V). Now, in order to compute the set $\mathcal{P}^u$ of permitted paths at each node u in the graph, we need to know the eBGP routes injected by border routers and to enumerate all valid signaling paths. To do that, we first extract eBGP routes from the BGP Routing Information Base (RIB) of each border router. Second, we simulate the propagation of each route through G. Observe that, during the simulation, iBGP attributes of a route might be changed by traversed routers according to their BGP configuration. At the end of this process, which we call the *Dissemination* phase, we end up with a set of BGP routes at each router u, which are used to compute the set of permitted paths $\mathcal{P}^u$. As a final step, we need to define the ranking function λ^u at each node u

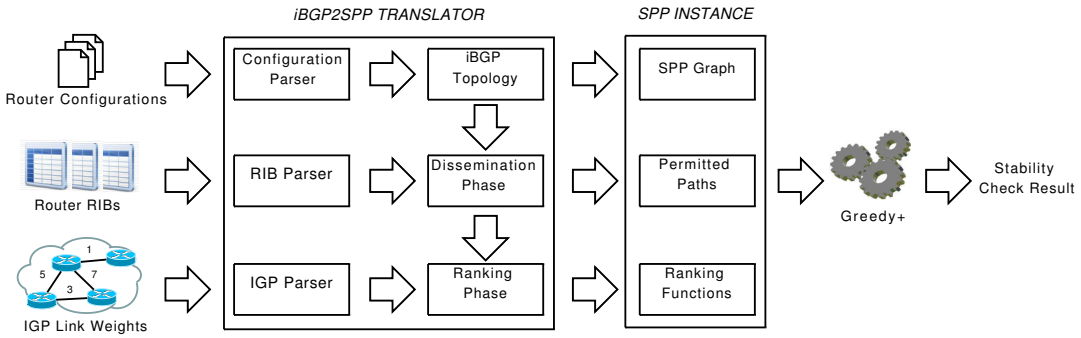

Fig. 4. Architecture of the stability checker tool.

(*Ranking* phase). To this end, we run the full BGP decision process at each node u, in order to obtain a sorted list of the BGP routes that were collected during the Dissemination phase. The corresponding ranking is used to define function λ^u. Notice that, to perform Step 6 of the BGP decision process, we need to know the underlying IGP topology.

Figure 4 summarizes the architecture of our tool. It takes BGP configuration files, RIBs and a map of IGP weights as inputs, performs Dissemination and Ranking, and produces an instance S of SPP which is then passed to the GREEDY+ algorithm [16]. This algorithm either correctly reports the instance as stable, or pinpoints a set of nodes that might be responsible for routing oscillations.

Our tool has a core Java component which performs the Dissemination phase, computes rankings, creates an SPP instance, and runs GREEDY+ on it. Besides that component our prototype currently features:

(i) a minimal parser for Cisco configuration files, which is able to parse the most common BGP statements, based on some code from `BGP2CBGP` [17];
(ii) an MRT [18] parser for RIBs; and
(iii) an SNMP-based OSPF link weight parser, which computes the all-pairs shortest distance matrix.

We tested our prototype both on in vitro and on real world iBGP configurations. Namely, in order to evaluate how much our approach can scale to large networks, we analyzed synthetic iBGP topologies consisting of up to 1100 iBGP speakers and route reflection hierarchies having at least three levels. The most time-consuming activity is the Dissemination phase, whose processing time depends on the number of eBGP routes that need to be propagated. Since this number is lower than 20 even for very large networks [6], we injected 20 eBGP routes for each prefix as a worst-case analysis. Figure 5 shows the processing time needed to run a worst-case analysis on three-levels hierarchies and a varying number of iBGP speakers. We ran our experiments on a entry-level server equipped with two 2.6 GHz quad-core CPUs and 16 GB RAM. Observe that checking the stability for a single prefix in a large network (e.g., 600 iBGP speakers) takes 0.3 seconds in the worst case. Running the analysis for the whole Internet routing table would take several hours. However, the stability check could be run only for the prefixes that experienced some

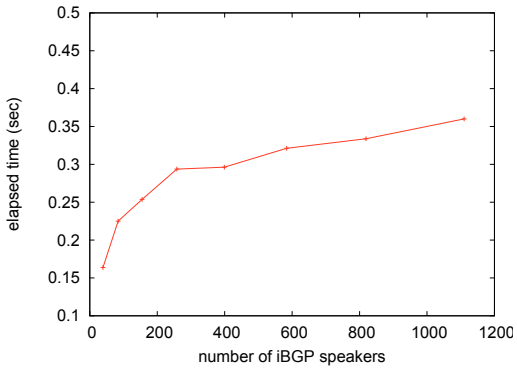

Fig. 5. Processing time to check the stability of three-levels iBGP configurations with 20 injected eBGP routes.

change in a given time frame, e.g., 15-30 minutes. Moreover, performance can still be improved if prefixes can be grouped in equivalence classes, which is frequently the case, since BGP policies are seldom specified on a per-prefix basis. The number of equivalence classes is usually one or two orders of magnitude lower than the number of prefixes (see, e.g., [6]).

Further, in order to test all the components of the prototype, we checked the iBGP configuration of a medium-sized Italian ISP, consisting of almost 40 iBGP speakers and two route reflectors. We ran a test for every prefix in the full Internet routing table ($\approx 300,000$ prefixes) and found the configuration stable in all cases. The full test took only a few minutes.

VII. PROFITABLE IBGP ATTRIBUTE MODIFICATION

Sections III and V suggest that an ISP willing to change iBGP attributes within its own network essentially faces a trade-off between flexibility and stability. In this section, we define policy configuration guidelines that safely exploit the flexibility of modifying iBGP attributes. The main concern here is to obtain benefits in terms of traffic load balancing (see, e.g., Figure 1b), while ensuring routing stability and keeping the complexity of BGP configuration manageable.

Our guidelines are meant to fulfill two main high level requirements: *(i)* Routes should be ranked according to revenues and costs; and *(ii)* Internal transit cost, i.e., the cost of forwarding traffic within the ISP network, should be minimized.

We assume that the neighbors of an ISP can be broadly classified, according to commercial relationships among ISPs, into customers, peers, and providers [8]. Selecting a route announced by a customer means forwarding traffic to that customer, which pays for it. Similarly, selecting a route announced by a peer implies that traffic is exchanged free of charge between the two ISPs. Selecting a route announced by a provider, instead, involves paying a cost. We then implement requirement *(i)* by mandating that customer routes have an higher `local-preference` than peer routes that, in turn, have an higher `local-preference` than provider routes. Moreover, to avoid offering transit service for free, routes learned from a peer or a provider are not exported to other peers or providers. This is one of the most typical way of expressing routing policies in BGP [9] and it provides the additional benefit of ensuring global interdomain routing stability [8]. Requirement *(ii)* is implemented by forcing each route reflector to prefer routes learned from its own clients, assuming that the cost of sending traffic from a route reflector to a client is less than the one of sending traffic to a non-client. This is very frequently the case, as route reflection topology design should be congruent with the network topology [2].

Guideline A. *Every iBGP speaker assigns a local preference value LP_{cust} to the routes announced by customer ASes, LP_{peer} to the routes announced by peer ASes, and LP_{prov} to the routes announced by provider ASes, in such a way that $LP_{cust} > LP_{peer} > LP_{prov}$.*

Guideline B. *Route reflectors modify the local preference value with LP_{mod} when receiving a route R from one of their clients, in such a way that*

- *if R is from a customer AS, $LP_{mod} > LP_{cust}$*
- *if R is from a peer AS, $LP_{cust} > LP_{mod} > LP_{peer}$*
- *if R is from a provider AS, $LP_{peer} > LP_{mod} > LP_{prov}$*

Figure 6 shows a simple implementation of our guidelines. First, the `community` attribute is used to tag routes according to our requirements. Then, the `local-preference` attribute is modified according to the tags. Since a very similar technique is commonly used by ISPs to manage traffic from eBGP neighboring ASes [9], we argue that our guidelines do not add significant configuration complexity.

We now prove that our guidelines guarantee iBGP stability.

Lemma 1. *If the configurations of all iBGP speakers of an AS comply with Guidelines A and B, then eventually either:* (i) *all iBGP speakers select routes learned from customer ASes,* (ii) *all iBGP speakers select routes learned from peer ASes, or* (iii) *all iBGP speakers select routes learned from provider ASes.*

Proof: Consider an AS in the steady state, and let W be the set of BGP routes to a given destination prefix that are selected as best by at least one iBGP speaker. Let C_1 be the set (*class*) of customer ASes, C_2 be the class of peer ASes, and C_3 be the class of provider ASes.

The statement is trivially true if $|W| = 1$ or if all routes in W are learned from neighboring ASes belonging to the same class. Then, assume by contradiction that there exist at least two routes r_1 and r_2 in W such that r_1 (r_2) is learned from a neighboring AS belonging to class C_i ($C_j \neq C_i$). Without loss of generality, let $i < j$. Since each iBGP speaker only propagates its best route, there must exist a border router u which selects r_1 and a border router v which selects r_2.

Let P be a valid signaling path between u and v (P must exist, see Section II). Because of the iBGP propagation rules in Table II, there must exist two speakers x and y in P such that x selects r_1, y selects r_2, and there is an iBGP peering between x and y. We have the following cases:

- x acts as a route reflector for y (or vice versa). Then, according to iBGP route propagation rules in Table II, x eventually announces r_1 to y.
- x and y are peers. In this case, we have that x learned route r_1 either from an eBGP neighbor or from a client. In both cases, iBGP route propagation rules in Table II ensure that x eventually announces r_1 to y.

Hence, y is aware of r_1 in the steady state. Guidelines A and B imply that y eventually selects route r_1 because it has a higher `local-preference` than r_2 (a contradiction). ■

Theorem 2. *If every router configuration complies with Guidelines A and B, then the resulting iBGP configuration is free from routing oscillations under arbitrary link failures.*

Proof: Consider the translation to an SPP instance S

Configuration for Border Routers

(i) Tag routes according to commercial relationships

```
if msg from customer
        add community comm_cust
if msg from peer
        add community comm_peer
if msg from provider
        add community comm_prov
```

(ii) Prefer customers to peers, and peers to providers

```
if comm_cust in community
        set local-pref 200
if comm_peer in community
        set local-pref 100
if comm_prov in community
        set local-pref 50
```

Configuration for Route Reflectors

(i) Tag routes announced by clients

```
del community comm_client
if msg from client
        add community comm_client
```

(ii) Prefer customers to peers, and peers to providers
Prefer clients to non-clients

```
if comm_cust in community
        set local-pref 200
if comm_cust and comm_client in community
        set local-pref 220
if comm_peer in community
        set local-pref 100
if comm_peer and comm_client in community
        set local-pref 120
if comm_prov in community
        set local-pref 50
if comm_prov and comm_client in community
        set local-pref 70
```

Fig. 6. A simple configuration complying with Guidelines A and B.

computed as described in Section V. By Lemma 1, we know that we can restrict our attention to routes announced by the same class of neighboring ASes. We now direct some of the edges in S and then show that the resulting instance satisfies the sufficient conditions for robustness (i.e., stability under arbitrary link failures) described in [8].

Take each edge in S and direct it from a client to its route reflector. Namely, if router u is a client of v, then we have edge (u, v). We say that v is a *parent* of u, and, similarly, u is a *child* of v. Under this convention, edges are oriented from a child to its parent. According to the conditions in [8], a partially oriented instance is free from routing oscillations if all the following conditions hold.

valley-free Permitted paths can be written as an "uphill" part, i.e., a (possibly empty) sequence of child-to-parent edges, optionally followed by a "step", i.e., an undirected edge, and terminated by a "downhill" part, i.e., a (possibly empty) sequence of parent-to-child edges.

prefer-child Each node prefers routes announced by its children to the routes announced by other neighbors.

no-directed-cycle There are no directed cycles in the graph.

The *valley-free* condition holds since the set of permitted paths $\mathcal{P}^u$ at each node u consists only of valid signaling paths, according to the translation to SPP described in Section V. Recall from Section II that any valid signaling path can be written as a (possibly empty) sequence of **up** edges, an optional **over** edge, and a (possibly empty) sequence of **down** edges, and that we oriented each edge from a client to its route reflector. The *prefer-child* condition is ensured by Guideline B. The *no-directed-cycle* condition follows from the fact that route reflectors are organized in a hierarchy (Section II), hence the orientation we defined cannot result in a directed cycle.

The statement hence follows by Theorem 5.1 of [8]. ■

Observe that Guidelines A and B act on the `local-preference` attribute. Since this attribute is evaluated at the first step in the decision process, the ISP's policy takes the highest precedence and the selected routes are guaranteed to be compliant with the policy no matter what 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, are only considered as tie breakers. 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.

VIII. Conclusions

BGP configuration languages offer the possibility to change iBGP attributes en route, but there is little understanding on the extent to which routing could be affected. This paper discusses the potential benefits and drawbacks, and proposes a systematic way to mitigate the risks of this practice. We stress that our results should not be taken as an argument supporting (nor discouraging) modification of iBGP attributes.

We show a simple scenario where changing iBGP attributes yields better traffic engineering, however we also prove that changing iBGP attributes can result in creating routing oscillations that would not be possible otherwise.

By analyzing BGP update traces collected at multiple vantage points, we estimate that at least $1,800$ ASes in the Internet exhibit a set of selected routes which cannot be explained if iBGP attributes are left untouched.

Since neither known theoretical models [4] nor practical techniques for oscillation detection [6] allow iBGP attributes to be changed, we define a way to translate an iBGP configuration to an instance of a well-known model for policy-based path vector protocols like BGP. We use this translation both to formally prove stability properties and as a practical tool to implement a prototypical oscillation detection system.

Finally, we propose configuration guidelines to change iBGP attributes in a profitable way. Compliance to our guidelines guarantees stability under faulty conditions and enforces reasonable traffic engineering policies, not depending on BGP attributes that could be modified by other ASes.

A natural question that arises is how hard it is to translate complex traffic engineering requirements into BGP configurations with iBGP attribute changing. This paper gives a preliminary answer for the case where policies follow the customer-provider pattern. However, this is far from a complete methodology tackling this issue.

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. on Networking*, 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] ——, "Analysis of the MED Oscillation Problem in BGP," in *Proc. ICNP*, 2002.

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

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

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

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

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

[11] A. Di Menna, T. Refice, L. Cittadini, and G. Di Battista, "Measuring Route Diversity in the Internet from Remote Vantage Points," in *Proc. ICN*, 2009.

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

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

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

[15] L. Cittadini, G. Di Battista, and M. Rimondini, "(Un)-Stable Routing in the Internet: A Survey from the Algorithmic Perspective," in *Proc. WG*, 2008.

[16] L. Cittadini, M. Rimondini, M. Corea, and G. Di Battista, "On the Feasibility of Static Analysis for BGP Convergence," in *Proc. IM*, 2009.

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

[18] L. Blunk, M. Karir, and C. Labovitz, "MRT routing information export format," Internet-Draft, draft-ietf-grow-mrt-10.txt, 2009.