Reducing the Complexity of BGP Stability Analysis with Hybrid Combinatorial-Algebraic Models

Debbie Perouli[§] Stefano Vissicchio[⋆] Alexander Gurney[†] Olaf Maennel[‡]
Timothy G. Griffin[≀] Iain Phillips[‡] Sonia Fahmy[§] Cristel Pelsser[×]
[§]Purdue University, USA [⋆]Universite Catholique de Louvain, Belgium [†]University of Pennsylvania, USA
[‡]Loughborough University, UK [≀]Cambridge University, UK [×]Internet Initiative Japan

***Abstract*—Routing stability and correctness in the Internet have long been a concern. Despite this, few theoretical frameworks have been proposed to check BGP configurations for convergence and safety. The most popular approach is based on the Stable Paths Problem (SPP) model. Unfortunately, SPP requires enumeration of all possible control-plane paths, which is infeasible in large networks.**

In this work, we study how to apply algebraic frameworks to the BGP configuration checking problem. We propose an extension of the Stratified Shortest Path Problem (SSPP) model that has a similar expressive power to SPP, but enables more efficient checking of configuration correctness. Our approach remains valid when BGP policies are applied to iBGP sessions – a case which is often overlooked by previous work, although common in today's Internet. While this paper focuses mainly on iBGP problems, our methodology can be extended to eBGP if operators are willing to share their local-preference configurations.

I. Introduction and Related Work

The Internet is a composition of Internet Service Providers (ISPs) that exchange routing information about Internet destinations via the Border Gateway Protocol (BGP). BGP has two parts: eBGP, used to exchange routing information between different ISPs; and iBGP, used within each ISP to distribute eBGP routes. In this paper, we focus on iBGP.

The iBGP behavior can be as complex and non-intuitive as eBGP. As a result, several factors have to be carefully taken into account when configuring iBGP, including: i) the deployment of route reflection, which involves a partial reduction in route visibility; ii) the dependency between BGP and IGP, which makes routing and forwarding decisions potentially inconsistent; and iii) the configuration of iBGP policies, which allows custom iBGP route ranking and filtering at some routers. These factors make iBGP configurations prone to both routing and forwarding anomalies [1]. Most critical are route oscillations, which can prevent convergence to a stable state.

The correctness of the iBGP configuration can affect the core business of ISPs, namely packet delivery. For this reason, significant research has been devoted to studying iBGP configuration correctness. iBGP routing correctness has been formalized in [1], where previous theory [2] developed for eBGP convergence is re-applied to iBGP, and sufficient conditions for iBGP convergence are defined. Practical techniques, based on graph-based formal models (e.g., [3], [4]) have also been developed. With a few notable exceptions [5], all previous work assumes BGP messages remain unchanged among iBGP peers, i.e., no iBGP policies are deployed. However, iBGP policies enable network operators to perform traffic engineering [5], and private discussions with operators have confirmed that complex iBGP policies are used in larger networks.

One heuristic to check iBGP configurations for guaranteed routing convergence has been proposed by Cittadini *et al.* in [5]. This technique checks the convergence of a BGP network separately for each combination of egress points receiving an eBGP announcement to the same Internet destination. Unfortunately, for each of these checks, the enumeration of all iBGP paths is required in the worst case. Depending on the iBGP topology, this may take a long time. For example, checking the configuration of a 600 node network by applying the technique in [5] to each prefix in a full $400,000$ routing table would take more than one day. This makes the approach infeasible to use in an online tool for conducting stability tests after configuration changes, e.g., changes to optimize network performance using traffic engineering [6].

In this paper, we extend the SSPP algebraic model described in [7], and propose algorithms to check iBGP convergence properties in the resulting model. To the best of our knowledge, ours is the first attempt to leverage routing algebras to model iBGP policies. Previous work on routing algebras has focused on eBGP, proposing approaches to build provably correct routing protocols [8], or to revise the current protocols [9].

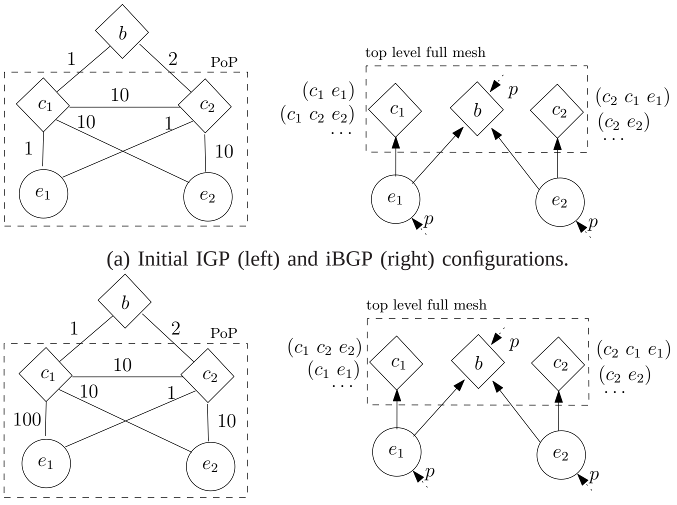
II. The Need for Efficient Convergence Checking

The primary role of iBGP is to distribute eBGP routing information to all routers inside an ISP. However, configuring iBGP to produce a given dynamic behavior is not straightforward, as several interacting factors have to be considered.

Route reflection [10] is commonly deployed in today's networks. Every iBGP router can either be a client, peer or route-reflector. The propagation rules [10] limit eBGP route visibility to be partial, i.e., not all routers will know the eBGP routes received by all egress points in the network.

Beyond route reflection, the dependency among iBGP and IGP must be considered. Indeed, one criterion that iBGP routers use to select the best route to a given destination is the IGP distance to the egress point (i.e., the closest iBGP router that receives the eBGP route) [10]. It has been shown that the dependency among iBGP and IGP, combined with the partial route visibility introduced by route reflection, can trigger

This work has been sponsored in part by Cisco Grant 570873.



(a) Initial IGP (left) and iBGP (right) configurations.

(b) IGP (left) and BGP (right) configurations after an attempt to prevent congestion of (C1,E1).

Fig. 1: A case in which an efficient stability checker helps to avoid BGP convergence problems due to traffic re-balancing.

different kinds of routing and forwarding anomalies in certain configurations [1], [11]. Such anomalies are exacerbated if iBGP policies are deployed.

Consider the configuration depicted in Fig. 1. The IGP configuration is represented by a simple weighted graph, with an IGP link weight associated with each edge in the graph. The graph represents a Point-of-Presence (PoP) and a simplified backbone encompassing a single router b. The link weights can be assigned proportionally to link capacities. Based on link weights, egress point e_2 is used by both c_1 and c_2 only as a backup. The iBGP configurations form a two-layer route reflection, in which routers b, c_1 and c_2 are in the top layer, while e_1 and e_2 are in the bottom layer. For simplicity, we restrict our analysis to the prefixes for which b, e_1, and e_2 are egress points (hence, selecting the eBGP route). The route-reflector relationships are indicated by directed edges, with the arrow pointing to the route-reflector.

Suppose that a primary traffic engineering goal for this network is to enforce locality of the traffic, so that routers in the PoP always prefer sending Internet-destined traffic to an egress point in the same PoP. Instead of setting ad-hoc link weights and losing their semantics, a simple iBGP policy can be enforced: each router prefers BGP routes received from an egress point in the same PoP over any other. This iBGP policy can be configured by increasing the local preference value of intra-PoP routes.

A first question may be: "Is iBGP guaranteed to reach a stable state in this configuration, whatever eBGP routes are received by the egress points in the network?" The answer in this case is yes, as it is possible to show that both c_1 and c_2 always select the same best eBGP route. Observe, however, that the question is not trivial in general, as the interaction between partial route visibility, IGP topology and iBGP policies must be taken into account. Specifically, all the scenarios corresponding to different eBGP routes received at the egress points need to be evaluated.

Even worse, slight modifications to any part of the configuration can change the answer to the question above. Suppose, for example, that the link between c_1 and e_1 starts to become overloaded at a given time t. In order to react to the congestion threat, one tempting option is to re-balance c_1's traffic by forcing c_1 to prefer e_2 over e_1. A reasonable approach is to increase the weight of the IGP link (c_1, e_1). However, this is not harmless, as c_1 and c_2 form a potentially oscillating structure called DISAGREE [2] in the iBGP configuration.

Indeed, both c_1 and c_2 de-prefer the route from b, by iBGP policy. Moreover, c_1 prefers the route from e_2 which it receives only from c_2, and symmetrically c_2 prefers the route received from e_1 and propagated by c_1. Again, observe that the DISAGREE is due to the combination of iBGP policies, partial route visibility, and IGP weights. Detecting this kind of routing anomaly is difficult for network operators without the support of an automatic tool to check convergence, especially if they administer large networks and have to make quick decisions (such as in this case where a sudden increase in traffic levels occurs). A tool may also be valuable to evaluate configuration tweaking alternatives to prevent congestion (or react to link failures and eBGP events) while ensuring iBGP convergence.

III. The Modeling Power of SSPP

We start by reviewing the basics of the SSPP model and discuss how it can applied to operational networks in the presence of iBGP policies. We will highlight cases in which the straightforward application of SSPP is insufficient for accurately modeling iBGP configurations. We assume that the local preference values have the same semantics network-wide, i.e., all iBGP routers use the same set of local preferences.

A. Original SSPP

The Stratified Shortest Path Problem (SSPP) [7] is an algebraic model that provides a sufficient condition for safety of protocols which resemble BGP. In this model, paths are characterized by two metrics (both non-negative integers): (1) the *stratum* level, and (2) the length of the path. When a node compares two paths, it compares their metrics lexicographically. The resemblance to BGP stems from the fact that the stratum and the path length are similar to the local preference attribute and the AS path length in BGP, respectively. The lexicographic comparison of the two attributes in SSPP matches the first two steps in the BGP decision process. One difference is that higher local preference in BGP means a more preferred route, whereas the opposite is true for the stratum attribute in SSPP. For instance, among the local preference values $\{200, 100, 80\}$, 200 corresponds to the most preferred route, but among the strata values $\{0, 1, 2\}$, stratum 0 corresponds to the most preferred route.

The policies on BGP sessions are viewed as functions on links in SSPP. These functions operate on the stratum and the path length as a route is announced from one node (router) to another. Assuming that the path length always increases during propagation in SSPP, a sufficient condition for safety is that the function operating on the stratum of a route is *inflationary*. In other words, the stratum of a route should never decrease as the route propagates from one router to another.

B. Straightforward Application of SSPP is Insufficient to Check iBGP Policies

Fig. 2 presents an example where the AS under consideration uses route reflection. Routers A and B act as route reflectors, while C is the client of A, and D the client of B. Assuming that AS1's neighbors can be classified into customers, peers and providers, the border routers of AS 1 assign local preference 200 to customer routes, 100 to peer routes and 50 to provider routes. The local preference attribute is only modified on the iBGP session between A and B, so that each of the reflectors prefers routes from their own clients over routes from the other reflector (e.g., to keep the traffic local).

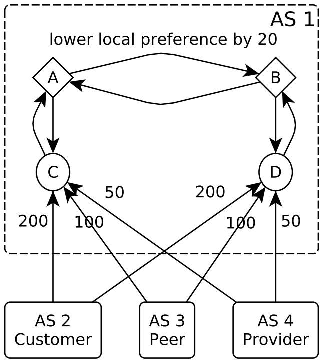


Fig. 2: Routers A and B are reflectors, while C and D are clients. The arrows show the direction of BGP announcements. Local preference is not modified along the links between A and C or B and D.

If the inflationarity property holds when the iBGP policies are modeled with SSPP, then the policies are guaranteed to be safe. The inflationarity property requires all the functions on the links (which have strata values as input and output) to be inflationary. Since there are six local preference values used across the iBGP network $(200, 180, 100, 80, 50, 30)$, one can start the modeling process by mapping them to six strata $(0, 1, 2, 3, 4, 5)$ respectively. Verifying that the functions on all the links of Fig. 2 are inflationary is straightforward. For instance, the function on the (C, A) link is $f(s) = s$, because local preference does not change when a route is announced from C to A. The function on the (A, B) link is $f(s) = s+1$, which is also inflationary, because local preference changes from $200, 100$ or 50, to $180, 80$ or 30, respectively.

Unfortunately, the analysis of an iBGP configuration is not always so simple using SSPP. For example, in order to enforce the same prefer-client rule as before, AS 1 can manipulate the local preference attribute inside its network in a different way. Namely, it can enforce that route reflectors increase the local preference of routes from their own clients with respect to those announced through other route reflectors. For this reason, the reflectors can assign 220 to customer routes, 120 to peer routes and 70 to provider routes, if such routes are learned from their own clients. Otherwise, they use the same local preference rules as the border routers ($200, 100$, and 50). The set of local preference values used throughout the iBGP network in this case is $\{220, 200, 120, 100, 70, 50\}$. The first attempt to model this network with SSPP is with six strata $(0, 1, 2, 3, 4, 5)$. For instance, the function $f(1) = 0$ on the (C, A) link of Fig. 3 captures the fact that a route which is assigned local preference 200 from the client border router C receives local preference 220 from the route reflector A. The functions on the (A, B) link are the same as the functions on the (B, A) link.

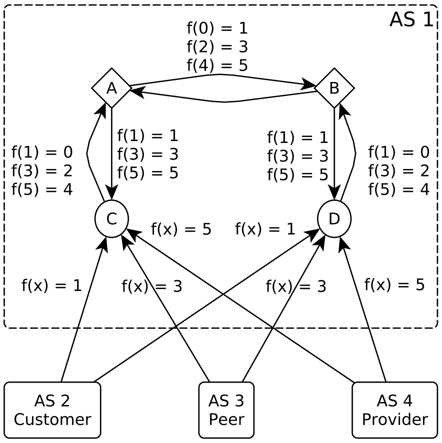


Fig. 3: The functions on the (C, A) and (D, B) links seem to violate the inflationary property that strata functions need to follow in SSPP.

With this strata assignment, the functions on the (C, A) and (D, B) links are not inflationary, since the stratum of a route is decreased across these two links. We now discuss the options to make the link functions inflationary by adjusting the strata value assignment. Consider the function $f(1) = 0$ on the (C, A) link. To make this function inflationary, two options are available, that is, either stratum 0 (which is the value of the function) is replaced by a value greater than 0, or stratum 1 (which is the argument of the function) is replaced by 0. Suppose we replace the function by $f(1) = 1$. Then, we have changed the preferences of A into equally preferring a customer route from B or C, which unfortunately does not reflect the original local preference semantics. Alternatively, replacing $f(1) = 0$ with $f(0) = 0$ does not change the semantics. If C replaces function $f(x) = 1$ with $f(x) = 0$ on the (AS 2, C) link, then the replacement $f(0) = 0$ on the (C, A) link is valid: C still prefers most the route from AS 2 and the better stratum (0) it assigns to it does not affect its other preferences. However, these changes will then make the functions on the (A, C) link non-inflationary resulting in a seemingly endless cycle of stratum changes.

IV. Using Strata Dependencies to Model BGP Policies

We design a data structure, the *Strata Digraph*, to capture the dependencies among the strata values that are assigned to routes across a network. The existence of a particular type of cycle in this graph means that there is no set of strata values which can satisfy all the requirements placed by both the iBGP policies and the sufficient conditions for safety. We also show how the “prefer routes learned from eBGP over iBGP” rule of the BGP decision process can be modeled without complicating the SSPP algebra.

A. Requirements

From the configuration file of each router r_i, we construct the set L_i which consists of all the local preference values that the router uses, including the default value. We sort $L_i = \{l_{i1}, l_{i2}, l_{i3}, ...\}$ in decreasing order so that $l_{i1} > l_{i2} > l_{i3} > ...$. We map each value l_{ij} to a stratum s_{ij} in the SSPP model. The strata values associated with a router need to respect the order imposed by the L_i values, i.e.: $s_{i1} < s_{i2} < s_{i3} < ...$. Note the difference in the direction of the inequalities between the local preferences and the strata.

In addition to respecting the local preference rankings of each router, the strata functions need to follow the inflationary property of SSPP. The inflationary property requires that when a route which has stratum s_a in router r_i is announced to $r_{i'}$ and has stratum s_b, then $s_b \geq s_a$. This type of inequality is enforced on every BGP session across which the local preference attribute changes value.

B. The Strata Digraph

We encode the requirements that the strata values need to follow in the *Strata Digraph*. A node in the graph represents a stratum. For each inequality $s_b > s_a$ or $s_b \geq s_a$, we add an arc from s_a to s_b. If the graph has a cycle in which at least one of the arcs corresponds to a strict inequality, then the set of inequalities is impossible to solve. Otherwise, the system is guaranteed to be safe since it satisfies the sufficient condition of SSPP.

Fig. 4 depicts examples of strata digraphs. Fig. 4a corresponds to the topology of Fig. 2. We observe that there is no cycle in this digraph, therefore strata values can be assigned and the iBGP policies are guaranteed to be safe. In contrast, the digraph in Fig. 4b contains cycles where at least one of the arcs corresponds to a strict inequality.

We examine the strata dependencies in one such cycle: C1 → A1 → A2 → C1. The (C1, A1) arc indicates that a route which has the highest local preference in border router C (customer route) is assigned the highest local preference of route reflector A. The (A1, A2) arc denotes that customer routes learned from C1 are assigned higher local preference by A than customer routes learned from reflector B. Finally, the (A2, C1) arc shows that customer routes that A learns from B are announced to C, then C assigns to them the same local preference used with customer routes it learns from eBGP.

A key observation is that the dependencies captured in the strata graph so far only model the iBGP policies that are expressed through the local preference attribute. Factors that come later in the BGP decision process are not taken into consideration. However, by modeling further steps, it is possible that initial cycles disappear. We consider the case where the eBGP routes have equal AS path length. We can then model the “prefer routes learned from eBGP over iBGP” step without adding features to the SSPP model. Fig. 4c presents the new strata digraph for Fig. 3. We replace each stratum of the border routers (C and D) with two strata. For example, stratum C1 is now split into strata C1e and C1i to denote that although routes learned from eBGP sessions may receive the same local preference as routes learned from iBGP, the eBGP routes will be preferred over the iBGP routes. For each stratum X that is split, the Xe stratum is strictly preferred over the Xi stratum.

The new strata digraph for Fig. 3 has no cycles that involve a strict inequality. This means that although the given iBGP policies are not guaranteed to be safe, the combination of these policies and the “eBGP over iBGP routes” rule results in a safe configuration.

V. Hybrid SSPP Model

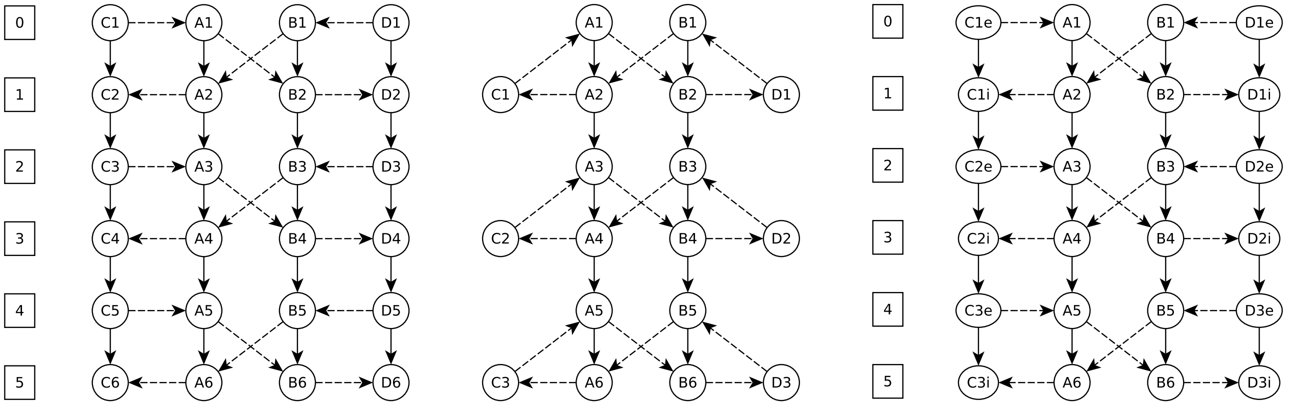
Algorithmically, the SPP model [12] requires (inefficient) path enumeration in order to determine whether a given instance is safe, since we can assume nothing about the preferences. We would, however, also like to avoid overly-elaborate algebraic definitions of routing instances, where many BGP nuances may be represented, but not necessarily in a way that is conducive to analysis or understanding.

Our proposed hybrid solution, an extended SSPP model, combines ‘arbitrary’ path ranking in its first attribute, with additional shortest-path structure in the second attribute. This means that more effective encodings of BGP preference can become possible. We can still, in the manner of SPP, encode various preference in the strata, but we can also make more parsimonious encodings (using fewer distinct stratum values) by making use of the second component.

Naturally, BGP has high complexity, and this simple two-component picture is insufficient for capturing some of the features of particular interest to us. In this section, we propose appropriately modified versions of SSPP that encode *IGP weights*, iBGP route *propagation rules*, and BGP *communities*.

Recall that an IGP weight is calculated for the component of a path that traverses the internal routing domain of an AS. No other links contribute to the total weight of a path. If we consider the propagation of a path through iBGP, we see that the IGP weight ‘starts’ at zero (at the border), and is increased with each traversed link. Meanwhile, the AS path length is not increased, because no AS boundary is being crossed. Under lexicographic attribute comparison, IGP weight is only taken into account if AS path length and a few other attributes (including the notorious MED) were indecisive.

We argue that a single distance metric is insufficient to capture both AS path length and IGP weight, under any encoding (that is, any method of computing a single number out of both values). It is tempting to try a simple lexicographic encoding. For example, given an upper bound W on IGP weight, which is the case in reality (it is a 32-bit field), we can try to use distance values where the inter-AS links have weight



(a) Strata digraph for the topology of Fig. 2.

(b) Strata digraph for the topology of Fig. 3. There are no strata values that can satisfy the inequalities of this topology.

(c) Strata digraph for the topology of Fig. 3, taking into account the eBGP $>$ iBGP rule in the BGP decision process.

Fig. 4: Strata digraphs where solid line arcs represent strict inequalities, while dotted lines represent non-strict inequalities. The value in the rectangular boxes on the left is the value that each stratum in the corresponding line has. If there exists a cycle that includes a strict inequality, then there are no strata values that can satisfy the inequalities.

W. Then, the computed path weights become $Wx + y$ if x AS boundaries were traversed, and y is the total IGP distance. Comparing these path weights is equivalent to lexicographic comparison of pairs (x, y). The problem with this encoding is that it propagates IGP weight information across ASes: this does not happen in reality, and leads to different paths being selected. The crucial issue is that the IGP weight should be *reset* to zero for each new AS, because the attribute is meant to measure distance to the egress point only. This cannot be done if all link traversals correspond to adding a single fixed number to the path weight.

Therefore, we must add another path weight metric in order to simultaneously model AS path length and IGP weight. The resulting algebra associates each path with a triple (s, d, w), where s is the stratum level, d is the AS path length, and w is the IGP weight within the current AS. On inter-AS arcs,

- The stratum value may change according to a function,
- The d value will increase, and
- The w value will be set to zero.

Within an AS, for each arc,

- The stratum value may change according to a function,
- The d value will not change, and
- The w value will increase by some non-zero value.

For correctness, the w component must be strictly inflationary, in addition to the previous criterion on the s and d values. This is achieved by requiring the IGP link weight a) to be strictly greater than zero, and b) to increase when the iBGP path becomes longer. We plan to tackle more complex cases in which (b) does not hold in future work.

BGP decision-making is also affected, indirectly, by *communities*. These are numeric tags attached to route announcements; they take no part in the BGP decision process, but they can affect the values of other attributes, and the way that routes are propagated. This is done by configuring routers to recognize particular community values and take corresponding actions—for example, excluding the route from export to a given neighbor, altering the local preference value, or editing the set of attached communities. The possible semantics are incredibly diverse, and we will not try to capture all possibilities in a uniform way within the model. However, given the importance of community-based mechanisms for BGP, we want to support the key feature of *action at a distance* on the local preference value. Since our SSPP model includes the special ‘top’ stratum for routes that are forbidden, this will suffice for community-based control of export as well. That mechanism replaces the loop-prevention feature of AS paths (since we only have an AS path length, the identity of the traversed ASes must live elsewhere, in order to stop paths from visiting the same AS more than once).

It is straightforward to augment the path tuple with a set of numeric values (drawn from some finite range). Thus, the path data is (s, d, w, C), where C is the set of communities associated with the route. Since we do not want the C sets to affect route choice, we must now allow the path comparison relation to be other than a total order: $(2, 3, 5, \{7\})$ is just as good as $(2, 3, 5, \{11\})$. Paths will now be ranked according to a *preference relation*, or *total preorder*, where there are several preference levels but multiple inhabitants of each level. Fortunately, the strict inflationary property on the initial (s, d, w) components remains sufficient for convergence, because every link traversal strictly increases the level of preference of a path.

In order to allow for communities to affect local preference, the functions that govern strata values must now be conditioned on communities, as well as on the input stratum. Fig. 5 gives an example of the use of communities to encode the iBGP behavior that routes learned from one iBGP peer are not propagated to another iBGP peer. Here, we use a community tag (comm-ibgp) to mark when a route has already traversed an iBGP link in this AS. If the tag is present, then the route is filtered out by mapping the stratum value to infinity; otherwise, the stratum value is preserved. Similarly, Fig. 6 shows an example of filtering behavior that depends on which ASes have been visited.

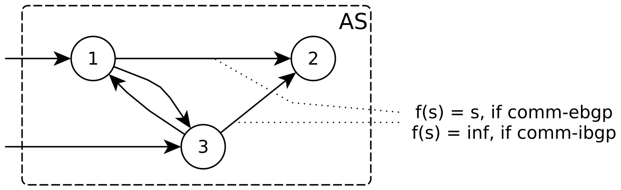


Fig. 5: On the (1,2) and (3,2) arcs, routes learned from iBGP need to be filtered while routes learned from eBGP are propagated, even if such routes are assigned the same local preference. We use "inf" for infinity.

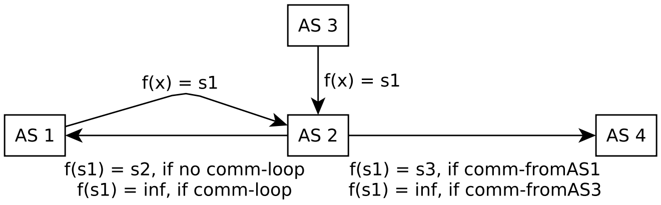

AS 3
f(x) = s1
f(x) = s1
AS 1
AS 2
AS 4
f(s1) = s2, if no comm-loop
f(s1) = inf, if comm-loop
f(s1) = s3, if comm-fromAS1
f(s1) = inf, if comm-fromAS3

Fig. 6: The use of communities is needed not only to model iBGP rules, but also eBGP loop prevention. AS 2 assigns the same stratum to routes learned from AS 1 and AS 3. The use of community in this case is required to prevent AS 2 from announcing AS 1 routes back to AS 1. Communities are also needed to implement filtering: AS 2 announces routes learned from AS 1 to AS 4, but filters routes learned from AS 3.

We now consider what operations must be supported on the community set, and how they are to be triggered. In BGP, a policy is capable of rewriting this set arbitrarily, and so it would seem that a faithful model would also include a function on each arc that could carry out such rewriting—perhaps also depending on local preference or other values. However, we argue that a simpler *append-only* model is sufficiently expressive for our purposes, while being easier to reason about. In this setting, a set of communities is attached to every arc, and these are combined with those already in the C set (by union) when the path is propagated along that arc. There is no mechanism for communities to be removed or rewritten, conditionally or otherwise.

As with the stratum encoding, it is possible to support arbitrary policy *if* greatly increasing the size of the representation is acceptable. Suppose that each arc has only one community attached, and that no two arcs have the same community. Then, each route tuple will end up with a representation of its exact path through the network, and this can be used as a 'hook' for any kind of policy we like. Now, if our policy is not quite so nuanced, then we do not need the communities to be globally unique. For example, we can have a community that means "I traversed an iBGP link in AS 1," attach it to all of the iBGP links in AS 1, and add policy as in Fig. 5 to filter out routes, on iBGP links, that already have this community applied. This is perfectly sufficient to support the iBGP mesh semantics, and does no harm in terms of the *model* for that community to remain as the route is propagated.

VI. Conclusions

Verifying the safety of an iBGP network with the SPP model requires complete path enumeration – a task of exponential complexity. More efficient algebraic approaches have been proposed, but they do not account for iBGP policies. In this paper, we propose a systematic method to efficiently verify iBGP safety with a previously proposed algebraic model: SSPP. We extend SSPP to capture functionality similar to that of BGP communities. Additionally, we model the IGP weight step of the decision process for cases when there is congruence between the iBGP and the IGP paths. We plan to consider the remaining cases in our future work. Our theoretical framework can be applied in a tool to be used by ISPs to safety-check their configurations.

References

[1] T. Griffin and G. Wilfong, "On the correctness of iBGP configuration," in *Proc. of ACM SIGCOMM*, 2002.

[2] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Netw.*, vol. 10, pp. 232–243, April 2002.

[3] A. Flavel, M. Roughan, N. Bean, and A. Shaikh, "Where's Waldo? practical searches for stability in iBGP," in *Proc. of ICNP*, 2008.

[4] 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, december 2011.

[5] L. Cittadini, S. Vissicchio, and G. D. Battista, "Doing don'ts: Modifying BGP attributes within an autonomous system," in *NOMS'10*, 2010, pp. 293–300.

[6] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE JSAC*, vol. 20, no. 4, pp. 756 –767, May 2002.

[7] T. Griffin, "The stratified shortest paths problem (invited paper)," in *Proc. of COMSNETS*, 2010.

[8] T. G. Griffin and J. L. Sobrinho, "Metarouting," in *Proc. of ACM SIGCOMM*. ACM, 2005, pp. 1–12.

[9] A. Flavel and M. Roughan, "Stable and flexible iBGP," in *Proc. of ACM SIGCOMM*, 2009.

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

[11] S. Vissicchio, L. Cittadini, L. Vanbever, and O. Bonaventure, "i-BGP Deceptions: More Sessions, Fewer Routes," in *Proc. INFOCOM*, 2011.

[12] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "Policy disputes in path vector protocols," in *Proc. of ICNP*, November 1999.