Analyzing Internet Routing Policies with Answer Set Programming

Anduo Wang
Temple University, Philadelphia, USA

1 The Route Oscillation Problem

Routing (path selection) on the Internet is driven by administrative policies made by individual networks (also called autonomous systems or ASes) that make up the Internet. In particular, border gateway protocol (BGP) [5], the current Internet routing protocol allows the participating ASes to independently set path preference based on local concerns without any global coordination. This flexibility comes at a cost: there exists local policy configurations under which the global Internet is unable to reach a stable path selection. The ASes can keep oscillate between several path selections as they continuously exchange routing information — announce or withdraw their selected path, which may further trigger path re-selection — re-computation of the best path based on newly received routing information.

How can such "oscillation" occur? In the absence of any AS policies, Internet routing is the *monotonic* computation of shortest path: For a particular destination, any shortest path to that destination selected by an AS X, denoted by P_x, must also extend another shortest path that has been selected by some neighbor AS Y that is one hop closer to the destination. But independently set AS policies make path computation *non-monotonic*: The local policy of AS X may prefer a path P_x that extends a path P_y, but P_y is less preferred by y — there exists another path P'_y that is more favored by Y. As Y discovers P'_y via routing information it received from its other neighbors, Y will move away from P_y and make it unavailable. This will result in Y sending X routing information that makes P_x unavailable at X, causing X to downgrade to a less preferred alternative.

To detect route oscillation, much efforts in the networking community [4] have been on developing abstract models of the Internet routing policies, and deriving sufficient conditions — combinatorial structures such as circular dependency graphs — that characterize "conflicting" AS policies. In this extended abstract, rather than relying on domain expert to manually check sufficient policy conditions, we seek to develop an executable specification of the policy-driven routing system as a means to automate oscillation detection.

2 Preliminary Results with Answer Set Programming

We adopt answer set programming (ASP) [1, 2] as the knowledge representation for routing polices: ASP uses datalog-like rules and provides native support for negation with stable model semantics [3] that makes ASP particularly convenient for specifying the non-monotonic behavior of routing policies.

We outline our ASP formulation by an example in Figure 1: the example BGP system has 4 AS nodes with `0` being the destination of interest. The routing policy — path ranking — is represented by a vertical list next to each AS, with the highest ranked path at the top going down to the lowest ranked path at the bottom. To encode these AS policies, we employ two predicates `r` and `b`: `r(p)` states that the

Submitted to:
TEASE-LP 2020

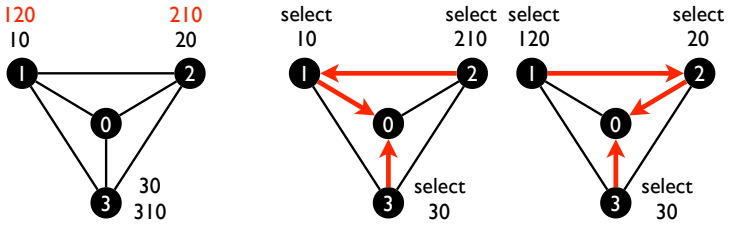

Figure 1: (left) Example (disagreeing) policy configuration. (right) Potential oscillation between two path selections (highlighted in red).

path `p` is permitted at some AS, `b(p)` says that `p` is selected as the best path, and `p` is a tuple that contains the list of nodes along the path. Intuitively, a path is selected as the best path only if no higher ranked paths are available. For example, the path ranking of `AS2` is captured by the following rules:

```
b((2,0)) :- r((2,0)), not r((2,1,0)).
b((2,1,0)) :- r((2,1,0)).
:- b((2,1,0)), b((2,0)).
```

Next we specify the interaction between these policies: A BGP system is a distributed system that computes the most preferred paths to `0`: ASes exchange routing information with their neighbors, announcing their current best path which may further affect the best paths available to the receiving neighbor — Upon receiving a route, an AS can discover a new path by appending itself to the received path. Note that, since BGP selects a single best path, a new announcement effectively withdraws a previously announced path. Continuing with `AS2`, it can learn routes either by the direct link to `0`, or by receiving route announcement `b((1,0))` from `AS1`:

```
r((2,0)).
r((2,1,0)) :- b((1,0)).
```

To fully specify the entire BGP system, we only need to repeat the same encoding for `AS1`,`AS3`. Running this ASP program yields two solutions shown in Figure 1 (right), representing two stable states of path selections where highlighted lines depict the selected best paths. Depending on the particular dynamics — the ordering or routing message exchanges, the BGP system may converge to either one or experience transient oscillation between the two. On the other hand, if we modify the policy configuration such that each AS prefers the counter-clockwise path of length 2 over its shorter direct path to `0` (e.g., `AS3` prefers the longer counter-clockwise path `310` over its direct path `30`), the BGP system has no stable path selection — the ASes will oscillate permanently, and the corresponding ASP program will produce 0 solution.

# of ASP solutions	oscillation analysis
0	permanent oscillation
>1	transient oscillation
$=1$	?

Table 1: Oscillation analysis with ASP solutions

More generally, as summarized in the first two roles in Table 1, the number of solutions our ASP formulation coincide with the number of stable path selections computed by the BGP system, thus providing an automated method to catch oscillations: If the ASP code yields zero solution, there does not

exist a stable path selection, hence the policies will result in permanent oscillation. On the other hand, when multiple solutions are produced, the policies can lead to temporary oscillation — the ASes will oscillate under certain ordering of routing information exchange, but will converge to one of the stable states under other cases.

3 Open questions

In the previous section, we illustrated how to detect oscillation. Can we also verify convergence? A BGP system converges if the ASes, regardless of the ordering of routing message being exchanged, will always select paths that form a stable path selection — no AS can find a better path based on routes learned from neighbors. In particular, as shown in the last row in Table 1, we ask: if the ASP formulation of a BGP system gives a unique solution, can we conclude that the BGP system is guaranteed to converge?

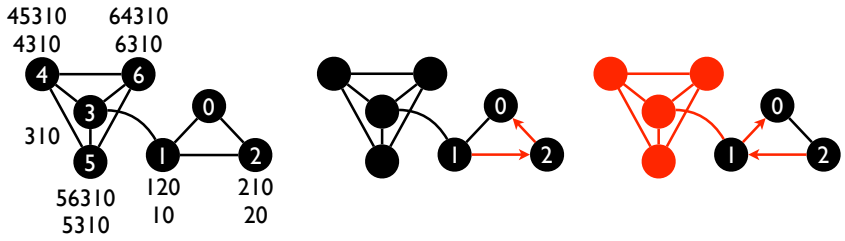

Figure 2: (left) A BGP system with a unique ASP solution. (middle) The stable path selected (highlighted). (right) Permanent oscillation (among nodes 4,5,6) can still occur if node 3 is "activated".

Unfortunately, the answer is no. Unique ASP solution does not guarantee convergence. There exists some BGP instance that has unique ASP solution but exhibits oscillation behavior. Consider such a counter-example [4] (Figure 2 left) with a unique ASP solution: ASes $4,5,6$ prefer a path through their counter-clockwise neighbors and the middle AS 3 to reach destination 0. AS 3 only accepts route 310 to 0. When 10 is selected by AS 1 (shown in Figure 2 right), 310 becomes available to 3 — 3 becomes "activated", which further triggers permanent oscillation among $4,5,6$. On the other hand, if AS 1 chooses 120 (middle of Figure 2), 310 becomes unavailable at 3, hence "deprecating" the oscillations among $4,5,6$. Indeed, the path selected here corresponds to the unique solution produced by the ASP formulation.

Based on the observation in the above, we conclude this extended abstract with open questions along two directions. First, the relation between our ASP formulation and convergence verification is fairly weak. When the ASP formulation of a BGP system has a unique solution, we can only rule out temporary oscillation between multiple stable states. Can we identify additional checkable conditions that distinguish between convergence and the mixed case of stable path selection in some scenarios but oscillation in the others? Second, our convergence analysis of BGP system also reveals a limitation of the ASP paradigm: The stable model semantics does not capture all interesting aspects of program execution. The answer set — stable model — only describes the final state of an ASP program after the execution "converges". It does not say how the program or whether the program will always reach that final state. Can we develop a stronger connection between stable model semantics and some sort of operational semantics?

References

[1] Gerhard Brewka, Thomas Eiter & Miroslaw Truszczynski (2011): *Answer set programming at a glance*. *Commun. ACM* 54(12), pp. 92–103, doi:10.1145/2043174.2043195. Available at `https://doi.org/10.1145/2043174.2043195`.

[2] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub & Marius Schneider (2011): *Potassco: The Potsdam answer set solving collection*. *Ai Communications* 24(2), pp. 107–124.

[3] Michael Gelfond & Vladimir Lifschitz (1988): *The Stable Model Semantics For Logic Programming*. MIT Press, pp. 1070–1080.

[4] Timothy G. Griffin & Gordon Wilfong (1999): *An Analysis of BGP Convergence Properties*. In: *SIGCOMM*.

[5] Y. Rekhter., T. Li. & S. Hares. (RFC 4271, 2006): *A Border Gateway Protocol 4 (BGP-4)*.