Internet Routing and Non-monotonic Reasoning

Anduo Wang, Zhijia Chen

Temple University, Philadelphia, USA,
adw,zhijia.chen@temple.edu

Abstract. Internet routing is the process of selecting paths across the Internet to connect the communicating hosts, it is unique in that path selection is *jointly* determined by a network of *independently* operated networks, known as domains or Autonomous Systems (ASes), that interconnect to form the Internet. In fact, the present routing infrastructure takes such an extreme position that it favors *local autonomy* — an AS can use arbitrary path preference to override the default shortest path policy, at the expense of potential *global oscillation* — a collection of AS preferences (policies) can fail to converge on a stable path, a path that is also the most preferred possible for every AS along the path. In this paper, we examine the route oscillation problem with non-monotonic reasoning. We observe that, in the absence of any AS specific policies, Internet routing degenerates into the monotonic computation of shortest path — a preferred (shorter) (super)path always extends another preferred (sub)path; But fully autonomous AS policies are *non-monotonic* — a path favored by one AS can be an extension of a less preferred path of a neighbor, to which an "upgrade" to a better path can cause this AS to downgrade to a less preferred path previously discarded. Based on this insight, we present an Answer Set Programming (ASP) formulation that allows for automatic oscillation detection. Our evaluation using the clingo ASP solver is promising: on realistic Internet topology and representative policies, clingo can detect anomalies within 35 seconds.

1 Introduction

The Internet is at once the world-wide information infrastructure that has revolutionized the computers and communications like nothing before. Behind its tremendous success as a means for information dissemination and a medium for collaboration and interaction between geographically distributed computers, is one common service — to provide end to end data paths that connect the communicating hosts. To select the much needed communication paths, the Internet relies on a process called routing.

Routing on the global Internet is unique in that, the Internet is a network of *independently* operated networks — known as domains or Autonomous Systems (ASes) that are driven by their own economic concerns. This makes routing a process jointly determined by all the ASes along the path. In particular, border gateway protocol (BGP) [9] is currently the only interdomain (i.e. across ASes) routing protocol that stitches together the Internet, it allows the ASes to set arbitrary path preference — local policies — to override the path decision based on shortest path metric — by default, without any AS-specific policies, BGP behaves like the distributed execution of the Dijkstra algorithm on a (AS-level) graph and always selects the path with fewest AS hops.

This extreme position taken by BGP — favoring local autonomy, allowing an AS to influence path selection based solely on its own concerns without any global coordination — has an important consequence. The BGP protocol suffers a global anomaly called route oscillation [7]: BGP system is not guaranteed to converge on a unique best path. There exists some AS policies when the BGP system is unable to agree upon a global policy-compliant path, and keeps oscillating between several sub-optimal ones. Much efforts in the networking community [5, 4, 6] have been on using abstract combinatorial models and derived structures — e.g., state transition machines, circular dependency graphs — to understand and detect such routing oscillation.

In this position paper, instead of relying on specialized combinatorial structures or expert-guided reasoning, we use non-monotonic reasoning as a means to understand, explain, and automatically detect policies that can lead to routing oscillation.

We observe that, in the absence of any AS policies, Internet routing degenerates into the *monotonic* computation of shortest path: For a particular destination, any shortest path to that destination, denoted by P_x, selected by AS x, must also extend another shortest path that has been selected by some neighbor AS y that is one hop closer to the destination. This has been captured by the Bellman-Ford equation that lies at the heart of the Dijkstra algorithm — $P_x = sp_y\{\text{x added to} P_y\}$, where sp is the aggregate function that returns the shortest path[1].

The Bellman-Ford equation implies that, path improvement to x's neighbor — y selects a better (thus shorter) path P'_y — can only benefit x, resulting in a P'_x no worse than before. Since the set of available paths are finite, all ASes will eventually converge on the path that is also their best possible choice. But independently set AS policies make path selection *non-monotonic*: The local policy of AS x may prefer a path (P_x) that extends a path (P_y) disliked by a neighbor y. As y learns and gets promoted to a better path — moving away from the less preferred P_y, it also unfortunately makes the much liked P_x unavailable at x, causing x to downgrade to a less preferred path.

Based on this observation, we developed a systematic encoding of policy configurations in Answer Set Programming (ASP) [2, 3] that allows for automated analysis. Our main result is that, the ASP solution to our policy formulation coincides with the paths selected by the policy-based routing system, thus providing a means to detect oscillation: The policies are guaranteed to converge on a single best path if the ASP solver gives a unique solution; The policies can lead to oscillation in some circumstances but converge in other cases if the ASP solver finds multiple solutions; The policies will result in permanent oscillation in any circumstances if the ASP solver cannot find any solution.

We also evaluated our ASP formulation using the clingo [1] solver, showing promising result: on realistic Internet topology with up to 10k nodes and 51k edges, clingo can correctly recognize routing oscillation problems for several representative policy configurations within 35 seconds; Our encoding also scales gracefully, the clingo searching time increases linearly with respect to the topology size (in terms of number of nodes under the same edge density).

[1] The original Bellman-Ford equation addresses computation of the cost of the shortest path. We presented a modified version for the shortest path — a list of nodes that constitute the path.

2 Internet Routing and Non-monotonic Reasoning

In this section, we provide the necessary background on BGP and a detailed analysis of route oscillation with non-monotonic reasoning. We begin with a brief review of BGP using an abstract model called the stable path problem (SPP) formalism. SPP offers a simple semantics for AS policies while abstracting away all nonessential details, such as implementation specifics of BGP.

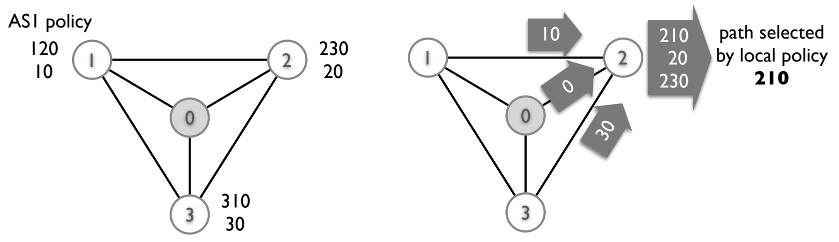

Fig. 1: (left) Example: the SPP model for AS policies. (right) Policy-based route selection.

An instance of SPP, $S = (G, P, R)$, as shown in Figure 1 (left), is an AS graph G — `0` is the pre-fixed destination of interests, together with the permitted paths P at each AS, and the path ranking (preference) functions R — 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. In Figure 1 (left), each AS prefers the counter-clockwise path of length 2 over all other paths to `0`. For example, `AS1` prefers the longer path `120` over its direct path `10`.

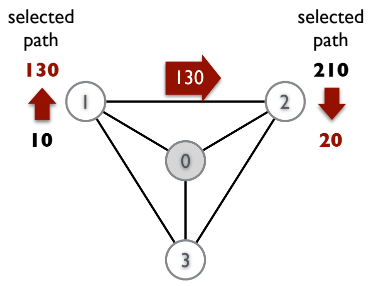

Fig. 2: Route selection with “disagreeing” AS policies is non-monotonic.

Given an SPP, BGP can be viewed as a distributed algorithm for finding the most preferred paths to `0`: ASes exchange routing announcements with their neighbors. Each announcement is the current best path chosen by the sending AS, and it indicates that the sending AS is willing to carry traffic destined to `0` from the receiving AS. That is, traffic flow in the opposite direction, from announcement receivers to senders. When multiple announcements are received by an AS, one single best path is selected by the AS’s local policy. In Figure 1 (right), `AS2`, upon receiving three announcements from its neighbors, uses its local policy to filter out the path `210` — which does not occur in the permitted path list, and picks the highest ranked path `210` as the selected one. It is also worth noting that, since an AS only selects one single best path, the announcement of a new (better) path implicitly retracts the previously exposed (older) path.

This path selection process was originally designed for using within an AS under one single administrative domain that often employs a single (global) ranking function — e.g., shortest path policy that always favors fewer hops. The global ranking ensures that all nodes will converge onto a stable path, a path that is not only globally optimal but also coincides with every node’s local best.

But fully autonomous policies of BGP can lead to non-monotonic behavior. By fully autonomous we mean an AS is free to prefer a path that extends a path disliked by a neighbor. The ASes do not need to take a consistent view of the paths based on some universally agreed criteria. For example, in Figure 1 (left), `210` is the most preferred path by `AS2`, but it extends the less preferred `10` from `AS1`. Note how this policy configuration diverges from the Bellman-Ford equation discussed in § 1. Such "disagreeing" policies lead to route oscillation shown in Figure 2: Suppose the current best path selected by `AS1` and `AS2` are `10` and `210`, respectively. When `AS1` learns a new path `30` from `AS3`, it will elevate `130` to be its best path and expose it to `AS2`, which also implicitly withdraws `10`, resulting in the retraction of `AS2`'s top choice `210`. Now, `AS2` needs to downgrade to the less preferred `10`. In fact, all three ASes will keep exchanging route announcements and exhibit similar oscillation: their choices of best path bounce back and forth as the subpaths they depends on are announced and withdrawn by the corresponding neighbors.

3 An ASP Formulation for Automatic Oscillation Detection

An advantage of our non-monotonic reasoning approach is that it lends itself to an ASP formulation [2, 3] that, readily recognizable by modern ASP solvers, allows for automatic analysis.

First, we present a straightforward ASP encoding of the AS policies defined in § 2, thanks to ASP's native support for negation and constraints. Our formulation involves two predicates `r` and `b`: `r(p)` states that the 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. To fully specify the policy-based route selection of an AS, we only need to describe how to generate the permitted and selected paths `r,b`.

A permitted path is either a direct path to the destination AS or an extension to paths received from a neighbor, which must also be that neighbor's current best path. Take `AS2` in Figure 1 (left) as an example, it can generate two permitted candidates, one direct path from itself to `0`, and an indirect one extending the best path of `AS1`.

```
% direct path as known fact(s)
r((2,0)).
% an indirect path generated by route announcements from a
    neighbor
r((2,1,0)) :- b((1,0)).
```

The best paths are determined by the path ranking function, the encoding of which is also straightforward. For every p_i in a list of paths $p_1 \cdots, p_n$ ranked from the most preferred to the least preferred, we only need a rule of the form `b(pi) :- r(pi), not r(p1-1), ..., not r(p1)`, meaning that, p_i can be promoted to be the best path only if it is available, while none of the more preferred paths are presented. Continuing with `AS2`, its ranking function is captured by the rules as follows.

```
% ranking function of the permitted paths at AS2
b((2,0)) :- r((2,0)), not r((2,1,0)).
b((2,1,0)) :- r((2,1,0)).
```

Finally, we need to ensure that only one best path is selected. To achieve this, we only need to make sure that, for any p_i from a list of permitted paths $p_1, \cdots, p_n$, the presence of p_i will prevent the derivation of $p_j, j \neq i$. In other words, for any $p_i, p_j (i \neq j)$, it follows that $b(p_i)$ and $b(p_j)$ cannot be simultaneously true. For the two paths — `210,20` — permitted at `AS2`, we have:

```
% only one permitted path can be selected as the best path
:- b((2,1,0)), b((2,0)).
```

The strength of this formulation is that, for a set of AS policies, the solution to this ASP encoding coincides with all the stable best paths that can be selected by the BGP system. For example, running the ASP program for Figure 1 yields no solution. More importantly, this correspondence gives a method for detecting BGP oscillation, stated as follows:

Let `L` *denote an ASP program that encodes a set of AS policies as described in this section, we can use the solution(s) to* `L` *to detect route oscillation as follows: (1)* 0 *solution implies permanent oscillation since the policies fail to give any stable path selection; (2)* 1 *solution indicates convergence since the policies will select a unique set of best paths under all circumstances — any ordering of route announcement exchanges; (3) multiple solutions mean possible oscillation: depending on the ordering of route announcement exchange, the policies can converge onto different set of best paths, but it is also possible that the BGP system may oscillate on other circumstances.*

4 Preliminary evaluation

In this section, we evaluate our ASP formulation with the clingo solver [1, 3], on various network topologies and policy configurations. Our preliminary result is promising. Clingo can correctly recognize oscillation problem in all cases within 1 minute. The solution time also scales well with the size of the network. All the experiments are performed on the macOS platform with 3.4 GHz Intel Core i5 processor and 16 GB 2400 MHz DDR4 RAM.

Setup We use the popular GT-ITM tool [8] to generate various Internet-like topologies — 2-level hierarchical graphs where the top level mimics a set of well connected network service providers, and the second layer represents smaller providers that access (and thus are customers of) the top providers. Compared to the actual Internet topology inferred from real-world network traffic, GT-ITM has the advantage of generating topologies of various sizes, allowing us to study the scalability of our ASP formulation. Specifically, we generate six topologies with 1000, 2000, 4000, 6000, 8000, 10000 nodes; 5092, 10304, 20766, 31595, 42880, 54954 edges; and a node-to-edge ratio of 0.196, 0.194, 0.192, 0.190, 0.187 and 0.182, respectively.

On every GT-ITM topology, we randomly pick a node as the destination of interest, and populate the rest of the nodes with all possible paths to that destination. The policy configurations we embed in the topology include three scenarios: In the "good" scenario, all paths are permitted and shortest path policy is used to rank the permitted paths. In the "disagree" scenario, we embed circular ranking (similar to Figure 1 (left)) between two randomly picked neighbors and use shortest path for the rest of the nodes. In the last "bad" scenario, we embed the circular ranking of Figure 1 (left). The bad configuration is known to exhibit permanent route oscillation while the disagree scenario only oscillates under certain ordering of route announcement exchanges.

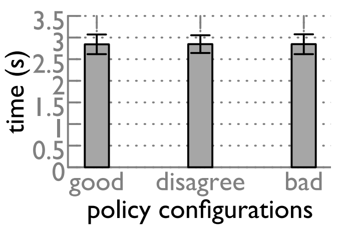

Fig. 3: Solution searching time for the good, disagree, and bad policies

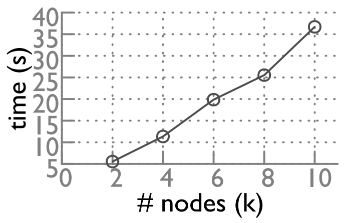

Fig. 4: ASP scales gracefully: the searching time increases linearly

Correctness and performance Using the analysis method described in § 3, clingo correctly recognizes the good, disagree, and bad policies on the graph with 1000 nodes and 5092 edges. The solution times for all three cases are depicted in Figure 3: we plotted the average running time (the box height) and the standard deviation (the bar length) for 10 runs. The solution time on different cases are very close — 2.843s, 2.850s and 2.846s for good, disagree and bad case, respectively.

Scalability Our formulation also scales gracefully. Figure 4 shows the searching time for the disagree case on various topology sizes: as the network size increases from 2000 nodes and 10304 edges to 10000 nodes and 54954 edges, clingo search time grows linearly from 6.578s to 34.786s. Each data point in the figure is averaged on 3 runs.

References

1. The clingo system. https://potassco.org/.
2. Brewka, G., Eiter, T., and Truszczyński, M. Answer set programming at a glance. *Commun. ACM 54*, 12 (Dec. 2011), 92–103.
3. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., and Schneider, M. Potassco: The potsdam answer set solving collection. *Ai Communications 24*, 2 (2011), 107–124.
4. Griffin, T. G., Shepherd, F. B., and Wilfong, G. The stable paths problem and interdomain routing. *IEEE Trans. on Networking 10* (2002), 232–243.
5. Griffin, T. G., and Wilfong, G. An analysis of BGP convergence properties. In *SIGCOMM* (1999).
6. Griffin, T. G., and Wilfong, G. A Safe Path Vector Protocol. In *INFOCOM* (2000).
7. McPherson, D., Gill, V., Walton, D., and Retana, A. Border Gateway Protocol (BGP) persistent route oscillation condition.
8. Modeling Topology of Large Internetworks. http://www.cc.gatech.edu/projects/gtitm/.
9. Rekhter., Y., Li., T., and Hares., S. A Border Gateway Protocol 4 (BGP-4).