Verifying Non-Deterministic Convergence on a Global Production WAN

Xing Fang[1,2], Fangdan Ye[2], Yifei Yuan[2], Zhongyu Guan[2], Duncheng She[2], Xiaobo Zhu[2], Qiao Xiang[1]
[1]Fujian Engineering Research Center of High-Performance Intelligent Computing Systems, School of Informatics, Xiamen University, [2]Alibaba Cloud

Abstract

This paper presents our experience deploying *TianYan* on Alibaba Cloud's global production WAN, which is, to the best of our knowledge, the first system for verifying non-deterministic convergence on a global production WAN. In daily operation, we rely on simulation-based configuration verifiers that assume a single converged data plane to ensure reliability and performance. However, non-deterministic convergence—where a configuration yields different converged data planes—undermines verification accuracy and has caused a production incident, motivating the need to analyze non-deterministic convergence itself. At scale, this is challenging because the analysis space grows exponentially with the number of routers. *TianYan* addresses this challenge with a key insight: by leveraging routing similarity among routers within the same group—a common fault-tolerance practice—it reduces exponential complexity from the number of routers to the number of groups, enabling efficient convergence analysis. Over a year of deployment, *TianYan* identified non-deterministic convergence in ~2% of all prefixes, exposed unnoticed design flaws, and improved simulation-based verification accuracy through integration. We share representative cases and evaluation results from our production WAN, distilling key operational lessons and practical guidelines for managing non-determinism at scale.

CCS Concepts

• **Networks** → **Network reliability**; **Network manageability**.

Keywords

Network Verification, BGP, Non-Deterministic Convergence, Stable Paths Problem

ACM Reference Format:
Xing Fang, Fangdan Ye, Yifei Yuan, Zhongyu Guan, Duncheng She, Xiaobo Zhu, and Qiao Xiang. 2026. Verifying Non-Deterministic Convergence on a Global Production WAN. In *ACM SIGCOMM 2026 Conference (SIGCOMM '26), August 17–21, 2026, Denver, CO, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3789240.3829180

1 Introduction

As a major cloud provider, Alibaba Cloud operates a global Wide Area Network (WAN) interconnecting thousands of routers and millions of prefixes across continents. This WAN is the backbone for critical services such as e-commerce, cloud computing, video collaboration, and emerging AI workloads.

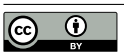
To ensure the reliability and performance of Alibaba Cloud's global WAN, we developed and deployed a series of verification tools [23, 24, 44, 45] to check the intent compliance of network configurations, which have successfully detected and prevented many errors before deployment. We adopt simulation-based configuration verification [6, 14, 44, 45] to simulate routing protocols and derive the resulting data plane for analysis, an approach widely adopted in practice for its scalability and expressiveness. In contrast, SMT-based verification [4, 12, 26, 30, 41] has limited scalability and is challenging to use and maintain in practice, while graph-based methods [1, 17] have restricted expressiveness (*e.g.*, cannot capture BGP's local preference).

Problem: simulation-based verifier is inaccurate under non-deterministic convergence. Non-deterministic convergence refers to the phenomenon that a network configuration yields different converged data planes (*e.g.*, BGP wedgies [18]). In such cases, a simulation-based verifier may produce a data plane different from the one realized in the network, which can lead to inaccurate verification results. A real-world incident in our WAN demonstrated this exact failure: a configuration deemed safe by our verifier led the production network to a state that violated a critical traffic load property, causing severe service degradation (§2.1).

Goal: prevent non-deterministic convergence from harming verification accuracy. We aim to enhance the accuracy of simulation-based verification by developing a tool that identifies prefixes that suffered from non-deterministic convergence and pinpoints their root causes. Once identified, accuracy can be improved by eliminating problematic configurations or applying targeted verification to these prefixes.

Challenge: scaling to a WAN with $O(10^3)$ routers and $O(10^6)$ prefixes. In our initial deployment, our simulation-based verifier Hoyan [44] incorporated a component based on the Stable Paths Problem (SPP) and SMT solving to identify non-deterministic convergence, which could handle networks with $O(10^2)$ routers and $O(10^4)$ prefixes. As our production WAN grew, this approach became impractical: constructing SPP instances requires enumerating an exponential number of permitted paths, and SMT solving is computationally expensive. We also explored other verification paradigms, including SMT-based [4, 12], graph-based [1, 17], and model-checking-based approaches [29], none of which can be applied because of limited scalability or expressiveness (§2.2).

Key insight: a general pruning method based on the router grouping and intra-group routing similarity. We make two observations in our global WAN. First, routers are organized into groups to improve resiliency. Second, routers within a group share



SIGCOMM '26, Denver, CO, USA

ACM ISBN 979-8-4007-2467-1/26/08
https://doi.org/10.1145/3789240.3829180

similar policies (*policy similarity*), enforced by using common configuration templates, with the goal of achieving ***routing similarity*** where they select next hops from the same group. We prove that if routing similarity holds for all converged data planes, the per-destination path search shrinks from $O((mn)!)$ to $O(n!)$, where m and n are the numbers of routers per group and groups, respectively. Here, $O((mn)!)$ represents an upper limit on the total number of simple paths to a fixed destination in a graph with mn nodes. In our WAN, m is on the order of 10, giving substantial search-space reduction.

This pruning is general in two respects. First, it can be used to reduce the path space in routing protocol convergence analysis, guide solver search in SMT-based approaches, and prune the state space in model-checking frameworks. Second, the routing pattern it exploits is a general design that arises in large-scale BGP networks due to operational needs such as reliability and automation, as observed in deployments including Meta's data centers [2]. Since policy similarity does not theoretically imply routing similarity, we discuss the validity of using routing similarity for pruning in §4.2.

Building on this insight, we design and deploy *TianYan*[1], the first reported verifier for checking non-deterministic convergence in a global WAN. *TianYan* adopts an SPP-based routing protocol analysis for two reasons: (1) our simulation-based verifiers can be easily adapted to generate SPP instances, enabling seamless integration; and (2) the well-established theory of SPP [8, 9, 19, 20] provides a solid foundation for characterizing routing behavior. As depicted in Figure 1, *TianYan* consists of three key designs (D1–D3).

D1: A simulation-based path generator to construct an SPP instance (§4). This generator takes the topology, router configurations, and input routes of our global WAN as input and enumerates paths through simulation. Leveraging the routing similarity assumption, we derive sufficient conditions for pruning paths that never appear in a converged data plane. In this way, the generator produces an SPP instance with far fewer paths than full enumeration.

D2: A convergence analyzer to identify prefixes suffering from non-deterministic convergence (§5). The analyzer evaluates each SPP instance in two phases. First, it uses a sufficient condition for a unique and stable solution to SPP that can be checked in polynomial time [9], to filter all prefixes that will have a unique convergence. Second, for the remaining prefixes, it employs an SMT-based method with completeness in detecting non-deterministic convergence.

D3: A root cause localizer to identify the source of non-deterministic convergence (§6). For each prefix identified as exhibiting non-deterministic convergence, we design a cyclic dependency checking algorithm to find the dispute wheels in the network that would lead to non-deterministic convergence, and return the related routers and paths.

Implementation (§7). Beyond detection, *TianYan* integrates with our simulation-based verifier. Since eliminating all problematic configurations to ensure unique convergence is impractical, *TianYan* keeps verification accuracy by aligning simulation with the observed data plane.

[1]We name our system *TianYan*, Chinese for "Heavenly Eye," after the mythological power used by the deity Erlang Shen to see through all variations.

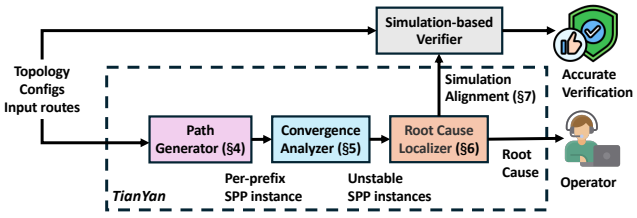


Figure 1: The architecture and workflow of *TianYan*.

Evaluation and Deployment (§8 and §9). Extensive evaluation shows that *TianYan* is highly efficient. *TianYan* has been deployed in our global WAN for over a year and has identified widespread non-determinism in ~2% of our 2 million prefixes, many of which lead simulation-based verifiers to converge to data planes that deviate from the real network without alignment. By analyzing hundreds of dispute wheels, *TianYan* pinpointed the operational root causes and exposed three recurring flawed policy patterns, two of which had not been noticed prior to deployment.

Lessons learned (§10). We share lessons learned from long-term deployment and operation, including the operational tradeoffs behind the blind spots of our simulation-based verifiers, why non-determinism is better prevented proactively, operational guidelines for reducing such risks in practice, and directions for future work.

Ethics. This work does not raise any ethical issues.

2 Background and Motivation

2.1 Why Are We Interested in Non-Deterministic Convergence?

Our production global WAN. As of January 2026, our WAN comprises $O(10^3)$ routers and $O(10^5)$ links, interconnecting $O(10^6)$ prefixes from our data centers and external ISP peers. It operates with distributed routing protocols (*e.g.*, BGP [27], IS-IS [28], and segment routing [13]). We designed and deployed a series of simulation-based verification tools to ensure reliability and performance in daily operations.

Incident: non-deterministic convergence leads to incorrect traffic load property verification. Figure 2 presents a simplified real incident from our WAN. The network consists of three autonomous systems (AS 100–300) and four device groups (A–D), each containing two routers, all running BGP. Groups A and B are core routers, while C and D are data center edge routers. The intent is that (1) all routers reach prefix P and (2) no link's traffic load exceeds its capacity threshold. The configuration of each router uses the default BGP settings except that (1) group A prefers iBGP routes learned from B over eBGP routes from D to utilize B's high-capacity backbone links, and (2) each router prefers next hops in other groups with the same index (*e.g.*, $A1$ prefers $B1$ over $B2$) to improve robustness against link failures. Before deployment, we used our simulation-based verifier and found it produced $DP1$, as group B first receives routes from A and prefers them for shorter AS-PATH. Using $DP1$ to verify the intents shows both reachability and performance satisfied, so we deem the configuration correct and deploy it.

The deployment is successful until routers in group D reboot, during which routers in groups A and C temporarily lose the routes

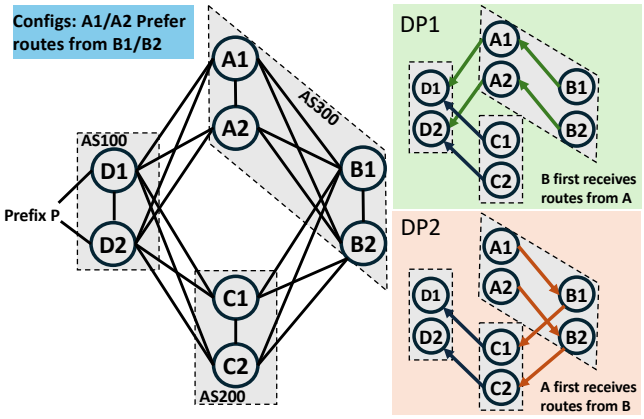


Figure 2: An example of multiple convergences.

learned from D. In the subsequent exchange of route updates, $A1$ and $A2$ select routes from $B1$ and $B2$, and, since they are configured to prefer iBGP over eBGP routes, they keep these routes even after receiving routes from D, leading the network to converge to $DP2$ in Figure 2. We omit the detailed message sequence for brevity. Although $DP2$ still satisfies the reachability intent, all traffic destined to P is now forced to detour via device group C. Given the very large traffic volume associated with this prefix, the detour causes the links $(C1, D1)$ and $(C2, D2)$ to exceed their capacity, thus violating the performance intent and resulting in a significant drop in throughput and an increase in latency.

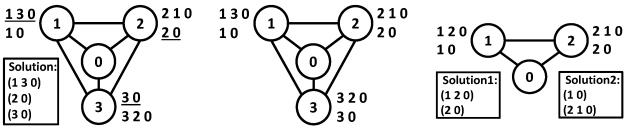
Root cause: DISAGREE in the real world. In the postmortem, we found this case exemplifies the DISAGREE instance of SPP (Figure 3c, §3.1), where neighboring routers (*i.e.*, group A and group B) prefer each other as the next hop to reach the destination results in the possibility of different converged data planes (*i.e.*, $DP1$ and $DP2$).

Takeaway: Non-deterministic convergence affects the accuracy of simulation-based verification tools, which could result in severe incidents in production networks.

2.2 Why Existing Tools Do Not Help?

The deprecation of our initial approach. Hoyan [44] is our production simulation-based verifier. In its early deployment, Hoyan incorporated an SPP-based model with an SMT solver to detect non-deterministic convergence, which worked for networks with $O(10^2)$ routers and $O(10^4)$ prefixes. However, as the network scale and the number of prefixes increased, the analysis became prohibitively expensive. Because non-deterministic convergence only affects a small fraction of prefixes at the time, this functionality was eventually disabled.

Our initial method has two inefficiencies. First, generating an SPP instance for each prefix requires enumerating all permitted paths and their rankings, whose complexity is exponential. This overhead also prevents us from leveraging existing efficient SPP-based methods [9, 19, 20, 43], which analyze a given SPP instance in polynomial time under sufficient (but not necessary) conditions for unique convergence. Second, even when an SPP instance can be constructed, the SMT-based analysis itself is highly inefficient, as shown in our evaluation (§8). Routing algebra [10, 21, 31–35]



(a) GOOD-GADGET (b) BAD-GADGET (c) DISAGREE

Figure 3: Examples of different SPP instances.

provides a mathematical framework based on algebraic composition and comparison of policy-induced path weights, but translating large-scale production configurations into precise algebraic models is non-trivial.

We also evaluated alternative verification paradigms, but none were suitable for our WAN. (1) SMT-based methods [4, 30, 41] encode routing behavior and intent as logical constraints and use solvers to exhaustively verify all stable data planes, which in principle can handle non-determinism. However, these methods cannot verify a single prefix within one hour in a simplified topology of $O(10^2)$ routers [12]. Despite recent work on improving performance [3, 12, 36, 38], these techniques either rely heavily on operational expertise or are limited to specific classes of intents. (2) Graph-based methods [1, 17] abstract configurations as weighted graphs and reason about properties across all data planes. Their applicability, however, is constrained by limited expressiveness, for example, the inability to encode BGP local preference. (3) Model-checking-based approaches [29] use explicit-state exploration to enumerate announcement sequences and derive all possible converged states. These methods suffer from exponential state-space growth and have only been demonstrated on BGP networks with up to 320 nodes.

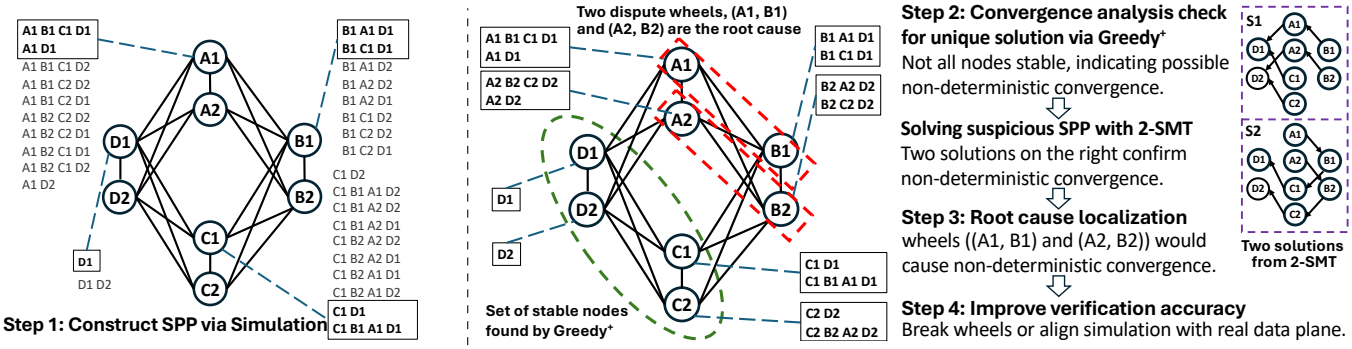
3 Overview of *TianYan*

3.1 Basic Idea

TianYan leverages SPP to analyze non-deterministic convergence, owing to its seamless integration with our verifiers and its solid routing protocol analysis foundation. Compared with Hoyan's initial SPP-based attempt (§2.2), *TianYan* makes SPP-based analysis practical at production scale by pruning the path space before SPP construction and filtering most prefixes before invoking SMT.

A primer on SPP. SPP abstracts routing protocols into a simple path vector protocol (SPVP), where each router receives paths from neighbors (*i.e.*, *permitted paths* determined by import and export policies), selects the best one based on its ranking function (*i.e.*, selection policies), and extends and sends the selected path to its neighbors in the form of a path vector. It studies, given the set of permitted paths and ranking function of each router, whether such a network admits a *stable path assignment* (also called a *solution*). A stable solution is reached when each router selects its highest-ranked path that is consistent—meaning its next-hop neighbor advertises the sub-path—leaving no incentive to change as all better options are unavailable.

Figure 3 illustrates canonical SPP instances. Each node is annotated with its permitted paths, ordered from high to low preference. (a) GOOD-GADGET has a unique solution (left box). First, node 3 selects its direct, highest-ranked path $[3, 0]$, which allows node 1 to select its preferred path, $[1, 3, 0]$. This choice in turn makes node



(a) Construct a simplified SPP by pruning unneeded permitted paths.

(b) Analyze constructed SPP instance to verify non-deterministic convergence, locate root cause, and improve accuracy of simulation-based verifiers.

Figure 4: Illustration on how *TianYan* verifies non-deterministic convergence for Figure 2.

2's most preferred path, $[2, 1, 0]$, inconsistent because node 1 is not advertising its required sub-path $[1, 0]$. Node 2 must therefore fall back to its next-best consistent path, $[2, 0]$. (b) BAD-GADGET has no solution. Nodes 1, 2, and 3 form a preference dependency cycle: each node prefers a path that goes through another in the trio (*e.g.*, node 1 via node 3, node 3 via node 2, and node 2 via node 1). Any node's selection will render another's preferred path inconsistent, resulting in perpetual oscillations (divergence). (c) DISAGREE-GADGET admits two solutions. Nodes 1 and 2 prefer transiting through each other, which creates two possible stable outcomes depending on timing. Unlike BAD-GADGET, it converges, but its final state is not unique, demonstrating non-deterministic convergence.

Relating SPP to verifying non-deterministic convergence. SPP theory provides a formal basis for verifying non-deterministic convergence. A network configuration exhibits non-deterministic convergence if and only if its corresponding SPP instance admits multiple stable solutions [19, 20]. However, the main hurdle is the scalability. Constructing an SPP instance requires enumerating all permitted paths, which is infeasible for a WAN of $O(10^3)$ routers and $O(10^5)$ links. One may think of resorting to the solvability conditions of SPP instances in the literature (*e.g.*, dispute-wheel free [20], monotonicity [21], Gao-Rexford [15], path graph [31], and Greedy$^+$ [9]). But checking them still needs enumerating all permitted paths or relying on exhaustive search via symbolic solving (*e.g.*, SMT), and neither is scalable.

Observations in our global WAN: router grouping and intra-group routing similarity. First, to improve the resiliency against router or link failures, we organize routers in groups based on their roles and locations. For example, in Figure 2, groups C and D are edge routers to data centers in different regions. Second, because of the role and location similarity of the routers in the same group, routers within the same group share similar policies, yielding ***routing similarity***: they prefer next hops inside the same group of routers. For example, in Figure 2, $A1$ and $A2$ both prefer routes from group B over group D. At first glance, these two observations would allow us to compress routers in the same group into an abstract one and hence simplify the verification [5]. However, routers within a group select different next-hop routers when multiple candidates exist, leading to distinct transfer functions and preventing safe compression.

Key insight: grouping and routing similarity allow substantial pruning of path exploration space. Leveraging these best practices in our WAN, we prove that *if routing similarity holds in all converged states*, two sufficient conditions enable substantial pruning of permitted paths that cannot appear in any stable assignment. (1) Among paths with the same *group path*—the sequence obtained by replacing each router with its group—at most one can appear. (2) No path containing multiple routers from the same group can appear. With these conditions, the number of possible permitted paths from any router to a destination shrinks from $O((mn)!)$ to $O(n!)$, where $m = O(10)$ is the number of routers per group and n is the number of groups. Thus we can construct an equivalent SPP instance with far fewer paths than full enumeration, speeding up convergence analysis.

3.2 Workflow

Figure 4 illustrates how *TianYan* leverages our key insight to verify non-deterministic convergence and integrates with our simulation-based verifier to improve accuracy.

Step 1: Construct a simplified equivalent SPP via simulation-based path generation. Given the router configurations, *TianYan* constructs the SPP instance for each prefix by simulating the routing protocols. In this simulation, and unlike standard protocol behavior, each router announces all received paths, not just its best one. To prevent the resulting path explosion, we enforce two pruning conditions derived from our routing similarity assumption. Specifically, a router only retains the highest-ranked path among those sharing the same group path, and prunes a path that traverses the same group more than once.

For example, router $B1$ receives eight paths (Figure 4a) corresponding to two group paths, $[B, A, D]$ and $[B, C, D]$. By applying our first pruning condition, $B1$ retains only the two highest-ranked paths: $[B1, A1, D1]$ and $[B1, C1, D1]$. Our second condition further prunes intra-group paths, such as the path $[D1, D2]$ at node $D1$. This combined pruning yields a highly simplified SPP instance. In our example, the final SPP contains merely 14 permitted paths (Figure 4b), representing a dramatic reduction from the original 116

paths (the total number of received paths from all nodes, including intra-group ones). Crucially, we prove this simplified instance is solution-equivalent to the original.

Step 2: Verify non-deterministic convergence via Greedy$^+$ and SMT. For each prefix, *TianYan* analyzes its simplified SPP instance using a two-pass approach. First, it uses the Greedy$^+$ algorithm [9], a polynomial-time sufficient condition for deterministic convergence, to determine whether the SPP instance has only one stable path assignment. In our example, the instance does not satisfy the Greedy$^+$ condition because not all nodes are stable, where a node is stable if it selects the same path across all potential stable assignments. As such, *TianYan* marks the instance as having a risk of non-deterministic convergence.

Second, *TianYan* encodes the suspicious instance as an SMT formula to solve the 2-SMT problem: determining whether two distinct assignments exist. The formula is built by conjoining per-path constraints that specify whether a path can appear in a stable assignment. When two solutions are found (Figure 4b), *TianYan* confirms non-deterministic convergence; otherwise the case is treated as a false positive.

Step 3: Localize routers and paths that lead to non-deterministic convergence. For each prefix with non-deterministic convergence, *TianYan* runs a cyclic-dependency checking algorithm to identify the dispute wheels—dependency loops formed by permitted paths preferred at different routers—that cause the instability. For example, in Figure 4b, $A1$ prefers $[A1, B1, C1, D1]$ from $B1$ while $B1$ prefers $[B1, A1, D1]$ from $A1$, forming a dependency loop that *TianYan* pinpoints as the root cause.

Step 4: Improve accuracy of simulation-based verifiers. The fundamental way is to eliminate the dispute wheel to ensure unique convergence, such as having $A1$ deny paths from group B. Alternatively, we align the simulation by forcing it to converge to the state observed in the real-world. However, this alignment is unsuitable for verifying configuration updates, as the state after update is unpredictable. Our pragmatic compromise is to report non-deterministic prefixes to operators for risk assessment.

4 Simulation-Based Path Generation

This section introduces a simulation-based algorithm that constructs an SPP instance for each destination prefix (§4.1), and shows how router grouping and intra-group routing similarity can be leveraged to prune unneeded permitted paths in practice while preserving correctness (§4.2).

4.1 Path Generation Framework

TianYan uses a simulation-based framework to generate an SPP instance for each prefix. Given router configurations, network topology, and input routes, it employs an iterative algorithm to simulate the routing protocols. The key difference between this framework and simulation-based verifiers [14, 44] is that at each step, each router advertises all its received paths, instead of only the selected one, thereby enumerating all permitted paths.

Algorithm 1 outlines the simulation process for a given prefix by iteratively propagating paths permitted by configurations. The origination path is initially added to the corresponding node and

Algorithm 1: Path generation for a given prefix.

```
Input: A: Set of announcement nodes for a given prefix
Output: P: Sorted permitted paths for all nodes
1  foreach v ∈ A do
2      P^v ← {(v)};
3      M.push((v));
4  while M is not empty do
5      P ← M.pop();
6      Let v_i be the first node in P;
7      /* Path pruning (§ 4.2)*/
8      if pruned(P) then
9          P^{v_i} ← P^{v_i} \ {P};
10         continue;
11     /* Advertise to neighbors, not restricted to the best */ foreach
         neighbor v_j of v_i do
12         if P passes export policy at (v_i, v_j) then
13             P' ← (v_j) ∪ P;
14             if P' passes import policy at (v_j, v_i) then
15                 P^{v_j} ← P^{v_j} ∪ {P'};
16                 M.push(P');
17 return P;
```

the path queue (lines 2-3). In each iteration, a path is extracted from the queue and advertised to all neighbors. If a path can be pruned (§4.2), it is removed from the result and not advertised to neighbors (lines 6-8). Otherwise, the path is processed through the export and import policies for each neighbor and, if permitted, extended and added to the neighbor's permitted path set (lines 9-15). The paths in each node are sorted by descending rank.

4.2 Pruning Permitted Paths

Best practices in our WAN: router grouping and intra-group routing similarity. To improve resiliency, our global WAN adopts two generic principles for large-scale networks: organizing routers in groups and configuring routers in the same group with similar policies with the goal of achieving *routing similarity*. Specifically, we configure routers in the same group to prefer the same next-hop group, but different next-hop routers within that group to enhance robustness against link failures. For example, in Figure 2, we configure routers in group A to both prefer routes from group B, but $A1$ prioritizes next-hop $B1$ while $A2$ prioritizes $B2$.

From assumptions based on best practices to pruning propositions. Taking the best practices as the underlying conditions allows *TianYan* to prune paths substantially. Specifically, (1) *intra-group routing similarity*—routers within the same group select next hops that are also in the same group for every converged state; and (2) *best-path only*—each router advertises only its best path. If the above assumptions hold, then the stable path assignment of the corresponding SPP instance satisfies the following propositions.

Proposition 1 ($\leq$ 1 Path Per Group Path). *For each router, among its permitted paths sharing the same group path (the sequence obtained by replacing each router on the path with its group), at most one path can appear in the stable path assignment.*

Proof Sketch: By induction on the group path length.

Base case. The length 1 path is uniquely the direct one, so the proposition holds trivially.

Induction step. Assume the proposition holds for all group paths of length n. Consider a group path $(g_{n+1}, g_n, \ldots, g_1)$. If a path with group path $(g_n, \ldots, g_1)$ is selected in a stable path assignment, then by the assumptions of routing similarity and best-path advertisement, together with the induction hypothesis, each router in g_n will select and advertise its own unique path in that assignment. As a result, the set of paths received by g_{n+1} from g_n is fixed, and the best path among them is uniquely determined.

Proposition 2 (No Path with Repeating Group). *For each router, no permitted path containing multiple routers from the same group can appear in a stable path assignment.*

Proof Sketch: The proof proceeds by contradiction. If such a path were selected, it would violate the intra-group routing similarity assumption.

Detailed proof is in Appendix A.

Leveraging the propositions to prune paths during simulation. At each step of the simulation, *TianYan* applies both propositions to prune paths that will not appear in any stable path assignment. For a router, among received paths sharing the same group path, it prunes all but the highest-ranked one. Paths containing multiple nodes from the same group are also pruned. Consequently, only the unpruned paths are considered for announcement. This pruning process is dynamic. If a newly received path is superior to the current best for a given group-path, the old path is pruned. This triggers its withdrawal and the announcement of the new one.

As a result of pruning, the number of paths that need to be considered for a given router and destination is reduced from $O((mn)!)$ to $O(n!)$, where m denotes the number of routers in a group and n the number of groups. Specifically, $O((mn)!)$ represents a worst-case bound on the number of paths in a network with mn routers. Under Propositions 1–2, path exploration collapses to group-level paths, which is equivalent to searching over a compressed graph with n nodes, yielding an $O(n!)$ upper bound. In our WAN, m is on the order of 10, so this pruning yields a simplified SPP instance with far fewer permitted paths. We now prove that this simplified instance is equivalent to the original, which enumerates all permitted paths in the network.

Theorem 1 (Equivalence of SPP Instances After Path Pruning). *The simplified SPP instance and the original one are equivalent in terms of their solutions. Both instances have the same set of stable path assignments.*

Proof. The pruning process removes only paths that cannot appear in any stable path assignment. Since no new paths are introduced, the simplified SPP instance has the same set of stable path assignments as the original instance, ensuring their equivalence in terms of solutions. □

Validity of the routing similarity assumption. The propositions above rely on the routing similarity assumption, requiring that it holds in all possible converged states. This assumption cannot be guaranteed in theory, as policy similarity does not necessarily imply routing similarity. For instance, two routers in the same group preferring each other can result in one selecting its peer as the next-hop, while the other chooses an external path. Nevertheless, we find this assumption practically sound for our WAN for several reasons. First, our simulation-based verifier has been used for years to enforce routing similarity in simulated data planes. Second, the non-deterministic scenarios detected by *TianYan* are all confirmed in the real network (§8), demonstrating the assumption's effectiveness. Finally, the pruning method based on this assumption is robust: it can still detect non-deterministic scenarios in networks where policy similarity holds but routing similarity does not (Appendix B).

5 Convergence Analysis

TianYan first employs an optimized polynomial-time Greedy$^+$ algorithm [9] to efficiently flag suspicious prefixes, and then uses a complete SMT-based method to precisely verify which of them are truly non-deterministic.

5.1 Convergence Analysis via Greedy$^+$

Rationale behind using Greedy$^+$. Polynomial-time algorithms [9, 20, 43] have been proposed to analyze the SPP instance and determine if it has a deterministic convergence. These algorithms provide a sufficient but not complete condition, ensuring all non-deterministic prefixes are detected at the cost of some potential false positives. Among these, *TianYan* leverages Greedy$^+$ [9] due to its strong balance of high efficiency and a low false positive rate. While more recent algorithms like UPEA [43] can further reduce false positives by performing additional path pruning, we found this unnecessary for our network where Greedy$^+$ already demonstrated sufficient accuracy. Thus, we avoid the computational overhead of this extra pruning step.

Greedy$^+$ overview. The Greedy$^+$ algorithm iteratively expands a set of stable nodes—those guaranteed to select a fixed best path. The process starts with nodes that originate the prefix and repeatedly identifies new stable nodes by verifying if their best path extends from an already stable neighbor, continuing until a fixed point is reached. If any nodes remain unstable, the prefix is flagged as suspicious for non-deterministic convergence. A key feature of Greedy$^+$ is the dynamic pruning of suboptimal paths from newly stabilized nodes, which progressively shrinks the SPP instance and thereby helps to discover new stable nodes. We refer the reader to the original paper for details [9].

Optimizing Greedy$^+$ for timely and comprehensive analysis. First, we introduce an incremental update mechanism that avoids redundant computation. *TianYan* only re-evaluates nodes whose stability determinants—such as the status of neighboring nodes—have changed, which prevents re-analyzing stable portions of the network in each iteration. Second, we accelerate the analysis by leveraging the per-node independence of the stability check to parallelize the search across multiple cores. Finally, to correctly handle anycast, we remove the assumption that announcing nodes are inherently stable. Instead, an anycast source is only considered stable if it deterministically prefers its own originated path over any received paths.

5.2 Convergence Analysis via SMT

SMT-based method becomes practical. Considering the NP-completeness of SPP, we directly encode it into an SMT problem and utilize the SMT solver (*i.e.*, Z3 [11]) to determine whether two

different solutions exist. Despite the high computational complexity, it becomes feasible for several reasons. First, instead of verifying millions of prefixes in our WAN, only the suspicious prefixes detected by the Greedy$^+$ algorithm are considered, a small subset of all prefixes. Second, the number of permitted paths is significantly pruned with the propositions. Finally, *TianYan* improves efficiency by grouping prefixes to avoid redundant computations.

Basic formulation for SPP. Two boolean variables $avail_p^v$ and $best_p^v$ are defined for each permitted path p in node v, indicating whether p is an available or the best at v. The following constraints are derived:

Available path constraint. A path is available if and only if it is an origin path or an extension of a neighbor's best path.

$$avail_p^{v_i} \iff \left(p = (v_i) \vee (p = (v_i, v_j) \circ p' \wedge best_{p'}^{v_j})\right) \tag{1}$$

Here, $p = (v_i, v_j) \circ p'$ represents a path starting at node v_i, traversing neighbor v_j, and extending with sub-path p'.

Best path constraint. A path is the best if and only if it has the highest rank among all available paths at the same node.

$$best_p^v \iff \left(avail_p^v \wedge \forall p' \in \mathcal{P}_v, avail_{p'}^v \implies rank(p) \geq rank(p')\right) \tag{2}$$

Here, $\mathcal{P}_v$ denotes the set of permitted paths at node v, and $rank(p)$ is the rank of path p in node v.

We construct an SMT formula from the conjunction of all permitted path constraints and query a solver for a stable assignment. If the formula is unsatisfiable (UNSAT), the prefix cannot converge. Otherwise (SAT), at least one stable state exists. To check for a second one, we add a constraint that blocks the most recently found solution and re-query the solver. This constraint is formulated as follows:

Previous solution negation constraint. The new solution must differ from the previously found converged state on at least one router.

$$\bigvee_{v \in V} \neg best_{\pi_s(v)}^v \tag{3}$$

Here, $\pi_s(v)$ represents the best path at node v in the previously found converged state. By adding this constraint and re-solving, a new solution confirms non-deterministic convergence, while unsatisfiability proves unique convergence.

By encoding the SPP into an SMT problem and leveraging the exhaustive search capability of the SMT solver, it guarantees that all solutions can be explored, ensuring no non-deterministic convergence is missed.

6 Root Cause Localization

The localization aims to identify dispute wheels, which are dependency loops formed by preferred paths at different routers. We associate dispute wheels with the root cause of non-deterministic convergence for two reasons. First, two different solutions imply the existence of a dispute wheel [20]. Second, the dispute wheel prevents nodes from being classified as stable by Greedy$^+$, hindering the SPP instance from achieving deterministic convergence.

Efficiently localizing dispute wheels in a limited node set. We leverage a key observation: a dispute wheel must exist among the unstable nodes that have a stable path but do not prefer it. This conclusion is generalized from the proof in [20], which shows that two different solutions imply a dispute wheel by constructing one between these nodes. Based on the stable path definition shown in §5.1, these nodes must be either the origin or neighbors of stable nodes.

Algorithm 2: Dispute Wheel Localization.

```
Input: Sorted Permitted paths P, node set V
Output: Dispute wheels C
1  P_stable ← FindStablePaths(P, V);
2  /* Find suspicious nodes */
3  V_s ← ∅;
4  foreach v ∈ V do
5      P^v_best ← Best path in P^v;
6      if P^v_best ∉ P_stable and ∃p ∈ P_v | p ∈ P_stable then
7          V_s ← V_s ∪ {v};
8  /* Construct preference dependency graph */
9  G_w ← ∅;
10 foreach v_i ∈ V_s do
11     P^{v_i}_best ← Best path in P̄^{v_i};
12     foreach v_j ∈ V_s do
13         if ∃p ∈ P^{v_j}_stable and P^{v_i}_best is extended from p then
14             G_w ← G_w ∪ {(v_i, v_j)};
15 /* Perform DFS to detect cycles */
16 C ← DFS(G_w);
17 return C;
```

Algorithm 2 presents our efficient process for localizing dispute wheels. It begins with the set of stable paths obtained from the output of the Greedy$^+$ algorithm (line 1) and then proceeds in three steps. First, it identifies a small set of suspicious nodes: those whose best path is unstable, despite having at least one stable path available (lines 2-7). Next, it constructs a dependency graph over these nodes, where a directed edge from v_i to v_j exists if v_i's best path extends a stable path of v_j (lines 8-14). Finally, it runs a Depth-First Search (DFS) [37] to find all cycles in this graph, which correspond to the dispute wheels (line 16).

Analyze the root cause of multiple convergences with dispute wheels. Only an even-numbered dispute wheel can cause non-deterministic convergence. Depending on route propagation timing, two stable states exist: either odd-numbered nodes select their preferred in-wheel paths, forcing even-numbered nodes to stable paths, or vice versa. In contrast, an odd-numbered wheel creates an unresolvable dependency, leading to persistent oscillation. To pinpoint the root cause, our analyzer examines any node in the wheel and compares the attributes of its preferred in-wheel path against those of its stable path. This comparison directly reveals the policy that establishes the circular dependency.

7 Implementation

This section details practical enhancements for scalability and expressiveness (§7.1), and integration with simulation-based verifiers to improve verification accuracy (§7.2).

7.1 Practical Enhancements

Optimize SMT analysis by prefix grouping. Even with path pruning, the SMT convergence analysis remains a bottleneck, as analyzing a single prefix takes seconds to minutes, making a check of $O(10^4)$ suspicious prefixes impractical. We observe that many prefixes share the same root cause of non-determinism: an identical

set of dispute wheels. Therefore, we first use our efficient localization to group prefixes by their root cause. This reduces the problem from $O(10^4)$ prefixes to just $O(10^2)$ unique groups. By verifying only one representative prefix per group, we significantly reduce the total number of SMT invocations.

Support ADD-PATH. The BGP ADD-PATH feature [39], enabled on a small subset of routers, requires a modification to our path pruning logic. This is because ADD-PATH allows exporting multiple paths, which invalidates our assumptions. Specifically, for these routers, among received paths sharing the same group path, both the best path and additional qualified paths allowed by ADD-PATH configuration are exported. Furthermore, paths traversing multiple nodes within the same group are also exported. A formal proof of correctness for modifications is left as future work.

7.2 Integration with Simulation-Based Verifiers

We present two complementary methods to improve the accuracy of our simulation-based verifiers.

Eliminating dispute wheels. The most fundamental way to ensure verification accuracy is to eliminate dispute wheels and thereby guarantee unique convergence, which can be accomplished once *TianYan* identifies non-deterministic prefixes and their root causes. In practice, operators employ two strategies. The first is a tactical patch: using a prefix-based filter to block a specific route advertisement forming the wheel. While effective, it fails to address underlying routing policy flaws and can compromise fault-tolerance. The second, more fundamental strategy is to redesign the routing policy. For example, the wheel in Figure 2 arises because group A unconditionally prefers all routes from group B to utilize high-capacity links, even if metrically inferior (*e.g.*, longer AS-PATH). A proper policy would apply this preference only to specific prefixes where such utilization is intended, thereby avoiding the dispute wheel.

Aligning simulation with real-world convergence. Since fundamentally eliminating all dispute wheels is impractical (§10), we address the resulting accuracy challenge by aligning our simulation-based verifier with reality. Specifically, for non-deterministic prefixes, we first obtain the actual routing state from the nodes involved in dispute wheels. Then, during the simulation, we steer the convergence to match this observed state by controlling the order of route advertisements on these nodes. For instance, if the network in Figure 2 has converged to $DP1$, we enforce this outcome by ensuring group B advertises its route only after receiving the update from group A. To address the limitation that Greedy$^+$ cannot find all dispute wheels, we iteratively align the simulation for identified wheels and re-run Greedy$^+$ to progressively uncover all sources of non-determinism.

Handling configuration updates. The alignment strategy is not applicable to configuration updates, especially when updates trigger re-convergence for non-deterministic prefixes. Since the final converged state is unpredictable and the number of possible outcomes grows exponentially with dispute wheels, *TianYan* does not yet enumerate all possible converged data planes. Instead, for prefixes affected by an update, *TianYan* detects and localizes the involved dispute wheels, and reports them to operators for risk assessment before deployment. Resolving these wheels still requires human involvement today: as discussed in § 10, eliminating a dispute wheel often requires network-wide policy changes that may affect many prefixes, reduce fault tolerance, or disrupt legacy design intent. Assessing these tradeoffs demands deep operational expertise, since operators must weigh the benefit of deterministic convergence against the risk of the fix. Automating this resolution process is left for future work.

8 Performance Evaluation

In this section, we evaluate *TianYan* on our production WAN with $O(10^3)$ routers and $O(10^6)$ prefixes.

Experiment setup. All evaluations are conducted on a single server with Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz (96 logical cores), 754GB RAM, and 1T SSD. While *TianYan* can be easily deployed in a distributed manner, our evaluation shows that a single server is sufficient.

Baseline. We compare *TianYan* with routing-protocol convergence analysis techniques [8–10, 15, 19, 20, 40], which rely on enumerating the entire path space as a prerequisite. Instead, we exclude SMT- and model-checking-based verifiers due to their inability to scale to our production WAN. For instance, existing SMT verifiers [4, 30, 41] exhibit prohibitive latency, while recent acceleration techniques [3, 12, 36, 38] require substantial domain-specific knowledge. Similarly, model-checkers like Plankton [29] face inherent state-explosion barriers and are not open-sourced. Although we evaluate *TianYan* against a specific baseline, its key insight—pruning the exploration space via intra-group routing similarity—is a general principle that can also accelerate SMT and model-checking approaches.

Overall runtime of *TianYan*. Figure 8 shows the overall runtime breakdown of each stage in *TianYan*. The total runtime is about 14 hours, which is acceptable for daily analysis and can be further reduced by parallelizing different prefix sets across machines. Path generation and convergence analysis via Greedy$^+$ dominate the cost because every prefix must go through these steps. In contrast, convergence analysis via SMT and root cause localization require little time, benefiting from prefix grouping and efficient algorithms.

Scalable SPP Generation via Path Pruning. Figures 5 and 6 highlight a critical limitation of exhaustive path enumeration (purple curves). This method is only effective for the subset of prefixes where routing policies rigorously limit the scope of route propagation (*e.g.*, ISP routes cannot propagate into core routers and edge routers). We find that around half of all non-deterministic prefixes reside within this intractable set, meaning they would be completely missed by such an approach. In contrast, our path pruning technique (blue curves) successfully analyzes the vast majority of prefixes in under one second with modest overhead compared to the baseline BGP behavior (red curve), making it practical for achieving comprehensive non-deterministic convergence analysis.

Efficient convergence analysis and root cause localization. Figure 7 depicts the performance of three analysis methods (Greedy$^+$, SMT, and root cause localization) on the reduced SPP instances. A comparison with full enumeration is omitted for brevity, as it would invariably show worse performance due to their larger size. Greedy$^+$ analysis and root cause localization are highly efficient, completing in under 100 milliseconds for the majority of prefixes. In contrast, the SMT-based analysis is computationally expensive,

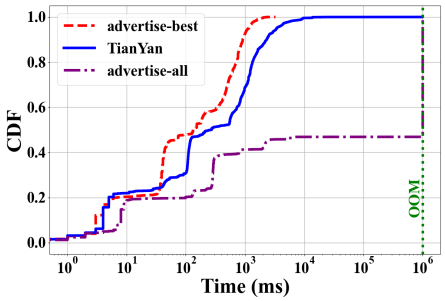


Figure 5: SPP instance generation time per prefix.

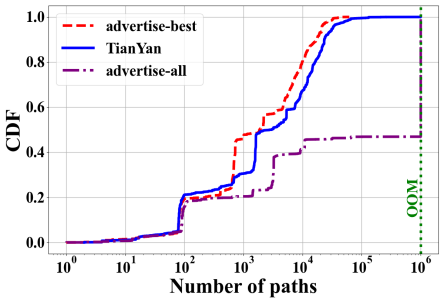


Figure 6: Number of paths in SPP instances per prefix.

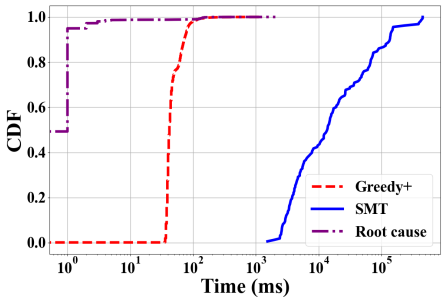


Figure 7: Analysis time of different methods per prefix.

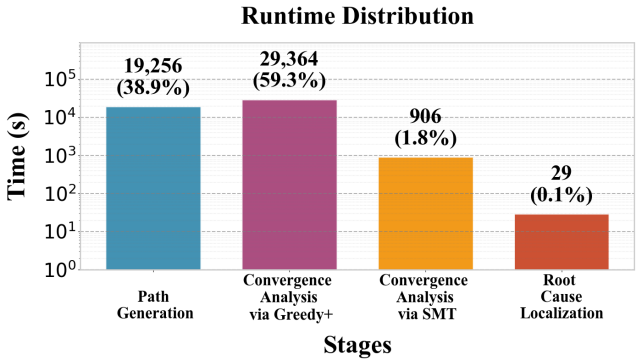


Figure 8: Overall runtime breakdown.

# Total Prefixes	# Non-deterministic Prefixes	# Root Causes Types
2×10^6	4×10^4	1×10^2

Table 1: Non-determinism detected by *TianYan*. Each root cause type is defined by a set of dispute wheels.

requiring seconds to minutes per prefix. Our prefix grouping optimization (§ 7.1) addresses this bottleneck by reducing the number of SMT analyses from one for each of the $O(10^4)$ prefixes to one for each of the $O(10^2)$ root-cause groups.

***TianYan* accurately identifies non-deterministic convergence.** All dispute wheels reported by *TianYan* have been validated on our production WAN. Our SMT-based verifier pruned false positives from Greedy$^+$, identifying three instances that converge uniquely despite not being Greedy-solvable. The first involved ADD-PATH-enabled routers with preference dependencies forming a routing loop carrying no service traffic. The second involved two VPN instances redistributing backup paths to each other. The third involved a subtle policy interaction pattern detailed in Appendix C.

9 Deployment

TianYan has been deployed in our production WAN for over a year and runs daily analyses to identify convergence risks and improve verification accuracy through alignment.

9.1 Deployment Overview

Widespread non-deterministic convergence. Non-deterministic convergence is not a corner case in our WAN. Our analysis (Table 1) shows that around 40,000[2] (~2%) of two million prefixes exhibit non-deterministic convergence, and most of these prefixes carry business traffic.

Impact on simulation accuracy. In our daily verification, approximately 30,000 prefixes would converge to incorrect outcomes without *TianYan*'s alignment, demonstrating the importance of addressing non-determinism for accuracy.

Diverse root causes. We identify ~100 distinct dispute wheel sets across different network regions and router roles. Despite their diversity, many stem from the same underlying configuration and design flaws, which we detail in §9.3.

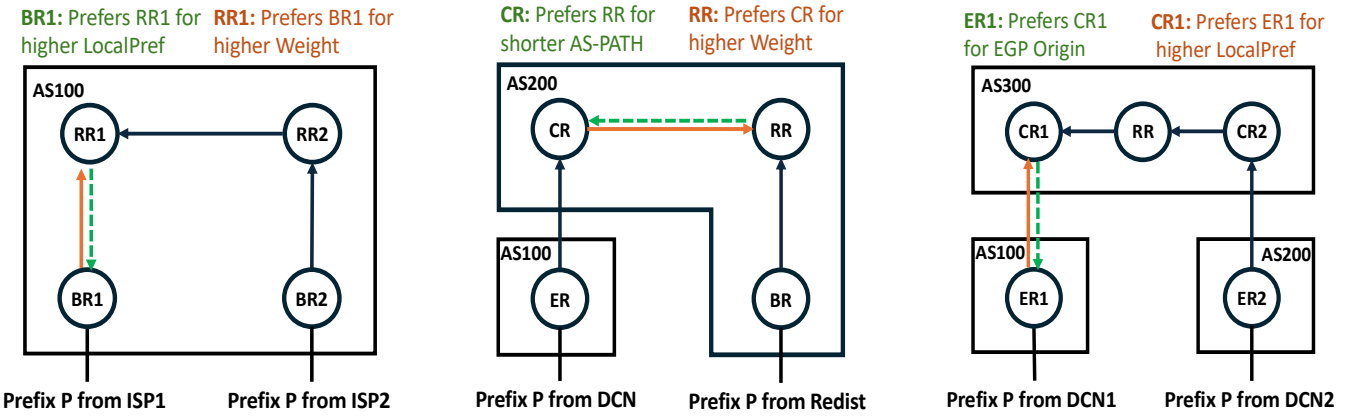
9.2 Real-World Cases

Overview of our WAN Topology. Edge Routers (ER) import routes from the DCN to facilitate communication between the DCN and external networks. Border Routers (BR) are placed at our WAN boundary to ensure connectivity with other ISPs. Core Routers (CR) form the backbone of the network, responsible for high-speed data forwarding across the WAN. Route Reflectors (RR) help distribute routing information efficiently by reducing the number of connections.

Case 1: Multiple convergences between RR and BR. Figure 9a shows a non-deterministic convergence where a dispute wheel forms between BR1 and RR1. RR1 prefers routes from BR1 for high weight, while BR1 prefers routes from RR1 for high local preference. Two converged states lead to BR1 selecting different ISPs for egress. Confirmed by operators, this non-determinism stems from a flawed design: RR1 assigned weights to routes from other neighbors but omitted RR2, causing unconditional de-preference of RR2's routes despite superior metrics. Moreover, the high local preference set by BR2 was intended to attract regional traffic, not influence BR1's egress decision. The converged state where BR1 selects ISP2 as egress violates this intent and has been observed in production.

Case 2: Multiple convergences between CR and RR. Figure 9b illustrates a dispute wheel between CR and RR. RR prefers routes from CR due to a high weight, while CR in turn favors the route from RR for its zero-length AS-PATH, a result of redistribution at BR. This

[2]Numbers are approximated for confidentiality reasons.



(a) An example of multiple convergences between RR and BR.

(b) An example of multiple convergences between RR and CR.

(c) An example of multiple convergences between ER and CR.

Figure 9: Typical scenarios of multiple convergences found by *TianYan* in our production WAN.

cycle creates two stable data planes, causing CR to unpredictably select either ER or BR as its egress. This non-determinism exposes two flawed designs: (1) RR unconditionally prefers routes from CR, similar to the motivating example in Figure 2, and (2) BR's redistributed route, a black hole route, propagates to CR despite being intended solely for external ISPs. Consequently, CR selecting BR as egress may cause traffic loss.

Case 3: Multiple convergences between CR and ER. Figure 9c illustrates a dispute wheel between CR1 and ER1. CR1 is configured to prefer ER1 with a high local preference as its nearest egress for anycast prefixes, while ER1 in turn prefers the route from CR1 due to a superior Origin attribute. This attribute difference stems from distinct redistribution methods: the route from DCN2 is marked as IGP, while the route from DCN1 is marked as Incomplete. This cycle creates two stable data planes, where CR1 selecting ER2 violates the traffic engineering intent. Unlike the previous two cases driven by unconditional preferences, this non-determinism is more subtle and difficult to notice without *TianYan*.

Beyond adjacent dispute wheels. The above cases focus on dispute wheels between adjacent router roles, since they dominate the non-deterministic prefixes observed in our WAN. However, *TianYan* is not limited to adjacent routers: its localization algorithm builds a preference dependency graph and detects cycles among suspicious nodes (§6). We also observed non-adjacent cases in deployment, where two routers form a preference dependency through one or more transit routers that only propagate the selected paths. Such cases are less common because dispute wheels usually arise between routers that explicitly apply preference changes, and these routers are generally adjacent in our WAN.

9.3 Root Cause Analysis

Table 2 summarizes the pairs of tie-breaking attributes forming the two-node dispute wheels, along with the percentage of impacted prefixes. Local preference and weight are grouped, and pairs affecting <0.1% of prefixes are excluded.

Independent decisions and anycast prefixes easily lead to multiple convergences. As shown in Table 2, local preference and weight dominate the attributes driving these dispute wheels, suggesting that operators frequently make localized routing decisions to engineer traffic paths. Besides, we find that anycast prefixes constitute a large portion of all non-deterministic prefixes. By their nature, anycast routes are announced from multiple locations, inherently increasing the complexity of route selection and the likelihood of preference dependency loop.

Attribute 1	Attribute 2	Impacted (%)
Local Pref./Weight	Local Pref./Weight	61.6%
Local Pref./Weight	AS-PATH Len	35.0%
Local Pref./Weight	MED	2.8%
Local Pref./Weight	Origin.	0.5%

Table 2: Dispute wheel attributes.

Flawed policy patterns behind non-deterministic convergence. From the identified ~100 dispute wheel sets, we summarize three recurring flawed routing policy patterns. (1) Unconditional de-preference caused by missing weight assignments (*e.g.*, Case 1). Operators confirmed this stemmed from unintended misconfiguration: weights were missing for certain neighbors, causing de-preference of their routes despite having superior metrics such as short AS-PATH or high local preference. (2) Unconditional preference (*e.g.*, Case 2), where routers indiscriminately prefer routes from a neighboring group due to higher-capacity links or closer egress points. (3) Mutual preference, which emerges from configuration evolution. Group X was historically configured to prefer routes from group Y. Later, to attract traffic to group X as an egress, operators modified group Y to prefer routes from group X, inadvertently creating mutual preference.

Notably, while operators were already aware of unconditional preference prior to deploying *TianYan*, the other two flawed designs were unexpected and only surfaced through our analysis. In addition, our deployment also uncovered several routing issues that are detectable in practice but not captured by our simulation-based

verifier. We discuss these blind spots, along with guidelines learned from deployment for preventing non-determinism, in §10.

10 Lessons-learned

Widespread non-determinism invalidates the core assumption of simulation-based verification. Simulation and emulation-based verifiers are widely favored in production networks for their high scalability and usability [6, 14, 16, 25, 45]. However, they rely on the foundational assumption that the control plane yields a deterministic outcome. Our deployment reveals the widespread and impactful reality of non-deterministic convergence, directly contradicting this premise. Consequently, non-determinism cannot be simply seen as a theoretical corner case and ignored, especially in production WANs.

Our simulation-based verifier's blind spot: undetected intent violations. In our production WAN, many prefixes affected by non-deterministic convergence have in fact converged to unintended data planes, leading to risks such as traffic detours and black holes. Despite being expressive enough to detect such violations, our simulation-based verifier failed in practice. This failure manifests in two blind spots. First, intent specifications are incomplete at scale: when the verifier correctly simulates an unintended converged state, no alarm is raised if the corresponding intent is absent. Second, accuracy checking is also incomplete: when the simulator converges to an intent-compliant state, discrepancies between the simulated and live routing states can remain undetected under our coarse-grained, traffic-based accuracy checking. Both blind spots stem from a fundamental tradeoff between completeness and practical operability in production networks. Constructing and maintaining exhaustive intents and fine-grained, prefix-level accuracy checks is operationally prohibitive at WAN scale and therefore intentionally avoided. Even if these blind spots were fixed, latent non-determinism would remain undetectable whenever both simulation and the live network coincidentally converge to a correct state, underscoring the need to verify stability itself.

Non-determinism should be avoided proactively. Most non-deterministic prefixes stem from three recurring design flaws we identified in §9.3. Contrary to the expectation that problematic configurations should be fixed immediately, operators often chose not to do so. The reason is that fixing design flaws requires network-wide changes (*e.g.*, re-configuring all route reflectors) affecting numerous prefixes, which are too risky in practice. Since the primary harm from non-determinism is traffic detours, which latency-insensitive traffic can tolerate, the operational risk of fixing outweighs the benefit. Thus, operators selectively enforce deterministic behavior only for business-critical prefixes, while tolerating flawed designs for the rest. This pragmatic choice is further reinforced by ongoing migrations to new network architectures, which reduce the incentive to fix legacy systems. This illustrates a key lesson: non-determinism must be prevented proactively, as it tends to persist once introduced.

Guidelines for preventing non-determinism. We summarize three guidelines to help operators proactively prevent non-determinism. (1) Use fine-grained routing policies. A major source of dispute wheels is indiscriminate preference applied to neighboring groups. Operators should instead use finer-grained controls, such as prefix- or community-based preferences, to encode when a preference is intended, thereby avoiding unintended preference loops. (2) Maintain an explicit intent database. Design intent is often lost during configuration evolution in large networks, for example due to personnel changes. Beyond recording high-level reachability or traffic-engineering goals, the intent database should also capture per-node path preference relationships, which can help identify potential preference loops before deployment. (3) Bridge the gap between routing design and implementation. Even when the design avoids non-determinism, incomplete or inconsistent configuration implementation may reintroduce dispute wheels. For example, as shown in Case 1, missing preference assignments caused routes that were considered in the design to be unintentionally de-preferred. While simulation-based verification helps detect such mismatches, ensuring complete and correct network intent remains challenging [42], suggesting the need for effective intent synthesis.

Toward faster and more comprehensive non-deterministic verification. Our deployment of *TianYan* highlighted two directions suggested by operational use. First, verification latency fundamentally limits usability during configuration updates. While hour-level runtimes suffice for periodic checks, operators emphasized that update verification requires minute-level feedback that can both ensure no new instabilities are introduced and enumerate all possible converged data planes for risk assessment before deployment. This creates a strong operational demand for incremental verification capabilities, where reusing prior results is essential for meeting these tight latency targets. Second, verification coverage is constrained by input-dependent analysis. Because *TianYan* reasons about convergence based on existing route inputs, latent instabilities triggered only under specific routing inputs can remain invisible. This necessitates exploring a wider range of inputs, implying a need for capabilities like test case generation or symbolic analysis.

11 Limitations

Correctness depends on routing similarity. *TianYan*'s pruning is sound only under the routing similarity assumption: routers in the same group select next hops from the same group in all stable states. In our WAN, we aim to realize this property through configuration similarity and validate group-level next-hop behavior using our daily simulation-based verifier. However, configuration similarity does not theoretically imply routing similarity, as further discussed in Appendix B. *TianYan* empirically validates routing similarity on the simulated converged data plane, but does not formally prove that the assumption holds across all possible stable states. Thus, *TianYan* trades absolute correctness for the scalability needed in production-scale analysis: if the assumption is violated, the pruned SPP instance may no longer be equivalent to the original one, causing *TianYan* to miss possible converged data planes. Nevertheless, Appendix B and our deployment results suggest that this tradeoff is practical for our production WAN.

Explosion of group paths. While *TianYan* is effective for most prefixes, it struggles with thousands where an explosion of group paths makes their SPP instance too large to construct. These are predominantly management routes that do not carry business traffic and are intentionally advertised network-wide. Given their low

operational impact, we currently exclude them from verification, leaving the handling of such worst-case scalability scenarios to future work.

Time-dependent and inter-protocol non-determinism. By design, *TianYan* cannot handle non-determinism rooted in time-dependent policies (*e.g.*, age-based route ranking [22]) or inter-protocol interactions (*e.g.*, non-determinism involving route redistribution [7]). Verifying these behaviors would require a more expressive model capable of capturing both temporal dynamics and multi-protocol logic.

Route aggregation. Route aggregation is currently unsupported due to its sparse use in our WAN. Supporting it would require non-trivial extensions to our model to capture cross-prefix dependencies.

Resilience to link failures. While *TianYan* can handle fixed link failures by analyzing the resulting topology, reasoning about arbitrary k-failures requires a more extensive framework. For instance, our convergence analysis could be extended to detect dispute wheels among all potential paths that become active under failures, not just the best ones. A deeper challenge is the validity of our routing-similarity assumption under failures. While we hypothesize the assumption largely holds for small failures (e.g., $k < 3$) due to our WAN's high redundancy, formally verifying this requires a comprehensive framework, which we leave for future work.

12 Conclusion

This paper presents the deployment of TianYan, a system for detecting non-deterministic convergence in Alibaba Cloud's global production WAN with $O(10^3)$ routers and $O(10^6)$ prefixes. Leveraging router grouping and intra-group routing similarity, TianYan enables efficient convergence analysis. Over one year of deployment, TianYan has identified widespread non-deterministic scenarios, uncovering underlying design flaws and improving the accuracy of our simulation-based verifier. We also share key lessons learned from deployment.

Acknowledgments

We thank our shepherd and the SIGCOMM reviewers for their insightful comments. We especially thank Ennan Zhai for his helpful feedback and support. We are also grateful to Ruzica Piskac, Ricardo Santos, and João Luís Sobrinho for their valuable feedback and discussions during this project. This work is funded by the National Key R&D Program of China (Grant No. 2024YFB2906900), the National Natural Science Foundation of China (Grant Nos. 62532010 and 62572408), the Fujian Provincial Science and Technology Project (Grant No. 2025H6021), and Alibaba Cloud through the Alibaba Research Intern Program. Qiao Xiang is the corresponding author.

References

[1] Anubhavnidhi Abhashkumar, Aaron Gember-Jacobson, and Aditya Akella. 2020. Tiramisu: fast multilayer network verification. In *USENIX NSDI '20*. USENIX Association, USA, 201–220.

[2] Anubhavnidhi Abhashkumar, Kausik Subramanian, Alexey Andreyev, Hyojeong Kim, Nanda Kishore Salem, Jingyi Yang, Petr Lapukhov, Aditya Akella, and Hongyi Zeng. 2021. Running BGP in Data Centers at Scale. In *USENIX NSDI '21*. USENIX Association, USA, 65–81. https://www.usenix.org/conference/nsdi21/presentation/abhashkumar

[3] Timothy Alberdingk Thijm, Ryan Beckett, Aarti Gupta, and David Walker. 2023. Modular Control Plane Verification via Temporal Invariants. *Proc. ACM Program. Lang.* 7, PLDI (June 2023), 50–75. doi:10.1145/3591222

[4] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2017. A General Approach to Network Configuration Verification. In *ACM SIGCOMM '17*. ACM, New York, NY, USA, 155–168. doi:10.1145/3098822.3098834

[5] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2018. Control plane compression. In *ACM SIGCOMM '18*. ACM, New York, NY, USA, 476–489. doi:10.1145/3230543.3230583

[6] Matt Brown, Ari Fogel, Daniel Halperin, Victor Heorhiadi, Ratul Mahajan, and Todd Millstein. 2023. Lessons from the evolution of the Batfish configuration analysis tool. In *ACM SIGCOMM '23*. ACM, New York, NY, USA, 122–135. doi:10.1145/3603269.3604866

[7] Enke Chen and Jenny Yuan. 2021. *Deterministic Route Redistribution into BGP*. Internet-Draft draft-chen-bgp-redist-01. Internet Engineering Task Force. Work in Progress. https://datatracker.ietf.org/doc/draft-chen-bgp-redist/01/

[8] Yichao Cheng, Ning Luo, Jingxuan Zhang, Timos Antonopoulos, Ruzica Piskac, and Qiao Xiang. 2021. Looking for the Maximum Independent Set: A New Perspective on the Stable Path Problem. In *IEEE INFOCOM '21*. IEEE, USA, 1–10. doi:10.1109/INFOCOM42981.2021.9488682

[9] Luca Cittadini, Massimo Rimondini, Stefano Vissicchio, Matteo Corea, and Giuseppe Di Battista. 2011. From Theory to Practice: Efficiently Checking BGP Configurations for Guaranteed Convergence. *IEEE Trans. Netw. Serv. Manag.* 8, 4 (2011), 387–400. doi:10.1109/TNSM.2011.110311.100109

[10] Matthew L. Daggitt, Alexander J. T. Gurney, and Timothy G. Griffin. 2018. Asynchronous convergence of policy-rich distributed bellman-ford routing protocols. In *ACM SIGCOMM '18*. ACM, New York, NY, USA, 103–116. doi:10.1145/3230543.3230561

[11] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, Berlin, Heidelberg, 337–340.

[12] Xing Fang, Feiyan Ding, Bang Huang, Ziyi Wang, Gao Han, Rulan Yang, Lizhao You, Qiao Xiang, Linghe Kong, Yutong Liu, and Jiwu Shu. 2024. Network Can Help Check Itself: Accelerating SMT-based Network Configuration Verification Using Network Domain Knowledge. In *IEEE INFOCOM '24*. IEEE, USA, 2119–2128. doi:10.1109/INFOCOM52122.2024.10621215

[13] Clarence Filsfils, Stefano Previdi, Les Ginsberg, Bruno Decraene, Stephane Litkowski, and Rob Shakir. 2018. Segment Routing Architecture. RFC 8402. doi:10.17487/RFC8402

[14] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. 2015. A General Approach to Network Configuration Analysis. In *USENIX NSDI '15*. USENIX Association, Oakland, CA, 469–483. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/fogel

[15] Lixin Gao and Jennifer Rexford. 2001. Stable Internet routing without global coordination. *IEEE/ACM Trans. Netw.* 9, 6 (2001), 681–692. doi:10.1109/90.974523

[16] Zhaoyu Gao, Anubhavnidhi Abhashkumar, Zhen Sun, Weirong Jiang, and Yi Wang. 2024. Crescent: Emulating Heterogeneous Production Network at Scale. In *USENIX NSDI '24*. USENIX Association, Santa Clara, CA, 1045–1062. https://www.usenix.org/conference/nsdi24/presentation/gao-zhaoyu

[17] Aaron Gember-Jacobson, Raajay Viswanathan, Aditya Akella, and Ratul Mahajan. 2016. Fast Control Plane Analysis Using an Abstract Representation. In *ACM SIGCOMM '16*. ACM, New York, NY, USA, 300–313. doi:10.1145/2934872.2934876

[18] Tim A. Griffin and Geoff Huston. 2005. BGP Wedgies. RFC 4264. doi:10.17487/RFC4264

[19] Timothy G. Griffin, Bruce Shepherd, and Gordon Wilfong. 1999. Policy Disputes in Path-Vector Protocols. In *IEEE ICNP '99*. IEEE, USA, 21.

[20] Timothy G. Griffin, Bruce Shepherd, and Gordon Wilfong. 2002. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Netw.* 10, 2 (2002), 232–243. doi:10.1109/90.993304

[21] Timothy G. Griffin and João Luís Sobrinho. 2005. Metarouting. In *ACM SIGCOMM '05*. ACM, New York, NY, USA, 1–12. doi:10.1145/1080091.1080094

[22] Ed Harmoush. 2021. BGP Oldest Path – It's Not What You Think... https://www.practicalnetworking.net/stand-alone/bgp-oldest-path/. Accessed: 21 January 2025.

[23] Ruihan Li, Fangdan Ye, Yifei Yuan, Ruizhen Yang, Bingchuan Tian, Tianchen Guo, Hao Wu, Xiaobo Zhu, Zhongyu Guan, Qing Ma, Xianlong Zeng, Chenren Xu, Dennis Cai, and Ennan Zhai. 2024. Reasoning about Network Traffic Load Property at Production Scale. In *USENIX NSDI '24*. USENIX Association, Santa Clara, CA, 1063–1082. https://www.usenix.org/conference/nsdi24/presentation/li-ruihan

[24] Ruihan Li, Yifei Yuan, Fangdan Ye, Mengqi Liu, Ruizhen Yang, Yang Yu, Tianchen Guo, Qing Ma, Xianlong Zeng, Chenren Xu, Dennis Cai, and Ennan Zhai. 2024. A General and Efficient Approach to Verifying Traffic Load Properties under Arbitrary k Failures. In *ACM SIGCOMM '24*. ACM, New York, NY, USA, 228–243. doi:10.1145/3651890.3672246

[25] Hongqiang Harry Liu, Yibo Zhu, Jitu Padhye, Jiaxin Cao, Sri Tallapragada, Nuno P. Lopes, Andrey Rybalchenko, Guohan Lu, and Lihua Yuan. 2017. CrystalNet: Faithfully Emulating Large Production Networks. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*. ACM, New York, NY, USA, 599–613. doi:10.1145/3132747.3132759

[26] Yu Liu, Pavle Subotic, Emmanuel Letier, Sergey Mechtaev, and Abhik Roychoudhury. 2023. Efficient SMT-Based Network Fault Tolerance Verification. In *FM 2023*. Springer, Berlin, Heidelberg, 92–100. doi:10.1007/978-3-031-27481-7_7

[27] Kirk Lougheed and Yakov Rekhter. 1989. Border Gateway Protocol (BGP). RFC 1105. doi:10.17487/RFC1105

[28] David Oran. 1990. OSI IS-IS Intra-domain Routing Protocol. RFC 1142. doi:10.17487/RFC1142

[29] Santhosh Prabhu, Kuan Yen Chou, Ali Kheradmand, Brighten Godfrey, and Matthew Caesar. 2020. Plankton: Scalable network configuration verification through model checking. In *USENIX NSDI '20*. USENIX Association, Santa Clara, CA, 953–967. https://www.usenix.org/conference/nsdi20/presentation/prabhu

[30] Xiaozhe Shao, Zibin Chen, Daniel Holcomb, and Lixin Gao. 2022. Accelerating BGP Configuration Verification Through Reducing Cycles in SMT Constraints. *IEEE/ACM Trans. Netw.* 30, 6 (June 2022), 2493–2504. doi:10.1109/TNET.2022.3176267

[31] João Luís Sobrinho. 2003. Network routing with path vector protocols: theory and applications. In *ACM SIGCOMM '03*. ACM, New York, NY, USA, 49–60. doi:10.1145/863955.863963

[32] João Luís Sobrinho. 2005. An algebraic theory of dynamic network routing. *IEEE/ACM Trans. Netw.* 13, 5 (2005), 1160–1173. doi:10.1109/TNET.2005.857111

[33] João Luís Sobrinho. 2017. Correctness of Routing Vector Protocols as a Property of Network Cycles. *IEEE/ACM Trans. Netw.* 25, 1 (2017), 150–163. doi:10.1109/TNET.2016.2567600

[34] João Luís Sobrinho. 2019. Fundamental Differences Among Vectoring Routing Protocols on Non-Isotonic Metrics. *IEEE Networking Letters* 1, 3 (2019), 95–98. doi:10.1109/LNET.2019.2913951

[35] João Luís Sobrinho and Miguel Alves Ferreira. 2020. Routing on Multiple Optimality Criteria. In *ACM SIGCOMM '20*. ACM, New York, NY, USA, 211–225. doi:10.1145/3387514.3405864

[36] Alan Tang, Ryan Beckett, Steven Benaloh, Karthick Jayaraman, Tejas Patil, Todd Millstein, and George Varghese. 2023. Lightyear: Using Modularity to Scale BGP Control Plane Verification. In *ACM SIGCOMM '23*. ACM, New York, NY, USA, 94–107. doi:10.1145/3603269.3604842

[37] Robert Tarjan. 1972. Depth-First Search and Linear Graph Algorithms. *SIAM J. Comput.* 1, 2 (1972), 146–160. doi:10.1137/0201010

[38] Timothy Alberdingk Thijm, Ryan Beckett, Aarti Gupta, and David Walker. 2024. Kirigami, the Verifiable Art of Network Cutting . *IEEE/ACM Trans. Netw.* 32, 03 (June 2024), 2447–2462. doi:10.1109/TNET.2024.3360371

[39] Daniel Walton, Alvaro Retana, Enke Chen, and John Scudder. 2016. Advertisement of Multiple Paths in BGP. RFC 7911. doi:10.17487/RFC7911

[40] Hao Wang, Haiyong Xie, Yang Richard Yang, Avi Silberschatz, Li Erran Li, and Yanbin Liu. 2005. Stable Egress Route Selection for Interdomain Traffic Engineering: Model and Analysis. In *IEEE ICNP '05*. IEEE, USA, 16–29. doi:10.1109/ICNP.2005.39

[41] Konstantin Weitz, Doug Woos, Emina Torlak, Michael D. Ernst, Arvind Krishnamurthy, and Zachary Tatlock. 2016. Scalable verification of border gateway protocol configurations with an SMT solver. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2016)*. ACM, New York, NY, USA, 765–780. doi:10.1145/2983990.2984012

[42] Xieyang Xu, Yifei Yuan, Zachary Kincaid, Arvind Krishnamurthy, Ratul Mahajan, David Walker, and Ennan Zhai. 2024. Relational Network Verification. In *ACM SIGCOMM '24*. ACM, New York, NY, USA, 213–227. doi:10.1145/3651890.3672238

[43] Wenwu Yan, Bo Hu, Weiqing Huang, Chao Ma, Xiaobin Tian, and Dong Wei. 2025. UPEA: A Novel BGP Convergence Verification Algorithm with Zero False Positives and Minimal False Negatives . In *IEEE ICNP '25*. IEEE, Los Alamitos, CA, USA, 1–14. doi:10.1109/ICNP65844.2025.11192437

[44] Fangdan Ye, Da Yu, Ennan Zhai, Hongqiang Harry Liu, Bingchuan Tian, Qiaobo Ye, Chunsheng Wang, Xin Wu, Tianchen Guo, Cheng Jin, Duncheng She, Qing Ma, Biao Cheng, Hui Xu, Ming Zhang, Zhiliang Wang, and Rodrigo Fonseca. 2020. Accuracy, Scalability, Coverage: A Practical Configuration Verifier on a Global WAN. In *ACM SIGCOMM '20*. ACM, New York, NY, USA, 599–614. doi:10.1145/3387514.3406217

[45] Yifei Yuan, Fangdan Ye, Yifan Li, Jingkai Zhang, Mengqi Liu, Yuyang Sang, Ruizhen Yang, Duncheng She, Zhiqing Ye, Tianchen Guo, Xiaobo Zhu, Xinji Tang, Li Jia, Zhongyu Guan, Lingpeng Su, Ci Wang, Ruiyang Feng, Shuo Wu, Zhonghui Xie, Cheng Jin, Peng Zhang, Qing Ma, Xianlong Zeng, Dennis Cai, and Ennan Zhai. 2025. New Evolution of Hoyan: Enhancing Scalability, Usability, and Accuracy for Alibaba's Global WAN Verification. In *ACM SIGCOMM '25*. ACM, New York, NY, USA, 809–825. doi:10.1145/3718958.3754343

Appendices are supporting material that has not been peer-reviewed.

A Proof

We now formally prove that the propositions can be implied when the key observations hold in the network. Table 3 formally defines the necessary symbols and concepts based on the SPP framework [19, 20].

Assumption 1 (Intra-group Routing Similarity). *In all stable states, nodes in the same group consistently have preferred next hops in the same group. Formally:* $\forall s \in \mathcal{S}, \forall v_i, v_j \in V$,

$$\gamma(v_i) = \gamma(v_j) \implies \begin{cases} P_1 = \epsilon \wedge P_2 = \epsilon, \\ or \\ \exists v_k, v_l \in V, \gamma(v_k) = \gamma(v_l) \\ \wedge P_1 \in s(v_k) \wedge P_2 \in s(v_l), \end{cases}$$

where $\pi_s(v_i) = (v_i\ P_1)$ *and* $\pi_s(v_j) = (v_j\ P_2)$.

Assumption 2 (Best Path Only). *Each node advertises only its best path to neighbors. Formally:* $\forall s \in \mathcal{S}, \forall (v_i, v_j) \in E$

$$(v_j\ P) \in s(v_j) \wedge P \in s(v_i) \implies P = \pi_s(v_i).$$

Formal illustration for Proposition 1: $\forall s \in \mathcal{S}, \forall v \in V, \forall Q \in \boldsymbol{Q}^v$,

$$\Gamma(\pi_s(v)) = Q \implies \pi_s(v) = P_Q^v,$$

where P_Q^v is the unique path in $\mathcal{P}^v$ with $\Gamma(P_Q^v) = Q$.

Lemma 1. *Proposition 1 can be implied by assumptions 1–2.*

Proof. The proof is based on mathematical induction.

Base Case: Consider any group path of length 1, denoted as $Q = (g_0)$. For all $s \in \mathcal{S}$ and $v \in V$ where $\gamma(v) = g_0$, there is only one best path with group path Q, so the proposition is trivially satisfied.

Inductive Hypothesis: Assume that for any group path of length $n-1$, denoted as $Q' = (g_{n-1}, g_{n-2}, \ldots, g_0)$, and for any node v_{n-1} where $\gamma(v_{n-1}) = g_{n-1}$, there is at most one unique path $P_{Q'}^{v_{n-1}}$ such that for all $s \in \mathcal{S}$:

$$\Gamma(\pi_s(v_{n-1})) = Q' \implies \pi_s(v_{n-1}) = P_{Q'}^{v_{n-1}}.$$

Here, $n - 1 \geq 0$.

Inductive Step: Consider any group path of length n, $Q = (g_n, g_{n-1}, \ldots, g_0)$. For any node v_n where $\gamma(v_n) = g_n$, we want to prove that $\forall s \in \mathcal{S}$:

$$\Gamma(\pi_s(v_n)) = Q \implies \pi_s(v_n) = P_Q^{v_n} \text{ and } P_Q^{v_n} = \arg\max_{P \in \mathcal{R}^{v_n}(Q)} \lambda(P),$$

where $\mathcal{R}^{v_n}(Q)$ is defined as:

$$\mathcal{R}^{v_n}(Q) = \{P = (v_n, P') \mid \gamma(v_{n-1}) = g_{n-1}, P' = P_{Q'}^{v_{n-1}}\}.$$

According to Assumptions 1 and 2, if $\Gamma(\pi_s(v_n)) = Q$, then for any node v_{n-1} where $\gamma(v_{n-1}) = g_{n-1}$, it must hold that $\pi_s(v_{n-1}) = P_{Q'}^{v_{n-1}}$ for state s. Consequently, the set of paths in $s(v_n)$ for group path Q is constant and equals $\mathcal{R}^{v_n}(Q)$, and $\pi_s(v_n)$ must be the best path in $\mathcal{R}^{v_n}(Q)$, which is $P_Q^{v_n}$ according to the definition. Therefore, the proposition is established. □

Formal illustration for Proposition 2: $\forall s \in \mathcal{S}, \forall v \in V$

$$\pi_s(v) \implies \nexists v_i, v_j \in \pi_s(v), \text{ where } v_i \neq v_j \text{ and } \gamma(v_i) = \gamma(v_j)).$$

Table 3: Symbol Definitions

Symbol	Definition
G	Graph with node set $V = \{v_0, v_1, \ldots, v_n\}$ and edge set E.
P	Path $(v_k\ v_{k-1}\ \ldots\ v_1\ v_0)$, where $v_i \in V$ and $(v_i, v_{i-1}) \in E$.
$\mathcal{P}$	Set of all permitted paths in the network. $\mathcal{P}^v$ denotes the permitted paths for node v.
λ^v	Ranking function for node v, where $\lambda^v(P_1) < \lambda^v(P_2)$ means P_2 is preferred.
ϵ	Empty path, $\forall v \in V, \epsilon \in \mathcal{P}^v$, and $\forall P \neq \epsilon, \lambda^v(\epsilon) < \lambda^v(P)$
$\mathcal{S}$	Set of stable states; $s(v)$ represents the paths at v in state s.
$\pi_s(v)$	Best path at v in state s: $\pi_s(v) = \arg\max_{P \in s(v)} \lambda^v(P)$.
$\mathcal{G}$	Set of device groups $\{g_0, g_1, \ldots, g_n\}$.
g_0	Origin group consisting of announcement nodes.
γ	Mapping from node $v_i \in V$ to group $g_k \in \mathcal{G}$.
Q	Group path $(g_k\ g_{k-1}\ \ldots\ g_0)$.
$\boldsymbol{Q}^v$	Set of group paths for $\mathcal{P}^v$.
Γ	Mapping from path $P = (v_k\ v_{k-1}\ \ldots\ v_0)$ to group path $Q = (\gamma(v_k)\ \gamma(v_{k-1})\ \ldots\ \gamma(v_0))$.

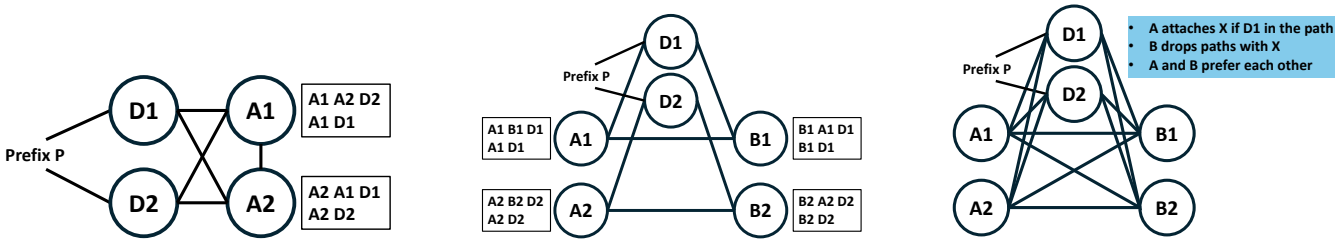
Lemma 2. *Proposition 2 can be implied by assumption 1–2.*

Proof. Assume for contradiction that there exists a path $P = (v_n, \ldots, v_0)$ with nodes v_i and v_j such that $0 \leq i < j \leq n$ and $\gamma(v_i) = \gamma(v_j)$. According to Assumption 2, each node $v_i \in P$, where $i > 0$, selects paths from v_{i-1}, and v_0 selects paths from ϵ, where ϵ represents a virtual source that originates the announcement.

Choose v_i and v_j as the closest pair of such nodes to v_0, meaning that there is no node v_k on the path P such that $k < i$ and there exists another node v_l satisfy $\gamma(v_k) = \gamma(v_l)$. We will show this leads to a contradiction with Assumption 1. First, consider the case where $i = 0$. Since v_0 selects paths from ϵ, no other nodes in P can have the same preference, directly leading to a contradiction. Otherwise, consider the case where $i > 0$. By the closest pair definition, v_{i-1} does not share the same group with other nodes in P. Thus, v_{i-1} and v_{j-1} must be in different groups, which also contradicts Assumption 1. Therefore, such a path P cannot exist with repeated group membership, affirming Proposition 2. □

B Discussion on Policy Similarity and Routing Similarity

The key insight of *TianYan* is to use the intra-group routing similarity assumption to prune path exploration space. Routing similarity results from the common practice of configuring similar policies within a group, but intra-group policy similarity does not necessarily imply routing similarity. Nevertheless, we believe routing similarity widely holds in our network and illustrate this with representative examples.



(a) Preferring routes from the same group. **(b) Incomplete inter-group connectivity.** **(c) Per-device rather than per-group config.**

Figure 10: Illustrative examples showing policy similarity and routing similarity.

B.1 Example 1: Preferring Routes from the Same Group

Figure 10a illustrates a simple setting where policy similarity holds but routing similarity is violated. Routers $A1$ and $A2$ are configured with similar policies that prefer each other's routes. However, in every converged data plane, one router chooses a route from group A and the other from group D, so their preferred next hops lie in different groups, violating intra-group routing similarity.

In our WAN, such configurations have been eliminated, since daily checks by our simulation-based verifier detect them as intent violations and remove them during operation. However, we have observed a practical variant under ADD-PATH in which both routers may simultaneously prefer each other as the next hop, forming a forwarding loop. In this variant the next hops remain within the same group, so routing similarity appears satisfied even though the data plane is incorrect. *TianYan* still detects these cases by uncovering the dispute wheel using Greedy$^+$, as discussed in §8.

B.2 Example 2: Incomplete Inter-group Connectivity

Figure 10b presents another example where policy similarity holds but routing similarity is violated. In this setting, routers in different groups are not fully connected, as each router connects to only one router in each other group and routers within the same group are not interconnected. Devices inside a group can thus be viewed as belonging to separate networks. Each network has two possible converged states, yielding four possible global converged states. In two of these states, routers in groups A and B choose different next-hop groups, which violates intra-group routing similarity.

Such a configuration does not occur in our WAN, since routers in different groups are normally fully connected for link-failure resilience. Even if such a topology were present, *TianYan* would still detect the dispute wheel, but under the routing similarity assumption it could identify only two converged data planes instead of four.

B.3 Example 3: Per-device Rather than Per-group Configuration

Figure 10c illustrates a setting where routers in group A attach community X only to routes whose AS-PATH contains $D1$, while routes containing $D2$ are not tagged. Routers in group B drop the routes carrying X and generally prefer routes from group A, and routers in group A likewise prefer routes from group B. At first glance, this per-device rather than per-group configuration may appear to violate routing similarity and hide non-deterministic convergence.

Under the routing similarity assumption, *TianYan* prunes redundant paths, which might be misunderstood as causing a miss if all paths that group A learns from $D2$ (e.g., $[A1, D2]$ and $[A2, D2]$) were pruned and the non-deterministic convergence between groups A and B were hidden. In reality, the algorithm prunes only lower-priority paths that share the same group path. For example, $[A1, D2]$ is pruned only when it is less preferred than $[A1, D1]$, and $[A2, D2]$ only when it is less preferred than $[A2, D1]$. When all such paths are suppressed, group B receives no route from group A because the routes originated from $D1$ are tagged with X and dropped, and the network converges uniquely with every router in group A selecting group B as the next hop. In this situation routing similarity still holds and, as proved in our analysis, the pruning is correct. If any of the paths that group A learns from $D2$ are not pruned, a dispute wheel exists and routing similarity continues to hold, with the illustration omitted for brevity.

B.4 Conclusion.

These three illustrative examples highlight that policy similarity does not necessarily imply routing similarity and demonstrate the robustness of our routing similarity assumption. They show that when routing similarity is truly violated, such as with mutual preference within a group or with incomplete inter-group connectivity, our system correctly detects the resulting non-deterministic convergence, and when routing similarity holds, as in per-device rather than per-group configuration, our pruning remains sound and does not miss any real violations.

We note that we do not attempt to formally characterize the conditions under which intra-group policy similarity implies intra-group routing similarity, which remains an interesting direction for future work.

C Case Study of a False Positive from Greedy$^+$

This appendix details a subtle false positive from Greedy$^+$ that our SMT-based verifier correctly proved to be converged uniquely, illustrating a scenario that eludes even state-of-the-art techniques like UPEA [43]. Figure 11 presents the simplified SPP instance. At first glance, a preference dependency cycle appears between nodes 1 and 3, as their most preferred paths ($[1, 3, 0]$ and $[3, 1, 0]$) transit

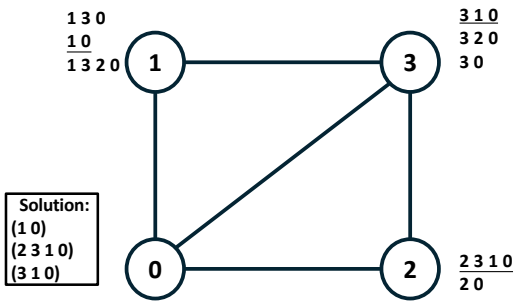


Figure 11: A false positive example of Greedy$^+$.

through each other. This mutual dependency prevents Greedy$^+$ from proving convergence, leading it to flag the instance as non-deterministic. However, the system deterministically converges to the solution as shown in the Figure 11. The key insight is that the dependency cycle is illusory: node 3 can never select the path $[3, 0]$ needed by node 1, because doing so would enable node 2 to advertise the path ($[2, 0]$) back to node 3, forcing it to abandon $[3, 0]$. We present this case as a practical example to aid the development of more accurate verification algorithms.