USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

AnyPro: Preference-Preserving Anycast Optimization based on Strategic AS-Path Prepending

Minyuan Zhou, *Nanjing University and Alibaba Cloud;*
Yuning Chen, *University of California, Merced, and Alibaba Cloud;*
Jiaqi Zheng, *Nanjing University;* Yifei Xu, *University of California, Los Angeles, and Alibaba Cloud;* Pan Hu, Yongping Tang, Wendong Yin, Jie Lin, Qingyan Yu, and Yuanchao Su, *Alibaba Group;* Guihai Chen and Wanchun Dou, *Nanjing University;* Songwu Lu, *University of California, Los Angeles;* Wan Du, *University of California, Merced*

https://www.usenix.org/conference/nsdi26/presentation/zhou-minyuan

This paper is included in the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation.

May 4–6, 2026 • Renton, WA, USA

ISBN 978-1-939133-54-0

Open access to the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

AnyPro: Preference-Preserving Anycast Optimization based on Strategic AS-Path Prepending

Minyuan Zhou[1,4*†], Yuning Chen[2,4*†], Jiaqi Zheng[1], Yifei Xu[3,4†], Pan Hu[4], Yongping Tang[4], Wendong Yin[4], Jie Lin[4], Qingyan Yu[4], Yuanchao Su[4], Guihai Chen[1], Wanchun Dou[1], Songwu Lu[3], Wan Du[2]
[1]*State Key Laboratory for Novel Software Technology, Nanjing University*
[2]*University of California, Merced*
[3]*University of California, Los Angeles*
[4]*Alibaba Cloud*

Abstract

Operating large-scale anycast networks is challenging because client-to-site mappings often misalign with operator's expectation due to opaque inter-domain routing. We present `AnyPro`, the first system to unlock the full potential of AS-path prepending (ASPP), efficiently deriving globally optimal configurations to steer clients toward performance-optimal sites at scale. `AnyPro` first employs an efficient polling mechanism to identify all clients sensitive to ASPP. By analyzing the routing changes during the process, the system derives a set of ASPP constraints that guide client traffic toward the desired sites. We then formulate the anycast optimization problem as a constraint-based program and compute optimal ASPP configurations. Extensive evaluation on a global testbed with 20 PoPs demonstrates the effectiveness of `AnyPro`: it reduces the 90th percentile latency by 37.7% compared to baseline configurations without ASPP. Furthermore, we show that `AnyPro` can be integrated with PoP-level anycast optimization techniques to achieve additional performance gains.

1 Introduction

IP anycast is a foundational and widely adopted technique in today's Internet infrastructure, which involves advertising the same IP prefix from multiple geographically distributed Points of Presence (PoPs). This design enhances key service qualities including resilience to failures, distributed load balancing, and reduced latency by routing users to topologically or geographically proximate instances [4, 7, 16, 19, 20, 24, 28, 43, 44]. Client-to-PoP routing is not governed by explicit centralized rules, but rather by the policy-driven inter-domain routing behavior of BGP, which selects paths based on configured policies and AS-level relationships. Due to its inherent flexibility, scalability, and robustness [45], IP anycast has become a critical technology underlying major global services, including root DNS systems [26, 33, 34], large-scale DDoS mitigation platforms [29], and content delivery networks (CDNs) [12], where improving performance and reliability is essential.

In a typical anycast deployment, each site serves a group of client IPs – forming what is known as its catchment. Ideally, these catchments align with geographic proximity, topological closeness, or minimum-delay partitions to ensure optimal quality of service, especially in terms of latency. However, anycast does not always achieve such performance. Due to the nature of BGP routing, clients may be directed to geographically distant sites [4, 5, 25, 26, 33], resulting in significant path inflation that can add over 100 ms of latency and severely degrade user experience.

Network operators often predefine optimal client-to-site mapping (catchment) based on operational experience or application needs [43]. Achieving optimal mapping has always been the north star for anycast operations, but existing approaches are still far from it due to limited visibility into cross-AS topology and external routing policies. For instance, embedding the PoP's geographic location in a BGP announcement is an intuitive idea [25], but it demands BGP attribute extensions across multiple routing domains, which is operationally impractical. AnyOpt [43], a recent optimization framework, selects a subset of sites through pairwise preference discovery. While it improves global catchment efficiency, the cost is sacrificing the latency of a significant fraction of clients, a compromise operators are unwilling to make.

In this landscape, AS-path prepending (ASPP) stands out as a remarkably versatile mechanism for fine-grained routing control [27]. Unlike more intrusive methods, ASPP extends AS-path lengths at selected PoPs to subtly steer BGP route selection for certain ASes. However, this powerful tool has never been systematically used for large-scale anycast tuning. The key challenge lies in the huge search space. Unlike binary site selection, ASPP introduces exponentially more combinations, making pairwise scan [43] computationally infeasible. Consequently, ASPP remains an underutilized yet highly promising mechanism for anycast tuning.

*The first two authors contributed equally to the work.
†Work done during the internships of Yuning Chen, Minyuan Zhou, and Yifei Xu at Alibaba Cloud.

In this paper, we propose AnyPro, which can achieve globally optimal ASPP configurations in a practically efficient way. AnyPro operates through three core phases: *max-min polling* identifies ASPP-sensitive clients and establishes preliminary constraints; *optimization solving* maximizes constraint satisfaction; and *contradiction resolution* uses binary search to refine conflicting constraints. The final output is an optimal set of ASPP configurations per PoP-transit access point that yield the intended client-to-ingress mappings.

Overall, AnyPro provides a comprehensive framework for anycast optimization with three key capabilities: (1) systematic detection of ASPP-sensitive clients, (2) efficient identification of configuration constraint to satisfy routing preferences, and (3) derivation of the optimal ASPP configurations. We evaluate AnyPro on a global testbed of 20 production PoPs. To further evaluate AnyPro's versatility, we integrate it with AnyOpt – a state-of-the-art anycast optimization technique that operates through strategic PoP enablement. This combined approach yields remarkable performance gains: the 90th percentile latency is reduced to 58.0 ms, representing a significant improvement over baseline measurements. Regional-scale validation in Southeast Asia demonstrates particularly impressive results, with a 17.9% improvement in the matching accuracy. These multi-scale experiments confirm AnyPro's effectiveness across different network topologies and operational requirements, while its constraint-based optimization framework provides network operators with both flexibility and precise control over anycast routing behavior.

The contribution can be summarized as follows. AnyPro pioneers production-grade anycast catchment optimization by introducing a robust and deterministic framework that automatically derives optimal AS-path prepending configurations aligned with operator performance objectives. It establishes comprehensive theoretical foundations and a practical methodology, supported by an efficient algorithm capable of identifying ASPP-sensitive clients, generating preference-aware constraints — even under third-party or middle-ISP changes — and computing near-optimal configurations through iterative refinement. Evaluated on production networks, the system demonstrates substantial gains in catchment efficiency and RTT across diverse operational scenarios, while supporting peering-aware configurations and enabling future integration of advanced BGP controls.

2 Background and Motivation

Global network services require providers to deploy distributed servers, ensuring clients connect to optimal sites for ideal Quality of Service (QoS) [41, 42]. Anycast simplifies this by advertising the same IP prefixes across all sites, leveraging BGP to route clients to the nearest or best-performing Point of Presence (PoP). We define a PoP as the physical access point and its connected clients as the catchment of that PoP. Each PoP may have multiple transit providers, and we refer to an ingress as a unique (PoP, transit provider) pair, representing a distinct entry point into the network.

Prior work has optimized anycast performance through various approaches, such as deploying additional PoPs [11], expanding peering relationships with other ASes [7, 19], and encoding PoP locations in BGP communities [25]. However, these methods often lack deployability in practice. AnyOpt [43] introduced PoP-level optimization by selectively enabling or disabling sites, but its reliance on pairwise BGP experiments creates operational overhead that limits scalability. To address this, we propose a finer-grained control mechanism: instead of operating at the PoP level, we optimize announcements at the ingress level — the granularity of a (PoP, transit provider) pair — enabling more flexible and scalable anycast routing.

Typically, network operators employ two mechanisms to customize route announcements. **(1) Community** [23]. BGP communities act as standardized tags that convey routing instructions to neighboring ASes, enabling functions like route propagation control and path selection optimization [21, 22]. However, their implementation lacks universal consistency — different networks assign distinct community numbers for identical functions [3, 10], creating significant operational complexity. This variability makes community-based routing configurations particularly challenging to manage at scale. Due to these inherent complexities, we exclude BGP community management from our primary focus and address it separately in §5. **(2) AS-path prepending** [17]. AS-path prepending (ASPP) is a BGP traffic engineering mechanism in which an AS artificially extends its AS-path length by inserting duplicate instances of its AS number. Service providers strategically vary prepending lengths when announcing anycast prefixes to transit providers (ingress), leveraging BGP's path selection algorithm, which prioritizes routes with shorter AS-paths. This enables fine-grained control over inbound traffic flows [24]. Importantly, ASPP policies exhibit remarkable stability across the Internet — only 0.3% of paths show prepending changes, and most modifications are temporary, reverting within days [45]. This stability ensures that our optimization operates in a predictable environment, free from interference by sporadic prepending adjustments in other networks (as we detailed in §3).

While ASPP offers significant potential for catchment optimization, its effective implementation presents three key challenges: (1) managing computational complexity, (2) inferring sufficient routing topology context to ensure accurate client-to-ingress mappings, and (3) constraining the total number of required configuration experiments. To address these challenges, we require a systematic approach that can both efficiently analyze ASPP's impact on individual client routing and determine globally optimal prepending configurations across all network ingresses.

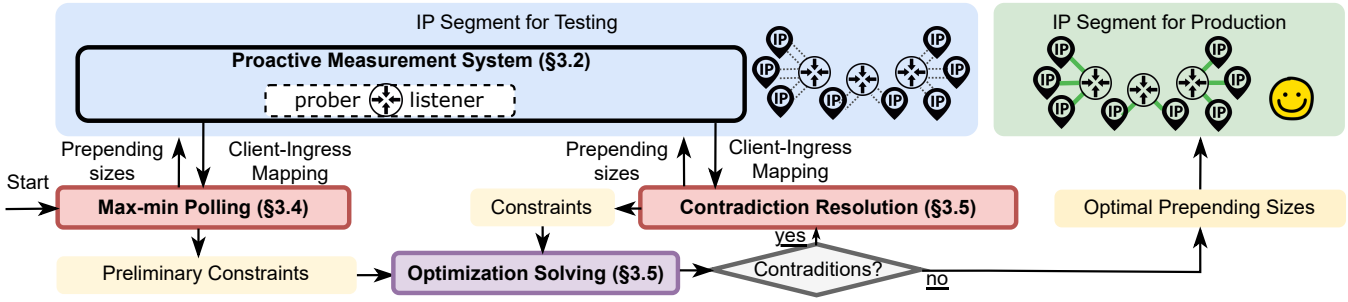


Figure 1: AnyPro system overview.

3 AnyPro

This section presents an overview of AnyPro, its practical challenges, methods for discovering ASPP-sensitive clients, optimization for preference-preserving configurations, and the rationale behind its operational algorithms.

3.1 Overview

Real-world anycast testbed. Our anycast testbed operates on a production-grade network comprising 20 globally distributed PoPs, each connected to 1~3 transit providers (totaling 38 ingresses, as listed in Appendix B). These PoPs peer with other ASes through IXPs, ensuring realistic routing conditions. The infrastructure supports two IP segments: one for live traffic and one dedicated to experiments. While each segment may employ independent ASPP configurations and exhibit distinct client-to-ingress mappings, identical settings always yield reproducible mappings since they share the same backbone network. Using the test segment, we deploy a proactive measurement system that efficiently characterizes global catchments for arbitrary ASPP configurations without exhaustive testing, enabling scalable optimization.

Max-min polling identifies ASPP-sensitive clients while generating preference-preserving routing constraints. To derive sufficient conditions for ingress-level preference compliance, we first design an exploration function, *max-min polling*, which systematically probes the configuration space by: (1) initializing all PoP-transit prepending lengths to their maximum values, then (2) iteratively reducing to the minimum length. This scanning process identifies all feasible ingress preferences per client IP (with formal proof in §3.4) while simultaneously generating preliminary preference-preserving constraints.

Constraint contradiction resolution. While *max-min polling* generates preliminary preference-preserving constraints for individual clients, combining constraints across all clients reveals frequent contradictions. This occurs because the initial exploration only tests prepending lengths at the boundaries (0 and *MAX*), resulting in coarse-grained length differences (limited to 0 and *MAX*) that fail to capture intermediate optimal values. To address this, we develop an efficient contradiction resolution algorithm that employs binary search to iteratively refine the prepending length difference thresholds. This approach produces a tightly constrained optimization problem while maintaining computational efficiency — reducing the search space by orders of magnitude compared to brute-force methods.

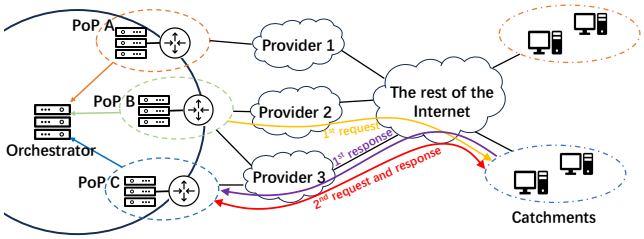


Figure 2: Illustration of the real-world testbed we used for anycast measurements.

3.2 Proactive measurement system

Although traditional passive measurement with client request logs can reflect real-time ingress behavior statistics, it has two major limitations. First, client request logs in some client IPs can be sparse, especially considering the peak and off-peak hours within different time zones. Therefore, it requires significant time to output a complete client-ingress mapping for all clients. Second, client request logs are collected directly from production networks, which prevents trying risky configurations and collecting the feedback effect on real networks.

To ensure the generalizability of our experimental findings across diverse network environments, we construct a representative and stable IP set derived from the authoritative ISI IPv4 hitlist [14] — the most comprehensive public source of active IPv4 addresses, offering broad coverage across geographic and topological regions. This IP selection strategy, combined with a client-agnostic proactive measurement approach that operates without relying on client requests, enhances the universal applicability of our experimental results [13, 38]. To further improve reliability, we filter the initial IPs through week-long active probing, retaining only addresses with under 10% packet loss, which excludes unstable nodes and strengthens measurement validity. The final set comprises ~2.4M IP

addresses, a substantial portion of which belong to network middleboxes.

To measure client-ingress mapping and client-to-PoP RTTs, we developed a custom measurement tool (Figure 2). Each ingress hosts a prober-listener pair that exchanges ICMP packets in an anycast format. The measurement process begins with proactive probing: anycast PoPs send ICMP requests (yellow lines) with anycast source addresses to target clients. Responses (purple lines) reveal catchment mappings by routing to the nearest enabled PoP. Upon receiving a response, the ingress immediately issues a follow-up ICMP request containing a unique identifier and timestamp. When the response (red line) returns, we compute RTT from the timestamp delta and log the result alongside ingress metadata (transit providers, peers, IXPs) to inform announcement optimization. This dual-phase approach simultaneously captures catchment boundaries and latency while minimizing measurement overhead.

3.3 Practical challenges

We have introduced the overview of `AnyPro`. However, to make it practical, we must address the following challenges.

Determining client-to-ingress mappings across all possible prepending configurations presents significant computational and methodological challenges. In anycast networks, client-to-ingress mappings are determined by BGP routing decisions, which respond dynamically to AS-path prepending adjustments. For a network with n ingresses (each supporting m prepending lengths), exhaustive testing of all possible configurations would require $O(m^n)$ BGP experiments. Given that each experiment incurs nontrivial convergence delays (often minutes), this brute-force approach becomes computationally intractable at scale, necessitating a more efficient optimization methodology.

Constraint contradiction resolution incurs significant computational overhead. When consolidating preliminary preference-preserving constraints across all clients, inherent contradictions frequently arise due to competing routing objectives. Resolving each contradiction through naive enumeration would necessitate $O(m)$ BGP experiments per conflict (systematically testing each ingress's prepending length from MIN to MAX). This approach becomes operationally infeasible for three key reasons: (1) BGP convergence delays introduce minutes of network churn per experiment, (2) the quadratic scaling of total experiments ($O(m \cdot k)$ for k contradictions) becomes prohibitive in large networks, and (3) transient Internet routing fluctuations may invalidate results before completion. Our measurements show that even mid-sized deployments (e.g., 20 PoPs with 38 ingresses) could require weeks of continuous testing to resolve all conflicts through brute-force methods — an untenable proposition for production networks requiring real-time optimization.

3.4 Max-min Polling

A client IP is considered ASPP-sensitive only when it can reach the prefix through at least two distinct ingresses. Among all ingresses, the client IPs prefer one over others due to service quality discrepancies. The matrix that records whether each client prefers an ingress or not is the desired client-ingress mapping, denoted as $\boldsymbol{M}^*$. To find out all ASPP-sensitive clients and their potential routes, we develop an efficient *max-min polling* method as depicted in Algorithm 1, which produces two outcomes:

Outcome 1: Discovering ASPP-sensitive clients. *Max-min polling* is executed by initially setting the prepending length to *MAX* for all ingresses, and iteratively adjusting each to zero while others remain *MAX*, where *MAX* is an upper bound of the prepending length, while *MIN* is set as zero. Initially, all prepending lengths are set to *MAX* (line 1). Under this initial configuration, the client-ingress mapping, denoted as $\boldsymbol{M}$, serves as a baseline mapping (line 2). We iteratively tune the ASPP configuration of each ingress and observe the corresponding client-ingress mapping changes (line 3–8). For each ingress, we first eliminate the prepending (line 4). At this moment, since the prepending lengths of others are kept at *MAX*, this ingress becomes the most preferred, in terms of AS-path length, and then we measure the client-ingress mapping, denoted as $\boldsymbol{M}'$ (line 5). Comparing $\boldsymbol{M}'$ and $\boldsymbol{M}$, we identify the clients whose ingresses change. These clients are identified as ASPP-sensitive, as their routing behavior responds directly to alterations in the prepending configuration of the specific ingress.

Algorithm 1 *max-min polling* execution and processing

Input: All ingresses $\boldsymbol{P} = \{p_{i,j}\}$ for the i_{th} PoP and j_{th} transit; the desired client-ingress mapping $\boldsymbol{M}^*$
Output: Pairwise preference constraints $\boldsymbol{\Gamma}$ for $\boldsymbol{M}^*$

1: Set the prepending length of each $p_{i,j} \in \boldsymbol{P}$ to *MAX*.
2: Gather client-ingress mapping $\boldsymbol{M}$ with §3.2.
3: **for** each $p_{i,j} \in \boldsymbol{P}$ **do**
4: Set the prepending length of $p_{i,j}$ to 0.
5: Gather client-ingress mapping $\boldsymbol{M}'$ with §3.2.
6: Compare $\boldsymbol{M}'$ and $\boldsymbol{M}$, generate pairwise preference constraints γ to achieve the desired mapping $\boldsymbol{M}^*$.
7: $\boldsymbol{\Gamma} = \boldsymbol{\Gamma} \cup \gamma$.
8: Set the prepending length of $p_{i,j}$ to *MAX*.

To prove the completeness of *max-min polling* in identifying ASPP-sensitive clients, we now introduce a lemma about *max-min polling* in a pair of ingresses.

Lemma 1 (Completeness of *max-min polling* for a pair of ingresses)**.** *For any two ingresses $p_{i,j}$ and $p_{m,n}$, max-min polling can effectively explore all potential routes for all clients by shrinking the length of the decision space from an interval $(s_{i,j}, s_{m,n} \in [MIN, MAX])$ to two points $(s_{i,j}, s_{m,n} \in$*

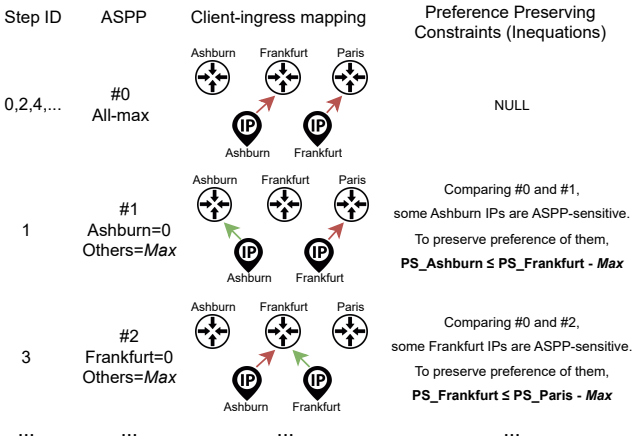


Figure 3: An illustrative example of *max-min polling*.

$\{MIN, MAX\}$), *where* $s_{i,j}$ *and* $s_{m,n}$ *is the prepending length of* $p_{i,j}$ *and* $p_{m,n}$*, respectively.*

Proof. During *max-min polling*, the prepending length difference between any two ingresses can be defined as $s_{i,j} - s_{m,n} = \{MIN - MAX, 0, MAX - MIN\}$. Therefore, considering that the preference of the route selection is monotonic to the prepending length difference, all possible routes are explored by *max-min polling*. □

Based on the discussion of two ingresses shown in Lemma 1, we can further derive a theorem to extend Lemma 1 to a general case with multiple ingresses.

Theorem 2. *max-min polling can explore all ASPP-sensitive clients as well as their potential routes, where the prepending length of each ingress belongs to the interval [MIN, MAX].*

Proof. The set of all ingresses can be divided into multiple pairs. Each pair can be explored independently by *max-min polling* as discussed in Lemma 1. By applying *max-min polling* to each pair, we ensure that the ASPP-sensitive clients and their potential routes are thoroughly explored. □

Outcome 2: Revealing preliminary preference-preserving constraints. During *max-min polling*, apart from discovering all sensitive clients, we also obtain all *candidate ingresses* for each client. Among them, we generate a set of inequation conditions to drive each client into the desired ingress. These inequation conditions are defined as *preliminary preference-preserving constraints*. For each client, pairwise comparisons between candidate ingresses can establish a critical threshold Δs^* that governs routing preference: if the operator intends to prioritize ingress A over ingress B, constraints are formulated to ensure A's ASPP configuration maintains a prepending advantage of at least Δs^* than B. However, since *max-min polling* tests only two ASPP configurations per ingress (*i.e.*, 0 and MAX prepending), the precise Δs^* cannot be empirically determined, resulting in preliminary constraints that approximate routing behavior.

Figure 3 illustrates an example of deriving preliminary preference-preserving constraints during *max-min polling* process (lines 6–7, Algorithm 1). Initially, each ingress is configured to prepend *MAX* length. When a client c_k in Ashburn switches from ingress Frankfurt ($p_{Frankfurt}$) to Ashburn ($p_{Ashburn}$) (for simplicity, we do not specify the transit provider here) after removing prepending at Ashburn ($s_{Ashburn}$) while others remain *MAX*, it reveals a partial preference order: $p_{Ashburn} \succ_{c_k} p_{Frankfurt}$ under $s_{Ashburn} = 0$, $s_{Frankfurt} = MAX$, but $p_{Frankfurt} \succ_{c_k} p_{Ashburn}$ under uniform MAX prepending. To enforce c_k's desired access to $p_{Ashburn}$, the preference-preserving constraint $s_{Ashburn} \leq s_{Frankfurt} - MAX$ is derived. This ensures $p_{Ashburn}$'s dominance by maintaining a prepending gap exceeding Δs^* between $s_{Ashburn}$ and $s_{Frankfurt}$, while intermediate configurations ($s_{Ashburn} = \frac{MAX}{2}, s_{Frankfurt} = MAX$) are excluded due to nondeterministic routing behavior (We also show why *min − max* polling fails to satisfy our requirements in Appendix C).

We now establish a formal proof demonstrating that for any pair of candidate ingresses, there exists a unique critical threshold Δs^* governing routing preference:

Theorem 3 (Existence and uniqueness of preference-preserving constraints for a pair of ingresses)**.** *Given a client* c_k*, if it can enter either ingress* $p_{i,j}$ *or* $p_{m,n}$ *under different prepending lengths, there exists a unique integer* Δs^**, s.t.,* $(p_{m,n} \succ_{c_k,(s_{m,n}-s_{i,j}=\Delta s^*)} p_{i,j}) \wedge (p_{i,j} \succ_{c_k,(s_{m,n}-s_{i,j}=\Delta s^*+1)} p_{m,n})$*, where* $\succ_{c_k,\boldsymbol{S}}$ *represents the partial routing preference of client* c_k *under prepending configuration* $\boldsymbol{S}$*, and* $s_{i,j}$ *represents the prepending length of* $p_{i,j}$*.*

Proof. **(Existence).** Since the client c_k switches route (from $p_{m,n}$ to $p_{i,j}$) when the prepending length difference $s_{m,n} - s_{i,j}$ increases monotonically, there must be an integer d where the preference flips.

(Uniqueness). Assume two distinct flip points $\Delta s_1 < \Delta s_2$ exist. This would imply that c_k switches back to $p_{m,n}$ for some $\Delta s \in (\Delta s_1, \Delta s_2)$, violating BGP's shortest-path selection principle [22], which shows that once a route becomes less preferred due to increased prepending, it cannot regain preference without external policy changes. Thus, no such interval can exist, and Δs^* must be unique. Thus, there cannot be more than one such flip point, and Δs^* must be unique. □

3.5 Solving with constraints and contradiction resolution

In this section, we describe: (1) Given the constraints, how we solve the optimization to get an optimal prepending configuration; (2) How we efficiently resolve contradictions among constraints to ensure optimality. (3) An exceptional observation that ingress shifts can be caused by third-party prepending length.

Optimization solving implementation and feasibility verification. Given the desired client-ingress mapping $\boldsymbol{M}^*$, we

wish to determine an optimal prepending configurations $\boldsymbol{S}$ such that the product of $\boldsymbol{M}^*$ and $\boldsymbol{M}$ is maximized (that is to say, $\boldsymbol{M}$ matches $\boldsymbol{M}^*$ as much as possible), where $\boldsymbol{M}$ is the resulting client-ingress mapping under the prepending configurations $\boldsymbol{S}$. Accordingly, we formulate this problem as the program (1).

$$\text{maximize} \sum_{c_k \in \boldsymbol{C}} \sum_{p_{i,j} \in \boldsymbol{P}} \boldsymbol{M}^*_{c_k,p_{i,j}} \cdot \boldsymbol{M}_{c_k,p_{i,j}} \tag{1}$$

$$\text{s.t.} \quad \sum_{p_{i,j} \in \boldsymbol{P}} \boldsymbol{M}_{c_k,p_{i,j}} = 1, \quad \forall c_k \in \boldsymbol{C} \tag{1a}$$

$$\boldsymbol{M}_{c_k,p_{i,j}} \in \{0,1\}, \quad \forall c_k \in \boldsymbol{C}, \forall p_{i,j} \in \boldsymbol{P} \tag{1b}$$

$$\boldsymbol{M}_{c_k,p_{i,j}} \geq \boldsymbol{M}_{c_k,p_{m,n}}, \quad \forall p_{i,j} \succ_{c_k,\boldsymbol{S}} p_{m,n} \tag{1c}$$

$$s_{i,j} \in \{0,1,\ldots,MAX\}, \quad \forall s_{i,j} \in \boldsymbol{S} \tag{1d}$$

The constraint (1a) characterizes that one client enters exactly one ingress. $M_{c_k,p_{i,j}}$ is a zero-one variable indicating whether the client c_k enters the ingress $p_{i,j}$ or not. The constraint (1c) preserves the partial order preference between $p_{i,j}$ and $p_{m,n}$ under the configuration $\boldsymbol{S}$. The optimization variable $s_{i,j}$ is an integer, representing the prepending length configured for the ingress $p_{i,j}$. This problem is NP-hard, which can be reduced from classic max-SAT problem [40] and the proof can be found in Appendix D.

Before running program (1) using commercial solver OR-Tools [32], we verify the solving feasibility with statistics. First, although the total number of clients is large, most clients exhibit identical ingress selection patterns across configurations, enabling aggregation into *client group* that the same set of routing constraints. This grouping is derived empirically from observed routing behavior rather than predefined structures such as BGP atoms. Specifically, despite there are ~2.4M clients, they form only ~14,700 unique client groups. Second, sparse candidate ingress distribution is observed: 58% of client groups have only 1–2 candidate ingresses (equivalently 0–1 constraints), while merely 15% have over 10 candidate ingresses. Third, each client group's constraints are structured in conjunctive normal form (CNF), requiring simultaneous satisfaction of all constraints. For example, constraints for a client group c_a selecting desired ingress might be: $(s_{i,j} \leq s_{m,n} - MAX) \wedge (s_{i,j} \leq s_{x,y} - MAX)$. while group c_b choosing the desired ingress could require $(s_{i,j} \leq s_{m,n} - MAX) \wedge (s_{i,j} \leq s_{v,w} - MAX)$. Since the MAX-SAT problem is given by a CNF formula, our original problem is reducible to maximizing satisfaction of distinct atomic constraints of all client groups: $(s_{i,j} \leq s_{m,n} - MAX) \wedge (s_{i,j} \leq s_{x,y} - MAX) \wedge (s_{i,j} \leq s_{v,w} - MAX)$. Therefore, the complexity of the problem is further reduced.

As a result, the total number of constraints is less than 1,000, and the solver can solve it in less than one second. For further scaling concerns, if the number increases significantly, we can prioritize large client groups to reduce the number or use the early stopping strategy to limit the inference time.

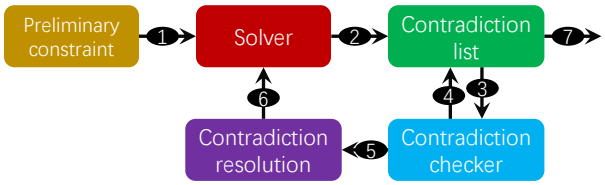


Figure 4: Workflow of the contradiction resolution.

Constraint contradiction resolution. Preliminary constraints derived from *max-min polling* are categorized into two types based on their operational origins and structural forms: **TYPE-I constraints**, expressed as $s_{i,j} \leq s_{m,n} - MAX$, originate from scenarios where a client's desired ingress $p_{i,j}$ becomes accessible only when its prepending length $s_{i,j}$ reaches zero. **TYPE-II constraints**, expressed as $s_{i,j} \leq s_{m,n}$, arise when a client can access $p_{i,j}$ when both $p_{i,j}$ and $p_{m,n}$ are prepended MAX ASes but turns to $p_{m,n}$ once $s_{m,n}$ depletes to zero.

Consider two clients c_k and c_l whose preliminary constraints are $s_{i,j} \leq s_{m,n} - MAX$ and $s_{m,n} \leq s_{i,j}$ respectively. As these two preliminary constraints cannot be satisfied simultaneously (the only solution for the first constraint is $s_{m,n} = 0, s_{i,j} = MAX$ which is inherently unsatisfiable for the second constraint), we denote it as a constraint contradiction.

Constraint contradictions arise from the preliminary constraints' maximal looseness. By introducing tighter bounds (Δs), we resolve the contradictions through interval overlap analysis. Taking the same example, for c_k's constraints, according to Theorem 3, there exists a Δs_1^* which satisfies: $\Delta s_1^* \leq MAX$ and $s_{i,j} \leq s_{m,n} - \Delta s_1^*$. Therefore, the feasible interval for c_k is $s_{m,n} - s_{i,j} \in [\Delta s_1^*, MAX]$. Similarly, there exists a Δs_2^* which satisfies: $\Delta s_2^* \geq 0$ and $s_{m,n} \leq s_{i,j} + \Delta s_2^*$, with its feasible interval defined as $s_{m,n} - s_{i,j} \in [0, \Delta s_2^*]$. Consequently, the resolvability of the contradiction depends on the intersection of these intervals: $s_{m,n} - s_{i,j} \in [\Delta s_1^*, \Delta s_2^*]$. This intersection is non-empty if and only if $\Delta s_2^* \leq \Delta s_1^*$. If this condition holds, the constraints are resolvable; otherwise, they can be asserted unresolvable.

TYPE-II constraints are inherently resolvable between themselves because their mutual contradiction (*e.g.*, $s_{i,j} \leq s_{m,n}$ and $s_{m,n} \leq s_{i,j}$) collapses into an equality $s_{i,j} = s_{m,n}$, which is always satisfiable. Conversely, conflicting TYPE-I constraints are irreconcilable when mutually imposed(*e.g.*, $s_{i,j} \leq s_{m,n} - MAX$ and $s_{m,n} \leq s_{i,j} - MAX$), as their combined logic enforces $MAX = 0$, violating the predefined $MAX > 0$ parameter. Based on this insight, we design a binary scan algorithm specifically targeting hybrid contradictions between TYPE-I and TYPE-II constraints.

Algorithm 2 resolves constraint contradictions through coordinated bisection refinement of slack variables ($\Delta s1$ and $\Delta s2$) for conflicting TYPE-I γ_1 and TYPE-II γ_2 constraints. Initialized with $\Delta s1 \in [0,k]$ (experience prior binary scan

Algorithm 2 Binary scan for Constraint Resolution

Input: Contradictions $\gamma_1 : s_{i,j} \leq s_{m,n} - k$ and $\gamma_2 : s_{m,n} \leq s_{i,j} + b$

Output: Refined constraints γ_1', γ_2'.

1: Initialize $\Delta s_1^{min} \leftarrow 0$, $\Delta s_1^{max} \leftarrow k$, $\Delta s_2^{min} \leftarrow b$, $\Delta s_2^{max} \leftarrow MAX$
2: **while** $\Delta s_1^{max} > \Delta s_2^{min}$ and $\Delta s_1^{min} < \Delta s_2^{max}$ **do**
3: **if** $s_{i,j} \leq s_{m,n} - \frac{\Delta s_1^{min} + \Delta s_1^{max}}{2}$ still holds **then**
4: $\Delta s_1^{max} \leftarrow \frac{\Delta s_1^{min} + \Delta s_1^{max}}{2}$
5: **else**
6: $\Delta s_1^{min} \leftarrow \frac{\Delta s_1^{min} + \Delta s_1^{max}}{2}$
7: **if** $s_{m,n} \leq s_{i,j} + \frac{\Delta s_2^{min} + \Delta s_2^{max}}{2}$ still holds **then**
8: $\Delta s_2^{min} \leftarrow \frac{\Delta s_2^{min} + \Delta s_2^{max}}{2}$
9: **else**
10: $\Delta s_2^{max} \leftarrow \frac{\Delta s_2^{min} + \Delta s_2^{max}}{2}$
return $s_{i,j} \leq s_{m,n} - s_1^{max}$, $s_{m,n} \leq s_{i,j} + s_2^{min}$

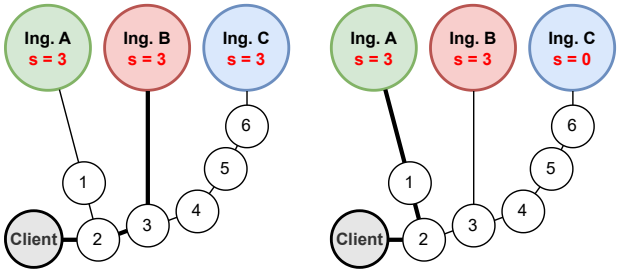


Figure 5: Illustration of an ingress shift caused by the ASPP change of a third-party ingress .

adjustments but remain insufficiently tightened) and $\Delta s2 \in [b, MAX]$, the algorithm iteratively tightens both constraints by bisecting their respective threshold ranges, followed by client-ingress mapping validation to verify whether the modified constraints preserve desired client access patterns (lines 3&7). If validation fails, the search continues in subranges favoring larger $\Delta s1$ and smaller $\Delta s2$ to resolve the contradictions (lines 6&10), otherwise, it narrows toward stricter adjustments (lines 4&8). The process ends either when overlapping intervals $\Delta s_1^{max} < \Delta s_2^{min}$ indicate resolvable conditions or when disjoint intervals $\Delta s_1^{min} > \Delta s_2^{max}$ confirm irreconcilable contradictions. Each time, the adaptive binary scan strategically avoids the exact determination of Δs^*. Instead, it employs progressive boundary refinement by continuously narrowing feasibility regions until either the contradictions converge or are demonstrated to be unresolvable. This dual bisection mechanism ensures minimal constraint relaxation while systematically probing the solution space.

In Figure 4, we present a systematic constraint solution workflow that begins with the preliminary constraints generated from max-min polling. We first use a commercial solver OR-Tools to try to solve the constraint lists (❶). If the constraint set is unsolvable, the solver identifies contradictory constraint pairs $\xi_i = (\gamma_{i1}, \gamma_{i2})$ that cannot coexist, forming a contradiction set $\Xi = \{\xi_1, \xi_2, \dots\}$ (❷). These contradictions are prioritized by their client impact count, then evaluated through a verification process: first assessing whether either constraint in the pair constitutes a tight inequation (precisely bounded by Δs)(❸), where any tight constraint pairs automatically designates the contradiction as unresolvable (❹), and second attempting resolution via binary scan optimization for untightened pairs (❺). After each resolution attempt, the constraint set is revalidated through solver re-execution (❻) to propagate resolution effects across interdependent constraints. Since no additional contradictions would be generated, the entire set Ξ can be processed in one pass. The workflow terminates when all contradictions are either resolved or definitively classified as unresolvable, yielding an optimized prepending configuration through this closed-loop verification-optimization cycle (❼). Constraints processed through the binary scan are formally defined as finalized preference-preserving constraints.

3.6 Deep Dive: Algorithm Design Rationale

This section discusses the design rationale behind the algorithms used in `AnyPro`'s operational deployment.

Third-party impact: When analyzing the results in *max-min polling*, we have identified two distinct types of clients that show different reactions to prepending adjustments: (1) 95.1% of client groups shift to one ingress when the prepending length of this ingress is tuned to smaller values, which aligns with intuitions and the Bi-ingress competing model in Theorem 3. (2) Conversely, 4.9% clients make the ingress shift from B to A when the prepending length of an unrelated third-party C is adjusted. For instance, a client group at ⟨California, AS132203⟩ switches from the ingress in Frankfurt to Ashburn when the prepending length of Malaysia is changed to zero.

To understand the causes behind the second category's behavior, we delve into an analysis of traceroutes from clients to their respective ingresses under different configurations. We demonstrate this phenomenon with an example in Figure 5. In this figure, each circle symbolizes an individual AS, and connecting lines between ASes indicate the propagation of announcements. In this network with three ingresses A, B, and C. The client whose desired mapping is A receives anycast announcements. Suppose we are conducting *max-min polling* with $MAX = 3$. In the base mapping (left), "AS 2" prefers ingress B when the path to B is the shortest. When C is set as 0 (right), paths to C and A are both the shortest, but "AS 2" chooses A instead of C, which actually adjusts itself. This behavior occurs because when AS path lengths are equivalent, "AS 2" has a bias for routes via "AS 1" rather than those via "AS 3" due to lower-tier-breaking metrics (e.g., origin code, MED or router ID) [9] of BGP. Finally, "AS 2" will advertise the path originating from ingress A to the client.

AnyPro is compatible with this case through a new preference-preserving constraint format:

$$(p_{m,n} \succ_{c,(s_{m,n}-s_{i,j}\leq \Delta s^*)} p_{i,j}) \wedge (p_{i,j} \succ_{c,(s_{m,n}-s_{i,j}>\Delta s^*)} p_{m,n}),$$

where s_1 and s_2 are no longer required to be $s_{i,j}$ and $s_{m,n}$, exactly matching $p_{i,j}$ and $p_{m,n}$. Instead they can be the prepending lengths of any other ingresses. Note that our optimization formulation intrinsically allows this, i.e., the $\boldsymbol{S}$ in $M_{c_k,p_{i,j}} \geq M_{c_k,p_{m,n}}, \forall P_{i,j} \succ_{c_k,\boldsymbol{S}} P_{m,n}$ can be any prepending length array. And the input to the solver is still in the same format.

Middle ISP's impact: In empirical deployments of AnyPro, a primary operational concern arises from ISPs dynamically adjusting ASPP configurations — either by truncating prepending length or reinflating AS paths. Actually this type of change on the ASPP configuration of the internal ISPs will not influence the correctness of our preference-preserving constraints, provided ISP reactions to these configurations remain consistent — a robustness validated by prior studies [27]. This resilience stems from the system's inherent integration of client response modeling to ASPP changes, which inherently accounts for ISP-driven configuration fluctuations. For instance, if a middle ISP shortens the ASPP length to 3, our dynamic parameter tuning from 9 to 3 (if we set *MAX* as 9) is transparent to clients — no ingress shifts occur. However, if further tuning ASPP to 2 triggers ingress changes, the system infers a preference-preserving constraint with $\Delta s = 7$, which inherently accommodates both operator-initiated tuning and ISP-driven ASPP variations, ensuring stable routing behaviours despite configuration dynamics.

4 Evaluation

In this section, we first derive the computational complexity of AnyPro, and then validate it on the real-world anycast testbed described in § 3.2. Specifically, we aim to answer the following research questions.

RQ1 How effective is AnyPro in reducing latency? How much gain is introduced by correctly inferring preference constraints?

RQ2 Does the optimization objective function value align well with latency? Does AnyPro correctly infer preference constraints?

RQ3 What are the computational and operational complexity of AnyPro? What are the actual overhead and flexibility in production networks?

RQ4 How does AnyPro's optimization efficacy scale with varying numbers of enabled PoPs?

4.1 Overall Performance (RQ1)

4.1.1 Experiment Setup.

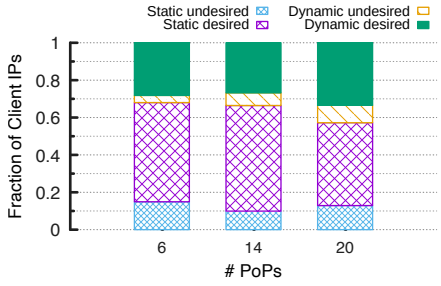
As outlined in § 3.4, AnyPro requires desired client-to-ingress mappings as input, which are derived by operators from historical data and application-specific requirements; for our evaluation, we use geographical proximity as the primary mapping criterion to approximate latency, and we quantify anycast performance by comparing observed mappings against these geo-optimal ones. To ensure the correctness of the observed client-ingress mappings, consecutive ASPP adjustments are spaced 10 minutes apart to allow the ASPP updates to fully propagate and stabilize across the global routing table before initiating ICMP probes. This duration is consistent with established inter-domain routing studies [16, 43], ensuring that we capture the steady-state routing behavior rather than transient path-hunting effects. We specify $MAX = 9$ as our practical upper bound for prepending, a value informed by prior studies [27] and our empirical observations that transit providers commonly accept AS-path lengths up to this threshold without filtering.

Baselines. We compare the following schemes with AnyPro.

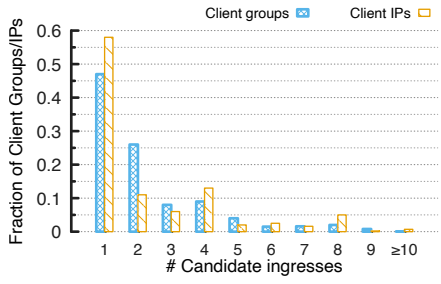
- **All-0**: In this configuration, we enable all available ingress points without AS-path prepending (i.e., $\forall i, j, s_{i,j} = 0$).
- **AnyOpt**: We additionally implement AnyOpt [43], an anycast optimization approach that operates by selectively enabling/disabling PoP subsets.
- **AnyPro (Preliminary)**: We derive preference constraints from the output of *max-min polling* as described in §3.4, obtaining only preliminary constraints since we omit the resolution of the contradictions. From these constraints, we generate a preliminary ASPP configuration where each ingress's ASPP length is either 0 or 9.
- **AnyPro (Finalized)**: After applying contradiction resolution using the method described in §3.5, we obtain the finalized preference constraints. In the resulting configuration, the ASPP of each ingress is selected from the discrete set $\{0, 1, \cdots, 9\}$.

Metrics. We evaluate the optimization using both **RTT** and **normalized objective**, where **normalized objective** is calculated as $\frac{\sum_{c_k \in \boldsymbol{C}} \sum_{p_{i,j} \in \boldsymbol{P}} \boldsymbol{M}^*_{c_k,p_{i,j}} \cdot \boldsymbol{M}_{c_k,p_{i,j}}}{|C| \cdot |P|}$. Since $|C|$ and $|P|$ are constants, this metric is essentially the optimization objective function itself, and we further demonstrate the correlation between these two metrics. A normalized objective value closer to 1 indicates that the observed mappings are more closely aligned with the desired ones.

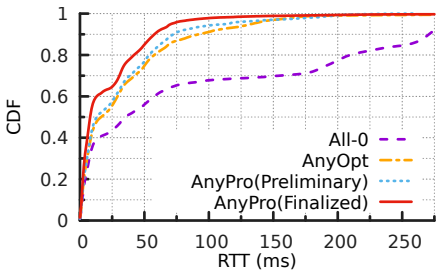
Our experiment first quantifies the client rerouting potential through ASPP by measuring how many clients can reach their desired ingresses. Figure 6(a) presents the results of three anycast deployments. In the 20-PoP configuration, 57.2% of clients maintain stable catchment PoPs during *max-min polling*, with 44.3% reaching desired ingresses (static desired) and 12.9% diverted to undesired ones (static undesired).



(a) The proportion of clients exhibiting divergent reactions to the ASPP.



(b) The distribution of client (groups) by their number of candidate ingresses.



(c) The accuracy in predicting client accessibility to their desired PoPs.

Figure 6: (a) AnyPro can optimize performance for a substantial portion of clients. (b) Most clients and client groups are associated with only a single candidate ingress. (c) AnyPro-optimized configuration substantially outperforms other approaches in terms of RTT.

Table 1: Normalized objective of the optimized anycast system across different methods, both with peers (w/ peer) and without peers (w/o peer).

Method	Normalized Objective	
	w/o peer	w/ peer
All-0	0.60	0.68
AnyOpt	0.66	0.76
AnyPro (Preliminary)	0.72	0.82
AnyPro (Finalized)	0.76	0.85

Among the 42.8% experiencing ingress shifts, 30.7% ultimately connect to desired ingresses (dynamic desired) while 9.3% do not (dynamic undesired). This yields a 77.8% total normalized objective (static desired + dynamic desired).

Table 1 presents the normalized objective for AnyPro-optimized configurations. The left column shows normalized objective values excluding clients connected via peering links. For the All-0 deployment, the peer-inclusive configuration demonstrates a 0.08 lower normalized objective than the transit-only version. Since both configurations consider desired ingresses, this result suggests that substantial intercontinental routing occurs when using transit-only paths. Notably, AnyPro achieves competitive performance even without finalized preference constraints — its normalized objective is merely 0.05 below "AnyPro (Finalized)". This effectiveness stems from two key factors: (1) Figure 6(b) reveals that most clients face $\leq$2 candidate ingresses, meaning few constraints are needed to ensure desired ingress connectivity, and (2) AnyPro's optimization strategically prioritizes high-weight constraints during contradiction resolution, preferentially serving the majority client base to maximize overall normalized objective.

We further assess the RTT of AnyPro-optimized configurations against alternative deployments. Figure 6(c) presents the CDF of client RTTs across different setups, showing that "AnyPro (Finalized)" achieves significantly reduced tail latency compared to All-0. The 90th percentile RTT improves from 271.2 ms (All-0) to 58.0 ms ("AnyPro (Finalized)").

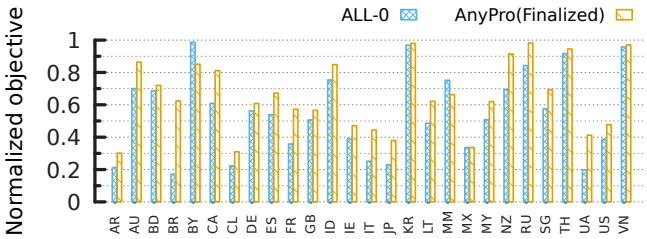
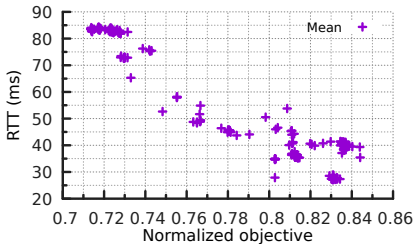


Figure 7: AnyPro-optimized configuration can achieve high normalized objective for most countries simultaneously.

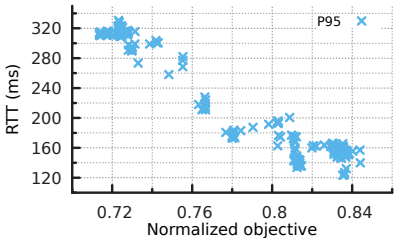
This optimal performance stems from the two-stage optimization: AnyOpt first selects an optimal PoP subset, eliminating poorly-performing nodes, and AnyPro then fine-tunes ASPP values within this subset to precisely steer clients to the lowest-latency ingresses. These dramatic reductions demonstrate the ability of AnyPro, especially when combined with AnyOpt, to effectively leverage anycast routing for minimal latency.

To better quantify the effectiveness of AnyPro, we analyze country-level normalized objective improvements across AnyPro and AnyOpt deployments, focusing on the 27 countries with the highest transit-connected client populations. Figure 7 reveals that AnyPro optimization enhances normalized objective for most countries, with Brazil showing the most dramatic improvement — increasing from 0.17 to 0.62 under the "AnyPro (Finalized)" configuration. Our trace analysis indicates this stems from shifting Brazilian clients from Bangkok to Ashburn PoPs. The sole exception is Myanmar (MM bar), where normalized objective decreases as its lower traffic volume leads to deprioritization during constraint resolution, redirecting more than 10% of Ho Chi Minh clients to European PoPs.

While effective, AnyPro faces limitations when handling contradictory constraints during optimization. The system prioritizes higher-weight client groups at the expense of smaller populations, potentially routing them to suboptimal PoPs. For instance, two competing constraints emerge: (1) 1,388 U.S. clients require $s_{(\text{Frankfurt,Telia})} \geq s_{(\text{Ashburn,Level3})} + 9 \geq 9$ to reach Ashburn-Level3, while (2) 467 German clients need



(a) Matching accuracy vs. mean RTT.



(b) Matching accuracy vs. 95th percentile RTT.

Figure 8: Correlation Analysis between AnyPro 's optimization objective (matching accuracy) and RTT performance.

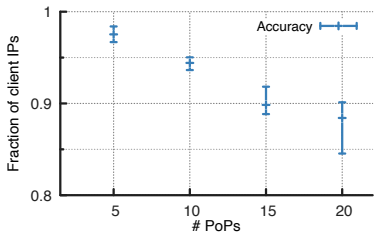


Figure 9: The preference-preserving constraints derived by AnyPro are both complete and accurate.

$s_{(\text{India, Airtel})} \geq s_{(\text{Frankfurt,Telia})} + 9$. Given $s_{(\text{India, Airtel})} \in [0,9]$, the system sets $s_{(\text{Frankfurt, Telia})} = 0$, forcing the German clients to connect to undesired PoPs. This demonstrates how weight-based prioritization can disadvantage smaller client groups despite their legitimate routing preferences.

Takeaways. Our results demonstrate that ASPP-based optimization of IP anycast is both practical and effective. The AnyPro-optimized configuration achieves significant improvements, successfully redirecting the majority of clients to their desired PoPs. Furthermore, performance gains can be substantially enhanced through integration with AnyOpt, combining PoP selection optimization with intelligent path preference configuration to maximize anycast routing efficiency.

4.2 Optimization Effectiveness (RQ2)

4.2.1 Correlation between normalized objective and RTT

To validate the alignment between AnyPro's optimization objective — normalized objective— and actual latency performance, we systematically analyze the relationship between normalized objective and RTT. Note that this analysis is specific to AnyPro's internal configuration space and is intended to verify that maximizing matching accuracy effectively minimizes RTT. As illustrated in Figures 8(a) and 8(b), improvements in normalized objective consistently correspond to reductions in both mean RTT and the 95th-percentile RTT. This consistent inverse relationship is further substantiated through statistical analysis, revealing a strong negative correlation between normalized objective and RTT metrics. The Pearson correlation coefficients are approximately -0.95 for mean RTT and -0.96 for the 95th-percentile RTT[1], underscoring the reliability of this trend. These findings confirm that normalized objective serves as a robust and effective predictor of latency performance in anycast networks, capable of accurately reflecting both average and tail-end latency behavior. Higher normalized objective values are consistently associated with enhanced end-to-end latency characteristics across the entire performance distribution, thereby reinforcing its utility as a key metric for network optimization.

Takeaways: The normalized objective, which serves as the optimization objective, exhibits a strong negative correlation with RTT.

4.2.2 Constraint identification

We evaluate AnyPro's effectiveness in identifying sufficient preference-preserving constraints to address RQ2.

While AnyPro does not predict exact catchments for all possible configurations — as exhaustively determining ingress preference orders would be computationally prohibitive — it efficiently identifies preference-preserving constraints that guarantee client groups reach their desired PoPs by: (1) selecting a random subset of PoPs and disabling others to isolate ASPP-sensitive clients via *max-min polling*; (2) deriving constraints using our tie-breaking method (§3.5); and (3) validating accuracy through testing of 10 random ASPP configurations per deployment and comparing predicted versus observed PoP connections.

Figure 9 demonstrates the prediction accuracy variability across different deployment scales, where we evaluate AnyPro with 5, 10, 15, and 20 enabled PoPs (including all associated transit providers). The results show that AnyPro maintains strong predictive performance, particularly in smaller deployments — achieving over 95% accuracy with 5 enabled PoPs. However, as the number of PoPs increases, we observe a gradual decline in prediction accuracy for client-to-PoP mappings. This degradation primarily stems from two factors: (1) the quadratic growth in unresolved constraint contradictions as more PoPs are added (despite our binary scan-based resolution mechanism), and (2) external complexities such as uncommon BGP policies [43] and multi-path routing effects. Notably, even at 20 PoPs (our largest test case), AnyPro maintains 88.5% accuracy, demonstrating its robustness for real-world anycast networks.

Takeaways. The preference-preserving constraints discovered by AnyPro are effective for determining client accessibility to their desired PoP.

[1]The Pearson correlation coefficient ranges from -1 to 1, where values approaching -1 indicate a strong inverse linear relationship.

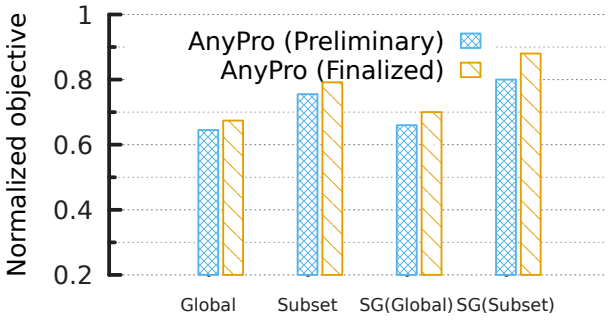


Figure 10: Subset optimization can yield greater performance for certain clients.

4.3 Complexity Analysis (RQ3)

The initial *max-min polling* pass over the n ingresses incurs a cost of only $O(n)$. For $|\Xi|$ contradictions, the resolution process performs at most $|\Xi|$ binary scans, each requiring $O(\log m)$ time, resulting in a total complexity of $O(|\Xi|\log m)$. Overall, `AnyPro` completes in $O(n+|\Xi|\log m)$ time, which is optimal and several orders of magnitude more efficient than the naïve $O(m^n)$ approach. In our large-scale deployment scenario — where $m = 10$, $n = 38$, and approximately 2.4 million IP addresses are considered — the system identifies 513 distinct preliminary constraints and resolves all contradictions with only 84 ASPP adjustments. Combined with the 76 ASPP adjustments from the initial *max-min polling* process (i.e., 38×2), a full optimization cycle requires a total of 160 ASPP adjustments. This low-frequency, targeted approach ensures that `AnyPro` scales to large WANs without compromising Internet control-plane stability. With each adjustment taking 10 minutes to ensure convergence, the total computational time per cycle amounts to 26.6 hours — a significant reduction from the 190 hours required by AnyOPT [43]. To validate the robustness of our derived constraints over the 26.6-hour window, we randomly sampled 50 non-contradiction preference constraints generated during the max-min polling, and derived a ASPP configurations to satisfy them simultaneously. 48 hours later, we reapplied the configurations. By measuring the resulting client-ingress mappings again, we observed that 99.2% of the mappings remained identical to the initial results and the 50 constraints still hold. This high degree of consistency indicates that inter-domain routing policies and path preferences exhibit significant persistence over multi-day periods, confirming the consistency of global constraints during each round, guaranteeing the effectiveness of `AnyPro`.

4.4 Subset Optimization (RQ4)

Our prior analysis reveals that unresolved constraint contradictions disproportionately impact clients in low-traffic regions, reflecting a common operational practice that prioritizes service quality in specific critical areas over global optimization. To mitigate this imbalance, we propose to selectively deploy `AnyPro` across curated subsets of PoPs, allowing targeted optimization for priority client groups. We demonstrate that the optimization framework of `AnyPro` is particularly well-suited for several practical scenarios: (1) regionally constrained services, (2) regional IP anycast implementations [44], and (3) mitigation of temporary ingress outages. Using Southeast Asia as a case study, we activate six regional PoPs — specifically in Malaysia, Manila, Ho Chi Minh City, Singapore, Indonesia, and Bangkok — along with their transit links, while disabling all others. This configuration creates an isolated test environment in which we rigorously evaluate the efficacy of localized optimization.

Figure 10 compares normalized objective values for Southeast Asian countries under both global and localized optimization configurations. The leftmost bars present results from global optimization — where all ingresses are enabled — showing that these countries achieve a relatively lower normalized objective due to their reduced priority in worldwide routing policies. In contrast, localized subset optimization increases the overall normalized objective from 0.67 to 0.78, representing a 16.4% improvement. Country-level analysis reveals that Singapore benefits most significantly, with its normalized objective rising from 0.70 to 0.88 — a 25.7% gain. A detailed analysis indicates that under global optimization, 16.6% of Singapore clients are misrouted (12.5% to U.S. and 4.1% to Europe), whereas subset optimization successfully directs all traffic to local Singapore PoPs.

Takeaways: Localized optimization in Southeast Asia significantly improves routing accuracy, demonstrating `AnyPro`'s capability for region-specific tuning. This is exemplified by Singapore's 25.7% normalized objective improvement after eliminating all transcontinental misroutes. The results confirm that `AnyPro` effectively enforces geographic prioritization while preserving the benefits of anycast.

5 Discussion

This section critically examines the inherent limitations of `AnyPro` and proposes promising avenues for future research to address these constraints.

Why not Traceroute? While traceroute-derived AS-path information could theoretically inform route optimization, a naive approach of comparing AS paths to get the finalized preference constraints faces two critical limitations. First, experimental data indicates that traceroute-collected path information often lacks completeness, particularly in documenting intermediate hops. This makes it difficult to characterize the complete AS path for optimization purposes. Second, certain ISPs implement BGP regular expression filters that dynamically truncate excessive route prepending — for instance, observed cases where 9× is compressed to 3×Prepending - rendering direct AS-path length comparisons invalid, as the ISP's real-time processing of ASPP remains opaque. Our methodology overcomes these challenges by employing em-

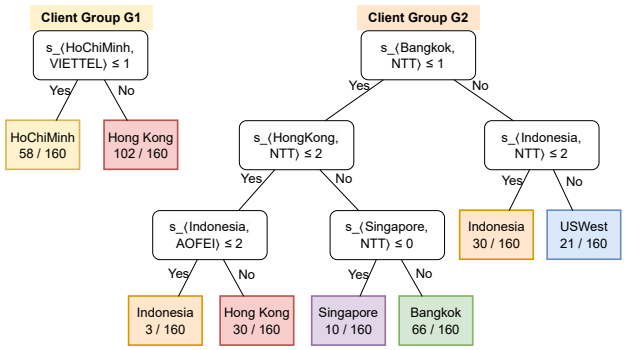


Figure 11: Instability of decision tree models during catchment prediction.

pirical performance evaluation (via max-min polling and binary scan) to derive both preliminary and finalized constraints. This approach ensures reliable client-side routing predictability independent of opaque ISP ASPP handling mechanisms.

Data-driven catchment modeling and inference. `AnyPro` employs a deterministic approach for catchment modeling, inference, and optimization. While data-driven methods like machine learning (ML) may appear appealing, we demonstrate that ML catchment inference proves fundamentally unreliable and inefficient. As an illustrative case, we trained decision tree models using 160 random ASPP configurations and their corresponding client-ingress mappings as training data. The models take prepending length arrays as input and predict client-ingress relationships. Figure 11 visualizes the resulting tree structures for two representative client groups: G1 in Vietnam (2 candidate ingresses) and G2 in Indonesia (6 candidate ingresses), represented as leaves in their respective trees. This exercise reveals the inherent limitations of probabilistic approaches compared to our deterministic methodology.

The decision tree for G1 predicts with 100% confidence that clients enter Ho Chi Minh City when $s_{(\text{HoChiMinh,VIETTEL})} \leq 1$, and Hong Kong otherwise. However, this proves unreliable when tested with $s_{(\text{HoChiMinh,VIETTEL})} = 1$ and $s_{(\text{HongKong, NTT})} = 0$, where clients undesiredly connect to Hong Kong. These inconsistencies worsen for complex cases like G2 (with 6 candidate ingresses), and frequently occur when clients ingress through PoPs absent from training data. We identify two root causes: (1) BGP policies are fundamentally deterministic and ill-suited to probabilistic modeling, and (2) random ASPP configurations fail to capture sensitivity and constraint contexts. `AnyPro` overcomes these limitations by systematically discovering all possible ingresses and their constraints, guaranteeing optimal configurations.

More advanced machine learning methods, such as graph neural networks (GNNs) [15] that model global correlations in evolutionary optimization [31] and hybrid CNN–GRU–GA pipelines for structured feature learning and parameter search [18], demonstrate that graph-structured learning can effectively capture complex network dependencies. While they may exceed the decision tree, and potentially approximate the optimal, they cannot match `AnyPro`'s efficiency, as every configuration exploration in `AnyPro` provides essential information.

Comparison with Alternative BGP Controls. While BGP communities and announcement scoping (e.g., restricting route propagation to specific regions) offer alternative anycast steering, they present challenges in granularity and operational consistency. Unlike binary scoping, which is limited to discrete PoP-level advertisements, `AnyPro` leverages the scalar nature of ASPP (prepending $1 \ldots 9$ times)) to enable *fine-grained* steering of catchment boundaries without sacrificing global reachability. Furthermore, while we acknowledge that higher-priority policies like `Local-Pref` (often set via communities) can override AS-path length, `AnyPro` is specifically designed to empirically isolate *ASPP-sensitive* clients through max-min polling (§ 3.4). Clients behind ISPs with rigid, community-driven policies that mask prepending effects are naturally identified as non-sensitive and excluded from the optimization variables. This focus on the universal AS-path attribute avoids the operational complexity of non-standardized communities across heterogeneous ISPs, though integrating community-based controls for anomaly detection remains a promising direction for future work.

Peering connections. Our anycast network leverages both transit and peering connections, which differ fundamentally in their routing characteristics and performance impacts. While prior work typically separates peer optimization from transit configuration (e.g., AnyOpt's sequential approach), we adopt a distinct strategy by enabling all peering connections before transit optimization. This design reflects two key considerations: (1) Performance benefits — despite known risks of remote peering [8, 30], most peering paths in practice offer lower latency than transit alternatives, and (2) Relationship preservation — frequent prefix announcement changes may violate peering agreements that stipulate route stability. Our approach thus maximizes potential performance gains while maintaining critical peering relationships.

6 Related works

Anycast catchment measurement and inference. Verfploeter [13] supports inferring any clients' catchment sites as long as the clients are capable of responding to ICMP requests. Leveraging this capacity, researchers have extended its utility to assess client-specific metrics such as RTTs, traceroutes, and ingresses to anycast PoPs [1, 12, 37, 43]. Another research approach utilizes RIPE Atlas to send active probes towards an anycast address [7, 11, 25, 39]. For instance, Zhou *et al.* [44] infer the anycast catchment by interpreting the penultimate hop from traceroute results, which typically corresponds to the on-site routers situated at the same locale as the anycast PoPs. Despite its wide utility in research, the RIPE Atlas method is constrained by the limited availability of probes. A related research line focuses on inferring catchments for In-

ternet routing [35, 36]. Sermpezis *et al.* [35] suggest utilizing inferred AS-level Internet topology as a means of predicting catchments. However, this methodology does not effectively scale to the magnitudes of anycast networks and cannot guarantee accurate inferences at such scales. Singh *et al.* [36] train a probabilistic model for Internet routing by feeding a corpus of traceroutes. While insightful, this approach also falls short when applied to a practical anycast networks.

Anycast catchment optimization. Existing catchment optimization can be broadly classified into two categories: PoP-level [16,25,33,43,44] and ingress-level [2,4,28] optimization. (1) For the former, Zhang *et al.* [43] construct the total preference order of anycast sites for each client. This strategy helps forecast anycast catchments and effectively minimizes RTT, at the expense of time-consuming measurements. Zhou *et al.* [44] deeply explore the potential of regional anycast, concluding that it can mitigate path inflation and enhance the performance over global anycast in practice. (2) For the latter, DailyCatch [28] captures the performance shift across varying snapshots, each representing a particular configuration. Ballani *et al.* [4] suggest connecting all anycast PoPs to a single tier-1 provider, leveraging the intra-domain routing for optimal PoP selection. Bertholdo et al. [6] aim to manage anycast catchment by adjusting BGP communities. However, their work primarily showcases the performance of specific configurations without deriving the optimal one. Their approach thus functions more as a monitoring tool rather than an optimization method. To compare, `AnyPro` constructs a white-box model that encapsulates the critical conditions needed for each ASPP-sensitive client to connect to their expected ingress. `AnyPro` has been demonstrated to be effective across any subset of ingresses so that it can work well with ingress-level optimization.

7 Conclusion

This paper introduces `AnyPro`, the first production-ready system that optimizes anycast catchment efficiency through strategic AS-path prepending configuration tuning. `AnyPro` efficiently characterizes global catchments and performs *max-min polling* to identify ASPP-sensitive clients and derives preliminary routing preference constraints. If there arises contradiction among constraints, it conducts iterative contradiction resolution and optimization solving to obtain the optimal global ASPP configuration. Evaluations on a production testbed confirm the effectiveness of `AnyPro` with significant latency improvements.

8 Acknowledgment

Minyuan Zhou, Yuning Chen and Yifei Xu were supported by Alibaba Group through Alibaba Research Intern Program. We also sincerely thank our shepherd Oliver Hohlfeld and all anonymous reviewers for their valuable feedback.

References

[1] Thomas Alfroy, Thomas Holterbach, and Cristel Pelsser. Mvp: Measuring internet routing from the most valuable points. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'22)*, 2022.

[2] Hussein A. Alzoubi, Seungjoon Lee, Michael Rabinovich, Oliver Spatscheck, and Jacobus Van Der Merwe. A practical architecture for an anycast cdn. *ACM Trans. Web*, 5(4), 2011.

[3] Arelion. Telia BGP communities. Retrieved in Apr, 2024 from https://www.arelion.com/our-network/bgp-routing/bgp-communities.

[4] Hitesh Ballani and Paul Francis. Towards a Global IP Anycast Service. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'05)*, 2005.

[5] Hitesh Ballani, Paul Francis, and Sylvia Ratnasamy. A Measurement-Based Deployment Proposal for IP Anycast. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'06)*, 2006.

[6] Leandro M Bertholdo, Joao M Ceron, Lisandro Z Granville, Giovane CM Moura, Cristian Hesselman, and Roland van Rijswijk-Deij. Bgp anycast tuner: Intuitive route management for anycast services. In *International Conference on Network and Service Management (CNSM'20)*. IEEE, 2020.

[7] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitu Padhye. Analyzing the Performance of an Anycast CDN. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'15)*, 2015.

[8] Ignacio Castro, Juan Camilo Cardona, Sergey Gorinsky, and Pierre Francois. Remote Peering: More Peering without Internet Flattening. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies (CoNEXT'14)*, 2014.

[9] Cisco. BGP Attributes and Path Selection. Retrieved in Dec, 2024 from https://networklessons.com/bgp/bgp-attributes-and-path-selection.

[10] Cogent. Cogent Customer User Guide. Retrieved in Apr, 2024 from https://cogentco.com/files/docs/customer_service/guide/global_cogent_customer_user_guide.pdf.

[11] Ricardo de O. Schmidt, John Heidemann, and Jan Kuipers. Anycast Latency: How Many Sites Are Enough? In *Proceedings of Passive and Active Measurement (PAM'17)*, 2017.

[12] Wouter B de Vries, Salmān Aljammāz, and Roland van Rijswijk-Deij. Global-Scale Anycast Network Management with Verfploeter. In *Proceedings of 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS'20)*, 2020.

[13] Wouter B. De Vries, Ricardo de Oliveira Schmidt, Wes Hardaker, John Heidemann, Pieter-Tjerk de Boer, and Aiko Pras. Broad and Load-Aware Anycast Mapping with Verfploeter. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'17)*, 2017.

[14] Xun Fan and John Heidemann. Selecting representative ip addresses for internet topology studies. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'10)*, 2010.

[15] Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Krzysztof Rusek, Shihan Xiao, Xiang Shi, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Routenet-fermi: Network modeling with graph neural networks. *IEEE/ACM transactions on networking*, 31(6):3080–3095, 2023.

[16] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. FastRoute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *Proceedings of 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI'15)*, 2015.

[17] Ruomei Gao, Constantinos Dovrolis, and Ellen W. Zegura. Interdomain ingress traffic engineering through optimized as-path prepending. In *Proceedings of the 4th IFIP-TC6 International Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems*, page 647–658, 2005.

[18] Zong Ke, Jiaqing Shen, Xuanyi Zhao, Xinghao Fu, Yang Wang, Zichao Li, Lingjie Liu, and Huailing Mu. A stable technical feature with gru-cnn-ga fusion. *Applied Soft Computing*, page 114302, 2025.

[19] Thomas Koch, Ethan Katz-Bassett, John Heidemann, Matt Calder, Calvin Ardi, and Ke Li. Anycast in context: A tale of two systems. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'21)*, pages 398–417, 2021.

[20] Thomas Koch, Shuyue Yu, Sharad Agarwal, Ryan Beckett, and Ethan Katz-Bassett. Painter: Ingress traffic engineering and routing for enterprise cloud networks. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'23)*, 2023.

[21] Thomas Krenc, Robert Beverly, and Georgios Smaragdakis. AS-Level BGP Community Usage Classification. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'21)*, 2021.

[22] Thomas Krenc, Matthew Luckie, Alexander Marder, and kc claffy. Coarse-grained inference of bgp community intent. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'23)*, pages 66–72, 2023.

[23] Tony Li, Ravi Chandra, and Paul S. Traina. Bgp communities attribute,. RFC 1997, RFC Editor, 1996.

[24] Weitong Li, Zhexiao Lin, Md Ishtiaq Ashiq, Emile Aben, Romain Fontugne, Amreesh Phokeer, and Taejoong Chung. Rovista: Measuring and analyzing the route origin validation (rov) in rpki. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'23)*, pages 73–88, 2023.

[25] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Internet Anycast: Performance, Problems, & Potential. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'18)*, 2018.

[26] Ziqian Liu, Bradley Huffaker, Marina Fomenkov, Nevil Brownlee, and kc claffy. Two days in the life of the dns anycast root servers. In Steve Uhlig, Konstantina Papagiannaki, and Olivier Bonaventure, editors, *Proceedings of Passive and Active Measurement (PAM'07)*, 2007.

[27] Pedro Marcos, Lars Prehn, Lucas Leal, Alberto Dainotti, Anja Feldmann, and Marinho Barcellos. As-path prepending: there is no rose without a thorn. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'20)*, pages 506–520, 2020.

[28] Stephen McQuistin, Sree Priyanka Uppu, and Marcel Flores. Taming Anycast in the Wild Internet. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'19)*, 2019.

[29] Giovane C.M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Muller, Lan Wei, and Cristian Hesselman. Anycast vs. ddos: Evaluating the november 2015 root dns event. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'16)*, 2016.

[30] George Nomikos, Vasileios Kotronis, Pavlos Sermpezis, Petros Gigis, Lefteris Manassakis, Christoph Dietzel, Stavros Konstantaras, Xenofontas Dimitropoulos, and Vasileios Giotsas. O Peer, Where Art Thou?: Uncovering Remote Peering Interconnections at IXPs. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'18)*, 2018.

[31] Kaichen Ouyang, Zong Ke, Shengwei Fu, Lingjie Liu, Puning Zhao, and Dayu Hu. Learn from global correlations: Enhancing evolutionary algorithm via spectral gnn. *arXiv preprint arXiv:2412.17629*, 2024.

[32] Laurent Perron and Vincent Furnon. Or-tools.

[33] Sandeep Sarat, Vasileios Pappas, and Andreas Terzis. On the Use of Anycast in DNS. In *Proceedings of the 2005 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'06)*, 2006.

[34] Kyle Schomp, Onkar Bhardwaj, Eymen Kurdoglu, Mashooq Muhaimen, and Ramesh K. Sitaraman. Akamai dns: Providing authoritative answers to the world's queries. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'20)*, 2020.

[35] Pavlos Sermpezis and Vasileios Kotronis. Inferring Catchment in Internet Routing. *ACM Meas. Anal. Comput. Syst.*, 3(2), 2019.

[36] Rachee Singh, David Tench, Phillipa Gill, and Andrew McGregor. Predictroute: A network path prediction toolkit. *Proc. ACM Meas. Anal. Comput. Syst.*, 5(2), 2021.

[37] R. Sommese, L. Bertholdo, G. Akiwate, M. Jonker, R. van Rijswijk-Deij, A. Dainotti, k. claffy, and A. Sperotto. MAnycast^2 - Using Anycast to Measure Anycast. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'20)*, 2020.

[38] Raffaele Sommese, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, Roland van Rijswijk-Deij, Alberto Dainotti, KC Claffy, and Anna Sperotto. Manycast2: Using anycast to measure anycast. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'20)*, pages 456–463, 2020.

[39] Lan Wei and John Heidemann. Does Anycast Hang Up on You? In *Proceedings of the Network Traffic Measurement and Analysis Conference (TMA'17)*, 2017.

[40] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[41] Yifei Xu, Yuning Chen, Xumiao Zhang, Xianshang Lin, Pan Hu, Yunfei Ma, Songwu Lu, Wan Du, Z Morley Mao, Ennan Zhai, et al. Cloudeval-yaml: A realistic and scalable benchmark for cloud configuration generation. 2023.

[42] Yifei Xu, Yuning Chen, Xumiao Zhang, Xianshang Lin, Pan Hu, Yunfei Ma, Songwu Lu, Wan Du, Zhuoqing Mao, Ennan Zhai, et al. Cloudeval-yaml: A practical benchmark for cloud configuration generation. *Proceedings of Machine Learning and Systems*, 6:173–195, 2024.

[43] Xiao Zhang, Tanmoy Sen, Zheyuan Zhang, Tim April, Balakrishnan Chandrasekaran, David Choffnes, Bruce M Maggs, Haiying Shen, Ramesh K Sitaraman, and Xiaowei Yang. Anyopt: Predicting and optimizing ip anycast performance. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'21)*, pages 447–462, 2021.

[44] Minyuan Zhou, Xiao Zhang, Shuai Hao, Xiaowei Yang, Jiaqi Zheng, Guihai Chen, and Wanchun Dou. Regional IP Anycast: Deployments, Performance, and Potentials. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'23)*, 2023.

[45] Jiangchen Zhu, Kevin Vermeulen, Italo Cunha, Ethan Katz-Bassett, and Matt Calder. The best of both worlds: high availability cdn routing without compromising control. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'22)*, 2022.

Appendix

A Ethical Considerations.

We issue the ICMP requests at a reasonably low rate to avoid additional load on the Internet infrastructure. Additionally, our BGP announcements are restricted to prefixes used for testing and using our own AS number. In system logs, only IP, timestamps, and ingresses are parsed for analysis. User information is not included. This work raises no other ethical issues.

B Infrastructure

Table 2 presents the PoPs along with their respective transit providers used in our study.

Malaysia	NTT_2914, AIMS_24218
Madrid	TATA_6453
Manila	PLDT-iGate_9299, Globe_4775
Hong Kong	PCCW_3491, NTT_2914
Seoul	SKB_9318, TATA_6453
Vancouver	TATA_6453
Ashburn	Level3_3356, Cogent_174
Moscow	Rostelecom_12389, Megafon_31133
Chicago	CenturyLink_3356, Cogent_174
Ho Chi Minh	VIETTEL_7552, CMC_45903
California	NTT_2914, TATA_6453
Frankfurt	Telia_1299, TATA_6453
Bangkok	TATA_6453, TrueIntl.Gateway_38082
Singapore	Singtel_7473, TATA_6453, PCCW_3491
Sydney	Telstra_4637, Optus_7474
Toronto	TATA_6453
India	TATA_4755, Airtel_9498
Indonesia	NTT_2914, AOFEI_135391
London	TATA_4755, Telia_1299
Tokyo	NTT_2914, SoftBank_17676

Table 2: Transits and ASNs in the testbed for each PoP. Some are listed after a city while others are after a country.

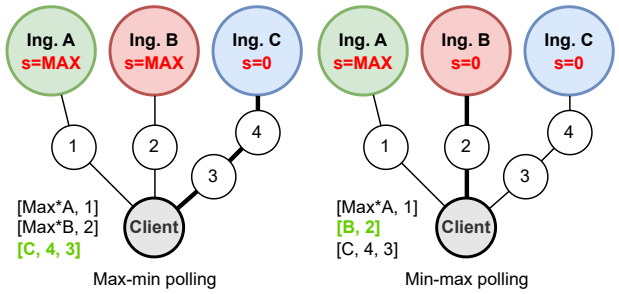


Figure 12: Difference between *max-min polling* and *min-max polling*. *min-max polling* will not explore the path from C for the client because there will always be a shorter path available, either from A or B.

C Why not Min-Max Polling

The *max-min polling* offers a clear method to analyze the individual ingress's effect on traffic catchment areas. Figure 12 illustrates the rationale behind choosing a *max-min polling* instead of a *min-max polling* (*i.e.*, initially prepending zero to all route announcements and then sequentially reverting to *Max* for all ingresss one by one). Specifically, consider a client that receives routing information from three different PoPs and AS path length is the most important route selection criterion. When prepending sizes are zero, routes from A and B always have shorter AS paths than that from C. Thus, in the case of *min-max polling*, path originating from C would never be selected by the client because there will always be a preferable route available via either A or B. However, by employing *max-min polling*, we are able to explore the scenario where the client might select the path from C when routes from A and B both prepend 3 ASes.

D Reduction from Max-SAT to ASPP Optimization

In this section, we show why and how we reduce NP-hard max-SAT problem into the ASPP optimization.

Taking another look into preference preserving constraints in the formal manner, we can express them as clauses of boolean values. The preference preserving constraints can be formally expressed as clauses of boolean values. The preference preserving of each client is a clause, consisting of booleans connected with *conjunction* logic. Each boolean is a pairwise preference preserving constraint, which is an inequation involving two prepending sizes. We use conjuction logic because for clients with more than two candidate ingresses, to guarantee matching, several inequation constraints need to be satisfied simultaneously, so that the preferred ingress wins over every other ingresses. To this end, it follows the nature of MAX-SAT problem, which we try to satisfy as much clauses as possible.

Proof. Consider an arbitrary Max-SAT instance given by a CNF formula

$$\Phi = \bigwedge_{k=1}^{m} C_k, \quad \text{with } C_k = \bigvee_{j=1}^{r_k} \ell_{k,j}.$$

For each clause C_k, create a client c_k; for each literal $\ell_{k,j}$ in C_k, define a candidate preference $p_{k,j}$ with weight

$$\mathbf{M}^*_{c_k,p_{k,j}} = 1.$$

Formulate the ASPP optimization instance as

$$\text{maximize} \sum_{c_k \in \boldsymbol{C}} \sum_{p_{i,j} \in \boldsymbol{P}} \boldsymbol{M}^*_{c_k,p_{i,j}} \cdot X_{c_k,p_{i,j}} \tag{1}$$

$$\text{s.t.} \quad \sum_{p_{i,j} \in \boldsymbol{P}} X_{c_k,p_{i,j}} = 1, \quad \forall c_k \in \boldsymbol{C} \tag{1a}$$

$$X_{c_k,p_{i,j}} \in \{0,1\}, \quad \forall c_k \in \boldsymbol{C}, \forall p_{i,j} \in \boldsymbol{P} \tag{1b}$$

$$X_{c_k,p_{i,j}} \geq X_{c_k,p_{m,n}}, \quad \forall p_{i,j} \succ_{c_k,\boldsymbol{S}} p_{m,n} \tag{1c}$$

$$s_{i,j} \in \{0,1,\ldots,MAX\}, \quad \forall s_{i,j} \in \boldsymbol{S} \tag{1d}$$

Considering each constraint satisfaction as a clause C_k, and reduce the exactly logical negations of each other as $\neg C_k$. Then any solution to this ASPP instance corresponds to a truth assignment for Φ that satisfies as many clauses as possible. Since the reduction is performed in polynomial time and Max-SAT is NP-hard, ASPP optimization is NP-hard. □