修士論文

OSPF ベースの AS 内経路最適化に向けたデジタルツインによる経路制御支援

西村 優典

主指導教員 宇多 仁

北陸先端科学技術大学院大学
先端科学技術専攻
（情報科学）

令和 8 年 3 月

Abstract

In large-scale network environments, primarily autonomous systems (AS), automation is achieved using IGPs such as OSPF. In recent years, management has become increasingly complex, requiring high reliability in its operation.

By default, OSPF link costs are calculated based on the bandwidth of each interface. However, in real networks, there are many situations where operators need to reflect their own intentions, such as "bypassing a specific node for maintenance" or "prioritizing a specific link," from a traffic engineering perspective.

Currently, achieving these intentions relies on trial and error based on operator experience, requiring significant effort. Furthermore, changing a single link cost can affect the traffic distribution ratio across the entire network, carrying a significant risk of unintended traffic concentrations or communication outages. It is also difficult to analyze the cause of problems when they occur or to reproduce past conditions for verification. Network simulators and emulators are commonly used as existing verification methods. However, these often rely on fixed configuration information. As a result, there are limitations to faithfully reproducing and predicting real-time, large-scale traffic distribution flowing within an actual network.

To address these challenges, this study proposes the creation of a "digital twin" environment that faithfully recreates an identical network in a virtual space based on dynamic operational data from the actual network. This method enables "what-if analysis," which allows for proactive and quantitative verification of the impact of configuration changes on the entire network without affecting the physical environment. It also accepts abstract operator intent as input and converts it into link costs, aiming to enable verification and prediction of their impact in the digital twin.

The proposed system consists of four independent modules, each with its own function, to reflect the dynamic changes of the real network and realize simulations that reflect operational intent.

The first module, the "Collector Module," uses SNMP to obtain OSPF link state databases (LSDBs) from actual routers. The obtained data is used to generate a network topology based on graph theory. Other methods estimate neighbor relationships by collecting interface information (iftables) and ARP tables from each router. However, our method reconstructs a detailed L3 topology, including neighbor relationships, metrics, and multi-access segments, which these methods cannot capture.

The second module, the "Processor Module," normalizes traffic data obtained from NetFlow/sFlow. Destination and source addresses in the flow information

are mapped to nodes within the topology. If an address is not within the topology, it is determined that the communication is with an AS outside the AS, and that address is mapped to an ASBR.

The third module, the "Simulator Module," performs shortest path calculations (SPF) based on NetrkX to reproduce OSPF behavior. It supports traffic superposition and ECMP modeling, calculating the load on each link and recreating the current network state in a virtual space.

The fourth module, the "Visualizer Module," visualizes the results of the calculations made so far as a heat map using logarithmic normalization. This intuitively presents network congestion, potential bottlenecks, and changes in traffic flow over time.

The core result of this research is the implementation of an "intent resolver" that allows operators to achieve their goals without complex cost calculations, and injects threads into the digital twin. Operators only need to declaratively describe abstract threads, such as "what state they want to achieve," rather than "how to configure it."

This system defines six types of operational intents to be resolved. First, as intents related to structural constraints, we implemented "Node Drain" for hosting and "Link Blocking" for physical disconnection. Next, as intents related to route guidance, we implemented "Low Latency Guidance," which prioritizes specific paths while minimizing side effects, and "Load Balancing," which automatically adjusts cumulative costs to establish ECMP. Furthermore, we implemented "Flow Reduction," which gradually adjusts flow rates, and "Safety Range Analysis," which calculates the boundary value below which configuration changes do not affect other routes. These algorithms apply the theory of sensitivity analysis in shortest path problems, enabling the calculation of configuration values ⊠⊠ based on mathematical evidence.

We also focused on the order in which the derived link costs are set to ensure the safety of network operations. If cost changes are applied in the wrong order, temporary routing loops, black holes, and other failures may occur during OSPF convergence. This research proposes the automatic generation of "safe sequences" to minimize these risks. First, a route to which traffic can escape is established by performing a cost-decreasing operation. After that, traffic is removed from the original route by performing a cost-increasing operation. In this way, the sequence is determined automatically. This maximizes availability during large-scale configuration change processes and reduces the risk of service interruptions caused by human error.

To verify the effectiveness of the proposed method, we evaluated it using actual data from the WIDE project's backbone network. In verifying the comprehen-

siveness of topology construction, the proposed LSDB analysis method identified 45 routers and 116 links. This was a significantly higher detection rate than conventional CLI-based and config-based methods. This demonstrated the proposed method's ability to accurately capture configuration errors and dynamic segment changes. The traffic estimation accuracy was compared with actual measurements from the Zabbix operations monitoring system. We confirmed extremely high reproducibility, with an error of 0.08 % at edge nodes. For backbone links, we observed an underestimation of approximately 30 % due to sampling bias in flow information and L2 overhead. However, the relative load relationships were consistent with actual measurements, demonstrating their usefulness for the practical purpose of identifying congestion points.

Furthermore, we confirmed the usefulness of the intent resolver using multiple intents created using data from the WIDE project. In the "load balancing" scenario, we successfully split approximately 963.5 Mbps of traffic concentrated on a specific path equally between two paths, each at 481.76 Mbps, through automatic cost adjustment. In the "flow reduction" scenario, we also identified an optimal intermediate cost value through sensitivity analysis that diverted only bursty traffic while maintaining nearby reachability. These results demonstrate that the proposed method abstracts complex metric calculations and dramatically improves operational efficiency and safety.

This research has demonstrated that intent-based control using digital twins will be an extremely effective foundation for next-generation network operation automation. In the future, we plan to introduce a statistical complementary model for traffic sampling, integrate it with other routing protocols, and expand it to multi-area OSPF.

概要

大規模なネットワーク環境、主に自律システム (AS) では OSPF などの IGP を用いて自動化している。近年ますますその管理の複雑性は増しており、その運用には高い信頼性が求められる。

デフォルトでは OSPF のリンクコストは各インターフェースの帯域幅に基づき算出される。しかし、実際のネットワークにおいて、トラフィックエンジニアリングの観点より「特定のノードをメンテナンスのために迂回させる」や「特定のリンクを優先したい」といった運用者の意図を反映させたい場面が多々ある。

現在、意図の実現には、運用者による経験則に基づく試行錯誤に依存しており、多大な労力えお要する。また、単一のリンクコストの変更がネットワーク全体のトラフィック分散比率に影響を及ぼすこともあるため、意図しないトラフィック集中や通信切断を引き起こす大きなリスクを伴う。問題が発生した場合の原因分析や、過去の状態を再現しての検証も容易ではない。既存の検証手段として、ネットワークシミュレータやエミュレータがよく用いられる。しかし、これらは性的な構成情報に依存することが多い。そのため、実際にネットワーク内を流れるリアルタイムかつ大規模なトラフィック分布を忠実に再現・予測することには限界がある。

本研究ではこれらの課題を解決するため、実際のネットワークの動的な運用データに基づいて仮想空間上に同じネットワークを忠実に再現する「デジタルツイン」環境の構築を提案する。本手法により、物理環境に影響を与えることなく、設定変更のネットワーク全体への影響を事前かつ定量的に検証できる「What-If 分析」を可能にする。また、運用者の抽象的な意図を入力として受け、その意図をリンクコストへ変換。その影響をデジタルツイン上で検証・予測が可能になることを目指す。

提案システムは、実ネットワークの動的な変化を反映し、運用意図を反映したシミュレーションを実現するために、機能ごとに独立した 4 つのモジュールで構成される

第 1 のモジュール「Collector Module」は、SNMP を用いて実機ルータより、OSPF のリンクステートデータベース (LSDB) を取得する。取得したデータを用いて、グラフ理論に基づいてネットワークトポロジを生成する。各ルータのインターフェース情報 (iftable) や ARPtable を収集して隣接関係を推定する手法もある。しかし、本手法ではこれらでは把握がこんな隣接関係や、メトリック、およびマルチアクセスセグメントを含む詳細な L3 トポロジを再構築する。

第 2 のモジュール「Processor Modeule」は、NetFlow/sFlow から取得したトラフィックデータを正規化する。フロー情報にある送信先・元のアドレスをトポロジ内のノードにマッピングする。トポロジ内にないアドレスの場合は AS 外との通信であることがわかるため、そのアドレスに ASBR をマッピングする。

第 3 のモジュール「Simulator Module」は、Net を rkX を基盤とした最短経路

計算（SPF）を行い、OSPF の挙動を再現する。トラフィックの重畳や ECMP のモデル化をサポートする。これにより各リンクへの負荷を算出し、現状のネットワーク状態を仮想空間上に再現する。

大４のモジュール「Visualizer Module」がこれまでの計算結果を対数正規化を用いて、ヒートマップとして可視化する。これによりネットワークの混雑状況や潜在的なボトルネック、時系列でのトラフィクの流れ方の変化を直感的に提示する。

本研究の中核的な成果は、運用者が複雑なコスト計算を行うことなく目標を達成できる「インテント・リゾルバ」の実装を行い、デジタルツインに糸を注入することである。運用者は「どのように設定するか」ではなく、「どのような状態にしたいか」という抽象的な糸を宣言的に記述するだけで良い。

本システムは、解決する対象として６種類の運用意図を定義した。まず、構造的制約に関わる意図として、ほすのための「Node Drain」および物理的断線をもした「リン遮断」を実装した。次に経路誘導に関する意図として、副作用を抑えつつ特定のパスを優先する「低遅延誘導」や、累積コストを自動調整して ECMP を成立させる「負荷分散」を実装した。さらに、流量を段階的に調整する「流量削減」や、設定変更が他の経路に影響を与えない境界値を算出する「安全範囲解析」を実現した。これらのアルゴリズムには、最短経路問題における感度分析の理論が応用されており、数理的な根拠に基づいた設定値の算出を可能にしている。

また、ネットワーク運用の安全性を担保するため、導出したリンクコストを設定する順序にも着目した。コスト変更の適用順序を誤ると。OSPF の収束中に一時的なルーティングループやブラックホールなどの障害が発生する恐れがある。そこで本研究は、それらのリスクを最小化するための「安全シーケンス」の自動生成を提案する。まずコスト減少操作によってトラフィックの逃げる先となる経路を先に確立する。その後に、コストを増加させる操作によって元の経路からトラフィックを排除する。このようにして、順序を自動的に決定する。これにより、大規模な設定変更プロセスにおける可用性を最大化し、人的ミスに起因するサービス中断のリスクを減らしている。

提案手法の有効性を検証するため、WIDE プロジェクトのバックボーンネットワークの実際のデータを用いて評価を行なった。トポロジ構築の網羅性の検証では、提案する LSDB 解析手法により 45 台のルータと 116 本のリンクを特定した。これは従来の CLI ベースやコンフィグベースの手法と比較して大幅に高い検出数であった。これが提案手法が設定ミスや動的なセグメントの変化を正確に捉えられることを示した。

トラフィックの推定精度については、運用監視システムである Zabbix の実測値と比較検証を行った。エッジノードにおいては誤差 0.08%という極めて高い再現性を確認した。バックボーンリンクにおいては、フロー情報のサンプリングバイアスや L2 オーバーヘッドに起因する約 30%の過小評価が見られた。しかし、負荷の相対的な大小関係は実測値と一致しており、輻輳箇所の特定という実運用上の目的において十分な有用性を証明した。

さらに、WIDE プロジェクトのデータを用いて作った複数の意図により、インテント・リゾルバの有用性を確認した。「負荷分散」のシナリオでは、特定パスに集中していた約 963.5Mbps のトラフィックを、自動的なコスト調整により 2 つのパスへ各 481.76Mbps ずつ正確に等分割することに成功した。また「流量削減」のシナリオでは、感度分析を通じて近隣の到達性を維持しつつ、バーストトラフィックのみを迂回させる最適な中間コスト値を特定した。これらの結果は、提案手法が複雑なメトリック計算を抽象化し、運用の効率性と安全性を飛躍的に向上させることを示している。

本研究は、デジタルツインを用いた意図ベースの制御が次世代のネットワーク運用自動化において極めて有効な基盤となることを立証した。今後は、トラフィックサンプリングの統計的補完モデルの導入や、他のルーティングプロトコルとの連携、マルチエリア OSPF への拡張に取り組む予定である。

目 次

第 1 章　はじめに **1**
1.1 研究背景 1
1.2 研究目的 2
1.2.1 実ネットワークの忠実な再現 2
1.2.2 What-If 分析による事前検証の実現 2
1.2.3 運用意図に基づくコスト算出の自動化 3

第 2 章　既存のネットワーク検証および構成手法 **4**
2.1 構成情報およびトポロジ記述に基づく手法 4
2.2 シミュレータおよびエミュレータを用いる手法 5
2.2.1 再現環境の概要 5
2.2.2 大規模環境におけるスケーラビリティの制限 5
2.3 トラフィックエンジニアリングとネットワーク・デジタルツイン . . 6
2.3.1 従来の重み最適化手法 6
2.3.2 デジタルツインを用いた検証の進展 6
2.4 既存手法の課題と本研究の位置付け 7

第 3 章　提案手法：動的データ収集に基づくデジタルツイン構築システム **8**
3.1 システム全体概要 8
3.2 データ収集とトポロジモデル化 (Collector Module) 9
3.2.1 OSPF LSDB 解析によるトポロジ構築 9
3.2.2 トラフィック分析のためのサブネットマップ生成 11
3.2.3 BGP NextHop 解析による境界ルータ (ASBR) の特定 11
3.3 トラフィック需要の正規化とマッピング (Preprocessor Module) . . 12
3.3.1 フローデータの収集と正規化処理 12
3.3.2 ノード解決アルゴリズム 13
3.4 基本シミュレーションエンジン 15
3.4.1 経路計算と負荷の重畳プロセス 15
3.5 運用意図に基づく自律的経路制御ロジック 16
3.5.1 運用意図の定義と詳細設計 16
3.5.2 移行リスクを考慮した安全な実行順序決定アルゴリズム . . 18
3.6 トラフィック状態の視覚的表現手法 19

3.6.1 対数スケールを用いた負荷集中度の可視化手法 19
3.6.2 時系列フロー再現とレイヤー分離手法 19

第 4 章 実装と実証実験環境 20
4.1 開発環境とソフトウェア構成 20
4.2 データ収集・前処理モジュールの実装 20
4.2.1 SNMP 通信 21
4.2.2 Scapy を用いた OSPF LSDB の解析 21
4.2.3 NetworkX によるグラフオブジェクトの操作 21
4.2.4 トラフィック紐付けのためのサブネット抽出モジュール . . 21
4.2.5 BGP NextHop 解析による ASBR 特定の実装 22
4.3 トラフィックシミュレータの実装 22
4.3.1 多段階ノード解決ロジックの実装 23
4.3.2 分析レポート生成機能の実装 23
4.4 運用意図解決エンジンの実装 24
4.4.1 安全な実行順序とフェーズ制御の実装 24
4.4.2 意図ハンドラの実装 24
4.5 実証実験環境およびデータの正規化 25
4.5.1 対象ネットワークとトラフィックデータの諸元 25
4.6 対象ネットワーク環境 25
4.7 実験データの諸元 25
4.8 比較検証用基盤 26

第 5 章 評価と考察 27
5.1 評価環境と手法 27
5.2 トポロジ構築手法の網羅性と優位性 27
5.2.1 比較対象手法 28
5.2.2 構成要素の検出数比較 28
5.2.3 ノード識別精度の定量的評価 28
5.2.4 考察: 提案手法の優位性 29
5.2.5 ノード負荷推定の精度検証 29
5.3 インテント・リゾルバによる運用シナリオの評価 32
5.3.1 カテゴリ A：トラフィック負荷分散と輻輳緩和 32
5.3.2 カテゴリ B：保守・障害シミュレーションと運用安定性 . . . 34
5.3.3 カテゴリ C：QoS とサービス品質最適化 35
5.4 提案システムの優位性に関する考察 38
5.4.1 運用の抽象化によるヒューマンエラーの低減 38
5.4.2 デジタルツインによる「事前保証」と副作用の可視化 38
5.4.3 安全手順による過渡的障害の回避 38

5.4.4　推計精度による信頼性の確保　. 38

第 6 章　結論 **40**

6.1　まとめ　. 40

6.2　今後の展望　. 40

図 目 次

3.1 提案システムの機能モジュール構成と各節との対応関係 9
3.2 トラフィックフローのノード解決アルゴリズム 14

5.1 リンク負荷のシミュレーション値と Zabbix 実測値 30
5.2 インテント適用前後のトラフィックヒートマップ比較 34
5.3 保守排除インテント適用前後のヒートマップ比較 36
5.4 低遅延制御適用前後のヒートマップ比較 37

表 目 次

4.1 実装に用いた主要なソフトウェアとライブラリ 20

5.1 従来手法群と提案手法による検出要素数の比較 28
5.2 Router ID の識別精度および一致率の比較 28
5.3 各観測ノードにおける推計負荷と Zabbix 実測値の比較 30
5.4 カテゴリ C におけるインテント適用前後の流量変化 36

第1章　はじめに

1.1　研究背景

昨今の自律システム（AS：Autonomous System）では、ネットワーク内部の経路制御を OSPF や IS-IS などの IGP（Interior Gateway Protocol）を用いて自動化している。これらのルーティングプロトコルは、各ノード間のリンクに設定されたコスト値などに基づいて最短経路を計算し、パケット転送経路を決定している。

このリンクコストは、デフォルトでは各インターフェイスの帯域幅をもとに自動で算出される。一方で、実際のネットワーク運用においては、単なる接続性の観点だけでなく、トラフィックエンジニアリング（TE）の観点より、「特定のリンクの輻輳を回避したい」「メンテナンスのために特定経路を迂回させたい」といった運用者の明確な「意図（Intent）」を反映させたい場面が多々存在する。現在、このような意図の実現には、運用者自身の経験則や複雑な計算に基づいてリンクコストを手動で調整し、実際の流量を観察し、さらに調整というような試行錯誤的に設定変更を余儀なくされているのが実状である。特に、等コストマルチパス（ECMP）の存在するネットワークでは、単一のリンクコストの変更がトラフィックの分散比率やネットワーク全体のフローに及ぼす影響を完全に予測することは極めて困難で、高度な専門性と多大な労力を必要とする。

さらに、実際に運用中のネットワークに対して安易に直接的な設定変更を行うことには、重大なリスクを伴う。変更された設定が即座に反映されてしまうため、意図しない経路へのトラフィック集中や、予期しない通信切断を引き起こす可能性がある。問題が発生した場合の原因分析や、過去の状態を再現しての検証も容易ではない。

既存の検証手段としては、ネットワークシミュレータやエミュレータがよく用いられる。しかし、これらは静的な構成情報をもとに検証していることが多く、実際のネットワークを流れるリアルタイムなトラフィック分布や構成変化、複雑な運用意図の影響まで正確に反映・予測することには限界がある。そのため、実際のネットワークトポロジ、経路情報、トラフィックを仮想空間のなかで忠実に再現し、設定変更の影響を事前かつ安全に検証できる「ネットワーク・デジタルツイン」環境の構築が重要であると考える。本デジタルツインは、単なるシミュレータではなく、ネットワークの現在の状態を把握する「観測可能性」と、変更後の挙動を予測する「予測可能性」を提供するものである。

1.2 研究目的

昨今、ネットワークバックボーンの複雑化に伴い、ネットワーク運用において設定変更の影響範囲を正確に予測することは極めて困難になっている。本研究の目的は、実際のネットワークから収集した運用データに基づき忠実度の高い「デジタルツイン」を構築することである。さらに、この「デジタルツイン」を用いて、OSPF ネットワークにおける運用者の意図を反映した経路制御を支援し、事前に物理環境に影響を与えることなく検証することのできるシステムを構築する。なお、対象ネットワークとして大規模かつ、OSPF で実運用されている WIDE プロジェクトのバックボーンネットワークを用いる。

具体的には、本研究では以下の 3 点の達成を目標とする。

1.2.1 実ネットワークの忠実な再現

既存のネットワークシミュレータの多くでは、トポロジやトラフィックを静的な情報や人工生成したものに依存していて、実際の運用環境に近い挙動を完全に模倣することは困難である。そこで、本研究では、SNMP（Simple Network Management Protocol）を用いて実機ルータより取得した OSPF のリンクステートデータベース（LSDB）を解析することで、L3 ベースの隣接関係だけでなく、ノード間のリンクのメトリックやエリア構成を含めたコントロールプレーンの状態を基に「デジタルツイン」を構築する。加えて、sFlow からサンプリングした Flow 情報を実トラフィック情報として取得し、再現したネットワークトポロジ上にフローをマッピングする。これにより、人工的に生成されたトラフィックでは把握しきれない実際のトラフィック変動やルーティングプロトコルの挙動に近い、高度な検証基盤を確立する。

1.2.2 What-If 分析による事前検証の実現

実装したデジタルツイン環境上で、OSPF のリンクコスト変更が経路選択やトラフィック分布にどのような影響を与えるかをシミュレーションする「What-If 分析」機能を実現する。本システムによって、ネットワーク運用者は事前に、デジタルツイン上で「特定のリンクコストを変更した場合、トラフィックがどのように迂回するか」「新たなボトルネックが発生しないか」といった波及効果を定量的かつ視覚的に評価することが可能となる。これにより、リンクコスト変更による実ネットワークへのリスクを軽減し、安全性の高い運用変更計画の策定を支援する。

1.2.3　運用意図に基づくコスト算出の自動化

従来のトラフィックエンジニアリングでは、「ここからここへのトラフィックはここを経由して欲しい」というような運用者の意図を実現するには、複雑なリンクコストの計算が必要であり、極めて属人性が高く、実際にトラフィックを流すまではそれが正しいのかすらわからなかった。本研究では、このような運用者の意図を入力として、それを満たすために最適な OSPF リンクコスト設定を数理的あるいは探索的に計算し、実際のトラフィックを流して検証するシステムを構築する。これにより、設定変更に伴う副作用（意図しない経路変更や輻輳）を最小化しつつ、目標とするトラフィック分散や遅延最適化を実現する設定値を、客観的かつ論理的な根拠に基づいて提示することを目指す。

第2章　既存のネットワーク検証および構成手法

本章では、ネットワークの構成把握および検証に用いられる既存手法を整理する。具体的には、設定ファイルやトポロジ記述から論理的な挙動を導出する手法、およびシミュレータやエミュレータを用いてパケットレベルの挙動を再現する手法の二つを取り上げる。さらに、OSPF 等のリンクコスト最適化に関する従来の手法についても概観し、大規模な実運用環境における課題を明らかにする。

2.1　構成情報およびトポロジ記述に基づく手法

設定ファイル（コンフィグ）や特定の記述言語からネットワークの状態や構成を把握する手法として、静的解析ツールを用いる方法や、テンプレートベースの構成管理プラットフォーム [8] を使う方法が存在する。

コンフィグ解析の代表例である Batfish 等は、各ルータの設定ファイルを読み込み、ベンダーごとに異なる構文を抽象化して、論理的な到達性やルーティングポリシーの検証を可能にする。この手法の特徴として、実機を稼働させずに大規模なネットワークの論理チェックが可能であることと、ACL やルーティングの矛盾を網羅的に検出できることが挙げられる。しかし、実機の設定ファイルに依存する手法のため、以下の制約がある。

- **動的状態の欠如：**インターフェースの Up/Down や、OSPF の Router ID が自動選出される際の動的な挙動を正確に反映できない。
- **収集のオーバーヘッド：**全ルータから最新のコンフィグを常に収集し続ける必要があり、実ネットワークとの同期にタイムラグが生じる。

また、dot2net[8] のように、DOT 形式のグラフ構造から設定ファイルを自動生成する手法は、トポロジと機能設定を分離し、大規模展開時の管理コストを下げる面で有効である。dot2net の性能評価によれば、1,000 ノード規模の CLOS ネットワークにおいて、以下の性能が示されている。

- **設定生成時間：**3.5 秒以内

- **メモリ使用量：**2.65GB

しかし、これらは「トポロジから設定を作る」アプローチであり、既存の実ネットワークから現在のリアルタイムな状態を取得し再現、トラフィックの変化を動的にシミュレーションすることまでは考慮されていない。

2.2 シミュレータおよびエミュレータを用いる手法

本節では、パケットレベルの挙動を再現するシミュレータとエミュレータの概要および、それらを用いた実運用ネットワーク再現の制限について議論する。

2.2.1 再現環境の概要

ネットワークの動作検証には、ns-3[11] のようなディスクリートイベントシミュレータや、Containerlab[3] のようにコンテナ仮想化技術を用いたエミュレータが広く利用されている。

ns-3 は、プロトコルスタックの詳細なモデル化により、物理層からアプリケーション層までのパケットの挙動を精密にシミュレートする。一方、Containerlab は Docker 等のコンテナ技術を活用して、実際のネットワーク OS image をホスト上で動かす。これにより、運用者は実機と同じかそれに類するコマンド操作やプロトコル挙動を安価に検証・再現できる。

これらの環境には、以下に挙げる機能がある。

- **プロトコルの忠実な再現**: OSPF や BGP 等のルーティングプロトコルを実機同様のデータ構造やモデルで動作させる。
- **仮想リンクの模倣**: 遅延、帯域幅、パケットロス等の物理特性を仮想的なインターフェース間に付与する。
- **スケーラビリティの確保**: 物理的なハードウェアを揃えることなく、単一のサーバ上に数十〜数百の仮想ノードを展開することができる。

2.2.2 大規模環境におけるスケーラビリティの制限

しかし、これらを実運用のデジタルツインとして活用するには、計算資源の面で大きな課題がある。シミュレータ上で実環境を再現するには、サンプリングされたフロー情報や動的に変化する OSPF LSDB をリアルタイムに同期させる必要があるが、パケット処理や OS のエミュレーション負荷はノード数に対して指数関数的に増大する。

その結果、WIDE プロジェクトのような数万フローが定常的に流れるバックボーンネットワークを、パケットレベルのシミュレータで完全同期して再現し続けることは現実的ではない。そのため、本研究では精密なパケットエミュレーションではなく、グラフ理論に基づくフローレベルのシミュレーションを用いて、スケーラビリティを確保する。

2.3 トラフィックエンジニアリングとネットワーク・デジタルツイン

大規模ネットワークにおいては、流れるトラフィックをどのように制御するかが重要となる。本節では、従来の OSPF 重み最適化手法と、最新のデジタルツインを用いたアプローチを整理する。

2.3.1 従来の重み最適化手法

Fortz と Thorup の研究 [5] では、OSPF の重み設定問題が NP 困難であることを示し、ローカル探索アルゴリズムによる最適化を提案した。その後、遺伝的アルゴリズム（GA）を用いたアプローチ [4, 1] や、ネットワークの負荷状況に応じて動的にコストを調整する CA-OSPF[16] などが提案されてきた。しかし、これらの従来手法は、網全体のパフォーマンス最大化を主目的としており、以下の課題が残されていた。

- **ブラックボックス化**: 最適化アルゴリズムが算出したコスト値の根拠が、運用者にとって直感的ではない。
- **移行プロセスの欠如**: 最終的な最適値は提示されるが、設定投入順序による過渡的な障害リスクが考慮されていない。

2.3.2 デジタルツインを用いた検証の進展

近年、ネットワーク運用にデジタルツイン（NDT）を導入する研究が急速に進展している。Poorzare ら [13] は、最新のサーベイにおいてトラフィックエンジニアリングにおける NDT の重要性を指摘している。具体的な活用例として、Polverini ら [12] は、設定変更がトラフィックに与える影響を事前に評価する「What-If 分析」のフレームワークを提案した。また、Zalat ら [15] は、実ネットワークを監視し、平均遅延や損失率を最小化する重み設定を自動適用する実用的な NDT の構成を示している。

2.4　既存手法の課題と本研究の位置付け

本章で述べた通り、既存手法には実運用環境への適用において以下の共通課題が存在する。

トポロジ構築においては、コンフィグ解析等の静的情報に依存する手法では、バイナリレベルの LSDB 解析が捉える「現在の生の動的状態」を再現できない。

また、トラフィックエンジニアリングの面では、既存の最適化アルゴリズム [7, 6, 16] は網全体の効率化を優先するあまり、運用者の具体的な「意図（Intent）」[2, 9] を取り込む柔軟性に欠けていた。

これに対し、本研究は以下の独自性を持つ。第一に、LSDB と実フローデータの動的結合により、絶対的な精度のみならず「トラフィック遷移の傾向」を捉える予測可能性（Predictability）を重視する点である。第二に、最短経路問題における感度分析の理論 [14] を応用した「Intent Resolver」を導入し、複数の解の中から副作用が最小となる設定を論理的に選定する点である。さらに、設定変更の実行順序（Safe Sequence）までを自律的に構成することで、定常状態のみならず「運用プロセス」そのものの安全性を担保する点に特徴がある。

第3章　提案手法：動的データ収集に基づくデジタルツイン構築システム

本章では、実ネットワークより、構成情報およびトラフィック情報を動的に収集し、「デジタルツイン」を構築するための提案手法について説明する。

3.1　システム全体概要

本システムは、実ネットワークの動的な変化を反映し、運用意図を反映したシミュレーションを実現するために、機能ごとに独立した 4 つのソフトウェアモジュールによって構成される。

図 3.1 にシステムの全体構成とデータの流れを示す。各モジュールの役割および本章における構成は以下の通りである。

1. **Collector Module（データ収集部）：** → **3.2 節**実機ルータ（SNMP）やログ基盤（Elasticsearch）から生データを収集し、物理トポロジのグラフ化を行うモジュール。

2. **Preprocessor Module（前処理・マッピング部）：** → **3.3 節**収集されたトラフィックデータ（Flow）の正規化や、IP アドレスとトポロジ上のノード ID を紐付けるノード解決を行うモジュール。

3. **Simulator Module（シミュレーション・意図解釈部）：** → **3.4 節, 3.5 節** OSPF のルーティングを再現するための最短経路計算（SPF）による現状分析に加え、運用者の意図（Intent）を解釈してトポロジを仮想的に変形させ、現実の Flow を流した際の挙動をシミュレーションする。

4. **Visualizer Module（可視化・レポーティング部）：** → **3.6 節**シミュレーション結果をヒートマップや時系列アニメーションとして可視化し、運用判断のための定量レポートを出力するモジュール。

次節以降では、これらの各モジュールの詳細な実装アルゴリズムについて順に説明する。

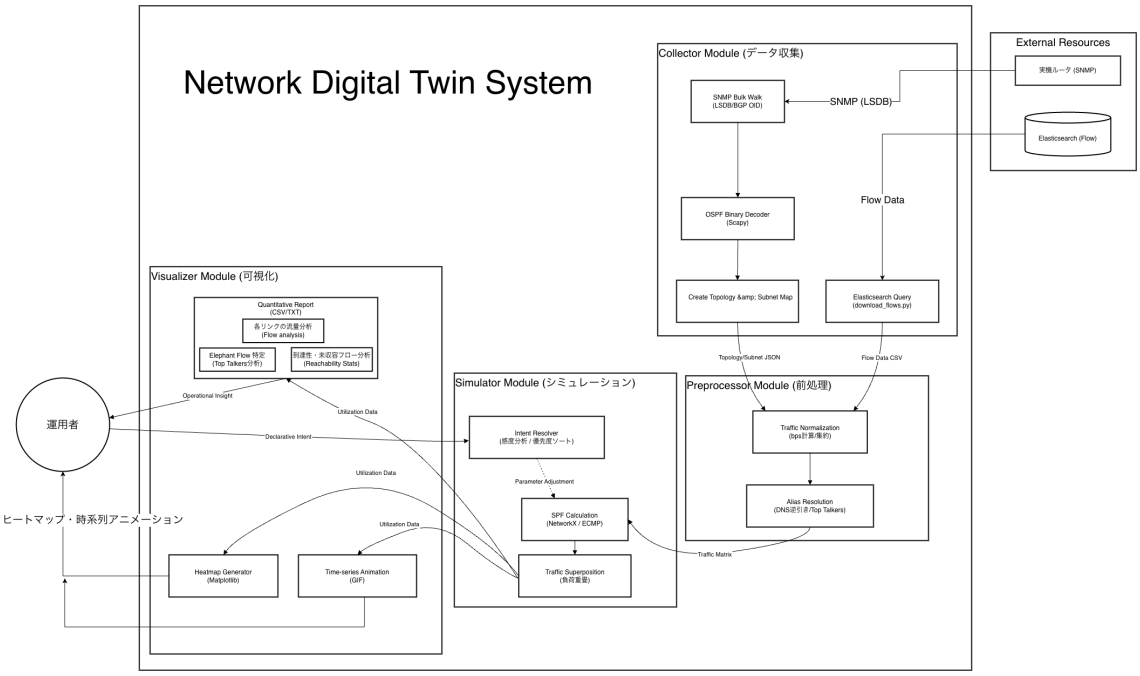


図 3.1: 提案システムの機能モジュール構成と各節との対応関係

3.2 データ収集とトポロジモデル化 (Collector Module)

本節では、システム入力の起点となる **Collector Module** の詳細について説明する。本モジュールは、SNMP を用いて OSPF のリンクステート（LSA: Link State Advertisement）情報を取得し、グラフ理論に基づいてネットワークトポロジを生成する。

3.2.1 OSPF LSDB 解析によるトポロジ構築

本システムのネットワークトポロジの再現は、SNMP によって実機ルータから取得した OSPF の Link State Database（LSDB）の解析によって実現されている。各ルータのインターフェース情報（iftable）や ARPtable を収集して隣接関係を推定する手法もあるが、ブロードキャストドメインの正確な把握や、論理的なコスト計算の再現が難しくなる。一方で、本研究では OSPF プロトコルが内部で保持するバイナリデータをデコードするアプローチを採用した。これにより、ルータ間の物理的な隣接関係だけでなく、サブネットマスクを含む詳細なアドレッシング情報を取得することができている。また、OSPF の特性上、同一エリア内の全ルータは同一の LSDB を保持するため、本システムではネットワーク内の任意のルータ 1 台、もしくは冗長性を考慮した数台から SNMP 経由でデータを取得することで、エリア全体のトポロジを網羅することを可能にしている。

LSDB の取得とパケット構造のデコード

トポロジ生成の第一段階として、対象 AS 内の任意のルータに対して、SNMP を用いて OSPF LSDB（OID：1.3.6.1.2.1.14.4.1.8）を取得する。取得したデータは 16 進数のバイナリ列であるため、Python のパケット解析ライブラリである Scapy を用いて OSPF ヘッダ構造に従ってデコード処理を行う。本システムでは主に下記の LSA を用いた。

- **Router LSA (Type 1):** 各ルータが自身のリンク状態を広告するもの。
- **Network LSA (Type 2):** イーサネット等のマルチアクセスネットワークにおいて、Designated Router (DR) がセグメント情報を広告するもの。

グラフ理論に基づくトポロジ構築アルゴリズム

解析した LSA オブジェクトより、ネットワークをグラフ $G(V, E)$ としてモデル化する。ここで V はノード（ルータおよびネットワークセグメント）、E はリンクを表す。

ルータノードとリンク種別の識別　Type 1：Router LSA の解析では、Advertising Router ID をグラフ上のルータノード v_r として定義する。次に、LSA 内部に含まれるリンク記述を見て、リンクタイプに応じた処理を行う。

1. **Type 1 (Point-to-Point 接続):** リンク ID が隣接ルータの Router ID の場合、該当ルータ間に有向エッジ $e(v_{src}, v_{dst})$ を生成する。エッジの重みには、LSA に含まれるメトリック（OSPF Cost）を使用する。
2. **Type 2 (Transit Network 接続):** リンク ID が DR のインターフェース IP を示している場合、その接続先は特定のルータではなく「共有ネットワークセグメント」である。本システムでは、このセグメントを仮想ノード v_{net} とし、ルータと仮想ノード間にエッジを張ることで、L2 スイッチを含む多地点接続を単一の仮想ノードで表現した。
3. **Type 3 (Stub Network 接続):** リンク先がルータでもトランジットネットワークでもない末端セグメント（ループバックアドレスやユーザー収容 VLAN など）である場合、これらはトポロジグラフ上の通過ノードとしては扱わない。しかし、後述するサブネットマップの生成ソースとして利用する。

Network LSA による共有セグメントの補完　Router LSA だけでは、トランジットネットワークの正確なサブネットマスクや、そのセグメントに接続している他のルータ一覧を完全には網羅できない。そのため、Network LSA を見て、以下の処理を行う。

- 当該セグメントに接続するすべてのルータをグラフ上で相互にエッジを張る。
- 当該セグメントから各ルータへのコストは、OSPF の仕様により 0 として設定する。
- LSA に含まれるネットマスク情報を取得し、セグメントの属性として保持する。

3.2.2　トラフィック分析のためのサブネットマップ生成

ネットワークトポロジとは別に、トラフィックデータに含まれている送信先・元 IP アドレスが帰属するルータを特定するためにサブネットマップを作成する。トポロジ図の描画とは別に、トラフィックデータに含まれる IP アドレスが「どのルータに帰属するか」を解決するためのデータベースとして「サブネットマップ」を生成する。前述の Router LSA Type 3 リンクおよび Network LSA から抽出されたネットワークアドレスとサブネットマスクのペアを、以下のデータ構造を持つ Json を作成・保存する。

```
{
    "network": "192.168.10.0", // link.id
    "mask": "255.255.255.0",   // link.data
    "owner": "10.0.0.1",       // Router ID
    "type": "Stub"
}
```

このサブネットマップにより、トポロジ上にはノードとして現れない末端の IP アドレスであっても、その収容ルータ（Owner）を一意に特定することが可能となる。

3.2.3　BGP NextHop 解析による境界ルータ (ASBR) の特定

前述までの情報で、AS 内のアドレスは解決することができるようになった。一方、これだけでは AS 外の IP アドレスは解決できない。OSPF において AS 外との通信は、ASBR (Autonomous System Boundary Router) を通してやり取りされ

る。そのため、ASBR をトラフィックの流入出点として正確に特定・可視化する必要がある。

ASBR を特定する際、各ルータの BGP ネイバー確立状態（bgpPeerTable）を監視するだけでは不十分である。なぜなら、BGP セッションが確立していても、そのルータが必ずしも外部経路を直接保持しているとは限らない。例えば、iBGP を用いて ASBR から経路情報を受け取っているだけの内部ルータや、経路を中継するだけの Route Reflector なども BGP ピアを持つため、これらを ASBR と誤認する恐れがある。

そこで本システムでは、BGP の経路情報に含まれる NextHop 属性に着目した。代表ルータから BGP4-MIB の bgp4PathAttrNextHop (OID: 1.3.6.1.2.1.15.6.1.6) を取得する。BGP の仕様上、外部 AS から学習した経路の NextHop 属性は、AS 内部に伝播される際も通常書き換えられず、「その経路を最初に AS へ注入した ASBR の IP アドレス」が維持される特性がある。NextHop アドレスと OSPF トポロジ上のルータ ID を照合することで、単なる中間ルータを排除し、確実に外部との接点のみを特定している。判定されたノードにはトポロジデータ上で is_asbr=True のフラグを与える。

3.3 トラフィック需要の正規化とマッピング (Preprocessor Module)

本節では、収集された生データをシミュレーション可能な形式に変換する **Preprocessor Module** について説明する。NetFlow/sFlow データは本来サンプリングされた「点」の情報に過ぎないが、これをトポロジ上の「線」の負荷としてシミュレーションに反映させるため、データの正規化と論理アドレスを物理ノードへ紐付ける必要となる。

3.3.1 フローデータの収集と正規化処理

WIDE プロジェクト側でフローデータを Elasticsearch に収集していたため、それを使用した。Elasticsearch から取得したフローデータに対して前処理を行う。元データには送信元/宛先 IP、転送バイト数、フロー継続時間に加え、BGP によって学習した bgp_next_hop 情報が含まれる。本システムでは以下の手順で、シミュレーション用のトラフィックマトリクスを生成する。

1. **通信速度の算出:** フローの総バイト数 B と継続時間 T (秒) から、平均通信速度 R (bps) を次式で算出する。

$$R = \frac{B \times 8}{\max(T, 0.001)}$$

ここで、計測誤差による $T = 0$ となるフローに対しては、ゼロ除算を防ぐために 0.001 秒の最小値を適用する補正を行っている。

2. **BGP 属性の保持:** IP アドレスのマッチングでは解決できない外部トラフィックを識別するため、フローレコードに含まれる bgp_next_hop カラムを欠損処理（デフォルト値 `0.0.0.0` 埋め）した上で保持し、後のプロセスへ引き渡す。

3.3.2 ノード解決アルゴリズム

フローデータ上の IP アドレス（特に外部 IP や IPv6）が、トポロジグラフ上のどのノードに対応するかを特定するため、本研究では図 3.2 に示す手順で実装した。

1. **Alias Map 照合（DNS リゾルバ）:** IPv6 アドレスを持つフローに対し、DNS 逆引き（PTR レコード）および正引き（A レコード）を連鎖的に行う。これにより、ホスト名を経由して IPv4 ベースのルータ ID への変換（例: IPv6 → Hostname → IPv4 Router ID）を行っている。

2. **Core Router の自動推定（Degree Centrality）:** DNS 解決が不可能で、かつトポロジマップにも存在しない未知の IP アドレスについては、ネットワークの中心ノードへ集約させるフォールバック処理を行う。具体的には、トポロジグラフ G における各ノードの次数（Degree）を計算し、最も接続数が多いルータを「Core Router」として自動選出する。

$$v_{core} = \arg\max_{v \in V} \deg(v)$$

このロジックにより、所属不明なトラフィックであっても解析対象から除外することなく、最も帯域が集まるコアスイッチを経由するものと仮定してシミュレーションを継続可能としている。

3. **BGP NextHop Check:** 前節で保持した bgp_next_hop がトポロジ内に存在する場合、宛先 IP が不明であっても、そのネクストホップルータを出口（Exit）とみなしてマッピングを行う。AS 外からの通信であっても同様に送信元 IP を解決する。

4. **Subnet Mapping:** サブネットマップを探索し、対象 IP が属するサブネットを広告しているルータ（Owner）を特定する。

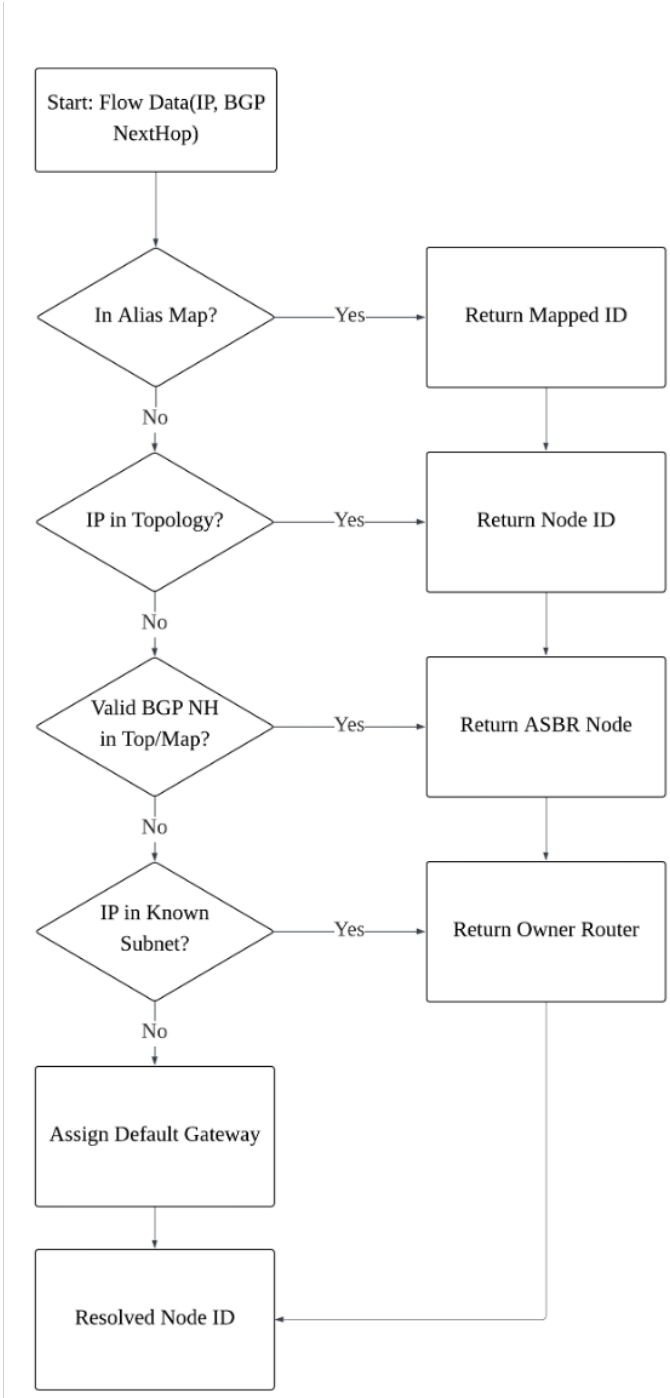


図 3.2: トラフィックフローのノード解決アルゴリズム

3.4　基本シミュレーションエンジン

本節および次節では、本システムの中核となる **Simulator Module** の内部ロジックについて説明する本節では、前節までに作られたトポロジとトラフィック Matrix、ノード解決ロジックを用いて現状のトラフィックフローを再現する基本機能について説明する。

本システムでは、Python のグラフ理論ライブラリ `NetworkX` を用いて、OSPF と同様の動作をする最短経路計算（SPF: Shortest Path First）を行い、各リンクへの負荷を算出する。

3.4.1　経路計算と負荷の重畳プロセス

本システムは、以下の手順でトラフィックフローを物理的なリンク負荷へと変換し、ネットワーク全体の状態を再現する。

1. **ネットワークグラフの構築:** 収集したトポロジを用いて、ルータをノード V、リンクをエッジ E とする有向グラフ $G(V, E)$ を構築する。各エッジには、実際のネットワーク設定から抽出した OSPF コストを重み（Weight）として付与する。

2. **フローの集約とノード解決:** フロー情報は、送信元・先および BGP NextHop 情報の組み合わせごとに集約する。論理的な IP アドレスをグラフ上の始点ノード s および終点ノード d へマッピングする。

3. **最短経路計算:** マッピングされたペアをダイクストラ法を用いてコストが最小となる経路 $P(s, d)$ を算出する。これによって、OSPF による実際のフローの挙動を再現する。

4. **負荷の重畳 (Traffic Superposition):** 算出された経路 $P(s, d)$ を構成する各エッジ $e \in P(s, d)$ に対し、当該フローの通信量 f_{sd} を加算する。全フローに対して同様の動作を繰り返し、各リンク e の総トラフィック量 $L_e = \sum f_{sd}$ を導出する。

複数経路における負荷分散モデル

最短経路計算の結果、複数の最短経路が見つかった場合、等コストマルチパス（ECMP）の挙動を再現するために、フローの通信量を候補数 n で等分して各経路に配分する。

具体的には、各リンク e における総トラフィック負荷 L_e は、以下の式によって算出される。

$$L_e = \sum_{f_{sd} \in F} \left(f_{sd} \times \frac{\delta_{e,P(s,d)}}{n(s,d)} \right) \tag{3.1}$$

ここで、各変数の定義は以下の通りである。

- F: ネットワーク内を流れる全フローの集合
- f_{sd}: 送信元 s から宛先 d へのフローの通信量 (bps)
- $n(s,d)$: s から d への最短経路の総数
- $\delta_{e,P(s,d)}$: エッジ e が s から d への最短経路集合に含まれる場合に 1、含まれない場合に 0 となる指示関数

3.5 運用意図に基づく自律的経路制御ロジック

本節では、デジタルツイン上で運用者の意図（Intent）をネットワークに反映するために、意図をリンクコストへと変換し、その影響を予測・検証する手法について説明する。

3.5.1 運用意図の定義と詳細設計

本システムにおいて意図とは、運用者がネットワークの具体的な構成変更（OSPF コストの直接操作等）を意識することなく、到達したいネットワーク状態を宣言的に記述したものである。これら抽象的な要求をトポロジ情報およびトラフィックデータと付き合わせ、最適な OSPF メトリックへと変換を行なっている。本システムが解決対象とする 6 種類の運用意図について、その定義、想定される運用シナリオ、および解決アルゴリズムの説明をする。

構造的制約に関する意図

ネットワークのトポロジ構成を変更して、メンテナンスや故障への耐性を高めるための意図である。

1. ノード排除 (Node Drain)

運用シナリオ ルータの保守作業に伴い、通信断を避けるために、対象ルータを経由している全トラフィックを事前に他の経路へ迂回させる必要がある。

解決アルゴリズム 対象ノードに隣接する全てのリンクコストを OSPF の最大値である 65535 に設定する。これにより、最短経路計算において当該ノードを通過するコストが極大化され、ネットワーク全体から当該ノードが論理的に排除される。

2. リンク遮断 (Link Close)

運用シナリオ 特定の回線の物理的な切断作業や、故障発生時の影響を事前に確認したい場合に使用する。「もしこのリンクが切れたら通信は維持できるか」という What-if 分析を行うためのものである。

解決アルゴリズム デジタルツイン上のトポロジグラフから対象リンクを前述同様に論理的に削除し、全トラフィック行列に対して経路の再計算を行う。

経路誘導に関する意図

特定の通信品質（QoS）を最適化し、リソースの利用効率をあげるための制御意図である。

3. 低遅延誘導 (Smart Low Latency)

運用シナリオ 特定のノード間を「ネットワーク内で特定のパス」へ強制的に誘導したい場合に使用する。

解決アルゴリズム 特定のパスを優先する際、単に最小コスト「1」を設定すると、意図しないフローまで流入し、新たな輻輳を招く恐れがある。そこで、対象経路 P に対し、現状の代替経路の最短コスト C_{rival} を基準として、必要十分な優位性を確保するコスト w_i を算出する。

$$\sum_{e_i \in P} w_i = C_{rival} - 1 \tag{3.2}$$

これにより、経路上の各リンクへコストを等分配することで、ネットワーク全体のメトリック整合性を維持しつつ、対象フローを確実に最短路へ誘導する。この手法は、網全体のパフォーマンスを考慮した重み設定の重要性を説いた Fortz らの知見 [5] に基づいている。

4. 負荷分散 (Load Balancing)

運用シナリオ 特定のパスが混雑し、並行する迂回路が空いている場合、これら 2 つのパスを等コストマルチパス（ECMP）として 50:50 で均等に利用するために使用する。

解決アルゴリズム パス P_A とパス P_B の累積コストを比較し、以下の等式を満たすようにコスト調整を行う。

$$C(P_A) = \sum_{e \in P_A} w_e = \sum_{e \in P_B} w_e = C(P_B) \tag{3.3}$$

一方のコストを下げきれない（下限値 $w_e = 1$ に達する）場合は、もう一方のコストを自動的に引き上げることで、OSPF の仕様（RFC 2328 [10]）に準拠した等コスト状態を自律的に形成する。

5. 流量削減 (Traffic Thinning)

運用シナリオ 特定のリンクの負荷を、目標とする削減率 r に基づいて調整したい場合に使用する。

解決アルゴリズム リンクコストを 1 単位ずつ段階的に増加させながら、トラフィック行列の経路遷移を繰り返し試算する。これは最短経路問題における感度分析（Sensitivity Analysis）の理論 [14] を応用したものである。削減後の流量 L_e^{new} が、目標値 $L_e^{new} \leq (1-r)L_e^{old}$ を満たした瞬間の最小コスト増分 Δw を特定することで、過度な迂回による他リンクの二次輻輳（副作用）を最小限に抑制する。

6. 安全範囲解析 (Safe Range Analysis)

運用シナリオ 設定変更による「他の重要な通信の経路変化（Side-effect）」を回避するため、副作用が発生しない安全な設定変更の限界値を特定したい場合に使用する。

解決アルゴリズム 対象リンク e のコスト w_e を変数とし、全フローの最短経路木が不変（Invariant）である境界値を算出する。Tarjan による感度分析のアルゴリズム [14] を用い、以下の条件を維持する範囲 $[w_{min}, w_{max}]$ を特定・提示する。

$$\forall f \in F, \quad P_{shortest}(s, d) \text{ is constant} \tag{3.4}$$

これにより、管理者はデジタルツイン上での全数探索に基づいた「数学的に保証された安全圏」の中で運用操作を行うことが可能となる。

3.5.2 移行リスクを考慮した安全な実行順序決定アルゴリズム

実際のネットワークにおいてコスト変更の適用順序を誤ると、ネットワーク収束中に一時的なルーティングループやブラックホールが発生するリスクがある。本システムでは、操作を 3 段階に分類して実行手順を整理した。

1. **Phase 1:** [**安全度：高**]
   コスト減少を伴う操作を最優先にする。ネットワーク内にトラフィックの「受け皿」を先につくり、パケットが破棄されるリスクを回避する。

2. **Phase 2:** [**安全度：中**]
   負荷分散等の微調整を行う操作。受け皿が確保された状態で、トラフィックの均衡化を図る。

3. **Phase 3:** [**安全度：低**]
   コスト増加や遮断を最後に行う。安全な迂回経路が確立された後に、対象箇所からトラフィックを安全に排除する。

3.6 トラフィック状態の視覚的表現手法

本システムで構築されたデジタルツインの状態を運用者が直感的に理解するため、以下の 2 つの視覚化手法を提案する。

3.6.1 対数スケールを用いた負荷集中度の可視化手法

ネットワークトラフィックは、リンクごとの負荷差が数桁に及ぶことがよくある。そこで、線形スケールではなく、対数正規化を用いたカラーマッピング手法を用いた。各リンク e の描画色 C_e および描画幅 W_e は、当該リンクの負荷 L_e に基づき、以下の通り定義する。

$$C_e = \mathrm{Colormap}(\log(L_e)) \tag{3.5}$$

$$W_e = W_{min} + \alpha \cdot \frac{\log(L_e)}{\log(L_{max})} \tag{3.6}$$

これにより、機関回線に比べて負荷の小さすぎる回線も同一図面上で識別できるようにした。

3.6.2 時系列フロー再現とレイヤー分離手法

動的なトラフィック変動を視覚的に認識するために、時間スライスによる状態の再現を行なった。時間ステップ Δt ごとに、以下の条件を満たすアクティブなフロー集合 F_{active} を抽出する。

$$F_{active} = \{f \mid T \leq f_{end} \wedge f_{start} < T + \Delta t\} \tag{3.7}$$

第4章　実装と実証実験環境

本章では、第 3 章で提案したデジタルツインを実際にどのように実装したのかを具体的に説明する。また、本システムを適用して評価を行うための実証実験環境について述べる。

4.1　開発環境とソフトウェア構成

システムは主に Python 3.9 を用いて開発している。また、ネットワークプロトコルの解析、グラフ理論に基づく計算、および可視化のために、表 4.1 に示すライブラリ群を使用した。

表 4.1: 実装に用いた主要なソフトウェアとライブラリ

Category	**Name**	**Description**
Runtime	Docker	コンテナ仮想化による実行環境の統一
Language	Python 3.9+	主要実装言語
Collection	PySNMP	SNMP による MIB 情報の取得
	Scapy	OSPF LSA のパケット構造解析とデコード
	Elasticsearch-Py	NetFlow/sFlow ログデータの検索と取得
Simulation	NetworkX	トポロジグラフの構築、最短経路探索 、中心性解析
	Pandas	大規模なトラフィックデータの正規化、結合、統計処理
Visualization	Matplotlib	ヒートマップおよび時系列スナップショットの描画
	Pillow	フレーム画像の結合による GIF アニメーション生成

4.2　データ収集・前処理モジュールの実装

本節では、トポロジ構築手法を実現するためのソフトウェア実装について説明する。

4.2.1 SNMP 通信

本実装では、SNMP によるデータ収集を、PySNMP ライブラリを用いて Bulk Walk を行うことで、数千行に及ぶ LSDB を取得させている。

4.2.2 Scapy を用いた OSPF LSDB の解析

サブネットマスク情報やブロードキャストドメイン内の接続関係を抽出するため、パケット解析ライブラリ Scapy を使用した。SNMP で取得したバイナリ列をバイトストリームとして読み込んでから、LSA Type ごとに異なるフィールド構造をパースさせている。

4.2.3 NetworkX によるグラフオブジェクトの操作

トポロジ情報の表現には、Python のグラフ操作ライブラリ NetworkX を使用した。Router LSA および Network LSA から取り出された隣接関係は、nx.Graph クラス（無向グラフ）のインスタンスとして構築される。

トポロジ図を可視化する際のレイアウトアルゴリズムには、NetworkX がライブラリで提供している Kamada-Kawai 法を採用した。トポロジデータは、他のモジュールで利用しやすいように、JSON 形式にされて保存している。

4.2.4 トラフィック紐付けのためのサブネット抽出モジュール

本モジュールは、OSPF LSDB からルーティング情報を網羅的に抽出し、トラフィック解析時の「IP アドレス対ルータ」の解決に使用される。

LSA からのプレフィックス抽出ロジック

実装においては、Router LSA 内の「Stub Network」リンク記述や Network LSA のネットワークマスク情報を解析した。SNMP により取得された情報より、ネットワークアドレスとサブネットマスクのペアをデコードし、その LSA を生成したルータを Owner としてマッピングする。

重複排除と最長一致の実装

実際の OSPF LSDB には、同一のサブネット情報が複数のルータから広告されることや、集約ルートと詳細ルートが混在することがある。これらをそのままにしてしまうと、トラフィック分析時に解決の競合が発生してしまう。そのため、本実装では以下の優先順位に基づくフィルタリング処理を行なった。

1. **重複排除:** ネットワークアドレスとマスク長が完全に一致するエントリは、Router ID が小さい（または大きい）方を代表として残し、重複したものを削除する。

2. **プレフィックス長のソート:** ipaddress ライブラリを用いてネットワークオブジェクトを生成し、サブネットマスク長が長い順にリストをソートして保存する。

この処理により、のちのシミュレータが IP アドレスを照合する際、リストを先頭から見るだけで「最長一致」に近い解決ができ、誤ったルータへの紐付けを防ぐことになる。生成されたデータは subnet_map.json として保存され、他のモジュールから参照される。

4.2.5 BGP NextHop 解析による ASBR 特定の実装

第 3 章で述べた「ASBR 特定ロジック」の実装について説明する。

NextHop 属性のバルク収集

ASBR の候補となる IP アドレスリストを取得するために、bgp4PathAttrNextHop (OID: 1.3.6.1.2.1.15.6.1.6) をターゲットとした SNMP Walk を実行する。バックボーンルータが保持する BGP 経路数は膨大であったため、通常の getCmd ではなく、PySNMP の bulkCmd を使用して一度のリクエストで複数のエントリを取得するように実装した。

OSPF トポロジとの照合ロジック

取得された NextHop アドレス情報を、先に取得したトポロジ内のノードリストと積集合をとった。これにより、NextHop に現れるルータを特定した。

特定されたノードに対しては、グラフデータの属性として node['is_asbr'] = True を与える。このフラグは、後のシミュレーション時に「対外通信の出口」として重み付けを行うための識別子として利用される。

4.3 トラフィックシミュレータの実装

本モジュールでは、前処理で生成された Traffic Matrix、DNS 逆引き（Alias Map）、トポロジ、サブネットマップを用いて OSPF におけるトラフィックの再現を行なった。

4.3.1　多段階ノード解決ロジックの実装

第 3 章で提案したノード解決アルゴリズムは、シミュレータ内の関数として実装した。本関数は、入力された不明な IP アドレスに対し、以下の順序で解決を試みるフォールバック構造を持つ。

1. **Alias Map 照合:** 事前に DNS 逆引きで生成した Alias Map を Hash Map として保持し、フロー内の IPv6 アドレスから OSPF 上のルータ ID（IPv4）への変換を行う。

2. **BGP NextHop 優先:** フローレコードに有効な BGP NextHop が含まれ、かつそのルータがトポロジ内に存在する場合、宛先 IP の解決をスキップして NextHop ルータを強制的に出口として採用する分岐処理を実装した。

3. **Subnet Mapping:** 前述のとおり、最長プレフィックス一致により収容ルータを特定する。

4. **Core Router Fallback:** 上記いずれの手法でも特定できない場合、NetworkX の degree_centrality 関数を用いて算出された「最も次数の高いルータ」をデフォルトの宛先として割り当てることで、シミュレーションの継続性を担保した。

これにより、宛先 IP が解決不能なグローバルアドレスであっても、最もらしい ASBR へトラフィックを誘導することが可能となり、Inter-AS トラフィックの再現を可能としている。

最近傍 ASBR 探索と負荷分散の実装

BGP 情報が欠損している外部トラフィックに対しては、NetworkX の shortest_path_length を応用して ASBR を割り当てた。ソースノードから候補となる全 ASBR までの経路コストを計算し、最小コストのノードを選択する。また、等コストの経路が複数存在する場合には、random.seed を固定した上で random.choice を用いてトラフィックを分散させている。

4.3.2　分析レポート生成機能の実装

シミュレーションによって得られた膨大なリンク負荷データを運用知見へと変換するため、Pandas をベースとした分析モジュールを実装した。

リンク負荷の集計と高負荷箇所の特定

各リンクを通過するトラフィック量を、統計的な Peak Load として算出し、リンクごとの負荷状況を求める。本システムでは、物理帯域が不明な環境においても客観的なボトルネックを特定するため、全リンクのトラフィック量をソートして、絶対的な流量が最も高いリンクを「潜在的な輻輳箇所」として抽出する実装とした。最終的な結果は CSV ファイルに出力され、運用者が実環境の回線スペックと照らし合わせることで、増強が必要な区間を即座に判別できるようにしている。

4.4 運用意図解決エンジンの実装

本節では、第 3 章で定義した意図のコストへの変換に関する実装詳細について説明する。本エンジンは NetworkX を基盤とし、Traffic Matrix とトポロジ情報、サブネットマップを用いることで意図の解決を行う。

4.4.1 安全な実行順序とフェーズ制御の実装

入力された複数の意図に対して、ネットワーク収束時のリスクを低減する「安全シーケンス」を生成するため、意図のソートロジックを実装した。実装においては、各変更操作のコスト増減（$\Delta Cost$）および絶対値を評価し、以下の優先度を付与している。

- **Priority 1** [**安全度：高**]**:** $Cost_{new} < Cost_{old}$ となる操作。
- **Priority 2** [**安全度：中**]**:** 通常のコスト微調整（1〜60000 の範囲）。
- **Priority 3** [**安全度：低**]**:** $Cost_{new} \geq 65535$ となる排除操作。

Python の `sorted` 関数により、リスト化された全変更操作を優先度順に再配列し、バイパスを確保した後に元の経路を絞るという安全なシーケンスの自動生成を行なった。

4.4.2 意図ハンドラの実装

各意図をコスト値へ変換するため、シミュレーションベースの個別ハンドラを実装した。

- **流量削減** (`handle_traffic_thinning`)**:** 対象リンクを通過するフロー集合に対し、コストを 1 単位ずつ加算しながら経路再計算を行う、感度分析を行なった。目標削減量（bps）を満たしたときの値を最適解として採用する。

- **低遅延誘導** (handle_low_latency_smart)**:** 対象外の最短経路のコスト C_{rival} を算出し、対象パスの合計コストが $C_{rival} - 1$ となるように各リンクへコストを等分配する。これにより、極端な値「1」に頼らない整合性の高い優先制御を可能にした。
- **負荷分散** (handle_load_balancing_bidirectional)**:** 2 パス間のコスト差分を埋めるときに、対象パスのコストを下げるだけでなく、最小値「1」に達した場合は自動的に対向パスのコストを引き上げるようにし、確実な ECMP を実現した。

4.5 実証実験環境およびデータの正規化

本システムを実運用環境へ適用し、その妥当性を検証するための実験条件である。

4.5.1 対象ネットワークとトラフィックデータの諸元

実証実験は、WIDE プロジェクトが運用するバックボーンネットワークを対象として行なった。収集したトラフィックデータは、主要ルータから送信される Flow ログであり、これらは約 1/1024 ～ 1/2048 の割合でサンプリングされている。

4.6 対象ネットワーク環境

本節以降では、上述の実装を用いた実証実験の条件について述べる。実証実験は、WIDE プロジェクトが運用するバックボーンネットワークを対象に行った。本ネットワークは、複数の研究拠点を接続する AS であり、商用 ISP とのピアリングやトランジット接続を持っている。また、主要な拠点間を結ぶ高速なバックボーン回線と、各拠点内のアクセス回線によって構成されており、OSPFv2 による IGP および iBGP/eBGP による経路制御が行われている。

4.7 実験データの諸元

シミュレーションに入力するデータとして、以下の期間・条件で収集された実際のトラフィックログを使用した。

- **対象期間:** 2026 年 1 月 16 日 04:42 ～ 04:57（約 15 分間）
- **データソース:** ネットワーク内の主要ルータから送信される NetFlow v9 および sFlow データ。Elastic Common Schema (ECS) 形式で Elasticsearch 基盤に蓄積されたログから、15 分間のレコードを取得して使用した。

- **サンプリングレート:** ルータの機種設定に基づき、1/1024 ～ 1/2048 の範囲でサンプリングされている。

4.8 比較検証用基盤

デジタルツインによる推定結果の妥当性を検証するために、オープンソースの統合監視ソリューションである Zabbix を利用した。

実ネットワーク上の全ルータおよび L2 スイッチは SNMP ポーリングを通じて常に監視されており、各物理インターフェースのトラフィックカウンタが 1 分間隔でデータベースに記録されている。本実験では、Zabbix API を使用してフロー情報と同様の期間のヒストリデータを取得し、シミュレータが算出した「リンク負荷の推定値」と突き合わせることで、シミュレーション精度の定量的評価を行なった。

第5章　評価と考察

本章では、構築したデジタルツインシステムの有効性を検証するために行った実証実験の結果について説明する。評価は主に「トポロジ構築の網羅性」、「推定精度の検証」、「運用上の有用性」の観点から行った。

5.1　評価環境と手法

実証実験は、WIDE プロジェクトが運用するバックボーンネットワークを対象に行った。比較対象の正解データとして、運用監視システムである Zabbix に収集された SNMP インターフェースカウンタの値を用いた。

- **対象期間:** 2026 年 1 月 16 日 04:42 ～ 04:57（約 15 分間）
- **トラフィックデータ:** Elasticsearch より取得した NetFlow/sFlow レコード サンプリングレート [1/1024 ～ 1/2048]
- **検証指標:**
    1. **トポロジ構築の網羅性:** 従来のトポロジ構築の手法である「CLI 解析」および「コンフィグ解析」によるノードおよびリンクの検出数を比較した。
    2. **ノード負荷の推定精度:** シミュレータが算出した bps 値と、Zabbix の実測値の相関および誤差率を調べた。
    3. **有用性検証:** 運用意図の注入と反映度を調べた。

5.2　トポロジ構築手法の網羅性と優位性

対象とした WIDE のバックボーンネットワークでは長年、管理図面が更新されておらず、その間に絶えず設定変更や機器の入れ替えがあったために、完全な正解データを得ることができなかった。そのため、本研究では CLI ベースの解析とコンフィグベースの解析それぞれと、提案手法を比較し、生成されたトポロジの網羅性や実態の反映度を検証する。

5.2.1 比較対象手法

比較対象として、以下の 2 つの手法を採用した。

手法 1: CLI ベースの解析　各ルータに SSH/Telnet でログインし、show ip ospf neighbor 等のコマンド出力を収集・解析して隣接関係を特定する手法。実稼働状態を取得できる反面、全ルータへのログイン権限が必要となる。

手法 2: コンフィグベースの解析　Oxidized 等のツールを用いて収集・管理されている各ルータのコンフィグファイルを、ネットワーク構成検証ツール Batfish を用いてベンダー差異を抽象化した形式に書き換えて L3 トポロジを生成する手法。大規模な解析が可能だが、コンフィグ管理リポジトリに含まれていない機器や、動的なリンクステート状態は反映されない。

5.2.2 構成要素の検出数比較

上記の 2 つの手法と、提案手法それぞれによって生成されたネットワーク構成要素数を表 5.1 に示す。

表 5.1: 従来手法群と提案手法による検出要素数の比較

要素	手法 1	手法 2	提案手法
Active Routers	35	33	**45**
Links	23	72	**116**

5.2.3 ノード識別精度の定量的評価

生成したトポロジ上のノードを一意に決定する Router ID の識別精度を検証するため、提案手法で検出された 45 ノードを基準とし、各手法の一致率を表 5.2 に示す。

表 5.2: Router ID の識別精度および一致率の比較

比較手法	検出ノード数	提案手法との一致数	一致率
手法 1	35	33	73.3%
手法 2	33	20	44.4%
提案手法	**45**	**45**	

5.2.4 考察: 提案手法の優位性

上記の結果より、提案手法は網羅性と精度の双方において従来手法を大きく上回ることが確認された。それぞれの要因について考察する。

- **CLI 手法のリンク検出数の少なさ:** CLI 手法のリンク検出数が 23 本と極端に少ないのに対し、提案手法は 116 本を検出した。CLI 手法では、直接隣接するルータしか認識することができない。よって、ルータ間に L2 スイッチなどが入っていた場合、その先を追うことができない。一方、提案手法ではそうした際も Network LSA として認識することができる。また、SSH 等が設定されていない機器がいるとその先の機器を追うことができないのも欠点である。

- **Batfish の識別精度の低さ：** コンフィグ解析において、Router ID の一致度が著しく低くなっていた主な原因は、OSPF 設定における Router ID の明示的な設定漏れにある。多くのベンダーの機器では Router ID が未設定のままでも OSPF を扱うことができる。その場合、デフォルトではルータに設定されたインターフェースの IPv4 アドレスから自動で選出される。しかし、Batfish はどのインターフェースのアドレスが選出されたのか解決できず、代替としてホスト名をノードの識別子として扱うケースが多発してしまった。一方、提案手法では LSA か実際に広告されている Router ID を抽出するので、設定の不備やベンダー仕様の差異に影響を受けなかった。

以上の結果より、提案手法は設定ファイルの網羅性やベンダーごとの挙動の違いなどに左右されず、現在のネットワーク構成を最も忠実かつ網羅的に再現できる手法であると言える。

5.2.5 ノード負荷推定の精度検証

本節では、第 3 章で提案したフロー情報を用いたトラフィックシミュレーションの精度を評価する。フロー情報を元に行われたシミュレーションにより求められた各ノードの負荷と、運用監視システム Zabbix から取得した実測負荷の比較評価を行った。

シミュレーション上の観測点と物理インターフェースの紐付け

本研究のシミュレーションでは、フロー情報を生成したトポロジ上で経路計算して、重ね合わせていく。一方、Zabbix の監視データは、ルータの各インターフェースごとに実測値が記録されている。

本システムのトポロジは OSPF 上での Router ID しか持たず、一方で Zabbix は OSPF の情報を持たない。そのため、デジタルツインによる推定値と実測値の比較を行うにあたり、まずトポロジ上の論理ノードと Zabbix 上の物理デバイスの対応付けを行わなければならなかった。フローの観測点となっているノードだけは Zabbix 上のどのノードかわかっていたため、そのノードでの結果を分析した。結果は以下のとおりである。

表 5.3: 各観測ノードにおける推計負荷と Zabbix 実測値の比較

ノード名	ポート数	提案手法 (Mbps)	実測値 (Mbps)	誤差
Router_6000_7	20	1649.27	2503.59	34.12%
Router_1800_0	9	899.34	1321.13	31.93%
Router_6000_6	5	231.55	231.73	0.08%
Router_6400_6	3	0.002	0.00	-

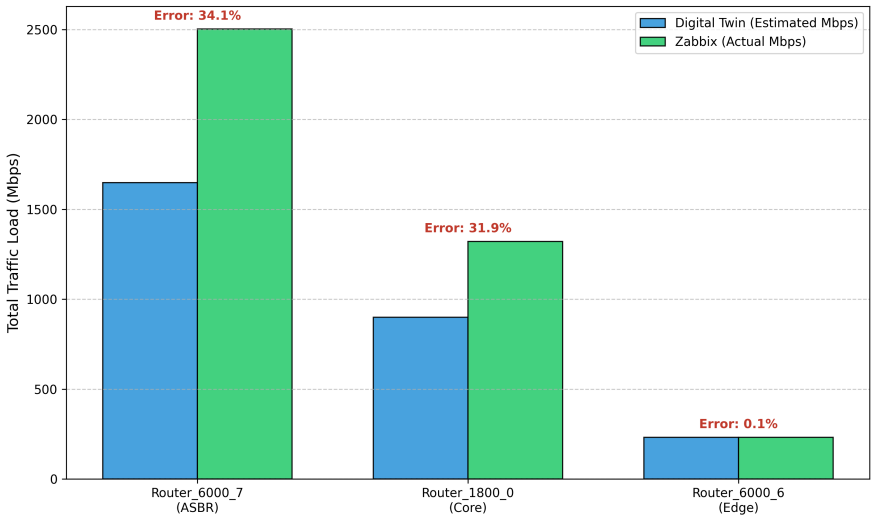


図 5.1: リンク負荷のシミュレーション値と Zabbix 実測値

表 5.3 より、以下の傾向が確認できた

シミュレーションロジックの正当性　Router_6000_6 において、推定値 231.55Mbps に対して実測値 231.73Mbps が得られ、誤差が 0.08% という極めて高い一致が見られた。このノードはトポロジ上ではエッジノードであり、本システムで使用した最短経路計算およびフローを用いたトラフィック再現が、現実のネットワークのパケット転送の挙動を再現していると考えられる。またこのノードにおいて推

定値および実測値のトラフィック量が少ないことから、不要なトラフィックがトポロジ上に現れておらず、経路計算が適切に動いていると考えられる。

大規模ノードにおける誤差の要因　Router_6000_7（ASBR）および Router_1800_0（Core）において、推計値が実測値を約 30%以上下回る乖離が確認された。この要因を特定するため、フローデータおよび解決ログの定量的調査を行った。その結果、以下の 2 点が主因であることが判明した。

- **エイリアス解決不能な外部 IPv6 トラフィック:** 本システムは OSPFv2 トポロジを基盤とするため、フローデータ内の IPv6 アドレスを router_alias.json を用いてルータ ID（IPv4）へ紐付けている。しかし、対外接続点である ASBR を通過する通信の多くは組織外のグローバル IPv6 アドレスであり、これらはエイリアスリストに存在しないため、最短経路計算の対象から除外される。
- **サンプリングバイアスと累積誤差:**1/1024 ∼ 1/2048 という高いサンプリングレート下では、定常的なフローは補正可能だが、基幹網特有のマイクロバーストや短命なフローを捕捉しきれず、過小評価につながる傾向がある。

一定の乖離は見られるものの、ノード負荷の相対的な大小関係は実測値と一致している。よって、ネットワーク内の潜在的な輻輳地点を予測するという目的においては実用可能な範囲だと考える。また、フローの観測点やサンプリング数を増やしたり、サンプリングを考慮したトラフィックのシミュレーションを行えば精度も上がるのではないかと考えられる。

誤差の定量的分析とロジックの正当性　誤差の原因がデータ網羅性に起因することを証明するため、各ノードにおける「名前解決不能な IPv6 トラフィック」の比率を算出した。

調査の結果、全トラフィックの 38.05%が IPv6 であり、そのうち ASBR（Router_6000_7）を通過するトラフィックの 30.24%がエイリアス解決不能な組織外通信であった。これは、同ノードで報告された誤差（34.12%）と極めて近い値となっている。対照的に、エッジノード（Router_6000_6）では、IPv6 の比率自体は 11.89%存在するものの、そのほとんどが組織内ノードとして解決可能であり、解決不能な通信は 0.23%に留まっていた。このノードで誤差 0.08%という極めて高い精度が得られた事実は、入力データ（エイリアス）が完備されていれば、最短経路計算および負荷重畳のアルゴリズム自体は極めて正確に動作していることを裏付けている。以上の分析より、本システムの推計精度はロジックの欠陥ではなく、参照するエイリアス情報の網羅性に依存しており、今後の IPv6 エイリアス管理の拡充によって大幅に改善可能であるとの見通しを得た。

5.3　インテント・リゾルバによる運用シナリオの評価

本節では、第 4 章で実装した運用意図解決エンジン（インテント・リゾルバ）の有効性を検証する。実際にインテント・リゾルバが想定している意図を 3 つのシナリオに分類して想定通りに解決できるか実験を行った。

5.3.1　カテゴリ A：トラフィック負荷分散と輻輳緩和

本カテゴリでは、インテント・リゾルバの感度分析によって、特定のリンクに集中した負荷を最小限のコスト変更で分散できるかどうか検証した。

```
{
  "policies": [
    {
      "id": "scenario_A1_thinning",
      "type": "thinning",
      "description": "【輻輳緩和】8->0 リンクのフローに対し、10%の流量を
迂回させるコストを確認する。",
      "target_link": ["203.178.136.8", "203.178.136.0"],
      "reduction_ratio": 0.1
    },
    {
      "id": "scenario_A2_ecmp_balance",
      "type": "load_balancing",
      "description": "【負荷分散】コアリンク（0-8）とバイパス（1-50-8）
のコストを、双方向調整ロジック（v11 以降）により完全に一致させ、トラフィッ
クが 50:50 に分散されることを証明する。",
      "path_a": ["203.178.136.1", "203.178.136.50", "203.178.136.8"],
      "path_b": ["203.178.136.1", "net_203.178.136.102", "203.178.136.10", "203.178
    }
  ]
}
```

シナリオ A1：バーストトラフィックの段階的間引き

本シナリオでは、203.178.136.0 =¿ 203.178.136.8 間のリンクに対して、その 10% の流量を迂回させる意図を検証した。OSPF はそもそもリンクコストのみでトラフィックの挙動が決まるため、細かく流量を調整するのが難しいがそれがどこまで可能かを評価する狙いがある。

検証結果と考察: インテント・リゾルバによって、現状コスト **475** に対し、他のフローへ影響を与えずに一部を迂回させる中間値として **546** が算出された。これを適用したトポロジを用いたシミュレーションの結果、当該リンクの最大負荷は **1262.12 Mbps** から **227.55 Mbps** へと大幅に低減された。

指定した削減割合10%を大きく上回る削減結果となってしまったが、これはOSPFがフロー単位の細かな制御ではなく、宛先プレフィックス単位で経路を決定する特性が原因である。インテント・リゾルバは、当該リンクを通過する複数のフローのうち、迂回させた際の合計流量が指定されたしきい値を初めて超える最小のコスト増分を感度分析によって特定した。その結果、1262.12 Mbps を占めていた主要なトラフィック群が別経路を選択したため、結果的に約 80%の負荷削減が達成されたと考えられる。

また、コストを極端に大きく設定してリンクを遮断するのではなく、546 という中間値を選択しており、これにより、当該リンクを本来通るべき近隣ノード間の通信を維持しつつ、遠隔拠点からのバーストトラフィックのみを狙い通りに迂回させることには成功している。

シナリオ A2：等コストマルチパスの自律形成

ここでは、メイン経路（.0-.8）とバイパス（.1-.50-.8）のコストを意図的に一致させ、ECMP を成立させることで負荷を 50:50 に分散する検証を行った。

検証結果と考察: インテント・リゾルバによって、指定された 2 つの経路の積算コストを一致させるため、複数のリンクコストが再計算された。具体的にはメイン経路上のリンク（.0 → .8）のコストを **475** から **1** へ、同時にバイパス経路上のリンク（.50 → .8）のコストを **5** から **2511** へと大幅に調整した。

この極端なコスト調整は、OSPF トポロジ全体の整合性を保ちつつ、対象となる 2 拠点間においてのみ合計メトリックを完全に一致させようとした結果である。

シミュレーションによる効果測定: このコスト設定を反映したデジタルツイン上でトラフィックシミュレーションを実施した結果、以下の効果が確認された。

- **トラフィックの等分割**: 以前のトポロジではPathBに集中していた約963.5Mbpsのトラフィックが、リゾルバによるコスト調整（リンク 0–8 をコスト 1 へ、50–8 をコスト 2511 へ変更）の結果、各パスへ 481.76 Mbps ずつ正確に分散された。これにより、特定のリンクに集中していた負荷を 50% 低減させることに成功し、網全体のリソース利用効率が大幅に向上したことが実証された。

- **自律的な経路生成**: 管理者は各リンクの具体的なコスト値を試行錯誤することなく、「2 つのパスを等分する」という抽象的なインテントを定義する

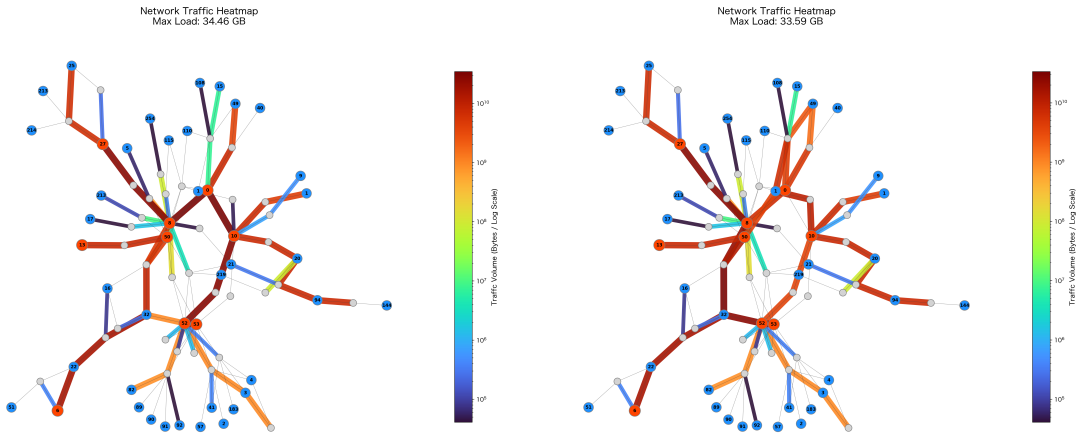
だけで、OSPF の制約下で複雑なメトリック設計を完結させることが可能となった。

以上の結果から、本システムは大規模で複雑なネットワークにおいても、トポロジの整合性を破壊することなく、管理者の意図に基づいたトラフィックエンジニアリングを自動で実行できる能力を有していることが実証された。

ネットワーク負荷分布の変化（Before/After）

本項では、インテント適用前後のトラフィック分布の変化を、可視化ツールによるヒートマップ（図 5.2）を用いて定量的に評価する。

ヒートマップによる視覚的評価　ヒートマップでは、各リンクのトラフィック積算量を色の変化と線の太さで表現している。



(a) Before: 特定リンクへの負荷集中　　(b) After: インテント適用による負荷分散

図 5.2: インテント適用前後のトラフィックヒートマップ比較

適用前（図 5.2a）において濃い茶色で表示されていたコアリンク（.0-.8）付近の負荷が、適用後（図 5.2b）では橙色へと変化していることが視覚的に確認できる。これは、インテント・リゾルバが算出した非対称コスト調整により、最短経路が再計算され、トラフィックがネットワーク全域に平準化されたことを示している。

5.3.2　カテゴリ B：保守・障害シミュレーションと運用安定性

本カテゴリでは、ネットワークの保守作業や予期せぬリンク故障を想定し、デジタルツイン上でトラフィックの再収容が正常に行われるかを検証した。

シナリオ B1：中心ノードの保守に伴うトラフィック排除

本シナリオでは、コアノードである 203.178.136.10 の保守を想定し、当該ノードに隣接する全インターフェースのコストを最大値に設定する Drain ポリシーを適用した。

検証結果と考察: リゾルバは対象ノードに隣接する全てのリンク（.0、net_...102、net_...185 等）を特定して、一括でコストを 65535 へ変更している。

シナリオ B2：主要リンク故障時の代替経路推計

本シナリオでは、最もトラフィック密度の高い .0 – .10 間のリンクを論理的に遮断した際、網内の到達性が維持されるかを検証した。

検証結果と考察: シミュレーションの結果、当該リンクの流量が 0 Mbps となっていることが確認された。一方で、経路割り当てができなかったフローは検出されず、総フロー数 156,456 に対する到達性の喪失は見られなかった。

これは、デジタルツイン内の最短経路計算が、リンク遮断を検知した瞬間にトポロジ内の代替経路を即座に計算し、全トラフィックを再収容した結果である。管理者は具体的な代替経路を指示することなく、単に「リンクを閉じる」という意図を定義するだけで、障害発生時の網全体への影響を事前に、かつ正確に把握できることが証明された。

カテゴリ B における負荷分布と到達性の変化

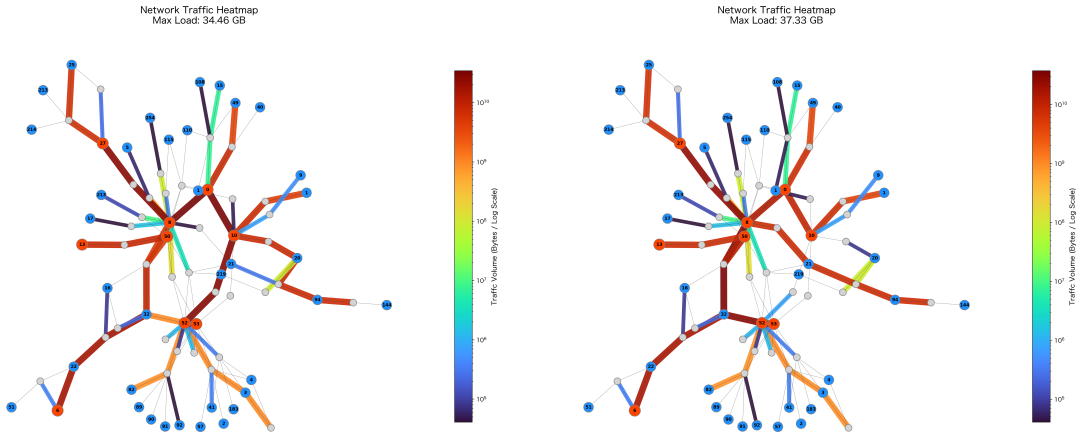
カテゴリ B のインテント適用前後の視覚的変化（図 5.3）および、排除操作に伴う主要指標の変化（表 ??）を以下に示す。

5.3.3 カテゴリ C：QoS とサービス品質最適化

本カテゴリでは、網全体のサービス品質を維持しつつ、特定のパスの遅延最適化や設定変更時の副作用を最小化するための検証を行った。本システムが、単なるコスト計算に留まらず、運用リスクを考慮した「安全な実行手順（Safe Sequence）」を自律的に構成できるかを評価した。

シナリオ C1：低遅延制御と負荷分散

本シナリオでは、高負荷状態にあった .190 → .52 間のリンク負荷を軽減するため、周辺リンクのコストを動的に調整する「Smart Low Latency」ポリシーを適用した。



(a) Before: 中心ノード .10 への流入　　(b) After: .10 回避経路への遷移

図 5.3: 保守排除インテント適用前後のヒートマップ比較

表 5.4: カテゴリ C におけるインテント適用前後の流量変化

対象リンク	適用前 (Mbps)	適用後 (Mbps)	変化
.190 → .52 (高負荷路)	927.98	396.23	-57.3%
.8 → .50 (迂回受入路)	0.00	481.61	New Flow

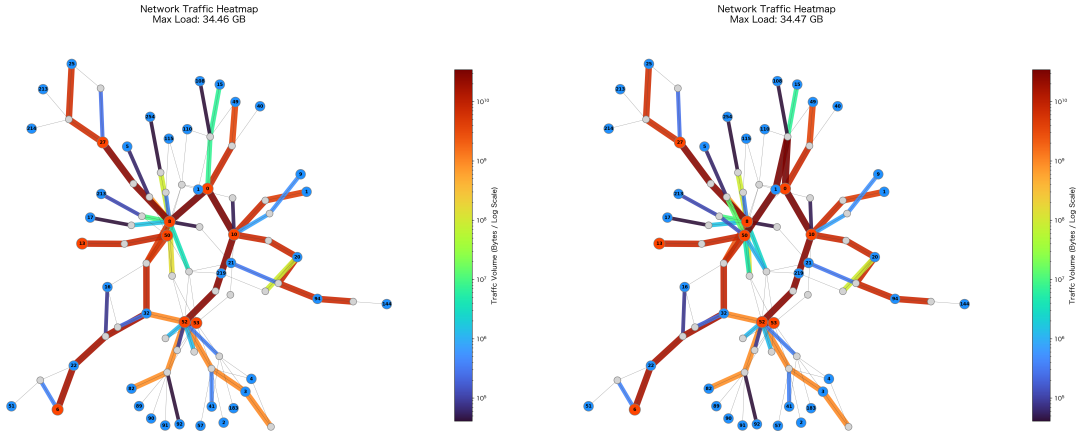
検証結果と考察:　インテント・リゾルバは、単一のリンク操作ではなく、周辺 5 箇所のリンクコストを多角的に調整する実行プランを導き出した。具体的には、迂回先となるリンク（.190 → .52）のコストを 11,000 から 3,004 へ引き下げることでトラフィックを引き込み、同時に過密な既存経路のコストを 3,004 へ引き上げることでトラフィックを押し出すという、多段階の制御を導出した。

この結果、表 5.4 から読み取れるように、当該リンクの負荷を 57.3%削減することに成功した。これは、デジタルツインが網全体のフロー分布を把握しているからこそ可能な最適化であり、一部のリンクを閉塞させることなく、QoS 要件を満たす意図が達成されたと言える。

シナリオ C2：副作用を排除した安全圏の自動探索

本シナリオでは、管理者がリンクコストを変更する際、他の経路に予期せぬ影響を与えない「安全な設定範囲」をデジタルツインが提示できるかを検証した。

検証結果と考察:　リンク .1 → .50 に対して感度分析を実行した結果、リゾルバは **1 – 999** という具体的な安全値を算出した。この範囲内であれば、OSPF の再計算が発生しても、特定の重要経路における輻輳が発生しないことがデジタルツイン上の全フロー・シミュレーションによって事前に保証された。



(a) Before: リンク .1–.50 変更前　　(b) After: コスト変更後

図 5.4: 低遅延制御適用前後のヒートマップ比較

5.4　提案システムの優位性に関する考察

5.4.1　運用の抽象化によるヒューマンエラーの低減

従来のネットワーク運用では、管理者がトラフィックを迂回させる場合に、トポロジ全体を把握した人が適切なコスト値を手動で算出し、各ルータへ設定を投入する必要があった。また、WIDE バックボーンのように全容の把握が困難な場合も想定される。これに対し、本システムは「排除」や「負荷分散」といった抽象的な意図を記述することで、具体的なコスト値を導出する。これにより、複雑なメトリック計算を行うことなく、パラメータ設定ミスや計算ミスといった人為的要因によるネットワーク障害のリスクを大幅に低減できることが示された。

5.4.2　デジタルツインによる「事前保証」と副作用の可視化

本システムの最大の優位性は、実環境への設定投入前に、デジタルツイン上で現実と同様のフローの挙動をシミュレーションできる点にある。カテゴリ C で検証した「安全圏の探索」機能により、あるリンクのコスト変更が、一見無関係に見える遠隔のリンクにおいて輻輳を引き起こさないかを事前に把握できる。この「設定変更が網全体に与える影響の定量的な予測能力」は、ネットワーク運用において、保守作業に伴うコストを劇的に削減するものである。

5.4.3　安全手順による過渡的障害の回避

単一の設定値の妥当性だけでなく、設定投入の「順序」に着目した点も本システムの重要な要素である。大規模なネットワークでは、複数のルータの設定を一括で変更する際、その投入順序によって一時的なルーティングループや局所的な輻輳が発生することが知られている。本システムが生成する安全手順は、トラフィックを「引き込む」操作を先行させ、逃げ道を確保してから「押し出す」という、ネットワーク工学的な知見に基づいたフェーズ分けを行っている。この順序制御機能により、定常状態のみならず、設定変更という「動的なプロセス」における可用性が担保される。

5.4.4　推計精度による信頼性の確保

本システムの実用性を支える基盤として、推計精度が挙げられる。WIDE プロジェクトの実データを用いた検証において、絶対精度における約 30%の乖離は、主に組織外 IPv6 通信の解決不能に起因することが定量的に特定された。しかし、運用実務における What-if 分析では、特定のリンクコスト変更に伴う「トラフィック遷移の傾向（トレンド）」や「相対的な負荷の増減」の把握が極めて重要である。

本システムは、解決可能なフロー群に対しては0.1%精度の再現性（エッジでの検証結果）を有しており、ネットワーク内の「どちらの経路がより混雑するか」という比較評価においては十分に実用的な判断材料を提供できる。今後は、OSPFv3（IPv6）のLSDB解析を統合し、エイリアス解決に依存しないトポロジマッピングを実装することで、絶対精度のさらなる向上と、より複雑なトラフィックエンジニアリングへの応用が期待される。

第6章　結論

6.1　まとめ

本研究ではOSPFをベースとした大規模なネットワークを対象として、実際の運用データに基づいたデジタルツインの提案・実装を行った。従来のネットワーク運用において「設定変更の影響予測の困難さ」および「ヒューマンエラーのリスク」という課題があったが、シミュレーションによる事前検証と運用意図を入力としたトポロジ操作というアプローチで解決を図った。

本研究の主な成果は以下の3点である。第一として、実機ルータのLSDBとNetFlow/sFlowによるフロー情報を動的に解析することで、L3トポロジ構成とトラフィック分布をデジタルツイン上に忠実に再現する手法を確立した。WIDEプロジェクトのバックボーンネットワークを対象とした実証実験では、実測値との誤差は認められたものの、フローの観測点やサンプリング数を調整すれば精度の向上も期待できる。

第二に、運用者の抽象的な意図を入力として受け取り解釈し、OSPFのリンクコストに自動で変換するアルゴリズムを開発した。これにより、運用者は「低遅延化」「負荷分散」といった目的を記述するだけで、複雑なコスト計算を行うことなく必要な設定値を求めることが可能になった。

第三に、設定導入時の意図しない障害を回避するための安全な手順を生成する手法を実装した。各変更操作の障害を引き起こすリスクを評価し、動的な設定変更プロセスにおける可用性を担保した。

以上の成果により、デジタルツインを用いた意図ベースのネットワーク制御が、次世代のネットワーク運用自動化において極めて有効な基盤技術となり得ることを示した。

6.2　今後の展望

本研究で構築したシステムは、実運用データに基づいた高度なシミュレーションを実現したが、実用化に向けては以下の課題が残されており、今後の展望として挙げる。

2. **IPv6環境への完全対応とOSPFv3の統合**

本研究で特定した外部IPv6トラフィックによる推計誤差を解消するため、OSPFv3

の LSDB 解析および IPv6 ベースのフロー重畳ロジックを統合し、エイリアス情報の網羅性に依存しない、より高精度なデジタルツインの構築を目指す。

2. **フロー観測点の拡充による推定精度の向上**

本研究では特定の観測点から得られたフロー情報に基づいたが、ネットワーク全体のトラフィックをより詳細に把握するためには、観測ポイントの動的な拡充が必要である。より多地点のフローデータを収集し、さらに精緻に分析することで、デジタルツインの再現性を限りなく実環境に近づけることが期待される。

3. **フロー情報のサンプリング特性を考慮したシミュレーションの実装**

本研究ではサンプリングされたフロー情報を利用したが、サンプリングには統計的な偏りがあり、特にバースト的なトラフィックや小規模なフローの検出には限界がある。サンプリング率に基づく統計的な推計モデルをシミュレータに統合し、不完全なフロー情報から網全体のトラフィック分布をより正確に復元する手法の実装が必要である。これにより、マイクロバーストによる瞬間的な輻輳予測など、より精緻なシミュレーションが可能になると考えている。

4. **意図記述の高度な抽象化と AI による複雑な意図分析**

本研究では、運用意図は定型化されたカテゴリに依存しているが、将来的には大規模言語モデル等の AI 技術の活用により、自然言語による複雑なポリシー解釈が必要である。例えば「対外接続の遅延を最小化しつつ、特定リンクの予備帯域を 30%確保せよ」といった複合的かつ高度な意図に対し、その意図を AI が正しく解釈し、デジタルツイン上で数万通りの設定変更パターンを自動検証。副作用が最小となる最適解を提示するインターフェースの構築が期待できる。

5. **自動最適化アルゴリズムへの応用と自動適用**

本研究では運用者の意思決定を支援するに留まっているが、デジタルツイン上で算出された最適解を、実ネットワークへ自動的にフィードバックするクローズドループ制御への拡張が必要である。ネットワークの状態変化をデジタルツインがリアルタイムに検知し、自律的にトラフィックを再配置するようなシステムの実現が次の目標である。

6. **複数プロトコルおよびマルチエリアへの対応**

本研究では単一エリアの OSPFv2 を主対象としていたが、BGP とのより密接な連携や、Segment Routing、IPv6 環境への適応についても検証が必要である。複数のルーティングプロトコルが混在する環境において、本システムが提供する「事前検証能力」はさらにその重要性を増すと考えている。

謝辞

本論文の完成にあたり、多くの方々から多大なるご指導とご鞭撻を賜りました。本研究にご協力いただいた全ての皆様に深く感謝いたします。

主指導教員である宇多 仁 准教授には、終始熱心かつ的確なご指導を賜りました。特に、本研究の根幹となる実運用データを用いた検証にあたっては、WIDE プロジェクトをはじめとする関係各方面へのご紹介や、外部機関との連携に向けた多大なるご調整をいただくなど、多方面にわたり多大なるご尽力を賜りました。先生の積極的なお力添えと橋渡しがなければ、本研究をこれほど実効性の高い形にまとめ上げることは叶いませんでした。深く御礼申し上げます。

副指導教員のリム 勇仁 教授をはじめとする丹 康雄 教授、BEURAN Razvan Florin 准教授には、中間発表ならびに本論文の審査において、客観的かつ専門的な立場から的確なご意見をいただきました。深く感謝いたします。

また、篠田 陽一 特任教授には、在学中、同じ研究室にて、日頃より暖かな励ましと多くのご助言を賜りました。日々新たな視点や気づきを与えてくださり、深く御礼申し上げます。

本研究のフィールドとして貴重な実運用環境を提供していただきました WIDE プロジェクトの関係各位、ならびに運用に携わる皆様に深く感謝いたします。実際のバックボーンネットワークから得られた LSDB や Flow データなくして、本研究のデジタルツインの構築および精度検証は成し得ませんでした。実運用の最前線に触れる機会をいただいたことは、私にとってかけがえのない経験となりました。

本研究室の高田 敦生 氏、金田 昂大 氏、中村 一貴 氏、Lu Xingche 氏、斉藤 翼 氏、岡坂 直徒 氏には、日々のゼミや研究室での議論を通じて、多くの刺激と励ましをいただきました。行き詰まった時に交わした何気ない会話や、技術的なトラブルの際に手を貸してくれた仲間のおかげで、最後まで研究を完遂することができました。深く感謝いたします。

最後に、これまでの学生生活を物心両面で支え、常に温かく見守ってくれた家族に、心からの感謝を捧げます。

関連図書

[1] Luciana S Buriol, Mauricio GC Resende, Celso C Ribeiro, and Mikkel Thorup. A hybrid genetic algorithm for the weight setting problem in ospf/is-is routing. *Networks: An International Journal*, Vol. 46, No. 1, pp. 36–56, 2005. 動的な最短経路計算と GA を組み合わせ、OSPF 重み設定問題を効率的に解く手法.

[2] A. Clemm, L. Ciavaglia, L. Granville, and J. Tantsura. Intent-Based Networking - Concepts and Definitions, 2022. 管理者が How ではなく What を宣言的に定義する IBN の概念を規定.

[3] Containerlab Project. Containerlab: defined by code, built for speed. `https://containerlab.dev/`, 2023.

[4] M Ericsson, Mauricio GC Resende, and Panos M Pardalos. A genetic algorithm for the weight setting problem in ospf routing. *Journal of Combinatorial Optimization*, Vol. 6, No. 3, pp. 299–333, 2002. 重み設定問題（NP 困難）に対し、GA を用いた初期の最適化アプローチ.

[5] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing ospf weights. In *Proceedings IEEE INFOCOM 2000*, Vol. 2, pp. 519–528. IEEE, 2000.

[6] H. T. Kaur, Tao Ye, S. Kalyanaraman, and K. Vastola. Minimizing packet loss by optimizing ospf weights using online simulation. In *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS 2003)*. IEEE, 2003. 待ち行列モデルを用いてパケット損失を最小化する OSPF 重みの動的最適化手法.

[7] Hema Tahilramani Kaur, Shivkumar Kalyanaraman, and Saroj Yadav. Dynamic optimization of ospf weights using online simulation. In *IEEE International Conference on Communications (ICC 2001)*. IEEE, 2001. リンク負荷の変動に対し、オンラインで OSPF 重みを動的に再構成する初期の重要研究.

[8] Satoru Kobayashi, Ryusei Shiiba, Ryosuke Miura, Shinsuke Miwa, Toshiyuki Miyachi, and Kensuke Fukuda. dot2net: A labeled graph approach for

template-based configuration of emulation networks. In *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023.

[9] Aris Leivadeas and Matthias Falkner. A survey on intent-based networking. *IEEE Communications Surveys & Tutorials*, Vol. 25, No. 1, pp. 625–655, 2022.

[10] J. Moy. OSPF Version 2, 1998. OSPF メトリックの最大値（65535）や SPF 計算の基本仕様.

[11] NS-3 Consortium. The ns-3 network simulator. `https://www.nsnam.org/`, 2023.

[12] Marco Polverini, Francesco Giacinto Lavacca, Jaime Galán-Jiménez, and Marco Listanti. A digital twin based framework to enable "what-if" analysis in bgp optimization. In *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023. BGP の設定変更がトラフィックに与える影響を事前に評価する What-If 分析フレームワーク.

[13] Reza Poorzare, Dimitris N Kanellopoulos, Varun Kumar Sharma, Poulami Dalapati, and Oliver P Waldhorst. Network digital twin toward networking, telecommunications, and traffic engineering: A survey. *IEEE Access*, 2025. トラフィックエンジニアリングにおける NDT の役割と進化を網羅した最新サーベイ.

[14] R. E. Tarjan. Sensitivity analysis of minimum spanning trees and shortest path trees. *Information Processing Letters*, Vol. 14, No. 1, pp. 30–33, 1982. 最短経路木が変化しない重みの範囲を特定する感度分析の理論的基礎.

[15] Mohamad Zalat, Maede Davoudzade, Chris Barber, David Krauss, Babak Esfandiari, and Thomas Kunz. A practical network digital twin for igp weight optimization. In *2024 20th International Conference on Network and Service Management (CNSM)*. IEEE, 2024. 実ネットワークを監視し、平均遅延や損失率を最小化する重み設定を自動適用する NDT を提案.

[16] H. J. Zhou, J. Pan, and P. B. Shen. Cost adaptive ospf. In *Proceedings of the Fifth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'03)*. IEEE, 2003. 利用率に応じて動的にコストを調整する CA-OSPF を提案.