博士論文

トラヒックエンジニアリングにおける負荷分散を考慮した機械学習によるリアルタイム経路設計法に関する研究

Study on Path Planning Method in Real Time using Machine Learning Considering Load-balancing for Traffic Engineering

令和6年2月

日本大学大学院 工学研究科

情報工学専攻・博士後期

伊藤 真

指導教員

源田 浩一

目 次

第1章　序論

増加し続けるインターネットトラヒックを起因とする様々な問題や課題への取り組みは重要な研究課題の1つである．そのような問題の1つに，ネットワーク内を流れるトラヒックの時間変動が大きくなることによる，急激なトラヒックの増減を起因とした輻輳の発生がある．一般的なインターネット通信は，ベストエフォート型のサービスであるため，トラヒックの優先度や重要度に関わらず，輻輳の発生によりトラヒックが破棄されてしまう可能性がある．しかしながら，通信時の経路はトラヒックの送信前に決定されるため，経路を通信途中で制御することはできず，ネットワーク状況に合わせたトラヒック制御が行えないためである．このような問題を解決するための技術として，トラヒックエンジニアリング (TE:Traffic Engineering) がある．TE とは，サービスの品質を維持しながら，時間変動の大きなトラヒックを効率的に制御する方法である [1]．具体的には，サービスの目的や要件に合わせて，多種多様な方式が検討されている [2] [3] [4] [5] [6] [7].

本研究では，輻輳制御と代替経路設計に焦点を当てる．先ず，輻輳制御のための最も簡単なトラヒック制御例は，複数のトラヒックの経路に同一リンクを使用しないように，一部の経路を迂回させることで，ネットワークの負荷分散を行うことである（図 1.1）．そのためには，ネットワーク全体の状況を考慮した集中管理型の経路設計を行う必要がある．次に，最も簡単な代替経路の設計例は，主経路に含まれるリンクやノードを一切含まないような経路を設計することで，ネットワーク内で故障が発生したとしても安定した通信を行うことである．また，本研究ではトラヒックの変動に対して，最適な経路を設計するような場面を想定しており，このような場面ではトラヒックデマンドに含まれる全てのフローを同時に再計算する手法が用いられている [8] [9] [10].

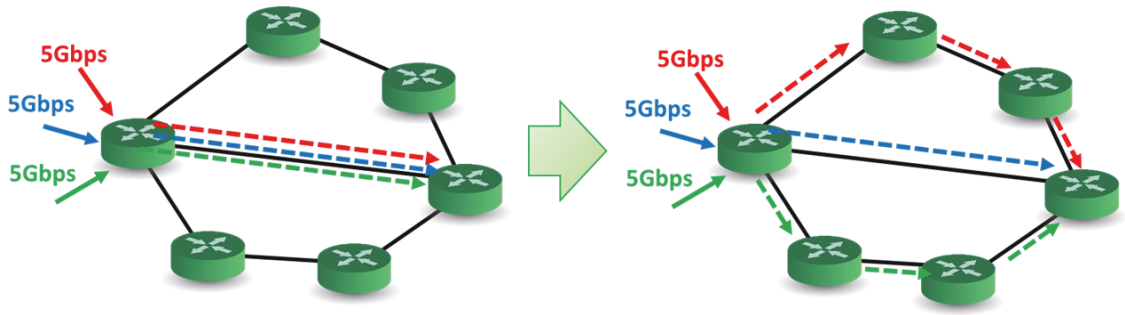


図 1.1 最も単純な輻輳制御例

トラヒックエンジニアリングには，オフライン型とオンライン型がある．オフライン TE では，定期的なトラヒックの観測によりノード間のトラヒック量を表すトラヒック行列を求め，これを基に経路設計を行う．これにより，単純な最短経路ルーティングと比べて最大リンク使用率を大幅に減少することができる．その一方で，オフライン TE では，長期的なトラヒック予測と実際のトラヒックには乖離があるため，観測期間と経路の最適化をどの程度の頻度で行うべきか検討することが重要である．そこで，オフライン TE よりも短い期間，即ちリアルタイムに近いトラヒック観測やフロー単位での通信を要求されるような場合に，最適な経路を設計する方式としてオンライン TE がある．オンライン TE は理論上最もネットワーク資源を効率的に利用できるものの，これらの経路は一般的に複雑なアルゴリズムや線形計画法 (LP:Linear Programing) によって求められるため，計算量やスケーラビリティの問題を無視することができないという課題がある．

このような経路計算における課題の解決に，機械学習技術を用いる研究 [11] [12] [13] [14] [15] [16] が近年見られる．文献 [11] [12] [13] では， 目的に合わせて深層学習モデルと教師データを設計し，教師あり学習により生成した学習モデルを用いて，拠点間トラヒック経路を設計させる方式を提案している．文献 [14] [15] [16] では，周期的にトラヒックの観測を行い，強化学習を用いてルーティングプロトコルに用いられるリンクコストの最適化 [14] や，ネットワーク内で高負荷の原因となっているクリティカルフローに対して経路の再計算を行う方式 [15] [16] を提案している．機械学習技術を用いる最大の利点は，経路の計算時間を大幅に短縮できることである．複雑なアルゴリズムや LP 法を用いた経路設計法では，一般的に膨大な計算時間を要し，最悪の場合は解を得ることができないこともある．そのため，計算時間に関する問題を無視することができない．これに対して，機械学習モデルを用いた場合の計算時間は，ニューラルネットワークの構成や規模に応じて増減するものの，一般的には即時性が高く，非常に短い計算時間である．また，教師データの作成や学習にかかる時間は長い時間を要することもあるが，あくまでも事前の準備時間であり，経路計算にかかる時間とは直接の関係がない．よって，経路設計アルゴリズムの持つ特徴を正しく学習した機械学習モデルを作成することで，計算時間の課題を解決できる可能性がある．また更なる利点として，サービスの目的や要件に応じた異なる方式やネットワーク機器を導入する必要がないことも挙げられる．学習モデルは教師データの作成時に用いる経路設計法を適宜変更することで，様々な経路を設計することができる．つまり，機械学習を用いた経路設計法の導入のみで，複数サービスを運用し得ることから，コストの引き下げやサービスの汎用性向上が見込まれる．以上より，機械学習技術を用いることで，従来の経路設計法における計算時間の課題を解決しつつ，様々な要件に対応することができる汎用性の高い経路設計法として期待できる．

しかし，従来の機械学習を用いた経路設計法には，多くの課題がある．一つは，機械学習モデルに設計させる経路を，送信元から宛先ノード間全体とした場合に，学習精度が著しく低下し，十分な経路設計が行えないことである．そのため，文献 [11] [12] では，精度

向上のために設計する経路情報を次ノードのみとし，ネットワーク内の全ノードが機械学習モデルを持つ分散型の経路設計としている．また，文献 [13] では，送受信間経路全体を出力するために，回帰型ニューラルネットワーク (RNN:Recurrent Neural Network) を用いて，通過すべきノード番号を制約条件として組み込むことで生成される教師データの種類を減らし，学習難易度を低下させ精度の向上を図っている．これらの方式では，ある拠点間トラヒックの経路最適化は図れるものの，複数の拠点間トラヒックを考慮したネットワーク全体の輻輳を回避する経路を設計できない．加えて，文献 [14] [15] [16] では，周期的なトラヒックの観測を前提としており，トラヒックの変化に対して全フローを瞬時に計算できないことや，クリティカルフロー以外の経路の最適化は行えないことから，最悪の場合，輻輳が発生する．よって，本研究の目的である，ネットワーク全体を考慮した複数の拠点間トラヒックからなるデマンド集合に対し，輻輳制御しつつ最短経路を設計するような機械学習を用いた集中管理型の経路設計法は提案されていない．

以上のような背景より，本研究では，ネットワーク全体を考慮した複数の拠点間トラヒックからなるデマンド集合に対して，輻輳制御しつつ最短経路を設計するような機械学習を用いた集中管理型の経路設計法を検討することを目的とした．はじめに，集中管理型の経路設計を行うための深層学習モデルと入出力信号を設計し，教師データにリンクコストを残余帯域の逆数としたダイクストラ法を用いて学習させた．学習モデルは様々なトポロジの構造や帯域の幅環境下を用いてハイパーパラメータのチューニングを行い，最適な構造を検討した．これらの学習モデルを用いて，小規模および中規模トポロジに対する評価実験を行った．先ず，経路設計成功率，最大負荷リンクにおける帯域使用率および経路計算時間の観点からダイクストラ法と比較評価を行った．加えて，関連研究で検討されている seq2seq モデルを用いた経路設計法と比較評価を行った．また，耐故障性を評価するために，トポロジの変化に対する汎化性能評価を行い，これらの結果から提案方式の有効性を示した．次に，教師データに線形計画法を用いることで，提案方式の更なる性能向上を図る．そこで，ネットワーク全体の収容率向上と平均ホップ数の最小化を目的とした整数線形計画法を提案した．この経路を教師とし学習したモデルを用いて，実験用の小規模トポロジにおいて，経路設計成功率と最大負荷リンクにおける使用帯域幅および経路計算時間の観点から，ダイクストラ法，ILP 法，Dijkstra ベースの機械学習モデル，seq2seq 方式らと比較評価を行った．これらの結果から提案方式の有効性を示した．　また，トラヒックエンジニアリングにおける代替経路の設計に対して，機械学習を用いた方式を検討した．代替経路は主経路と全く異なる経路であることが望ましいため，主経路に応じて適切に設計する必要がある．そこで，ある入力に対して同時に主経路と代替経路を設計する機械学習を用いた経路設計法を提案した．評価実験により，主経路と代替経路を同時に高い成功率で設計できることと，これらの経路が非常に低いリンク重複率であることを示す．

本研究の構成は以下の通りである．第 2 章では，関連研究の概要と課題を述べる．第 3 章では，輻輳制御のためのダイクストラ法ベースの深層学習を用いた経路設計法について

述べる．第4章では，教師データを改良し，輻輳制御のための整数線形計画法ベースの深層学習を用いた経路設計法について述べる．第5章では，代替経路設計のためのダイクストラ法ベースの深層学習を用いた経路設計法について述べる．第６章では，結論として本研究で得られた結果をまとめる．

第2章　関連研究

2.1　Traffic Engineering

2.1.1　Traffic Engineeing

一般的なインターネットにおけるベストエフォート型の通信では，通信品質は保証されないため，輻輳が発生した場合，トラヒックの優先度に寄らず破棄されてしまう．また，一度輻輳が発生してしまうと意図的にトラヒックを制御しない限り，状況は緩和されずトラヒックの発生量が減少するのを待つか，ネットワークを遮断するしかなく，ネットワーク全体への影響が大きくなってしまう．このようなトラヒックを起因とする問題の解決を図るための技術が，トラヒックエンジニアリング (TE:Traffic Engineering) である [1].

TE は，ネットワーク資源を効率的に使用できるように，トラヒックが流れる経路を制御することで，輻輳の回避や帯域幅の節約等の目的を達成とする．その一方で，実際のネットワークでは，様々な通信方式が混在しているため，対象とするネットワーク内で用いられる各種プロトコルや伝送方式に応じた TE を実現する必要がある．例えば，Internet Protocol(IP) をベースとしたものや，Generalized Multi-Protocol Label Switching(GMPLS) をベースとしたものが挙げられる [2] [3] [4] [5] [6] [7].

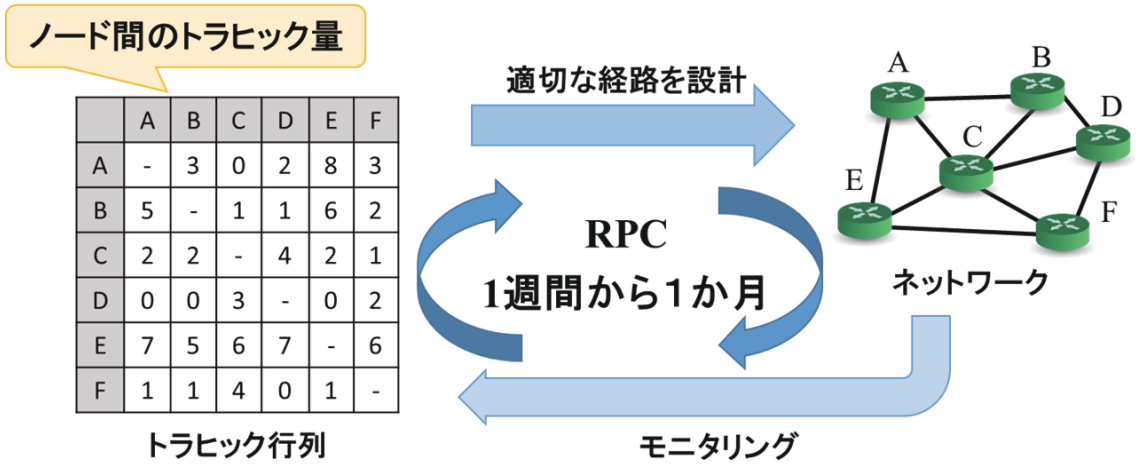
2.1.1.1　Offline Traffic Engineering

オフライン型のトラヒックエンジニアリングでは，観測により推定したトラヒック行列に対して，最適な経路を計算する方式である [1]．代表的に方式として，OSPF-TE や MPLS 方式が挙げられる．一般的に最短経路ルーティングと比較して，最大リンク使用率を大幅に減少させることができる．

図 2.1 はオフライントラヒックエンジニアリングの動作例を示したものである．オフライントラヒックエンジニアリングでは，ネットワークを定期的にモニタリングし，RPC と呼ばれる一定期間におけるノード間のトラヒック量を観測し，トラヒック行列を生成する．そして，このトラヒック行列を基に最適な経路を設計する．また，RPC は一般的に 1 週間から 1 か月程度であり，サービス設計者が決める．

オフライントラヒックエンジニアリングには大きく 2 つの課題がある [8]．一つは RPC の最適な期間を求めることである．経路設計のためのトラヒック行列は RPC により求め

られるものの，RPC は通常 1 週間から 1 か月程度であり，長期的なトラヒック予測による経路設計である．そのため，実際のトラヒックの発生頻度とは異なり，時間変動やトラヒック需要の変化，ネットワーク内外部機器の故障やサイバー攻撃といったトラヒックの急激な変動に対して最適な経路を設計することができない．もう一つはネットワーク故障に対する代替経路の最適化である．トラヒックエンジニアリングにおいては，特定の故障を想定し，事前に代替経路を設計しておくことで故障に備えた経路設計を行う．その一方で，ネットワークの故障には膨大な種類があり，事前に想定した故障が発生するとは限らない．そのため，事前に設計された代替経路では，常に適切な経路設計が行えないという課題がある．



	A	B	C	D	E	F
A	-	3	0	2	8	3
B	5	-	1	1	6	2
C	2	2	-	4	2	1
D	0	0	3	-	0	2
E	7	5	6	7	-	6
F	1	1	4	0	1	-

図 2.1 オフライントラヒックエンジニアリングの例

2.1.1.2　Online Traffic Engineering

オフライントラヒックエンジニアリングの課題を受け，研究されたのがオンライントラヒックエンジニアリングである [1]．オンライントラヒックエンジニアリングでは，分当たり，時間あたり等の RPC よりも短い期間で，リアルタイムにトラヒック行列やネットワーク故障を観測可能である場合や，拠点間のフロー単位での通信を要求された場合を想定した，トラヒックの変化に応じた経路設計を目的とする経路設計法である．既存方式として GMPLS や TeXCP が挙げられるものの，オフライントラヒックエンジニアリングと比較して，十分な検討はされていない．これは，オンライントラヒックエンジニアリングが理論上最もネットワーク資源を効率的に利用することが可能な一方で，現実的な実現のためには大きな課題が残されているためである．それは，リアルタイムに発生するトラヒック要求に対して，ネットワーク全体を考慮した経路設計は，現実的な時間では解けないような NP 困難な問題であるためである．その一方で，NP 困難な問題に対して，トラヒック

の変動に応じて短時間で経路計算を行う必要がある．これらの計算量とスケーラビリティの問題から，オンライントラヒックエンジニアリングにおけるリアルタイムな経路設計の実現は非常に困難である．

図 2.2 はオンライントラヒックエンジニアリングの動作例を示したものである．オンライントラヒックエンジニアリングでは，ネットワーク全体を考慮しつつ負荷分散した経路を瞬時に設計する事が求められる．この図では，ある状況下で新しく発生したトラヒックにより，ネットワーク内で輻輳の発生が検知された場合に，輻輳を回避するために可能な限り早く経路の再計算を行うような場面を表している．このような経路設計を行うためには，ネットワーク全体を考慮した集中管理型の経路設計が必要である．

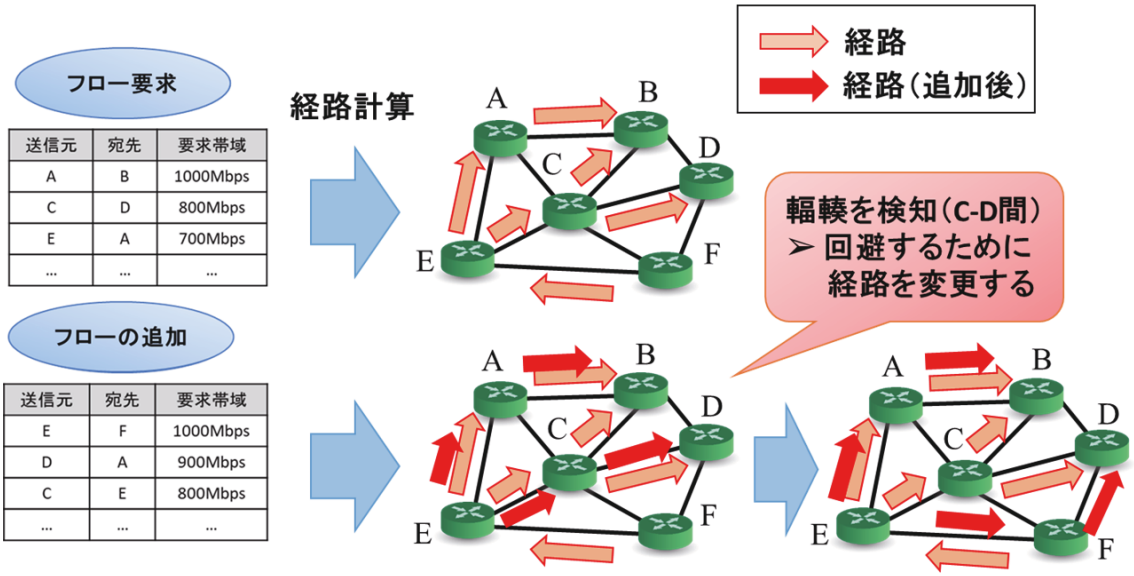


図 2.2 オンライントラヒックエンジニアリングの例

2.1.2 Traffic Engineeing の課題

トラヒックを起因とする問題の解決を図るための技術が，トラヒックエンジニアリング (TE:Traffic Engineering) である [1]．TE は，ネットワーク資源を効率的に使用できるように，トラヒックが流れる経路を制御することで，輻輳の回避や帯域幅の節約等の目的を達成する．その一方で，実際のネットワークでは様々な通信方式が混在しているため，対象とするネットワーク内で用いられる各種プロトコルや伝送方式に応じた TE が提案されている [2] [3] [4] [5] [6] [7]．TE における各経路設計法は，目的や要件に応じて優れた経路を設計できる一方で，計算時間の問題を無視することができない．特に，優れた経路を設計するためには，線形計画法 (LP:Linear Programming) を用いた膨大な計算を行う必要がある．線形計画法を用いた経路設計では，トポロジの規模に応じた計算時間の増加を避ける

ことができず，事実時間では解が得られないといった課題がある．更に言えば，トラヒックの時間変動は益々大きくなりつつ，全体のトラヒック量も増え続けていることから，最適な経路を計算し設計するための難易度は上がり続けている．それにもかかわらず, トラヒックの変動に対して最適な経路を設計するような場面 [8] [9] [10] では，可能な限り短い時間で経路設計を行うことが望ましい．事実時間では計算が難しい問題に対して，可能な限りリアルタイムに近い経路計算を行わせるという矛盾した課題の解決が求められている．

2.2 QoS 保障サービス

2.2.1 Integrated Service

2.2.1.1 Integrated Service

Integrated Services(IntServ) [17] は，IP ネットワークにおいて，トラヒックフローごとにリンク帯域などのネットワーク資源を確保することによって，高い QoS を保証するための通信サービスである．IntServ には，帯域と最大遅延を保証する Guaranted Service(GS) と，遅延の目標値を持って，フローを自ら制御する Controlled Load Service(CLS) の 2 つのモデルが想定されている．特に，CLS では，ネットワークが混雑している場合でも，利用率が低いベストエフォートネットワークと同等程度に動作することを目標としている [18].

これらのサービスを実現するためには，トラヒックデマンドの送信前に，トラヒックフロー毎のネットワーク資源を確保する必要がある．このとき，各トラヒックフローの経路上にある資源を予約するためのプロトコルとして，Resource reSerVation Protocol(RSVP) が規定されている．

2.2.1.2 RSVP

Resource reSerVation Protocol(RSVP) は，IP ネットワークにおいて，送信元から宛先までの帯域をあらかじめ予約することで，ネットワーク内リンクの QoS 保証を行うプロトコルであり，RFC2205 によって規定されている [19]．RSVP の特徴としては，マルチキャストとユニキャスト両方で使用可能な資源予約プロトコルである．一方向のデータフローに対して資源予約を行う．また，送信元ノードではなく，中継ノードや宛先ノードが予約を行うといったものが挙げられる．

IntServ における動作としては，はじめに送信元ノードが送信前にリンクの資源を確保するために RSVP の Path メッセージを宛先ノードに向けて送信する．Path メッセージを受信した中継ノードは，Path メッセージ内に自身を通過したという情報を書き込み，次のノードに転送する．この動作を宛先ノードに到着するまで繰り返す．Path メッセージを受信した宛先ノードは，リンク資源の予約を完了するために RSVP の Resv メッセージを送信元ノードに向けて送信する．このとき Resv メッセージは，Path メッセージが辿った経路を逆順に辿る．Resv メッセージを受信した中継ノードは，Path メッセージと同様に自身を通過したという情報を書き込む．その後，Resv メッセージが送信元ノードに到達することで，リンク資源の確保が完了する [18].

2.2.1.3　Int Serv の課題

IntServ を用いることで，QoS 保証が行えると考えられていたが，重大な問題点が複数指摘されている．代表的な問題は，スケーラビリティに関する問題であった．例えば，RSVP による資源予約は，ネットワーク上を流れるトラヒックフローの状態を，各中継ノードが保持する必要がある．特に大規模バックボーンネットワークでは，状態数がノード数に比例して増大してしまい，スケーラビリティが著しく低下する問題が挙げられる．

そのため，大規模なネットワークにおいては，IntServ による QoS 保証された経路の計算は不可能であった．加えて，RSVP プロトコル自体の複雑さも重大な欠点であり，運用への足かせとなっていた．これらの理由から，IntServ と RSVP は実用化されなかった．

2.2.2　Differentiated Services

Differentiated Services(DiffServ) [19] は，IntServ の様な厳密な QoS 保証を諦め，フローをクラスにまとめ優先度を付与する事で，資源の確保を行わずに QoS 保証を行う通信サービスである [18].

DiffServ では，エッジルータがトラヒックの送信前に所属クラスを表す Differentiated Services CodePoint(DSCP) という値を設定する．その為，各インナールータは DSCP 値によって異なる QoS 保証を行う様に，独立したパケットスケジューリングを行うだけで構わない．また，どの様な基準で DSCP 値を設定するかは，ISP があらかじめ設計する．

DiffServ を用いることで，比較的容易に QoS 保証サービスを提供する事が可能である．その一方で，ユーザが QoS 保証を実感し難いというサービスとしての決定的な問題点がある．原因として，サービス提供側はクラス毎に優先度を付与することで通信品質の保証を行っているが，実際に高優先度クラスのユーザが低優先度クラスのユーザよりも良いサービスを受けていると実感できるかと言えば難しい．

この様に，QoS 保証サービスとしてユーザの所望する品質レベルを提供できると断言できない事こそが大きな問題点である [18].

2.3 機械学習を用いた経路設計法

近年，機械学習技術を用いた研究が，様々な情報工学分野において盛んに行われている．学習難易度の観点から，ネットワーク工学の分野ではあまり積極的に機械学習技術が適用されてこなかった．しかしながら，近年の機械学習技術の発展により，トラヒック分析やトラヒック監視，経路制御などに適用する研究が行われるようになってきた．本章では，機械学習を用いて経路設計や経路制御を行う方式を検討している研究を紹介する．なお，各研究で用いられている機械学習技術は，深層学習 [11] [12]，回帰型深層学習 [13]，強化学習 [14] [16] [17] であり，様々な学習モデルが検討されている．

2.3.1 深層学習を用いた分散型トラヒック制御法

文献 [11] [12] は，モバイル機器を中心とした異機種間ネットワークにおけるトラヒックの急激な増加を制御する必要性を述べており，機械学習技術を用いたトラヒック制御法を提案している．従来の機械学習では，大規模で異機種間からなるネットワークが持つ動的な特徴を学習することが非常に困難であった．加えて，学習を行う上で最適な学習データをどの様に設計すべきか，検討が必要であった．そこで，近年急速に発達している深層学習に焦点を当て，深層学習モデルの各レイヤの構成と学習データの設計を行っている．

この方式では，深層学習を用いたトラヒック制御の実現のために，ネットワーク内の各ルータが深層学習モデルを持ち，分散制御する方式を導入している．このとき，各ルータの深層学習モデルへの入力信号は，ルータ自身に流入する直近 3 つのトラヒックパターンであり，トラヒックパターンは各ルータのインバウンドパケットを用いる．次に，深層学習モデルの出力信号は，経路の設計精度の観点から，送信元ノードから宛先ノード間全体の経路ではなく，次ホップとしてどのルータに送信するかのみ決定させている．なお，集中管理型の送受信間経路全体を設計する方式も検討されたが，最も良い場合でも経路設計成功率が 30%以下であったことから実現が見送られている．

評価では，Open Shortest Path First(OSPF) による経路と，機械学習による経路を比較している．その結果，機械学習によって設計された経路の方が，シグナリングオーバーヘッドが約 70%削減され，スループットが約 2%向上し，平均ホップ遅延が約 90%削減されたと述べられている．その一方で，分散型の経路設計法であることから拠点間における経路最適化は行えるものの，本研究の目的であるネットワーク全体を考慮した経路最適化は行えない．

2.3.2　回帰型深層学習を用いたトラヒック制御法

文献 [13] は，インターネットトラヒックの爆発的な増加によるネットワークの問題に対処するために，転送制約を考慮した機械学習型トラヒック制御法を提案している．特に，学習対象である経路情報に対して時系列性を見出し，自然言語処理で多く利用されている，Seq2Seq モデルを適用している．Seq2Seq モデルは入力データを有用な情報に変換する Encoder と，変換された情報を再度変換し時系列データを作成する Decoder という，2 つのリカレントニューラルネットワークから構成されている．加えて，学習モデルの性能を向上させるために，入力信号と出力信号の各要素の関連性を結びつける Attention メカニズムと，探索アルゴリズムの一種であり，評価値の高い解候補を上位から数個のみ選択することで探索必要な時間や資源を削減する Beam Search を適合し，経路の持つ順次的特徴を捉えることを狙っている．

この方式では，入力信号として送信トラヒックの送信元と宛先ノード番号および制約条件の三つを用いている．ここで，制約条件とは，必ず通過しなければならないノード番号である．出力信号は送信トラヒックに対する，制約条件を満たしたループの無い送受信ノード間の経路全体としている．また，教師データの作成は二つの工程に分かれており，はじめにダイクストラ法を用いて，ネットワーク内の全ノード間の静的な経路を探索する．その後，各入力信号に対して事前に計算された静的な経路を組み合わせて，拠点間経路全体を作成し出力信号としている．

評価では，ダイクストラ法による経路と機械学習による経路を比較しており，ダイクストラ法と同様な経路を設計できたと述べられている．また，学習データには含まれていない入力信号に対しても，目的を達成するような経路が設計できることが確認されている．その一方で，ある拠点間トラヒックの最短経路を設計するため，複数トラヒックの経路を設計させた場合に，ネットワーク全体を考慮した経路最適化は行えない．

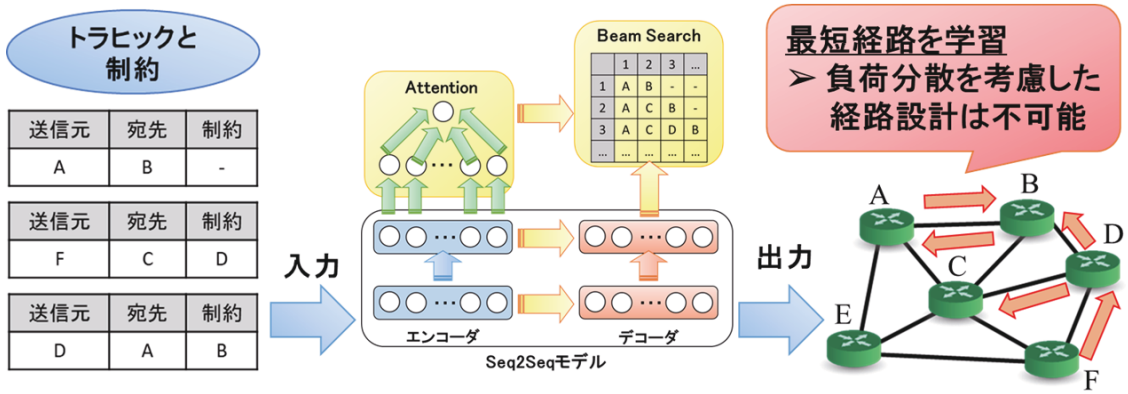


図 2.3 回帰型深層学習を用いたトラヒック制御法の動作例

2.3.3 トラヒックエンジニアリングのための強化学習を用いたリンクコスト最適化手法

文献 [14] は，トラヒックエンジニアリングにおける課題の解決に強化学習を用いた解決を図っている．この方式では，マルチエージェント強化学習 (MARL:Multi-Agent Reinforcement Learning) とグラフニューラルネットワーク (GNN:Graph Neural Networks) を組み合わせた学習モデルを用いて，測定されたトラヒックに対して，ネットワーク内の最大負荷を下げるようにリンクコストを再計算させる．これによって，ネットワークの輻輳を最小化することができる．

評価では，MARL+GNN とトラヒックエンジニアリングにおける制約プログラミングに基づくネットワーク最適化システムである DEFO [15] と比較している．結果として，MARL+GNN 方式が 3 種類の現実で用いられているネットワークトポロジにおけるネットワークシナリオにおいて，DEFO と同等の性能であると述べられている．加えて，実行時間の観点からは，DEFO と比較して大幅に短縮されており，分オーダから秒オーダへ改善されていると述べられている．

2.3.4 強化学習を用いたクリティカルパスにおける経路最適化手法

文献 [16] [17] は，機械学習を用いた TE において，ネットワーク内で高負荷の原因となっているクリティカルフローについて経路の再計算をすることで，ネットワーク内の帯域使用率の改善に貢献している．一般的にネットワーク内でトラヒックフローを頻繁に迂回させる場合，パケットの順番が維持できない問題などの悪影響は考慮されていない．これらの問題を解決するためには，一部の重要なトラヒックフローを選択的に迂回することで，帯域使用率の改善を達成する必要がある．しかし，適切なクリティカルフローの選択は膨大な計算を要するため，迂回路の設計は容易ではないという課題がある．

この方式は，強化学習と線形計画法を用いて，クリティカルフローの再ルーティングを目的とした CFR-RL 方式を提案している．CFR-RL 方式では，強化学習によって入力されたトラヒック行列に対して，クリティカルフローを選択するルールを学習し，選択されたクリティカルフローを線形計画法により再計算させる方式である．

評価実験により，CFR-RL 方式を用いることで，すべてのトラヒックのうち約 1 割から 2 割の経路を再設計することで，負荷を分散しネットワークの性能を向上できると述べられている．

その一方で，文献 [14] [16] [17] の方式は，周期的なトラヒックの観測を前提としており，トラヒックの変化に対して全フローを瞬時に計算することができないことや，クリティカルフロー以外の経路の最適化は行えないことから，最悪の場合は輻輳が発生するため，負荷分散を考慮した経路設計という観点では必ずしも信頼できないという課題がある．

第3章　輻輳制御のためのダイクストラ法を用いた深層学習による経路設計法

3.1　機械学習を用いた経路設計法の課題

従来の機械学習を用いた経路設計法では，負荷分散を考慮した経路設計を行えないという課題がある．文献 [11] [12] では，先ず入力されたすべてトラヒックフローに対して，集中管理型の送信元から宛先間の経路を設計させるようなモデルを検討している．集中管理型の経路設計では，すべての拠点間ノードに対する経路を一度に出力させる必要がある．しかしながら出力層の規模が大きいため十分な学習が行えるようなモデルを生成できなかった．そこで次に，入力されたある一つのトラヒックフローに対して送信元から宛先間の経路を設計させるように変更したものの，エラー率が 70%より改善されなかった．このように，集中管理型の機械学習を用いた経路設計には，経路情報を十分に学習することができないという課題がある．文献 [13] では，回帰型深層学習である seq2seq モデルに BeamSearch と AttentionAlgorithm を用いることで，ある一つのトラヒックフローに対する送信元から宛先間の経路を設計できる方式が検討されている．この方式ではホップ数をコストとした最短経路と，それらを結合させることで生成した経路群を用いて学習を行わせている．これらの経路は最短経路であることから，ネットワーク状況に応じた経路の変更をすることができない．そのため，負荷分散を考慮した経路設計が行えないという課題がある．本研究では，これらの課題を解決した新しい機械学習を用いた経路設計法を検討するのが目標である．

3.2　輻輳制御のための機械学習を用いた経路設計法の検討

図 3.1 は，本研究で想定する機械学習を用いた経路設計におけるシナリオを表している．ネットワークのコントローラは，常にネットワークの状態を監視しており，何らかの異常検知により学習モデルによる経路の再設計を行う．図では，輻輳の発生を検知した場合の動作例である．コントローラは輻輳を検知した際に，その時点での拠点間トラヒックの集合であるデマンドを学習モデルに入力する．学習モデルは，入力されたデマンドに対して経路を直ちに再設計することで，輻輳制御されたリアルタイムな経路設計を実現する．

このような機械学習を用いた経路設計法を実現するために，三つの目標を立てる．一つ目の目標は，高い成功率で経路設計を行える学習モデルを設計することである．本研究では，機械学習を用いて経路設計を行う方式を検討する．しかしながら，機械学習を用いた場合，確率的に出力が定まるため，必ず適切な解が得られるとは限らない．そのため，機械学習を用いて適切な経路を高い成功率で設計する必要がある．二つ目の目標は，残余帯域を考慮した負荷分散された経路を設計することである．本研究では，負荷分散された経路を設計するために，最大負荷リンクにおける帯域使用率を低減させる事が必要である．そのため，教師データとして負荷分散を考慮した経路設計法を用いて生成し，学習モデルが経路設計法の特徴を学習できるか検証する必要がある．三つ目の目標は，経路計算にかかる時間を削減することである．本研究では，ネットワークを監視しており輻輳の発生を検知した場合に直ちに経路を再計算することを想定しているため，ミリ秒オーダでの経路計算が必要である．そのため，機械学習を用いて経路設計を行う場合にどの程度の計算時間を要するのか検証する必要がある．しかしながら，既存の機械学習を用いた経路設計では，これらの目標を達成することができないため，まったく新しい方式の検討を行うのが本研究の目的である．

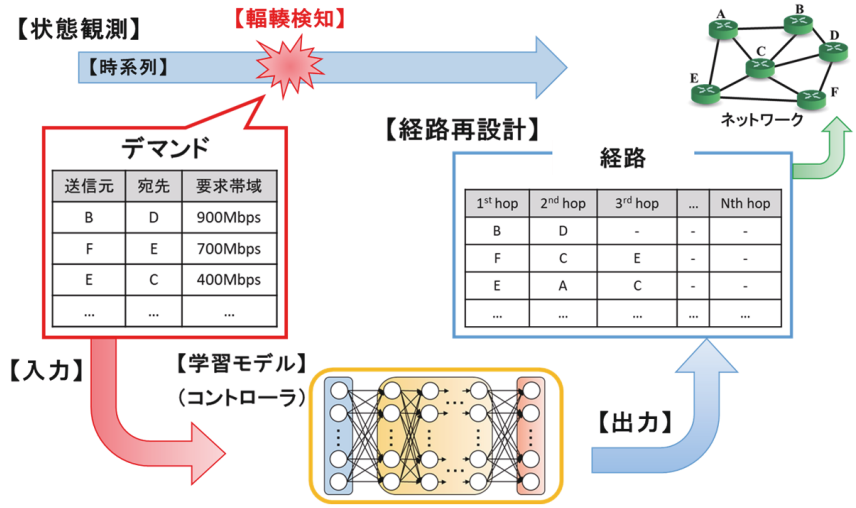


図 3.1 輻輳制御のための機械学習を用いた経路設計の想定するシナリオ

3.3 深層学習モデルと教師データの設計

本研究における課題は以下の二点である．一つ目は，新しい深層学習モデルの設計を行うことである．具体的には，入力層，出力層，中間層の構成の設計である．特に，入力層は複数の拠点間トラヒックを受け付けることができるようにする必要がある．また，出力層は入力された複数のトラヒックに対して，それぞれの送信元から宛先間の経路を出力することができるようにする必要がある．加えて，中間層は最良の学習結果を得られるように適切な層の数やユニット数，他の層との結合などを決定する必要がある．二つ目は，新しい教師データの設計を行うことである．具体的には，訓練データと正解データの構成の設計である．特に訓練データは，学習モデルの入力であり，どのような情報を用いるのが適切であるか検討する必要がある．また，正解データは出力させたい経路情報であり，どのような情報として表現するか検討する必要がある．加えて，正解となる経路をどのような経路設計法により生成するかについても検討する必要がある．

機械学習モデルおよび教師データの設計と学習，評価の工程は以下の通りである．先ず，入力層と訓練データを設計し，次に出力層と正解データの設計，最後に中間層および学習モデル全体の構成を設計する．この設計を基に機械学習モデルと学習データを生成し事前学習により，ハイパーパラメータ等のチューニングを行う．チューニング済みの学習モデルを用いて，性能評価を行う．

3.3.1 深層学習モデルの入力層と訓練データの設計

訓練データには，複数の拠点間トラヒックからなるデマンド集合を用いる．このとき，各拠点間トラヒックは，送信元ノード番号，宛先ノード番号，中継ノード候補，要求帯域幅の情報を持つベクトルで表される．ここで，送信元ノード番号と宛先ノード番号は送受信間経路を設計させるために必要な情報であり，要求帯域幅は負荷分散を考慮させるために必要な情報である．また，中継ノード候補となるその他のノード情報を加えることで，拠点間トラヒック毎の情報の違いを明確にし，学習難易度の低減を図る．これらの設計を基に入力層を設計する．

図 3.2 は，入力層の設計をまとめたものである．はじめに，入力層の構成を検討する前に，入力信号中に何個の拠点間トラヒックが含まれるかを定める必要がある．仮に，含まれる拠点間トラヒックの最大数が定まっていない場合，入力層は無限個の拠点間トラヒックを処理しなければならず，学習しなければならない情報量の際限が無くなり，学習難易度の大幅な上昇を防ぐことができない．これは，計算機資源は有限であるという観点から見ても，適切な設計ではない．そこで，本研究では対象とするネットワーク内のすべてのノード間で，拠点間トラヒックが一つずつ発生するとする．よって，デマンド集合中に含まれる拠点間トラヒックの数は，ネットワーク内のノード数を N としたとき，${}_N\mathrm{P}_2$ 個であ

る．ただし，各拠点間トラヒックは要求帯域幅の値を基準に降順にソートする．これは，帯域の大きい順に経路を割り当てることで，より効果的な負荷分散を達成することができるためである．

このとき，一つの拠点間トラヒックは $N+1$ 個の要素を持つベクトルで表され，ベクトルの i $(i \in N)$ 番目の要素が -1 であるときノード i は送信元ノードを表し，1 であるときノード i は宛先ノードを表す．また，それ以外の 0 である要素は，中継ノードの候補であることを表している．加えて，$N+1$ 番目の要素は，拠点間トラヒックの要求帯域幅である．以上より，入力層は ${}_N\mathrm{P}_2 \times (N+1)$ 個のユニットで構成される．

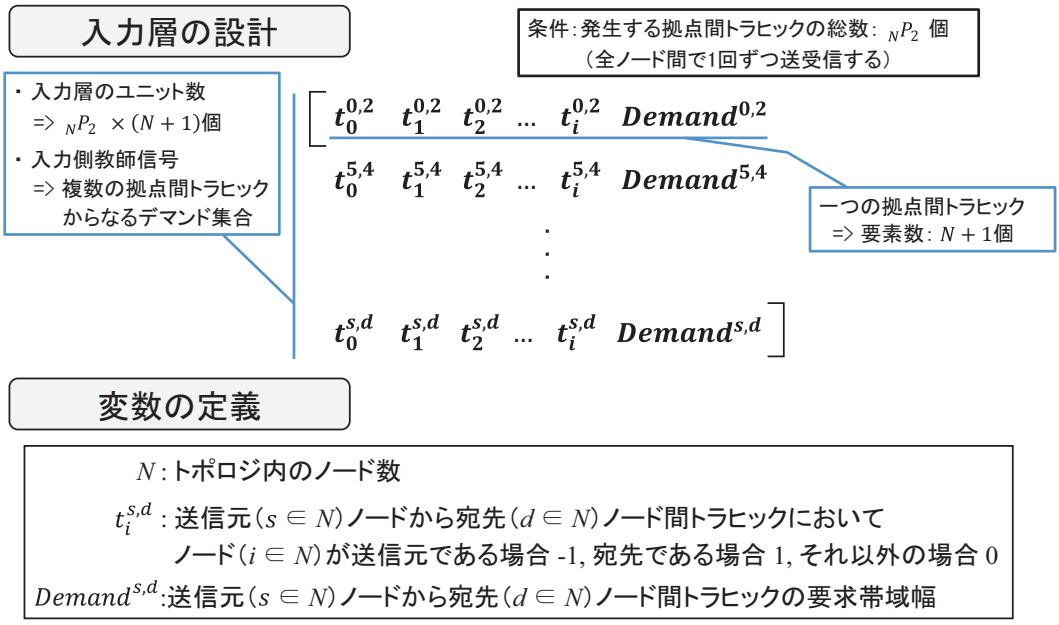


図 3.2 入力層の設計と定義

3.3.2　深層学習モデルにおける出力層の設計

正解データには，各拠点間トラヒックに対する送信元から宛先間の経路情報を用いる．このとき，各経路情報は送信元から宛先ノードを含むすべての中継ノードを 1hop 毎に分割し，送信元，宛先，中継ノード番号または経路未使用フラグのいずれか一つの情報を持つ one-hot ベクトルで表される．one-hot ベクトルとは，一つの要素のみが 1 で，他のすべての要素が 0 であるベクトルであり，教師あり学習における分類問題の出力に用いられるデータ形式である．なお，送受信間経路情報の計算には，目的や要件に合わせて，ダイクストラ法や線形計画法などの経路設計法を用いて作成する．

図 3.3 は，出力層の設計についてまとめたものである．提案する深層学習モデルの出力層は，デマンド集合に含まれる各拠点間トラヒックに対して，ホップ毎の経路情報を出力

する必要がある．ここで，分類問題として学習を行わせるために，ホップ毎の経路情報には，送信元ノード番号，中継ノード番号，宛先ノード番号，ノード未使用を表すフラグのいずれか一つのみを示すのが望ましい．そこで，$N+1$ 個の要素を持つ one-hot ベクトルによって表現する．また，送信元から宛先ノード間において最長である経路長は，トポロジ内のすべてのノードを 1 度ずつ通過するときであり，この数を V とすると，V はノード数 N と等しい．よって，一つの拠点間トラヒックにおける経路全体を表すためには，N 個の one-hot ベクトルが必要である．ただし，送信元から宛先ノード間の経路が h ホップ $(h < V)$ の場合，$h+1$ 番目以降は経路情報として使用しないことを示す必要があるため，ノード未使用フラグにより表す．具体的には，one-hot ベクトルの j $(j \in N)$ 番目の要素が 1 であるとき，ノード j は送信元ノード，宛先ノード，または中継ノードであることを示し，それ以外のすべての要素を 0 とする．また，one-hot ベクトルの $N+1$ 番目の要素が 1 であるとき，この要素はノード未使用フラグを示し，それ以外のすべての要素を 0 とする．以上より，出力層は全部で ${}_N\mathrm{P}_2 \times N$ 個であり，各出力層は $N+1$ 個のユニットで構成される．

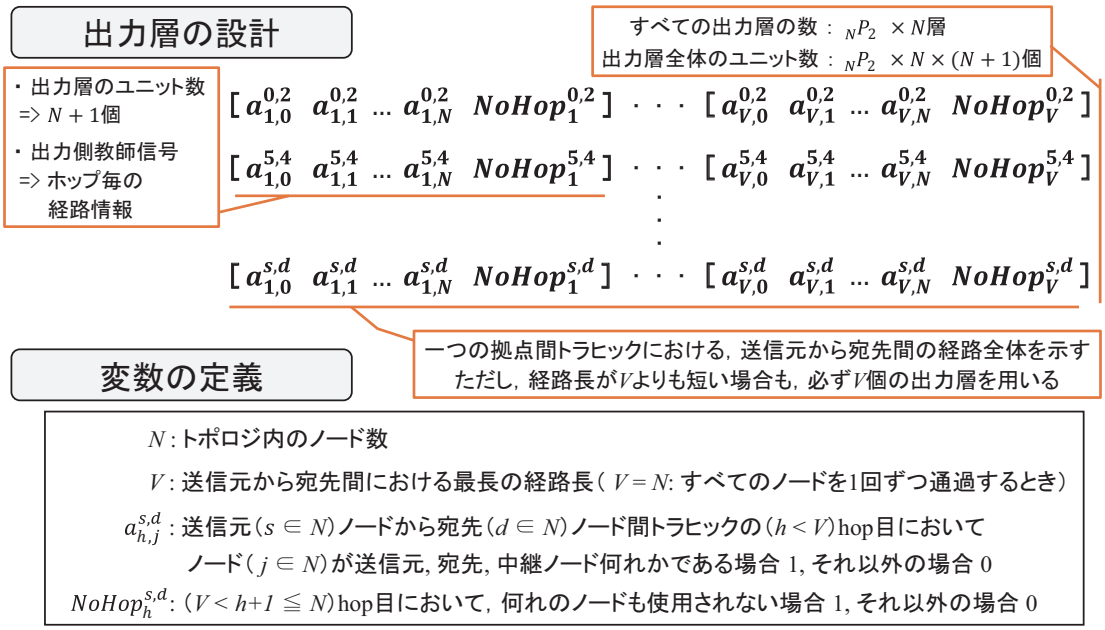


図 3.3 出力層の設計と定義

3.3.3 深層学習モデルにおける中間層の設計

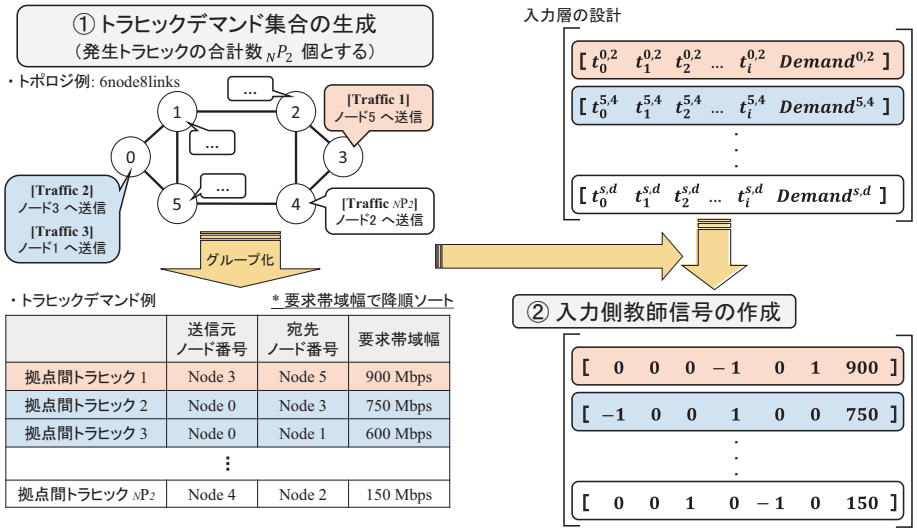
中間層とは，入力層と出力層を繋ぐ学習の根幹を担っており，ニューラルネットワークにおいて最も重要な構成要素である．一般的に，深層ニューラルネットワークにおける中間層とは 2 層以上の層からなる状態を指す．その一方で，中間層の適切な層数は規定され

ていない．そこで，本研究では関連研究において中間層が4層であるとき，最も学習精度が高い結果を得られたと示唆されていることから，この4層を基準に3層から6層程度の中間層を適宜使用することとした．なお，実際の構成で使用される中間層の数は，学習実験を繰り返し行い，最良な結果となったものを適用することとする．

3.4 深層学習モデルの学習

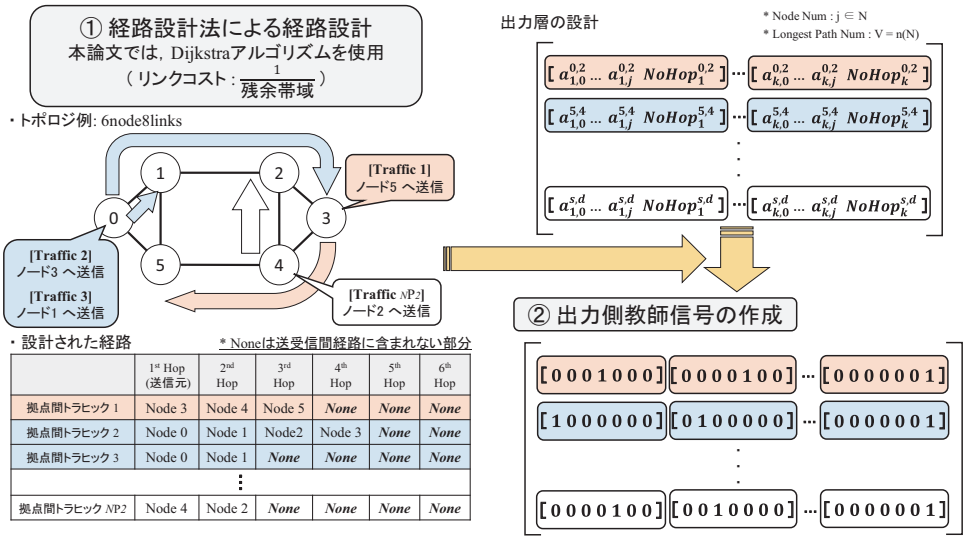
3.4.1 教師データの作成

図 3.4 は訓練データ，図 3.5 は正解データの作成例を表したものである．先ず，訓練データは，対象とするネットワークにおいて，${}_N\mathrm{P}_2$ 個の拠点間トラヒックに対して要求帯域順に降順ソートしたものをある一つのデマンド集合とし，ベクトルとして作成する．このとき，各拠点間トラヒックの要求帯域幅は，設定した複数の帯域幅より完全ランダムに割り当てたものとする．次に，正解データは，各デマンド集合に対して経路設計アルゴリズムを用いて最適経路を計算し，one-hot ベクトルとして作成する．ただし，計算された経路を用いて，すべての拠点間トラヒックを送信したとき，リンクのどこか一か所でも輻輳が発生している場合は教師データとして用いない．なお，本研究における教師データの作成に用いる経路計算アルゴリズムは，リンクコストを残余帯域の逆数としたダイクストラ法である．これは，IP ベースのトラヒックエンジニアリングにおいてリンクコストを計算するための最も基本的なアルゴリズムであることを理由に採用した．



	送信元ノード番号	宛先ノード番号	要求帯域幅
拠点間トラヒック 1	Node 3	Node 5	900 Mbps
拠点間トラヒック 2	Node 0	Node 3	750 Mbps
拠点間トラヒック 3	Node 0	Node 1	600 Mbps
⋮			
拠点間トラヒック $_NP_2$	Node 4	Node 2	150 Mbps

図 3.4 訓練データの生成例



	1st Hop (送信元)	2nd Hop	3rd Hop	4th Hop	5th Hop	6th Hop
拠点間トラヒック 1	Node 3	Node 4	Node 5	*None*	*None*	*None*
拠点間トラヒック 2	Node 0	Node 1	Node2	Node 3	*None*	*None*
拠点間トラヒック 3	Node 0	Node 1	*None*	*None*	*None*	*None*
⋮						
拠点間トラヒック $_NP_2$	Node 4	Node 2	*None*	*None*	*None*	*None*

図 3.5 正解データの生成例

3.5 深層学習モデルの学習

最後に学習過程について説明する．機械学習を用いる際の重要な要素として，様々なハイパーパラメータの設定や教師データのデータセット数などが挙げられる．本研究では，実験を通して最適化するパラメータと，予め見当をつけて設定するパラメータの 2 種類に分けて考える．先ず，実験による最適化を行うパラメータは，学習モデルの中間層の層数，教師データおよびテストデータのデータセット数である．本研究において，1 つの訓練データは ${}_N\mathrm{P}_2$ 個の拠点間トラヒック情報で構成され，正解データはダイクストラ法によって計算された当該トラヒックの経路情報で構成される．また，予め見当をつけて設定するパラメータは，最大学習回数を表す Epoch 数，1 回の学習で何個のデータセットを用いるかを表すバッチサイズ，中間層の活性化関数として Relu 関数，出力層の活性化関数として SoftMax 関数，学習精度の指標となる損失関数は多クラス交差エントロピー，最適化アルゴリズムには Adamax 関数を用いる．なお，学習の繰り返し回数は，十分に大きな値を設定し，一定回数の学習更新が閾値よりも行われなくなった場合に学習を早期終了する Early Stopping を導入している．

3.6　深層学習モデルを用いた経路設計法の性能評価実験

3.6.1　実験の目的

本章では，深層学習モデルを用いて集中管理型の経路設計が行えるかどうかについて検証するのが目的である．そのため，複数の小規模トポロジを用いて，学習を行い経路設計成功率や，出力された経路が教師と比較してどの程度負荷分散を行えるか確認する．加えて，経路設計にかかる時間を計測し，深層学習モデルを用いた経路設計にかかる時間についても調査を行う．

3.6.2　実験に用いる深層学習モデルの構成

生成したディープニューラルネットワークモデルを図 3.6 に示す．

入力層は 1 層 ${}_N\mathrm{P}_2 \times (N+1)$ ユニットの単層レイヤとする．

中間層は 1 層 ${}_N\mathrm{P}_2 \times (N+1)$ ユニットの全結合レイヤを 3 層または 4 層重ねたものとし，活性化関数は全て Relu 関数を用いる．

出力層は 1 層 $N+1$ ユニットを並列 ${}_N\mathrm{P}_2 \times N$ 層とし，全ての出力層の活性化関数は Softmax 関数とする．

また，最適化アルゴリズムは Adamax，損失関数は CategoricalCrossentropy，バッチサイズは 128，エポック数は 1000 回とする．

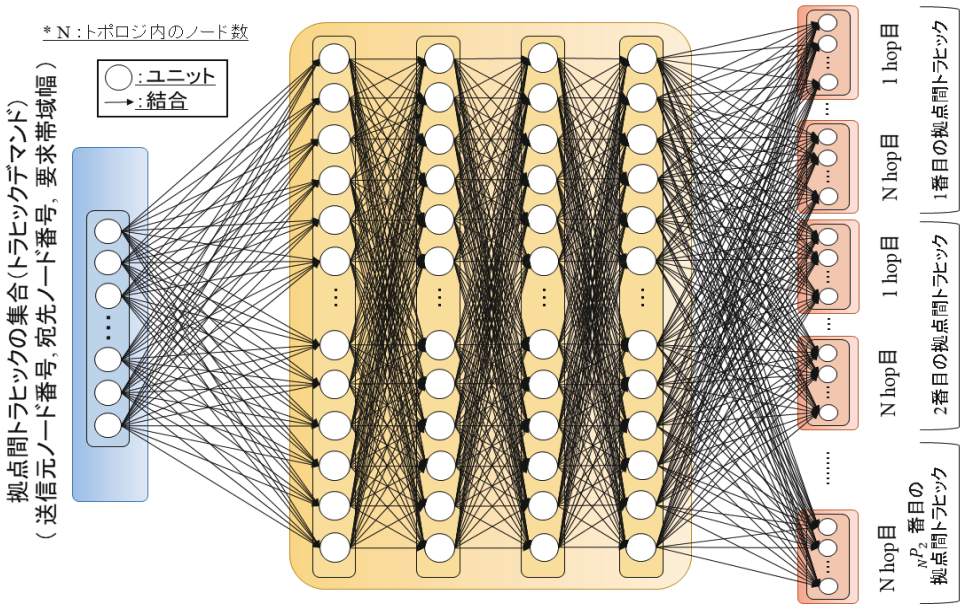


図 3.6 設計したディープニューラルネットワークモデル

3.6.3 実験条件

本章では，深層学習モデルの性能を確認するのが目的であるため，深層学習モデルの評価のために小規模ネットワークである，複数パターンの 6 ノードトポロジ (図 3.29) [21] [22] [23] を用いる．ただし，評価実験を行う前に，提案方式の設計に基づいて，10000 セットの教師データと 5 セットのテストデータを生成し最適化を行った．なお，要求帯域幅の設定については，(株）KDDI 法人・ビジネス向け「国内イーサネット専用サービス」における，帯域選択型の専用線サービスを参考にした [24]．本章では，この最適化されたモデルを用いて，経路設計成功率，最大負荷リンクにおける使用帯域幅，および経路計算時間の観点から評価を行った．

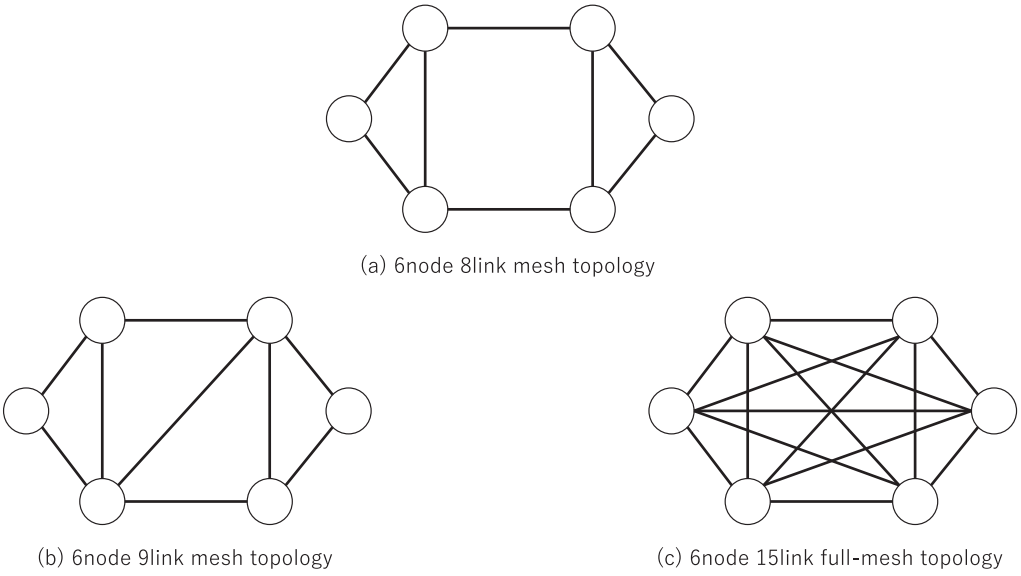


図 3.7 実験用トポロジ

3.6.4　実験用トポロジを用いた性能評価

3.6.4.1　経路設計成功率

提案方式によって設計された経路の経路設計成功率を図 3.8 に示す．実験により，生成した 5 つのテストデータすべてにおいて経路設計成功率が 100%であることが確認された．

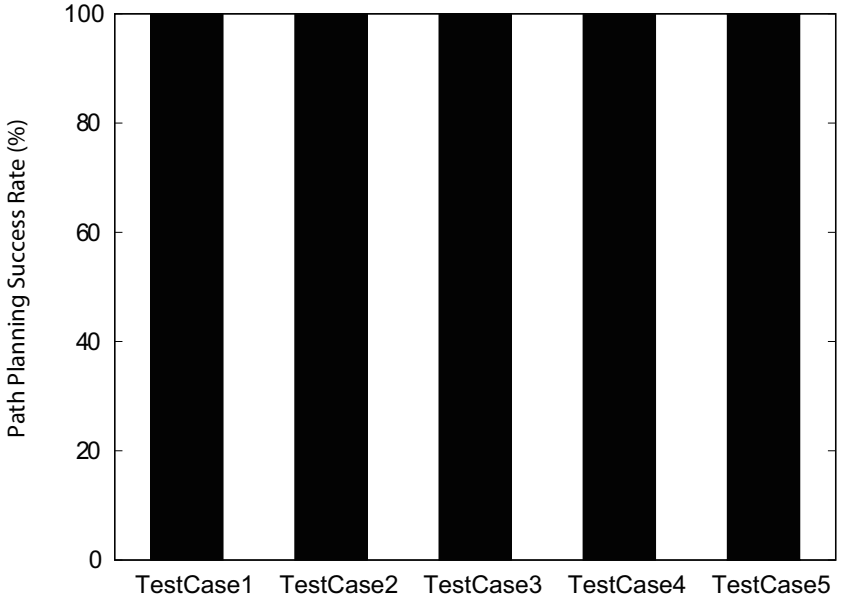


図 3.8 経路設計成功率

3.6.4.2 最大負荷リンクにおける使用帯域幅

提案方式によって設計された経路の最大負荷リンクにおける使用帯域幅について，図 3.9, 図 3.10, 図 3.11 に示す．実験により，すべてのトポロジにおいて，最大負荷リンクのトラヒック量が帯域幅を下回っており，輻輳が発生していない事を確認した．また，ダイクストラ法によって設計された経路と比較すると，トポロジ（a）と（c）でほぼ同等，トポロジ（b）ではやや劣るものの十分に負荷分散された経路であることが確認された．

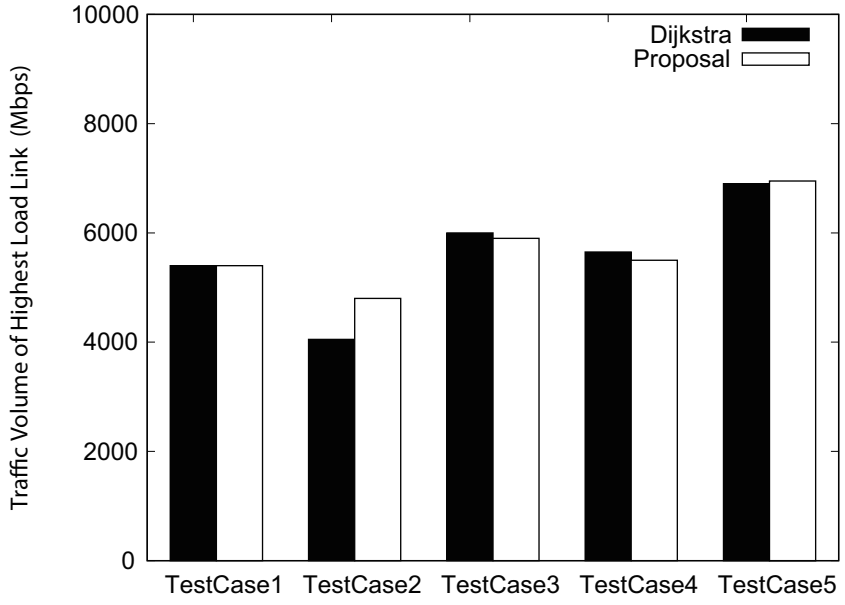


図 3.9 トポロジ (a) における最大負荷リンクにおける使用帯域幅

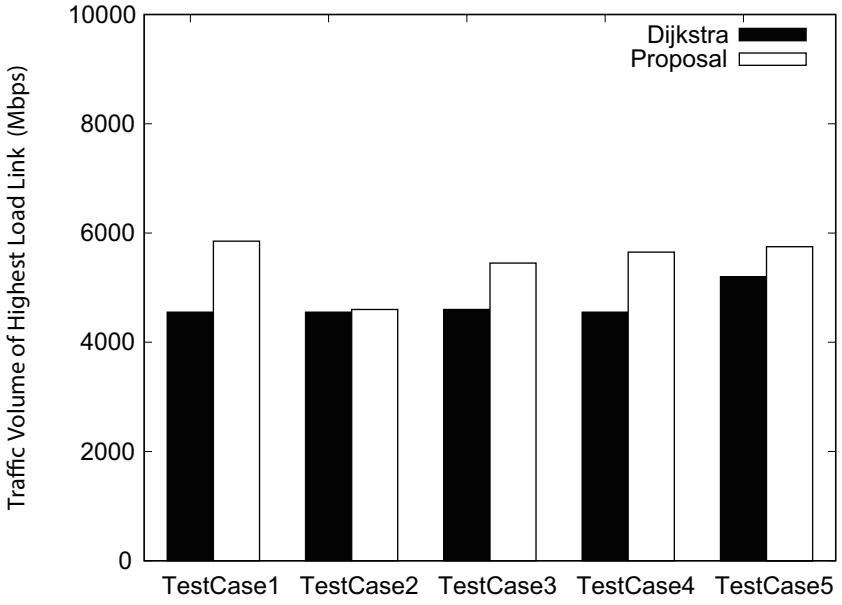


図 3.10 トポロジ (b) における最大負荷リンクにおける使用帯域幅

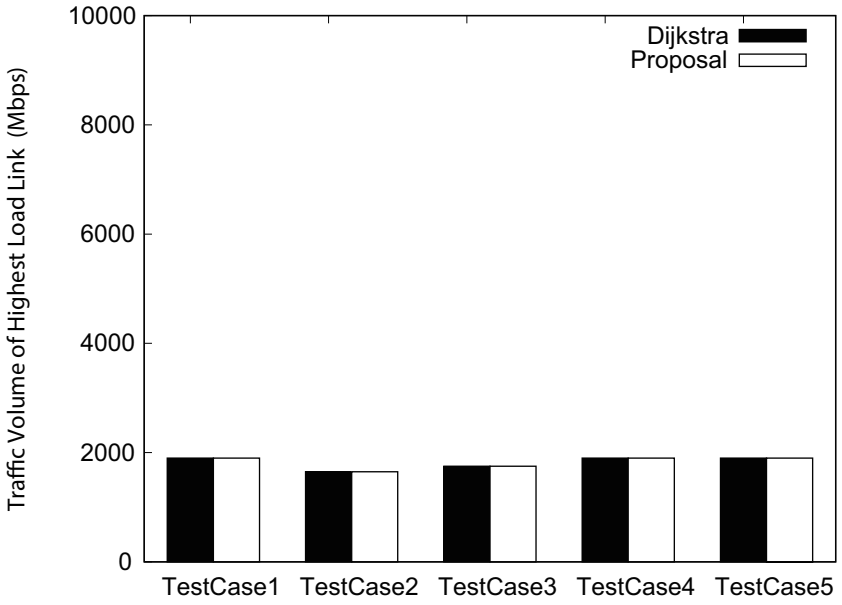


図 3.11 トポロジ (c) における最大負荷リンクにおける使用帯域幅

3.6.4.3 経路計算時間

図 3.12 は，提案方式とダイクストラ法によって，入力トラヒック 10,000 セット分の経路を設計するまでにかかる時間の平均である．ただし，図 3.12 はダイクストラ法を基準に正規化を行っている．実験により，提案方式を用いた経路設計にかかる時間は，ダイクストラ法と比較して約 40%から 61%短縮されていることが確認された．ここで，アルゴリズムに従って順次処理を行うダイクストラ法では，リンク数の増加と共に経路設計時間が長くなる傾向が見られる．その一方で，提案方式における経路設計時間は，リンク数に依存せず，学習モデルの規模に依存して長くなる傾向が見られる．よって，トポロジ (a),(b),(c) の様に対象とするトポロジの規模が変わらなければ，経路設計にかかる時間は変わらないため，ダイクストラ法よりも高速に処理できると考えられる．

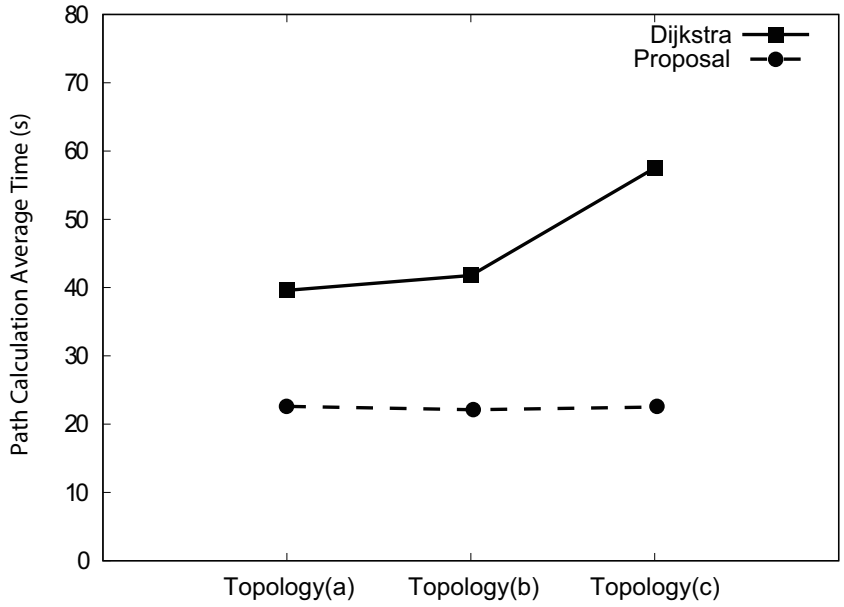


図 3.12 実験用トポロジにおける平均計算時間 [s]

3.6.5　深層学習モデルを用いた経路設計法の考察

実験結果より，実験用の簡易な小規模トポロジにおいて，高い経路設計成功率と教師データと同等の負荷分散，ダイクストラ法よりも短い時間で経路設計ができることが確認された．先行研究と異なり，入力にすべての拠点間トラヒックの情報と要求帯域幅を用いて，出力に送受信間経路を 1hop ずつ one-hot ベクトルで表現することで，集中管理型の経路設計を機械学習によって実現できることが確認できた．今後は，より規模の大きなトポロジを用いて提案方式の有用性を検証していくことが必要であり，11 ノード 26 リンクからなる COST239 トポロジを用いて，性能評価実験を行う．

3.6.6　深層学習モデルを用いた経路設計法の課題

実験により，実験用の簡易な小規模トポロジにおいては，高い経路設計成功率で，負荷分散された経路設計が，ダイクストラ法よりも短い計算時間で行えることを確認した．しかしながら，適用するネットワークトポロジを 11 ノード 26 リンクからなる COST239 トポロジに変更したところ，学習が行えなかった．そこで，中規模トポロジでも提案方式による経路設計が行うために，機械学習モデルの改良を行う．具体的には，回帰型構造を取り入れることでこの課題の解決を図る．

3.7 回帰型深層学習モデルを用いた経路設計法の性能評価実験

3.7.1 本章の目的

深層学習モデルは，適用するトポロジの規模が大きくなった場合，学習が収束せず正しい経路が設計できなくなる課題がある．より大規模なトポロジに対する経路を学習するためには，学習モデルを改良する必要があると考えた．そこで，トラヒックの発生に着目する．本研究における入力は，複数の拠点間トラヒックの集合である．これらの経路は，要求帯域幅の大きい順に経路が決定されることから，トラヒック毎の経路には時系列性を見出すことができると考えた．そこで，字句解析をはじめとする時系列性のあるデータを学習させる場合によく用いられる，回帰型ニューラルネットワークを参考に深層学習モデルの改良を行うのが目的である．

3.7.2 深層学習モデルの改良

回帰型ニューラルネットワークとは，比較的小さな規模のニューラルネットワークモデルを連結し，入力に一つ前の出力を加えることで，時系列性のあるデータを学習，予測し易くした学習モデルである．そこで，本章では回帰型ニューラルネットワークを参考に，3.5 章で用いた深層学習モデルを分割および連結を行い，回帰型構造とすることで学習精度の向上を図る．

具体的には，訓練データはある一つの拠点間トラヒックとし，この拠点間トラヒックの送受信間経路を設計させるための，小さな深層ニューラルネットワークの単位に分割する．このとき，訓練データと正解データは以下の様に変形される．

[訓練データ]

$$\begin{bmatrix} t_1^{s,d} & t_2^{s,d} & t_3^{s,d} & \dots & t_i^{s,d} & Demand^{s,d} \end{bmatrix}$$

[定数の定義]

N : ノード集合

$t_i^{s,d}$: ルータ $s \in N$ とルータ $d \in N$ 間のトラヒックにおいて，ルータ $i \in N$ が送信元ノードならば-1，宛先ノードならば 1，それ以外の場合 0

$Demand^{s,d}$: ルータ $s \in N$ とルータ $d \in N$ 間のトラヒック要求帯域

[正解データ]

$$\begin{bmatrix} \begin{bmatrix} a_{0,1}^{s,d} & a_{0,2}^{s,d} & a_{0,3}^{s,d} & \dots & a_{0,j}^{s,d} & NoHop_0^{s,d} \end{bmatrix} \\ \begin{bmatrix} a_{1,1}^{s,d} & a_{1,2}^{s,d} & a_{1,3}^{s,d} & \dots & a_{1,j}^{s,d} & NoHop_1^{s,d} \end{bmatrix} \\ \vdots \quad \ddots \quad \vdots \\ \begin{bmatrix} a_{h,1}^{s,d} & a_{h,2}^{s,d} & a_{h,3}^{s,d} & \dots & a_{h,j}^{s,d} & NoHop_h^{s,d} \end{bmatrix} \end{bmatrix}$$

[定数の定義]

$a_{h,j}^{s,d}$: ルータ $s \in N$ からルータ $d \in N$ 間のトラヒックにおいて，ルータ s からの $h \in N$ ホップ目のルータ $j \in N$ で送信元ノード，または中継ノード，または宛先ノードならば 1，そうでなければ 0

$NoHop_h^{s,d}$: ルータ $s \in N$ とルータ $d \in N$ 間のトラヒックにおいて，ルータ s からの $h \in N$ ホップ目に送信元ノード，または中継ノード，または宛先ノードとなるものが存在しない場合 1，そうでなければ 0

3.7.3　実験に用いる深層学習モデルの構成

設計した回帰型深層学習モデルを図 3.13，図 3.14 に示す．

図 3.13 がある一つの拠点間トラヒックを学習する小さな深層ニューラルネットワークの構成である．従来との違いは出力層の数である．従来と比較して入力層や中間層の構成は変わらないのに比べて，出力層が少なくなる分学習難易度が低減されることが期待される．また，図 3.14 は，回帰型ニューラルネットワークの全体構成である．このように，デマンドに含まれる拠点間トラヒックの数と同数の小さな深層ニューラルネットワークが回帰型構造で連結している．なお，図では省略しているものの，小さな深層ニューラルネットワーク間に全結合レイヤーからなる処理層を設け，中間層同士を連結させる．

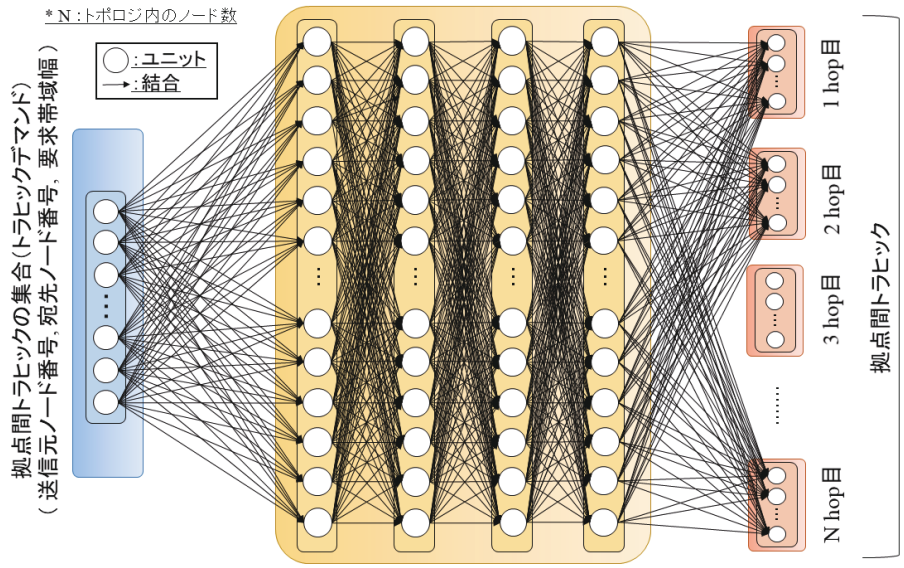


図 3.13 小規模深層ニューラルネットワークの構成

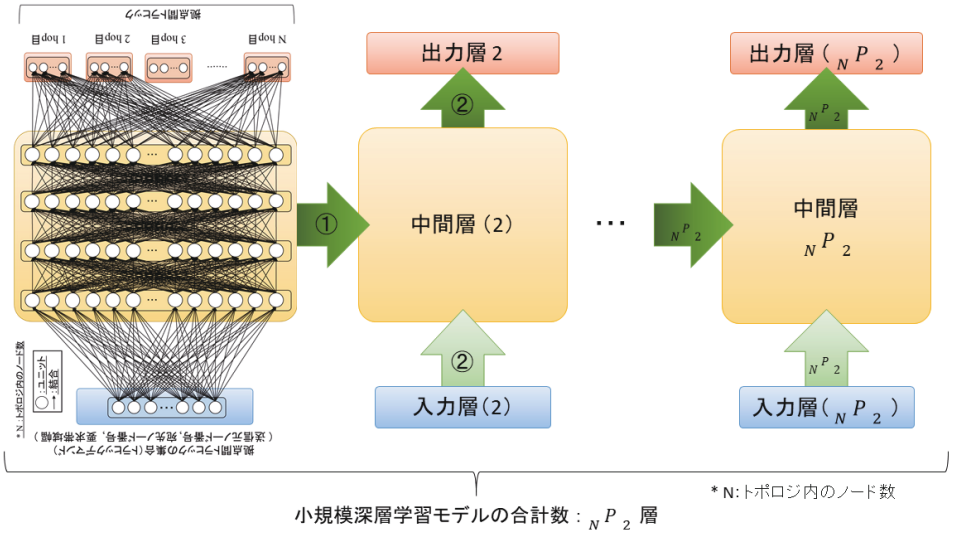


図 3.14 回帰型深層ニューラルネットワークの構成

3.7.4　実験条件

本章では，回帰型深層学習モデルを用いて中規模トポロジにおける提案方式の性能を確認するのが目的である．よって，評価用トポロジとして中規模ネットワークである 11 ノード 26 リンクの cost239(図 3.15) [25] を用いる．ただし，評価実験を行う前に，提案方式の設計に基づいて，10000 セットの教師データと 5 セットのテストデータを生成し最適化を行った．なお，要求帯域幅の設定については，(株）KDDI 法人・ビジネス向け「国内イーサネット専用サービス」における，帯域選択型の専用線サービスを参考にした [24]．本章では，この最適化されたモデルを用いて，経路設計成功率，最大負荷リンクにおける使用帯域幅，および経路計算時間の観点から評価を行った．

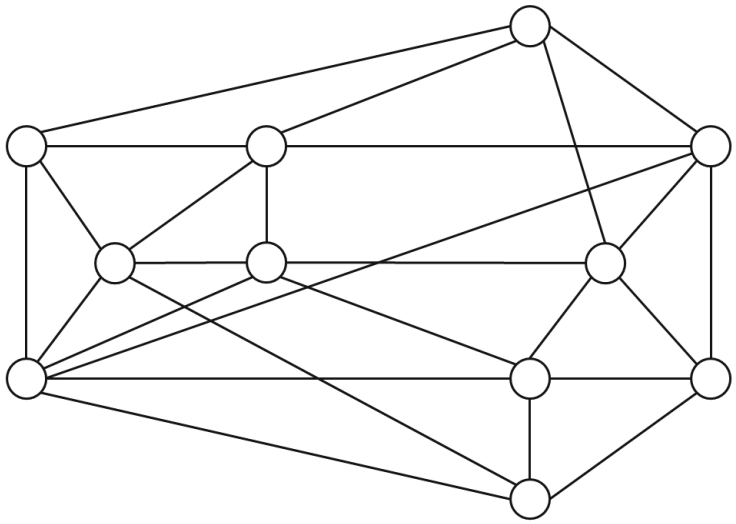


図 3.15 Cost239 トポロジ

3.7.5 評価用トポロジを用いた性能評価

3.7.5.1 経路設計成功率

提案方式によって設計された経路の経路設計成功率を図 3.16 に示す．実験により，生成した 5 つのテストデータすべてにおいて経路設計成功率が 100%であることが確認された．

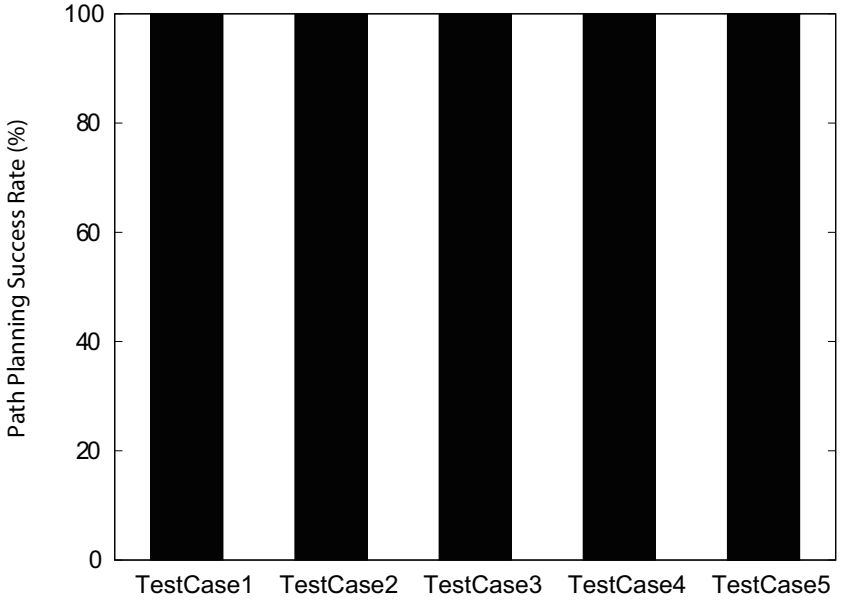


図 3.16 経路設計成功率

3.7.5.2　最大負荷リンクにおける使用帯域幅

提案方式によって設計された経路の最大負荷リンクにおける使用帯域幅について，図 3.17 に示す．実験により，COST239 トポロジにおいて，最大負荷リンクのトラヒック量が帯域幅を下回っており，輻輳が発生していない事を確認した．また，ダイクストラ法によって設計された経路と比較すると，やや劣るものの十分に負荷分散された経路であることが確認された．

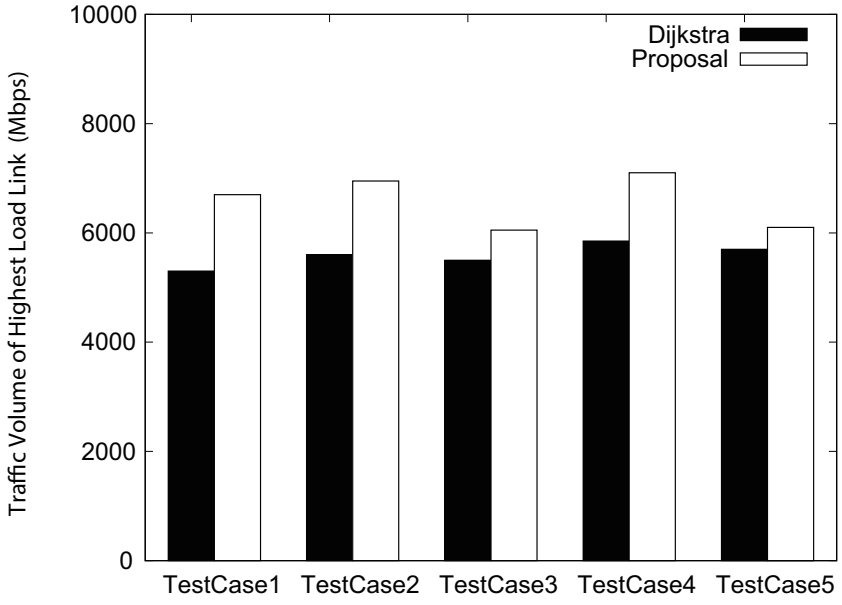


図 3.17 COST239 における最大負荷リンクにおける使用帯域幅

3.7.5.3 経路計算時間

図 3.18 は，提案方式とダイクストラ法によって，入力トラヒック 10,000 セット分の経路を設計するまでにかかる時間の平均である．ただし，図 3.18 はダイクストラ法を基準に正規化を行っている．実験により，提案方式を用いた経路設計にかかる時間は，ダイクストラ法と比較して約 23%短縮されていることが確認された．

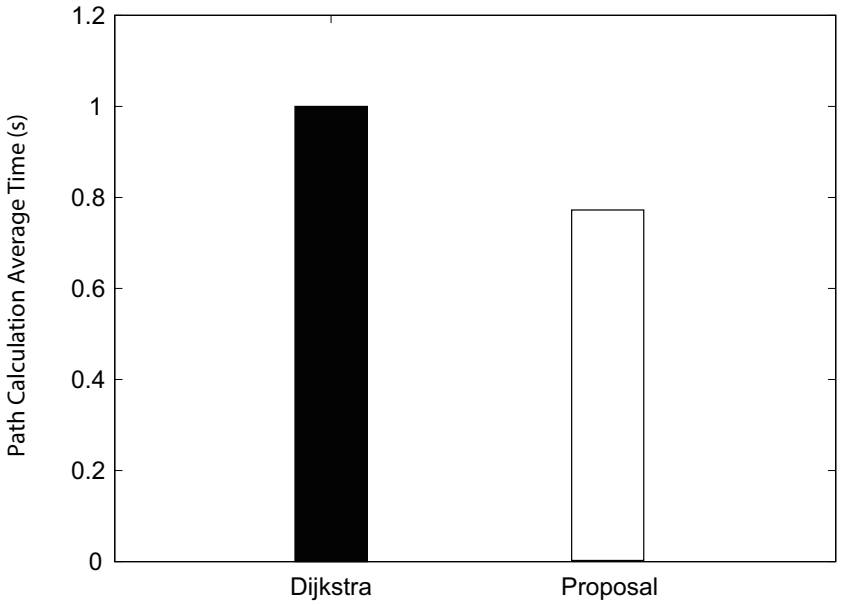


図 3.18 評価用トポロジにおける平均計算時間 [s]

3.7.6　回帰型深層学習モデルの考察

実験結果より，回帰型深層ニューラルネットワークを用いることで，これまで経路を学習し設計することができなかった中規模トポロジにおいても経路設計が行えることが示された．しかし，計算時間の削減率に着目すると，単純な深層ニューラルネットワークと比べて削減効果が低下していることが分かった．これは，回帰型深層ニューラルネットワークを用いているため，一度にすべての経路を設計するのではなく，一つの拠点間トラヒックに対する経路を設計し，その情報を加えて更に次の経路を設計する逐次型の経路設計になっているため，ニューラルネットワーク間の入力および出力がボトルネックになっているものと考えられる．また，回帰型深層ニューラルネットワークの場合，単純な深層ニューラルネットワークと比較して，学習モデルの規模が増大し易いため，スケーラビリティが劣ると考えられる．

3.7.7　回帰型深層学習モデルの課題

回帰型深層ニューラルネットワークを用いることで，提案方式の適用可能なトポロジ規模の拡大を達成できる．しかしながら，課題が多く残されている．一つ目はスケーラビリティの点である．回帰型深層ニューラルネットワークの場合，入力層が拠点間トラヒックの数だけ重複してしまうことや，ニューラルネットワーク間の結合部が必要なため，学習モデルの規模が増大してしまう．二つ目は計算時間の削減効果の点である．回帰型深層ニューラルネットワークを用いても，ダイクストラ法よりも短い時間で経路計算が行えることが確認された．その一方で，単純な深層ニューラルネットワークと比較すると，回帰型構造を用いたことで計算時間の削減率は低下してしまった．これらの事項は，今後のトポロジ規模の拡大を想定した場合，計算資源や計算時間の観点から懸念事項となり得る．

そこで，可能な限り単純な構造からなる深層ニューラルネットワークを用いて，中規模トポロジにおいても経路設計ができるような方式の検討が必要であると考える．

3.8 不均一ネットワークにおけるアンサンブル学習型深層学習モデルの性能評価実験

3.8.1 本章の目的

これまでの提案方式は，ネットワーク内のリンクがすべて同一の帯域幅を持つことを想定して設計されている．しかしながら，現実的なネットワークでは，大都市間の帯域幅と地方間の帯域幅では異なる帯域幅であることが一般的である．そこで，深層ニューラルネットワークを用いて不均一な帯域幅を持つネットワークにおいて経路設計が行えるか検証する．

3.8.2 不均一リンクを学習した深層学習モデルを用いた経路設計法の課題

深層学習を用いた経路設計法は，均一な帯域幅を持つトポロジ（以下，均一リンクトポロジ）においては，100%の精度で輻輳のない経路を設計することができるが，基幹ネットワークの様に実際に運用されているネットワークは均一な帯域幅を持つとは限らない．そのため，深層学習を用いて不均一な帯域幅を持つネットワークについて学習し，正しい経路を設計できるか検証する必要がある．

そこで，先ず不均一な帯域幅を持つネットワークとして，帯域幅に傾斜を与えた小規模トポロジ（以下，傾斜リンクトポロジ）を作成し（図 3.19），均一リンクトポロジを学習することができた深層学習モデルと同じ構成を持つ複数のモデルを用いて，学習を行い経路を設計させた．なお，今回の実験で用いる深層学習モデルの構成は，3.5 章で生成した学習モデルのチューニングを用いる．図 3.20 は，その一例である．

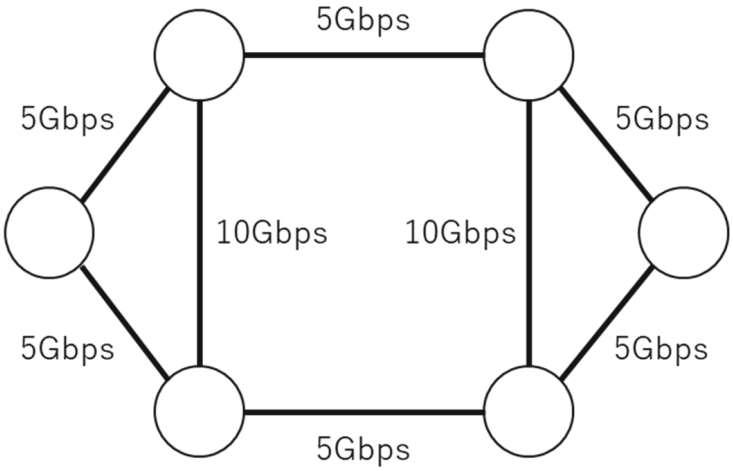


図 3.19 不均一な帯域幅を持つトポロジの例

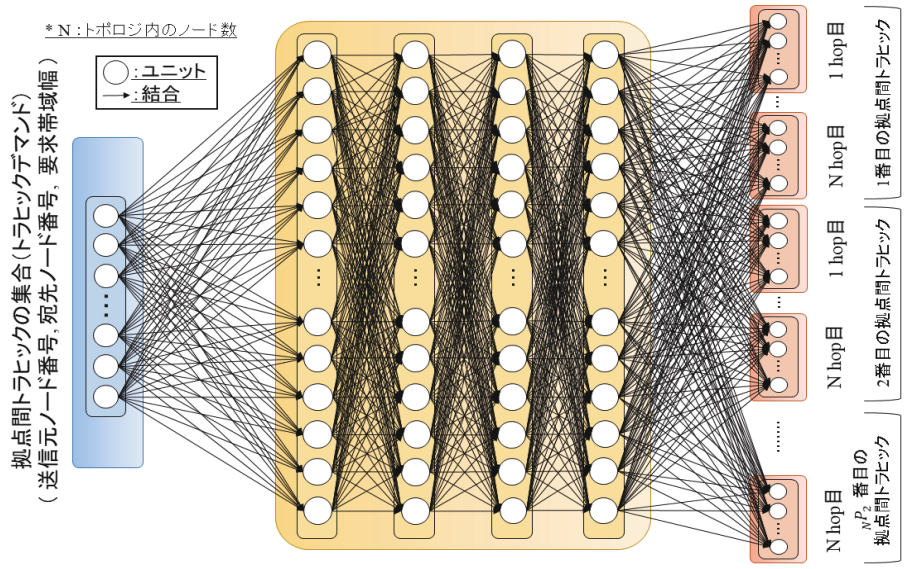


図 3.20 深層学習モデルの構成例

表 3.1 および表 3.2 は，傾斜リンクトポロジを学習した 4 つの学習モデルを用いて，5 つのテストケースに対する経路設計成功率についてまとめたものである．これより，均一リンクトポロジを学習した場合と比較して経路設計成功率が約 5%〜10%弱程度低下している．また，テストケース毎に着目するとデマンド集合全体の経路を設計することができたのは 2 つのみであり，精度の低下が分かった．

その後，深層学習モデルのハイパーパラメータを変更し更なる最適化を図った．しかしながら，結果として 100%の精度で経路を設計することができるような学習モデルを生成することはできなかった．以上より，完全に最適化された深層学習モデルが生成できるまで，繰り返しチューニングを行うのは適切ではないと考える．

表 3.1 傾斜リンクトポロジを学習したモデルにおける経路設計成功率

モデル番号	経路設計成功数	経路設計成功率 (%)
ModelA	143	95.3
ModelB	131	87.3
ModelC	137	91.3
ModelD	133	88.7

表 3.2 傾斜リンクトポロジを学習したモデルにおけるテストケース毎の経路設計成功率

モデル番号	TEST1	TEST2	TEST3	TEST4	TEST5
ModelA	93.3%	96.7%	93.3%	93.3%	100.0%
ModelB	76.7%	86.7%	100.0%	83.3%	90.0%
ModelC	83.3%	96.7%	90.0%	90.0%	96.7%
ModelD	86.7%	90.0%	90.0%	80.0%	96.7%

3.8.3　均一リンクを学習した深層学習モデルを用いた不均一リンクにおける経路設計の検討

ここでネットワークトポロジに着目すると，均一リンクトポロジと傾斜リンクトポロジは，帯域幅のみが異なる同一構成のトポロジである．つまり，均一リンクトポロジを学習した深層学習モデルを用いて，傾斜リンクトポロジにおける経路を設計させることができると考えた．

表 3.3 および表 3.4 は，均一リンクトポロジにおいて 100%の精度で経路を設計することができた 4 つの学習モデルを用いて，前述の 5 つのテストケースに対する経路設計成功率についてまとめたものである．これより，ほとんどの学習モデルで 100%精度で通信可能な経路を設計することができている．しかしながら，ここで設計された経路のうち，全体の半分以上のテストケースにおいて輻輳の発生が確認された．

そのため，均一リンクトポロジを学習したモデルを用いた経路設計は，輻輳制御の観点から，傾斜リンクトポロジに用いるのは適切ではないと考える．

表 3.3 均一リンクトポロジを学習したモデルにおける経路設計成功率

モデル番号	経路設計成功数	経路設計成功率 (%)
ModelA	146	97.3
ModelB	150	100.0
ModelC	150	100.0
ModelD	150	100.0

表 3.4 均一リンクトポロジを学習したモデルにおけるテストケース毎の経路設計成功率

モデル番号	TEST1	TEST2	TEST3	TEST4	TEST5
ModelA	96.7%	96.7%	100.0%	96.7%	96.7%
ModelB	100.0%	100.0%	100.0%	100.0%	100.0%
ModelC	100.0%	100.0%	100.0%	100.0%	100.0%
ModelD	100.0%	100.0%	100.0%	100.0%	100.0%

3.8.3.1　アンサンブル学習の検討

以上より，不均一なリンク帯域幅を持つネットワークのような複雑なトポロジに対して提案方式を用いる場合，高い経路設計成功率の維持が困難であり，輻輳制御された通信を実現できないことが明らかとなった．その一方で，均一リンクトポロジを学習したモデル

を流用した場合，高い経路設計成功率の維持は可能なものの，適用するトポロジと学習するトポロジの帯域幅差が輻輳を発生させる要因となっていることが明らかとなった．

そこで，本章では機械学習の一種である，アンサンブル学習を用いて解決を図る．アンサンブル学習とは，複数の機械学習アルゴリズムを持つ学習モデルに同一の教師データを学習させ，各学習モデルが予測した結果を多数決的に判断し，全体の最終的な予測結果とする機械学習技術の一つである．アンサンブル学習は一般的に，学習精度の向上や未知のデータに対する予測能力の向上を図ることができると言われている．

ただし，本提案では学習アルゴリズムの異なる複数の学習モデルを持つアンサンブル学習モデルを作成するのではなく，これまで提案した深層学習モデルに様々な条件のトポロジを学習させ，複数の学習モデルを生成し，これらの学習モデルを組み合わせたアンサンブル学習モデルとする．つまり，各学習モデルに傾斜リンクトポロジにおけるトラヒックデマンドを入力し，それぞれが設計した経路を用いて多数決により最終的な経路とする（図 3.21）．次章では，複数の学習モデルの組み合わせを設計し，経路推定実験により最適なモデルの組み合わせについて考察を行う．

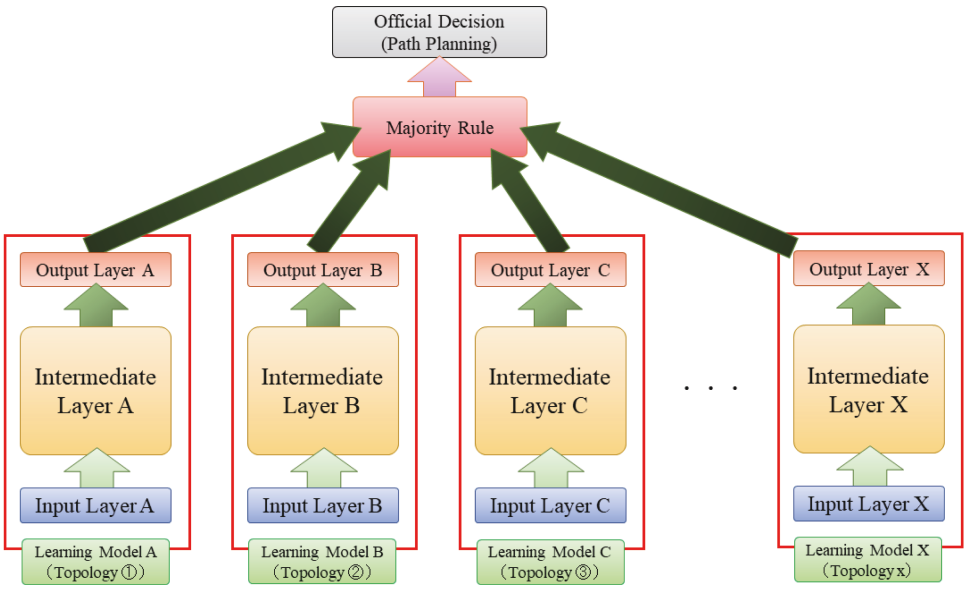


図 3.21 アンサンブル学習モデルの例

3.8.4　アンサンブル学習を用いた深層学習モデルの構成

実験に用いる学習モデルは，傾斜リンクトポロジを学習したものを 4 種類（以下，タイプ A-1〜4）と，帯域幅を 5Gbps から 10Gbps まで 1Gbps 毎に変化させた均一リンクトポロジを学習したものをそれぞれ 4 種類（以下，タイプ B から G-1〜4）．加えて，帯域幅を傾斜リンクトポロジにおける帯域幅の平均値とした均一リンクトポロジを学習したものをそれぞれ 4 種類（以下，タイプ H-1〜4）の計 32 種類作成した．

また，アンサンブル学習モデルを作成する際の組み合わせは，(1) 傾斜リンクトポロジを学習したモデル（タイプ A）のみで構成，(2) 均一リンクトポロジを学習したモデル（タイプ B からタイプ G）のみで構成，(3) 傾斜リンクトポロジ，均一リンクトポロジ，平均リンクトポロジを学習したモデル（タイプ A からタイプ H）からなる混合構成，以上の 3 種類とする．一般的にアンサンブル学習モデルでは，多数決の原理より奇数の組み合わせが推奨されているが，本実験では奇数，偶数問わず様々なパターンを検証し，性能評価を行う．

3.8.4.1　アンサンブル学習 パターン (1) の総数

パターン (1) は，傾斜リンクトポロジを学習したモデル（タイプ A）のみで構成されたアンサンブル学習である．この場合に想定される組み合わせは，${}_4C_2 + {}_4C_3 + {}_4C_4$ の計 10 パターンである．

3.8.4.2　アンサンブル学習 パターン (2) の総数

パターン (2) は，均一リンクトポロジを学習したモデル（タイプ B からタイプ G）のみで構成されたアンサンブル学習である．この場合に想定される組み合わせは，${}_{24}C_2 + {}_{24}C_3 + {}_{24}C_4 \cdots {}_{24}C_{24}$ の計 16,777,191 パターンである．

3.8.4.3　アンサンブル学習 パターン (3) の総数

パターン (3) は，傾斜リンクトポロジ，均一リンクトポロジ，平均リンクトポロジを学習したモデル（タイプ A からタイプ H）から構成されたアンサンブル学習である．この場合に想定される組み合わせは，${}_{32}C_2 + {}_{32}C_3 + {}_{32}C_4 \cdots {}_{32}C_{32}$ の計 4,294,967,263 パターンである．

3.8.5 実験条件

本章では，アンサンブル学習を用いた深層学習モデルを用いて傾斜リンクトポロジにおける提案方式の性能を確認するのが目的である．実験評価トポロジには，第 3.7.2 章にて用いた帯域幅に傾斜を与えた小規模なネットワークである 6 ノードトポロジ（図 3.22）を用いる．なお，従来の実験と同様に教師データは 10,000 セット，テストデータは 5 セット生成する．なお，要求帯域幅の設定については，(株）KDDI 法人・ビジネス向け「国内イーサネット専用サービス」における，帯域選択型の専用線サービスを参考にした [24]．この最適化されたモデルを用いて，経路設計成功率，最大負荷リンクにおける使用帯域幅，および経路計算時間の観点から評価を行った．

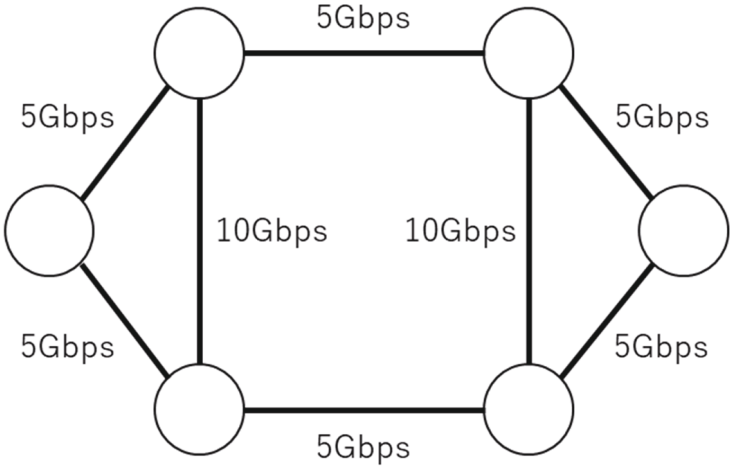


図 3.22 不均一な帯域幅を持つトポロジの例

3.8.6　実験用トポロジを用いた性能評価

3.8.6.1　経路設計成功率

先ず，表 3.5，図 3.23 はパターン (1) の条件における経路設計成功率の上位 5 組である．傾斜リンクトポロジ学習モデルのみを組み合わせた場合のアンサンブル学習モデルによる経路設計は，傾斜リンクトポロジ学習モデル単体での経路設計と比較すると，全体的に経路設計精度が向上していることがわかる．しかし，経路設計成功率が 100%となる構成が存在しないうえに，経路設計に成功したテストケースに着目すると，ほとんどの場合で最大負荷リンクの使用帯域幅が 5Gbps を超えており，輻輳が発生しているのが確認できる．なお，図 3.23 には一部，グラフの存在しないアンサンブル学習モデルが見られるが，これはテストケースにおける経路設計成功率が 100%とならず評価できなかったものである．

表 3.5 アンサンブル学習 パターン (1) における経路設計成功率

モデル番号	経路推定成功数	経路設計成功率 (%)
(1)-1：A-2,3	147	99.3
(1)-2：A-1,2,4	147	99.0
(1)-3：A-1,2,3,4	147	99.0
(1)-4：A-1,2,3	146	97.3
(1)-5：A-1,2	146	97.3

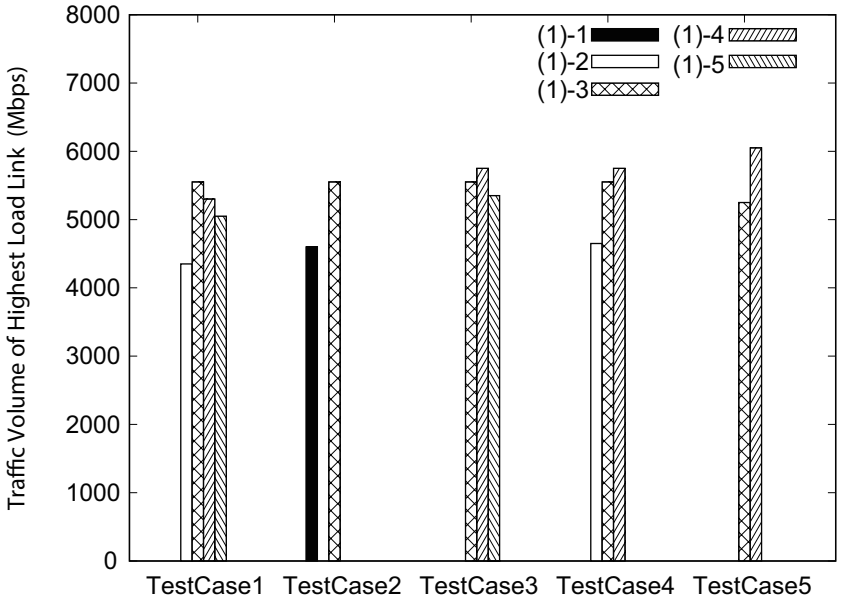


図 3.23 アンサンブル学習 パターン (1) における最大負荷リンクの使用帯域幅

次に，表 3.6，図 3.24 はパターン (2) の条件における経路設計成功率の上位 5 組である．均一リンクトポロジ学習モデルのみを組み合わせた場合のアンサンブル学習モデルによる経路設計は，上位 5 組全てにおいて経路設計成功率が 100%であり，安定して高精度の経路設計が行えている．その一方で，輻輳の発生は均一リンクトポロジ学習モデル単体で設計された経路と比較すると，全体的に抑えられているものの，テストケース 3 を中心に輻輳が発生しているのが確認できる．

表 3.6 アンサンブル学習 パターン (2) における経路設計成功率

モデル番号	経路推定成功数	経路設計成功率 (%)
(2)-1：F,G-1,2,3	150	100.0
(2)-2：F,G-1,2,3,4	150	100.0
(2)-3：F,G-2,3,4	150	100.0
(2)-4：F,G-1,3,4	150	100.0
(2)-5：G-1,2,3,4	150	100.0

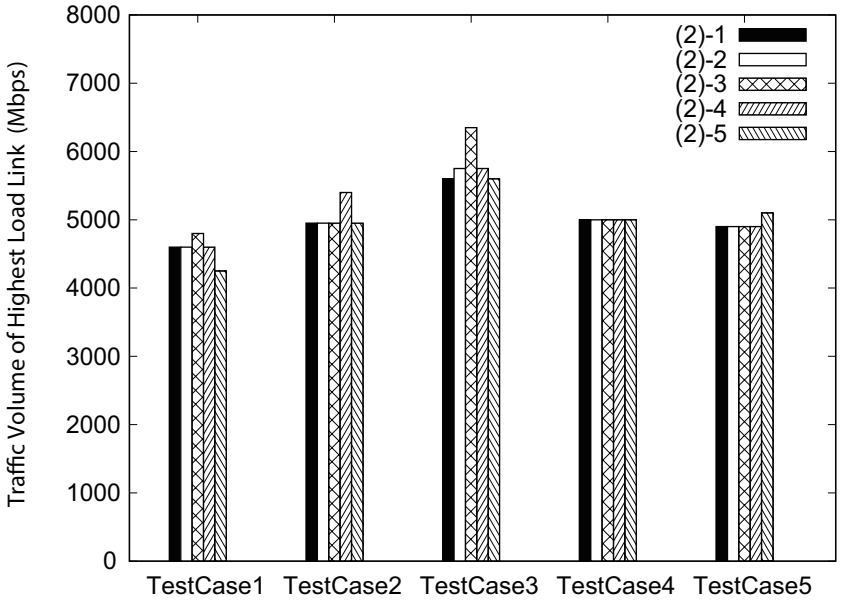


図 3.24 アンサンブル学習 パターン (2) における最大負荷リンクの使用帯域幅

最後に，表 3.7，図 3.25 はパターン (3) の条件における経路設計成功率の上位 5 組である．この場合も上位 5 組全てにおいて，経路設計成功率が 100%を達成している．また，モデル番号 (3)-1，(3)-2，(3)-3 において，全てのテストケースで最大負荷リンクにおける使用帯域幅が 5Gbps を下回っており，輻輳が発生していないことが確認できた．

表 3.7 アンサンブル学習 パターン (3) における経路設計成功率

モデル番号	経路推定成功数	経路設計成功率 (%)
(3)-1：A-1,2+F,G-2,3,4+H-1,2,3	150	100.0
(3)-2：A-1,2+F,G-1,2,3+H-2,3	150	100.0
(3)-3：A-1,2+F,G-1,2,3,4+H1,2	150	100.0
(3)-4：A-1,2+F,G-2,3,4	150	100.0
(3)-5：A-1,2+F,G-1,2,3,4	150	100.0

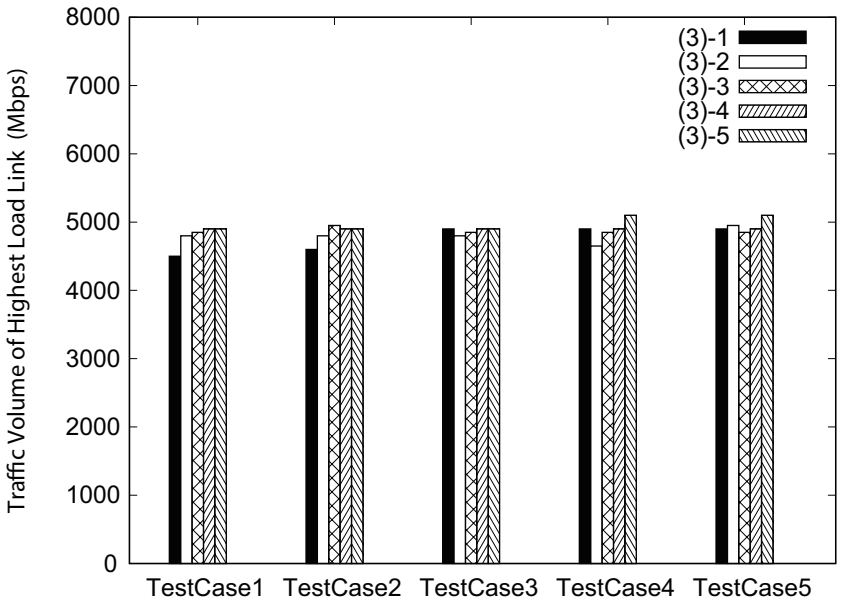


図 3.25 アンサンブル学習 パターン (3) における最大負荷リンクの使用帯域幅

3.8.6.2 帯域幅使用率

ここで，最も性能が良いモデル (3)-1 を用いて設計された経路と，ダイクストラ法を用いて設計された経路における，最大負荷リンクの使用帯域幅を比較した結果が図 3.26 である．全てのテストケースにおいて，Dijkstrah 法と同等かやや劣る結果ではあるが，均一リンクトポロジに対して設計された経路とほぼ同等の輻輳制御性能となっており，複数のモデルを組み合わせたアンサンブル学習モデルを用いた場合でも，十分に負荷分散された経路が設計できることを示した．

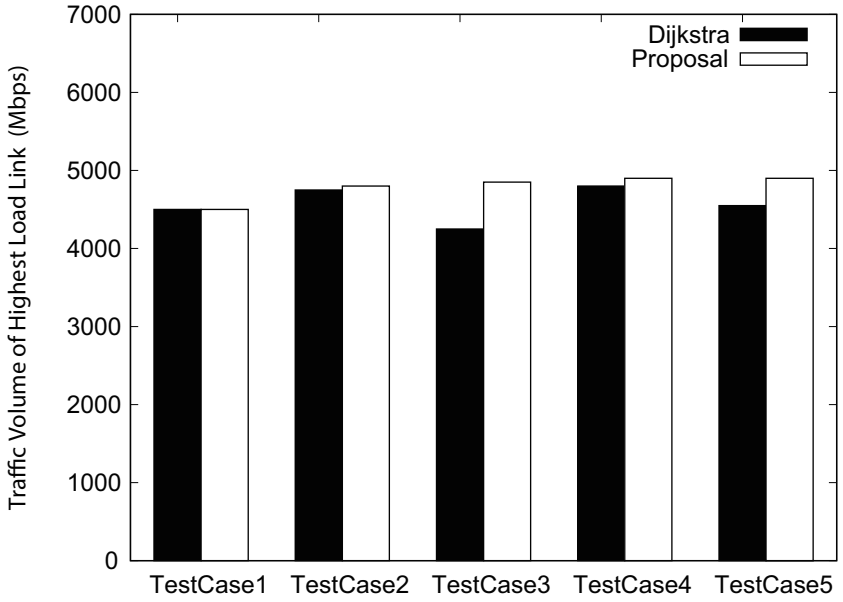


図 3.26 アンサンブル学習を用いた最良モデルにおける最大負荷リンクの使用帯域幅

3.8.7 アンサンブル学習を用いた深層学習モデルの考察

実験結果より，アンサンブル学習モデルを作成する際の深層学習モデルの最適な組み合わせは，大きく分けて 3 つの構成からなる．先ずは，適用するトポロジを学習したモデルが，経路設計の中心的な役割を持つベースモデルとして働く．次に，均一リンク（適用するトポロジの最大帯域幅）トポロジを学習したモデルが，接続関係を持つ正しいリンクを選択し経路設計成功率を向上させる補助モデルとして働く．

最後に，均一リンク（適用するトポロジの平均帯域幅）トポロジを学習したモデルが，トポロジ内の帯域幅の差から生じる輻輳を抑制するために働く．この様な構成からなるアンサンブル学習モデルを用いて，基幹ネットワークの様な帯域幅が均一でないトポロジにおいても輻輳制御された経路の設計が行えると考えている．

3.8.8　アンサンブル学習を用いた深層学習モデルの課題

アンサンブル学習を用いることで，現実的なネットワークモデルである不均一な帯域幅を持つトポロジにおいても，負荷分散を考慮した経路を高い成功率で設計できることが分かった．しかしながら，アンサンブル学習の課題として，適用したいネットワークトポロジに対して適切な学習モデルの構成を簡単に見つけることが困難であることが挙げられる．実験結果から，適切な学習モデルの組み合わせについて一定の指標を得ることはできたものの，少なくない種類の学習モデルを生成する必要がある．そのため，今後は適切なアンサンブル学習の組み合わせを一般化する研究やアンサンブル学習を用いずに不均一な帯域幅を持つトポロジを学習できるような学習モデルを検討する必要がある．

3.9 輻輳制御のためのダイクストラ法を用いた深層学習による経路設計法の提案

3.9.1 本章の目的

これまで，様々な深層学習モデルを用いて負荷分散を考慮した経路を設計させることを検討してきた．結果として，単純な深層ニューラルネットワークからなる学習モデルは短い時間で経路設計が行えるものの，適用可能なトポロジの規模の限界が非常に低かった．また，回帰型深層ニューラルネットワークからなる学習モデルを用いることで，適用可能なトポロジの規模の限界は大きくなるものの，経路計算にかかる時間が増加してしまうことが分かった．ここで，回帰型ニューラルネットワークに着目すると，単純な深層ニューラルネットワークとの違いは，一つ前のニューラルネットワークの出力が入力されることで入力情報が増えることと，拠点間トラヒックの数と同じ数だけ中間層ブロックが増えていることである．特に，中間層はある一つの拠点間トラヒックに対してのみ経路情報を学習すれば良いため，学習精度が向上しているのではないかと考えた．そこで，本章では単純な深層ニューラルネットワークにおける中間層の構成を，拠点間トラヒックの数と同数に分割させた新しい深層ニューラルネットワークモデルを提案する．加えて，これまでの実験では適切な中間層の層数やデータセットのデータ数などを検討していなかったため，これらについても事前実験により適切な設定を検討する．

3.9.2 深層学習モデルの改良

図 3.27 は，深層学習モデルの構成例を示している．提案する深層学習モデルの中間層は，複数のブロックから構成され，一つのブロックは入力層と同じ数のユニットを持つ．また，中間層は深さが 3 層以上で，それぞれの深さで複数のブロックを持つ．このとき，一つの深さにつき ${}_N\mathrm{P}_2$ 個のブロックを持つ．加えて，ブロックは各拠点間トラヒック毎に連結関係を持つ．なお，一般的に深層学習における中間層は深さが 3 層以上であり，図 3.27 は中間層が 4 層の場合を示している．

入力層から中間層への結合は，深さが 1 であるすべての中間層のユニットに対して全結合させる．また，ブロックは同一拠点間トラヒックに関する前段と後段のユニットに対して全結合させる．このとき，異なる拠点間トラヒックに関する中間層ブロック同士は一切の結合関係を持たない．そして，中間層から出力層は，各拠点間トラヒックと対応しているブロックの最深から，該当するトラヒックの出力層とのみ全結合させる．ただし，出力層は一つの拠点間トラヒックに対して N 個の one-hot ベクトルから構成される．

従来の機械学習を用いた経路設計法との大きな違いは，単一の中間層にすべての経路情報を学習させ，経路設計を行わせるのではなく，中間層と出力層のブロックが，一つの拠点間トラヒックに対する経路情報を学習し，経路設計を行うことである．これにより，単

一の中間層を用いて多数の正解データを学習させた場合に，学習が収束せずに十分な性能を得られないという課題を解決することができる．この課題は，一度に膨大な出力を計算させてしまうことで，出力パターンが非常に複雑化することで，特徴量を十分に学習することができないことが最大の原因である．特に経路設計の場合は，類似している訓練データに対して，全く異なる正解データが複数存在することもあり，学習が著しく阻害される要因となっている．提案方式では，中間層と出力層をブロック化し学習させる範囲を一つの拠点間トラヒックにおける経路情報に限定することで，学習精度の向上を果たしている．

その一方で，中間層と出力層だけではなく，入力層も一つの拠点間トラヒックのみを入力とする方法も挙げられる．この場合，学習モデルの規模が縮小することで，性能が限られたデバイスでも機械学習を用いた経路設計が行えると考えられる．しかし，この方法では出力パターン数が減少し学習が収束しやすくなる一方で，全く同じ要素からなる訓練データに対して，複数の正解データを学習することができないという問題がある．つまり，本研究が目的とするネットワーク状況に応じた動的な経路設計が行えない．また，学習に必要な情報量を増やす目的で，訓練データにトラヒックの受け付け順やネットワークの残余帯域などの情報を組み込むこと方法も挙げられる．しかし，この方法では訓練データの種類が，正解データの種類よりも大幅に増えてしまうことで，入力パターンの複雑化が増し，特徴量を十分に学習することができないという問題がある．以上より，訓練データをトラヒック単位とした場合は，いずれの手段においても学習自体が成り立たず経路設計を行えるような性能を持つ学習モデルを生成することができない．提案方式の場合は，訓練データをデマンド集合とすることで，デマンド集合に含まれる他の送信元，宛先ノード番号や要求帯域幅の情報により，残余帯域などの情報を用いずとも，デマンド集合全体を総合的に収容するような経路を学習できる．

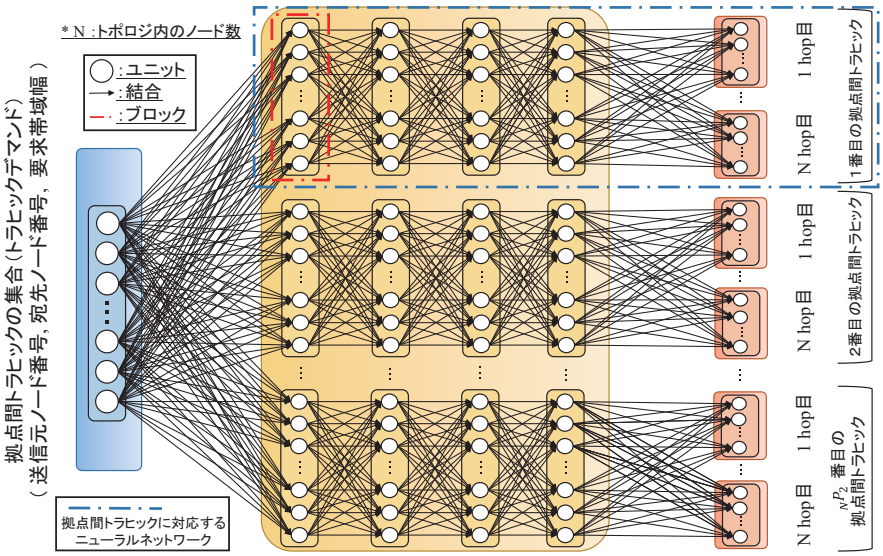


図 3.27 深層学習モデルの構成例

3.9.3 深層学習モデルの事前実験

事前実験は，以下の 2 つの実験からなる．事前実験 1 では，中間層の深さの異なる複数の学習モデルを作成し，実験結果から以降採用する中間層の深さを決定する．事前実験 2 では，教師データおよびテストデータのデータセット数を複数種類検討し学習させる．この実験により，経路設計成功率と学習時間を考慮した最適なデータセット数を決定する．

なお，実験に用いるトポロジは以下の複数の 6 ノードトポロジ (図 3.28) である．これは，実験的な意味合いで用いられる小規模ネットワークトポロジ [21] [22] [23] であり，事前実験 1, 事前実験 2 にて用いる．なお，各トポロジ内のすべての帯域幅は 10Gbps で統一する．深層学習モデルの実装と実験は，OS:Ubuntu 20.04.1， 物理メモリ:128GB， GPU:GeForce RTX 3090， プログラミング言語:Python 3.8.5， Tensorflow2.3.1+Keras を用いて行った．

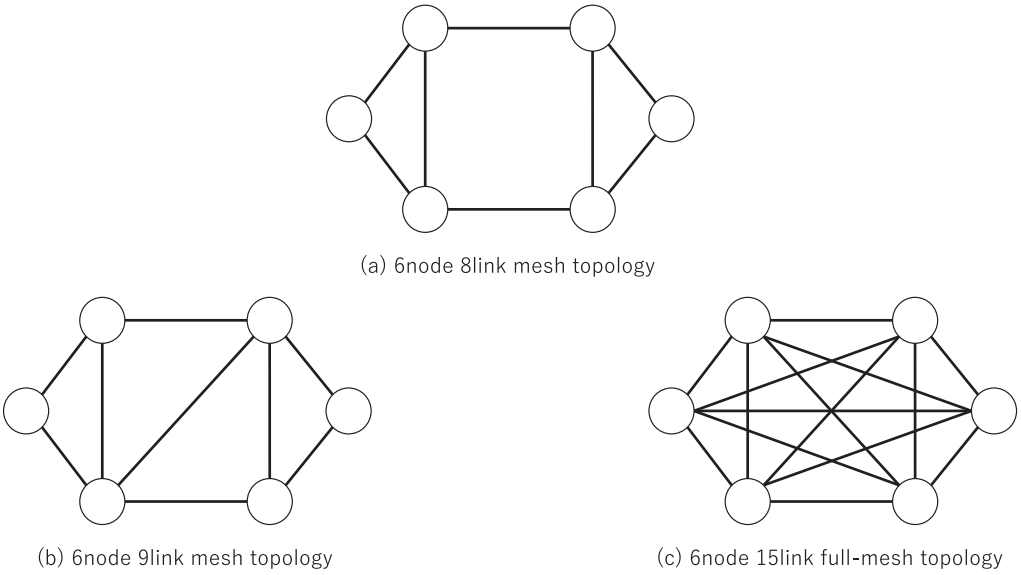


図 3.28 実験用トポロジ

3.9.3.1　事前実験 1 深層学習モデルの中間層数の検討

表 3.8，3.9，3.10 は，実験用トポロジにおける経路設計成功率を示したものである．なお，経路設計成功率は，ダイクストラ法による経路と一致している送受信間経路の割合を用いる．この実験で用意した深層学習モデルの構成は，次の通りである．中間層の深さは 3 層，4 層，5 層．一度の学習で用いるデータセット数であるバッチサイズは BS:64, BS:128, BS:256．これらを組み合わせた合計 9 種類とする．また，教師データおよび評価用データのデータセット数は 10,000 セットとした．

結果より，すべてのトポロジにおいて中間層の深さが 4 層および 5 層のとき，高い成功率を示している．この結果から，実験 2 以降では深さを 4 層および 5 層を基本として，適宜ハイパーパラメータの調整を行いつつ高い性能を持つ学習モデルを作成する．

表 3.8 事前実験 1 トポロジ (a) における経路設計成功率 [%]

	number of batch size		
中間層	**BS:64**	**BS:128**	**BS:256**
中間層 3 層	54.07	32.53	18.80
中間層 4 層	59.13	41.66	21.59
中間層 5 層	54.46	41.24	16.04

表 3.9 事前実験 1 トポロジ (b) における経路設計成功率 [%]

	number of batch size		
中間層	**BS:64**	**BS:128**	**BS:256**
中間層 3 層	74.91	44.60	45.18
中間層 4 層	82.13	77.20	57.55
中間層 5 層	81.33	84.72	52.20

表 3.10 事前実験 1 トポロジ (c) における経路設計成功率 [%]

	number of batch size		
中間層	**BS:64**	**BS:128**	**BS:256**
中間層 3 層	99.76	63.32	65.28
中間層 4 層	99.68	99.85	99.91
中間層 5 層	99.59	99.56	99.69

3.9.3.2 事前実験 2 適切なデータセット数の検討

表 3.11，3.12，3.13 は，実験用トポロジにおける平均経路設計成功率を示したものである．なお，経路設計成功率は，ダイクストラ法による経路と一致している送受信間経路の割合を用いる．表中に記載されている，() 内の数値は全 5 回の試行における最大値と最小値の差 [%] であり，成功率の揺らぎである．実験に用いる教師データと評価用データのデータセット数は，それぞれ 10,000 セット，100,000 セット，1,000,000 セットの 3 種類とした．また，中間層は深さを 4 層とし，バッチサイズを 128 に固定した．

結果より，教師データに着目した場合，すべてのトポロジにおいてデータセット数が増加するほど，精度が高くなっていることが分かる．また，評価用データに着目した場合，すべてのトポロジにおいてデータセット数が多くなるほど，成功率の揺らぎが小さくなっていることが分かる．ただし，評価用データの計算時間は，100,000 セットでは約 20 分程度に対して，1,000,000 セットでは約 3 時間程度を要した．加えて，計算時間はトポロジ内のノード数が増加するほど，急激に増加する傾向が見られるため無視することができない．以上より，学習精度，精度誤差，計算時間それぞれの観点から総合的に判断し，実験 3 および 4 以降では教師データ数を 1,000,000 セット，テストデータ数を 100,000 セットとする．

表 3.11 事前実験 2 トポロジ (a) における平均経路設計成功率 [%]

	number of test data		
number of train data	**Test:10,000**	**Test:100,000**	**Test:1,000,000**
Train:10,000	42.476	42.697	42.739
	(0.01190)	(0.00205)	(0.00051)
Train:100,000	85.050	85.034	85.061
	(0.00310)	(0.00257)	(0.00011)
Train:1,000,000	97.402	97.419	97.421
	(0.00540)	(0.00048)	(0.00036)

表 3.12 事前実験 2 トポロジ (b) における平均経路設計成功率 [%]

	number of test data		
number of train data	**Test:10,000**	**Test:100,000**	**Test:1,000,000**
Train:10,000	28.456	30.073	30.051
	(0.08010)	(0.00351)	(0.00067)
Train:100,000	90.404	90.387	90.425
	(0.00770)	(0.00510)	(0.00025)
Train:1,000,000	97.882	97.979	97.986
	(0.00200)	(0.00059)	(0.00002)

表 3.13 事前実験 2 トポロジ (c) における平均経路設計成功率 [%]

	number of test data		
number of train data	**Test:10,000**	**Test:100,000**	**Test:1,000,000**
Train:10,000	99.880	99.908	99.895
	(0.00040)	(0.00087)	(0.00002)
Train:100,000	99.848	99.885	99.862
	(0.00060)	(0.00152)	(0.00003)
Train:1,000,000	99.954	99.958	99.954
	(0.00050)	(0.00062)	(0.00005)

3.9.3.2.1　データセットの組み合わせ総数

実験により適切なデータセット数について検討したが，このデータセット数が全体の組み合わせのどの程度の割合を占めるか確認する必要がある．ここで，本研究で想定する環境では，すべてのノード間で一つずつトラヒックが発生するとしている．また，拠点間トラヒック毎に M 通りの要求帯域幅からランダムに割り当てられる．よって，組み合わせの総数は M の ${}_N\mathrm{P}_2$ 乗 通りである．

表 3.14 は，要求帯域幅が 7 種類の場合の組み合わせ数である．これより，事前実験によって定めたデータセット数は，膨大な組み合わせの一部であることが分かる．よって，正解データと成り得る解空間は十分に広いことから，これらのデータセットを用いた評価によって高い経路設計成功率や負荷分散が確認できた場合，学習モデルは十分に汎用性を学習できていると考えられる．

表 3.14 要求帯域幅が 7 種類の場合の組み合わせ数

number of node	**number of required banwidth**	**formula**	**number of combinations**
6	7	7^{30}	2×10^{25}
7	7	7^{42}	3×10^{35}
8	7	7^{56}	2×10^{47}
9	7	7^{72}	7×10^{60}
10	7	7^{90}	1×10^{76}
11	7	7^{110}	9×10^{92}

3.9.4 深層学習モデルの性能評価実験

事前実験の結果を基に学習モデルを最適化し，性能評価実験を行う．性能評価実験は，以下の 4 つの実験からなる．実験 1 では，これまでに決定した各ハイパーパラメータを用いて複数の学習モデルを実装し学習を行い，実験用トポロジを用いて提案方式の評価を行う．実験 2 では，実験 1 と同様に複数の学習モデルを実装し学習を行い，評価用トポロジを用いて提案方式の評価を行う．実験 3 では，実験 2 で用いた複数の学習モデルを用いて，文献 [13] の Seq2Seq 方式と比較評価を行う．実験 4 では，実験 2 で用いた複数の学習モデルを用いて，トポロジの変化に対する提案方式の汎化性能評価を行う．

評価項目は，経路設計成功率，帯域使用率，計算時間の 3 つである．はじめに，経路設計成功率は次の式から求められる．

$$\frac{\text{ダイクストラ法による経路と一致している送受信間経路の数}}{{}_N\mathrm{P}_2 \times \text{データセット数}} \tag{3.1a}$$

$$\frac{\text{本研究における成功条件を満たした送受信間経路の数}}{{}_N\mathrm{P}_2 \times \text{データセット数}} \tag{3.1b}$$

評価項目 (5.1a) は，ダイクストラ法による経路との一致率であり，一般的な機械学習における正答率に該当する．評価項目 (5.1b) は，本研究における正答率の計算法である．このとき成功条件を満たす経路とは，学習モデルの出力経路において，使用するリンクがネットワーク上に存在し，送信元から宛先まで到達可能であり，ループフリーであり，かつどのリンクにも輻輳が発生していない経路である．次に，帯域使用率は最大負荷リンクにおける使用帯域幅の各値から求める．本研究では，教師データ作成のための経路設計法として，リンクコストを残余帯域の逆数としたダイクストラ法を用いており，輻輳制御を目的としている．そこで，学習モデルが教師データの特性を正しく学習できているかを，帯域使用率の観点からダイクストラ法と比較評価を行う．最後に計算時間は，提案方式とダイクストラ法それぞれを用いて，１つのトラヒックデマンド集合の経路をすべて計算するまでにかかる時間を 10 回計測し，その平均値により比較評価を行う．ただし，計測時間はデータの前処理や後処理を含まず，経路計算のみにかかる時間を対象とする．

なお，一般的に機械学習の分野では，教師データとして作成したデータセットを学習用のデータと評価用のデータに分割し，学習に用いたデータでは評価を行わない．これは，API などの用意されたデータセットや，実際に運用された履歴情報などを用いることから，データ数に限りがあるためである．その一方で，本研究の想定環境では，デマンド集合に含まれる拠点間トラヒックの組み合わせ数は，すべてのノード間の組み合わせ数と要求帯域幅の値，更にはそれらを要求帯域幅の値で降順にソートするため，非常に膨大なパターンが存在する．そのため，教師データとテストデータを別々に生成しても，データセット中に同一のデマンド集合が生成される確率は非常に低く，評価実験において未知のデータに対する検証を行えると考えている．よって，教師データの一部を分割し，学習と評価に

それぞれ用いる方法ではなく，教師データとは別にデータセットを生成し評価することで，学習モデルが汎用性のある学習を行っているか検討する．なお，要求帯域幅は組み合わせ数に大きな影響を与え，種類が多くなるほど学習難易度が上昇してしまう．よって本研究では，難易度の低減を図るために 150Mbps から 900Mbps までの 150Mbps 刻みの 6 種類に，1Gbps を合わせた 7 種類からランダムで選択する．この値は，KDDI の帯域保障サービスにおける選択可能な帯域幅を参考にした [24].

実験に用いるトポロジは以下の通りである．実験用トポロジは複数の 6 ノードトポロジ (図 3.29) である．これは，実験的な意味合いで用いられる小規模ネットワークトポロジ [21] [22] [23] であり，実験 1 にて用いる．また，評価用トポロジは 6 ノードから 11 ノードの複数のトポロジ (図 3.30) である．これらは，実際に世界各地で運用経験のある小規模なトポロジであり，Topology Zoo [26] から参照し抜粋し，実験 2,3,4 にて用いる．なお，各トポロジ内のすべての帯域幅は 10Gbps で統一する．深層学習モデルの実装と実験は，OS:Ubuntu 20.04.1， 物理メモリ:128GB， GPU:GeForce RTX 3090， プログラミング言語:Python 3.8.5， Tensorflow2.3.1+Keras を用いて行った．

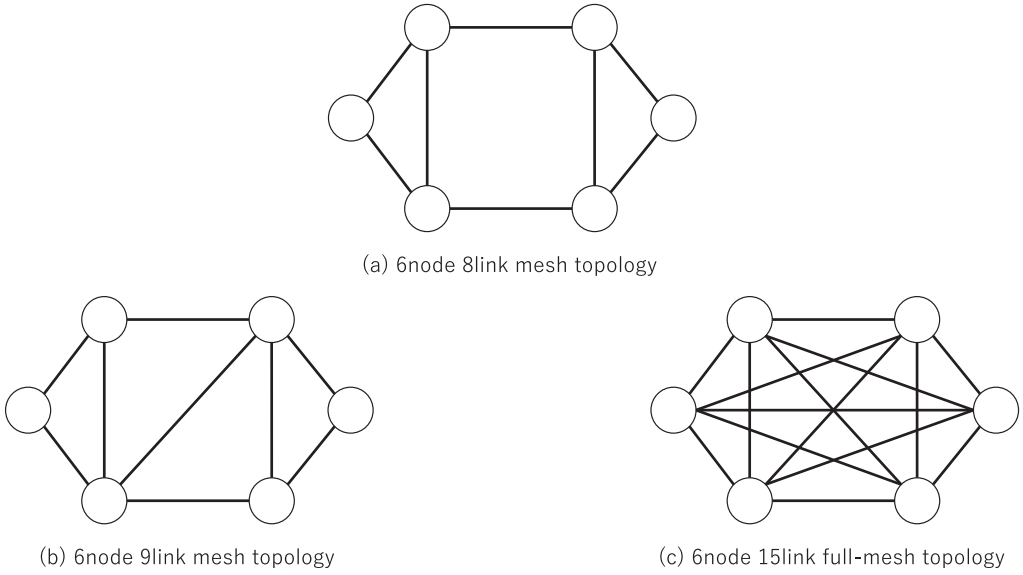


図 3.29 実験用トポロジ

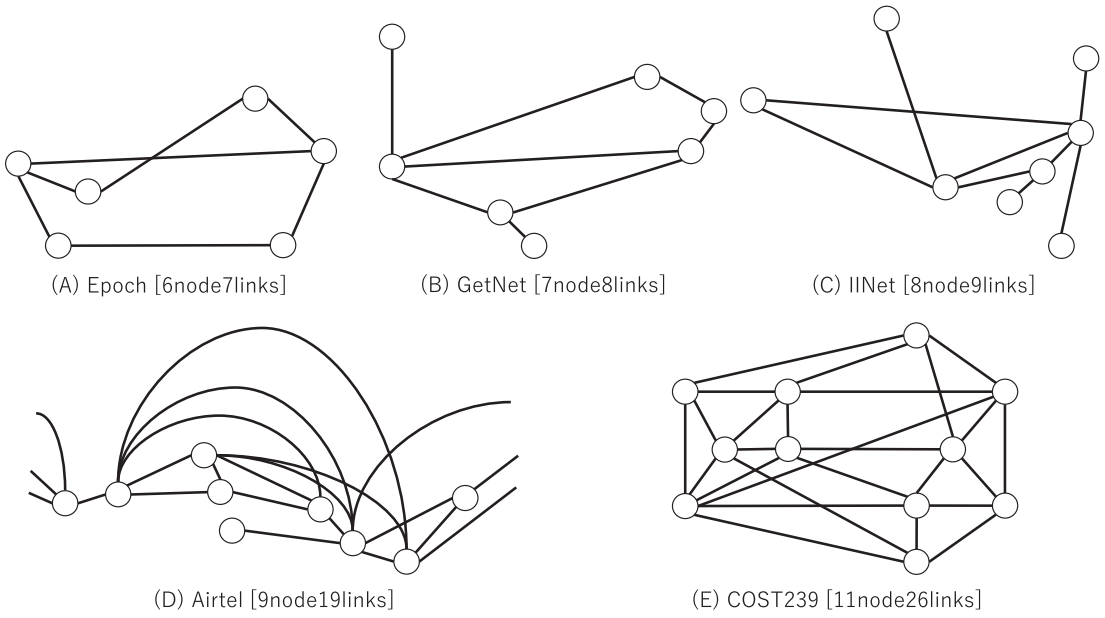


図 3.30 評価用トポロジ

3.9.4.1　実験 1 実験用トポロジを用いた性能評価

はじめに，実験 1 で用いるモデルの学習にかかった Epoch 数を示す．それぞれ最大学習回数 1,000 回に対して，トポロジ (a) は 124 回，トポロジ (b) は 145 回，トポロジ (c) は 166 回であった．以上より，すべてのモデルにおいて学習は早期終了しており，学習が収束していることを確認した．

3.9.4.1.1　経路設計成功率

図 3.31 は，実験用トポロジにおける拠点間トラヒック毎の経路設計成功率である．トポロジ (a) における評価 (5.1a) は，92.23% (2,766,769/3,000,000 トラヒック)，評価 (5.1b) は，99.91% (2,997,411/3,000,000 トラヒック) であった．トポロジ (b) における評価 (5.1a) は，88.45% (2,653,587/3,000,000 トラヒック），評価 (5.1b) は，99.99% (2,999,970/3,000,000 トラヒック) であった．トポロジ (c) における評価 (5.1a) は，99.99% (2,999,961/3,000,000 トラヒック)，評価 (5.1b) は，99.99% (2,999,989/3,000,000 トラヒック) であった．

図 3.31 より，教師データとの一致率は全体的に高いものの，トポロジ (a),(b) では約 1 割程度が誤答であった．これは，少なくとも約 1 割のデマンド集合において，通信が行えないような経路が含まれていることを表している．これに対して，本研究における成功条件を満たした経路は，非常に高い精度で通信可能な経路を設計できている．よって，学習モデルは教師データとは異なる経路を設計する場合があるものの，通信可能な経路を選択しておりダイクストラ法の持つ動的な性質を正しく学習していることが分かった．

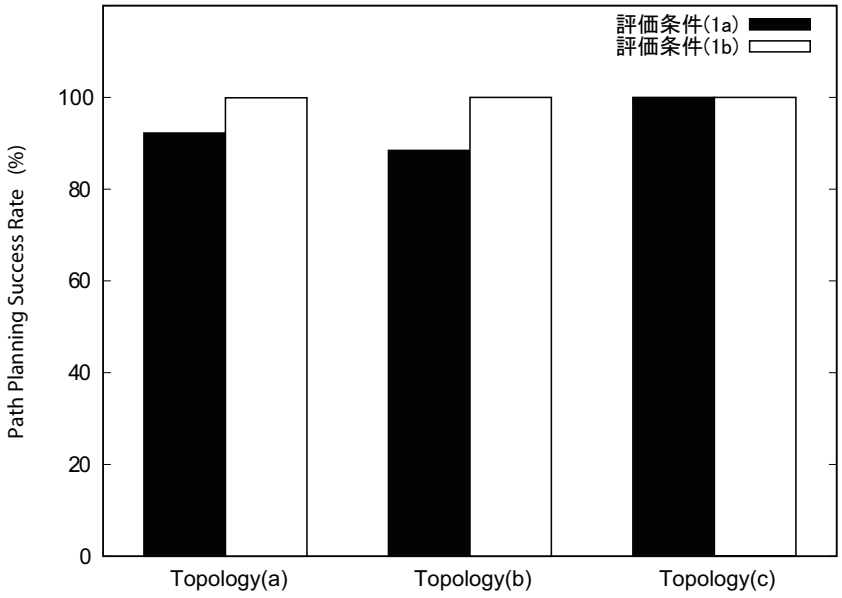


図 3.31 実験 1 拠点間トラヒック毎の経路設計成功率

3.9.4.1.2 最大負荷リンクにおける帯域幅使用率

図 3.32 は，すべての拠点間トラヒックを流した際の最大負荷リンクにおける使用帯域幅の平均値を比較したグラフであり，表 3.15 は，最大負荷リンクにおける最大値 (Max)，最小値 (Min)，平均値 (Average)，標準偏差 (SD:StandardDeviation)，標準誤差 (SE:StandardError)，中央値 (Median) を記載したものである．ただし，提案方式はデマンド集合に含まれるすべてのトラヒックの経路が設計できている場合にのみ計算する．これは，経路設計に失敗している場合，本来流れるべきトラヒックが流れず計算できないためである．

図 3.32 および，表 3.15 より，帯域使用率の観点から見ると，教師データとは異なる経路を設計している影響によりダイクストラ法と比較して使用帯域幅の平均値がやや増加しているものの，すべてのトポロジにおいて最大値が帯域幅を超えていないため，輻輳は発生していない．加えて，標準誤差を用いて 99%信頼区間を求めた．先ず，トポロジ (a) のとき，ダイクストラ法では 3,205.68 ± 3.496，提案方式では 3,491.68 ± 4.298．トポロジ (b) のとき，ダイクストラ法では 2,586.68 ± 2.778，提案方式では 2,892.12 ± 3.429．トポロジ (c) のとき，ダイクストラ法と提案方式共に 998.99 ± 0.082 であった．標準誤差は非常に小さい値であることから，想定環境において性能が大幅に低下するような場面は存在しない．以上より，学習モデルは輻輳制御を目的としたダイクストラ法が持つリンク選択の動的性質を学習できることが示された．

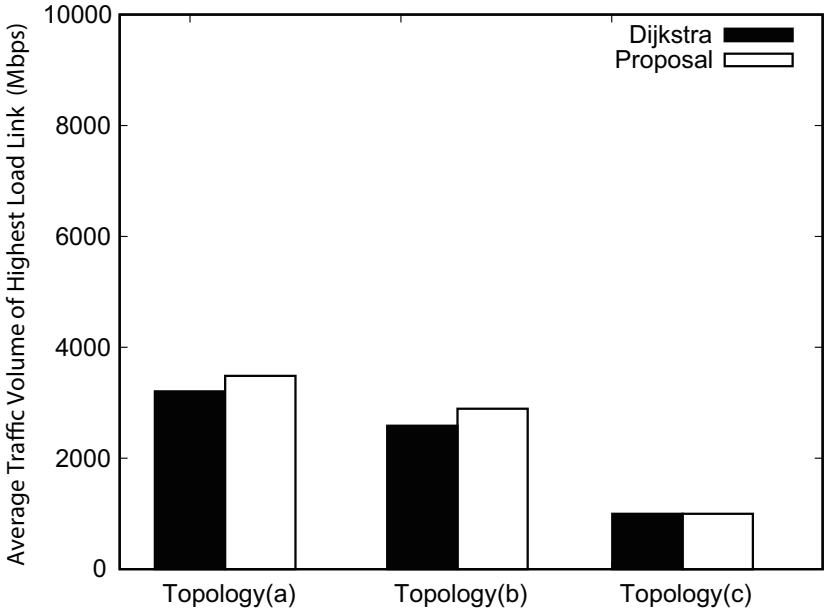


図 3.32 実験 1 最大負荷リンクにおける使用帯域幅の平均値

表 3.15 実験 1 最大負荷リンクにおける使用帯域幅の詳細値 [Mbps]

		Traffic Volume of highest Load Link					
Topology	**Method**	**Max**	**Min**	**Average**	**SD**	**SE**	**Median**
Topology(a)	**Dijkstra**	5,600	1,500	3,205.68	429.14	1.357	3,200
	Proposal	5,800	1,650	3,491.68	527.67	1.669	3,450
Topology(b)	**Dijkstra**	4,300	1,350	2,586.68	341.08	1.079	2,550
	Proposal	4,800	1,500	2,892.12	420.99	1.331	2,850
Topology(c)	**Dijkstra**	1,000	750	998.99	10.05	0.032	1,000
	Proposal	1,000	750	998.99	10.05	0.032	1,000

3.9.4.1.3　経路計算時間

最後に，表 3.16 は，${}_N\mathrm{P}_2$ 個の拠点間トラヒックからなるデマンド集合に対する経路設計時間の平均値である．提案方式は，ダイクストラ法よりもほんの僅かに計算時間を要するものの，ほぼ同等の結果であった．以上より，提案方式は小規模トポロジに対して経路設計を行う場合に，多項式時間で経路計算が可能なダイクストラ法とほぼ同等の計算時間であることが示された．

表 3.16 実験 1 実験用トポロジにおける平均計算時間 [s]

	Topology		
Method	**(a)**	**(b)**	**(c)**
Dijkstra	0.010	0.010	0.009
Prpposal	0.014	0.014	0.014

3.9.4.2 実験 2 評価用トポロジを用いた性能評価

はじめに，実験 2 で用いるモデルの学習にかかった Epoch 数を示す．それぞれ最大学習回数 1,000 回に対して，トポロジ (A) は 322 回，トポロジ (B) は 183 回，トポロジ (C) は 199 回，トポロジ (D) は 285 回，トポロジ (E) は 144 回であった．以上より，実験 3 と同様にすべてのモデルにおいて学習は早期終了しており，学習が収束していることを確認した．

3.9.4.2.1 経路設計成功率

図 3.33 は，評価用トポロジにおける拠点間トラヒック毎の経路設計成功率である．トポロジ (A) における評価 (5.1a) は，82.54% (2,476,067/3,000,000 トラヒック)，評価 (5.1b) は，99.97% (2,999,104/3,000,000 トラヒック) であった．トポロジ (B) における評価 (5.1a) は，95.13% (3,995,642/4,200,000 トラヒック)，評価 (5.1b) は，99.99% (4,199,901/4,200,000 トラヒック) であった．トポロジ (C) における評価 (5.1a) は，98.26% (5,502,690/5,600,000 トラヒック)，評価 (5.1b) は，99.99% (5,599,943/5,600,000 トラヒック) であった．トポロジ (D) における評価 (5.1a) は，88.94% (6,403,509/7,200,000 トラヒック)，評価 (5.1b) は，99.96% (7,197,219/7,200,000 トラヒック) であった．トポロジ (E) における評価 (5.1a) は，81.77% (8,995,080/11,000,000 トラヒック)，評価 (5.1b) は，99.52% (10,946,756/11,000,000 トラヒック) であった．

図 3.33 より，教師データとの一致率は最大 2 割程度が誤答であり，少なくとも 2 割を超えるデマンド集合において，通信が行えないような経路が含まれていた．これに対して，本研究における成功条件を満たした経路は，非常に高い精度で通信可能な経路を設計できている．よって，実験用トポロジと同様に学習モデルはダイクストラ法の性質を正しく学習できることが分かった．

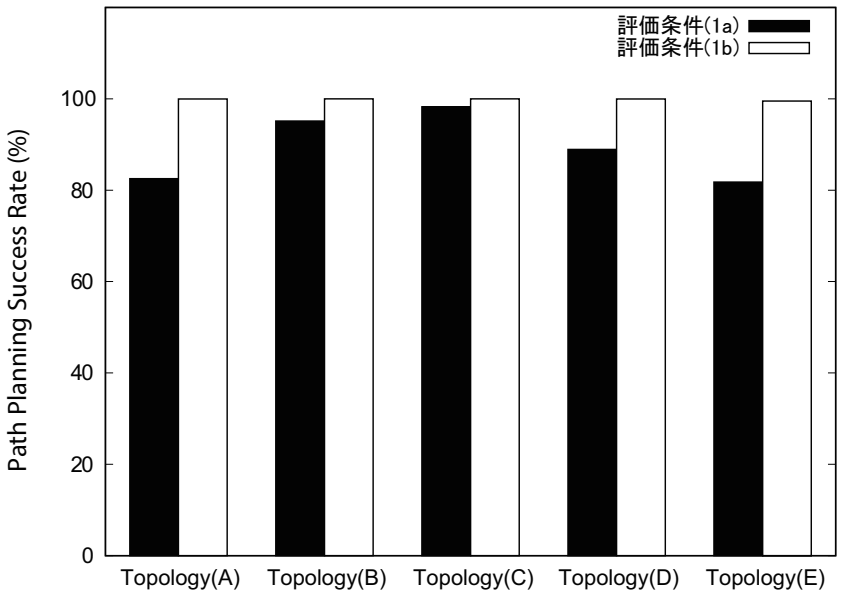


図 3.33 実験 2 拠点間トラヒックにおける経路設計成功率

3.9.4.2.2　最大負荷リンクにおける帯域幅使用率

図 3.34 は，すべての拠点間トラヒックを流した際の最大負荷リンクにおける使用帯域幅の平均値を比較したグラフであり，表 3.17 は，最大負荷リンクにおける最大値 (Max)，最小値 (Min)，平均値 (Average)，標準偏差 (SD:StandardDeviation)，標準誤差 (SE:StandardError)，中央値 (Median) を記載したものである．ただし，トラヒックの計算は実験 3 と同様にデマンド集合全体で経路設計に成功した場合のみ計算する．

図 3.34 および表 3.17 より，帯域使用率の観点から見ると，実験 3 と比べてダイクストラ法との差が全体的に小さくなっているのが分かる．特に，トポロジ (B) や (C) は次数が 1 となるノードが含まれていることや，トポロジ (D) や (E) はノード数，リンク数ともに大きいため選択すべき経路が多い．つまり，正しく教師データの動的性質を学習できていない場合，リンク選択に偏りが生まれ，最大負荷リンクにおける使用帯域幅が増加し，最悪の場合は輻輳が発生する．しかし実際には，トポロジ (B) や (C)，(D) はダイクストラ法と遜色がない負荷分散が行えており，最も小さな規模であるトポロジ (A) の方がやや劣る結果であった．これは，リンク数の少ないトポロジ (A) は選択可能な経路パターンが少なく，一部のリンクが偏って選択されたためと考えられる．加えて，実験 3 と同様にすべてのトポロジにおいて使用帯域幅の最大値が帯域幅を超えていないため，輻輳が発生していないことが確認された．

また，標準誤差を用いて 99%信頼区間を求めた．先ず，トポロジ (A) のとき，ダイクストラ法では 3,032.21 $\pm$ 2.919，提案方式では 3,398.79 $\pm$ 3.808．トポロジ (B) のとき，ダイクストラ法では 4,408.95 $\pm$ 3.736，提案方式では 4,450.35 $\pm$ 3.703．トポロジ (C) のとき，ダイクストラ法では 5,231.99 $\pm$ 3.694，提案方式では 5.236.83 $\pm$ 3.710．トポロジ (D) のとき，ダイクストラ法では 5,204.43 $\pm$ 5.457，提案方式では 5.204.48 $\pm$ 5.411．トポロジ (E) のとき，ダイクストラ法では 3,351.98 $\pm$ 2.524，提案方式では 3,902.73 $\pm$ 3.753 であった．実験 4 においても，標準誤差は非常に小さい値であることから，評価用トポロジにおいても想定環境において性能が大幅に低下するような場面は存在しない．以上より，実験 3 と同様に学習モデルは，現実的なネットワークトポロジにおいても，輻輳制御を目的としたダイクストラ法が持つリンク選択の動的性質を学習できることが示された．

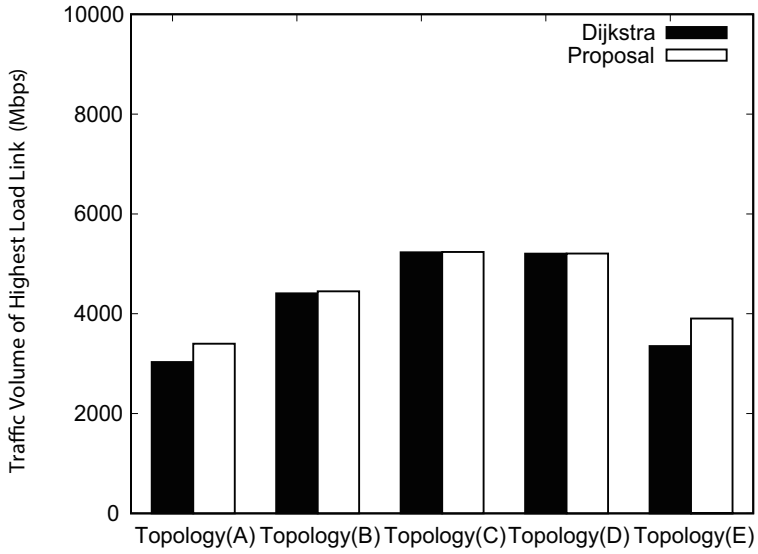


図 3.34 実験 2 最大負荷リンクにおける使用帯域幅の平均値

表 3.17 実験 2 最大負荷リンクにおける使用帯域幅の詳細値 [Mbps]

		Traffic Volume of highest Load Link					
Topology	**Method**	**Max**	**Min**	**Average**	**SD**	**SE**	**Median**
Topology(A)	**Dijkstra**	4,800	1,750	3,032.21	1.133	358.32	3,000
	Proposal	5,900	1,950	3,398.79	1.478	467.46	3,400
Topology(B)	**Dijkstra**	6,000	2,700	4,408.95	1.450	458.63	4,400
	Proposal	6,450	2,800	4,450.35	1.436	454.58	4,450
Topology(C)	**Dijkstra**	6,900	3,550	5,231.99	1.434	453.46	5,200
	Proposal	7,200	3,550	5,236.83	1.440	455.51	5,200
Topology(D)	**Dijkstra**	7,700	2,800	5,204.43	2.116	669.93	5,200
	Proposal	7,700	2.800	5204.48	2.101	664.26	5,200
Topology(E)	**Dijkstra**	5,300	2,250	3,351.98	0.990	309.87	3,300
	Proposal	6,950	2,600	3,902.73	1.457	460.78	3,850

3.9.4.3　経路計算時間

最後に，表 3.18 は，${}_N\mathrm{P}_2$ 個の拠点間トラヒックからなるデマンド集合に対する経路設計時間の平均値である．提案方式は，実験用トポロジと同じノード数であるトポロジ (A) の場合は，実験 3 と同様に僅かに劣るものの，トポロジ (C) つまりノード数が 8 を超えたところから提案方式の方が優れており，ノード数の増加と共に差が広がっていくのが分かる．以上より，提案方式はトポロジの規模が大きくなるにつれて，多項式時間で経路計算が可能なダイクストラ法よりも高速に経路設計できることが示された．

表 3.18 実験 2 評価用トポロジにおける平均計算時間 [s]

	Topology				
Method	**(A)**	**(B)**	**(C)**	**(D)**	**(E)**
Dijkstra	0.010	0.021	0.043	0.072	0.199
Proposal	0.014	0.021	0.029	0.053	0.079

3.9.4.3.1 実験 3 seq2seq 方式との性能比較評価

提案方式の有効性を示すために，2 章で述べた関連研究の中から文献 [13] の方式を選出した．文献 [13] は，逐次的に送受信経路全体を設計可能な方式であることから，提案方式の比較方式として適切だと考えたここで，文献 [13] では，Seq2Seq モデルに Attention と Beam Search を組み合わせた方式を用いており，以下この方式を Seq2Seq 方式と呼ぶ．比較評価条件は，経路設計成功率ならびに最大負荷リンクにおける使用帯域幅とする．加えて，経路設計成功率には評価項目 (5.1b) を用いる．実験には，最も小さなトポロジであるトポロジ (A) ならびに，1 番目，2 番目に大きなトポロジであるトポロジ (D) とトポロジ (E) を用いた．なお，Seq2Seq 方式は一つの拠点間トラヒックに対して送受信間経路を設計することから，本提案との条件を合わせるためにトラヒックデマンド内のすべての拠点間トラヒックに対して，繰り返し経路を設計させたときの結果を用いて比較評価を行う．

表 3.19 は提案方式と Seq2Seq 方式，それぞれの経路設計成功率である．提案方式はすべてのトポロジにおいて，安定して 99%以上の精度で経路設計が可能である．その一方で，Seq2Seq 方式は，トポロジサイズの増加とともに経路設計精度が低下している．これは評価項目 (5.1b) より，設計された経路のうち輻輳が発生している経路を除外しているためである．表 3.20 は提案方式と Seq2Seq 方式，それぞれの最大負荷リンクにおける最大値 (Max)，最小値 (Min)，平均値 (Average)，標準偏差 (SD:StandardDeviation)，標準誤差 (SE:StandardError)，中央値 (Median) を記載したものである．提案方式はすべてのトポロジにおいて，輻輳が発生していない一方で，Seq2Seq 方式は輻輳が発生してしまっている．これは，Seq2Seq 方式は教師データに，リンクコストをホップ数としたダイクストラ法による最短経路，および，送信元から制約ノード，制約ノードから宛先までの最短経路を結合し，ループフリーであるものを制限付き経路として学習している．特に，制限付き経路はホップ数が増加し，帯域幅を過剰に使用してしまう．その結果，頻出リンクの使用が過剰になり，輻輳の発生や平均負荷の増加が発生している．加えて，標準偏差からデマンド内のフローの組み合わせやランダムに選択される制限付き経路よって，最大負荷に大きなばらつきがある．これに対して，提案方式はリンクコストを残余帯域の逆数としたダイクストラ法を学習することで，負荷分散された経路を設計できていることから輻輳の発生を抑制し，平均負荷の低減，最大負荷のばらつきの低減を達成できていることが示されている．以上より提案方式は，本研究の目的であるネットワーク内の最大負荷を低減するような，負荷分散された経路を設計できることが示された．

表 3.19 実験 3 拠点間トラヒック毎の経路設計成功率 [%]

	Topology		
Method	**Topology(A)**	**Topology(D)**	**Topology(E)**
Seq2Seq	99.96	98.79	97.22
Proposal	99.97	99.96	99.52

表 3.20 実験 3 最大負荷リンクにおける使用帯域幅の詳細値 [Mbps]

		Traffic Volume of highest Load Link					
Topology	**Method**	**Max**	**Min**	**Average**	**SD**	**SE**	**Median**
Topology(A)	**Seq2Seq**	10,150	2,500	5,144.94	827.88	2.618	5,050
	Proposal	5,900	1,950	3,398.79	467.46	1.478	3,400
Topology(D)	**Seq2Seq**	12,700	3,600	6,188.67	945.42	2.990	6,050
	Proposal	7,700	2.800	5204.48	664.26	2.101	5,200
Topology(E)	**Seq2Seq**	14,250	4,350	7,262.41	1,075.93	3.402	7,150
	Proposal	6,950	2,600	3,902.73	460.78	1.457	3,850

3.9.4.3.2 実験 4 トポロジの変化に対する汎化性能

機械学習を用いた経路設計では，教師データ作成時のトポロジに依存した経路設計を行う．そのため文献 [13] では，リンク故障などの要因でトポロジが変化した場合に，設計される経路がどの程度汎化性能を持つかについて議論されている．実験 6 では，実験 4 で用意した学習済みの学習モデルを用いて，ネットワーク内でリンク故障が発生した場合に，経路設計成功率にどの程度影響を与えるかについて検証する．実験には，最も小さなトポロジであるトポロジ (A) ならびに，1 番目，2 番目に大きなトポロジであるトポロジ (D) とトポロジ (E) を用いた．実験におけるリンク故障の設定は，文献 [13] と同様にした．具体的には，ランダムに N 個のリンク故障が発生することを想定し，故障リンクを変更することで，最大 10 種類のリンク故障パターンを生成する．また，リンク故障数 N は最大で 10 個とし，テストデータは実験 4 で用いたものと全く同じデータセットを使用した．なお，評価条件である経路設計成功率には評価項目 (5.1b) を用いた．実験により，各リンク故障パターンに対して経路設計成功率を計測し，最大値 (Max)，最小値 (Min)，平均値 (Average) を算出する．ただし，リンク数や故障の組み合わせ数が 10 種類に満たない場合は，可能な限りすべての組み合わせを検証する．

図 3.35， 3.36， 3.37 は，トポロジ (A)，(D)，(E) における最大 10 個のリンク故障が発生した場合の経路設計成功率の変化を示している．先ず，図 3.35 より，トポロジ (A) では経路設計成功率の極端な劣化が起きている．これは，ノード数 6 に対して，リンク数が 7 であるため，リンク故障によって受ける影響が大きくなることが挙げられる．しかしながら，トポロジの特性を考慮した場合，妥当な結果であると言える．次に，図 3.36 より，トポロジ (D) ではリンク故障数の増加に対して段階的に経路設計成功率が低下している．トポロジ (D) はリンクの総数が 19 本であり，全体の 3 分の 1 に該当するリンク故障数 $N = 6$ のとき，平均経路設計成功率が約 50% であり，全体の半分に該当する最大故障数 $N = 10$ のとき，平均経路設計成功率が約 35% 程度であった．原因として，トポロジ (D) は各ノードの次数の差が大きいネットワークであることから，リンク故障の発生する箇所によっては，大きく経路設計成功率が低下する要因となり得る．その一方で，全体の半分のリンクが使用不可になった場合でも，平均で 30%以上の通信が行えることから，負荷分散によってリンクの偏りが少なくなっており，一定の頑健性を有していると言える．最後に，図 3.37 より，トポロジ (E) では，リンク故障数の増加に対して緩やかに経路設計成功率が低下している．最大リンク故障数 $N = 10$ のとき，平均経路設計成功が約 50% 程度であり，これまでのトポロジと比較してリンク故障に対する耐性が高い．要因として，トポロジ (E) は各ノードの次数の差が小さいため，負荷分散された経路の設計により，満遍なくリンクが使用されることで，リンク故障に対する頑健性を得ていると推測できる．これは，図 3.37 中の各リンク故障数 N に対して，最大と最小の経路設計成功率の差が，トポロジ (A)，(D) と比較して小さいことからも，特定の経路に依存せず，常に一転の頑健性を有していると言える．

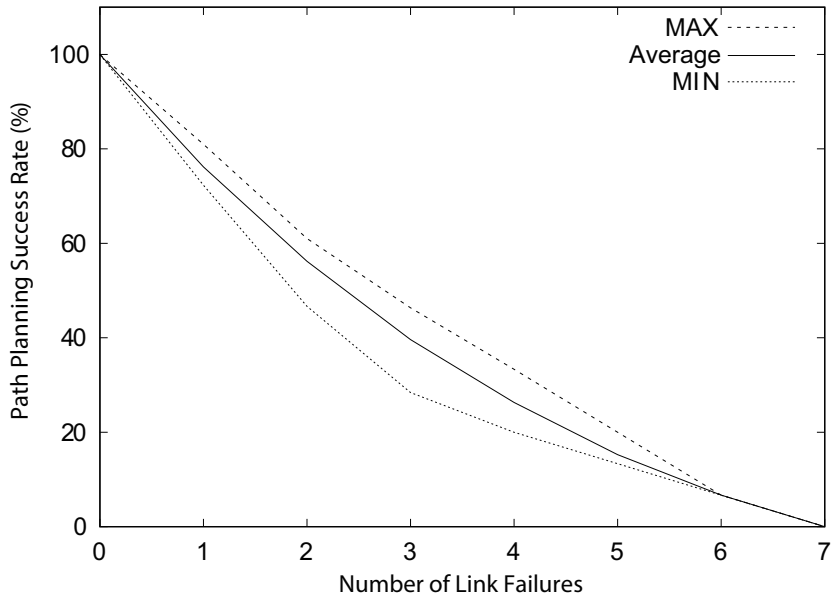


図 3.35 トポロジ (A) におけるリンク障害に伴う経路設計成功率

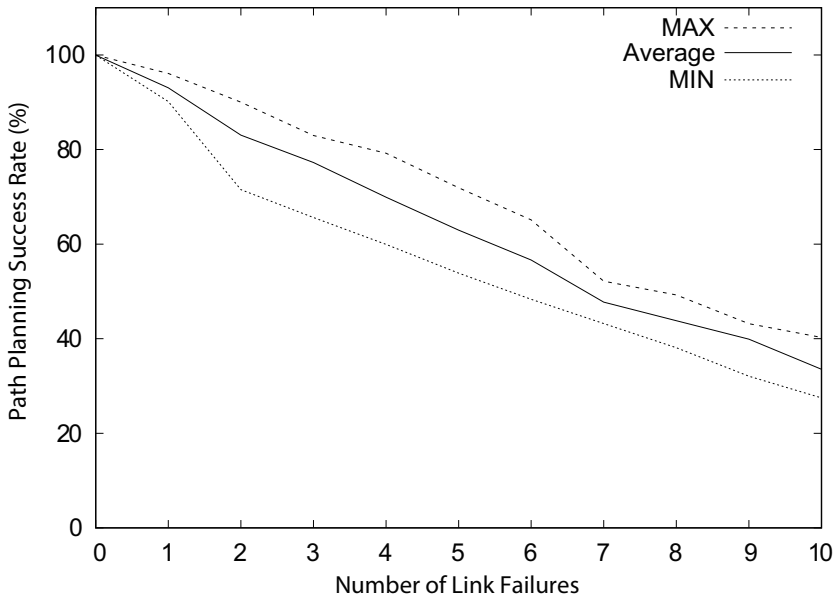


図 3.36 トポロジ (D) におけるリンク障害に伴う経路設計成功率

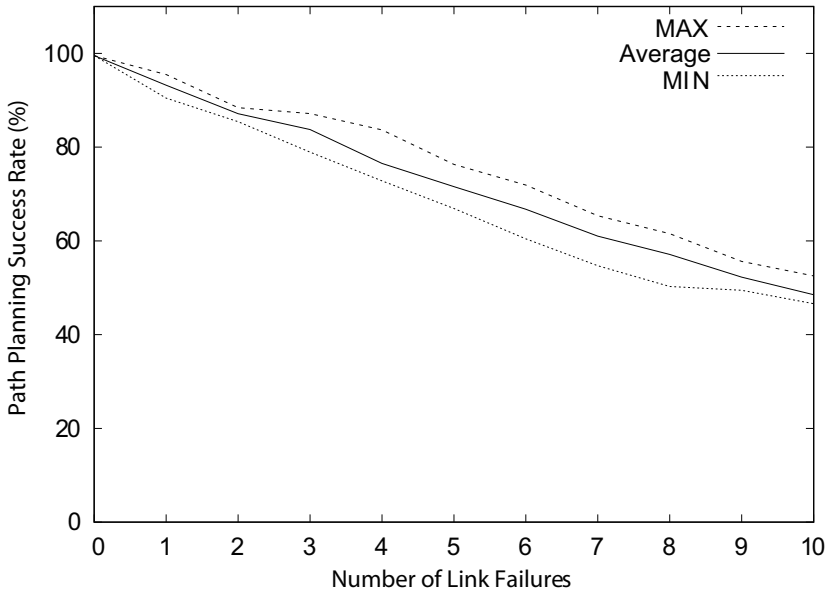


図 3.37 トポロジ (E) におけるリンク障害に伴う経路設計成功率

以上より，トポロジ (D) や (E) において，平均経路設計成功率が約 50%となるのはリンク故障数が 7 のときであった．また，提案方式による経路は，1 つや 2 つ程度の軽微なリンク故障により，直ちにネットワーク内すべての通信が行えなくなる様な事象は確認できない．よって，提案方式が設計する経路はリンク故障に対して，一定の頑健性を有していると考えられる．

3.9.5 深層学習モデルの考察

性能評価より, 提案方式の有効性は示された．特に，従来の機械学習を用いた経路設計では，複数の拠点間トラヒックからなるデマンド集合に対して，すべての送受信間経路を一度に計算することは精度の観点から不可能であり，実現されてこなかった．本研究では，教師データとの一致率も高い精度であったのに加えて，ネットワーク上に存在するリンクであり，送受信間経路として使用可能で，ループフリーであり，輻輳がないという条件で経路を評価したところ，非常に高い精度で正しい接続性を持つ経路であることが分かった．また，帯域使用率の観点から中規模程度のトポロジにおいて，リンクコストを残余帯域分の逆数としたダイクストラ法と同等レベルの輻輳制御が行えていることが分かった．加えて，従来の機械学習を用いた送受信間経路設計法よりも優れた経路を設計できることを比較評価により示した．次に，経路計算時間の観点では多項式時間で経路設計が可能なダイクストラ法と比較して，小規模トポロジの場合は同等程度であり，中規模トポロジになるにつれて，より短い時間で計算できることが分かった．本実験で評価したトポロジにおいては，十分にミリ秒オーダの計算時間であると言える．最後に，これらの経路はリンク故

障などのトポロジの変化に対しても，一定の頑健性を有していると考えられる．この結果から，教師データを作成する際の経路設計法をより優れた方式を用いることで更なる性能の向上が見込まれる．例えば，線形計画法を用いた経路設計法では，本来非常に長い計算時間を要し，最悪の場合経路が計算できない場合がある．これに対して，提案方式を用いることで線形計画法によって求められる優れた経路を，ダイクストラ法と同等以下の速度で設計できる可能性がある．

また，本研究ではもう一つ優れた結果を得ている．それは，経路設計に成功したデマンド集合において，すべての拠点間トラヒックを流した場合に輻輳するようなテストデータは確認されなかったことである．加えて，ダイクストラ法を用いて経路計算した際に輻輳が発生するような拠点間トラヒックの組み合わせであるデマンド集合を，学習モデルに入力した場合，経路設計成功率はほぼ 0%に近い値で，経路として成り立たないことが分かった．これら二点より，深層学習を用いた経路設計法では，デマンド集合毎の経路設計成功率の値のみで輻輳制御が可能がどうか判別することができると考えられる．なぜならば，学習モデルの経路設計結果から，高い成功率の場合は輻輳制御が行えるようなデマンド集合であり，低い成功率の場合は輻輳制御が行えないようなデマンド集合であることを判別することができるためである．

3.9.6　今後の課題

今後の課題は大きく三つある．第一に，提案方式が適用可能なトポロジ規模の拡大．これは，より大きな規模のトポロジにおける評価実験が必要である．第二に，より優れた負荷分散を考慮した経路設計法の学習．これは，現在の教師である，リンクコストを残余帯域の逆数としたダイクストラ法よりも優れた経路が設計できる方式を用いて新しい教師データを作成し，学習させる必要がある．第三に，教師データの変更における経路計算時間の維持．教師データを変更してもミリ秒オーダの計算時間を維持できるかどうかについても確認する必要がある．

3.10 むすび

本研究では，トラヒックエンジニアリングにおける経路設計法の持つ経路計算における課題に対して機械学習技術を適用する．そこで，複数の拠点間トラヒックを考慮しつつ，輻輳制御された経路を出力できるような深層学習モデルを検討した．

提案方式では，最も基本的な輻輳制御法であるリンクコストを残余帯域の逆数としたダイクストラ法を用いて教師データを作成し，深層学習モデルに学習させた．評価実験より，小中規模のトポロジにおいて非常に高い精度で，輻輳制御された経路をダイクストラ法と同等以上の速さで設計できることが分かった．また，従来の機械学習を用いた送受信間経路設計法よりも，優れた負荷分散を達成している．特に，提案方式はトポロジの規模が大きくなるにつれて，ダイクストラ法よりも短い時間で経路を計算できることを示した．加えて，リンク故障などを起因とするトポロジの変化に対しても，一定の頑健性を有していることを示した．以上の結果より，本提案は，トラヒックデマンドの変化がある度に，瞬時にすべてのトラヒックに対してネットワーク全体の負荷分散を考慮した経路設計を行うことができる．

その一方で，負荷分散の観点でより優れた経路設計法を学習させることで，提案方式の性能の向上を図りつつ，計算時間の課題を解決できるかについて検証することが必要である．第 4 章では，この課題を解決するために線形計画法を用いた経路設計法により教師データを生成し，学習させることで従来の提案方式よりも優れた方式を提案する．

第4章　輻輳制御のための整数線形計画法を用いた深層学習による経路設計法

4.1　整数線形計画法を用いた教師データの検討

本研究では，機械学習を用いた経路設計法の課題の中でも，教師データに関する課題を取り上げる．特に，機械学習モデルの性能は教師データを作成する際に用いる経路設計法の性能に大きく依存する．

これまでのシミュレーション実験で用いてきた教師データには，トラヒックエンジニアリングにおける最も基礎的な輻輳制御のための経路設計法である，リンクコストを残余帯域の逆数としたダイクストラ法により生成している．しかしながら，トラヒックエンジニアリングに関する研究では，より優れた輻輳制御を達成するために，線形計画法を用いて入力されたトラヒック全体に対する経路の最適化を行う方式が提案されている [2] [3] [4]．その一方で，既存のトラヒックエンジニアリングにおける線形計画法を用いた経路設計法は，教師データを作成することを目的としておらず，線形計画法の入力であるトラヒックパターンも想定する環境や前提が異なるため，そのまま適用することができない．そこで，機械学習のための線形計画法を用いた，教師データの生成法を検討する必要がある．

4.1.1　本章の目標

本章における提案方式の目標は三つある．一つ目は，高い経路設計成功率である．これは，線形計画法を学習しても従来の提案方式と同等成功率で経路設計が行えるか検証する必要がある．二つ目は，負荷分散を考慮した線形計画法の特徴を学習することである．これは，従来の提案方式よりも優れた負荷分散された経路を学習モデルが設計する事ができるか検証する必要がある．三つめは，深層学習を用いた経路設計法の計算時間である．これは，教師データを作成するための経路設計法が変更されても，従来の提案方式と同等の計算時間であるか検証する必要がある．これらの目標を解決するためには大きく二つの課題がある．

4.1.2　本章の課題

一つ目の課題は線形計画法を用いた経路設計法の検討と性能評価である．先ず，線形計画法を用いた教師データの生成法について検討し，この教師データを用いて実際にどの程度の性能を有することができるか性能評価により確認する．特に，ダイクストラ法と比較してどの程度性能が向上しているか確認する必要がある．二つ目の課題は線形計画法を用いた教師データの作成と性能評価である．教師データは第 3 章の訓練データ，正解データの設計に基づいて作成される．性能評価では，教師データの変更に寄らず性能が維持できるか検証する必要がある．特に，ダイクストラ法を学習した深層学習モデルよりも優れているか，経路設計成功率，負荷分散，経路計算時間の観点で評価を行う．

4.1.3　線形計画法を用いた教師データの生成法

本研究では，入力されたすべての拠点間トラヒックに対して，すべてのリンクに輻輳が無いかつ，最大負荷リンクの帯域使用率を最小化するような経路を線形計画法を用いて探索する．よって，帯域使用率の取りうる範囲は 0 から 1 の範囲である．すべての拠点間トラヒックの経路を割り当てた際の最大負荷リンクの帯域使用率が最小となるような経路を探索させる．

[定数の定義]

N :　ノード集合
$d_{s,d}$:　ノード $s \in N$ とノード $d \in N$ 間の要求帯域幅
$Lc_{i,j}$:　ノード $i \in N$ とノード $j \in N$ 間の帯域幅
α, β :　CMAX と $X_{i,j}^{s,d}$ の優位を決定する定数

[変数の定義]

$X_{i,j}^{s,d}$:　ノード $s \in N$, $d \in N$ 間におけるリンク $i \in N$, $j \in N$ の状態を表すバイナリ変数であり，使用する場合は 1，使用しない場合は 0
CMAX :　最大負荷リンクの帯域使用率

[目的関数]

$$min\ \alpha \cdot CMAX + \beta \cdot \sum_{s,d,i,j \in N} X_{i,j}^{s,d} \tag{4.1}$$

[制約式]

$$\sum_{s,d \in N} X_{i,j}^{s,d} \cdot d_{s,d} \leq CMAX \cdot Lc_{i,j} \qquad i,j \in N \tag{4.2}$$

$$\sum_{k \in N} X_{i,k}^{s,d} - \sum_{k \in N} X_{k,i}^{s,d} = 1 \qquad i = s \qquad s,d,i \in N \tag{4.3}$$

$$\sum_{k \in N} X_{i,k}^{s,d} - \sum_{k \in N} X_{k,i}^{s,d} = -1 \qquad i = d \qquad s,d,i \in N \tag{4.4}$$

$$\sum_{k \in N} X_{i,k}^{s,d} - \sum_{k \in N} X_{k,i}^{s,d} = 0 \qquad i \neq s,d \qquad s,d,i \in N \tag{4.5}$$

$$0 < CMAX \leq 1 \tag{4.6}$$

先ず，式 (1) は最大負荷リンクにおける帯域使用率と，合計ホップ数を足し合わせた値の最小化を目的とする関数である．次に，式 (2) は帯域幅に関する制約であり，各リンクにおいて通過する拠点間トラヒックの要求帯域幅の総量が，帯域幅を超えないための式である．続いて，式 (3),(4),(5) は送信元，宛先，中継ノードにおけるフロー保存則に関する式である．式 (3),(4) は送信元ノード，宛先ノードはそれぞれトラヒックを出力する，入力されるという関係を表しており，式 (5) は中継ノードは必ず入力されたトラヒックを出力するという関係を表している．最後に，式 (6) は最大負荷リンクにおける帯域使用率の取りうる範囲を定めた式である．

なお，帯域使用率とルーティング変数は，異なる桁を持つ変数であるため，α と β という定数によって補正する必要がある．ただし原則として，$\alpha < \beta$ となるように値を設定する．また，β は α に比べて十分に小さい値とする．

4.1.4　線形計画法を用いた経路の性能評価

4.1.4.1　評価条件

先ず，評価実験で用いるネットワークトポロジは，中規模トポロジである 11 ノード 26 リンクからなる，COST239(図 4.1) [25] を用いて，各ノード間の帯域幅はすべて 10Gbps で統一する．

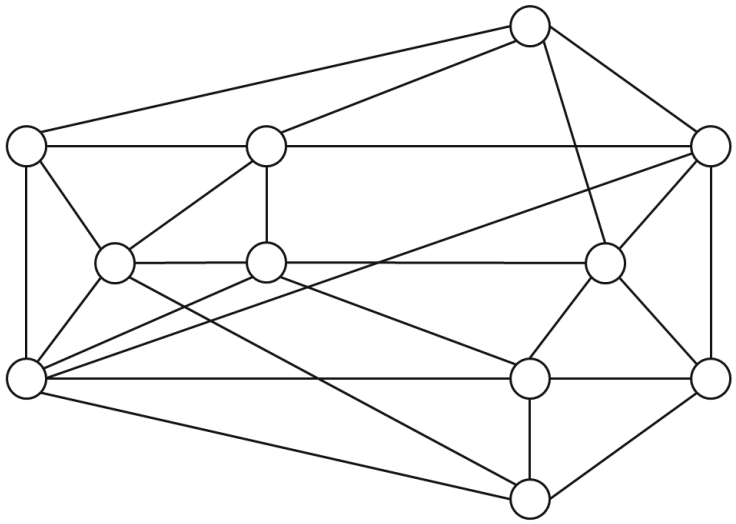


図 4.1 Cost239 トポロジ

次に，各経路計算の実行環境は以下の通りである．ダイクストラ法の計算は，OS:Ubuntu 20.04.1, CPU:Intel Core i9 10900K 3.7Ghz, 物理メモリ:128GB, プログラミング言語:Python 3.8.5 を用いた．また，数理計画法の計算には，OS:Ubuntu 21.10.0, CPU:Intel Core i7 12700K 3.6Ghz, 物理メモリ:128GB, プログラミング言語:Python 3.9.7, 数理計画法ソルバ:Gurobi Optimizer 9.5.1 を用いた．

実験で用いるデータセットは以下のように生成する．先ず，データセットの構造をトポロジ内のすべてのノードが互いに一つずつトラヒックを発生させた場合を想定し，${}_N\mathrm{P}_2$ 個の拠点間トラヒックからなるトラヒックデマンドを入力とすると定める．そのうえで，ダイクストラ法と線形計画法による経路を比較評価するために，次の二種類のデータセットを生成する．一つは，ダイクストラ法を用いて経路を計算させた場合に最大負荷リンクにおいて輻輳が発生していないようなデータセットであり，計 5 セット．また，同様の条件で輻輳が発生しているデータセットを計 7 セット用意する．このとき，すべてのデータセットは全く異なるデータであることを確認し，これらの計 12 セットのデータを，線形計画法を用いて再計算させる．

最後に評価条件は，最大負荷リンクにおける使用帯域幅と，トポロジ内すべてのリンクにおける使用帯域幅の平均値の観点から評価を行う．

4.1.4.2 線形計画法を用いた経路の負荷分散性能評価

図 4.2，図 4.3 は，ダイクストラ法によって輻輳が発生していないデータセットにおける結果を表したものである．先ず，図 4.2 は，最大負荷リンクにおける使用帯域幅について表したものである．データ (1) はダイクストラ法が 8000Mbps(ノード 2,5 間), 線形計画法が 4900Mbps(ノード 0,1 間) で 3100Mbps の低減．データ (2) はダイクストラ法が 8000Mbps(ノード 0,1 間), 線形計画法が 5000Mbps(ノード 0,1 間) で 3000Mbps の低減．データ (3) はダイクストラ法が 8000Mbps(ノード 0,1 間), 線形計画法が 6000Mbps(ノード 1,4 間) で 2000Mbps の低減．データ (4) はダイクストラ法が 8000Mbps(ノード 4,10 間), 線形計画法が 6900Mbps(ノード 6,0 間) で 1100Mbps の低減．データ (5) はダイクストラ法が 10000Mbps(ノード 0,3 間), 線形計画法が 5350Mbps(ノード 0,2 間) で 4650Mbps の低減であった．

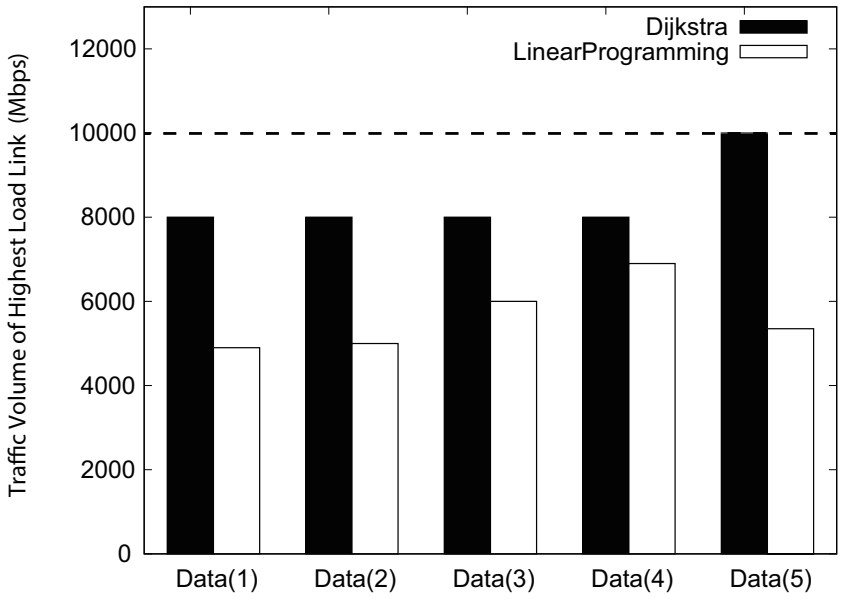


図 4.2 最大負荷リンクにおける使用帯域幅 (輻輳なし)

次に，図 4.3 は，すべてのリンクにおける使用帯域幅の平均値について表したものである．ただし平均値は，すべて小数点以下の四捨五入を行った値である．データ (1) はダイクストラ法が 6408Mbps, 線形計画法が 6109Mbps で 299Mbps の低減．データ (2) はダイクストラ法が 6456Mbps, 線形計画法が 6133Mbps で 323Mbps の低減．データ (3) はダイクストラ法が 5672Mbps, 線形計画法が 5600Mbps で 72Mbps の低減．データ (4) はダイクストラ法が 5484Mbps, 線形計画法が 5184Mbps で 300Mbps の低減．データ (5) はダイクストラ法が 6170Mbps, 線形計画法が 5703Mbps で 467Mbps の低減であった．以上より，線形計画法によって求めた経路が，最大負荷を低減させることに成功しており，かつネットワーク全体の平均負荷も低減していることから，ダイクストラ法と比較して効率的な経路を設計できることが示された．

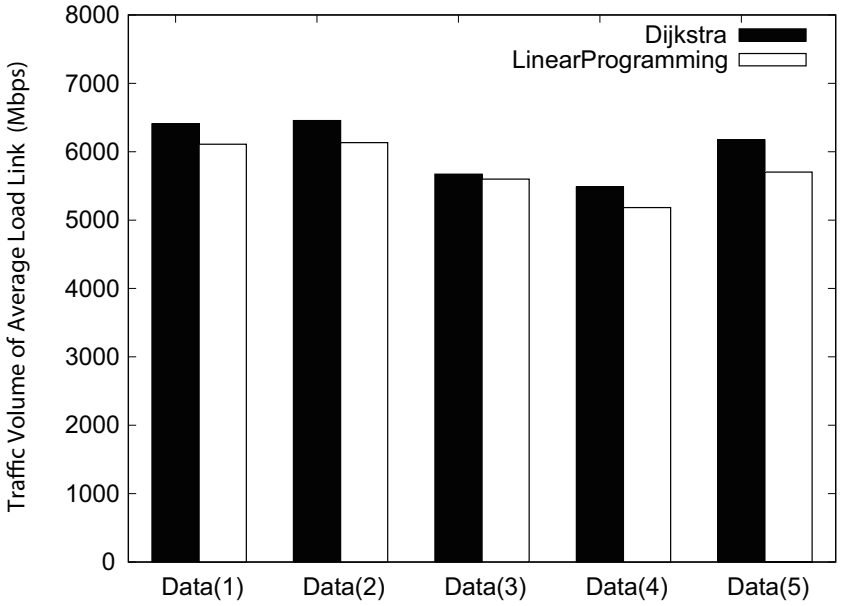


図 4.3 すべてのリンクにおける使用帯域幅の平均値 (輻輳なし)

図 4.4，図 4.5 は，ダイクストラ法によって輻輳が発生しているデータセットにおける結果を表したものである．先ず，図 4.4 は，最大負荷リンクにおける使用帯域幅について表したものである．データ (1) はダイクストラ法が 12000Mbps(ノード 6,0 間), 線形計画法が 5800Mbps(ノード 0,6 間) で 6200Mbps の低減．データ (2) はダイクストラ法が 12000Mbps(ノード 0,6 間), 線形計画法が 6000Mbps(ノード 0,2 間) で 6000Mbps の低減．データ (3) はダイクストラ法が 10750Mbps(ノード 10,7 間), 線形計画法が 8000Mbps(ノード 8,5 間) で 2750Mbps の低減．データ (4) はダイクストラ法が 12000Mbps(ノード 6,5 間), 線形計画法が 6350Mbps(ノード 7,3 間) で 5650Mbps の低減．データ (5) はダイクストラ法が 10450Mbps(ノード 6,0 間), 線形計画法が 8000Mbps(ノード 3,0 間) で 2450Mbps の低減．データ (6) はダイクストラ法が 11000Mbps(ノード 2,3 間), 線形計画法が 6250Mbps(ノード 0,1 間) で 4750Mbps の低減．データ (7) はダイクストラ法が 10750Mbps(ノード 10,4 間), 線形計画法が 8000Mbps(ノード 1,4 間) で 2750Mbps の低減であった．

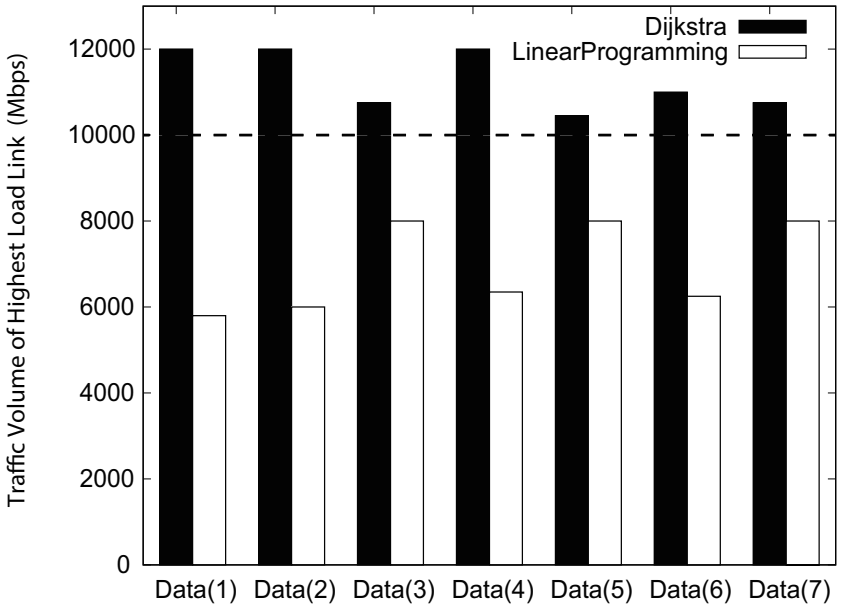


図 4.4 最大負荷リンクにおける使用帯域幅 (輻輳あり)

次に，図 4.5 は，すべてのリンクにおける使用帯域幅の平均値について表したものである．ただし平均値は，すべて小数点以下の四捨五入を行った値である．データ (1) はダイクストラ法が 6213Mbps, 線形計画法が 5581Mbps で 632Mbps の低減．データ (2) はダイクストラ法が 6206Mbps, 線形計画法が 5887Mbps で 319Mbps の低減．データ (3) はダイクストラ法が 5317Mbps, 線形計画法が 5156Mbps で 162Mbps の低減．データ (4) はダイクストラ法が 5153Mbps, 線形計画法が 4639Mbps で 513Mbps の低減．データ (5) はダイクストラ法が 5325Mbps, 線形計画法が 5219Mbps で 106Mbps の低減．データ (6) はダイクストラ法が 5315Mbps, 線形計画法が 4643Mbps で 672Mbps の低減．データ (7) はダイクストラ法が 4986Mbps, 線形計画法が 4928Mbps で 58Mbps の低減であった．以上より，線形計画法によって求めた経路が，輻輳を解消しており，かつネットワーク全体の平均負荷も低減していることから，ダイクストラ法と比較して教師データとして使用可能なトラヒックパターンが増えていることが示された．

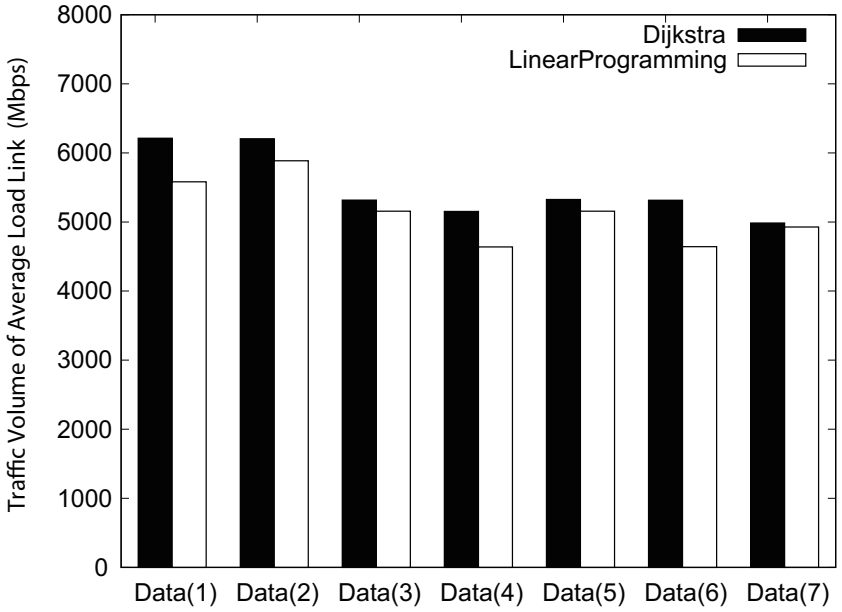


図 4.5 すべてのリンクにおける使用帯域幅の平均値 (輻輳あり)

よって線形計画法を用いて，最大負荷リンクの使用帯域幅を低減しつつ，教師データとして使用できるトラヒックパターンを増やすことができることから，提案方式の有効性が示された．

4.2 線形計画法を用いた深層学習モデルの性能評価実験

4.2.1 実験に用いる深層学習モデルの構成

本章で用いる深層学習モデルの構成は第 3 章にて，ダイクストラ法を学習させたモデルと全く同じ構成を持つ．

図 4.6 は，深層学習モデルの構成例を示している．提案する深層学習モデルの中間層は，複数のブロックから構成され，一つのブロックは入力層と同じ数のユニットを持つ．また，中間層は深さが 3 層以上で，それぞれの深さで複数のブロックを持つ．このとき，一つの深さにつき ${}_N\mathrm{P}_2$ 個のブロックを持つ．加えて，ブロックは各拠点間トラヒック毎に連結関係を持つ．なお，一般的に深層学習における中間層は深さが 3 層以上であり，図 4.6 は中間層が 4 層の場合を示している．

入力層から中間層への結合は，深さが 1 であるすべての中間層のユニットに対して全結合させる．また，ブロックは同一拠点間トラヒックに関する前段と後段のユニットに対して全結合させる．このとき，異なる拠点間トラヒックに関する中間層ブロック同士は一切の結合関係を持たない．そして，中間層から出力層は，各拠点間トラヒックと対応しているブロックの最深から，該当するトラヒックの出力層とのみ全結合させる．ただし，出力層は一つの拠点間トラヒックに対して N 個の one-hot ベクトルから構成される．

従来の機械学習を用いた経路設計法との大きな違いは，単一の中間層にすべての経路情報を学習させ，経路設計を行わせるのではなく，中間層と出力層のブロックが，一つの拠点間トラヒックに対する経路情報を学習し，経路設計を行うことである．これにより，単一の中間層を用いて多数の正解データを学習させた場合に，学習が収束せずに十分な性能を得られないという課題を解決することができる．この課題は，一度に膨大な出力を計算させてしまうことで，出力パターンが非常に複雑化することで，特徴量を十分に学習することができないことが最大の原因である．特に経路設計の場合は，類似している訓練データに対して，全く異なる正解データが複数存在することもあり，学習が著しく阻害される要因となっている．提案方式では，中間層と出力層をブロック化し学習させる範囲を一つの拠点間トラヒックにおける経路情報に限定することで，学習精度の向上を果たしている．

その一方で，中間層と出力層だけではなく，入力層も一つの拠点間トラヒックのみを入力とする方法も挙げられる．この場合，学習モデルの規模が縮小することで，性能が限られたデバイスでも機械学習を用いた経路設計が行えると考えられる．しかし，この方法では出力パターン数が減少し学習が収束しやすくなる一方で，全く同じ要素からなる訓練データに対して，複数の正解データを学習することができないという問題がある．つまり，本研究が目的とするネットワーク状況に応じた動的な経路設計が行えない．また，学習に必要な情報量を増やす目的で，訓練データにトラヒックの受け付け順やネットワークの残余帯域などの情報を組み込むこと方法も挙げられる．しかし，この方法では訓練データの種

類が，正解データの種類よりも大幅に増えてしまうことで，入力パターンの複雑化が増し，特徴量を十分に学習することができないという問題がある．以上より，訓練データをトラヒック単位とした場合は，いずれの手段においても学習自体が成り立たず経路設計を行えるような性能を持つ学習モデルを生成することができない．提案方式の場合は，訓練データをデマンド集合とすることで，デマンド集合に含まれる他の送信元，宛先ノード番号や要求帯域幅の情報により，残余帯域などの情報を用いずとも，デマンド集合全体を総合的に収容するような経路を学習できる．

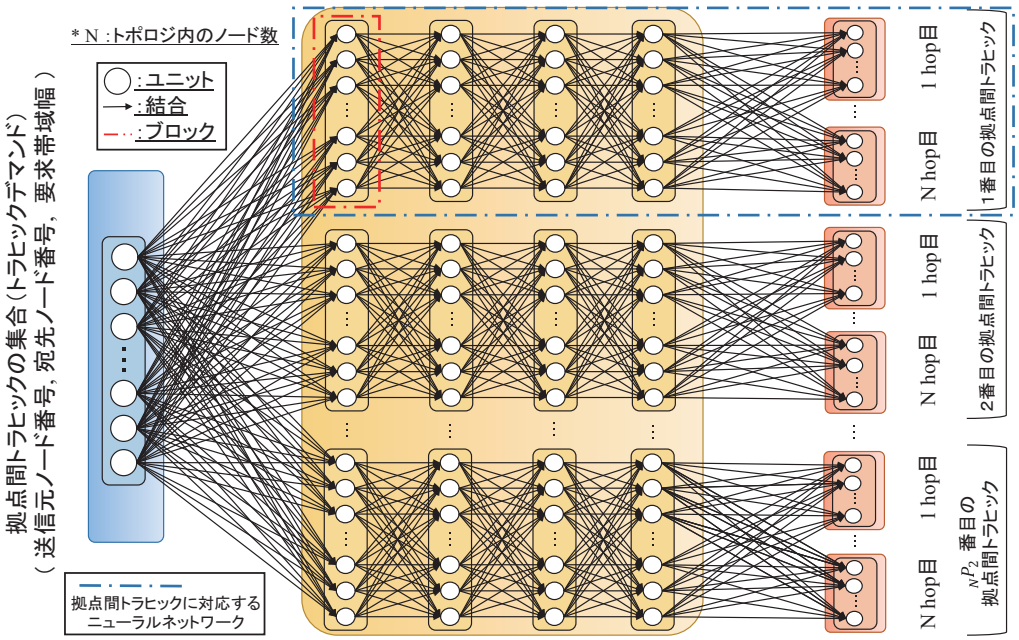


図 4.6 深層学習モデルの構成例

4.2.2　評価条件

線形計画法を学習した深層学習モデルの評価を行うために，実験用トポロジを用いて実験を行う．実験評価項目は，経路設計成功率，使用帯域幅，経路計算時間の三点である．なお，比較対象方式は，リンクコストを残余帯域の逆数とした動的なダイクストラ法，ネットワークの負荷分散を考慮した線形計画法，ダイクストラ法を学習した提案方式，文献 [13] にて提案されている seq2seq 学習モデルを用いた経路設計法の四種類である．ここで，経路設計成功率は次の式から求められる．

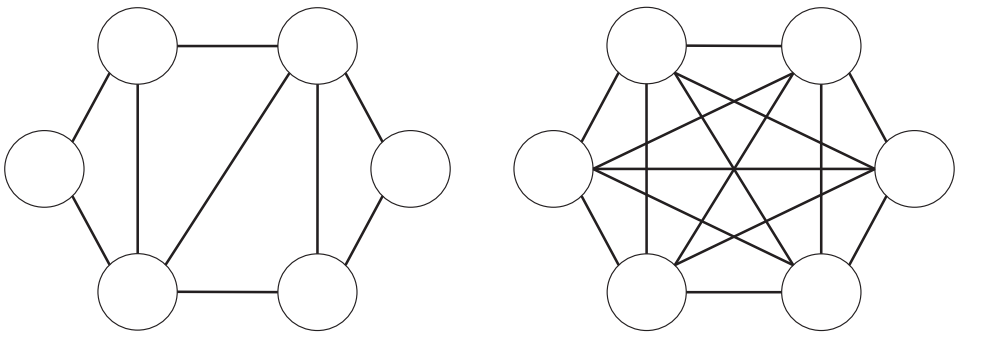
$$\frac{\text{ダイクストラ法による経路と一致している送受信間経路の数}}{{}_N\mathrm{P}_2 \times \text{データセット数}} \tag{4.7a}$$

$$\frac{\text{本研究における成功条件を満たした送受信間経路の数}}{{}_N\mathrm{P}_2 \times \text{データセット数}} \tag{4.7b}$$

評価項目 (5.1a) は，ダイクストラ法による経路との一致率であり，一般的な機械学習における正答率に該当する．評価項目 (5.1b) は，本研究における正答率の計算法である．このとき成功条件を満たす経路とは，学習モデルの出力経路において，使用するリンクがネットワーク上に存在し，送信元から宛先まで到達可能であり，ループフリーであり，かつどのリンクにも輻輳が発生していない経路である．次に，帯域使用率は最大負荷リンクにおける使用帯域幅の各値から求める．本研究では，教師データ作成のための経路設計法として，リンクコストを残余帯域の逆数としたダイクストラ法を用いており，輻輳制御を目的としている．そこで，学習モデルが教師データの特性を正しく学習できているかを，帯域使用率の観点からダイクストラ法と比較評価を行う．最後に計算時間は，提案方式とダイクストラ法それぞれを用いて，１つのトラヒックデマンド集合の経路をすべて計算するまでにかかる時間を 10 回計測し，その平均値により比較評価を行う．ただし，計測時間はデータの前処理や後処理を含まず，経路計算のみにかかる時間を対象とする．

なお，一般的に機械学習の分野では，教師データとして作成したデータセットを学習用のデータと評価用のデータに分割し，学習に用いたデータでは評価を行わない．よって，教師データの一部を分割し，学習と評価にそれぞれ用いる方法ではなく，教師データとは別にデータセットを生成し評価することで，学習モデルが汎用性のある学習を行っているか検討する．また本研究では，難易度の低減を図るために 150Mbps から 900Mbps までの 150Mbps 刻みの 6 種類に，1Gbps を合わせた 7 種類からランダムで選択する．この値は，KDDI の帯域保障サービスにおける選択可能な帯域幅を参考にした [24].

実験に用いるトポロジは以下の通りである．実験用トポロジは複数の 6 ノードトポロジ (図 4.7) である．これは，実験的な意味合いで用いられる小規模ネットワークトポロジ [22] [23] である．なお，各トポロジ内のすべての帯域幅は 10Gbps で統一する．深層学習モデルの実装と実験は，OS:Ubuntu 20.04.1， 物理メモリ:128GB， GPU:GeForce RTX 3090， プログラミング言語:Python 3.8.5， Tensorflow2.3.1+Keras を用いて行った．



(a) 6nodes - 9links mesh network topology　(b) 6nodes - 15links full mesh network topology

図 4.7 実験用トポロジ

4.2.3　実験用トポロジを用いた性能評価

4.2.3.1　経路設計成功率

図 4.8 は，実験用トポロジにおける経路設計成功率である．実験用トポロジは，6 ノードから構成されているため，すべてのノードが 1 回ずつ送受信を行うことを想定した場合，100,000 件のデータセットに含まれる拠点間トラヒックの総数は 3,000,000 である．これは ${}_6\mathrm{P}_2 \times 100,000$ から計算することできる．トポロジ (a) において，ダイクストラ法を学習した提案方式の経路設計成功率は 99.99% (2,999,970/3,000,000 トラヒック)，線形計画法を学習した提案方式の経路設計成功率は 99.99% (2,999,506/3,000,000 トラヒック) であった．トポロジ (b) において，ダイクストラ法を学習した提案方式の経路設計成功率は 99.99% (2,999,989/3,000,000 トラヒック)，線形計画法を学習した提案方式の経路設計成功率は 99.99% (2,999,970/3,000,000 トラヒック) であった．

以上の結果より，線形計画法を学習した場合であっても，従来のダイクストラ法を学習した場合と遜色ない経路設計成功率であることが分かった．よって，線形計画法を学習した深層学習モデルを用いることで，線形計画法の経路設計に関する動的な性質を学習できていると考えられる．

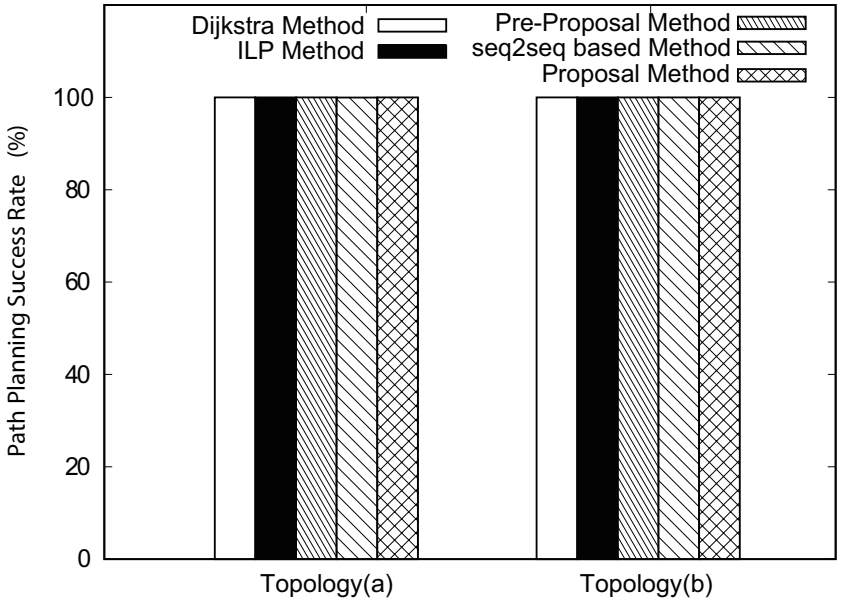


図 4.8 拠点間トラヒック毎の経路設計成功率

4.2.3.2 帯域幅使用率

図 4.9 は，すべての拠点間トラヒックを流した際の最大負荷リンクにおける使用帯域幅の平均値を比較したグラフであり，表 4.1 は，最大負荷リンクにおける最大値 (Max)，最小値 (Min)，平均値 (Average)，標準偏差 (SD:StandardDeviation)，標準誤差 (SE:StandardError)，中央値 (Median) を記載したものである．ただし，提案方式はデマンド集合に含まれるすべてのトラヒックの経路が設計できている場合にのみ計算する．これは，経路設計に失敗している場合，本来流れるべきトラヒックが流れず計算できないためである．

図 4.9 および，表 4.1 より，最も重要な結果はトポロジ (a) における使用帯域幅である．線形計画法を学習することで，リンクコストを残余帯域の逆数としたダイクストラ法やダイクストラ法を学習した深層学習モデルと比較して最大負荷リンクにおける使用帯域幅を低減することができた．その一方で，教師データである線形計画法と比較すると，最大負荷リンクにおける使用帯域幅が増加している．よって，線形計画法の学習には余地が残されており，更なるモデルの最適化や提案方式の改良などが必要である．また，トポロジ (b) は動的ダイクストラ法，線形計画法，ダイクストラ法を学習した深層学習モデル，線形計画法を学習した深層学習モデル，いずれも同じ値であった．これは，トポロジ (b) がフルメッシュトポロジであり，最適解がダイクストラ法と線形計画法で全く同じであり，それぞれを正しく機械学習モデルが学習できているため，4 つのモデルが全て同じ結果を出力したと考えられる．加えて，関連研究である seq2seq 方式はトポロジ (a) とトポロジ (b) ともに他の方式と比較して明らかに劣る結果となっている．これは，seq2seq 方式は最短経路を学習しているため，負荷分散を考慮した経路設計が行えないためである．以上より，深層学習モデルは線形計画法の経路設計に関する特徴を一定量学習することができると考えられる．

加えて，標準誤差を用いて 99%信頼区間を求めた．先ず，トポロジ (a) のとき，ダイクストラ法では 2,586.68 $\pm$ 2.778，線形計画法では 2,092.21 $\pm$ 1.994，ダイクストラ法を学習した学習モデルでは 2,892.12 $\pm$ 3.429，線形計画法を学習した学習モデルでは 2,860.42 $\pm$ 3.140，seq2seq 学習モデルでは 4,096.26 $\pm$ 6.432．トポロジ (b) のとき，ダイクストラ法では 998.99 $\pm$ 0.082，線形計画法では 998.99 $\pm$ 0.082，ダイクストラ法を学習した学習モデルでは 998.99 $\pm$ 0.082，線形計画法を学習した学習モデルでは 998.99 $\pm$ 0.082，seq2seq 学習モデルでは 4,097.44 $\pm$ 6.448．標準誤差は非常に小さい値であることから，想定環境において性能が大幅に低下するような場面は存在しない．以上より，学習モデルは輻輳制御を目的とした線形計画法が持つリンク選択の動的性質を学習できることが示された．

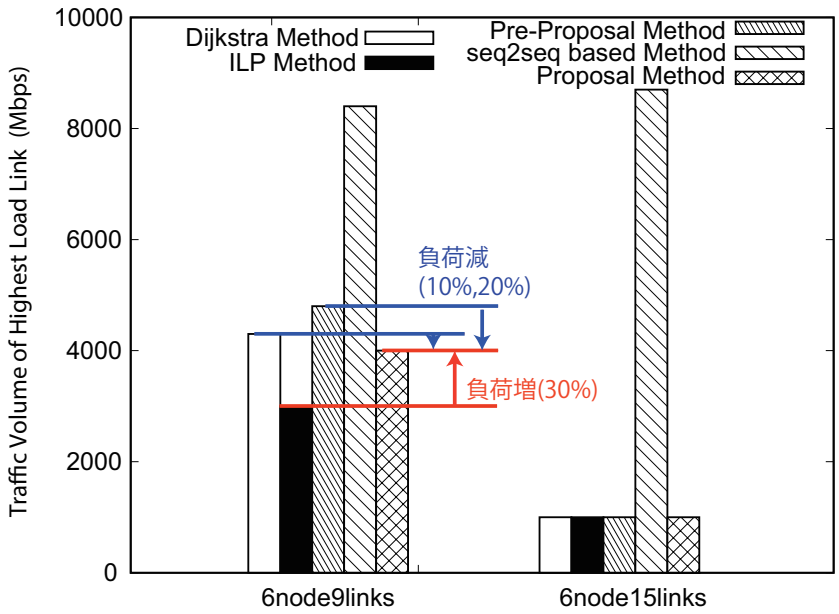


図 4.9 最大負荷リンクにおける使用帯域幅の平均値

表 4.1 最大負荷リンクにおける使用帯域幅の詳細値 [Mbps]

		Traffic Volume of highest Load Link					
Topology	**Method**	**Max**	**Min**	**Average**	**SD**	**SE**	**Median**
Topology(a)	**Dijkstra**	4,300	1,350	2,586.68	341.08	1.079	2,550
	ILP	3,000	1,200	2,092.21	244.66	0.774	2,100
	Pre-Proposal	4,800	1,500	2,892.12	420.99	1.331	2,850
	seq2seq	8,400	1,800	4,096.26	789.67	2.497	4,000
	Proposal	4,000	1,500	2,860.42	385.63	1.219	2,850
Topology(b)	**Dijkstra**	1,000	750	998.99	10.05	0.032	1,000
	ILP	1,000	750	998.99	10.05	0.032	1,000
	Pre-Proposal	1,000	750	998.99	10.05	0.032	1,000
	seq2seq	8,700	1,950	4,097.44	791.46	2.503	4,000
	Proposal	1,000	750	998.99	10.05	0.032	1,000

4.2.3.3 経路計算時間

最後に，表 4.2 は，${}_N\mathrm{P}_2$ 個の拠点間トラヒックからなるデマンド集合に対する経路設計時間の平均値である．先ず，ダイクストラ法を学習した学習モデルと線形計画法を学習した学習モデルの計算時間に着目すると，計算時間は同じ値であった．これは，ニューラルネットワークを用いた経路設計では，計算時間はモデルの構成や規模に依存していることから，同一のトポロジに対しては，教師データの性能や性質は影響を与えないためである．次に，線形計画法の経路計算時間に着目すると，非常に長い計算時間が必要である．具体的には，学習モデルを用いた経路設計法と比較すると，トポロジ (a)，トポロジ (b) ともに約 8 倍の計算時間を要する．以上より，線形計画法を学習した学習モデルを用いることで，ダイクストラ法よりも優れた経路をダイクストラ法と同等の速度で設計が可能であり，線形計画法よりも大幅に短い時間で経路設計が可能であることが示された．

表 4.2 実験用トポロジにおける平均計算時間 [s]

	Topology	
Method	**(a)**	**(b)**
Dijkstra	0.010	0.009
ILP	0.078	0.055
Pre-Prpposal	0.014	0.014
Prpposal	0.014	0.014

4.2.4 深層学習モデルの考察

性能評価より, 提案方式の有効性は示された．ダイクストラ法を学習した深層学習を用いた経路設計では，非常に高い成功率で経路を設計可能であるものの，負荷分散の観点からはダイクストラ法と同等であり，改善の余地がある．本研究では，負荷分散の考慮した線形計画法による経路を学習することで，従来と同等の経路設計成功率で経路が設計できることが分かった．また，帯域使用率の観点から実験用の小規模トポロジにおいて，リンクコストを残余帯域分の逆数としたダイクストラ法やダイクストラ法を学習した深層学習を用いた経路設計法よりも優れた輻輳制御が行えていることが分かった．次に，経路計算時間の観点では，非常に長い計算時間オーダを持つ線形計画法と比較して，非常に短い時間で経路計算ができることが分かった．本実験で評価したトポロジにおいては，十分にミリ秒オーダの計算時間であると言える．その一方で，深層学習モデルは線形計画法の特徴を十分に学習できていない．今後は学習モデルの構成やハイパーパラメータのチューニング等を行い，より線形計画法の特徴を学習できるように改良する必要がある．

4.2.5　今後の課題

今後の課題は大きく三つある．第一に，提案方式が適用可能なトポロジ規模の拡大．これは，ダイクストラ法を学習した深層学習モデルと同様に，より大きな規模のトポロジにおける評価実験が必要である．第二に，線形計画法の性能に近い負荷分散された経路の設計．これは，学習モデルの最適化や再検討により，線形計画法の性能により近づけられるような改良が必要である．第三に，モデルの最適化や適用トポロジ規模の拡大に応じた計算時間の検証．更なる性能の検証や向上のためには，学習モデルに様々な変更を加える必要がある．その結果，経路計算時間が延長される可能性がある．そのため，これらの変化による影響について検証が必要である．

4.3 むすび

本研究では，深層学習を用いた経路設計において更なる負荷分散を達成するために線形計画法を教師データとする方式を検討した．そこで，線形計画法を用いて負荷分散を考慮した経路を設計し，教師データを作成する方法を検討し，実際に教師データを作成し，深層学習モデルに学習させた．

評価実験より，実験用の小規模トポロジにおいて非常に高い精度で，輻輳制御された経路を非常に短い時間で設計できることが分かった．また，ダイクストラ法やダイクストラ法を学習した深層学習モデルよりも，優れた負荷分散が行えることを確認した．特に，線形計画法と比較して大幅に計算時間を削減することができることを示した．以上の結果より，教師データをより優れた経路設計法に変更することで，深層学習モデルの性能が向上できることが分かった．

その一方で，教師データである線形計画法と比較すると，その性質を完全に学習し，同等の負荷分散が行えているとはいえないため，今後改良が必要である．

第5章　代替経路設計のためのダイクストラ法を用いた深層学習による経路設計法

5.1　Trafiic Engineeringと代替経路

5.1.1　代替経路

トラフィック エンジニアリングでは，ネットワーク [27] [28] [29] でノードまたはリンクに故障が発生した場合に，代替経路を使用することで通信の継続を図る．代替経路を設計する際には，主経路に含まれるノードやリンクを含まない経路を設計することが一般的である．よって，最も単純な代替経路はノードやリンクの故障を想定した状況で任意の経路設計法により，主経路と異なる経路を設計する必要がある．例えば，はじめにネットワークの状況に応じて，経路設計法を用いて主経路を設計する．その後，主経路に含まれるリンクのリンクコストを無限大に設定し，再び適切な経路設計法により経路を設計することで代替経路が設計できる．このような代替経路はリンクディスジョイントな経路と呼ばれる．

代替経路には，特定のノードの故障やリンクの途絶に対して設計するような場合と，特定の故障ではなく汎用的に耐故障性を持つ経路を設計するような場合がある．例えば，リンクディスジョイントな経路やノードディスジョイントな経路は，特定の故障ではなく汎用性のある代替経路である．

図5.1はリンクディスジョイントな経路やノードディスジョイントな経路の例である．リンクディスジョイントな経路は，主経路に含まれている中継ノードは重複が許されている一方で，主経路に含まれているリンクは重複してはならない．よって，赤色の主経路に対して，青色や緑色の代替経路を設計することができる．また，ノードディスジョイントな経路は，主経路に含まれている中継ノードもリンクも重複してはならない．よって，赤色の主経路に対して，緑色の代替経路しか設計することができない．

ノードディスジョイントな経路は，リンクディスジョイントな経路と比べて経路計算難易度が高いものの，ノードの故障に応じて経路を変化することができるため耐故障性が高い．

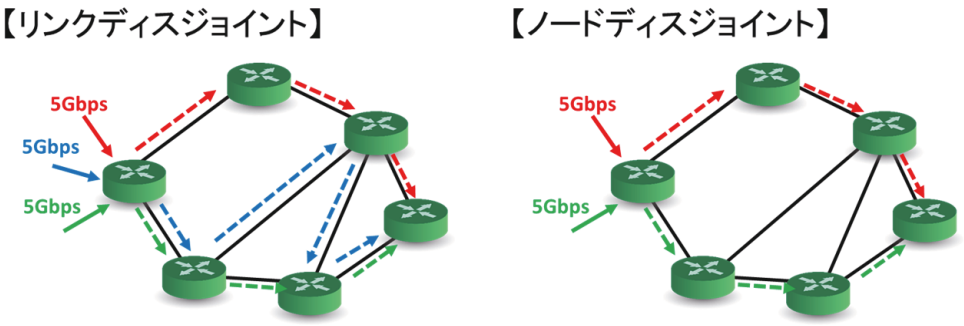


図 5.1 代替経路の例

5.1.2　代替経路設計の課題

一般的に代替経路の設計では，ホップ数が増加し，帯域負荷が増大するなどの問題がある．例えば，線形計画法などの複雑な経路設計法を用いることで，ホップ数を削減し帯域の負荷分散を図ることができるものの，非常に長い計算時間を要し，最悪の場合経路を設計する事ができないという課題がある．これらの経路設計法では，設計する経路の品質と経路計算時間の増加という，トレードオフの関係を無視することができない．

よって，これまでの輻輳制御と同様に，オンライントラヒックエンジニアリングにおける計算時間の課題を解決する必要がある．そこで，本研究では輻輳制御のための機械学習を用いた経路設計法を応用することで，これらの課題を解決することを目的とする．そのためには，深層学習モデルを用いて複数の経路を設計することと，異なる経路情報を同時に学習する必要がある．

5.1.3　本章の目標

本章における提案方式の目標は三つある．一つ目は，高い経路設計成功率である．これは，二つの異なる経路を学習させた場合に，従来の提案方式と同等の成功率で経路設計が行えるか検証する必要がある．二つ目は，負荷分散を考慮した経路の特徴を学習することである．これは，主経路と代替経路ともに教師データと同等の負荷分散された経路を深層学習モデルにより設計できるか検証する必要がある．三つめは，特定の目的を満たすような代替経路の設計である．これは，主経路に対して正しい代替経路を設計する事が必要である．本章においては，最も重要な目標である．これらの目標を解決するためには大きく二つの課題がある．

5.1.4 本章の課題

一つ目の課題は新しい教師データの設計である．先ず，目的に応じた代替経路の設計として，本研究では主経路に対してリンクディスジョイントな経路を設計させる．また，リンクディスジョイントな経路をどのような経路設計法によって生成するか検討する必要がある．二つ目の課題は新しい深層学習モデルの設計である．出力層は入力されたある拠点間トラヒックに対して，異なる二つの送受信間経路を設計できるように変更する必要がある．その一方で，可能な限りこれまでの優良な設定を用いることで，深層学習モデルの適用難易度を低減させる．

5.2　代替経路設計のための深層学習モデルの検討

5.2.1　代替経路用教師データの作成

図 5.2 は代替経路の作成例を表したものである．代替経路設計のための教師データの生成手順は以下のとおりである．先ず，リンクコストを残余帯域の逆数としたダイクストラ法を用いて主経路を設計する．次に，主経路に含まれるリンクのリンクコストを無限大に設定し，再び動的ダイクストラ法を用いて代替経路を設計する．このような代替経路を用いて，代替経路を教師データとして生成する．

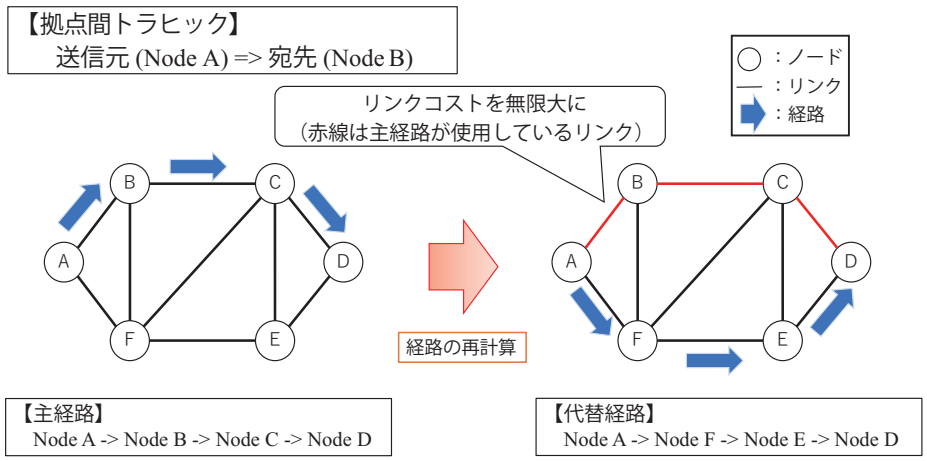


図 5.2 代替経路の設計例

5.2.2 深層学習モデルにおける出力層の改良

図 5.3 は，改良後の出力層を表したものである．従来の深層学習モデルを用いて，複数の経路を同時に設計できるように，出力層の構造を改良する．本研究では，従来の学習モデルにおける出力層に，もう一つの拠点間トラヒックに対する経路を設計するための出力層を加える方法を検討する．これらの追加した出力層は，ブロック範囲毎に中間層の各最深層と全結合するように設計する．また，出力層以外の構成に関しては，従来の深層学習モデルと全く同じ構成とする．

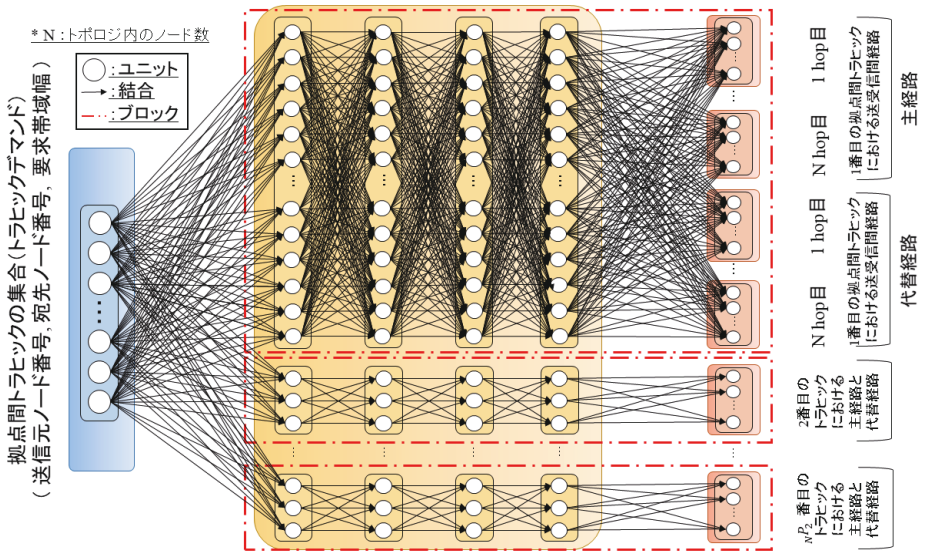


図 5.3 深層学習モデルにおける出力層の改良

5.3　線形計画法を用いた深層学習モデルの性能評価実験

5.3.1　評価条件

異なる二つの経路を学習した深層学習モデルの評価を行うために，実験用トポロジを用いて実験を行う．実験評価項目は，経路設計成功率，使用帯域幅，リンク重複率の三点である．なお，比較対象方式は，リンクコストを残余帯域の逆数とした動的なダイクストラ法である．ここで，経路設計成功率は次の式から求められる．

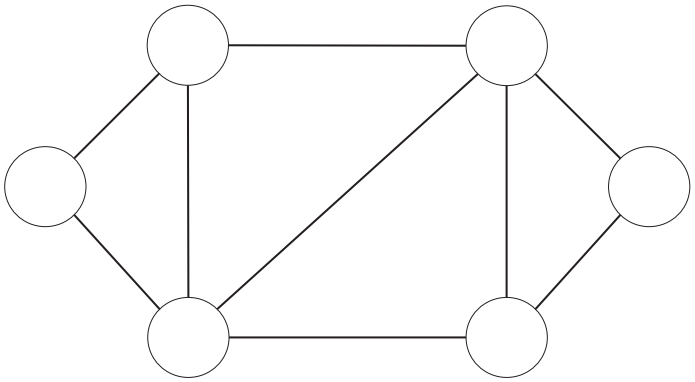
$$\frac{\text{ダイクストラ法による経路と一致している送受信間経路の数}}{{}_N\mathrm{P}_2 \times \text{データセット数}} \tag{5.1a}$$

$$\frac{\text{本研究における成功条件を満たした送受信間経路の数}}{{}_N\mathrm{P}_2 \times \text{データセット数}} \tag{5.1b}$$

評価項目 (5.1a) は，ダイクストラ法による経路との一致率であり，一般的な機械学習における正答率に該当する．評価項目 (5.1b) は，本研究における正答率の計算法である．このとき成功条件を満たす経路とは，学習モデルの出力経路において，使用するリンクがネットワーク上に存在し，送信元から宛先まで到達可能であり，ループフリーであり，かつどのリンクにも輻輳が発生していない経路である．次に，帯域使用率は最大負荷リンクにおける使用帯域幅の各値から求める．本研究では，教師データ作成のための経路設計法として，リンクコストを残余帯域の逆数としたダイクストラ法を用いており，輻輳制御を目的としている．そこで，学習モデルが教師データの特性を正しく学習できているかを，帯域使用率の観点からダイクストラ法と比較評価を行う．最後にリンク重複率は，ダイクストラ法をと深層学習モデルが設計したそれぞれの経路について，すべての送受信間経路に含まれるリンクの中で主経路と代替経路が使用している同一経路の割合である．また，平均ホップ数についても合わせて記載する．

なお，一般的に機械学習の分野では，教師データとして作成したデータセットを学習用のデータと評価用のデータに分割し，学習に用いたデータでは評価を行わない．よって，教師データの一部を分割し，学習と評価にそれぞれ用いる方法ではなく，教師データとは別にデータセットを生成し評価することで，学習モデルが汎用性のある学習を行っているか検討する．また本研究では，難易度の低減を図るために 150Mbps から 900Mbps までの 150Mbps 刻みの 6 種類に，1Gbps を合わせた 7 種類からランダムで選択する．この値は，KDDI の帯域保障サービスにおける選択可能な帯域幅を参考にした [24].

実験に用いるトポロジは以下の通りである．実験用トポロジは 6 ノード 9 リンクからなるメッシュトポロジ (図 5.4) である．これは，実験的な意味合いで用いられる小規模ネットワークトポロジ [22] である．なお，各トポロジ内のすべての帯域幅は 10Gbps で統一する．深層学習モデルの実装と実験は，OS:Ubuntu 20.04.1， 物理メモリ:128GB， GPU:GeForce RTX 3090， プログラミング言語:Python 3.8.5， Tensorflow2.3.1+Keras を用いて行った．



6node and 9link mesh topology

図 5.4 実験用トポロジ

5.3.2　実験用トポロジを用いた性能評価

5.3.2.1　経路設計成功率

図 5.5 は，実験用トポロジにおける経路設計成功率である．実験用トポロジは，6 ノードから構成されているため，すべてのノードが 1 回ずつ送受信を行うことを想定した場合，100,000 件のデータセットに含まれる拠点間トラヒックの総数は 3,000,000 である．これは ${}_6\mathrm{P}_2 \times 100,000$ から計算することできる．提案方式の経路設計成功率は主経路では 99.99% (2,999,659/3,000,000 トラヒック)，代替経路では 98.93% (2,967,938/3,000,000 トラヒック) であった．

以上の結果より，異なる二つの経路を学習した場合であっても，従来の深層学習モデルと遜色ない経路設計成功率であることが分かった．また，代替経路は若干の成功率の低下が見られるものの，十分に高い経路設計成功率であると言える．よって，出力層を改良した深層学習モデルを用いることで，異なる二つの経路を学習できていると考えられる．

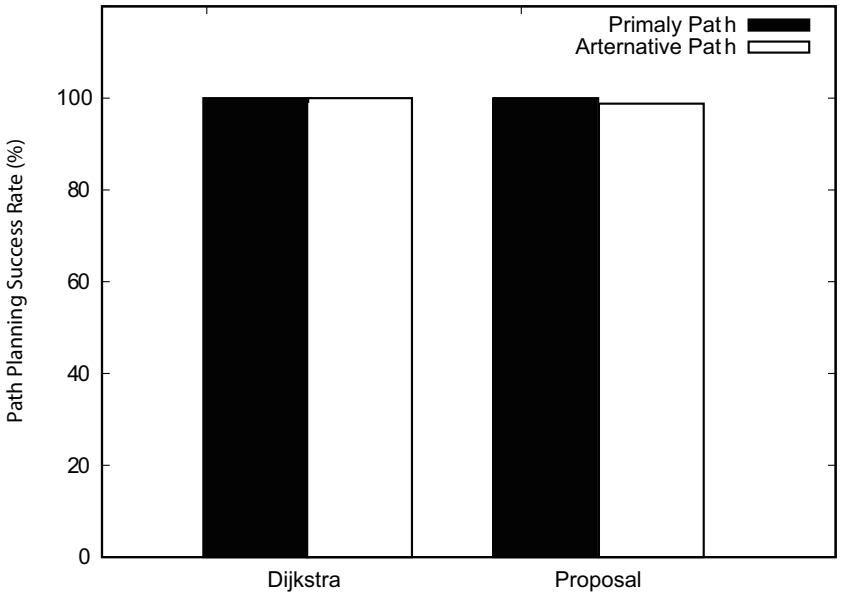


図 5.5 拠点間トラヒック毎の経路設計成功率

5.3.2.2 帯域幅使用率

図 5.6 は，すべての拠点間トラヒックを流した際の最大負荷リンクにおける使用帯域幅の平均値を比較したグラフであり，表 5.1 は，最大負荷リンクにおける最大値 (Max)，最小値 (Min)，平均値 (Average)，標準偏差 (SD:StandardDeviation)，標準誤差 (SE:StandardError)，中央値 (Median) を記載したものである．ただし，提案方式はデマンド集合に含まれるすべてのトラヒックの経路が設計できている場合にのみ計算する．これは，経路設計に失敗している場合，本来流れるべきトラヒックが流れず計算できないためである．

図 5.6 および，表 5.1 より，主経路と代替経路の使用帯域幅に着目すると，ダイクストラ法と深層学習モデルともに主経路に比べて代替経路の帯域負荷が増加する傾向にあることが分かった．これは，代替経路は主経路と比較してホップ数が増加することから，主経路と比較して帯域負荷が増加するためである．また，深層学習モデルが設計した経路はダイクストラ法による経路と比較して，主経路で約 10%の負荷増，代替経路で約 3%の負荷増であった．以上より，深層学習モデルは複数の経路情報を同時に学習し，それぞれの目的に応じた経路を設計できていると考えられる．

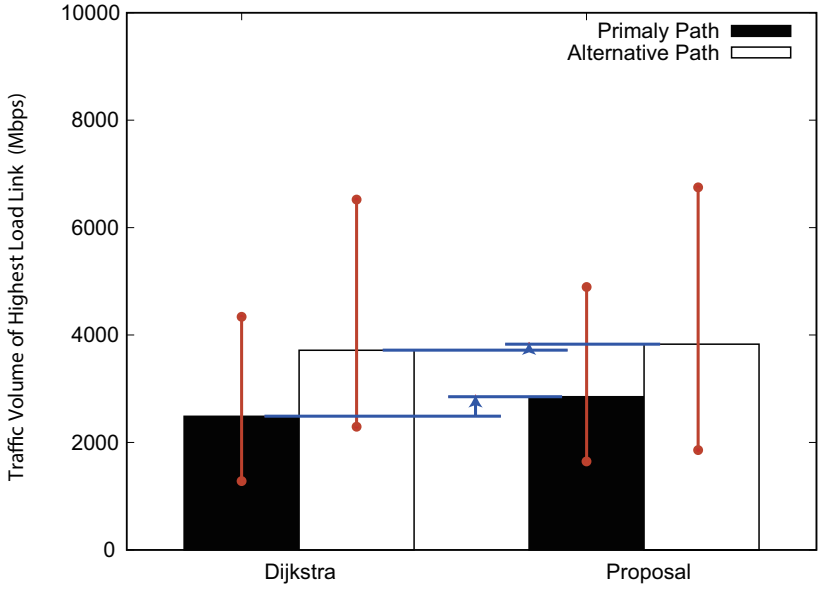


図 5.6 最大負荷リンクにおける使用帯域幅の平均値

表 5.1 最大負荷リンクにおける使用帯域幅の詳細値 [Mbps]

Method	Path	Traffic Volume of highest Load Link Max	Min	Average
dijkstra	**Primaly**	4,200	1,350	2,586.76
	Alternative	6,300	1,950	3,715.92
Proposal	**Primaly**	4,900	1,450	2,851.67
	Alternative	6,400	2,100	3,829.03

5.3.2.3　リンク重複率

最後に，表 5.2 は，ダイクストラ法および深層学習モデルにおける主経路と代替経路の平均ホップ数とリンク重複率である．先ず，平均ホップ数に着目するとダイクストラ法ならびに深層学習モデルともに，主経路に対して代替経路の経路長が長いという傾向が見られる．経路長の傾向が似ていることからも，深層学習モデルは正しくダイクストラ法の経路を学習できていると考えることができる．

次に，リンク重複率に着目すると，深層学習モデルを用いた経路設計では 0.23%と十分に低い値だった．これは，100,000 件のテストデータセットにおいて，重複リンク数は約 10,000 件であったことを示している．僅かな重複リンクについては，単純なアルゴリズムによる修正方式などを用いて完全にリンクディスジョイントな経路とすることを検討したいと考えている．

表 5.2 平均ホップ数とリンク重複率

Method	**Path**	**Average hops**	**Overlapping Rate [%]**
dijkstra	**Primaly**	1.460	
	Alternative	2.312	0.000
Proposal	**Primaly**	1.467	
	Alternative	2.202	0.230

5.3.3　考察

性能評価より, 提案方式の有効性は示された．本研究では，深層学習を用いて代替経路を設計することを目的とした学習モデルと教師データの改良を行った．実際に主経路と代替経路，異なる二つの経路を深層学習モデルに学習させたところ，従来の深層学習モデルと同等の経路設計成功率であることが分かった．また，深層学習モデルにより設計された経路は，ダイクストラ法による経路と同等の負荷分散が行えていることや，主経路と代替経路の帯域負荷に同様の傾向が見られることから，教師データの特徴を正しく学習できていると考えられる．次に，リンク重複率では，完全ではないものの深層学習モデルを用いて，十分に重複の無い経路が設計されていることが分かった．重複リンクがわずかであるため，単純なアルゴリズムによって完全にリンクディスジョイントに修正することも十分に可能だと考えている．今後は，経路計算時間についての検証や異なる状況を想定した代替経路設計について検討する必要がある．

5.3.4 今後の課題

今後の課題は大きく三つある．第一に，提案方式が適用可能なトポロジ規模の拡大．これは，負荷分散を考慮した深層学習を用いた経路設計と同様に，より大きな規模のトポロジや現実的なトポロジを用いた評価実験が必要である．第二に，重複リンク修正手法の検討．これは，重複している経路を修正し，完全にリンクディスジョイントな経路を設計することが必要である．第三に，様々な代替経路を設計するための改良．本研究では，リンクディスジョイントな経路を想定したが，ノードディスジョイントな経路や特定のノードが故障している状況を想定した代替経路の設計などについても検討したいと考えている．そのためには，深層学習モデルや教師データの再設計などが必要である．

5.4　むすび

本研究では，深層学習を用いた経路設計において代替経路を設計させる方式を検討した．そこで，ダイクストラ法を用いて異なる二つの経路を教師データとし，深層学習モデルの出力層を改良することで同時に設計させる方式を検討した．

評価実験より，実験用の小規模トポロジにおいて非常に高い精度で，異なる二つの経路を高い成功率で設計できることが分かった．また，ダイクストラ法による主経路および代替経路と同様の経路を深層学習モデルにより設計できることを確認した．特に，リンク重複率は完全に 0%ではないものの非常に少ない重複リンク数であった．以上の結果より，深層学習モデルの構成と教師データを変更することで，異なる二つの経路を同時に設計できることが分かった．

今後は，重複リンクの修正手法の検討や，更なる代替経路の設計を想定し深層学習モデルの改良などを行っていく必要がある．

第6章　結論

本研究では，増加し続ける時間変動の大きなトラヒックの効率的な制御の必要性について説明し，その解決法の一つであるトラヒックエンジニアリングを説明した．一般的にトラヒックエンジニアリングにおける経路設計法は，複雑なアルゴリズムや線形計画法を用いており，最適な経路を計算するためには非常に長い計算時間を要する．加えて最悪の場合には，事実時間では経路が計算できないことや，最適解が存在しないため計算が修了しないといった課題がある．また，トラヒックの変動に対して，リアルタイムに最適な経路を設計する様な場面では，トラヒックデマンドに含まれる全てのフローを同時に再計算する手法が用いられており，可能な限り短い時間で経路計算を行うことが望ましい．よって，最適化が困難な経路設計を行いつつ，計算時間は短くする必要があるという矛盾した課題を解決する必要がある．このような経路計算における課題に対して機械学習技術を適用することで解決を図るのが本研究の目的である．そこで，複数の拠点間トラヒックを考慮しつつ，集中管理型の輻輳制御された経路を出力できるような深層学習モデルを検討した．

第 3 章では，最も基本的な輻輳制御法であるリンクコストを残余帯域の逆数としたダイクストラ法を用いて教師データを作成し，深層学習モデルに学習させることで集中管理型の経路設計が行える方式を提案した [30] [31] [32] [33]．評価実験より，小中規模のトポロジにおいて非常に高い精度で，輻輳制御された経路をダイクストラ法と同等以上の速さで設計できることが分かった．また，従来の機械学習を用いた送受信間経路設計法よりも，優れた負荷分散を達成している．特に，提案方式はトポロジの規模が大きくなるにつれて，ダイクストラ法よりも短い時間で経路を計算できることを示した．加えて，リンク故障などを起因とするトポロジの変化に対しても，一定の頑健性を有していることを示した．以上の結果より，本提案は，トラヒックデマンドの変化がある度に，瞬時にすべてのトラヒックに対してネットワーク全体の負荷分散を考慮した経路設計を行うことができる．

第 4 章では，提案方式の更なる性能向上のために教師データに線形計画法を用いた機械学習を用いた経路設計法を提案した [34] [35] [36]．はじめに，整数線形計画法を用いてネットワーク全体の収容率を改善しつつ，平均ホップ数を最小化するような経路設計法を検討し，この経路を教師データとして作成する．次に，深層学習モデルにこの教師データを用いて学習を行い，実験用の小規模トポロジを用いて性能評価を行う．提案方式は，リンクコストを残余帯域の逆数としたダイクストラ法と比較して，最大負荷リンクにおける使用帯域幅を改善し，整数線形計画法と比較して短い時間で経路設計が行えることを示した．

第 5 章では，代替経路設計のための機械学習を用いた経路設計法を提案した [37]．代替経路の設計には，主経路と代替経路を同時に設計することが望ましいため，機械学習モデルの出力層を改良する．性能評価により，提案方式はあるトラヒックデマンドに対して，主経路と代替経路を同時に設計することができる．また，これらの経路は主経路とのリンク重複率が低い代替経路を設計できることを示した．

以上より，本研究では，トラヒックエンジニアリングにおける負荷分散を考慮した機械学習によるリアルタイム経路設計法に関する研究を行い，ダイクストラ法ベースおよび線形計画法ベースの輻輳制御のための深層学習を用いた集中管理型の経路設計法，代替経路設計のための深層学習を用いた経路設計法について，それぞれ新たな方式を提案するとともに，提案方式の有効性を明らかにしている．これらの研究成果は，ダイクストラ法より最大負荷リンクの使用帯域幅を低減した経路を線形計画法より短い時間で設計することが可能であり，今後のより優れたトラヒックエンジニアリングの実現を果たすための重要な技術となるものである．

今後の最も重要な課題は，より大規模なトポロジにおいても輻輳制御された経路を高い精度で設計することである．また，教師データの作成に用いる経路設計法を線形計画法に変更することで，より優れた経路を学習し設計できるような深層学習モデルを提案する．加えて，適用するトポロジの規模に応じた，深層学習モデルの構成やハイパーパラメータの規定値について検討し，深層学習モデルの導入難易度低減を図る．

謝辞

本論文は筆者が日本大学大学院工学研究科情報工学専攻博士後期課程に在籍中の研究成果をまとめたものである．

本研究を実施し，学位論文をまとめるにあたり，学部 4 年の研究室所属時から多くのご支援とご指導を賜りました，指導教員である日本大学工学部情報工学科准教授 見越大樹先生に深く感謝いたします．また，博士後期課程在籍時より，指導教員として研究活動ならびに研究論文の執筆など，多くのご指導を賜りました，日本大学工学部情報工学科教授 源田浩一先生にも深く感謝いたします．

東京都市大学情報工学部知能情報工学科教授 塩本公平先生，福井大学工学系部門工学領域情報・メディア工学講座教授 橘拓至先生，日本大学工学部情報工学科教授 菊間一宏先生には，副査としてご助言頂き深く感謝いたします．

日本大学工学部情報工学科教授 上田 清志先生には，研究会や論文審査会におきましてご助言頂き深く感謝いたします．

日本大学工学部情報工学科見越研究室の各位には研究遂行にあたり日頃より有益なご討論ご助言を戴き，感謝申し上げます．特に，学部 4 年生から博士前期課程 2 年生までの 3 年間共に研究活動に従事した，同研究室卒業生 安斎優也氏，石井翔氏，遠藤敦博氏，及び，本研究の実施にあたり，熱心な協力を戴いた，同研究室卒業生 雫石樹生氏に感謝申し上げます．

参考文献

[1] N. Wang, K. H. Ho, G. Pavlou and M. Howarth, "An overview of routing optimization for internet traffic engineering," in IEEE Communications Surveys and Tutorials, vol. 10, no. 1, pp. 36-56, First Quarter 2008, doi: 10.1109/COMST.2008.4483669.

[2] M. Pioro, et al., "On open shortest path first related network optimization problems," Journal of Combinatorial Optimization, Volume 48, Page 201-223, Volume 48, April 2002. doi: 10.1016/S0166-5316(02)00036-6.

[3] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), Tel Aviv, Israel, 2000, pp. 519-528 vol.2, doi: 10.1109/INFCOM.2000.832225.

[4] D. Mitra and K. G. Ramakrishnan, "A case study of multiservice, multipriority traffic engineering design for data networks," Seamless Interconnection for Universal Services. Global Telecommunications Conference. GLOBECOM'99. (Cat. No.99CH37042), Rio de Janeiro, Brazil, 1999, pp. 1077-1083 vol. 1b, doi: 10.1109/GLOCOM.1999.830281.

[5] Y. Lee, et al., " A Framework for the Control of Wavelength Switched Optical Networks (WSON) with Impairments, " IETF Internet draft, draft-ietf-ccamp-wson-impairments-07, April 2011.

[6] H. Yonezu, K. Kikuta, D. Ishii, S. Okamoto, E. Oki and N. Yamanaka, "QoS aware energy optimal network topology design and dynamic link power management," 36th European Conference and Exhibition on Optical Communication, Turin, Italy, 2010, pp. 1-3, doi: 10.1109/ECOC.2010.5621098.

[7] D. Ishii, K. Nakahara, S. Okamoto and N. Yamanaka, "A novel IP Routing/Signaling based service provisioning concept for ubiquitous grid networking environment,"

2010 IEEE Globecom Workshops, Miami, FL, USA, 2010, pp. 1746-1750, doi: 10.1109/GLOCOMW.2010.5700240.

[8] S. Kandula, D. Katabi, B. Davie, and A. Charny, “Walking the tightrope: responsive yet stable traffic engineering,” in Proceedings of ACM SIGCOMM 2005, Aug. 2005, pp.253-264, doi: 10.1109/GLOCOM.1999.830281.

[9] A. Elwalid, C. Jin, S. Low and I. Widjaja, "MATE: MPLS adaptive traffic engineering," Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), Anchorage, AK, USA, 2001, pp. 1300-1309 vol.3, doi: 10.1109/INFCOM.2001.916625.

[10] T. Otoshi et al., "Traffic engineering based on stochastic model predictive control for uncertain traffic change," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 2015, pp. 1165-1170, doi: 10.1109/INM.2015.7140461.

[11] N. Kato et al., "The Deep Learning Vision for Heterogeneous Network Traffic Control: Proposal, Challenges, and Future Perspective," in IEEE Wireless Communications, vol. 24, no. 3, pp. 146-153, June 2017, doi: 10.1109/MWC.2016.1600317WC.

[12] Nei Kato, "The Deep Learning Vision for Heterogeneous Networks Control –Proposal, Challenges, and Future Perspective –," The 2017 International Conference on Wireless Internet (WICON ’17), Dec. 2017.

[13] Yuan Zuo, Yulei Wua, Geyong Min, Laizhong Cui, "Learning based network path planning for traffic engineering ," Future Generation Computer Systems 92, (2019) 59 67. doi: 10.1016/j.future.2018.09.043

[14] G. Bernardez et al., "Is Machine Learning Ready for Traffic Engineering Optimization?," 2021 IEEE 29th International Conference on Network Protocols (ICNP), Dallas, TX, USA, 2021, pp. 1-11, doi: 10.1109/ICNP52444.2021.9651930.

[15] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils,T. Telkamp, and P. Francois, “A declarative and expressive approach to control forwarding paths in carrier-grade networks,” ACM SIGCOMM computer communication review, vol. 45, no. 4, pp. 15–28, 2015.

[16] J. Zhang, M. Ye, Z. Guo, C. -Y. Yen and H. J. Chao, "CFR-RL: Traffic Engineering With Reinforcement Learning in SDN," in IEEE Journal on Selected Areas in Communications, vol. 38, no. 10, pp. 2249-2259, Oct. 2020, doi: 10.1109/JSAC.2020.3000371.

[17] M. Ye, J. Zhang, Z. Guo and H. J. Chao, "DATE: Disturbance-Aware Traffic Engineering with Reinforcement Learning in Software-Defined Networks," 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), Tokyo, Japan, 2021, pp. 1-10, doi: 10.1109/IWQOS52092.2021.9521343.

[18] R. Braden, D.Clark, and S. Shenker,“Integrated services in the internet architecture: An overview,” RFC 1633, IETF, June. 1994.

[19] 長 健二朗（ソニーコンピュータサイエンス研究所）,（社）日本ネットワークインフォメーションセンター編 "QoS 技術〜Intserv と diffserv," Internet Week 98 国立京都国際会館, 1998 年 12 月 16 日.

[20] R. Braden ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin,“Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification,” RFC 2205, IETF, Sep. 1997.

[21] Rubio-Largo Á, et al., "Multiobjective swarm intelligence for the trafc grooming problem," Computational Optimization and Applications, vol. 60, no. 2, pp. 479–511, Mar. 2015, doi: 10.1007/s10589-014-9682-8.

[22] I. Popescu, B. Uščumlić, A. Triki, Y. Pointurier, A. Gravey and P. Gravey, "Scalable routing, scheduling and virtualization for TWIN optical burst switching networks," 2015 20th European Conference on Networks and Optical Communications - (NOC), London, UK, 2015, pp. 1-6, doi: 10.1109/NOC.2015.7238609.

[23] Javier Vales-Alonso, et al., "Performance analysis of optimal schedulers in single channel dense radio frequency identification environments", EURASIP Journal on Embedded Systems 2013, Jan. 2013, doi: 10.1186/1687-3963-2013-11.

[24] 株式会社 KDDI 国内イーサネット専用サービスサービスメニュー (帯域選択型), https://www.kddi.com/business/network/intranet/ethernet-senyo/service/menu2/

[25] Alvaro L. Barradas and Maria do Carmo R. Medeiros, "An Intrinsic TE Approach for End-to-End QoS Provisioning in OBS Networks Using Static Load-Balanced Routing Strategies," Future Internet 2010, Oct. 2010

[26] The Internet Topology Zoo, http://www.topology-zoo.org/index.html

[27] Guo, Y., Kuipers, F. and Van Mieghem, P. (2003), Link-disjoint paths for reliable QoS routing. Int. J. Commun. Syst., 16: 779-798.

[28] J. Yallouz, O. Rottenstreich, P. Babarczi, A. Mendelson and A. Orda, "Optimal link-disjoint node-"somewhat disjoint" paths," 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, 2016, pp. 1-10.

[29] J. Yallouz, O. Rottenstreich, P. Babarczi, A. Mendelson and A. Orda, "Minimum-Weight Link-Disjoint Node-"Somewhat Disjoint" Paths," in IEEE/ACM Transactions on Networking, vol. 26, no. 3, pp. 1110-1122, June 2018.

[30] 伊藤 真, 見越 大樹, 源田 浩一, " トラヒックエンジニアリングのための深層学習を用いた集中管理型の経路設計法 ", 信学論 (B), Sep 2023, Vol.J107-B, No.1, pp.23-37.

[31] Makoto Ito and Taiju Mikoshi, " Path Planning Method using Deep Learning Model for Traffic Engineering in Small Networks ", Proceedings of the 31st International Telecommunication Networks and Applications Conference, pp. 65-70, Nov. 2021.

[32] 伊藤 真, 見越 大樹, 大山 勝徳, 西園 敏弘, " 帯域保障サービスのための回帰型深層学習モデルを用いた経路設計法 ", 電子情報通信学会 技術研究報告 NS2019-240, 2020 年 3 月, pp. 353-358.

[33] 伊藤 真, 見越 大樹, " 基幹ネットワークにおける輻輳制御のための深層学習を用いた経路設計法 ", 電子情報通信学会 技術研究報告 NS2020-146, 2021 年 3 月, pp. 137-142.

[34] Makoto. Ito, Taiju. Mikoshi and Kouichi. Genda, "Deep Learning Based Path Planning Using Integer Linear Programming Method to Teacher Signal," 2023 33rd International Telecommunication Networks and Applications Conference (ITNAC), Sydney, Australia, 2023, pp. 92-97.

[35] 雫石 樹生, 伊藤 真, 見越 大樹, " 機械学習を用いた経路設計法における線形計画法による教師信号作成法の提案 ", 令和 4 年度情報処理学会東北支部研究会 Section 2-1, 2023 年 1 月.

[36] 伊藤 真, 雫石 樹生, 見越 大樹, " 機械学習を用いた経路設計法のための最適な教師信号の検討 ", 電子情報通信学会 技術研究報告 NS2023-4, 2023 年 4 月, pp. 19-24.

[37] Makoto Ito, Taiju Mikoshi, and Kouichi Genda, "Planning of Primary and Alternative Path using Deep Learning Model for Traffic Engineering in Small Networks ", Proceedings of the 18th International Conference on Ubiquitous Information Management and Communication.

付 録A　本論文を構成する論文

A. 基本論文

1) 伊藤 真, 見越 大樹, 源田 浩一,“トラヒックエンジニアリングのための深層学習を用いた集中管理型の経路設計法”, 信学論 (B), Sep 2023, Vol.J107-B, No.1, pp.23-37.

C. 日本大学工学部紀要及びこれに準ずるもの (国際会議)

1) Makoto Ito and Taiju Mikoshi,“Path Planning Method using Deep Learning Model for Traffic Engineering in Small Networks”, Proceedings of the 31st International Telecommunication Networks and Applications Conference, pp. 65-70, Nov. 2021.
2) Makoto. Ito, Taiju. Mikoshi and Kouichi. Genda, ”Deep Learning Based Path Planning Using Integer Linear Programming Method to Teacher Signal,” 2023 33rd International Telecommunication Networks and Applications Conference (ITNAC), Sydney, Australia, 2023, pp. 92-97.
3) Makoto Ito, Taiju Mikoshi, and Kouichi Genda,“Planning of Primary and Alternative Path using Deep Learning Model for Traffic Engineering in Small Networks”, Proceedings of the 18th International Conference on Ubiquitous Information Management and Communication.

付 録B　学会の大会, 支部大会などの口頭発表等

(学外)

1) 伊藤 真, 雫石 樹生, 見越 大樹, “ 機械学習を用いた経路設計法のための最適な教師信号の検討 ”, 電子情報通信学会 技術研究報告 NS2023-4, 2023 年 4 月, pp. 19-24.
2) 雫石 樹生, 伊藤 真, 見越 大樹, “ 機械学習を用いた経路設計法における線形計画法による教師信号作成法の提案 ”, 令和 4 年度情報処理学会東北支部研究会 Section 2-1, 2023 年 1 月.
3) 伊藤 真, 見越 大樹,“基幹ネットワークにおける輻輳制御のための深層学習を用いた経路設計法 ”, 電子情報通信学会 技術研究報告 NS2020-146, 2021 年 3 月, pp. 137-142.
4) 伊藤 真, 見越 大樹, 大山 勝徳, 西園 敏弘, “ 帯域保障サービスのための回帰型深層学習モデルを用いた経路設計法 ”, 電子情報通信学会 技術研究報告 NS2019-240, 2020 年 3 月, pp. 353-358.

(学内)

1) 伊藤 真, 見越 大樹, 源田 浩一, “ 中規模トポロジにおける深層学習を用いた経路設計法の性能評価 ”, 第 65 回日本大学工学部学術研究報告会講演要旨集, 2022 年 12 月, pp. 71-74.
2) 雫石 樹生, 伊藤 真, 見越 大樹, “ 機械学習型経路設計法における教師信号のためのトラフィックと経路の ILP による作成法 ”, 第 65 回日本大学工学部学術研究報告会講演要旨集, 2022 年 12 月, pp. 67-70.
3) 伊藤 真, 見越 大樹, 大山 勝徳, 西園 敏弘, “ トラヒック受付制御における残余帯域を考慮した機械学習による経路決定法 ”, 第 62 回日本大学工学部学術研究報告会講演要旨集, 2019 年 12 月, pp. 44-47.