特別研究報告

題目

コアペリフェリ構造の導入による仮想化ネットワーク機能の効率性および安定性の向上に関する研究

指導教員
村田 正幸 教授

報告者
津久井 佑樹

平成 30 年 2 月 14 日

大阪大学 基礎工学部 情報科学科

平成 29 年度 特別研究報告

コアペリフェリ構造の導入による仮想化ネットワーク機能の効率性および安定性の向上に関する研究

津久井 佑樹

内容梗概

ネットワーク機能提供の柔軟性を高めつつ提供コストを削減することを目的として、NFV (Network Functions Virtualization) の導入が進められている。NFV の運用にあたっては解決すべき課題がいくつかあり、その一つに VNF (Virtual Network Function) の配置問題がある。VNF 配置問題に対する既存の解法では常に最良の VNF 配置を求め、その結果として、VNF 配置場所の変更が多数必要となる。多数の配置変更を必要とする場合、配置変更の過程で生じるネットワーク機能提供の継続性が損なわれる恐れがある。そこで、効率性を高めつつも適応性を有する VNF 配置を得ることを目的として、コアペリフェリモデルに着目する。コアペリフェリモデルは、システムの環境変化に対する適応性と効率性を説明するモデルの一つであり、システム要素をコアとペリフェリに分類する。コアのシステム要素は効率的かつ安定的に動作し、一方、ペリフェリのシステム要素は環境変化に応じた変化を許容している。これにより、コアペリフェリモデルにもとづく VNF 配置では、効率性と柔軟性の両立が期待できる。

NFV において、頻繁に実行される VNF をコアと解釈し、コアではない VNF をペリフェリと解釈すれば良い。しかし、このような解釈を NFV における VNF 配置問題に応用するには、いくつかの課題がある。一つは、コアサイズに対する動作の効率性であり、もう 1 つの課題は、コア/ペリフェリ分類の時間軸方向での安定性である。本報告では、NFV における VNF 配置問題を対象とし、コアペリフェリモデルの適用にもとづく VNF の分類と、その分類により得られる効率性と安定性を定量的に明らかにする。ただし、NFV はまだ標準化を策定する段階であり、実測にもとづくサービスチェイン要求パターンを用いた定量的評価を行うことはできない。そこで、まず経年的変化の実測データが有る AS (Autonomous System) ネットワークを対象とし、コアペリフェリモデルの適用によるコアサイズとコアの安定性を評価する。次に、生成モデルにもとづくサービスチェイン要求に対してコアペリフェリモデルにより VNF を分類し、コアの効率性および安定性の知見が有効であることを確認しつつ、定量的な評価を行う。評価の結果、コアサイズを 128 とした場合、約 24% の

サービスチェイン要求が処理可能であることがわかった。また、要求される VNF の偏在性が高い場合はコアの効率性と安定性が高まるため配置する VNF 数を 128 から 32 に減らしても多くのリクエストを継続的に処理できることがわかった。また、要求される VNF の偏在性が低い場合はコアの効率性と安定性が低下し、配置変更の頻度を高めることによる対応が必要であることが明らかとなった。

主な用語

ネットワーク仮想化 (NFV)、仮想ネットワーク機能 (VNF)、コアペリフェリモデル、サービスチェイン、有向グラフ、DC グラフ

目 次

図 目 次

表 目 次

1　はじめに

スマートフォンやタブレットの普及を背景に、情報ネットワークと情報ネットワークを利用するアプリケーションやサービスは人々の生活に浸透している [1]。利用者は従来の料金設定で従来通り、あるいはより良いサービスを求め、さらにはサービス利用に伴って通信量が増大する。通信事業者は、通信量の増大に対処しつつ、様々な利用者が求める様々なサービスを効率的に提供することが求められている。利用者にサービスを提供するにあたって、通信事業者は、例えばファイアウォール機能などのネットワーク機能を提供する。一般に、ネットワーク機能は専用ハードウェアを用いて提供されるため、ユーザの需要の変化に合わせたネットワーク機能の処理能力の調整が困難となる。また、専用ハードウェアの導入コストや運用コストも増大するため、コスト効率性を高めることが課題となる。この問題の解決策として、ネットワーク仮想化 (NFV: Network Functions Virtualization) が注目されている [2,3]。NFV ではこれまで専用のハードウェアで実現されていたネットワーク機能を分離し、汎用サーバ上で動作する仮想ネットワーク機能 (VNF: Virtual Network Function) として実現する。VNF は一つの汎用ハードウェア上で複数のインスタンスを動作させることが可能であるため、通信事業者が抱えるハードウェア数を減らし、設備投資や運用にかかるコストを軽減することが期待できる。また、利用者が所望するネットワーク機能群の提供は、VNF を鎖状に接続したサービスチェイン要求として通信事業者が受け入れ、これを通信事業者が有する汎用ハードウェア上で実行処理することにより行われる。VNF は仮想化されるため、要求されるネットワーク機能の変化に応じて実行される汎用ハードウェアを変更したり、新たな機能を提供する VNF を作成し展開することが可能となる。これにより、ユーザの需要の変化に合わせてネットワーク機能の処理能力を調整可能とする柔軟性が提供可能となる。

NFV によって、ネットワーク機能提供のコスト効率性と柔軟性を高めることが可能となるが、運用にあたっては解決すべき課題がいくつかある。その一つに VNF の配置問題がある [4] 。VNF 配置問題は、汎用ハードウェアの数量が与えられるものとして、サービスチェイン要求の変化に対して利用者品質の向上や運用コストを削減するための適切な VNF の配置を求める問題である。文献 [5] では MIP (Mixed Integer Programming) を用いて、配置コストの最小化と負荷の分散の観点で最適な VNF 配置手法を示している。しかし、この手法では MIP によって VNF 配置を求めるため、計算に膨大な時間を必要とする。そのため、サービスチェイン要求の変化に応じて VNF の再配置を行う場合、それに伴う計算時間が問題となる。さらに、ヒューリスティック手法等によって計算時間の問題が解決できると仮定しても、これらの手法によって常に最良の VNF 配置を求める結果として、VNF 配置場所の変更が多数必要となる懸念がある。多数の配置変更を必要とする場合、配置変更の過程で生

じるネットワーク機能提供の継続性が損なわれる恐れがある。また、変更後の VNF の配置において、一定の性能品質を確保するためには、すべての VNF において実行品質を担保しなければならない。近年は、ネットワーク機能の仮想化だけではなく、サービスアプリケーションが利用する機能の仮想化についても検討がなされており、すべての VNF に対して実行品質を担保することは困難となる。そこで、効率性を高めつつも適応性を有する VNF 配置を得ることを目的として、コアペリフェリモデル [6] に着目する。コアペリフェリモデルは、システムの環境変化に対する適応性と効率性を説明するモデルの一つであり、システム要素をコアとペリフェリに分類する。コアのシステム要素は効率的かつ安定的に動作し、一方、ペリフェリのシステム要素は環境変化に応じた変化を許容している。これにより、コアペリフェリモデルにもとづくシステム構成では、効率性と柔軟性が両立される [6]。このコアペリフェリモデルを用いて VNF 機能を解釈し、さらには設計に応用することで、効率性と柔軟性を両立する VNF 配置の実現が期待できる。すなわち、VNF 全体の実行効率を高めるのではなく、コアに分類される VNF の実行効率を高めることによってシステムの効率性を高めつつ、機能要求の多様化についてはペリフェリに分類される VNF のみを組み替えることによって VNF 再配置時の変更量の抑制しつつ対処することが期待される。

NFV において、効率的かつ安定的に動作させたい VNF は、頻繁に実行される VNF である。実際に、文献 [7] のシミュレーション結果では、サービスチェインの要求量の変化に迅速に対応するためには、共通して実行される VNF 集合を保持することが望ましいことが示されている。したがって、頻繁に実行される VNF をコアと解釈し、コアではない VNF をペリフェリと解釈すれば良い。しかし、このような解釈を NFV における VNF 配置問題に応用するには、いくつかの課題がある。一つは、コアサイズに対する動作の効率性である。例えば図 1 のように、ある特定の VNF 集合をコアに分類し、エッジノードに配置するものとする。すべての VNF をコアとしエッジに配置すれば、エッジノードですべてのユーザリクエストが処理できる。しかし、エッジノードのリソース量はコスト観点から有限かつ最小としなければならず、すべての VNF をコアとすることは困難であり、一部の VNF をコアとせざるを得ない。その際に、何種類の VNF をコアとすれば、何割のサービスチェイン要求（の一部）を処理することが可能となるのかを定量的に明らかにしておく必要がある。もう一つの課題は、コア/ペリフェリの分類の時間軸方向での安定性である。ある時刻のサービスチェイン要求パターンにもとづいて VNF をコアとペリフェリに分類したとしても、サービスチェイン要求が変化する中で、その分類が長期にわたって有効となるとは限らない。もちろん、コアペリフェリモデルにおけるコアは、ペリフェリに比べて安定的ではあるが、サービスチェイン要求が変化する中でコア自身も効率性を高めるために緩やかに組み替えて進化させなければならないが、現状では適正な進化速度や進化量は明らかになっていない。また、ペリフェリをどの程度変化させれば良いかを示しつつ、 VNF 再配置時の変更

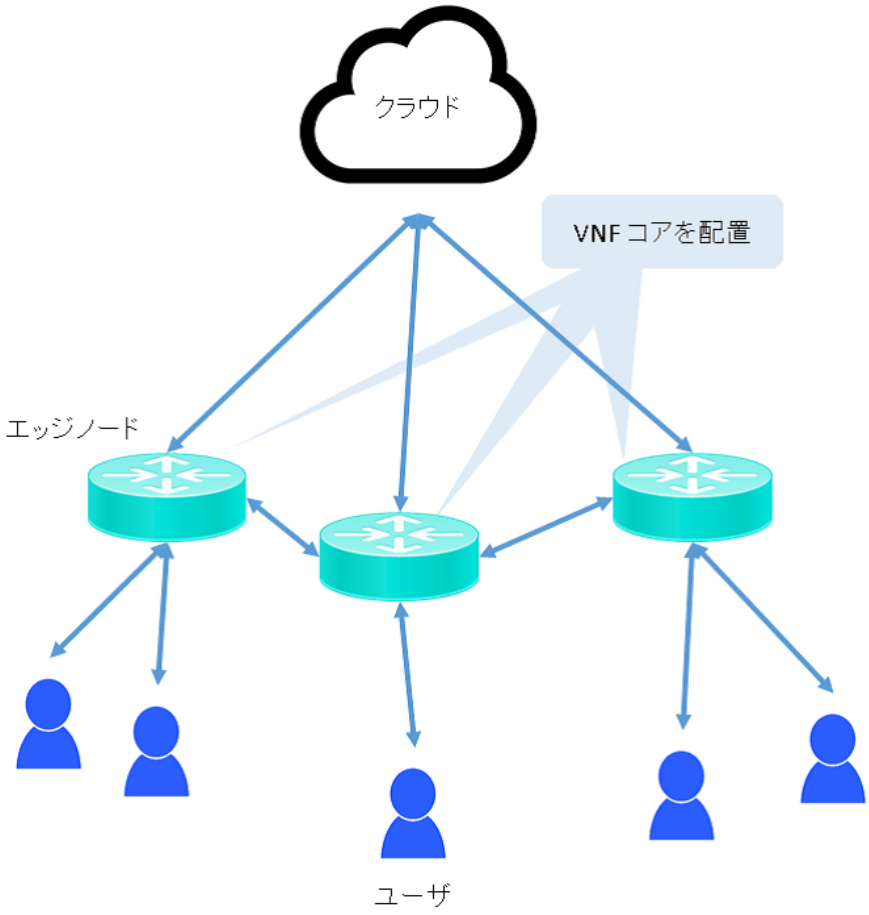

図 1: VNF 配置の例

がどの程度抑えられるかを明らかにする。

本報告では、NFV における VNF 配置問題を前提とし、コアペリフェリモデルの適用にもとづく VNF の分類と、その分類により得られる効率性と安定性を定量的に明らかにする。ただし、NFV はまだ標準化を策定する段階であり、実測にもとづくサービスチェイン要求パターンを用いた定量的評価を行うことはできない。そこで、まず経年的変化の実測データが有る 自律システム (AS: Autonomous System) のネットワークを対象とし、コアペリフェリモデルの適用によるコアの効率性および安定性を評価する。AS ネットワークの実測データは AS パス情報であり、NFV におけるサービスチェイン要求情報と親和性が高い。次に、生成モデルにもとづくサービスチェイン要求に対してコアペリフェリモデルにより VNF を分類し、コアの効率性および安定性の知見が有効であることを確認しつつ、定量的な評価を行う。

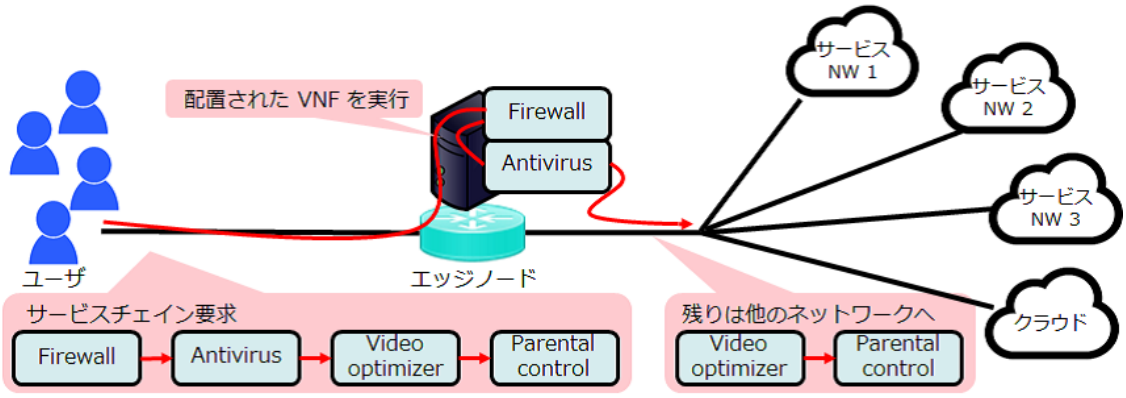

図 2: サービスチェイン要求の例

2　有向グラフを対象としたコア抽出方法

本章では、有向グラフを対象としたコアの抽出方法について述べる。また、NFV のサービスチェイン要求からコアを抽出するにあたってこの方法を用いる理由についても述べる。

2.1　サービスチェイン

NFV ではユーザからのリクエストを、VNF を鎖状に連鎖させたサービスチェインとして扱う。ユーザにサービスを提供する際には、サービスチェインの順序にしたがって VNF を実行する。サービスチェインにはトラヒックフローの自動化や、実行される VNF の組み換えの柔軟性が高いといった利点がある。

図 2 にサービスチェイン要求の例を示す。サービスチェイン要求の順序に従い、 VNF1, VNF2, VNF3 の順序で VNF が実行される。

2.2　サービスチェイン要求の有向グラフ化

サービスチェインからコアを抽出するにあたり、サービスチェインを有向グラフに変換してそこからの抽出を試みる。有向グラフにコアペリフェリモデルを適用する場合、効率性の観点より他のノードから頻繁に呼び出される、あるいは他のノードを頻繁に呼び出しているノードの集合をコアとして認識することができる。ここで、頻繁に呼び出されているということは入次数が高いことを表し、頻繁に呼び出しているということは出次数が高いことを示している。したがって、有向グラフ全体から次数が高いノードで構成されたサブグラフの切り出しを行うことでコアの抽出が期待できる。

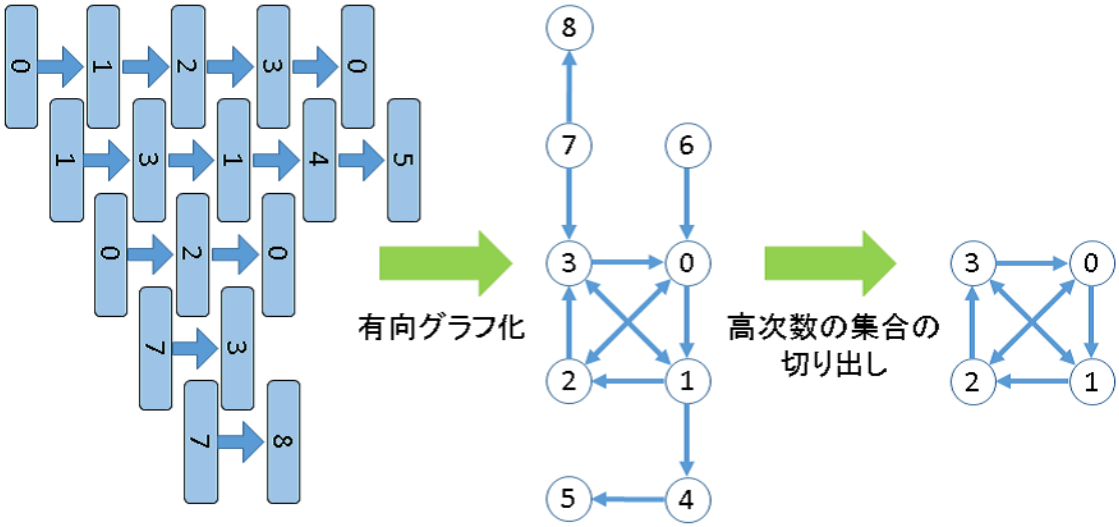

図 3: リクエストのグラフ化と集合の切り出しの例

図 3 ではリクエストを有向グラフに変換し、そこから次数の高いノードの集合を切り出す様子を示している。図の左側はサービスチェイン要求の例である。図の中央はそのサービスチェイン要求を有向グラフに変換したものである。図の右側には有向グラフから切り出した、次数の高いノードで構成されたサブグラフを示している。図 3 において、入次数と出次数はともに 2 が最高であり、切り出したサブグラフは入次数と出次数が 2 のノードで構成されている。

2.3 サブグラフ分解にもとづく VNF コアの抽出

有向グラフから次数の高いノードで構成されたサブグラフの切り出しを行うために、文献 [8] で提案されたメトリクスを用いる。文献 [8] で提案されたメトリクスでは、それぞれのノードのハブ/オーソリティ性のみではなく各ノード間の有向辺についても着目することによって、有向グラフ全体からコラボレーティブな特徴を有するサブグラフを抽出する。このメトリクスを用いることで、有向グラフを次数にもとづいて複数のサブグラフに分解する。このメトリクスの定義を以下に示す。

まず、有向グラフを $D = (V, E)$ とおく。V は D の頂点集合であり、E は D の辺集合である。このとき、D を構成するノードの最低入次数 $\delta^{in}(D)$ と最低出次数 $\delta^{out}(D)$ を式 (1) および式 (2) でそれぞれ定義する。

$$\delta^{in}(D) = min\{x \mid \mathbf{deg}^{in}_{D}(x) \mid x \in V\} \tag{1}$$

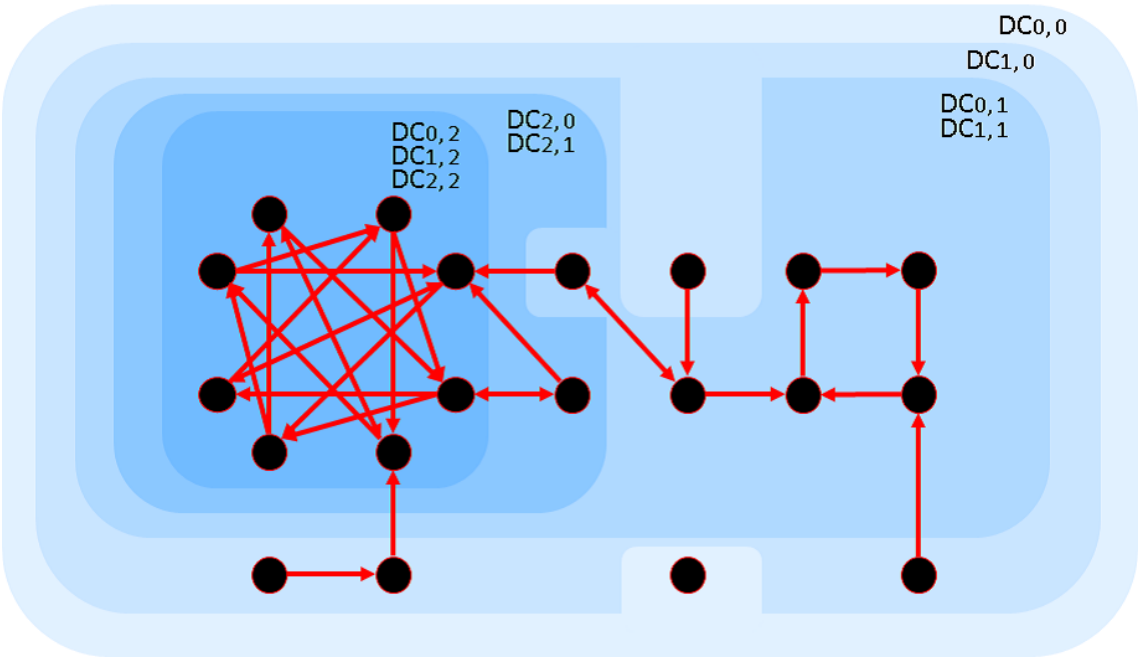

図 4: メトリクスを用いたサブグラフ分解の例

$$\delta^{out}(D) = min\{x \mid \mathbf{deg}_D^{out}(x) \mid x \in V\} \tag{2}$$

ここで、$\mathbf{deg}_D^{in}(x)$ は D を構成するノード x の入次数、$\mathbf{deg}_D^{out}(x)$ は D を構成するノード x の出次数を表す。

非負整数 k、l について式 (3) を満たす D の最大のサブグラフ F を $\mathbf{DC}_{k,l}(D)$ と定義する。ここで、$\mathbf{DC}_{0,0}(D) = D$ であることに注意する。

$$\delta^{out}(F) \geq k,\ \delta^{in}(F) \geq l \tag{3}$$

式 (3) を満たす F が存在しない場合、$\mathbf{DC}_{k,l}(D)$ は空であるとする。また、式 (3) で得られるサブグラフについて常に式 (4) が成り立つ。

$$k_1 \geq k_2 \wedge l_1 \geq l_2 \Rightarrow \mathbf{DC}_{k_1,l_1} \subseteq \mathbf{DC}_{k_2,l_2} \tag{4}$$

式 (4) が成立するため、ある有向グラフ D を分解して得られる $\mathbf{DC}_{k,l}(D)$ の組み合わせは有限個である。また、Algorithm1 に示すアルゴリズムにしたがって k, l をそれぞれ 0 から変化させることで D を分解して得られる全てのサブグラフを求めることができる。

メトリクスを用いたサブグラフの分解例を図 4 に示す。この図では Algorithm1 を用いて分解して得られる全てのサブグラフを求めている。求めたサブグラフについては色の濃淡で

Algorithm 1 有向グラフをサブグラフに分解するアルゴリズム

Require: D は有向グラフ

1: $k \Leftarrow 0$
2: **while** $\mathbf{DC}_{k,l}(D)$ が空でない **do**
3: $\quad l \Leftarrow 0$
4: $\quad$ **while** $\mathbf{DC}_{k,l}(D)$ が空でない **do**
5: $\quad\quad \mathbf{DC}_{k,l}(D)$ を求める
6: $\quad\quad l \Leftarrow l + 1$
7: $\quad$ **end while**
8: $\quad k \Leftarrow k + 1$
9: **end while**

領域を示している。より薄い色の領域はそれよりも濃い色で示された領域を含んでおり、サブグラフの包含関係を表している。

本節で定義したメトリクスを用いることで有向グラフ化したサービスチェインを次数にもとづいてサブグラフに分解することが可能である。しかし、このメトリクスを用いただけではサブグラフに分解することまでしかできない。そのため、得られたサブグラフが次数の高いノードで構成されていた場合でも、そのサブグラフが NFV のサービスチェイン要求においてコアとしての性質を持つかどうかは保証されない。したがって、分解して得られたサブグラフについて NFV のサービスチェイン要求における効率性と安定性の評価を別途行う必要がある。

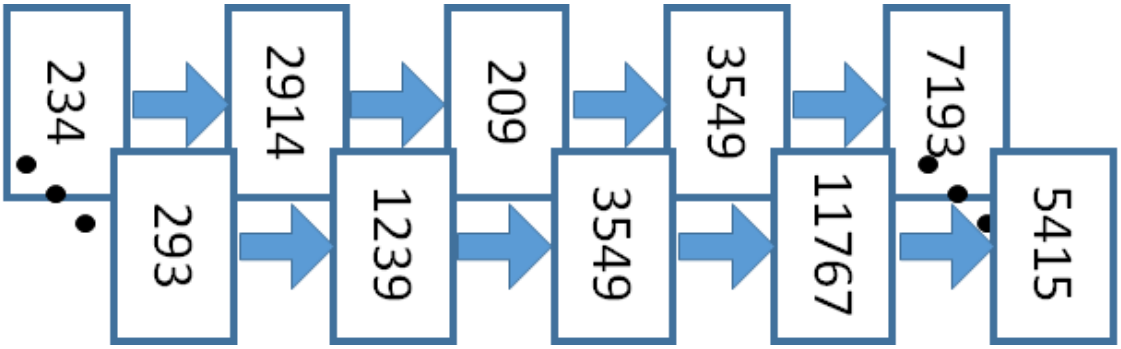

図 5: AS パスデータの例

3　インターネットの経路データを用いたコア抽出方法の適用性評価

本章では、2 章で述べたコアの抽出方法を NFV のサービスチェイン要求に適用する前段階として行った、実測されたインターネットの経路データを用いた抽出方法の適用性評価の指標およびその結果について述べる。

3.1　AS パスデータ

本報告では、2 章で述べたコアの抽出方法を NFV のサービスチェイン要求に適用し、得られたサブグラフの効率性と安定性の評価を行う。しかし、NFV の標準化は策定段階にあるため、実測にもとづいたサービスチェイン要求を用いた定量的評価を行うことはできない。そこで、本報告ではまず実測されたインターネット経路のデータを用いる。インターネット経路データに対してコアの抽出方法の適用および得られたサブグラフの効率性と安定性の評価を行い、定性的にどのような特徴が得られるのかを求める。

インターネットの経路データとして文献 [9] で使用されている AS パスデータを用いる。AS パスデータは RouteViews プロジェクトのサーバで収集された BGP テーブルから抽出した。 RouteViews のサーバは、プロジェクト開始からほぼ同一の ISP から BGP テーブルを収集しているため、抽出した AS パスデータは経年での一貫した比較が可能である点で優れている。図 5 のように、AS パスデータはトラヒックの通った AS をリストとして表している。また、抽出した AS パスデータは複数のエントリで構成されており、これらのエントリを統合することで NFV のサービスチェイン要求と同様に有向グラフに変換することができる。

3.2 タプル被覆率

本節では、抽出したコアの効率性と安定性を評価する際の評価指標として用いる、グラフのタプル被覆率の定義を述べる。

3.2.1 効率性および安定性の評価指標

サービスチェイン要求において、隣接 VNF の組み合わせをタプルと呼称する。ここで、全てのサービスチェイン要求に含まれるタプルのうち、あるグラフを構成する有向辺と被覆するものの割合を、そのグラフのタプル被覆率と呼称する。タプル被覆率は、そのグラフをコアとしたときに、コアだけで処理可能なサービスチェイン要求の割合を反映し、タプル被覆率が $x\%$ であるならば $x\%$ のサービスチェイン要求を処理できるとみなす。NFV においてコアがもつ効率性は、コアを物理ネットワークに配置した場合にその部分だけで、確率的にサービスチェイン要求の何 % を処理することができるのかを評価指標として用いることで明らかにする。また、時間経過に伴ってサービスチェイン要求に含まれるタプルは追加・削除が発生し変化していく。コアがもつ安定性はサービスチェイン要求が変化した場合に、コアで処理できるサービスチェイン要求の割合がどのように変化するのかを評価指標として用いることで明らかにする。

なお、サービスチェインの場合と同様に AS パスにおける隣接 AS をタプルとみなせる。加えて、AS も同様に時間経過に伴ってタプルの追加・削除が行われる。

3.2.2 タプル被覆率の定義

まず、ある時刻 t における NFV のサービスチェインや AS のパスデータを変換した有向グラフを分解して得られたサブグラフを $\mathbf{DC}^t_{k,l}$ とする。また、$\mathbf{DC}^t_{k,l}$ を構成する有向辺の集合を $E(\mathbf{DC}^t_{k,l})$ で表す。一方で、ある時刻 t におけるサービスチェインまたはパスデータに含まれるタプルの集合を $L(t)$ とする。$L(t)$ については重複するタプルを含む。

ここで、時刻 t_1 におけるサブグラフ $\mathbf{DC}^{t_1}_{k,l}$、時刻 t_2 におけるタプル集合 $L(t_2)$ について、$L(t_2)$ のうち $E(\mathbf{DC}^{t_1}_{k,l})$ に含まれるタプルの集合 $LE(t_1, t_2, k, l)$ を式 (5) で定義する。

$$LE(k, l, t_1, t_2) = \{x | x \in L(t_2), x \in E(\mathbf{DC}^{t_1}_{k,l})\} \tag{5}$$

このとき、時刻 t_2 における $\mathbf{DC}^{t_1}_{k,l}$ のタプル被覆率 $A(k, l, t_1, t_2)$ を式 (6) で定義する。

$$A(k, l, t_1, t_2) = \frac{|LE(k, l, t_1, t_2)|}{|L(t_2)|} \tag{6}$$

式 (6) は時刻 t_2 のタプル集合 $L(t_2)$ のうち $E(\mathbf{DC}^{t_1}_{k,l})$ と被覆するタプルの割合を表す。

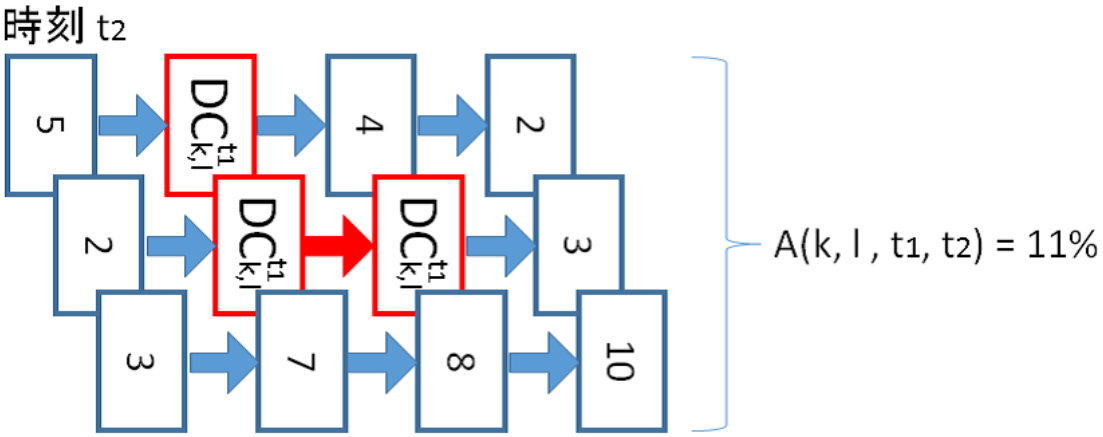

図 6: 時刻 t_2 における $\mathbf{DC}^{t_1}_{k,l}$ のタプル被覆率の例

図 6 は時刻 t_1 における NFV のサービスチェインや AS のパスデータを変換した有向グラフを分解して得られたサブグラフ $\mathbf{DC}^{t_1}_{k,l}$ の、時刻 t_2 におけるタプル被覆率を例示している。赤色で示す VNF または AS とタプルはそれぞれ $\mathbf{DC}^{t_1}_{k,l}$ を構成するものを表し、青色で示すものはそれ以外を表している。

3.2.3 リクエストに追加・削除されたタプルの割合の定義

安定性の評価を行う際にユーザからのリクエストがどの程度変化したのかを明らかにするため、ある時刻 t_1 から別の時刻 t_2 までの間に追加されたタプルの割合と削除されたタプルの割合を求める。これらの定義を以下に示す。

ある時刻 t_1 における全てのサービスチェイン要求またはパスデータに含まれるタプルの集合は $L(t_1)$ で表される。一方で別の時刻 t_2 におけるタプル集合は $L(t_2)$ で表される。$L(t_1)$ と $L(t_2)$ については重複するタプルを含む。このとき、時刻 t_1 から時刻 t_2 までの間に サービスチェイン要求または AS パスに対して追加されたタプルの割合を式 (7) で、削除されたタプルの割合を式 (8) で求める。

$$\frac{|L(t_2) - L(t_1)|}{|L(t_1)|} \tag{7}$$

$$\frac{|L(t_1) - L(t_2)|}{|L(t_1)|} \tag{8}$$

式 (7) と式 (8) の分子はそれぞれ差集合の要素数を表す。

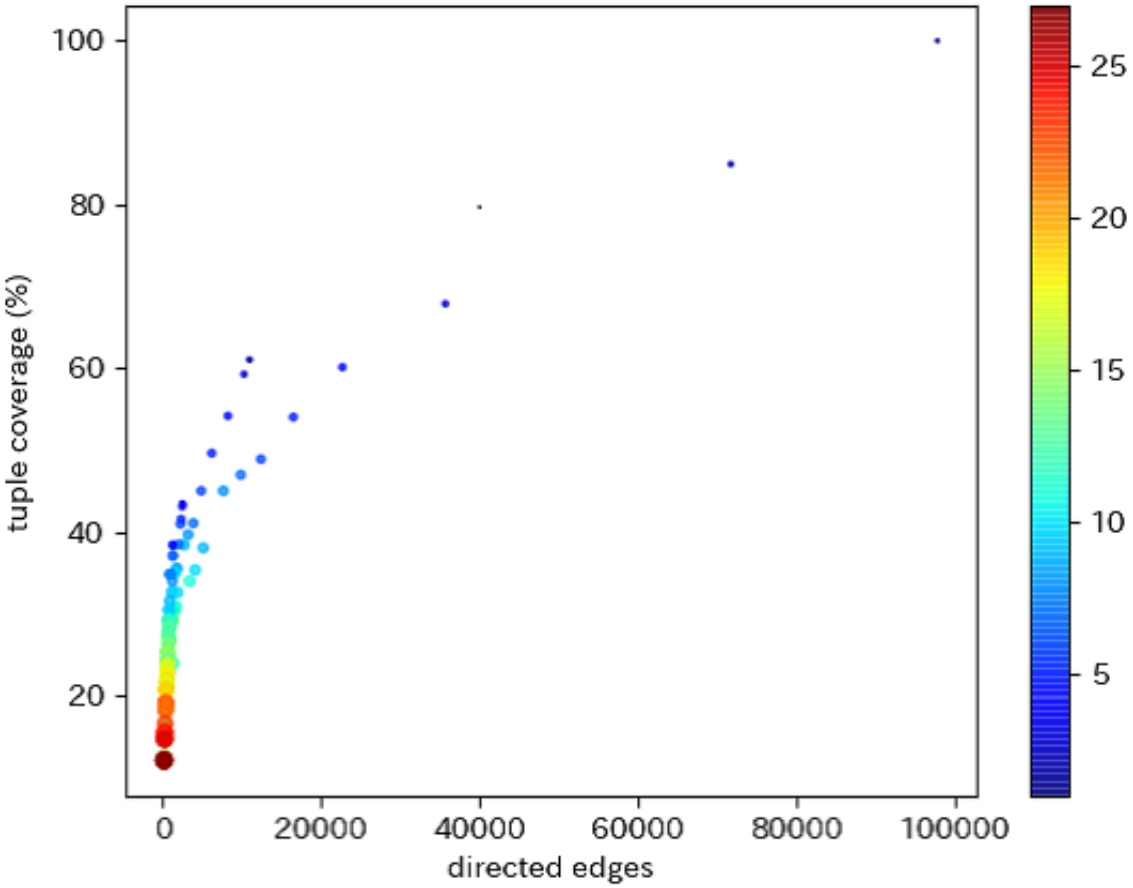

図 7: サブグラフごとの構成する有向辺の数とタプル被覆率

3.3 コアサイズに対するタプル被覆率の時間的変化

3.3.1 コアサイズとタプル被覆率の相関関係

ある時刻の AS パスデータから得られたサブグラフのその時刻におけるタプル被覆率の導出を行い、サブグラフを構成する有向辺の数とタプル被覆率がどのような相関関係にあるかを求める。ここでは、サブグラフとして 2014 年 1 月 15 日 12 時の AS パスデータを分解して得られたものを用いて、同時刻である 2014 年 1 月 15 日 12 時におけるタプル被覆率を導出している。

図 7 にサブグラフを構成する有向辺の数とタプル被覆率について散布図で表したものを示す。図の横軸にはサブグラフを構成する有向辺の数を、縦軸にはタプル被覆率を示している。また、プロットされたポイントのサイズはサブグラフ $\mathbf{DC}_{k,l}$ の $k+l$ を反映している。図の右側に位置するカラーマップについても同様に $k+l$ を反映している。図より、最も構

成する有向辺の数の少ないサブグラフが最も低いタプル被覆率を有していることがわかる。対して、最も構成する有向辺の数の多いサブグラフが最も高いタプル被覆率を有している。サブグラフを構成する有向辺の数とタプル被覆率について $y = x^{\frac{1}{2}}$ で近似される曲線状にプロットしている。したがって一定の大きさまではサブグラフのサイズに対して、高いタプル被覆率見込める。図 7 だと約 40% 付近まではサブグラフのサイズの増加に対するタプル被覆率の増加効率が高い。AS パスデータについてはサイズに対して高いタプル被覆率を有するサブグラフをコアとして抽出することで、そこに効率性を確認することができる。

3.3.2 時間経過に伴う AS パスとタプル被覆率の変化

時間経過に伴って AS パスデータに追加・削除されたタプルの割合とタプル被覆率の変化を示す。ここでは、サブグラフとして 2014 年 1 月 15 日 12 時の AS パスデータを分解して得られたもののうち、構成する有向辺の数と同時刻におけるタプル被覆率が最も低い $\mathbf{DC}_{13,14}$、構成する有向辺の数とタプル被覆率が最も高い $\mathbf{DC}_{0,1}$、タプル被覆率が 50% を超えるもののなかで最も構成する有向辺の数が少ない $\mathbf{DC}_{1,3}$ を用いている。また、AS パスデータとタプル被覆率の変化を求める期間は 2014 年 1 月 15 日 12 時 から、 2014 年 12 月 15 日 12 時 までの 1 ヵ月ごととしている。ここでは、 2014 年 1 月 15 日 12 時 を時刻 $t = 0$ とし、2014 年 12 月 15 日 12 時 を $t = 11$ とするまで順に時刻を割り当てる。

図 8 に、各時刻において追加・削除されたタプルの割合とタプル被覆率を示す。図の横軸の目盛りには時刻 t を示している。一方で縦軸の左側には各時刻におけるサブグラフのタプル被覆率を示している。縦軸の右側には 1 ヵ月前から追加・削除されたタプルの割合を示している。図より、 三つのサブグラフのなかで最も構成する有向辺の数が少ない $\mathbf{DC}^0_{13,14}$ は他のサブグラフと比べてタプル被覆率の減少が少なく、AS のパスの変化に対して安定的であることがよみとれる。AS パスデータに対しては 1 ヵ月ごとに前月からおおよそ 5 ∼ 10% ほどタプルの追加・削除が行われている。これに対して、$\mathbf{DC}^0_{1,3}$ では 1 年間で 5% 程、$\mathbf{DC}^0_{0,1}$ では 1 年間で 15% 程のタプル被覆率の減少がみられる。一方で、$\mathbf{DC}^0_{13,14}$ の 1 年間でのタプル被覆率の減少は 1% にも満たない程度に留まっている。また、約 20% と多くのタプルが追加された 8 月の 9 月の間においても、$\mathbf{DC}^0_{13,14}$ のタプルの減少は少なく、安定的であるといえる。

以上のことから、サブグラフのサイズが小さいほど AS のパスの変化に伴うタプル被覆率の変化が少なく、安定性が高いと考えられる。3.3.2 章もふまえると 3 章を用いた場合、効率性と安定性のトレードオフを考慮することによって AS パスデータからコアを抽出することが可能であると言える。

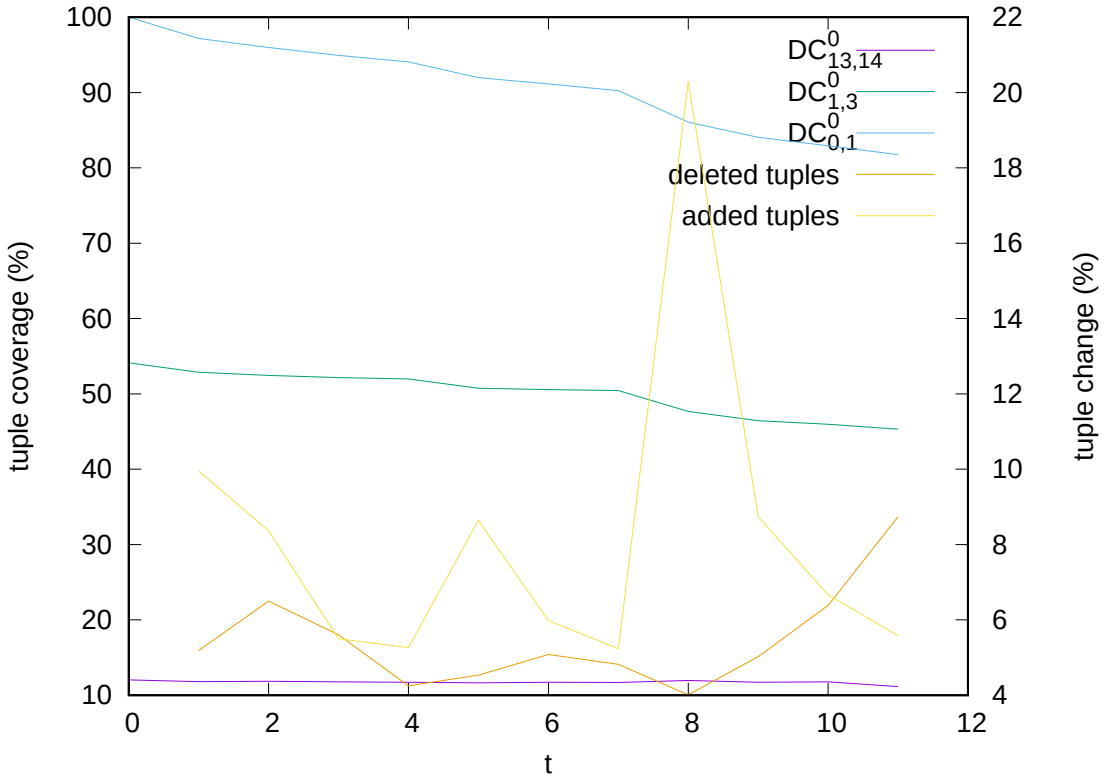

図 8: AS パス全体に追加・削除されたタプルの割合とタプル被覆率の変化

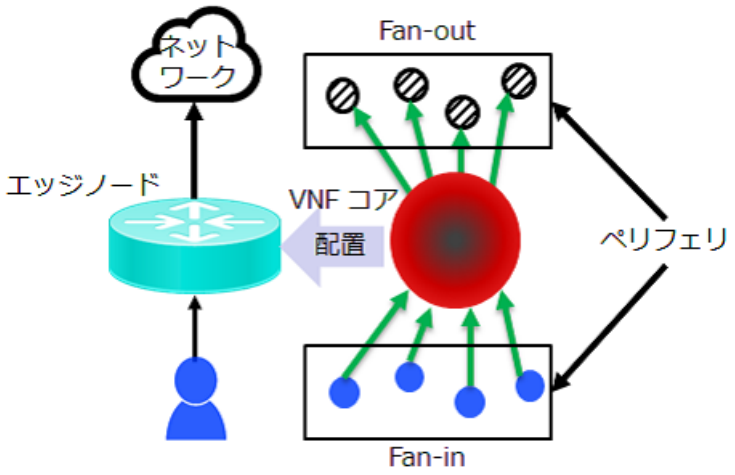

図 9: 想定する物理ネットワーク環境

4　コア抽出方法にもとづく VNF 配置の効率性及び安定性の評価

3 章で得られた定性的な特徴を踏まえてコアペリフェリモデルにもとづく VNF 配置の効率性及び安定性の評価を行う。

4.1　ネットワーク構成

物理ネットワーク環境は、図 9 に示すように他のネットワークに 1 台のエッジノードが接続されているものを想定する。このエッジノードに 2 章の方法を用いて抽出したコアを配置する。ユーザはこのエッジノードを介してリクエストを送信する。なお簡単化のため、ネットワーク間回線の帯域は考慮しないものとする。

4.2　サービスチェイン生成モデル

NFV におけるサービスチェインの実測データは存在しない。したがって、過去の VNF 配置問題を扱う研究では、サービスチェイン要求の生成モデルを導入している。

文献 [7] では、VNF に必要なコンポーネント数に焦点をあて、サービスチェイン内の VNF で必要とされるコンポーネント数をランダムに決定する生成モデルを用いていた。文献 [10] ではまず、サービスチェインに含まれる VNF 数を 2 ～ 7 個の範囲でランダムに決定し、要求される VNF を 5 種類の中からランダムに決定する生成モデルを用いている。これらの文献では、要求される VNF はランダムに定まるとしていた。しかし、実際には要求される

VNF には偏在性があり、サービスチェインへの登場頻度の高い VNF と低い VNF があることが予測される。文献 [11] では、配置された VNF で処理できるサービスチェイン要求のヒット率を向上させるために、要求される VNF の偏在性にもとづいた VNF 配置を提案している。それに伴い、要求される VNF の偏在性に基づいたサービスチェイン要求の生成モデルを用いている。本報告では、文献 [11] で用いられた生成モデルを使用してコアペリフェリモデルにもとづく VNF 配置の効率性および安定性の評価を行う。

文献 [11] で提案された生成モデルにおいて、ある一つのサービスチェイン要求は以下の手順で生成される。

step1 送信元と送信先のノードをそれぞれ決定する

step2 リクエストされる VNF 数を 10 ∼ 20 の範囲で一様にランダムで決定する

step3 リクエストされる各 VNF をジップの法則にしたがって決定する

文献 [11] の生成モデルでは、要求される VNF の偏在性をジップの法則を用いてモデル化している。そのため、生成の手順における **step3** ではジップの法則を用いてリクエストされる VNF を決定する。ジップの法則では要素の人気度と出現頻度をモデル化している。

ジップの法則において j 番目に人気な要素の出現頻度は式 (9) で示される。

$$p_j = \frac{\Omega}{j^{\alpha}} \tag{9}$$

ここで、α はパラメータ定数であり Ω は式 (10) で表される。

$$\Omega = 1/\sum_{i=1}^{N} \frac{1}{i^{\alpha}} \approx 1/\int_{1}^{N} \frac{1}{i^{\alpha}} di = \frac{1-\alpha}{N^{(1-\alpha)}-1} \tag{10}$$

式 (10) 内の N は VNF の総種類数である。ジップの法則に対して逆関数法を適用することで要求される VNF を決定することができる。

4.3 VNF コア抽出にもとづく VNF 配置

生成したサービスチェイン要求から抽出した VNF コアはユーザとクラウド間に配置されたエッジノード内の汎用サーバ上に配置するものとする。一方で、この汎用サーバ上に配置できる VNF 数には上限があるものとする。ここで、配置できる VNF 数の上限を x としたとき、エッジノードで VNF 間のリクエストを最大 $x(x-1)$ 種類処理可能となる。

エッジノードに配置する VNF コアは、サービスチェイン要求の有向グラフを分解して得られたサブグラフの中で、汎用サーバ上に配置できる VNF 数の上限を満たし最も多くのサービスチェイン要求を処理できるものを構成する VNF とする。

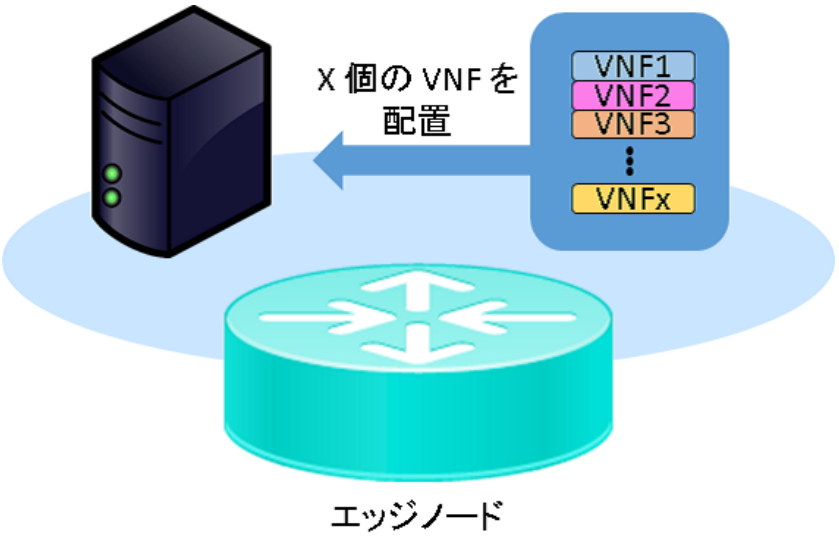

図 10: エッジノードへの VNF コアの配置

4.4 VNF 配置の効率性および安定性の評価

本節では生成したサービスチェイン要求から VNF コアを抽出し、エッジノードに配置した際の効率性および安定性の評価シナリオを述べ、評価シナリオに則り導出した評価結果を示す。

4.4.1 評価指標

エッジノードに配置可能な VNF 数の上限を x とした場合に、 VNF コアを配置することでサービスチェイン要求に含まれる VNF 間のリクエストをどの程度処理可能かを明らかにする。効率性および安定性の評価指標としてこれを用いる。エッジノードに配置した VFN コアで処理可能なリクエスト量はグラフのタプル被覆率を用いて示す。VNF コアを表すグラフは 2 章のメトリクスを用いて得られたサブグラフのなかで、エッジノードに配置可能な VNF の上限数 x 以下のノード数で構成され最も高いタプル被覆率を有するサブグラフを完全グラフ化したものである。これは VNF コアが得られたサブグラフのなかで、エッジノードに配置可能な VNF の上限数 x 以下のノード数で構成され最も高いタプル被覆率を有するものを構成する VNF から成り、これらの VNF 間のリクエストをすべて処理可能であるためである。VNF コアを表すグラフのタプル被覆率は、サービスチェイン要求に含まれる VNF 間のリクエストのうち、VNF コアを配置したエッジノードで処理可能なものの割合を示す。以後は VNF コアを表すグラフのタプル被覆率を単に VNF コアのタプル被覆率と

表 1: サービスチェイン要求の生成パラメータ: 効率性評価

パラメータ	値
サービスチェイン要求の本数	30000
サービスチェイン要求一本あたりに要求される VNF 数	$[10, 20]$
VNF の総種類数	30000
ジップの法則のパラメータ	$[0.1, 2.0]$

表現する。

4.4.2 効率性の評価シナリオ

エッジノードに配置可能な VNF 数の上限を x とした場合に、 VNF コアを配置することでサービスチェイン要求に含まれる VNF 間のリクエストをどの程度処理可能かを明らかにすることにより、 VNF コアの効率性を示す。

効率性の評価シナリオでは、表 1 に示すパラメータを用いて、ジップの法則のパラメータ α が $[0.1, 2.0]$ の範囲で 0.1 ずつ異なる 20 パターンのサービスチェイン要求を生成した。なお、生成した 20 パターンのサービスチェイン要求について α 以外のパラメータ設定は共通である。生成するサービスチェイン要求の本数は 30000 本とし、サービスチェイン要求一本あたりに要求される VNF 数は $[10, 20]$ の範囲で一様ランダムに決定している。また、要求される VNF の候補は 30000 種類に設定している。20 パターンのサービスチェイン要求についてエッジノードに配置できる VNF 数の上限を $128, 64, 32, ..., 1$ に設定したときに、これらの配置上限を満たす VNF コアのタプル被覆率を導出する。なお、タプル被覆率の導出に際して各 α ごとに 10 回ずつサービスチェイン要求を生成し、そこから得られたタプル被覆率の平均をとっている。ただし、エッジノードに配置可能な VNF 数の上限を x としたとき、サービスチェイン要求の有向グラフを分解して得られるサブグラフのなかで、構成するノード数が x 以下のものが存在せず、VNF コアの抽出を行えない場合がある。各 α ごとに 10 回ずつサービスチェイン要求を生成し、s 回分のサービスチェイン要求で VNF コアを抽出できた場合、$s \geq 5$ ならばそれらのタプル被覆率の平均を示す。$s < 5$ ならばその α では VNF コアの抽出を行えないとみなす。

図 11 に、各サービスチェイン要求から抽出した VNF コアのタプル被覆率を示す。図の横軸は各サービスチェイン要求の生成時に用いたパラメータ α を $[0.1, 2.0]$ の範囲で 0.1 ずつ示している。縦軸にはタプル被覆率を示している。図より、VNF コアのタプル被覆率は $\alpha = 0.1$ に近づくと 0% に収束し、$\alpha = 2.0$ に近づくと 100% に収束する。また、α が高ま

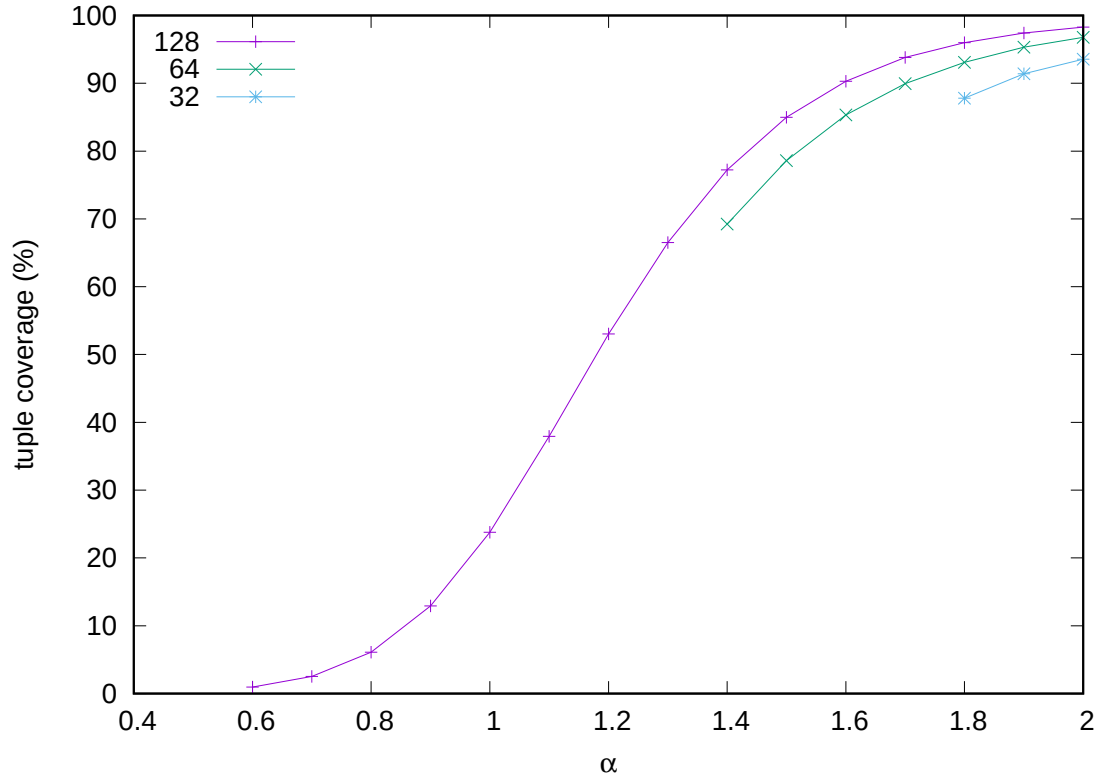

図 11: VNF コアのタプル被覆率

るほど、エッジノードに配置可能な VNF 数の上限がより少ない場合でも VNF コアの抽出が可能になる。VNF コアについて、VNF 数の上限が 128 以下の場合は $\alpha = 0.6$ から、64 以下の場合は $\alpha = 1.4$ から、 32 以下の場合は $\alpha = 1.8$ から抽出されていることが確認できる。これは α が高まると要求される VNF の偏在性が高くなり、より少数の VNF にリクエストが集中するためである。要求される VNF の偏在性が高いほど、より少ないエッジノードのリソースでより多くのリクエストを処理できるようになり、 VNF コアは高い効率性を示す。一方で、α が低下すると配置上限が少ない場合の VNF コアでは多くのリクエストを処理することが困難になり、効率性は減少する。$\alpha = 0.6$ において、エッジノードに配置できる VNF 数の上限を 128 とした場合の VNF コアが抽出されているが、そのタプル被覆率は 1% に満たない。要求される VNF の偏在性が低い場合により多くのリクエストを処理できるようにするためには、エッジノードに配置可能な VNF 数の上限を増やす等の設備投資が必要となる。

表 2: サービスチェイン要求の生成パラメータ: 安定性評価

パラメータ	値
サービスチェイン要求の本数	30000
サービスチェイン要求一本あたりに要求される VNF 数	$[10, 20]$
VNF の総種類数	30000
ジップの法則のパラメータ	1.0

4.4.3 安定性の評価シナリオ 1

ある時刻のサービスチェイン要求から抽出した VNF コアをエッジノードに配置し、時間経過でサービスチェイン要求が変化した場合に処理できるリクエスト量がどの程度変化するかを明らかにすることにより、 VNF コアの安定性を示す。そのために、3.2 章で定義したタプル被覆率の変化を導出する。また、時間経過でサービスチェイン要求に追加・削除されたタプルの割合を導出することでリクエストの変化を示す。

安定性の評価シナリオ 1 では、まず時刻 $t = 0$ において、表 2 に示すパラメータを用いて生成したサービスチェイン要求から VNF コアを抽出し、エッジノードに配置するものとする。生成したサービスチェイン要求の本数は 30000 本とし、サービスチェイン要求一本あたりに要求される VNF 数は $[10, 20]$ の範囲で一様ランダムに決定している。また、要求される VNF の候補は 30000 種類に設定し、ジップの法則のパラメータは $\alpha = 1.0$ としている。時間が 1 ステップ進むごとにサービスチェイン要求の本数と VNF の総種類数を 1.1 倍してサービスチェイン要求を生成しなおし、$t = 0$ において抽出した VNF コアのタプル被覆率を導出する。エッジノードに配置可能な VNF 数の上限は 128 としている。また、配置上限が 128 のときの VNF コアとの比較用に、配置上限が 4096, 8192 のときの VNF コアのタプル被覆率についても導出する。ただし、一つのエッジノードに 4096 個や 8192 個ものVNF を配置することはリソースコストの観点より困難かつ現実的ではないため、あくまで比較用であることに注意されたい。t の範囲は $[0, 10]$ とし、効率性の評価シナリオと同様にタプル被覆率の導出に際して各時刻ごとに 10 回ずつサービスチェイン要求を生成し、そこから得られたタプル被覆率の平均をとっている。

図 12 に時刻 $t = 0$ において抽出した VNF コアのタプル被覆率の時間変化を示す。図の横軸には時刻 t を、縦軸にはタプル被覆率を示している。図 12 より、配置上限が 128 のときの VNF コアは配置上限が 4096 および 8192 のときと比べてサービスチェイン要求の変化に伴うタプル被覆率の減少が少ないことがわかる。時間が 1 ステップ進むごとにもとのサービスチェイン要求から約 63 ~ 64% のタプルが削除され、新たに約 73 ~ 74% のタプルが追

加されていることがわかる。サービスチェイン要求の変化に伴い時刻 $t = 0$ において抽出した配置上限が 128 のときの VNF コアのタプル被覆率は約 $0.35 \sim 0.42\%$ ずつ減少する。一方で配置上限が 8192 のときの VNF コアは時刻 $t = 0$ から $t = 1$ の間にタプル被覆率が約 4.8% 減少し、それ以後は約 $1.1 \sim 1.4\%$ ずつ減少する。配置上限が 4096 のときの VNF コアは時刻 $t = 0$ から $t = 1$ の間にタプル被覆率が約 2.8% 減少し、それ以後は約 $0.91 \sim 1.2\%$ ずつ減少する。したがって、配置上限が 128 のときの VNF コアはタプル被覆率の減少量が比較的に少なく安定性が高い。

ここで、サービスチェイン要求の変化にあわせて VNF コアを抽出しなおしエッジノードに再配置を行うことを考える。図 13 に各時刻のサービスチェイン要求から新たに抽出した VNF コアのタプル被覆率を示す。それぞれのタプル被覆率は 10 回生成したサービスチェイン要求の平均をとっている。図のラインは $t = 0$ において抽出した VNF コアのタプル被覆率の変化を示し、ポイントは各時刻において生成しなおしたサービスチェイン要求から新たに抽出した VNF コアのタプル被覆率を示している。エッジノードに配置可能な VFN 数の上限は 128 としているまた、横軸には時刻 t を、縦軸にはタプル被覆率を示している。図 13 より、時刻 $t = 0$ において抽出した VNF コアと各時刻において新たに抽出した VNF コアとの間でタプル被覆率の差が少ないことがわかる。また、$t \geq 7$ については VNF 数の上限 128 を満たす VNF コアは存在していない。したがって、エッジノードに配置可能な VNF 数の上限を 128 に設定した場合、サービスチェイン要求の変化に合わせて VNF コアの再抽出を行い、エッジノードへの再配置を行ったとしてもタプル被覆率はあまり上昇しない。

サービスチェイン要求の変化に伴うタプル被覆率の減少量が少なく再配置の必要性もすくないことから、エッジノードに配置可能な VNF 数の上限を 128 としたときに時刻 $t = 0$ において抽出した VNF コアは高い安定性を有すると言える。一方でこのことは VNF 数の上限が 128 で固定の場合、図 12 でみられるようなタプル被覆率の減少は VNF コアの再抽出による再配置手法では改善されないことを示している。より多くのリクエストを処理する必要がある場合にはエッジノードに配置可能な VNF 数の上限を増やすといった工夫を要する。

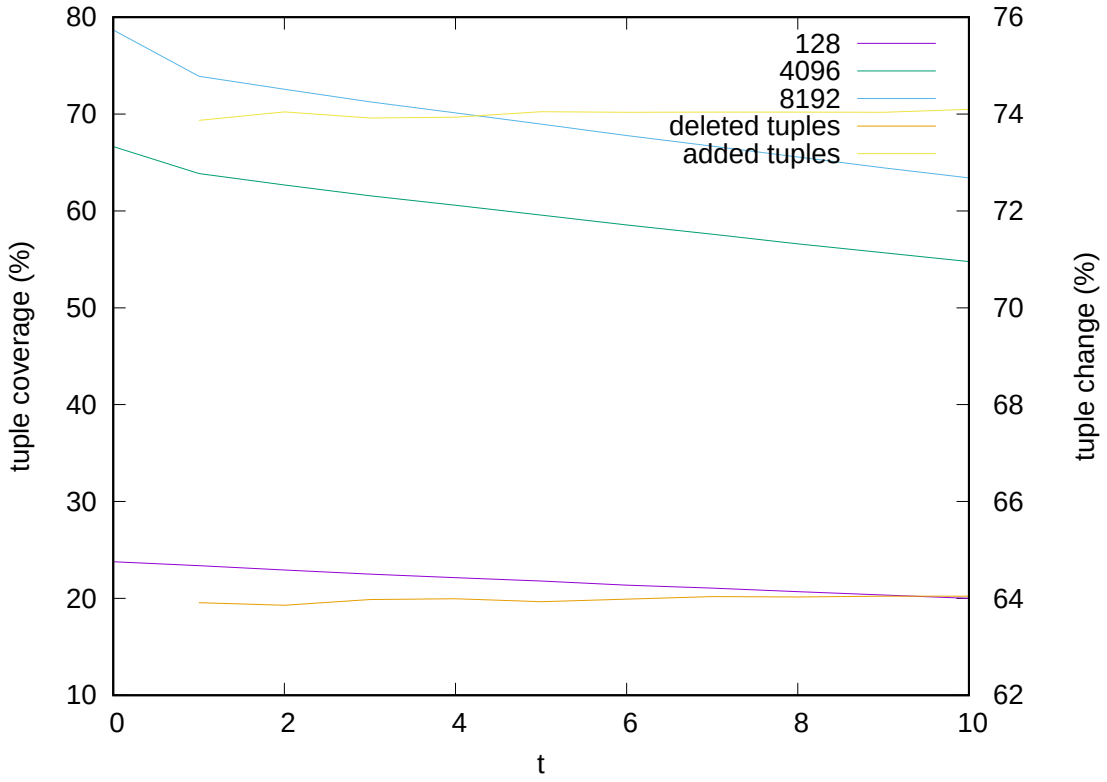

図 12: 時間変化に伴う VNF コアのタプル被覆率の変化: 安定性の評価シナリオ 1

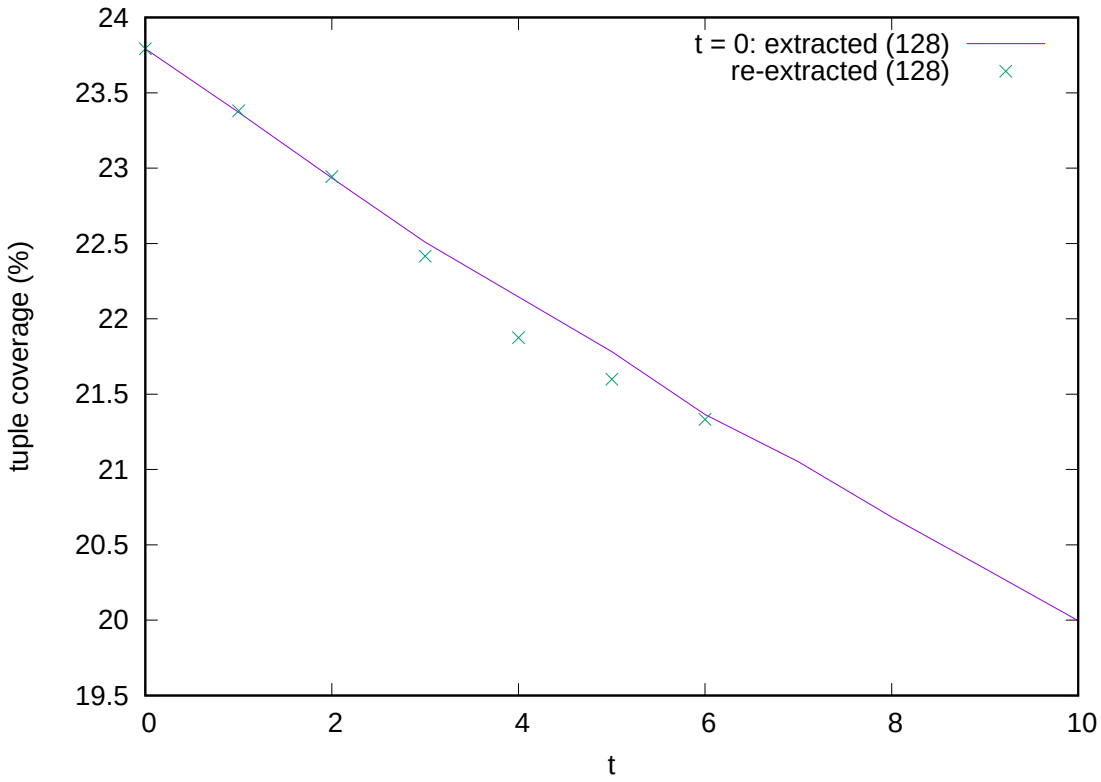

図 13: 各時刻のサービスチェイン要求から抽出した VNF コアのタプル被覆率: 安定性の評価シナリオ 1

4.4.4 安定性の評価シナリオ 2

安定性の評価シナリオ 2 では、まず時刻 $t = 0$ において評価シナリオ 1 と同様に表 2 に示すパラメータを用いて生成したサービスチェイン要求から VNF コアを抽出し、エッジノードに配置するものとする。時間が 1 ステップ進むごとに元のサービスチェイン要求に含まれるタプルの 10% に相当するタプルを含んだサービスチェイン要求新たに追加生成する。このとき、追加生成するサービスチェインについて、4.2 章で述べた生成モデルと異なり要求される VNF は一様な確率で決定されるものとする。エッジノードに配置可能な VNF 数の上限は 128 としている。また安定性の評価シナリオ 1 と同様に、配置上限が 128 のときの VNF コアとの比較用に、配置上限が 4096, 8192 のときの VNF コアのタプル被覆率についても導出する。なお、t の範囲は $[0, 10]$ であり、タプル被覆率の導出に際して各時刻ごとに 10 回ずつサービスチェイン要求を生成し、そこから得られたタプル被覆率の平均をとっている。

図 14 に時刻 $t = 0$ において抽出した VNF コアのタプル被覆率の時間変化を示す。図の横軸には時刻 t を、縦軸にはタプル被覆率を示している。図 14 より安定性の評価シナリオ 1 と比べ VNF コアのタプル被覆率の減少量が大きいことがわかる。10 ステップでのタプル被覆率の減少量を比べた場合、エッジノードに配置できる VNF 数の上限が 8192 のときは約 54%、4096 のときは約 40%、128 のときは約 15% 減少している。これは新たに要求されるようになった VNF が一様な確率で決定されていることにより、VNF コアで処理できるリクエストが相対的に減るためである。このような場合においても 配置上限が 128 のときの VNF コアは比較的安定的に動作することが確認できる。

一方で、安定性の評価シナリオ 1 と同様にサービスチェイン要求の変化にあわせて VNF コアを抽出しなおしエッジノードに再配置を行うことを考える。図 13 に各時刻のサービスチェイン要求から新たに抽出した VNF コアのタプル被覆率を示す。ここで、エッジノードに配置可能な VFN 数の上限が 128 の場合について、10 回生成したサービスチェイン要求のタプル被覆率の平均を導出している。図のラインは $t = 0$ において抽出した VNF コアのタプル被覆率の変化を示し、ポイントはそれぞれのサービスチェイン要求から新たに抽出した VNF コアのタプル被覆率を示している。また、横軸には時刻 t を、縦軸にはタプル被覆率を示している。図 13 より、安定性のシナリオ 1 と同様に VNF コアの再抽出を行った場合でもタプル被覆率の変化は少ない。

本項の評価シナリオのように、時間変化によって要求される VNF の偏在性が低下する場合においてもエッジノードに配置可能な VNF 数の上限を増やすことにより多くのリクエストを処理することが可能となる。しかし、このようなサービスチェイン要求に対して VNF コアの安定性は低下する。したがって、リクエストが変化した場合でも多くのリクエストを

処理できるようにするためには、変化にあわせた VNF 配置変更の頻度をあげるといった工夫を要する。

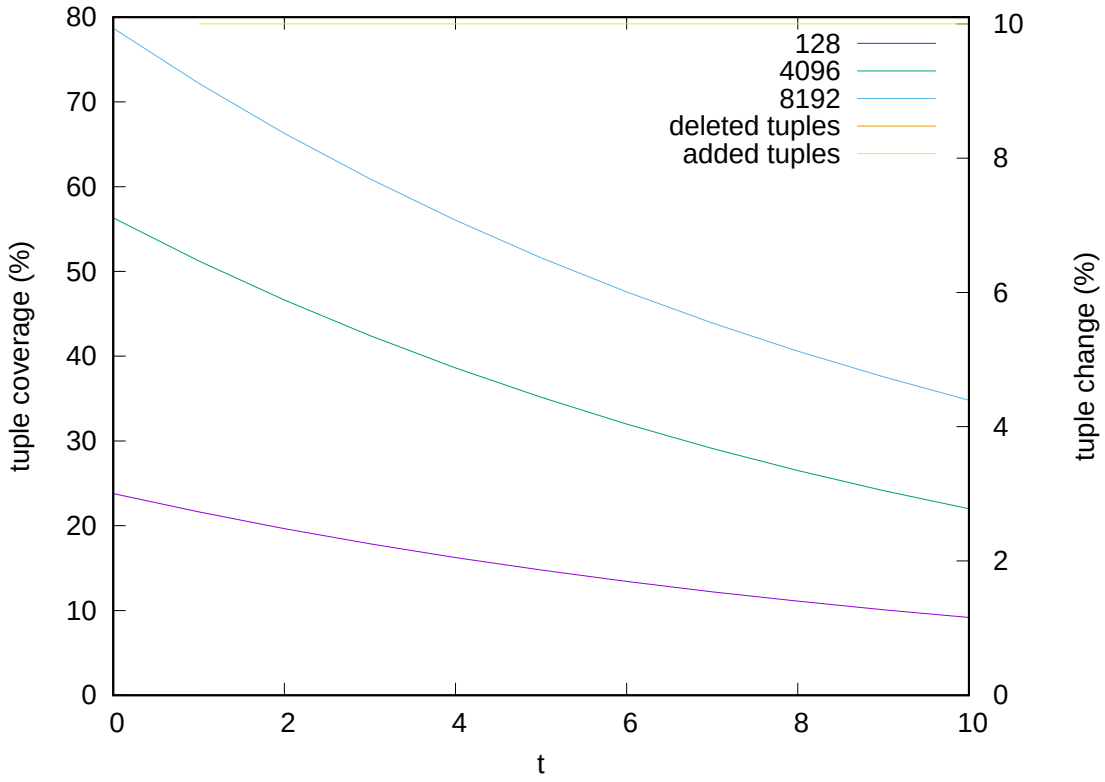

図 14: 時間変化に伴う VNF コアのタプル被覆率の変化: 安定性の評価シナリオ 2

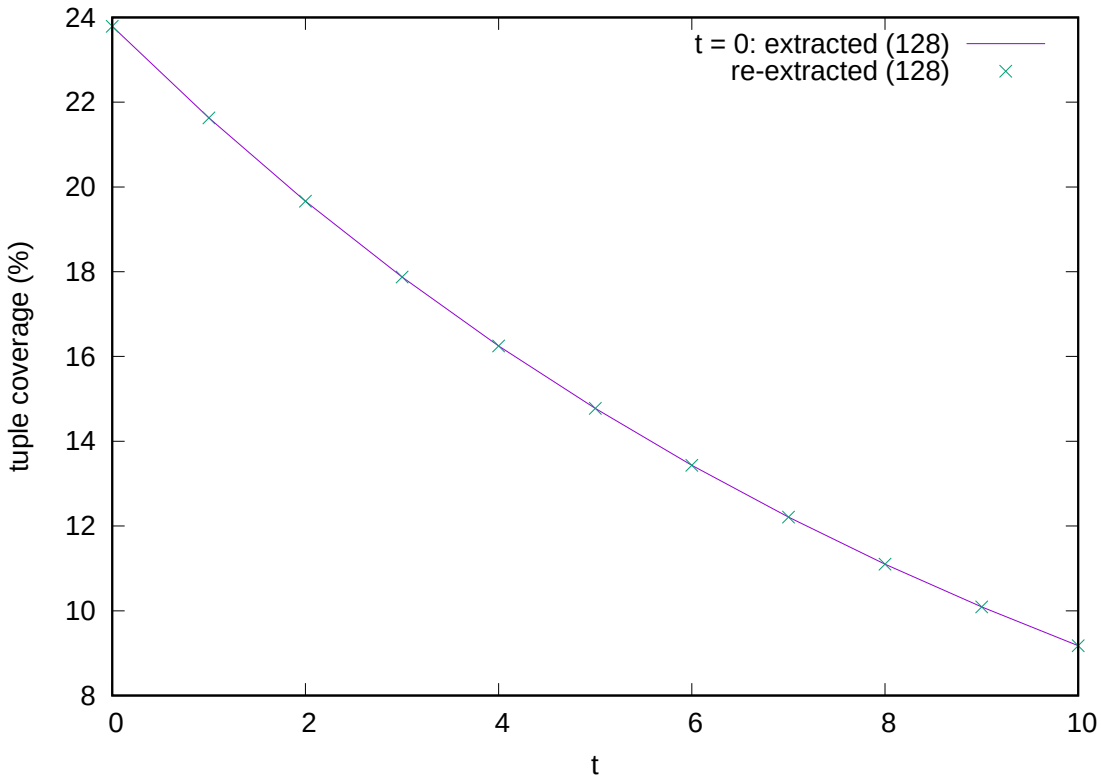

図 15: 各時刻のサービスチェイン要求から抽出した VNF コアのタプル被覆率: 安定性の評価シナリオ 2

4.4.5 α の変化に伴うサービスチェイン要求の変化

4.4.2 章の図 11 に α を $[0.1, 2.0]$ の範囲で 0.1 ずつ変化させたときに生成したサービスチェイン要求から VNF コアを抽出し、そのタプル被覆率を示した。本節ではサービスチェイン要求を $[0.1, 2.0]$ の α を用いて生成した後に、$[0.1, 2.0]$ の異なる α を用いて生成しなおした場合、どの程度のタプルが追加・削除されるのかを示す。なお、α 以外のパラメータについてはサービスチェイン要求の本数を 30000 本、要求される VNF の候補を 30000 種類に設定し、サービスチェイン要求一本あたりに要求される VNF 数は $[10, 20]$ の範囲で一様ランダムに決定している。

図 16 に追加されたタプルの割合を、図 17 に削除されたタプルの割合をそれぞれ示す。各図はサービスチェイン要求生成時のパラメータ α の変化に伴うタプルの変化をヒートマップで表している。図の縦軸は変化前の α の値、横軸は変化後の α の値である。図に示す追加・削除の割合は α の変化前と変化後でそれぞれ 10 回ずつサービスチェイン要求を生成し導出した平均をとっている。図より、変化後の α の値が小さいほど追加・削除されるタプルの割合は大きくなる。特に変化前と変化後のサービスチェイン要求について、一方でも $\alpha \leq 0.5$ を満たす場合 99% 以上のタプルが削除されている。これは要求される VNF の偏在性が非常に低いためであり、変化後にサービスチェイン要求の内容のほぼ全てが入れ替わることを示す。このため、VNF コアを用いて安定的にリクエストを処理することは困難となる。対して、変化前と変化後でともに α の値が大きくなると、要求される VNF の偏在性が高まり追加・削除されるタプルの割合はともに小さくなる。サービスチェイン要求の変化が小さくなるほど、VNF コアを用いて安定的にリクエストを処理することが可能となることは明らかである。

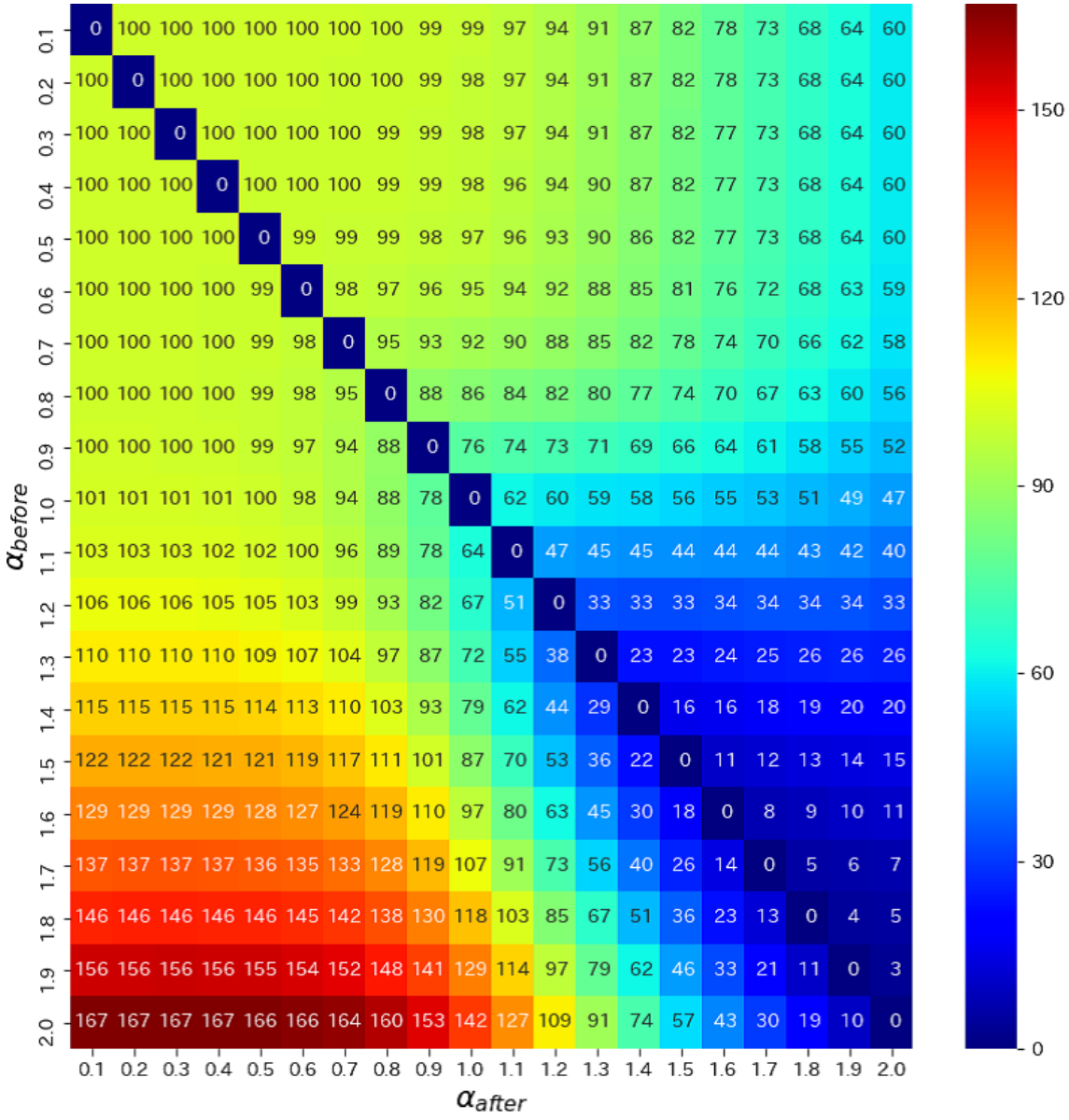

図 16: 追加されるタプルの割合: $0.1 \leq \alpha_{before} \leq 2.0$

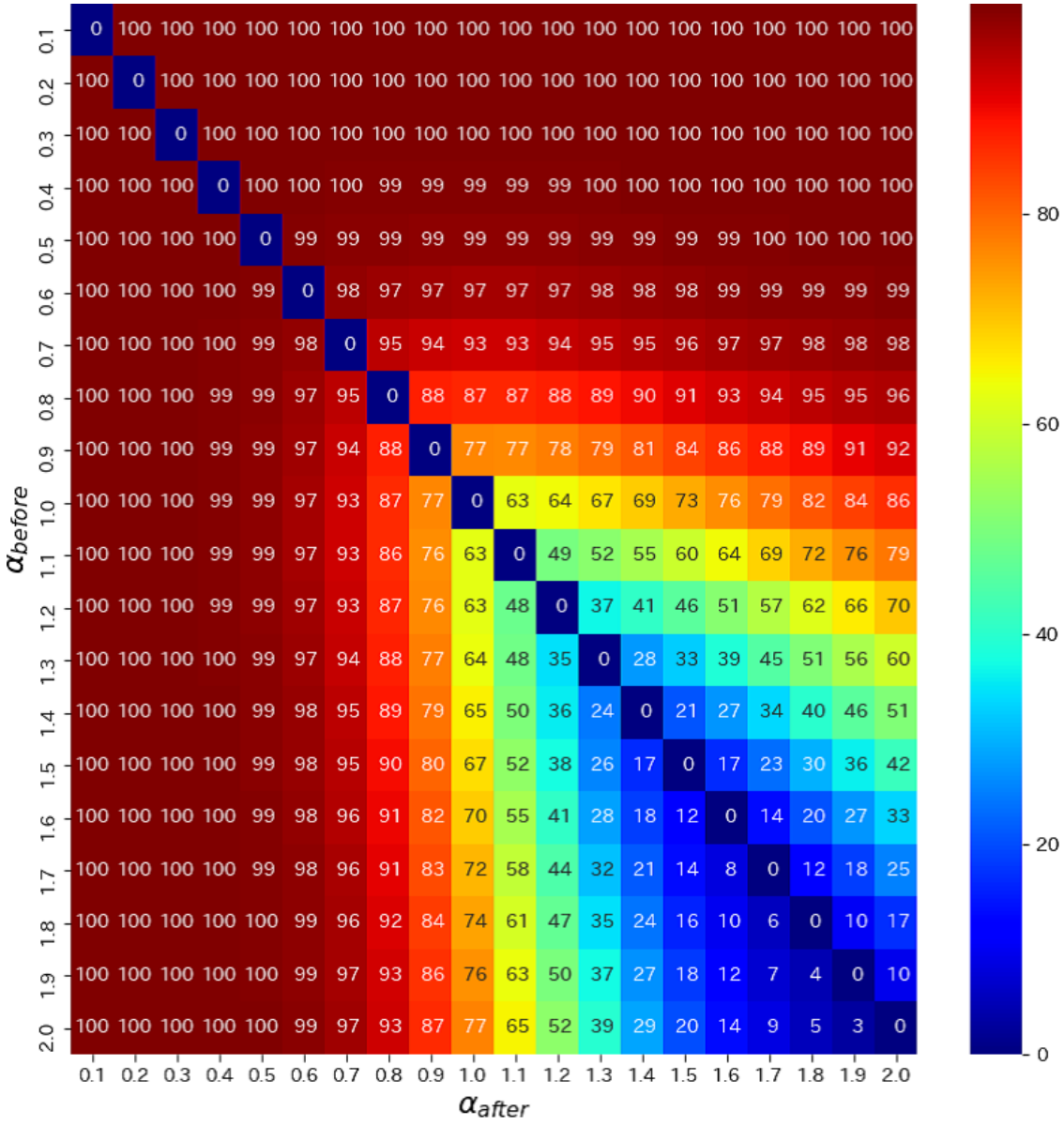

図 17: 削除されるタプルの割合: $0.1 \leq \alpha_{before} \leq 2.0$

5　おわりに

ネットワーク機能提供のコスト効率性と柔軟性を高めることを可能とする NFV だが、その運用にあたって解決すべき問題のひとつに VNF の配置問題がある。本報告では NFV にコアペリフェリモデルを適用することで VNF を物理ネットワークに配置した際の効率性と安定性の向上を図った。NFV から VNF コアを抽出するために、サービスチェイン要求を有向グラフに変換し、それに対してサブグラフ分解メトリクスを用いた。サービスチェイン要求の標準化は策定段階にあることから実測データを用いた評価を行えないため、AS パスデータを用いてコア抽出の適用性の評価をまず行った。その後、ジップの法則にもとづいた生成モデルを用いてサービスチェイン要求を生成し、VNF コアの効率性と安定性を明らかにした。ただし、効率性と安定性はともに VNF の配置リソースとサービスチェイン要求の影響を受ける。要求される VNF の偏在性が高い場合は配置する VNF コアを少数に限定しても高い効率性と安定性を期待することができる。対して、要求される VNF の偏在性が低くリクエストがランダムに分散する場合は、配置リソースを増やしつつ配置変更の頻度をあげるといった工夫が必要となる。

今後の課題としては、異なる生成モデルを用いてサービスチェイン要求を生成した場合に抽出したコアの効率性と安定性を明らかにし本報告で得られた結果との比較を行うこと、また、既存の評価手法を用いて他の VNF 配置手法との比較を行うことがあげられる。

謝辞

本報告を終えるにあたり、ご多忙の中、日頃より熱心にご指導、ご教授いただきました大阪大学大学院情報科学研究科の村田正幸教授に、心より深く感謝を申し上げます。ならびに、本報告の作成にあたって、ご多忙の中、多くの時間を割いてご指導いただき方向性を示していただいた大阪大学大学院情報科学研究科の荒川伸一准教授に厚く心よりお礼申し上げます。また、平素よりご助言をいただきました大阪大学大学院情報科学研究科の大下裕一助教、大阪大学大学院経済学研究科の小南大智助教に感謝を申し上げます。最後に日頃より様々な面でご助言、ご協力いただきました大場斗士彦氏、井上昴輝氏、坂本昂輝氏、佐竹幸大氏をはじめとする研究室の皆様に感謝いたします。

参考文献

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, pp. 2347–2376, Jun. 2015.

[2] "Network Functions Virtualisation Introductory White Paper," Oct. 2012.

[3] "Network Functions Virtualisation - White Paper #3," Oct. 2014.

[4] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, pp. 90–97, Feb. 2015.

[5] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proceedings of IEEE INFOCOM 2009*, pp. 783–791, Apr. 2009.

[6] P. Csermely, A. London, L.-Y. Wu, and B. Uzzi, "Structure and dynamics of core/periphery networks," *Journal of Complex Networks*, vol. 1, pp. 93–123, Oct. 2013.

[7] M. Otokura, K. Leibnitz, T. Shimokawa, and M. Murata, "Evolutionary core-periphery structure and its application to network function virtualization," *IEICE Nonlinear Theory and Its Applications*, vol. 7, pp. 202–216, Apr. 2016.

[8] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis, "D-cores: measuring collaboration of directed graphs based on degeneracy," *Knowledge and Information Systems*, vol. 35, pp. 311–343, May 2013.

[9] Y. Nakata, S. Arakawa, and M. Murata, "Analyzing the evolution and the future of the internet topology focusing on flow hierarchy," *Journal of Computer Networks and Communications*, vol. 113, pp. 13–18, 2015.

[10] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo, "Towards making network function virtualization a cloud computing service," in *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 89–97, May 2015.

[11] Y. Nam, S. Song, and J.-M. Chung, "Clustered NFV service chaining optimization in mobile edge clouds," *IEEE Communications Letters*, vol. 21, pp. 350–353, Oct. 2017.