社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

進化適応性を備えた仮想ネットワーク機能配置手法の提案と評価

乙倉　麻里[†]　ライプニッツ賢治[††,†]　小泉　佑揮[†]　小南　大智[†††]　下川　哲也[††,††††]
村田　正幸[†]

† 大阪大学 大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1–5
†† NICT/大阪大学 CiNet　〒 565–0871 大阪府吹田市山田丘 1–4
††† 大阪大学 経済学研究科　〒 560–0043　大阪府豊中市待兼山町 1–7
†††† 大阪大学 大学院生命機能研究科　〒 565–0871 大阪府吹田市山田丘 1–3
E-mail: †{m-otokura,ykoizumi,murata}@ist.osaka-u.ac.jp, ††leibnitz@nict.go.jp,
†††d-kominami@econ.osaka-u.ac.jp, ††††simokawa@fbs.osaka-u.ac.jp

あらまし　ネットワーク機能仮想化（NFV）はネットワーク機能の柔軟な提供を可能とし、動的で多種多様な通信ネットワークサービス提供のために必須の技術である。仮想化されたネットワーク機能（VNF）を物理ネットワークに動的に配置する動的 VNF 配置問題の重要な目的として、到着した要求を素早く収容することが挙げられる。この目的を達成するために、我々は生物の環境変動の中での進化の知見 (Modularly Varying Goals; MVG) を応用する。本稿では、MVG を動的 VNF 配置問題に応用した手法である Evolvable VNF Placement（EvoVNFP）を提案する。シミュレーション評価により、EvoVNFP を利用することで計算時間を短縮し動的な要求変動に追随して制御を行うことが可能であり、時間内の配置生成の失敗率を最大約 50%低減させることが示された。
キーワード　ネットワーク機能仮想化；サービスファンクションチェイニング；進化的アルゴリズム

Application of Evolutionary Mechanism to Dynamic Virtual Network Function Placement

Mari OTOKURA[†], Kenji LEIBNITZ[††,†], Yuki KOIZUMI[†], Daichi KOMINAMI[†††],
Tetsuya SHIMOKAWA[††,††††], and Masayuki MURATA[†]

Abstract　Recently, communication network services have become increasingly diverse and dynamic. The important goals of the dynamic VNF placement problem include accommodating new requests following the dynamics and reducing the time to calculate solutions. To tackle this problem, we utilize the concept of Modularly Varying Goals (MVG), which is based on a genetic algorithm (GA) and generates solutions that can easily adapt to time-varying goals in short time. In this paper, we propose Evolvable VNF Placement (EvoVNFP) that applies the concept of MVG to the dynamic VNF placement problem to reduce the time to obtain solutions. Results from numerical evaluations show that our method is able to better follow the dynamics of VNF requests and also reduce time until adapting to successive objectives.
Key words　Network Function Virtualization; Service Function Chaining; Evolutionary Algorithm

1. はじめに

近年、ユーザに提供される通信サービスが動的で多種多様となっており、サービス提供のために必要なネットワーク機能（ファイアウォールや侵入検知システムなど）も動的で多種多様となることが予想される。今後の IoT の実現を考慮すると、この傾向はますます強くなることが予想される。従来通信サービスプロバイダは、ハードウェアで実装されたネットワーク機能を用いてサービスを提供してきた。しかしながらこの提供手法では、サービスを新たに提供する際に多くの設備投資（CAPEX）と運用維持費（OPEX）が必要となる。

ネットワーク機能仮想化（Network Function Virtualization; NFV）は、ネットワーク機能の柔軟な提供を可能にする技術である [1] 。NFV では、ネットワーク機能と物理リソースを分離

する。NFV におけるネットワーク機能は、仮想マシン（VM）上で動作するソフトウェア（Virtual Network Function; VNF）として実装される。VM は物理ネットワーク内の物理マシン（PM）へ配置される。NFV により、ネットワーク機能の性能を動的に変更することや、ネットワーク機能を鎖状に接続したもの（チェーン）を 1 つのサービスとして提供することが可能となる（Service Function Chaining; SFC）[2]。

多くの研究者が、VNF の物理ネットワークへの配置を特定の目標に従って決定する問題である VNF 配置問題に取り組んできた [3] 。VNF 配置問題は、NP 困難な問題として知られる施設配置問題の一種である。また、ETSI NFV ISG specification [4] では、VNF の性能をより細かく制御するために VNF を細分化してコンポーネントとして提供することが提案されており、これを考慮すると VNF 配置問題がさらに複雑化する。さらに、SFC においてはユーザからの VNF チェーン要求が動的に到着/離脱することを考慮すると、VNF 配置問題を動的に解く必要がある [5]。VNF 配置問題に関しては多くの既存研究が存在するが、動的 VNF 配置問題に取り組んだものはほとんど存在しない。動的 VNF 配置問題では、現在の配置を資源利用率などの面で最適化することに加えて、サービスを安定的に提供することが求められる。このため、配置を計算するための時間が短いことが求められる。また、新しい配置への変更を行っている間はネットワークの帯域が圧迫され、場合によってはサービスの停止が必要となるため、要求到着/離脱に伴う配置の変更時に必要な操作（VM マイグレーション等）が少ないことが求められる。この問題を扱うための直観的な手法として、要求到着/離脱の度に最適化問題を解き直す手法が挙げられる。しかし、VNF 配置問題は NP 困難な問題であり、配置の計算に膨大な時間が必要となることが予想されるため、適用は困難である。

この問題に対処するために、本稿では生物の進化の知見を利用する。生物は、環境変動の中で進化したために環境変動への適応性を有するようになったという説がある。文献 [6] は、これを進化的アルゴリズム（EA）を用いて検証し、Modularly Varying Goals（MVG）の概念を提唱している。MVG とは、規則性を持った目標変動のことである。MVG を導入した EA により進化した個体は目標変動に対する適応性を有することがすでに示され [6]、また MVG の導入によって進化速度（進化に必要な世代数）を促進可能であることも示されている [7]。我々はこれまでに、Evolutionary Varying Goals（EVG）を提案し、MVG の概念が動的 VNF 配置問題に適用可能であることを示した [8]。EVG は、環境変動に伴う目標変動時に集団の初期化を行わないことで、配置生成に必要な世代数と配置変更に必要な操作数を削減する手法である。

本研究では、EVG を MVG の知見に基づき再度改良した手法である Evoluvable VNF Placement（EvoVNFP）を提案する。またその手法を、文献 [8] のものよりもさらに現実的な動的 VNF 配置問題に適用する。EvoVNFP では、目標に適応する解の生成に必要な時間を短縮するために、要求変動の周期より短い規則的な目標変動を行う。シミュレーション評価により、EvoVNFP は新しい目標へ適応する解の生成に必要な時間を短縮し、要求の動的変動に追随可能であることが示された。

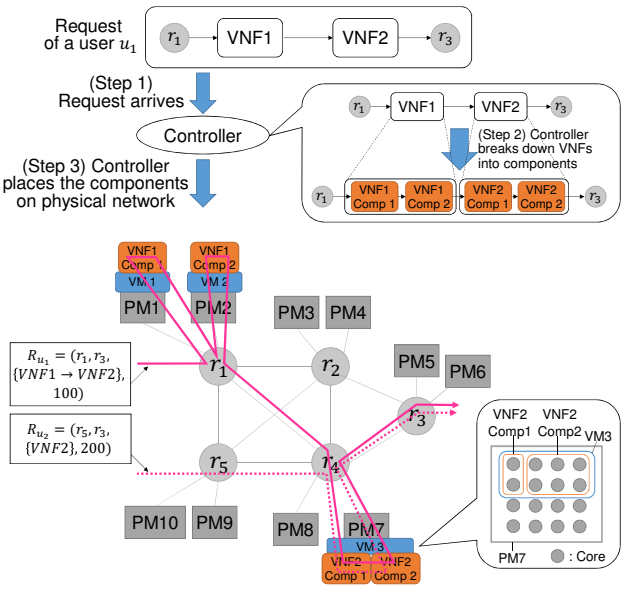


図 1: NFV システムモデルの概念図

本稿の構成は以下の通りである。2 章では NFV システムのモデルを説明し、問題の定式化を行う。3 章では提案手法 EvoVNFP について説明する。4 章では提案手法の性能評価を行う。最後に 5 章でまとめと今後の課題について述べる。

2. NFV システムモデルと問題の定式化

本章ではまず本稿で想定する SFC を前提とした NFV システムモデルを示し、それを用いて VNF 配置問題を定式化する。

2.1 NFV システムモデルの概要

本稿で想定する NFV システムモデルの概略図を図 1 に示す。本システムでは、ユーザが送信した VNF チェーン要求がシステムコントローラへ到着し（Step 1）、コントローラはチェーン中の VNF をコンポーネント（Comp）に分割し（Step 2）、そのチェーン（コンポーネント群）を物理ネットワーク中の物理計算機（Physical Machine; PM）に配置する（Step 3）。ユーザのトラフィックは物理ネットワークのイングレスルータから物理ネットワークに流入し、要求したチェーン（コンポーネント群）を経由し、エグレスルータから外部へと流出する。トラフィックがコンポーネントを通過するまでに伝搬遅延とキューイング遅延の 2 種類の遅延が発生する。伝搬遅延は、トラフィックがリンクを通過するときに発生する。キューイング遅延は、トラフィックがルータやコンポーネントなどのネットワーク機器によって処理されるときに発生する。本稿では、計算資源として CPU コアのみを考える。VM には PM の CPU コアが 1 つ以上割り当てられ、コンポーネントには VM に割り当てられている CPU コアが 1 つ以上割り当てられる。図 1 の右下にコア割り当ての概念図を示す。1 つの VM に複数のコンポーネントを配置可能であり、同様に 1 つの PM に複数の VM を配置可能である。

2.2 コンポーネントの性能に関する定義

コンポーネントに割り当てられたコア数と処理能力を関係づ

けるため、1 コアの性能、すなわち処理可能な命令量を処理量と定義し、1 コアが 1 秒間で処理可能な最大処理量を処理能力と定義する。n コアが割り当てられたコンポーネントまたは VM の処理能力は $n \cdot C$（C は処理能力）となると想定する。

2.3 要求フローの定義

コントローラに要求を送信したユーザのトラフィックフローのことを要求フローと定義する。ユーザ u の要求フロー R_u は、$R_u = (src_u, dst_u, chain_u, b_u)$ と表現される。src_u はイングレスルータ、dst_u はエグレスルータ、$chain_u$ は VNF チェーン、 b_u は要求伝送速度を表す。要求フローの例を図 1 の R_{u_1} と R_{u_2} に示す。複数のチェーンが同じ VNF を含む場合は、その VNF のコンポーネントが共有される可能性がある。

2.4 配置の遅延の計算方法

以下では配置の遅延の計算方法を解説する。本稿では、キューイング遅延は要求フローがルータとコンポーネントを通過した際に生じると仮定し、キューイング遅延を理論的に導出する。キューイング遅延は、ルータやコンポーネントの負荷やコンポーネントの性能に依存する。一方、伝搬遅延は各リンクに定数で設定され、要求フローがリンクを通過するときに発生すると仮定する。ユーザ u の要求フロー R_u で生じるキューイング遅延と伝搬遅延の和を要求フロー遅延 t_u と定義する。

キューイング遅延を理論的に導出するため、ルータとコンポーネントは M/M/1 待ち行列システムであると仮定し、ルータおよびコンポーネントへのトラフィック到着率を次のように設定する。ルータ r のサービス率は定数 M_r [packet/s] であり、コンポーネント j のサービス率は $\mu_{k,j,a} = (n_{k,j,a} \cdot C)/T_a$ [packet/s] （$n_{k,j,a}$ は VM k に配置された VNF a のコンポーネント j に割り当てられたコア数、T_a は VNF a のコンポーネントがパケット 1 つを処理するために必要な処理量）である。ルータ r へのトラフィック到着率は $\lambda_r = \sum_{r'} p^u_{(r,r')} \cdot b_u$ [bit/s]（$p^u_{(r,r')}$ は要求フロー R_u がルータ r、r' 間のリンクを通過する場合 1 、通過しない場合 0 となる変数） である。コンポーネント j へのトラフィック到着率 v_a [bit/s] は、VNF a を含む要求フロー全ての要求伝送速度の和である。

配置の遅延 $\hat{d}$ は、各要求フローの遅延を要求伝送速度で重み付けした平均、すなわち次式で計算される。

$$\hat{d} = \sum_u \frac{b_u}{\sum_u b_u} \cdot t_u.$$

2.5 問題の定式化

本稿で扱う VNF 配置問題は、以下のように定式化される。

$$\text{minimize} \quad \hat{d} + W \cdot \sum_{i,k} m_{i,k} \tag{1}$$

$$\text{subject to} \quad T_a \cdot \frac{v_a}{S} \leqq n_{k,j,a} \cdot C \qquad \forall k, j, a \tag{2}$$

$$\sum_k m_{i,k} \leqq N_i \qquad \forall i \tag{3}$$

$$\sum_{j,a} n_{k,j,a} \leqq m_{i,k} \qquad \forall k, i \tag{4}$$

$$\text{variables} \quad m_{i,k},\ n_{k,j,a},\ p^u_{(r,r')}$$

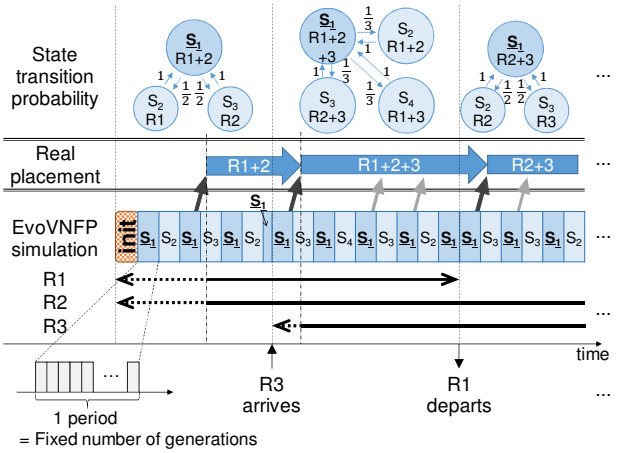


図 2: EvoVNFP の実行の概念図

$m_{i,k}$ は VM k に割り当てられている PM i のコア数を表す。W は重み付け係数である。N_i は PM i のコア数を表す。S は 1 パケットのサイズを表す。式 (2) は、各コンポーネントにはそれを通過する全要求フローを処理するために必要な処理能力を持つために必要なコア数が割り当てられていなければならないことを表す。式 (3) は、ある PM に配置された全 VM に割り当てられたコア数の合計は、その PM のコア数を超過してはならないことを表す。式 (4) は、ある VM に配置された全コンポーネントに割り当てられたコア数の合計は、その VM に割り当てられたコア数を超過してはならないことを表す。

3. 提 案 手 法

本稿では動的 VNF 配置手法 Evolvable VNF Placement（EvoVNFP）を提案する。

3.1 EvoVNFP の概要

EvoVNFP は EVG [8] を拡張した手法である。EVG の特徴は、要求が到着/離脱し EA の目標を変更する際に、集団の初期化を行わないことである。EvoVNFP の実行の概念図を図 2 に示す。EvoVNFP は、システム中に 2 つ以上の要求フローが存在する場合 EVG と異なる挙動となる。この時 EvoVNFP は、図 2 中の「EvoVNFP simulation」のように、EA の目標を周期的に変動させながら実行する。目標の周期的な変動の間隔をピリオドと定義する。ピリオドごとの目標変動は文献 [7] の MVG に着想を得たものであり、現在の目標に対する解を生成するまでの時間を短縮するためのものである。ピリオドごとの変動の状態遷移図を図 2 の上部に示す。初期状態は S_1 であり、これは EvoVNFP がシステム中に配置すべき全ての要求フローを考慮した目標を用いている場合を示す。使用中の配置状態図中の「real placements」）の更新タイミングは次の 2 種類である。1 つは、図 2 の「real placements」と「EvoVNFP simulation」の間の黒矢印で表されている、目標変更後新しい目標に対する許容解を初めて得たタイミングである。もう 1 つは、同様に灰色の矢印で表されている、 1 つ前の更新時の配置と比べて一定割合良い配置状態が生成された場合である。図 2 中の破線は、新しい要求を収容するために必要な時間を表す。これは配置を計算する時間と配置を変更する時間の和である。

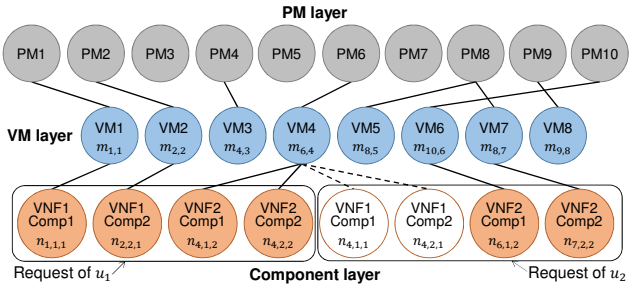


図 3: 個体の一例

3.2 EA の設計

EvoVNFP で用いている EA の設計について説明する。

3.2.1 個体の設計

EvoVNFP で用いる EA の個体の一例を図 3 に示す。個体は固定長の 3 層ネットワークであり、各個体が 1 つの配置状態を表す。最上位層、中間層、最下位層のノードはそれぞれ、PM、VM、コンポーネントを表す。VM 層と PM 層のノードは、対応する VM が対応する PM に配置されているときに接続される。同様に、コンポーネント層と VM 層のノードは、対応するコンポーネントが対応する VM に配置されているときに接続される。さらに、VM 層とコンポーネント層の各ノードは、対応する VM とコンポーネントに割り当てられたコア数の情報も保持している。物理ネットワーク内での各要求フローの経路は、必要なコンポーネントを接続する最短経路となる。

コンポーネント層のノードと接続されていない VM 層のノードは、その個体が配置状態に変換される際には無視される。また、図 3 では白いノードと破線で表されている、コンポーネント層の使われていないノードやリンクも、その個体が配置状態に変換される際には無視される。

3.3 適応度関数の設計

適応度関数とは、その個体が目標にどれだけ適応しているかを評価するための関数である。EvoVNFP の EA の適応度関数を式 (5a) および (5b) に示す。

$$F = \begin{cases} \left(\dfrac{\hat{d}}{d_{max}} + \dfrac{W(\sum_{i,k} m_{i,k})}{c_{max}} \right)^{-1} & \text{if the individual meets (2)–(4)} \quad (5a) \\ \alpha Z & \text{otherwise} \quad (5b) \end{cases}$$

d_{max} は遅延の基準値、c_{max} はコア数の最大値であり、正規化のために用いられている。本稿では、あるチェーンの遅延の基準値を、（そのチェーンの長さ＋3）の長さのチェーンの、チェーン上の全コンポーネントが使用率 80%で動作する時の遅延であるとする。Z は個体中の制約条件に違反している要素の数を表す。α は負の定数である。式 (5a) は、個体が表す配置状態が制約条件 (2)〜(4) を満たす、すなわち評価中の個体が配置状態へ変換可能である（許容解である）場合のみ使用される。式 (1) の値が小さいほど、式 (5a) の値は大きくなる。評価中の個体が配置状態へ変換可能でない場合は、式 (5b) が適応度関数として用いられる。個体中に制約条件に違反する要素数が多いほど、式 (5b) の値は低下する。

3.4 突然変異の設計

EvoVNFP 中の EA では、突然変異のみを用い、交叉を用いない。突然変異ステップでは、集団中のエリート以外の全ての個体に対し、あらかじめ定められた突然変異率で突然変異を行うかを決定する。EvoVNFP では、突然変異を行う場合は、PM 層と VM 層間のリンクをランダムに張り替える、VM 層とコンポーネント層間のリンクをランダムに張り替える、VM 層のノードのコア数をランダムに変更する、コンポーネント層のノードのコア数をランダムに変更する、の 4 操作のうち 1 つをランダムに選択し実行する。

4. シミュレーション評価

本章では、シミュレーション評価の設定、比較手法、評価指標について解説した後、評価結果を解説する。

4.1 シミュレーション設定

物理ネットワークとして、図 1 のような、5 つのルータが存在し各ルータに 2 つの PM が接続するネットワークを想定する。各 PM は同じ処理能力の CPU コア 16 個を搭載する。ルータ間の物理リンクで生じる伝搬遅延は 20[ms] とする。到着間隔（世代数）と各チェーンがシステム内に留まる時間（世代数）は、パラメータ p の幾何分布に従う。一般的にチェーン中の VNF の順番には制約があること [9] を考慮し、本稿ではチェーンとして、{VNF1}、{VNF1 → VNF2}、{VNF1 → VNF2 → VNF3}、{VNF1 → VNF2 → VNF3 → VNF4} のうち 1 つをランダムに選択し用いる。EA のパラメータは、集団中の個体数は 1000、集団中のエリート数は 100、突然変異率は 0.8、適応度関数中の重み付け係数 W は 1.0 とする。その他のパラメータは、実際に用いられている計算機やルータの性能を考慮し、ユーザ u の要求伝送速度 $b_u(t)$ は 200 [Mbit/s]、1 コアの処理能力 C は 3.0 [GHz]、1 パケットのサイズ S は 1500 [bit]、ルータ r のサービス率 M_r は 3.0 [Gpacket/s] とする。

4.2 比 較 手 法

本稿では以下 2 つの比較手法を用いる。

4.2.1 Conventional EA

Conventional EA では、EvoVNFP と同様に、要求到着/離脱時に目標を変更する。EvoVNFP との違いの 1 つは、Conventional EA では目標変更時に集団を初期化することである。もう 1 つは、Conventional EA ではピリオドごとの目標変動を行わないことである。この手法を比較手法とすることで、過去の情報を保持しないことが動的 VNF 配置問題に対する有効な戦略であるかが明らかになる。

4.2.2 Random Immigrant GA

Random Immigrant GA [10] でも、EvoVNFP と同様に、要求到着/離脱時に目標を変更する。EvoVNFP との違いの 1 つは、Random Immigrant GA では突然変異ステップ後に、あらかじめ定められた replacement rate に従って集団中の個体を初期化することである。もう 1 つは、Random Immigrant GA ではピリオドごとの目標変動を行わないことである。この手法を比較手法とすることで、集団に多様性を持たせることが動的 VNF 配置問題に対する有効な戦略であるかが明らかになる。

4.3 評価指標

評価指標として、以下に示す 4 種類の指標を用いる。

4.3.1 配置生成失敗率

目標変更後、次の目標変更までに許容解を生成できなかったケース数の、全目標変更回数に対する割合を評価する。ここで、許容解とは、配置状態への変換が可能な個体のことを指す。

4.3.2 最初の許容解を生成するために要する世代数

ある目標変更後、最初の許容解を生成するために必要となった世代数を評価する。

4.3.3 配置再構成コスト

2 つの連続する目標に対する配置状態に対し、前の配置状態から後の配置状態へ変更するために必要な操作のコストを評価する。配置再構成操作の一覧とその重みは、ルータ内 VM マイグレーション：30、ルータ間 VM マイグレーション：120、VM リサイジング：1、VM 追加：60、VM 削除：1、ルータ内コンポーネントマイグレーション：30、ルータ間コンポーネントマイグレーション：120、コンポーネントリサイジング：1 である。VM/コンポーネントマイグレーションは、ある PM に配置されていた VM/コンポーネントを、別の PM へ移動させることを指す。本稿では、ルータ間マイグレーションがルータ内マイグレーションよりも多くの時間を必要とする [11] ことを考慮し、この 2 種類のマイグレーションを区別する。VM/コンポーネントリサイジングは、ある VM/コンポーネントに割り当てられているコア数を変更することである。VM 追加は配置中に VM を 1 つ追加すること、VM 削除は配置中の VM を 1 つ削除することを指す。重みは、各操作に必要な時間に基づいて設定されており [12]、これを用いた重み付き平均を配置再構成操作数のコストとする。

4.3.4 配置の性能

生成された配置の性能として、配置中の VM に割り当てられているコア数（システム視点の性能）と、2.4 節で解説した配置の遅延（ユーザ視点の性能）を評価する。

4.4 シミュレーション結果

本章ではシミュレーション結果を示しその考察を行う。以降では、Conventional EA を Conv、Random Immigrant GA を RandImm と表記する。各実行では EA を 10000 世代実行する。評価では、安定状態にあるシステムを評価するため、5000〜10000 世代での結果を用いる。1 ピリオドの長さは 20 世代である。40 世代ごとに現在使用されている配置状態よりも 10 パーセント良い配置状態が生成されているかのチェックが行われ、生成されている場合は配置状態の更新が行われる。シミュレーションは、5 つの要求がシステム内に存在する状態から開始する。システム内に配置可能なチェーンの最大数は 10 であり、チェーンが 10 個配置されているときに新しい要求が到着した場合はその要求を棄却する。図 4 の X 軸は世代数を表し、Y 軸は集団中で最も適応度の高い個体の適応度を表す。図 5(a)〜5(e) の X 軸は到着率を表し Y 軸は各指標の 100 実行の平均を表す。到着率としては $1/3000, 1/2000, 1/1500, 1/1000, 1/500$ [リクエスト数/世代] を用いる。システムの負荷をほぼ一定とするために、チェーンがシステム内に留まる時間の平均（幾何分布のパラメータ）は、到着間隔平均（幾何分布のパラメータ）を、システム内初期要求数の逆数倍にしたもの、すなわち到着率の 1/5 倍としている。図 5(b)〜5(e) では、手法が次の要求到着/離脱までに許容解を見つけることができた場合のみを評価している。RandImm の replacement rate は 0.3 である。

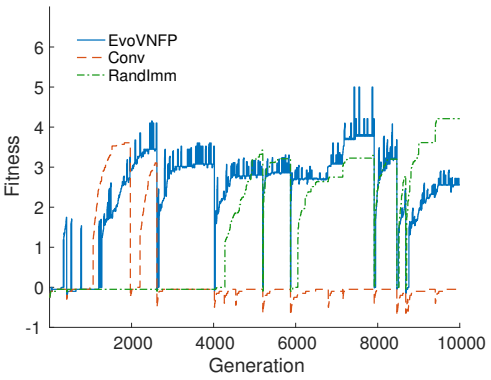


図 4: fitness 推移の一例

4.4.1 適応度変動の一例

試行の一例である図 4 で手法ごとのふるまいを説明する。到着率は 1/1000 である。EvoVNFP は、ピリオドごとの目標変動のために、他の手法と比較して適応度が変動していることが分かる。また初めは、EvoVNFP は生成した解を維持できておらず、RandImm は解を生成できていない。しかし、約 1000 世代経過したのちに許容解を生成した後は、目標変更時に集団を初期化しないため、一度生じた許容解を維持できている。一方で、Conv は早い段階で解を生成できているが、目標変更時に集団の初期化を行うため、許容解を維持できていない。

4.4.2 最初の許容解までの世代数と配置生成失敗率

図 5(a) と 5(b) に、最初の許容解を生成するまでに必要とする世代数と配置生成失敗率の評価結果を示す。到着率が高い場合には、短時間で解を生成しなければならないので、配置生成失敗率が高くなる。ここで、失敗率が高い手法は、配置すべきチェーン数が少ない簡単な問題のみを解いていることになる。Conv と RandImm は EvoVNFP より失敗率が高いため、図 5(b)〜5(e) において EvoVNFP よりも簡単な問題を解いていることになる。すなわちこれらの比較は EvoVNFP にとって不利なものとなっている。次に、図 5(b) において 3 手法を比較すると、EvoVNFP は全ての到着率において最も良い結果となっている。失敗率の差のためにこの比較が EvoVNFP にとって不利であることも考慮すると、EvoVNFP は解を短時間（少ない世代数）で生成できていると言える。この理由の 1 つは、EvoVNFP はピリオドごとの目標変動により局所解に陥ることが少なくなっているためである。もう 1 つは、ピリオドごとの目標変動の方法が有効に働いたためだと考えられる。すなわち、N 個のチェーンを収容する配置状態は $N-1$ 個のチェーンを収容する配置状態を拡張するだけで生成でき、さらに $N-1$ 個のチェーンを収容する配置状態の方が N 個のチェーンを収容する配置状態よりも簡単に生成できるため、EvoVNFP は比較手法よりも短時間で解を生成できたのだと考えられる。

4.4.3 配置再構成操作数のコスト

図 5(c) に、2 つの連続する目標に対する配置状態に対し、前の配置状態から後の配置状態へ変更するために必要な操作数のコストの評価結果を示す。本稿では、前の目標に対する最後の配置更新後の配置と、後の目標に対する最初の許容解を用いて評価した。結果より、RandImm の方が EvoVNFP よりコストが低くなっているが、失敗率の差のためにこの比較が EvoVNFP にとって不利であることを考慮すると、RandImm の方が良い結果であるとは言えないと考えられる。

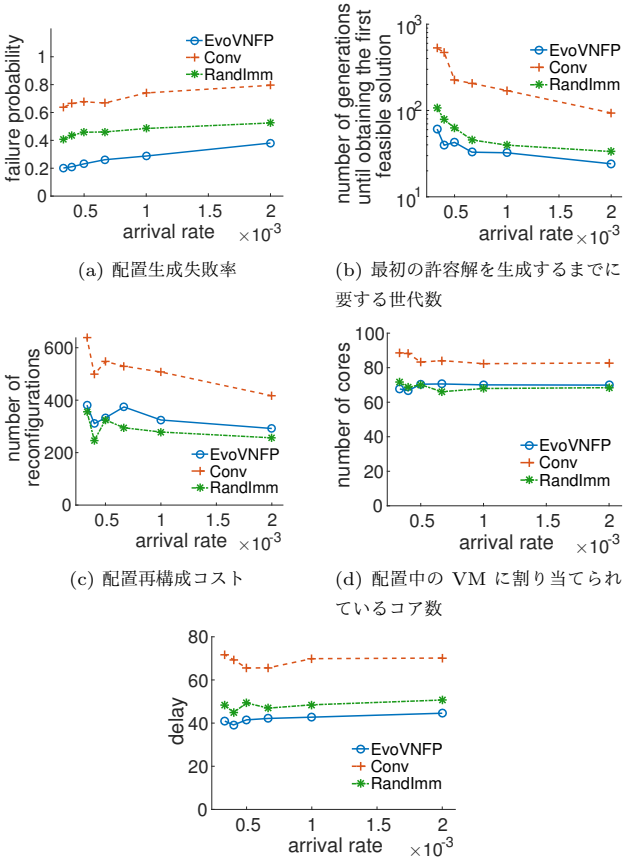
4.4.4 配置の性能

図 5(d) と 5(e) に、生成された配置の性能の評価結果を示す。ここでは、目標に対して初めて生成した許容解、および全ての配置更新時において生成された配置状態の性能を評価している。図 5(d) においては EvoVNFP と RandImm の評価結果はほぼ同じで、Conv より良い結果となっている。図 5(e) においては EvoVNFP が最も良い結果となっている。この結果と、失敗率の差のためにこの比較が EvoVNFP にとって不利であることを考慮すると、EvoVNFP は失敗率を低減させるにもかかわらずシステム視点の性能やユーザ視点の性能におけるオーバヘッドを発生させないことが分かる。

5. まとめと今後の課題

本稿では、動的 VNF 配置手法 EvoVNFP を提案した。EvoVNFP では、進化速度を促進するために、目標を一定世代ごとに、本来の目標とそれをわずかに変化させた目標の間で変化させる。さらに、要求が到着/離脱する際に、前の目標に対して生成された集団を、次の目標に対する初期集団として用いる。シミュレーション評価により、EvoVNFP は許容解の生成失敗率が比較手法より低い、すなわち要求変動に追随して制御を行うことが可能であることが示された。さらにその際生成される配置の性能は比較手法と同程度または比較手法よりも良い、すなわち性能のオーバヘッドが生じないことも示された。今後は、本稿で用いた以外の様々な目標変動方法や評価手法を用いてさらに評価を行う予定である。

謝辞 本研究の一部は、文部科学省博士課程教育リーディングプログラムの補助によるものである。ここに記して謝意を表す。



(a) 配置生成失敗率

(b) 最初の許容解を生成するまでに要する世代数

(c) 配置再構成コスト

(d) 配置中の VM に割り当てられているコア数

(e) 配置中のチェーンで生じる平均遅延

図 5: 評価結果。X 軸は到着率を表す。(b)–(e) では、配置生成成功時のみの結果を示している。

文　献

[1] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 18(1), pp. 236–262, Jan. 2016.

[2] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research Directions in Network Service Chaining," in *Proceedings of SDN4FNS 2013*, pp. 1–7, Nov. 2013.

[3] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On Orchestrating Virtual Network Functions in NFV," in *Proceedings of CNMS 2015*, Nov. 2015.

[4] ETSI, *GS NFV-SWA 001 - V1.1.1 - Network Functions Virtualisation (NFV); Virtual Network Functions Architecture*, Dec. 2014.

[5] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The Dynamic Placement of Virtual Network Functions," in *Proceedings of NOMS 2014*, pp. 1–9, May 2014.

[6] N. Kashtan and U. Alon, "Spontaneous Evolution of Modularity and Network Motifs," *PNAS*, vol. 102, pp. 13773–13778, Sept. 2005.

[7] N. Kashtan, E. Noor, and U. Alon, "Varying Environments Can Speed Up Evolution," *PNAS*, vol. 104, pp. 13711–13716, Aug. 2007.

[8] M. Otokura, K. Leibnitz, T. Shimokawa, and M. Murata, "Evolutionary Core-Periphery Structure and its Application to Network Function Virtualization," *IEICE Transactions on NOLTA*, vol. 7, Apr. 2016.

[9] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, "The Middlebox Manifesto: Enabling Innovation in Middlebox Deployment," in *Proceedings of HotNets-X*, pp. 21:1–21:6, Nov. 2011.

[10] J. Grefenstette, "Genetic Algorithms for Changing Environments," in *Proceedings of PPSN 1992*, pp. 137–144, Elsevier, Sept. 1992.

[11] J. Barrera, M. Ruiz, and L. Velasco, "Orchestrating Virtual Machine Migrations in Telecom Clouds," in *Proceedings of OFC 2015*, pp. 1–3, Mar. 2015.

[12] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A Cost-Aware Elasticity Provisioning System for the Cloud," in *Proceedings of ICDCS 2011*, pp. 559–570, June 2011.