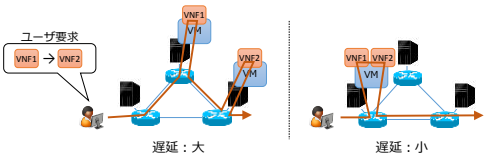
進化適応性を備えた 仮想ネットワーク機能配置手法の 提案と評価

○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

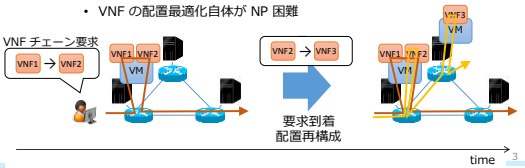
研究背景

- 仮想ネットワーク機能（VNF）
  - ネットワーク機能仮想化（NFV）技術によって仮想化されたネットワーク機能
  - VNF は仮想計算機（VM）上で動作
- VNF 配置問題
  - VNF を実行する VM の物理計算機（PM）への配置を静的に決定



動的 VNF 配置問題

- 動的 VNF 配置問題
  - VNFチェーン要求の到着/離脱時に、VM の配置を再構成
- 動的 VNF 配置問題の要件
  - 配置を短時間で計算
  - 高い性能を持つ配置を生成
- 静的な VNF 配置手法を適用する際の問題点
  - システムが大規模化するほど、要求変動よりも短い時間スケールでの最適化計算が困難
    - VNF の配置最適化自体が NP 困難



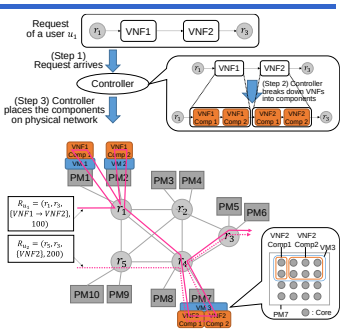
研究の目的とアプローチ

- 本研究の目的
  - 動的 VNF 配置問題の要件を満たす手法の提案
    - 配置を短時間で計算可能
    - 高い性能を持つ配置を生成可能
- 本研究のアプローチ
  - 生物の環境変動の中での進化の知見を利用
    - 生物が環境変動の中で進化する際の特性
      - 環境変動に強い構造を獲得 [6]
      - 環境変動がない場合に比べて進化が加速 [2]

[6] N. Kashtan and U. Alon, “Spontaneous Evolution of Modularity and Network Motifs,” *PNAS*, vol. 102, pp. 13773–13778, Sept. 2005.
[2] N. Kashtan, E. Noor, and U. Alon, “Varying Environments Can Speed Up Evolution,” *PNAS*, vol. 104, pp. 13711–13716, Aug. 2007.

システムモデル

1. VNF チェーン要求が到着
2. コントローラはチェーン中の VNF をコンポーネントに分割
   - コンポーネント：VNF を細分化したもの [4]
3. コントローラはコンポーネントと VM を物理ネットワークに配置しコアを必要数割当
   - コアが多いほど性能良
4. 各ユーザのトラフィック経路を、必要なコンポーネントを経由するように設定



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

VNF 配置問題の定式化

- 平均ユーザ遅延と使用コア数を最小化
  - 式 (2) の制約条件：各 VNF には、通過しているトラフィックを処理するために十分なコア数を与えなければならない
  - 式 (3) の制約条件：VM に割り当てられるコア数は、それが動作する PM のコア数を超過してはならない
  - 式 (4) の制約条件：コンポーネントに割り当てられるコア数は、それが動作する VM のコア数を超過してはならない

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

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

$$\sum_{k} m_{i,k} \leqq N_i \qquad \forall i \qquad (3)$$

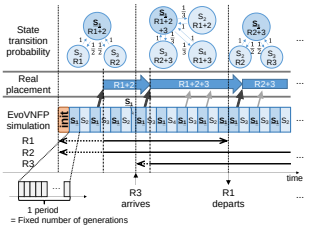
$$\sum_{j,a} n_{k,j,a} \leqq m_{i,k} \qquad \forall k, i \qquad (4)$$

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

$m_{i,k}$：PM i に配置された VM k に割り当てられたコア数
T_a：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
v_a：VNF a のコンポーネントへのトラフィック到着率
S：パケット長
$n_{k,j,a}$：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
W：重み付け係数
C：1コアが1秒で処理可能な命令数
N_i：PM i のコア数
$p^{u}_{(r,r')}$：ユーザ u の VNF チェーンがルータ r と r' 間のリンクを通過していれば1、そうでなければ0

Evolvable VNF Placement (EvoVNFP)

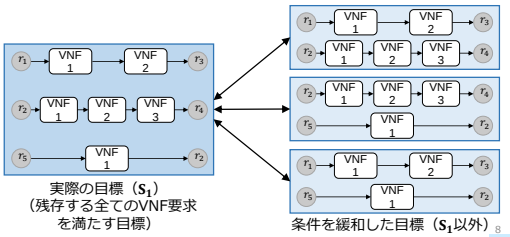
- VNF チェーン要求の動的な到着/離脱への対応を想定した動的 VNF 配置手法
  - EA を用い、目標変動時の集団初期化はなし
    - 先行研究 [8] の知見を利用
  - 一定世代（ピリオド）ごとに目標を意図的に変動
    - 適応性の高い構造を生み出し、進化を加速させるため



7

ピリオドごとの目標変動の方法

- ある要求到着/離脱と次の要求到着/離脱の間の EA では、一定世代（ピリオド）ごとに、残存する要求に応じた形で目標を変動
  - 実際の目標とその条件を緩和した目標を切替
  - 例：実際の目標でチェーンを３個配置するとき、条件を緩和した目標では、３つのうち１つを除いた2個のチェーンを配置



実際の目標（$\mathbf{S_1}$）
（残存する全てのVNF要求を満たす目標）

条件を緩和した目標（$\mathbf{S_1}$以外）

8

EA の適応度関数

- 目標に対する配置性能の良さ（適応度）を計算するための関数
  - 個体が配置状態に変換可能な場合は、平均ユーザ遅延と使用コア数が小さいほど大きい
  - 個体が配置状態に変換不可能な場合には負の値となり、個体中の制約条件に違反している要素の数が少ないほど 0 に近い

$$F = \begin{cases} \left(\frac{\bar{d}}{d_{max}} + \frac{W(\sum_{l,k} m_{l,k})}{c_{max}}\right)^{-1} & \text{（個体が配置状態に変換可能な場合）} \\ \alpha Z & \text{（個体が配置状態に変換不可能な場合）} \end{cases}$$

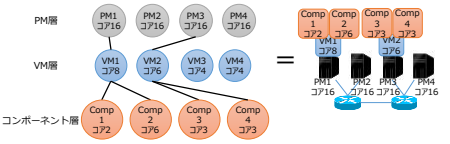
※d_{max}: 遅延の基準値
c_{max}: コア数の最大値
α：負の定数
Z：個体中の、制約条件に違反している要素の数

- EvoVNFP における「条件を緩和した目標」での計算では、実際の目標では考慮している全チェーンのうち１つを除いて適応度の計算を行う

9

EA の個体と突然変異の設計

- 個体：３層ネットワーク（下図）
  - 配置をネットワークの接続状態で表現
    - PM 層と VM 層の接続：PM への VM の配置
    - VM 層とコンポーネント層の接続：VM へのコンポーネントの配置
- 突然変異：要素を１つランダムに選択・変更
  - リンク接続状態の変更、ノードのコア数の変更



10

シミュレーション設定

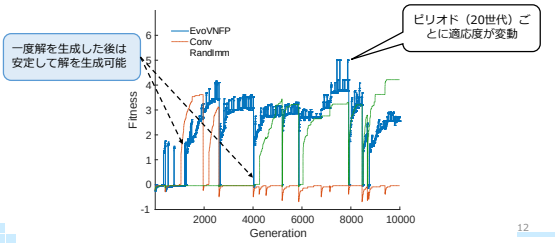
- 物理ネットワーク：5 ルータ、10 PM、各 PM は 16 コア保有
- VNF チェーン要求：イングレスルータ、エグレスルータ、VNF チェーン、要求伝送速度の組
  - 例：(r_1, r_3, $\{VNF1 \rightarrow VNF2\}$, 200 [Mbps])
- 比較手法：
  - 通常の EA を要求到着/離脱の度に再実行（Conventional EA; Conv）
  - Random Immigrant GA（RandImm）[10]
    - 各世代の突然変異ステップ後、ランダムに選択された個体群を初期化
    - 要求到着/離脱の際に集団初期化を行わない
- EA のパラメータ：
  - 集団中の個体数：1000、集団中のエリート数：100、突然変異率：0.8
  - EvoVNFPの１ピリオドの世代数：20、RandImmの個体置換率：0.3
- 要求到着/離脱のパラメータ：
  - (平均到着率, 平均滞在世代数)：(1/3000, 600), (1/2000, 400), (1/1500, 300), (1/1000, 200), (1/500, 100)
    - 到着率の単位は [リクエスト数/世代]

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

11

各手法の適応度変動の一例

- EvoVNFP はピリオドごとの目標変動のために適応度が変動
- EvoVNFPとRandImmは、一度解を生成した後は安定して解を生成可能
  - 目標変動時に集団を初期化しないため



12

評価指標

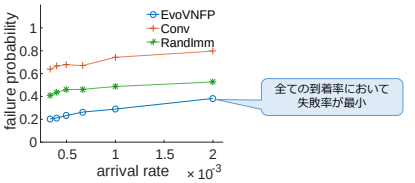
- 配置生成失敗率
  - 目標変更後、次の目標変更までに解を生成できなかったケース数の、全目標変更回数に対する割合
- 配置生成に要する世代数
  - 目標変更後、次の目標に対する配置を生成するために必要となった世代数
- 配置再構成コスト
  - 配置の再構成に必要とした操作数を、以下の表に示す重みを用いて重み付け平均したもの

ルータ内 VM/コンポーネント マイグレーション	30
ルータ間 VM/コンポーネント マイグレーション	120
VM/コンポーネント リサイジング、VM削除	1
VM 追加	60

- 配置の性能
  - 使用コア数
  - 平均ユーザ遅延

配置生成失敗率

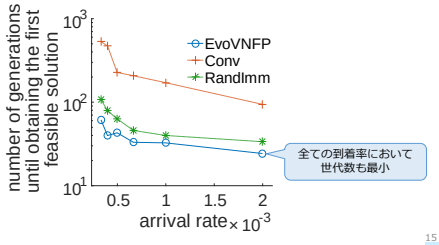
- EvoVNFP は全ての到着率において最も低い失敗率
  - 要求の動的到着/離脱に追随して配置生成可能＝配置計算時間が短い



- これ以降の比較では成功した場合のみの評価を行っているため、失敗率が低い EvoVNFP にとって不利な比較
  - EvoVNFPは、VNFチェーン要求数が多い問題、すなわち解の探索が困難かつ生成した配置の性能も悪くなりやすい問題を多く解いているため

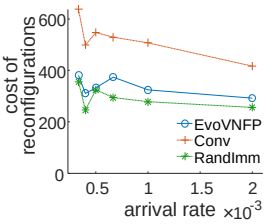
配置生成に要する世代数

- 不利な比較にも関わらず、 EvoVNFP は全ての到着率において配置生成に要する世代数が最小
  - 世代数が少ない＝計算時間が短い



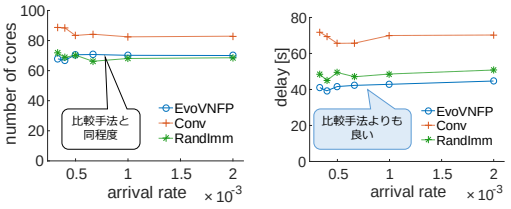
配置再構成コスト

- RandImm が全ての到着率において最も良い結果
  - ただしこの比較は EvoVNFP にとって不利であることを考慮すると、EvoVNFP と RandImm は同程度の結果
- 要求到着/離脱時に集団の初期化をしていない手法のコストが小さい



配置の性能

- 不利な比較にも関わらず、EvoVNFP は他の手法と同じまたは他の手法よりも良い結果
  - 失敗率を低く抑えながら高い性能を持つ配置を生成可能

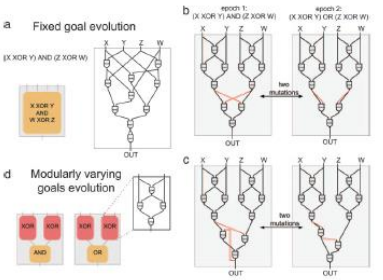


まとめと今後の課題

- まとめ
  - 動的 VNF 配置手法 EvoVNFP を提案
    - EA を用いて、各ユーザ処理要求を満たす配置を生成
    - VNF チェーン要求到着/離脱時は、集団を初期化せずに EA を再実行
    - 一定世代ごとに実際の目標とその条件を緩和した目標を切替
  - シミュレーション評価により EvoVNFP の有効性を確認
    - 配置生成失敗率と最初の許容解を生成するまでの世代数を低減
    - かつ、良い性能の配置を生成可能
- 今後の課題
  - 今回用いたもの以外の様々な目標変動の方法や指標を用いた評価

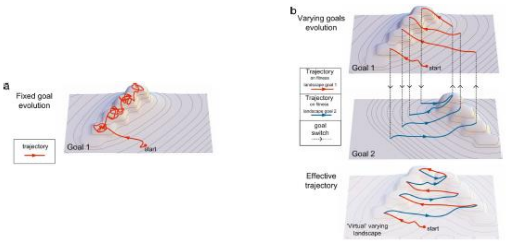
19

補足１　MVG



20

補足２　環境変動下の進化の加速



21