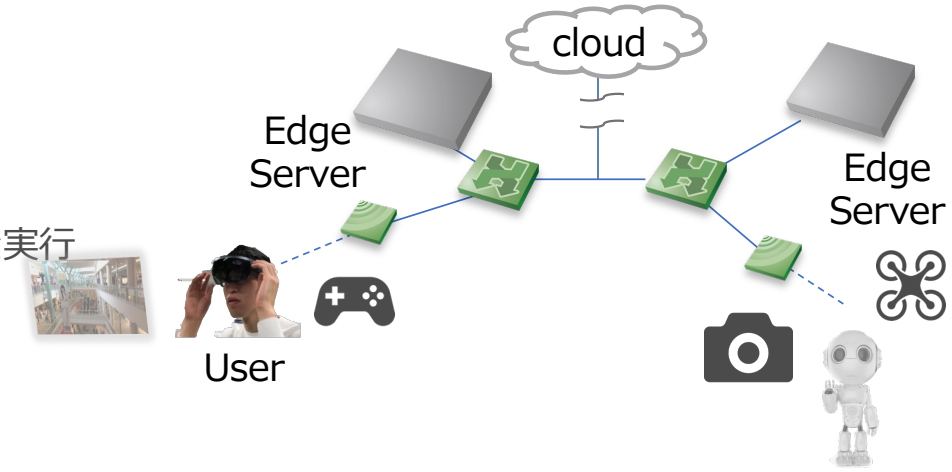
コア/ペリフェリー構造に基づく 進化適応性を持つサービス機能ネットワーク

大阪大学 大学院情報科学研究科
情報ネットワーク学専攻 村田研究室

高木 詩織

研究の背景

- **ネットワークサービスのソフトウェア化**
  - サービス機能同士が API などを介して連携
  - Multi-access Edge Computing (MEC) への期待
    - ユーザに近い場所にエッジサーバを配置しサービス機能を実行
    - 通信距離の削減、負荷の分散により遅延を軽減



- **ユーザ体感型アプリケーションの登場**
  - Telexistence: 遠隔ロボットや MR (Mixed Reality) を用いた臨場体験
  - MEC によってリアルタイム性が求められるサービスの質が向上

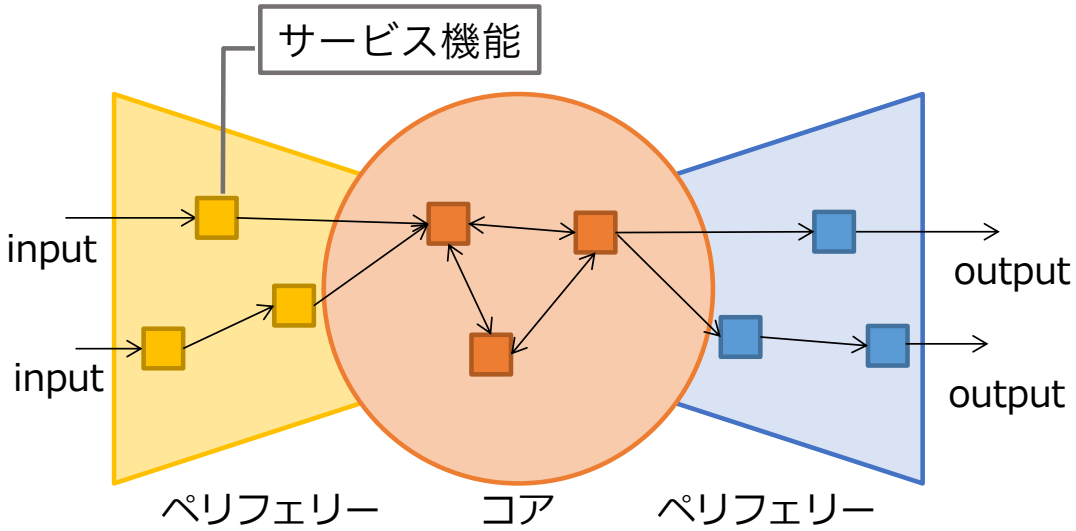
- **MEC 環境におけるサービス機能配置**
  - リソースには限りがありすべてのサービス機能を配置するのは困難
  - 変化するユーザの要求やデバイスの発展などの実環境に合わせたサービスを提供するため、少ないコストで環境変動に対応できる設計が必要

サービス機能のネットワークにおける課題

- **サービス機能のネットワーク上で多様なサービスを収容**
  - API 等によって利用可能なサービス機能同士が接続されたネットワーク
  - ネットワークを構成するサービス機能の一部を使用しサービスを構成
- **環境変動に対応してサービス機能の構成を変更できるサービス機能の構造**
  - 長期的な開発コストを削減できるサービス機能の構造が必要
    - サービス機能の構成の変更にはサービス機能やインタフェースの開発を伴う
    - 環境変動に応じてサービス全体を作り直す場合、開発コストが大きく増加
  - 将来どのようなサービス要求が発生するかを予測することは困難
- **進化適応性を持つサービス構造**
  - サービスを提供する能力を維持したまま、少ないコストでサービス機能全体の構成を変更し続ける能力
  - サービス機能の構造が、新たなサービスを収容可能な構造に進化することが必要

研究の目的とアプローチ

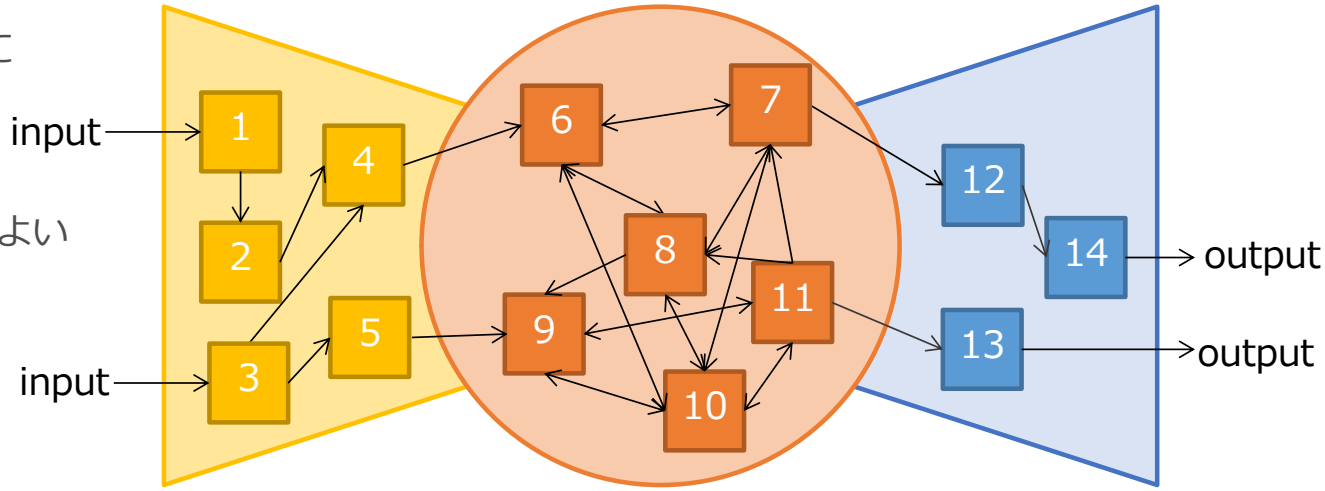
- **研究の目的**
  - 進化適応性を持つサービス構造の実現
    - 進化適応性: サービスの提供能力を維持したまま、少ないコストでシステムを変える能力
- **アプローチ**
  - コア/ペリフェリー構造に着目
    - 生物における柔軟で効率的な情報処理を行うシステム
    - コア: 密に構成され、効率的に情報処理
    - ペリフェリー: 多様な機能を提供できる構造
  - コア/ペリフェリー構造をサービス設計に活用
    - 再利用の可能性が高い機能をコアとすることで、ペリフェリーの接続構造のみを変化させ多様な入出力に対応
    - さらに、ペリフェリー機能の一部をコアに取り入れながら変化させることで新しいサービスを収容できる可能性が上昇



サービス機能ネットワークとサービスチェイン

- **サービス機能ネットワーク**
  - 入力側のペリフェリー、コア、出力側のペリフェリーで構成
    - コア: 互いに利用可能なサービス機能が多く高密度
    - ペリフェリー: 低密度
  - リンク: API などのサービス機能同士を連携可能にするインタフェース

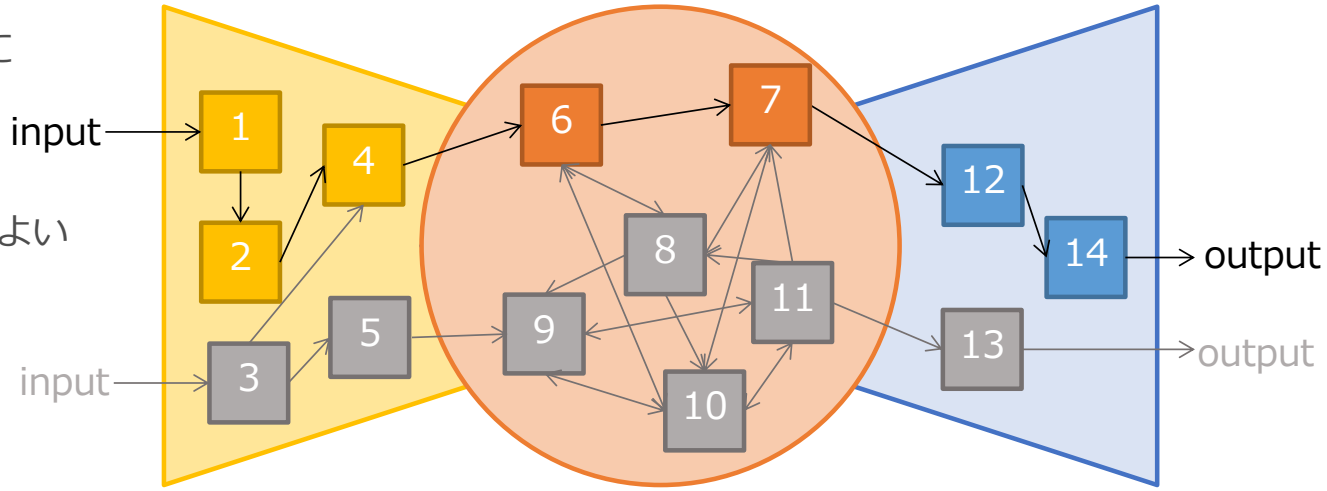
- **サービスチェイン**
  - 使用したいサービス機能と順序を示した組み合わせ
  - 与えられたサービス機能をその順に使用する経路が存在するとき、**収容可能である**と定義
    - 不必要なサービス機能が含まれてもよい
  - サービスチェインを収容した際の経路の長さを**チェイン長**と定義



サービス機能ネットワークとサービスチェイン

- **サービス機能ネットワーク**
  - サービス機能（ノード）：入力側のペリフェリー、コア、出力側のペリフェリーで構成
    - コア: 互いに利用可能なサービス機能が多く高密度
    - ペリフェリー: 低密度
  - リンク: API などのサービス機能同士を連携可能にするインタフェース

- **サービスチェイン**
  - 使用したいサービス機能と順序を示した組み合わせ
  - 与えられたサービス機能をその順に
    使用する経路が存在するとき、
    収容可能であると定義
    - 不必要なサービス機能が含まれてもよい
  - サービスチェインを収容した際の
    経路の長さを**チェイン長**と定義



コア/ペリフェリー構造を用いたサービス機能ネットワーク

- **環境適応性を持つサービス機能の構造**
  - コアを再利用し、ペリフェリー機能のみを変化させることによって少ないコストで環境変動に適応
  - サービス機能ネットワーク全体を進化させ、新しいサービスを収容する必要
    - ペリフェリー機能の著しい増加によりサービスの効率が低下
    - コアではなかった機能の利用の増加

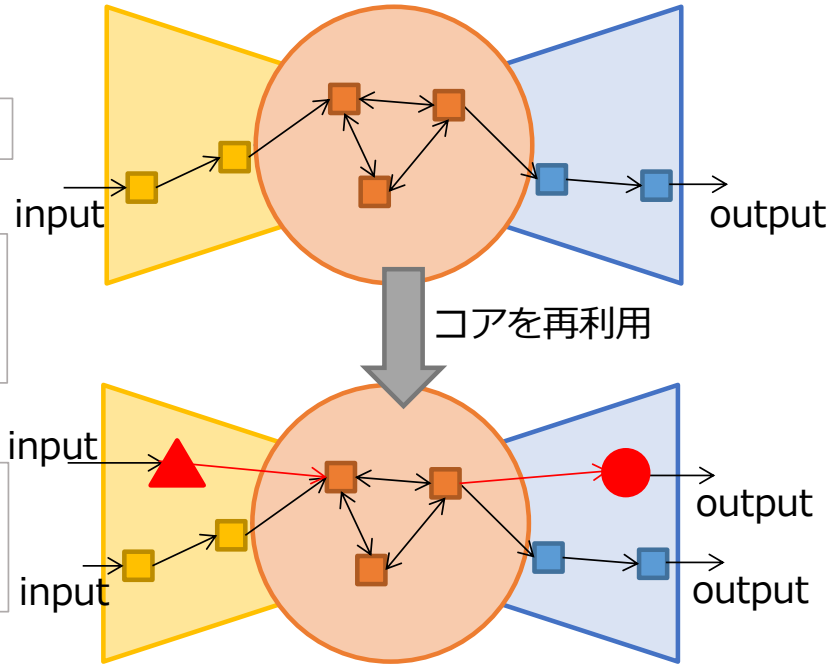
有効性を数値的に確認

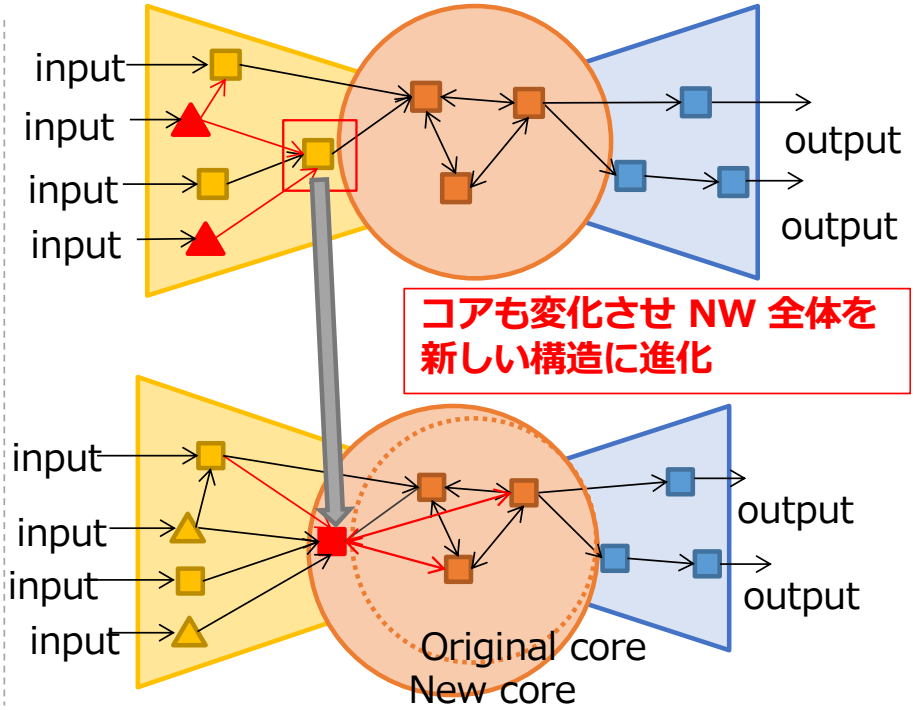
↓

特定のサービスシナリオの実機実装による有効性の確認

↓

環境適応性を持つサービスの実装による評価





進化適応性を実現するためのアプローチ

- **ペリフェリー機能の一部をコアとして利用できるようサービス機能の構成を変更**
  - コアとペリフェリーを適切な大きさと密度に調整
    - サービス機能を密に接続する（コアを大きくする）→サービスを効率よく収容することができるが、高コスト
    - サービス機能を疎に接続する（コアを小さくする）→コストは低いが、サービスの効率が悪化
    - 適切な大きさと密度に調整することで、長期にわたりサービス機能の構成を変え続けることが可能

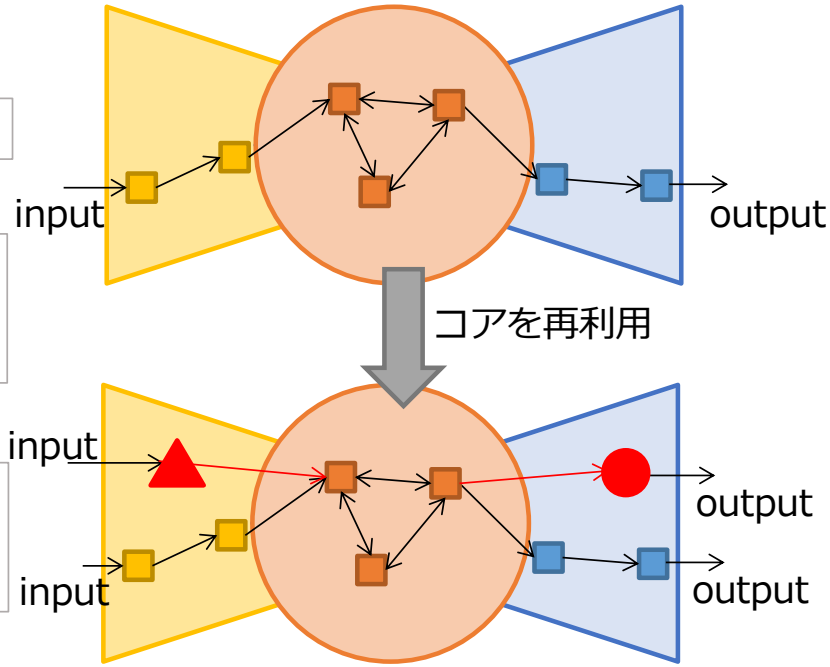
有効性を数値的に確認

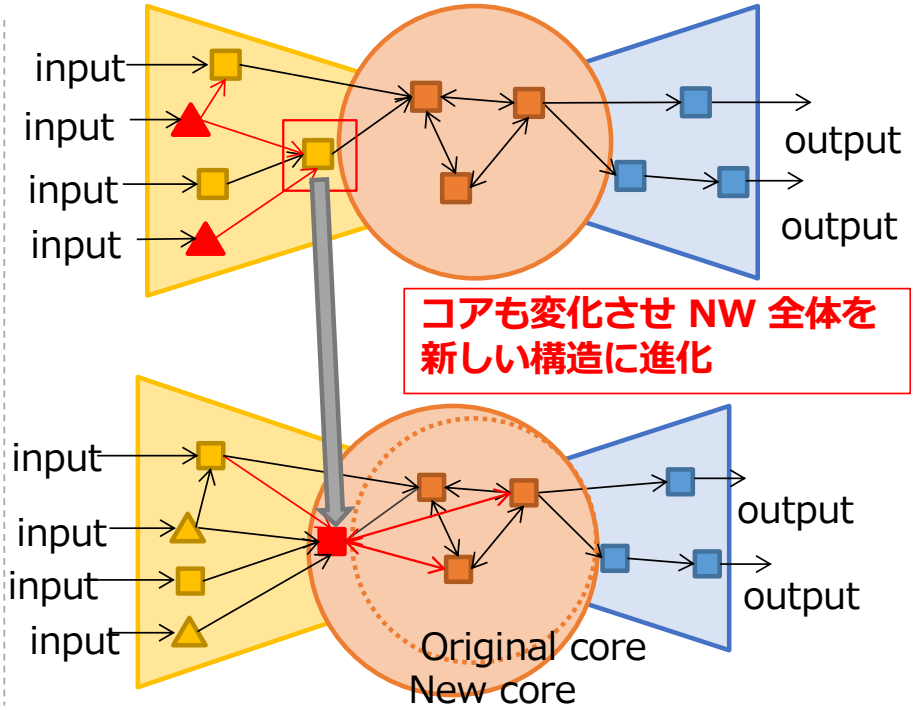
↓

特定のサービスシナリオの実機実装による有効性の確認

↓

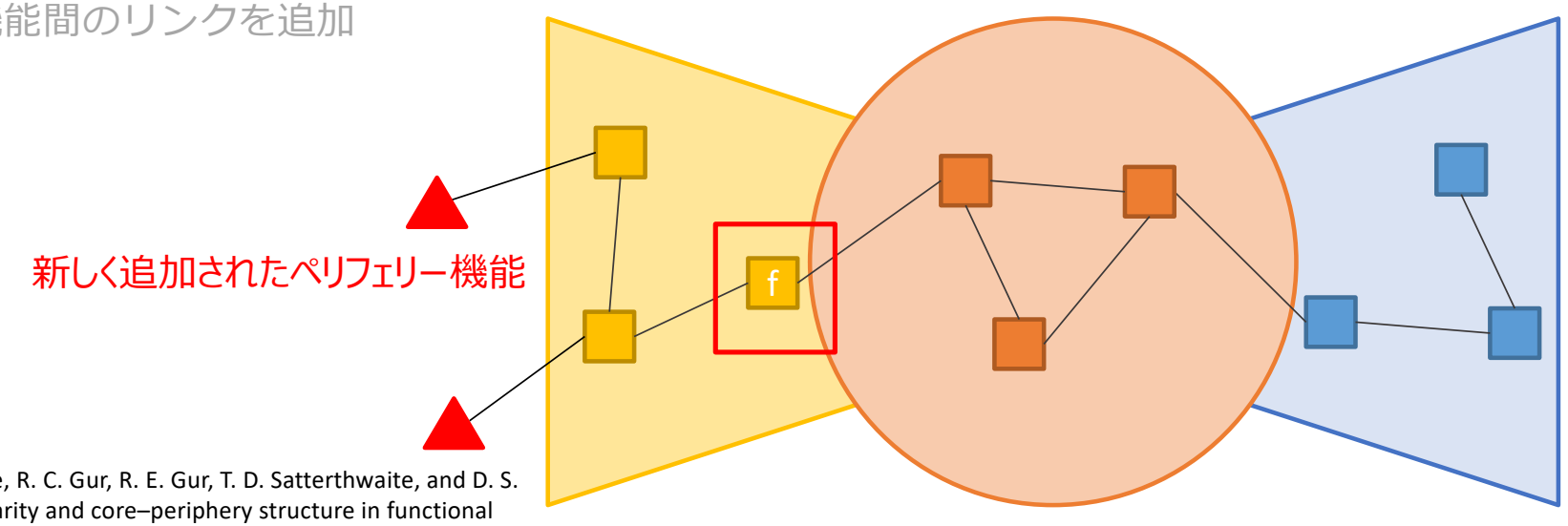
環境適応性を持つサービスの実装による評価





サービス機能の密度の制御

- **コアの密度 > コア/ペリフェリー間の密度 > ペリフェリーの密度になるように密度を制御**
  - コアとペリフェリーの密度が分離しているほど、各ブロックの機能の発展を促進[56]
  1. コアに隣接するペリフェリー機能から注目するサービス機能 f をランダムに選択
  2. f をコア機能とみなせるよう f とコアの間にリンクを追加
  3. 密度の大小関係を保つように、f を含む新しいコアとペリフェリーの間のリンク
  4. ペリフェリー機能間のリンクを追加

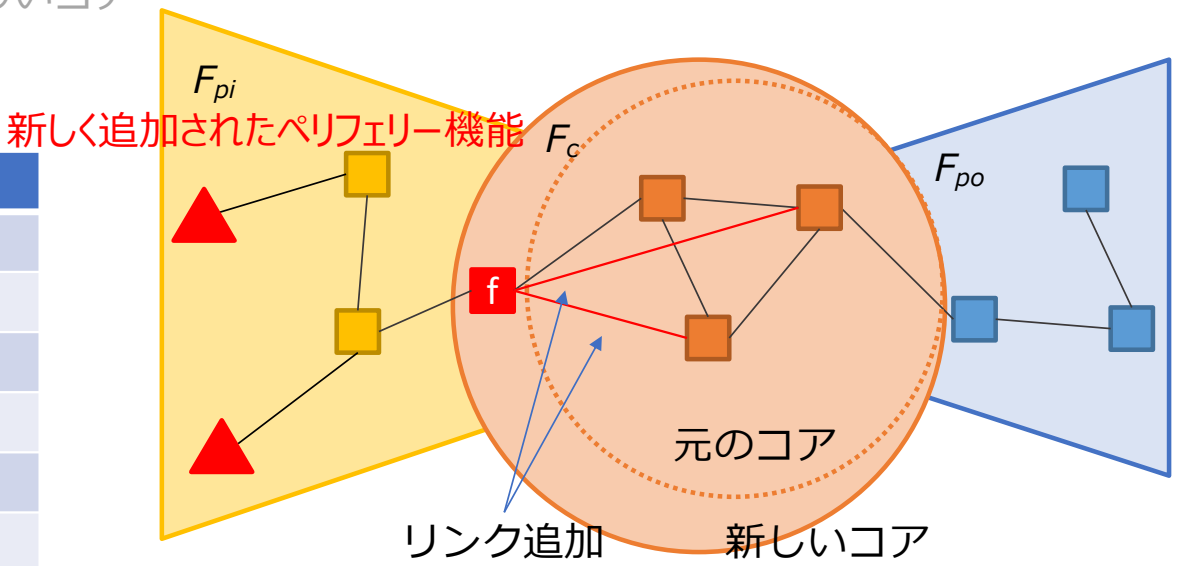


[56] S. Gu, C. H. Xia, R. Ciric, T. M. Moore, R. C. Gur, R. E. Gur, T. D. Satterthwaite, and D. S. Bassett, “Unifying the notions of modularity and core–periphery structure in functional brain networks during youth,” Cerebral Cortex, vol. 30, no. 3, pp. 1087–1102, Aug. 2019.

サービス機能ブロックの密度の制御

- **コアの密度 > コア/ペリフェリー間の密度 > ペリフェリーの密度になるように密度を制御**
  1. コアに隣接するペリフェリー機能から注目するサービス機能 f をランダムに選択
  2. f をコア機能とみなせるよう f とコアの間にリンクを追加
     - コアの大きさ *coresize* は $coresize_{min} < coresize < coresize_{max}$
     - コアの密度 d_c は $dmin_c < d_c$ かつ $d_c - d_{c,cp} > th_{c,cp}$
  3. 密度の大小関係を保つように f を含む新しいコアとペリフェリーの間のリンクを追加
  4. ペリフェリー機能間のリンクを追加

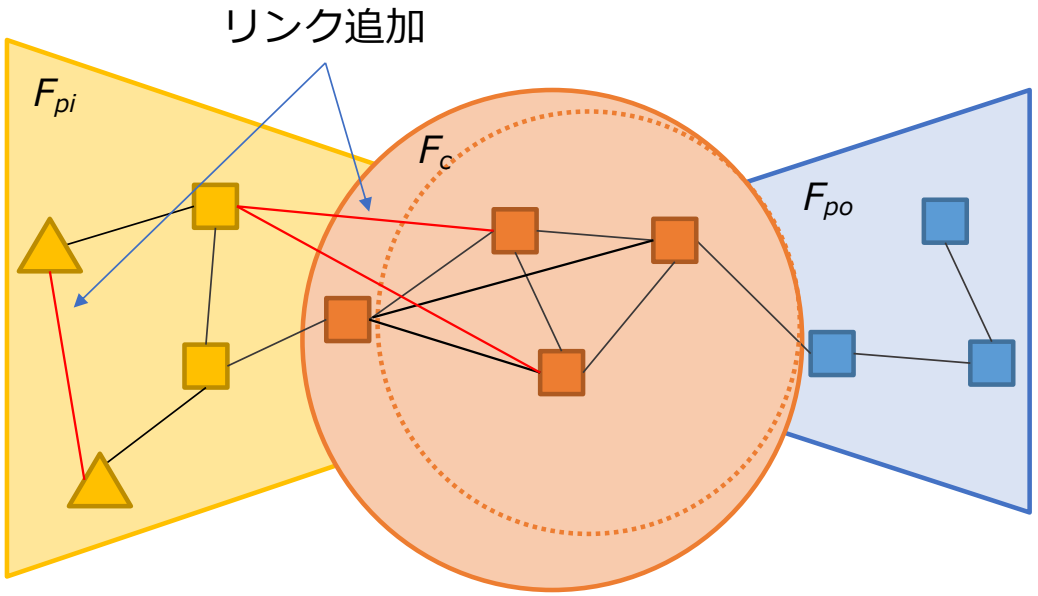
変数名	説明
dc, dp	コア、ペリフェリーの密度
$d_{c,cp}$	コア/ペリフェリー間の密度
coresize	コアの大きさ
$coresize_{min}$, $coresize_{max}$	コアの大きさの最小・最大
$dmin_c$, $dmin_p$	密度の最小値
$th_{c,cp}$, $th_{cp,p}$	密度の差の閾値



サービス機能ブロックの密度の制御

- **コアの密度 > コア/ペリフェリー間の密度 > ペリフェリーの密度になるように密度を制御**
  1. コアに隣接するペリフェリー機能から注目するサービス機能 f をランダムに選択
  2. f をコア機能とみなせるよう f とコアの間にリンクを追加
  3. 密度の大小関係を保つように、f を含む新しいコアとペリフェリーの間のリンクを追加
     - コア/ペリフェリー間の密度とペリフェリーの密度の差は $d_{c,cp} - d_p > th_{cp,p}$
  4. ペリフェリー機能間のリンクを追加
     - ペリフェリーの密度 d_p は $d_p > dmin_p$

変数名	説明
dc, dp	コア、ペリフェリーの密度
$d_{c,cp}$	コア/ペリフェリー間の密度
coresize	コアの大きさ
$coresize_{min}$, $coresize_{max}$	コアの大きさの最小・最大
$dmin_c$, $dmin_p$	密度の最小値
$th_{c,cp}$, $th_{cp,p}$	密度の差の閾値



比較手法と評価方法

- **Density Control: 提案手法**

- **比較手法**
  - Random: ランダムにノードの組を選択し、リンクを追加
    - Density Control とほぼ同等の開発コストを使用
  - Shortest-Path Accommodation: チェイン長が最も短くなるようにリンクを追加
    - SPA with limited cost: Density Control とほぼ同等の開発コストを使用
    - SPA with unlimited cost: 開発コスト無制限
  - Low-Cost Accommodation: サービスチェイン収容率が最も上昇するノードの組を選択しリンクを追加
    - 低コストで多くのサービスチェインを収容できるが、チェイン長が長い手法

評価方法

- **シミュレーションによる評価**

1. 環境変動を想定し、サービス機能ネットワークが変化
   - サービス機能の末端に新しくペリフェリー機能を追加
2. 新しいサービスチェインが 30 個発生
3. 各手法を適用しサービス機能ネットワークを更新
4. 1. と 2. を 40 ステップが経過するまで反復
5. 40 ステップの実行を 100 回試行
   - サービス機能ネットワークが 100 通りの進化を行った場合について評価

パラメータ設定

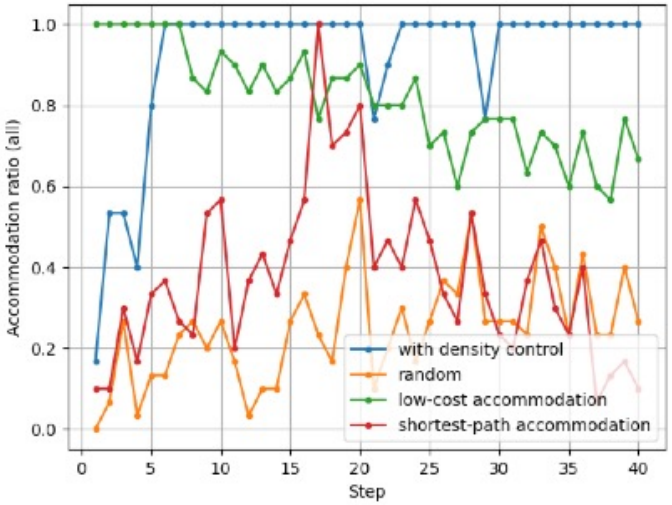
変数名	説明	値
d_c	コアの密度	0.7 以上
$d_{c,cp}$	コア/ペリフェリー間の密度	0.3 以上 *dc-0.3* 以下
d_p	ペリフェリーの密度	0.1 以上 $d_{c,cp}$ *-0.2* 以下
coresize	コアの大きさの割合	0.3 以上 0.4 以下

評価指標

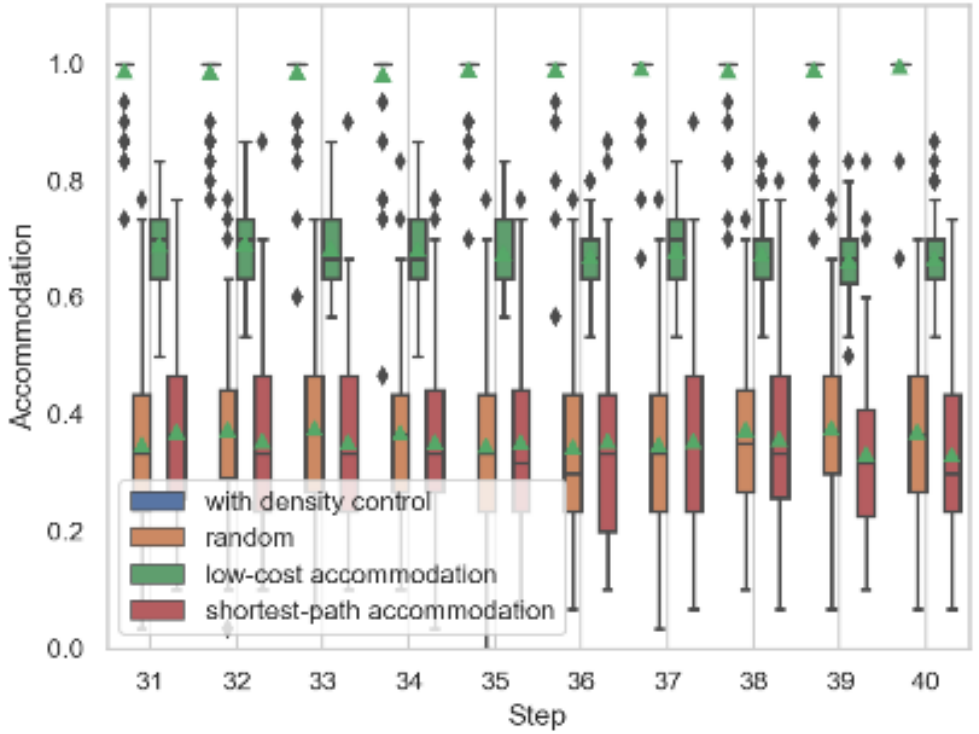
- **サービスチェインの収容率**
  - 各ステップ 30 個生成したサービスチェインのうち、収容可能であったものの割合
  - 新しいサービスチェインをどれだけ収容できるようになったかを評価
  - Density Control, Random, Shortest-Path Accommodation with limited cost, Low-Cost Accommodation を比較
- **開発コスト**
  - 各手法での目標を達成するために各ステップで追加したリンクの数
  - Density Control, Random, Shortest-Path Accommodation with unlimited cost, Low-Cost Accommodation を比較
- **収容可能であったサービスチェインについて、収容時点でのチェイン長**
  - 最短のチェイン長（サービスチェインのノード数 -1）との比
  - 短いほど余計な通信経路が少ないため効率よくサービスを提供可能
  - Density Control, Low-Cost Accommodation を比較

評価結果: サービスチェイン収容率

- **Density Control で安定して高い収容率を達成（青）**
  - コア（密に接続されたブロック）を一定の大きさに保ったことによる効果
  - Shortest-Path Accommodation with limited cost（赤）では、収容率のばらつきが大
    - 長いサービスチェインが多いと収容しきれないため
  - Low-Cost Accommodation（緑）では収容率が低下
    - コア機能を増やす進化をしておらず、
      双方向に利用できるサービス機能が少ないため



収容率の変化の一例

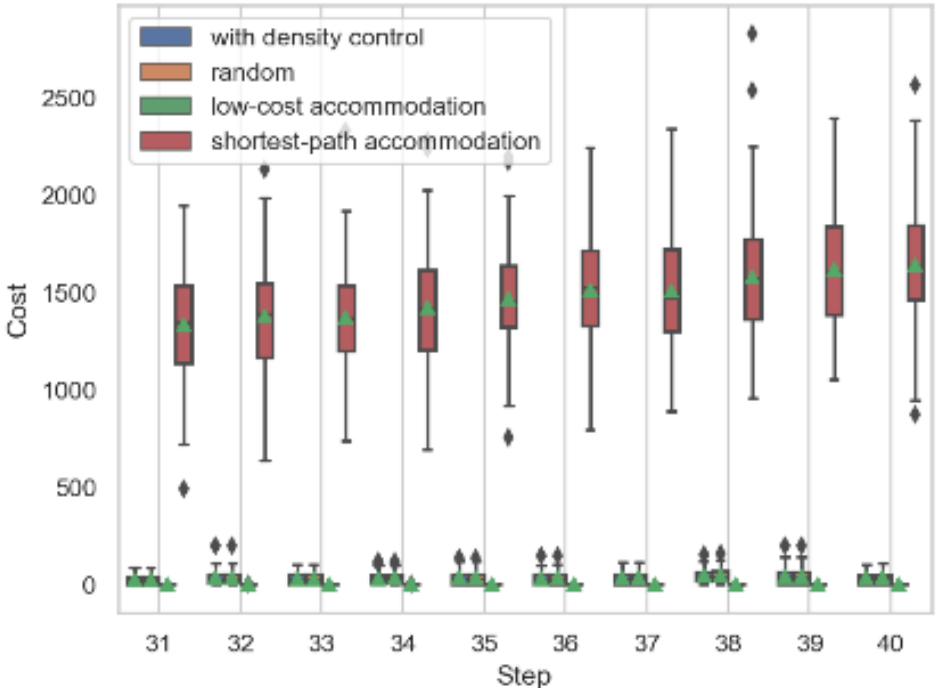


100 回試行時の収容率

評価結果: 開発コスト

- **Density Control は Shortest-Path Accommodation with unlimited cost よりも少ないコストで同等の収容率を達成**
  - Shortest-Path Accommodation では、過去に追加したリンクが再利用されることが少ない
  - Low-Cost Accommodation と比較すると 10 倍程度多くのコストが必要

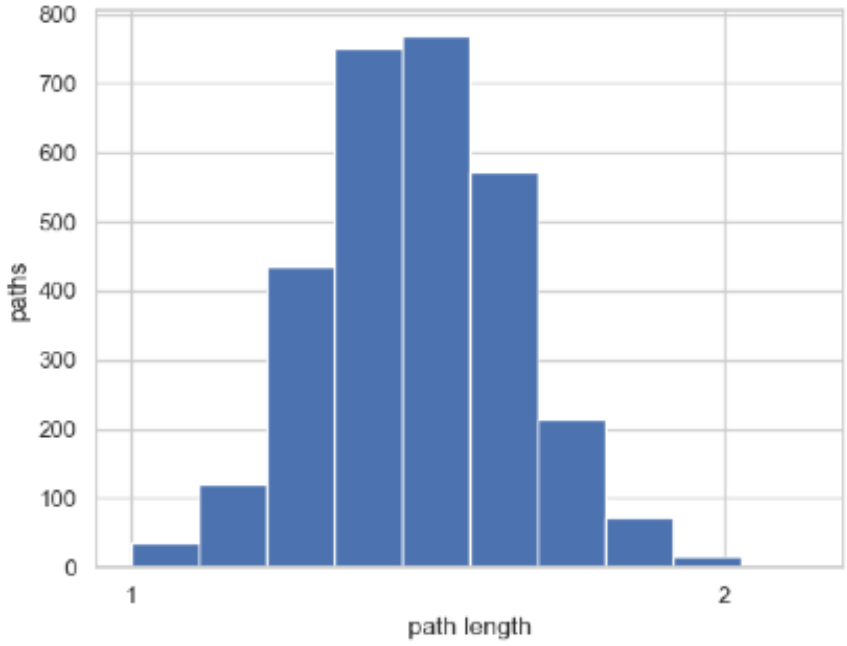
- **Density Control の開発コストは サービスチェインの数や長さに 左右されない**
  - サービスチェインが 30 個/ステップの場合の結果
  - Low-Cost, Shortest-Path Accommodation with unlimited cost では発生したサービスチェインの数や長さに依存
    - 新しい・長いサービスチェインが多く発生するほどコストが増大



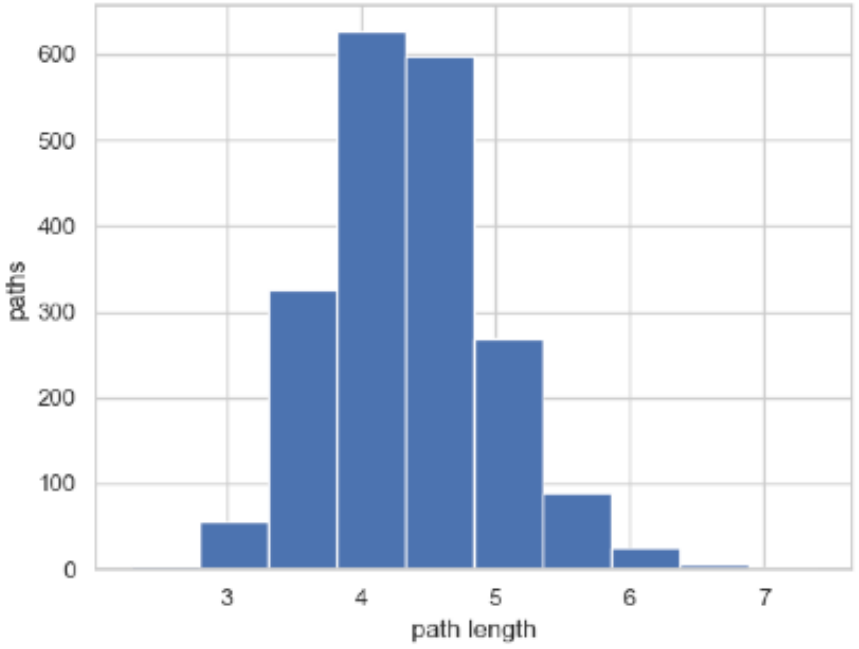
100 回試行時の開発コスト

評価結果: チェイン長

- **Density Control では短いチェイン長で効率よくサービスを提供可能**
  - Density Control では最短チェイン長の 2 倍以下でチェインを構成
    - ペリフェリー機能とコア機能を接続するリンクが用意されるため
  - Low-Cost Accommodation では最短チェイン長の 3 ~ 7 倍程度のチェイン長
    - ペリフェリー機能がリング状に近い形になり、通信経路が長くなっているため



Density Control



Low-Cost Accommodation

まとめと今後の課題

- **進化適応性を持つサービス機能ネットワークの構造を実現**
  - コアとペリフェリーの密度を制御することで、少ないコストで多様なサービスチェインを収容可能
  - サービス構造がどのような進化を行っても、安定して高い収容率を達成
  - 短いサービスチェインを構成することができるため、効率的なサービスを提供可能
  - 開発コストはサービスチェインの数や長さに左右されず安定

- **今後の課題**
  - コアとペリフェリーの配置問題
    - ペリフェリーからコアに変化したサービス機能をどのように配置するか
  - サービス機能が削除された場合への対応