地域IoTサービスに対する計算需要に応じた適応型地産地処リソース配分手法の提案

中村 優吾[1,2,a)] 水本 旭洋[1] 諏訪 博彦[1] 荒川 豊[1,3] 山口 弘純[4] 安本 慶一[1]

概要：本研究では，「地域で生成されたIoTデータ流を，地域で処理して有効活用する：地産地処」をコンセプトに，データ発生源に近いIoTデバイスの計算資源を有効活用して効率的にIoTデータを処理し，低コストかつ質の高いIoTサービスを提供することを目指している．地域IoTサービスを利用するユーザのQoE（Quality of Experience）を保証するためには，各IoTサービスに対してユーザが要求する遅延制約を満たすように，必要となる計算資源をいかに効率的に確保するかが重要な課題である．多種多様なIoTデバイスで構成された分散システムは，デバイスの計算能力，デバイス間のネットワーク性能，デバイスの配置密度などが異なるため，ユーザの計算需要を満たすように計算資源を確保し，QoEの高いサービスを提供することは実用時間で計算が困難な組合せ最適化問題となりうる．本論文では，はじめに上記の問題を，(a) 対象の地域エリアに存在するIoTデバイス群で構成されるリソースグラフ，(b) 地域エリアで展開されるIoTサービスの集合，(c) 一定時間にユーザから各IoTサービスに対して発行されたクエリの集合，(d) クエリに対応する処理タスクグラフの集合，を入力として，QoEの最大化に向けてIoTコンテンツを素早く生成するための (e) タスク実行スケジュール（対象地域に存在するIoTデバイスに対する各タスクの割り当てプラン）を求める遅延制約付き地域IoTサービスプロビジョニング問題として定式化する．そして，この問題を多項式時間で解決するために，(1) IoTサービスの計算需要に応じて動的に計算資源の選択エリアを拡張する適応型スケールアウトと (2) タブー探索に基づく地産地処タスクスケジューリングからなる適応型地産地処リソース配分手法を提案する．提案されたアルゴリズムの有効性を評価すべく，対象地域に2000個のIoTデバイスと10個のIoTサービスが展開された状況を想定したシミュレーションを行い，提案したアルゴリズムがベースラインの手法と比べてより高いユーザQoEを得ることが可能であることを示す．

A proposal of Demand-Aware Adaptive In-situ Resource Provisioning for Regional IoT Services

Yugo Nakamura[1,2,a)] Teruhiro Mizumoto[1] Hirohiko Suwa[1] Yutaka Arakawa[1,3] Hirozumi Yamaguchi[4] Keiichi Yasumoto[1]

1. はじめに

近年，Internet of Things（IoT）は社会を大きく変える可能性が高いことから大きな注目を集めている．IoTデバイスの数は指数関数的に増加し，2020年には500億以上に達すると予想されている [1]．これに伴い，実世界に展開された無数のIoTデバイスを有効活用し，人々の生活や地域社会の質を向上させる様々な地域密着型のIoTサービス（地域IoTサービス）を実現することが期待されている．地域IoTサービスの典型的な例としては，地域の人々が身につけているウェアラブルデバイスから得られるセンサデータを解析し，認識した行動を時間軸にマッピングするライフロギングサービスや，地域の主要スポット（公園/店舗）やモビリティ（バス/車）に設置されているIoTデバイスから得られる騒音や混雑度などの環境情報を地図にマッピングする街の状況可視化サービスなどを想定している．これらのサービスは，地震/洪水のような自然災害時には，避難行動の支援や生存確認などに活用できることから，平常時だけでなく非常時においても，継続的に提供されることが望ましい．そのため，地域IoTサービスを効率的かつ低コストで提供/維持するためのデータ処理プラットフォームの存在が重要である．

現状，クラウドコンピューティングが，IoTサービスを

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[2] 日本学術振興会特別研究員
JSPS Research Fellowships for Young Scientists
[3] JST さきがけ JST PRESTO
[4] 大阪大学 Osaka University
[a)] nakamura.yugo.ns0@is.naist.jp

実現する上での標準的なプラットフォームとしてみなされている [2]．しかしながら，クラウドベースのアプローチは，運用コストが高く，ネットワーク障害に対する回復力が低いことから，地域 IoT サービスを実現するという点では適していない可能性がある．また，遅延の制約がある IoT サービスを実現する上で，データの発生源から遠いサーバ上で処理タスクを実行するアプローチは通信コストや通信遅延の点で効率的ではない．近年は，これらの背景から，クラウドベースのアプローチの限界に対処するべく，フォグ・エッジコンピューティングと呼ばれる新しいコンピューティングのパラダイムが注目を集めている [3][4]．これらのパラダイムでは，遅延の影響を受けやすいタスクをサービスユーザに近いコンピューティングリソースに割り当てる手法が採用されている．

これらの背景を踏まえて，我々の研究グループでは，エッジコンピューティングのアプローチをさらに拡張した次世代型 IoT データ流処理基盤として IFoT（Information Flow of Things）[5] の実現を目指している．IFoT では，「地域で生成された IoT データ流を，地域で処理して有効活用する：地産地処」をコンセプトに，データ発生源に近い IoT デバイスの計算資源を有効活用して効率的に IoT データを処理し，低コストかつ質の高い IoT サービスを提供することを目指している．地域 IoT サービスを利用するユーザの QoE（Quality of Experience）を保証するためには，各 IoT サービスに対してユーザが要求する遅延制約を満足するように，必要となる計算資源をいかに効率的に確保するかが重要な課題である．しかしながら，IFoT で想定する多種多様な IoT デバイスで構成された分散システムは，デバイスの計算能力，デバイス間のネットワーク性能，デバイスの配置密度などに関して異質性を有するため，ユーザの計算需要を満たすように計算資源を確保し，QoE の高いサービスをプロビジョニングすることは困難な制約付き最適化問題となりえる．

本稿では，上記の問題を，(a) 対象の地域エリアに存在する IoT デバイス群で構成されるリソースグラフ，(b) 地域エリアで展開される IoT サービスの集合，(c) 一定時間にユーザから各 IoT サービスに対して発行されたクエリの集合，(d) クエリに対応する処理タスクグラフの集合，を入力として，QoE の最大化に向けて IoT コンテンツを素早く生成するための (e) タスク実行スケジュール（対象地域に存在する IoT デバイスに対する各タスクの割り当てプラン）を求める遅延制約付き地域 IoT サービスプロビジョニング問題として定式化する．そして，この問題を多項式時間で解決するために，(1) IoT サービスの計算需要に応じて動的に計算資源の選択エリアを拡張する適応型スケールアウトと (2) タブー探索に基づく地産地処タスクスケジューリングからなる適応型地産地処リソース配分手法を提案する．提案手法の有効性を確認するために，2000 台の IoT デバイスが展開された地域エリアを想定したコンピュータシミュレーションによる評価実験を行う．その結果，データ発生源に近いエリアに存在するデバイス群を対象としたタブーサーチに基づく地産地処タスクスケジューリングによって，コンテンツ生成おける遅延時間を短縮し，適応型スケールアウトを実施することで各サービスにおけるクエリの受託数を増加させることを確認した．

以降の章構成は，以下の通りである．第 2 章では，提案手法に関連する既存研究を概説すると共に，本提案の位置付けを明らかにする．その後，第 3 章では，問題についての前提条件および問題設定を記述する．第 4 章では，提案手法であるについて述べ，第 5 章で提案手法の有効性を示すために評価実験について述べる．最後に，第 6 章で本論文のまとめとする．

2. 関連研究

本章では，本研究の関連する既存研究として，分散コンピューティングシステム，クラウドコンピューティング，エッジコンピューティングの分野で検討されてきたタスク割り当て手法について概説し，本研究の位置付けを明確にする．

2.1 分散コンピューティングシステム

分散処理システムにおけるプロセッサへのタスクの割当てを最適化する問題は長年関心が集まっており，ヒューリスティックな手法を活用した様々な手法が検討されてきた [6][7][8]．Shem ら [9] は，グラフマッチングアプローチに基づいて分散コンピューティングシステムのタスク割り当てモデルを提案している．彼らは，タスクの割り当てをグラフマッチング問題として定式化した．これは，最小のタスクターンアラウンドタイムでタスクグラフからプロセッサグラフへの準同型を見つける最適化問題である．彼らは，状態空間探索アルゴリズムを適用することによって，単一タスクグラフに対するタスク割り当てを解決した．したがって, このアプローチは，我々が対象とする複数タスクグラフの割り当てを考慮していない．また，Salman らは，均質の分散コンピューティングシステムに対する複数タスクの割り当て問題を解決するために，パーティクル群最適化に基づくタスク割り当てアルゴリズムを提案した [10]．このアプローチは，我々が対象とする異種の処理資源を有する分散 IoT システム環境には適用が難しい．

2.2 クラウドコンピューティングシステム

クラウドコンピューティングシステムは，分散コンピューティングシステムの一種であり，クラウドコンピューティングシステムにおけるタスク割り当ては，分散コンピューティングシステムと同様に重要である．クラウドコンピューティングシステムにおけるタスク割り当ての研究 [11][12][13] に

おける課題は，計算資源を適応的にプロビジョニングし，クラウド上で提供するサービス内のすべてのジョブの期限を守りながら, ランニングコスト（金銭的）を最小化することである．これらの研究では，処理資源として仮想マシンをほぼ無限にプロビジョニングすることを仮定しているため，使用可能な処理資源の制限は考慮されていない．さらに，仮想マシンのランニングコストを最小限に抑えることに重点が置かれているため，処理遅延を考慮しても，データ通信遅延は考慮されていない．したがって，これらのアプローチは，利用可能な物理 IoT デバイスおよび処理リソースの制限，ならびにタスク間のデータ通信遅延を考慮する必要がある分散 IoT システム環境には適していない．

2.3 エッジコンピューティングシステム

最近の研究では，エッジおよびフォグコンピューティングシステムにおけるリソースプロビジョニング手法にも注目が集まっている．Xu ら [14] は，エッジコンピューティングシステムで位置ベースの拡張現実感ゲームなどの位置情報に基づく遅延に敏感なアプリケーション実行するプラットフォームを提案している．このプラットフォームでは，地域エッジのマイクロデータセンターまたは大規模データセンターの処理資源をクラウドにプロビジョニングする．ただし，大規模なサービスエリアをカバーしながら遅延を減らすためには，多数のエッジサーバーを展開する必要があるため，プラットフォームの実現に高い資本コストと保守コストが必要である． Ardagna ら [15][16] は，クラウドコンピューティングとフォグコンピューティングを組み合わせたリソースプロビジョニングプラットフォームを提案している．このプラットフォームでは，フォグコロニーと呼ばれる（フォグサーバ集合）の処理資源を活用し，追加のリソースが必要な場合にのみクラウドのリソースを使用するアプローチを採用している．しかし，このアプローチは，計算資源の需要に応じてフォグコロニーをスケールアウトすることは検討されていない．したがって，利用可能な計算資源が近くに存在する場合にも，クラウド上の追加計算資源を活用することから，近隣のリソースを活用する場合と比較して，遅延時間が長くなる場合がある．

2.4 本研究の位置付け

本研究のアプローチは，我々が知る限りでは，地域の存在する IoT デバイス群を最大限に有効活用して，低コストかつ質の高い地域 IoT サービスを実現するための最初の方法である．本稿で提案する手法は，サービスの需要に応じて現場の計算資源を動的に確保するとともに, メタヒューリスティックアルゴリズムを活用して多項式時間で準最適なタスク割り当てプランを生成する新規的かつ効率的なメカニズムを提供する．

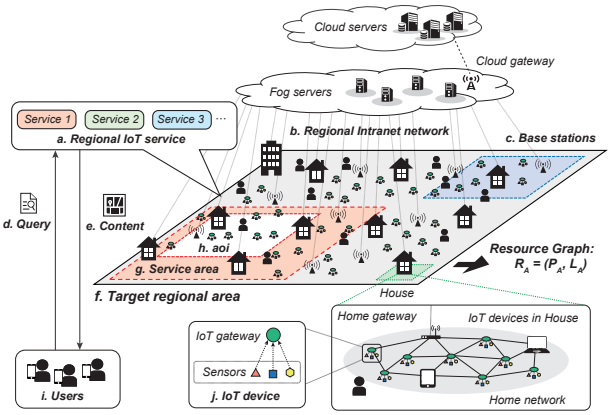

図 1: 想定する地域エリア

3. 地域 IoT サービスプロビジョニング問題

本章では，遅延制約付き地域 IoT サービスの QoE 最大化に向けて，対象エリアに存在する IoT デバイスを有効活用し，IoT コンテンツを素早く生成するためのタスク実行スケジュール（IoT デバイスに対する各タスクの割り当てプラン）を求める遅延制約付き地域 IoT サービスプロビジョニング問題の定式化を行う．以下では，想定する地域のエリアと IoT サービスに関する前提条件を説明し，その後，問題設定を記述する．

3.1 想定環境と前提条件

3.1.1 地域エリア

図 1 に，想定する地域エリアの概要を示す．図のアルファベットは，上から下，左から右に並んでいる．地域エリアには，無数の IoT デバイス (Fig.1.j) と地域 WiFi 基地局 (Fig.1.c) が導入されており，各 IoT デバイスは，地域ネットワーク (Fig.1.b) を介して相互に通信可能である．また，各 IoT デバイスは，割り当てられたタスクを実行するための計算資源（最低でも Raspberry Pi クラスの計算能力を想定する）を持っていると仮定している．

本稿では，地域に展開された計算資源（IoT デバイス）とそれらを結ぶオーバーレイリンクで構成されるグラフ構造をリソースグラフと呼び，$R = (P, L)$ と表記する．ここで，P は頂点（IoT デバイス）の集合であり，L は 2 つの頂点間を結ぶ無向リンクの集合を表す．以降では，R の頂点をプロセッサと呼ぶ．

3.1.2 地域 IoT サービス

対象の地域エリア (Fig.1.f) では，複数の地域 IoT サービス (Fig.1.a) が展開されている．地域エリアで提供される地域 IoT サービスの集合を $S = \{s_1, s_2, ..., s_n\}$ と表す．地域 IoT サービスの典型的な例としては，地域の人々が身につけているウェアラブルデバイスから得られるセンサデータを解析し，認識した行動を時間軸にマッピングするライフロギングサービスや，地域の主要スポット（公園/

店舗）やモビリティ（バスや車）に設置されている IoT デバイスから得られる騒音や混雑度などの環境情報を地図にマッピングする街の状況可視化サービスなどを想定している．基本的に，地域 IoT サービスは，地域で生成されるセンサデータを収集し，何らかの処理を行った後に，意味のある情報として集約し，コンテンツとしてユーザに提供するものとする．各地域 IoT サービス s には，サービスを提供可能なエリアを表すサービスエリア $area(s)$(Fig.1.g) があらかじめ設定されている．加えて，サービス提供する際の水準を表す SLA（Service Level Agreement）として，満足できる遅延（preferable delay）閾値 pd_s，許容可能な遅延（marginal delay）閾値 md_s，サービスが提供できる最大空間解像度 SR（測定点の数/単位面積 (m^2)），最大時間分解能 TR（サンプリング数/単位時間 (s)）が設定されている．サービス s を提供する上で，センサデータをどのように処理するかを示した順序付きのタスクの集合をタスクグラフと呼び G_s と表す．タスクグラフ G_s は，深さ 5 の有向非循環グラフ（DAG）として記述されており，(1) Start, (2) 収集タスク, (3) 処理タスク, (4) 集約タスク, (5) end で構成されている．各タスクの処理負荷や並列数に関しては，後に説明するユーザからサービス s に対して発行されるクエリ q の内容に応じて決定される．また，クエリ q に応じて決定されたタスクグラフは，$G_q = (T_q, E_q)$ と表される．各サービス $s \in S$ は，G_q に含まれる各タスクを実行することでユーザに対して IoT コンテンツを提供することができる．このとき，各タスク $t \in T_q$ は，各サービスに対応するリソースサブグラフである R_s 内のプロセッサに割り当てられることを基本方針とする．サービス提供開始時は，サービスエリア内に存在する計算資源がリソースサブグラフ内のプロセッサに紐付けられる．

3.1.3 ユーザとクエリ

地域エリアに存在するユーザ (Fig.1.i) は，各サービス $s \in S$ に対してクエリ q (Fig.1.d) を発行することで，コンテンツ (Fig.1.e) を受け取る（サービスを利用する）ことが可能である．地域エリアに存在する全てのユーザから発行されたクエリの集合を Q と表す．各クエリ $q \in Q$ は，タプル $\langle s, aoi, sr, tr \rangle$ で表現される．s は，クエリ発行対象のサービス，aoi は，ユーザの関心領域である AoI（Area of Interest）(Fig.1.g) を表す．ユーザは基本的にサービスエリア $area(s)$ の中からのみ aoi を選択することが許可されるものとする．sr と tr は，それぞれ要求するコンテンツの空間解像度と時間解像度を表す（これらの値は，各サービスで設定されている最大空間解像度 SR と最大時間解像度 TR 以下であるものとする．）．

3.2 問題設定

本研究で対象とする問題は，(a) 対象の地域エリアに存在する IoT デバイス群で構成されるリソースグラフ $R = (P, L)$，(b) 地域エリアで展開される IoT サービスの集合 $S = \{s_1, s_2, ..., s_n\}$，(c) 一定時間にユーザから各 IoT サービス $s \in S$ に対して発行されたクエリの集合 $Q_s \subset Q$，(d) それぞれのクエリ q に対応する処理タスクグラフの集合 $\cup_{q \in Q_s} G_q (= (T_q, E_q))$ を入力として，QoE の最大化に向けて IoT コンテンツを素早く生成するための (e) タスク割り当てプランを求める問題である．

以下では, 問題の定義について詳しく説明する．

3.2.1 各サービスに対する動的リソースサブグラフ

各サービス $s \in S$ は, それぞれ自身のサービスを提供するための計算資源プールとして $R = (P, L)$ のサブグラフである R_s を確保する．リソースサブグラフ R_s は以下のように定義される．ここで，$pos(p)$ はプロセッサ p の位置を示し，$area(s)$ はサービス s のサービスエリアを示す．

$$R_s = (P_s, L_s) \tag{1}$$

where

$$P_s = \{p | p \in P_A \wedge pos(p) \in area(s)\}$$
$$L_s = \{l | pos(l.st) \in area(s) \wedge pos(l.ed) \in area(s)\}$$

上記に示すように，初期状態では，各サービス s のリソースサブグラフ R_s の計算資源選択エリアはサービスエリアに $area(s)$ 絞られているが，各サービスはこのリソースグラフを動的に拡張することでスケールアウトすることが許可されている．拡張可能な動的リソースグラフ R_s^{i+} を以下のように定義する．$R_s^{(i-1)+} = (P_s^{(i-1)+}, L_s^{(i-1)+})$ から拡張されたサービス s のリソースサブグラフを $R_s^{i+} = (P_s^{i+}, L_s^{i+})$ と表す．ここで $R_s^{0+} = R_s$ であり，$|P_s^{i+}| = |P_s^{(i-1)+}| + 1$ が成り立つ．つまり，計算資源選択エリアを拡張することによって，最低 1 つのプロセッサが R_s^{i+} に追加される[*1]．

3.2.2 タスク割り当てと遅延時間

ここでは，説明を簡略化するために，対象の地域エリアに存在するユーザから発行されるすべてのクエリ集合 Q に付随する全てのタスクグラフの集合 G を以下のように定義する．

$$G = (T, E) \text{ where } T = \bigcup_{q \in Q} T_q, E = \bigcup_{q \in Q} E_q \tag{2}$$

そして，対象の地域エリアに存在する全てのプロセッサ $p \in P$ とタスク $t \in T$ のすべての組み合わせを表す行列の要素として，変数 $x_{t,p}$ を定義する．t が p に割り当てられている場合は 1 になり，それ以外の場合は 0 になる．本問題では，T のすべてのタスクが 1 つのプロセッサ $p \in P$

[*1] 各サービスの資源選択エリアは，対象の地域エリア全体まで拡張され，他のサービスのリソース領域と重複しても良い．

に割り当てられるものとする．この制約条件を以下の式で表す．

$$\forall t \in T, \sum_{p \in P} x_{t,p} = 1 \tag{3}$$

各タスク t は $comp(t)$ で示される必要な計算量を持ち，各プロセッサ p は $cap(p)$ で示される所定のタスク受け入れ上限計算キャパシティを持つ．p は $cap(p)$ の範囲内で，できるだけ多くのタスクを受け入れることが可能である．この制約条件を以下の式で表す．

$$\forall p \in P_A, \sum_{t \in \{t | x_{t,p}=1\}} comp(t) \leq cap(p) \tag{4}$$

各クエリ $q \in Q$ に対応するタスクグラフ G_q は解像度パラメータである $q.sr$ と $q.tr$ に基づいて収集タスクと処理タスクの並列処理数を適切に選択することによって $G_{q.s}$ から生成される．タスクグラフ G_q を実行する際の総遅延時間は，Start から End までのすべての実行パス間（各パスには，収集タスク，処理タスク，および集約タスクが順に 1 つずつ含まれている）の最大処理/通信遅延である．

タスクグラフ $G_q = (T_q, E_q)$ の経路数を n_q とする．$path_{q,i} = \langle Start, tc_{q,i}, tp_{q,i}, ta_{q,i}, End \rangle$ は，G_q の i 番目の実行経路を表す．ここで，$0 \leq i \leq n_q$ であり，$tc_{q,i}$, $tp_{q,i}$, $ta_{q,i}$ は，それぞれタ収集タスク，処理タスク，集約タスクを表す．ここで，$T_q = \bigcup_{0 \leq i \leq n_q} \{tc_{q,i}, tp_{q,i}, ta_{q,i}\}$ である．

タスク t が割り当てられるプロセッサを $p(t)$ とし，t が出力するデータのサイズを $dsize(t)$ とする．ここでは，プロセッサ p 上でタスク t が実行されるときの処理遅延を見積もる関数 $pt(t,p)$ およびサイズ $dsize$ のデータがリンク $l \in L$ を介して転送されるときの通信遅延を見積もる関数 $dt(l, dsize)$ が利用可能であると仮定する．

各実行パスについて，$delay(path(q,i)) = delay(\langle Start, tc_{q,i}, tp_{q,i}, ta_{q,i}, End \rangle$ で表される処理遅延と通信遅延の合計は，次の式で計算される．

$$\begin{aligned} delay(path(q,i)) = \\ \sum_{p \in P_A} [pt(tc_{q,i}, p) \cdot x_{tc_{q,i},p} + pt(tp_{q,i}, p) \cdot x_{tp_{q,i},p} \\ + pt(ta_{q,i}, p) \cdot x_{ta_{q,i},p}] \\ + dt(link(tc_{q,i}, tp_{q,i}), dsize(tc_{q,i})) \\ + dt(link(tp_{q,i}, ta_{q,i}), dsize(tp_{q,i})) \\ + dt(link(ta_{q,i}, End), dsize(ta_{q,i})) \end{aligned} \tag{5}$$

where

$$link(t, t') = (p(t), p(t')), \forall t, t' \in T_q$$

そして，G_q を実行するための総遅延時間 $D(q)$ は，以下の式で表すことができる．

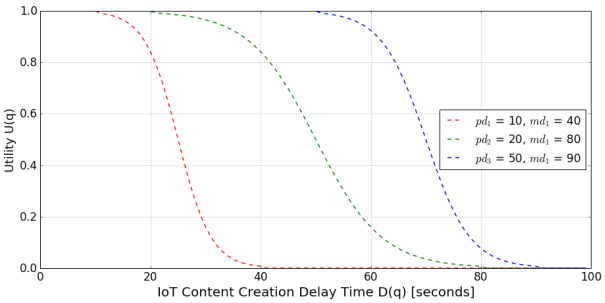

図 2: QoE ベース効用関数グラフ

$$D(q) = \max_{0 \leq i \leq n_q} delay(path(q,i)) \tag{6}$$

3.2.3 QoE ベースの効用関数

本研究では，ユーザの QoE は，発行されたクエリ q に対応する G_q を実行する際の総遅延時間 $D(q)$ によって決定するものとする．具体的には，遅延が pd の範囲内で最大であり，遅延が md に近づくにつれて徐々に減少し，遅延が md を超えた場合に最終的に 0 になるものと仮定する．我々は，QoE に関する既存研究 [17] に基づいて，各サービスの好ましい遅延時間 pd および限界遅延時間 md を超えた度合いに応じて値が変化する QoE ベースの効用関数 $U(q)$ を次の式で定義する[*2]．図 2 は，pd と md の異なる値に対する効用関数の例である．

$$U(q) \stackrel{\text{def}}{=} \begin{cases} 1 & (D(q) < pd) \\ 1 - \dfrac{1}{1 + e^{5(ad - D(q))/(ad - pd)}} & (pd \leq D(q) \leq ad) \\ \dfrac{1}{1 + e^{5(D(q) - ad)/(md - ad)}} & (ad < D(q) \leq md) \\ 0 & (md < D(q)) \end{cases} \tag{7}$$

where

$$ad = \frac{pd + md}{2} \tag{8}$$

3.2.4 問題定義

本研究で対象とする問題は，(a) 対象の地域エリアに存在する IoT デバイス群で構成されるリソースグラフ $R = (P, L)$，(b) 地域エリアで展開される IoT サービスの集合 $S = \{s_1, s_2, ..., s_n\}$，(c) 一定時間にユーザから各 IoT サービス $s \in S$ に対して発行されたクエリの集合 $Q_s \subset Q$，(d) それぞれのクエリ q に対応する処理タスクグラフの集合 $\cup_{q \in Q_s} G_q (= (T_q, E_q))$，を入力として，全てのクエリに対する効用関数を最大化するような (e) タスク割り当てプラン $x_{t,p}$ を決定することが目的である．したがって，本問題の目的関数を以下のように定義する．

[*2] pd と md はサービスごとに定義されている．

$$Maximize \quad \sum_{s \in S} \sum_{q \in Q_s} U(q) \qquad (9)$$
$$subject\ to \quad (3)\ (4)$$

この問題は，NP-hard であると知られている RCPSP（Resource-Constrained Project Scheduling Problem）[18] のインスタンスであるため，NP-hard 問題である．

4. 提案手法

3 章で述べた問題は NP 困難であると予想されるため，本章では，多項式時間での解決に向けたヒューリスティックアルゴリズムを提案する．

4.1 基本方針

我々は遅延制約付き地域 IoT サービスプロビジョニング問題を解決するために次の 2 つの基本方針を採用している．**(*1*) 地産地処タスク割り当て**：クエリ q によって生成されたタスクグラフ G_q は，サービスエリア $area(q.s)$ および拡張された計算資源選択エリア内の IoT デバイスに対して優先的に割り当てられ，処理される．**(*2*) *FCFS* タスク割り当て**：クエリの集合 Q に対して，Q 内の各クエリ q に対応するタスクグラフ G_q のすべてのタスクは，クエリの生成順序に従い，到着順（FCFS, First-Come-First-Served）で利用可能な IoT デバイスに割り当てられる．

4.2 システムモデル

提案するアルゴリズムのシステムモデルを図 3 に示す．このモデルは，図 3（A）に示す**リージョナルブローカー（*R* ブローカー）**と図 3（B）に示す（S ブローカー）という 2 種類のブローカーで構成されている．R ブローカーは，すべてのデバイスが事前に知っている well-known サーバーである[*3].R ブローカーは，対象エリアに存在するすべてのプロセッサ（計算リソース）とそのエリアに配置された地域 IoT サービスの管理を担当し，S ブローカーは特定のサービスの計算資源確保とタスク割り当てを管理する．各

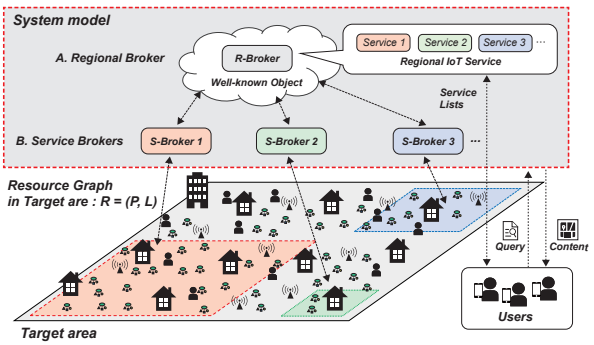

図 3: 提案手法のシステムモデル

[*3] 通常，クラウドサーバーまたはフォグ・エッジサーバーは，R ブローカーとして選択される可能性が高いが，強力な IoT デバイス/ゲートウェイも選択対象である．

サービス s に割り当てられた各 S ブローカーは，s に対する各クエリ q の受付制御とタスクグラフ G_q のスケジューリングを実行する．

4.3 提案手法の処理ステップ

図 4 に提案手法の処理ステップを示す．提案手法は，(1) 各サービスの起動と初期リソースの確保，(2) 各クエリの受付制御とリソース選択領域の適応型スケールアウト，(3) 受理したクエリの地産地処タスクスケジューリングの 3 ステップで構成される．

Step 1: サービス起動と初期リソース確保

ステップ 1 は，R ブローカー上で実行される．このステップでは，R ブローカーは各サービスを起動すると共に，それぞれのサービスを提供する際の初期リソースとなるリソースサブグラフ $R_s = (P_s, L_s)$ を選択する．図 5 に示すように P_s から S ブローカーとして最も強力で最も安定したプロセッサ（IoT デバイス）が選択され，その後，S ブローカーが起動し，ステップ 2 とステップ 3 が実行される．

Step 2: クエリ受付制御と適応型スケールアウト

このステップは，ステップ 2a：クエリ受付制御とステップ 2b：適応型スケールアウトという 2 つのサブステップで構成されている．ステップ 2a において，S ブローカーがクエリ q を受信すると，タスクグラフ G_q が生成される．具体的には，以下の手順でタスクグラフ G_q が生成される．

(1) まず，クエリ発行時に指定される各パラメータ（aoi, sr, tr）によって入力データサイズが算出される; (2) 次に，サービスのタスクグラフ G_s における各タスク (T_c, T_p, T_a) の処理負荷とクエリで指定された入力データから 3 ステップ（Collecting, Processing, Aggregating）それぞれのタスク負荷を算出する．(3) 最後に，$G_{q.s}$ の各タスク t は，複数のサブタスク $st_1, ..., st_{k_t}$ に分割され，$comp(st_i) \leq cap(p)(1 \leq i \leq k_t)$ が成り立つ．次に，各タスク t を $G_{q.s}$ のサブタスク $st_1, ..., st_{k_t}$ の並列実行で置き換えることによって，クエリ G_q のタスクグラフが生成される．

G_q を生成した後，S ブローカーはヒューリスティックな方法で，タスクグラフ G_q を現在のリソースサブグラフ R_s（または R_s^{i+}）を使用して限界遅延 $md_{q.s}$ 以内で処理できるかどうかを判断する．ステップ 2b において，S ブローカーはサービス s のリソース使用量を定期的にチェックし，サービス s の計算リソースが不十分であれば計算資源選択エリアを拡張してリソースグラフ R_s^{i+} を生成する**適応型スケールアウト**を実行し，クエリが処理される．

Step3: 地産地処タスクスケジューリング

最後に，ステップ 3 において，S ブローカーは，受け入れられたクエリ q に対するタスクグラフ $G_{q.s}$ をスケジューリングする．具体的には，タブー探索アルゴリズムを使用して，行列サイズ $|T_q| \times |P_q.s^{i_s+}|$ をもつ準最適なタスク割り当て

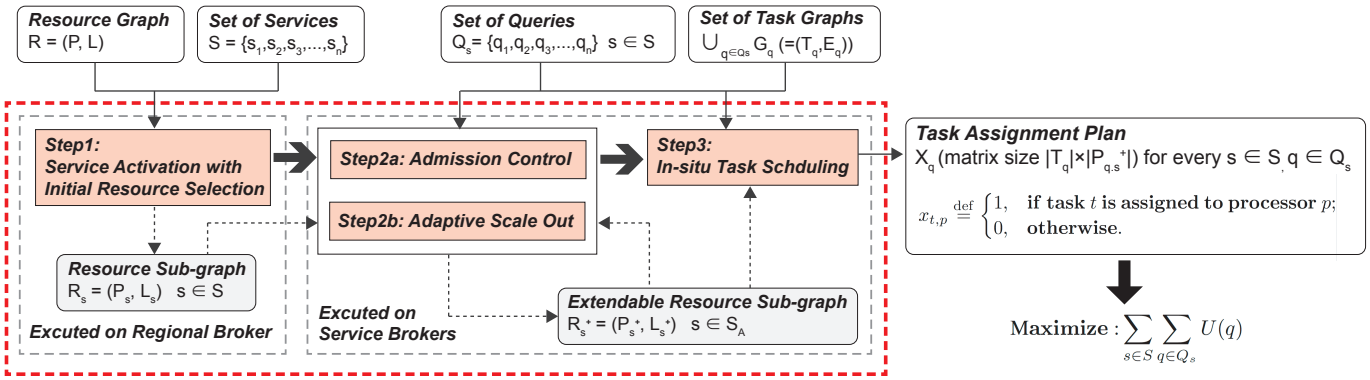

図 4: 提案手法の処理フロー

X_q を生成する．

ステップ 2 およびステップ 3 の詳細なアルゴリズムを以下に説明する．

4.4 クエリ受付制御と適応型スケールアウト

S ブローカーはクエリ q を受け取ると，指定された遅延制約の観点から q が許容可能かどうかを判断する（予測遅延時間が $md_{q.s}$ 以下であるか否かを判断）．アルゴリズム 1 は，クエリ q に対する受付制御のための処理関数 $isAcceptable$ を示す．入力パラメータは，クエリ q，リソースサブグラフ $R_{q.s}$（または拡張リソースグラフ $R^{i+}_{q.s}$），タスクグラフ G_q および限界遅延時間 $md_{q.s}$ である．出力は，クエリ q が受け入れ可能かどうかを示すブール値である．クエリ q が受け入れ可能な場合，最初のリソース割り当て X_0 が，グリーディアルゴリズムに従って生成される．

アルゴリズム 1 では，変数 $P, X, CPD, falsecount$ が初期化される（2 行目～5 行目）．ここで，P は割り当てられていないプロセッサの集合，X は要素が $x_{t,p}$，CPD は，タスクを実行できるプロセッサが見つからない場合の計算資源の不足分の合計値，$falsecount$ は誤割り当ての数を示している．6 行目では，md_c, md_p, md_a と表記されているタスクの収集，処理，および集計の限界遅延時間 md_{qs} を分割することによって決定される．ここで，それぞれの分割率は経験的に決定された値（事前に与えられている値）を使用する．7 行目から 16 行目では，タスク T_q とプロセッサ P の間のタスク割り当てが決定される．具体的には，遅延制約を満たす t から p の代入が見つかった場合，$x_{t,p}$ は 1 に設定され，p は P から削除される（9 行目～12 行目）．それ以外の場合，t の実行に必要な計算リソースは CPD に加算され，$falsecount$ が 1 増加する（13～15 行目）．最後に，17 行目～21 行目で，$falsecount$ の値に応じて true（最終タスク割り当て X_0 も記録される．）または false（CPD も同様）が返される．

X_0 は，$isAcceptable$ の生成された行列であり，アルゴリズム 2 に使用される．CPD は適応的スケールアウトに使用される（図 4 のステップ 2b）．$CPD > 0$ の場合，S ブローカーは拡張リソースグラフ R^{i+}_s に対して i を増加させる．これによって図 5 (b). 示すように，CPD で表される計算資源の不足を補うことが可能になる．

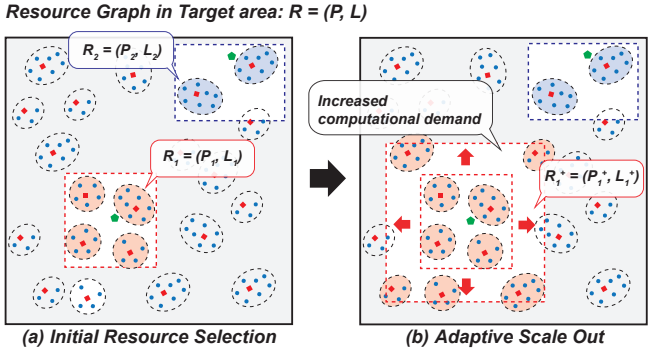

図 5: 初期リソース確保と適応型スケールアウトの様子

4.5 地産地処タスクスケジューリング

クエリ q が $isAcceptable$ によって受け入れられると，タスクグラフ G_q がスケジュールされる．このステップでは，S ブローカーは，ユーティリティ関数 $\sum_{q \in Q} U(q)$ の合計を可能な限り増加させるために，$isAceptable$ によって導出された解よりも，優れたタスク割り当てプランを見つけようとする．そこで，本研究では，メタヒューリスティック検索を用いて初期割り当てプラン X_0 を改善するアルゴリズムを提案する．具体的には，タブー検索アルゴリズム [6] を使用してタスク割り当てプランを調整する．

アルゴリズム 2 は，タブー探索に基づいて提案された現場タスクスケジューリング手法を示す．入力パラメータは，クエリ q 対するタスクグラフ G_q，リソースサブグラフ $R_{q.s}$，および $isAcceptable$ によって導出された初期タスク割り当て X_0 である．出力は，タブー探索によって調整されたタスク割り当てプラン X_q である．

アルゴリズム 2 では，最初に変数 $X, X^*, Tabu, cnt$ が初期化される（1～4 行目）．ここで X, X^* は現在のタスク割り当てプランと最適な割り当てプランである．$Tabu$ はタブー行列で cnt は反復回数を制御するカウンタを示す． $p(t)$ はタスク t が割り当てられているプロセッサを

Algorithm 1 アドミッション制御アルゴリズム

Require: $G_q = (T_q, E_q)$, $R_{q.s} = (P_{q.s}, L_{q.s})$, $md_{q.s}$
Ensure: Bool Value that indicates if a query is acceptable, X_0 (if acceptable), CPD (if not acceptable)

```
 1: procedure ISACCEPTABLE(G_q, R_{q.s}, md_{q.s})
 2:     P ← P_{q.s}
 3:     X ← O
 4:     CPD ← 0
 5:     falsecount ← 0
 6:     determine values of md_c, md_p, md_a so that md_c + md_p + md_a = md_{q.s}
 7:     for all t ∈ T_q do
 8:         md ← md_c/md_p/md_a depending on type of t
 9:         find p ∈ P s.t. proctime(t, p) ≤ md
10:         if such p is found then
11:             x_{t,p} ← 1
12:             P ← P\{p}
13:         else
14:             CPD ← CPD + comp(t)
15:             falsecount ← falsecount + 1
16:         end if
17:     end for
18:     if falsecount = 0 then
19:         X_0 ← X
20:         return true
21:     else
22:         return false
23:     end if
24: end procedure
```

示す．メインプロセスは 5 行目〜28 行目にあり，5 行目の $MAXCNT$ は反復回数を指定する定数である．8 行目では，最良近傍解 X' と近傍解 $\bar{X}$ が X で初期化される． 9 行目では，変数 t' と p' が初期化されている．ここで，t' と p' は，それぞれ T_q と X で割り当てられたプロセッサからランダムに選択されたタスクである.10 行目〜19 行目のループは，現在の解 X で 1 つのタスク割り当てを変更することによって生成されたすべての近傍解 $\bar{X}$ の中から選択された最良の近傍解 X' を見つける．具体的には，p に t を実行するための残りの計算能力（$emcap()$）があれば，すべてのタスク割り当て $x_{t,p(t)} = 1$ は，新しい割り当て $\bar{x}_{t,p} = 1$ に変更される（12〜13 行目）．次に，新しいタスク割り当てプランによってタスクグラフを実行するための全体的なユーティリティが増加すると，X' が更新される（14-17 行目と 20 行目）．14 行目では，$Utility(X)$ は，現在のタスク割り当てプラン X に基づいてタスクグラフを実行するときの QoE を推定する関数である（タスクグラフを処理するための遅延が計算され，その後，遅延に基づいてユーティリティが計算される）．22 行目〜27 行目では，最良解 X^* は，現在の最良解 X^* よりも優れている場合，最良近傍解 X' によって更新される．反復が停止した後，現在の最良解 X^* が最終解として X_q に代入される（29〜30 行目）．

Algorithm 2 タブー探索に基づく地産地処タスクスケジューリング

Require: $G_q = (T_q, E_q)$, $R^{i+}_{q.s} = (P^{i+}_{q.s}, E^{i+}_{q.s})$, $X_{q.init}$
Ensure: X_q: Task Assignments

```
 1: /*Initialization*/
 2: X, X* ← X_0
 3: Tabu ← O
 4: cnt ← 0
 5: while cnt ≤ MAXCNT do
 6:     reduce_best ← ∞
 7:     /* obtain the best neighbor plan*/
 8:     X', X̄ ← X
 9:     t' ← randselect(T_q), p' ← p(t')
10:     for all t ∈ T_q do
11:         for all p ∈ P^{i+}_{q.s} | p ≠ p(t) do
12:             if Tabu_{t,p} = 0 & comp(t) ≤ remcap(p) then
13:                 x̄_{t,p(t)} ← 0, x̄_{t,p} ← 1
14:                 gain ← Utility(X̄) − Utility(X)
15:                 if gain > gain_best then
16:                     gain_best ← gain
17:                     t' ← t, p' ← p
18:                 end if
19:             end if
20:         end for
21:     end for
22:     x'_{t',p(t')} ← 0, x'_{t',p'} ← 1
23:     /* update the best plan*/
24:     if gain_best ≥ 0 then
25:         if Utility(X') > Utility(X*) then
26:             X* ← X'
27:         end if
28:     else
29:         Tabu_{t',p(t')} ← 1
30:     end if
31:     X* ← X'
32:     cnt ← cnt + 1
33: end while
34: X_q ← X*
35: return X_q
```

5. 評価実験

本章では，提案手法の有効性を評価するために，シミュレーションを用いた比較実験を行う．以下では，実験環境について説明した後に，比較手法と評価指標ついて述べ，最後に，実験結果とその考察を示す．

5.1 実験環境

提案アルゴリズムの有効性を明らかにするために，図 6 のような仮想的な IoT 環境を作成し，シミュレーションによる比較実験を行った．比較実験に用いたシミュレータは SimPy と NetworkX という Python フレームワークを使って実装されている．シミュレーションは，R ブローカ，IoT デバイス（*M-nodes*，*S-nodes*），地域 IoT サービス，およびクエリで構成されている．M ノードおよび S ノードは，それぞれ高性能のラップトップなどのハイパワーな IoT デ

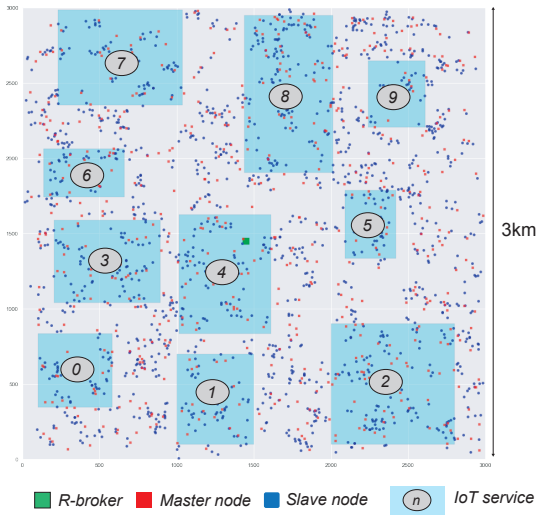

図 6: シミュレーション上の地域 IoT エリア

バイス，Raspberry Pi などのローパワーな IoT デバイスを想定してモデル化されている．シミュレーションでは，対象の地域エリアのリソースグラフ $R = (P, L)$，サービスの集合 S，各サービスのサービスエリア $area(s)$，各サービスに対するクエリの集合 Q_s，各クエリに対応するタスクグラフの集合 $\cup_{q \in Q_s} G_q (= (T_q, E_q))$ が与えられる．各クエリ q とそれに対応するタスクグラフ G_q は，ポアソン到着プロセスに基づいて生成される．

シミュレーションのパラメータを表 1 に示す．R-ブローカー，M ノード，S ノード，IoT サービスの数は，それぞれ 1, 600, 1400, および 10 です．各デバイスの処理能力およびデバイス間の通信速度は，これまでの研究での実測値 [19], [20] に基づいて決定されている．各サービスに対して，クエリは $\lambda = 1/(min)$ のポアソン到着過程に基づいて発行され，各クエリに対応するタスクグラフの入力データサイズは，1〜20(MB) の範囲でランダムに選択される．これは，クエリで指定された AoI（Area of Interest）から

表 1: シミュレーションのパラメータ

Parameter	Value(s)
Simulation duration	10 (h)
Number of IoT Services	10
Preferable delay for each service	10 (s)
Marginal delay for each service	30 (s)
Query arrival rate for each service λ	1 (min)
Raw sensor data size per each query	1 to 20 (MB)
Number of R-broker	1
Computation amount of Collecting task	500
Computation amount of Processing task	2000
Computation amount of Aggregation task	1000
Processing power of R-broker	10000
Processing power of M-nodes	1000 to 3000
Processing power of S-nodes	500 to 1000
Network speed of M-node to R-broker	5 to 50 (Mbps)
Network speed of S-node to M-node	50 to 500(Mbps)

表 2: 各サービスエリアの面積

Area name	Area(km^2)
A	9.0
A_0	0.235
A_1	0.300
A_2	0.640
A_3	0.376
A_4	0.471
A_5	0.150
A_6	0.169
A_7	0.511
A_8	0.597
A_9	0.165

生成されるセンサデータのサイズと時空間解像度パラメータから算出された値を想定したものである．

5.2 評価手法と評価指標

シミュレーションによって，評価する手法として，(1) ランダムタスクスケジューリングと (2) グリーディタスクスケジューリングという 2 つのベースライン手法と 2 パターンの提案手法 (3)(4) を設定した．

(1) ランダムタスクスケジューリング: 各サービスのリソースサブグラフからランダムにプロセッサを選出し，タスクを割り当る．

(2) グリーディタスクスケジューリング：各サービスのリソースサブグラフからグリーディ手法に基づいてプロセッサを選択し，タスクを割り当る．

(3) タブー探索に基づくタスクスケジューリング: 各サービスのリソースサブグラフから 4.5 節で説明したタブー探索アルゴリズムに基づいてプロセッサを選択し，タスクを割り当る．

(4) タブー探索に基づくタスクスケジューリング+適応型現場スケールアウト．（3）の手法に加えて，サービスの負荷状況に応じて適応型スケールアウトを実施し，リソースサブグラフの拡張を行う．

各手法のパフォーマンスを評価するための評価指標は，遅延時間，受け入れられたクエリの数，および効用関数の値である．効用関数の値は，pd=10(s) および md=30(s) の効用関数曲線に基づいて計算される．

5.3 評価結果と考察

5.3.1 遅延時間

図 7 は，クエリを処理するための遅延時間の累積分布関数（CDF）曲線を示している．各手法の平均遅延時間は，ランダム：17.83 秒，グリーディ：14.95 秒，提案手法 1 ：9.79 秒，提案手法 2 ：10.07 秒である．この結果は，提案手法 1，2 は，ランダムやグリーディよりも遅延時間が短いことを示している．また，遅延時間の標準偏差は，ラン

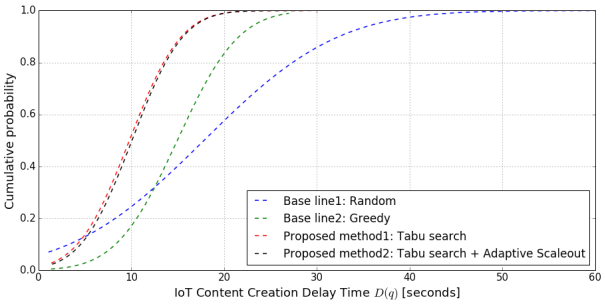

図 7: 各種法における遅延時間の累積分布関数（CDF）曲線

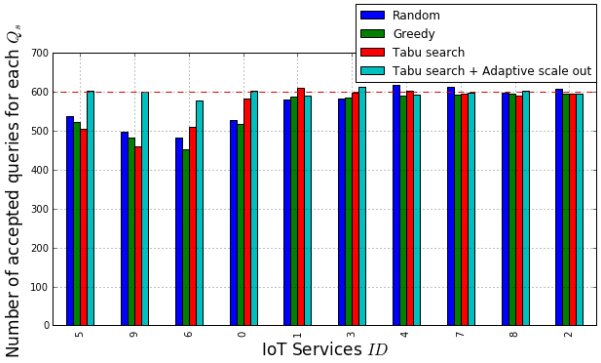

図 8: 各サービスで受託されたクエリの数

ダム：11.33 秒，グリーディ：5.19 秒，提案手法 1 ：4.38 秒，提案手法 2 ：4.30 秒である．この結果は，提案手法 1 ，2 は，ランダムやグリーディよりも各クエリに対する遅延時間の差が小さいこと示している．

5.3.2 受託されたクエリの数

図 8 は，各サービスで受託されたクエリの数を表している．x 軸はサービスエリアサイズの小さい順に並べられたサービス ID である（表 2 を参照）．各手法の受託クエリの総数は，ランダム：5532 件，グリーディ：5526 件，提案手法 1 ：5660 件，提案手法 2 ：5981 件である．この結果は，スケールアウトする提案手法 2 が，他の手法よりも多くのクエリを受諾できることを示している．図 1 に示すように，ベースラインの手法や提案手法は，領域サイズが小さい（サービス ID = 5,9,6,0）ときに受け入れられたクエリの数が減少することを示している．一方，提案手法 2 では，サービスエリアのサイズによらず，ほぼ同数のクエリを受け付ける．この結果は，適応型スケールアウトを使用することによって，面積の小さいサービスでも多くのクエリを受け入れることができることを示している．

シミュレーション中の適応型スケールアウトのスナップショットを図 9 に示す．赤い領域は，スケールアウトされた領域を表す．初期状態の場合，適応型スケールアウトは，まだサービスエリアにも適用されていない．その後，時間が経過と共に一部のサービスが計算資源選択エリアをスケールアウトしている様子が確認できる．中間状態の場合，サービス A_0，A_5，A_6，A_9 はサービスエリアを拡大している．最後状態では，広いサービスエリアを持つサービス A_7 も拡大していることが確認できる．結果として，

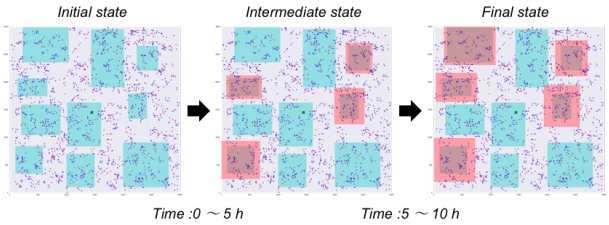

図 9: 適応型現場スケールアウトのスナップショット

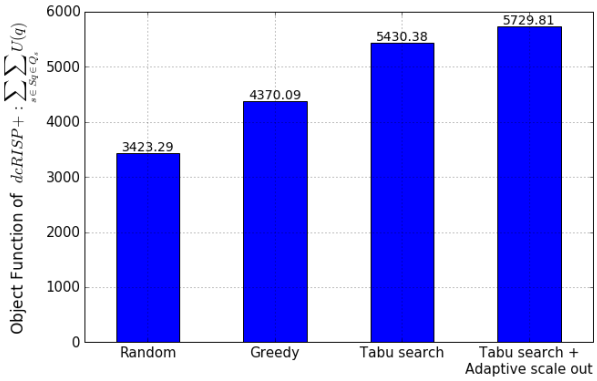

図 10: 全てのクエリ Q_A に対する効用関数値の合計 $U(q)$

サービスエリアが小さめのサービスであっても，適応型スケールアウトを実施することによって，計算資源選択エリアを適応的に拡張し，サービスエリアが大きいサービスと同等の数のクエリに対処することが可能になっている．

5.3.3 効用関数の値

図 10 は，4 つの手法のすべてのクエリに対する効用関数の値の合計を示している．結果は，提案された方法 1 および 2 がベースライン方法よりも QoE の観点で良いパフォーマンスを得られることを示唆している．また，適応型現場スケールアウトを行う提案手法 2 は，提案手法 1 よりも効用関数の値の合計が上回っており，より良い QoE を達成していると言える．

5.3.4 評価結果のまとめ

評価結果を以下のように要約する．

- タブー探索を用いた提案手法 1 ， 2 は，各クエリに対する遅延時間を短くすることを確認した（Fig. 7）．遅延時間の短縮は，サービスの質の向上を意味する．このことから，タブー探索に基づく地産地処タスクスケジューリングは，各クエリに対するサービスの質の向上に寄与している．
- 適応型スケールアウトを用いた提案手法 2 は，クエリの受託数を増加することを確認した（図 8）．クエリ受託数の増加は，サービス量の向上を意味する．このことから，適応型スケールアウトはサービス量の向上に寄与している．
- 提案手法 2 は，タブー探索によって各クエリに対するサービスの質を，適応型スケールアウトによってサービスの量を向上させている．その結果，最も高い QoE を得ることができた．

6. おわりに

本論文では, 対象の地域エリアに存在する IoT デバイスを活用して，より QoE（Quality of Experience）の高い IoT サービスを提供するための資源確保およびタスク割り当て手法として，(1) IoT サービスの計算需要に応じて動的に計算資源の選択エリアを拡張する適応型スケールアウトと (2) タブー探索に基づく地産地処タスクスケジューリングからなる適応型地産地処リソース配分手法を提案した．提案手法の有効性を確認するために，2000 台の IoT デバイスが展開された地域エリアを想定したコンピュータシミュレーションを行い，近いエリアに存在するデバイス群を対象としたタブーサーチに基づくタスクスケジューリングによって，コンテンツ生成おける遅延時間を短縮し，適応的な現場スケールアウトを実施することで各サービスにおけるクエリの受託数を増加させることを確認した．その結果，提案手法がベースラインの手法と比べて，より高い QoE を達成することを明らかにした．今後は，時間帯別の混雑などを想定したシミュレーションを実施するとともに，実機デバイスを用いた検証を行い，提案手法のオーバーヘッドなどの評価を行う予定である．

謝辞 本研究の一部は，JSPS 科研費（17J10021, 16H01721, 26220001）の支援を受けて実施されたものである．

参考文献

[1] Evans, D.: The internet of things: How the next evolution of the internet is changing everything, http://blogs.cisco.com/news/the-internet-of-things-infographic/.

[2] Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions, *Future generation computer systems*, Vol. 29, No. 7, pp. 1645–1660 (2013).

[3] Bonomi, F., Milito, R., Zhu, J. and Addepalli, S.: Fog Computing and Its Role in the Internet of Things, *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, New York, NY, USA, ACM, pp. 13–16 (online), DOI: 10.1145/2342509.2342513 (2012).

[4] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P. and Riviere, E.: Edge-centric computing: Vision and challenges, *ACM SIGCOMM Computer Communication Review*, Vol. 45, No. 5, pp. 37–42 (2015).

[5] Yasumoto, K., Yamaguchi, H. and Shigeno, H.: Survey of Real-time Processing Technologies of IoT Data Streams, *Journal of Information Processing*, Vol. 24, No. 2, pp. 195–202 (online), DOI: 10.2197/ipsjjip.24.195 (2016).

[6] Porto, S. C. and Ribeiro, C. C.: A tabu search approach to task scheduling on heterogeneous processors under precedence constraints, *International Journal of high speed computing*, Vol. 7, No. 01, pp. 45–71 (1995).

[7] Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D. et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, *Journal of Parallel and Distributed computing*, Vol. 61, No. 6, pp. 810–837 (2001).

[8] Page, A. J. and Naughton, T. J.: Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing, *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, IEEE, pp. 8–pp (2005).

[9] Shen, C.-C. and Tsai, W.-H.: A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion, *IEEE Transactions on Computers*, Vol. 100, No. 3, pp. 197–203 (1985).

[10] Salman, A., Ahmad, I. and Al-Madani, S.: Particle swarm optimization for task assignment problem, *Microprocessors and Microsystems*, Vol. 26, No. 8, pp. 363–371 (2002).

[11] Zhu, Q. and Agrawal, G.: Resource provisioning with budget constraints for adaptive applications in cloud environments, *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ACM, pp. 304–307 (2010).

[12] Mao, M. and Humphrey, M.: Auto-scaling to minimize cost and meet application deadlines in cloud workflows, *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, p. 49 (2011).

[13] Ardagna, D., Ciavotta, M. and Passacantando, M.: Generalized nash equilibria for the service provisioning problem in multi-cloud systems, *IEEE Transactions on Services Computing*, Vol. 10, No. 3, pp. 381–395 (2017).

[14] Xu, J., Palanisamy, B., Ludwig, H. and Wang, Q.: Zenith: Utility-aware resource allocation for edge computing, *Edge Computing (EDGE), 2017 IEEE International Conference on*, IEEE, pp. 47–54 (2017).

[15] Skarlat, O., Schulte, S., Borkowski, M. and Leitner, P.: Resource provisioning for IoT services in the fog, *Service-Oriented Computing and Applications (SOCA), 2016 IEEE 9th International Conference on*, IEEE, pp. 32–39 (2016).

[16] Skarlat, O., Nardelli, M., Schulte, S. and Dustdar, S.: Towards QoS-aware fog service placement, *Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on*, IEEE, pp. 89–96 (2017).

[17] Abdeljaouad, I. and Karmouch, A.: Monitoring IPTV quality of experience in overlay networks using utility functions, *Journal of Network and Computer Applications*, Vol. 54, pp. 1–10 (2015).

[18] Brucker, P., Drexl, A., Möhring, R., Neumann, K. and Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods, *European journal of operational research*, Vol. 112, No. 1, pp. 3–41 (1999).

[19] Nakamura, Y., Suwa, H., Arakawa, Y., Yamaguchi, H. and Yasumoto, K.: Design and Implementation of Middleware for IoT Devices toward Real-Time Flow Processing, *Distributed Computing Systems Workshops (ICDCSW), 2016 IEEE 36th International Conference on*, IEEE, pp. 162–167 (2016).

[20] Nakamura, Y., Suwa, H., Arakawa, Y., Yamaguchi, H. and Yasumoto, K.: Middleware for Proximity Distributed Real-Time Processing of IoT Data Flows, *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, IEEE, pp. 771–772 (2016).