令和元年度

修士学位論文

エッジコンピューティング環境における モバイル端末の移動予測を利用した リソース割当て手法

Resource allocation algorithm using mobile device movement prediction in edge computing environment

1225125　　福永 昂輝

指導教員　　横山 和俊

2020 年 2 月 28 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報学コース

要 旨

エッジコンピューティング環境における
モバイル端末の移動予測を利用した
リソース割当て手法

福永 昂輝

IoT 機器の普及により，データセンタへ送られるデータ量が日々増加している．それらのデータはネットワークを介してクラウド上のデータセンタに送信されるため，実時間処理が必要なアプリケーションには，通信遅延が問題となる．この問題に対し，ネットワークの周縁部にエッジサーバを設置することで分散処理を実現し，通信遅延時間の削減，サーバやネットワークの負荷を低減するモバイルエッジコンピューティングの研究がおこなわれている．しかし，豊富なリソースを持つデータセンタと比較して配置可能なエッジサーバは貧弱なため，一部のエッジサーバに負荷が集中するとリソース不足が発生する可能性がある．この課題に過去研究では，モバイル端末と通信しているエッジサーバの近傍にあるエッジサーバを利用する集中アルゴリズムを提案した．しかし，全てのモバイル端末のリソース割当てを一括管理するのは現実的ではない．本研究では，各エッジサーバが割当ての決定を処理する分散アルゴリズムを提案し，有効性をシミュレータを作成し，評価する．

キーワード　キーワード モバイルクラウドコンピューティング，モバイルエッジコンピューティング，分散処理

Abstract

Resource allocation algorithm using mobile device movement prediction in edge computing environment

Koki Fukunaga

As mobile devices increase, the amount of data sent to data centers is increasing every day. Since such data is transmitted to a data center on the cloud via a network, communication delay is a problem for applications that require real-time processing. For this problem, mobile edge computing has been studied in which distributed processing is realized by edge servers at the edge of the network to reduce communication delay time and reduce the load on the server and the network. Mobile edge computing places edge servers around the network and performs distributed processing. Mobile edge computing can improve real-time processing and reduce server and network load. However, edge servers are poorer than resource-rich data centers, so a heavy load on some edge servers can result in resource shortages. Previous research has proposed a centralized algorithm that uses an edge server near the edge server that communicates with the mobile device. However, it is not practical to manage resource allocations for all mobile devices together. In this research, we propose a distributed algorithm and create a simulator to evaluate its effectiveness.

key words Mobile cloud computing, Mobile edge computing, Distributed processing

目次

図目次

表目次

第 1 章

はじめに

近年，スマートフォンなどのモバイル端末やセンサーなどの IoT 機器が増加し，インターネットを通してクラウド上のデータセンタに送信されるデータが年々増加している．シスコシステムズの調査によれば，世界中のモバイル端末によるデータトラフィック量は 2016 年から 2021 年にかけて 7 倍に増加し，2021 年には月間 48.3EB に達する見込みになる [1]．このような大量データをデータセンタにサーバを集中配置し，アプリケーションの要求を処理する従来のクラウドコンピューティングでは，モバイル端末が集中した地域において，実時間処理が重要なサービスや通信トラフィックの増加や計算能力の不足に対応できない可能性がある．そこで，アプリケーションの実行をデータセンタではなく，ユーザーにより近いネットワーク周縁部に計算機を配置し，処理を行うというモバイルエッジコンピューティングを利用する研究が進められている [2][3]．しかし，モバイルエッジコンピューティングには課題が多く，特に重要な問題としては，データセンタのように豊富なリソースを持たないため，モバイル端末が集中し，処理負荷が増えるとリソースが不足してしまう可能性がある．過去研究では，この問題に対し，モバイル機器の位置から許される遅延時間で通信できる近傍のモバイルエッジコンピューティングサーバ (以下 MEC サーバと略す) を利用し，モバイル端末の要求リソースを割り当てることで MEC サーバのリソースの枯渇を抑えること，モバイル端末と MEC サーバ間の通信遅延を小さくすることを目的とするリソース割当てアルゴリズムの提案を行った [4]．しかしながら，過去研究でのリソース割当てアルゴリズムは，集中アルゴリズムで全てのモバイル端末をデータセンタにて制御する中央制御型にしていたが，モバイル端末の増加に合わせて，集中管理するのは現実的ではなく，多くの計算機と大容量のネットワークが必要になる．それらを改善するには各 MEC サーバが割当て

を判断し，近傍の MEC サーバへ割当てを依頼する自律分散型の処理をして中央へのシグナリングを削減する必要がある [5]．そこで，本研究では，各 MEC サーバが割当ての決定を処理する分散アルゴリズムを提案し，有効性をシミュレータを作成し，評価する．また，より正確な評価を行いたいため，岡山駅周辺の道路ネットワーク上を走らせた車両データと KDDI の基地局配置を利用したシミュレーションを作成し，評価を行う．

第 2 章

研究背景

本章では，モバイルクラウドコンピューティングとモバイルエッジコンピューティングの相違点について説明する．

2.1 モバイルクラウドコンピューティング

クラウドとは，利用者に提供されるサービスや利用形態は様々になるが NIST(アメリカ国立標準技術研究所) が定めたクラウドコンピューティングの定義によると，クラウドコンピューティングとは，共用の構成可能なコンピューティングリソース (ネットワーク，サーバ，ストレージ，アプリケーションサービス) の集積に，どこからでも，簡単に，必要に応じて，ネットワークを経由してアクセスすることを可能とするモデルであり，最小限の利用手続きまたはサービスプロバイダとのやり取りで速やかに割り当てられ提供されるものと定義されている [6]．クラウドコンピューティングのメリットとして，使用したい機能を好きな期間サービスとして利用できたり，データセンタ上で集中管理することで効率的に更新作業や保守作業を行うことでコストを抑えられるなどが挙げられる [7]．

図 2.1 にモバイルクラウドコンピューティングモデルを示す．モバイルクラウドコンピューティングとは，クラウドコンピューティングを利用してアプリケーションをモバイル端末に配信する技術である．図 2.1 のようにネットワークエッジのモバイル端末は基地局を介してクラウド上のデータセンタを利用することで，モバイル端末のリソース不足を改善することができる技術である．しかしながら，モバイル端末が年々増加傾向にある中，大量のデータが送信されるとクラウドでの集中型管理では，ネットワークの過負荷や遅延が大きくなり，

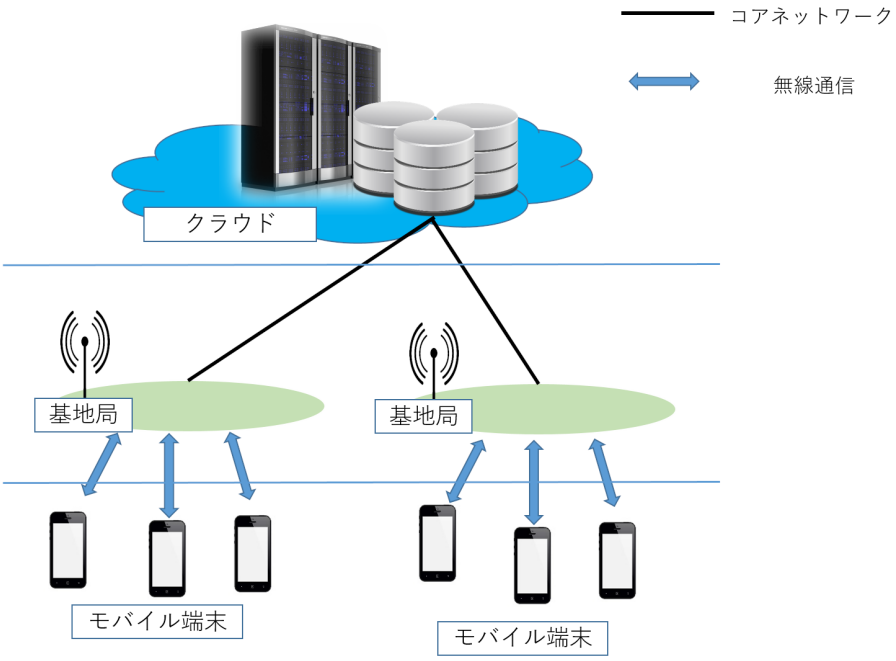

図 2.1　モバイルクラウドコンピューティングモデル

サービスを提供できない可能性がある．

2.2　モバイルエッジコンピューティング

モバイルエッジコンピューティングとは現在，European Telecommunications Standards Institute (ETSI) で標準化されている新技術のことである [8]．ETSI によると，モバイルエッジコンピューティングはモバイルネットワークのエッジ，無線アクセスネットワーク内に近接した環境でクラウドコンピューティング機能を提供することと定義されている．図 2.1 にモバイルエッジコンピューティングモデルを示す．図 2.1 のように，モバイルエッジコンピューティングはクラウドとモバイル端末の間に存在するレイヤーで，クラウド，モバイルエッジコンピューティングサーバ (MEC サーバ)，モバイル端末の三層階層とされている．モバイルエッジコンピューティングは計算機をアクセスポイント，基地局などに展開し，モバイル端末の処理をエンドユーザーに近い場所で処理することでモバイルクラウドコンピューティングの問題点であるネットワークの過負荷や遅延を改善することができる [9]．

2.2　モバイルエッジコンピューティング

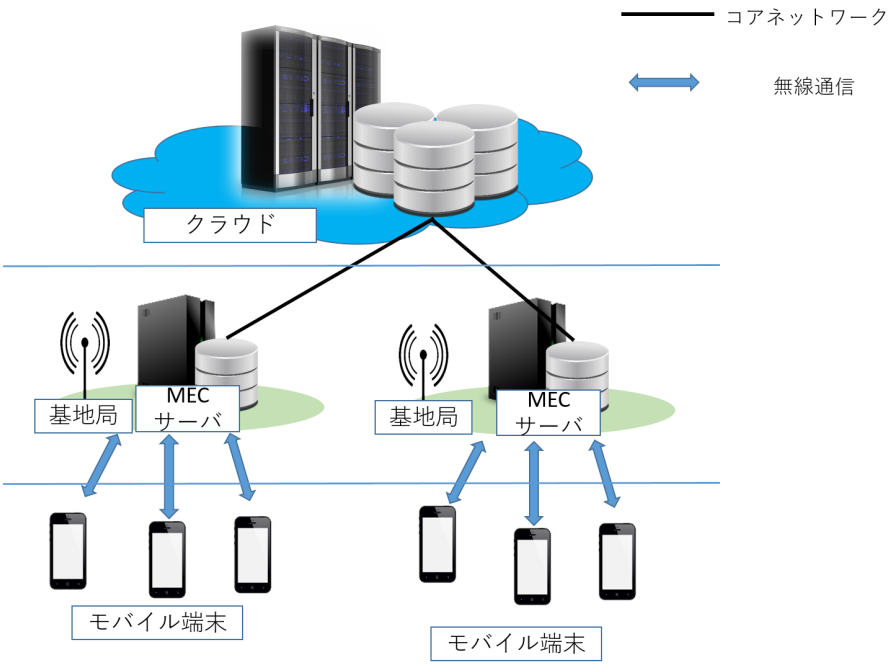

図 2.2　モバイルエッジコンピューティングモデル

第 3 章

関連研究とこれまでの取り組み

3.1 関連研究について

以下に関連研究について説明する．関連研究の種類として，クラウド環境でのリソース割当てアルゴリズムとモバイル端末の移動を考慮したリソース割当てアルゴリズムの 2 点を述べる．

(1) クラウド環境でのリソース割当てアルゴリズム

クラウド環境下でのリソース割当てアルゴリズムに関する研究として [10][11][12] がある．

文献 [10] では，複数のアプリケーションが物理サーバで動作する際，物理サーバの CPU 使用率の和が想定を超えてしまう可能性があることから仮想マシンを物理サーバ間を移送させるが，仮想マシンの移送はコストがかかることが多い．そのことから仮想マシンの移送コストが極力少ない厳密アルゴリズムの提案を行なっている．

文献 [11] では，仮想マシンの集約効率をベクトルパッキング問題として定式化し，古典的ヒューリスティックなアルゴリズムである First-Fit，Best-Fit，Worst-Fit，Next-Fit を改良したアルゴリズムを提案している．

文献 [12] では，スマートフォンなどの移動端末には，処理能力，電気容量などに制限がかかってしまうため，移動端末がクラウドサービスを効率良く利用できるシステムが必要になることから，移動端末の要求データをより近いネットワーク上にプロキシサーバを配置し，データをキャッシュしておくことで応答時間の改善を行っている．

文献 [10][11][12] は，クラウド上のデータセンタ内の仮想マシンや物理サーバのリソースの使用率が閾値を超えないようにリソースを割り当てるアルゴリズムであり，モバイルクラウドコンピューティングには適応できる可能性があるが，モバイル端末の増加によりネットワークの遅延や，負荷の軽減が求められる現在では，適応できない．また，モバイル端末の移動性を考慮する必要がある．

(2) モバイル端末の移動を考慮したリソース割当てアルゴリズム

モバイル端末の移動を考慮したリソース割当てアルゴリズムとして文献 [13][14][15] がある．

文献 [13] では，スマートフォンなどの移動端末には，処理能力，電気容量などに制限がかかってしまうため，移動端末がクラウドサービスを効率良く利用できるシステムが必要になることから，移動端末の要求データをより近いネットワーク上にプロキシサーバを配置し，データをキャッシュしておくことで応答時間の改善を行っている．

文献 [14] では，モバイル端末が Cloudlet 環境で動作するには，モバイル端末の移動先にある Cloudlet 環境に対し，アプリケーションの実行環境が必要になることから，実行環境を移送しなければならない．しかし，移動先の Cloudlet 環境でリソースが不足した場合，実行環境が移送できずサービスの継続ができない場合がある．この問題を改善するため，予め Cloudlet に移送用のリソースを予約することを提案している．

文献 [15] では，Cloudlet やモバイルアドホックネットワークベースのクラウドコンピューティング環境においてオンラインスケジューリングとバッチスケジューリングを適応し，評価している．オンラインスケジューリングは他のジョブやスケジューリングを考慮せずにジョブが到着するとすぐにスケジュールされる手法で，バッチスケジューリングは所定の時間間隔後、タスクをグループ化し、スケジューリングを行う手法である．

文献 [13][14][15] はいずれも，ネットワークエッジにリソース配置し，ユーザに近い場所で処理することで応答時間を改善する研究であるが，リソースを大きく超えるような

負荷を考慮していない．モバイルエッジコンピューティングにおいて，モバイル端末は移動体であるため，なんらかのイベントが発生し，ネットワークや計算機に過負荷がかかることを想定する必要がある．

3.2　過去研究

共同で研究していた大崎らの Cloudlet において移動端末を対象したアルゴリズムの提案を過去研究とする [4]．文献 [4] は，Cloudlet 環境におけるリソース割当て手法としてモバイル端末の割当て順序を決定するアルゴリズムとモバイル端末の割当て先を決定するアルゴリズムを提案し，シミュレーションにて評価している．しかし，この研究には以下の 2 点の問題がある．

(1) 集中アルゴリズムで全てのモバイル端末を一括管理

図 3.1 に過去研究でのモバイル端末のリソース要求の流れを示す．過去研究は，集中アルゴリズムでモバイル端末のリソース要求を管理し，割り当てる順序や割当て先を決定していた．しかしながら，非常に多くのモバイル端末に対して，高性能な計算機が必要かつ大容量なネットワークが必要になるため，自律分散的に処理できる分散アルゴリズムが今後適している．よって本研究では，分散アルゴリズムの提案を行う．

(2) 基地局の配置やモバイル端末の移動網が格子状

過去研究では，基地局の配置が 2 次元の格子状になっており，モバイル端末の移動も基地局の配置に合わせて格子状の上を上下左右するモデルだった．よって本研究では，より厳密に評価したいと考え，モデルの変更を行う．

3.2　過去研究

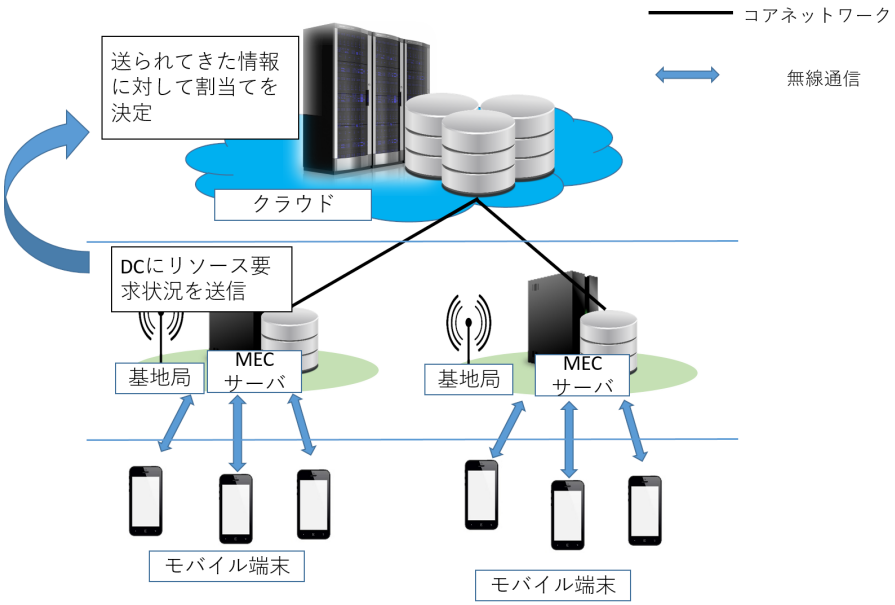

図 3.1　モバイル端末のリソース要求の流れ

第 4 章

システムモデル

システムモデルを図 4.1 に示す．図 4.1 では，ネットワーク周縁部の基地局に MEC サーバを配置し，その基地局の情報を集約する集約局，集約局の情報をデータセンタへ送信する中間局が仲介となり，基地局間やデータセンタとの通信が実現される．また，基地局には通信可能範囲があり，モバイル端末は基地局の通信可能範囲に入っていれば通信が可能になる．以下，システムモデルについて説明する．

(1) MEC サーバ

MEC サーバは基地局に配置され，図 4.1 のように各基地局に MEC サーバを配置する．各 MEC サーバを m_i で表す．各 MEC サーバ m_i にはそれぞれ所有リソース量が設定されており，MEC サーバ m_i の所有リソースが s とすると，$S(m_i)$ で求められる..

$$S(m_i) = s$$

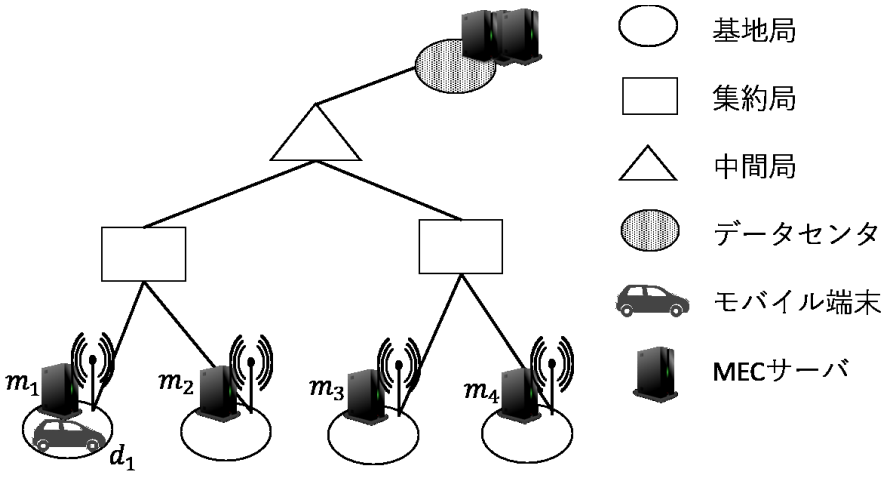

図 4.1　システムモデル

また，実際の環境では，所有リソースは記憶容量や，CPU コア数など様々な種類のリソースがあるが，複数種類を考慮すると問題が複雑になるため，ここではリソースを 1 種類に限定する. また，実際に割り当てられ，所有リソースから割り当てたリソース量を引いた値を割当て可能リソース量 R_i とする．MEC サーバの R_i を超えない限り，モバイル端末のリソース要求を処理できる．

$$V(m_i) = R_i$$

最後に，MEC サーバには位置情報があり，$L(m_i)$ で求められる．

$$L(m_i) = (x, y)$$

(2) モバイル端末

モバイル端末は道路ネットワーク上を走行する．図 4.2 のようにモバイル端末が道路上を走行し，基地局の通信範囲内であれば通信が可能になる．また，モバイル端末を d_j で表し，各モバイル端末 d_j には通信している MEC サーバに要求するリソース量が設定されている．あるモバイル機器 d_j の要求リソース量を r と表すと，$R(d_j)$ で求められる．また，モバイル端末は道路ネットワーク上で，最も近い基地局と通信する．

$$R(d_j) = r$$

また，表 4.1 のようにモバイル端末は直近 t 秒間の走行履歴を持っており，直近 t 秒間の移動距離 s，平均方位角 α がわかる．モバイル端末の直近 t 秒間の移動時間 s，平均方位角 α は，$P(d_j, t)$ で求められる．またモバイル端末には，位置情報があり，$Ld(d_j)$ で求められる．

$$P(d_j, t) = s, \alpha$$

$$Ld(d_j) = (x, y)$$

(3) 周辺 MEC サーバ

各基地局の MEC サーバ m_i は，半径 $\ell.km$ 以内の周辺の MEC サーバと定期的に情報

表 4.1　走行履歴の例

時刻	移動速度	方位角
$0s$	0	180.461097
$1s$	1.304631	180.461097
$2s$	2.962383	180.461097
$3s$	4.93081	180.461097
$4s$	6.837284	180.461097

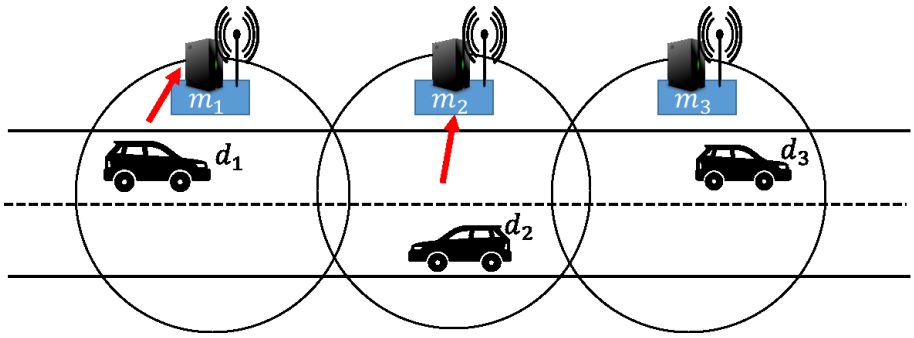

図 4.2　モバイル端末の走行

交換しており，周辺 MEC サーバの情報を得ることができる．図 4.3 に例を示す．図 4.3 では，m_1 は半径 $\ell.km$ 以内の緑の円で囲まれた中の周辺 MEC サーバを知ることができ，m_2, m_3, m_4, m_5, m_6 の周辺 MEC サーバを知ることができる．周辺 MEC サーバを m_n で表すと，$Am(m_i)$ で求められる．また，各 MEC サーバ m_i は通信しているモバイル端末の全ての要求リソース量がわかる．全ての要求リソース量を S_i とすると，$Ar(m_i)$ で求められる．また，各 MEC サーバ m_i は $A(m_i)$ で周辺 MEC サーバ m_n を求め，$Ar(m_n)$ で周辺 MEC サーバ m_n の S_n を知ることができる．また，MEC サーバと周辺 MEC サーバの距離は近いため，情報伝達にかかる遅延がないものとする．

$$Am(m_i) = m_n$$

$$Ar(m_i) = S_i$$

(4) 割当て

割当ては一定時間ごとに各 MEC サーバ m_i が判断する．図 4.4 に例を示す．m_1 は，

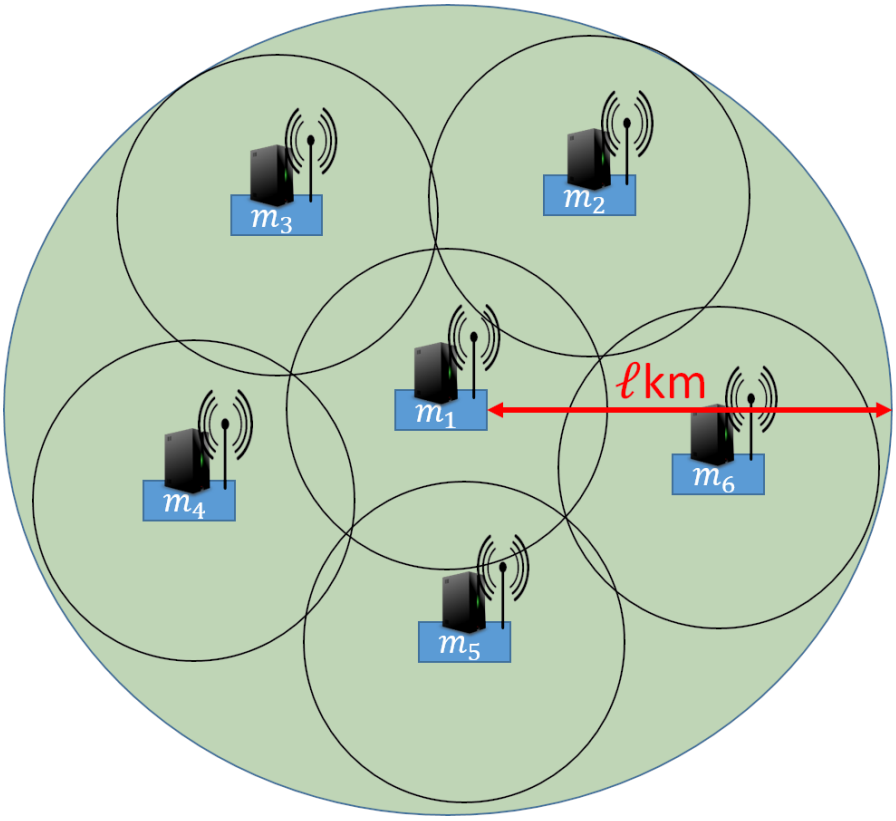

図 4.3　周辺 MEC サーバの範囲

m_2, m_3, m_4, m_5, m_6 の周辺 MEC サーバを知ることができ，モバイル端末が最も距離が近い m_1 に対してリソース要求している．この時 m_1 は m_6 に対して，割当てを依頼する．m_6 は依頼を受けた後，割当て可能なら割り当てるが，割当て可能ではなかったので，割当て失敗としてデータセンタへ割当てを依頼をする．各 MEC サーバ m_i は周辺 MEC サーバ m_n に対して割り当てたい場合は，対象の周辺 MEC サーバ m_n に対して割当てを依頼する．割当て依頼を受けた周辺 MEC サーバ m_n は自身の R_n を超えない場合は割り当てるが，超えた場合に割当て失敗として DC に割当てを依頼する．

(5) 遅延時間

システムモデルでの遅延時間をホップ数で表す．モバイル端末のリソース要求を割り当てる場合，様々な転送・中継設備を経由するほど通信遅延が発生する．本研究では，1 つの転送・中継設備を経由することを 1 ホップとする．図 4.1 を例に説明する．モバイル端末 d_1 は MEC サーバ m_1 と通信しており，MEC サーバ m_1 が割当て先を自身に決

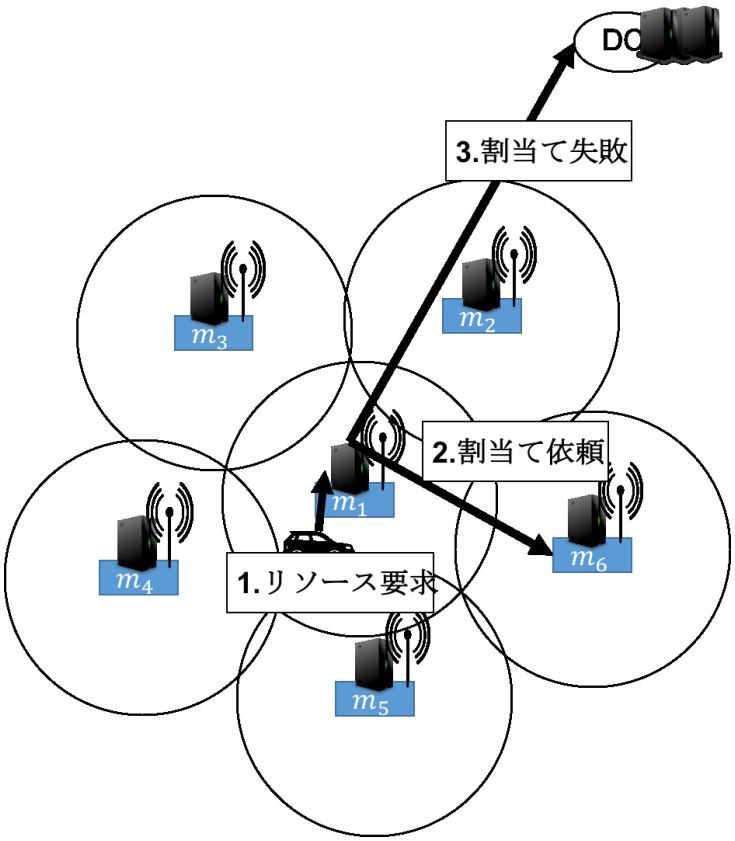

図 4.4　リソース要求の割当て順序

定した場合，基地局を経由するので，1 ホップになる．割当て先を m_2 にした場合，基地局，集約局，基地局を経由するので 3 ホップになる．割当て先をデータセンタにした場合，基地局，集約局，中間局，データセンタを経由するので 4 ホップになる．

システムモデルで用いた記号の定義を表 4.2 にまとめる．

表 4.2　システムモデルで用いる記号の定義

記号	意味
m_i	MEC サーバ
$S(m_i)$	MEC サーバ m_i の所有リソース
$V(m_i)$	MEC サーバ m_i の割当て可能リソース量
$L(m_i)$	MEC サーバ m_i の座標
d_j	モバイル端末
$R(d_j)$	モバイル端末 d_j の要求リソース量
$P(d_j, t)$	直近 t 秒間のモバイル端末 d_j の移動距離 s と平均方位角
$Am(m_i)$	MEC サーバ m_i の周辺 MEC サーバ m_n
$Ar(m_i)$	MEC サーバ m_i の全ての要求リソース量 S_i

第 5 章

分散アルゴリズムの設計

5.1 分散アルゴリズムの設計方針

以下に，リソース割当てアルゴリズムの設計方針を示す.

(1) 分散アルゴリズム

過去研究では，モバイル端末からのリソース要求を一括管理していたが，本研究では，モバイル端末のリソース要求は通信している MEC サーバがどこで処理するかを判断する分散アルゴリズムを設計する．また，分散アルゴリズムを設計する中で，平均ホップ数を抑えることを目標とする．平均ホップ数はホップ数の総数に対して割当ての総回数を割ったもの.

(2) 周辺 MEC サーバの利用

本研究で検討する分散アルゴリズムでは，モバイル機器が利用できる MEC サーバは，モバイル端末が通信している MEC サーバのみではなく，通信している MEC サーバの判断によって周辺 MEC サーバを利用することができる.

(3) 平均ホップ数の抑制

平均ホップ数を減らすには割当て失敗回数をできるだけ減らすことが必要になる．これは，周辺 MEC サーバに割当てを依頼し，割当て失敗が発生した場合，割当て依頼がデータセンターに送られるため，ホップ数が最大になってしまうからである．したがって，割当て失敗をできるだけ減らす必要がある．どのような状況下で割当て失敗が発生するかを示す．図 5.1 では，モバイル端末が m_2 と m_4 とそれぞれ通信しており，リ

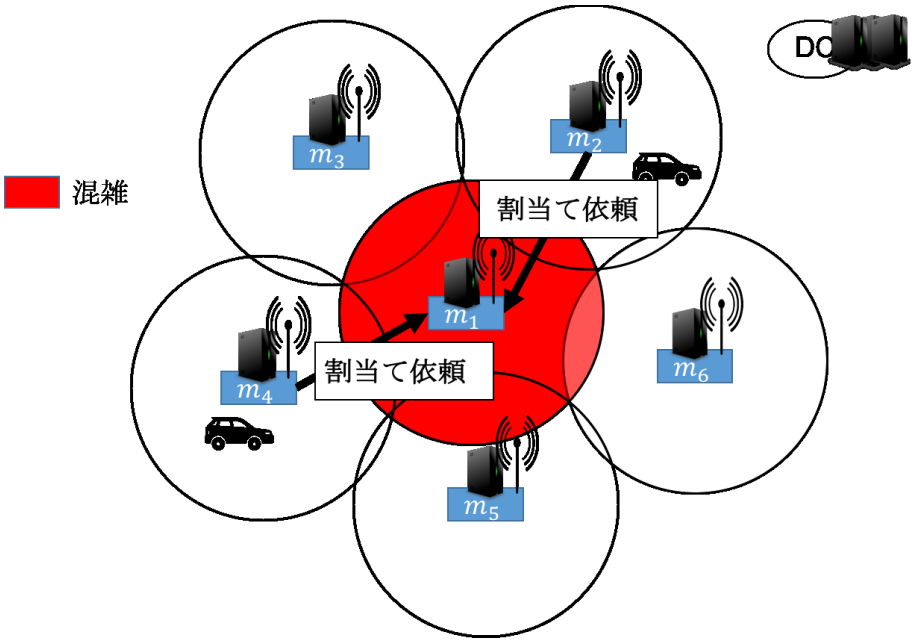

図 5.1　混雑状況における割当て依頼

ソース要求をしている．m_2, m_4 は割当て依頼を m_1 に依頼する判断を決定とするが，m_1 の通信エリアは非常に混雑しており，割当てを依頼した場合，割当て失敗が発生してしまう．このような場合，周囲の MEC サーバの混雑状況を判断し，割当て可能な MEC サーバに対して割当てを依頼するアルゴリズムが必要になる．

(4) 移動体の考慮

モバイル端末は移動体なので移動を考慮した割当てを行う必要がある．この理由は，モバイル端末の移動方向とまったく別の方向に割り当ててしまった場合に，次回の割当て開始前にホップ数が増えてしまう可能性があるからである．これは，割当てが一定時間ごとに MEC サーバが決定するため，一定時間の間は同じ MEC サーバに割り当て続けられるため発生する．どのような状況下で移動によるホップ数の増加が発生するかを示す．図 5.2 では，m_1 と通信しているモバイル端末が一定時間後の割当て開始時に m_2 の通信エリアに移動する予定である．また，モバイル端末が移動する前に m_1 が割当て処理を開始し，m_4 に割当てを依頼することを判断している．ここで，実際に割当て依頼が成功した場合，次回の割当て開始時まで，モバイル端末が m_1 と通信している間は，

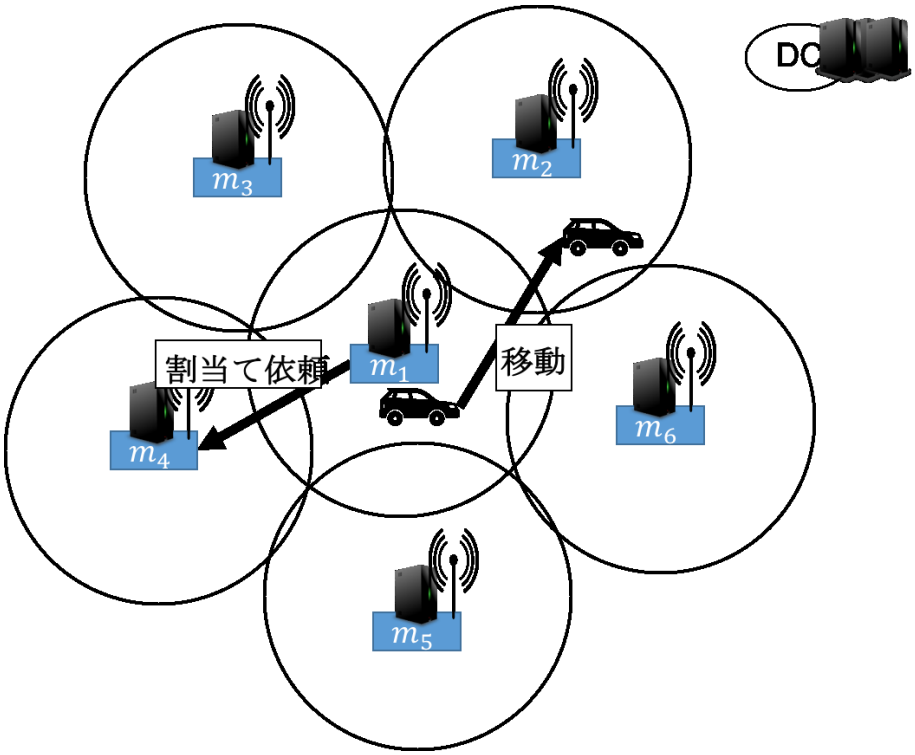

図 5.2　モバイル端末の移動に対しての割当て依頼

m_1 の基地局，集約局，m_4 の基地局を介して 3 ホップ，モバイル端末が移動し，m_2 に通信が切り替わった場合は，m_2 の基地局，集約局，m_4 の基地局を介して 3 ホップになる．しかしながら，最初に m_2 に割り当てていた場合，次回の割当て開始時まで，モバイル端末が m_1 と通信している間は，m_1 の基地局，集約局，m_2 基地局を介して 3 ホップ，モバイル端末が移動し，m_2 に通信が切り替わった場合は，m_2 の基地局のみで 1 ホップになり，ホップ数が減少する．このようにモバイル端末の移動を考慮した割当てアルゴリズムを設計することでホップ数を抑えることができる．

第 6 章

リソース割当てアルゴリズム

リソース割当てアルゴリズムは，各 MEC サーバが一定時間ごとに実行し，各モバイル端末に対して MEC サーバを割り当てる．その際，自身を含む，周辺 MEC サーバのうち，どの MEC サーバを選択するかを決めるアルゴリズムが重要になる．

6.1 従来手法

単純な手法として以下の 2 種類の方法を検討した.

(1) 最近傍割当て

最近傍割当ては，単純に物理的な距離が最も近い MEC サーバに割り当てる手法．各 MEC サーバは割当て開始時に次の順序で割当てを決定する．

（a）割当て処理開始

（b）各 MEC サーバは m_i は通信しているモバイル端末 d_j の要求リソース量 $R(d_j)$ が割当て可能リソース量 $V(m_i)$ を次式の条件を満たす場合，自身 m_i に割り当てて，処理を終了する．

$$V(m_i) - R(d_j) \geq 0$$

（c）条件 $V(m_i) - R(d_j) \geq 0$ を満たさなかった場合，周辺 MEC サーバ m_n の座標 $L(m_n)$ と自身 m_i の $L(m_i)$ のユークリッド距離 yl を計算する．ユークリッド距離は次式で求められる．

$$YL((x1, y1), (x2, y2)) = yl$$

（d）求めた yl の内，最小の周辺 MEC サーバ m_n に対して割当てを依頼する．

（e）割当て処理終了

(2) 最近傍・最小リソース量による割当て

最近傍・最小リソース量による割当ては，まず，モバイル端末が通信している MEC サーバに割り当て，その MEC サーバの割当て不可の場合に周辺 MEC サーバの中で全ての要求リソース量が最小の MEC サーバに割り当てる手法．各 MEC サーバは割当て開始時に次の順序で割当てを決定する．

（a）割当て処理開始

（b）各 MEC サーバは m_i は通信しているモバイル端末 d_j の要求リソース量 $R(d_j)$ と割当て可能リソース量 $V(m_i)$ を次式の条件を満たす場合，自身 m_i に割り当てて，処理を終了する．

$$V(m_i) - R(d_j) \geq 0$$

（c）条件 $V(m_i) - R(d_j) \geq 0$ を満たさなかった場合，周辺 MEC サーバ m_n の全ての要求リソース量 $Ar(m_n)$ が最小の MEC サーバに対して割当てを依頼する．

（d）割当て処理終了

6.1.1 従来手法の問題点

(1) 最近傍割当て

最近傍割当ての問題点はモバイル端末のリソース要求に対して，割当て先の MEC サーバが 2 つしかないこと．1 つは，モバイル端末が通信している MEC サーバで，2 つは，その MEC サーバの割当て可能リソース量が 0 になった場合，その MEC サーバから物理的な距離が最も近い MEC サーバだけである．この場合，一部の MEC サーバにリソース要求が多く送られたされた場合，割当て失敗が多く発生してしまう．

(2) 最近傍・最小リソース量による割当て

最近傍・最小リソース量による割当ては，最近傍のデメリットである割当て先が 2 つしかない中，モバイル端末の要求リソース量が最小の MEC サーバに割当てを依頼するため，最近傍割当てよりは比較的良くなると考えられる．しかし，分散アルゴリズムの

ため，他の MEC サーバがどこに割当てを依頼したかわからない．そのため，割当て依頼の量が増えた場合，割当て可能リソース量が最小の MEC サーバに対して他の MEC サーバが割当てを依頼してしまい，割当て失敗が増加してしまうと考えられる．

6.2 提案手法

提案手法では，割当て失敗を減らすこと，移動体であることを考慮し，モバイル端末の移動に対して割当て先を決定することを目的とし，次の手法を提案する．

6.2.1 走行履歴による割当て

この手法は，一定時間後のモバイル端末の移動先を走行履歴から予想し，移動先に近い MEC サーバに対して割当てを依頼する手法．また，割当て失敗を抑えるため，割り当てできない MEC サーバに対しては，依頼しないようにする．以下，走行履歴による割当てについて説明する．

(1) 推定移動場所

モバイル端末の走行履歴から移動先を予測し，移動場所を推定移動場所とする．推定移動場所の求め方を説明する．モバイル端末の直近 t 秒間を予測時間とし，平均方位角 α 方向に移動距離 s ほど動かした座標を推定移動場所 G とする．推定移動場所は $G(P(d_j, t))$ で求められる．推定移動場所は Vincenty の順解法を用いて算出する [16]．

$$G(P(d_j, t), Ld(d_j)) = (x, y)$$

(2) 割当ての流れ

各 MEC サーバは割当て開始時に次の順序で割当てを決定する．

（a）割当て処理開始

（b）各 MEC サーバは m_i は通信しているモバイル端末 d_j の走行履歴を参照し，直近 t 秒間の移動距離，平均方位角 $P(d_j, t)$ を抽出し，推定移動場所 $G(P(d_j, t,), Ld(d_j))$ を算出する．

（c）周辺 MEC サーバ m_n を取得する．取得した周辺 MEC サーバ m_n の座標 $L(m_n)$ と推定移動場所 $G(P(d_j, t,), Ld(d_j))$ のユークリッド距離 D_n を計算する．D_n は次式で求められる．

$$D_n = YL(L(m_n), G(P(d_j, t,), Ld(d_j)))$$

（d）取得した周辺 MEC サーバ m_n に対して割当て可能リソース量 $V(m_n)$ と全てのモバイル端末の要求リソース $Ar(m_n)$ を求める．

（e）周辺 MEC サーバ m_n とモバイル端末の推定移動場所との距離 D_n，周辺 MEC サーバ m_n の割当て可能リソース量 $V(m_n)$，m_n の全てのモバイル端末の要求リソース $Ar(m_n)$ を次の評価関数に入力する．

$$D_n + f(V(m_n), Ar(m_n))$$

関数 f は $V(m_n) > Ar(m_n)$ の時，0 を出力し，$V(m_n) < Ar(m_n)$ の時，ペナルティとなる値を出力する．

（f）評価関数で最も値が小さい周辺 MEC サーバ m_n に対して割当てを依頼する．

（g）割当て処理終了

関数 f によってペナルティを与え，割当て失敗を抑えることができ，評価関数によってモバイル端末の推定移動場所から近い MEC サーバに割り当てることができ，移動に合わせた割当てが可能になる．

分散アルゴリズムで用いた記号の定義を表 6.1 にまとめる．

表 6.1　分散アルゴリズムで用いる記号の定義

記号	意味
$YL((x1, y1), (x2, y2))$	座標 $(x1, y1)$ と座標 $(x2, y2)$ のユークリッド距離
$G(P(d_j, t), Ld(d_j))$	モバイル端末 d_j の推定移動場所
D_n	MEC サーバ m_n とモバイル端末 d_j の推定移動場所 G の距離
$f(V(m_n), Ar(m_n))$	ペナルティ関数
$D_n + f(V(m_n), Ar(m_n))$	走行履歴による割当ての評価関数

第 7 章

評価

7.1　評価内容

ここまで説明した手法の性能を調べるため，シミュレーションにより評価を行う．評価をより厳密にしたいため，表 7.1 の座標 4 点の内部の岡山駅周辺の道路ネットワークと基地局を調査し，割当て処理を実行するシミュレータの作成し，実行した．以下に，過去研究とのシミュレーションの評価の相違点について述べる．

7.1.1　過去研究と本研究の相違点

(1) 基地局配置と MEC サーバの配置

図 7.1 に過去研究の基地局配置と MEC サーバの配置を，図 7.2 に本研究の基地局配置を示す．過去研究と本研究では，基地局配置が異なる．過去研究では，図 7.1 のように，二次元の格子状に配置された各エリアに 1 つずつ基地局が配置されていたが，本研究では，岡山駅周辺の KDDI の基地局配置を利用する．KDDI の基地局配置は世界中の携

表 7.1　岡山駅周辺の座標 4 点

緯度	経度
34.673946	133.879051
34.629747	133.879051
34.629746	133.932266
34.629747	133.932266

帯基地局を公開する OpenCellID から，岡山駅周辺の表 7.1 の座標 4 点内部の KDDI の LTE 基地局配置を抽出し，利用した [17]．図 7.2 は岡山駅周辺の基地局を抽出し，マップ上に表示した図になる．青マークは 4G の基地局，赤マークは 3G に基地局を示す．本研究では，図 7.2 の基地局配置を利用し，各基地局に MEC サーバを配置し，シミュレーションを実行する．また，基地局の通信範囲も異なる．過去研究では，1 つのセル内が通信範囲であったが，本研究では，基地局の座標から半径 $500m$ としており，基地局同士の通信範囲の重複も考慮してある．

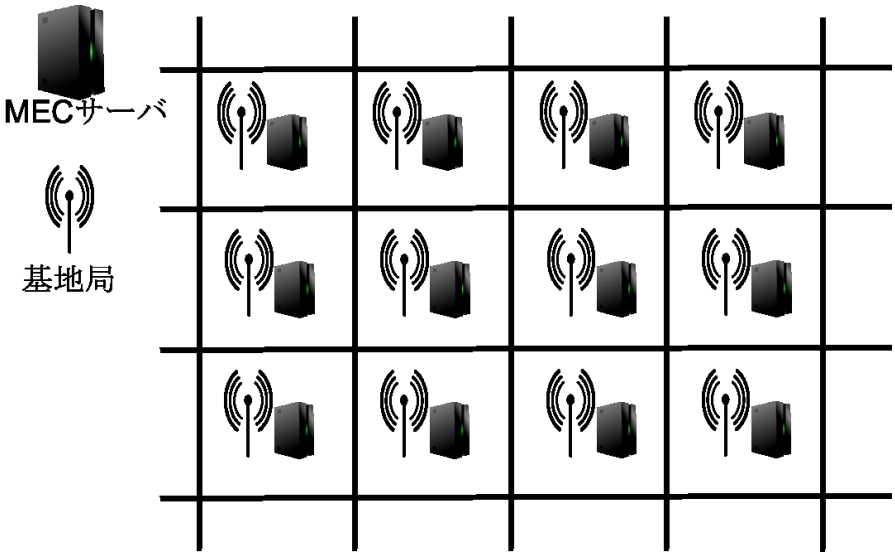

図 7.1　過去研究の基地局と MEC サーバの配置

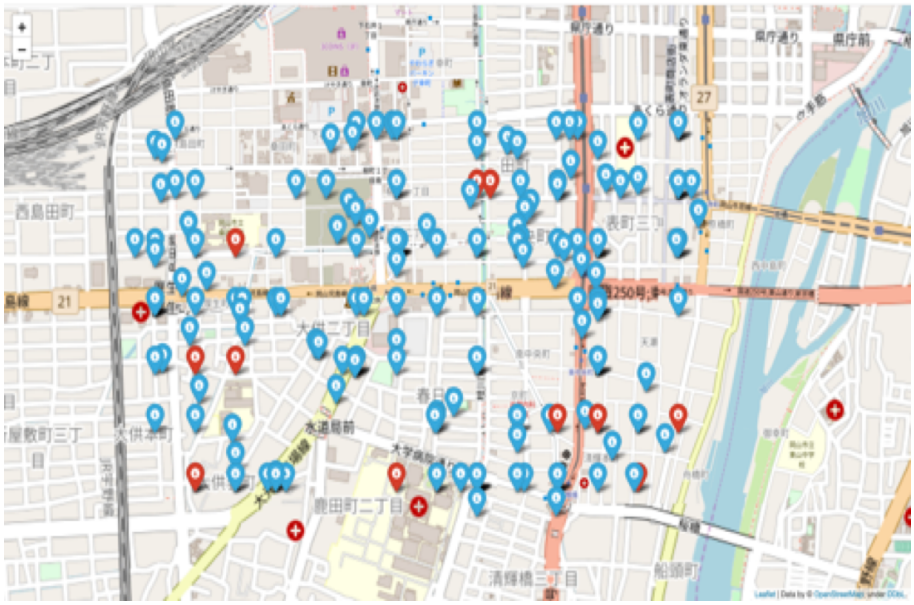

図 7.2　本研究の基地局配置

(2) 道路ネットワークとモバイル端末の経路

図 7.3 に過去研究の道路ネットワークとモバイル端末の経路を，図 7.4 に本研究の道路ネットワークを示す．まず，過去研究と本研究では，道路ネットワークが異なる．過去研究では，図 7.3 のように，二次元の格子状に配置された各エリアをモバイル端末が上下左右に動くが，本研究では，岡山駅周辺の表 7.1 の座標 4 点内部の道路ネットワーク

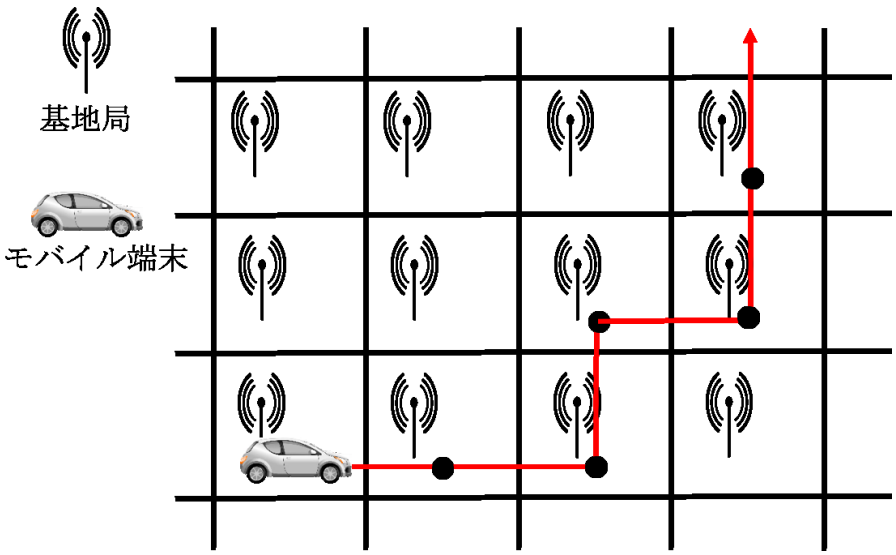

図 7.3　過去研究の道路ネットワークとモバイル端末の経路

上をモバイル端末動く．岡山駅周辺の表 7.1 の座標 4 点内部の道路ネットワークは自由な地図作成プロジェクトである OpenStreetMap を利用し，抽出した [18][19]．また，抽出した道路ネットワーク上でモバイル端末を動作させるために交通シミュレータである SUMO を利用した [20]．図 7.5 に交通シミュレータ SUMO に抽出した道路ネットワークを入力した図を示す．交通シミュレータ SUMO で実際に車両を生成し，走行させたデータをモバイル端末として利用した．

7.1　評価内容

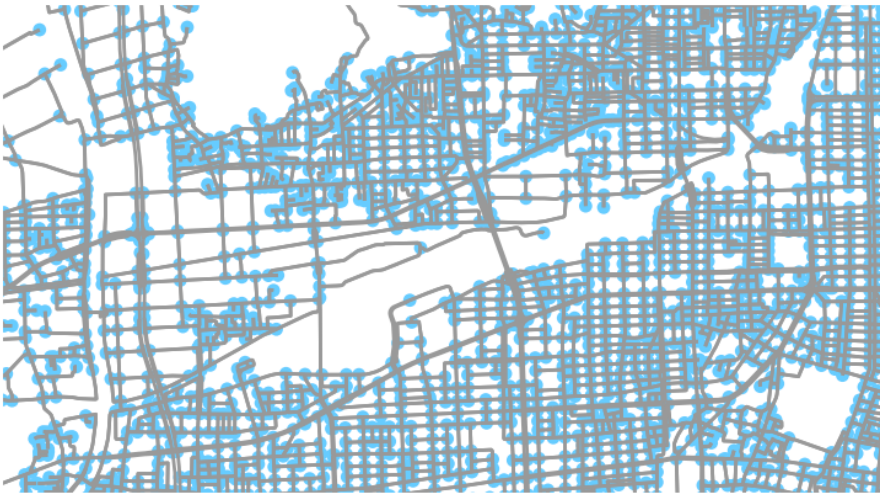

図 7.4　本研究の道路ネットワーク

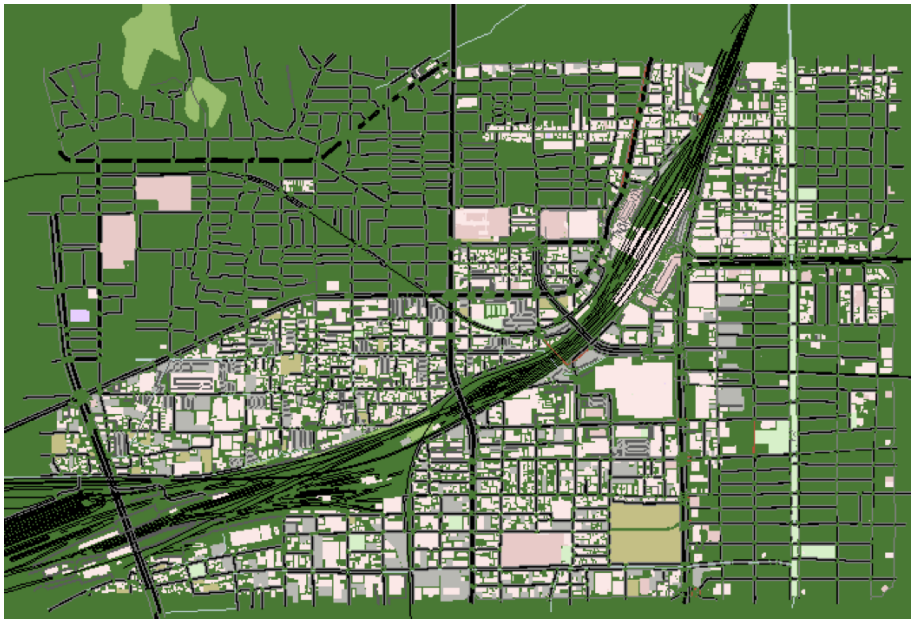

図 7.5　SUMO に入力した道路ネットワーク

7.2　評価方法

表 7.2 に実験パラメータを示す．基地局配置は，岡山駅周辺の基地局配置を利用する．また，基地局の数は 537 であったため，MEC サーバ台数は 537 台とする．道路ネットワークは図 7.4 の岡山駅周辺の道路ネットワークを利用し，交通シミュレータで道路ネットワーク上を動作させた車両データをモバイル端末として扱い，基地局と通信し，従来手法と提案手法の割当て処理をシミュレーションする．なお交通シミュレータ SUMO の車両生成によりモバイル端末数は 700 台ずつ増加させてシミュレーションする．シミュレーション時間は 1000 秒，基地局の通信範囲は文献 [21] を参考に 500m と設定する．また，文献 [22] を参考にモバイル端末の要求リソースは 50 から 500 のランダム，MEC サーバの所有リソースは 2500 とし，最大 50 台，最小 5 台のモバイル端末のリソース要求を処理できるように設定した．

上記の条件で最近傍割当て，最近傍・最小リソース量の割当て，走行履歴による割当てを比較する．また，各手法の割当て時間間隔を変化させた場合，走行履歴による割当ての予測時間を変化させて，シミュレーションする．評価項目としては，平均ホップ数と割当て失敗率とする．割当て失敗率は，割当て失敗回数を全ての割当て回数で割ったものとする．また，各手法の実行時間について評価する．

表 7.2　実験パラメータ

シミュレーション時間	1000 秒
MEC サーバ台数	537
モバイル端末数	700–4200
基地局の通信範囲	半径 500m
周辺の MEC サーバの範囲	半径 500m
モバイル端末の要求リソース	50–500 のランダム
MEC サーバの所有リソース	一律 2500

7.3　評価結果

以下に評価方法で述べたシミュレーションの結果と考察を示す.

7.3.1　割当て時間間隔を変化させた場合の結果と考察

割当て時間間隔を 10 秒，20 秒，30 秒と変化させシミュレーションし，比較した結果と考察を以下に示す．また，この時の走行履歴による割当ての予測時間は割当て時間間隔の半分，ペナルティ関数 f の値は 0 か 500 を出力させるよう設定した.

(1) 割当て時間間隔を変化させた場合の結果

（a）割当て時間間隔が 10 秒の時，図 7.6 に各割当て手法の平均ホップ数の結果を，図 7.7 に割当て失敗率を示す．図 7.6 より，平均ホップ数では，走行履歴による割当てが最も良いが，図 7.7 より，割当て失敗率はモバイル端末数が 2800 までは，最近傍・リソース量が良く，モバイル端末数が 2800 を超えると走行履歴による割当ての結果が良くなる.

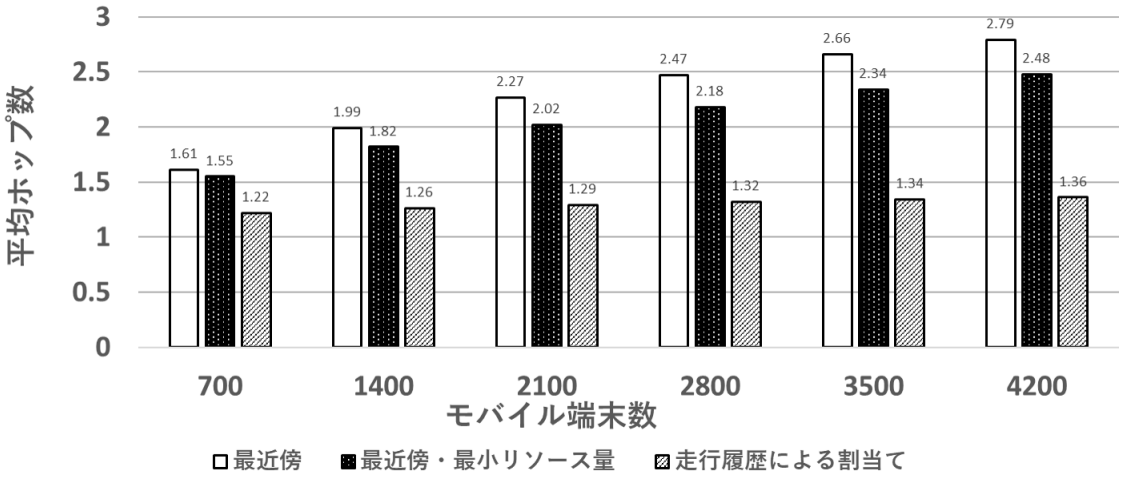

図 7.6　割当て時間間隔 10 秒，各割当て手法の平均ホップ数

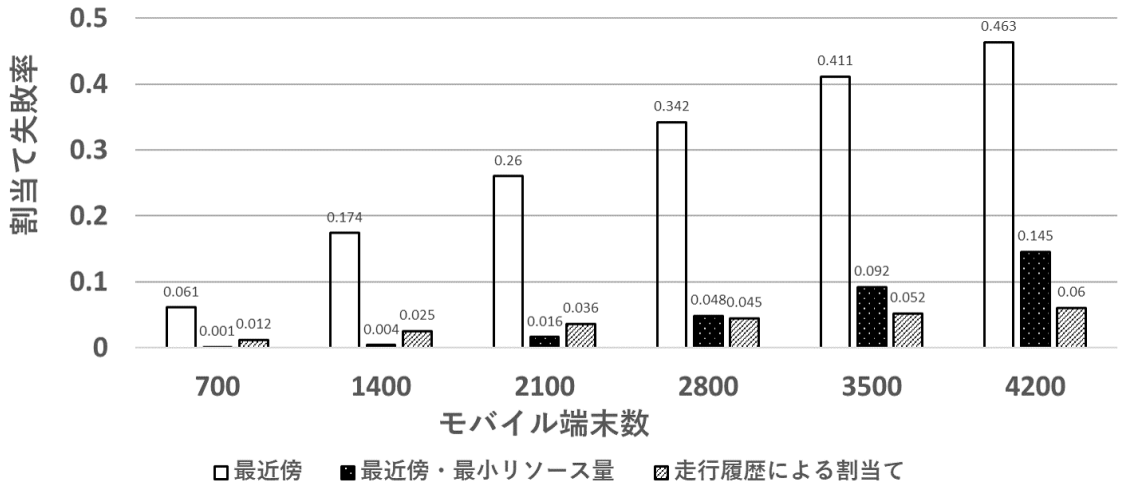

図 7.7　割当て時間間隔 10 秒，各割当て手法の割当て失敗率

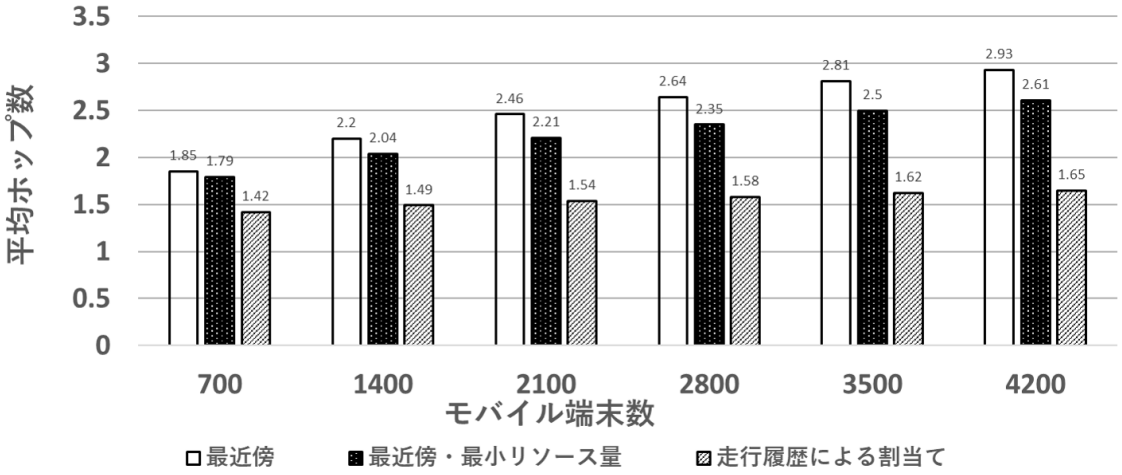

図 7.8　割当て時間間隔 20 秒，各割当て手法の平均ホップ数

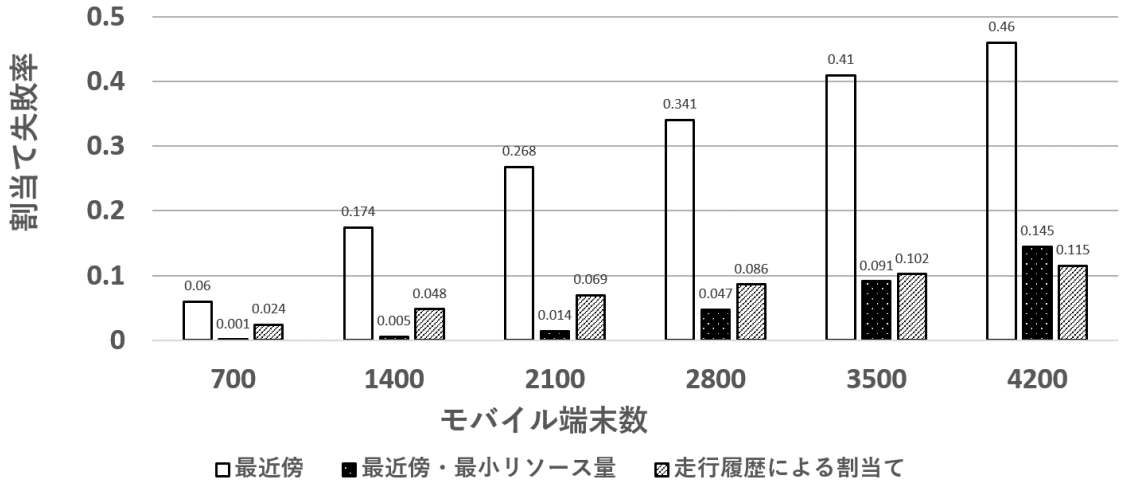

図 7.9　割当て時間間隔 20 秒，各割当て手法の割当て失敗率

（b）割当て時間間隔が 20 秒の時，図 7.8 に各割当て手法の平均ホップ数の結果を，図 7.9 に割当て失敗率を示す．図 7.8 より，平均ホップ数では，走行履歴による割当てが最も良いが，図 7.9 より，割当て失敗率はモバイル端末数が 3500 までは，最近傍・リソース量が良く，モバイル端末数が 3500 を超えると走行履歴による割当ての結果が良くなる．

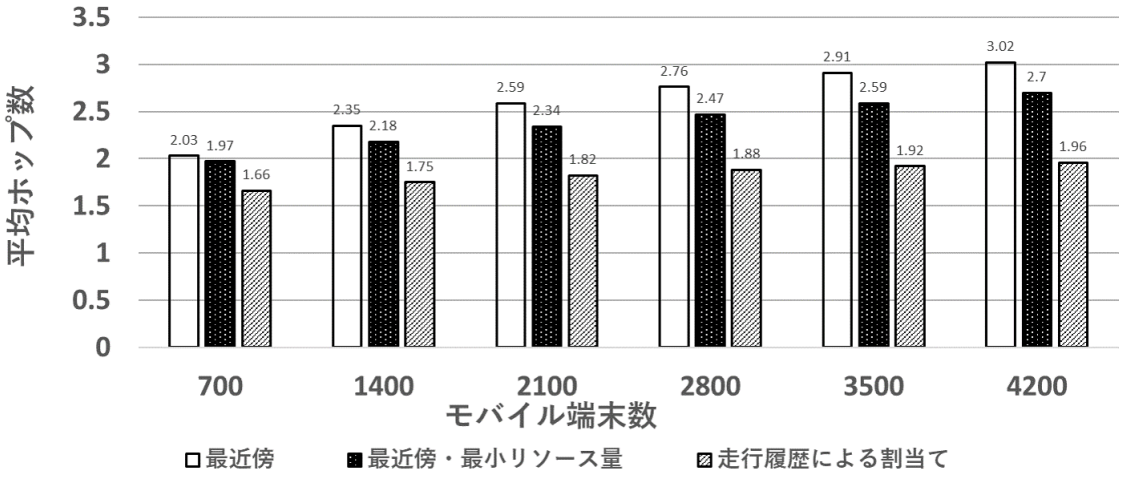

図 7.10　割当て時間間隔 30 秒，各割当て手法の平均ホップ数

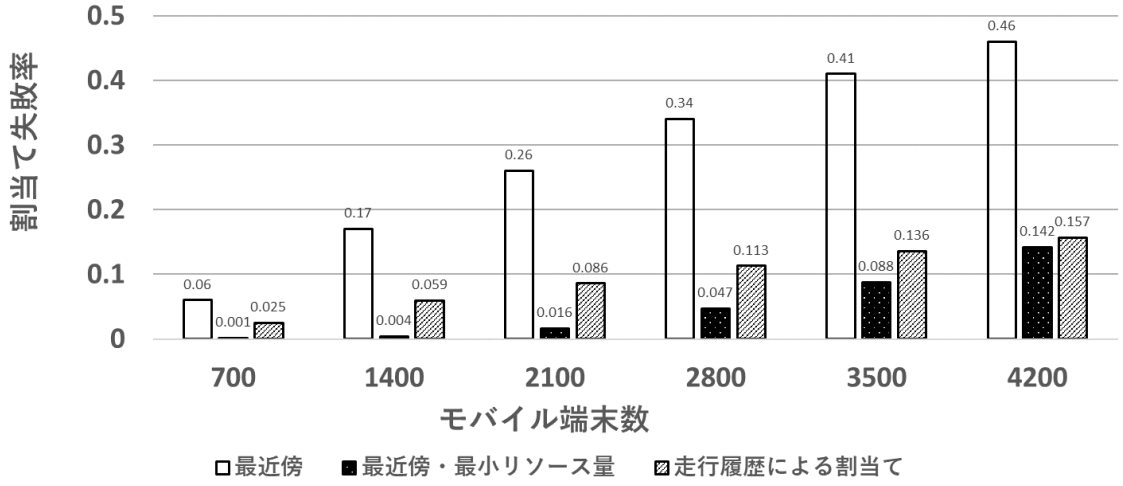

図 7.11　割当て時間間隔 30 秒，各割当て手法の割当て失敗率

（c）割当て時間間隔が 30 秒の時，図 7.10 に各割当て手法の平均ホップ数の結果を，図 7.11 に割当て失敗率を示す．図 7.11 に割当て失敗率を示す．図 7.8 より，平均ホップ数では，走行履歴による割当てが最も良いが，図 7.11 より，割当て失敗率は常に最近傍・リソース量が最も良い．

(2) 割当て時間間隔を変化させた場合の考察

図 7.6，図 7.8，図 7.10 より平均ホップ数において，走行履歴による割当てが最も良い理由は，移動を考慮しているため，割当てが決定し，次の割当て開始時までの間，1 ホップになることが多いからである．一方，図 7.7，図 7.9，図 7.11 より，割当て失敗率は，モバイル端末数が一定数までは，最近傍・最小リソース量が最も良く，一定数を超えると走行履歴による割当てが最も良くなる．この理由は，MEC サーバの割当て可能リソース量をモバイル端末のリソース要求が超える状態が増えると，全てのモバイル端末の要求リソース量が最も小さい MEC サーバに周辺 MEC サーバが割当てを多く依頼するため，割当て失敗が増加するからである．走行履歴による割当ては，全体的に割当て失敗が発生している．この理由は，多くのモバイル端末の移動方向が同じ場合は，同じ周辺 MEC サーバに割当てを依頼を送るため，割当て失敗が一定数発生するからである．また，割当て時間間隔が大きくなるにつれて，走行履歴による割当てが最近傍・最小リソース量よりも良くなるために必要なモバイル端末数が多くなる．この理由について各手法の時間間隔ごとに比較しながら説明する．

割当て時間間隔ごとに図 7.12 に最近傍・最小リソース量の割当て，図 7.14 に走行履歴による割当ての平均ホップ数の変化を示す．図 7.13 に最近傍・最小リソース量の割当て，図 7.15 に走行履歴による割当ての割当て失敗率の変化を示す．

図 7.12，図 7.13 より，最近傍・最小リソース量の平均ホップ数は割当て時間間隔によって約 0.1 から 0.2 ほど時間間隔が大きくなるにつれて差が生じている．一方，割当て失敗率は，ほぼ変化がない．平均ホップ数が増加する理由は，割当て時間間隔が大きくなることで，モバイル端末の移動距離も大きくなるため，1 ホップに割り当てていたが，モバイル端末が移動したため，3 ホップになってしまうから．割当て失敗率が変わらない理由は，時間間隔が大きくなったとしても 4 ホップ以上にならないため，つまり，割当て失敗の数が増えないからである．

図 7.14，図 7.15 より，走行履歴による割当ての平均ホップ数は割当て時間間隔によって約 0.2 から 0.3 ほど時間間隔が大きくなるにつれて差が生じている．一方，割当て失

7.3　評価結果

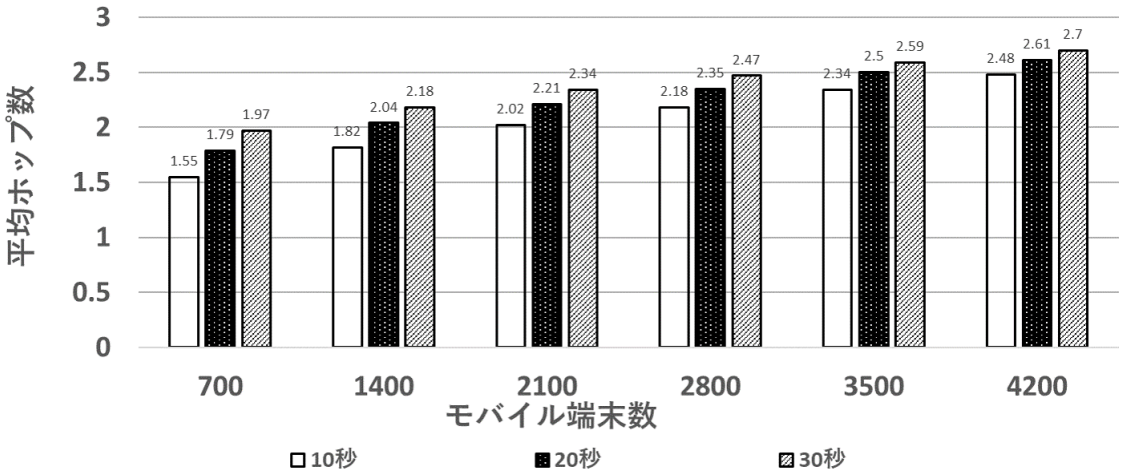

図 7.12　時間間隔ごとの最近傍・最小リソース量の割当ての平均ホップ数の変化

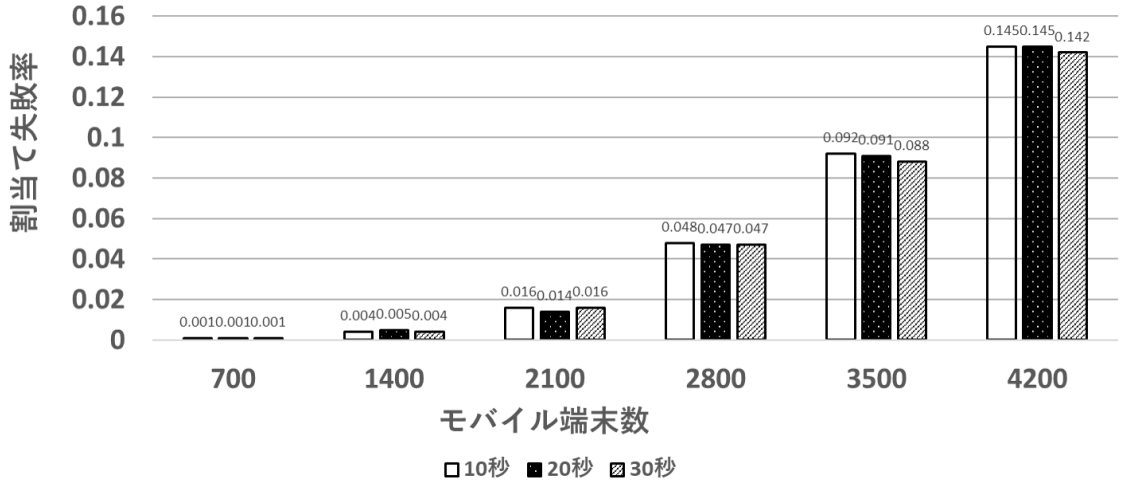

図 7.13　時間間隔ごとの最近傍・最小リソース量の割当ての割当て失敗率の変化

敗率は，時間間隔が大きくなるに比例して高くなっている．平均ホップ数も割当て失敗率も時間間隔の変化によって最近傍・最小リソース量よりも結果が悪くなる．この理由は，推定時間によるものだと考えられる．なぜなら，割当て時間間隔の半分の秒数が推定時間と設定しており，推定時間が大きくなるにつれて推定移動場所とモバイル端末が移動した場所との差が大きくなるからである．

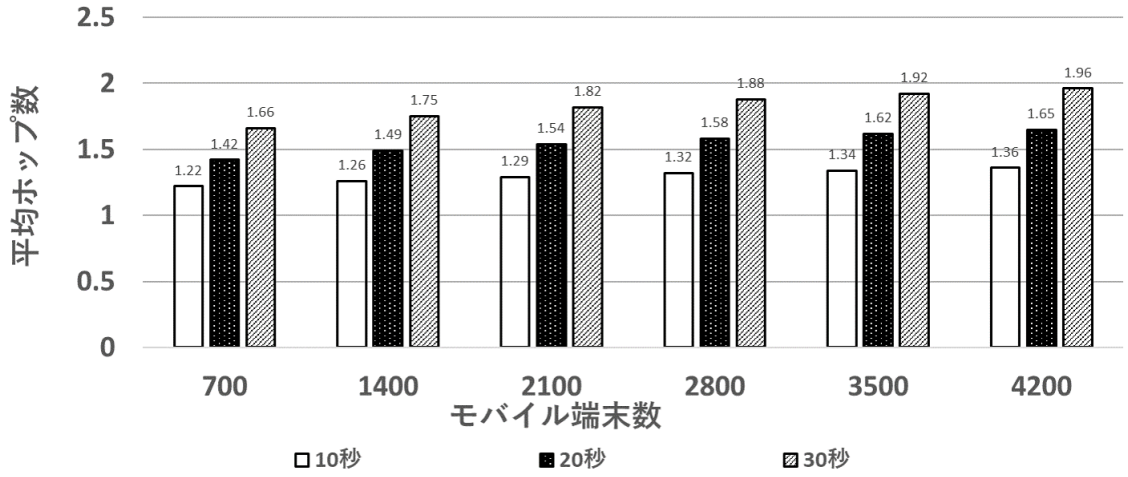

図 7.14　時間間隔ごとの走行履歴による割当ての平均ホップ数の変化

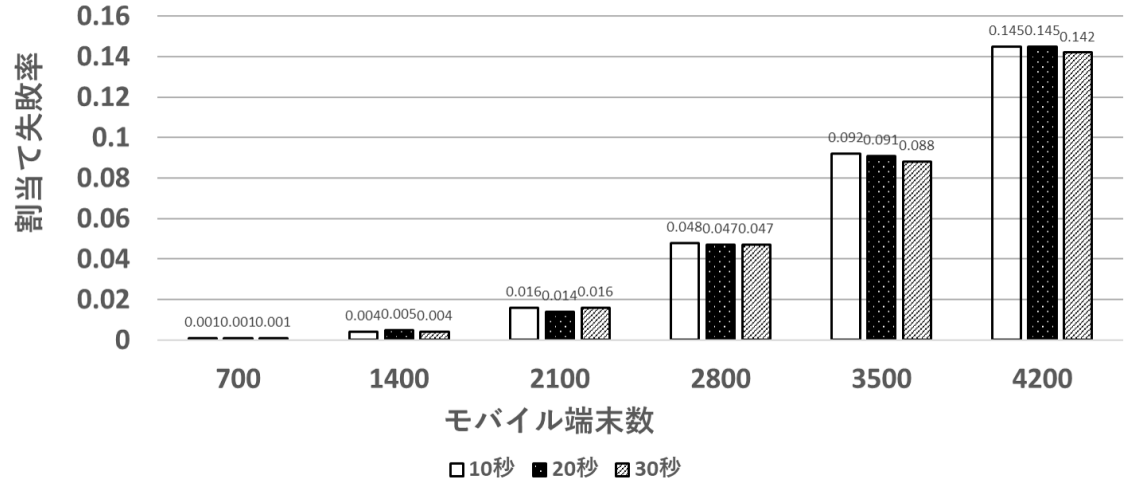

図 7.15　時間間隔ごとの走行履歴による割当ての割当て失敗率の変化

7.3.2　走行履歴による割当ての推定時間を変化させた場合の結果と考察

割当て時間間隔を 10 秒とし推定時間 5 秒から 15 秒まで変化させシミュレーションした結果と考察を以下に示す．負荷をかけるためモバイル端末数は 4200 に設定する．ペナルティ関数 f の値は 0 か 500 を出力させるよう設定した．

(1) 走行履歴による割当ての推定時間の変化させた場合の結果

図 7.16 に走行履歴による割当ての推定時間を 5 秒から 15 秒に変化させた平均ホップ数

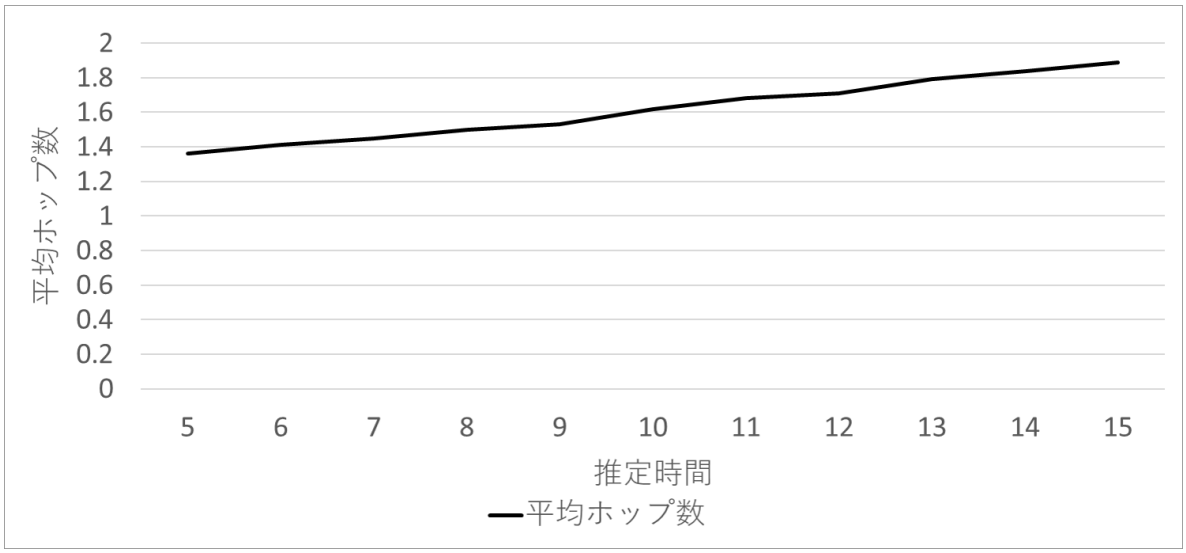

図 7.16　推定時間の変化させた走行履歴による割当ての平均ホップ数の変化

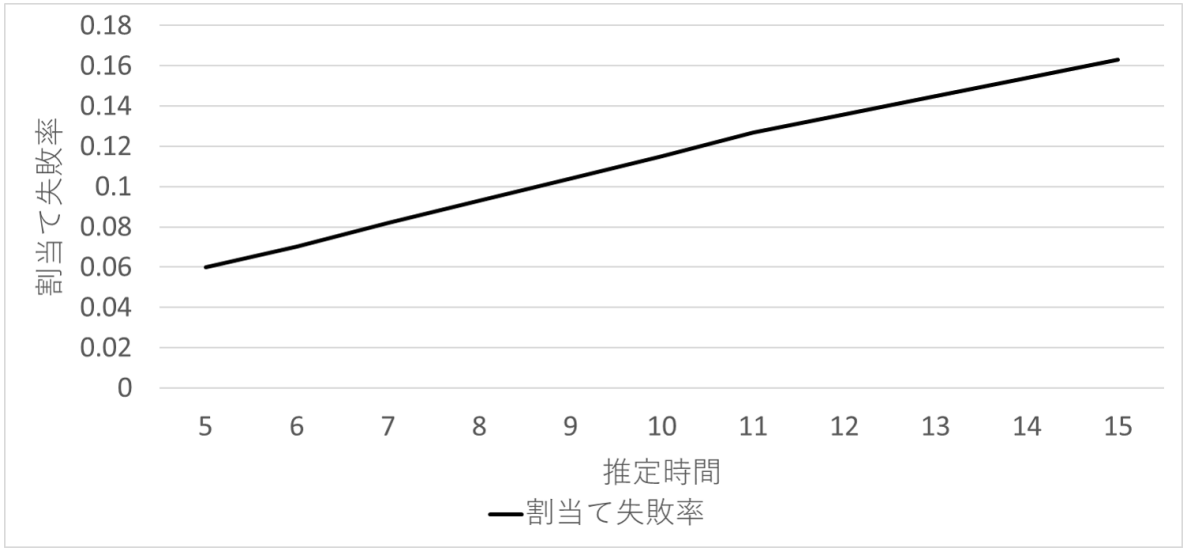

図 7.17　推定時間の変化させた走行履歴による割当ての割当て失敗率の変化

の結果，図 7.17 に割当て失敗率を示す．図 7.16，図 7.17 より，平均ホップ数，割当て失敗率ともに推定時間の増加に比例して結果が悪くなる．

(2) 走行履歴による割当ての推定時間の変化させた場合の考察

図 7.16，図 7.17 のように走行履歴による割当ての推定時間の増加につれて平均ホップ数と割当て失敗率が高くなるのは，推定時間が大きくなるにつれて推定移動場所とモバイル端末が移動した場所との差が大きくなるためである．モバイル端末の移動と異なる

OS	Microsoft Windows 10 Pro
プロセッサ	Intel(R) Core(TM) i5-6500
メモリ (RAM)	8.00 GB
使用言語	python3.6

表 7.3　実験環境

モバイル端末数	88	168
最近傍割当て	0.55134[sec]	1.63347[sec]
最近傍・最小リソース量による割当て	0.01561[sec]	0.02393[sec]
走行履歴による割当て	0.06881[sec]	0.13653[sec]

表 7.4　各手法の実行時間

方向の周辺 MEC サーバに割り当てると 3 ホップになる時間が長くなり，モバイル端末の移動データにもよるが，モバイル端末の推定移動場所が被ってしまい割当て失敗が発生してしまう．

7.3.3　各手法の実行時間

最近傍割当て，最近傍・最小リソース量による割当て，走行履歴による割当ての実行時間を計測する．シミュレーター上で 1 つの MEC サーバがモバイル端末数が 88，168 の時に割当て処理が開始し，割当て先を決定するまでに必要な処理時間を測定する．実験環境を表 7.3 に示す．実行時間を計測した結果を表 7.4 に示す．

結果より，走行履歴による割当てはモバイル端末数が 88 の時に約 0.06 秒，168 の時に約 0.13 秒となった．通信範囲半径 500m の基地局に対して，168 台のモバイル端末がリソース要求するのは非常に混雑している状態であると言える．混雑している状態において処理時間が約 0.13 秒であれば，問題ないと言える．

第 8 章

おわりに

8.1 終わりに

近年，モバイル端末の増加に伴い，従来のモバイルクラウドコンピューティングではなく，モバイルエッジコンピューティングを用いた構成が研究されていることを述べ，モバイルエッジコンピューティングの問題である計算機資源の制約を解決する過去研究を紹介した．

しかし，過去研究の課題として，提案した手法が集中アルゴリズムだったため，年々増加するモバイル端末全てを一括管理していた．モバイル端末全てを扱うためには大量の計算機資源，ネットワーク容量が必要になるため，自立分散的に処理する必要がある．そこで，本研究では，分散アルゴリズムの提案を行い，シミュレーション実験を行った．今回のシミュレーション実験により，走行履歴を用いてモバイル端末の移動場所を推定し，推定したモバイル端末の移動場所に近い MEC サーバに割当てすることで最近傍割当て，最近傍・最小リソース割当てと比較して大きく平均ホップ数を抑制することができ，割当て失敗率に関しても，モバイル端末数が増加すると最近傍・最小リソース割当てよりも優位な手法であることを示した．ただし，推定時間が大きくなると，本来のモバイル端末の移動場所と離れた場所の MEC サーバに対して，割り当ててしまう．また，各手法の実行時間を計測し，提案手法の処理が高速であることを示した．

8.2　今後の課題

実際の環境では，MEC サーバの所有リソース量は MEC サーバの CPU やメモリによって異なるため，所有リソースの定義を厳密にする必要がある．

また，モバイル端末の利用するアプリケーションは多種類であり，MEC サーバによっては実行環境が必要である．本研究では，1 種類のアプリケーションを全てのモバイル端末が要求し，実行環境がもともと用意されているものとしているため改良が必要になる．

謝辞

本研究を進めるにあたり，懇切丁寧に御指導頂きました指導教員であり，主査の横山和俊教授，お忙しい中，本研究の副査を務めて頂きました松崎公紀教授，高田喜朗准教授に心より感謝致します．また，共同研究としてシミュレーション環境を共同で作成した杉村侑起君，修士 2 年の黒木勇作さんを始めとする研究室の皆様に支えられ研究をすることができ，心より感謝致します．

参考文献

[1] シスコシステムズ合同会社, ”CiscoVisualNetworkingIndex:予測と方法論 2016-2021 年ホワイトペーパー”, ⟨https://www.cisco.com/c/ja_jp/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360⟩(参照 2020/1/23).

[2] 田中裕之, 高橋紀之, 川村龍太郎, ”IoT 時代を拓くエッジコンピューティングの研究開発”, NTT 未来ねっと研究所, NTT ジャーナル (2015).

[3] 塩田純, 滝澤充, 田中裕之, 高橋紀之, 小林英嗣, ”画像処理におけるエッジコンピューティングを用いた垂直分散処理方法の検討”, 情報処理学会研究報告, Vol.2016-DPS-168.No.2(2016).

[4] 大崎康平, 福永昂輝, 横山和俊, ”Cloudlet 環境における移動経路計画を用いたリソース割当て手法の検討 ”, 情報処理学会研究報告, Vol.2017-DPS-173, No.8, pp.1-8(2018).

[5] 柚木克夫, 新保宏之, ”5G におけるエッジコンピューティングのリソース活用に向けた制御手法”, KDDI 総合研究所, ⟨https://5g-miedge.eu/wp-content/uploads/2018/05/IEICE_KDDI-Research_Control-method-for-resource-utilization-of-edge-computing-in-5G.pdf⟩(参照 2020/1/23).

[6] 独立行政法人情報処理推進機構, ”5NIST によるクラウドコンピューティングの定義”, ⟨https://www.ipa.go.jp/files/000025366.pdf⟩(参照 2020/1/23).

[7] 林雅之, ”イラスト図解式この一冊で全部わかるクラウドの基本第 2 版”, pp.12-16. SBCreative.

[8] Y.C.Hu,M.Patel,D.Sabella,N.Sprecher,V.Young, ”Mobile edge computing—A key technology towards 5G”, Sophia Antipolis, France:ETSI, No.11(2015).

[9] Nasir Abbas,Yan Zhang,Amir Taherkordi,Tor Skeie, ”Mobile Edge Computing: A Survey”, IEEE INTERNET OF THINGS JOURNAL, Vol.5, NO.1, pp450-

465(2018).

[10] 田添聡士, 福永アレックス, ”仮想マシン配置問題に対する厳密アルゴリズム”, 情報処理学会研究報告, 2009-MPS-76, No.11, pp1-7(2009).

[11] 網代育大, ”仮想マシンのためのパッキングアルゴリズム”, 研究報告システム評価(EVA), Vol.2010-EVA-33, No.4(2010).

[12] 藪崎仁史, 中越洋, 村山耕一, 加藤崇利, ”クラウドにおける自律的な応答性向上に向けた広域スケーリング”, 情報処理学会論文誌, Vol.57, No2, pp562-572(2016).

[13] 鈴木俊裕, 張勇兵, 計宇生, ”移動端末からクラウドサービスを利用するためのプロキシサーバ構築に関する研究”, 情報処理学会研究報告, Vol.2013-MBL-69, No.15(2013).

[14] Rong Yu,Yan Zhang,Stein Gjessing,Wenlong Xia,and Kun Yang, ”Toward cloud-based vehicular networks with efficient resource management”, IEEE Network, Vol.27, pp.48–55(2013).

[15] Bo Li,Yijian Peri,Hao Wu,Bin Shen, ”Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds”, J Supercomput71, pp.3009-3036(2015).

[16] T.Vincenty, ”Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations”, No.176, Survey Review XXIII, pp.88-93

[17] Theworld'slargestOpenDatabaseofCellTowers, ⟨https://opencellid.org/#zoom=16&lat=37.77889 122.41942⟩(参照 2020/1/23).

[18] OpenStreetMap Japan, ⟨https://openstreetmap.jp/#zoom=5&lat=38.06539&lon=139.04297&la 照 2020/1/23).

[19] OSM 利用入門, ⟨https://openstreetmap.jp/node/762⟩(参照 2020/1/23).

[20] Simulation of Urban MObility, ⟨https://sumo.dlr.de/docs/index.html⟩(参照 2020/1/23).

[21] Shiqiang Wang,Rahul Urgaonkar,Ting He,Murtaza Zafer,Kevin Chan,Kin K. Leung, ”Mobility-Induced Service Migration in Mobile Micro-clouds”, 2014 IEEE

Military Communications Conference, pp.835-840(2014).

[22] 中野弘一, ”エッジコンピューティング環境におけるユーザの移動性を考慮したコンテナの同期先の選択手法の提案”, 〈”https://library.naist.jp/mylimedio/dllimedio/showpdf2.cgi/DLPDFR015532_P1-58”〉(参照 2020/1/23).