車両ネットワークにおけるタスク成功率向上のための タスクオフロード手法の提案

豊田 睦[1] 佐竹 颯太[1] 重野 寛[1]

概要： 現在，Internet of Vehicles (IoV) の発展に伴い車両アプリケーションの時間制約が高まっている．車両ネットワークの既存のオフロード手法では，タスクの処理時間を予測し，実行にかかる遅延が最小となる計算資源へオフロードを実行する．しかし，既存手法では道路上の他の車両やアプリケーションの時間制約に対する検討が十分でない．そこで，道路上の車両全体でタスク成功率を向上するための集中制御による動的なタスクオフロード手法を提案する．タスク成功率を最大化する目的関数を複数ナップザック問題として定式化し，近似アルゴリズムによってタスク成功率の向上を目指す．車両とエッジサーバの情報を集中制御に集約し，近似アルゴリズムに基づいてタスク成功率を最大化するエッジサーバを決定する．近似アルゴリズムでは，車両のタスク成功状況に応じて車両優先度を算出し，車両優先度に基づいてエッジサーバを決定する．本研究では提案したタスクオフロード手法のプロトタイプを実装して実験を行い，動作確認と評価を行なった．実験により，実装したオフロード手法によって効率的なエッジサーバへのオフロードを実行し，既存手法と比較してタスク成功率が最大 38%向上したこと確認した．

Proposal of Task Offloading for Improving Task Success Rate in Vehicular Networks

MUTSUMI TOYODA[1] HAYATA SATAKE[1] HIROSHI SHIGENO[1]

1. はじめに

現在，Internet of Vehicles (IoV) の発展に伴い，衝突回避や歩行者検知などの車両アプリケーションの時間制約が高まっている．また，センサー情報や画像情報を処理する車両アプリケーションの計算量は，車載コンピュータの処理能力に対し膨大である．車両の計算負荷を軽減するために，タスクオフローディングが注目されている．車両はアプリケーション実行に必要なタスクを，計算資源が豊富なサーバへ転送し，実行をすることにより，車載コンピュータの負荷軽減を実現する．また，ユーザ近傍のエッジサーバにオフロードし，アプリケーション実行にかかる時間を削減するモバイルエッジコンピューティング技術 [1] も研究されている．

車両ネットワークの既存のオフロード手法では，タスクの処理時間を予測し，実行にかかる遅延が最小となる計算資源へオフロードを実行する．しかし，既存のオフロード手法では，道路上の他の車両の遅延状況は考慮されておらず，エリア全体にとって最適なタスクオフロード戦略とは言えない．また，要求される車両アプリケーションの時間制約に対する検討も十分ではない．そのため，ミリ秒単位の許容遅延を考慮し，エリア全体の車両に対して最適なタスクオフロード手法が必要であると考える．

本稿では，道路上の車両全体でタスク成功率を向上するための集中制御による動的なタスクオフロード手法を提案する．タスク成功とは，タスクが許容遅延内に実行を完了することと定義する．提案手法では車両ネットワークに集中制御装置を導入し，制約を満たすエッジサーバにタスクをオフロードする最適化問題を近似アルゴリズムで解く．集中制御によって道路上を走行する車両やエッジサーバの情報を集約し，エリア全体で最適なオフローディングを達成する．近似アルゴリズムでは車両アプリケーションの時間制約とエッジサーバの計算資源制約を考慮することで，

[1] 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University,Yokohama, Kanagawa, 223-8522, Japan

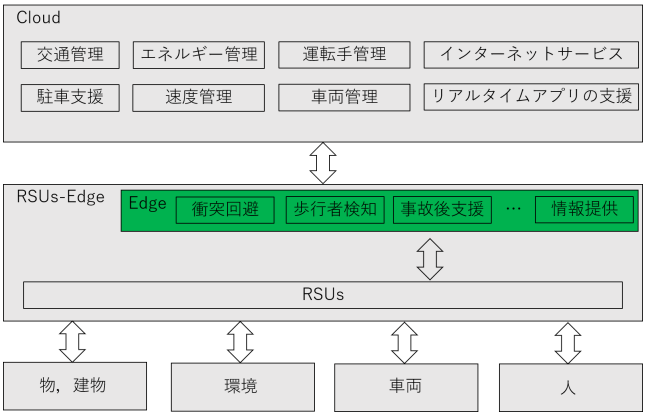

図 1 各層でサポートされるアプリケーションの分類

エリア全体のタスクがミリ秒単位の許容遅延内に実行を達成する．以上から，車両のタスク成功状況やエッジサーバの負荷状態に応じた動的なタスクオフロード手法を実現する．

2. 関連研究

本章では，車両ネットワークにおけるモバイルエッジコンピューティング，車両アプリケーションの時間制約について述べ，これらから考えられる既存の車両ネットワークにおけるタスクオフロード手法の問題点を抽出する．

2.1 車両ネットワークにおけるモバイルエッジコンピューティング

IoV の発展に伴い，車両は他の車両との車車間通信 (V2V) や路側機 (RSU) との路車間通信 (V2I) が実現し，事故回避アシスタントや道路経路情報などの車両アプリケーションの需要が高まっている．車両アプリケーションでは，車両のセンサや車載カメラによってリアルタイムに生成されるデータを処理するため，計算能力，ストレージ容量の要求も益々高まっている．しかし，車載コンピュータの計算資源，ストレージ容量はこの要求を満たしていない．そこで，車両の計算負荷軽減を目的に，車両ネットワークへクラウドコンピューティング [2] やエッジコンピューティングの導入 [3] が提案されている．

図 1 では車両アプリケーションの展望を示す [4]．図 1 のように分散化されたエッジサーバと中央のクラウドサーバの共存は互いに異なる種類の車両アプリケーションを補い合っている．クラウドで処理するアプリケーションは主にサービスマネジメントに使われ，トラフィック情報などのグローバルな視点が必要である．エッジで処理するアプリケーションはリアルタイムな情報を処理する．そのため，エッジサーバは車両と近い場所に配置する必要がある．

2.2 車両アプリケーションの時間制約

表 1 にはリアルタイム性が要求されるため，エッジで処理すべきアプリケーションを示す [4]．これらの車両アプリケーションの実行には QoS 要求として許容遅延があるため，本稿では，100msec 以内にアプリケーションの実行を完了することを一つの基準とする．

表 1 車両アプリケーションの QoS 要求
Table 1 QoS requirements for vehicle applications.

アプリケーション	許容遅延	アプリケーション例
Advanced driving	1ms-20ms	自動運転，遠隔運転
Efficient driving	<100ms	自動駐車，リアルタイムナビゲーション
Infotainment	<100ms	AR ゲーム，地域広告

2.3 車両ネットワークにおけるタスクオフロード手法

X.Wang ら [3] は，メッセージの平均応答時間を最小化するオフローディング手法を提案している．路車間通信や車車間通信を使用し，エッジサーバや駐車中の車両，走行中の車両に対し，オフローディングにかかる遅延を予測し，最小な計算資源へオフロードする．

L.Li ら [5] は，車両アプリケーションの時間制約を満たし，金銭コストを最小とするタスクオフロード手法を提案している．タスクサイズや路車間，車車間通信の帯域幅，エッジサーバの計算資源からタスクの処理時間を予測し，金銭コストが最小となるエッジサーバへオフロードする．

H.Wang ら [6] は，道路上の計算資源を統合し，タスクを実行するフェデレーテッドオフロード手法を提案している．タスクを 3 つのサブタスクに分割し，それぞれのサブタスクを他の車両やエッジサーバに配置し，遅延を削減する．

しかし，既存のオフロード手法では，道路上の他の車両の遅延状況は考慮されておらず，エリア全体にとって最適なタスクオフロード戦略とは言えない．また，2.2 章で述べた時間制約に対する検討も十分ではない．そのため，ミリ秒単位の許容遅延を考慮し，エリア全体の車両に対して最適なタスクオフロード手法が必要であると考えられる．

3. 提案手法

本章では，道路上の車両全体でタスク成功率を向上するための集中制御による動的なタスクオフロード手法を提案する．

3.1 タスクオフロード手法の概要

タスクオフロード手法では，従来の車両ネットワーク環境に集中制御装置を導入する．集中制御装置では各車両情報，各エッジサーバ情報を集約し，エリア全体で最適なオフロード先エッジサーバを算出する．各車両は結果を受け取り，算出されたエッジサーバにオフロードを行う．また，本稿ではタスク成功率の最大化を複数ナップザック問題として定式化する．しかし，複数ナップザック問題は NP 困難である [7]．そこで，集約された各車両情報，各エッジサーバ情報をもとに，車両のタスク成功率を最大化するオ

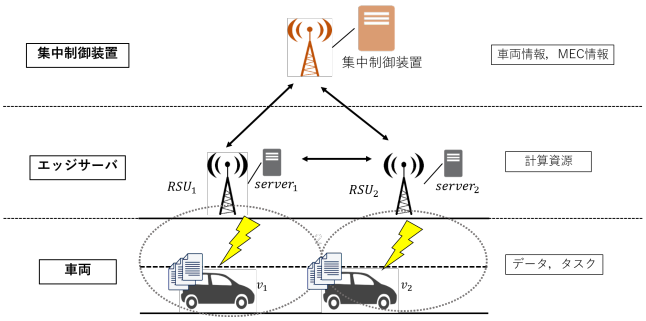

図 2　想定環境

フロード先エッジサーバを近似アルゴリズムによって算出する．近似アルゴリズムでは，エッジサーバの負荷状況と各車両のタスク優先度によって目的のオフロード先エッジサーバを決定する．エッジサーバの負荷状況は集中制御装置がエッジサーバにリクエストを送り，CPU 使用率を格納したメッセージを返答することで得られる．各車両のタスク優先度は，各車両のタスク成功状況によって算出する．

3.2　想定環境

図 2 は提案オフロード手法における想定環境を示しており，従来の車両ネットワークに集中制御装置を導入し，車両，エッジサーバ，集中制御装置の 3 層構造からなる．各エッジサーバは通信能力を持った路側器 (RSU) に備わっており，各 RSU および集中制御装置はバックホールネットワークを介して通信が行われる．また，各 RSU は通信範囲ごとに配置されており，道路上の車両は少なくとも 1 台の RSU と路車間通信をする．

3.3　タスク成功率

道路上の各車両が，エリア全体においてタスク成功率が最大となるエッジサーバを集中制御装置によって決定する．タスク成功率とは，オフロードしたタスクの内，許容遅延内に実行が完了したタスクの割合を指す．道路上の車両全体のタスク成功率を $F_{success}$ とし，複数ナップザック問題に基づいて目的関数 (1) を定式化する．

$$maximize \quad F_{success} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} \cdot success_{ij}}{N} \tag{1}$$

制約は以下の式で表される．

$$\forall j \quad \sum_{i=1}^{N} x_{ij} \cdot c_i \leq C_j \tag{2}$$

$$\forall i \quad \sum_{j=1}^{M} x_{ij} \leq 1 \tag{3}$$

$$x_{ij} \in \{0, 1\} \tag{4}$$

$$success_{ij} = \begin{cases} 1 & T_{ij} < T_{max} \\ 0 & T_{ij} \geq T_{max} \end{cases} \tag{5}$$

目的関数 (1) において，$V = \{v_1, v_2 ... v_i ... v_N\}$ は道路上の車両，$E = \{e_1, e_2 ... e_j ... e_M\}$ は道路上のエッジサーバ，x_{ij} は車両 v_i がエッジサーバ e_j にオフロードをするかを表す決定変数である．$x_{ij} = 1$ の場合，v_i が e_j にオフロードし，$x_{ij} = 0$ の場合，オフロードをしない．また，$success_{ij}$ は v_i が e_j において，1 ステップ前のタスクがタスク成功したかを表している．$success_{ij}$ は v_i が e_j において，1 ステップ前のタスクの実行が許容遅延内であれば $success_{ij} = 1$，許容遅延を超えた場合，$success_{ij} = 0$ の値をとる．

制約 (2) において，c_i は v_i のタスク処理に必要な計算容量，C_j は e_j の実行可能な計算容量を表す．つまり，制約 (2) は各エッジサーバにオフロードされるタスクの総計算量がエッジサーバの計算容量を超えないことを表す．制約 (3) は車両が同時に複数のエッジサーバにオフロードできないことを示す．制約 (4) はタスクを分割したり，タスクの一部のみをオフロードしないことを示す．制約 (5) において T_{ij} は v_i が e_j においてオフロードした際の遅延，T_{max} はタスクの許容遅延時間を示す．

目的関数 (1) は車両数に対し複数ナップザック問題であり，NP 困難である．つまり，複数ナップザック問題を複数の部分問題に分割し，それぞれを独立に解く必要がある．

3.4　車両優先度

車両のタスク成功状況によって変動する車両優先度について定式化する．Osorio ら [7] は，大規模な複数ナップザック問題が近似アルゴリズムによって比較的短時間で解けることを示している．本提案では複数ナップザック問題を貪欲法を用いて近似的に解を得る．貪欲法では時間計算量を $O(n \log n)$ で最適解の 50%以上の解を得られる [8]．

v_i の車両優先度 p_i の算出は車両のタスク成功状況に依存する式 (6) に基づいて行う．

$$p_i = \begin{cases} \frac{1}{M - \sum_{j=1}^{M} fail_{ij}} & if \sum_{j=1}^{M} fail_{ij} = M \\ 0 \end{cases} \tag{6}$$

$$fail_{ij} = \begin{cases} 1 & T_{ij} \geq T_{max} \\ 0 & T_{ij} < T_{max} \end{cases} \tag{7}$$

ここで，$fail_{ij}$ は v_i が e_j においてタスク許容遅延内でタスク処理をしたかを表す．式 (6) は v_i の車両優先度 p_i が

タスク失敗するごとに増加することを表す．つまり，車両優先度 p_i の初期値は $1/M$ であり，v_i が全てのエッジサーバでタスク失敗した場合，p_i は 0 となる．

4. 近似アルゴリズム

3.3 で定式化したタスク成功率の最大化問題を解くために，3.4 で述べた車両優先度と貪欲法を用いて，道路上の車両全体のタスク成功率が最大となるエッジサーバを決定する近似アルゴリズムについて述べる．アルゴリズム 1 は本稿で目的関数 (1) の近似解を求めるアルゴリズムを示す．

Algorithm 1 提案オフロード手法

Require: Vehicle identifier i,Task success status of vehicle $fail_{ij}$,Task size of vehicle c_i, Computational capacity of Edge server C_j
Ensure: Determined edge server
  Calculate vehicle priority for all vehicles
  Sorted i by vehicle priority
  for $i = 0...N$ **do**
    while $j < M$ **do**
      if $c_i < C_j$ **then**
        if $fail_{ij}$ *is* 0 **then**
          Determine edge server, break
        end if
      end if
    end while
  end for

アルゴリズム 1 において，集中制御装置は車両のタスク成功状況とエッジサーバの負荷状態を得る．次に式 (6) に基づき車両優先度を算出する．車両優先度 p_i を降順にソートし，車両優先度の高い順に，$success_{ij} = 1$ であれば，オフロード先エッジサーバを決定する．計算資源が不足している場合は他のエッジサーバにオフロードを行う．また，$success_{ij} = 0$ の場合，最寄りエッジサーバに隣接したエッジサーバにオフロードを行う．このアルゴリズムにより，道路上の車両はタスク成功状況に応じて動的にオフロードするエッジサーバを決定することが可能である．

4.1 システム構成とタスクオフロード手順

車両の最寄りエッジサーバと，集中制御装置によって算出されたオフロード先エッジサーバが異なる場合を例に，提案機構のオフロード手順について述べる．図 3 は最寄りのエッジサーバと集中制御装置によって算出されたオフロード先エッジサーバが異なる場合のタスクオフロード手順例を示している．

(1) 車両 v_i は，車両識別子，$success_{ij}$ が格納されたリクエストを最寄りエッジサーバへ転送する．

(2) エッジサーバで車両からのリクエストを受信し，リクエストを集中制御装置へ転送する．

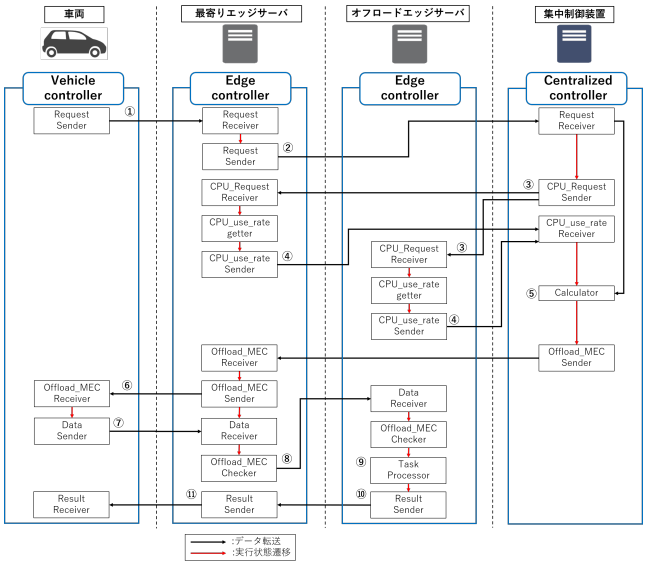

図 3 提案機構におけるタスクオフロード手順例

(3) 集中制御装置は，リクエストに格納された v_i，$success_{ij}$ を取り出し，エッジサーバに CPU 使用率を要求する．

(4) 各エッジサーバは要求を受信し，CPU 使用率を集中制御装置に転送する．

(5) 集中制御装置は CPU 使用率と v_i，$success_{ij}$ から近似アルゴリズムによってオフロード先エッジサーバを決定する．

(6) 集中制御装置によって決定したオフロード先エッジサーバを v_i に転送する．

(7) 車両は最寄りエッジサーバに処理データとオフロード先エッジサーバを格納したメッセージを送信する．

(8) 最寄りエッジサーバではオフロード先エッジサーバを参照し，バックホール通信を介してオフロード先エッジサーバに処理データを転送する．

(9) オフロード先エッジサーバでは，タスク処理を実行する．

(10) オフロード先エッジサーバは，タスク処理した実行結果をバックホール通信を介して最寄りエッジサーバに転送する．

(11) 最寄りエッジサーバは実行結果を車両に転送する．

5. 実験・評価

本章ではプロトタイプを用いた実験による動作確認と評価について述べる．

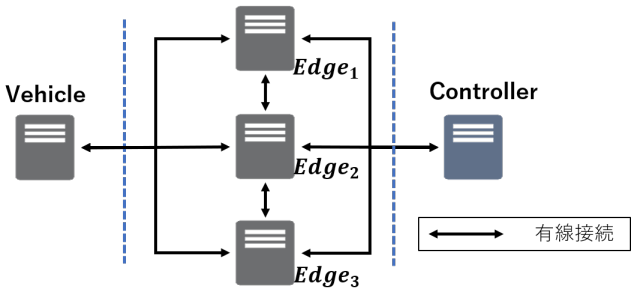

図 4　実装環境

表 2　使用機材
Table 2　Material used.

	CPU	Memory
Vehicle	Intel(R) Core(TM)i5-2400 CPU@3.1GHz	2GB
$Edge_1$	Intel(R) Core(TM)i7-2600 CPU@3.4GHz	4GB
$Edge_2$	AMD Ryzen3 2200G with Radeon Vega Graphics	8GB
$Edge_3$	Intel(R) Pentium(R)CPU G4560@3.5GHz	4GB
Controller	Intel(R) core(TM)i7-3770 CPU@3.4GHz	8GB

5.1　実験環境

図 4 に実装するタスクオフロード手法の実装環境を示す．実験では，5 台のコンピュータをそれぞれ Vehicle，$Edge_1$，$Edge_2$，$Edge_3$，Controller と想定し，図 4 のように有線接続した．表 2 に使用機材の性能を示す．また，使用した OS は Ubuntu 16.04.3 LTS である．

実装するタスクオフロード手法において，各コンポーネントの実装には Python [9] を用いた．使用した Python のバージョンは Python 3.5.2 である．4.1 章で述べたような各計算資源上で必要となるソフトウェアおよびコンポーネントを作成し，動作させた．データの転送は Socket による UDP コネクションを用いた．また，計算資源間のルーティングテーブルは事前に設定されているものとした．車両の移動性は，車両速度と RSU の通信範囲から，車両が RSU を通過する時間を算出し，通過時間で最寄りのエッジサーバを切り換えることで実装した．

また，実験において実行する車両アプリケーションは人物数検出アプリケーションであり，タスク実行する際にタスクのオフロードを行う．実験で使用する画像は 1 枚で，複数人の全身画像から構成されており，アプリケーションは転送された画像に何人存在するか検出する．人物の検出には OpenCV [10] を利用した．OpenCV には学習済の Haar-like 特徴 [11] を用いた分類器のデータがあり，比較的短い時間で人物数検出が実行可能である．実験で測定する遅延時間は，車両が集中制御装置にリクエストを送信してから，集中制御装置によって最適なオフロード先のエッジサーバが決定し，車両が決定されたエッジサーバでオフロードを実行完了するまでの時間とする．

実験では車両の流入が一定の場合と，一定ではない場合においてタスク成功率を評価した．図 5 に示すように，1 つの RSU の通信範囲内に走行する車両の集合を車両群と

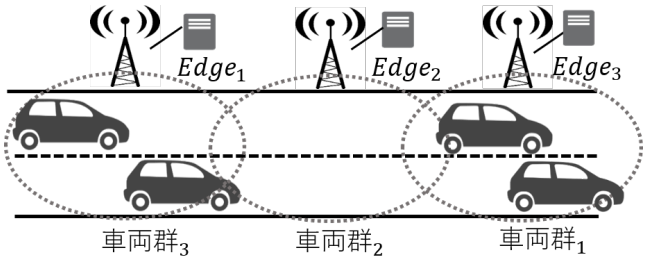

図 5　道路上を通過する車両群

表 3　実験パラメータ
Table 3　Experimental Parameters.

パラメータ	値
実験時間	180 [sec]
道路距離	3 [km]
エッジサーバ数	3
RSU の通信範囲	半径 500 [m]
車両速度	60 [km/h]
画像サイズ	66.3 [KB]
ファイル転送レート	5 [fps]
T_{max}	100 [msec]

した．この実験では各車両群の車両数を 20, 0, 20 として車両の流入を不均一にした．

また，以下の既存のオフロード手法とタスク成功率について比較する．

- ダイレクトモード：最寄りエッジサーバにのみオフロードを行う．
- ランダムモード：ランダムにエッジサーバを決定し，オフロードを行う．

5.2　実験パラメータ

実装したタスクオフロード手法の実験パラメータを表 3 に示す．車両は時速 60km/h で単一方向の 1 車線の直線道路を走行し，各 RSU の通信範囲は半径 500m であるため，車両は実験時間で 3 つの RSU 通信範囲を通過する．また，2.2 章で述べたように，車両アプリケーションの許容遅延 T_{max} を 100 msec に設定した．

5.3　評価結果

図 6 に道路上の車両の流入が一定の場合における，1km あたりの車両数とタスク成功率の関係について示す．車両の流入が一定の場合，提案手法がどの既存手法よりもタスク成功率が向上していることがわかる．提案手法では，1 つのエッジサーバに負荷が集中した場合に，他のエッジサーバへオフロードを実行するため，エッジサーバの負荷が分散する．既存手法では，エッジサーバの負荷は考慮していないため，タスク成功率は低下した．また，車両数とエッジサーバにかかる負荷は比例関係にあることから，車両数の増加はタスク成功率の低下を示した．

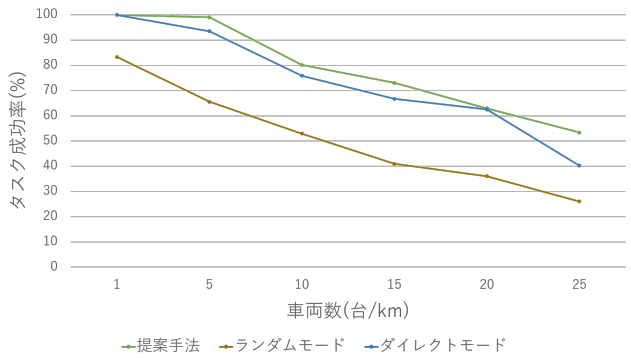

図 6 車両の流入が一定の場合におけるタスク成功率

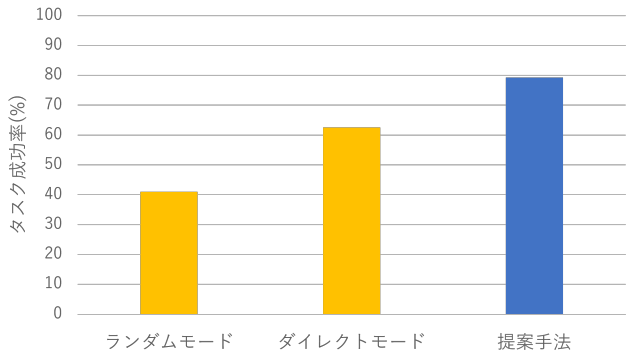

図 7 車両の流入が一定ではない場合のタスク成功率の比較

次に，車両の流入が一定ではない場合について述べる．図 7 に車両の流入が一定ではない場合における，既存オフロード手法と提案オフロード手法のタスク成功率の比較を示す．提案手法のタスク成功率は，ランダムモードと比較し 38.3%，ダイレクトモードと比較し 16.8%向上した．提案手法では，集中制御によって効率的にオフロードを行うため，エッジサーバの計算資源を効率的に利用する．また，提案手法では，他の車両のタスク成功状況を考慮し，タスク成功率が低い車両は優先的にエッジサーバの計算資源が割り当てられる．そのため，車両の流入が一定の場合と同様に，提案手法のタスク成功率が向上した．既存手法では，エッジサーバへのオフロードが車両の分布によって偏るため，エッジサーバの計算資源を有効に利用できない．

以上より，車両の流入が一定である場合と，一定ではない場合どちらにおいても提案オフロード手法のタスク成功率が向上することを確認した．

6. おわりに

本稿では，道路上の車両全体でタスク成功率を向上するための集中制御による動的なタスクオフロード手法の提案を行った．本稿で提案したオフロード手法では，車両ネットワークに集中制御装置を導入し，各車両の優先度やエッジサーバの負荷状態を考慮した近似アルゴリズムに基づいてオフロード先エッジサーバを算出する．車両の優先度は時間制約を満足したかのタスク成功状況によって算出される．これにより車両のタスク成功状況やエッジサーバの負荷状態に応じた動的なタスクオフロード手法を実現した．

また，本稿で提案したタスクオフロード手法を実装して実験を行い，動作確認と評価を行った．実験では 5 台のコンピュータを用いて複数層の環境を構成した．実験により，タスク成功率はダイレクトモードと比較し最大 38%，ランダムモードと比較して最大 35%の向上を確認した．

以上より，本稿で提案したタスクオフロード手法は道路上の車両全体でタスク成功率を向上するオフロード手法であることを示した．

謝辞 本研究は JSPS 科研費 JP20H04180 の助成を受けたものです．

参考文献

[1] Hayata Satake, Ryotaro Tani, Hiroshi Shigeno, A Task Placement System for Face Recognition Applications in Edge Computing. In *IEEE Consumer Communications & Networking Conference (CCNC 2020, Work-in-Progress Session)* , pp.441-446, 10-13 January 2019.

[2] Salma M. A. Ataallah, Salwa M. Nassar, and Elsayed E. Hemayed. Fault tolerance in cloud computing - survey. In *2015 11th International Computer Engineering Conference (ICENCO)*, Cairo, Egypt, pages 241-245, 2015.

[3] X.Wang，Z.Ning and L.Wang. Offloading in Internet of Vehicles: A Fog-Enabled Real-Time Traffic Management System. In *IEEE Transactions on Industrial Informatics*, Vol.14, pp4568-4578，March.2018.

[4] J.Li , X.Shen , L.Chen , D.Van , J.Ou , L.Wosinska , and J.Chen. Service Migration in Fog Computing Enabled Cellular Networks to Support Real Time Vehicular Communications. In *IEEE Access*, Vol. 7, pp .13714-13704, Feb

[5] L.Li, H.Zhou, S.Xiaoli Xiong, J.Yang and Y.Mao. Compound Model of Task Arrivals and Load-Aware Offloading for Vehicular Mobile Edge Computing Networks. In *IEEE Access*, Vol. 7, pp26631-26640, Feb. 2019.

[6] H.Wang，X.Li，H.Ji，Heli Zhang．Federated Offloading Scheme to Minimize Latency in MEC-enabled Vehicular Networks. In*IEEE Globecom Workshops*, 9-13 Dec. 2018.

[7] M.A Osorio, G. Cuaya. Hard problem generation for MKP. In *Proceedings of the Sixth Mexican International Conference on Computer Science (ENC' 05)*, Sept. 2005.

[8] Cormen, Leiserson, Rivest, and Stein. Introduction to Algorithms In *Greedy Algorithms*, 2001, Chapter 16.

[9] Python Software Foundation. "python". [Online]. Available: https://www.python.org/, [Online; accessed 26-Jan-2019].

[10] Intel Corporation. "opencv (open source computer vision library) ". [Online]. Available: https://opencv.org, [Online; accessed 26-Jan-2019].

[11] T. Mita, T. Kaneko, and O. Hori. Joint haar-like features for face detection. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Beijing, China, 2005.