

Google Distance Matrix APIを利用した 再配達を含めた配達ルート最適化

松下 義孝^{1,a)} 相川 勝^{3,b)} 井上 健太郎^{2,c)} 山森 一人^{2,d)}

概要：本論文では、再配達に対応した配達経路最適化法の提案、評価する。配達経路の決定には、遺伝的アルゴリズムと列挙法をアレンジした解法を用いる。遺伝的アルゴリズムと列挙法は、どちらも巡回セールスマン問題（TSP:Traveling Salesman Problem）の主な解法の1つである。本研究では、Google が提供する Distance Matrix API(Application Programming Interface) を用いて配達先間の実距離を取得し、経過探索に利用する。配達先や配達エリアなどのパラメータを変化させ、配達シミュレーションによって評価したところ、配達先が増えるにつれ配達の成功率が90%に近づくことがわかった。

キーワード：Google Distance Matrix API, 巡回セールスマン問題, 遺伝的アルゴリズム

Delivery Route Sub-optimization Combined by Google Distance Matrix API and Genetic Algorithm

YOSHITAKA MATSUSHITA^{1,a)} MASARU AIKAWA^{3,b)} KENTARO INOUE^{2,c)} KUNIHITO YAMAMORI^{2,d)}

Abstract: In this paper, we propose and evaluate a delivery route optimization method including random redelivery requests. The combination of Genetic Algorithm and enumeration method is used to determine the delivery route. Genetic algorithm and enumeration method are one of the popular solution methods to solve Traveling Salesman Problem. In this research, we will use the Distance Matrix API(Application Programming Interface) provided by Google to obtain the real distances between destinations. The proposed method is evaluated by delivery simulation with varying parameters such as delivery destinations and delivery area.

Keywords: Google Distance Matrix API, Traveling Salesman Problem, Genetic Algorithm

1. はじめに

近年、インターネットによる通信販売の普及やネット通販における送料無料などのサービス拡大により、宅配便の取扱個数が年々増加傾向にある。平成29年度の宅配便取

扱個数は約42億個で、10年前の平成19年度から約10億個も増加している。宅配会社は、企業規模を問わず人材不足で、ドライバーの時間外労働は月80時間前後に達している[1]。再配達も問題の1つであり、平成30年10月期の宅配便の再配達率は約15.2%に上る。再配達問題の対策として宅配ボックスの設置なども進められているが、大規模マンションでは宅配ボックスがすぐ荷物で埋まってしまう、宅配会社同士の競合といった問題も生じている。これらの現状から、宅配ドライバーの負担を減らす配達業務のサポートは非常に重要である。

宅配便の配達業務のサポートとして、より少ない走行距離で効率良く配達業務を行う配達ルート最適化サービスがある。クラウド上で利用できる最適ルート自動作成システ

¹ 宮崎大学 工学研究科
Graduate School of Engineering, University of Miyazaki, Japan
² 宮崎大学 工学教育研究部
Faculty of Engineering, University of Miyazaki, Japan
³ 宮崎大学 工学部 教育研究支援技術センター
Technical Center, Faculty of Engineering, University of Miyazaki, Japan
^{a)} matsushita@taurus.cs.miyazaki-u.ac.jp
^{b)} aikawa@cs.miyazaki-u.ac.jp
^{c)} inoue@cs.miyazaki-u.ac.jp
^{d)} yamamori@cs.miyazaki-u.ac.jp

ムである GuRutto(グルット) [2] や、顧客の実績データを参考に輸配送の最適化シミュレーションを実施し、その結果を基にして利用者に合わせた輸配送計画システムを構築する RootCreator(ルートクリエイター) [3] などが挙げられる。配達ルート最適化は現在も研究が進んでおり、膝杰ら [4] による宅配便ドライバの動的ルートスケジューリングや、Ohsugi ら [5] による電力消費量に注目した在宅予測に基づいた配達ルート最適化などがある。これらは主に再配達を考慮した配達ルート最適化の研究といえる。

配達ルート最適化は、配達拠点を出発し配達先をすべて訪問した後に配達拠点に戻る最短距離となるルートを求める問題である。この問題は巡回セールスマン問題(TSP:Traveling Salesman Problem)に相当する。TSP とは、都市の集合と各都市間の距離が与えられたとき、すべての都市を1度だけ巡回して出発点に戻る経路のうち、最短の経路(以下、最適解と呼ぶ)を求める問題である。TSP には必ず最適解を求められる厳密解法と、早い時間で最適解に近い準最適解を求められる近似解法の2つが存在する。

配送問題において、再配達依頼が発生した場合には再配達先を考慮しつつ再度配達ルートの最適化を行う必要が生じる。その場合、再配達依頼を受けた地点が出发点、配達拠点が終着点となるため、TSP にはならない。よって、再配達依頼発生時には、出発前に行う配達ルートの最適化とは異なる方法により配達ルートの最適化を行う必要がある。本研究では、厳密解法をアレンジした解法を用いる。

より正確な最適化を行うために、本研究では各配達先間の距離を、直線距離ではなく実際の道路上の距離である実距離とする。実距離の取得には、Google が提供する Web 地図サービスである Google Maps Platform にて利用できる Web API の1つである Distance Matrix API を用いる。

本研究では、Web API によって取得した実距離の利用と、厳密解法をアレンジして再配達に対応した配達経路最適化法の提案と評価を目的とする。不在率、再配達依頼率を変化させ、シミュレーションにより提案手法を評価する。

本論文の構成は以下の通りである。第2章では、巡回セールスマン問題について説明する。第3章では、Google Maps Platform について説明する。第4章では、vcopt について説明する。第5章では、提案手法の評価手法と結果について説明する。第6章では、研究のまとめと今後の課題である。

2. 巡回セールスマン問題

巡回セールスマン問題(TSP)とは、都市の集合と各都市間の距離が与えられたとき、全ての都市を1度だけ巡回して出発点に戻る最短の経路を求める問題である。最短の巡回経路を最適解と呼ぶ。TSP の例を図1に示す。TSP の解法には、大別すると厳密解法と近似解法の2つが存在する。厳密解法は必ず最適解を求めることができる解法であ

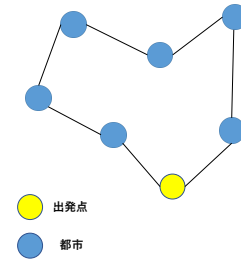


図1 巡回セールスマン問題の例。

るが、都市数が多くなると時間がかかりすぎるため、大規模な TSP を厳密解法で解くのは現実的ではない。近似解法は、最適解に比較的近い準最適解を求めることができる解法であり、厳密解法より短時間で解を求めることができるため、大規模な TSP を解く際に使われる。

2.1 厳密解法

厳密解法の主な例として、列挙法や分枝カット法が挙げられる。列挙法は巡回路を全て列挙することで最適解を求める解法である。分枝カット法は高速化のために分枝限定法と切除平面法を組み込み、無駄な探索を除いた解法である。分枝カット法による問題解決の例として、スウェーデンの 24,987 都市の問題がある [6]。この問題では 96 台からなる PC クラスタが使用され、約 8 年をかけた計算により最適解が求められた。この例から分かるように、厳密解法は最適解を必ず求められる代わりに都市数が多くなると現実的な時間では計算できない。

2.2 近似解法

近似解法とは厳密解法より比較的短い時間で最適解に近い準最適解を求めることのできる解法である。例として、経路を繋ぎ変えることで準最適解を求める 2-opt 法や LK 法、生物の進化の仕組みを模倣した遺伝的アルゴリズム [9] などが挙げられる。

3. Google Maps Platform

Google Maps Platform とは、Google が提供する Web 地図サービスである [7]。本研究では Google Maps Platform 上で利用できる Web API の1つである Distance Matrix API を利用する。Distance Matrix API は、訪れる地点が複数存在する際に、すべての組み合わせの2点間の距離と移動に要する時間を行列形式で取得することができる。Distance Matrix API で取得する距離は、Google Maps Platform で推奨される経路の実距離である。Distance Matrix API では、HTTP リクエストを送ることにより JSON 形式で応答が返される。また、Google Maps Platform が提供する Web API を利用するには API キーが必要となる。

3.1 Web API

Application Programming Interface (API) とは、ある機能に特化したプログラムと、利用者がそのプログラムを利用できるようにするためにサービス提供者が提供するインターフェースである。API を用いると、プログラムを初めからすべて作成する必要が無くなるため、ソフトウェアの作成を単純化できる。Web API は、HTTP プロトコルによって利用できるネットワーク経由のリモート API である。Web API は HTTP / HTTPS ベースの API であるため、異なるプログラミング言語で開発されたアプリケーション同士を連携させることが可能である。また、Web ブラウザでも利用できるなど、汎用的に利用することが可能である。図 2 に Web API を利用したプログラムの例を示す。

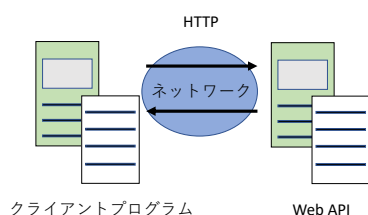


図 2 Web API を利用したプログラムの例。

3.1.1 HTTP

HTTP (Hyper Text Transfer Protocol) とは、インターネット上で Web サーバーとクライアントが相互に通信するために使用されるプロトコルである。HTTP では、クライアントが Web サーバーに HTTP リクエストを送り、Web サーバーは受信したリクエストを処理した後で HTTP レスポンスを返すことにより情報のやり取りを可能にする。図 3 に HTTP 通信の流れを示す。HTTP リクエストは、HTTP メソッドや URL などの情報で構成される。代表的な HTTP メソッドの一覧を表 1 に示す。

HTTP レスポンスは、ステータスコードや、HTTP リクエストへの応答結果となるデータが格納されているボディなどの情報で構成される。ステータスコードの一覧を表 2 に示す。

ボディのコンテンツタイプは HTML や XML、JSON などである。本研究では、JSON を使用する。

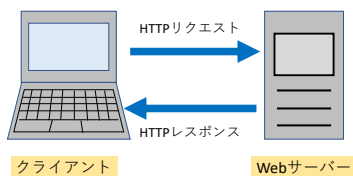


図 3 HTTP 通信の流れ。

表 1 HTTP メソッド。

GET	リソースの取得
POST	子リソースの作成、そのほかの処理
PUT	リソースの更新、リソースの作成
DELETE	リソースの削除
HEAD	リソースのヘッダ (メタデータ) の取得
OPTIONS	リソースがサポートしているメソッドの取得
CONNECT	プロキシ動作のトンネル接続への変更

表 2 よく使われるステータスコード。

200 OK	リクエストの成功
201 Created	リソースの作成成功
301 Moved Permanently	リソースの恒久的な移動
303 See Other	別 URI の参照
400 Bad Request	リクエストの間違い
401 Unauthorized	アクセス権の不正
404 Not Found	リソースの不正
500 Internal Server Error	サーバ内部エラー
503 Service Unavailable	サービスの停止

3.1.2 JSON

JSON (JavaScript Object Notation) は、JavaScript の記法でデータを記述できるデータ記述言語である。記法は JavaScript であるが、そのシンプルさから多くの言語がライブラリを用意しており、プログラム言語間でデータを受け渡すことに使用できる。Web サービスでは、ブラウザが JavaScript を実行できるので相性が良いこと、XML と比べてデータ表現の冗長性が低いことなどの利点から Ajax 通信におけるデータフォーマットとして活用されている。

3.2 Distance Matrix API

Distance Matrix API では、訪れる地点が複数存在する際に、すべての組み合わせの 2 点間の距離と時間を行列形式で取得することができる。Distance Matrix API は、必要なすべてのデータを一度に取得することが可能であるため、一度に多点間の距離と時間を取得したい場合に有力な API である。以下で、Distance Matrix API の詳細な利用方法と設定について説明する。

3.2.1 利用方法

Distance Matrix API は、HTTP プロトコルによりリクエストを送ることで情報を取得することができる。例として、outputFormat では出力結果を何のデータフォーマットで記述するかを指定する。利用できるデータフォーマットは XML と JSON の 2 種類であるが、Google Maps Platform では JSON の利用が推奨されている。parameters では、リクエストパラメータを指定する。次節でリクエストパラメータについて説明する。

3.2.2 リクエストパラメータ

リクエストパラメータは、Distance Matrix API にて HTTP リクエストを送る際に必要なパラメータ群である。リクエ

ストパラメータは、記述が必須であるパラメータと、記述が選択可能であるオプションのパラメータに分けられる。

3.2.2.1 必須のパラメータ

origins : 移動距離と移動時間を計算するための、出発点となる1つ以上の地点である。パイプ文字で区切られた1つ以上の場所を、場所 ID、住所、または緯度・経度の座標形式で指定する。

destinations : 移動距離と移動時間を計算するための、終点となる1つ以上の地点である。destinations パラメータの指定方法は origins パラメータと同じである。

key : Google Maps Platform の API キーを指定する。API キーにより特定の環境に限って API を利用できるように制限することができる。

3.2.2.2 オプションのパラメータ

mode : 移動距離を計算する際に使用する移動モードを指定する。デフォルトでは driving モードである。

language : 結果を返す言語を指定する。指定されていない場合、Distance Matrix API は HTTP ヘッダーの Accept-Language で指定されている優先言語、またはリクエストの送信元のドメインの母国語を使用する。

region : ccTLD (国別コードトップレベルドメイン) の2文字の値として指定されたリージョンコードを指定する。

avoid : 回避するルートを指定することでルートに制限を導入する。指定できる値は1つだけである。

units : 距離をテキストで表す際に使用する単位系を指定する。指定できる値は、キロメートルとメートルで表す metric と、マイルとフィートで表す imperial の2つである。

arrival_time : 公共交通機関による移動の際の到着時刻を指定する。1970年1月1日 UTC の午前0時から時間を秒単位の整数で指定できる。

departure_time : 公共交通機関による移動の際の出発時刻を指定する。1970年1月1日 UTC の午前0時から時間を秒単位の整数で指定できる。

traffic_model : 移動時間を計算する際に用いる仮定を指定する。指定できる値は、過去と現在の交通情報から、移動時間の推定値が最良であると仮定する best_guess、出力された移動時間が実際の移動時間より長いと仮定する pessimistic、出力された移動時間が実際の移動時間より短いと仮定する optimistic の3つである。

transit_mode : 優先する公共交通機関を1つ以上指定する。

transit_routing_preference : 移動する際の条件を指定する。transit_routing_preference パラメータの値は、徒歩での移動をできるだけ少なくする less_walking と、乗り換え回数をできるだけ少なくする few_transfers がある。

3.2.3 レスポンスの出力結果

Distance Matrix API のリクエストに対するレスポンスは、リクエストの URL 内にて指定された形式で返される。本研究では、JSON 形式に指定する。図4は、JSON 形式によるレスポンスの例である。

```
{
  "status": "OK",
  "origin_addresses": [
    "Vancouver, BC, Canada",
    "Seattle, État de Washington, États-Unis"
  ],
  "destination_addresses": [
    "San Francisco, Californie, États-Unis",
    "Victoria, BC, Canada"
  ],
  "rows": [
    {
      "elements": [
        {
          "status": "OK",
          "duration": {
            "value": 340110,
            "text": "3 jours 22 heures"
          },
          "distance": {
            "value": 1734542,
            "text": "1 735 km"
          }
        }
      ]
    },
    {
      "status": "OK",
          "duration": {
            "value": 24487,
            "text": "6 heures 48 minutes"
          },
          "distance": {
            "value": 129324,
            "text": "129 km"
          }
        }
      ]
    },
    {
      "status": "OK",
          "duration": {
            "value": 28834,
            "text": "3 jours 8 heures"
          },
          "distance": {
            "value": 1489604,
            "text": "1 490 km"
          }
        }
      ]
    },
    {
      "status": "OK",
          "duration": {
            "value": 14368,
            "text": "4 heures 0 minutes"
          },
          "distance": {
            "value": 135822,
            "text": "136 km"
          }
        }
      ]
    }
  ]
}
```

図4 JSON 形式でのレスポンスの例。

4. vcopt

vcopt [8] は Vignette&Clarity 合同会社により開発された、GA を使用した組合せ最適化ソルバーである。調整が難しい GA に対し、すべてのハイパーパラメータを自動的、かつ動的に決定することで、高い汎用性とユーザビリティ、および計算効率の実現を目的として作成されている。Python で記述されており、vcopt には様々な機能が実装されている。

4.1 vcopt の機能

vcopt の機能の一覧を表3に示す。本研究では、巡回セールスマン問題の大域最適化を行う機能を持つ vcopt().tspGA() 関数を用いる。

表3 vcopt の機能一覧。

機能	関数
順序の局所最適化	vcopt().opt2()
順序の大域最適化	vcopt().tspGA()
離散地の組合せ最適化	vcopt().dcGA()
選択枝の組合せ最適化	vcopt().setGA()
連続値の組合せ最適化	vcopt().rcGA()

4.2 vcopt().tapGA() 関数

vcopt.tspga() 関数では、巡回セールスマン問題の大域最適化が可能である。親のもつ辺を、2 手先までを重み付けして確率的に採用する交叉法を採用し、独自の高速化アルゴリズムを用いている。

4.2.1 ハイパーパラメータ

vcopt では、ハイパーパラメータは自動的、かつ動的に決定される。表 4 にハイパーパラメータの一覧を示す。

表 4 ハイパーパラメータの一覧.

個体数、親の数、子の数	10n,2,4 (n は次元数)。
交叉法	親のもつ辺を、2 手先までを重み付けして確率的に採用する。
突然変異率	各辺ともおおむね $1/n$ 。
選択法	エリート選択。
終了条件	目標達成、もしくは個体群の平均評価値に変化がなくなった場合。

4.2.2 利用方法

vcopt().tspGA() 関数には、必須の引数とオプションの引数が存在し、それらの引数に値を適切に設定する必要がある。表 5 と表 6 に引数の一覧を示す。

表 5 必須引数一覧.

para_range	順序を並び替えたい要素の 1 次元配列。
score_func	評価関数。単一の数値である評価値を返すように設定する。
aim	目標評価値。比較演算子を含む文字列も指定可能。

表 6 オプション引数一覧.

show_pool_func	並び替えの様子を可視化する関数。
seed	乱数シードを指定。
pool_num	個体数を指定。必ず 2 以上の偶数とする。
max_gen	最大世代数。
core_num	並列計算用のノード数。

5. 提案手法

本研究では、Google Distance Matrix API によって取得した実距離の利用と、厳密解法をアレンジして再配達に対応した配達経路最適化法を提案する。5.1 節では本研究にて用いられる 2 つの最適化法について、5.2 節では提案手法を評価するための配達シミュレーションについて説明をする。

5.1 配達ルートの最適化

研究では、配達の出発前に行う配達ルートの最適化と、再配達依頼が発生し配達ルートを再度最適化する必要が生じた際に行う配達ルートの最適化の 2 種類の最適化を用いる。以下にて、それぞれの最適化について詳しく説明する。

5.1.1 出発前の最適化

配達に出発する前に、まず配達ルートを決める必要がある。このときの配達ルートとは、配達拠点を出発してすべての配達先を訪れた後に再び配達拠点に戻るルートを指し、この最適化は TSP となる。そこで出発前の最適化では、TSP の解法を用いることとし、短時間で最適化を行うために近似解法である遺伝的アルゴリズムを利用する。以下にて遺伝的アルゴリズムについて説明する。

遺伝的アルゴリズム (GA:Genetic Algorithm) は、生物の進化をモデルとしたアルゴリズムである。GA は、解の候補を個体を持つ遺伝子として記述する。遺伝子に対して選択、交叉、突然変異などの遺伝的操作を繰り返し行い、より優秀な個体を生成する。図 5 に一般的な GA の流れを示す。

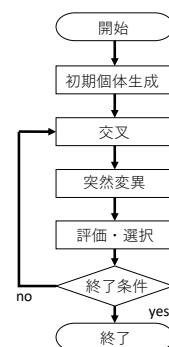


図 5 遺伝的アルゴリズムの一般的な流れ.

5.1.1.1 巡回経路の遺伝子表現.

GA を TSP に対して用いるためには、巡回経路を遺伝子として表現する必要がある。本研究では、出発点となる配達拠点から巡回する順番に配達先の番号を列挙した数列を遺伝子とするパス表現 (path representation) を用いる。図 6 のように、A~F までの配達先の番号が振られた配達先群を、A から順番に巡る巡回経路は (A,B,C,D,E,F) と表す。

5.1.1.2 交叉

交叉とは、複数の遺伝子間でその一部を交換することで新たな個体を作り出す操作のことである。TSP を GA で解く際、交叉法によっては巡回経路ではない個体が生み出される場合がある。このような個体は「致死性遺伝子を持つ個体」と呼ばれる。GA で TSP を解く場合、致死性遺伝子を持つ個体の生成を抑制する交叉法を用いることが重要である。致死性遺伝子を持つ個体の生成を抑制する交叉法の例として、前川ら [10] によって提案された、親に含ま

れる経路のみを用いて巡回経路を生成する枝交換交叉法 (EXX:Edge Exchanging Crossover) などがある。

5.1.1.3 突然変異

突然変異とは、一定の確率で遺伝子の一部を変化させる操作のことである。遺伝子が無作為に変更することにより、交叉のみでは見つけないことのできない経路の発見を可能とする。一般的には局所解からの脱出に効果があるとされる。

5.1.1.4 評価・選択

個体の評価は巡回経路の長さに基づいて行い、巡回経路長が短いほど評価は高くなる。個体を評価した後、次世代の親個体となる個体を選択する。もっとも単純な選択として、巡回経路長が短い順に並び替え、評価が高い順に次世代の親個体として選択するランキング選択がある。各世代の最良解を必ず次世代に残すエリート選択、評価が高い程、次世代の親個体として選ばれる確率が高くなるルーレット選択がよく用いられる。

5.1.2 再配達依頼発生時の最適化

再配達の依頼が発生した場合、通常の巡回経路となる TSP ではなく、出発地点と到着地点が異なる経路の最適化を行うこととなる。再配達時の最適化においては、再配達の依頼が発生した際の位置が出発地点、配達開始時点での出発地点が到着地点となる。出発地点から到着地点までの配達ルートは遺伝的アルゴリズムで既に最適化されている。その配達ルートに、再配達先を組み込む必要がある。よって、再配達時の最適化は再配達先をいずれの配達先間に挿入するかを列挙することによって行う。図 7 で再配達時の最適化の様子を表す。

5.2 配達シミュレーション

配達シミュレーションを通じて提案手法を評価する。5.2.1 節にてシミュレーションの流れについて説明する。配達先数や配達エリアなどのシミュレーションにおける設定については 5.2.2 節～5.2.4 節にて説明する。

5.2.1 シミュレーションの流れ

配達シミュレーションの流れを図 8 に示し、以下で説明

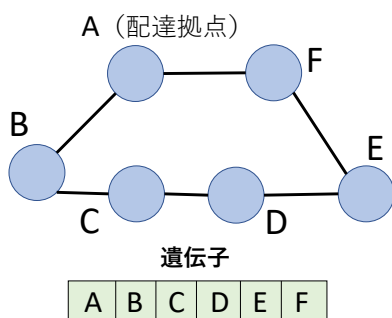


図 6 巡回経路の遺伝子表現。

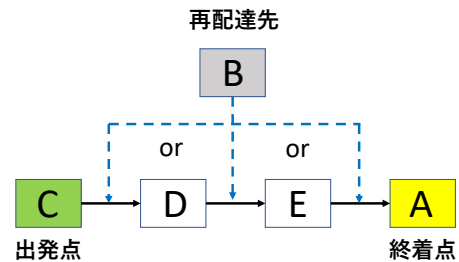


図 7 再配達依頼発生時の最適化。

する。

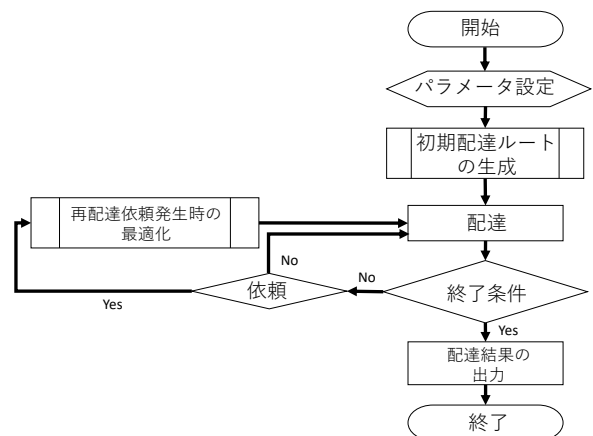


図 8 配達シミュレーションの流れ。

STEP1 パラメータを設定する。

STEP2 遺伝的アルゴリズムで配達経路の最適化を行い、初期配達ルートを生成する。

STEP3 生成した配達経路に沿って次の配達先へ移動する。

STEP4 すべての配達先をまわり拠点に戻るか、配達制限時間に達するまで、STEP3 へ戻すことを繰り返す。再配達依頼があれば再配達時の最適化を行う。

STEP5 配達履歴を出力して終了する。

5.2.2 配達エリアと配達先の配置

配達シミュレーションにおけるエリアは、宮崎市清武町、吉村町、赤江地域自治区の3つとする。各エリアごとに、エリア内に存在する建物の中から配達先をランダムに選択する。配達先数は10件から50件まで10ずつ変化させる。

5.2.3 配達制限時間

シミュレーションにおいては、配達制限時間を設定する。これには配達を任うドライバーの過剰な労働を防ぐ意図がある。配達制限時間は出発地点からの走行距離で制御する。本研究のシミュレーションでは、配達先間に信号や渋滞がなく、その道のりを常に同じ速度で走行していると仮定しているためである。本研究では、配達制限時間（最大走行距離）は、配達先数10件につき32kmとする。

表 7 パラメータの値.

パラメータ	値
配達エリア（宮崎市）	清武町、吉村町、赤江地域自治区
配達先数	10,20,30,40,50 件
不在率	20%
配達時間（距離）	配達先数 10 件ごとに 32km
再配達依頼発生率	60%
実験回数	配達エリア、配達先数毎に 10 回

5.2.4 再配達依頼の発生率

配達シミュレーションでは、最初の訪問時に不在だった配達先から再配達依頼が来る場合がある。再配達依頼は、不在宅を出発して次の配達先へ移動するたびにランダムに発生する。再配達依頼の確率はどの配達先でも同じとする。また、再配達依頼発生確率は、パラメータとしてシミュレーションのはじめに設定する。本研究では、再配達依頼の発生率は 60 % とする。また、不在率を 20 % とする。

6. 評価と結果

6.1 評価方法

表 7 に示すような配達エリアや不在率、配達先数、配達時間、再配達依頼発生率の設定のもと配達シミュレーションを行う。再配達の成功率と配達時間内での配達の成功率、および初期ルートの生成時間で提案手法を評価する。再配達の成功率 R は式 (1) で、配達制限時間内での配達の成功率 T は式 (2) により定義される。

$$R(\%) = \frac{\text{再配達}}{\text{再配達依頼の発生数}} \times 100. \quad (1)$$

$$T(\%) = \frac{\text{配達制限時間内での配達}}{\text{総配達先数}} \times 100. \quad (2)$$

6.2 実験結果

再配達依頼の発生率を 60%、不在率を 20% とし、配達エリアと配達先数を変化させたシミュレーションをそれぞれ 10 回ずつ行った。平均 R と平均 T を図 9 と図 10 にそれぞれ示す。また、初期ルートの生成時間を図 11 に示す。図 9 と図 10 から、配達エリアによって、配達先数が増えると平均 R 、平均 T とも大きく影響を受けることが分かる。これは、配達エリアによっては配達先が広く散らばっており、各配達先間の距離が長くなってしまったためであると考えられる。図 11 から、初期配達ルートの生成時間でも配達エリアによって約 2 倍の差が生じることが分かる。

7. おわりに

インターネットによる通信販売の普及や、ネット通販における送料無料などのサービスにより、宅配便の取扱個数は年々増加傾向にある一方、宅配会社の人材不足も相まって、ドライバーの労働負荷も増す一方である。再配達の多

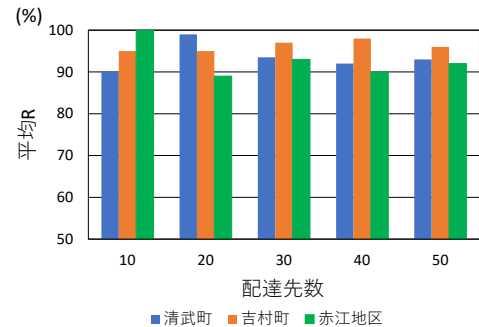


図 9 各平均 R .

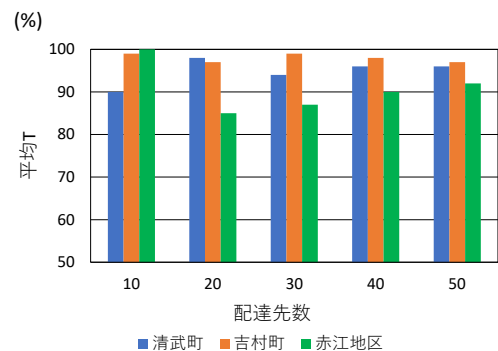


図 10 各平均 T .

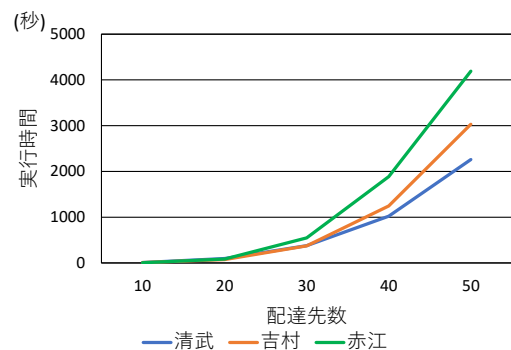


図 11 初期ルートの生成時間.

さも問題の一つであり、宅配ボックスの設置などの対策も進められているが、大規模マンションでは宅配ボックスがすぐ荷物で埋まってしまい、宅配会社同士の競合といった問題も生じている。

これらの現状から、ドライバーの負担を減らすための配達業務のサポートは大変に重要であり、より少ない走行距離で効率良く配達業務を行うための配達ルート最適化サービスが注目されている。

本研究では、Web API によって取得した実距離の利用と、厳密解法をアレンジすることで再配達に対応した配達経路最適化法の提案と評価を目的とした。

Web API として、Google が提供する Web 地図サービスである Google Maps Platform 上で利用できる Distance Matrix API を用いることで、各配達先間の実距離の取得を可能にした。

実験の結果、配達先数が 10 から 50 に増えても、配達成功率は 90% 以上を保てることが分かった。

今後の課題として、初期配達ルート生成の高速化や、渋滞や通行制限などの交通情報を考慮した配達シミュレーションが挙げられる。

謝辞

本研究は JSPS 科研費 19K12139 の助成を受けたものです。

参考文献

- [1] 「アマゾン大過ぎ」ヤマトドライバーから悲鳴続出、「利便性」が生んだ過酷な実態. https://news.infoseek.co.jp/article/bengoshi_5530, Dec. 2016.
- [2] Gurutto — 複数地点をグルっと一筆書き. <https://gurutto.geoinfo.co.jp/Gurutto-ssl/aboutGurutto.htm>, Aug 2018.
- [3] Routecreator — キャノン it ソリューションズ. <https://www.canon-its.co.jp/products/rc/>, Nov 2018.
- [4] 藤本, 山本学, 寺野隆雄. 宅配便ドライバの動的ルートスケジューリング. 計測自動制御学会第 5 回社会システム部会研究会資料, 第 05 巻, pp. 77–84, 2014.
- [5] S. Ohsugi and N. Koshizuka. Delivery route optimization through occupancy prediction from electricity usage. In 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Vol. 01, pp. 842–849, July 2018.
- [6] 柳浦 睦憲, 野々部 宏司, 分枝限定法: さらなる計算効率の希求 (特集号) 強く柔らかく…数理計画アプローチ再訪, システム/制御/情報, 50 巻, 9 号, p. 350–356, 2017.
- [7] Google Maps Platform — Google Developers, <https://developers.google.com/maps>
- [8] お手軽最適化パッケージ「vcopt」仕様書, <https://vigne-cla.com/vcopt-specification/>
- [9] W. Ellili, M. Samet and A. Kachouri, “Traveling salesman problem of optimization based on genetic algorithms,” 2017 International Conference on Smart, Monitored and Controlled Cities (SM2C), Sfax, pp. 123–127, 2017.
- [10] 前川景示, 玉置久, 喜多一, 西川緯一, “遺伝アルゴリズムによる巡回セールスマン問題の一解法”, 計測自動制御学

会論文集, Vol. 31, No. 5, pp. 598–605 (1995).