「マルチメディア，分散，協調とモバイル
(DICOMO2021) シンポジウム」 令和3年6月-7月

乗客の移動を考慮した相乗り経路探索の高速化手法

天野 雅人[1]　金 鎔煥[1]　山本 大介[1]　高橋 直久[1]

概要：近年，海外では相乗り配車サービスが普及している．相乗りの利点として，交通量の減少，環境負荷の低減，交通料金の低下などが挙げられる．ただし，乗車位置が固定されていると，一方通行などで遠回りが生じ，総経路長が長くなったり，乗客が多いほど探索時間が膨大にかかるという問題点がある．そこで本研究では，乗客が少し移動することで総経路長を短縮する高速な相乗り経路探索の実現を目的とする．本論文では乗客の移動を考慮した相乗り経路探索システムを提案，実装した．さらに，2 つの提案手法を用いて，経路探索の高速化の検証と，乗客の最大移動距離と乗客数の増加に伴う相乗り経路長と探索時間の推移の評価を行った．提案手法の 1 つは事前探索不要な手法で，もう 1 つは事前探索が必要だが高速な手法である．評価実験の結果，後者の手法の探索時間は前者に比べて，最大移動距離が 400m のとき約 150〜200 倍高速になった．さらに，事前探索は一度だけで済み，400 秒程度しかかからないため，実用上は後者の手法が良いと考える．また，乗客に 1km の移動を許容した場合，乗客が移動しないときの相乗り経路と比較すると，乗客数 5 人のとき約 30%，20 人のとき約 45%，50 人のとき約 50%，経路長を短縮できることがわかった．ほかにも，乗客の最大移動距離を約 250m にすると，最も効率よく相乗り経路を短縮できることがわかった．

A Fast Routing Method of Ride Sharing Considering Passenger Movement

MASATO AMANO[1]　YONGHWAN KIM[1]　DAISUKE YAMAMOTO[1]　NAOHISA TAKAHASHI[1]

1. はじめに

近年，日本では自動車の保有台数の増加が見られる．自動車検査登録情報協会 [1] によると，2020 年での日本の乗用車の保有台数は約 6180 万台にもなり，年々増え続けていることがわかる．しかしながら，自動車の保有台数の増加に伴って自動車の維持費がかかることや交通量が増加するという問題が考えられる．これらの問題を解決するため，相乗り配車サービスへの期待が高まっている．

相乗り配車サービスは海外で普及しており，代表例としてアメリカの Uber[2] や中国の DiDi[3] が挙げられる．このサービスはスマートフォンのアプリを利用して自家用車の所有者と自動車に乗りたいユーザを結びつけるものである．そのうえで，運転手は乗客を迎えに行き，目的地まで送るタクシーのような役割を果たす．この相乗り配車サービスを利用することで，ユーザは必要な時に自動車を手配するため，個人で自動車を保有する必要がなく，自動車の維持費に対する負担を解消できる．また，自動車 1 台に同じ目的地をもった複数の乗客が乗車することで，交通量が減少し，交通渋滞の解消や環境負荷の低減が見込まれる．さらに，乗車人数が増えるほど交通料金が低下することも考えられる．しかしながら，乗客数が複数人でそれぞれの乗車位置が異なる場合，乗客を迎えに行く場所が増えるため，乗客が 1 人の時と比べて，経路がより複雑になり総経路長が長くなることが予想される．

現在日本では，一般人が自家用車を用いて有償で他人を運送することは「白タク」行為となり，法律で禁止されているため，現在では一部の地域を除いて認められていない．ただし，国土交通省 [4] によると，平成 30 年 1 月から「相乗りタクシー」の実証実験が行われた．これは，配車アプリを活用して複数の利用者を 1 台のタクシーにマッチングするサービスである．このように，日本でもタクシー業界と連携して相乗り配車サービスの導入を目指している．

[1] 名古屋工業大学大学院工学研究科
Graduate School of Engineerong, Nagoya Institute of Technology

そこで本研究では，相乗りの特徴である乗車位置が自由であることを活かして，欠点である総経路長が長くなってしまうことを少しでも補うような，乗客が少し移動することで総経路長を短縮する高速な相乗り経路探索の実現を目的とする．

本論文では提案システムにより以下の課題を解決する．

課題 1 相乗り経路では乗客一人一人迎えに行ったり，乗車位置によっては一方通行などで遠回りが生じたりするため，乗客一人の時と比べて経路長が長くなってしまう．

課題 2 相乗り経路探索では乗客数の増加に伴って考えられる経路数も増加する．そのうえ，乗客の移動を許容すると，さらに経路数が増加する．そのため，経路探索にかかる計算時間が膨大になってしまう．

以上の課題を解決するために，提案システムは以下の特徴を持つ．

特徴 1 乗客の移動を許容して相乗り経路探索をすることで経路長を短縮する．

特徴 2 まず，地図上のすべての 2 地点間の最短経路をワーシャルフロイド法を用いて事前に探索する．さらに，焼きなまし法を用いた巡回経路探索から乗客の乗車順を決定した後，Viterbi アルゴリズムを用いて相乗り経路探索を行う．これにより経路探索を高速化する．

2. 関連研究

複数の乗客に対する相乗り経路探索について様々な研究がなされている．

文献 [5] では，複数の SAV(Smart Access Vehicle) でのライドシェアにおいて，リアルタイムで最適な SAV に個別の目的地をもった乗客を割り当てるアルゴリズムの性能評価を行っている．SAV とはライドシェアサービスを提供する車両のことである．このアルゴリズムでは，ある時間に乗車地点と降車地点をもつデマンドが発生した時に，デマンドを割り当てる計算時間とユーザの目的地到着時間に関わる新経路のコストとのバランスを考慮してデマンドの割り当てをしている．

文献 [6] では，共通の目的地をもつ複数の乗客を最適なタクシーに割り当てるアルゴリズムを提案している．このアルゴリズムでは，複数の乗客の乗車位置が与えられたうえで，タクシーの総走行距離が長くならない中で，乗客の総移動距離が最小になることを目的として，乗客を最適なタクシーに割り当てている．

文献 [7] では，さまざまな乗客定員を備えた複数の車両を使用して，最適な車両に乗客を動的に割り当てたり需要に応じて再配置したりするシステムを提案している．このシステムでは，ペアワイズ共有可能性グラフから実行可能なトリップを計算し，次にトリップを車両に割り当てることによって問題を解決する．このシステムはリアルタイムの要求に基づいて継続的に経路を変更できる自動運転車に特に適している．

3. 提案システムの概要

3.1 提案システムの特徴

本論文では，以下の特徴を有したシステムを提案する．

3.1.1 乗客の移動を許容することでの相乗り経路長の短縮

乗客が初期位置から徒歩で，あらかじめユーザが設定した最大移動距離以内で訪れることができるノード（交差点）をその乗客の乗車位置候補とする．そのうえで，最短の相乗り経路を探索することで相乗り経路長を短縮する．これにより課題 1 を解決する．

3.1.2 相乗り経路探索の高速化

まず，地図上のすべての 2 地点間の最短経路をワーシャルフロイド法を用いて事前に探索する．これにより，使用する 2 地点間の最短経路を逐一探索する必要がなく，探索時間を短縮できる．また，巡回経路探索では焼きなまし法を用いて近似解を求め，これにより得られた乗車順を乗客の乗車順として決定する．さらに，相乗り経路探索では乗客の乗車順と乗車位置の候補から Viterbi アルゴリズムを用いて最短の相乗り経路を探索する．このように，乗車順を限定してから相乗り経路探索を行うことで，考えられる相乗り経路の候補を削減でき，計算時間を短縮する．このようにして相乗り経路探索の高速化を図り，課題 2 を解決する．

3.2 提案システムで使用するデータ

提案システムにおいて使用するデータは，道路データである．

3.2.1 道路データ

道路データの構造を図 1 に示す．道路データは，ポイント，ノード，アークの 3 つのデータで構成されている．ポイントは，道路上の点を表し，位置座標（緯度，経度）からなる．道路の形状は図のように道路上の隣接ポイントを順番につないでできる折れ線で近似的に表現される．ノードとは，道路上の交差点である．アークは始点ノードから終点ノードに至る道路であり，道路上のポイントの系列で表す．リンクはアークとノードをまとめたものである．リンクは表 1 に示すデータ構造を持つ．また，アークは線を表現するジオメトリ型 LineString の形式で表現する．

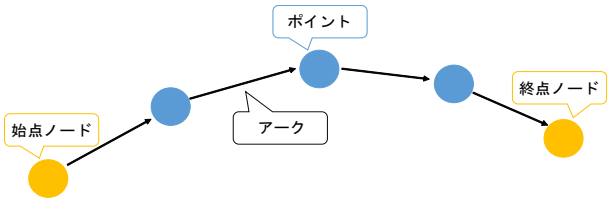

図 1: 道路データの構造

表 1: リンクテーブル

カラム名	データ型	説明
ID	Integer	リンクを一意に識別するリンク ID
clazz	Integer	道路のクラス
source	Integer	リンクの始点ノード ID
target	Integer	リンクの終点ノード ID
x1	Double	リンクの始点ノードの経度
y1	Double	リンクの始点ノードの緯度
x2	Double	リンクの終点ノードの経度
y2	Double	リンクの終点ノードの緯度
km	Double	ノード間のノードの長さ (単位は km)
kmh	Double	制限速度
reverse_cost	Double	反対方向へのコスト
geom_way	geometry	リンクの形状を表す (アーク)

3.3 提案システムの構成

提案システムの構成を図 2 に示す．まず，事前探索として道路データから地図上のすべての 2 地点間の最短経路を探索し，最短経路データに格納する．次に，ユーザが入力した，出発地点，中間地点，到着地点，乗客の最大移動距離の 4 つの入力データと，最短経路データから相乗り経路探索を行う．最後に，アプレット上に相乗り経路を表示する．

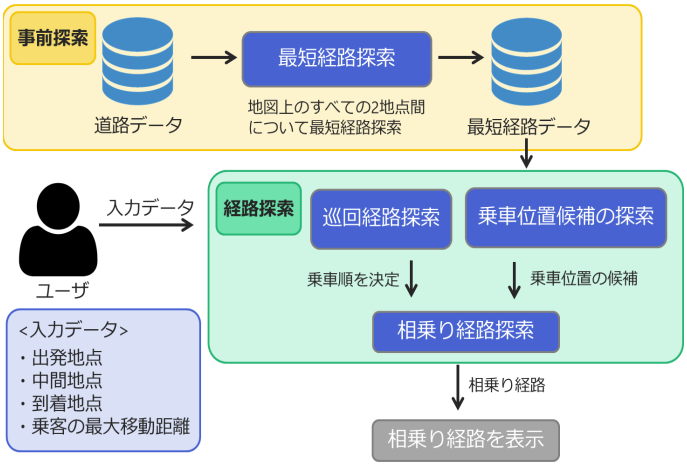

図 2: 提案システムの構成

4. 提案システムの実現法

提案システムは，最短経路探索，巡回経路探索，乗車位置候補の探索，相乗り経路探索からなる．それぞれの実現法を以下に示す．

4.1 最短経路探索

この節では事前探索で行う最短経路探索の実現法について述べる．事前探索で取得する最短経路のデータは提案システムでの巡回経路探索や乗車位置候補の探索，相乗り経路探索で利用する．例えば図 3 のような道路ネットワークがあった場合，巡回経路探索を行うとき，地点 S，A，B，C から考えられるすべての 2 地点間の最短経路のデータが必要になる．さらに，乗客の移動を考慮した相乗り経路探索を行う場合では，地点 S，A，B，C 以外にもそれぞれの乗客が移動できる乗車位置の候補も含めた 2 地点間の最短経路のデータが必要になる．そのうえ乗客の人数が増加すると，必要な 2 地点間の最短経路データがさらに膨大になる．そのため提案システムでは，必要な 2 地点間の最短経路をその都度一組ずつダイクストラ法で探索するのではなく，ワーシャルフロイド法 [8] を利用して地図上のすべての 2 地点間の最短経路を一度に探索する．

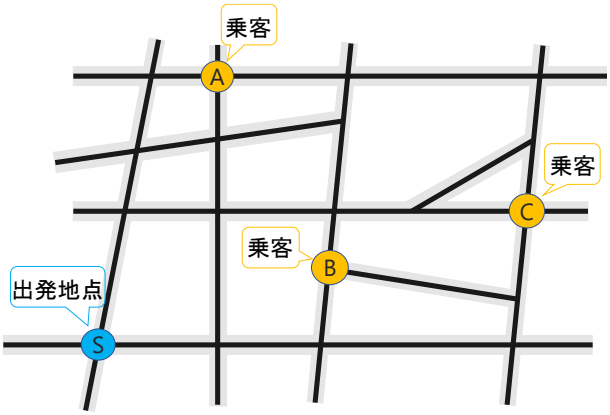

図 3: 道路ネットワークの例

4.1.1 ワーシャルフロイド法

ワーシャルフロイド法は，グラフの全ての頂点の間の最短路を見つけるアルゴリズムである．具体的には，「3 つの頂点 a，b，c を選んで，a → b → c という道が a → c という道より短ければ a → c の距離を更新する」という操作をすべての頂点の組み合わせで繰り返して最短経路を確定させていく．これは単純な 3 重ループで実現可能であり，計算量は頂点数を V とすると $O(V^3)$ である．V=1,...,n 上のグラフ G=(V,E) についてのワーシャルフロイド法のアルゴリズムを Algorithm1 に示す．

Algorithm 1 ワーシャルフロイド法

```
グラフ G=(V,E) および各辺 e ∈ E の長さ length(e) を入力とする
//初期化
for i ∈ V do
  for j ∈ V do
    d_{i,j}← i と j を結ぶ辺 e があれば length(e)，なければ無限大
  end for
end for

//本計算
for k ∈ V do
  for i ∈ V do
    for j ∈ V do
      if d_{i,j} > d_{i,k} + d_{k,j} then
        d_{i,j}← d_{i,k} + d_{k,j}
      end if
    end for
  end for
end for
```

次に 3 つのスケールでのワーシャルフロイド法の探索時間を表 2 に示す．いずれの探索時間も 5 回ずつ探索を行ったときの平均を求めた．

表 2: ワーシャルフロイド法の探索時間

スケール	縦横の距離	ノード数	探索時間
16	約 1.35km	495	0.155s
15	約 2.70km	1861	9.153s
14	約 5.40km	6776	412.773s

4.2 巡回経路探索

この節では巡回経路探索の実現法について述べる．本研究における巡回経路とは，出発地点からすべての乗客の初期位置を巡って出発地点に戻ってくる経路のことを言う．提案システムでは図 4 のように，一方通行を考慮しない巡回経路を焼きなまし法を利用して探索をすることで乗客の乗車順を決定する．

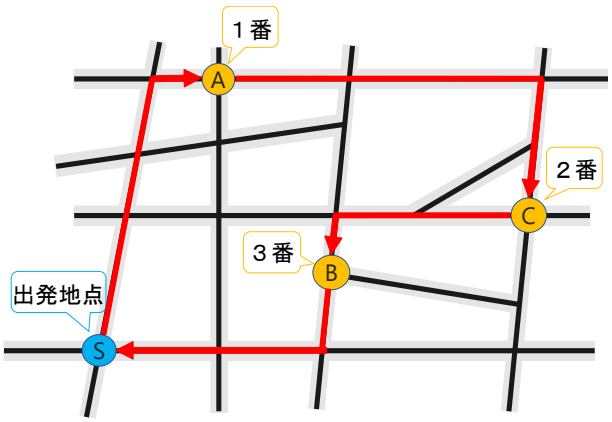

図 4: 巡回経路の例

4.2.1 焼きなまし法の更新確率

本研究において，反復回数 L での試行回数 k 回目の焼きなまし法の更新確率 P(k) は

$$P(k) = exp\left(-\frac{\Delta E}{T}\right)$$

とする．ただし，

$$\Delta \mathrm{E} = \text{変更後の経路長 (m) - 変更前の経路長 (m)}$$

$$T = \frac{L-k}{L} \times 100 (0 < T \leqq 100)$$

である．

4.2.2 巡回経路探索の手順

巡回経路探索は以下のような手順で行う．

手順 1 初期解の設定
初期解として訪れる順番を貪欲法で決定する．

手順 2 2 つの訪れる順番を入れ替える
訪れる順番を 2 か所ランダムで決め，それらを入れ替える．

手順 3 コスト比較
入れ替える前と入れ替えた後のコストを比較し，入れ替えた後のほうがコストが良ければ，暫定解を更新し，手順 5 へ進む．

手順 4 焼きなまし法
入れ替える前の方がコストが良い場合，焼きなまし法により確率でコストが悪くても暫定解を更新する．

手順 5 ループ脱出判定
一定回数実行したらループを抜けて終了する．一定回数満たしていない場合は，手順 2 へ戻る．

4.3 乗車位置候補の探索

この節では乗車位置候補の探索の実現法について述べる．乗車位置候補の探索では，ユーザが入力した乗客の最大移動距離をもとにそれぞれの乗客の乗車位置の候補を探索する．これについて，以下のような手順をすべての乗客に対して行う．

手順 1 最短経路データを取得
ある乗客の初期位置が始点となるすべての最短経路データを取得する．

手順 2 乗客の最大移動距離と比較
手順 1 で取得した最短経路データの経路長のうち，乗客の最大移動距離以下の最短経路データを探索する．

手順 3 データを格納
手順 2 で取得した乗客の最大移動距離以下の最短経路データのうち，終点までの経路長が短いデータから順に，終点と最短経路のデータを格納する．この終点の集合が乗客の乗車位置の候補となる．

4.4 相乗り経路探索

この節では相乗り経路探索の実現法について述べる．相乗り経路探索では，乗客の乗車順と乗車位置の候補から，Viterbi アルゴリズム [9][10] を用いて乗客の移動を考慮した最短の相乗り経路を探索する．本研究での相乗り経路とは図 5 のように，それぞれの乗客に対して一定距離の移動を許容した上で，出発地点からすべての乗客を巡回して出発地点に戻る最短経路と定義している．例えば図 3 の道路ネットワークに対して相乗り経路を探索するとき，図 4 のように 3.2 節の巡回経路探索で決定した乗車順と 3.3 節で求めた乗車位置候補から，乗客 A と C が初期位置から移動することで経路が短縮されるため，相乗り経路は図 5 のような経路となる．

4.4.1 Viterbi アルゴリズム

Viterbi アルゴリズムは，何かの状態が与えられた時に，現在の状態に基づいて，その後に生じる状態の最も尤らしい並びを探索するアルゴリズムで，動的計画法の一種であり，特に隠れマルコフモデルに基づいている．したがって，相乗り経路探索に Viterbi アルゴリズム適用すると，複数の経路が考えられる中で，始点と終点とノード間のコストから最短経路を探索することができる．このときのアルゴリズムを Algorithm2 に示す．

Algorithm 2 Viterbi アルゴリズム

V = 乗車順に並んだ乗客の集合
V'_u = u 番目に乗車する乗客の乗車位置の候補ノード
E[u][v] = ノード u からノード v までの最短経路長
L[u] = 始点からのノード u までの最短の相乗り経路長 (初期値：∞)

```
//始点から 1 番目の乗客までの最短の相乗り経路長を格納
for i ∈ V'_1 do
    L[V'_1[i]] ← E[S][V'_1[i]]
end for

//始点から 2 番目以降の乗客までの最短の相乗り経路長を格納
for i ∈ {1, ..., V.length − 1} do
    for j ∈ V'_i do
        for k ∈ V'_{i+1} do
            if L[V'_{i+1}[k]] > L[V'_i[j]] + E[V'_i[j]][V'_{i+1}[k]] then
                L[V'_{i+1}[k]] ← L[V'_i[j]] + E[V'_i[j]][V'_{i+1}[k]]
            end if
        end for
    end for
end for

//始点から終点までの最短の相乗り経路長を格納
for i ∈ V'_{V.length} do
    L[T] ← L[V'_{V.length}[i]] + E[V'_{V.length}[i]][T]
end for
```

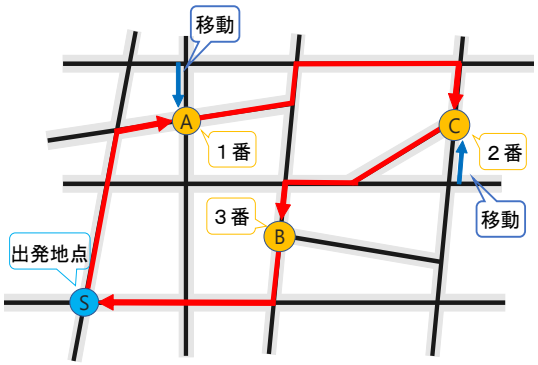

図 5: 相乗り経路の例

5. プロトタイプシステム

提案システムを実現するために，Java 言語を用いてプロトタイプシステムを実装した．開発環境として Eclipse，Windows10，データベースとして，PostgreSQL を使用している．プロトタイプシステムで使用しているデータベースは，道路データである．

5.1 システムの概要

プロトタイプシステムは，提案システムである乗客の移動を考慮した相乗り経路探索を実装した．このシステムでは，乗客の最大移動距離をユーザが適宜変更できるようになっている．地図データは OpenStreetMap[11] の地図データが利用された地図サーバから取得する．入力した緯度経度により地図サーバから地図画像を取得し，研究室サーバ内にあるデータベースから範囲内に存在する道路データを取得する．

5.2 システムの動作

プロトタイプシステムの操作画面を図 6 に示す．上部に入力インターフェースを，下部に地図画像を配置している．システムを実行すると，入力インターフェースで地図のスケールと乗客の最大移動距離を選択し，地図画像内から出発地点，中間地点，到着地点をクリックして選択し，相乗り経路探索ボタンをクリックすることで相乗り経路探索を行い，図 7 のように経路が描画される．探索した経路の経路長と探索時間は Eclipse のコンソールに出力される．

プロトタイプシステムの使い方を以下に示す．

(1) 使用したい地図の位置とスケールを設定する．

(2) 「道路データ」ボタンをクリックして事前探索を行う．

(3) 出発地点，中間地点，到着地点，乗客の移動距離を設定する．

(4) 「相乗り経路探索」ボタンをクリックして経路探索を実行する．

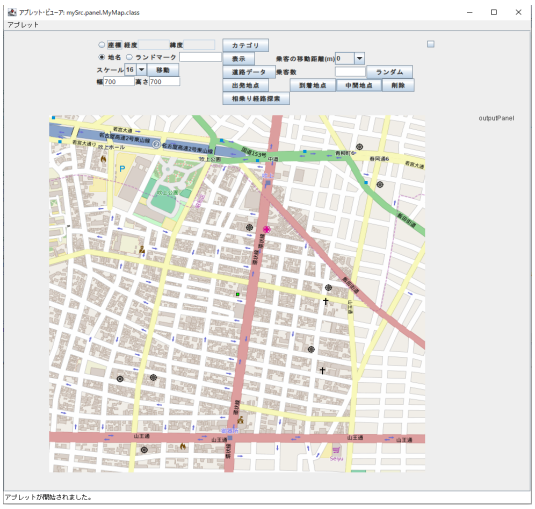

図 6: プロトタイプシステムの外観

図 7: 相乗り経路探索

6. 評価実験

本章ではプロトタイプシステムを用いた評価実験とその結果，考察について述べる．

6.1 実験目的

本実験の目的は，経路探索の高速化を検証することと，乗客の最大移動距離と乗客数の増加に伴う相乗り経路長と探索時間の推移を評価すること，乗客の総移動距離と短縮距離を比較することである．

6.2 実験方法

プロトタイプシステムを用いて，名古屋市内の約 5.4km 四方のある矩形範囲の中から任意の出発地点，中間地点，到着地点の組み合わせを選定して相乗り経路探索を行う．ただし，出発地点と到着地点は同じ位置とする．また，中間地点とは乗客の初期位置のことであり，中間地点の数が乗客数となる．

6.2.1 評価実験 1

乗客数を 5 人，20 人，50 人として出発地点，中間地点，到着地点の組み合わせをそれぞれ 10 組ずつ選定する．この組み合わせに対して，許容する乗客の最大移動距離を 0m から 100m おきに，5 人のときは 800m まで，20 人では 600m まで，50 人では 500m まで変化させて，以下の 2 パターンの相乗り経路探索を行う．

- 提案手法 1

  事前探索が必要な手法．ワーシャルフロイド法を利用して事前に地図上のすべての 2 地点間の最短経路を探索してから相乗り経路探索を行う．
- 提案手法 2

  事前探索不要な手法．ダイクストラ法を利用してそれぞれの経路探索で必要とする 2 地点間の最短経路のみを探索して相乗り経路探索を行う．

6.2.2 評価実験 2

乗客数を 5 人，20 人，50 人として出発地点，中間地点，到着地点の組み合わせをそれぞれ 100 組ずつ選定する．この組み合わせに対して，許容する乗客の移動距離を 0m から 200m おきに 1600m まで変化させて相乗り経路探索を行う．

6.3 評価尺度

6.3.1 評価実験 1

評価実験 1 の評価尺度として以下の尺度を用いる．また，この尺度は 10 回の探索の平均をとる．

- 探索時間

  3 つのパターンの探索時間を比較する．1 つ目は事前探索を含めた提案手法 1 の探索時間である．これは事前探索の探索時間と提案手法 1 での相乗り経路の探索時間を足した時間である．2 つ目は提案手法 2 の探索時間である．これは提案手法 2 での相乗り経路の探索時間である．3 つ目は提案手法 1 の探索時間である．これは提案手法 1 での相乗り経路の探索時間のみである．また，事前探索の探索時間は，表 2 で求めた 412.773 秒を用いる．

6.3.2 評価実験 2

評価実験 2 での評価尺度として以下の 3 つの尺度を用いる．また，これらの尺度は 100 回の探索の平均をとる．

- 相乗り経路長

  出発地点から移動を許容したすべての乗客を巡回して到着地点に戻ってくるまでの総経路長．

- 短縮率

  乗客の移動を許容せず乗客の初期位置を巡回する相乗り経路（巡回経路）と比較した，乗客の移動を許容した相乗り経路長の短縮率を求める．短縮率は以下の式で定義する．

$$\text{短縮率} = \frac{\text{巡回経路長 - 相乗り経路長}}{\text{巡回経路長}} \times 100$$

- 探索時間

  入力データを入力し終えた後から相乗り経路を探索し終えるまでの時間を計測する．

- 短縮効率

  乗客の総移動距離に対する相乗り経路長の短縮距離の割合．つまり，乗客が移動することで効果的に経路長を短縮できているかを表す指標である．短縮効率は以下の式で定義する．

$$\text{短縮効率} = \frac{\text{巡回経路長 - 相乗り経路長}}{\text{乗客の総移動距離}} \times 100$$

6.4 結果と考察

6.4.1 評価実験１の結果と考察

乗客数を 5 人としたときの探索時間の推移を図 8 に，乗客数 20 人のときを図 9 に，乗客数 50 人のときを図 10 に示す．それぞれの図の横軸は乗客の最大移動距離 (m) であり，縦軸は探索時間 (s) である．ただし，縦軸は対数目盛で表示している．

まず，事前探索を含めた提案手法１の探索時間は最大移動距離が増加してもほぼ一定であることがわかる．提案手法１では事前探索の探索時間に対する相乗り経路探索の時間が非常に短いため，探索時間の大半は事前探索の探索時間となる．したがって，事前探索を含めた提案手法１では最大移動距離が増加しても探索時間の増加は少ないと考えられる．

次に，提案手法２は移動距離の増加に伴って指数関数的に増加していることがわかる．これは，最大移動距離の増加に伴ってそれぞれの乗客の乗車位置の候補が増加し，相乗り経路探索で必要とする 2 地点間の最短経路が指数関数的に増加したためだと考えられる．

最後に，提案手法１の探索時間は提案手法２と比べて，最大移動距離が 0m のとき約 10 倍，400m のとき約 150〜200 倍高速になった．さらに，事前探索にかかる時間だけでも，乗客 5 人のとき最大移動距離 700m，20 人のとき 500m，50 人のとき 400m，の提案手法２の 1 回にかかる相乗り経路の探索時間と同程度であった．このように相乗り経路探索の高速化を実現できたため，課題 2 を解決できた．

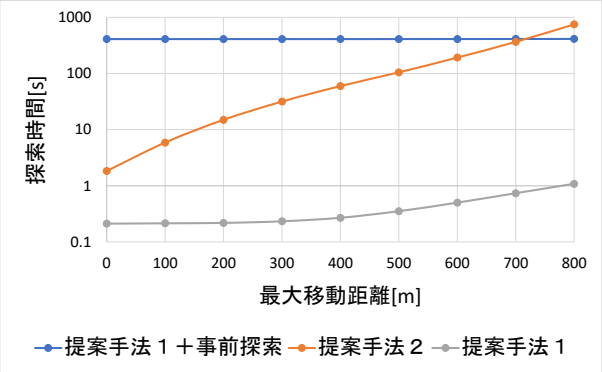

図 8: 乗客数 5 人の探索時間

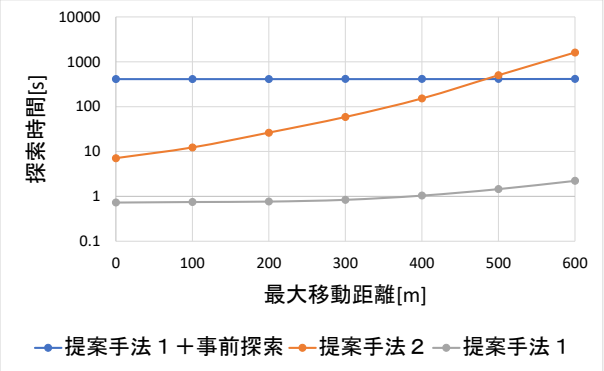

図 9: 乗客数 20 人の探索時間

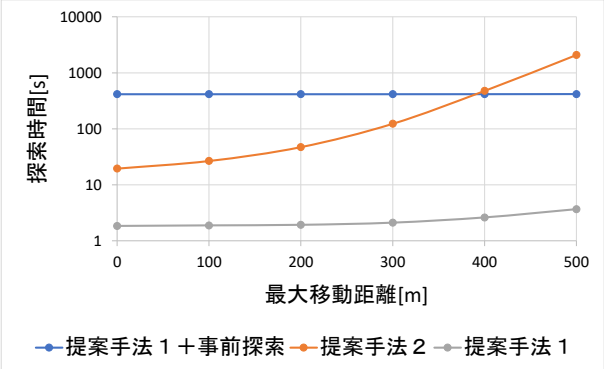

図 10: 乗客数 50 人の探索時間

6.4.2 評価実験２の結果と考察

- 相乗り経路長

  乗客数を 5 人，20 人，50 人としたときの相乗り経路長の推移を図 11 に示す．図 11 の横軸は乗客の最大移動距離 (m) であり，縦軸は相乗り経路長 (km) である．

  相乗り経路長は乗客数にかかわらず，最大移動距離が増加するにつれて短縮している．ただし，乗客数が 20 人と 50 人の場合は最大移動距離が短いときのほうが経路長の減少度合が大きいことがわかる．これは，最大移動距離が 0m のときに一方通行などにより生じ

てた遠回りなどが少し最大移動距離を増やすことで解消されたためだと考えられる．このように乗客の移動を許容することで経路長が短縮されたため，課題 1 を解決できた．

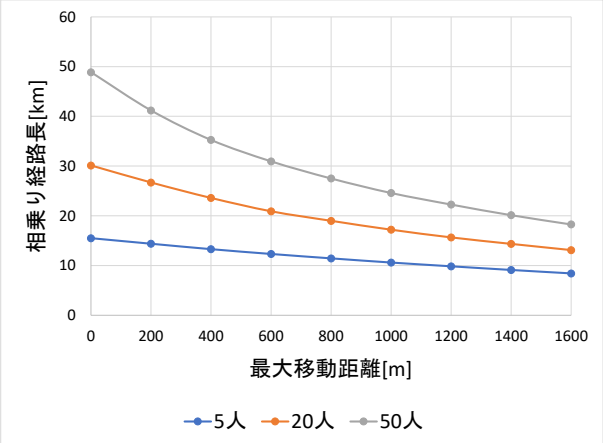

図 11: 相乗り経路長の推移

- 短縮率

乗客数を 5 人，20 人，50 人としたときの短縮率の推移を図 12 に示す．図 12 の横軸は乗客の最大移動距離 (m) であり，縦軸は短縮率 (%) である．

短縮率は乗客数にかかわらず，最大移動距離が増加するにつれて増加している．ただし，乗客数が 20 人と 50 人の場合は最大移動距離の増加に伴って短縮率の増加度合は小さくなっていることがわかる．また，乗客数が多いほど短縮率が大きいことがわかる．具体的に挙げると，最大移動距離が 1km のときの短縮率は，乗客数が 5 人のとき約 30%，20 人のとき約 45%，50 人のとき約 50%である．これは，乗客数が多いほど移動できる乗客の数が増えるため，経路を短縮できるポイントが増えるためだと考えられる．

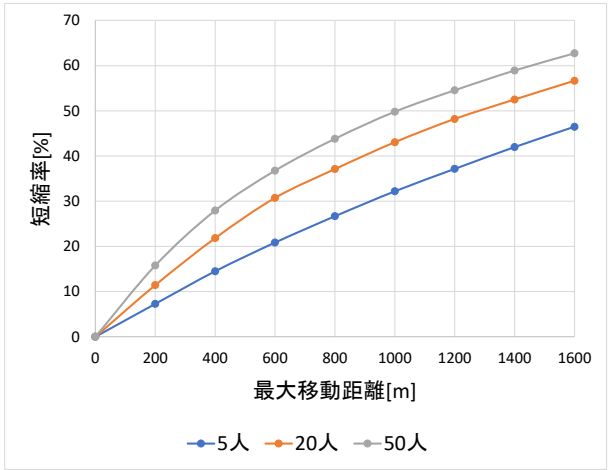

図 12: 短縮率の推移

- 探索時間

乗客数を 5 人，20 人，50 人としたときの探索時間の推移を図 13 に示す．図 13 の横軸は乗客の最大移動距離 (m) であり，縦軸は探索時間 (s) である．

探索時間は乗客数にかかわらず，最大移動距離の増加に伴って指数関数的に増加していることがわかる．また，乗客数が多いほど探索時間の増加度合も大きくなっていることがわかる．これは，最大移動距離が増加するほど，それぞれの乗客の乗車位置の候補が増加するので，Viterbi アルゴリズムにかかる探索時間が増加したり，乗客数が増加するほど，巡回経路や相乗り経路で考えられる経路の候補が増加するためだと考えられる．探索時間を 15 秒まで許容したとすると，最大移動距離が 800m のとき乗客 50 人でも探索時間は約 13.3 秒であったため，最大移動距離が 800m 以下の相乗り経路探索であれば 15 秒以内に探索可能である．

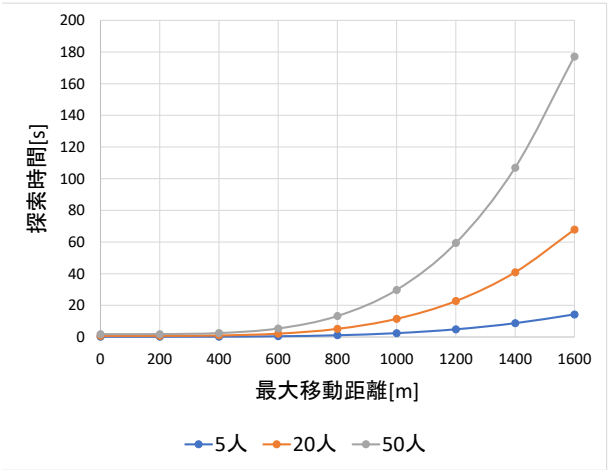

図 13: 探索時間の推移

- 短縮効率

乗客数を 5 人，20 人，50 人としたときの短縮効率の推移を図 14 に示す．図 14 の横軸は乗客の最大移動距離 (m) であり，縦軸は短縮効率 (%) である．

短縮効率は乗客数にかかわらず，乗客の最大移動距離が 250m 程度で最大の値をとることがわかる．このことから，同じ経路長を移動する自動車と乗客にかかるコストを等しいと仮定したとき，乗客の最大移動距離が約 250m のとき最も効率よく相乗り経路を短縮できており，これを超えると相乗り経路長は短縮されるものの，乗客の総移動距離も増加するため短縮の効率は悪くなってしまうと考えられる．

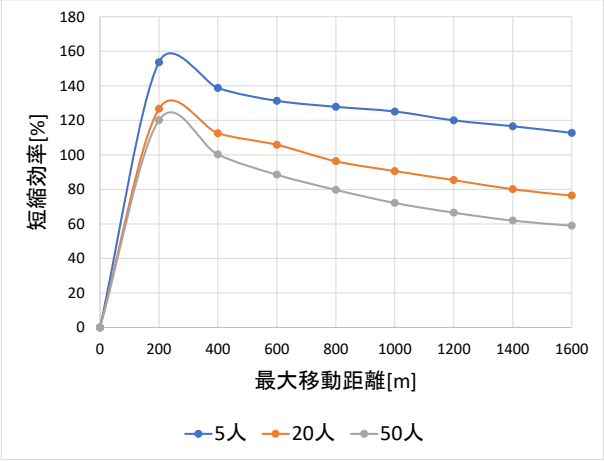

図 14: 短縮効率

7. おわりに

本論文では，乗客の移動を考慮した相乗り経路探索システムを提案し，その実現方法について述べた．また，提案システムに基づいてプロトタイプシステムを実装し，評価実験を行った．

評価実験では，2 つの提案手法を用いて経路探索の高速化の検証と，乗客の最大移動距離と乗客数の増加に伴う相乗り経路長と探索時間の推移の評価を行った．提案手法の 1 つは事前探索不要な手法で，もう 1 つは事前探索が必要だが高速な手法である．評価実験の結果，後者の手法の探索時間は前者に比べて，最大移動距離が 0m のとき約 10 倍，400m のとき約 150〜200 倍高速になった．さらに，事前探索は一度だけで済み，事前探索にかかる時間だけでも 400 秒程度で，乗客 5 人のとき最大移動距離 700m，20 人のとき 500m，50 人のとき 400m，の前者の手法の 1 回にかかる相乗り経路の探索時間と同程度であった．そのため，実用上は後者の手法が良いと考える．また，乗客に 1km の移動を許容した場合，乗客が移動しないときの相乗り経路と比較すると，乗客数 5 人のとき約 30%，20 人のとき約 45%，50 人のとき約 50%，経路長を短縮できることがわかった．ほかにも，乗客の最大移動距離を約 250m にすると，最も効率よく相乗り経路を短縮できることがわかった．

今後の課題としては，乗車位置の変更，近辺の乗車位置の統合，乗客別の移動距離の設定などが挙げられる．提案手法では乗車位置が交差点上になっており，実際には道路交通法に違反することになってしまうため，乗車位置をリンク上に変更する必要がある．また，相乗りでは乗降に一定の時間がかかるため，近辺の乗客の乗車位置を統合し，乗車位置を減らすことで，実際の乗降にかかる総時間の短縮が期待できる．ほかにも，提案手法では乗客の移動距離を一律に設定していたが，実際では，あまり移動できない高齢者や多く移動できる若者などが混在すると考えられるため，乗客別に移動距離を設定できるように改良する必要があると考えられる．

参考文献

[1] 自動車保有台数の推移 https://www.airia.or.jp/publish/file/r5c6pv000000g7vb-att/r5c6pv000000g7vq.pdf 2021-05-06 参照.
[2] Uber プラットフォームの詳細 | Uber https://www.uber.com/jp/ja/ 2021-05-06 参照.
[3] DiDi で今までにない移動体験を | DiDi モビリティジャパン株式会社 https://didimobility.co.jp/ 2021-05-06 参照.
[4] 報道発表資料：平成 30 年 1 月から「相乗りタクシー」実証実験開始 - 国土交通省 https://www.mlit.go.jp/report/press/jidosha03_hh_000273.html 2021-05-06 参照.
[5] 吉塚裕生，内田英明，藤井秀樹，吉村忍：ライドシェアサービス向け経路探索アルゴリズムの性能評価, The 32nd Annual Conference of the Japanese Society for Artificial Intelligence，2018，4F1-OS-11c-02.
[6] 吉田岳人，矢野正基，堀川健一郎，佐藤啓太，南翔太，繁野麻衣子：共通の目的地をもつ顧客によるタクシー相乗りのためのモデル作成と評価，情報処理学会研究報告，Vol.2018-MPS-117，No.3，2018/3/1.
[7] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus：On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment, PNAS January 17, 2017 114 (3) 462-467, first published January 3, 2017.
[8] 最短経路問題（ベルマンフォード法・ワーシャルフロイド法） - アルゴリズム講習会 https://dai1741.github.io/maximum-algo-2012/docs/shortest-path/ 2021-05-06 参照.
[9] ビタビアルゴリズム https://www.mnc.toho-u.ac.jp/v-lab/yobology/viterbi/viterbi.htm 2021-05-06 参照.
[10] ビタビアルゴリズム【入門】具体例でわかりやすく解説！（Viterbi） https://spjai.com/viterbi-algorithm/ 2021-05-06 参照.
[11] OpenStreetMap Japan | 自由な地図をみんなの手に/The Free Wiki World Map https://openstreetmap.jp/ 2021-05-06 参照.