道なり優先巡回経路生成システムとシミュレーションによる評価法

岡田　拓也† 　山本　大介† 　高橋　直久†

† 名古屋工業大学 大学院工学研究科 情報工学専攻　〒 466-8555 愛知県名古屋市昭和区御器所町
E-mail: †okadat@moss.elcom.nitech.ac.jp, ††{yamamoto.daisuke,naohisa}@nitech.ac.jp

あらまし　オンラインショッピングの普及により配送業が生活に欠かせない存在になっている。その配送業が取り決めている配送ルートは巡回経路問題の一つとして考えることができる。これは運送コストが最小になることを目的とした問題であり、基本的には最短距離を目指している。しかし、その場合右左折の回数が増えることがある。特に大型トラックといった運転をするのが難しい車では、この右左折での事故が発生しやすい。またストップ・アンド・ゴーの回数も増え、CO2 の量も増加する。さらに曲がる場所を勘違いする認知的なコストが増加するという点も挙げられる。そこで道なり優先の配送ルート、すなわち巡回経路を考える。道なり優先とは、道なり道路（ストロークと呼ぶ）を優先的に選ぶことである。ストロークとは、直感的には道なり道路であり、我々は、実際の道路データを用いて、交差点における道路のなす角度によりストロークを構成する道路を定め、データベース化している。ストロークをもとに 2 点間の経路を求め巡回経路を決定し、その経路をシミュレーションに反映し、運転の負担や二酸化炭素の排出といった環境への影響を測る。

キーワード　巡回経路問題, 道案内, 地理情報システム, 交通シミュレーション,SUMO

1. はじめに

1.1 研究の背景

Amazon や楽天市場といったオンラインショッピングが大きく普及し、利用者が増加している。利用者が購入した商品は、佐川急便やヤマト運輸などの配送業が請け負っており各利用者を巡っている。このため、配送業がより生活に欠かせないものになりつつある一方、運転手の負担が増加しており、負担の軽減が求められている。そのため、より効率の良い配送ルートを求める必要がある。この配送ルートの決定は巡回経路問題に当てはめて考えることができる。

巡回経路問題とは、出発地点からすべてのノードを 1 回ずつ訪れて、1 周するための運送コストが最小となるルートを求める最適化問題のことである。しかし最短距離で求めた経路は、曲がる回数が増えることがある。この回数が増えると、曲がる場所を勘違いする認知的なコストが増加したり、右左折待ちによる渋滞も増える。また右左折による事故が多いため、その発生も起きやすくなる。

そこで、道なりを優先とした巡回経路を考える。移動する距離は最短距離と比べると長くなってしまうが、右左折の回数が少なくなり、また視覚的にもわかりやすい経路になる。本研究では、道なりを優先とした巡回経路を求め、その経路を SUMO と呼ばれる交通シミュレーションソフトに反映して、その影響を測る (SUMO については別の章で解説する)。

1.2 関連研究

通常、巡回経路の経路探索に関しては、ダイクストラ法により最短経路となるように任意の 2 地点間の経路を求める。その次に 2 地点間の距離をコストとし、どの順番で訪れるのかの組み合わせを様々な手法で求める。一般的な手法として、求めた近傍解が暫定解よりも良ければ解を更新する局所探索法や、悪い解でも受理する場合を考慮する焼きなまし法、生物の遺伝のメカニズムを応用した遺伝的アルゴリズム選択等が考えられている [5] [6]。道なりに関する研究としては、右左折の回数が少なくなるように、右左折に対して重みを付けて経路探索をしたり、ストロークと呼ばれる道なりの道路で、ストロークネットワークを構築し、その上で道なり優先経路探索手法を用いて、経路を求めるシステムを開発したものもある [1]。

また二酸化炭素の排出量をコストに加味した配送計画問題や、交通事故の発生による、渋滞を避ける経路探索システムも考えられている。

オンラインショッピングでは時間指定を都合に合わせて設定することはできるが、今回は配送時間の指定を考慮しない。また積載については 1 台ですべての顧客を巡ることができるものとする。また交通渋滞については、他の車による自然渋滞を考え、道路の工事や事故といった要素は考慮しないものとする。

2. 提案システムの概要

本研究で提案するシステムは次のような特徴を持つ。

特徴 1　道なり優先の巡回経路生成システム

巡回経路問題を解く時と同様に、ストロークネットワークを使って、道なり優先の経路探索プログラムを使って 2 地点間の経路を求め、設定された経由地を辿る順番を決める。この時使用するプログラムは文献 [1] で作成された既存のものを使用する。以下、このプログラムをストロークプログラムと呼ぶ。

特徴 2　巡回経路を SUMO で扱えるように変換

SUMO 向けに道路データを改良した際、道路 ID やノード ID が全く異なっている、道路や交差点が存在しないといった理由により、求めた巡回経路は SUMO へそのまま反映することができない。このため、SUMO で扱えるように求めた経路を変換する機能を実現する。

3. 道なり優先の巡回経路生成システム

道なり優先の巡回経路生成のために、ストロークプログラム [1] を使用する。このストロークプログラムはストロークネットワークと呼ばれる道路データを使って経路を求めている。まずは使用している道路ネットワークについて説明する。

3.1 道路データ

道路データは、ノード、ポイント、アークの３つのデータがある。ポイントは道路上の点を示し、ノードは、道路ネットワークの交差点、アークは始点ノードから終点ノードに至る道路で、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り、始点ノードと終点ノードの組をエッジと呼ぶ。

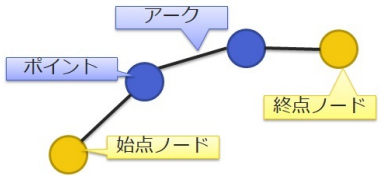

図 1　道路データの３つの要素

この道路データを、表で示すようなリンクテーブルで表現する。表にある道路クラスは高速道路や国道といった道路の種類を識別するものである。

表 1　リンクテーブル

カラム名	データ型	説明
ID	int	このテーブルでリンクを一意に識別するリンク ID
osm_id	String	OSM 上でリンクを一意に識別するリンク ID
clazz	int	道路のクラス
source	int	リンクの始点ノードの ID
target	int	リンクの終点ノードの ID
osm_source_id	String	OSM 上でノードを一意に識別する始点ノードの ID
osm_target_id	String	OSM 上でノードを一意に識別する終点ノードの 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	ノード間の道路の長さ (単位は km)
arc	LineString	道路の形状を表す

表 2　道路クラスについて

clazz	説明
11	高速道路
12	高速道路の出入り口
13	国道
14	国道への連絡路
15	主要地方道
16	主要地方道への連絡路
21	一般地方道
31	2 車線以上の一般道
41	2 車線未満の一般道

3.2 ストローク

背景でも述べたように、最短経路となるような巡回経路では右左折の回数が増えてしまい、認知的なコストの増加や、事故の遭遇確率、ストップ・アンド・ゴーの回数が増加するといった問題が発生し運転手への負担や環境への影響が大きくなる。

提案手法では、この影響を少なくするため道なり優先の巡回経路を求める。道なり優先とは、このストロークと呼ばれる道路を優先的に選ぶことだが、具体的には、ストローク数が最小となるような経路のこという。しかし道なりかどうかは個人の感覚によって違いがあり、ストロークの定義の仕方によって変わってくる。本研究では、以下のルールに従って、連続したリンクを同一のストロークと判定する [1]。

（1） 注目するリンク L の端点に接続するリンクがリンク L_i の 1 つのみの場合、リンク L_i とリンク L は同じストロークに属する。

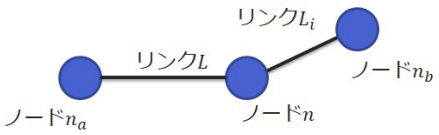

図 2　ストロークルール 1

（2） 注目するリンク L の端点に接続するリンクが複数の場合、注目するリンクと同じ道路のクラスで、隣接するアークの成す角が 45 度以下であり、かつ最小となるアークを持つリンク L_i をリンク L の系列の対象とする。

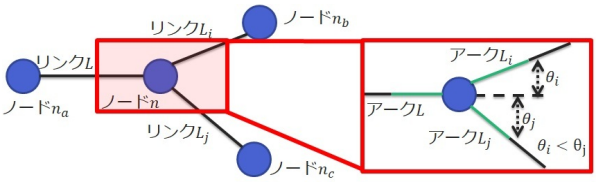

図 3　ストロークルール 2

（3） 大きな交差点内において交差点内リンクが存在する場合、交差点内リンクを挟むリンクを全て接続するものとみなし、条件 2 を満たすリンクをリンク L の系列の対象とする。

このルールにしたがって作成されたストロークネットワークを使って道なり優先の巡回経路を求める。

3.3 巡回経路生成システム

提案するシステムは次のような特徴を持つ。

- ストロークプログラムによって求められた任意の 2 地点間の経路をデータベースに保存
- データベースに保存することで、同じ 2 地点間の経路の再探索の必要がない
- 様々な経由地の組み合わせによる巡回経路の作成が容易に実行可能

まず、一般的な巡回経路を求める時と同様に、２点間の経路をストロークプログラムを用いて求める。求めた 2 地点間の経路はデータベースに保存する。保存するデータ構造は次の表の

表 3　2 地点間の道なり優先の経路のデータ構造

カラム名	データ型	説明
start	int	スタート地点のノード ID
goal	int	ゴール地点のノード ID
length	double	経路の距離
stroke_count	int	経路上のストローク数
route_node	int[]	ノード ID の配列で表した経路

とおり。

巡回経路作成にはシミュレーションソフトである SUMO が先ほどのデータベースから経路データを獲得し、スタート地点、経由地を設定し距離もしくはストローク数をコストとして、具体的な経路を求める。訪れる順番は次の処理を一定回数実行して求める。

手順 1　初期解設定

初期解として訪れる順番をランダムに決定する。

手順 2　2 つの訪れる順番を入れ替える

訪れる順番を 2 か所ランダム決め、それを入れ替える。

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、入れ替えた後の方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。悪い解を受理する確率は、処理する回数が増えるごとに小さくしていく。焼きなまし法は局所探索法に比べて、極所解に陥りにくいという性質がある [4]。

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し、このときの暫定解が最終的な巡回経路となる。一定回数満たしていない場合は、手順 2 へ戻る。

ルーティングからデータベースに保存し、巡回経路を求める一連の流れを図 4 に示す。

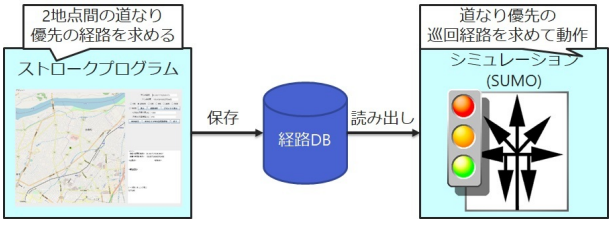

図 4　巡回経路生成の流れ

4.　シミュレーションへの巡回経路

4.1　SUMO について

本研究で用いるシミュレーションソフトは、Simulation of Urban Mobility、通称 SUMO と呼ばれるフリーのシミュレーションソフトを用いる。このシミュレーションソフトは、交通シミュレーションができるオープンソフトで、車の渋滞を発生させて交通の流れを観察したりバスやタクシー、それを利用する人を発生させて交通機関の利用の流れを再現できるシミュレーションソフトである。また OpenStreetMap から地図データをエクスポートし、それを加工することで OSM の地図データを使ったシミュレーションも行うことができる。関連研究として、複数の車両が混在するデマンド型交通システムにおいて個々の車両制御が可能なことからシミュレーションとして用いられている [2] [3]。

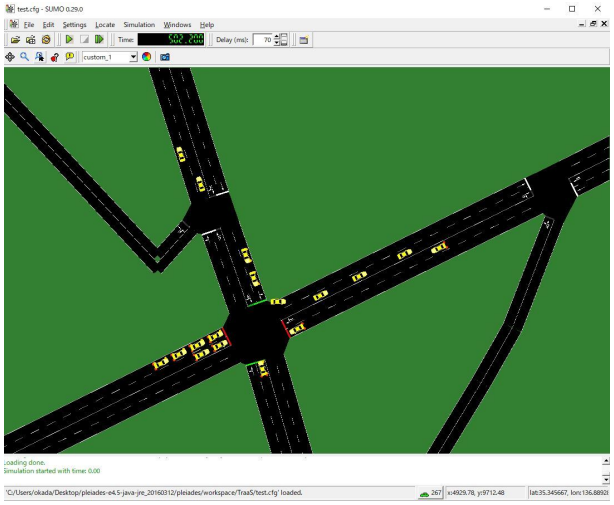

図 5　SUMO の画面

4.2　問題点とアプローチ

前章のストロークプログラムにより求められた巡回経路をシミュレーションに反映するためには、求めた経路を道路 ID の配列に直す必要がある。しかし、求めた経路をそのまま道路 ID に反映してもシミュレーションではエラーにより動かなかった。この原因は、ストロークプログラムで使用しているストロークネットワークの道路データとシミュレーションで用いる道路データには差異が発生していることである。元は同じ OSM からエクスポートしてきたファイルだが、エクスポートした日時が違ったりデータの加工により内容に違いが発生してしまう。

そこでストロークプログラムにより求められた巡回経路をノード ID の配列で表現し、これを SUMO の道路データで使われているノード ID に直し、変換したノード ID の配列を元に道路 ID を求める。この時、道路 ID が求められないことがあるためこの部分に対してケースに応じて修正をする必要がある。修正後にできた道路 ID の配列を最終的なルートとしてシミュレーションに定義する。

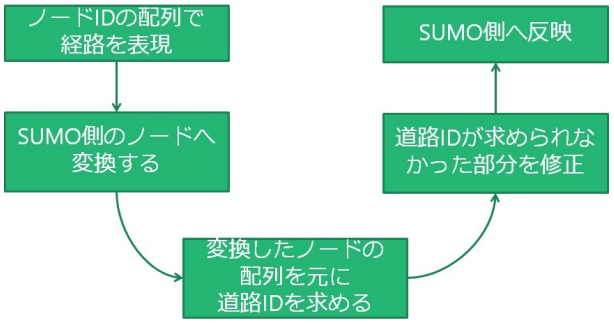

図 6　経路変換手順

4.3 ストロークプログラムで求めたノード ID の配列を SUMO 側の道路 ID へ変換

ストロークプログラムの道路データと SUMO の道路データで使用されているノード ID は異なっている。そこでストロークプログラム側のノード ID から SUMO 側のノード ID へ変換する必要がある。ここでは、ストロークプログラムで用いている道路データを元に作成されたデータベースを用いる。

表 4　ノード ID 変換用のデータベース

murase_node	sumo_node
ストロークプログラムで用いているノード ID	SUMO で用いているノード ID

このデータベースを使って、ストロークプログラム側のノード ID から SUMO 側のノード ID へ変換する。一例を図 7 に示す。

図 7　ストロークプログラム側のノード ID から SUMO 側のノード ID へ

SUMO 側のノード ID へ置き換えた後、SUMO 側のノード ID から道路 ID へ変換する。このとき、SUMO 側の道路データを元に作成されたデータベースを使って変換する。

表 5　道路 ID 変換用のデータベース

link_id	source_Id	target_Id
SUMO 側の道路 ID	始点のノード ID	終点のノード ID

ノード ID による配列から始点と終点のノード ID を決め、そこから SUMO 側の道路 ID を求める。道路 ID への変換の際、図 8 のように一部の経路では 2 つのノードを結ぶ道路が見つからないことがある (図 8 では「??????」となっている部分)。この部分については、次章で説明するケースに応じて修正をする必要がある。

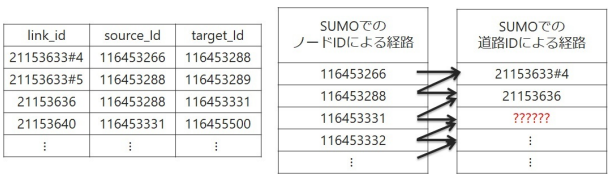

図 8　ノード ID から道路 ID への変換例

4.4 道路 ID の不明箇所の修正手法

道路 ID が不明となった部分について、3 つのケースが考えられた。そのうちの 2 つは SUMO 側でノードが消失している場合であり、残り 1 つは SUMO 側でノードが追加されている場合である。それぞれのケースについて順番に説明していく。

（1） SUMO 側でノードが消失しており、消失したノードの前後で結んでいる道路が存在している場合

OSM ファイルをシミュレーション向けのファイルに変換する場合、密接している 2 つのノードが 1 つに統合されることがある。そのため、SUMO 側の道路ネットワークを元に作成された道路 ID 変換用のデータベースに、消失したノードから出ている道路が存在せず修正をする必要がある。図 9 を例に考えてみる。

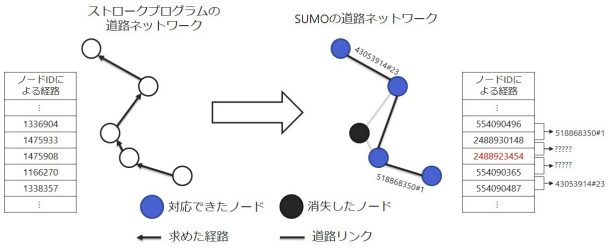

図 9　ケース 1 の事例

SUMO 側のノード ID へ変換した後の図 9 の右の部分に注目する。「2488923454」というノードが SUMO 側の道路データでは消失しており、該当の道路 ID を見つけることができない。しかし、消失したノードの前後の 2 つのノードで結んでいる道路が見つかることがある。この場合、ノード ID の配列から「2488923454」を消すことで、「2488930148」と「554090365」を結ぶ道路を見つけることができ、修正をすることができる。このようにノードが消えることで道路が見つけられなくても、消失したノードの前後のノード同士で結んでいる道路を見つけることができる場合がある。

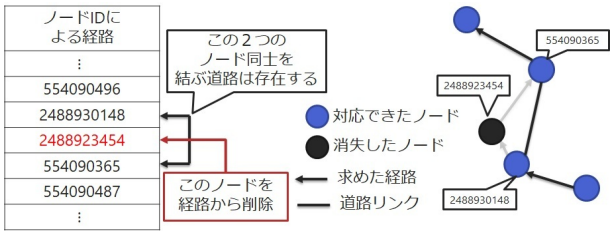

図 10　ケース 1

（2） SUMO 側でノードが消失しており、消失したノードの近くにある別のノードが道路と繋がっている場合

1 つ目のケースと同じように、消失したノードが近くの別のノードに統合された場合だが、その消失したノードの前後のノード同士で結んでいる道路が存在しないことがある。2 つ目のケースは消失したノードの前後のノード同士を結ぶ道路がなかった場合を考える。この場合、消失したノードの 1 つ前のノードから、消失したノードへのベクトルを求めて、そ

のベクトルの方向に最も近いノードを見つけ、消失したノードを見つけたノードに置き直すことで修正することができる。図 11 で、「1356394155」というノードが消失した場合を考えてみる。この時、一つ前の「567350535」から消失したノード「1356394155」へのベクトルを求める。次に、「567350535」から出ている道路のベクトルを計算する。消失したノードへのベクトルと求めた道路のベクトルを比較して、一番近い (なす角が一番小さい) ものを選ぶ。一番近い道路のベクトルを指した方向にあるノード「1356394008」を求めることができたのでノード ID の配列をこれに置き直す。これにより、経路を修正することができる。

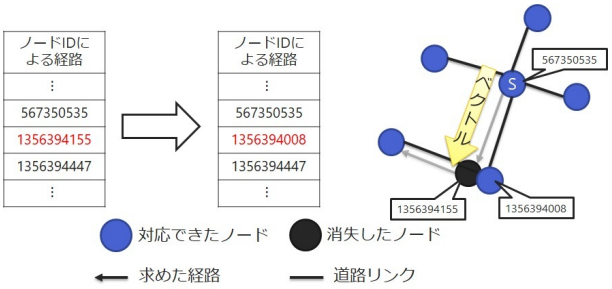

図 11　ケース 2

ベクトルで修正をする理由としては、例えば最短経路で経路修正をすると、別の道路が採択される可能性があるため、できるだけ元の経路を崩さないようにするためである。

（3） SUMO 側でノードが新しく追加された場合

OSM の道路データの更新により、ストロークプログラム側にはなく SUMO 側に新しくノードが追加されていることがある。この場合は、抜けている部分を SUMO 側の道路データを用いて最短経路による経路探索を実行して修正をする。一例を図 12 で示す。SUMO 側のノード ID の配列に変換した後の配列を見ると、2 つのノード「567350530」と「567350555」を結ぶ道路が存在せず、新しいノードにより複数の道路で表現されている。このとき始点を「567350530」、終点を「567350555」として最短経路によって経路を求めることにより、新しいノード ID の配列を求めることができる。求めた配列を、経路に追加することで修正することができる。最短経路による経路探索にした理由として、このケースによる経路修正は比較的に距離が短いことが多いと考えたためである。

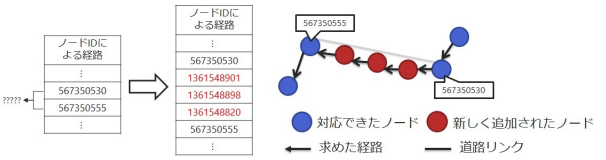

図 12　ケース 3

5.　評価実験

5.1　実験内容

ダイクストラ法による最短経路による巡回経路と距離をコストとしたストロークプログラムによる巡回経路の 2 つの経路を用いて環境への影響と操作の負担を測った。

使用する道路データは、江南市と扶桑町の一部を含む OSM データ (緯度 35.3349～35.3748, 経度 136.8632～136.9242 の範囲) とする。

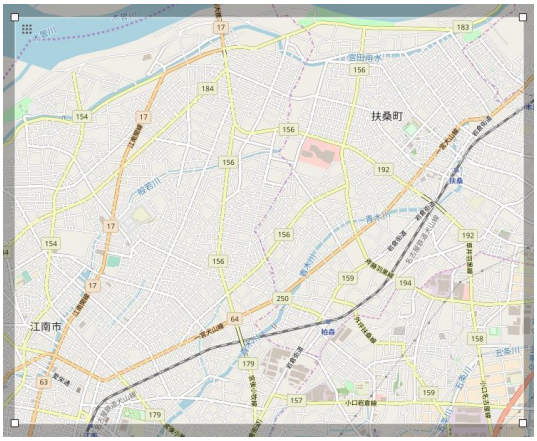

図 13　実験で使用する地域

また経由地は、商業施設への配送を想定し、幹線道路に経由地を設定した幹線パターンと、住宅への配送を想定し、住宅街の道路を経由地にした路地パターンの 2 つを用意した。

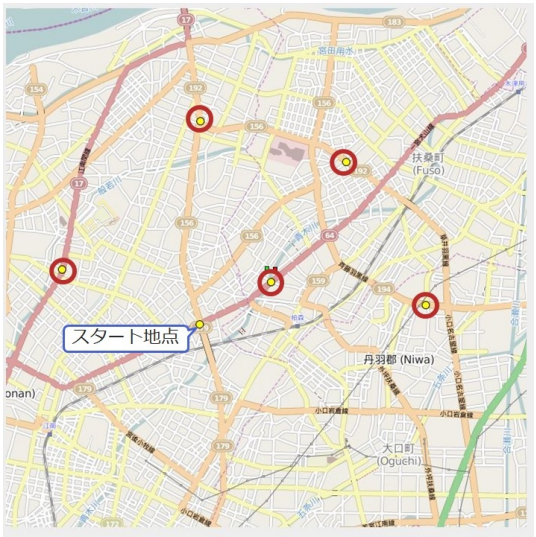

図 14　幹線パターンの経由地

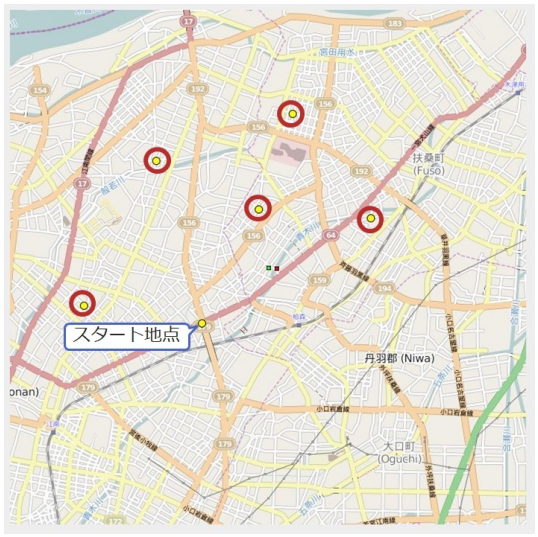

図 15　路地パターンの経由地

なおスタート地点は固定し経由地に到着した際、荷下ろし等のための一時停止はないものとして考える。csv ファイルにおけるパラメータは以下のように設定する。道路の速度制限は、個別の道路によって様々だが今回は OSM で定義されている道路の種類ごとに速度制限を設け、またその数値もなるべく日本の環境に合わせた。

表 6　実験で使用する標準設定

項目	値	項目	値
lefthand	TRUE	car_interval	1
highway.primary	50	car_accel	2.9
highway.residential	25	car_decel	7.5
highway.secondary	50	car_minGap	2.5
highway.service	10	car_length	6.5
highway.tertiary	40	car_width	2.1
highway.trunk	50	car_maxSpeed	180
highway.unclassified	25	car_vclass	delivery
start_time	0	car_emissionClass	HBEFA3/LDV
end_time	10000		

実験の結果として出力する内容は、基本的な移動距離、移動時間のほかに環境への影響と操作の負担である。具体的な項目は次の通り。

- 移動距離 (km)
- 移動時間 (秒)
- 停止時間 (秒)
- ブレーキランプ点灯 (回)
- ウィンカー点灯 (回)
- CO2 排出量 (kg)
- 消費燃料量 (L)

このほかに、燃費 (km/L)、ブレーキ頻度 (回/分) を計算して評価の指標にした。また各項目に対して、道なり優先が最短経路に比べてどのくらい良くなったのかを示す、改善率も計算した。改善率の計算方法は次の式 (1) の通り。道なり優先の値が最短経路よりも悪い場合、計算結果にマイナスを付けることにする。

$$\text{改善率} = \left| \left(\frac{\text{道なり優先の値}}{\text{最短経路の値}} - 1 \right) \times 100 \right| \tag{1}$$

5.2　幹線パターンの結果

幹線パターンの結果を、表 7 に示す。移動時間や停止時間は、移動距離に比例し増加がみられた。改善率の項目を見ると、ブレーキランプ点灯は 43%、ウィンカー点灯は 65%、ブレーキの頻度は 50%改善されている。これらの項目は運転手の操作負担になるため、最短経路時よりも負担を軽減することができた。しかし、環境の影響の指標となる CO2 排出量の改善率は-11%、燃料消費量は-14%と悪化が見られた。幹線パターンにおける、道なり優先の道路では操作負担と事故のリスクを軽減することができたが、環境にやさしい巡回経路にはできなかった。道なり優先での遠回りによる移動距離の増加が大きく、それに比例して CO2 排出量、燃料消費量が大きくなったと考えられる。

表 7　道路の速度制限変更の結果 (幹線パターン)

項目	道なり優先	最短経路	改善率
移動距離 (km)	13.9	11.1	-25.3%
移動時間 (秒)	1436.7	1260.4	-14.0%
停止時間 (秒)	299.7	126.3	-137.3%
ブレーキランプ点灯 (回)	22	39	43.6%
ウィンカー点灯 (回)	8	23	65.2%
ブレーキ頻度 (回/分)	0.92	1.86	50.5%
CO2 排出量 (kg)	3.12	2.72	-11.5%
燃料消費量 (L)	1.33	1.16	-14.6%
燃費 (km/L)	10.43	9.55	9.21%

5.3　路地パターンの結果

路地パターンの結果を、表 8 に示す。幹線パターンと違い、道なり優先時の移動時間は最短経路よりも約 2 分縮めることができた。改善率の項目を見ると、ブレーキランプ点灯は 37%、ウィンカー点灯は 56%、またブレーキの頻度も 32%改善された。環境の影響の指標となる CO2 排出量や燃料消費量は、改善率が 1%と、わずかに良い結果となり、そこまで差が大きいものではなかった。路地パターンにおける、道なり優先の道路では安全に運転することができ、環境への影響も最短経路とほぼ変わらず、早く巡回経路を辿ることができた。

表 8　道路の速度制限変更の結果 (路地パターン)

項目	道なり優先	最短経路	改善率
移動距離 (km)	11.6	10.1	-14.8%
移動時間 (秒)	1328.2	1435.2	-7.45%
停止時間 (秒)	54.3	122.5	55.7%
ブレーキランプ点灯 (回)	43	69	37.7%
ウィンカー点灯 (回)	16	37	56.8%
CO2 排出量 (kg)	2.62	2.66	1.50%
燃料消費量 (L)	1.12	1.14	1.75%
燃費 (km/L)	10.34	8.89	16.3%
ブレーキ頻度 (回/分)	1.94	2.88	32.6%

6. おわりに

ストロークプログラムを用いた簡単な巡回経路作成のシステムを作ることができ、２つの道路データの差異を埋めて、巡回経路をシミュレーションに反映することができた。また、住宅地への配送を想定した巡回経路の場合、最短距離よりも所要時間を短くし、また環境にやさしく、より交通事故のリスクが少ない運転を実現することができた。今後の課題として、今回提示した経路変換時の修正方法が別の道路データでも対応できるか、また道路データに含まれている道路の種類の割合によって、実験結果に変化は現れるのかを実験する必要がある。さらに、配送する車が経由地で一時停止することや、急発進や停車中における排出量も考慮する必要がある。

本研究は JSPS 科研費 26330136，25700009、および、総務省 SCOPE の助成を受けたものです。

文　　献

[1] 新帯里奈, 山本大介, 高橋直久, 多層ストロークネットワークの構築と道なり優先経路探索への応用, DEIM Forum 2017, E4-4, 2017.

[2] 落合純一, 宮地将大, 野田五十樹, 複数タイプの車輌が混在するデマンド型交通サービスの利便性評価, The 29th Annual Conference of the Japanese Society for Articial Intelligence, pp. 1-3, 2015.

[3] 中島秀之, 野田五十樹, 松原仁, 平田圭二, 田柳恵美子, 白石陽, 佐野渉二, 小柴等, 金森亮, バスとタクシーを融合した新しい公共交通サービスの概念とシステムの実装, 土木学会論文集 D3(土木計画学), Vol. 71, No. 5 (土木計画学研究・論文集第 32 巻), pp. I_875-I_888, 2015.

[4] 山崎紘揮, 宗久知男, 宗久保子, 巡回セールスマン問題への確率的状態選択法: 集団的焼きなまし法, 電子情報通信学会技術研究報告, 知能と知識処理, Vol. 106, No. 617, pp. 33-35, 2007.

[5] 青木健, 井上裕介, 海野紘旭, 高橋伸典. 巡回セールスマン問題の解法
http://www.info.kanagawa- u.ac.jp/˜hatori/study/yousi2004-5.htm,

[6] 原健太, 古川武志, 塚原荘一, 狩野均, 多目的遺伝的アルゴリズムによるカーナビゲーションのための経路探索, 情報処理学会研究報告, 数理モデル化と問題解決, Vol. 2004, No. 130, pp. 49-52, 2004.

[7] JA_Key_highway - OpenStreetMap Wiki,
https://wiki.openstreetmap.org/wiki/JA:Key:highway, 2018 年 1 月 17 日アクセス