地図データを使った配達経路の最適化アルゴリズム

倉田　是

1．はじめに

以前から、生活協同組合は日常の食生活を支える冷凍食品や生鮮食料品をはじめとする商品を組合員に配達して販売してきた。数名の組合員が班組織を作り、この班を一括して商品を配達するという効率の良いものであった。

しかしながら、特に都市及び都市近郊においては、共稼ぎ世帯の増加傾向が産業のリストラによって増加傾向にあり、隣近所のつきあいが希薄になる傾向も多く、班組織でなく戸別に配達する希望者が多くなってきた。この傾向は、1年間で約2倍に達するほど顕著である。この配達を戸別配達あるいは個別配達と呼ぶが、以下個配と簡略化して呼ぶこととする。

また、高齢化社会では、買い物に出られないことが多くなるので、ますます個配の増加傾向に拍車が掛かることは明らかである。自治体が業者に委託して、給食や日用品を届けることも多くなってきた。

一方、経費は掛かる。たとえば、直接の人件費を時間給1,000円としても、5箇所配達するだけで、1箇所200円となる。しかし、個配などの配達の場合には、利用者から配達費用を確実に取ることができない場合が多い。現在は0～400円／回を商品とは別に徴収している場合が多い。なかには、商品価格に上乗せしているものも見受けられる。

人件費は配達に関わる労働時間に関係するから、配達時間をもっとも短くする配達経路を選ぶことが重要である。また、これで浮いた有効時間を他の目的に使うことができ、労務管理上も有意義と考えられる。

本報告では、個配を前提として、配達コストを低下させるために、配達時間をもっとも短くする配達経路を探索する問題をコンピュータで解くアルゴリズムを考え、実際の地図データを利用することを含めて、検証してみた。

2．本研究の特徴と従来の研究

最短経路の探索問題は下記に述べるいくつかの種類について多くの報告がなされている。

（1） 巡回セールスマン問題[1)]

（2） 配送計画問題[2)]

（3） カーナビゲータ[3)]

本報告の最短経路問題と比較しての相違点を述べる。本報告の配達箇所は主として住宅街にあり、道路は広いものではない。このような道路ではＵターンをして方向転換を行う余地は少なく、無理である。前後にある交差点を使って方向転換も可能ではあるが、後方の安全確認を必要とするので、事故による有形無形の損失を想定するとＵターンを考えない方が良い。(1)の巡回セールスマン問題は、すべての都市を最短経路で巡回する問題である。本報告の複数の配達箇所を巡る点では同一である。巡回セールスマン問題は都市間の最短距離が与えられて、同一箇所を通らずにその最短距離を求める方法である。Ｕターン禁止条件は、この最短距離を利用できるとは限らない。

（2）の配送計画問題は、工場から部品を複数の組み立て工場に配送する問題や、コンビニエンスストアに商品を配送する問題である。これも一般的には、各配送箇所には荷受けの場所があり、荷下ろしをした後に方向転換をすることもできる。本報告の問題は、配達箇所を通る道路の進入方向を考慮しなければならないので、地図上での最短距離の問題とは異なる。

巡回セールスマン問題はアルゴリズムや計算速度の問題であり、配送計画問題は実際の地図を利用し、配送荷物の重量やトラックの積載量を問題とする実用的な計画と相違はあるが、道路の進入方向は考慮しなくても良いので、どちらも最短距離のみを考慮すればよい点では同じである。

（3）のカーナビゲータは、実際の道路を使し、進入方向を考慮する点では本報告と同じであるが、複数の箇所を巡る問題ではない。

3．アルゴリズムおよびプログラム

3．1 MapInfo 地図システムからの道路データの抽出プログラム

MapInfo は[4)]、各種地図データを持ったデータベースである。本報告で使用するのは、道路情報であって、交差点、道路区間距離、道路の種類などである。MapInfo のデータは表形式である。このうち、使用する表は、表１に示す「道路点」と表２に示す「区間」である。このデータは各表に示すような属性を持っている。この表は、MapInfo の地図

表１　MapInfoの道路点データ

都道府県コード CHAR(2)	道路点ID INTEGER	道路点種別 CHAR(2)	隣接都道府県コード CHAR(2)	隣接道路点ID INTEGER	交差点名称 CHAR(2)	接続道路数 SMALLINT	区間(1) INTEGER	区間(2) INTEGER	区間(3) INTEGER	区間(4) INTEGER	区間(5) INTEGER	区間(6) INTEGER	区間(7) INTEGER	区間(8) INTEGER	追加道路点ID INTEGER	フェリー接続航路総数 SMALLINT
13	49838	1		0		3	77390	77374	77389	0	0	0	0	0	0	0
13	49839	1		0		4	77389	77393	77391	77392	0	0	0	0	0	0
13	49845	1		0		5	77411	77409	77408	77407	77410	0	0	0	0	0
13	49870	1		0		4	77483	77482	77480	77481	0	0	0	0	0	0

表２　MapInfoの区間データ

区間ID INTEGER	区間種別 CHAR(2)	都道府県コード CHAR(2)	市区町村コード CHAR(3)	区間長 INTEGER	幅員区分コード CHAR(1)	車線数コード CHAR(1)	12時間交通量 SMALLINT	ピーク時速度 SMALLINT	接続点ID 1 INTEGER	接続点ID 2 INTEGER	路線番号 SMALLINT	有料道路コード CHAR(1)	管理者コード CHAR(1)	主･従道路区分コード CHAR(1)	重要路線総数 SMALLINT	車道幅員 SMALLINT	最小車道部幅員 SMALLINT	中央帯幅員 SMALLINT	中央帯設置延長 INTEGER
77389	6	13	204	87	2	2	102	96	49838	49839	134	2	5	1	0	65	65	0	0
77390	6	13	204	43	2	2	102	96	49838	52043	134	2	5	1	0	65	65	0	0
77391	6	13	204	84	2	2	102	96	49839	51989	134	2	5	1	0	65	65	0	0
77392	6	13	204	100	2	2	100	207	49839	52009	121	2	5	1	0	70	70	0	0
77409	6	13	204	68	2	2	102	96	49845	51853	134	2	5	1	0	65	65	0	0

をウインドウ画面に表示して、これらの表を取り出す領域を限定した後に得ることができる。

図１は本報告で、アルゴリズムの検証用に使用したMapInfoの地図である。これは東京近郊のＭ市の一部である。図の中の破線で囲った部分が、対象となる領域である。この対象領域を選択し、表１と表２のデータが得られる。なお、この図には道路点（交差点）番号と区間距離（m）が表示してある。道路点のデータは日本全国でユニークな番号を持っている。また道路点と道路点の間は区間であり、これもユニークな番号を持っている。道路点データには、その道路点から分岐する区間のデータがある。区間のデータには、区間両端の道路点のデータが隣接点ID1と隣接点ID2で表されている。

本報告では各交差点で、下記の条件で通過時間が変わることを考慮することにしている。

（１）　信号の有無による交差点の通過時間の差違

（２）　道路の優先度による走行速度の差違

また、将来は直進・左折・右折により差違を付けること、また通過時間帯による走行速

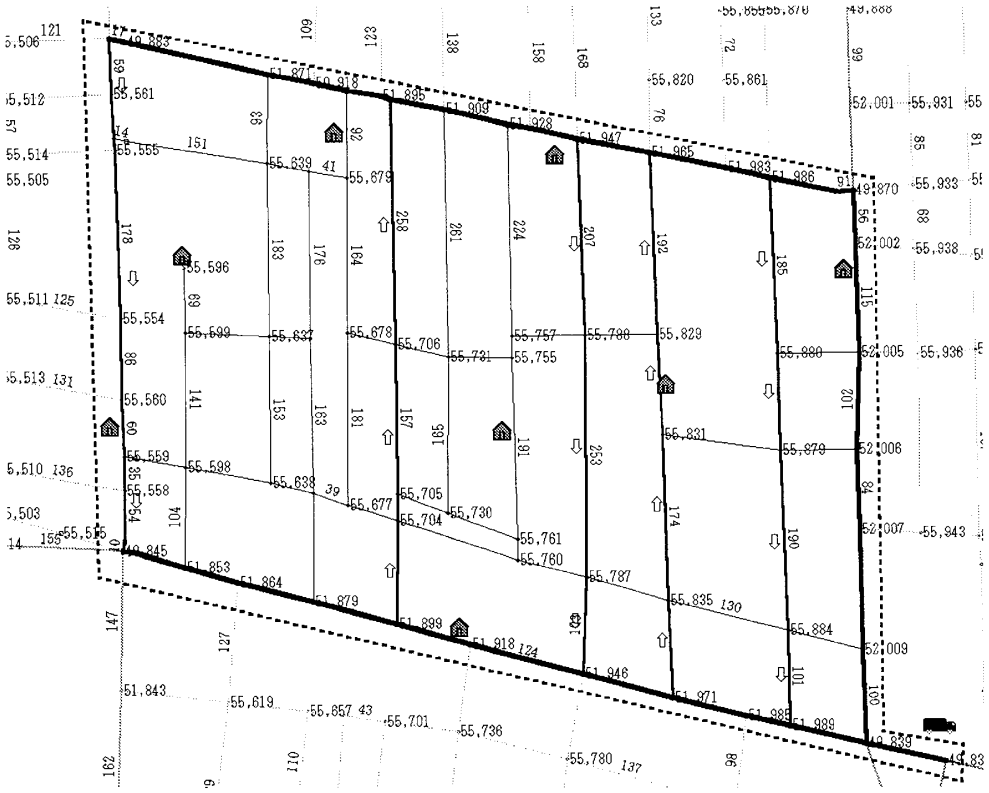

図１　本報告のアルゴリズム検証用に用いたMapInfoの地図

度や交差点通過時間の変化を考慮したいので、表３のデータ属性を与えることにした。ここで、主・従は道路の優先度を表している。なお、閑散時間や渋滞時間などの通過時間帯については、この表の末尾に記号を付け加える予定であるのが、本報告では使用しなかったので、省いた。

本報告の配達経路循環の最短時間を求めるための地図の必要データは表３に表すものである。表にある出発交差点と進路変更交差点、進路変更先交差点１，２，３の関係と

表３　アルゴリズムを実現するための交差点データ

出発交差点	区間距離(m)	区間主・従別	進路変更交差点	進路変更交差点の信号機の有無	進路変更先交差点１	進路変更先交差点２	進路変更先交差点３	進路変更先交差点と変更先交差点１の区間主・従区別	進路変更交差点と変更先交差点２の区間主・従区別	進路変更交差点と変更先交差点３の区間主・従区別	直進・左折・右折の区別	直進・左折・右折の区別	直進・左折・右折の区別
49838	43	13	52043	0	0	0	0	0	0	0	0	0	0
49838	87	13	49839	1	51989	52009	0	2	2	0	0	0	0
49839	87	13	49838	0	52043	0	0	2	0	0	0	0	0
49839	84	13	51989	0	51985	55884	0	2	1	0	0	0	0
49839	100	13	52009	0	55884	52007	0	0	2	0	0	0	0

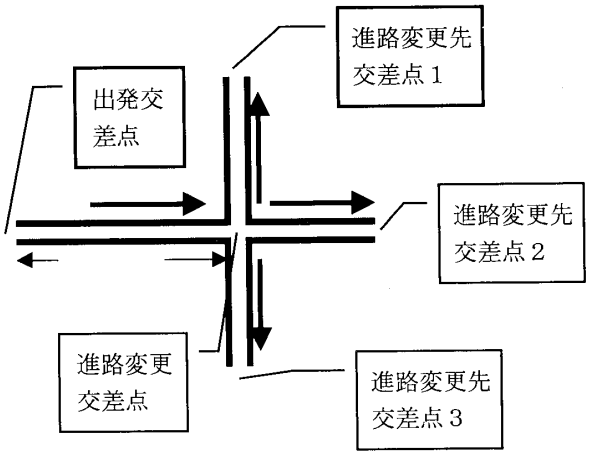

図２　各交差点の関係と区間距離
なお，進路変更先交差点の１，２，３
は，左折・直進・右折には対応していない

その他の属性をわかりやすく説明するために図示すると図２となる。道路の優先度は主・従の欄で道路区間の優先度を表しているが、図では省いた。

MapInfo の２種類のデータから、表３のデータを導き出すために Map Basic を利用したプログラムを作成した。Map Basic はやや特殊な Basic であり、配列は１次元のみである。ただし、Type 文が使えるので、実質的に２次元配列までは利用できる。表３のデータ構造全体は Type 文の１次元配列で各変数を作成し、進路変更先交差点や主・従の別、直進・左折・右折（今回は参考に設けた）は Type 文内部の変数の１次元配列を利用した。なおこのたび使用した地域には、４差路の交差点のみであったので、配列要素数は３であるが、５差路以上の場合にも容易に増やすことができる。

Map Basic では、16ビットの整数は Smallint を使用しているなどの一般の Basic との変数の定義の上でも相違がある。

MapInfo の地図データには、信号機のある交差点や一方通行路の記述がない。今回は使用しなかったが、直進・左折・右折の識別情報もない。
このために、信号機のある箇所を MapInfo 地図のウインドウ表示上の交差点を領域選択し、表３の交差点の属性欄に１を入れるプログラムを作成した。

一方通行路のデータは、一方通行路を逆走する表３の出発交差点と進路変更交差点を削除するプログラムを作成した、MapInfo 地図のウインドウ表示上の一方通行路を選択領域として、この中で対になる２箇所の交差点を入力して除去するプログラムである。

なお、MapInfo の道路の優先度を表す主従の属性は、２車線道路を２とし、一方通行路を１とし、住宅地内の道路を０とした。これも、同様の手段で変更できるプログラムを作成した。

Map Basic は、原則としてコンパイル後に実行するのであるから、実行速度はＣ言語

とそれほど変わらないとは思える。しかしながら、プログラム作成の編集機能が悪く、よく使われているVisual Basicのようにプログラム作成中の文法エラーチェック機能もない。さらにインタプリターではないので実行時のエラーも把握しにくい。一方、交差点（道路点）の入力などには、MapInfoの領域選択と連動し、さらにダイアログボックスが用意されているので、ユーザーインターフェースは程々に良い面もある。しかしながら、次に述べる多次元配列を多用するプログラムを作るには適さないので、表3を作るまでに利用を留めた。

3．2 最短経路を求めるアルゴリズム

図3は、例題として利用したデポと8箇所の配達箇所を示した図である。ここで、一方通行路や道路の優先度は矢印と線幅で示している。この領域には道路点が71箇所(MapInfoの特徴上、行政区域の異なる道路区間でも道路点としているので実際の交差点数は60であるが、簡単のためすべてを交差点とした）ある。このすべての交差点をデポ及び配達箇所を結ぶ経路を探索するのは、すべての交差点を十字路とすると、3^{71}のオーダーに近く、計算量が多くなり困難である。

デポと各配達箇所を根(root)とした、デポ・配達箇所前面道路の車両の進行2方向ごとに、進出及び進入経路の時間を木構造の連結リストを求めた。また、この木構造の枝の

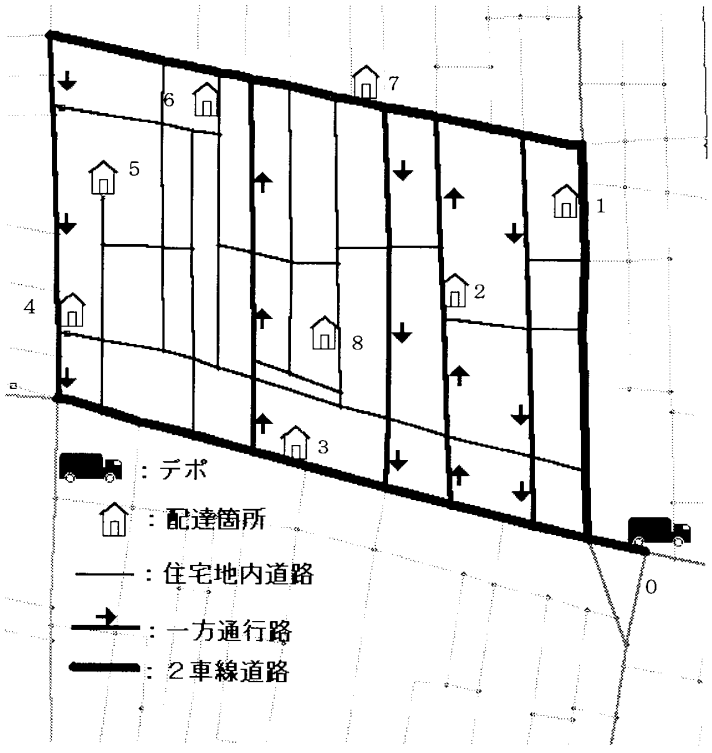

図３　デポと配達箇所

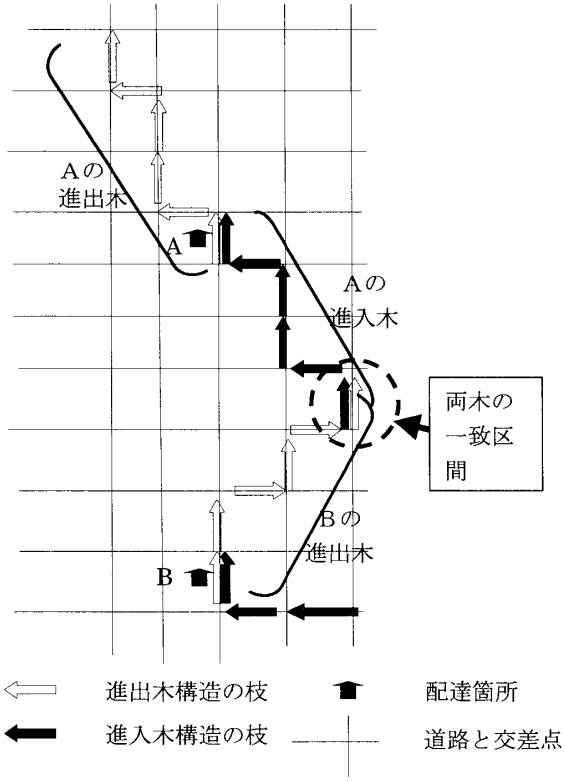

図４　進出木と進入木の木構造による経路の接続

長さをある長さに制限した。このことによっても、複数のデポまたは各配達箇所間の木構造の枝が絡み合い、経路が連結するの可能性があるので、デポ及び配達箇所を結ぶ経路求めることができる。この例題では、デポと各配達箇所間の道路区間の最大数は10であるから、進出木構造、進入木構造とも枝の長さを６まで選べば図４のようにＢ配達箇所からの進出木とＡ配達箇所の進入木とが必ず一致する区間が存在することになる。なお、ルートから数えると、この木構造の枝(branch)の長さはこれよりも１つ多い７である。このことにより、各木構造はこれもすべての交差点を十字路として3^7のオーダーとなり、極めて少ない計算量となる。

この木構造を作るためのアルゴリズムを図５と図６で示した。デポまたは配達箇所が面する道路区間の進行元の交差点と進行先の交差点と区間距離を、車両の進行方向２方向に関係づけた図７で示す。

最初は、この進行元と進行先の交差点と表３のデータの出発交差点と進路変更交差点の一致するデータの行を抽出する。この進路変更交差点と進路変更先交差点と一致する出発交差点と進路変更交差点のデータを捜す。この時に、進入木の場合には、出発交差点と進路変更交差点を入れ替えて捜す。つまり、進出木は進路変更先交差点が次の進路変更交差点となって進行し、進入木は進路変更交差点が出発交差点となって進入するように捜すのである。

この結果、進出木と進入木の表４，表５に示すものが得られる。これは連結リスト構造の一種である。

交差点数が71であるから、交差点を０から70までの番号に割り振り、71×71の行列を作り、表４，５のデポまたは配達箇所より（まで）の総走行経過時間、区間内の道路のみの走行経過時間、進出木あるいは進入木連結リスト構造の番地の３種類を別の行列の各要素に張り付ける。これによりデポ・配達箇所計９組みと車両方向２方向の計18×３組の行列ができる。

これを、出発木構造交差点行列と進入木構造交差点行列と名付けた。この配達箇所より（まで）の総走行経過時間の行列で要素が一致したものが、図４で示した経路の接続

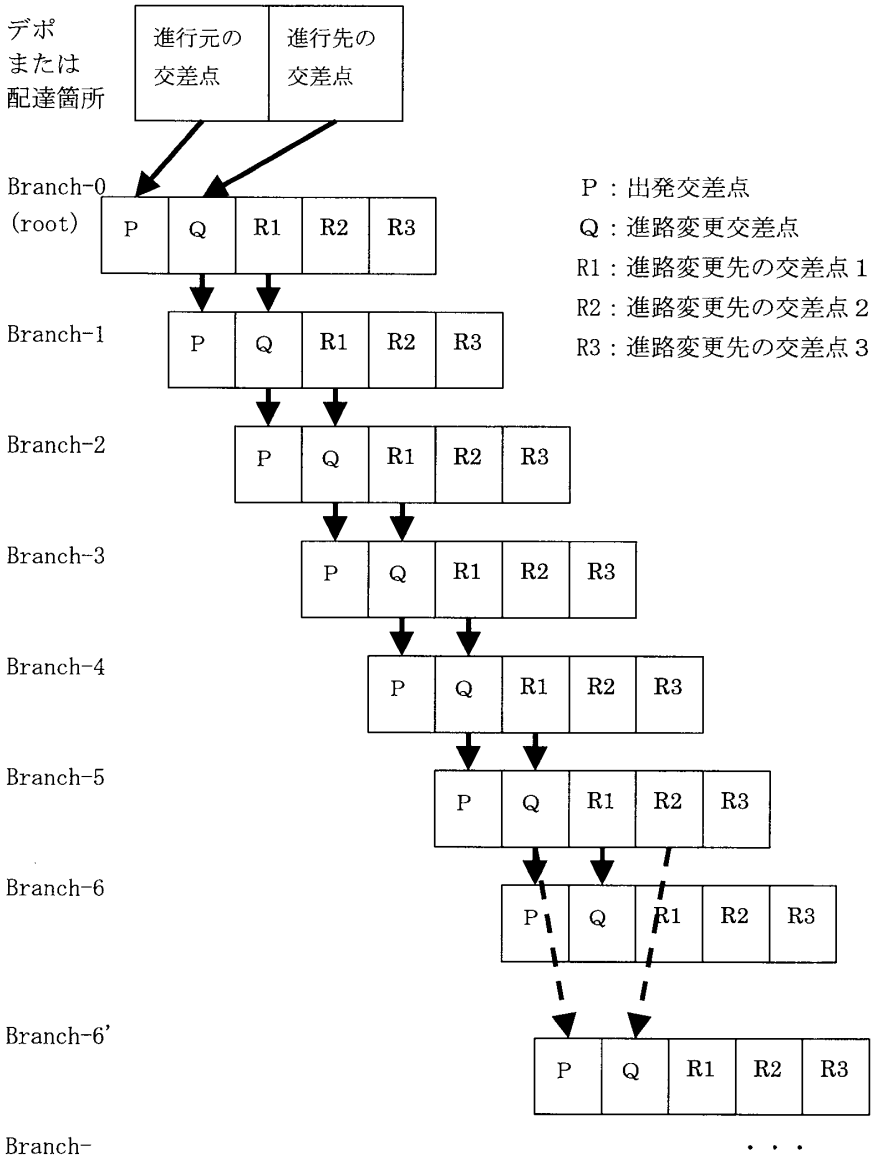

図５　進出木構造作成のアルゴリズム

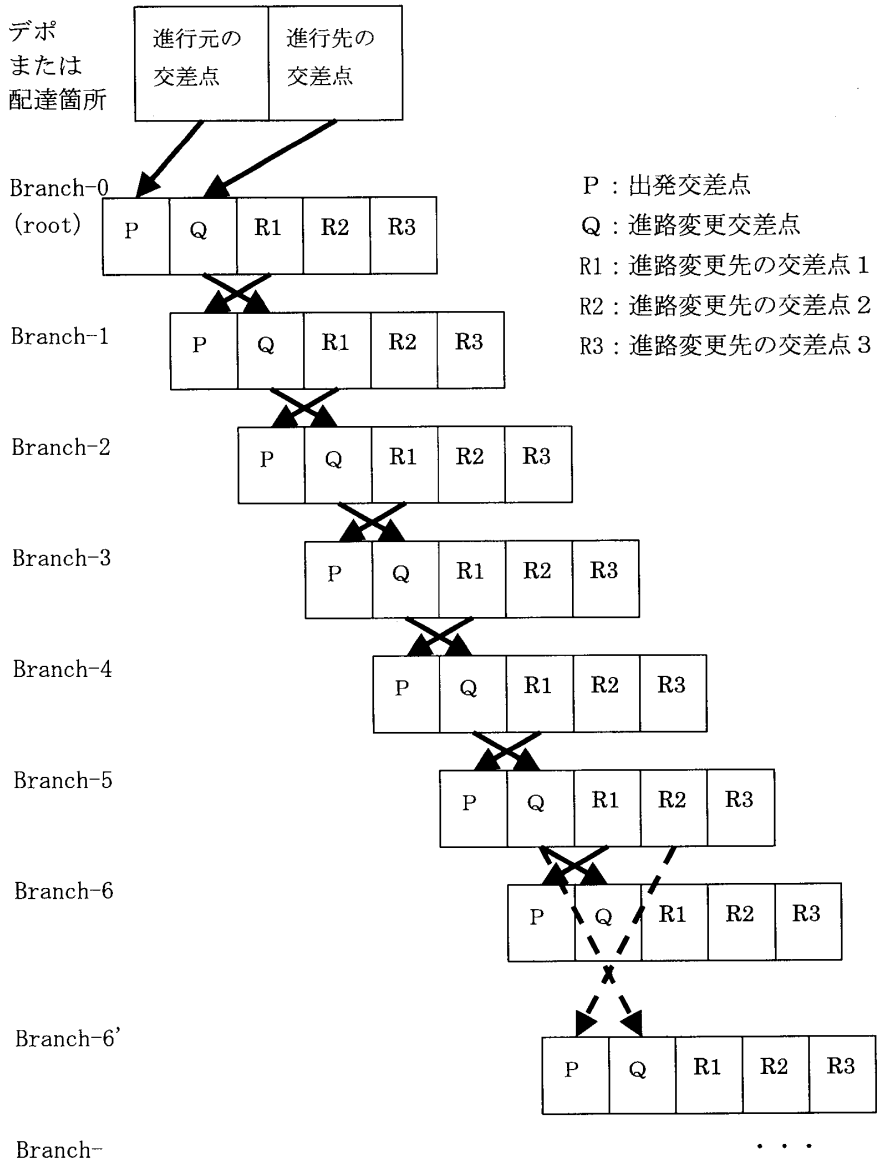

図６　進入木構造作成のアルゴリズム

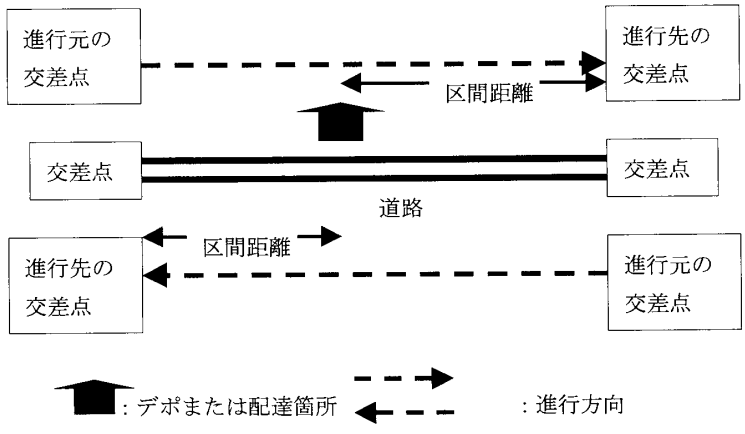

図７　デポまたは配達箇所の交差点と区間距離の関係

表４　進出木構造の一部

出発交差点	進路変更交差点	デポまたは配達箇所よりの総走行経過時間	区間内の道路のみの走行経過時間	枝(BRANCH)番号
49838	49839	13	13	0
49839	51989	25	12	1
51989	51985	52	7	2
51985	51971	64	12	3
51971	51946	78	14	4
51946	51918	96	18	5
51918	51899	128	12	6

表５　進入木構造の一部

出発交差点	進路変更交差点	デポまたは配達箇所よりの総走行経過時間	区間内の道路のみの走行経過時間	枝(BRANCH)番号
49839	49838	26	26	0
51989	49839	71	25	1
51985	51989	85	14	2
51971	51985	129	24	3
51946	51971	158	29	4
51918	51946	195	37	5
51899	51918	219	24	6

になる。すなわち、この出発木構造交差点行列を

$$\mathrm{F}_0 \cdots \mathrm{F}_8 = \begin{bmatrix} f_{0,0} & & f_{71,0} \\ & & \\ f_{0,71} & & f_{71,71} \end{bmatrix} \tag{1}$$

とし、進入木構造交差点行列

$$\mathrm{B}_0 \cdots \mathrm{B}_8 = \begin{bmatrix} b_{0,0} & & b_{71,0} \\ & & \\ b_{0,71} & & b_{71,71} \end{bmatrix} \tag{2}$$

とすれば、

$$\mathrm{T} = \mathrm{t}_f + \mathrm{t}_b - \mathrm{t}_{i,j} \tag{3}$$

でデポまたは配達箇所相互間の走行時間Tが求められる。ただし

$$f_{\mathrm{P}i,j} \neq 0 \quad \text{かつ} \quad b_{\mathrm{A}i,j} \neq 0 \tag{4}$$

である。ここで＝t_fは出発木構造交差点行列のデポまたは配達箇所よりの総走行経過時間の要素で、t_bは進入木構造交差点行列デポまたは配達箇所までの総走行経過時間の要素で、$t_{i,j}$は$_{i,j}$の区間内の道路のみの走行時間である。この$t_{i,j}$を減算することにより、$t_f + t_b$による共通する道路区間の走行時間の2重加算を避けることができる。この結果は、一致箇所が多いので、多数のデータができる。

次はこのデポ・配達箇所間それぞれの最短時間を求める。この結果、一種の有向グラフができあがる。この有向グラフを使って、たとえばDijkstra法などでデポを出発点とする巡回配達の最短ルートを見つけだす方法もあると思われる。[5] しかい、デポ・配達箇所前の道路の進行方向を考える必要があるので、このたびはこれらの手法を使わずに、全ての経路をたどる方法で計算した。この結果には、それぞれ最短経路のデポ・配達箇所間の進出木および進入木連結リスト構造の番地のデータがあるので、表4及び表5の枝番号を逆にたどることにより、通過する交差点を全て表示することができる。

4．実行結果

前章で述べたアルゴリズムを実現するプログラムは、Map Basicで6個のプログラム、C言語（正式にはVisual C＋＋言語）で7個のプログラムを作成した。なお、Map Basicは今後とも独立したプログラムとなるが、C言語のプログラムは、デバッグのために7個のプログラムを使ったが、2個位のプログラムにまとめることも可能である。表6にこのプログラムと、その内容を一覧した。

道路の走行時間と交差点の通過時間が重要である．単なる距離でなく，道路の種類と交差点種類によって変わる通過時間を区間距離とそれぞれの道路の速度から、計算で求

表6　プログラム一覧

言語	プログラム番号	内　　容
Map Basic	0	道路点情報の抽出
	1	区間情報の抽出
	2	主・従道路の変更
	3	信号機のある交差点の設定
	4	一方通行路の設定
	5	交差点データの作成
C	0	デポ及び配達箇所の交差点と区間距離の設定
	1	進出木作成
	2	進入木作成
	3	進出木及び進入木行列の作成とデポ及び配達箇所間の走行時間等の結果出力
	4	デポと配達箇所の最短時間データの結果出力
	5	デポと配達箇所の最短ルート探索
	5	デポからデポへの配達箇所巡回交差点と経過時間の結果出力

める方法を採用した。

① ２車線道路　　40km/h

② 一方通行道路（１車線）　　30km/h

③ １車線道路　　20km/h

また、交差点の種類による交差点の通過時間については、おおよそ次のように定めた。

① ２車線の信号機のある交差点　0.2m

② １車線道路から２車線道路　0.6m

③ 住宅地内の道路の交差点　0.2m

④ １車線道路から一方通行路　0.2m

⑤ その他　0 m

直進・左折・右折を区別する予定で、交差点の実測データを使って統計的なデータを利用する予定であったが、今回間に合わなかったので省いた。

前に図３でデポと８箇所の配達箇所を示した。Ｕターンは今回も扱わないが、配達箇所５では、袋小路の終点にあるのでＵターンと同様に扱っている。この際には、配達箇所の進入方向と進出方向を逆転させるプログラム手法が必要である。このプログラム手法は、Ｃ言語のプログラム４で用いている。この手法を用いれば、配達箇所でＵターンさせることも可能である。

表７に最終結果を表した。また、図８にこの結果による経路を示した。また、総走行時間は18.7分であった。この地域は周囲約2.8kmの地域である。もし40km/hで周囲を

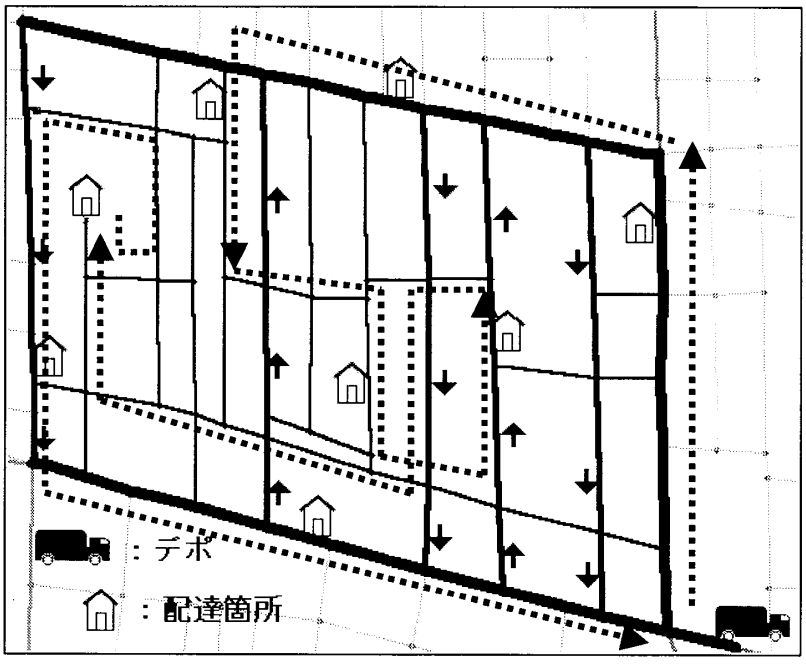

図８　最短走行時間の配達経路

表7　最終結果

交差点	デポまたは配達箇所	配達箇所間の走行時間(0.01m)		交差点	デポまたは配達箇所	配達箇所間の走行時間(0.01m)
49839	0			55760		0
52009				55704		0
52007				55677		0
52006				55660		0
52005	1	128		55638		0
52002				55598		0
49870				55599	5	416
51986				55596		0
51983				55599		0
51965				55637		0
51947	7	123		55639		0
51934				55564		0
51928				114914		0
51909				55555		0
51897				55554		0
51895				55560	4	307
51888	6	82		55559		0
55679				55558		0
55678				55553		0
55706				49845		0
55731				51853		0
55755	8	235		51864		0
55761				51879		0
55760				51899	3	193
55787				51918		0
55835				51946		0
55831	2	245		51971		0
55829				51985		0
55788				51989		0
55787				49839	0	141

巡ったとすると、4.2分で1周可能である。

5．考察

最終結果をみると、Uターンのところはもとより、同一交差点と道路区間を利用していることがわかる。また、地図の左側から右に移動し、再度左へ移動している。あるブロックをまとめて配達して、次のブロックに移るであろうという予想に反したものであった。このような結果は、おそらくいくつかの一方通行路があるために起こった問題であろうと思われる。都会の住宅地には、多数の一方通行路がある。このような場合には、最短時間で配達をする問題は、本報告のアルゴリズムを使わないと解けないように思える。

本報告では、進出木と進入木を用いたが、各木の枝の総数が1,300～1,400と多い。実

際の生活協同組合の配達は、一度に10～20台のトラックが各々15箇所位を配達する。大まかに、本報告の30倍である。したがって、枝の総数は40,000を軽く数えることが予想される。このような場合には、コンピュータのメモリの容量や処理の時間が掛かることになる。各木をみると、枝が入り交じり、同一道路区間を２重にも３重にも使っていることがみられる。特に枝数が多くなればなおさらである。庭木や果樹では、内側に回り込んだ枝は日当たりや風通しが悪くなるので剪定作業をする。同様に、本報告の木構造も剪定するアルゴリズムを考えることにより、コンピュータのメモリの節約と、処理時間の軽減ができると思われる。

また、このような場合には地図上の交差点数は2,000を数えることになると思われる。本報告では、71であったから、71×71の行列を作ったが、2,000以上の２次元配列はメモリの容量を大量に必要とする場合が予想される。この結果、仮想空間を利用することも多くなり、コンピュータの負担が増加する。行列を使わずに、進出木と進入木のみを使って同一結果を求めることも可能である。このようなアルゴリズムを開発する必要がある。

本報告は小規模であるから、１台の配達車で済むが、実際の場合には10～20台以上の配車計画も同時に考えなければならないし、実際の労務管理でも重要な問題である。このアルゴリズムは今後の問題である。

一種の有向グラフができるが、このグラフから最短ルートを見いだすアルゴリズムの考察ができていない。この最短ルートを求めるプログラムのみが、やや時間が掛かった。と言ってもElapse timeで１分位である。他のプログラムは全て、即時結果が得られる感じであった。なお、使用しているパソコンのCPUはPentium II 400MHzであり、主メモリ128MB(100Mhz)である。大規模な配達問題を扱う場合には、当然ながらCPUの性能向上を含めて、このアルゴリズムを考えなければならない。

結果を地図データ上にプロットする方策がまだできていない。MapInfoのデータには、主要交差点の名称はあるようであるが、特殊な場合である。配達の運転者の初期教育には、是非通過する交差点の位置をプロットするソフトウエアが必要である。

６．むすび

交差点間の区間長や区間の優先度、信号の有無、分岐先交差点などの属性を持たせたデータベースを、MapInfoの地図データベースから作り上げた。

このデータベースを使って、デポ・配達箇所を根とする有限な枝を持つ木構造の交差点連結リストを作成した。この木の根は、デポ・配達箇所の面している道路区間の両端の交差点である。またこの根それぞれからデポ・配達箇所を通って２方向に向かうことと、デポ・配達箇所から進出する方向と、進入する方向２種類の木構造である。つまり、

デポ・配達箇所からは、４本の木構造の交差点連結リストを作成した。

この各デポ・配達箇所の木構造の枝が絡み合うことを利用して、計算量が少なくて相互の経路を捜すことができた。また、この経路の中から最短の経路を求め、最後にデポから各配達箇所を巡ってデポに戻る最短の経路を、途中の経由する交差点を含めて求めることができた。

交差点の種類と道路の種類による、進行速度の変化と通過時間を読み込んである。

また、今後に残された問題を考察で指摘した。

参考文献

1）玉木久夫“巡回セールスマン問題の近似アルゴリズム：天才アローラによる20年ぶりの急進展”,情報処理,Vol.39,No.6,pp.566-573(1998)等

2）澤本潤、辻秀一、徳永寿郎、小泉寿男“分散協調による配送計画問題解決方式の提案とその実現法”,電気学会論文誌C，Vol.117，No.7,pp.896-906,1997等

3）加藤誠巳“経路探索問題とその応用”，情報処理,Vol.39,No.6,pp.552-557(1998)等

4）MapInfo はUSA に本社があり，日本総代理店は三井造船システム技研株式会社である.

5）増本忠幸，百合本茂、片山直登“ロジスティクスのＯＲ”槇書房,1998等