情報システム　12－1
（１９８６．９．１６）

首都圏電車網における巡回経路探索システム

加藤　誠巳　　高木　啓三郎

上智大学理工学部

近年、首都圏の電車網等の公共交通網は益々複雑化しつつあり、利用者はそれらの中から適切な経路を見い出し、効率的に移動することが極めて困難となってきている。一般に任意の複数個の地点を巡回する経路の適切な解を求める問題は巡回セールスマン問題として知られているが、この最適巡回経路探索を実際の大規模なネットワークに適用した例はあまり多くないように見受けられる。ここでは、従来より筆者らが手懸けて来た首都圏電車網における任意の２点間の最短もしくは第ｋ最短経路の探索を行なうシステムを更に拡張して任意のｎ地点を巡回する最適あるいは準最適な経路の探索をパーソナルコンピュータをもちいて行なうシステムを開発したので御報告する。なお、本システムでは例えばセールスマンが会社を出て顧客を訪問し再び会社に戻って来るような始点と終点とが等しいモード（閉ループ）だけでなく、朝セールスマンが家を出て顧客を訪問した後に会社へ出社するような始点と終点とが異なるモード（開ループ）についての探索も可能である。

An Information System for Optimal Round Trip Tour of Tokyo Metropolitan Railway Networks

Masami KATO　　Keizabro TAKAGI

Faculty of Science and Technology, Sophia University
Kioi-cho 7-1, Chiyoda-ku, Tokyo 102

The authors have already proposed a system which finds the shortest time route between any two stations of the Tokyo metropolitan railway networks. As an extension of this system, we have developed an information system for the optimal round trip tour of the same networks. For a plurality of given n stations, the objective of the system is to find the round trip tour that includes every station once and only once and has the minimal trip time. The proposed system employs an improved depth-first search algorithm, which gives the optimal or sub-optimal solution with a reasonable calculation time. Examples are given to demonstrate the usefulness of the proposed system.

１．まえがき

近年、首都圏の電車網等の公共交通網は益々複雑化しつつあり、利用者はそれらの中から適切な経路を見い出し、効率的に移動することが極めて困難となってきている。一般に任意の複数個の地点を巡回する経路の適切な解を求める問題は巡回セールスマン問題として知られているが、この最適巡回経路探索を実際の大規模なネットワークに適用した例はあまり多くないように見受けられる。ここでは、従来より筆者らが手懸けて来た首都圏電車網における任意の２点間の最短もしくは第ｋ最短経路の探索を行なうシステム(1) を更に拡張して任意のｎ地点を巡回する最適あるいは準最適な経路の探索をパーソナルコンピュータをもちいて行なうシステムを開発したので御報告する。(2) なお、本システムでは例えばセールスマンが会社を出て顧客を訪問し再び会社に戻って来るような始点と終点とが等しいモード（閉ループ）だけでなく、朝セールスマンが家を出て顧客を訪問した後に会社へ出社するような始点と終点とが異なるモード（開ループ）についての探索も可能である。

２．対象とした電車網ネットワーク

本システムは図１に示すように木更津、成田、土浦、小山、熊谷、奥多摩、大月、御殿場、小田原、三崎口で囲まれる範囲内の全ての国鉄、私鉄、地下鉄を含む東京近郊の電車網を対象としている。

この対象とする電車網に含まれる駅は1053駅あり、ここでは後述するように１つの駅を複数のノードによって表現しているためこの電車網を表現するネットワークは、計算機内部では約4000個のノードと約9000本のリンクで構成されている。また、リンクの走行遅延時間は１分を単位として表わしている。

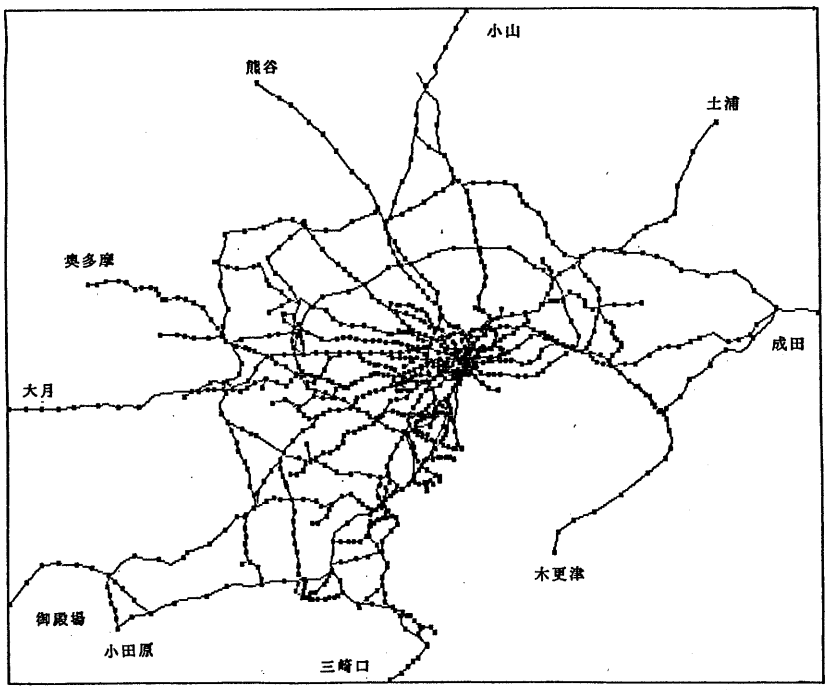


図１　対象とした電車網

3．改良縦型探索による巡回経路探索のアルゴリズム

与えられたｎ地点に対する最適巡回経路の探索には種々の方法が知られている。(3)(4)　例えば０－１整数計画法を用れば次のように定式化できる。

地点ｉから地点ｊへの費用をＣij　、０－１変数をＸij　、頂点集合をＮとしたとき

$$\text{最小化：}\quad Z=\sum_{i\in N}\sum_{j\in N}Cij\cdot Xij$$

$$\text{条件：}\quad \sum_{j\in N}Xij=1,\ \forall i\in N$$

$$\sum_{i\in N}Xij=1,\ \forall j\in N$$

$$\sum_{i\in V}\sum_{j\in N/V}Xij\geqq 1,\ \forall V\subset N\quad (V\neq\phi,\ V\neq N)\quad (N/V\ \text{はＮからＶを除いた差集合})$$

$$Xij=\{0,\ 1\}$$

しかしながら、上述の式を直接解くことができるのはｎがせいぜい数地点までであり、それ以上になると変数および条件式の数が多くなり実用的計算時間での探索は不可能となる。

ここでは分岐限定法（Branch and Bound Method ）を基にした改良縦型探索と名付けた手法により最適あるいは準最適な解を求めている。そのアルゴリズムは次の通りである。

＜提案するアルゴリズム＞

(1) 巡回すべきｎ個の地点に対する費用（所要時間）行列を作り、ｋ←１とする。

(2) 各行各列の禁止フラグの立っていない０要素の有無を調べ、禁止フラグの立っていない０要素の存在しない各行各列からその行あるいは列中の最小値を引き各行各列にすくなくとも１つの０要素を作る。

(3) 各０要素に対して最小除外費用を求め、最小除外費用が最大の０要素を解の枝の一部として採用し、同時にこの採用した要素の行および列に禁止フラグを立て、この行と列を深さｋより深いところでは以後考慮の対象から外す。

(4) サブループを形成する０要素の有無を調べ、もし存在する場合にはその０要素に禁止フラグを立てる。

(5) ｋ←ｋ＋１とし、ｋ＜ｎ＋１なら(2) へ行く。ｋ＝ｎ＋１なら１つの解が求まったことになり、この解を当面の最適解候補とする。つぎに、求めた解よりさらに良い解の存在する可能性のある未探索の枝があればそれらの枝の中で最も良い解が得られると予想される深さｉの枝に対してｋ←ｉとし、バックトラックしてその枝の探索を開始し(2) へ行く。このバックトラックの対象となるべき枝が１つも存在しない場合には直前に求めた解が最適解であり探索を終了する。

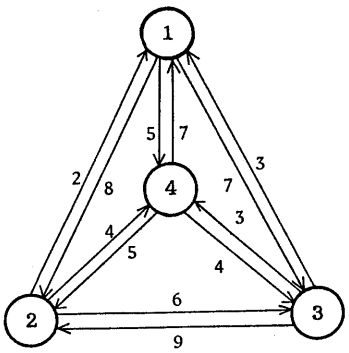


図２　ネットワークの例

	1	2	3	4
1	∞	8	7	5
2	2	∞	6	4
3	3	9	∞	3
4	7	5	4	∞

図３(a) 費用行列

	1	2	3	4	引いた値
1	∞	3	2	0	5
2	0	∞	4	2	2
3	0	6	∞	0	3
4	3	1	0	∞	4

図３(b) 各行に０要素を作る　計１４

	1	2	3	4
1	∞	2	2	0
2	0	∞	4	2
3	0	5	∞	0
4	3	0	0	∞
引いた値	0	1	0	0

計 1

図３(c) 各列に０要素を作る

	1	2	3	4
1				2
2	2			
3	0			0
4		2	2	

図３(d) 最小除外費用

例として図２のように４地点の簡略化されたネットワークモデルにおいて、始点と終点とが等しい最適巡回経路を求める場合について説明する。

ステップ(1) の費用行列は図３(a) のようになり、k←1とする。次に、ステップ(2) により各行各列に少なくとも１つの０要素をつくる。その結果を図３(b) および図３(c) に示す。

さらに、ステップ(3) により各０要素に対する最小除外費用を求める。この最小除外費用を求めるには、例えば図３(c) の(2,1) の０要素について注目すれば、もし、この(2,1) を採用しないと仮定すれば第２行中の他の最小の要素は(2,4) の費用２の要素であり、また、第１列中の他の最小の要素は(3,1) の費用０の要素であるので、求める(2,1) に対する最小除外費用は２＋０＝２となる。この操作を全ての０要素に対して行なえば図３(d) のようになる。ある０要素に対しての最小除外費用とは、その要素をもし除外したとすると最終的な解の費用が、採用した場合に比べて少なくともどのくらい多くなるかを表わしている。したがって最小除外費用が最大の要素を解の枝として採用すればよく、ここで例えば(1,4) の０要素を採用したとすると、最終的な解の下界値LBは14+1であるのに対し、(1,4) の０要素採用しないと最終的な解の下界値LBは14+1+2となり探索木は図３(e) に示す２つの枝を派生する。また、第１行および第４列に禁止フラグを立てる。

ステップ(4) においては、ステップ(3) にて(1,4) の０要素を採用したのでサブループを形成する要素は(4,1) となり、(4,1) に禁止フラグ（∞）を立て（図３(f)）、ステップ(5) によりk←2とし、ふたたびステップ(2) へ行く。

図３(g) は、このような操作を繰り返すことにより最初の１つの最適解候補が求められた様子を示している。求まった解は(1,4), (3,1), (2,3), (4,2) 、つまり１→４→２→３→１

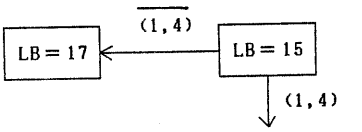


図３(e) 探索木（成長過程）
(LB＝Lower Bound)

	1	2	3	4
1	∞	∞	∞	∞
2	0	∞	4	∞
3	0	5	∞	∞
4	∞	0	0	∞

図３(f) 禁止フラグ

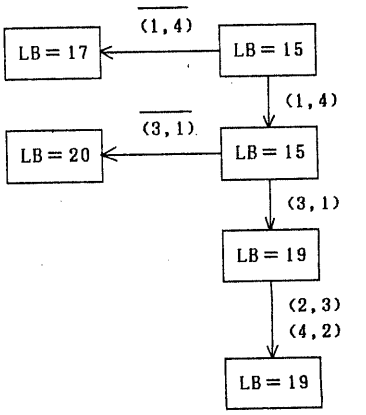


図３(g) 探索木（成長過程）

で、費用は19となる。つぎに、ステップ(5) にしたがいバックトラックを開始する。通常の縦型探索では最も近い未探索の枝にバックトラックするのであるが、ここで提案する改良縦型探索では次のようにしてバックトラックすべき未探索の枝を選定する。図３(g)では、未探索の枝が２つあるが、LB（下界値）＝20の枝は、既に求まった費用が19の解より良い解は存在しないのでこの枝の探索は不要である。一方、LB＝17の枝は求めた解よりさらに良い解が求められる可能性があるため、バックトラックしてこの枝の探索を開始する。可能性のある未探索の枝が複数個ある場合にはＬＢが最も小なるものを選択するがこの例ではLB=17 の枝のみしか存在しないのでたまたま通常の縦型探索と一致する。その結果、図３(h) に示すような解、すなわち１→３→４→２→１で、費用17なる解が発見される。次に、再びバックトラックしようとするが今度はどの未探索の枝もさらに良い解を与える可能性がないため求まった２番目の解が最終的な最適解となり、全探索が終了する。

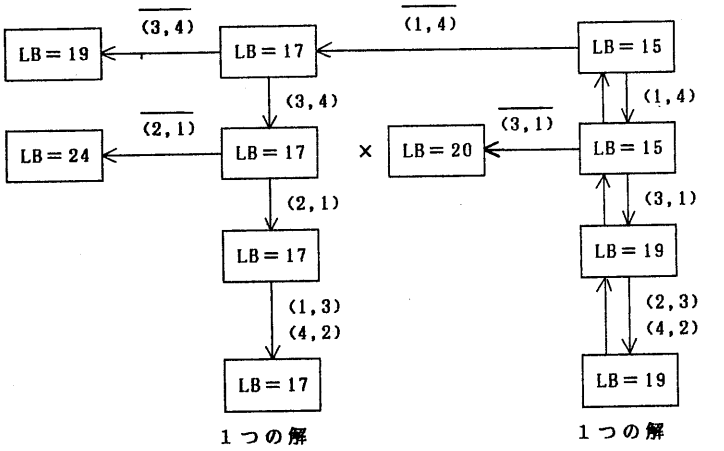


図３(h) 最終的な探索木

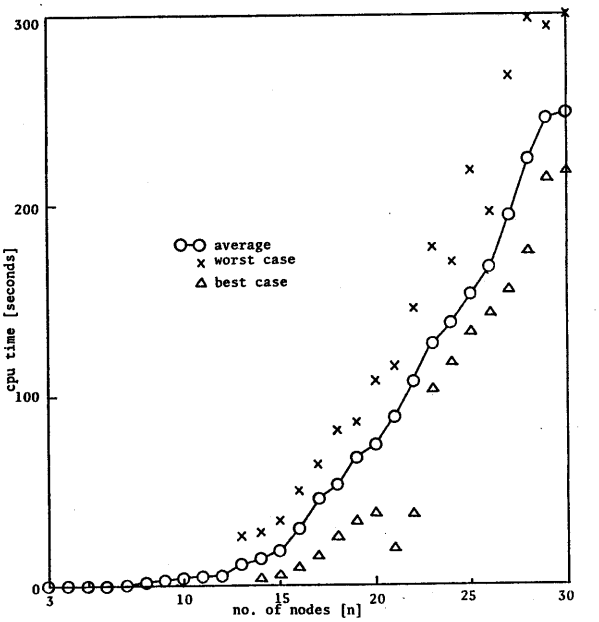


図４　地点数ｎとCPU 時間との関係

以上の説明から明らかなように、ここで採用した改良縦型探索とは、１つの解が求まるか、もしくは、これ以上先へ進めなくなるまでは縦方向優先で探索を行ない、バックトラックが起動された場合には Best First Search Method（最良優先探索）で進むべき未探索の枝を選択する手法であると言える。

始点と終点とが異なるような巡回経路の探索を行なうには上述のアルゴリズムに対して若干の変更を行なえばよい。即ち始点をＳ、終点をＤとすると、探索の深さｋが１のときに限ってＤからＳへ向かう費用０の枝を強制的に採用させ、しかも、以後この枝を他のものと取り替えることを禁止すれば、この問題は始点と終点とが等しいときの探索問題に帰着され、同様に取り扱うことが可能となる。

一般に、枝探索問題において厳密な最適解を求めたい場合、 Best First Search Method 、すなわち常に最良の枝を選択しながら探索を進めて行く手法が最も優れていると言えるが、この手法を用いた場合には最適解を見出すまで探索が終了せず、そのため問題によっては最適解を見出すまでに伸ばす枝の数が膨大になり、多くのメモリと時間を必要とするという欠点がある。しかし、改良縦型探索を用いた場合には、逐次解を改良して行く方法であるため、伸ばした枝の数が許容限度を越したとき既に近似的に良い解が求まってい

る場合には適宜探索を打ち切り、その時点で求まっている解を近似的な最適解として与えることができる。電車網の巡回経路の場合、１日の最大乗車時間は高々１０時間程度、地点数ｎにして２０地点程度であると考えられるので、ここでは経験的に、最初の解が求められた後に伸ばした枝の数がｎ×１５になった時点で探索を打ち切っている。この場合の地点数ｎとＣＰＵ時間との関係を図４に示す。

４．全地点対間の最短経路探索の手法

与えられた任意のｎ地点に対する巡回経路の探索の前段階として、ｎ地点のすべての地点対間の最短経路を求め図３(a) のような費用行列を作成する必要がある。本システムにおいては、これらのすべての地点対間の最短経路を効率良く求めるためにバケット法を採用した。この手法はノード間の移動に要する時間を１分を単位として表わし、計算機内部で出発地に相当するノードから可能なあらゆる方向へリンクを通して波を等速度で四方八方に拡散するシミュレーションを行なっていることと等価である。

図５(a) に示すネットワークモデルを用いてバケット法の原理を説明する。

(1) ノード１を出発地とし、初期状態としてノード１の時間は０と確定しており、まずノード１より発するリンク①およびリンク⑨を時間の早い順に単位幅１のバケットの所定の位置に登録する（図５(b)）。

(2) バケットの先頭からリンク①を取り出す。リンク①は最初にノード２に到達しているのでノード２の時間が１と確定し、ノード２に最初に到達するリンク①をＬＬ（２）＝１として記憶し、同時にノード２より発するリンク②、⑥および⑦を時間の早い順にバケットに登録する（図５(c)）。

(3) バケットの先頭よりリンク②を取り出す。しかしリンク②は既に時間の決定しているノード１へ到着するのでさらにバケットの先頭よりリンク⑨を取り出す。リンク⑨はノード１より出発して最初にノード４に到着するのでノード４の時間が２と確定し、前回と同様、リンク⑨をＬＬ（４）＝３として記憶し、ノード４より発するリンク④およびリンク⑨をバケットの所定の位置に登録する（図５(d)）。

以後、バケットが空になるまで同様な手順を繰り返すことにより全てのノードの時間が確定し、全てのノードに最初に到着するリンクが求められる。その様子を図６に示す。

図６を用いて例えばノード１からノード６への最短経路を求める場合は、まず、ノード６の時間が７であることから最短経路の所要時間は７分となる。次に、ノード６に最初に到着するリンクは⑫でありリンク⑫はノード３より出発し、ノード３に最初に

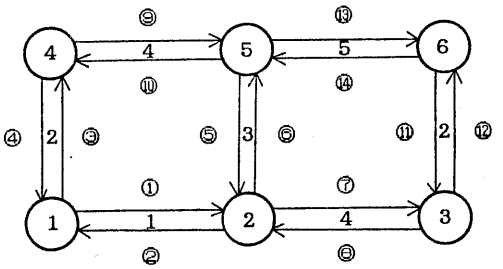
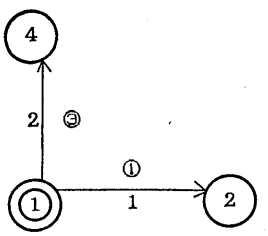
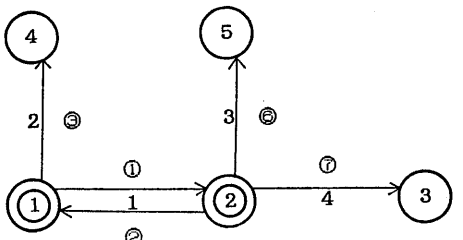


図５(a) ネットワークの例



バケット＝ (1) (2) (1、3)
図５（b）



ＬＬ（２）＝１
バケット＝ (2) (2) (4) (5) (2、3、6、7)
図５(c)

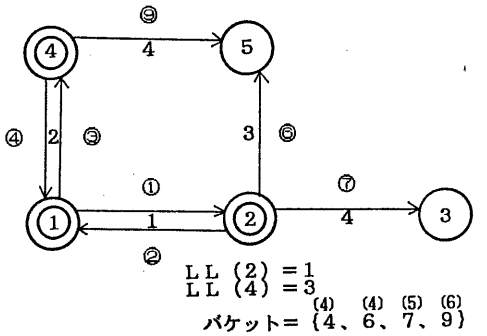


図５(d)

ノード	ノード時間	最初に到着するリンク（ＬＬ）
1	0	φ
2	1	①
3	5	⑦
4	2	⑤
5	4	⑥
6	7	⑫

図６　ノードに最初に到着するリンク

到着するリンク⑦はノード２より出発し、ノード２に最初に到着するリンク①はノード１より出発している。従って求める最短経路は１→２→３→６となる。また、ノード１からノード５への最短経路は同様に１→２→５となり所要時間は４分である。

以上のように始点から他のあらゆる点への最短経路が１回の操作により求めることができるので図３(a)のような費用行列を求めるにはｎ回の操作で済むことになる。

５．計算機内部での駅の表現方法

計算機内部で駅に相当するノードを表現するには、各駅に出入りする路線の情報に加えて各駅の構造上の特徴に起因する情報（例えばホームと改札口の移動時間）も必要なため、ここでは１つの駅を複数のノードで表現している。

例えば図７(a)に示す各停のみ停車するような通常の中間駅では、改札口１、上りホーム２、下りホーム３の３つのノードで表現する。例えば上り線を利用してこの駅で下車する場合、隣の駅から到着した電車は上りホームに相当する２番のノードに到着する。次いでこのホームから改札口に出るのにはその移動時間として１分が加えられることになる。逆に、この駅から例えば上り方面の電車に乗車する場合を考えてみると、まず先の例と同様、改札口からホームへの移動に１分かかり、さらにこの電車の平均運行間隔を４分とするとその平均の待ち時間２分が加えられ、１分＋２分が改札口１から上りホーム２への所定時間となる。

次に、図７(b)は、例として御茶ノ水駅のように各停と快速の両方が停車する中間駅を示している。この駅で下車する場合はどの線を使った場合でもホームから改札口への移動時間は１分であり、逆にこの駅で乗車する場合、快速は上下線共に平均待ち時間が３分であるとしているので改札口４からホーム５あるいはホーム８への移動時間は４分となる。さらに、このような駅では快速と各停の相互間で乗り換える場合があるので下り快速のホームと中野方面各停のホーム間、および上り方面快速のホームと千葉方面各停のホーム間はそれぞれリンクで接続されている。この場合、乗り換えを想定している線はそれぞれ同一ホームの反対側であるので移動時間は０分としてよく、例えば、中野方面

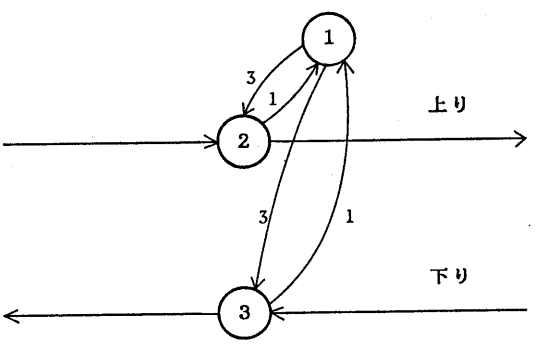


図７(a)　通常の中間駅

各停から下り快速に乗り換える場合には快速の平均待ち時間である３分のみ考慮に入れ、ホーム６からホーム５への所要時間は３分となる。その他の乗り換えは一度改札口の所まで行く形をとることによりデータの簡略化を図っている。

以上、２つの駅を例に挙げて説明したが、他の駅についても同様に、それぞれの駅の特徴を考慮に入れた表現をおこなっている。

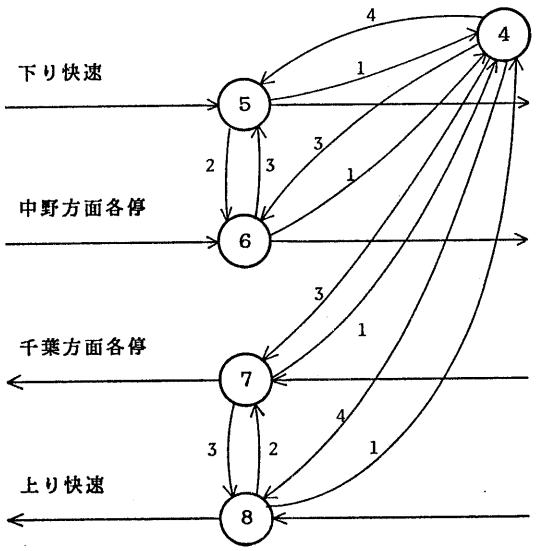


図７(b) 乗り換え駅（御茶ノ水）

６．巡回経路のグラフィックス表示

求めた経路の停車・乗り換え駅や使用する路線等の情報を表示する方法としては文字表示による場合が一般的であるが、複雑な経路を文字のみで表示した場合は一見して理解しにくい。特に巡回する地点数ｎが大となるとその傾向は一層顕著となる。そこで本システムではこれらの情報を利用者がより理解し易い形にするために、文字により経路情報を提供するだけでなく、巡回経路の路線図によるグラフィックス表示も提供している。このグラフィックスによる経路の表示は次の基準に従って行なわれる。

(1) 相対的な位置関係を与える基本路線として山手線、中央線、中央本線、総武線、京浜東北線、埼京線、東北本線、東海道線を表示する。（これらの路線はたとえ利用しなくても表示される）
(2) 求めた巡回経路で利用する路線を表示する。
(3) 山手線全体が必ず含まれるという条件の下で、求めた巡回経路全体が画面上でなるべく大きくなるように拡大して表示し、また、使用する路線を太線で表示する。
(4) 巡回経路に沿って一定の速度でマーカを移動させる。

８．巡回経路の探索例

［例１］１０地点（春日部、新橋、仙川、江戸川台、志木、三ツ境、習志野、武蔵小山、鶴見、東青梅）に対し、出発地を春日部、最終目的地を新橋としたときの開ループの最適巡回経路。（図８）

◎春日部－－（東武野田線）－→◎江戸川台－－（東武野田線）－→柏－－（常磐線［快］）－－－

→松戸－－（新京成）－→◎習志野－－（新京成）－－新津田沼－－（徒歩３分）－→津田沼－－－

－（総武線［各］）－→御茶ノ水－－（営団丸の内線）－→池袋－－（東武東上線［急］）→成増－

－（東武東上線［各］）－→◎志木－－（東武東上線［各］）－→朝霞台－－（徒歩１分）－－－－

→北朝霞－－（武蔵野線）－→西国分寺－－（中央線［快］）－→立川－－（青梅線）－→◎東青梅

－（青梅線）－→立川－－（南武線）－→分倍河原－－（京王線［急］）－→千歳烏山－－－－－－

－－（京王線［各］）－→◎仙川－－（京王線［各］）－→新宿－－（山手線）－→目黒－－－－－

－（東急目蒲線）－→◎武蔵小山－－（東急目蒲線）－→田園調布－－（東急東横線［急］）－－－

→横浜－－（相模鉄道［急］）－→◎三ツ境－－（相模鉄道［急］）－→横浜－－（京浜東北線）－

→◎鶴見－－（京浜東北線）－→◎新橋

全所要時間６２５分、CPU 時間は約１０秒。

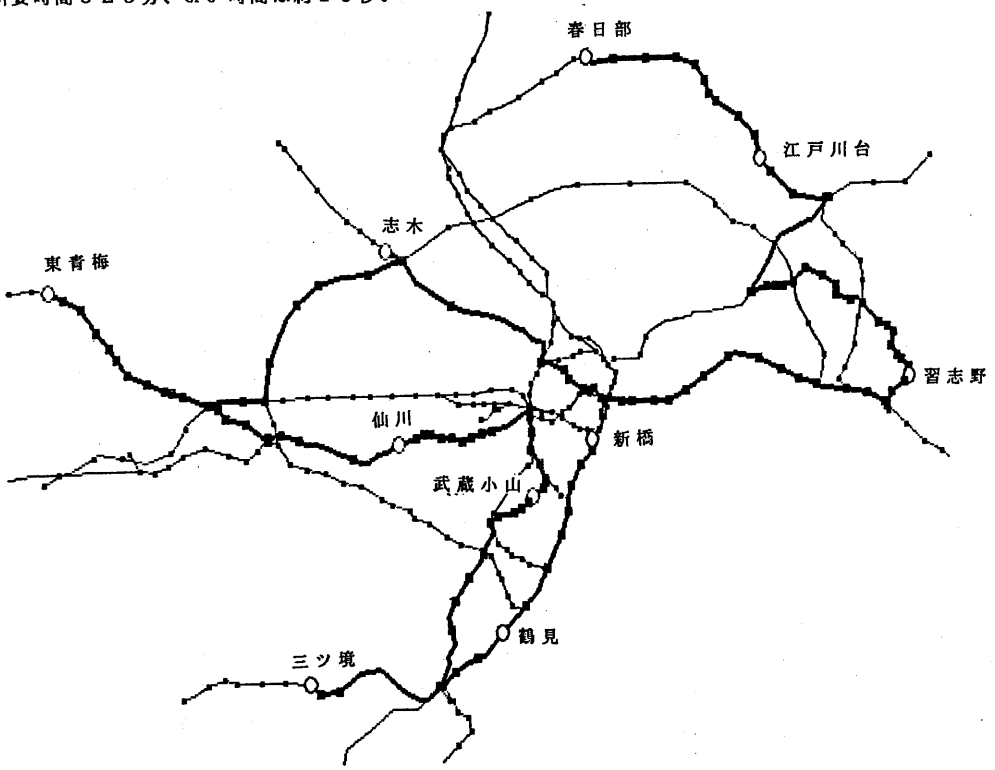


図８　最適巡回経路（開ループ）

［例２］１０地点（田端、渋谷、東京、高輪台、茗荷谷、早稲田、吉祥寺、国会議事堂前、浅草橋、永田町）に対し、東京を出て東京に戻って来る閉ループの最適巡回経路。（図９）

◎東京－－（営団丸の内線）－→◎国会議事堂前－－（徒歩６分）－→◎永田町－－－－－－－－－

－（営団有楽町線）－→飯田橋－－（営団東西線）－→◎早稲田－－（営団東西線）－→中野－－－

－（総武線［各］）－→◎吉祥寺－－（京王井の頭線［急］）－－－→◎渋谷－－（山手線）－－－

→五反田－－（都営浅草線）－→◎高輪台－－（都営浅草線）－→◎浅草橋－－（総武線［各］）－

→秋葉原－－（京浜東北線）－→◎田端－－（山手線）－→池袋－－（営団丸の内線）－→◎茗荷谷

－（営団丸の内線）－→◎東京

全所要時間２０６分、CPU 時間は約１０秒。

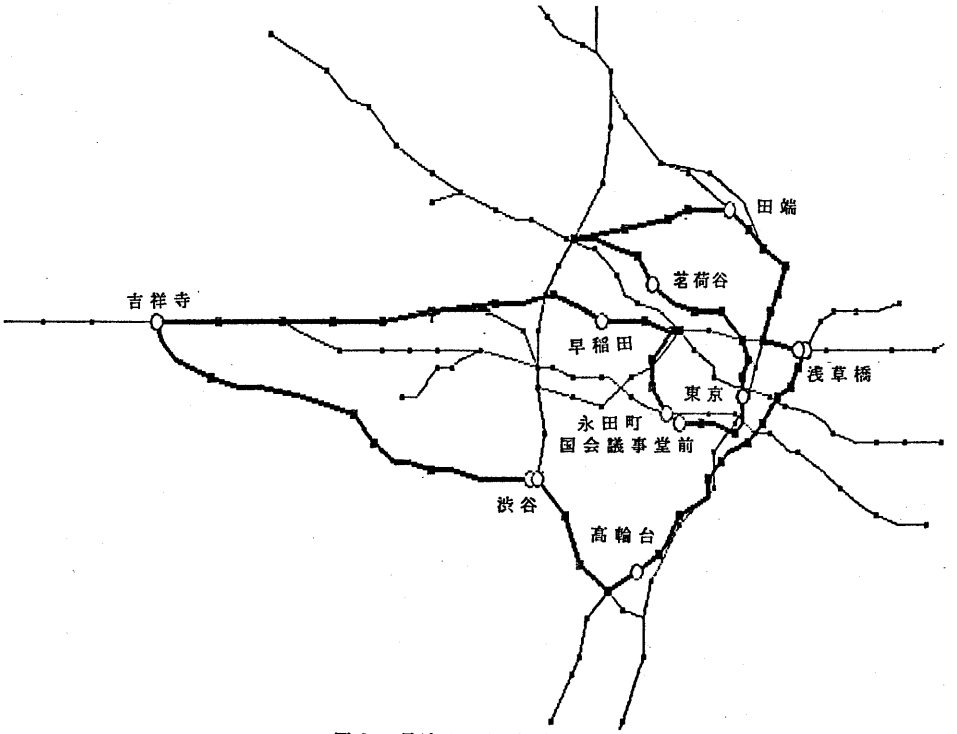


図９　最適巡回経路（閉ループ）

９．むすび

首都圏電車網を対象として、分岐限定法を基にした改良縦型探索を用いて任意の複数個の地点間の最適もしくは準最適な巡回経路をパーソナル・コンピュータを用いてリアルタイムで探索するシステムの概要について述べた。なお計算機はパーソナル・コンピュータ　PC-9801ｍ2　（メモリ 640kB）を使用し、プログラムは FORTRANを用いて作成した。計算所要時間はｎ＝10のとき 2～3 秒、ｎ＝20のとき１分程度であり十分実用的であると考えられる。

今後、この最適巡回経路探索の手法を京都の社寺等の名所巡り順路、デパートにおける買物順路、新聞・宅配便等の配達順路、遊園地におけるアトラクションの見物順路等の探索に応用することを検討している。

(1) 加藤、是永：”ランドマーク指定も可能な代替ルートを含む首都圏電車網乗継情報提供システム”、情処学会第２８回全大、4L-1(S59-03)．

(2) 加藤、高木：”首都圏電車網における準最適一巡経路探索システム”、情処学会第３２回全大、3W-2(S61-03)．

(3) 今野、鈴木：”整数計画法と組合せ最適化”、日科技連、p.330 (1982)．

(4) Claude McMillan,Jr.:"Mathematical Programming"、John Wiley & Sons 、p.650 (1975)．