旅客サービス・システムにおける列車選定の一方式†

石井 博章††

国鉄における旅客サービス・システムの一機能である座席予約においては列車名をキーとする方式がとられている．この場合旅客はあらかじめ時刻表などで希望する列車を調べておかなくてはならない．現在そのシステムを全体的に見なおし機能向上をはかろうという検討がなされており，その一環として旅客の希望する時間帯からシステム側で列車を選定する方式の開発が進められている．それは時間帯予約方式と呼ばれているが，ここではその問題を一種のネットワーク問題，特に枝の長さが時間の関数で与えられたネットワーク上の経路問題，として取り扱い，問題点を明らかにすると共に，それらをマン・マシンにより実用的に解決し，列車選定を行う方式を提案する．この方式は現在の国鉄網の規模にたえうるものであり，かつ優等列車ばかりでなく普通列車をも対象にしたきめの細かいサービスが期待できるものと考えられる．

1. まえがき

現在，国鉄における旅客サービスシステム MARS においては，旅客の希望する列車名をキーとして座席を確保する方式がとられている．この場合，旅客が希望する時間帯に目的駅方面に発車する列車を，あらかじめ時刻表などで，調査しておかなければならない．この点に着目し，旅客が希望する出発または到着の時間帯のみからシステム側で列車を探索し座席を確保しようとする機能向上の検討が，システム全体のサービス向上の検討の一環として，続けられている[1),2)]．ここではそのような列車探索方法を列車選定方式と呼ぶことにする．

列車選定方式は2つの大きな役割が考えられる．第1は旅客の希望に最も適した列車を選定すること，第2は，第1の列車がすでに満席などの理由によって利用不可能な場合に，その代替列車を選定することである．

ここでは，旅客の希望が目的駅に最短時間で到着することと仮定し，発または着時間帯から列車を選定する方式をネットワーク上の経路問題として取り扱う．ネットワーク上にある種の交通機関を仮定した経路問題は，ネットワークの構成要素である枝の長さが時間の関数となる場合の経路問題となることが知られており，特に最短経路問題の効率的な解法も発表されている[3),4)]．列車選定方式はこの最短経路問題のほかにK番目の最短経路問題に関係している．

† A Method of Train Selection in the Passenger Transportation Service System by HIROAKI ISHII (Operations Research Laboratory, Railway Technical Research Institute, JNR).

†† 日本国有鉄道鉄道技術研究所計画管理研究室

従来の最短経路問題の解法を列車選定方式に適用した場合，乗換回数に関係して必ずしも現実的な解が得られないなどの問題が生ずる．

ここではまず，ネットワーク上に鉄道における列車を想定した場合の最短経路アルゴリズムを具体的に述べる．

次に，一定の時間帯の列車の着発時刻を初期値として，そのアルゴリズムをくり返し実行する，あるいは得られた結果を人間が判断することにより，原ネットワークを列車を取り除くなどして変形し，その上で再度アルゴリズムを実行する，などいわゆるマン・マシンにより上記問題点を現実的に解決し，第1希望列車およびその代替列車を選定する方式を提案する．

同時に，モデル線区とその上の現実に近い列車を想定したネットワークによる実験について述べ，その結果を評価することによりここに提案した列車選定方式が実用的なものであることを示す．

2. 鉄道網および列車によるネットワークの構成

国鉄には約5千の駅が存在するが，実用上優等列車（特急・急行）の停車する駅あるいは分岐駅を節点としたネットワークを対象にすればよいと考えられる．この規模を前提として，以下のように節点，枝およびその長さ，そして記号を定義する．

（1） 節点　国鉄における主要な駅を節点とする．節点の総数をnとし，節点を$v_1, v_2, \cdots, v_n$で表わす．

（2） 枝　2節点間に，その節点を停車駅とし，ほかの節点を中間の停車駅としない列車が存在するとき，その列車と同じ方向の有向枝を設定する．たとえ

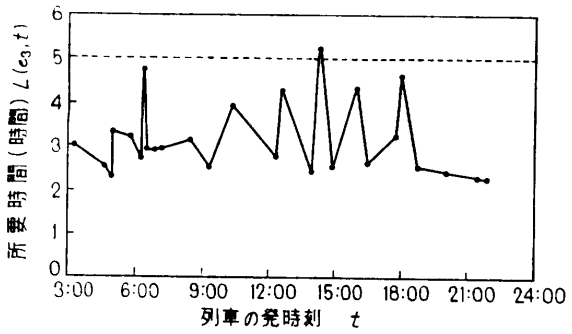

図 1 (盛岡→青森) 間の列車の所要時間
e_3

Fig. 1 Train Hours between Morioka and Aomori.

ば，列車の中には節点として取り上げた駅をも通過駅とするものがあるが，この場合はこの列車の隣接する停車節点間に枝を設定する．この意味で枝は鉄道線区をモデル化したものといえる．一般に鉄道の駅間には上り，下りの列車が存在するから，各節点間には方向が逆になる2本の枝を設定することになる．枝の総数を m とし，枝を $e_1, e_2, \cdots, e_m$ で表わす．

(3) 枝の長さ 枝の一方の端節点を発車してからほかの端節点に到着するまでの列車の所要時間を枝の長さとする．これは節点における列車の出発時刻 t の関数となる．枝 $e_l, l=1,2,\cdots,m,$ の長さを $L(e_l,t)$ と表わす．**図1**は例として盛岡→青森間に枝を設定した場合のその枝の長さの変化を示したものである．

ここでその他の記号を次のように定義する．

$d_l, l=1,\cdots,m$: 枝 e_l 上の列車本数，

$r_{l,k_l}, k_l=1,\cdots,d_l$: 枝 e_l 上の列車番号，

s_{l,k_l}, a_{l,k_l}: 枝 e_l 上の r_{l,k_l} に対応する列車の発時刻と着時刻．

そして次の仮定をおくことにする．

$$|a_{l,k_l}-s_{l,k_l}|\leq T,$$

ここで，$l=1,\cdots,m, k_l=1,\cdots,d_l,$ そして T は列車の周期を表わすものとする．実際は $T=1,440$(分) (24時間) とするが，この仮定は国鉄の現状からみて，列車の隣接節点間の所要時分は 24 時間を越えないことを意味している．

このとき，アーク e_l の長さは $t=s_{l,k_l}, k_l=1,\cdots, d_l,$ にたいし次のように定義される．

$$L(e_l,t)=\begin{cases}a_{l,k_l}-s_{l,k_l}, & a_{l,k_l}>s_{l,k_l},\\ a_{l,k_l}-s_{l,k_l}+T, & a_{l,k_l}<s_{l,k_l}.\end{cases}$$

3. 列車による最短経路アルゴリズム

2章に述べたネットワーク上の最短経路問題は基本的に Dijkstra のアルゴリズムで解けることが知られている[4]．ここでは列車の時刻表を用いてラベリング法を実行する具体的な方法について述べる．

まず次のように枝 $e_l, l=1,\cdots,m,$ にたいする列車の発時刻，着時刻，列車番号を編集する．

(1) もし $s_{l,k_l}>a_{l,k_l}$ ならば $a_{l,k_l}+T$ をあらためて a_{l,k_l} とする．次に着時刻が

(2) $a_{l,1}<a_{l,2}<\cdots a_{l,k_l}\cdots<a_{l,d_l}$

となるように発時刻，着時刻，列車番号をならびかえる．

枝 $e_l=(v_i,v_j)$ とする．v_i を時刻 s_0 に出発可能とするとき，e_l 上の列車によって v_j に最も早く到着するための列車の番号は，$k_l=1,\cdots,d_l$ にたいし，

$$s_0\leq s_{l,k_l}$$

となる最初の k_l に対応する r_{l,k_l} か，あるいはそのような k_l が存在しなければ

$$s_0-T\leq s_{l,k_l}$$

となる最初の k_l に対応する r_{l,k_l} となる．このような k_l を $k_l=\mathrm{fast}\,1\,(e,i,s_0)$ と定義する．

その他の記号を次のように定義する．

origin: 出発節点番号，

destin: 目的節点番号，

t_0: v*origin* における出発可能時刻，

f: 時刻を分単位になおす関数，

g: 分単位を時刻になおす関数．

各節点には次の5種類のラベルを仮定する．

$l_1(i)$: v*origin* における t_0 から v_i までの経過時間，

$l_2(i)$: v_i に到着する1つ前の節点番号，

$l_3(i)$: 枝 (v_j,v_i) の番号，ただし $j=l_2(i)$，

$l_4(i)$: v_i に到着するために利用した，$e_l, l=l_3(i),$ 上の，列車の識別子，

$l_5(i)$: v_i の状態．すなわち $l_5(i)=0$ ならば v_i がまだラベルをつける対象になっていない，$l_5(i)=1$ ならば v_i のすべてのラベルは仮のものである，$l_5(i)=2$ ならば v_i のすべてのラベルは確定している，とする．

アルゴリズム：

ステップ0 $l_\alpha(i)=0, \alpha=1,\cdots,5, i=1,\cdots,n, i=origin$ $\bar{t}_0=f(t_0)$，と初期設定する．

ステップ1 $l_5(i)=2$ とし，$i=destin$ ならば終了，そうでなければ，$t_f=l_1(i)+\bar{t}_0, t_g=g(t_f)$ とする．

次に v_i の隣接節点 v_j の中 $l_5(j)\neq 2$ であるすべての v_j について以下のことを行う．

$$z=\begin{cases} l_1(i)+a_{l,k_l}-t_\theta, & t_\theta \le s_{l,k_l}, \\ l_1(i)+a_{l,k_l}-t_\theta+T, & t_\theta > s_{l,k_l}, \end{cases}$$

とする．ここで．$e_l=(v_i, v_j)$，$k_l=\text{fast}\,1(l, i, t_\theta)$．そして，$z<l_1(j)$ の場合についてのみ，$l_1(j)=z, l_2(j)=i$，$l_3(j)=l$，$l_4(j)=k_l$，$l_5(j)=1$ とラベルを設定しなおす．

ステップ2　$l_5(h)=1$ であるすべての節点 v_h について，$l_1(h)$ が最小となる節点の1つ v_h をみいだし，$i=h$ としてステップ1にもどる．

以上述べたアルゴリズムを TLROAD (Train Line Road) と名づけ以後引用することにする．

ここで旅客が到着時間帯を指定した場合，到着時刻から逆に列車を探索するアルゴリズムについて述べる．

まず列車に関する情報を(1)と同様な処理の後，

$$s_{l,1}<s_{l,2}<\cdots<s_{l,k_l}<\cdots<s_{l,d_l}$$

となるように編集する．

時刻 s_0 に v_i にいるとするとき，$e_l=(v_j, v_i)$ の列車の中 v_i の発時刻から s_0 までの時間が最小となるものの列車番号を次のように選択する．$k_l=1, \cdots, d_l$ にたいし，

$$s_0 \ge a_{l,k_l}$$

となる最後の k_l に対応する r_{l,k_l} か，あるいはそのような k_l が存在しなければ

$$s_0+T \ge a_{l,k_l}$$

となる最後の k_l に対応する r_{l,k_l} をとる．このような k_l を $k_l=\text{fast2}\,(l, i, s_0)$ と定義する．次に

g'：分単位の時間を，目的駅における到着時刻からその分単位の時間だけ以前の，時刻に変換する関数，を定義する．そして3章のアルゴリズムのステップ1において，$t_{\theta'}=g'(t_f)$ とする．そして $k_l=\text{fast}\,2(l, i, t_{\theta'})$ にたいし，

$$z=\begin{cases} l_1(i)-s_{l,k_l}+t_{\theta'}, & a_{l,k_l}<t_{\theta'} \\ l_1(i)-s_{l,k_l}+t_{\theta'}+T, & a_{l,k_l}-T<t_{\theta'}<a_{l,k_l} \\ l_1(i)-s_{l,k_l}+t_{\theta'}+2T, & t_{\theta'}<a_{l,k_l}-T \end{cases}$$

というように改造する．このアルゴリズムを LTROAD と呼ぶことにする．

4. TLROAD の問題点と解決方法

前節のアルゴリズム TLROAD は，通常のラベリング法の応用であり，1つの枝上の列車が1操作の対象となっているため，いくつかの枝にわたって走行する列車も枝ごとにあたかも異なった列車であるかのように取り扱われる．そこで以下に述べる問題が生ずる可能性がある．

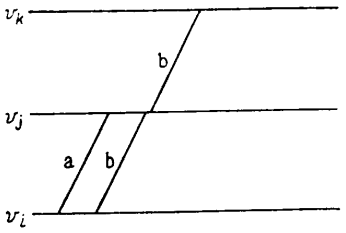

図 2　列車選択（1）
Fig. 2　Train selection (1).

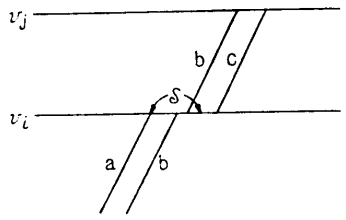

図 3　列車選択（2）
Fig. 3　Train selection (2).

（1）　目的駅における到着時刻が等しい列車による経路がいくつか存在する場合は，その中で出発駅における出発可能時刻に最も近い発時刻の列車を選択する．それは途中駅にたいしても最も早く到着する列車を選択するからであるが，現実問題として，到着時刻が同一であれば最も遅い発時刻の列車あるいはできる限り直通列車を選択する方が望ましいと考えられる．**図2**によりその現象を説明すると，v_i から v_k に行くには直通列車bを選択する方が現実的であるにもかかわらず，v_i, v_j 間はa列車，v_j, v_k 間はb列車というように選択することになる．

（2）　途中駅において列車の乗り換えに必要となる余裕時分を考慮していない．それは乗り換え余裕時分を導入すると目的駅への最早到着列車を選択できない場合が生ずるからである．**図3**は v_j に最も早く到着するために直通列車bを利用すればよいことを示している．しかし乗り換え余裕時分 δ を導入すると，問題点(1)により v_i に到着するためにa列車を選択した後，v_j に到着するためにc列車を選択することになる．Dijkstra のアルゴリズムは過去に経由してきた枝に依存せずに最短経路を求める方法であり，そのためにこの問題が生ずることは当然考えられる．乗り換え余裕時分を考慮した経路問題は，過去に経由した枝に依存する経路問題，たとえば Turn Penalty をもつ経路問

題，に関係していると考えられその解決は今後に残されている．

上に述べた2つの問題点を実用的に解決し直通列車あるいは乗り換え回数の少ない列車を選択する一方法として，出発駅における旅客の希望する発時間帯に発車するすべての列車の発時刻を初期値 t_0 として，その時刻の数だけ TLROAD を実行し，その結果を人間が判断して希望列車を選択することが考えられる．

この方法では，その時間帯に直通列車が存在しかつその列車による所要時分が，出発駅におけるその列車の発時刻からみて，最少であればそれを必ずみいだすことができる．その列車の発時刻も初期値として TLROAD を実行するからである．

直通列車が存在しないとき，乗り換え回数のより少ない列車をみいだすためには次のようにする．得られている任意の列車による経路を R_1 とするとき，R_1 の1列車をネットワークから取り除き TLROAD を実行する．この操作をくり返して得られた，R_1 と同時刻に目的駅に到着する，最初の経路を R_2 とする．次に R_1 と R_2 の各経路から1列車ずつネットワークから取り除き TLROAD を実行するということをくり返し，得られた R_1 と同時刻に目的駅に到着する，最初の経路を R_3 とする．一般に $R_1, R_2, \cdots, R_r$ の経路が得られているとし，各経路から1列車ずつ取り除き TLROAD を実行することをくり返すことにより，新しい経路 R_{r+1} が存在すれば得ることができる．到着時刻の等しい経路の数は有限であるから，このステップは有限回で終了する．

このステップは希望する乗り換え回数の経路が得られた時点で終了させてもよい．しかし R_1 と同時刻に到着する経路の中で希望するものが得られなかったときは，第2番目に最早到着する経路について上と同様なステップを実行すればよいのであるが，その経路の求め方については次章にゆずることにする．

人間を介入させて取り除く列車を判断させることには次の利点が考えられる．

まず列車にたいする人間の知識を利用できる．特にある区間に直通列車が存在するかどうかの知識があれば無意味な乗り換え列車，すなわち取り除く対象になる列車，を判断しやすい．そのために無意味な列車を1度にすべて取り除くなど上記ステップを大幅に減少させることができると考えられる．次に乗り換え余裕時分は駅設備あるいは乗り換え線区などによって変動する量であるが，乗り換え駅ごとの細かい判断が期待できる．

5. TLROAD および LITROAD による列車選定実験

図4は東北地方の鉄道網を中心にモデル化したネットワークである．ただし，枝の存在するすべての節点間に，列車の上り，下りに対応する，一対の有向枝が存在するので，煩雑をさけるためにそれらを1本の無向枝で示している．節点数は36，有向枝の数は122である．対象とした列車は特急，急行，普通を合せて約1,100本であり，各枝上の列車本数を合計した延べ列車本数は約3,000本である．これらの列車は現実のものを用いており，規模も現実に近いものである．図4において，たとえば v_{10} と v_7 間の枝は v_9 と v_8 を通過駅とする列車が存在するために設定されたものである．

実験1　上野発列車の発時刻を初期値として青森に最

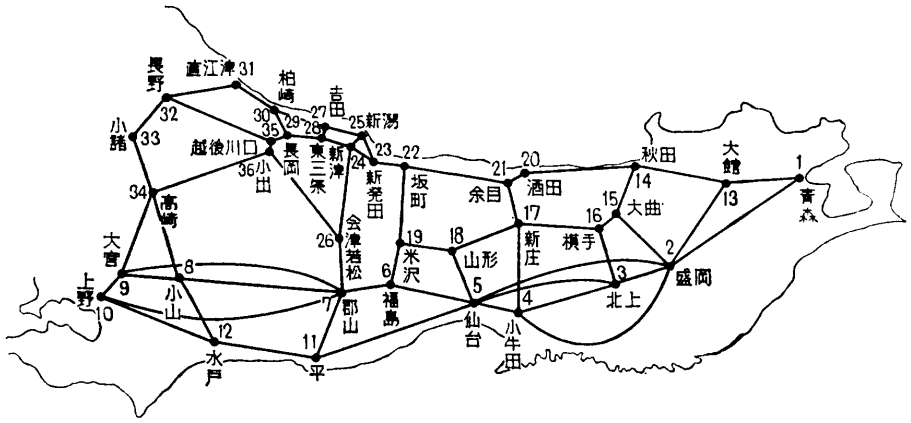

図4　実験に用いたネットワーク（数字はノード番号）
Fig. 4　Network used in the tests.

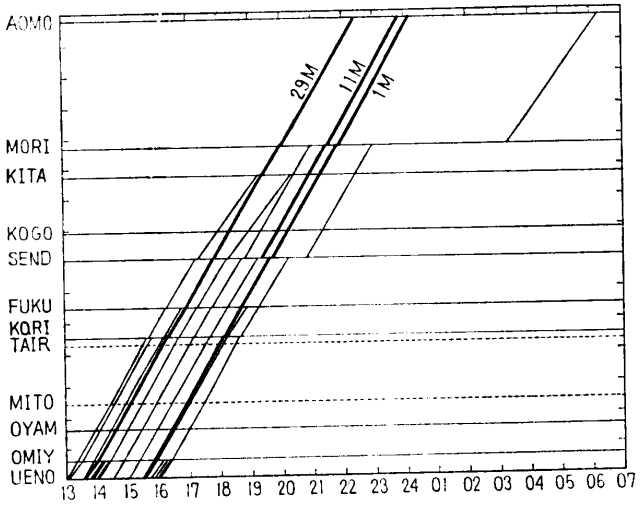

図 5　13：00～16：00 の上野発列車の発時刻による青森への最早到着列車

Fig. 5　The fastest trains from Ueno to Aomori starting at Ueno.

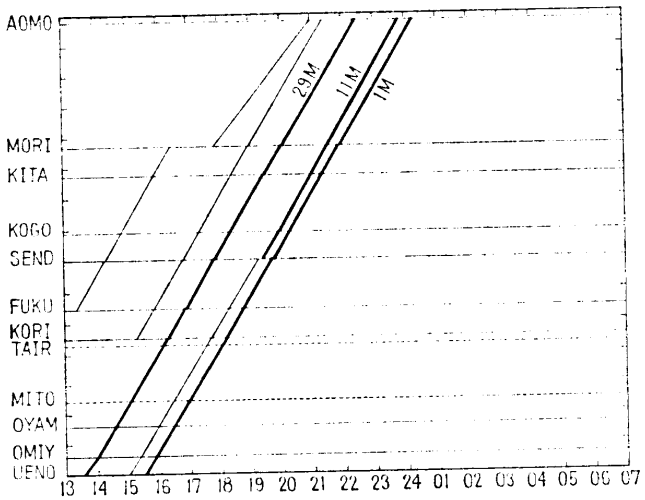

図 6　21：00～1：00 の青森到着列車の到着時刻による上野発青森最早到着列車

Fig. 6　The fastest trains obtained from arrival times.

早到着する列車の選定

図 5 は 13：00 から 16：00 までの間に上野を発車する列車（東北本線，常磐線，高崎線）の発時刻を t_0 として実験を行った結果を列車ダイヤ形式に表現したものである．発時刻数は計 41 であるが，すべて東北線の発列車が選定されるという結果になった．この図から 13：00～16：00 の時間帯で上野を出発し青森に最早で到着する直通列車として 29M がみいだせる．

実験 2　青森到着列車の到着時刻を初期値とした上野発青森最早到着列車の選定

図 6 は 21：00 から 1：00 までの間に青森に到着する列車（東北本線，奥羽本線）の到着時刻を t_0 として，その到着時刻の数 10 だけ LTROAD を実行した結果を列車ダイヤ形式に表現したものである．実験 1 と同様に直通列車 29 M がみいだせるが，図 6 に現われている途中の無意味と考えられる多数の列車が取り除かれ，直通列車がみやすくなっている．それは一般に大都市周辺の駅に着発する列車の数が地方の駅に着発する列車の数より多いためでもあり，大都市周辺の駅を出発駅とし地方の駅を目的駅とする場合は LTROAD による方が有利であると考えられる．

6. 代替列車の選定

第 1 希望列車が満席などの理由によって採用できないとき，その代替列車を求める問題について考える．この問題は 2 章に述べたネットワーク上の Kth Path 問題と考えることができる．

枝の長さが時間の関数でないネットワーク上の Kth Path 問題にたいし多くの解法が発表されている．その中で，$1, 2, \cdots, k$-1 番目の経路上の枝を取り除くことによりネットワークを変形し，それに最短経路アルゴリズムを適用する方法[5)~7)]が代替列車選定に容易に拡張できると考えられる．

その場合，枝を取り除くかわりに列車を取り除くことによりネットワークを変形し，それに TLROAD を適用することになる．しかし 4 章に述べたごとく，TLROAD は列車を枝ごとに分割して取り扱うために，実用的には無価値と考えられる多数の経路を算出する可能性がある．たとえば，何本かの直通列車が存在するとき，それらを枝ごとに細切れに乗り換えていく経路も解として算出される．K を十分大きくとってこの問題に対処する方法も考えられるが，処理時間あるいは記憶容量の面で現実的ではない．**5** 章で用いたネットワーク上で上野→青森間の Kth Path を求める実験において，処理時間 0.61 $k(2 \leq k \leq K)$ 秒 (IBM 3031) が観測され，余分に発生した経路数は $5K$～$15K$ となった．その上，$K=10$ 程度ではまだ目的駅への到着時間がすべて等しいという結果となった．

そこで，実用的な解決方法として，取り除く列車を人

間が判断しネットワークを変形する方式を考察した．

まず第1希望列車と同時刻に目的駅に到着するすべての経路を4章に述べた，乗り換え回数のより少ない経路を求める方法と同様な，手段によって求める．その中に代替列車となる経路が存在しなければ，各経路から目的駅に到着するために使用された最終列車を当該枝から取り除き，TLROAD を実行する．その結果，第2番目の最早到着時刻の経路が得られる．上と同様にその時刻と同時刻に目的駅に到着するすべての経路を求める．その際，取り除かれた最終列車は取り除いたままにしておく．以下同様に代替列車経路が得られるまでくり返す．

この取り除く列車を人間の判断による方法は4章に述べたと同様な利点のほかに，発生する経路をいちいち計算機に記憶させる必要はなく，代替列車経路の対象となる経路および取り除く列車を記憶させておけばよいため，記憶容量の面でも有利と考えることができる．

図7は図5の 29 M の代替列車を求めるために，(v_6, v_5) 上の 29 M を含めて 29 M と同時刻に青森に到着可能な列車をすべて取り除いて，実行した結果であるが，代替列車として 11 M がみいだせる．

7. ワン・マシン・インタフェース

インタフェースとしてグラフィック・ディスプレイを用い，画面には縦軸に距離，横軸に時間をとって，計算の結果得られた経路を列車ダイヤ形式に表示できればよい．その際距離は 1,000 km 程度，時間は 12 時間程度まで表示できれば，国鉄のネットワークの現状と首都圏を中心とした乗り継ぎ形態からみて，十分であろう．それより長時間あるいは長距離の経路，そして同一画面で表示できないような，異なった線区による，代替経路などの表示のために画面の切り換えが容易にできるよう工夫する必要はある．

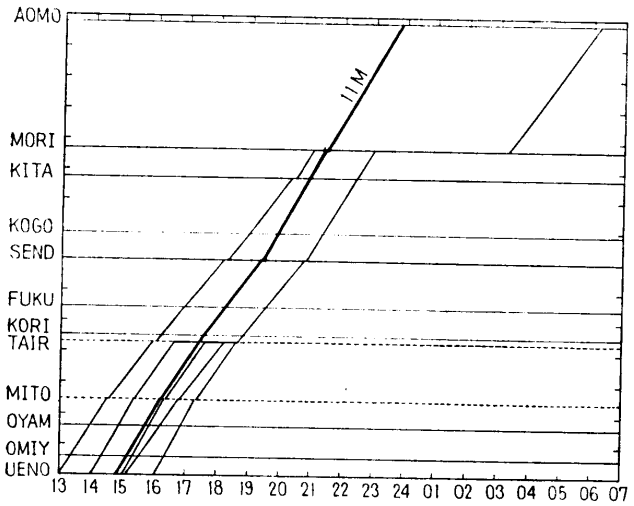

図7 ネットワークの変形による代替列車の選定

Fig. 7 The second train routes obtained from modified network.

画面には列車ダイヤに沿って列車番号，あるいはそれを線区ごとにコード化されたコード番号，を表示することにし，経路探索における列車の取り除きはそれらの番号をキー入力することにより行えばよい．

画面は経路を列車ダイヤ形式に表示し，列車の走行順序あるいは乗り換え余裕時分などを視覚によって判断しやすくするために用いるのであって，図形処理を目的としないので，グラフィック装置は蓄積管形のものでよいと考えられる．しかし，具体的なシステム設計の検討は今後に残されている．

8. 国鉄の現状からみたアルゴリズムの評価

図5のネットワークについて考察すると，節点数36にたいし無向枝数61であるから，節点の平均度数は $2\times61/36\fallingdotseq3.4$ であり，国鉄全体の鉄道網をモデル化した場合，それを4とすれば十分であると考えられる．そこで各節点の度数，隣接節点，隣接節点との間の枝にたいしおよそ $n+4n+4n=9n$ の格納場所が必要になる，ここで n は節点の総数である．次に2章で定義した有向枝上の平均列車本数を L とするとき，発時刻，着時刻，列車番号，列車本数にたいしおよそ $3L+m$ の格納場所が必要になる，ここで m は枝の数である．したがって入力情報全体では，$m\fallingdotseq4n$ を考慮すると，

$$9n+3Lm+m=(13+12L)n$$

の格納場所が必要になる．このほかに作業領域として3章に述べた5種のラベルにたいし $5n$，6章に述べた列車を取り除くか否かの識別にたいし $Lm=4Ln$ が必要になる．国鉄の現状では分岐駅が約400存在するが主要なものは 200 以下と推定され，各枝の平均列車本数 L も 30 本程度と推定される．そこで全体で必要となる領域は 99,600≒100 kW と推定される．

次に演算の回数であるが，Dijkstra のアルゴリズムは最大 n^2 回の加法および比較演算が必要であるが，TLROAD はその他に各節点で列車を選択するために最大 Ln 回の比較を行うの

で，Ln^2の比較演算が追加され，加法演算回数も2～3倍になると推定される．なお，5章の実験では，$n=36$, $L=3,000/122 \fallingdotseq 25$ であるが，IBM 3031 により1回の TLROAD の実行は約 70 ms と観測された．この時間はネットワークの形状および出発駅と目的駅間の経路上の枝などに関係すると考えられるが，枝数が1増加すると実行時間は約5 ms～10 ms 増加することが観測された．国鉄のネットワークを本州全体の規模とした場合，分岐駅間を1区間とすると，平均的にみて2駅間約25区間とみられるから，上の結果から1回の実行に 125 ms～250 ms かかると推定できる．

9. 中間駅の扱い方

いままで考察の対象としてきたネットワークは国鉄における分岐駅などの主要な駅からなる規模のものであり，その上の節点間について列車選定方式を提案した．ここでは主要駅として取り上げなかった駅，簡単のためそれを中間駅と呼ぶ，が出発駅あるいは目的駅となる場合にたいする処理について述べる．

まず出発駅と目的駅のどちらか一方が中間駅である場合について以下のようにネットワークを部分修正する．すなわち v_α を主要駅 v_i, v_j 間の中間駅とする．このときネットワークに節点 v_α および枝 (v_α, v_i), (v_i, v_α), (v_α, v_j), (v_j, v_α) をその上の列車と共に追加し，枝 (v_i, v_j), (v_j, v_i) をその上の列車と共に削除する．出発駅と目的駅の両方とも中間駅であればその両中間駅について上と同様の操作を行う．この修正は中間駅も主要駅として扱えるようにすることに相当する．この処理方式は外部記憶装置から中間駅にたいする情報を転送するオーバヘッドがあるが，国鉄には約5千の中間駅が存在することを考慮すると実用的な方式であり，この考え方はほかにも利用されている．

10. むすび

ここに提案した方式は優等列車ばかりでなく普通列車をも対象にすることができ，きめの細かい旅客サービスが期待できると考えられる．今後，列車ダイヤの改正，臨時列車などを考慮したデータ構造，国鉄の鉄道網の地域的特徴を生かした分割アルゴリズム，あるいは列車の自動選定を目的とした列車による経路の評価方法などいくつかの課題が残されている．

参考文献

1) 槻木，木村他：時間帯指定による列車案内および予約方式，第14回鉄道におけるサイバネティクス利用国内シンポジウム論文集 (1977).
2) 木村，槻木他：電話による列車案内，第15回鉄道におけるサイバネティクス利用国内シンポジウム論文集 (1978).
3) Cooke, K. L. and Halsey, E.: The shortest route through a network with time-dependent internodel transit times, J. Math. Anal. and Appl., Vol. 14, pp. 493-498 (1966).
4) Dreyfus, S.: An appraisal of some shortest-path algorithms, Opns. Res., Vol. 17, pp. 395-412 (1969).
5) Lawler, E. L.: Combinatorial Optimization: Network and Matroids, p. 374, HOLT, RINEHART AND WINSTON (1976).
6) Pollack, M.: The Kth best route through a network, Opns. Res., Vol. 9, pp. 578-579 (1961).
7) Yen, J. Y.: Finding the k shortest loopless paths in a network, Man. Sci., Vol. 17. pp. 712-716 (1971).

(昭和55年5月28日受付)
(昭和56年1月22日採録)