日本オペレーションズ・リサーチ学会和文論文誌
Transactions of the Operations Research
Society of Japan
2008 年 51 巻 1-24

鉄道運賃計算のための最安運賃経路探索
― 複数の鉄道会社を含む場合 ―

池上 敦子
成蹊大学

森田 隼史
成蹊大学
(現在 日本信号)

山口 拓真
日本信号

菊地 丞
日本信号

中山 利宏
日本信号

大倉 元宏
成蹊大学

(受理 2007 年 7 月 7 日；再受理 2008 年 1 月 16 日)

和文概要　本研究では，運賃設定の異なる複数の鉄道会社を含む鉄道ネットワーク上の運賃計算を正確かつ高速に行えるネットワーク表現とアルゴリズムについて報告する．鉄道運賃は，利用者の乗車経路が明らかであるとき，多くの場合，その経路に含まれる各鉄道会社が定めた運賃を足し合わせることによって得られる．一方，利用者の乗車経路が明確でない場合，利用可能経路の中で最も安い経路を利用したとみなし，その運賃を採用することが一般的である．しかし，鉄道運賃は，基本的には「距離が長くなればなるほど高く」なるように設定されているものの，同じ距離でも，会社によって異なる料金が設定されていることや，乗車区間によって割引ルールや特別運賃が設定されていることなどから，物理的距離に基づくショーテストパスが最も安い経路になるわけではない．よって，与えられた 2 駅間の正しい運賃を計算するためには，その 2 駅間の可能経路の運賃をすべて，もしくは，その 1 部を列挙して比較判断する必要があることがこれまでにも報告されてきた．本研究では，物理的構造に基づくネットワーク上での経路探索を行う代わりに，ダイクストラ法が利用可能な運賃計算用ネットワークを構築し，ダイクストラ法と，少ないケースではあるが K-shortest paths 問題用のアルゴリズムを利用することにより，複数社を含む鉄道ネットワーク運賃計算の大幅な高速化に成功した．

キーワード: 交通, 経路探索, アルゴリズム, ネットワークフロー

1. はじめに

鉄道ネットワークは，駅をノード，線路をアークとした物理的な要素と，鉄道会社間で定義された連絡関係（他社の駅への乗換え可能性）といった運用上の要素を含んでいる．1 社内のネットワークに限った場合，異なる路線への乗換えはそれほど複雑ではない．例えば，東日本旅客鉄道（JR 東日本）の山手線の新宿駅と総武線の新宿駅は同じ JR 新宿駅として考えられるので，ネットワーク上では 1 つのノードとして扱うことで対応することができる．一方，会社間の連絡関係の表現は難しく，例えば，JR 有楽町駅と地下鉄有楽町駅は連絡可能で，地下鉄有楽町駅と都営日比谷駅も連絡可能であるが，JR 有楽町駅と都営日比谷駅は連絡可能とされていない．すなわち，JR 東日本から東京都交通局（都営地下鉄）への JR 有楽町駅 (都営日比谷駅) を乗換駅とするような連絡乗車券は販売されていないのである．よって，1 つのノードとして扱えないばかりでなく，連絡をアークで表現するにも注意が必要である．ここで，連絡可能であるか否かは，駅間の地理的関係に基づき，運賃計算を考慮して定義されているという．

鉄道の運賃計算を考えるとき，与えられた経路に対する運賃計算と，与えられた 2 駅間の運賃計算との違いを意識しなければならない．与えられた経路に対する運賃は，利用鉄道会社，利用路線，利用距離，各社の特別ルール等に従って計算されるが，基本的には，各社内の始点と終点の駅の間の運賃を足し合わせれば求めることができる．また，各社初乗り運賃

が高く設定されていることへの対応策として，複数社を短い距離で乗り継いだ場合に適用される併算割引というルールも存在するが，与えられた経路が，併算割引対象であるかをチェックして必要であれば適用すればよい．

一方，与えられた2駅間の運賃は，通常，その2駅間の利用可能経路の中で最も安い運賃の経路を利用したと仮定し，その運賃を適用しなければならない．1利用者に対する運賃計算は，一般に，その発駅（乗車駅）と着駅（降車駅）に対して発生するので，鉄道運賃計算とは，最安運賃経路探索問題と捉えることができる．しかし，首都圏の鉄道ネットワークのように，その規模が大きかったり，密になればなるほど可能経路の数は膨大になり，あらかじめ全2駅間の運賃を求めようとした場合，列挙と比較による経路探索では，多くの計算時間が費やされてしまうという問題がある．

鉄道を利用する立場に立った乗換え案内サービスシステムとしては，市販ソフトウェア [16][17][30][20][24]，Web上のシステム [15][40][10][19][22][25][31][37][18][41][32][42][21][26][34][36][23][11][20][27][28][29][12][13][14][39][35] 等，多くの人に利用されている．半田，田中 [3] は，乗換え案内サービス駅探 [15] において，ネットワーク上のアークのコストに平均所要時間を設定し，K-shortest paths 用のアルゴリズムである MPS 法 [5][6] を利用して，与えられた2駅間における可能経路を列挙・比較し，良好な経路を高速に探索できるようにした．このように，利用者の視点に立って考えれば，長い時間をかけて最適解を得るより，最適解である保証はなくとも良好な解（高い確率で最適解となり，そうでなくとも最適解にほとんど劣らない解）を高速に得ることの方が重要である場合が多い[1]．

一方，鉄道サービスを運営する立場での運賃計算では，間違いは許されず，必ず最適解（正しいという保証がある解．多くの場合，最も安い運賃となる経路）を得る必要がある．その運賃計算の方法については，鉄道関係各社が，独自の工夫を凝らして行っていることから，その具体的な内容が公表されることがない．野末 [8] は，運賃計算方式のモデルとして，複数の利用可能な経路を生成し，その運賃をすべて計算して，最も安い運賃を選択することを提案している．そして，併算割引等の運賃計算ルールを自由に定義できるよう，宣言型汎用運賃計算システムを構築した．

しかし，与えられた2駅に対して膨大な数の経路列挙が必要となる場合，運賃計算アルゴリズムの速さは列挙アルゴリズムの速さ，例えば，K-shortest paths[5][6][4][9][1] のアルゴリズムの速さに大きく依存することになる．特に，まとめて大量の運賃計算を行う場合は，その影響が大きい．

これに対して，我々は，最適解を少ない手間で見つけることにより，列挙や比較をほとんど必要としない運賃計算アルゴリズム構築を目指す．1社内の運賃計算については，JR6社と全国私鉄の運賃計算が，すでに可能になっていることを論文 [7] で述べた．その中でも，首都圏エリアの IC カード乗車券 [38][33] が利用可能になる範囲に含まれる JR 東日本 510 駅を対象に構築した「ダイクストラ法 [2] に基づくアルゴリズム」は，従来は数時間を必要としていた 510×509=約 25 万の運賃計算を数秒 (最も工夫した場合には，一般的な PC 上であっても 1 秒未満) で行うことに成功した．本論文では，これら各社の運賃計算結果を利用し，首都圏エリアの複数の会社を含む鉄道ネットワークを対象にして行った運賃計算の結果を報告する．物理的な構造に基づくネットワークを利用する代わりに，運賃計算用のネットワークを構築して利用することにより，ダイクストラ法を有効に利用できるアルゴリズムを構築

[1]市販ソフトウェア 5 種類と Web 上のシステム 27 種類のそれぞれを利用して，吉祥寺・西船橋間の最安運賃を求めた結果を付録に付ける（2007 年 5 月 23 日現在の調査結果）．

し，運賃計算の高速化を図った．

2. 対象とする鉄道ネットワーク

本研究で利用したデータは，実際に改札機などに使用されている検証済みデータを基に，2005年2月18日，日本信号(株)が研究用に作成したものである．対象とする鉄道ネットワーク（以降，関東IC範囲と呼ぶ）は，首都圏エリアのICカード乗車券が利用可能な範囲のうち，首都圏新都心鉄道（つくばエクスプレス）運用開始前の26社のネットワークと，その連絡関係を含んでいる．図1に対象ネットワークの大まかな形を示し，表1に各社の駅数を示す[2]．

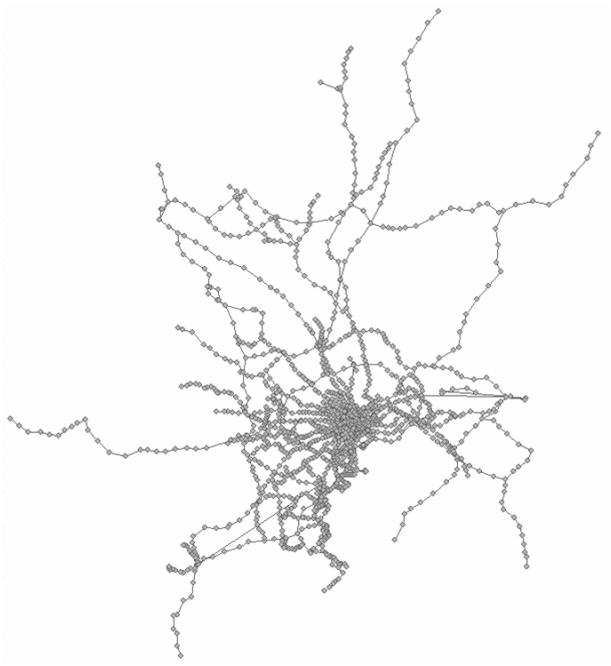

図 1: 鉄道ネットワーク（関東IC範囲）

鉄道ネットワークは，各鉄道会社が独自に形成している物理的ネットワークを，連絡可能であると定義された駅間でつなげたものと考えることができる．この連絡関係は，駅間の地理的関係に基づいて定義されるが，連絡（乗換え）のために歩ける距離の上限だけでなく，運賃計算の都合で決められたルールも考慮しなくてはならないことから，利用者にとって距離的に連絡可能であっても，連絡不可能と定義される場合もある．また，1節で挙げたJR有楽町駅，地下鉄有楽町駅，都営日比谷駅のように，連絡可能な駅と不可能な駅が混在する地域では，これらの駅を1つのノードとして扱うことが不可能である．そこで，本研究においては，連絡関係のある駅間に架空の駅（以降，架空連絡駅と呼ぶ）と距離ゼロのアークを導入することで，連絡可能，連絡不可能を表すことにした．

図2に，架空連絡駅を導入した鉄道ネットワークの例（簡略化し，各鉄道会社が4〜5駅しか持っていない例）を示す[3]．ここで，A社のa2駅とB社のb2駅，B社のb2駅とC社のc2駅は連絡可能だが，a2駅とc2駅は連絡不可能であることから，この地域（b2駅周辺）には2つの架空連絡駅が存在することになる．

[2]2007年3月18日現在と比べると，首都圏新都心鉄道（つくばエクスプレス）の全20駅と，東武鉄道の「流山おおたかの森」，ゆりかもめの「有明テニスの森」「市場前」「新豊洲」「豊洲」の合計25駅が少ない．
[3]本論文中では，簡略化のため，双方向の2本のアークを方向性のないエッジ1本で表現している．

表 1: 関東 IC 範囲 26 社の駅数（2005 年 2 月 18 日現在)

会社名	駅数	会社名	駅数
東日本旅客鉄道	510	江ノ島電鉄	15
箱根登山鉄道	11	相模鉄道	25
横浜市交通局	32	東武鉄道	203
京成電鉄	64	新京成電鉄	24
西武鉄道	92	京王電鉄	69
東京急行電鉄	87	京浜急行電鉄	72
小田急電鉄	70	東京地下鉄	139
東京都交通局	99	東京モノレール	10
北総鉄道	15	伊豆箱根鉄道	12
埼玉新都市交通	13	横浜新都市交通	14
ゆりかもめ	12	多摩モノレール	19
東京臨海高速鉄道	8	東葉高速鉄道	9
埼玉高速鉄道	8	横浜高速鉄道	6
		合計	1638

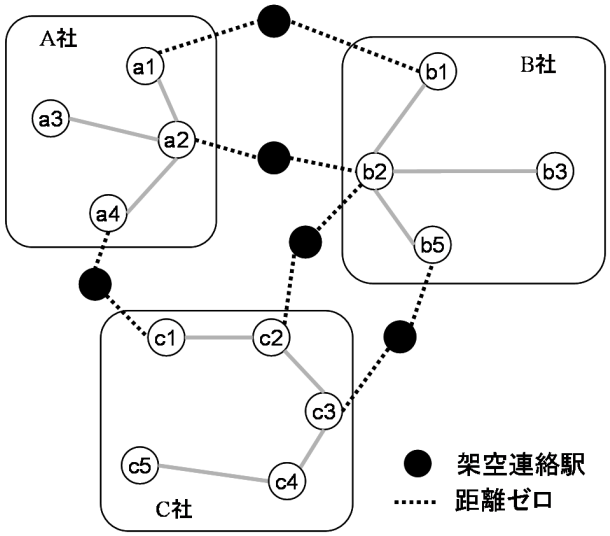

図 2: 架空連絡駅を含む鉄道ネットワーク

このようなケースも含め，本研究で対象とする関東IC範囲には，会社間で連絡できる関係が175ヶ所[4]定義されているため，架空連絡駅が175駅存在することとなった．よって，関東IC範囲に存在するノード数は，1638+175の1813となる．表2に本論文で扱う鉄道ネットワークの大きさを示す.

表 2: 関東IC範囲の大きさ

駅数	1638
架空連絡駅数	175
ノード数	1813
アーク数	4090

3. ショーテストパス問題を解くための工夫

2節で述べたネットワーク上で，ショーテストパス問題を解いたとき，連絡可能，連絡不可能な駅が混在する地域において，この定義を守れない場合がある．ここでのショーテストパスとは，利用した線路の距離の和が最も短い経路のこととする．

2節の図2の例を利用して説明すると，a2駅からb2駅に乗換えるものの，B社を利用することなく，b2駅から連絡可能なc2駅に乗換える経路を与えてしまう場合が起こり得るのである．つまり，結果的に，連絡不可能と定義されているa2駅とc2駅の乗換えを許すことになってしまう．

そこで，b2駅のように利用することなく通過する可能性がある駅（関東IC範囲には16駅存在）と，それとコストゼロでつながるa2駅やc2駅のような駅（11駅存在）のそれぞれにダミーノードを導入し，元の駅（今後，元ノードと呼ぶ）の間の連絡アーク（距離ゼロのアーク）をダミーノードに向けてつなぎなおし，ダミーノードからは自社内の隣の駅にのみアークを加えるといったネットワークの修正を行い（図3を参照[5]），ダイクストラ法等でショーテストパスを求める．

１つの駅を２つのノードで表現する関係上，その両方のノードを経路に含む（現実にはループを含む）経路を避けられなかった場合には，一方のノードの利用を禁止して2通りのショーテストパス問題を解き直し，その結果を比較して短い方の経路を採用する．

元のネットワーク上では，距離ゼロのアークを利用しようとして現実には許されない経路を与える確率が高く（関東IC範囲では全経路の14.38%と）なるが，ここで提案する方法では，ダミーノードから出るアークにコスト（隣の駅までの距離）が設定されるため，ループを構成する経路は生じにくくなっている（全経路の0.06%）．つまり，1回目に与えられた解が最適解となる確率が高いので解き直す手間を大きく削減できることになる．

本論文では，1つの駅が2つのノードで表されているネットワークに対し「ダイクストラ法で得られた解にループが存在した場合に，ループの原因になる駅の片方のノードを禁止しながら2回ダイクストラ法を適用する方法」を，ダイクストラM法（a modification of the Dijkstra's algorithm）と呼ぶことにする．

[4]3つ以上の会社が相互にすべて連絡可能なら，1ヶ所として数える．
[5]a1駅とb1駅，a4駅とc1駅，b5駅とc3駅の連絡関係は省略してある．

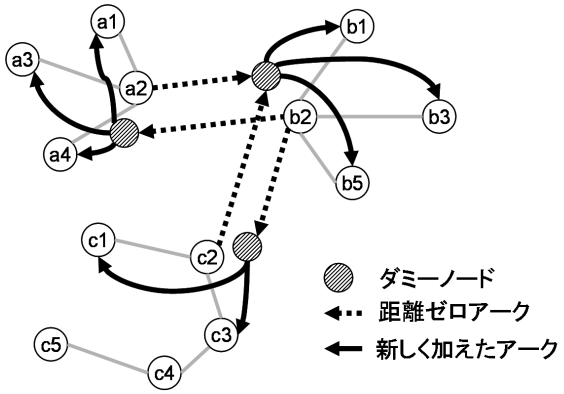

図 3: ショーテストパス問題を解くためのネットワーク修正

4. 運賃計算のためのネットワーク

与えられた 2 駅間の運賃計算においては，複数存在する可能経路の中で最も安い運賃となる経路を見つけなくてはならない．しかし，1 節で述べたように，距離的な意味合いでのショーテストパスを求めても最安運賃経路が得られるわけではない．また，2 節の図 2 に示したネットワークや 3 節で述べた修正ネットワーク上で，アークのコストを距離ではなく，駅間の運賃を設定してショーテストパスを求めたとしても，1 つの経路の運賃は，経路を構成するアークのコストの和では表せない（例えば，JR 吉祥寺駅と西荻窪駅を結ぶアークのコストが 130 円，西荻窪駅と荻窪駅を結ぶアークのコストが 130 円であっても，吉祥寺駅から荻窪駅までの運賃は 130+130=260 円ではなく，150 円である）ので，運賃計算には利用することができない．

よって，鉄道運賃計算においては，これまでに報告されてきたように，可能経路を出来るだけ列挙し，それぞれの経路の運賃を計算して，その結果の比較により最も安い経路を得ることが一般的とされてきた [3][8]．具体的には，2 節や 3 節に示したような物理的構造に基づくネットワーク上で，K-shortest paths 用のアルゴリズム等を利用するなどして，出来るだけ高速に経路を列挙することになるが，何を基準に列挙するか（例えば，距離を基準にするのか，所要時間を基準にするのか），いつまで列挙し続けるか（例えば，経路列挙数に上限をつけるか，評価基準に上限をつけるか，もしくは全列挙など），といった設定の仕方によっては，最適である保証がない場合も起こり得る．また，最適であることの保証を得るために多くの時間を費やす必要があることも容易に想像できる．さらに，1 つの運賃（与えられた 2 駅間のみの運賃）の計算であれば，その計算時間の大きさはあまり問題にならないと思われるが，例えば，関東 IC 範囲 1638 駅の間の 1638 × 1637=約 260 万の運賃計算を，すべて間違いなく行うためには非常に多くの時間を費やさなければならないことになる（1 つの運賃の計算が 0.001 秒でできたとしても，全部の計算に約 40 分弱かかってしまう）．

そこで，我々は，出来る限り早く最適である保証のある経路を見つけ出すことにより，列挙する経路数を少なくすること（出来れば 1 つ）を目指した．そして，ダイクストラ法を最大限に利用することを念頭に，扱うネットワークの構造を根本的に作り直すことを考えた．

鉄道を利用する場合，基本的に 1 社内の経路は，その会社の路線群利用のための始点（利用を開始する駅）と，終点（利用を終了する駅）の，2 駅間のものになる．つまり，複数の鉄道会社を利用するような経路でも，鉄道会社ごとにその経路を区切れば，必ず各会社内の始点と終点の 2 駅の組合せのセットになる．

1 社内の全駅間の運賃計算については，我々の論文 [7] で，その高速化について報告したが，ここでは，その結果として与えられたものを利用することにし，この各社ごとに与えられた全 2 駅間の運賃対応表（会社内の駅数×駅数のマトリックス）のことを以降，四角表[6] と呼ぶことにする．そして，これら四角表の要素をネットワーク上のアーク，その数値をアークのコストとして表されるネットワークを構築することを考える（図 4 参照）．こうすることによって，1 社内では 1 つのアークのみ利用して，複数社を乗り継いで利用した場合の経路の運賃計算は，利用したアークのコストの和で表されることになり，ダイクストラ法の利用の可能性を引き出すことになる．このように，運賃の意味を持つアークで構成されたネットワークを，2 節で述べたような物理的構造に基づくネットワークと区別し，今後，総称して（広義での）**Farenet**，もしくは運賃計算ネットワークと呼ぶことにする．

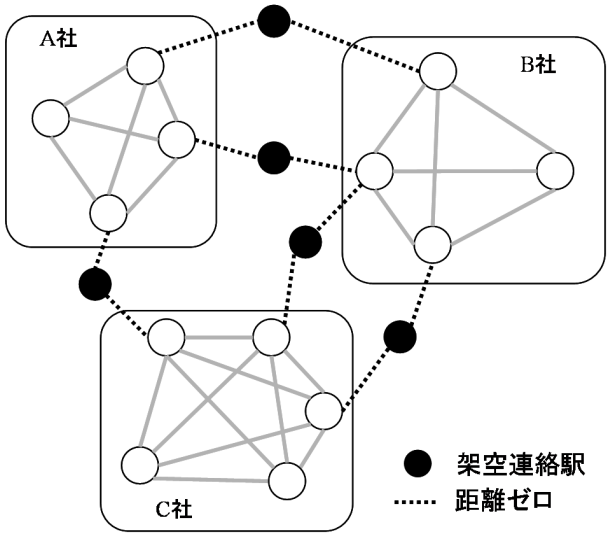

図 4: 運賃に対応したアークを持つネットワーク

また，複数社利用経路の運賃計算を対象にしているので，経路の発駅や着駅を含む会社以外の，経由のみの会社では，他社との連絡を持つ駅間をつなぐアークのみが利用されることに着目し，Farenet 上の他社との連絡を持つ駅だけ残したネットワーク利用も考えた．

物理的ネットワーク，全駅を含む Farenet，連絡可能駅のみ含む Farenet，それぞれについて 1 社分だけ表したものを，図 5 に示す．

図 5 の (b) のネットワークのように，全ての駅を含んだまま架空連絡駅や運賃ゼロのアークで結ばれたネットワーク全体を，狭義での **Farenet** と呼ぶことにする．

同様に，図 5 の (c) のネットワークが複数社結ばれたネットワークを，**Simplenet** (Simplified Farenet) と呼ぶことにするが，本論文で扱うデータにおいては，3 節で述べたネットワークの修正を行った方がショーテストパスを求める際に効率がよかったため，この修正を行った後のネットワークを，Simplenet と呼んで話を進める（図 6 参照）．

この Simplenet に対し，3 節のダイクストラ M 法で，与えられた 2 駅間のショーテストパス問題を解いた場合，アークのコストに 1 社内利用の最安運賃が設定されているため，ショーテストパスとして得られた経路のコストが最安運賃になる仕組みになっている．

[6] この表の対角線で切った 2 つの三角部分のことを一般に三角表という．

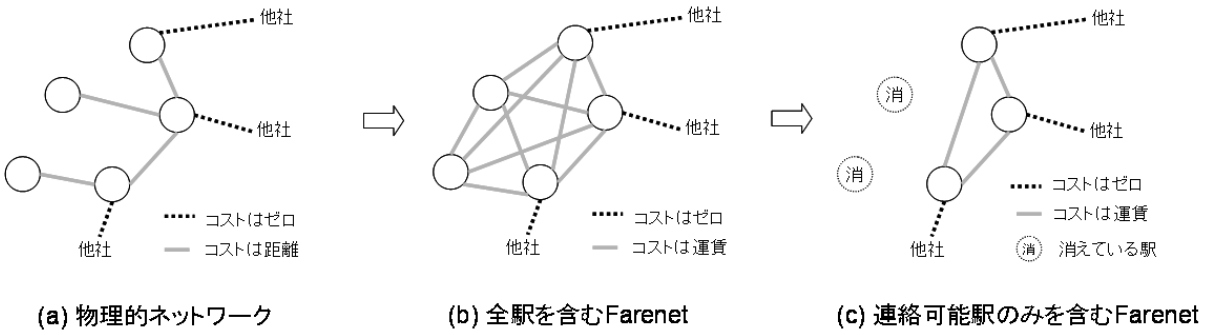

図 5: 1 社内のネットワーク

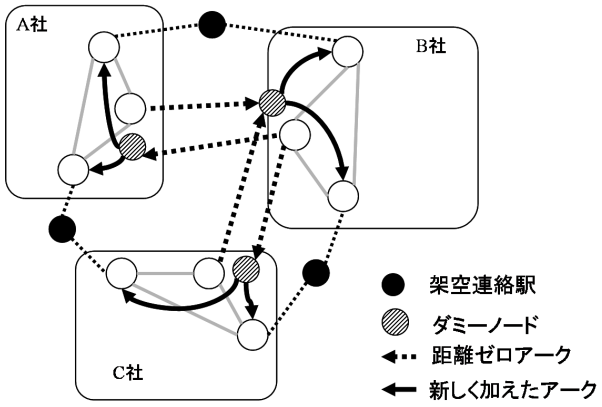

図 6: ネットワーク修正後の Simplenet

運賃計算対象の発駅がこのネットワーク上に存在しない場合は，その駅をネットワーク上に加え，その駅から自社の他の駅に向けて，運賃のコストを持ったアークを引く．同様に，着駅が存在しない場合には，その駅をネットワーク上に加え，自社の他の駅からその駅に向けて運賃のコストを持ったアークを引く．もしも発駅と着駅の両方とも同じ会社内に加えられた場合には，発駅から着駅に向けて運賃のコストを持つアークを引けばよい．

つまり，この Simplenet を構築したことにより，5 節に述べる問題点は残るものの，基本的には，わずかな工夫を加えながらショーテストパス問題をダイクストラ法で解くことで，運賃計算ができる仕組みを作り上げたことになる．

表 3 に，関東 IC 範囲における Simplenet の大きさ（ネットワーク修正前と修正後）を示す．

表 3: 関東 IC 範囲における Simplenet(ネットワーク修正前と修正後) の大きさ

修正前	
駅数	358
架空連絡駅数	175
ノード数	533
アーク数	20168

修正後	
駅数	358
ダミーノード数	27
架空連絡駅数	161
ノード数	546
アーク数	21814

5. 運賃計算を複雑にする 2 つの要因

4 節で説明した Simplenet を扱って，与えられた 2 駅間の最安運賃経路を求める際，1 社内の 2 駅間の運賃（四角表の値）の間の関係や割引サービスの影響により，実際の運賃と異なる場合が存在することを考慮しなければならない．以下にそれぞれの要因について述べる．

5.1. 1 社内の運賃設定

Farenet や Simplenet では，1 社内における運賃が各社で設定した四角表の値に従うようにするため，1 社内での複数アークの連続利用を対象としていない．また，四角表における i 駅と j 駅の間の運賃を c_{ij} とすると，任意の i，j，k，3 駅間の運賃（図 7 参照）が，

$$c_{ij} + c_{jk} > c_{ik} \tag{5.1}$$

という関係を保っていれば，1 社内複数アークの連続利用を含む経路が最安運賃経路として得られることもない．しかし，現実の運賃設定（四角表の値）では，この関係を満たす保証はなく，実際の運賃を計算するためには，1 社内のアークの連続利用を回避できる仕組みを作らなければならない．

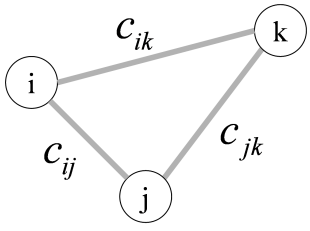

図 7: 3 駅間の運賃

5.2. 併算割引

図 8 に，Simplenet を利用して，JR 三鷹駅と京王井の頭線高井戸駅の間の最安運賃経路を求めた結果を示す.

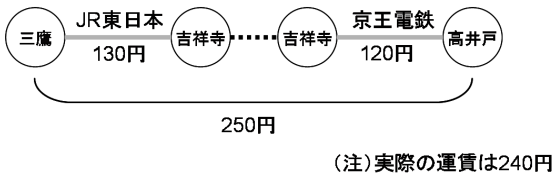

図 8: Simplenet を利用して得られた三鷹駅・高井戸駅間の最安運賃経路

この 2 駅間の運賃は，JR 東日本の三鷹駅と吉祥寺駅間の運賃 130 円と，京王電鉄の吉祥寺駅と高井戸駅間の運賃 120 円が足された 250 円になる．しかし，実際の最安運賃は 240 円に設定されていて，Simplenet で得た最安運賃 250 円より 10 円安い．このようなことが起こる原因は，会社間を乗り継いだときに発生する割引制度によるものである．複数会社を短い区間ずつ利用する場合，初乗り運賃が重なることから運賃が高くなる傾向がある．これを解消するために，会社間で導入されている上記のような割引を併算割引（以降，併割）という．図 9 は，吉祥寺駅で JR 東日本と京王電鉄の間を乗り継いだ場合の併割適用範囲を示したものである.

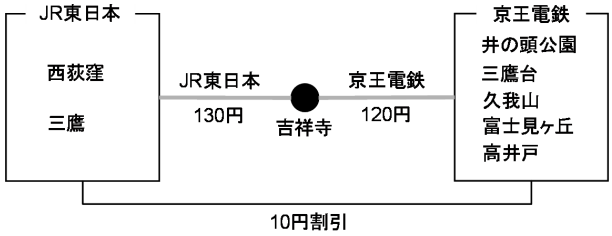

図 9: 吉祥寺駅連絡の併割適用範囲

この例のように，一般的には，対象駅（乗換え駅）からの範囲を定め，その範囲に定められている駅で電車を乗り降りした場合にのみ，割引額が適用されるように定義されている.

また，東京地下鉄（メトロ）と都営地下鉄の間では，図 10 に示すような特別な併割も存在する.

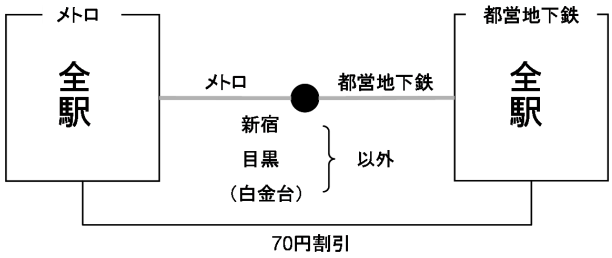

図 10: メトロと都営地下鉄の併割適用範囲

この併割ではメトロと都営地下鉄では全駅が対象となり，新宿駅と目黒駅[7]以外で乗換えた場合に 70 円割引が適用されるように定義されている．図 9，図 10 のような 2 社間の併割の他にも，現在，3 社間の併割，4 社間の併割まで存在する．表 4 に，関東 IC 範囲に存在する併割の数を示す.

表 4: 関東 IC 範囲に存在する併割数

	併割数	対象区間数
2 社間の併割	84	15317
3 社間の併割	4	1926
4 社間の併割	4	1960

また，与えられた経路が複数の併割適用対象になっている場合，特に対象範囲が重なっている場合には，結果としての割引額が最も高くなるよう，1 つ，もしくは複数の併割を選択して適用しなければならない[8]．図 11 に示した経路は，2 つの併割が適用対象となっているものである．

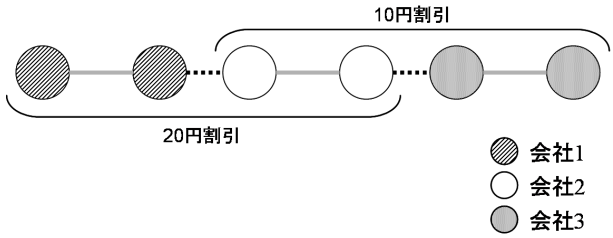

図 11: 複数の併割が適用対象となる経路（適用範囲が重なっている例）

この経路では，会社 1 と会社 2 を乗り継いだ場合に適用される 20 円割引の併割と，会社 2 と会社 3 を乗り継いだ場合に適用される 10 円割引の併割が重なっているが，20 円割引が適用されることになる．一方，複数の併割適用範囲が重なっていない場合には，その全てが適用される．

6. 運賃計算ネットワーク

5 節で述べた四角表の要素間の関係や併割の存在を考慮できる運賃計算のために，4 節で述べた Farenet の考え（運賃アークでネットワークを構成）を基礎とする新たなネットワークの構築を考えた．そのベースとなるネットワークは，全ての駅を含む Farenet であっても，他社に連絡可能な駅のみ含む Simplenet でも可能と考えたが，便宜上（他社に連絡可能でない駅が発駅になったり着駅になった場合の説明を省くため），全ての駅を含む Farenet を利用して説明することにする．

[7]利用者にとっては乗換可能だが，事業者間の連絡運輸規定で連絡駅と定義されていない白金台駅での乗り換えも割引対象外となる．
[8]併割の優先順位については，公になっていないものが存在するかも知れないが，本研究においては，この前提で議論を進めた．

6.1. 1社内における複数アーク連続利用の回避

全ての駅を含み，1社内の全駅間にアークが導入されているFarenetに対し，1社内でのアークの連続利用を避けるよう，各駅にダミーノードを導入し，その会社利用の始点（発駅，もしくは他社からの連絡を受けるもの）とし，元ノードをその会社での終点（着駅，もしくは他社へ連絡するもの）となるようにアークを付け替えることを考えた．

図12の左側に示すFarenetにおける各駅に対し，ダミーノードを導入し，アークを付け替えたものが右側の図である．各ダミーノードからは，社内の他の駅の元ノード全てにアークが引かれた状態を示している．この右側の構造を持つネットワークにおいて，ダミーノードを発駅，元ノードを着駅として，ショーテストパス問題を解けば，ダミーノードからしか社内の他の駅に向かうアークが出ていないため，1社内での複数アークの連続利用を回避できるようになる．

また，これらのダミーノードを導入したことにより，他社と連絡関係を持つ駅の元ノードから，連絡先の駅のダミーノードに直接，運賃ゼロのアークを引くことで，2節で述べた許されない乗換えを避けることが出来る（架空連絡駅の必要性もなくなる）．

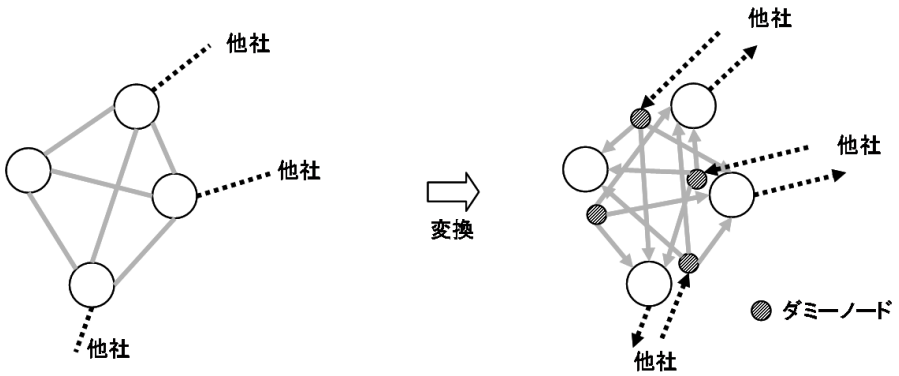

図 12: 1社内ネットワークの変換

図12の右側に示した1つの会社のネットワークが，複数社合わさったネットワーク全体を **Directnet** (Direct Farenet) と呼ぶことにする．図13にDirectnetの形を示すとともに，関東IC範囲におけるDirectnetの大きさを表5に示す．

表 5: 関東IC範囲におけるDirectnetの大きさ

駅数	1638
ダミーノード	1638
架空連絡駅数	0
ノード数	3276
アーク数	353166

Directnetでは，全ての駅にダミーノードが導入されているため，同じ駅を2度（元ノードとダミーノードの両方を）利用する経路を与えてしまう危険がある．その例[9]を図14に示す．

[9]この例は，説明のため，擬似的に作成したものである．

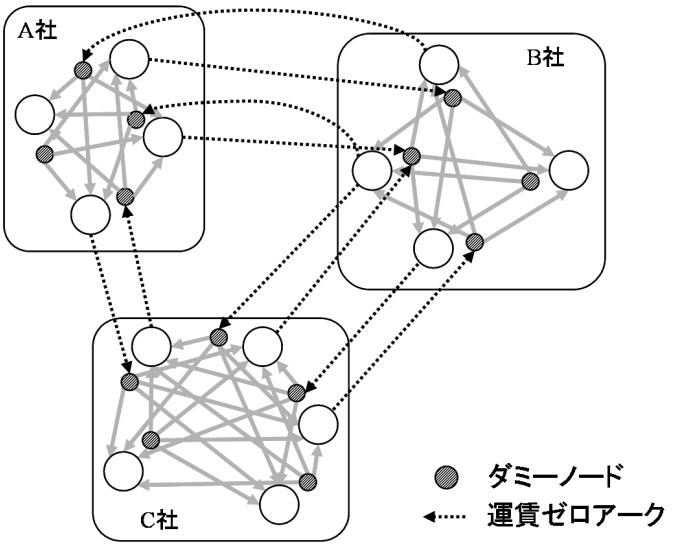

図 13: Directnet：1 社内での複数アーク連続利用を回避するネットワーク

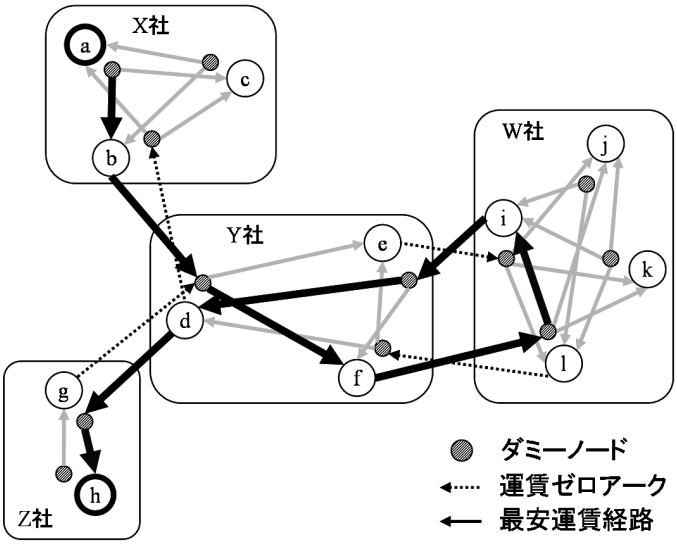

図 14: 同じ駅を 2 度（元ノードとダミーノードの両方を）利用する経路

図 14 は，Directnet 上，X 社の a 駅から Z 社の h 駅までの最安運賃経路をダイクストラ法で得た結果である．この経路は，a 駅のダミーノード，b 駅の元ノード，d 駅のダミーノード，f 駅の元ノード，l 駅のダミーノード，i 駅の元ノード，e 駅のダミーノード，d 駅の元ノード，g 駅のダミーノード，h 駅の元ノードという順で巡る．しかし，d 駅のダミーノードと元ノードの両方を利用していることから，同じ駅を 2 回通るループを含む経路になっている．これは，X 社の b 駅から Z 社の g 駅の乗換えが許されていない（アークが存在しない）ために起きた例であるが，もっと多くの会社（この図の範囲外の会社）を利用してでも，ループを含まない経路を見つけなくてはならない．

ループを含む経路を回避するためには，3 節のダイクストラ M 法の利用を考える．しかし，関東 IC 範囲における Directnet 上では，ダイクストラ法で得た解において，同じ駅を 2 回通る経路は存在したものの，発駅と着駅のみで起きていた[10]ため，発駅の元ノードと着駅のダミーノードの利用を禁止することで，ループを含む経路を全て回避することができた．

6.2. 併算割引への対応

併割を考慮した最安運賃を得る方法として，2 つのアイデアが考えられた．最終的には，その両方を採用したので，それぞれを以下に示す．

1 つ目は，Directnet 上での最安運賃経路に対し，併割適用による，もっと安い経路が存在する可能性を考慮し，複数の経路を列挙して併割適用後の運賃を計算・比較していく方法である．複数経路列挙にあたっては，Directnet 上，与えられた 2 駅間の併割適用前の運賃の安い順に列挙することにする．Directnet 上，ダイクストラ法で得た最安運賃を基準とし，併割適用後にその値以下になる可能性のある運賃の併割適用前運賃の上限を求め，併割適用前運賃が上限より安い経路を Directnet 上で列挙する．以下に，上限計算の方法を示す．

先ず，併割適用が定義されている各 2 駅間について，併割が適用される前の運賃をすべて明らかにしておく．そして，割引額（10 円，20 円，...）ごとに，適用対象の区間の併割適用前運賃を調べ，その最小値を求める．表 6 に，関東 IC 範囲で定義されている，2 社対象併割，3 社対象併割，4 社対象併割，それぞれについて，割引額の値ごとに，併割適用対象運賃の最小と，その適用後の運賃を示す．

例えば，併割適用前に 240 円以上の運賃ならば，20 円以上の割引の可能性を考慮しなければならないことになる．また，1 つの経路に対して，複数の併割適用の可能性を考慮しなくてはいけないので，表 6 の値の組合せを行い，併割適用前にいくら以上の運賃のものに対して，最大いくらの割引を考慮すべきかがわかるよう，表 7 を作成した[11]．ここには，考慮すべき割引額 z の値，併割適用対象運賃の最小 $g(z)$ と適用後の $g(z)-z$ を示してある．

Directnet 上で最安運賃経路として得られた経路の運賃を x 円とすると，この表を使用して $x > g(z)-z$ となる最大の割引額 z を求め，$x+z$ の金額を上限として採用する．例えば，Directnet 上の最安運賃が 250 円の場合，$x > g(z)-z$ となる最大の割引額 z は 20 円となり，上限金額は，250 円 +20 円 =270 円となる．

そして，この方法で求めた上限金額より運賃の安い複数の経路を，K-shortest paths 用の

[10]例えば，図 14 の d 駅から h 駅の最安運賃経路を求めようとした場合に起きていた．Y 社の d 駅を発駅とする場合，Y 社の路線利用からスタートする経路を前提としているため（d 駅のダミーノードが始点となり），Y 社の路線を利用することなく Z 社の g 駅から h 駅に向かう経路は対象外となる．つまり，図 14 のネットワークの例では，g 駅から h 駅の最安運賃経路は存在していても，d 駅から h 駅の実行可能経路は（ループを避けるために，d 駅の元ノードの利用を禁止するので）存在しないことになる．

[11]現実に起こりうる数をカバーできる十分大きな数の組合せを考える．ここの例では，12 回の連絡を許した場合までを考慮し，6 種類の併割が同時に起こった場合まで考慮して作成してある．

表 6: 割引額ごとの併割適用前後の最小運賃

	2 社対象併割		3 社対象併割		4 社対象併割	
割引額	適用前	適用後	適用前	適用後	適用前	適用後
10	250	240				
20	240	220	400	380		
30	370	340	550	520		
40			850	810		
60	610	550				
70	320	250				
80					1080	1000
90			1650	1560	710	620
100					1010	910

表 7: 割引額と併割適用対象運賃の最小と割引適用後の運賃

割引額 z	適用前 $g(z)$	適用後 $g(z)-z$
20	240	220
70	320	250
90	560	470
140	640	500
160	880	720
210	960	750
230	1200	970
280	1280	1000
300	1520	1220
350	1600	1250
370	1840	1470
420	1920	1500

アルゴリズム（[5][6][4][9][1] 等）を利用して列挙する．それぞれの経路に対して併割適用の可能性をチェックして，適用対象であれば適用後の運賃を計算し，これまでに与えられている適用後最小運賃より小さければ，適用後最小運賃を更新する．このように，上限を設定して経路列挙を行うことは，列挙経路数をさほど大きくしないと考えられる．ただし，列挙した多く（または全て）の経路が併割適用対象でなかった場合など，列挙が無駄に終わる可能性を含んでいる．

そこで，2 つ目のアイデアとして，併割を考慮できるアークの導入を考えた．具体的には，併割が適用される 2 駅間を，直接アークでつなげることにより，併割適用前と適用後を意識せずに最安運賃経路を得ることができるネットワークを構築した．

図 15 は，Directnet 上で，併割が適用される 2 駅間全てに直接アークを導入したものである．導入されたアークを併割アーク，そのネットワーク全体を **DDnet** (Direct and Discount Farenet) もしくは併割考慮ネットワークと呼ぶことにする．

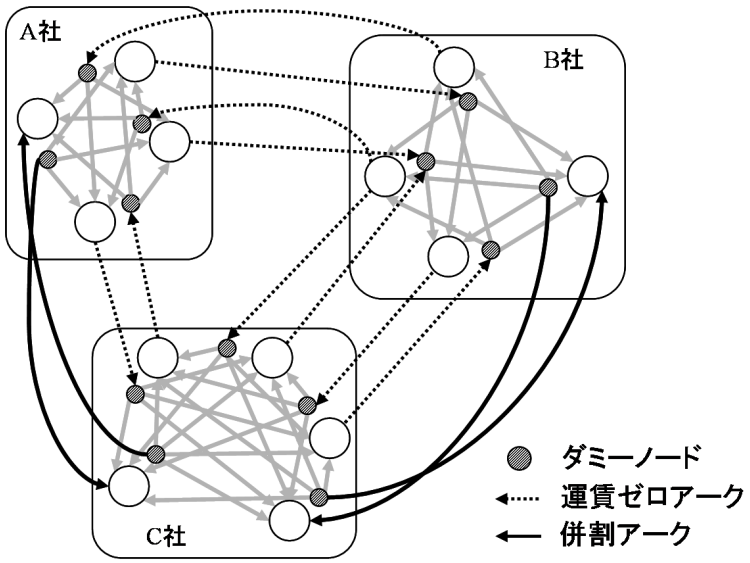

図 15: DDnet：併割考慮ネットワーク

関東 IC 範囲における DDnet の大きさを表 8 に示す．併割アークは，図 16 に示すように，併割適用後の値を設定する．これにより，最安運賃経路を求める際，併割アークを通る経路を得る仕組みができた．

表 8: 関東 IC 範囲における DDnet 大きさ

駅数	1638
ダミーノード	1638
架空連絡駅数	0
ノード数	3276
アーク数	391564

構築した DDnet では，Directnet と同様に，図 14 で示したような同じ駅を 2 度利用する（ループする）経路が得られる可能性がある．このことに加え，併割アークを導入したことから，以下に示す経路が生じる可能性もある．

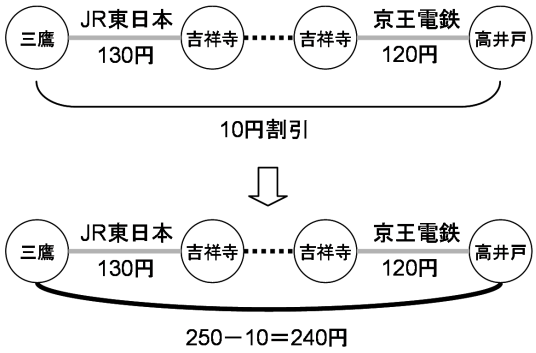

図 16: 併割アークのコスト設定

図 17 は，DDnet 上で，会社 1 の C 駅から会社 4 の T 駅について得られた最安運賃経路である．この経路は，会社 2 の H 駅と会社 4 の T 駅の間を結ぶ併割アークを利用する経路であるが，この併割アークは，H 駅と T 駅の間に定義された「4 社間の併割対象区間」の利用を表す．よって，実際の経路は，図 17 の下の図に表されるような C 駅を 2 度通る経路となる．

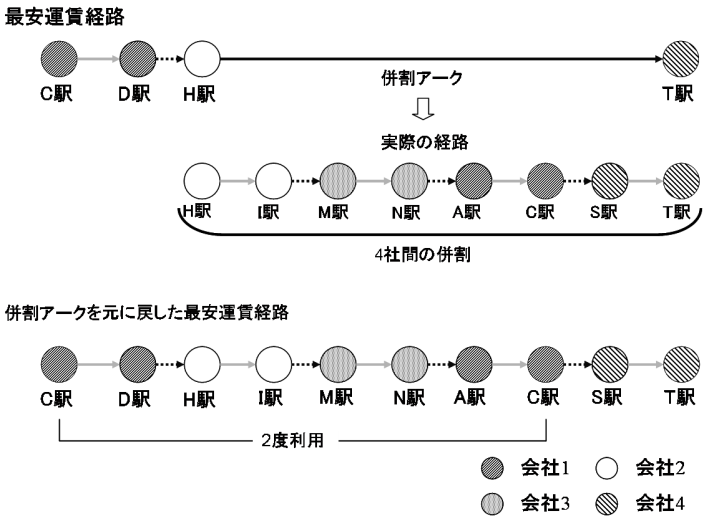

図 17: 同じ駅（元ノードとダミーノード両方を）利用する経路

併割アークを利用することで生じるループを含む経路を回避するためには，1 つ目のアイデアとして挙げたような，ある程度の経路列挙と比較が必要であるものの，ループ経路の出現率は極めて小さい．関東 IC 範囲における DDnet においては，1638 駅全ての 2 駅間の組合せ（1638 × 1637=2681406 組）に対して，8 組しかループ経路は得られなかった．

ここで示したように，ループを含む経路は，ダミーノードの導入や併割アークの導入によって引き起こされるが，その出現率はデータ依存である．しかし，元々の運賃設定の考え方（利用した距離が長くなればなるほど運賃は高くなる）に基づけば，多くの鉄道ネットワークにおいて，ループの出現率は非常に低いものと考えられる．

7. 最安運賃計算アルゴリズム

6 節で構築した Directnet と DDnet を利用して，与えられた 2 駅間の最安運賃を計算する手順について述べる．

先ず，与えられた2駅間に対して，DDnet上で最安運賃経路をダイクストラ法で求める．得られた経路に併割アークが含まれていない場合は，その経路の運賃を最安運賃として採用するが，併割アークが含まれている場合には，ループの存在のチェックが必要である．もし，併割アークを通ることでループが起こっているならば，複数（実際には非常に少ない数）の経路列挙が必要になる．経路列挙についてはDirectnet上で行い，列挙した複数経路の中で（併割を考慮した実際の）最安運賃を求める．その際，Directnet上で得られた最安運賃を基準とし，併割を適用すると基準値以下になる運賃の上限を求め，その上限より安い経路をDirectnet上で列挙する．上限の計算方法としては，6.2節で説明した方法を利用する．

また，DDnet上で得られたループを含む経路の運賃は，最安運賃の下限値と考えられる．Directnet上で経路列挙を行う際に，得られた経路の運賃が下限値と等しくなった場合には，そこで処理を終了する．

以上のことを考慮した最安運賃計算アルゴリズムの流れを以下に示す．

記号

s, $\bar{s}$：発駅の元ノードとダミーノード
t, $\bar{t}$：着駅の元ノードとダミーノード
$g(z)$：割引額zが適用対象となる適用前運賃の最小値
UB：最安運賃の併割適用前金額の上限
LB：最安運賃の下限
min：発着2駅間の最安運賃の計算結果保存用の変数

アルゴリズム

DDnet 上，s と $\bar{t}$ の利用を禁止して，$\bar{s}$,t 間のショーテストパス **path0** を求める．
if（**path0** 中に併割アークが含まれない）
　■ **path0** の運賃を min に格納し，処理終了．
else
　path0 の運賃を LB に格納する．
　Directnet 上，s と $\bar{t}$ の利用を禁止し，$\bar{s}$,t 間のショーテストパス **path1** を求める．
　if（**path1** の運賃と LB が同じ金額）
　　■ **path1** の運賃を min に格納し，処理終了．
　else
　　path1 の運賃 $> g(z) - z$ となる最大割引額zを求める．
　　UB =path1 の運賃+z とする．
　　UB 未満の経路を，運賃 LB の経路が見つからない限り，全て列挙する．
　　if（運賃 LB の経路が見つかった）
　　　■ LB を min に格納し，処理終了．
　　else
　　　■列挙した経路において，併割考慮後の最安運賃を min に格納し，処理終了．

関東IC範囲を対象にした場合には，DDnetやDirectnet上で最安運賃経路（ショーテストパス）を求める際に，発駅の元ノードと着駅のダミーノードの利用を禁止すればダミーノード挿入によるループをすべて回避できることを6.1節で述べた．よって，ここで示したアルゴリズムは，そのことが前提で書かれている．しかし，扱うデータによっては，ループ

を含む経路を回避できない場合が考えられるので，その場合には，$\bar{s}$,t 間のショーテストパスをダイクストラ M 法で求めることで対応する．

対象範囲の全 2 駅間の最安運賃（ならびにその経路）をまとめて計算することを前提とし，提案アルゴリズムの実装において，1 駅からその他の駅へのショーテストパスを全て求め，その後の処理を各 2 駅間で行うようしたところ，関東 IC 範囲にある 1638 駅全 2 駅間の最安運賃計算を 20.3 秒で行うことができた（CPU : Xeon 3.6GHz，メモリ : 2G，OS : Windows XP）．この時間の多くは，ループを含むか否かのチェックに費やされていることもわかっているので，これらの工夫によってさらに時間削減が可能と考える．

鉄道運賃計算システムを扱っている現場において，Directnet や DDnet を利用して，提案アルゴリズムで運賃計算を行ってもらったところ，首都圏エリア IC カード乗車券（Suica・Pasmo）では複数の会社を乗り継いで利用する場合，改札を出場しない 4 社以内の最安運賃経路の算出が必要とされる (最安運賃経路が 5 社以上の場合でも，4 社以内の経路を算出する必要がある[12]) ことをはじめ，運賃計算上の様々な特例に対応して手を加える必要があったものの，正確な運賃計算が可能であることがわかった．また，この確認計算や現在の運賃計算システムに適用する際に加えた修正の詳細については，別報にて報告の予定であるが，現時点では，従来のシステムで数 10 分かかっていた関東 IC 範囲の全 2 駅間の運賃計算と，それに伴う様々な計算を 2 分弱で行えるようになっている．提案手法は運賃計算上の特例に対応させることで，関東 IC 範囲だけではなく，全国各社の連絡運賃計算に対応できると思われる．

8. おわりに

本論文では，鉄道運賃計算を正確かつ出来る限り高速に行うため，これまでの列挙に依存した探索方法に対して，ダイクストラ法が直接利用できる考え方を提案した．

具体的には，各鉄道会社内の運賃に対応したアークで構成されるネットワーク Farenet を提案し，Farenet を簡略化した Simplenet，1 社内での複数アークの連続利用を回避する Directnet，さらに併算割引をも考慮できる DDnet を構築した．また，これらのネットワークを利用したアルゴリズムを構築することにより，正確かつ高速な運賃計算を可能にした．

ここでは，最安運賃経路探索に話を絞ったが，提案したネットワークの構築法や，アルゴリズムの考え方は，運賃だけでなく別の評価尺度にも適用可能と考える．

今後は，与えられた 2 駅間に対して，最安だけではなく安い順に複数の経路を得る必要がある場合，つまり，運賃をコストとした K-shortest paths を解く必要がある場合等についても検討していきたい．

謝辞

本論文について貴重なご助言を頂いた査読委員の先生方に心より感謝致します．

参考文献

[1] D. Eppstein: Finding the k shortest paths. *SIAM Journal on Computing*, **28** (1998), 652–673.

[2] E.W. Dijkstra: A note on two problems in connection with graphs. *Numeriche Mathematics*, **1** (1959), 269–271.

[12] http:/www.pasmo.co.jp/howto/train_calculate.html

[3] 半田恵一, 田中俊明: 乗換え案内サービスにおける経路探索手法. 電子情報通信学会論文誌, **J88-D1** (2005), 1525–1533.

[4] N. Katoh, T. Ibaraki, and H. Mine: An efficient algorithm for K shortest simple paths. *Networks*, **12** (1982), 411–427.

[5] E.Q.V. Martins, M.M.B. Pascoal, and J.L.E. Santos: The K shortest paths problem. *Research Report, CISUC* (1998).

[6] E.Q.V. Martins, M.M.B. Pascoal, and J.L.E. Santos: An algorithm for ranking loopless paths. *Research Report, University de Coimbra* (1999).

[7] 森田隼史, 池上敦子, 菊地丞, 山口拓真, 中山利宏, 大倉元宏: 大幅に計算時間削減を達成した鉄道最安運賃経路探索アルゴリズム − Suica/PASMO 利用可能範囲の JR510 駅を対象とした場合−（投稿中）.

[8] 野末尚次: 道具箱としての OR. オペレーションズ・リサーチ, **52** (2007), 135–140.

[9] J.Y. Yen: Finding the K shortest loopless paths in a network. *Management Science*, **17** (1971), 712–716.

[10] ASK.jp 路線検索, アスクドットジェーピー, http://ask.jp/transit/index.asp

[11] BIGLOBE 乗換案内, NEC ビッグローブ, http://map.biglobe.ne.jp/cgi-bin/norikae.cgi?cmd=2

[12] DION 路線, KDDI, http://www.dion.ne.jp/route/

[13] えきから時刻表, ぐるなび, http://www.ekikara.jp/top.htm

[14] えきねっと時刻・乗換え案内, JR 東日本, http://www.jnavi.eki-net.com/cgi-bin/jreast/jreast.cgi

[15] 駅探, 駅前探険倶楽部, http://ekitan.com/

[16] 駅探エクスプレス, ソースネクスト, http://www.sourcenext.com/

[17] 駅すぱあと, ヴァル研究所, http://ekiworld.net/

[18] エキサイト乗換案内, エキサイト, http://www.excite.co.jp/transfer/

[19] goo 路線, NTT レゾナント, http://transit.goo.ne.jp/

[20] ハイパーダイヤ, 日立情報システムズ, http://www.hyperdia.com/

[21] いつもガイド, ゼンリンデータコム, http://www.its-mo.com/cgi-bin/cp/expwww/exp.cgi

[22] infoseek 乗換案内, 楽天, http://transfer.www.infoseek.co.jp/

[23] ジョルダン乗換案内, ジョルダン, http://www.jorudan.co.jp/

[24] JR トラベルナビゲータ, ジェイアール東日本企画, http://www.jnavi.ne.jp/

[25] livedoor 路線案内, ライブドア, http://transit.livedoor.com/

[26] マップファンウェブ, インクリメント・ピー, http://www.mapfan.com/railway/exp.cgi

[27] MSN 路線, マイクロソフト, http://transit.msn.co.jp/

[28] NAVITIME, ナビタイムジャパン, http://www.navitime.co.jp/

[29] @ nifty 路線検索, ニフティ, http://www.nifty.com/rosen/pda/

[30] 乗り換え案内 VER.5, ジョルダン, http://norikae.jorudan.co.jp/

[31] OCN 路線, NTT コミュニケーションズ, http://ocntransit.goo.ne.jp/

[32] ODN 路線検索, ソフトバンクテレコム, http://www.odn.ne.jp/cgi-bin/ekispa/exp.cgi?para=tool
[33] PASMO ホームページ, (株) パスモ, http://www.pasmo.co.jp/
[34] らくらくお出かけネット，交通エコロジー・モビリティ財団，http://www.ecomo-rakuraku.jp/rakuraku/index/
[35] ルートどっとナビ, 交通新聞社, http://www.doconavi.com/cgi-bin/rou.cgi
[36] 路線検索:AOL, AOL, http://travel.aol.co.jp/etc/transit.html
[37] So-net 路線, ソネットエンタテインメント, http://www.so-net.ne.jp/rosen/
[38] Suica, 東日本旅客鉄道 (株), http://www.jreast.co.jp/suica/
[39] スパナビ時刻表, JTB, http://supanavi.rurubu.com/supanavi/index2.html
[40] 首都圏乗換え案内, 駅前探険倶楽部, http://www8.ekitan.com/ss/N1?an=0
[41] Tokyo のりかえ案内, 東京都交通局, 東京メトロ, http://www.tokyo-subway.net/japan/
[42] Yahoo!路線情報, ヤフー, http://transit.yahoo.co.jp/

付録

乗換え案内サービスシステム

市販ソフトウェア 5 種類 [16][17][30][20][24] と Web 上のシステム 27 種類 [15][40][10][19][22][25][31][37][18][41][32][42][21][26][34][36][23][11][20][27][28][29][12][13][14][39][35] のそれぞれを利用して，吉祥寺・西船橋間の最安運賃を求めた結果を表 9 に示す（2007 年 5 月 23 日現在の調査結果）．料金順に検索できるものについては，その 1 番目に与えられた運賃を示し，料金順に検索できないシステムにおいては，出来る限り多くの経路を表示した中で最も安い運賃を取り上げた．Web 上のシステムについては，同じ検索エンジンを利用しているものも多かったため，利用していると思われるエンジンも示した．ここで，検索エンジンとして，駅前探険倶楽部と駅探はまとめて「駅探」，ヴァル研究所と駅すぱあともまとめて「駅すぱあと」と示してある．最安運賃を得たものは，33 種類のうち 12 種類，エンジンとしては 1〜3 種類だけであった．

この 2 駅間を選んだ理由は，2006 年初めに予備調査を行った際に，最安運賃 450 円を与えるシステムを見つけられなかったからである．また，これらの結果は，日々改善されており，現在は，すべてが 460 円以下，多くのものが最安運賃の 450 円を与えるようになった（2007 年 4 月 6 日の調査では 620 円以上の経路しか与えられないものも存在した）．

ここで，最安運賃 450 円の経路と，460 円の経路，そして多くの場合に利用されると思われる 620 円の経路の例を，1 つずつを以下に示す．

450 円の経路：吉祥寺（JR）→荻窪（メトロ丸の内線）→大手町（メトロ東西線）→西船橋
460 円の経路：吉祥寺（JR）→中野（メトロ東西線）→西船橋
620 円の経路：吉祥寺（JR）→西船橋

それぞれの経路は異なる利点（速さ，乗換え数，運賃等を考慮した利点）を持ち合わせている．各システムは，利用者に合わせて，様々の尺度を考慮するように作成されており，本論文で取り上げた，運賃計算のための最安運賃経路探索とは，目的が異なる．

表 9: 乗換え案内システムによる「吉祥寺駅から西船橋駅までの最安運賃」検索結果

(2007 年 5 月 23 日現在)

インストール版	検索方法	最安運賃	
駅探エクスプレス	料金順トップ	450	
駅すぱあと	料金順のトップ	460	
乗り換え案内 VER.5	3 経路中の最小	460	
ハイパーダイヤ	5 経路中の最小	460	
JR トラベルナビゲータ	5 経路中の最小	460	
Web 版	検索方法	最安運賃	検索エンジン
駅探	料金順のトップ	450	駅探
首都圏乗換え案内	料金順のトップ	450	駅探
ASK.jp 路線検索	料金順のトップ	450	駅探
goo 路線	料金順のトップ	450	駅探
infoseek 乗換案内	料金順のトップ	450	駅探
livedoor 路線案内	料金順のトップ	450	駅探
OCN　路線	料金順のトップ	450	駅探
So-net　路線	料金順のトップ	450	駅探
エキサイト乗換案内	料金順のトップ	450	駅探
Tokyo のりかえ案内	料金順のトップ	460	駅探
ODN 路線検索	5 経路中の最小	460	駅すぱあと
Yahoo!路線情報	6 経路中の最小	460	駅すぱあと
いつもガイド	5 経路中の最小	460	駅すぱあと
マップファンウェブ	5 経路中の最小	460	駅すぱあと
らくらくお出かけネット	20 経路中の最小	460	駅すぱあと
路線検索：AOL	5 経路中の最小	460	駅すぱあと
ジョルダン乗換案内	3 経路中の最小	460	駅すぱあと
BIGLOBE 乗換案内	4 経路中の最小	460	駅すぱあと
ハイパーダイヤ	10 経路中の最小	460	ハイパーダイヤ
MSN 路線	5 経路中の最小	460	ハイパーダイヤ
NAVITIME	料金順のトップ	460	NAVITIME
@ nifty 路線検索	料金順のトップ	450	不明
DION　路線	料金順のトップ	450	不明
えきから時刻表	料金順のトップ	460	不明
えきねっと時刻・乗換え案内	3 経路中の最小	460	不明
スパナビ時刻表	3 経路中の最小	460	不明
ルートどっとナビ	5 経路中の最小	460	不明

池上敦子
成蹊大学理工学部情報科学科
〒180-8633 東京都武蔵野市吉祥寺北町 3-3-1

ABSTRACT

FINDING THE MINIMUM COST PATH FOR A RAILWAY FARE CALCULATION — A CASE STUDY INVOLVING MORE THAN ONE RAILWAY COMPANY —

Atsuko Ikegami
Seikei University

Shunji Morita
Seikei University
(currently affiliated with *Nippon Signal*)

Takuma Yamaguchi
Nippon Signal

Jo Kikuchi
Nippon Signal

Toshihiro Nakayama
Nippon Signal

Motohiro Ohkura
Seikei University

When calculating the fare between a given pair of stations, the minimum cost path from among the many feasible paths between the stations must be determined. The railway fare for a specific path is usually calculated by distance, i.e. the longer the distance, the higher the fare. The fare rate, however, differs between railway companies and there are additional rules to be used in the fare calculation, e.g. discounts for specific paths. Therefore, the shortest-distance path is not always the minimum cost path. Most previous research efforts have focused on the enumeration of feasible paths between a pair of stations and a comparison of their resulting fares in order to choose the minimum fare for the pair. Using this approach, computational time is inevitably long. In this paper, we propose some network representations for the fare calculation of a railway network and use an efficient algorithm based on Dijkstra's algorithm to calculate fares between all pairs of 1638 stations in the area where IC-ticket (Suica/Pasmo) is usable. Our algorithm reduces the computational time because the algorithm need not enumerate a large number of paths for every pair of stations.