大都市近郊区間の経路の効率的な列挙と検索

堀山　貴史，羽室　行信

JR 各社が定める「大都市近郊区間」内では，出発駅・到着駅の間の最短距離によって運賃が決められ，乗車経路は同じ駅を 2 度通らなければ自由に選択できる．たとえば，東京駅から隣の有楽町駅までには 4,152,859 通りの経路がある．乗客は，そうした膨大な経路の中から，各自の好みに合わせて経路を選択している．本稿では，ZDD（Zero-suppressed Binary Decision Diagram; ゼロサプレス型二分決定グラフ）を利用して，与えられたグラフ中の経路を効率的に列挙する方法を紹介する．また，列挙された経路を実行可能解として，異なる目的関数ごとに最小化問題や最大化問題を何度も簡単に解き直す方法について述べる．そして，東京近郊区間の例を中心に，経由する単線区間が最多となる経路などを紹介する．

キーワード：ZDD（ゼロサプレス型二分決定グラフ），グラフ列挙，索引化，Ekillion

1. はじめに

東京駅から隣の有楽町駅まで，JR 山手線に乗って 1 駅 140 円で移動できる．しかし，山手線は環状に回っているため，東京駅で逆方向の電車に乗ってグルッと 1 周しても，料金は変わらず，有楽町駅まで無事にたどりつける．もっと頑張って，千葉駅・大宮駅・横浜駅などの関東圏のあちこちの駅を“経由”して大回りしても，料金は変わらず，楽しい旅の末に有楽町駅までたどりつける．これは，JR 各社が定める「大都市近郊区間」内では，実際の乗車経路にかかわらず，出発駅と到着駅の間の最短距離によって運賃を決めるという特例のおかげである．

東京駅と有楽町駅の間の経路を，同じ駅を 2 度通らないという制約のもとで数え上げてみると，4,152,859 通りもの経路があり，その中から好みの経路を選んでよいことになる．東京駅から有楽町駅までの例では，山手線に乗って 1 駅の，最短の経路を多くの人が選択するだろう．しかし，経由する駅数が最少の経路と，乗車距離が最短の経路とが異なる場合には，どちらの経路を選択するかは人によるだろう．実際にはもっと複雑で，電車がどの路線を走っているかや，どの駅を通過するかなどを勘案して経路を選択することになる．また，人によっては，なるべく多くの駅を経由したかったり，乗車距離を長くしたかったり，なるべく多くの単線や名駅百選の駅を回ってみたかったりするだろう．

ほりやま　たかし
埼玉大学理工学研究科
〒 338–8570 埼玉県さいたま市桜区下大久保 255
はむろ　ゆきのぶ
関西学院大学経営戦略研究科
〒 662–8501 兵庫県西宮市上ケ原一番町 1–155

本稿では，こうした要望に対し，(1) 与えられたグラフ中の経路を効率的に列挙する，(2) 列挙された経路を実行可能解として，異なる目的関数ごとに最小化問題や最大化問題を解き直すというアプローチをとる．ここで，経路を効率的に列挙し保持するための鍵となる要素技術として，ZDD（Zero-suppressed Binary Decision Diagram; ゼロサプレス型二分決定グラフ）を利用する．以下では，まず ZDD について説明した後に，ZDD を用いた高速なパス列挙手法と ZDD の利用方法について解説する．そして，東京近郊区間の例を中心に，「大都市近郊区間」についての実験結果を示す．

2. ZDD とグラフ列挙索引化

ZDD は，有向非巡回グラフによる組合せ集合の表現法である．ここで，組合せ集合は，台集合 $X = \{x_1, x_2, \ldots, x_n\}$ に対して定義される集合 2^X の部分集合である．以下では，ZDD を直感的に理解するために，まず二分決定木により組合せ集合を表現する方法について説明し，より効率的な表現法として ZDD へと話を進める．

二分決定木は，図 1(a) のように，定数節点と変数節点からなる．定数節点には 0 または 1 がラベル付けされており（以降，「0-定数節点」「1-定数節点」と呼ぶ），変数節点には X のいずれかの要素 x_i がラベル付けされている．各変数節点からは，0-枝と 1-枝と呼ばれる 2 種類の有向辺が，それぞれ 一つずつ出ている．また，根節点（図 1 (a) の節点 a）から定数節点へ，途中の変数節点で 0-枝と 1-枝をどのように選んでも，変数は同じ順序でちょうど 1 回ずつ出現する．この出現順を変数順序と呼ぶ．

 Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

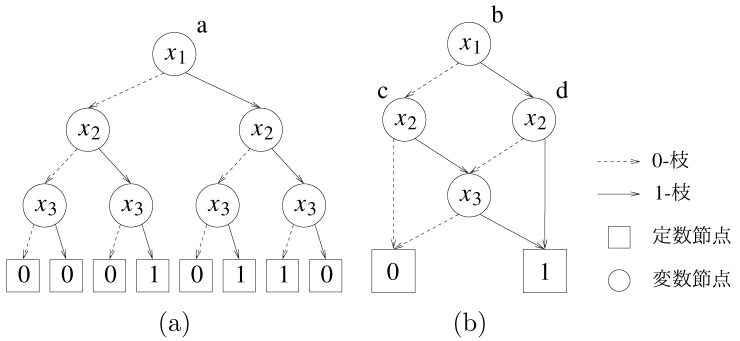

図 1　(a) 二分決定木と (b) ZDD

根節点から 1-定数節点への各経路は，それぞれが組合せに 1 対 1 対応をしている．たとえば図 1 (a) の二分決定木において，x_1 の 1-枝，x_2 の 1-枝，x_3 の 0-枝とたどって 1-定数節点に至る経路は，組合せ $\{x_1, x_2\}$ に対応している．ここで，1-枝をたどった変数のみが対応する組合せに含まれることに注意が必要である．同様に，x_1 の 1-枝，x_2 の 0-枝，x_3 の 1-枝とたどる経路や，x_1 の 0-枝，x_2 の 1-枝，x_3 の 1-枝とたどる経路は，それぞれ $\{x_1, x_3\}, \{x_2, x_3\}$ に対応している．図 1 (a) の二分決定木は，上記の 三つの経路のみをもつので，組合せ集合 $\mathcal{S} = \{\{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\}\}$ を表している．逆に，与えられた組合せ集合 $\mathcal{S}$ に対し，変数順序が定まれば，図 1 (a) のように二分決定木が定まる．

ZDD は，図 1 (a) の二分決定木に対して，(b) のように部分グラフの共有を許したものとみなすことができる．より正確には，二分決定木に対し，(1) 同型な部分グラフは一つにまとめて節点を共有する（等価な節点の共有），(2) 図 2 のように，1-枝が 0-定数節点を指している節点を削除する（冗長な節点の削除）という 2 種類の縮約を繰り返す（既約化する）ことで，ZDD を得ることができる．なお，等価な節点の共有や冗長な節点の削除をどのような順番で繰り返しても，得られる ZDD は一意に定まることが知られている．また，図 1 (b) では節点の共有は 1 カ所のみであるが，ZDD が大規模になるにつれて既約化による節点数の削減の効果は大きくなり，圧縮によるコンパクトなデータ構造としての利点が活かされることになる．

ZDD でも二分決定木と同様に変数順序が定義でき，根節点から定数節点へどのようにたどっても，同じ順序で変数が出現する．ただし，ZDD では，二分決定木と違って，各変数が必ず 1 回ずつ出現するのではなく，高々 1 回ずつ出現する．たとえば，図 1 (b) の

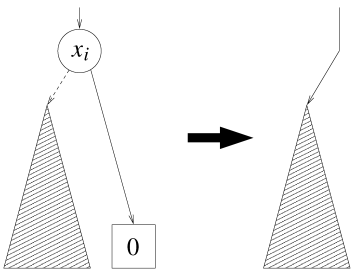

図 2　ZDD における冗長な節点の削除

ZDD では，変数順序は図 1 (a) の二分決定木と同じく x_1, x_2, x_3 であるが，根節点（図 1 (b) の節点 b）から x_1 の 1-枝，x_2 の 1-枝とたどったときには，x_3 が現れない．

根節点から 1-定数節点への経路と組合せとの対応関係は ZDD でも変わらず，たとえば図 1 (b) の ZDD は，(a) の二分決定木と同じく組合せ集合 $\mathcal{S} = \{\{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\}\}$ を表している．同様に，ZDD の各節点でも，それぞれの節点から 1-定数節点への経路をもとに，その節点が表す組合せ集合が定義できる．たとえば，図 1 (b) の節点 c は $\mathcal{S}_0 = \{\{x_2, x_3\}\}$ を表し，節点 d は $\mathcal{S}_1 = \{\{x_2\}, \{x_3\}\}$ を表している．節点 b, c, d の表す $\mathcal{S}, \mathcal{S}_0, \mathcal{S}_1$ の間には，

$$\mathcal{S} = \mathcal{S}_0 \cup \{S \cup \{x_1\} \mid S \in \mathcal{S}_1\} \tag{1}$$

という関係が成り立つ．見方を変えると，$\mathcal{S}$ の中で x_1 をもたない組合せが $\mathcal{S}_0$ に集められ，x_1 をもつ組合せが $\mathcal{S}_1$ に（x_1 を取り除いて）集められている．こうして，それぞれの節点も組合せ集合を表すことを考えると，複数の ZDD を扱いたい場合には，それらの間でも等価な節点の共有を行うことで，節点数をさらに大きく削減することができる．

なお，これまでの説明では二分決定木を既約化して

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

ZDD を得ているため，ZDD を構築するには，変数の数 n に対して $O(2^n)$ 時間が必要なように思われるかもしれない．しかし，以下に述べる巧妙な ZDD 構築法を利用することで，二分決定木を経ることなく効率的に ZDD を得ることができる．

グラフ列挙問題では，与えられたグラフ $G=(V,E)$ に対し，所望の条件を満たす部分グラフを列挙する[1]．たとえば，G の全域木を列挙する問題であったり，G と共に与えられた頂点 s,t に対して，s から t に至る s-t パスを列挙する問題であったりである．

こうした列挙問題に対し，まず，Simpath (Simple Paths) と呼ばれる s-t パスの集合を表す ZDD を構築するアルゴリズムが Knuth により提案された [1]．このアルゴリズムでは，各 s-t パスを辺の組合せとみなし，動的計画法の要領で，s-t パスの集合を表す ZDD を上から下へと効率的に構築していく．ZDD の構築途中では，各辺を s-t パスに採用するか否かが途中まで決まった状態であり，このときにあちこちででき上がりつつある部分パスから，以降の ZDD 構築で解くべき部分問題が定まる．この部分パスの情報を巧妙に利用することで，ZDD の節点の共有を行い，同じ探索を（すなわち ZDD の同型な部分グラフの構築を）2 回以上行わない点が，このアルゴリズムの肝である．この Simpath のアイデアを一般化して，さまざまなグラフ列挙問題で ZDD を構築できるようにしたものが，フロンティア法と呼ばれる [2]．

より詳しくは，各文献を参照していただきたい．また，ZDD やフロンティア法の解説と，その応用について，オペレーションズ・リサーチ誌 57 巻 11 号 [3] で特集されていたり，より詳しい入門書 [4] も刊行されているので，こちらも参考にしていただきたい．

3. ZDD の利用

以下では，構築した ZDD をどのように利用するかについて述べる．せっかく圧縮した状態でもっている ZDD を，わざわざ二分決定木に戻してから演算を行うと，$O(2^n)$ の時間がかかってしまう．したがって，ZDD のままで，つまり圧縮をほどくことなく，演算を行えることが望ましい．

まず，ZDD が与えられたときに，その ZDD が表す組合せ集合 $\mathcal{S}$ が何個の要素をもつか，すなわち $|\mathcal{S}|$ を求めてみよう．これは，式 (1) から，

$$\begin{aligned}|\mathcal{S}| &= |\mathcal{S}_0| + |\{S \cup \{x_1\} \mid S \in \mathcal{S}_1\}| \\ &= |\mathcal{S}_0| + |\mathcal{S}_1|\end{aligned}$$

であるので，以下に示すアルゴリズム 1 を再帰的に利用して，Count（根節点）を実行すればよい．

アルゴリズム 1　v を根節点とする ZDD が表す組合せ集合について，その要素数を求めるアルゴリズム

```
Algorithm Count（節点 v）

if（v = 0-定数節点）then return 0
if（v = 1-定数節点）then return 1
v0 := v の 0-枝が指す節点
v1 := v の 1-枝が指す節点
return Count(v0) + Count(v1)
```

アルゴリズムの実行過程において，同じ節点 v' について何度も Count(v') を実行するのを避けるには，v' をキーとして Count(v') の値を蓄えるキャッシュ（演算結果表とも呼ぶ）を利用するとよい．これにより，ZDD の節点数に比例する時間で計算が完了する．また，複数の ZDD を扱う場合には，それらの間で等価な節点の共有が行われていることを思い出してほしい．これは，別の ZDD の Count( ) で利用した演算結果をキャッシュに保存しておけば，（ユーザが明示的に意識せずとも）再利用できることを意味している．

なお，$|\mathcal{S}|$ は，動的計画法の要領でも求めることができる．この場合には，与えられた ZDD を下から上へとスキャンして，0-枝が指す節点と 1-枝が指す節点の要素数を加算していくことで $|\mathcal{S}|$ が得られる．再帰的な方法でも，動的計画法でも，本質的に同じことをしているので，どちらでも好みのほうを利用していただきたい．

次に，ZDD が表す組合せ集合 $\mathcal{S}$ の各要素に適当な順番を付けて，その中で N 番目の組合せを取り出してみよう．変数節点 v の 0-枝，1-枝の指す節点をそれぞれ v_0, v_1 とすると，アルゴリズム 1 を利用することで，$|\mathcal{S}_0|$ と $|\mathcal{S}_1|$ が求められる．したがって，$0 < N \le |\mathcal{S}_0|$ のときには，v_0 の表す $\mathcal{S}_0$ の中で N 番目を求めればよい（N 番目の組合せには x_1 は含まれない）．また，$|\mathcal{S}_0| < N \le |\mathcal{S}_0| + |\mathcal{S}_1|$ のときには，v_1 の表す $\mathcal{S}_1$ の中で $N - |\mathcal{S}_0|$ 番目を求めればよい（N 番目の組合せには x_1 が含まれることになる）．これを再帰的に繰り返すことで，N 番目の組合せにはどの変数が含まれるかが順にわかっていく．こうして，N を 1 から $|\mathcal{S}|$ の間でランダムにとることで，$\mathcal{S}$ からのランダムサンプ

[1] ZDD とグラフ $G=(V,E)$ の間での混乱を避けるため，ZDD では節点と枝，グラフでは頂点と辺と，用語を使い分ける．

 Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

リングを実現できる．

では次に，各変数 x_i に重み w_i が付いているとして，ZDD が表す組合せ集合 $\mathcal{S}$ に対して，重みの総和の最大値 MaxWeight($\mathcal{S}$) や最小値 MinWeight($\mathcal{S}$) を求めてみよう．最大値については，式 (1) から，

$$\begin{aligned}&\mathrm{MaxWeight}(\mathcal{S})\\&= \max\left\{\begin{array}{l}\mathrm{MaxWeight}(\mathcal{S}_0),\\ \mathrm{MaxWeight}(\{S\cup\{x_1\}\mid S\in\mathcal{S}_1\})\end{array}\right\}\\&= \max\{\mathrm{MaxWeight}(\mathcal{S}_0),\, w_1+\mathrm{MaxWeight}(\mathcal{S}_1)\}\end{aligned}$$

であるので，以下に示すアルゴリズム 2 を再帰的に利用して，MaxWeight（根節点）を実行すればよい．同様に，最小値 MinWeight($\mathcal{S}$) を求める場合には，max{ } ではなく min{ } をとればよい．また，$\max\{\mathrm{MaxWeight}(v_0),\, w_i+\mathrm{MaxWeight}(v_1)\}$ を計算する際にどちらを採用したかを覚えておくことで，重みの総和が最大の組合せ（または同様のアイデアで最小の組合せ）を取り出せる．

アルゴリズム 2　v を根節点とする ZDD が表す組合せ集合について，重みの総和が最大の組合せを求めるアルゴリズム

Algorithm MaxWeight（節点 v）

if（v = 定数節点）then return 0
$v_0 := v$ の 0-枝が指す節点
$v_1 := v$ の 1-枝が指す節点
v にラベル付けされている変数を x_i とする

$$\text{return } \max\left\{\begin{array}{l}\mathrm{MaxWeight}(v_0),\\ w_i+\mathrm{MaxWeight}(v_1)\end{array}\right\}$$

特筆すべきなのは，ZDD を最初に 1 回だけ構築しておけば，各変数の重みをさまざまに変えて，それぞれに対する重みの総和が最大／最小の組合せを簡単に（ZDD の節点数に比例する時間で）得られることである．たとえば，与えられたグラフに対して s-t パスの集合を表す ZDD を構築しておけば，さまざまな重みに対して MaxWeight( ) や MinWeight( ) を適用するだけで，それぞれの重みのもとでの最長経路や最短経路を求めることができる．各変数の重みが動的に変わるネットワークでも，ネットワークの情報を ZDD の形に圧縮する前処理をしておけば，各時点での重みに対応する最長経路や最短経路が簡単に得られることになる．最後に，組合せ集合 $\mathcal{S}$ のうち，指定した制約条件を満たす組合せのみからなる組合せ集合 $\mathcal{S}'$ を ZDD として取り出してみよう．たとえば，s-t パスの集合のうち，ある辺 e_k をもつものを取り出すとしよう．これには 2 通りのアプローチがあり，一つは ZDD をフロンティア法で構築する際に，辺 e_k を採用することを制約として加えて，$\mathcal{S}'$ を表す ZDD を構築する直接的な方法である．詳細については省略する．

もう一つは，$\mathcal{S}$ を表す ZDD に演算をほどこして $\mathcal{S}'$ を表す ZDD を得る方法である．辺 e_k を採用することは，対応する変数 x_k のラベル付けされた ZDD の節点では 1-枝側をたどることのみを許し，0-枝側は許さない，つまり 0-枝の先を 0-定数節点に変更する操作に相当する．

この操作は，Apply 演算という形で一般化されている．Apply 演算では，二つの組合せ集合 $\mathcal{S}, \mathcal{T}$ を，

$$\begin{aligned}\mathcal{S} &= \mathcal{S}_0\cup\{S\cup\{x_i\}\mid S\in\mathcal{S}_1\}\\ \mathcal{T} &= \mathcal{T}_0\cup\{T\cup\{x_i\}\mid T\in\mathcal{T}_1\}\end{aligned}$$

という形で，x_i を含むものと含まないものそれぞれ二つの組合せ集合に分割し，$\mathcal{S}\circ\mathcal{T}$ を

$$\begin{aligned}\mathcal{S}\circ\mathcal{T} = \{&S\circ T\mid S\in\mathcal{S}_0, T\in\mathcal{T}_0\}\\ &\cup\{(S\circ T)\cup\{x_i\}\mid S\in\mathcal{S}_1, T\in\mathcal{T}_1\}\end{aligned}\tag{2}$$

として再帰的に求める．（これまでに見てきた ZDD の演算と同様に，演算結果表を利用することで 2 度同じ計算を行わないため，効率的に演算ができる．）式 (2) において，演算 $\circ$ を $\cap$ や $\cup$ とすることで，$\mathcal{S}$ と $\mathcal{T}$ の積集合や和集合が得られる．したがって，$\mathcal{S}$ から辺 e_k をもつ組合せ集合を取り出すには，台集合 X に対して x_k をもつ組合せの集合 $\mathcal{T} = \{T\mid T\in 2^X, x_k\in T\}$ と $\mathcal{S}$ との積集合を求めればよい．この $\mathcal{T}$ は，要素数の多い組合せ集合であるが，等価な節点の共有が効率的に働き，変数節点がわずか n 個の ZDD で表すことができる．より複雑な制約条件を指定したい場合にも，$\mathcal{T}$ の取り方を変えることで，Apply 演算を活用することができる．

4. 大都市近郊区間

JR の運賃は，実際の乗車経路の営業距離（以下「距離」と略称）によって決まるというのが原則である．しかし JR 各社が定める「大都市近郊区間」内においては，実際の乗車経路にかかわらず，出発駅・到着駅間の最短経路によって運賃を決めるという特例がある．大都市近郊区間は東京・新潟・大阪・福岡の 4 都市に設定されている．東京近郊区間（2014 年 1 月現在）を図 3 に例示する．

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

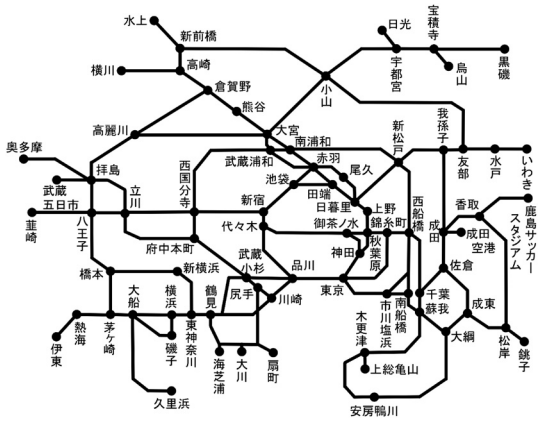

図 3　東京近郊区間（JR 時刻表 [5] より転載）

表 1　各近郊区間のグラフ統計

区間	頂点数	辺数	総距離 (km)
東京	623	653	2,055.6
大阪	355	364	941.0
福岡	128	131	323.0
新潟	58	59	174.2

この特例を利用した遊びとして，なるべく安く，なるべく長く電車の旅を楽しもうという「大都市近郊区間大回り」がある．たとえば東京駅から隣の有楽町駅までの運賃は 140 円であるが，千葉駅・大宮駅・横浜駅を“経由”して大回りしても運賃に変わりはないのである．ただし，この大回り旅の途中では，同じ駅を 2 度通ることはできない（同じ駅を 2 度通ると，運賃計算の特例が適用されない）．

ここで，近郊区間における駅を頂点集合 V，駅間の接続を辺集合 E としたグラフ $G = (V, E)$ を考え，前節で紹介した s-t 経路に対するフロンティア法を用い，任意の駅間の経路探索を行う．表 1 に各近郊区間のグラフ統計を示す．ただし，以下では誌面の都合上，東京近郊区間における例を中心に紹介していく．なお，以下の実験では，駅データ.jp より提供された路線データを利用した [6]．

5.　単線区間，名駅百選の駅を多く通る経路

前節で論じたように，ZDD によるパス列挙の利点の一つは，列挙された全経路を圧縮された状態で保持したまま，指定された条件にマッチする経路を抽出できることにある．そこで，東京近郊区間におけるある 2 駅間の経路を全列挙したのちに，辺重みを変更することで，以下に示すような目的と条件にマッチする経路を探索するケースを紹介する．

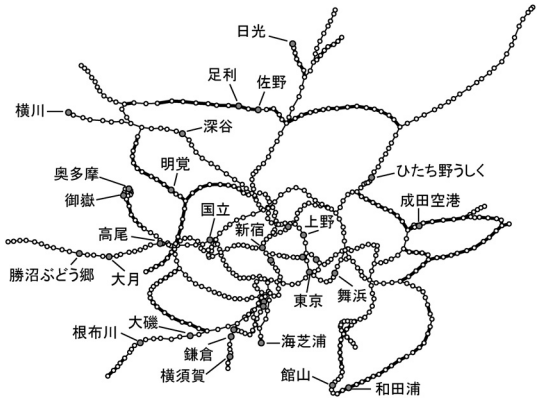

図 4　東京近郊区間における単線路線と名駅（各駅の緯度経度に基づき初期配置し，バネモデルで再描画したものであり，単線を太線，関東の駅百選に選ばれた駅を灰色円で表記）

表 2　東京近郊区間内の単線区間

区間		路線	区間数	距離
宝積寺駅	烏山駅	烏山線	7	20.4
宇都宮駅	日光駅	日光線	6	40.5
小山駅	友部駅	水戸線	15	50.2
小山駅	岩舟駅	両毛線	4	19.3
佐野駅	駒形駅	両毛線	9	48.3
前橋駅	新前橋駅	両毛線	1	2.5
日進駅	高麗川駅	川越線	9	26.9
香取駅	鹿島サッカースタジアム駅	鹿島線	5	17.4
成田駅	松岸駅	成田線本線	13	62.3
成田駅	成田空港駅	成田線空港支線	2	10.8
我孫子駅	成田駅	成田線我孫子支線	9	32.9
佐倉駅	銚子駅	総武本線	16	65.2
大網駅	成東駅	東金線	4	13.8
君津駅	安房鴨川駅	内房線	20	81.1
木更津駅	上総亀山駅	久留里線	13	32.2
上総一ノ宮駅	東浪見駅	外房線	1	3.2
長者町駅	御宿駅	外房線	4	13.3
勝浦駅	安房鴨川駅	外房線	6	22.4
八王子駅	北藤岡駅	八高線	21	88.4
東青梅駅	奥多摩駅	青梅線	13	20.0
拝島駅	武蔵五日市駅	五日市線	6	11.1
尻手駅	川崎新町駅	南武線浜川崎支線	2	2.0
浜川崎駅	扇町駅	鶴見線本線	2	1.3
新芝浦駅	海芝浦駅	鶴見線海芝浦支線	1	0.8
安善	大川駅	鶴見線大川支線	1	1.6
横須賀駅	久里浜駅	横須賀線	2	8.0
茅ケ崎駅	橋本駅	相模線	17	33.3
来宮駅	伊東駅	伊東線	4	15.7
		合計	213	744.0

目的 1：単線数最大化

できる限り多くの単線区間を経由するような経路を求めることを目的とする．東京近郊区間には，図 4，表 2 に示されるとおり，総延長 213 区間，744.0 km にも及ぶ単線区間が郊外を中心に散在している．単線区間は盲腸線（路線の一端がほかの路線と接続しておらず，袋小路となっている路線）に多いため，s-t の指定

 Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

表 3 関東の名駅百選に選ばれた東京近郊区間内にある駅一覧（全 32 駅．新横浜駅は新幹線の駅として選ばれたので除外している）

横川，足利，佐野，日光，ひたち野うしく，明覚，深谷，舞浜
成田空港，館山，和田浦，東京，上野，両国，御茶ノ水，駒込
新宿，原宿，国立，高尾，御嶽，奥多摩，海芝浦，横浜，桜木町
横須賀，北鎌倉，鎌倉，大磯，根府川，大月，勝沼ぶどう郷

表 4 東京–有楽町間の経路における各種最大化の結果

目的	距離	駅数	単線数	単線距離	名駅数
名駅数最大化	802.6	265	116	430.3	16
単線数最大化	850.2	247	140	519.1	10
単線数名駅数最大化	878.6	280	140	519.1	15
駅数最大化	986.0	343	132	483.0	15
距離最大化	1003.8	328	120	440.2	14

によっては経由できない単線区間がある．たとえば，宇都宮–日光は 40.5 km の単線区間ではあるが，その末端である日光でほかの路線と接続していないため，宇都宮–日光間の駅を始点と終点に選ばない限り，この単線区間を経由する経路は選ばれないことになる．

目的 2：名駅数最大化

できる限り多くの名駅[2]を経由するような経路を求めることを目的とする．「鉄道の日」の記念行事の一環として，1997〜2001 年の 4 年間に関東運輸局により選定された「関東の駅百選」を名駅として利用する．図 4 および表 3 に示されるとおり，東京近郊区間からは 32 の駅が選ばれている．熱海駅は路線の末端の駅ではないが，東京近郊区間の端に位置するため，結果として茅ヶ崎–熱海間は盲腸線となってしまい，その区間に存在する 2 駅（根布川，大磯）は，*s*-*t* の指定によっては経由できない．

条件 1：できる限り安い運賃

隣接駅間の経路を対象とすることで条件を満たしたものと考える．東京近郊区間における隣接駅間の運賃は 130 円〜190 円[3]の範囲にあり，約半数の隣接駅間の運賃が 140 円である．最長の隣接駅間は成田–空港第 2 ビル間の 9.8 km（190 円）である．

条件 2：できる限り少ない駅数

上記の目的を最大化する解は複数ある可能性がある．そこで，できるだけ効率的に移動することを考え，それらの解のうち経由する駅数が最も少ないような経路を選択することを条件とする．

[2] 名古屋駅のことではない．
[3] 2014 年当時の運賃．現在は 140 円〜200 円．

まずは東京と有楽町間の経路を ZDD により列挙すると，その数は 4,152,859 通りになる．メモリに格納された ZDD オブジェクトに対して，条件に応じた辺重みを与えることで目的とする経路を得ることが可能となる．たとえば，辺重みをすべて均等に設定した場合，辺重みが最大となる経路を計算することで最多駅経路が得られる．また辺重みとして距離を設定すれば最長経路が得られる（表 4）．

目的 1 の単線区間の最大化については，単線であるかどうかで 1 と 0 をそれぞれ辺重みとして割り当てればよい．また目的 2 の名駅かどうかは，頂点の属性であるため，辺重みに変換する必要がある．ここでは辺に接続された名駅の数が 0, 1, 2 のとき，それぞれ 0, 0.5, 1 を辺重みとして与えることで，名駅を通過すれば辺重みの合計が 1 となるように設定できる．ただし，始点もしくは終点が名駅の場合は，接続される辺重みを 1 に修正する．

また単線数最大化と名駅数最大化を同時に考慮した多目的問題（以下「単線数名駅数最大化」と呼ぶ）については，二つの重みに対する加重和を新たな辺重みとして定義すればよい．以下の実験では二つの重みの合計を用いており，単線を 1 区間通過するのと一つの名駅を通過するのを等しく評価することになる．

最後に，条件 2 の駅数の最小化条件については少し工夫が必要となる．ここでの目的は，たとえば単線経路の重みを最大化する問題において，重みの総計が同点の経路において駅数重みが最小になる経路を選ぶことである．辺 e の単線重み $w_e^\tau \in \{0, 1\}$ およびすべての辺に共通の微小な駅数重み w^σ (< 0) について，これらの和としての重み $w_e^\tau + w^\sigma$ を各辺に与えるとする．ここで，選択された経路のすべての駅数重みの合計の大きさが，単線区間の重み 1 を超えなければ，駅数重みが単線数最大化に影響を与えることはない．得られる経路の辺数は $|E|$ を超えないので，$|w^\sigma| < 1/|E|$ の条件を満たすような小さな値を w^σ として設定すればよい．

6. 結果

表 4 に，東京駅–有楽町駅間の経路における名駅数最大化，単線数最大化，名駅数単線数最大化についての結果の要約を示す．表 5 と図 5 に単線数最大化の経路を，そして表 6 に，名駅数最大化の経路を示す．

140 円の切符を購入するだけで，32 名駅のうち 16 駅をめぐることが可能となり，総延長 213 区間 744.0 km の単線区間のうち，140 区間 519.1 km の単線区間をめぐることができることがわかる．単線数名駅数最大化

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

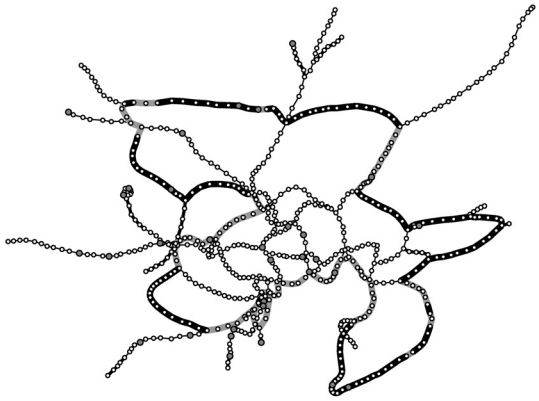

図 5　東京–有楽町間の経路で，単線区間を最も多く経由する経路（太線が経路であり，うち黒の太線が単線区間，グレーの太線が複線区間）

表 5　有楽町–東京間の単線区間最長かつ総駅数最少の経路（＝は複線区間，－は単線区間，★は関東名駅百選の駅）

有楽町＝新橋＝浜松町＝田町＝品川＝大井町＝大森＝蒲田＝川崎－尻手－八丁畷－川崎新町＝浜川崎＝武蔵白石＝安善＝浅野＝弁天橋＝鶴見小野＝国道＝鶴見＝新子安＝東神奈川＝★横浜＝保土ケ谷＝東戸塚＝戸塚＝大船＝藤沢＝辻堂＝茅ケ崎－北茅ケ崎－香川－寒川－宮山－倉見－門沢橋－社家－厚木－海老名－入谷－相武台下－下溝－原当麻－番田－上溝－南橋本－橋本＝相原＝八王子みなみ野＝片倉＝八王子－北八王子－小宮－拝島＝昭島＝中神＝東中神＝西立川＝立川＝★国立＝西国分寺＝新小平＝新秋津＝東所沢＝新座＝北朝霞＝西浦和＝武蔵浦和＝中浦和＝南与野＝与野本町＝北与野＝大宮＝日進－西大宮－指扇－南古谷－川越－西川越－的場－笠幡－武蔵高萩－高麗川－毛呂－越生－★明覚－小川町－竹沢－折原－寄居－用土－松久－児玉－丹荘－群馬藤岡－北藤岡＝倉賀野＝高崎＝高崎問屋町＝井野＝新前橋－前橋＝前橋大島＝駒形－伊勢崎－国定－岩宿－桐生－小俣－山前－★足利－富田－★佐野＝岩舟－大平下－栃木－思川－小山－小田林－結城－東結城－川島－玉戸－下館－新治－大和－岩瀬－羽黒－福原－稲田－笠間－宍戸－友部＝岩間＝羽鳥＝石岡＝高浜＝神立＝土浦＝荒川沖＝★ひたち野うしく＝牛久＝佐貫＝藤代＝取手＝天王台＝我孫子－東我孫子－湖北－新木－布佐－木下－小林－安食－下総松崎－成田－久住－滑河－下総神崎－大戸－佐原－香取－水郷－小見川－笹川－下総橘－下総豊里－椎柴－松岸－猿田－倉橋－飯岡－旭－干潟－八日市場－飯倉－横芝－松尾－成東－求名－東金－福俵－大網＝永田＝本納＝新茂原＝茂原＝八積＝上総一ノ宮－東浪見＝太東＝長者町－三門－大原－浪花－御宿＝勝浦－鵜原－上総興津－行川アイランド－安房小湊－安房天津－安房鴨川－太海－江見－★和田浦－南三原－千歳－千倉－九重－★館山－那古船形－富浦－岩井－安房勝山－保田－浜金谷－竹岡－上総湊－佐貫町－大貫－青堀－君津＝木更津＝巌根＝袖ケ浦＝長浦＝姉ケ崎＝五井＝八幡宿＝浜野＝蘇我＝千葉みなと＝稲毛海岸＝検見川浜＝海浜幕張＝新習志野＝南船橋＝二俣新町＝市川塩浜＝新浦安＝★舞浜＝葛西臨海公園＝新木場＝潮見＝越中島＝八丁堀＝★東京

表 6　有楽町–東京間の名駅数最多かつ総駅数最少の経路（記号の意味は表 5 と同様）

有楽町＝新橋＝浜松町＝田町＝品川＝大崎＝五反田＝目黒＝恵比寿＝渋谷＝★原宿＝代々木＝千駄ケ谷＝信濃町＝四ツ谷＝市ケ谷＝飯田橋＝水道橋＝★御茶ノ水＝秋葉原＝御徒町＝★上野＝鶯谷＝日暮里＝西日暮里＝田端＝★駒込＝巣鴨＝大塚＝池袋＝目白＝高田馬場＝新大久保＝★新宿＝大久保＝東中野＝中野＝高円寺＝阿佐ケ谷＝荻窪＝西荻窪＝吉祥寺＝三鷹＝武蔵境＝東小金井＝武蔵小金井＝国分寺＝西国分寺＝★国立＝立川＝西国立＝矢川＝谷保＝西府＝分倍河原＝府中本町＝南多摩＝稲城長沼＝矢野口＝稲田堤＝中野島＝登戸＝宿河原＝久地＝津田山＝武蔵溝ノ口＝武蔵新城＝武蔵中原＝武蔵小杉＝新川崎＝鶴見＝新子安＝東神奈川＝★横浜＝★桜木町＝関内＝石川町＝山手＝根岸＝磯子＝新杉田＝洋光台＝港南台＝本郷台＝大船＝藤沢＝辻堂＝茅ケ崎－北茅ケ崎－香川－寒川－宮山－倉見－門沢橋－社家－厚木－海老名－入谷－相武台下－下溝－原当麻－番田－上溝－南橋本－橋本＝相原＝八王子みなみ野＝片倉＝八王子－北八王子－小宮－拝島－東福生－箱根ケ崎－金子－東飯能－高麗川－毛呂－越生－★明覚－小川町－竹沢－折原－寄居－用土－松久－児玉－丹荘－群馬藤岡－北藤岡＝倉賀野＝高崎＝高崎問屋町＝井野＝新前橋－前橋＝前橋大島＝駒形－伊勢崎－国定－岩宿－桐生－小俣－山前－★足利－富田－★佐野＝岩舟－大平下－栃木－思川－小山－小田林－結城－東結城－川島－玉戸－下館－新治－大和－岩瀬－羽黒－福原－稲田－笠間－宍戸－友部＝岩間＝羽鳥＝石岡＝高浜＝神立＝土浦＝荒川沖＝★ひたち野うしく＝牛久＝佐貫＝藤代＝取手＝天王台＝我孫子－東我孫子－湖北－新木－布佐－木下－小林－安食－下総松崎－成田＝酒々井＝佐倉－南酒々井－榎戸－八街－日向－成東－求名－東金－福俵－大網＝永田＝本納＝新茂原＝茂原＝八積＝上総一ノ宮－東浪見＝太東＝長者町－三門－大原－浪花－御宿＝勝浦－鵜原－上総興津－行川アイランド－安房小湊－安房天津－安房鴨川－太海－江見－★和田浦－南三原－千歳－千倉－九重－★館山－那古船形－富浦－岩井－安房勝山－保田－浜金谷－竹岡－上総湊－佐貫町－大貫－青堀－君津＝木更津＝巌根＝袖ケ浦＝長浦＝姉ケ崎＝五井＝八幡宿＝浜野＝蘇我＝千葉みなと＝稲毛海岸＝検見川浜＝海浜幕張＝新習志野＝南船橋＝二俣新町＝市川塩浜＝新浦安＝★舞浜＝葛西臨海公園＝新木場＝潮見＝越中島＝八丁堀＝★東京

においては，単線数は単線数最大化の結果と変わらない一方で，名駅数を 5 駅余分に回ることができている．

表 7　最多経路数をもつ駅ペア

近郊	隣接駅間		任意の駅間	
区間	駅ペア	経路数	駅ペア	経路数
東京	布佐–木下	9,427,117	本郷台–東十条	28,328,716
大阪	東淀川–新大阪	103	栗東–塚口	392
福岡	香椎–九産大前	9	原町–東水巻	16
新潟	田上–矢代田	3	越後石山–長岡	4

表 8　最長経路をもつ駅ペア

近郊	隣接駅間		任意の駅間	
区間	駅ペア	距離 (km)	駅ペア	距離 (km)
東京	浅野–安善	1,016.8	いわき–韮崎	1,201.3
大阪	河内永和–俊徳道	557.0	難波–塚口	743.5
福岡	香椎–九産大前	165.2	今山–水巻	212.0
新潟	田上–矢代田	121.0	北三条–長岡	142.2

表 9　最多駅経路をもつ駅ペア

近郊	隣接駅間		任意の駅間	
区間	駅ペア	駅数	駅ペア	駅数
東京	浅野–安善	340	南橋本–上総亀山	390
大阪	河内永和–俊徳道	208	難波–塚口	286
福岡	香椎–九産大前	66	今山–水巻	85
新潟	田上–矢代田	41	北三条–長岡	48

7.　さまざまな最高経路

前節では，始点と終点を固定した列挙について紹介してきた．ここでは，任意の頂点ペアを始点と終点とした例を紹介する [4].

まず，ある駅から隣の駅までの経路（すなわち初乗り運賃で乗車可能な経路）について，その経路数，距離，駅数のそれぞれが最大となる駅ペアを各近郊区間別に表 7〜9 の左列に示す．隣接駅という制約の強さから，最長経路と最多駅経路はすべての近郊区間で同じ結果となっている．

次に，隣接駅に限らず，任意の 2 駅を始点終点とした場合についての結果を表 7〜9 の右列に示す．隣接駅とは違い，東京近郊区間において，最長経路と最多駅経路が異なっている．これは，最多駅経路において，久留里線という，短い距離の割に駅数が多めの盲腸線がゴールに選ばれているためである．

8.　おわりに

本稿では，JR の大都市近郊区間において，多様な条件にマッチする経路を ZDD を用いて列挙する方法を紹介した．鉄道網における経路の探索については，これまで整数計画法を中心として高速な手法がいくつか提案されてきたが，本稿で紹介した手法の利点は，整数計画法などの特殊な知識を必要とせず，経路データと重みデータさえ用意すれば，誰でも容易にさまざまな制約条件を満たす経路を求められることにある．

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

この気軽さを皆さんも是非とも体感してもらいたい．なお，本稿で紹介した方法の一部は [4] においても紹介されており，JR の全国路線データおよび各種描画プログラムが利用可能である．さらに，プログラミングの知識がなくても地図上で経路探索を可能とするWEB サービス「Ekillion（えきりおん）」を公開している（http://www.nysol.jp/apps/ekillion）．皆さんの身近な路線において，多様な条件を指定した経路探索を楽しんでもらいたい．

謝辞　東海大学の鵜飼孝盛先生，および鉄道総合技術研究所の佐藤圭介先生には，実験結果の誤りをご指摘いただきました．ここに感謝の意を表します．

参考文献

[1] D. E. Knuth, *The Art of Computer Programming, Vol. 4A, Combinatorial Algorithms: Part 1*, Addison-Wesley Professional, 2011.
[2] J. Kawahara, T. Inoue, H. Iwashita and S. Minato, “Frontier-based search for enumerating all constrained subgraphs with compressed representation,” Technical Report TCS-TR-A-14-76, Division of Computer Science, Hokkaido University 2014.
[3] 湊真一ら，特集　BDD/ZDD を用いた新しい列挙索引化技法（フロンティア法）とその応用，オペレーションズ・リサーチ：経営の科学，**57**, pp. 596–628, 2012.
[4] 湊真一（編），『ERATO 湊離散構造処理系プロジェクト，超高速グラフ列挙アルゴリズム—〈フカシギの数え方〉が拓く，組合せ問題への新アプローチ—』，森北出版，2015.
[5] JR 時刻表，2014 年 1 月号，交通新聞社，2014.
[6] 株式会社コードブラス，駅データ.jp，http://www.ekidata.jp（2014 年 1 月閲覧）

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.