

ZDD を用いた避難所割り当て集合からの全てのパレート解の抽出手法

○瀧澤 重志*1

キーワード：避難所割当， 列挙， ZDD， パレート解

1. はじめに

大規模災害が発生した際に、どこに避難すればよいのかわからない住民が多いことが東日本大震災で明らかになり、避難所の地域への割当が重要な課題となっている。避難所を地域に割り当てる数理的な問題は集合分割問題などに属し NP 困難な問題だが、割り当てられた地域が住人にとって意味のあるまとまりを確保しながら厳密解を求めるのはさらに困難である。そこで本報では、不整形な小地域単位で避難所に割り当てる問題を定式化し、全ての割当を Zero-suppressed Binary Decision Diagram (ZDD)¹⁾²⁾ と呼ばれる圧縮データ構造で全列挙した既往研究³⁾を発展させ、無数の列挙案の中から、移動距離と収容率に関するすべてのパレート解を効率的に抽出する方法を提案する。

アルゴリズムック・デザインとの関係でいうと、本研究で提案する直積演算による列挙手法は、フロアプラン列挙などの問題に応用できると考えられる。既報⁴⁾ではフロンティア法と呼ばれる異なる列挙手法を応用したが、本研究で提案する手法がより有望な印象がある。もう一つは、厳密なパレート解の抽出である。特に構造最適化の分野では多目的最適化のパレート解を、メタヒューリスティクスで抽出する研究が盛んだが、一般にすべてのパレート解を列挙することは困難である。一方本研究では、目的関数を若干単純化してはいるが、ZDD の関数等をうまく利用することですべてのパレート解を現実的な時間で抽出できる。

2. 問題設定

図 1 に示すような小地域のポリゴンで構成される対象地域全体を R 、各小地域を $r \in R$ と表す。小地域を区別する際には、 $R = \{r_1, \dots, r_{|R|}\}$ と r にインデックスを付けて表記する。本研究では収容避難所を想定し、避難所の点の集合を E 、各避難所を $e \in E$ と表す。避難所 e を含む小地域を r_e と表す。反対に、 r に含まれる避難所の集合を $E_r \subseteq E$ と表す。二つ以上の異なる避難所が同じ小地域に属する場合があるが、その場合は E_r の中から任意の一つの避難所を選んで記号表現上代表させる。そうして小地域と避難所を一对一対応させた避難所の集合を $E' \subseteq E$ と表す。避難所を区別する際には、 $E' = \{e'_1, \dots, e'_{|E'|}\}$ とインデックスを付けて表す。また、各小地域の想定避難者数を p_r 、避難所 e の収容者数を c_e とする。複数の避難所が小地域に存在する場合があるので、小地域単位で収容者の和を求め cs_{r_e}

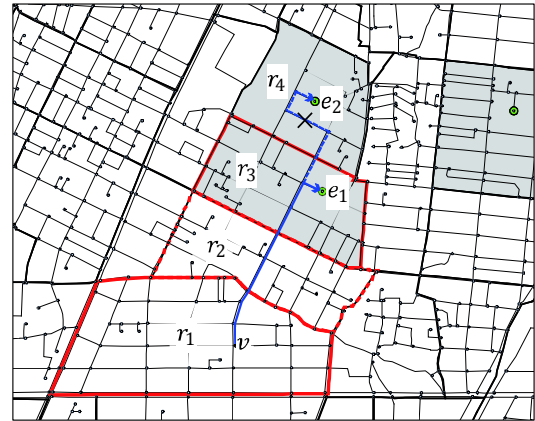


図 1 対象地域と小地域の包含関係の説明：細線は道路のエッジ，グレーの○は交差点を示す。

とおくと、 $cs_{r_e} = \sum_{e \in E_{r_e}} c_e$ となる。

次に、現実的な避難所割り当てを行うために、以下の制約を導入する。まず、避難所 $e' \in E'$ が被覆する小地域の集合のうち、実行可能な組み合わせの集合を $\mathcal{R}_{e'}$ と表し、その一つの組み合わせを $R_{e'} \in \mathcal{R}_{e'}$ と表す。各 $e' \in E'$ について $R_{e'}$ を求めて組み合わせたものが、 $\cup_{e' \in E'} R_{e'} = R, R_{e'} \cap R_{e''} = \emptyset (e' \in E', e'' \neq e')$ を満たすとき、その組み合わせは、厳密被覆 (exact cover) と呼ばれる。これは小地域がどこにも割り当てられなかったり、複数の避難所に割り当てられることを禁止する (C1)。

次の制約は、避難所を含む小地域の住人は原則その避難所に逃げるものとする。すなわち以下の制約が満たされる必要がある。 $r_e \in R_{e'} (C2)$ 。

次の制約は、同じ避難所に被覆される各小地域が連結し、さらに異なる避難所へ向かう避難者の動線が交差しないようにするものである (C3)。各小地域の避難者は、道路の交差点を母点とするボロノイ領域で面積案分されて交差点に配分され、そこから最短経路で避難所まで避難する。このとき、小地域の避難者のうち xxx が他の小地域を経由する場合、その経由する小地域も同じ避難所に割り当てられる必要がある。例えば、図 1 に示すように、避難所 e_1 を含む小地域 r_3 が小地域 r_1 を被覆しようとしている状況だとする。もし $r_1 \in R_{e_1}$ が成立する場合、過半の避難者が r_2 を経由するので、 $r_2 \in R_{e_1}$ となっている必要がある。

また、図 1 において、避難所 e_2 を含む小地域 r_4 が小地域 r_1 を被覆しようとしている状況だとする。例えば同

図において v から e_2 への最短経路が、他の避難所がある r_3 を横断している。このように、他の避難所を含む小地域を横断する経路を通る避難者が一定数以上ある割当は禁止される(C4).

max_dist , min_cap , max_cap をパラメータとする。もし $r \in R_{e'}$ が成立する場合、 r 内の交差点から避難所 e' までの平均距離を $d_r^{e'}$ とすると、 $d_r^{e'} \leq max_dist$ 以下となっている必要がある(C5).

避難所のキャパシティの和も、避難者数の合計のある範囲に収まっている必要がある。すなわち、 $min_cap \leq \sum_{r \in R_{e'}} p_r / cs_{r_{e'}} \leq max_cap$ が成立している必要がある(C6).

3. 2分木による割当の表現

ZDD を用いた列挙アルゴリズムの適用のために、割当を2分木で表現する。避難所を含む小地域は $|E'|$ 個ある。各小地域からみれば、(C1)よりどれか一つの避難所を含む小地域に被覆される必要がある。これは $|E'|$ 次元の0-1ベクトルで表現できる。例えば、 e'_3 を含む小地域に r_1 が含まれるときは $(0,0,1,\dots,0)$ と表現される。小地域は全部で $|R|$ 個あるので、全ての小地域の被覆は $|E'| \times |R| = n$ 次元のベクトルで表現される。このベクトルを $x = (x_1, x_2, \dots, x_n)$ とおき、その集合 X を ZDD で表現・保持する。図2は、このベクトルを2分木で表現した例である。この場合先に述べた、 r_1 が e'_3 を含む小地域に含まれることを示している。ZDD はこのような2分木を圧縮して保持したものである。

4. 割当の列挙アルゴリズム

アルゴリズムは2段階からなる。各避難所に対する小地域の割当の列挙を行い、次に、それらを組み合わせて地域全体の避難所割当の列挙を行う。本報では紙面の都合により詳細なアルゴリズムは省略し、以下にその概要のみ説明する。以下では簡単に説明するために、図3左上に示す2つの避難所と6つの小地域からなる例を用いる。

4.1 避難所毎の割当の列挙

まず、避難所 $e' \in E'$ を含む小地域が被覆する小地域の組み合わせを列挙して ZDD として保存する。図3右にアルゴリズムのイメージを示す。これは避難所がある小地域 r_{e1} に他の小地域を割り当てている例である。まず左の列では、 r_{e2} を除いて各小地域を避難所に割り当てている。この際、制約 C2, C4, C5 を満たすかどうかチェックする。そして制約を満たす割当だけを ZDD 変数の Z に入れる。次に ZDD の直積演算を使って $Z \leftarrow Z \times Z$ の組み合わせを求め、組み合わせが増えなくなるまでこの演算を繰り返す、この避難所に対するすべての小地域の割当を列挙できる。ただし、制約 C3,C4,C6 を満たすものだけを ZDD 変数の Z_{e1} に入れる。なお、制約を満たすかどうかのチェックは、制約違反となる割当の極小のパターンを予め列挙して ZDD 変数としておき、ZDD の集合演算でそのパターンを含むもの

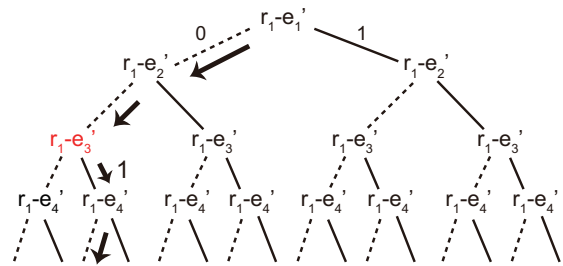


図2 2分木による割当の表現例

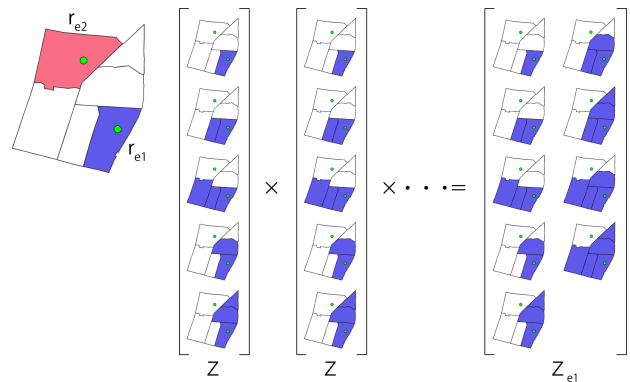


図3 例題の地域(左上)と、避難所を含む小地域 r_{e1} への地域割当の列挙アルゴリズムのイメージ(右図)

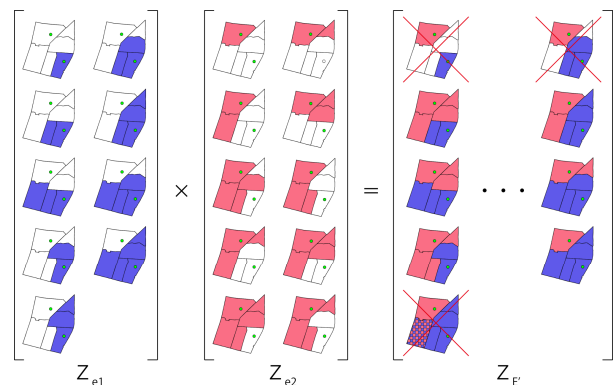


図4 地域全体の割当の列挙アルゴリズムのイメージ

だけを削除するといった方法で高速に行える。

4.2 全避難所の割当の列挙

各避難所で割当の列挙を行った後に、それらの直積をとることで対象地域全体の割当を列挙する。図4にこのアルゴリズムのイメージを示す。 Z_{e2} は先のアルゴリズムで避難所 r_{e2} に対する割り当てを列挙した ZDD 変数である。これと先に求めた Z_{e1} の直積をとることで、すべての割当のパターンが列挙できる。ただし、同図右上のどこにも割り当てられてないパターンや右下の2重割当となっている場合は排除されて、 $Z_{E'}$ に格納される。

5. 目的関数の設定

列挙された割当の集合から、避難計画に適した割当を抽出するための目的関数を設定する。本研究ではある割当 x

を評価するために、平均移動距離と平均収容率に関する二つの目的関数を設定し、それらを $o_1(x)$ と $o_2(x)$ として、それぞれ式(1)(2)のように定義する。ここで $x_r^{e'} \in x$ は、 r が e' に被覆されている場合 1 を、そうでない場合 0 となる。また、 $s_r^{e'} = p_r / cs_{r_{e'}}$ とする。

$$o_1(x) = \frac{1}{|R|} \sum_{r \in R} \sum_{e' \in E_r} x_r^{e'} \cdot d_r^{e'} \quad (1)$$

$$o_2(x) = \frac{1}{|E'|} \sum_{r \in R} \sum_{e' \in E_r} x_r^{e'} \cdot s_r^{e'} \quad (2)$$

6. パレートの抽出

多数の列挙解の中で、短い移動距離と少ない収容率を満たす被覆だけが必要であるが、それらは両目的関数に関するパレート最適解になっている。パレート最適解の分布が凸の場合、図 5 に示すように、それらは凸包の頂点となっているサポート解とそれ以外の非サポート解に分けられる。以下に、非サポート解も実用的な時間で抽出するアルゴリズムについて述べる。

6.1 単目的最適化への変換

もとの問題は 2 目的の最小化問題であるが、ZDD を用いて問題を解くために、式(3)に示す目的関数の重み $\lambda \in [0,1]$ に関する、単目的のパラメトリックな線形計画問題を定義する。ここで max_{o_1} , max_{o_2} はそれぞれ、 $o_2(x)$, $o_1(x)$ の単目的最適化問題を解いた解の $o_1(x)$, $o_2(x)$ の値である。また、 $Exist(x)$ は、 $Z_{e'}$ に含まれる被覆集合の中で、組み合わせが x となるものが存在する場合に 1 を、そうでなければ 0 を返す。このような制約が付いた線形計画問題を Boole 線形計画問題と呼ぶ⁵⁾。

$$\min_{x \in X} \left\{ \lambda \frac{o_1(x)}{max_{o_1}} + (1-\lambda) \frac{o_2(x)}{max_{o_2}} \right\} \quad (3)$$

s.t. $Z_{e'}, Exist(x) = 1$

いま、一つのノード集合 x の各要素に対する重みの実数値ベクトルを $w = (w_1, w_2, \dots, w_n)$ とし、ノード $x_r^{e'} \in x$ に対応する重みを $w_r^{e'} \in w$ とし、参照できるものとする。また、 x の各ノードの重み付き和を $sum = w_1x_1 + w_2x_2 + \dots + w_nx_n$ とする。ある λ の元で、問題(3)を解くために ZDD のライブラリに含まれる $Choose_Best(w)$ というメソッドを利用する。このメソッドは、当該 ZDD において、重みベクトル w が引数として与えられた際に、 sum が最大となる x を返す。

ある λ の元での各ノード $x_r^{e'}$ の重みを

$$w_r^{e'} = \frac{\lambda d_r^{e'}}{|R| max_{o_1}} + \frac{(1-\lambda) cr_r^{e'}}{|E'| max_{o_2}}$$

とすると、 sum は問題(3)の {} 内の値と等しくなる。 w の各要素にマイナスをかけて引数とすることで、この解法を最小化問題 (3) に適用することができる。 λ を次に示す方法で決めて問題(3)を解き、解を ZDD から削除することを繰り返し、すべてのパレート解を抽出することができる。

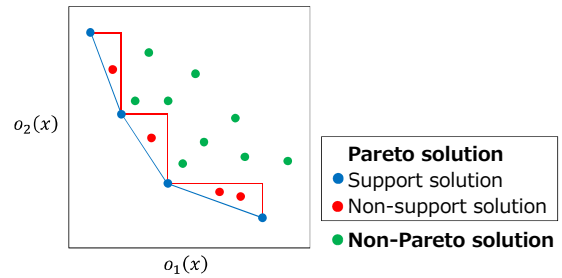


図 5 2 目的最小化問題におけるパレート解

6.2 サポート解の抽出

まずは、パレート解のうちサポート解を抽出する。サポート解は、先ほど述べたように解集合を目的関数空間にプロットした際にできる凸包の左下側の頂点になるので、計算幾何学で開発された hand probing oracle と呼ばれる手法を用いることで、解の個数だけ 2 分探索を行うことで、効率的にサポート解をすべて抽出することができる⁶⁾。

このアルゴリズムのイメージを図 6 に示す。"1"で、 max_{o_1} , max_{o_2} を含むそれぞれの目的関数の単目的最適化の解をプロットする。"2"でその 2 点をつなぐことで線分の傾き ($\lambda = 0.5$) が得られる。この λ の値で問題 3 を解くと、"4"で示した中間の点で新しいサポート解が得られたので、元の直線を"5"のように 2 分割する。分割された線分についてそれぞれの傾きを求めて同じことを繰り返すと、下側の線分では新しいサポート解が得られなかった ("6", "7")。すなわち、当該 λ で問題(3)を解いた場合に、得られた解が元の線分のいずれかの端点と一致した。一方、上の線分ではさらに一つ新しいサポート解が見つかり ("8"), 最終的に"9"で示す 4 つのサポート解が得られた。

6.3 非サポート解の抽出

次に、非サポート解の抽出を行う。図 7 に示すように、まず、見つかったサポート解を頂点とする直角三角形を"1"のように与える。次に左上の線分の傾き (すなわち λ) と等しい傾きで、位置が上の直角三角形の頂点を通る線分を上界として設定する ("2")。この λ で問題 3 を解き、得られた解が非サポート解かどうかを判定し、非サポート解の場合はそれを保存する ("3"- "6")。解が設定した上界を超えた場合 ("6"), その λ での探索を中止し、次の凸包の線分に移って同じ処理を繰り返す ("7"- "8")。最終的に "9" のような非サポート解集合を得る。

7. 計算例

大阪市住吉区を対象地域とし、平成 22 年度国勢調査の小地域統計と、全国デジタル道路地図のデータを用いた。小地域は 104 個、避難所を含む小地域は 31 地域である。プログラムは Visual C++2013 で作成し、ZDD のライブラリには Graphillion⁷⁾を用いた。 $min_p = 0.49$, $max_dist = 1200m$, $min_cap = 1.0$, $max_cap = 8.1$ とし、PC (OS: Windows 8.1 64Bit, CPU: Core i7-5960x, Memory: 64GB) を

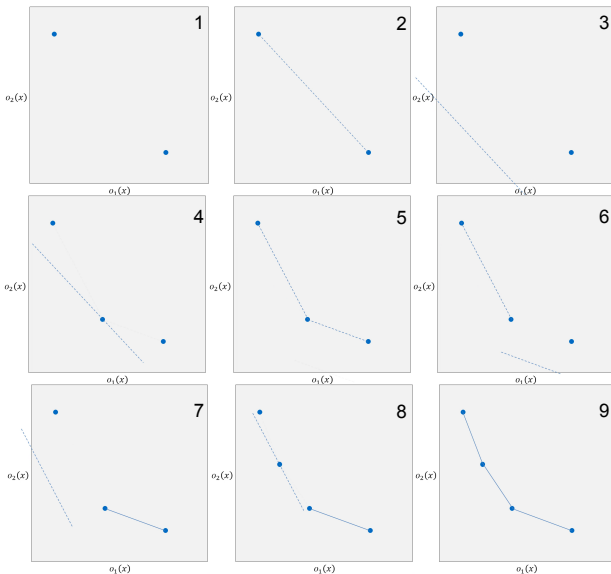


図 6 サポート解の抽出アルゴリズムのイメージ

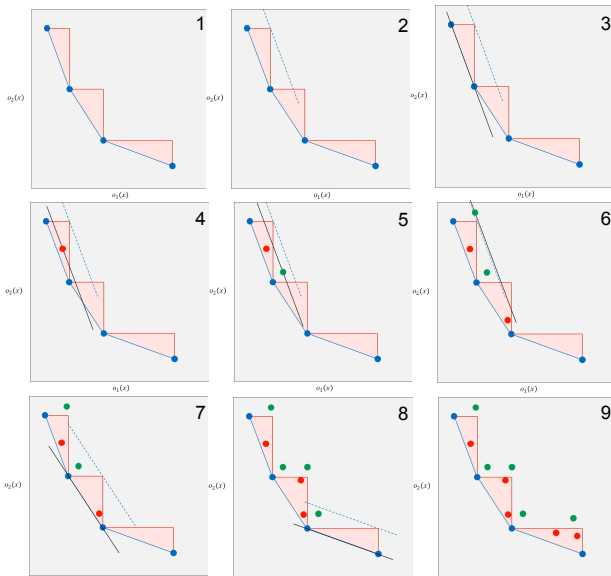


図 7 非サポート解の抽出アルゴリズムのイメージ

使い計算を行った結果、約 25 億個の割当を得た。列挙にかかった計算時間は約 0.5 秒、使用メモリは約 110MB だった。次にこれらの解からパレート解を抽出した。サポート解は 0.3 秒、非サポート解は 976 秒の時間がかかった。サポート解は 19 個、非サポート解は 61 個抽出された。

抽出されたサポート解の散布図を図 8 に示す。パレート解だけでなく非サポート解も抽出されているので、偏り無く代替案を選ぶことが可能である。

図 8 の(a)~(c)の割当を図 9 に示す。(a)は平均移動距離が最も短い割当、(c)は平均収容率が最も小さい割当、(b)はそれらの中間である。中央やや右下の地区は平均収容率の低下とともに割り当てられる小地域の数も少なくなっており、この地域の避難所のキャパシティ不足が示唆される結果となっている。

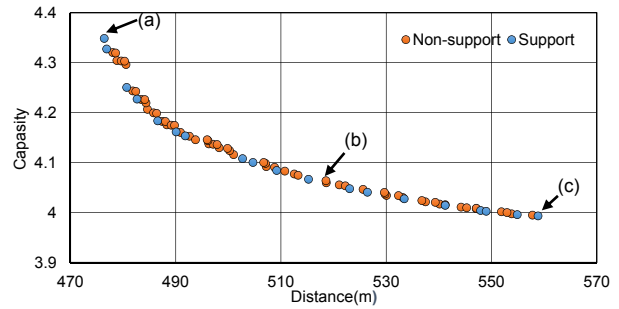


図 8 パレート解の分布

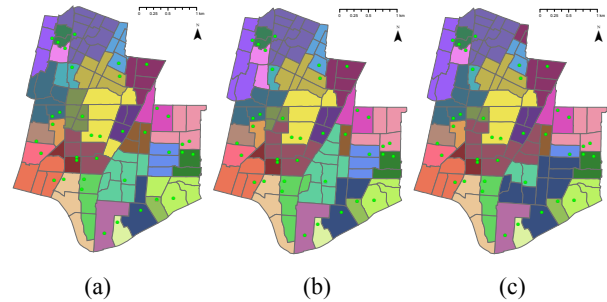


図 9 避難所割当の例

8. まとめ

本研究では ZDD を用いた小地域単位の避難所の地域割当手法を拡張し、移動距離と収容率に関する非サポート解を含むパレート解を、短時間で抽出できる事を示した。

謝辞

本研究は東京大学 CSIS の共同研究制度、科研費基盤研究(A)、CREST の援助の下で行われました。

[参考文献]

- 1) S. Minato: Zero-Suppressed BDDs for set manipulation in combinatorial problems, Proc. of the Design Automation Conference, 272-277, 1993
- 2) 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
- 3) 瀧澤重志: ZDD を用いた小地域単位の避難所割り当て案の高速列挙・抽出手法, 大阪市立大学都市防災研究プロジェクト, 都市防災研究論文集, 第 1 巻, pp.69-74, 2014
- 4) A. Takizawa, Y. Miyata and N. Katoh: Enumeration of Floor Plans Based on A Zero-Suppressed Binary Decision Diagram, International Journal of Architectural Computing, 13(1), pp.25-44, March 2015.
- 5) D. E. Knuth: The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1, Addison-Wesley Professional, 2011
- 6) N. Katoh: Finding an Optimal Region in One- and Two-Dimensional Arrays, IEICE transactions on information and systems, E83-D(3), 438-446, 2000
- 7) T. Inoue et al.: Graphillion: Software Library Designed for Very Large Sets of Labeled Graphs, International Journal on Software Tools for Technology Transfer, Springer, 2014, <https://github.com/takemaru/graphillion>

*1 大阪市立大学, JST CREST 准教授 博士(工学)