ZDD を用いた立体ペントミノパズルの解の列挙
Enumerating Solutions of Three-dimensional Pentomino Puzzle Using ZDDs

鈴木 拡 †　　湊 真一 †
Hiromu Suzuki　　Shin-ichi Minato

1. はじめに

制約充足問題とは，与えられた複数の制約条件を満たす入力の組を探索する問題である．制約充足問題を解くためのアルゴリズムは，これまでにいくつも提案されてきた．実際に，バックトラック法により数秒で解を求められた例は数多く存在する．しかし，本研究で扱う方法では，全解を過不足なく求められたか否かの論証や，求めた解集合に別の制約を加えた条件での解の列挙，などといったことが効率良く行えるという利点がある．

我々はこれまでに，制約充足問題の例題としてペントミノパズルの解法を提案した [5]．本研究ではその手法を拡張し，立体ペントミノパズルに適用した場合の効率的な解法を示す．その上で基本となるのは，二分決定グラフ (BDD: Binary Decision Diagram) と，その発展であるゼロサプレス型二分決定グラフ (ZDD: Zero-Suppressed BDD) である．

2. BDD/ZDD について

2.1 BDD

BDD は，ブール関数のグラフ表現である．変数をノード，変数の値 (0/1) をそれぞれノードから出る枝とし，定数値を終端ノードとすると，二分決定木で表現できる．この二分決定木を，変数の順序を固定した状態で 2 つの簡約化規則「冗長ノードの削除」「等価ノードの共有」[1] を施すことによって既約な BDD が得られる．BDD は多くの実用的なブール関数を比較的コンパクトかつ一意に表現できる．さらに，2 つの BDD を入力とした二項演算 (AND，OR など) の結果を表す BDD も生成できる．これは，入力 BDD のサイズにほぼ比例する時間で実行できる．

2.2 ZDD

BDD はブール関数をコンパクトに表現するために考案されたものであるが，組合せ集合を表すデータ構造として見ることもできる．組合せ集合とは，「n 個の変数の中から任意個数を選ぶ組合せ」を要素とする集合である．例えば $\{ab, abc, c\}$ は，変数 a, b, c からなる組合せ集合である．このような組合せ集合データの処理を効率良く行うことのできる BDD が，ゼロサプレス型二分決定グラフ (ZDD)[4] である．

ZDD は，通常の BDD とは簡約化規則が異なる．すなわち，1 枝が終端ノード 0 を直接指しているノードを取り除き，0 枝に直結させる (図 2(b))．これにより，組合せ集合に無関係な変数のノードが自動的に除去される．したがって，BDD よりも効率良く組合せ集合を表現・操作することができる．

3. ペントミノパズルとその解法

ペントミノパズルとは，正方形 5 個を辺に沿って繋げた多角形のピースを特定の形 (長方形など) をした盤面に隙間なく詰める，箱詰めパズルの一種である．ピースは異なる形状の 12 種類が存在する．ここでは，各ピースをその形状により図 3 のようにアルファベット 1 文字で表すこととする．

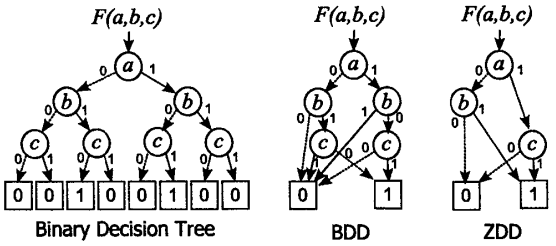


図 1: 二分決定木と BDD，ZDD

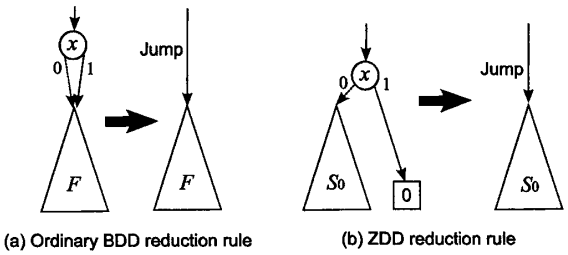
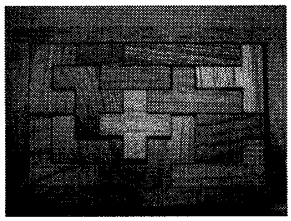
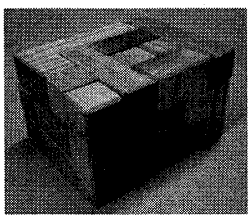


図 2: BDD と ZDD の簡約化ルールの違い

立体ペントミノパズルは，本来 2 次元平面で扱うペントミノパズルを 3 次元空間に拡張したものである．つまり，各ピースに幅 1 の厚みを持たせ，直方体状の立体盤面に隙間なく詰めるパズルである．



6 × 10 平面ペントミノ



3 × 4 × 5 立体ペントミノ

以後，ペントミノパズルを単にペントミノと呼ぶ．ペントミノは制約充足問題として定式化することができる．すでにペントミノパズルの解を探索し列挙するプログラムは開発されており，バックトラック法により，最近のコンピュータでは 1 分以内で解けることが知られている [2]．しかし，ZDD を用いて解く方法はあまり多くは研究されておらず，最近注目を集めている．ZDD に基づく解法は，Knuth が扱った「畳敷き詰め問題」の解法が

†北海道大学 大学院 情報科学研究科

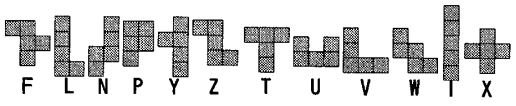


図 3: ペントミノの構成ピース

基本となっている [3]．本研究で扱うのは，これをペントミノに適用できるように独自に工夫したものである．

我々はこれまでに 6×10 サイズの平面ペントミノを ZDD に基づき解く方法を提案している [5]．

3.1 組合せ集合による解の表現

立体ペントミノの解が満たすべき制約条件を記述する．まず，「どの種類のピースをどのような向きで，どの場所に置いたか」を1つの変数で表す．ここでは，ピース自体の回転や鏡像，x/y/z 軸方向へのピースの平行移動，といったことをすべて考慮する必要がある．6×10 平面ペントミノの場合，例えば線対称でも点対称でもないピース F ならば向きは回転4通りとその鏡像を合わせて8通り．盤面上に置ける位置は32通りであるから，掛け合わせて合計256通りとなる．他の11種類も同様に合計すると 2,056 通り．つまり変数は 2,056 個必要ということになる．立体ペントミノの場合は次元が1つ増えるため，ピースの90度回転が3通り存在することに注意する．$3 \times 4 \times 5$ 立体ペントミノの場合は，変数の個数は 2,440 個にもなる．

用意した変数を，対応するピースの形状ごとに分類し変数集合をつくる．ピースは12種類あるので，12個の変数集合に分類される．それらの変数集合すべての直積をとると，12個の変数からなる組合せの集合ができる．そうしてできた膨大な数の組合せに制約条件を施し，ペントミノの解となり得る組合せのみを抽出することで，最終的に解集合が完成する．

制約条件は単純であり，「すべてのマスについて，そのマスを占有するピースがただ1つ存在しなければならない」というものである．すなわち，1つのマスに2つ以上のピースが重なってはならなく，かつ，どのマスにも空きがあってはならない．具体的には，ある1つのマス A を占有するような置き方を表す変数を含む組合せをすべて集め，それらの択一条件をとればよい．それが終われば，今度はマス B，C，...というようにすべてのマスについて制約条件を施す．こうした制約条件は，ZDD の処理系を用いて実装することができる [6]．

3.2 立体ペントミノの対称解の除去

実際は，立体ペントミノには回転対称な解が3通り存在する (x/y/z/軸に対する 180 度回転) ため，元の解と合わせると解の総数は4倍となる．さらにその各々に鏡像が存在するため，解の総数が元の8倍となり，大きく無駄が生じる．したがって，このような対称解を除去することが求められる．特定のピース1つの向きに制約を掛けることで回転対称解，ピースの置く位置に制約を掛けることで鏡像解を取り除くことができる．

まず，F や P などの対称性のないピース1つを基準とする．これらの向きは前述したように8通りある．そのうちの任意の1つおよびその90度回転1つを選び，残りの6つの向きは使用しない．こうすることで，180度回転解を取り除くことができる．残りの鏡像解は，基準としたピースの置ける位置を制限することで取り除く．つまり，ピースと垂直な軸に対する平行移動の範囲を半分に減らす (移動範囲幅4ならば1...2，幅5ならば1...3というように)．しかし，基準ピースがマス空間全体を等分割するような位置にある (奇数長の辺の中央に位置する) とき，分割面に対する鏡像解が現れてしまう．その場合は，新たに別の種類のピースに制約を掛けて鏡像解を取り除かなければならない．それでも鏡像解が残る場合は，そのような解がなくなるまで同じ処理を繰り返す．

4. 実験結果

以下に，$3 \times 4 \times 5$ 立体ペントミノの解を表す ZDD のノード数，解の個数，および実行時間を記載する．実験環境は，AMD Opteron 885(2.6GHz)，Red hat 64 Linux，主記憶 128Gbyte．ZDD の上限ノード数は10億とした．

対称解除去	ノード数	解の個数	実行時間 (分)
なし	183,399	31,520	4,394
あり	53,500	3,940	2,047

冗長な対称解が取り除かれ，実際に解の個数が丁度 1/8 になっている．

5. おわりに

本稿では立体ペントミノという例を用いて制約充足問題の解を ZDD で表現する方法を述べた．対称解を除去することで冗長な部分を省くことができたが，ピースの置き方によっては，いくつものピースに制約を施さなければ対称解を取り除けない場合もあった．できるだけ制約を掛けるピースの種類を少なくするような，シンプルな方法を探すことが今後の課題である．また，ペントミノのような娯楽的なものに限らず，他の実用的な問題についても応用できる可能性がある．今後の研究課題として，そのような問題を見つけていきたい．

参考文献

[1] Bryant, R. E.: Graph-based algorithms for Boolean function manipulation, IEEE Trans. Comput., C-35, 8 (1986), 677-691.

[2] Donald E. Knuth: Dancing links. Millennial Perspectives in Computer Science. edited by Jim Davies. Bill Roscoe, and Jim Woodcock(Houndmills. Basingstoke, Hampshire: Palgrave, 2000), 187 214.

[3] Donald E. Knuth: "The Art of Computer Programming: Bitwise Tricks & Techniques Binary Decision Diagrams" Volume 4, Fascicle 1.

[4] Shin-ichi Minato: Zero-suppressed BDDs for set manipulation in combinatorial problems, In Proc. 30th ACM/IEEE Design Automation Conf. (DAC-93), (1993), 272-277.

[5] 鈴木 拡，湊 真一: "BDD/ZDD を用いたペントミノパズルの解の列挙"，電子情報通信学会コンピュテーション研究会，信学技報 Vol.109, No.54, COMP2009-9 pp.1-7, May.2009.

[6] 湊 真一: "VSOP: ゼロサプレス型 BDD に基づく「重み付き積和集合」計算プログラム", IEICE Technical Report COMP2005-13(2005-5)