1D5-OS-02b-3in2

BDD/ZDDを用いたマインスイーパーの爆弾配置パタンの列挙

Generating All Mine Assignments of Minesweeper Using BDD/ZDD

鈴木　浩史　　孫　浩　　湊　真一
Hirofumi Suzuki　Sun Hao　Shin-ichi Minato

北海道大学　大学院情報科学研究科
Graduate School of Information Science and Technology, Hokkaido University

Minesweeper is one of the most popular puzzle game, but is also a typical example of the Constraint Satisfuction Problem (CSP). In this paper, we consider the new problem, which enumerates all possible assignments of mines for a give Minesweeper board. We propose a method using the compact data structures, BDD and ZDD, for manipulating a Boolean function or a set of combinations. In addition, we formulate the problem using a graph structure, called degree constrained subgraph, and propose a method using the ZDD-based graph enumeration technique, frontier-based search. We show experimental results of our methos for various settings of Minesweeper boards, and examine the difference among them.

1. はじめに

マインスイーパーは，Windows や X11，KDE，GNOME などの OS や GUI に付属している，1 プレイヤーゲームである．ゲームの目的は，与えられた盤面において閉じたマスに隠れている爆弾を，いくつかのヒントを頼りに，すべて見つけ出すことである．

マインスイーパーに関連する問題として，*Minesweeper Consistency Problem* [4] や *Minesweeper Counting Problem* [6] が知られている．これらの問題は，マインスイーパーの盤面が与えられたとき，可能な爆弾配置が存在するか否か，存在するならば何通り考えられるのかを問う，すなわち，決定問題および数え上げ問題の一つとして研究されてきた．

一方で，本来のマインスイーパーは，いくつかの制約のもとで可能な爆弾配置を求める問題であると捉えることができ，これは資源割当て問題や施設配置問題に通じる，制約充足問題の典型例の一つである．この観点から，本稿ではまず，上述の決定問題および数え上げ問題を含む新たな問題として，可能な爆弾配置をすべて列挙する問題を定義する．この問題を解くことは，コスト付きの爆弾が存在する場合における最適配置の計算や，あるマスに爆弾がないことの検定，各マスに対する爆弾の存在確率の計算，などの応用にも繋がる．特に，最適配置の算出は典型的なソルバで解くことが可能であるが，検定や確率計算には，列挙が基盤技術として重要である．

可能な爆弾配置を一つ求める問題であれば，バックトラック法をはじめとする素朴な探索アルゴリズムを用いれば良い．しかしながら，すべて列挙する問題になると，組合せ爆発に伴う計算時間やメモリ消費量の増加のために，素朴なアルゴリズムでは解くことが困難となってしまう．

本稿ではまず，組合せ爆発を抑えつつ，効率的に解くことが可能な，ニ種類の手法を提案する．一つめは，盤面の各マスに対して論理変数を設け，ヒントに起因する制約を基に論理関数を構築し，それを充足する真偽値 (1/0) の割当てを列挙することで解く手法である．論理関数の構築には Binary Decision Diagram (BDD) [1] と呼ばれるデータ構造を用いた．BDD は論理関数を効率的に表現することができ，かつすべての充足割当てを容易に知ることができる．二つめは，盤面の各マスを組合せアイテムとみなし，ヒントに起因する制約を満たす組合せを列挙する (組合せ集合を求める) ことで解く手法である．組合せ集合の算出には BDD を基にしたデータ構造である Zero-suppressed Binary Decision Diagram (ZDD) [5] と呼ばれるデータ構造を用いた．ZDD は組合せ集合を効率よく表現し，多種多様な演算処理を備えていることから，様々な組合せ問題に適用され成功を収めている [2][5].

本稿では，これら二つの手法について述べる他，可能な爆弾配置を求める問題が，次数制約部分グラフ探索問題と呼ばれるグラフ上の問題に定式化可能であることを述べる．この定式化を用いると，グラフ列挙技法を用いて解の列挙が可能となり，我々はその技法の一つであるフロンティア法 [3] を適用した．また，計算機実験を行い，上記の三手法について，計算時間を比較しそれぞれの特徴を調査した．

連絡先: 鈴木　浩史，h-suzuki@ist.hokucai.ac.jp

2. 定義と記法

2.1 マインスイーパーの爆弾配置パタン列挙

マインスイーパーは，閉じたマスを持つ格子状の盤面からなる (図 1)．閉じたマスを開くと爆弾，ヒント，空のいずれかのマスに変わる．爆弾マスは一つの爆弾を持ち，ヒントマスは隣接するマスに爆弾がいくつあるかを表す数字 (カウントと呼ぶ) を持ち，空マスには何もない (以降，カウントが 0 であるとして扱う)．一度の操作において，プレイヤーは一つの閉じたマスを開くことができる．ゲームの勝利条件は，すべてのヒントマスおよび空マスを開くことである (図 2)．ただし，開いたマスが爆弾マスであった場合は，即座にゲームが終了し負けとなる (図 3).

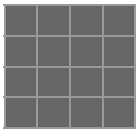

図 1: 初期状態

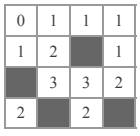

図 2: 勝ちの状態

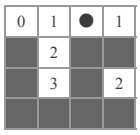

図 3: 負けの状態

我々は，マインスイーパーの途中盤面が与えられたとき，可能な爆弾配置をすべて列挙する問題を考え，爆弾配置列挙問

題と呼ぶ．例えば，図 4 に示す途中盤面には 66 個の可能な爆弾配置が存在するため，それらすべてを出力する (図 5 は出力の一つである)．与えられる盤面が，m 個の 1 から m で番号付けられた閉じたマスと，n 個の 1 から n で番号付けられたヒントマスからなるとする．ここで，隣接するマスにヒントマスが存在しないような閉じたマスは，m 個のうちに入らない．このとき，爆弾配置列挙問題を，以下のように定義する．

定義 1 爆弾配置列挙問題

入力 マインスイーパーの途中盤面 $MB(m,n,C,A)$．ここで，C は長さ n の系列 $c_1c_2\ldots c_n$ であり，c_i はヒントマス i の持つカウントである．A は n 個の整数集合 $A_1,A_2,\ldots,A_n$ からなり，A_i はヒントマス i に隣接するすべての閉じたマスの番号からなる．

出力 MB に対する，可能な爆弾配置すべて．

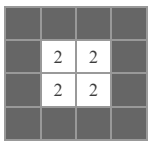

図 4: 途中盤面の例

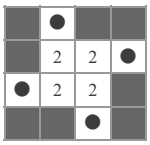

図 5: 図 4 の可能な爆弾配置

2.2 グラフ

頂点集合 V と辺集合 $E \subseteq \{\{u,v\}|\forall u,v \in V\}$ からなるグラフ G を $G=(V,E)$ と表記する．グラフ G の部分グラフとは，$V' \subseteq V$ かつ $E' \subseteq E$ であるようなグラフ $G'=(V',E')$ のことを言う．グラフ G の各頂点に対して，v に接続する辺の数 $|\{e|e \in E \text{ かつ } v \in e\}|$ を v の次数と呼び d_v と表記する．部分グラフ G' における頂点 v の次数は d'_v と表記する．

2.3 次数制約部分グラフ探索問題

グラフ G における頂点 v の次数制約とは，v が部分グラフ G' において取ることのできる次数の集合 $dc_v \subseteq \mathbb{N} \cup \{0\}$ である．G の各頂点 $\forall v \in V$ に対して，次数制約が与えられたとき，部分グラフ G' が，$d'_v \in dc_v (\forall v \in V)$ を満たすならば，G' を次数制約部分グラフと言う．次数制約部分グラフ探索問題は，以下のように定義される．

定義 2 次数制約部分グラフ探索問題

入力 グラフ $G=(V,E)$ と $dc_v (\forall v \in V)$.

出力 任意の次数制約部分グラフ $G'=(V,E' \subseteq E)$.

3. 主要なデータ構造とその応用技法

3.1 Binary Decision Diagram

BDD[1] は論理関数の二分決定木による表現を簡約化したものである (図 6)．木には根と呼ばれる一つの節点と，0-終端および 1-終端と呼ばれる二つの節点がある．各内部節点は一つの論理変数に対応したラベルを持ち，0-枝および 1-枝と呼ばれる二つの枝が伸びる．0-枝は節点の持つ論理変数に 0 を割当てること，1-枝は 1 を割当てることを表す．根から 1-終端に到達するパスは，木の表現する論理関数における一つの充足割当てを表現している．一般に，根から節点をたどっていく際の，変数の出現順序は固定されている．

BDD は、以下の簡約化規則に基づいて，二分決定木を簡約化することで得られる．

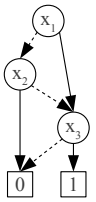

図 6: $f(x_1,x_2,x_3) = (x_1 \vee \neg x_2) \wedge x_3$ を表す BDD

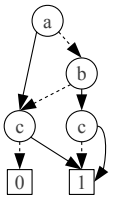

図 7: $\{\{a,c\},\{b,c\},\{b\},\{c\}\}$ を表す ZDD

共有規則 ：同じラベル，子を持つ節点を共有する．

削除規則 ：両枝が同じ子を指す節点を削除する．

すると，BDD は論理関数の一意かつコンパクトな表現となる．

BDD は，簡約化されたままで処理可能な，論理和や論理積などの様々な論理演算が整備されている．これらの演算にかかる計算時間は，二つの BDD の節点数の積に比例する．

3.2 Zero-suppressed Binary Decision Diagram

ZDD[5] は BDD を基にしたデータ構造であり，組合せ集合の表現に特化している (図 7)．ZDD の内部節点は，一つの組合せアイテムに対応するラベルを持ち，0-枝はアイテムが組合せに出現しないことを，1-枝は出現することを表す．すると，根から 1-終端へのパスは一つの組合せに対応する．

ZDD の簡約化規則は BDD と異なり，1-枝が 0-終端を指す節点を削除する，という削除規則を持つ．

ZDD は，簡約化されたままで処理可能な，和集合や積集合などの様々な集合演算が整備されている．これらの演算にかかる計算時間は，二つの ZDD の節点数の積に比例する．

3.3 グラフ列挙技法

与えられたグラフ中で，様々な構造制約を満たす部分グラフ (辺の組合せ) をすべて列挙し，その結果を表す ZDD を構築する技法として，フロンティア法が知られている [3] (図 8)．解を直接出力はせず，ZDD を構築しながらまとめて列挙することで，組合せ爆発を抑えつつ，効率的な列挙を実現する．

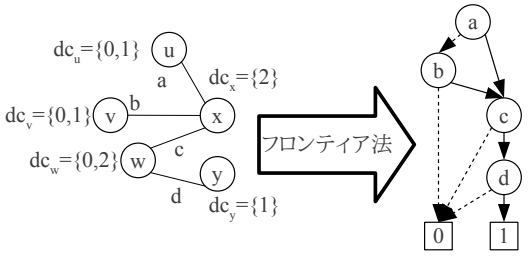

図 8: 次数制約部分グラフを列挙するフロンティア法

4. 提案手法

4.1 BDD を用いた列挙手法

与えられた盤面 $MB(m,n,C,A)$ に対して，閉じたマス i の論理変数として x_i を用意し，爆弾が配置されたときに 1，そうでないときに 0 を取ると考えると，MB における可能な爆弾配置ならば真，そうでないならば偽を取る m 変数の論理関数 $F_{MB}(x_1,x_2,\ldots,x_m)$ が存在する．我々は，F_{MB} を表現する BDD を構築することで，可能な爆弾配置の列挙を実現する．

F_{MB} は n 個のヒントに起因する論理関数を用いて構築できる．ヒントマス i に対し，A_i の中から c_i 個だけ選ぶ任意の方法について，選んだマス $j \in A_i$ に爆弾を配置したときに真，そうでないときに偽となる論理関数を考える．$x_1, \ldots, x_m$ の m 変数と整数集合 X に対して，$j \in X$ であるような x_j のうちちょうど k 個が真であるときに真，そうでないときに偽となる論理関数を $F(m, X, k)$ と表記すると，可能な爆弾配置は，すべてのヒントにおいて論理関数を真にすべきであるから，

$$F_{MB} = \wedge_{i=1}^{m} F(m, A_i, c_i)$$

である．よって，各 i に対して $F(m, A_i, c_i)$ を表現する BDD を構築できれば，それらの論理積で F_{MB} の BDD を得られる．$F(m, A_i, c_i)$ の BDD は，組合せ数 $\binom{a}{b}$ を計算するアルゴリズムに類似した，メモ化再帰の技法により効率的に構築できる．

4.2 ZDD を用いた列挙手法

与えられた盤面 $MB(m, n, C, A)$ に対して，閉じたマス i の組合せアイテムとして x_i を用意し，爆弾が配置されたときに組合せに出現し，そうでないときに出現しないと考えると，MB の可能な爆弾配置を表す組合せからなる集合 S_{MB} が存在するはずである．我々は，S_{MB} を表現する ZDD を構築することで，可能な爆弾配置の列挙を実現する．

S_{MB} は，n 個のヒントに起因する組合せ集合に基づいて構築することができる．ヒントマス i に対し，A_i の中から c_i 個だけ選ぶ任意の方法について，選んだ $j \in A_i$ に対して x_j が出現するような任意の組合せからなる集合を考える．m 個の組合せアイテム $x_1, \ldots, x_m$ と整数集合 X に対して，$j \in X$ であるような x_j のうちちょうど k 個が含まれる組合せからなる集合を $S(m, X, k)$ と表記すると，可能な爆弾配置は，すべてのヒントに共通する組合せであるから，

$$S_{MB} = \cap_{i=1}^{m} S(m, A_i, c_i)$$

である．よって，各 i に対して $S(m, A_i, c_i)$ を表現する ZDD を構築できれば，それらの積集合で S_{MB} の ZDD を得られる．$S(m, A_i, c_i)$ の ZDD も，組合せ数 $\binom{a}{b}$ を計算するアルゴリズムに類似した，メモ化再帰の技法により効率的に構築できる．

4.3 次数制約部分グラフとグラフ列挙技法の利用

与えられた盤面の可能な爆弾配置を一つ見つける問題を考えると，その問題の解をすべて列挙することで爆弾配置列挙問題に答えられる．我々は，この問題に対して，次数制約部分グラフ探索問題を用いた定式化を行うことで，BDD/ZDD を用いた演算技法とは異なる，グラフ列挙技法 (フロンティア法) の適用を可能にする．

盤面 $MB(m, n, C, A)$ に対して，頂点集合 B および H と辺集合 E を持ち，各 $i \in \{1, 2, \ldots, m\}$ について $b_i \in B$，各 $j \in \{1, 2, \ldots, n\}$ について $h_j \in H$ であり，$i \in A_j$ であるときに $\{b_i, h_j\} \in E$ であるような，グラフ $MG = (B \cup H, E)$ を構築する．つまり，MG は，閉じたマスを表す頂点集合と，ヒントマスを表す頂点集合からなる二部グラフであり，隣接するマスを表す頂点の間に辺が張られている．閉じたマス i は，爆弾が配置されたとき隣接するヒントマスすべてに一つの爆弾を与え，ヒントマス j は，隣接するマスに c_j 個の爆弾を必要とすると考え，MG に対して次数制約

$$dc_{b_i} = \{0, d_{b_i}\}, \quad dc_{h_j} = \{c_j\} \tag{1}$$

を与えることで，次数制約部分グラフ探索問題として定式化することができる．得られた次数制約部分グラフから爆弾配置に変換するには，次数が 0 でない頂点を持つ閉じたマスに爆弾を配置すればよい．すると，以下の定理が成立する．

定理 1 (1) による MG の次数制約部分グラフと，可能な爆弾配置の間には一対一対応がある．

証明. 定式化した問題の解集合を MG_{dc}，MG_{dc} 中の部分グラフから得た爆弾配置の集合を MB_{dc}，可能な爆弾配置の集合を MB_{valid} とする．$MB_{dc} = MB_{valid}$ を示せば良い．

まず，$MB_{dc} \subseteq MB_{valid}$ を示す．MG_{dc} 中の部分グラフにおいて，次数が 0 でない頂点 $b \in B$ はすべての隣接する頂点間に辺を持ち，すべての $h \in H$ が次数制約を満たすことから，ヒントマス i に隣接してちょうど c_i 個の爆弾が配置されることになる．よって，$MB_{dc} \subseteq MB_{valid}$ である．

次に，$MB_{dc} \supseteq MB_{valid}$ を示す．ある可能な爆弾配置に対して，爆弾が配置されたマスの頂点集合 $B' \subseteq B$ と H からなる，MG の誘導部分グラフを考える．この誘導部分グラフは，辺集合 $E' = \{e | e \in E$ かつ $b \in e$ かつ $b \in B'\}$ を持つので，すべての $b \in B$ は次数制約を満たしている．さらに，ヒントマス i に隣接してちょうど c_i 個の爆弾があるから，すべての $h_i \in H$ は c_i 個の頂点と隣接し，次数制約を満たす．従って，この誘導部分グラフは MB_{dc} に含まれるので，$MB_{dc} \supseteq MB_{valid}$ が示された．以上より，定理が示された． □

よって，定式化した問題の解を，フロンティア法を用いて列挙することで，爆弾配置列挙問題に答えることが可能である．

5. 計算機実験

計算機実験により，各手法の比較を行い，それぞれの性質を調査した．実験環境は，CPU が Core i7-3930K 3.2GHz で 64GB のメモリを持つ．実験プログラムは，C++で記述し g++ で最適化オプション O3 を用いてコンパイルした．

インスタンスは，すべて格子盤面であり，爆弾の割合が $10\%, 20\%, 30\%$ のもの，さらにその中で既知のヒントの割合が $10\%, 30\%, 50\%$ のものを，それぞれ 10 個ランダム生成している．評価は 10 個のインスタンスに対する平均，最小，最大の計算時間によるが，次数制約部分グラフ探索問題への定式化およびフロンティア法を用いる手法では，入力からグラフと次数制約を構築する時間も含めている．結果は表 1 に示しており，最も高速だったものは太字になっている．ここで，bdd は BDD を用いた手法を，zdd は ZDD を用いた手法を，dc は定式化とフロンティア法を用いた手法を表す．

bdd は，多くのインスタンスで最速ではないものの，安定して高速な計算時間を実現している．これに対して，zdd は，各インスタンスで安定した計算時間を実現しているが，最も遅いという結果となった．この原因は，BDD による $F(m, X, k)$ の表現と，ZDD による $S(m, X, k)$ の表現における，節点数の差であると考えられる．なぜならば，X に起因する変数のみで 1/0 や有無の選択が生じ，他の変数はそれが問われないため，BDD の削除規則がより多くの節点を削除でき，演算が高速になりやすいからである．一方で，dc は，ヒントの割合が 50% 付近の場合に，爆弾の割合の増加による速度の著しい低下が見られ，速度の安定性も失われているが，多くのインスタンスで最速かつ安定した結果を示している．

以上の結果から，bdd による手法は様々な設定で無難な手法であると言える他，次数制約部分グラフ探索問題への定式化は，序盤にあるような途中盤面において，bdd/zdd と比較し

表 1: 各手法の計算時間 (秒)

	爆弾	10%			20%			30%		
ヒント	手法	bdd	zdd	dc	bdd	zdd	dc	bdd	zdd	dc
10%	平均	0.0898	3.2197	**0.0061**	0.1415	2.9522	**0.0056**	0.2015	2.6064	**0.0051**
	最小	0.0724	2.6686	**0.0054**	0.0828	2.4270	**0.0051**	0.1202	2.2604	**0.0047**
	最大	0.1082	3.6271	**0.0068**	0.2779	3.4220	**0.0064**	0.2766	2.9293	**0.0063**
30%	平均	0.3087	21.7822	**0.0963**	**3.3373**	29.5592	5.5909	19.7889	42.8664	**1.5040**
	最小	0.2676	19.8257	**0.0571**	0.6909	22.1323	**0.1563**	2.2616	22.4692	**0.0693**
	最大	0.3541	23.4192	**0.1718**	**14.3177**	40.2097	32.4244	60.7200	77.6534	**4.4572**
50%	平均	0.2748	23.1147	**0.0349**	**0.4428**	34.9944	0.8787	**1.4812**	37.8803	290.3275
	最小	0.2619	22.0187	**0.0270**	0.3472	33.6787	**0.0568**	0.4979	35.7802	**0.2947**
	最大	0.2921	24.2032	**0.0426**	**1.0526**	36.9539	3.2996	**5.7594**	43.1102	2239.0358

てより有効な手法であると言える．本稿では割愛しているが，終盤の途中盤面においても，同様の結果を得ている．

BDD/ZDD が構築できると，解の数え上げを節点数に線形な計算時間で行える．実際に 30% の一部のインスタンスについて，dc の結果を用いて可能な配置の数え上げを行った結果を，表 2 に示す．例えば，爆弾 30% ヒント 30% の一例で dc の結果を用いて数え上げると，節点数 686989 に対して解の総数は 50 桁を超えており，その計算に要した時間はわずか 0.0219 秒であった．

表 2: 可能な爆弾配置の総数の計算結果

ヒント 10%	ZDD 節点数	160923
	配置の総数	68858325064509287989248000
	計算時間 (秒)	0.0064
ヒント 30%	ZDD 節点数	686989
	配置の総数	28071907380012210997011782 025178938032573644800000000
	計算時間 (秒)	0.0219
ヒント 50%	ZDD 節点数	1216
	配置の総数	19238058211473347462495060554811486584700928000000000000
	計算時間 (秒)	0.0007

同様に，BDD/ZDD を用いることで確率計算も容易に可能である．すなわち，各マスに対する爆弾の存在確率を計算でき (図 9，図 10)，これもまた節点数に線形な計算時間で行うことができる．

0.333	3	1.0
0.667	0	0.667
1	0.667	2

図 9: 確率計算の例 1

0.424	0.394	0.394	0.424
0.394	2	2	0.394
0.394	2	2	0.394
0.424	0.394	0.394	0.424

図 10: 確率計算の例 2

6. まとめ

本稿では，マインスイーパーの可能な爆弾配置を列挙する問題を考え，それに対して BDD/ZDD を用いた手法と，次数制約部分グラフ探索問題への定式化を用いたグラフ列挙による手法を提案した．各手法の特徴を調査した結果，BDD を用いた手法は多くのインスタンスで高速であり，ZDD を用いた手法は低速であるということの他，提案の定式化が序盤や終盤に見られる途中状態で，BDD よりもさらに高速に動作するということがわかった．

今後の課題としては，本研究をマインスイーパーの戦略に用いることである．特に，BDD/ZDD は演算処理だけでなくコストや確率の計算も可能であるため，各マスにおける爆弾の存在確率を容易に計算できる．さらに，制約の追加が容易であることから，ヒントが次々と開かれていくオンライン問題として捉え，それを高速に解くアルゴリズムとして利用できる．

謝辞

本研究の一部は科研費基盤 (S)15H05711 の助成による．

参考文献

[1] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, Vol. 35, No. 8, pp. 677–691, August 1986.

[2] Olivier Coudert. Solving graph optimization problems with zbdds. In *European Design and Test Conference*, pp. 224–228, March 1997.

[3] 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, 東京, Japan, 4 2015.

[4] Richard Kaye. *Minesweeper is NP-complete*, Vol. 22 of *The Mathematical Intelligencer*, pp. 9–15. Springer-Verlag, 2000.

[5] Shin-ichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. *Proc. of 30th ACM/IEEE Design Automation Conf. (DAC 1993)*, pp. 272–277, 1993.

[6] Preslav Nakov and Zile Wei. MINESWEEPER, #MINESWEEPER. http://www.minesweeper.info, 2003.