情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2009-GI-22 No.8
2009/6/26

虫食い算の非探索的解決と問題作成への応用

小谷善行†

古来からある計算パズルの一つである虫食い算について，解記述の候補における制約を用いて虫食い算パズルを非探索的に解く方法を示した．それに最小分岐を行う探索アルゴリズムを付加することにより，普通の問題がみな解ける効率的な虫食い算解法アルゴリズムを作った．さらにそれを用いて虫食い算を作成するシステムを設計し，巨大な虫食い算を作った

A Non-Searching Solving Method of the Arithmetic Puzzle “Mushikuizan” and its Application to Problem Generation

Yoshiyuki Kotani†

A method of solving the traditional arithmetic puzzle “Mushikuizan” is shown, which has non-searching solution mechanism by using constraint restriction in partial solution description. Adding the searching algorithm of minimum branching factor to this method, we design and implement a Mushikuizan solver which can solve problems of standard size effectively. We also implement a Mushikuizan generator which the solver is embedded in and which can generate huge problems.

1. ペンシルパズルと確定的パズル解法

虫食い算問題の解法アルゴリズムを論じるまえに，ペンシルパズルの非探索的解法の位置づけを論じる．スウドクやカックロ，スリザーリンク，イラストロジック（別名ノノグラム）といったパズルはペンシルパズルと呼ばれ，近年の「脳トレ」のブームで，流行している．こういうパズルは，たくさんのマス目でできていて，そのなかに入れる数字や印や線を確定させるパズルである．

一方，一般的なパズル，たとえば，箱入り娘などのスライディングパズルや，ペントミノなどの詰め込みパズル，8 queen などの数理パズルを，コンピュータで解かせる際には，探索的手法を使うのが普通である．探索的手法は，縦型探索が代表的なものであるが，たいていの場合，探索木という形で，複数の可能性にしたがって分岐しつつ,「途中まで解いた状態の空間」のなかをたどりゴールに到達する．single agent search 問題としてこれらは広く研究されている．

ペンシルパズルにおいては途中まで解いた状態の空間を非探索的に，つまり，確定的に枝分かれせずにたどって解くということが行われ，筆者らはその研究を行ってきている（[1],[2],[3],[4]）．ペンシルパズルは，一般的なパズル問題と同様に探索的手法で解くことができる．しかし，それはペンシルパズルの本質とは異なり，その非探索的解法に重要な意味がある．その理由は，第一に人間はペンシルパズルをほとんど確定的に問題を解いているということがある．「確定的」とは，可能性による分岐をしないという意味である．具体的には，推理することにより確定した数字やマス目や線を問題図のなかに鉛筆などで書き込んでいく（ペンシルパズルという名称はここから出ている）．人間は，保持できる記憶（頭の中，及び紙の上）がコンピュータに比べて極端に少ないので，枝分かれする探索的方法が苦手である．ペンシルパズルの問題自体も，確定的に解くことを想定して作られているということがある．こうした人間的手法を新しい課題として解明すべきであると考えている．第二に，問題を解くということに話を限定しても，探索的手法は，問題が大きくなると組み合わせ論的爆発という困難につきあたる．これは問題のサイズ（問題のマス目の数）に対して計算量が指数的になるため，事実上計算できなくなることである．このことに対処する方法として非探索的手法は重要である．

なお，スウドクやカックロでは，一つのマス目に可能な数字を複数書き込んで解い

† 東京農工大学大学院情報工学専攻

Department of Computer Sciences,
Tokyo University of Agriculture and Technology

ていくので，可能性により分岐させていて，探索的であるように一見みえる．しかしこれは問題を三次元で表すと，イラストロジックのようなペンシルパズルになる．つまり数字１～9 ごとに 9 層の面を用意し，その数字が埋まるマス目のところにその層の升目にしるしを入れるパズルと思えばよい．

本稿の目的は，まずはそうしたペンシルパズル的な非探索的に解く方法を，虫食い算という問題について設計すること，またその可能性を調べることである．

2. 虫食い算

加減乗除の筆算のなかにある数字を隠しておき，それを推論して正しい筆算の式を復元するパズル問題の種類があり，昔からさまざまに作られてきた．数字の隠し方として，英字や漢字仮名で数字を隠すのを「覆面算」という（このとき，同じ文字は同じ数字を表す）．数字を虫食いの穴（普通，□で表す）で隠すものを「虫食い算」という．

日本では両方とも，良く遊ばれる（[6],[7]）．このうち，虫食い算は日本独自の貴重な文化である．本稿ではこの虫食い算を扱う（本手法を覆面算に適用することも十分可能である）． 欧米では虫食算はあまり多くなく，ほとんど覆面算が一般的であり（[8],[9]），cryptoarithmetic とか alphametics 呼ばれている．

虫食い算には，主として，掛け算のものと割り算のものとがある．ここでは掛け算の虫食い算の解法アルゴリズムを扱う．その例を図１に示す．虫食い算のルールは，通常の筆算の習慣に従うような数字を□に１文字ずつ入れて正しい計算式にすることである（図１の解は図２になる）．したがって数の最上位の□には０を入れることはできない．

```
  □□
×２□
------
  □□
□□
------
□□１□
```

図１．虫食い算の例

```
  ４６
×２２
------
  ９２
９２
------
１０１２
```

図２．図１の解

3. 虫食い算解法アルゴリズムの方針

ここでは虫食い算をこのような非探索的手法で解くことを第一に行う．虫食い算はスウドクのようなペンシルパズルではない．ペンシルパズルに比べて，人はずっと多様な手がかりを駆使して解く．人間も（枝分かれは少ないが）探索的手法をとる．しかし，ほとんどの操作は，やはり確定的にマス目を決めていく．枝分かれは最小限にする工夫をしている．

ペンシルパズルとは異なり，虫食い算には確定的に解けるという保証がない．そこで，解けない場合は通常の探索を最小限使って解くことを試みる．われわれは，最小分岐探索法というアルゴリズムを提案している（[5]）．これは問題の途中までの解答経過において，そのあと選択肢により分岐する枝を，もっとも分岐が少ないものからおこなう方法である．これをどの程度使わなくては成らないか（使わなくて済むか）を検討する．

4. 虫食い算の定式化とマス目の設計

解法アルゴリズムを設計するにあたり，まず虫食い算の定式化を行う．方法は次の通りである．被乗数を $x_i...x_1x_0$，乗数を $y_j...y_1y_0$ という数字の並びで表す．ここで添え字付き変数はすべて 0 から 9 までの数字である（以下同様である）．筆算の途中の数（中段）は上から，$u_{0i}...u_{01}u_{00}, u_{1i}...u_{11}u_{10}, ..., u_{ji}...u_{j1}u_{j0}$ で表す．積（答え）の数字の並びは $z_k...z_1z_0$ で表す．すなわち，

$$x_i...x_1x_0 \quad \times \quad y_j...y_1y_0 \quad = \quad z_k...z_1z_0$$

及び

$$x_i...x_1x_0 \quad \times \quad y_0 \quad = \quad u_{0i}...u_{01}u_{00}$$

$$x_i...x_1x_0 \quad \times \quad y_1 \quad = \quad u_{1i}...u_{11}u_{10}$$

$$\ldots$$

$$x_i...x_1x_0 \quad \times \quad y_j \quad = \quad u_{ji}...u_{j1}u_{j0}$$

という関係がある．

これらの変数をマス目として，スウドクのようなペンシルパズル的なパズルとして虫食い算をとらえようというのが設計の基本的アイデアであった．ただし，虫食い算の虫食いのマス目をそのままペンシルパズル的なマス目にするのは困難である．スウドクでは行，列，または３×３正方形内に出現する数字はかならず１個ずつである，という限定的なマス目間の関係だけがあった．しかし虫食い算では「繰り上がり」によってマス目間関係が全体的なものになってしまう．

そこで**繰り上がりのマス目**というものを用意して，限定的マス目間関係だけが成立するようにする．まず，中段の掛け算の繰り上がり値として

$$c_{0i}...c_{01}c_{00}, c_{1i}...c_{11}c_{10}, ..., c_{ji}...c_{j1}c_{j0}$$

を用意する．ここで c_{JI} は，中段の数字 u_{JI} を計算するときに下の桁から来る繰り上が

Vol.2009-GI-22 No.8
2009/6/26

りの数である．また，中段の和によって最後の結果を計算するときの，足し算の繰り上がり値として

$$d_k ... d_1 d_0$$

を用意する．ここでd_Kは，積の数字z_K（積の下から K+1 桁目の数字）を計算するときに下の桁から来る繰り上がりの数である．図３に被乗数と乗数が２桁の掛け算虫食い算のマス目と対応する変数名を示す．

虫食い算問題は，この変数群への候補数字の集合として表される．例えば，図１の問題であると，図４のような候補数字で記述することができる．空いた数字の変数は候補が一つである．それ以外は，基本的には 0～9 の数字すべてであるが，筆算の決まりなどで自動的に制限される候補が除外される．

		x_1	x_0
		y_1	y_0
	c_{02}	c_{01}	c_{00}
	u_{02}	u_{01}	u_{00}
c_{12}	c_{11}	c_{10}	
u_{12}	u_{11}	u_{10}	
d_3	d_2	d_1	d_0
z_3	z_2	z_1	z_0

図３．繰り上がりを含んだ虫食い算のマス目

		1234 56789	01234 56789
	×	2	1234 56789
	01234 56789	01234 56789	0
	0	01234 56789	01234 56789
01234 56789	01234 56789	0	
0	01234 56789	01234 56789	
01	01	0	0
1234 56789	01234 56789	1	01234 56789

図４．図１虫食い算の数字候補の初期値

5. 非探索的虫食い算解法アルゴリズム

マス目相互の制約として二つのものが基本的な制約（限定規則）となる．中段の数字を掛け算で求めるものと，結果を足し算でもとめるものとである．

5.1 二つの限定規則

掛け算についての制約は次のように表現することとした．中段の数字u_{JI}（つまり上から J 段目の数の下位から I+1 番目の数字）を計算するのに関わるのは，被乗数の数字x_Iと乗数の数字y_Jである．その積と，下からの繰り上がりc_{JI}との和を考える．その和の１の位がu_{JI}になる．その 10 の位は，上の桁への繰り上がり，$c_{J,I+1}$になる．このことは次の関係式で表せる．

$$x_I \times y_J + c_{JI} = 10 \times c_{J,I+1} + u_{JI}$$

足し算についての制約は次のように表現することとした．結果（積）の数字z_K（積の下から K+1 桁目）を計算するのに関わるのは，その真上の中段数字の縦列$u_{M,N+L-1}, u_{M-1,N+L-2}, ..., u_{M+L-2,N-1}, u_{M+L-1,N}$である．その総和と，下からの繰り上がり$d_K$との和を考える．その和の１の位が$z_K$になる．その 10 の位は，上の桁への繰り上がり，d_{K+1}になる．このことは次の関係式で表せる．

$$u_{M,N+L-1} + u_{M-1,N+L-2} + ... + u_{M+L-2,N-1} + u_{M+L-1,N} + d_K$$

$$= 10 \times d_{K+1} + z_K$$

これらの限定規則を使って，マス目の候補数字を減らす．その例を示す．図１の問題のu_{10}に関連する掛け算の限定規則は，

$$x_0 \times y_1 + c_{10} = 10 \times c_{11} + u_{10}$$

である．このとき，初期値（図４）のx_0，y_1，c_{10}，c_{11}，u_{10}の候補数字はそれぞれ{0,1,...,9},{2},{0},{0,1,...,9},{0,1,...,9}である．これらのなかで上記の式を満たしうる数字だけを残す．するとc_{11}，u_{10}の候補数字はそれぞれ{0,1}と{0,2,4,6,8}に減る．そ

のあと，仮に z_1 に関する足し算の限定規則を適用するとする．それは

$$u_{01} + u_{10} + d_1 = 10 \times d_2 + z_1$$

である．このとき式中の変数 u_{01}，u_{10}，d_1，d_2，z_1 の候補数字はそれぞれ，{0,1,...,9},{0,2,4,6,8},{0},{0,1},{1}である．このなかでやはり式を満たしうる値を残す．すると，u_{01} の候補数字は{1,3,5,7,9}に減少する．

ここでマス目の限定規則を二つ述べた．この二つは基本的な必須規則である．一方，これ以外にいくらでも限定規則を加えて良い．現在は数の最下位桁，最上位桁の処理の規則を加えている．さらに人間の解き手が使ういくつかの技法を入れる設計になっており，そうして効率化することを検討している．

5.2 解法アルゴリズム

非探索的虫食い算解法アルゴリズムは，上記のようなマス目の数字の候補を減らしていくプロセスを記述したものである．概略は下の通りである．

```
次のことを繰り返す{
    全ての限定規則について次のことを繰り返す{
        その限定規則のなかのマス目について，
            現在の候補数字の組合せのなかから
            限定規則式が成り立つ組合せの中に存在する数字だけを残す
    }
    どのマス目についても候補数字の減少がなければ，終了
}
```

この計算が終了した時点で，すべてのマス目について，候補数字が単一になっていれば虫食い算が解けたことを意味する．どれかのマス目の候補数字が複数あれば，計算に失敗したことを意味する（問題が正しくない場合にもそうなる）．

非探索的な手法で失敗する場合は存在する．候補というものがマス目に対して与えられるため，複雑な部分解を表現できないということがあるためである．

これによりスウドクなどのペンシルパズルの解き方と同様の枝分かれしない確定的な過程が実現する．これがどの程度有効か，というのが調べるべき課題であり，あとの実験を通して述べる．

確定的手法は，人間が解けるのでわかるように，組み合わせ論的爆発を生じない．計算量は，サイズ（マス目の数）に比例するか，多くても二乗に比例する程度であると思われる．結局これで，簡単な虫食い算は解けるようになった（後述）．ここまでは，スウドク，カックロなどのペンシルパズルで，数字の候補を限定していく方法と同様に，「仮定」することなしに解くということに相当する．

6. 探索的メカニズムの付加

次に，枝分かれ，つまり「仮定」して進める方法を付加した．これは縦型（深さ優先）探索として実現した．その際，一つのヒューリスティックスを採用している．つまり，仮定する数字のあるマス目を選ぶにあたって，候補の数，つまり分岐係数が最少のところを選ぶ．これについては最少分岐法と名づけて他のパズルでも研究している([5])．これにより探索木の拡大を防ぎ，効率的なアルゴリズムを実現している．以下に探索的解法システムのアルゴリズムを「探索（局面）」という手続の形で示す．非探索的解法の部分は，このなかに埋め込まれる形になっている．

解く過程はこの手続に，図２のような初期局面（初期状態候補数字データ）を引数として与えてスタートする．

```
探索（局面）{
    非探索的虫食い算解法アルゴリズムを適用
    if(それが解けた)解を表示する
    else{
        候補数字の個数が 2 以上で最も少ないマス目を選択
        for(その各候補数字について){
            探索（その候補数字を仮定した局面）
        }
    }
}
```

ここで，候補数字を仮定した局面とは，当該のマス目の候補をその数字１個だけにしたものである．現在のアルゴリズムでは，この仮定を行うのは非乗数と乗数のマス目に限定している．

7. 解法実験

まず，第一の実験として，文献[5]にある虫食い算 30 題ほどを解かせた．そのうち，非乗数×乗数が 8×8 までの問題は，みな解けた．解いた問題の一例を図 5 に示す。以下，AMD athlon64，2.19GHz のマシンによる結果である．

Vol.2009-GI-22 No.8
2009/6/26

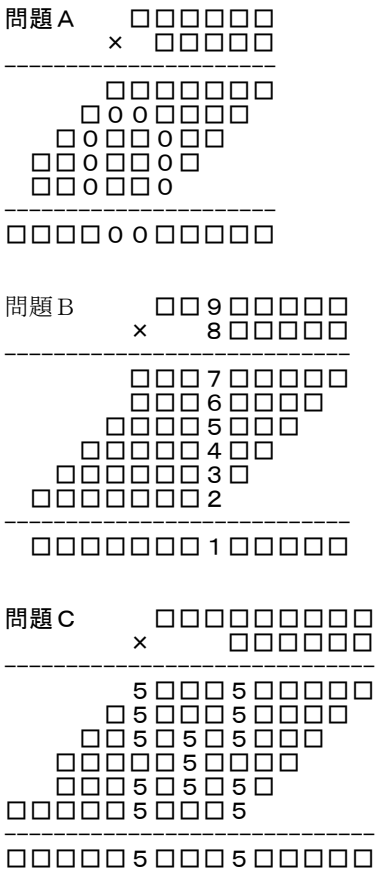


図５．解答の問題（[6]）

表１．　例題の解答状況

問題	計算時間（秒）	探索ノード数	探索木深さ	最大分岐幅
A	0.89	14	4	3
B	2.44	401	7	9
C	0.20	15	3	4

表 1 がそれらを解く際の計算時間，探索のノード数，探索木の深さ，最大分岐幅である．これらでは，非探索的解法アルゴリズムだけでは解けなかった．

しかし生成ノード数が非常に少なく，探索木が小さいことが分かる．現在の所上記アルゴリズムでは，10 桁×10 桁あたりから時間内に解けなくなる問題が出てくる．計算時間は問題により非常にばらついている．

次に第二の実験として，自動的に生成した問題について解答させた結果を説明する．問題を生成する方法については次節で述べる．問題は，非乗数×乗数が，2×2 から 6×6 までサイズを変えた 5 通りを調べた．それぞれ問題を 20 題ずつ作成した．　その結果は表 2 にまとめてある．解答率は 100%である．全 100 題に正しく答えを出した．

非探索的に，つまり分岐しないで，問題を解いたかどうかについては，20 題中で，非乗数×乗数が 2×2 で 12 題非探索的に解いた．しかし 3×3 では 20 題中 3 題のみなり，それより大きいサイズの問題はほとんど解けていない．探索アルゴリズムで，根ノードだけ分岐して解けたのは，2×2 で 6 題あり，分岐しないものと合わせると 90%になった．これらを図 6 に示す．結局簡単な問題，2×2～3×3 のサイズが非探索的に解けたことがわかる．

探索ノードについては図 7 に示す．通常の探索問題に比べノード数が非常に少ないことがわかる．とくにその中央値は，この範囲では指数的に増えているようには見えない．またノード数の平均が中央値に比べて非常に大きいことからもわかるように，ノード数のバラツキがに大きい．また極端に時間がかかる問題が少し混ざっている．

探索の木の深さも浅い．なにも工夫しない探索アルゴリズムであれば，探索深さは，「非乗数の桁数＋乗数の桁数」になる．その値の半分以下になった（図 8）．探索木のなかの分岐の枝数の最大値は図 9 の通りである．工夫しないアルゴリズムであれば，これは 10 になるものである．これはこの範囲では 6 あたりで抑えられている．

表２．　非乗数・乗数のサイズを変えた解答実験の結果

被乗数・乗数の桁数	非探索的に解けた割合（%）	分岐１回で済んだ割合（%）	探索ノード数平均	探索ノード数中央値	最大深さ平均	最大分岐数平均
2	60	30	2.70	1.0	0.50	1.45
3	15	30	15.25	8.5	1.90	4.30
4	5	10	48.90	25.0	3.25	5.45
5	0	10	63.55	34.0	4.20	6.10
6	5	0	140.40	39.5	5.35	5.85

Vol.2009-GI-22 No.8
2009/6/26

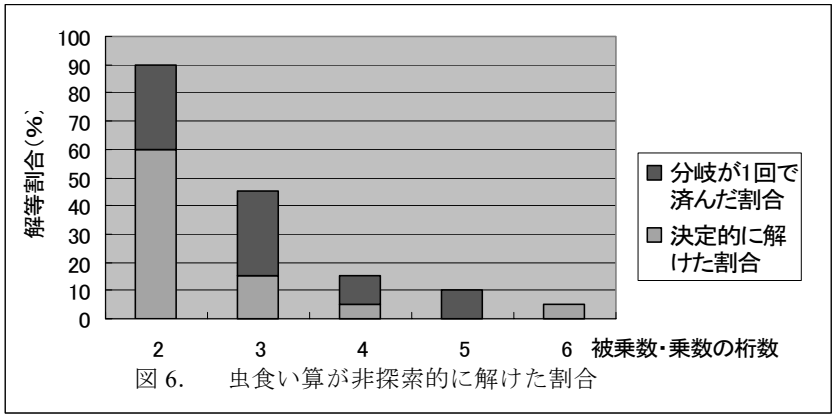


図 6. 虫食い算が非探索的に解けた割合

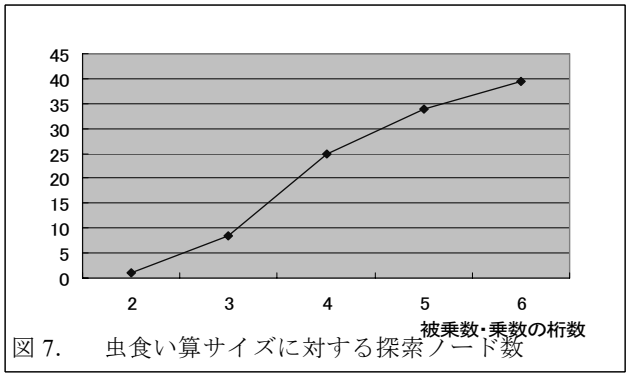


図 7. 虫食い算サイズに対する探索ノード数

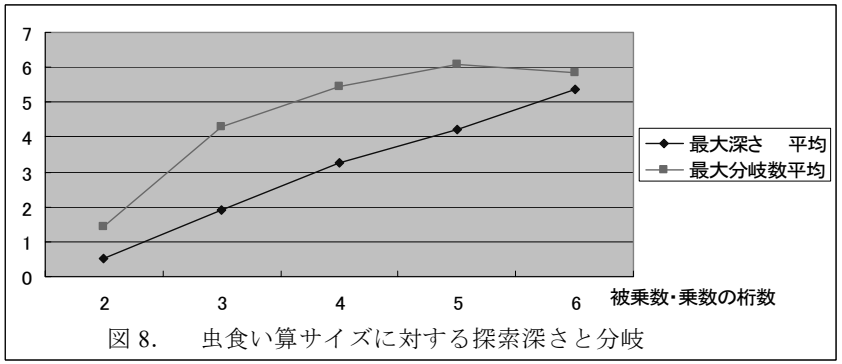


図 8. 虫食い算サイズに対する探索深さと分岐

8. 虫食い算作成アルゴリズム

虫食い算を高速に解くシステムができあがったので，それを用いて虫食い算を生成することを考えた．とくに非常に大きい虫食い算を作ることを試みた．

これは全部虫食い（□）だけでできている問題から出発して，虫食いを減らしていく方法である．数字すべてを虫食いにした問題は無数に答えがある．そこから，虫食いを減らしていき，答えが一つしかない問題に至るようにする．その際，なるべく虫食いが多く残るようにする．虫食いがなるべく多い方が問題として難しく，良い問題といえる．虫食いが意味無く少ないのは冗長であり，良い問題でない．そのアルゴリズムは下の通りである．

```
筆算のデータをランダムに作る
その中のマス目を全部虫食い（□）にした問題を作る
以下を繰り返す{
    問題を探索アルゴリズムで解く
    その解が一つの場合，繰り返し終了
    ランダムに□の一つを元の数字に戻す
}
for（それぞれの開いた数字について）{
    数字を□にする
    問題を探索アルゴリズムで解く
    その解が二つ以上のときは□を数字に戻す
}
```

このなかで探索アルゴリズムが常時，頻繁に使われる（4 行目と 9 行目）．したがってその効率は重要である．それによって問題が成立しているか，つまり解が 1 個か，2 個以上かを判定している．ここで解の数が 2 個より多いかどうかは判定する必要がないため，2 個解を見つけた後，探索が終了するようにしている．

for 文の繰り返しまでで，一応の問題ができあがる．そのままでは冗長であるので，そして for 文の繰り返し中で冗長な数字を□に戻している．

ここでいくつかの工夫をこのなかに加えている．問題を探索アルゴリズムで解こうとするときに時間がかかりすぎることがある．そのため探索を適当なノード数で打ち切っている（1000 ノード等）．打ち切られた場合は解が 2 個以上ある場合と同様に扱っている．また，掛け算限定規則の適用のところで，候補が多いときに計算を省略して効率化を図っている．両者により，問題が少し冗長になる（数字が多く開かれる）可能性がある．

こうした工夫の元に大きい虫食い算を作ることに挑戦した．付録にこの方法で生成した虫食い算 4 題を示す．問題 1 と問題 2 は，巨大さを追求したものである．問題 3 と問題 4 は，下位桁の数字を虫食いにすることにより難しい問題になるようにしたものである．これらのうちいくつかを虫食い算の趣味家に解いてもらったところ，1 題につき半日から 1 日かかったとのことであり，問題として成立している，という報告を受けた．

9. まとめと今後の課題

掛け算の虫食い算を解き，また作るという課題に取り組んだ．その結果をまとめるとつぎのようになる．

- ペンシルパズルを解くときと同様な，非探索的に虫食い算を解く方法を設計した．その方法は，マス目に候補数字を入れ，それを限定規則で減少させていくものである．
- それのみで解く場合，非乗数×乗数が 2×2 の問題なら半分以上解けた．それ以上では少ししか解けないことが分かった．
- 最小分岐を選ぶ探索を用いる探索アルゴリズムを加えた解法を設計・実現した．
- それは通常の大きさ（8×8 まで）の虫食い算をすべて解いた．また探索木が非常に小さいなどの実験結果を得た．
- この探索アルゴリズムを用いて，虫食いを減らしていく方法で虫食い算を生成するアルゴリズムを設計実現した．
- それを用いて巨大虫食い算を作った．

今後の課題として考えられるものを下に論じる．大きい課題としては人間的エキスパート知識を限定規則の形での記述する研究がありうる．これによって非探索的解法を効率的なものにできるはずである．

人は多様な推論を駆使する．それを表現しうるようにしたい．たとえば，積の数字がすべて開いた場合，積を因数分解して乗数と被乗数を推論することが行われる．さまざまな推論技法があり，それを「定理」として記述できるような問題表現を考えたい． その成果を使うと，巨大な虫食い算を解くことも可能にできるであろう．また究極的にはすべて非探索的に解くということも目標にできよう．

これは，虫食いを減らしていく方法であった．この他に全部数字の問題から出発して，虫食いを加えていく方法も考えられる．すなわち，数字を次第に虫食いにすることを繰り返す．解が複数になったところで停止し，その直前の状態を問題とするのである．

他には，制約充足問題として虫食い算を解くことも考えられる．スウドクではすでにある（[10],[11]）．また，関連することとして，割り算の虫食い算，また覆面算も意味ある取り組む課題である．図形や文字を出したり（図 5），解に誕生日の年月日など意味ある数を隠しておくなど，芸術的風味を出すことが，実際にたいてい行われており，そうしたことをコンピュータ生成させることも面白いテーマである．

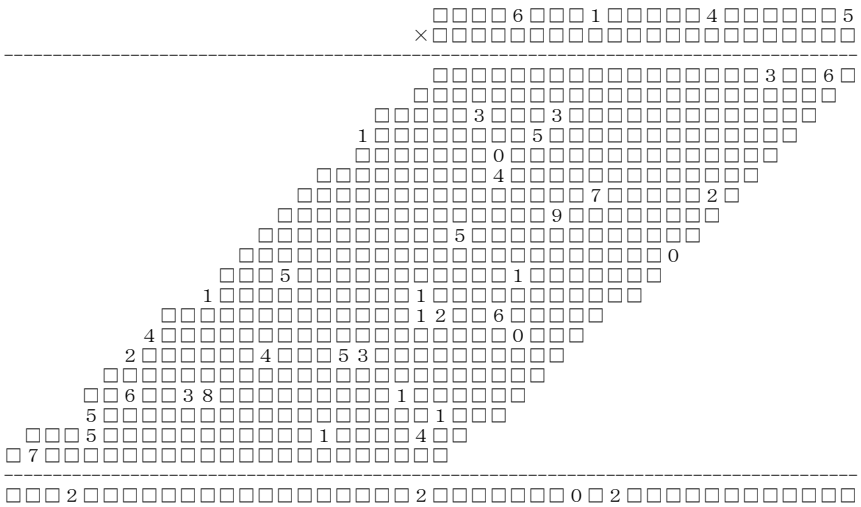
参考文献

[1] 是川空,五十嵐力,柴原一友,但馬康宏,小谷善行：ペンシルパズルにおける「解き筋」の概念の提案,Game Programming Workshop 2007 pp.99-106,2007.
[2] 是川空,白井裕己,柴原一友,小谷善行:ペンシルパズルにおける確定定理の自動獲得,Game Programming Workshop 2008 pp.112-115,2008.
[3] 乾伸雄,小谷善行：ナンプレの解法，難易度の算出，問題の作成,Game Programming Workshop 2002,pp.163-169,2002.
[4] 白井裕己,五十嵐力,但馬康宏,小谷善行:スリザーリンクの解答システムと問題作成システム,Game Programming Workshop 2006,pp.32-39,2006.
[5] Chikara Igarashi, Yasuhiro Tajima, Nobuo Inui, Yoshiyuki Kotani, A Logic Puzzle Solver by Selecting Smallest Branching Factor, IPSJ SIG Technical Reports,2005-GI-14,pp.41--45, Sep. 2005.
[6] 大駒誠一，武純也，丸尾学：虫食算パズル 700 選，共立出版，1985.
[7] 佐野昌一：推理学校虫食算大会，学生社，1968.
[8] Steven Kahan: Have Some Sums to Solve--The Compleat Alphametics Book, Baywood Publishing, 1978.
[9] Steven Kahan: At Last!!: Encoded Totals Second Addition, Baywood Publishing, 1994.
[10] Inês Lynce, Jöel Ouaknine : Sudoku as a SAT Problem, Proceedings of AIMATH 06, 2006.
[11] Helmut Simonis, : Sudoku as a Constraint Problem, In CP Workshop on Modeling and Reformulating Constraint Satisfaction Problems,pp.13-28,2005

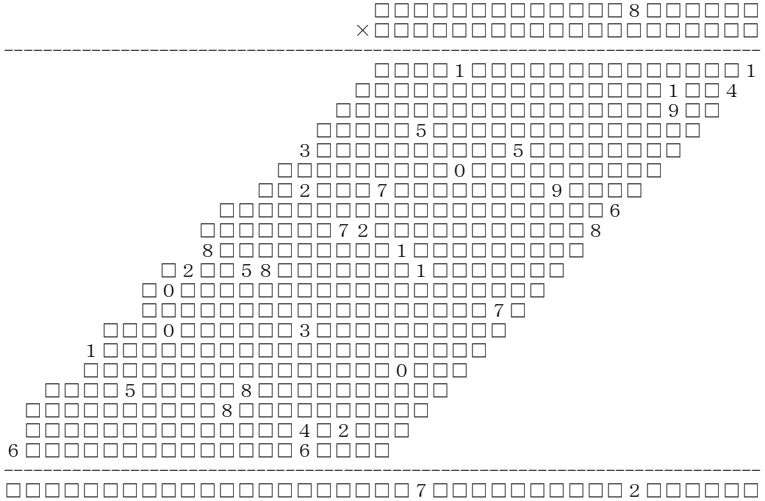
Vol.2009-GI-22 No.8
2009/6/26

付録　生成された虫食い算

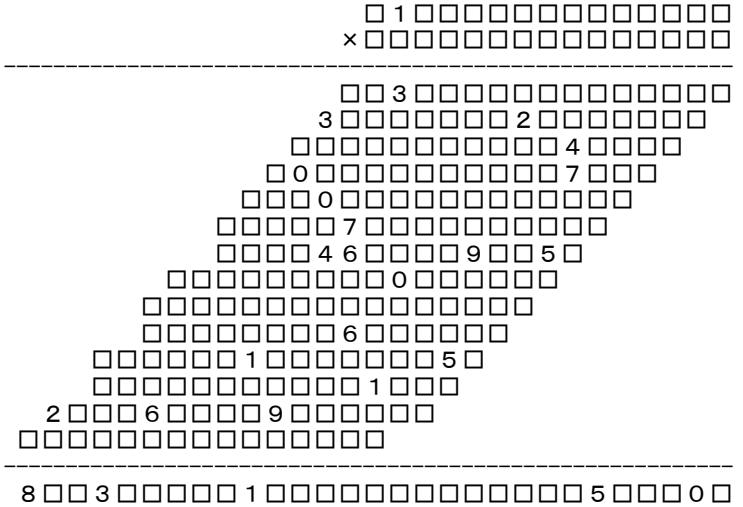
問題１



問題２



問題３



問題４