「倉庫番」の問題の自動作成

村瀬芳生[†] 松原仁[††] 平賀譲[†]

本研究ではパズル「倉庫番」の問題を自動作成するプログラムを作成した．プログラムは，問題の作成，解答，評価の3つのルーチンからなる．作成ルーチンは乱数を使ってできるだけ効率良く問題を作成し，解答ルーチンは作成されたものを幅優先探索によって解く．評価ルーチンでは，ここまでに得られたデータをもとに問題のおもしろさを評価する．この3つのフェーズを繰り返し，評価の低いものを捨て，おもしろい問題を残すことを試みた．その結果，残った中にはつまらない問題も含まれてはいるものの，難解なものを含めパズルとして成り立つ問題を創作することができた．人工知能において，コンピュータによる作品の自動作成は興味深い研究テーマである．本研究はその試みの1つとして位置付けられる．

Automatic Generation of "Sokoban" Problems

YOSHIO MURASE,[†] HITOSHI MATSUBARA[††] and YUZURU HIRAGA[†]

This paper describes a program that generates *Sokoban* problems automatically. *Sokoban* is a one-player puzzle invented in Japan. The rule is simple, but advanced problems are difficult to solve and very interesting for players. The program consists of three stages: generation, solving, and evaluation. The candidates for problems are generated randomly from a prototype and three templates. Unsolvable candidates are removed by the *Sokoban* solver. Finally trivial or uninteresting candidates are removed by the evaluator. Some problems that the program generated are judged good by human experts. Our work can be characterized as an attempt to pursue creative tasks on computers, which is increasingly becoming an important target in AI.

1. はじめに

我々はパズル「倉庫番」の問題をコンピュータによって自動作成することを試み，ある程度おもしろい問題を作成することに成功した．

「倉庫番」とは，コンピュータ上で遊ぶ1人パズルである．1982年に登場し，様々なパソコン機種，ゲーム機種に移植された．パズルは**図1**に示すように「番人」「ブロック」「荷物」「ゴール」「スペース」で構成されており，ブロックで形作られた倉庫の中を，番人を動かして荷物を押し，すべての荷物をゴールの上に運ぶことが目的である．その際，「荷物は押すことしかできない」「一度に1つしか押せない」というルールがある．初期状態から解決までの手順を「解」といい，一般に解は複数存在する．図1は例題で，(c) は解の途中の状態，(d) は解いた状態である（詳しい解は付録参照）．ルールは非常に単純だが，奥が深く，問題によっては難しいパズルになる．

図1は比較的小さな問題の例だが，既製のものはもっと大きなものが多い．**図2**はシンキングラビット社「倉庫番リベンジ」第294面[1]で，この問題は荷物が20個ある．荷物が70個以上の大規模なものも存在する．

倉庫番を扱った先行研究として，上野・中山・疋田の解答プログラム[2)~4)]がある．UNIXやパソコンで動作する倉庫番のフリーソフトも出回っており[5)~7)]，海外にも愛好者は多い．また，倉庫番関連のWWWページも多くある[8)~11)]．

本研究では倉庫番を題材に，おもしろい問題をコンピュータによって自動作成することを試みた[12)~14)]．「おもしろさ」については，意外さ，形のきれいさ，新鮮さなどいろいろな要因が考えられるが，本研究では「手順の複雑さ」に着目した．解決に複雑な手順を必要とする問題は難しく，おもしろいだろうとの予想からである．

研究の要点は，以下の3点をコンピュータによって

† 図書館情報大学
University of Library and Information Science
†† 電子技術総合研究所
Electrotechnical Laboratory

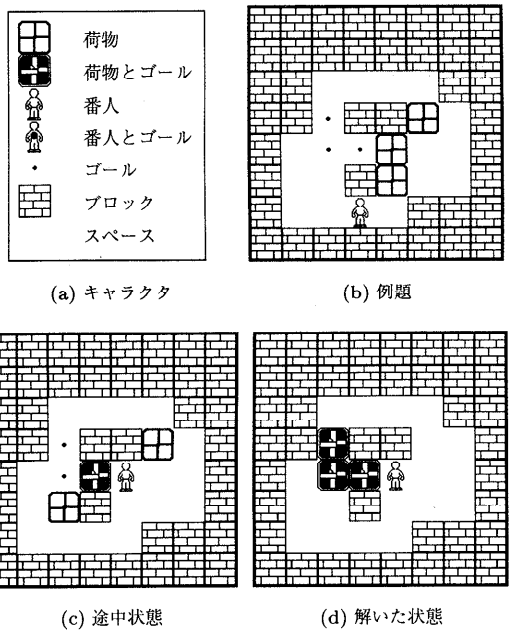


図1 倉庫番パズル
Fig. 1 Sokoban puzzle.

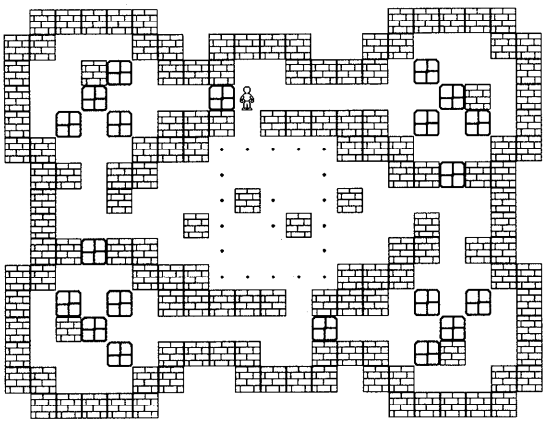


図2 大きな問題[1)]
Fig. 2 A large problem.

自動的に行うことである.

(1) 効率良く問題を作成する.
(2) 作成されたものを解き, 解のデータを作る.
(3) 得られたデータを基に, 問題のおもしろさを評価する.

図3 に示すように, 作成 → 解答 → 評価を1つの流れとして, これを繰り返す. Generate & Test 方式により, 良いものができたらそれを残そうという方針である. 作成する問題は, 盤の大きさ 8 × 8, 荷物3個とした.

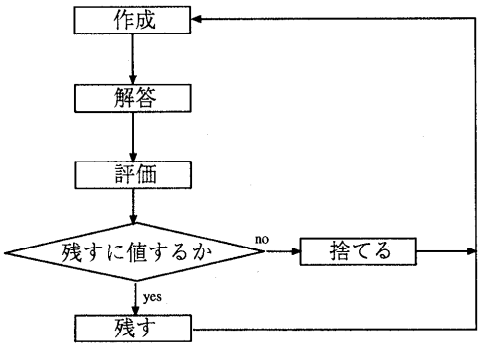


図3 創作の流れ
Fig. 3 Flow diagram of the generation.

倉庫番以外のパズルの作成・評価を扱った研究には, 詰め将棋の自動作成[15)~17)], 詰め将棋の感性評価[18)], パズル作成支援[19)]などがある.

2. 作成ルーチン

2.1 概 要

作成プログラムでは, 乱数を用いて問題を1つ作成する. その際, おもしろい問題ができる確率を高くするため, 以下のような点に留意した.

- 解のない問題をできるだけ作らない.
- 意味のない形をできるだけ作らない.

2.2 避けるべき形

倉庫番の性質上, 初期配置として明らかに意味のない形, 明らかに解けない形がある. それらは作成するときにある程度避けることができる (**図4**: 以下の(1)~(6)は図の(1)~(6)に対応する).

(1) 荷物を押し込むことができず, 番人が入る必要もないスペース. 図の×印のスペースにはゴールがなく, またそこに入れた荷物は出すことができない. またこのスペースに, 荷物を押したり移動のために番人が入る必要もない.
(2) 他のマスから荷物を押し込めないゴール. 図のゴールに荷物を入れることはできない.
(3) 初めから動かせない荷物. これらの荷物はいずれもブロックや他の荷物が障害になって, 動かすことができない.
(4) 動かすと手詰まりになってしまうことが明らかな荷物. この荷物を動かすとゴールからはずれ, しかも(3)の状態になってそれ以上動かせなくなる.
(5) 動かせる領域にある荷物の個数に見合う数のゴールがない配置. 他の荷物の存在を無視しても, 各荷物が移動できる範囲は初期配置から

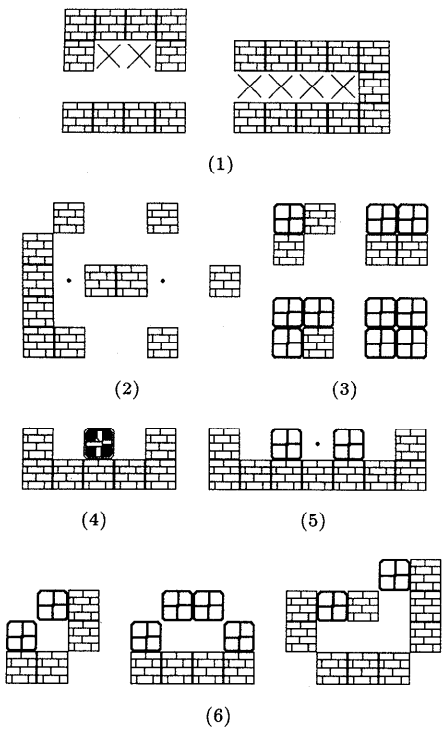


図4 避けるべき形
Fig. 4 Patterns to be avoided.

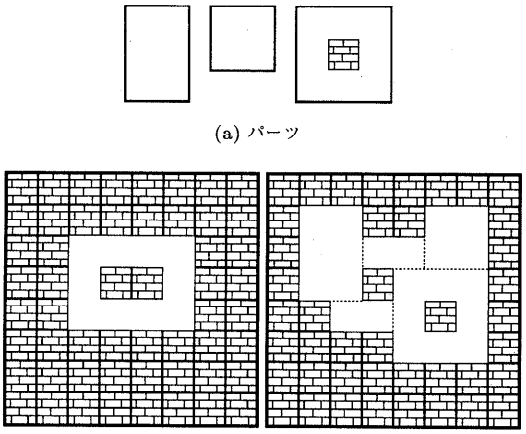


図5 輪郭作成
Fig. 5 Outline generation.

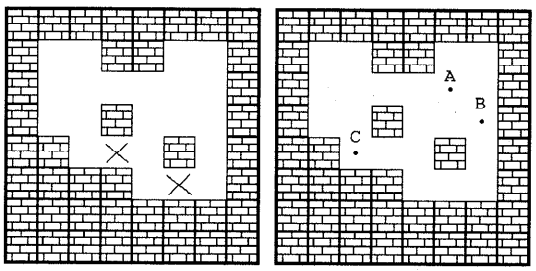


図6 ゴール配置
Fig. 6 Setting of goals.

限定される．したがってその領域内に対応するゴールがなければならない．図では2つの荷物は水平方向にしか動くことができないが，ゴールは1つしかない．

(6) 袋小路．荷物とブロックが作る閉領域で，解消できないものをいう．解消できないというのは，外側から可能な移動を重ねても，(3)の動かせない状態にしか至らないことを指す．

これらのうち(6)の「袋小路」は，閉領域が解消可能かどうかを簡単な方法で判定することは一般にはできない[2]．したがって実際に解決を試してみるしか方法がなく，「袋小路」の有無の判定は作成段階ではなく，次の解答ルーチンで行うことになる．

2.3 手　　順

問題の作成は輪郭作成，ゴール配置，荷物配置，番人配置の順に行う．

(1) 輪郭作成（図5）
初期配置を用意し，パーツをランダムに置く．パーツとは(a)に示すような部分盤面で，これが初期配置の上に「上書き」される．(a)のパーツを(b)の初期配置に重ねた結果が(c)である．パーツの形を選ぶことによって，意味のないスペースをある程度避けられる．

(2) ゴール配置（図6）
(a)の×印のところは外のマスから荷物を持っていくことができないので，ゴールを置くことはできない．そこを避けて，ランダムにゴールを配置する．

(3) 荷物配置，番人配置（図7）
荷物を置く前に，荷物を置けないゴールを求める．(a)の×印は図4の(3)より，荷物を置けないマスである．
図4(5)のような配置を避けるために，各ゴールへ移動可能な領域にそれぞれ1つずつ荷物を置く．ゴールAに動かせる領域は(b)の○印であり，その1つに荷物を置く(c)．
ゴールBについても同様である(d, e)．この結果，新たに荷物を置けないマスができた((e)の右側の壁に接した2つの×印)．
ゴールCについても，上を考慮して同様に行

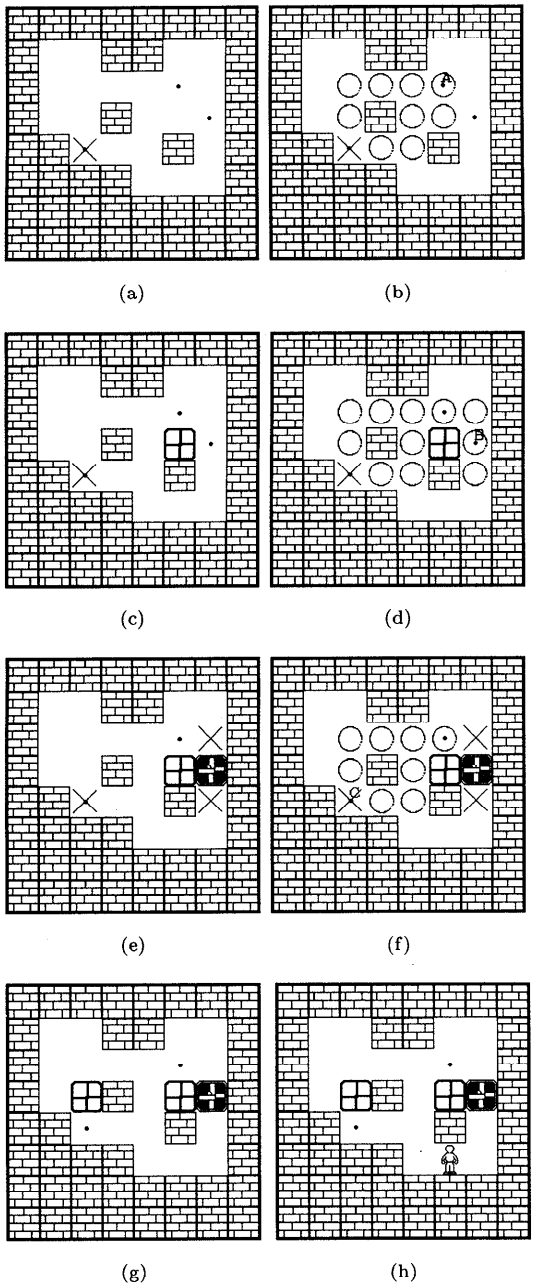


図7 荷物配置，番人配置
Fig. 7 Setting of objects and the keeper.

う (f, g)．最後に番人をランダムに置いて作成完了である (h)．袋小路がある場合，番人がその内側にいるか外側にいるかによって解の有無が分かれうる．したがって解の有無の最終判定は，次の解答ルーチンが行うことになる．

3. 解答ルーチン

作成された問題には，解があるものとないものがある．解答プログラムによって解のないものを排除し，解があるものについてはどのようにして解かれたかのデータを作成し，次の評価プログラムに渡す．

解答プログラムは上野ら[2]によるものがあるが，これは難題を含めて，問題を解くこと自体が目的のため，最短手で解くとは限らない．本研究では問題の評価のために「最良の手順」で解いたデータを必要とする．そのため，幅優先探索の解答プログラムを作成した．これは大きな問題は解けないが，解けるものは確実に最短手で解くことができる．本研究で実験した大きさ 8×8，荷物3個の問題ならば，ほとんど一瞬で解くことができる．

どのような手順が「最良の手順」であるかは一概にはいえないが，このプログラムでは以下の2つを判断基準とした．ただし (1) を (2) に優先させる．

(1) 最短手で解く．

(2) できるだけ荷物を持ち替えないで解く．

荷物の持ち替えとは，今押している荷物を離し，別の荷物を押すことを指す．同じ荷物を押す限り，番人が移動して方向転換をしても持ち替えたことにはならない．持ち替えの回数は，解の複雑さを評価するうえで重要な指針となる．

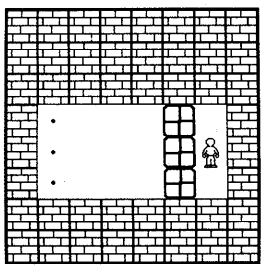
4. 評価ルーチン

人がどのような問題におもしろさを感じるかをコンピュータに判断させるのは，難しい問題である．人は解く際に何らかの戦略を持って解き，その戦略どおりに解けずに行き詰まるときに「おもしろい」と感じる，と推測できる．つまり解いている最中の試行錯誤におもしろさを感じるのである．しかし，本研究での解答プログラムは最短解を求めるための横型探索であり，人の解き方とは大きく異なるので，試行錯誤の評価には適さない．

そこで，解の手順が複雑になっている問題は，それを発見するための試行錯誤を要すると仮定し，解の手順と初期配置だけで複雑さを評価するプログラムを作成した．

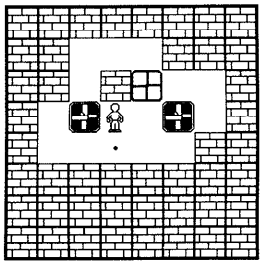
解答プログラムによって得られるデータは，手数，荷物の持ち替え回数などである．これらのデータと初期配置から得られるデータ（空間の広さ，荷物とゴールの距離など）を検討し，評価の基準として以下3つを使うことにした．

(1) 単純に荷物をゴールに近付けるだけでは解けず，



手数	12
持ち替え	3
遠回り	0
広さ	18

図8 簡単な問題
Fig. 8 Example of a trivial problem.



手数	11
持ち替え	7
遠回り	8
広さ	18

図9 おもしろい問題
Fig. 9 Example of an interesting problem.

遠回りをする必要がある.

(2) 何度も荷物を持ち替えなければならない.

(3) 盤面の大きさのわりに手数がかかる.

遠回りとは,荷物とゴールの間の見かけの最短距離よりも,解において実際に荷物が移動する距離が大きくなることを指す.距離はマンハッタン距離である.

簡単に解ける問題を図8に示す.これは各荷物を単純にゴールに近付けていくだけで解ける.荷物を1マス押すことを1手とすると手数は12手,余分な荷物の持ち替えを必要としないので,持ち替え数は荷物の数と同じ3,遠回りは0である.

一方,図9はおもしろい問題の例である.図8と比べると広さが同じで手数が1つ少ないが,持ち替えと遠回りが多い.

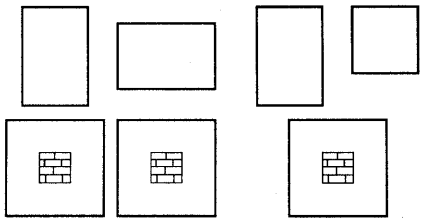
5. 問題作成実験

図10のような初期配置と2通りのパーツの組合せを用意し,作成実験を行った.それぞれの組合せで作成,解答,評価の一連の手続きをひとまとまりとして,1ループごとに評価の低い面は捨てるようにして,500回ループさせた.問題の大きさは8×8,荷物は3個とした.

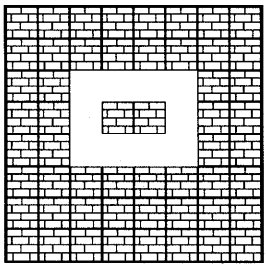
明らかにつまらない問題だけを捨て,誤っておもしろいものが捨てられることのないよう,ゆるい評価基準で実験した.具体的には,

- 遠回りせずに解ける
- 荷物の持ち替え回数が少ない(3以下)
- 解が短い(6手以内)

の3つのうち1つ以上当てはまったら捨てるようにした.



(a) パーツA (b) パーツB



(c) 初期配置

図10 パーツの組合せと初期配置
Fig. 10 Templates and a prototype.

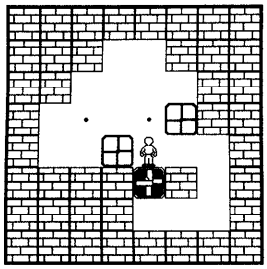
表1 実験結果
Table 1 Results of generation experiment.

パーツの組合せ		A	B
作成失敗		2	7
解なし		153	245
解あり	捨てられた	287	204
	残った	58	44
計		500	500
実際に解いてみておもしろい		13	14

実験結果を表1に示す.作成プログラムでは荷物を置く前に置けない場所を求めるため,荷物を置ける場所がなくなることがある.表の「作成失敗」はそれを指す.

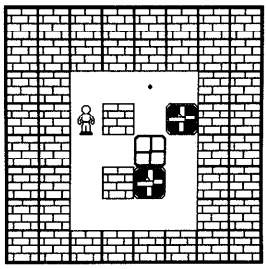
コンピュータによる評価で残った問題は,Aで58,Bで44となった.それらをすべて著者の1人が調べてみて,実際にある程度のおもしろさがあると判断したものはAで13,Bで14となった.その中でも特におもしろいものを図11に示す.この著者は倉庫番の問題を創作して商用ソフト[1]に採用された経験があり,倉庫番のエキスパートと見なすことができる.

パーツAの方が解のない問題ができにくかったのは,スペースの部分が広いためと考えられる.しかし



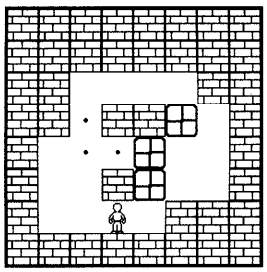
パーツ A

手数	15
持ち替え	9
遠回り	12
広さ	22



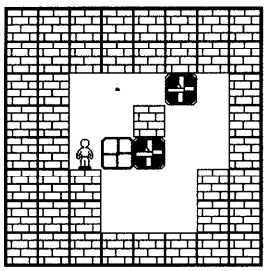
パーツ B

手数	14
持ち替え	6
遠回り	12
広さ	18



パーツ A

手数	17
持ち替え	10
遠回り	8
広さ	22



パーツ A

手数	10
持ち替え	5
遠回り	8
広さ	20

図 11 特におもしろい問題の例（パーツは図 10 参照）
Fig. 11 Examples of interesting problems generated by the system.

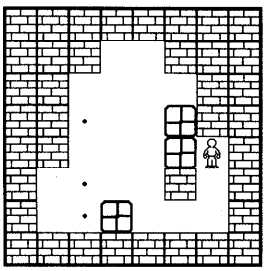
最終的におもしろい問題についてA，Bに差がなかったのは，Aは無意味に広すぎるために単純に解ける問題ができやすいためと考えられる．

捨てられた問題の例を**図 12** に示す．これは手数と遠回りは基準以上だったが，荷物の持ち替えが少ないので捨てられた．

6. 考　　察

創作された問題の中で特におもしろいものを観察すると，いくつかの特性が認められる．

- **荷物の持ち替え**
  荷物の持ち替え回数は，簡単な問題は4以下に集中しており，特におもしろいものはほとんど5以上である．したがってこのパラメータは，難しさの評価に有効といえる．
- **荷物の方向転換**
  おもしろい問題は，同じ荷物を押し続けるときでも直線的でなく，こまめに方向を変える必要がある傾向が認められる．
- **手数の多さ**
  手数が特に多い問題は，おもしろいことが多い．しかし，ただ単に荷物とゴールが遠いだけのものは，簡単に解けるのにこのパラメータが大きくなることもある．
- **遠回りの回数**
  荷物を遠回りすることなく，単純にゴールに荷物を近付けることによって解けるものは，簡単になることが多い．おもしろい問題はほとんど番人が遠回りをしている．ただし，発見しやすい遠回りもあるので，遠回りの程度がおもしろさと直結するわけではない．
- **ゴールがじゃま**
  ゴールが隅のじゃまにならないところにだけあるものは，簡単になりやすい．ゴールが道を塞ぐようなところにあると，単純にゴールに入れるわけにはいかず，おもしろさの一因となる．おもしろい問題の多くは，ゴールに荷物を入れたり出したりしなければ解けない．
- **無駄なスペースがない**
  無駄なスペースは，解を簡単にする要因になりうる．ただ，ダミーのスペースによって探索空間が増え，おもしろくなる可能性はある．
- **荷物の軌跡が複雑**
  荷物の軌跡が他の荷物の軌跡と交差したり，絡み合ったりしていると，おもしろい問題であることが多い．しかし，荷物どうしの軌跡が重ならなくてもおもしろいものは存在する．



パーツ A

手数	10
持ち替え	3
遠回り	2
広さ	26

図 12 捨てられた問題の例
Fig. 12 A discarded problem.

7. 今後の方向性

今後の方向性として，以下のようなことがあげられる．

- **解のない問題を作る率を下げる**
  効率の良いパーツの組合せでもある程度は解のない問題ができるので，解のないもののできる率を減らすことによって，よりおもしろい問題のできる確率が上がることが見込める．
- **おもしろい問題だけを選びだす**
  つまらない問題の多くは自動的に捨てることができた．しかし，残った中にもつまらないものは存在する．今回の実行例で示した方法ではおもしろいものとつまらないものをはっきり分けることはできなかったので，考察で示したようなおもしろい問題の特徴を評価基準に反映させれば，もっと精度を上げることが期待できる．
- **人間の問題作成と類似した作成方法を目指す**
  コンピュータは一度に大量の問題を作れるので，解のないものやつまらないものは捨て，おもしろそうなものだけを残す方法をとった．必然的に，「ちょっと直せば良くなる問題」も捨てられてしまうことになる．これに対し人間が作るときは，解のないものやつまらないものができた場合，それを捨てずに修正しようとする．このような方法で作ることによって，結果が変わることも期待できる．
- **問題に多様性を持たせる**
  決まったパーツを組み合わせるため，似た問題ができることがあった．パーツの種類を増やすことで，より多様な問題を創作することができる．また別の方法として，パーツを使用せずにひとマス単位で作成するようにすれば，より多様性を持たせることが期待できる．
- **「意外性」の評価を行う**
  コンピュータにとっては簡単な問題でも，人間にとって意外な，発見しにくい手が必要となるものはおもしろいと感じる．このような人間の解き方に即した評価ができれば，より正確におもしろい問題を選び出すことにつながるかもしれない．
- **きれいな形を目指す**
  難しさが同じでも問題の形が整っていたり，対称性があったりすると，人間にとってはおもしろさが増すようである．詰め将棋でいう「曲詰め」のような，形によるおもしろさの評価も興味深いところである．

8. おわりに

我々は，一度に大量の問題を作成しておもしろいものを残す方法で，パズル「倉庫番」の問題の創作を試みた．その結果 Generate & Test という単純なやり方で，ある程度効率良くおもしろい問題を創作できることを示し，計算機によるパズル問題の自動作成の可能性を明らかにすることができた．現在は規模を広げて荷物が4個や5個の実験を試みている．最近は「ロジックパズル」をはじめとして紙上でできるパズルが盛んであるが，その中には，我々のとった手法が応用できそうなものも多くある．それには各パズルでの適切な評価の指標を見い出す必要があるが，たとえば手数などの単純な評価基準でも，人よりも効率良く創作できるパズルはありそうである．そのほか，パーツを組み合わせる方法の詰め碁，詰め将棋などへの応用を期待している．本研究は，コンピュータによる価値のあるものの創作に向けての第一歩と考えている．

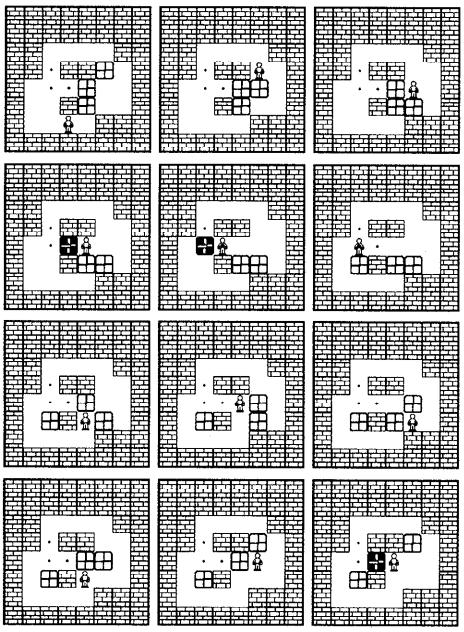
謝辞 解答プログラムを参考にさせて下さった上野篤氏，中山康氏，疋田輝雄氏に謹んで感謝の意を表する．

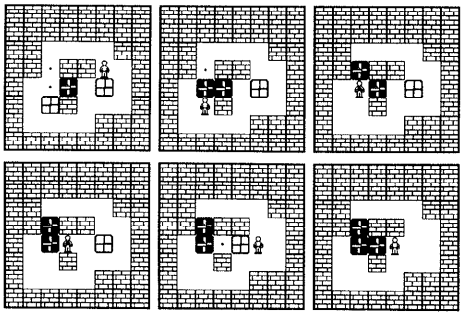
参考文献

1) シンキングラビット：倉庫番リベンジ，パソコンソフト (1991).
2) 上野 篤，中山 康，疋田輝雄：倉庫番，bit別冊ゲームプログラミング，松原 仁，竹内郁雄（編），chapter 3，共立出版 (1997).
3) 上野 篤，中山 康，疋田輝雄：「倉庫番」を解くプログラム（上），*bit*, Vol.26, No.12, pp.40–51 (1994).
4) 中山 康，上野 篤，疋田輝雄：「倉庫番」を解くプログラム（下），*bit*, Vol.27, No.1, pp.92–100 (1995).
5) Myers, A.C.: XSokoban Home Page, <http://xsokoban.lcs.mit.edu/xsokoban.html>.
6) Lindhurst, S.: Sokoban for the Macintosh, <http://math.wisc.edu/~lindhurs/sokoban/>.
7) Sleepless software: PocoMan, <http://www.sleepless.com/pocoman/>.
8) Dor, D. and Zwick, U.: Sokoban and other motion planning problems, <http://www.math.tau.ac.il/~ddorit/sokoban.ps.gz>.
9) 村瀬芳生：Resources and automatic generation of Sokoban, <http://www-student.ulis.ac.jp/~yoshio/sokoban.html>.
10) Shapiro, P.: Why I am so fond of the Sokoban logic puzzles, <http://www.his.com/~pshapiro/about.ss.html>.

11) Yossie: Sokoban, <http://www.blacksteel.com/~yossie/Sokoban/>.
12) 村瀬芳生，松原　仁，平賀　譲：「倉庫番」の問題の自動作成，第38回プログラミング・シンポジウム報告集，pp.99–106 (1997).
13) 村瀬芳生，松原　仁，平賀　譲：パズル「倉庫番」の問題を創作するプログラム，日本認知科学会第13回大会論文集，pp.76–77 (1996).
14) Murase, Y., Matsubara, H. and Hiraga, Y.: Automatic making of Sokoban problems, *PRICAI'96: Topics in Artificial Intelligence*, Lecture Note in Artificial Intelligence 1114, pp.592–600 (1996).
15) Noshita, K.: A note on algorithmic generation of Tsume-Shogi problems, ゲーム・プログラミングワークショップ'96，pp.27–33 (1996).
16) 広瀬正幸，伊藤琢巳，松原　仁：逆算法による詰め将棋の自動創作，ゲーム・プログラミングワークショップ'96，pp.34–43 (1996).
17) 広瀬正幸，伊藤琢巳，松原　仁：コンピュータによる逆算法を用いた詰め将棋の作成の試み，情報処理学会研究報告，Vol.96, No.36, pp.9–16 (1996).
18) 小山謙二：名作詰将棋の感性評価，*bit*, Vol.27, No.3, pp.26–36 (1995).
19) 瀧瀬明彦：パズル作成支援ツールの研究，図書館情報大学卒業論文 (1997).

付録　解答の手順（左から右へ）

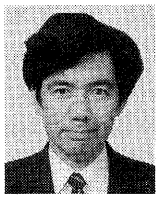




(平成9年 4 月7日受付)
(平成9年12月1日採録)

村瀬　芳生

1973年生．1996年図書館情報大学卒業．現在，図書館情報大学大学院図書館情報学研究科修士課程2年次在学．ディジタル図書館学会準会員．



松原　仁（正会員）

1959年生．1981年東京大学理学部情報科学科卒業．1986年同大学院工学系研究科情報工学専門博士課程修了．工学博士．1986年電子技術総合研究所に入所．1993年から1994年までスタンフォード大学言語情報研究センター滞在研究員．現在，電総研知能情報部主任研究官．協調アーキテクチャ，ゲームプログラミングの研究などに従事．AIUEO，人工知能学会，日本認知科学会，日本ソフトウェア科学会，日本ロボット学会などの会員．

平賀　譲（正会員）

1956年生．1979年東京大学理学部情報科学科卒業．1983年同大学院博士課程中退．同年図書館情報大学助手．現在，同助教授．理学修士．認知科学（音楽認知，認知科学の基礎），人工知能，パターン認識，マンマシンインタフェースなどに興味を持つ．著書「音楽と認知」（共著：東大出版会），訳書「コンピュータと認知を理解する」（産業図書）など．日本認知科学会，人工知能学会，ACM，ICMAなどの会員．