難しい詰めガイスター問題の生成法

石井岳史[†1,a]　川上直人[†1,b]　橋本剛[†2,a]　池田心[†1,c]

概要：ボードゲーム『ガイスター』は 6×6 のボード上で青赤 2 種 8 つの駒を交互に動かし，「脱出」「青駒全取り」「赤駒全取られ」のいずれかを狙う，互いの駒色がわからない 2 人用不完全情報ゲームである．著者らはガイスターにおけるコンテンツとして詰めガイスター問題を提案したが，生成アルゴリズムの要因から 11 手詰めまでの問題しか生成できず，さらに問題の質を評価することができなかった．そこで本稿は，生成アルゴリズムにおける必勝手探索の探索法に Df-pn を用いることで大幅に探索速度を改善し，19 手詰め問題を得ることに成功した．それに加え，元の問題から手を戻すことで新たな問題を生成する逆順生成法を用いることで，狙った手数の問題の生成を可能とした．さらに，被験者実験を行い生成した問題の面白さと難しさについてアンケートを取り，教師あり学習を行うことで特徴量から面白さと難しさの推定を行った．推定誤差は 5 段階評価の 0.5～0.6 程度で，ある程度の問題選別が可能であることを示した．

Generation of Difficult Geister Puzzle Instances

TAKEFUMI ISHII[†1,a]　NAOTO KAWAKAMI[†1,b]
TSUYOSHI HASHIMOTO[†2,a]　KOKOLO IKEDA[†1,c]

Abstract: "Geister" is a two-player, zero-sum, deterministic but imperfect information board game. Each player plays using 4 blue and 4 red pieces, and the colors are hidden from the opponent player. "Geister puzzle" is a miniature problem of Geister as chess mating problem is to chess, where there is a way the player can win if no mistake was made. In the previous paper, we proposed a way to generate Geister puzzle instances. But the employed method was slow, so the maximum number of moves to win an instance was only 11. In this paper, we improved the generation method by using df-pn, and the maximum number of winning moves was increased to 19. In addition, we proposed a reverse generation method to improve the generation efficiency. Further, we tried supervised learning for making a prediction model of interestingness/difficulty of instances. The training data were collected from experiments using human subjects. We successfully trained models which can predict interestingness/difficulty, where the root mean squared errors were around 0.5-0.6.

1. はじめに

近年，ハードウェアの進歩や新たなアルゴリズムの提案により，ゲームにおける人工知能の発展は目覚ましい．その中に人間を楽しませるコンテンツを生成する AI の研究がある．例として石飛らが楽しさに重きを置いた詰将棋問題の自動生成アルゴリズムの調査[1]について研究している．そのように将棋などの完全情報ゲームの研究が行われている一方で，不完全情報ゲームにおけるユーザーを楽しませる研究やコンテンツ生成についての研究は比較的数が少ない．

世界的に親しまれている将棋やチェスなどのボードゲームに近いルールを持ちながら，不完全情報ゲームである『ガイスター [2]』 というボードゲームがある．ガイスターは 2 人で行うゲームで，互いのプレイヤーが 2 種 4 個ずつの駒を用い，勝利条件の達成を目指すものである．駒の動かし方などは一見将棋やチェスに似通っているが，対戦相手の駒の種類がわからないようになっている．そのため，対戦相手の駒の種類をそれまでの動き等から予測する必要がある．そのことから，シンプルなルールでありながら心理戦の要素が多い．

このゲームを始めたばかりの初心者にとって，対戦相手の駒種の予測や自分の駒種を誤認させるテクニックなどの技術を身に着けることは難しい．さらには考えることが多く，本来では勝利が確定している盤面の見逃しなどが発生しうる．これらのことから初心者は理詰めよりも運や心理戦に頼りがちで，そのことが上達を困難にさせる．そのため，上達を促すためには，練習や教育のような技術向上支援を行う環境の提供が必要となる．将棋には詰将棋というコンテンツがある．これは娯楽と実戦における技術力向上の手段として用いられるパズルである．同じボードゲームであるガイスターにおいてもこのようなコンテンツは有用だと考えられる．

†1 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology
a) s1810010@jaist.ac.jp
b) s1910071@jaist.ac.jp
c) kokolo@jaist.ac.jp
†2 松江工業高等専門学校
National Institute of Technology, Matsue College
a) hashimoto@matsue-ct.jp

そこで，著者らは 2 種類の詰めガイスター問題の提案と考察[3]を行った．そこでは 2 種類の問題を定義し，生成と簡易的な評価を行った．その際は探索法の課題などから 11 手問題の生成が限度であった．

本研究の目的は，従来の問題生成法を改善し，高速化および長手数の問題生成を試みることである．さらに被験者実験を行うことで，面白さおよび難しさの推定・評価を行った．

2. ガイスター

2.1 ガイスターの概要

ガイスター（Geister）[2]は，2 人のプレイヤーが青駒と赤駒，2 種類の駒をそれぞれ 4 つずつ用いて遊ぶ不完全情報ゲームである．本ゲームの特徴として，各プレイヤーは自身の駒色を確認することはできるが，対戦相手の駒色を確認することはできないことが挙げられる．盤面は縦横に 6 マスずつ合計 36 マスとなっており，自身から見て一番奥の左右端マスをそのプレイヤーの脱出口としている．各プレイヤーはゲーム開始前に所定の範囲に 8 駒を自由に配置する．各プレイヤーは自身の手番で自身の駒のうち 1 つを縦横 4 方向のいずれかに 1 マス動かす．自身の駒が既にある方向に動かすことはできないが，動かした先に対戦相手の駒があればそれを取ることができる．その際に取った相手の駒色は確認することができる．そうして勝利条件を目指して手番をお互いに繰り返す．勝利条件は以下の 3 種類である．

勝利条件 1．自分の青駒を脱出させる（脱出口でもう 1 手番使うことで盤面の外に出すことができる）
勝利条件 2．相手の青駒を全て取る
勝利条件 3．自分の赤駒を全て取らせる

当ゲームは駒色を相手に悟られないよう動かすことが重要である．そのためシンプルなルールでありながら，人間同士のプレイでは論理だけでなく勘やハッタリを駆使した複雑な心理戦になることが多い．

2.2 先行研究

ガイスターの AI プレイヤーについては以下の先行研究がある．

末續・織田はルールベースを用いて行動を決定する AI を開発した[4]．これは対戦相手の駒の青らしさを過去の動きから評価し，それを基に予め決められたアルゴリズムで行動を決定するというものである．

佐藤は強化学習を用い，自己対戦を繰り返すことで行動価値関数を改善し，強い AI を開発することを試みた[5]．結果，AI が序盤の定石やブラフを指すことができるようになるなどの結果が得られている．

川上らは完全情報ゲームとしての探索を行うことで強さが向上するかを検証した[6]．結果，紫駒を用いた必勝判定法[5]も合わさり強力な AI となった．

一方，パズル生成やその面白さ推定も頻繁に行われる研究テーマである．

石飛らは証明数探索に用いる証明数と反証数を用いて詰将棋問題の評価を行った[1]．その結果，証明数と反証数は面白さに関わりがあることを示した．

及川らはテトリスにおける重要技術『T-spin』の詰め問題を生成した[7]．そして面白さの推定を行い，22 個の盤面特徴量を用いて面白さの平均絶対誤差 0.27 の推定を可能とした．

3. 詰めガイスター問題

詰めガイスター問題とはガイスターのルールを用いたパズルである（詳細は定義した[3]を参照）．通常のルールに加え，先手側が勝利する最短手順を見つけることが目的となる．問題として盤面，後手側の青赤駒数，最短の手数が公開される．本稿では詰め問題を図 1 のように表現する．

4. 自動問題生成アルゴリズム

4.1 従来手法

従来の問題生成アルゴリズムは以下の段階にわけた．本稿では要点のみを示す．

(1) 盤面のランダム生成

入力として互いの駒数を指定，それによって盤面をランダムに生成する．これによって様々な盤面を生成可能ではあったが，多くの生成盤面が詰問題として成立しないことが頻繁に見られた．

(2) $\alpha\beta$ 法を用いた必勝手探索

紫駒[5]の概念を導入し，色がわからない駒が存在する不完全情報盤面を全ての敵駒を紫駒として考えた完全情報盤面とし，その盤面を $\alpha\beta$ 法により探索，必勝手を発見する．ノードの評価値に深さを用いる

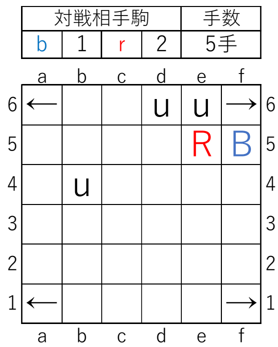

図 1．詰めガイスター問題例
Fig.1 Example of Geister’s puzzle

ことで確実に最短手数の必勝手を探し出すことができたが，深さが増すにつれ探索量は膨大となり，問題を大量生成することを考えると精々11 手程度の問題生成が限度であり，それにも駒数が多くなると 6～8 時間程度の時間を要した．

(3) 様々な条件による絞り込み

指定した手数に満たない問題や脱出口に直行するだけの問題など，簡単・単純すぎる問題の切り捨てなどを行う．その他に赤駒を壁にして解く問題など特定の問題の絞り込みもこのタイミングで行う．

4.2 提案手法

従来手法で生成した問題よりさらに難しく多様な問題を生成するためには，より長手数の問題を効率よく生成しなければならない．そこで本稿では生成法に対して以下の改善を行った．

(1) 探索法の変更

詰探索の手法をαβ法から Df-pn アルゴリズム[8]へと変更した．Df-pn は証明数と反証数を用いて，効率的に詰探索を行う手法であり，詰将棋の解探索などに用いられている．詳細は 5 章に示す．

(2) 逆順生成法の提案

盤面をランダムに生成する従来手法および，探索法を変更した(1)の手法には，長手数の問題ほど極めて生成されにくい課題があった．そこで(1)とは別に，新しい問題生成法として逆順生成法を提案する．これは既に生成された問題から両手番 1 手ずつ戻すことで 2 手深い問題を生成する手法である．詳細は 6 章に示す．

5. Df-pn による必勝手探索

5.1 Df-pn とは

Df-pn（Depth-First Proof-Number Search）[8]は，長井らによる証明数探索を改良した手法である．証明数探索同様 AND/OR 木に証明数・反証数の概念を追加しているが，それらの両方を閾値として用いることが違いとして挙げられる．さらに，ハッシュテーブルを用いることで探索した盤面の証明数と反証数を保存，再度同じ盤面を訪れた際に保存していた証明数と反証数を参照，再度の探索を省略することで探索効率を向上させている．証明数探索および Df-pn は，詰将棋などの詰探索に用いられており，True と False はそれぞれ詰み不詰みを表す．

5.2 本問題生成法における Df-pn の利用

本手法では，生成アルゴリズムにおける必勝手探索に Df-pn を用いる．従来手法同様，ランダムに得た盤面に紫駒[5]の概念を導入することで不完全情報盤面から完全情報盤面へとしている．Df-pn は詰み手が存在することを証明することはできるが，深さ優先探索の特性上最短の手順を見つけるのではなく詰み手順を見つけた時点で探索を終了してしまう．そこで本手法では反復深化を用いて，探索深さを 2 手ずつ深くしていき（対戦相手の手番で先手番が詰ませることはできないため 2 手ずつとした），詰みが証明できた時点での深さから最短手数を求めることとした．例として，5 手では詰み手が見つからず，次の 7 手で見つかった場合，その盤面は 7 手問題となる．

必勝手探索において計算するノードを減らすことは探索効率の向上に繋がる．本手法では，計算ノードを減らすために，残りの探索深さからして詰むことができない手があれば不詰としてそれ以降の探索を打ち切っている．詰めガイスター問題の一般問題において，自身の脱出口とそれに一番近い青駒とのマンハッタン距離は，脱出に必要な最低限の手数と相関がある．それを残りの探索深さと比較することで，詰むことが確実にできないかを判定する．これにより，一定手数以内に詰むことができない手を深くまで無駄に探索することを防ぐことができ，探索回数を大幅に削減した．

5.3 従来手法との探索時間比較

従来手法（αβ法）と本手法（Df-pn）の探索時間の比較を行った．用意した 3 種類の盤面をそれぞれに与え，盤面が詰みだと証明されるまでの時間を比較する．盤面はそれぞれ駒数の異なる 9 手問題である．

結果を表 1 に示す．Df-pn を用いた本手法はどの問題に対しても大きく探索時間を改善していることがわかる．Df-pn はハッシュの読み書き時間がかかってしまうが，計算ノード数が大幅に削減されたことで大きく差ができた．他のさまざまな問題に対しても検証したところ，よほど短手数の問題でない限り本手法が優れた結果を出したことを確認した．

表 1. 探索法による解答時間の比較
Table 1. Computation cost of solving problems

	解答時間[ms]		
	問題1	問題2	問題3
αβ	3452	127098	333467
df-pn	90	503	467

5.4 生成検証

本手法を用いて詰問題を生成し，生成速度および各手数における生成数を検証した．問題の設定は自分相手ともに青赤駒それぞれ 2 つずつ，19 手より深い探索は行わないとした．よって 21 手以上の問題は生成されないことになる．

結果を表 2 に示す．全生成に要した時間は約 36.6 時間である．結果が示すように，従来手法では探索時間の都合上限界であった 11 手問題が 660 問生成されており，それより長手数の問題も生成されている．しかし，長手数の問題になればなるほど生成数は大幅に少なくなっており，19 手問題に至っては 1 問しか生成されなかった．よって本手法では長手数の問題を狙って生成するのは難しいことがわかる．

5.5 まとめ

Df-pn を必勝手探索の手法として用いた本手法の生成速度の検証を行った．従来手法と比較すると大幅な時間削減を実現し，より深い問題の探索が現実的となった．さらに生成数についての検証により，従来手法では不可能であった 13 手以上の問題を生成した．しかし元の盤面はランダムで生成されることもあり，狙った長手数の問題を生成することは難しい．そこで通常の生成法とは別に，新たな手法として逆順生成法を提案する．

6. 逆順生成法

従来の生成法とは別の問題生成法として逆順生成法を提案する．これは盤面をランダム生成せずに，別に用意した問題を基に手を戻すことで新たな問題を生成する手法である．戻す手は後手番，先手番という順番になる．例として 7 手問題から 2 手戻した場合，9 手問題が生成されることになる．

6.1 アルゴリズム

逆順生成法のアルゴリズムは以下の流れになる．図 2 に例を示す．

(1) 問題を用意する

別の手法によって生成された問題を用意する．例の場合 3 手問題である．

表 2. 生成検証の結果

Table 2. Numbers of generated instances by random

問題手数	生成数	問題手数	生成数
不詰(21手以上)	121200	11	660
		13	158
3	29433	15	32
5	14495	17	7
7	4548	19	1
9	1666	合計	172200

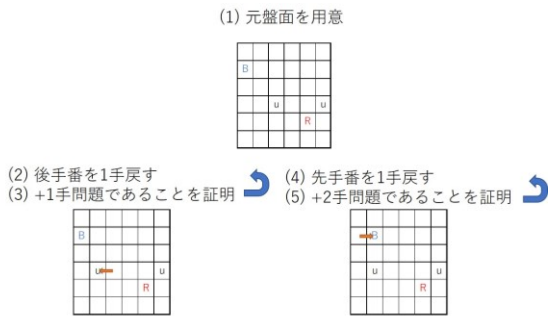

図 2. 逆順生成法の例

Fig.2 Example of reverse generation

(2) 後手番の 1 駒を戻す

対戦相手（後手番）の合法手を全て列挙し，ランダムに 1 手戻す．この際，通常の移動とは異なり移動先に何らかの駒があってはならない．移動後の盤面は一意的ではなく，①移動前のマスに駒がなかった場合　②移動前のマスに先手番青駒があった場合　③移動前のマスに先手番赤駒があった場合，の 3 通りが考えられる．これは正順で見た場合，先手番の駒が取られたか否かを考えている．

(3) 得られた盤面の最短手数を求める

(2)で得られた盤面の必勝手探索を行い，戻した手以外で先手番が元々の詰手数で詰めなくなるような手がないかどうかを調べる．もしあれば(2)に戻り別の手を探す．例の場合，4 手問題となっていれば次に進む．必勝手探索には 5 章に記した Df-pn を用いる．

(4) 先手番の 1 駒を戻す

(2)では後手番を戻したが，次は先手番を 1 手戻す．こちらは①駒がなかった場合　②後手番青駒があった場合　③後手番赤駒があった場合，に加え④⑤後手番非公開駒（青駒，赤駒）があった場合，の 5 通りを考える必要がある．

(5) 得られた盤面の最短手数を求める

(4)で得られた盤面の必勝手探索を行い，最短手数を求める．この際，元々の盤面＋2 手の必勝手を見つける必要はなく，＋1 手までの手数で必勝手を見つけることができなければ，最短手数が＋2 手であることがわかる．なぜなら(4)の時点で自身は本来の詰手数＋1 手で詰めることが確定しているため，そこから戻した手を辿れば必ず＋2 手で詰むことができるからである．しかし＋1 手までに必勝手が見つかれば，それは手数を伸ばすための戻し手が逆に手数を縮めたことになるため，(4)に戻り手を選びなおす必要がある．例の場合，5 手問題であることを証明するためには 4 手以内に詰めないことがわかればいい．これによって得られた盤面を新たな問題とする．

6.2 生成検証

逆順生成法による問題生成効率を検証した．検証項目は，元盤面から生成された逆順問題の数と生成率である．元となる盤面は 7 手～17 手の全駒種 2 つずつの問題であり，前章の手法により生成している．なお，今回の検証では 1 つの元盤面につき，最高 1 つの逆順問題しか生成していない．さらに都合上，表 2 の検証実験とは別に生成し直している．

結果を表 3 に示す．逆順生成数は元盤面から得られた該当する手数の問題数である．よって 9 手の逆順生成数 8051 とは，14124 問の 7 手（9-2 手）の元盤面から得られた逆順問題数，と見る．結果を見ると，ある手数の元盤面から得られた逆順生成数は＋2 手の元盤面数より多い．例として，既存手法で生成された 15 手の元盤面は 95 問である一方，同時間内で 13 手問題から逆順生成された 15 手問題は 191 問あるため得られた 15 手問題は合計で 286 問，逆向き生成の時間を加味しなければ同時間で 2.8 倍ほどの効率になっている．このことから，5 章での生成アルゴリズムよりも効率よく生成できていることがわかる．

6.3 まとめ

狙った手数の問題を効率よく生成する手法として逆順生成法を提案した．この生成法は元盤面を必要とするという欠点はあるが，それも通常の自動生成アルゴリズムにより収集できる．さらに+2 手だけではなく+4 手，+6 手と徐々に戻していくことで多様な問題が生成できることが期待できる．

7. 面白さ，難しさ評価

生成した問題には様々な質の物が存在する．例として，手数は長いがほぼ脱出口に直行するだけの簡単な問題，駒数が少なく一見シンプルに見えるが非常に複雑な動きを要求する問題，などである．本稿では，生成した問題に対して面白さと難しさに焦点を当て評価した．そのためにまず被験者実験を行う．そして得られたデータを基に学習モデルを生成，問題の特徴量から面白さと難しさを評価・推定した．

表 3. 逆順生成法による生成検証の結果
Table 3. Numbers of generated instances by reverse method

手数	逆順生成数	元盤面数（手数-2手）	逆順生成率
9	8051	14124	0.57
11	2510	4482	0.56
13	933	1829	0.51
15	191	416	0.46
17	29	95	0.31
19	4	20	0.2

7.1 被験者実験

プレイヤーが詰めガイスター問題を解くとき，どのような特徴を持った問題に面白さや難しさを感じるか分析するべく，ガイスターを殆どプレイしたことがない 8 人を対象とし被験者実験を行った．実験の手順は以下の通りである．

(1) 実験概要，目的，ルールの説明

(2) 予め用意した詰めガイスター問題を 1 問提示

(3) 制限時間を過ぎるか解けた時点で終了

(4) 解答例を表示

(5) 問題が面白かったか，難しかったか，解けたかどうかをアンケート評価

(6) (2)～(5)を合計 100 問分行う

被験者には図 3 のようなツールを使用してもらい，問題の解答および評価を行わせた．問題に関わる表示項目は通常の詰めガイスター問題同様，最低手数と相手の駒数，盤面である．解答には 90 秒の制限時間を設けた．評価項目である面白さ，難しさは 5 段階評価とし，解けたかどうかは 2 択とした．思考中の駒操作などの UI は，現実同様脳内で考えてもらうことが望ましいとして実装しておらず，解けたか否かは自己申告となる．

提示問題は合計 100 問，問題配分は表 4 のようにした．4 つのカテゴリに分け，考えることが他カテゴリより多い青駒全取り問題以外はほぼ均等に割り振った．手数と駒数については多すぎる，もしくは少なすぎる問題を少なめとし，中程度の問題に多く割り振った．問題の提示順番は一意的ではなく，被験者ごとにランダムとした．

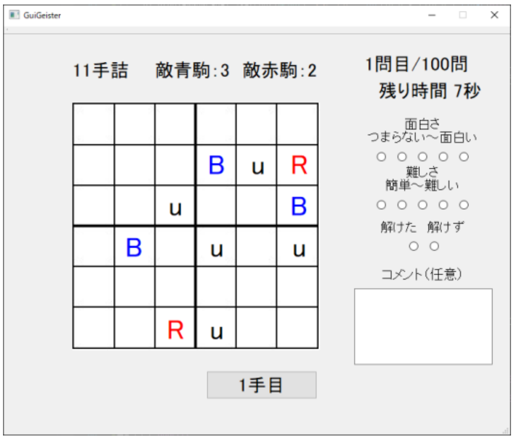

図 3. 被験者実験に用いたツール
Fig.3 Tool used for experiments using subjects

表 4. 被験者実験に用いた問題配分
Table 4. Numbers of evaluated instances

一般問題		一部公開	
青駒脱出	赤駒壁利用	青駒脱出	青駒全取り
35	25	30	10

7.2 問題の特徴量

各問題について面白さ・難しさを評価する際に必要となる特徴量を算出した．以下に特徴量を示す．

・各駒の数（自分の青駒赤駒，相手の青駒赤駒）
・問題の最低手数
・自分の青駒と自分の脱出口との最短距離
・相手の青駒（紫駒）と相手の脱出口との最短距離
・相手の駒と自分の脱出口との最短距離
・脱出口へ向かう以外の行動手数
・相手陣側に入っている自分の駒割合
・相手陣側に入っている相手の駒割合
・問題のカテゴリ
・探索中のルートノードが得た最大の証明数
・探索中のルートノードが得た最大の反証数
・詰みまでに減る駒数の最小値
・詰みまでに動かす自駒の最小値

7.3 面白さと難しさの相関

各問題における被験者全員の平均面白さ・難しさを用い，横軸を面白さ，縦軸を難しさとしてプロットを行った（図 4）．相関係数は 0.63 であり，面白さと難しさには正の相関があることがわかる．この結果から，難しい問題は面白いと評価される場合が多いことがわかる．ただし難しい問題の上位数問に着目すると，面白さはほぼ 0（-2～2）を中心としている．難しすぎる問題は面白いとは限らないということが考察できる．

7.4 被験者同士の評価差

人間はゲームの理解度などによって詰問題へ抱く印象に差が出てくると考えられる．そこで被験者同士の面白さ評価値を比較した．相関係数が一番高い比較と一番低い比較を図 5 に示す．左図の相関係数は 0.38 であり，弱い正の相関を持っている．右図の相関係数は-0.12 であり，極小ではあるが負の相関を持っている．被験者 C が面白くないと評価した問題の中でも縦軸の被験者 D は少し面白いと感じている問題が多いことがわかる．これに関して，被験者 C は被験者 D の約 1.8 倍程度の正答率があった．面白さと難しさには正の相関があるため，被験者 D が難しく面白いと感じた問題が被験者 C にとっては簡単でつまらないと感じたのだと考える．よって，面白さの感じ方は被験者の熟練度にも大きく左右されると考えられる．

7.5 面白さと難しさの推定

7.2 に示した各問題における特徴量を説明変数，被験者実験により得られた面白さと難しさを目的変数として，面白さ・難しさそれぞれの推定モデルを生成した．学習には決定木アルゴリズムに基づいた勾配ブースティングの機械学習フレームワークである「LightGBM」を用いた．さらに，過学習を抑制するため 10 分割交差検証を行った．

生成したモデルを用いて推定を行った結果を，横軸を推定値，縦軸を実値としてプロットしたものを図 6 に示す．精度を示す二乗平均平方根誤差は，面白さ（左図）については 0.52，難しさ（右図）については 0.64 となった．どちらにも正の相関が見られ，ある程度の推定ができていることがわかる．

7.6 高評価の問題

被験者に最も面白かった，難しかったと評価された問題を紹介する．図 7（左図）が最も面白かったとされた問題で，被験者全員の面白さ平均は 1.25（-2～2）であった．駒数は多いが左上部の攻防が軸となる問題で，初手が意外だったとの評価を受けた．図 7（右図）が最も難しかったとされた問題で，難しさ平均は 1.875 であった．一見簡単な問題に見えるが相手の駒を取っていいのは最悪な状況を考えると 1 駒までという制限が状況を難しくする問題である．

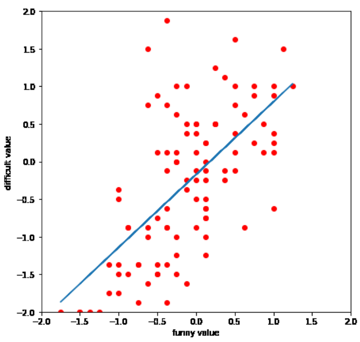

図 4. 面白さ（横軸）と難しさ（縦軸）のプロット図
Fig.4 Relation between interestingness (X-axis) and difficulty (Y-axis)

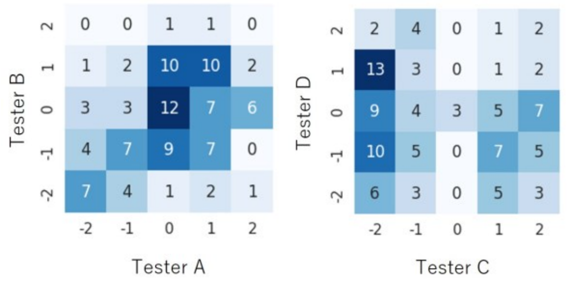

図 5. 被験者の評価相関．最大（左図）最小（右図）

Fig.5 Correlation between two subjects.
Positive case (left) and negative case (right)

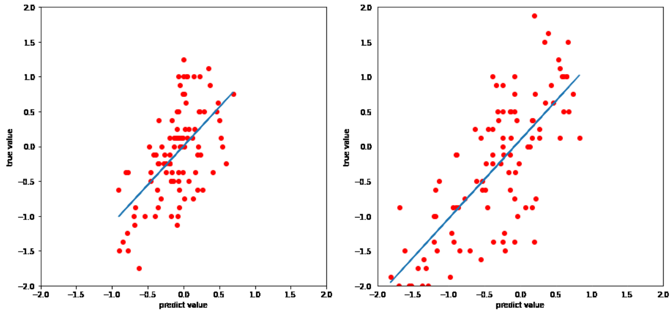

図 6. 面白さ推定（左図），難しさ推定（右図）

Fig.6 Estimation accuracy of
interestingness(left) and difficulty(right)

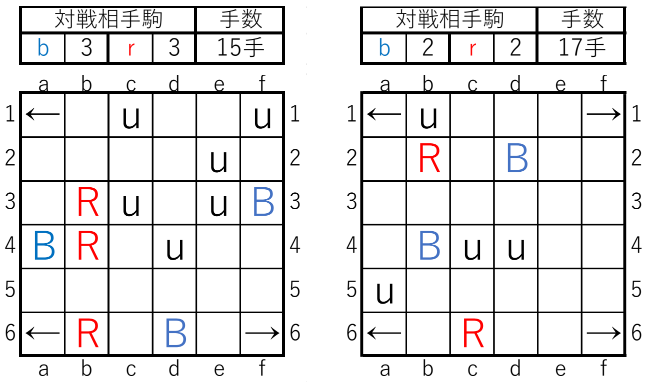

図 7. 最も面白い問題（左図），難しい問題（右図）

Fig.7 Most interesting instance (left)
and most difficult one (right)

7.7 その他の問題

その他の特徴的な問題を紹介する．図 8（左図）は長手数だが簡単で面白くないとされた問題である．手数が 15 手もあるが，青駒 1 つの脱出のための直進と敵駒 1 つの脱出妨害のみという単純すぎる問題である．一方で図 8（右図）は短手数だが面白いと評された問題である．駒数も少なくシンプルな盤面だが，青駒全取り問題でも赤駒壁利用問題でもある．

8. 別のアプローチ

詰め問題生成の別アプローチとして，後退解析，部分問題の評価利用を紹介する．

8.1 後退解析

駒数を限定すればすべての盤面を列挙できるため，互いの駒 1 つずつの 2 対 2 に限定して後退解析[9]をおこなった．2 対 2 の場合，相手駒色を 1 つ特定すれば，もう一方も定まるため，詰め問題には非公開問題，公開問題の 2 種類が考えられる．結果，非公開問題では勝ち 191992 盤面，その他 514868 盤面であり，最長勝ち盤面は図 9（左図）で 19 手であった．一方，公開問題では勝ち 783232 盤面，その他 630488 盤面であり，最長勝ち盤面は図 9（右図）で 37 手であった．公開問題の最長手数 37 手は，他生成法での最長手である 19 手よりも長い．よって，少ない駒数であれば後退解析の方が，長手数の問題をより効率的に生成できると考えられる．

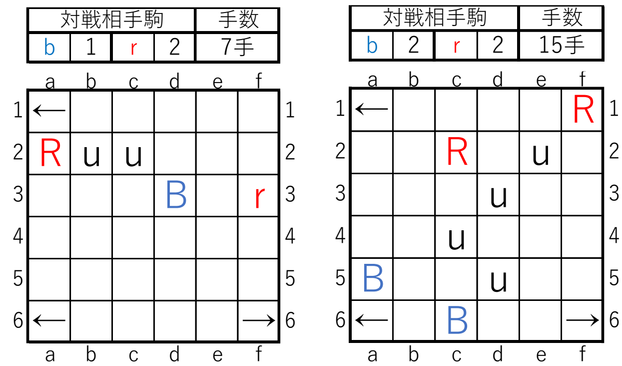

図 8.長手数でつまらない問題（左図）
短手数で面白い問題（右図）

Fig.8 Long but boring instance (left)
and short but interesting one (right)

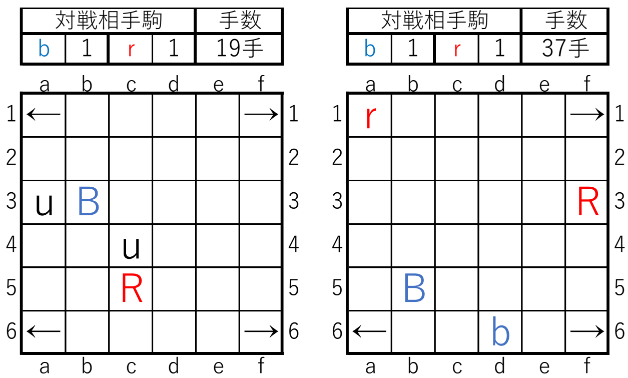

図 9. 2 対 2 での最長非公開問題（左図）
最長公開問題（右図）

Fig.9 The longest instances of 2 vs 2, imperfect
information case (left) and predictable case (right)

8.2 部分問題の評価利用

次に面白い問題の生成において，部分問題（左上 3×3 マスなど）の評価を利用できないか考えた．例えば，部分問題だけ見れば受けのない形をしているが，全体としては手数の稼ぎ合いで非自明な勝ち筋のある問題を構築できると面白いのではないかと考えた．問題例を図 10 に示す．図 10 は右下 3×3 マス（破線部）に注目すると，部分的には受けがない形である．すなわち，注目した 3×3 マス以外の駒を排除し，勝利条件を脱出に限定した場合，いずれゴールを許してしまう形である．しかし右下を放置し b2 の青駒を動かすと，青駒の脱出に対し敵青駒の脱出が 1 手早い分で負けが確定する．そこで f4 の青駒を f5 に動かすと，敵青駒がゴールするためには 4 手必要となり b2 青駒が先に脱出できるという仕掛けとなっている．このような問題を生成するために，部分問題の評価を利用できないか現在考察中である．

9. おわりに

本稿では，著者らが提案した不完全情報ボードゲーム『ガイスター』における詰め問題の，より難しい問題の生成法の提案を行った．探索法に Df-pn を用い，探索打ち切りなどの工夫を導入したところ，速度が大幅に改善され長手数の問題生成が現実的となった．それに加え，元の盤面から手を戻すことでより長い手順の問題を生成する逆順生成法を提案し，特定手数の問題生成を可能にした．

生成した問題を基に，被験者実験で被験者の感じる面白さと難しさ評価値を収集した．その評価値から，面白さと難しさにはある程度の正の相関があることや，解答者の熟練度によっては面白さを感じる問題に違いがあることがわかった．そして，問題の各特徴量を説明変数とした教師あり学習を行い，面白さと難しさを問題からある程度の精度で推定できた．

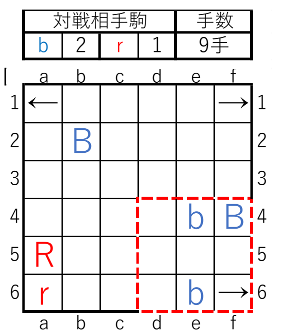

図 10．部分問題利用例

Fig.10 An instance constructed by two sub-problems

現在の問題生成法では盤面をランダムに生成している．そのため，問題生成時間が早くなったとは言え，多くの不詰盤面が生成される．そのため盤面生成をランダムではなく，解かせたい詰めの形などから駒を移動，新たに配置していくことで効率的に生成できるのではないかと考える．そのほか，面白さや難しさと相関のある特徴量を用いた生成をすることで，狙った面白さ・難しさの問題を生成したい．

現在生成できた問題の手数上限は 19 手であるが，後退解析を用いることでさらなる長手数の問題生成が期待できる．今後は 2 対 2 だけでなく，駒数を増やしても解析可能かを検証していきたい．

問題の質に関しては，AI が迷う問題は人間も迷うのではないかという仮定を基に，単純な答えを持たない問題，例えば脱出口が近いが実はもう片方の遠い脱出口に向かうことが正解である問題や，相手に駒を敢えて取らせることで勝ちに近づくことができるような「迷う問題」を探索結果から選ぶことができるのではないかと考えている．

部分問題の評価についても面白い問題生成への利用だけでなく，不偏ゲームにおける Grundy 数[10]のような概念を発見できれば，効率的な問題生成，詰み判定などに大きく貢献できると考えられる．

参考文献

[1] 石飛太一，飯田弘之，詰将棋問題の自動生成アルゴリズムに関する研究, 北陸先端科学技術大学院大学課題研究報告書, 2013

[2] “ガイスター”. http://www.mobius-games.co.jp/Gester.html, (参照 2019-02-10).

[3] 石井岳史，川上直人，橋本剛，池田心，不完全情報ゲーム『ガイスター』における 2 種の詰め問題の提案と考察，研究報告ゲーム情報学（GI），2019-3

[4] 末續鴻輝，織田祐輔，機械学習を用いないガイスターの行動アルゴリズム開発，GAT2018 論文集，pp13-16,2018

[5] 佐藤佑史, ガイスターにおける自己対戦による行動価値関数の学習，電気通信大学学術機関リポジトリ，2015

[6] 川上直人，橋本剛，完全情報ゲームの探索を用いたガイスター AI の研究，ゲームプログラミングワークショップ 2018 論文集，pp35-42，2018

[7] Taishi Oikawa et.al，Improving Human Players T-Spin Skills in Tetris with Procedural Problem Generation，The 16th International Conference on Advances in Computer Games 発表論文

[8] 長井歩，今井浩，df-pn アルゴリズムの詰将棋を解くプログラムの応用，情報処理学会論文誌，Vol.43，No.6，2002

[9] K.Thompson, Retrograde Analysis for Certain Endgames, ICCA Journal, Vol.9, No.3, 131-139, 1986

[10]“Sprague-Grundy theorem”, https://en.wikipedia.org/wiki/Sprague%E2%80%93Grundy_theorem, (参照 2019-10-8).

