構想法による詰碁生成

高木　希[1,a)]　全　炳東[2,b)]

概要：現在，囲碁や将棋などの完全情報ゲームでは，強さ以外に関する研究が盛んに行われている．本研究は，囲碁を学ぶ手段のひとつである詰碁に着目し，その自動生成を目標とする．これまでの生成に関する研究は「逆算法」を用いたものがほとんどだった．この研究では囲碁における手筋や形の構想を実現する「構想法」による生成手法を提案する．

キーワード：囲碁，詰碁，詰碁生成

Creation of *Tsume-Go* by Concept Method

Takagi Nozomu[1,a)]　Zen Heitoh[2,b)]

Abstract: This study proposes a novel approach to generate *Tsume-Go*, which is a problem based on life-and-death in the game of *Go*. Most of the conventional research uses the “inversion method”, while the proposed method utilizes the “concept method”, that realize the concepts of *tesuji* and shape in *Go*.

Keywords: *Go*, *Tsume-Go*, Creation of *Tsume-Go*

1. はじめに

囲碁や将棋などの完全情報ゲームにおいて，ゲーム AI はその分野におけるトッププロを凌駕する実力を獲得している [1][2]．現在は強さに関する研究だけではなく，人間への指導を目的としたものなど，学ぶことを支援する研究が注目されている．

本研究では，人間が囲碁を勉強する方法のひとつである詰碁に着目した．詰碁には，初心者向けから高段者向けのものまで多様な難易度の問題が存在するが，それらを作るためには高度な専門的知識が必要となる．「1 題の詰碁を作図するのは 100 題の詰碁を解くことに相当する」という言葉があるように，詰碁を創作することは非常に困難であり，プロ棋士の監修のもと作られているものがほとんどである．

これまでの詰碁生成に関する研究では，逆算法を用いたものしか提案されていない [3][4]．一方，詰将棋生成に関する研究では，逆算法の他にも遺伝的アルゴリズムを用いた手法や，将棋特有の駒の役割を利用した手法など，様々な手法によるものが提案されている [5][6][7]．将棋の駒にはそれぞれ強さや役割が存在するが，囲碁の石ひとつひとつにはそういったものが存在しない．また，詰碁は詰将棋よりも余詰め (別解の存在) の探索が困難である．これらの理由から，詰碁の生成手法は限られたものしか提案されていないと考えられる．

そこで本研究では，囲碁における手筋や形の構想を実現する「構想法」を用いた詰碁生成を行う．囲碁における手筋とは，通常より大きな効果を挙げることのできる着手のことであり，明確な形が定められている．手筋を利用した手順が解答となるよう，手筋に関する局面を与え，その局面から詰碁となる局面の生成を目指す．

2. 囲碁と詰碁

2.1 囲碁について

囲碁は，自分の色の石によって盤面のより広い領域 (地) を囲うことを目的としたボードゲームである．基本的なルールは次の通りである．

- 碁盤の縦横の路が重なる交叉点上に，自分の石と相手の石 (黒石と白石) を交互に置いていく
- 相手の石を囲むと取れる

[1] 千葉大学大学院 融合理工学府 数学情報科学専攻
Division of Mathematics and Informatics, Graduate School of Science and Engineering, Chiba University
[2] 千葉大学統合情報センター
Institute of Management and Information Technologies, Chiba University
[a)] t_nozomu@chiba-u.jp
[b)] zen.h@faculty.chiba-u.jp

- 相手の石に囲まれた点には打てない
- 最終的に囲った「地」が大きいほうが勝ち

2.2 詰碁について

詰碁は，囲碁における部分的 (局所的) な問題であり，どのように打てば自分の石を生かすことができるか，または相手の石を殺すことができるかという石の死活を問う一種のパズルである．先手後手の応酬が 1 通りであるものが詰碁の問題として望ましく，少なくとも初手は 1 か所に限定されなければならない．

詰碁では，一般的に局面と手番および問題の種類が与えられる．例えば，黒番で白の石を殺す問題は「黒先白死」，白番で白の石を生かす問題は「白先活き」と呼ぶ．また，様々な難易度の問題が存在し，「5 分で解ければ初段」といった棋力の目安が示されたものもある．

2.3 詰碁の作り方

プロ棋士らを顧問として 1999 年に発足した「詰碁を楽しむ会」(2009 年に活動終了) では，詰碁の作り方を次のように分類している [8].

検討法　生死不明の形を検討し詰碁の素材を探す
アレンジ法　実戦形や既存の詰碁から創作する
構想法　詰碁の筋や形の構想を実現する
移動法　中央の詰碁を辺や隅に移動する
逆算法　ある局面から手順を逆に戻して詰碁を創る手法
順算法　手数を延ばしながら石を追加する手法

2.4 手筋について

囲碁には石を取ったり殺したりする手筋が複数存在する．手筋には種類があり，例を以下に示す．ここで，アタリとは相手の石を完全に囲んで取る一歩手前の状態のことである．

ナカデ　相手の陣地の中に石を置くことで二眼を作らせず (二眼ができると生きとなる)，相手の石を殺す手のこと．ナカデには三目，四目，集四，五目，花五，花六，隅の曲四の 7 種類存在する．

シチョウ　アタリの連続で，逃げる側は盤がある限りは逃げることができるが，盤の端まで到達してしまうと結局取られてしまう手筋．

ウッテガエシ　アタリを回避するために相手の石を取っても再びアタリになり，結局取られてしまう手筋．

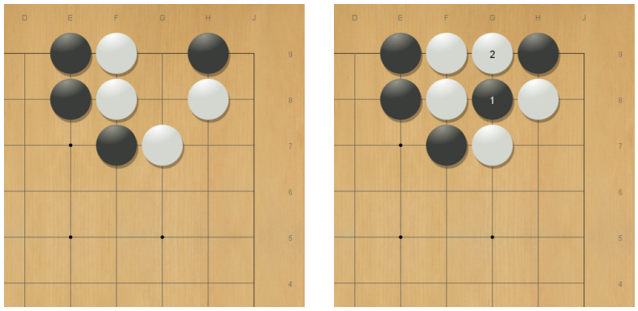
オイオトシ　アタリを回避するためにツグ手を打っても引き続きアタリになり，結局取られてしまう手筋．

石の下　意図的に相手に石を取らせて空いた交点に着手し，相手の石を取る手筋．

3. 構想法について

詰碁を生成するにあたって，[2.3] で述べた生死不明の形を扱う検討法や，特別な方針がない順算法の実装は困難である．またアレンジ法や移動法も変形が複雑となる．そこで，本研究では筋や形の構想を実現する構想法による詰碁生成を目指す．

構想法とは，あらかじめ手筋や形を定めて，その定めた構想を詰碁として実現する作り方である．ここでは，実際に人間が構想法を用いて詰碁を作るときの方法について紹介する．

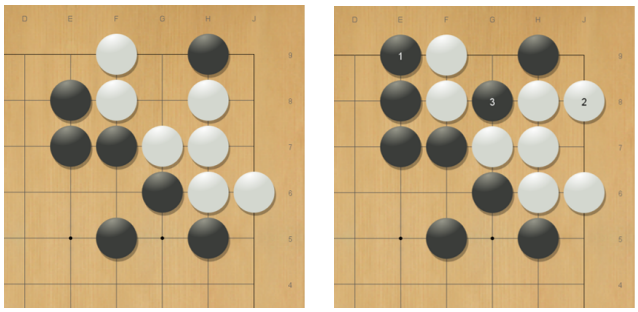


(a) 初期局面　(b) 手順

図 3.1　ウッテガエシの例

図 3.1 にウッテガエシの手筋例を示す．図 3.1(a) が初期局面である．図 3.1(b) のように，黒 1 に対して白 2 と取り返しても，黒に再び G8(黒 1) と打たれ取られてしまう．

この手順が詰碁の解答となるように，初期局面から石を増やしたり減らしたりを繰り返し，詰碁を作っていく．図 3.1(a) の局面を詰碁に変形した例を，図 3.2 に示す．



(a) 作成した詰碁の例　(b) (a) の解答

図 3.2　構想法を用いて作成した詰碁とその解答

図 3.2(a) が作成した詰碁であり，図 3.2(b) がその解答である．図 3.2(b) のように黒 1 に対して白 2 と打つと，黒 3 でウッテガエシとなる．白 2 で G8(黒 3) と打っても，黒に J8(白 2) と打たれ白石が取られてしまう．

以上が，構想法による詰碁の作り方である．

4. 提案手法

本研究では，[4.1] 詰碁の候補となる局面の自動生成を行うツリーの作成と，[4.2] 与えた石の配置がどれだけ詰碁らしいのかを判別する CNN の作成を行い，これらを用いて [4.3] 詰碁生成の実験を行った．扱う詰碁の盤面サイズは 9×9 とした．

4.1 局面生成ツリー

まず詰碁の候補となる局面をノードとするツリーを生成する．ルートノードには，実現したい手筋に関する局面を与え，黒石または白石を 1 つ増やした局面を子ノードとしていく．ノードの生成には，Bouzy's Algorithm[9] を用いる．

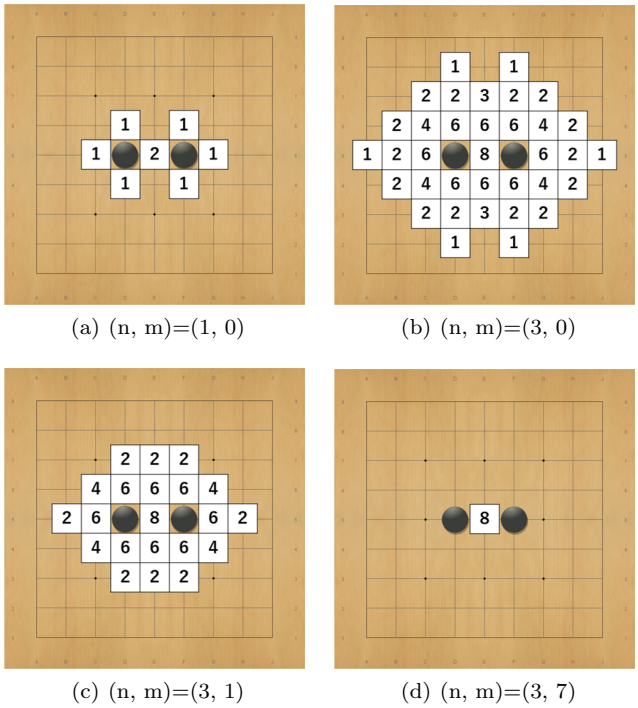
4.1.1 Bouzy's Algorithm

Bouzy's Algorithm とは，画像処理に用いられる膨張 (Dilation) と侵食 (Erosion) を囲碁に適用したアルゴリズムである．アルゴリズムの流れは以下の通りである．

1. 黒石のある交点には正のスコア，白石のある交点には負のスコアを与える．
2. 空点には，0 のスコアを与える．
3. すべての空点に対して，以下の Dilation を n 回繰り返す．
   - 空点のスコアが 0 以上かつ負のスコアを持つ交点が周りにない
     - 現在のスコアに周りにある正の交点の個数を足す
   - 空点のスコアが 0 以下かつ正のスコアを持つ交点が周りにない
     - 現在のスコアから周りにある負の交点の個数を引く
4. すべての空点に対して，以下の Erosion を m 回繰り返す．
   - 空点のスコアが正
     - 現在のスコアから周りにある 0 以下の交点の個数を引く
   - 空点のスコアが負
     - 現在のスコアに周りにある 0 以上の交点の個数を足す
   - それぞれスコアが 0 になったら Erosion の操作を止める

フリーソフトウェアの GNU Go のバージョン 2.0[10] では，局面の形勢判断にこのアルゴリズムを用いている．Dilation と Erosion の回数 (n, m) は，目算で (5, 21)，模様で (5, 10)，勢力図で (4, 0) であった．また，詰碁を解く探索プログラムの評価関数として用いられた例もある [11]．

次に，Dilation と Erosion の回数 (n, m)=(3, 7) として計算した流れを図 4.1 に示す．



(a) (n, m)=(1, 0)　(b) (n, m)=(3, 0)

(c) (n, m)=(3, 1)　(d) (n, m)=(3, 7)

図 4.1 Bouzy's Algorithm の計算例

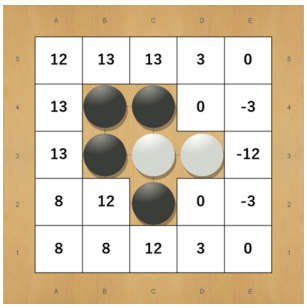
4.1.2 ツリーの作成手法

親ノード (現在の局面) から子ノード (次の局面) を生成する手順を次に示す．また，盤面サイズ 5×5 での計算例を図 4.2 に示す．

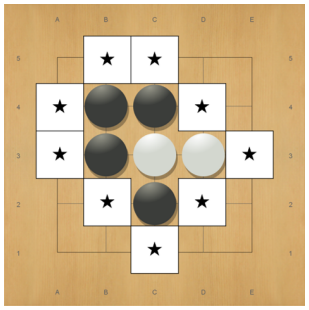
1. 親ノードを Bouzy's Algorithm で計算し，各空点にスコアをつける．(図 4.2(a))
2. 親ノードに対応する局面に置いてある石の 4 近傍の空点すべてに対して以下の手順を行う．(図 4.2(b))
   - 空点のスコアが負
     - その空点に黒石を置いた局面を仮のノードとする
   - 空点のスコアが 0
     - その空点に黒石を置いた局面，白石を置いた局面をそれぞれ仮のノードとする
   - 空点のスコアが正
     - その空点に白石を置いた局面を仮のノードとする
3. すべての仮ノードに対して以下の手順を行う．
   (a) 仮ノードを Bouzy's Algorithm で計算し，各空点にスコアをつける．(図 4.2(c))
   (b) 1. と 3(a). の計算結果の差分を計算し，すべての値を合計する．この合計した値をスコア差分と呼ぶ．(図 4.2(d))
4. 3. で計算したすべての仮ノードに対するスコア差分 (図 4.2(e), 図 4.2(f)) について，閾値を超えている仮

ノードを子ノードとする．閾値を超えていない仮ノードは子ノードとしない．

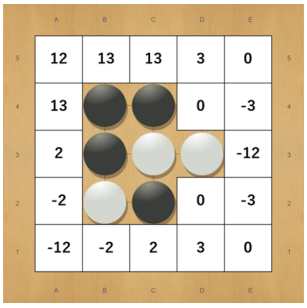
5. 閾値を超えている仮ノードがない，もしくは事前に定めたノード数を生成するまで，1.〜4. の手順を繰り返し，ツリーを作成する．



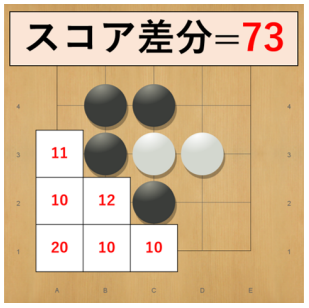
(a) 1. 親ノードのスコア



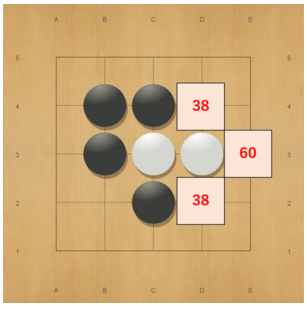
(b) 2. 石の 4 近傍の空点



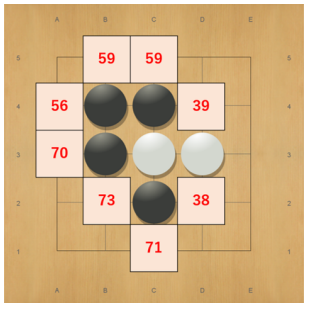
(c) 3. 仮ノードのスコア



(d) 4. スコア差分



(e) 5. 黒石を置いた仮ノードのスコア差分



(f) 5. 白石を置いた仮ノードのスコア差分

図 4.2 ツリーの作成手順

スコア差分は，石を増やす前の局面と石を増やした後の局面で，どれだけ値の差が生じるかを計算したものである．この値は，置いた石が局面上にどれだけの効果を与えるかを数値化したものと考え，事前に定めた閾値を超えた場合のみノードを生成する．

4.2 判別ネットワーク

このネットワークは，与えた石の配置がどれだけ詰碁らしいのかを判別するネットワークである．既存の死活判定システムや詰碁ソルバとして，インターネットを使った囲碁対局サービスを行っているパンダネットが提供している「死活の神様 パンダ先生」[12] などがあるが，これらは詰碁問題として成立するか否かを離散的に判定するものである．本研究では，その局面が詰碁として成立するか否かではなく，その局面がどれだけ詰碁らしいのかを連続的な数値として判定するネットワークの作成を試みた．

4.2.1 ネットワークの構成

ネットワークの構造を表 4.1 に示す．

表 4.1 判別ネットワークの構造

Input	9×9×25 [盤面サイズ × チャネル数]
Conv1	3×3 の 128 種類のフィルタと ReLU 関数
Conv2	3×3 の 96 種類のフィルタと ReLU 関数
Conv3	3×3 の 64 種類のフィルタと ReLU 関数
Conv4	1×1 の 1 種類のフィルタ
Dense	出力 64 個の全結合ネットワークと ReLU 関数
Output	出力 1 個の全結合ネットワークと Sigmoid 関数

入力層には，盤面サイズ 9 × 9 の各点ごとに 0-1 の 2 値で表現した特徴量を入力した．使用した特徴量は，石の位置情報と盤面情報から計算した囲碁的特徴量 [13] である．石の位置情報とは，黒石の位置，白石の位置，空点の位置の 3 チャネルである．また，囲碁的特徴量は，

- 石のタテヨコの空点の数を表す「呼吸点」
- 石の連なりの数を表す「連」
- 取れる黒石の数
- 取れる白石の数

の 4 項目について計算したものである．各項目について計算した値を 1〜4 および 5 以上のそれぞれの場合を 1 チャネルで表し，計 20 チャネルで表現した．これらにすべて 1，すべて 0 で埋めた 2 チャネルを合わせた合計 25 チャネルを入力する特徴量として用いた．

中間層の畳み込みフィルタサイズは第 1 層〜第 3 層では 3 × 3，第 4 層では 1 × 1 とした．また，出力層には全結合ネットワークを採用して出力 1 ノードに変換している．

出力層では，詰碁であるかどうかを評価する 0〜1 の値を出力する．1 に近いほど詰碁であり，0 に近いほど詰碁でないとした．

4.2.2 入力する特徴量の計算例

図 4.3 のような盤面サイズ 3×3 の局面を与えた場合，特徴量を計算すると図 4.4 のようになる．図 4.4(a)〜図 4.4(c) が石の位置情報，図 4.4(d)〜図 4.4(g) が囲碁的特徴量である．囲碁的特徴量 (*) はすべて図 4.5 のようにチャネルを分割して入力する．

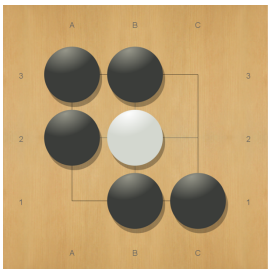
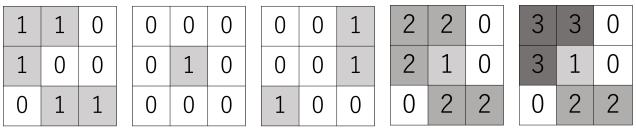
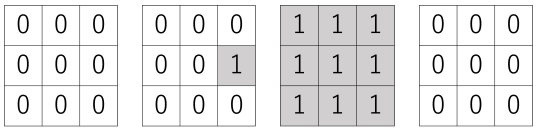


図 4.3 入力する局面の例

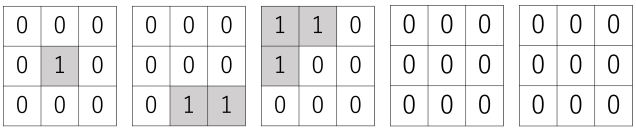


(a) 黒石 (b) 白石 (c) 空白 (d) 呼吸点* (e) 連*



(f) 取れる黒* (g) 取れる白* (h) すべて 1 (i) すべて 0

図 4.4 石の位置情報と囲碁的特徴量 (*)



(a) 値=1 (b) 値=2 (c) 値=3 (d) 値=4 (e) 値=5〜

図 4.5 図 4.4(e) の入力値

4.2.3 学習データセット

データセットには，正の教師データとして実在する詰碁の局面，負の教師データとして詰碁でない局面を用いた．詰碁でない局面 (負の教師データ) は次の 2 つの方法で生成した．

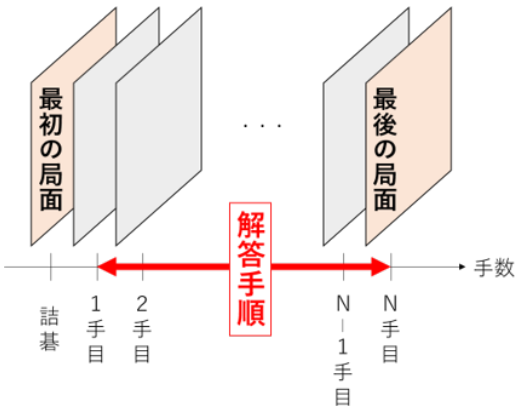


図 4.6 詰碁の解答手順について

1. ランダムで作成
   黒石と白石それぞれを 3〜15 個の範囲で生成し，ランダムに配置したものを詰碁でない局面として作成する．
2. 解答手順の最後の局面を利用
   詰碁の問題には解答手順が存在する．詰碁の解答手順について図 4.6 に示す．各問題における解答手順の最後の局面を詰碁でないと定義して利用する．

また，囲碁の局面は回転・反転の対称性が成り立つため，このことを利用すると 1 局面を回転 (0 度, 90 度, 180 度, 270 度)× 反転 (あり, なし) の 8 倍のデータに拡張することができる．元のデータ数と拡張したデータ数を表 4.2 に示す．

表 4.2 データセット数

		データ数	拡張データ数
正のデータ (1)	詰碁	191	1528
負のデータ (0)	最後の局面	191	1528
	ランダム	(都度作成)	

4.2.4 学習パラメータ

実験に用いたデータ数やパラメータを表 4.3 に示す．

表 4.3 学習データ数とパラメータ

学習データ	2956
テストデータ	100
epoch	30
batch size	8
optimizer	Adam
Loss	binary crossentropy

4.2.5 学習結果

負のデータセットによる判別結果の違いを表 4.4 に示す．

表 4.4 学習結果

負のデータセット	テストデータ正解率
ランダム	100%
最後の局面	67%
ランダム + 最後の局面	50%

ランダムで作成した局面との判別結果は 100%となったが，最後の局面との判別正解率は 67%であった．ランダムで作成した局面と最後の局面をどちらも負のデータセットとして学習すると．正解率は 50%となり，学習がうまくいかなかった．

4.3 詰碁生成の実験

[4.1] 局面生成ツリーを用いて，局面の自動生成を行い，[4.2] 判別ネットワークを用いて，生成した局面が詰碁であるかどうかの評価を行った．ネットワークは，負のデータセットとしてランダムを用いたネットワークと最後の局面を用いた 2 つのネットワークを使い，それらの平均値を評価値とした．また，生成する詰碁は「黒先白死」の問題とした．

4.3.1 パラメータ設定

実験に用いた各パラメータを表 4.5 に示す．

表 4.5 実験パラメータ

Bouzy's Algorithm	(n, m) = (5, 0)
生成するノード数	200
スコア差分の閾値	各ノードにおける最大スコア差分の 8 割

事前実験より，Dilation と Erosion の回数 (n, m) は (5, 0)，スコア差分の閾値は各ノードにおける最大スコア差分の 8 割とした．Erosion の値を大きくすると，各空点のスコアが小さくなったり 0 となってしまうことから，Erosion の値は 0 とした．

4.3.2 初期局面

ルートノードに与えた初期局面を図 4.7 に示す．この初期局面はウッテガエシの手筋に関する局面である．

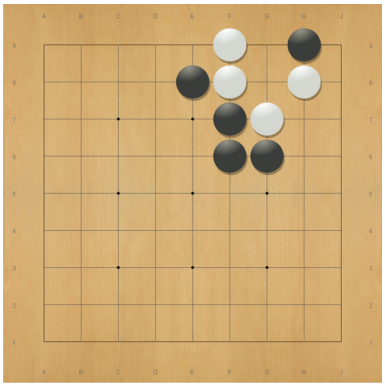
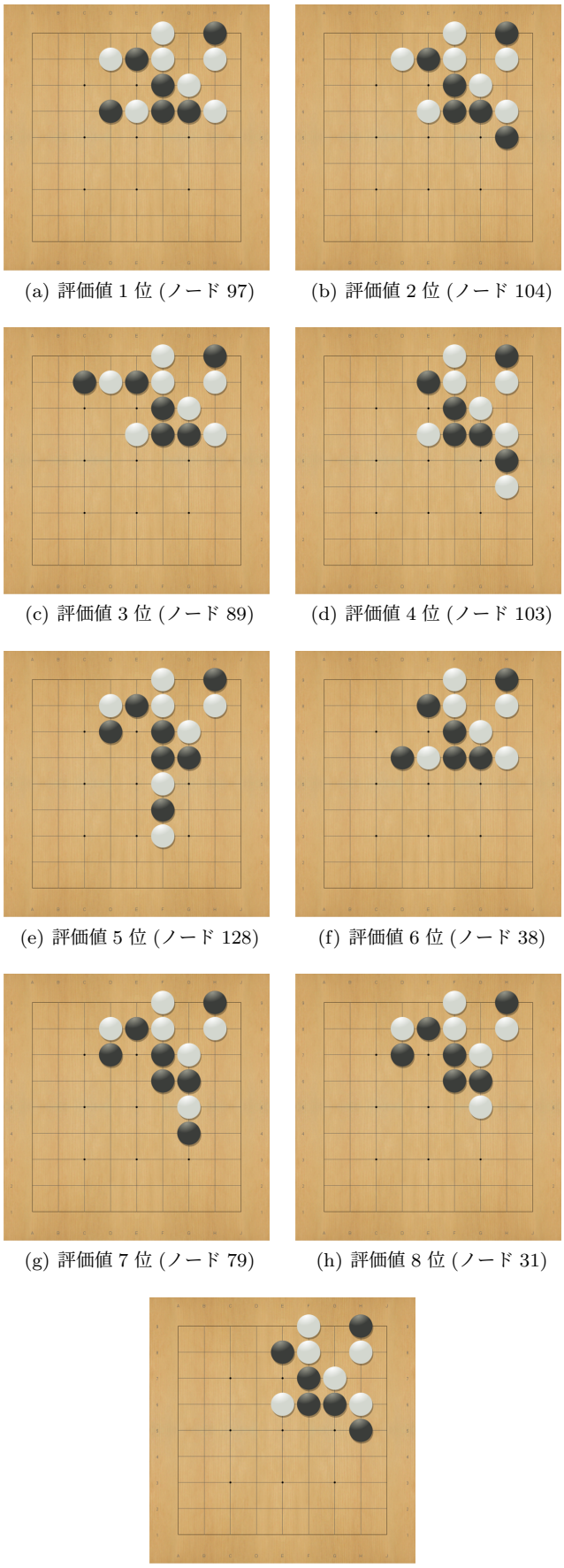


図 4.7 初期局面

4.3.3 実験結果

生成した 200 局面のうち，平均評価値の上位を表 4.6 に示し，その各生成局面を図 4.8 に示す．



(a) 評価値 1 位 (ノード 97)　(b) 評価値 2 位 (ノード 104)

(c) 評価値 3 位 (ノード 89)　(d) 評価値 4 位 (ノード 103)

(e) 評価値 5 位 (ノード 128)　(f) 評価値 6 位 (ノード 38)

(g) 評価値 7 位 (ノード 79)　(h) 評価値 8 位 (ノード 31)

(i) 評価値 9 位 (ノード 40)

図 4.8 生成局面

表 4.6 生成局面の評価値

順位	ノード番号	「ランダム」	「最後の局面」	平均評価値
1	97	0.99998	0.99996	0.999976
2	104	0.99998	0.99995	0.999972
3	89	0.99993	0.99990	0.99991
4	103	0.99992	0.99989	0.99990
5	128	0.99995	0.99981	0.999886
6	38	0.99993	0.99982	0.999880
7	79	0.99999	0.99973	0.999865
8	31	0.99999	0.99973	0.999863
9	40	0.99993	0.99978	0.999860
	初期局面	0.82657	0.99703	0.91180

評価値が上位であった局面のうちに，厳密な詰碁といえる局面はなかった．図 4.8(i) 評価値 9 位の局面は，実現したい手筋が解答となる局面となったが，初手で別解が存在するため，厳密な詰碁ではない．また，図 4.8(e) 評価値 5 位，図 4.8(g) 評価値 7 位の局面は白石を取ることができるものの，初期局面とほぼ同様の形であるため，詰碁ではない．そのほかの局面は白石を取ることができないため，詰碁ではない．

5. おわりに

構想法を用いた詰碁生成を提案した．詰碁の候補となる局面の自動生成を行うツリーでは，局面特徴を算出する Bouzy's Algorithm を用いてノード生成を行った．また，石の配置から詰碁であるかどうかを判別する CNN では，局面の位置情報から事前に計算した囲碁的特徴量を入力に使用した．これらを用いて詰碁生成を行ったが，厳密な詰碁を生成することはできなかった．ただ，構想法を実現した局面の生成はすることはできたので，今後はその実現したい手筋のみが解答となるような厳密な詰碁の生成を目指す．

参考文献

[1] David Silver, *et al.*, "Mastering the game of Go with deep neural networks and tree search", Nature 529, pp.484-489, (2016).

[2] David Silver, *et al.*, "Mastering the game of Go without human knowledge", Nature 550, pp.354-359, (2017).

[3] 上西知陽, 中村克彦. 逆算法による詰碁問題の生成方式. 情報処理学会第 75 回全国大会講演論文集 (1), pp.97-98, (2013).

[4] 岡田壱星, 中村貞吾. 詰碁解析プログラムを用いた詰碁問題生成. 2019 年度電気・情報関係学会九州支部連合大会, p.579, (2019).

[5] 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. 人工知能学会誌, Vol.13, No.3, pp.452-460, (1998).

[6] 宗藤大貴, 長尾智晴. 進化計算法を用いた詰将棋の自動生成. ゲームプログラミングワークショップ 2019 論文集, pp.1-6, (2019).

[7] 佐々木宜介, 今田鈴音, 長峯練. 駒の役割を考慮する詰将棋の生成及び改良手法の研究. 研究報告ゲーム情報学, 2020-GI-43(12), pp.1-8, (2020).

[8] the tsume-go social gathering, "https://www.h-eba.com/tsumego/index.html", 2021 年 10 月閲覧.

[9] Brono Bouzy, "Mathematical morphology applied to computer go", IJPRAI vol 17 no.2, (2003).

[10] GNU Go - GNU Project - Free Software Foundation (FSF), "https://www.gnu.org/software/gnugo/", 2021 年 10 月閲覧.

[11] 石井宏和, 横山大作, 近山隆. Bouzy's 5/21 Algorithm を用いた Df-pn+探索の詰碁への適用. 情報処理学会第 15 回ゲーム情報学研究会, (2006).

[12] 実践詰碁解答ソフト 死活の神様パンダ先生 | 囲碁ゲームのパンダネット, "https://www.pandanet.co.jp/members/lifedeath/", 2021 年 10 月閲覧.

[13] 美添一樹・山下宏・松原仁編 (2012)『コンピュータ囲碁-モンテカルロ法の理論と実践』共立出版.