社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－GI－11 (2)
2004／3／8

# 効率的な詰将棋探索のための評価関数

金子 知適 † 田中 哲朗 ††
山口 和紀 †† 川合 慧 †

本稿では，将棋の実戦であらわれる局面を対象に詰みやすさを予測する評価関数について提案し，df-pn$^+$ 探索と組み合わせることで探索を効率的にすることを試みた．df-pn$^+$ 探索とは詰将棋で有力な手法とされている df-pn 探索に，局面の証明数と反証数を予測する評価関数を組み合わせたものである．探索を制御する証明 (反証) 数の初期値の予測方法として詰 (不詰) の局面の証明木の大きさに着目し，(1) 証明木の大きさの予測値を証明 (反証) 数とする方法と，(2) 証明木の大きさの逆数をノードを展開する価値として用いる方法の二つの手法を提案した．評価関数のパラメータは，それぞれの手法について，実戦の棋譜に表れた詰 (不詰) の局面の証明木の大きさを用いて自動的に調整した．実戦で表れた局面を対象に実験を行ったところ，作成した評価関数を用いた df-pn$^+$ 探索が，評価関数を用いない df-pn 探索に比べて効果的に詰や不詰を発見できることを確認した．

# Evaluation Functions for Checkmate Search in Shogi Programs

T. KANEKO ,† T. TANAKA ,†† K. YAMAGUCHI †† and S. KAWAI †

Evaluation functions for checkmate search in Shogi-game are proposed, that are suitable for df-pn$^+$ search. They estimate initial values as proof number and disproof number in a newly expanded state in the search. We proposed two kind of evaluation functions: (1) the one based on the estimated size of (dis)proof trees, and (2) the other based on the "expansion value" which is the inverse of the estimated size of (dis)proof trees. We trained proposed evaluation functions by using the sizes of proof trees and disproof trees, found by using normal df-pn search, in states appeared in real game records. Our experiments showed that df-pn$^+$ search combined with the proposed evaluation functions are actually more efficient than df-pn search without any evaluation function for proving or disproving states appeared in real game records.

## 1. はじめに

長編詰将棋を解けるようになるなど詰将棋解答プログラムは大きく進歩してきたが，実戦での利用を考えるとさらなる効率化が必要とされている．強い将棋プログラムを作るためには，何手も進んだ局面での長手数の詰を正確に認識する必要があり，そのためには探索中に何度も詰将棋を呼びだす必要がある．しかし，現状の詰将棋解答プログラムの効率は充分ではないため，探索の末端付近ではごく短い詰しか読むことができていない．

本稿では，詰みやすさを評価する詰将棋専用の評価関数を作成することで，詰将棋の解答を効率化することを試みた．現実の詰将棋プログラムは，詰や不詰の証明に必要な局面の 10 から 100 倍の局面を読んでいる．もし詰みやすさを評価する良い評価関数があれば証明に不要な局面の探索を省くことができ，大幅に探索局面数 (探索コスト) を減らすことができるというアイデアである．

詰みやすさを評価する評価関数として，詰や不詰の証明木の大きさの予測に基づく 2 種類の関数を提案し，パラメータを自動的に調整した．詰将棋を解くすぐれた手法である df-pn 探索[12] に，作成した詰将棋用の評価関数を組み合わせたところ，棋譜から収集した詰む局面や詰まない局面に対して詰や不詰を証明するまでの探索ノード数を減らすことができた．

次の節で関連研究について紹介した後，3 節で証明数と反証数について説明する．続いて，4 節でそれらを予測する評価関数について提案し，5 節で実験結果を報告した後に，6 節で結論を述べる．

† 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
kaneko@graco.c.u-tokyo.ac.jp
†† 東京大学情報基盤センター
Information Technology Center, The University of Tokyo

## 2. 関連研究

近年発展している df-pn 探索は，pn 探索[1] と同じ振舞いを保ちつつ深さ優先に改良したアルゴリズムで，300 手以上の詰将棋を全て解くという成果をあげている[12] 他に，詰碁でも効果が確認されている[2]．また，将棋を含めてサイクルがあるゲームでは GHI 問題が生じる可能性があったが，df-pn 探索における対策[3] も提案されている．

さらに，評価関数を df-pn 探索に組み合わせたアルゴリズムとして df-pn$^+$ 探索が提案されており，オセロの終盤の探索で良い評価関数を用いると探索ノード数を $\frac{1}{6}$ にできるという効果が確認されている[4]．しかし，評価関数を設計することの難しさから一般にはそれほど広まっていない．☆他のアルゴリズムと組み合わせた詰将棋の評価関数としては，様々なアイデアが提案されているが[6,8]，df-pn$^+$ 探索を用いたものや実戦の局面を対象に充分な量の実験を行なった研究は著者らの知る限りではない．

他に詰将棋を解くアルゴリズムとしては，PN*探索[5,9] が早く考案され，成果をあげてきた．こちらも有力な手法であるが，反復深化で反証数を閾値にするという点で df-pn 探索の方が効率が良いとされている[12] ことと，探索手法と評価関数との組み合わせ方が既に議論されている[4] ことを考慮して．本稿では df-pn 探索を採用した．

## 3. 証明数と反証数

まず，df-pn 探索の振舞いを制御する変数である証明数と反証数について説明する．証明数と反証数はノード (状態) 毎に定義される値で次のような意味を持つ:[12]

**証明数** あるノードが詰であることを証明するために，詰であることを証明する必要がある先端ノードの数の最小値．

**反証数** あるノードを不詰であることとを証明するために，不詰であることを証明する必要がある先端ノードの数の最小値．

この証明数と反証数は，次に展開するノードを決めるために用いられる．すなわち，ルートノードから攻方では証明数の小さいノードを，受方では反証数の小さいノードをたどった末端の局面が順次展開される．これは pn 探索のアルゴリズムであるが，df-pn 探索でも計算方法は異なるものの同じノードが展開される．

**表 1** 証明数・反証数の計算方法

| ノードの種類 | | 証明数 | 反証数 |
|---|---|---|---|
| 先端 | 詰 | 0 | $\infty$ |
| | 不詰 | $\infty$ | 0 |
| | 不明 | 1 | 1 |
| 内部 | 攻方 | min(子ノードの証明数) | $\sum$(子ノードの反証数) |
| | 受方 | $\sum$(子ノードの証明数) | min(子ノードの反証数) |

その際に，証明数と反証数を定義通りに計算することは困難であるため，木の探索を仮定して表 1 のように再帰的に計算する．合流のあるゲームでは，木であるという仮定が成りたたないためこの計算方法では効率が悪くなるという問題があるが一般的な解決法はまだない．さらに，詰将棋ではサイクルが存在するため，ノードの深さを保持して上流への合流はカウントを遅らせるなどの対策[2] が必要である．

評価関数を使う df-pn$^+$ 探索では，表 1 の計算方法を一部変更し，先端ノードの (証明数，反証数) を (1,1) とする代わりに，ノード毎に予測した (証明数の予測値，反証数の予測値) を用いる．☆☆この予測を行う関数を本稿では評価関数と呼ぶ．ここで予測が正しければ，df-pn$^+$ 探索は，証明に必要なノード以外を一切展開しないため効率が良くなる．そのような評価関数を作ることが本研究の目標である．

## 4. 評価関数の設計とパラメータの調整

### 4.1 評価関数の枠組

本節では，先ほどの証明数と反証数の予測を行う評価関数を説明する．局面に関して，どのような証明数と反証数を予測すると良いかは，厳密には難しい問題である．しかし，おおまかにはすぐに詰 (不詰) を見つけられそうな局面に小さな証明 (反証) 数を割り当てれば良い．そこで本稿では，証明 (反証) 木の大きさを詰や不詰の難しさの指標として用いる評価手法と，そのような評価関数のパラメータを調整する方法を提案する．

予測の方法としては，次の二通りを試みた:

(1) 証明木の大きさによる予測: 局面が詰，不詰だと仮定した場合の証明木の大きさを予想し，それぞれ証明数と反証数とする．

(2) ノードを展開する価値による予測: 詰みやすさに関する 1 次元の指標に変換し，そこから証明数と反証数のペアに変換する．

☆ 「東大将棋」では持駒の枚数を考慮して証明数を予測しているという (岸本氏との個人的な会話)．

☆☆ この他に，df-pn$^+$ 探索では cost という別の評価を導入しているが，調整が難しいため本稿では用いなかった．

以下，それぞれについて説明する．

### 4.2 証明木の大きさによる予測

この方法では，局面のあるべき証明数 (反証数) は，詰 (不詰) だとした場合の証明木の大きさであると仮定する．評価関数は，証明数用の評価関数と反証数用の評価関数の二つを別々に用意し，それぞれ証明数，反証数を独立に予測する．それぞれの関数のパラメータは，訓練例中の詰 (不詰) の局面において，評価関数を用いないアルゴリズムが見つけた証明木☆の大きさを用いて調整する．

このようにする場合は，訓練例を注意して用意する必要がある．例えば，評価要素に持駒を使用する場合を考える．実戦から詰の局面と不詰の局面を単純に集めるとすると，持駒が多いと少ない時には詰められない局面でも詰めることができて，結果として証明木が大きい訓練例が増える傾向がある．すると持駒が多いと詰みにくくなるという誤った学習をしかねない．これを防ぐためには，証明数の予測に不詰の局面も使用して，持駒が多い方が少ないよりも詰みやすいということを学ばせれば良い．次の節ではこのアイデアに基づいて，詰む局面と不詰の局面の両方を一つの関数で扱う方法を提案する．

### 4.3 展開する価値に基づく予測

この方法では，ノードを展開する価値という詰みやすさに関する 1 次元の指標を考える．ノードを展開する価値とは，その局面が詰みで証明木の大きさが $N$ 時に $\frac{1}{N}$，同様にその局面が不詰だとして証明木の大きさが $N$ の時に，$-\frac{1}{N}$ と定義する．このように定義すると，全ての局面は，$[-1, 1]$ の範囲の値を持ち，値が 1 の局面は指手がなく詰，$-1$ の局面は指手がなく不詰に対応する．逆に 0 近辺は詰にしろ不詰にしろ証明に手数がかかる局面となる．

全ての局面を訓練に使うためパラメータの調整は簡単になると予測できる一方で，指手の価値から (証明数，反証数) のペアを求める手法が必要となる．探索においては，攻方は価値が大きいと予測されるノードを，受方は小さいと予測されるノードを展開させたい．そこで，以下のような単純な手法で，価値が大きいノードに小さな証明数を，価値が小さなノードには小さな反証数を割り当てた．

$$\text{証明数} = \begin{cases} 1 & (\textit{if}\ \text{価値} \geq 0) \\ 64 \times (\text{価値} + 1) & (\textit{otherwise}) \end{cases}$$

$$\text{反証数} = \begin{cases} 1 & (\textit{if}\ \text{価値} \leq 0) \\ 64 \times (1 - \text{価値}) & (\textit{otherwise}) \end{cases}$$

この方法は，一つの値から証明数と反証数のペアに変換するという点で，長井によるオセロでの研究[4] と同じ考え方である．長井の研究では，オセロの終盤の評価関数 (黒石と白石の数の差を予測) から，(証明数，反証数) のペアに変換している．その際に変換する関数は，詰/不詰が判明する直前のルートノードでの (証明数，反証数) の閾値を，そのノードの (証明数，反証数) の望ましい予測としてパラメータを調整している．将棋においては，終盤の評価関数は現段階ではオセロに比べて貧弱であるため，今回提案したノードを展開する価値の方がより信頼できる値と考えられる．一方，(証明数，反証数) のペアへの変換は何らかの方法で自動的に調整する方が望ましいと思われるが，今回は訓練例を充分用意できなかったため見送った．

### 4.4 特 徴 量

評価関数は簡単のために線形結合として，以下の評価要素のものを試した:

(1) 持駒
(2) 持駒 (1 枚目を区別)
(3) 持駒 (1 枚目を区別) + 玉の周囲 8 マスの利き
(4) 持駒 (1 枚目を区別) + 玉の周囲 8 マスの利き + 玉の位置

評価要素が簡単なものだけで数が比較的少ないのは，訓練例の数を充分用意できなかったためである．

### 4.5 局面の準備

パラメータ調整に用いる局面は，実戦の棋譜から集めた．使用した棋譜は将棋倶楽部 24☆☆で指された 8,000 棋譜である[7]．詰む局面は，棋譜を再生させながら，詰めろのかかっている (手番の側がパスすると詰む) 局面を集めた．同様に，詰まないの教師例には，その手番で詰めようとして詰められなかった局面を集めた．詰/不詰を見つけるために用いたアルゴリズムは，df-pn アルゴリズムに，優越関係の利用[12]，サイクル対策[2]，GHI 対策[3] を組み込んだものである．☆☆☆また詰将棋特有の工夫として，中合の後回しや，飛角歩が成らない指手は打不詰以外は読まないなどをいれている．1 局面につき 20 万ノードを探索し，それでも詰/不詰を見つけられなかった局面は使用しなかった．

☆ もちろん最小とは限らないが，最小の証明木を見つけることは一般には困難である．

☆☆ http://www.shogidojo.com/

☆☆☆ プログラムは将棋ライブラリの一部として利用可能である[11]．

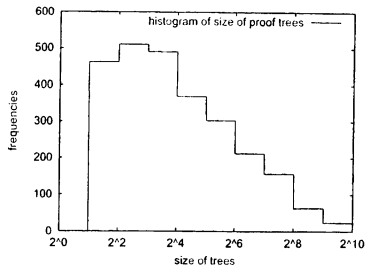


図 1　詰めろの証明木の大きさのヒストグラム

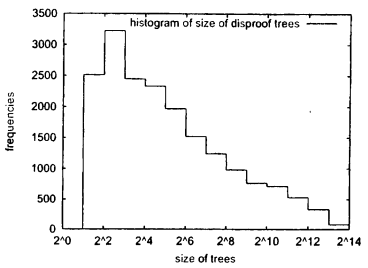


図 2　不詰の証明木の大きさのヒストグラム

また，初手から 40 手までの局面と王手をかけられなかった局面も使用しなかった．

詰めろ及び不詰の証明木の大きさの分布を図 **1** 及び図 **2** に掲載する．それぞれ，木の大きさを $2^n$ 刻みで頻度を集計し，頻度を縦軸に，大きさを横軸に描いたものである．このグラフから，大きさの小さな木が圧倒的に多いことが分かる．プロ棋士の棋譜ではまた別の傾向の可能性もあるが，証明木が大きな詰や不詰が実戦で表れる可能性は低いと予想される．一方で，パラメータの調整を安定させるためには，木の大きさがそろっている方が望ましいので将来の課題として何らかの対応が必要である．

## 5. 評価関数の効果

### 5.1 重みの調整結果

前節で説明した評価関数について，それぞれの線形結合中の重みを最小二乗法で推定した．実際の計算には，統計ソフトウェアの R☆を使用した．

#### 5.1.1 証明木の大きさによる予測

まず 4.2 節で説明した証明数と反証数の独立に予測する関数について結果を報告する．上記の棋譜には約 14,000 の詰の局面があり，そのまま全てを証明数を予測する評価関数のパラメータの調整に用いた．一方，不詰の局面は 12 万局面以上あったため，その $\frac{1}{5}$ の約

☆ http://www.R-project.org/

表 2　各評価関数の学習結果

| | | 証明 決定係数 $R^2$ | | 反証 決定係数 $R^2$ | |
|---|---|---|---|---|---|
| 種類 | 重みの種類 | $y$ | $\log(y)$ | $y$ | $\log(y)$ |
| (1) | 7 | 0.013 | 0.055 | 0.097 | 0.286 |
| (2) | 14 | 0.013 | 0.057 | 0.099 | 0.299 |
| (3) | 22 | 0.041 | 0.119 | 0.187 | 0.582 |
| (4) | 67 | 0.091 | 0.254 | 0.200 | 0.611 |

表 3　展開する価値を予測する関数の学習結果

| 種類 | 重みの種類 | 決定係数 $R^2$ |
|---|---|---|
| (1) | 7 | 0.098 |
| (2) | 14 | 0.113 |
| (3) | 22 | 0.351 |
| (4) | 67 | 0.422 |

25,000 局面を反証数を予測する評価関数のパラメータの調整に用いた．**表 2** にパラメータ調整の結果を掲載する．表中の評価関数の種類は，4.4 節で説明した，評価関数で使用する特徴量の種類に対応する．

証明木の大きさの分布が片寄っていることから，証明木の大きさの対数を予測する方が適している可能性を考えて，通常の予測 (表で $y$ と表記) と対数をとった後の値の予測 (表で $\log(y)$ と表記) の二種類の推定を行なった．決定計数 ($R^2$) で判断する限りでは，評価関数中の評価要素の項目が増えるほど予測は正確になっていることが分かる．また証明数と反証数では，証明数の方が予測が難しいことが読みとれる．

#### 5.1.2 展開する価値に基づく予測

続いて，4.3 節で説明した展開する価値の予測に基づく評価関数についての結果を報告する．詰の局面と不詰の局面の数を揃えた方が良いと思われたため，両方とも約 13,000 局面を用いた．**表 3** にパラメータ調整の結果を掲載する．

先程と同様に，決定計数 ($R^2$) で判断する限りでは，評価関数中の評価要素の項目が増えるほど予測は正確になっていることが分かる．

### 5.2 詰将棋探索での効果

評価関数の効果を確かめるために，詰めろのある局面で，見つけた証明木の大きさとそれを見つけるまでに探索したノード数の関係を測定した．訓練例とは別の棋譜の中で，詰めろのある 500 局面を例題とした．同様に不詰の局面で，見つけた証明木の大きさとそれを見つけるまでに探索したノード数の関係を測定した．こちらも同様に，詰めろのなかった 500 局面を例題とした．

まず，評価関数を使わない場合の分布を図 **3** 及び図 **4** に掲載する．探索ノード数は，局面表に登録され

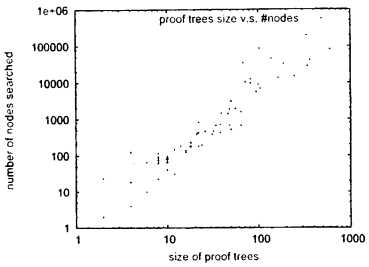


図 3　証明木の大きさと探索ノード数の散布図

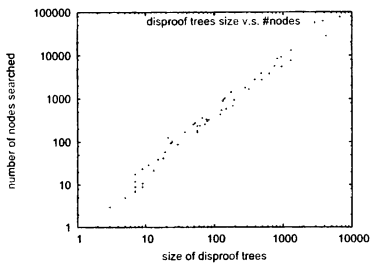


図 4　証明木の大きさと探索ノード数の散布図

た数で測った．☆ この結果から，評価関数を用いない通常の探索では，詰を証明するには木の大きさの 100 倍程度，不詰を証明するには 10 倍程度の局面の探索が必要になっていることが分かる．

### 5.2.1　証明木の大きさによる予測

続いて，証明木の大きさにより予測する評価関数を使用して，同じ局面を df-pn$^+$ 探索で探索した結果を**表 4** にまとめる．表で，「問題」は詰む局面か詰まない局面かどちらの例題を探索したか表し，また評価関数は，証明数と反証数の予測にどの評価関数を用いたかを示す．数字の後に log と書いてある場合は，パラメータ調整において対数を用いたことを示す．証明木と局面数は，それぞれ，各局面で見つけた証明木の大きさと探索したのノード数の合計で，比率はその合計後の比率である．また，比較のために評価関数を用いない場合の数値も掲載した．

表の結果から，詰む局面については，評価関数を用いると見つけた木の大きさは大きくなってしまっていることが分かる．また，探索局面数も (4) の評価関数を除いて増えてしまっている．(4) は実験した中では最も多くの評価要素を使っていることから，証明数の予測は難しく，持駒のみなどからパラメータを調整することは難しいことを示唆していると考えられる．この問題は，玉の周囲の 25 マスの駒を考慮するなど評価要素を増やすことで解決できるであろう．また木の大きさの対数をとってパラメータを調整することは，決定係数で判断する限りは効果的であったが，実際の探索では結果が悪かった．

一方，不詰の局面については，証明木の大きさは増えているものの，探索局面数は大きく減っており，半分以下のものも多い．従って，反証数の予想は簡単な特徴量でも効果的であると言える．

以上のように，詰の問題，不詰の問題とも (4) の評価関数の結果が良かったため，証明数の予測と反証数の予測の評価関数を同時に用いた場合の探索の効率を調べた．表の最後の 2 行がその結果である．それぞれ単独で用いるよりも若干性能は劣るが，評価関数なしのものと比較して，詰の問題，不詰の問題ともに性能が改善している．

### 5.2.2　展開する価値に基づく予測

最後に，ノードを展開する価値に基づいた評価関数を利用して同じ局面を df-pn$^+$ 探索で探索した結果を**表 5** にまとめる．評価関数がない探索と比較して，詰む局面に対しては，評価関数 (3), (4) が良い結果を残しており，先程同様に評価要素が多い方が効率が良くなる傾向を示している．また，不詰の局面に対しては全ての評価関数が効率を改善している．

一方，証明木の大きさにより予測する評価関数と比較すると，ノードを展開する価値に基づく予測を用いた方が，詰，不詰とも問題で小さな証明木を見つける傾向がある．一方，不詰の探索ノード数は劣る傾向に

表 4　証明木の大きさと探索局面数 (証明数と反証数の独立な予測)

| 問題 | 評価関数 | | 証明木 | 局面数 | 比率 |
|---|---|---|---|---|---|
| | 証明 | 反証 | | | |
| 証明 | なし | なし | 28726 | 4015138 | 0.007 |
| | (1) | なし | 33764 | 4260746 | 0.008 |
| | (2) | なし | 34352 | 4295551 | 0.008 |
| | (3) | なし | 56967 | 4414581 | 0.013 |
| | (4) | なし | 64265 | 2957751 | 0.022 |
| | (1) log | なし | 38329 | 6471862 | 0.006 |
| | (2) log | なし | 38191 | 6174945 | 0.006 |
| | (3) log | なし | 40356 | 5017261 | 0.008 |
| | (4) log | なし | 48346 | 4361708 | 0.011 |
| 反証 | なし | なし | 208037 | 1899894 | 0.117 |
| | なし | (1) | 298966 | 1299925 | 0.230 |
| | なし | (2) | 287488 | 1240033 | 0.232 |
| | なし | (3) | 274168 | 851212 | 0.322 |
| | なし | (4) | 289310 | 895990 | 0.323 |
| | なし | (1) log | 229189 | 975803 | 0.235 |
| | なし | (2) log | 242508 | 1051606 | 0.231 |
| | なし | (3) log | 324229 | 883421 | 0.367 |
| | なし | (4) log | 336765 | 888763 | 0.379 |
| 証明 | (4) | (4) | 78801 | 3568060 | 0.022 |
| 反証 | (4) | (4) | 302480 | 901660 | 0.335 |

☆ 訪れる局面はこの 2 倍程度になる．理由は df-pn アルゴリズムは多重反復深化を行うため，内部ノードの再展開が必要になるためである．

表 5 証明木の大きさと探索局面数の比率(展開する価値)

| | 評価関数 | 証明木 (合計) | 局面数 (合計) | 比率 |
|---|---|---|---|---|
| 証明 | (1) | 30156 | 4304044 | 0.007 |
| | (2) | 32111 | 6099030 | 0.005 |
| | (3) | 29895 | 3200853 | 0.009 |
| | (4) | 30540 | 3195096 | 0.010 |
| 反証 | (1) | 207262 | 1457535 | 0.142 |
| | (2) | 211205 | 1505405 | 0.140 |
| | (3) | 216056 | 1667634 | 0.130 |
| | (4) | 243019 | 1631208 | 0.149 |

ある．現時点ではどちらが良い評価関数かは結論づけがたいが，ノードを展開する価値に基づいた評価関数では価値から (証明数，反証数) への変換をアドホックに行っていることを考慮すると，変換をうまく調整すればノードを展開する価値に基づいた評価関数の方が成績が良くなると予測される．

全体として，成績は向上したものの，残念ながらオセロでの研究[4] ほどの効果を得ることはできなかった．主な原因は予測の正確さが足りないことと考えられるので，今後，評価項目を増やす[10] など正確さを改善することを試みたい．なお，原因の一部は，オセロでは終盤の正確な評価関数が利用可能である，勝負がつくまでの手数がほぼ一定である，黒の勝と白の勝は対称であるなど，オセロが詰将棋に比べて有利な特性を持つためであると考えられるので，詰将棋でオセロ以上の成果を出すことは容易ではなく価値のある研究課題と考えられる．

## 6. おわりに

本稿では詰将棋探索を効率的に行うための評価関数を考案し，df-pn$^+$ 探索と組み合わせた結果を報告した．探索を制御する証明 (反証) 数の初期値の予測方法として詰 (不詰) の局面の証明木の大きさに着目し，(1) 証明木の大きさの予測値を証明 (反証) 数とする方法と，(2) 証明木の大きさの逆数をノードを展開する価値として用いる方法の二つの手法を提案した．実際に棋譜に表れた詰めろ及び詰のない局面に対して探索を行い，詰及び不詰を証明する効率を調べたところ，評価関数を用いた探索では，用いない場合と比較して効率が向上することを示した．

今後の課題としては，さらなる効率の向上のために，より多くの評価要素と訓練例を用いた大規模な実験が必要である．特に，短い手数の詰と長い詰が同じ評価関数で良いかどうかは検討の余地がある．同様に，実践向けと長編詰将棋を解くのに向いている評価関数は異なると考えられる．

## 参考文献


1) L. V. Allis, M. van der Meulen, and H. J. van den Herik. Proof-number search. *Artificial Intelligence*, 66:91–124, 1994.
2) A. Kishimoto and M. Mueller. Df-pn in go: An application to the one-eye problem. In *Advances in Computer Games 10*, pp. 125–141. Kluwer Academic Publishers, 2003.
3) A. Kishimoto and M. Mueller. A solution to the ghi problem for depth-first proof-number search. In *7th Joint Conference on Information Sciences (JCIS2003)*, pp. 489–492, 2003.
4) A. Nagai and H. Imai. Application of df-pn$^+$ to othello endgames. In *Game Programming Workshop in Japan '99*, pp. 16–23, Oct. 1999.
5) M. Seo, H. Iida, and J. W. Uiterwijk. The pn*-search algorithm: Application to tsume-shogi. *Artificial Intelligence*, 129(1-2):253–277, 2001.
6) S. Tanaka, H. Iida, and Y. Kotani. An approach to tsume-shogi: Applying proof-number search with estimation function of mating. In *Game Programming Workshop in Japan '95*, pp. 138–147, Kanagawa, Japan, 1995. (In Japanese).
7) 久米. 将棋倶楽部 24 万局集. ナイタイ出版, 2002.
8) 野下. 詰将棋を解くプログラム T2. 松原 (編)，コンピュータ将棋の進歩, 第 3 章, pp. 50–70. 共立出版, 1996.
9) 脊尾. 共謀数を用いた詰将棋の解法. 松原 (編)，コンピュータ将棋の進歩 2, 第 1 章, pp. 1–21. 共立出版, 1998.
10) 金子, 田中, 山口, 川合. 駒の関係を利用した将棋の評価関数. 第 8 回ゲームプログラミング ワークショップ, Nov. 2003.
11) 田中, 副田, 金子. 高速将棋ライブラリ openshogilib の作成. 第 8 回ゲームプログラミング ワークショップ, Nov. 2003.
12) 長井, 今井. df-pn アルゴリズムと詰将棋を解くプログラムへの応用. 情報処理学会論文誌, 43(6):1769–1777, 2002.