ICOT Technical Memorandum: TM-0694

TM-0694

静的評価関数を使った 詰め碁プログラムの実現と評価

沖 廣明(未来技研)、実近憲昭(ETL)、
吉川貞行(NTT)、吉岡 勉、内田俊一

March, 1989

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191－5
Telex ICOT J32964

Institute for New Generation Computer Technology

静的評価関数を使った詰め碁プログラムの実現と評価

沖 廣明　　実近 憲昭*
(株) 未来技術研究所　　工業技術院 電子技術総合研究所
吉川 貞行†　　吉岡 勉　　内田 俊一
(財) 新世代コンピュータ技術開発機構

概要

詰め碁の問題をしらみつぶしに探索して解くプログラムの改良を目的として静的評価関数の導入を試みた. 静的評価関数の定式化に当っては,「中手」という石の死活に関する囲碁の知識を利用した. この結果, 静的評価関数の導入によって, 最終局面まで読み切る旧方式の探索より効率のよい探索が実現できた.

本報告では静的評価関数の導入方法について述べるとともに, 探索の効率がどれくらい向上したかについて報告する.

1　はじめに

ICOT では, 囲碁の対局を行うシステム「棋士システム」を現在開発中である[1]. 囲碁はその状態空間が巨大であること, また局面を評価する簡明な方法が知られていないことから, チェスなどで成功したしらみつぶしに探索するような単純な方式のみでは, 棋力の向上は望めない[2]. この為, 棋士システムでは知識指向型の立場をとって研究を進めており, 探索は知識として記述するのが困難な場合について, 知識を補完するものとして用いている.

棋士システムの探索には, 捕獲, 切断, 詰め碁, 攻め合いの 4 種類があるが, 現在, 実際に対局を行うシステムに組み込まれているのは捕獲と切断のみである. 詰め碁や攻め合いは, 探索に時間が掛かり過ぎるため組み込むことを見合わせており, このことがシステム全体の棋力向上の大きな障害になっている.

棋士システムの局所探索は, これまでどれも, 最終局面まで全てしらみつぶしに探索する, いわゆる読み切り型の方式をとってきた. 捕獲や切断の問題のように探索空間が比較的小さい場合はそれで充分であったが, 詰め碁や攻め合いの問題ではより効率の良い探索が必要になった.

ゲーム木探索の効率を上げる為の方法として, 静的評価関数によってある深さ以降の探索を省略する方法がよく用いられる. 静的評価関数とは, 与えられた局面に対して, 探索以外の何らかの発見的な方法でその局面の評価をするものである. しかし, 駒得のような局面を評価する簡明な因子を持つチェスなどのゲームについては, 静的評価関数の有効性が実証されているが, 囲碁のようなゲームでは, 有効な静的評価関数が設定できるかどうかさえよくわかっていない[3]. 今回実験的に,「中手」という死活に関する囲碁の知識を用いた静的評価関数を考案し実際に詰め碁プログラム中に組み込み, その有効性について調べた.

---

*現 (株)AI 言語研究所
†現 NTT データ通信株式会社

また, この詰め碁プログラムは, 論理型プログラミング言語 ESP[4] で記述され, 逐次型推論マシン PSI-II とそのオペレーティング・システム SIMPOS(4.2 版) という環境の下で, 実行している.

次章以降で, まず旧方式の読み切り型の詰め碁探索について説明し, 次に, 導入した静的評価関数について述べ, 最後に試作したプログラムによる実際の測定結果を示す.

2 読み切り型詰め碁探索

2.1 問題の定式化と探索法

詰め碁とは, 石の配置と手番とが与えられ, そこから黒白双方が最善の着手を続けた時の結末を答える問題である. 詰め碁のようなゲームの問題は, ゲーム木を定式化できれば, 「ゲーム木の探索法」によって解が求められることが分かっている[5]. ゲーム木を定式化する為には,

1. 与えられた局面に対する可能な手

2. 与えられた局面が最終状態の場合にはその局面に対する評価値

を決める必要がある[6].

通常の詰め碁の場合, 例えば図1のように, 外側に囲んでいる石があるので, その内側の全ての点を,1の「与えられた局面に対する可能な手」と決めた. ただし, 守り番の時はパスすることも可能とした.

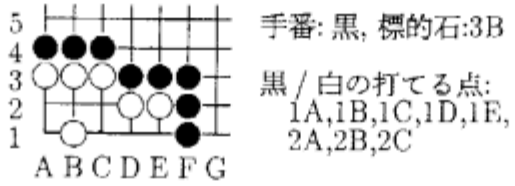

図 1: 詰め碁の例

次に,2の「局面の最終状態」は, 詰め碁では「生き」と「死に」の 2 種類がある. 「死に」と判定するのに, 囲まれている石全てが死ぬまで読む必要はほとんどない. 例えば, 図1からの変化図である図2の左の図の場合, 左上の 3 石が捕られてしまっては最早生きるべきスペースがない. このようにそれが殺されることが全体の石の死を意味するような石の死をもって「死に」の状態とした. 次に, 例えば黒番で図2の右図の場合, 打着禁止の場所以外もう打つ場所がない. このような状態を「生き」の状態とした.

また, ゲーム木の探索法は,$\alpha\beta$枝刈り法を採用した. 最終状態以外にその局面を評価することのない, いわゆる読み切り型の探索の場合,$\alpha\beta$枝刈り法が最も適しているからである.

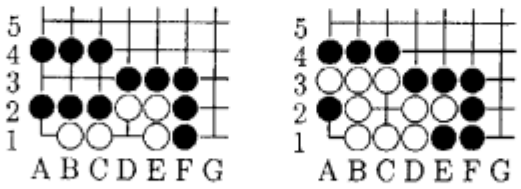

図 2: 「死に」の状態と「生き」の状態

2.2 候補手の生成順に関するヒューリスティック知識

$\alpha\beta$枝刈り法による探索の場合, 有望な手から打つほど, 探索の効率は格段によくなる. この為いくつかの囲碁の一般的な知識を, 候補手の生成順に関するヒューリスティック知識として導入した. 導入した囲碁の知識は以下の通りである.

1. 位置による急所
   「隅の急所は2ノ一,2ノ二」という格言もあるように, 「2ノ一」や「2ノ二」の手は隅の死活問題では急所になる場合が多い.

2. 眼形の急所
   眼を作るのに効率のよい手は, 一般的に死活問題の急所となる場合が多い.

3. フトコロを狭める／拡げる点
   「死はハネにあり」という格言もあるように, ハネてフトコロを狭めたり, またサガってそれを防ぐような手はまず考えてみなければならない.

4. 外ダメを詰める手
   外側からダメを詰めるような手は急がない手の場合が多く, 最後に考えればよい.

以上の4つの要素からなる多項式によって候補手の評価値関数を作り, 評価値の高い手の順に探索していくようにした. 候補手評価値関数内で重みとして使われる各係数は, 多くの詰め碁の問題に実際に適用しながら決めていった.

2.3 読み切り型詰め碁探索の特徴

読み切り型の探索を実際に試作し, いくつかの問題を解かせてみたところ, 例えば図3のような4つの問題 (全て黒先) については全て数秒で解いた. 問題は全て, 市販の詰め碁の問題集に載っていたものである[7].

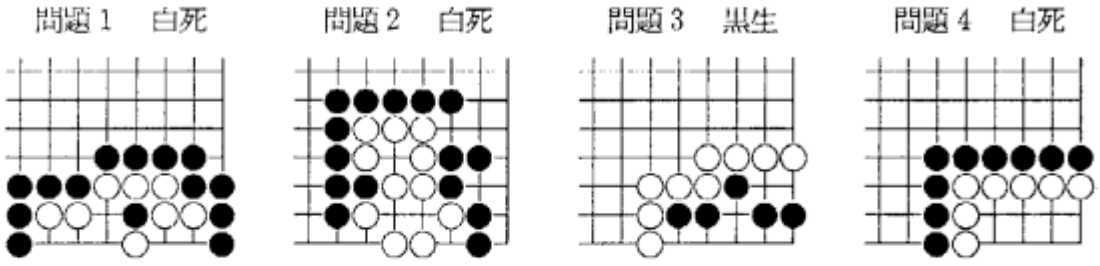

図 3: 詰め碁的な問題例

問題1は自らアタリに突っ込む手によって殺せる. 問題2は一見関係のなさそうな手が最後にダメヅマリを利用して眼をとることに利いている. 問題3はわざわざ石を大きくして取らせてから生きる. 問題4はセキを利用して生きる. というように, これらの問題は全て見落としがちな筋を含んでおり, いかにも詰め碁的 (頭の体操的) な問題である. このような問題については, 読み切り型の探索も満更でもないと言ってよいであろう.

これに対して, 例えば図4のように (全て黒先), 比較的実戦でよく生じるような問題も解かせてみたところ[8], 全て数分程度の時間を要した. 詰め碁的な問題と実戦的な問題について探索に要した手数と時間の比較を図5に示す.

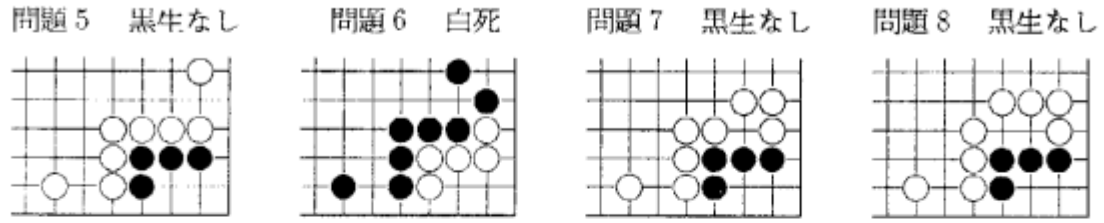

図 4: 実戦的な問題例

探索に要した手数と時間					
詰め碁的な問題例			実戦的な問題例		
問題 1	333 手	0.8 秒	問題 5	19,623 手	41 秒
問題 2	1,858 手	4.9 秒	問題 6	24,249 手	57 秒
問題 3	4,471 手	8.9 秒	問題 7	47,752 手	109 秒
問題 4	4,263 手	11 秒	問題 8	155,451 手	410 秒
平均	2,731 手	6.4 秒	平均	61,769 手	154 秒

図 5: 詰め碁的な問題例と実戦的な問題例の探索結果

実戦的な問題は詰め碁的な問題に比べて, 明らかに時間が掛っている. 一方, 人間が感じる難易度では, どちらの問題もそれほど差がないと思われる. ただし, 詰め碁的な問題の方が打てる点の範囲が少し狭い. 読み切り型の探索に掛かる手数は, 問題の難しさに関係なく, ほとんど打てる範囲の広さのみに比例するから, 少し広くなっただけで時間が掛かるのは当然とも言える. 従って, 問題 5,7,8 のように, 問題の本質には関わりのない外側のダメが増えるだけで, 掛かる時間が何倍もになることになる. 我々の詰め碁プログラムはあくまで実戦の対局の中で使われることを目的としているので, このような特徴は好ましくない.

3　静的評価関数

3.1　人間はどうしているか?

人間が詰め碁を解く場合は, 石が打ち上げられる最終の状態まで読んではいない. それよりずっと以前の局面で生死を判断している. 図6は問題 5 の解答の中で解説として現れた変化図である. このような変化図を示すことで生き死にの帰結に対する説明を終えている. つまり, 人間はこのような変化図から先はもう読む必要はなく, 知識によってその結論が推定できるのである. 図6の場合全て中手と呼ばれる囲碁の基本的な知識が使われている.

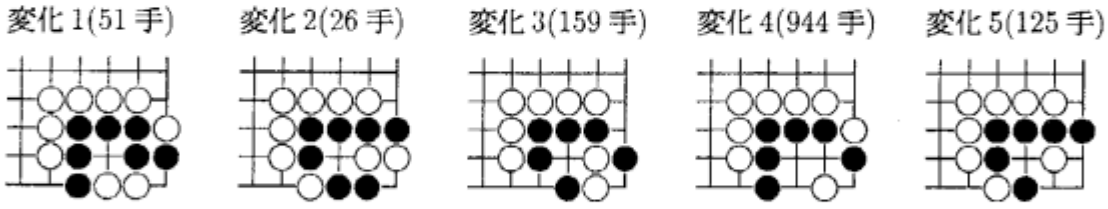

図 6: 問題 5 の変化図

また, 図6のカッコ内の手数は, 変化図からの読み切り型の探索に要する手数を示している. 人間がこれで終わりと判断する局面以降, 読み切り型の探索はまだ平均で 261 手の探索を行うので

ある．このような図をもって終了と判定できる中手の知識を導入できれば，それは探索手数を大幅に減らすことが期待できる．

3.2 静的評価関数の定式化

中手の知識は非常に多くのパタン的な知識により構成されている．パタンマッチングは一般にコストが掛かるが，静的評価関数にあまり時間が掛かり過ぎて，探索してしまった方が早かった，というのでは意味がなくなる．この為，効率のよいパタンマッチングを行うように，新たに次数コードという概念を導入した．この次数コードによって，パタンの取り扱いが高速かつ容易になった．以下，今回試作した静的評価関数について順次説明していこう．

1. 眼形
   同色の石と辺で囲まれた或る広さ以下の領域を眼形と呼ぶ．

2. 眼形の次数コードと急所
   眼形の次数コードとは眼形内の各点の次数の組み合わせのこと．次数とは点の隣接点の内同じ眼形に含まれる点の数のこと．また，急所とは石の死活にとって重要な点のことで，次数コードによって一意的に決まる．いくつかの例を図7に示す（・は空点か敵石の意）．

サイズ	眼形パタン	次数コード	急所
3	・・・	(2,1,1)	次数2の点
4	・・ ・・	(2,2,2,2)	なし
	・・・ ・	(3,1,1,1)	次数3の点
	・・・・	(2,2,1,1,)	次数2の点
5	・・・ ・・	(3,2,2,2,1)	次数3の点
	・・・・ ・	(3,2,1,1,1)	次数3と2の点
	・・・・・	(2,2,2,1,1)	次数2の点

図 7: 眼形パタンとその次数コード及び急所の例

3. 眼形の眼数
   眼形の眼数は，急所の点の数とその中で敵石に占められている点の数によって，図8のように決められる．

4. 仮想眼形の眼数
   図9の左図のような配置では，囲まれた領域はない．ただし，A の点に黒石を仮想的に置けば，その眼形から眼数を決定することは可能である．このような仮想局面における眼数を仮想眼形の眼数とする．

5. 欠け目
   欠け目は眼形や仮想眼形の1つであるが，最終的には眼形内の点を全て自分で埋めなくてはならないような特別な形であり，従って眼数は0眼になる．今回，いくつかの代表的な欠け目になる形について考慮した．例えば図9の右図の場合，B の点は仮想眼形であるが欠け目であるので0眼と判定される．

急所の点の数	急所内の敵石の数	眼数
3以上	—	2.0
2	2	1.0
	1	1.5
	0	2.0
1	1	1.0
	0	1.5
0	—	1.0

図 8: 眼形の眼数

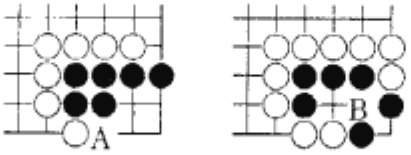

図 9: 仮想眼形の例と欠け目の例

6. 静的評価関数
   眼形の眼数合計が2眼以上ならば1(生き), 眼形と仮想眼形の眼数合計が1眼以下ならば−1(死に) とする. これら以外は全て0(不明) とし, 探索を続行する. 例えば, 図9の場合では, 両方の図とも眼数は1眼であり, 死にと判定される.

4 考察

前掲の問題5から問題8までについて, 読み切り型の探索と静的評価関数を使った探索の測定結果を図10に示す. ここで手数減少率とは, 以下の式のように, 静的評価関数の導入によって手数がどれだけ減少したかの度合いのことであり, 性能向上率とは時間がどれだけ減ったかの度合いのことである.

$$\text{手数減少率} = \frac{\text{読み切り型探索における探索手数}}{\text{静的評価関数を使った探索における探索手数}}$$

$$\text{性能向上率} = \frac{\text{読み切り型探索における探索時間}}{\text{静的評価関数を使った探索における探索時間}}$$

対象問題	読み切り型探索			静的評価関数を使った探索			手数減少率	性能向上率
	手数	時間	時間/1手	手数	時間	時間/1手		
問5	19,6234	41秒	2.1mSec	686	2.2秒	3.2mSec	29	19n
問6	24,249	57秒	2.4mSec	1,333	4.6秒	3.5mSec	18	12
問7	47,752	109秒	2.3mSec	1,024	3.4秒	3.3mSec	47	32
問8	155,451	410秒	2.6mSec	1,750	6.2秒	3.5mSec	89	66
平均	61,769	154秒	2.5mSec	1,198	4.1秒	3.4mSec	52	38

図 10: 静的評価関数による性能向上率の測定結果

静的評価関数の導入によって比較的不得意だった問題も数秒程度で解けるようになった. 特に, 問題7や問題8のような探索範囲のみが広がって問題の本質の変わらないようなものに大き

な効果が現われた．このことは中手による静的評価関数に導入よって，人間の読み筋に近づいている為と思われる．また，静的評価関数の導入によって，1 手あたりの処理時間の増加は 36% 程度で済んだ．これは次数コードによるパタンマッチングによって，中手の知識が高速に処理できた為であり，このことも性能向上率に大きく貢献した．

5 まとめ

今回匡碁の死活の知識としては最もポピュラーな中手の知識を使って静的評価関数を試作し，既存の読み切り型の詰め碁プログラムに導入した．新方式の探索では，今まで数分程度掛かっていた問題が数秒で解けるようになった．この性能向上の大きなポイントは，知識自体の有効性の他に，その知識を高速に処理できたことが上げられる．石の死活に対する同様の知識にはまだ「オス手なし」，「スミの曲り四目」，「眼あり眼なし」などがある．静的評価関数に入れる知識を増やせば増やすほど探索手数は減ることが期待できるであろうが，今回のような高速に処理するアルゴリズムがなければ，性能は必ずしも向上するとは限らないであろう．

今後の課題としては，まず，今回試作した中手の知識のアルゴリズムにはまだ改良の余地があり，その改良だけでも数倍程度の性能向上が望めると考えている．加えて，中手以外の有効な知識も増やしていき，より性能のよい探索の可能性を検討していく．

また，人間の場合大雑把に読んでみて問題がありそうだと判断してから精密に読み直すといったように，質の異なるいくつかの読み方をするようである．これなども，静的評価関数をいくつか用意することによって実現できるかもしれない．その可能性も合わせて検討していきたい．

[謝辞]
本研究の機会を与えて頂いた当 (財) 新世代コンピュータ技術開発機構の渕 一博研究所長と，有益な討論をして頂いた「棋士システム」タスクグループの皆様に深謝致します。

参考文献

[1] 実近憲昭, 沖 廣明, 吉川貞行, 吉岡 勉, 内田俊一：囲碁システム「碁世代の方法」, ICOT 研究速報 TM-0618, 1988

[2] 実近憲昭：碁における意思決定のプログラム化, 人口知能と対話技法研究会 26-4, 1982.6

[3] David B. Benson, Bruce R. Hilditch, J. Denbigh Starkey ; TREE ANALYSIS TECHNIQUES IN TSUMEGO

[4] 紺田茂實, 近山 隆：ESP プログラミング, ICOT 研究速報 TM-0134, 1988

[5] 白井良明, 辻井潤一：人工知能, 岩波講座情報化学 -22, 1982

[6] 武田晴夫：詰碁を題材とした人工知能の研究, (1980)

[7] 橋本宇太郎 解説 ; 二口外義 編集 ; 囲碁ブックス 詰碁を解くカギ, 山海堂, (1980)

[8] 趙 治勲：基本死活事典 上巻, 日本棋院, (1984)

[9] 実近憲昭：知識指向型碁プログラム Go.1 における探索モジュール, 情処第 33 回全国大会, 3M-1, 1986

[10] A.Barr, E.A.Feigenbaum ; 田中幸吉, 渕 一博 訳：人工知能ハンドブック, (1981)

[11] 実近憲昭：Expert プログラム — 位置づけと中間報告 —, 1982.4