ICOT Technical Report: TR-0837

TR-0837

棋士システム「碁世代」

清　慎一、実近　憲昭（電総研）、
沖　廣明（未来技研）、赤尾杉　隆（日立）

April, 1993

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191 ~ 5

Institute for New Generation Computer Technology

棋士システム『碁世代』

清 愼一 (財) 新世代コンピュータ技術開発機構
実近 憲昭 工業技術院 電子技術総合研究所　　沖 廣明 (株) 未来技術研究所
赤尾杉 隆*(財) 新世代コンピュータ技術開発機構

概要

本論文は，第五世代コンピュータプロジェクトにおける棋士システム「碁世代」の研究開発について紹介したものである。

棋士システムは大規模な知識処理システムであり，第五世代コンピュータプロジェクトにおける大規模並列推論マシンの性能検証，大規模知識処理システムの並列化方式の研究，負荷分散をはじめとする処理効率向上のための研究を目的としている。また棋士システムは，問題解決方式自体に不明な点を多く含むという特徴もあり，探索問題，曖昧性の処理，例外処理，協調問題解決などの人工知能の基本問題に関する新しい問題解決方法の研究題材としても適している。以上の研究目的から，並列推論マシン上で動く「並列版碁世代」と逐次推論マシン上で動く「逐次版碁世代」の二つが開発された。

本プロジェクトは第五世代コンピュータプロジェクトの中期から始められ，1992年のコンピュータ囲碁の世界大会であるICGC92では，逐次版碁世代は第4位となった。

*現 (株) 日立製作所

目次

1　序論

次の問いに答えることで，本研究の位置づけと目的を明らかにしよう。

- なぜ囲碁か？
- 囲碁に含まれる人工知能研究テーマとは？
- 人間を模倣することの意義は？
- 囲碁のプログラム化を困難にしている真の原因は？
- 第五世代プロジェクトにおける本研究の目的は？

1.1　なぜ囲碁か？

なぜ囲碁を研究するのか，チェスで十分ではないのか。元来，チェスと碁は同じ完全情報二人ゲームであり，数学的には同じ構造を持っている。チェスは人口知能研究の先駆けとされる1950年頃の当初から，研究対象として取り上げられてきた。そして今では，マスターレベルのプログラムも作られている。そこで，なぜ，今更囲碁を取り上げる必要があるのかという問いが生まれる。チェスが最初に注目された理由は，当時の人工知能研究の推進者であった研究者達の多くがチェスの文化圏に属していたという偶然もあったが，それよりも，チェスほどの複雑なゲームをうまくこなすプログラムを作ることができれば，人間の知能の仕組みに何らかの光を当てることができると考えられたからである。ここで，チェスプログラムの歴史を降り返ってみると興味深い事実が分かってくる。

発端は，Shannon C. の古典的論文にまで遡る。そこでは，既にミニマスク原理によるゲーム木探索の理論が明らかにされる共に，プログラム開発の段階として，次の３つのタイプを挙げている。

第一は，駒得を主とした簡単な評価関数を用いた固定深さ探索 (a タイプ)

第二は，先刈により候補手を絞り，チェスの知識を詰め込んだ複雑な静的評価関数による探索 (b タイプ)

第三は，目的駆動型の人間の思考の模倣 (c タイプ)

まずｃタイプは，将来の課題として除かれた。また，ａタイプは，当時，探索に関する研究も十分でなく，更にコンピュータの性能も著しく劣っていたため，実行不可能に近いことであったに違いない。唯一の可能性はｂタイプで，この線に沿ってプログラム作りが開始された。最初の全局を通してプレイするチェスプログラムは，Bernstein A. らによって作られた (1958)。1960年代の終り頃，Greenblatt ら (MIT) によるチェスプログラム MACHACK-6 は，C 級の実力 (平均的トーナメントプレイヤの水準) を示した。また，これはｂタイプ方式によるプログラムの頂点にあるといわれてる。その理由は，この方式に沿って評価関数などに様々な改良を試みたが，ついにこれ以上の向上は見られなかったためである。そして，当時たまたま計算機の性能テストのベンチマークとして実験的に作られたプログラム TECH が，なんと MACHACK-6 を負かしてしまった。TECH は単純な静的評価関数を用いた，深さ 6 迄の先刈りのない全幅探索方式によるもので，いわゆるａタイプに属していた。このことは，下手に先刈りして良い手を失うより，全ての手を考慮し，評価関数を簡単にしてその代わりに少しでも深く探索した方が，よい結果を生むことを示唆している。これは同時に，コンピュータの性能が ようやくａタイプ方式を可能にするほどに向上してきたことを示している。その後，多くのプログラムが競って，全幅探索方式を採用した。現在に至るまでこの傾向は続き，常に，時代の最先端を行くコンピュータを用いて，１手でも深く先読みすることにエネルギーが注がれた (1 プライ深まる毎に 200 点ランクが

向上すると言われている)。そして，ついに，チェス専用マシンや探索専用のカスタムメイドのVLSI迄登場するに至ったのである。以上のコンピュータチェスの歴史が示すように，最初はbタイプから始まり，除々に20年間発展を続け，そしてある時点でいきなりaタイプに切り替わり，以後急激にレベルアップを続け，既にマスターの域に達し，更に，限りなく最高峰のグランドマスターに近づこうとしている。結局，最も人工知能的アプローチとみられたcタイププログラムがコンピュータチェスの競技大会で活躍することはなかったし，今後の予測としても，永久に出番を失った印象を受ける。探索主体とはいいながら，一旦現在の高レベルを獲得した以上，人工知能研究のために，今更，レベルの劣るプログラムを作り続けるエネルギーをチェスから得られるかどうか疑問である。

そこで，登場するのが囲碁である。その理由は単に囲碁プログラムの開発が遅れているからというだけではない。それよりもチェスプログラムを成功に導いたゲーム木の「全幅探索」というパラダイムが，囲碁には全く通用しないという事実の方が大きい。つまり，最初，チェスに対して人間に対抗し得るプログラムができれば，人間の知能の仕組みがある程度分かるであろうと言う期待があったが，結局マシン性能に強く依存した「全幅探索」という意外な方向で決着がつき，ある意味で肩すかしを食わされた形になったため，それならば，いっそのこと，全幅探索が不可能なゲームにチャレンジした方が，残された人工知能テーマが鮮明になり，好都合と考えられるからである。

1.2 囲碁に含まれる人工知能研究テーマ

囲碁は，日常会話や車の運転のように，人間が比較的得意で，その機械化(プログラム化)が困難な事例の1つである。囲碁は，これらの事例に共通した人工知能の核心に迫るテーマをいくつか内蔵している。それらは，次のようなものである。

探索：探索パラダイムの限界の打破

曖昧：曖昧概念の処理，曖昧思考の導入

例外：未知の状況への対処(例外処理能力)

協調：システムの有機的統合処理

学習：経験の蓄積と法則化

各テーマについて，簡単に囲碁との関わりを述べよう。

1. 探索

探索は万能のアルゴリズムといわれており，人工知能においても基本的な手法の一つとして定式化されている。そのメカニズムの中心は，まず，可能な候補手を列挙し，次に各候補手を適用してその結果を調べることである。これはいわゆるgenerate and testと呼ばれている。そしてこのメカニズムそのものが組み合わせ爆発の根源である。この爆発を制御するために，検査するべき候補手をできるだけ絞る努力がなされる。絞るためには何らかの検査が必要になる。これは一間堂々巡りのように見えるがそうではない。このことは，絞る過程を複数段階に分けて行うことにより，比較的無理なく達成される。即ち，最初は粗い検査(コスト小)により，ふるいにかけ，候補手数が少なくなるにつれて検査の精度を上げていけばよい。この原理は日常生活でもよく使われている。例えば応募者の多い入学試験はよく二度に分けて行われる。この原理は，認識の段階的深化として，より一般的な立場から捉えることもできる。

2. 曖昧

曖昧概念とこれに基ずく曖昧思考はファジィ理論の中心テーマでもあるが，これは曖昧さが人間にとって本質的であるという観点に立っている。より狭い見方をすれば，定量的，数値的処理よりも定性的，記号処理を目指しているともいえる。従来，絶対的ともいえる信頼をえてきた科学的方法は分析的解析的な性格が強く，細かく分析してゆけばいつかは真理に到達するという考えが根底にあるようである。また，すべてのデータが努力すればやがては入手可能という考えも一方にある。しかし，人間の扱う概念の中には，分析すればかえって実態を失ったり，あるいは無意味になったりするものも多い。また，時間とメモリの制約から不完全な知識のまま行動したり，判断を下すことは日常的なことである。碁における抽象概念のいくつかは「自然界」に存在しないものであり (例. 厚み)，これに基づく思考は必然的に曖昧思考になる。

3. 例外

人間は，自分の知識外の状況に置かれたとき，非常事態の知識体系に切り替えて問題解決を試みる。このメカニズムを模擬するためには，まず定常状態と非常状態とを区別できることが必要であり，次に非常状態の意味を知る必要がある。非常状態は不利な場合もあれば有利な場合もある。例えば，不利の場合は不利と判断する原因を突きとめる。その原因を排除または回避する手段を，原因の性質に応じていくつか用意する。非常事態における価値基準は定常状態のときと異なる。非常事態で，ある目的を達成するために提示される手段は，定常状態で同じ目的を満たす手段と比べて，一般に条件が緩められる。例えば部屋から出るとき，通常はドアから出るが，鍵が壊れてドアが開かないときは窓から出ることをも考える。ドアも窓も外に通じる通路という点で共通した手段だからである。一般に非常事態での処理は，非能率で不正確である。なるべく定常状態でカバーした方がよい。即ち，なるべく多くの場合を想定して定常状態として処理する方が良い。非常事態も，同じことを何度か経験すれば，新しい定常状態として分類されるようになる。従ってこのテーマは学習と密接な関係にある。

4. 協調

特に定まった解法のない問題解決状況に置かれたとき，人間はまず，その問題に関係のありそうな要因をできるだけ洗い出す。次にその要因間の因果関係を明らかにしようとする。このように，いくつかの同時生起，または同時進行する事象を有機的に統合し，統合処理する能力を持っている。碁においては，いくつかの味 (可能性) を見ながら相手の出方に応じて手段を作って行く場合に，特にこの能力が発揮される。例えば，ａとｂという２つの目標があり，それぞれ単独に追求した場合は，ともに今一歩で失敗に終るような状況を考えてみよう。このとき，ａを追求する過程で生じる環境の変化によって，ｂが成功する条件が成立するようになる場合がある。即ち，まずａを仕掛け，相手がこれを防ぐ隙にｂを達成しようとする発想が生まれる。このテーマは，協調処理の一つの形態を示唆している。

5. 学習

人間の囲碁能力は，経験的に習得した知識 (記憶) に依存しているものと考えられる。ここで，心理学からの知見によると，知識 (記憶) は，長期記憶と短期記憶の区別があり，更に長期記憶は，定石，手筋，過去の対局経験 (エピソード)，各種の囲碁概念といった，宣言的な知識 (意識にのぼり，記述可能な知識) と曖昧属性を評価する能力，作戦を立てる技術，先読みを制御する能力等の手続き的知識 (無意識に行われ，記述不可能な知識) に分けられる。宣言的な知識は，コンピュータにデータとして，人間の手を介して，まがりなりに入

力可能であるが，手続き的知識の獲得メカニズムに関しては，現状ではほとんど明らかにされておらず，この知識を組み込むためには，その表現形式を含めて，大胆な仮説に基づくモデルによるしかない。

1.3 人間を模倣することの意義

コンピュータ囲碁の開発において，現状では，人間の碁を模倣する以外に方法がないというのが正直な所である。しかし，同じ模倣でも，

・人間の碁にヒントを得た機能的模倣

・人間の碁をなるべく忠実に再現しよう

とする直接的模倣の二つの行き方があり，その目的とするところがかなり違ってくる。機能的模倣では，人間の碁はあくまでヒントを求めることにあり，これを機能的に改良することは勿論として，途中で，機能的によりすぐれた新たな方法が見つかれば，いつでもそれに移行しようとする態度である。一方，直接的な模倣においては，思考形態まで含めてできるだけ忠実に模倣を試みることで，人間の意思決定における思考モデルを構築し，そのメカニズムに対する理解を深めることを目指している。

本研究の目標は，最終的には，人間の碁の直接的な模倣に重点を置いている。しかし，過渡的な開発段階では，機能的模倣の区別はつけ難い。特に，ハードウェアとしての人間とコンピュータの差を考慮したとき，人間の忠実な模倣が，機能面はともかく，性能面からみて必ずしも最適になるとは考えにくいからである。人間の碁のモデル化を進めるにあたり，多くのことが無意識下で処理されているため、最初に遭遇する困難は，表現の問題である。ここで，人間の碁の特徴，特に能力的な特徴を列挙してみよう。これらのほとんどは，無意識下の処理と考えられる。

- 注意の焦点化

  詳細な読みをしなくても，問題の有りそうな場所に，多くの場合ほとんど一瞬の内に注意が向けられる。少なくとも，数箇所に絞り込むことができる (本当に最も重要な問題のある場所が，注意の集中した場所の候補に常に，含まれているかどうかは，今は問わない)。

- 読みの制御

  手所では，場合によって深さ 20 手を越えるような先読みもできる。
  - アジ／利きを利用して読み筋の組み立てができる。
  - 条件不足でもう少しの所で，失敗したような読み筋も新しいアジ／利きとして記憶しておき，後で利用することができる。
  - 多重標的も必要に応じて設定できる。

- 意図に基づく作戦

  - 意図に基づいて作戦を立てることができる。
  - 一段落の判断があり，そのタイミングで次の作戦を立てる。
  - 先手／後手を意識し，作戦作成に活用できる。
  - 状況 (形勢，局相) に応じて作戦が切り替えられる。
  - 相手の着手意図が理解でき，それに呼応した着手ができる。

- 一貫性

  - 作戦に沿って一貫性のある着手ができる。

以上は，囲碁以外のゲームまたは一般の問題解決にも共通して，人間が示す一般的な能力である。従って，これらの機能のモデル化は，一般の人工知能的問題解決のモデル化にもつながる。

1.4 囲碁のプログラム化を困難にしている真の原因

- コンピュータチェスとの比較

  チェスで成功した全幅探索方式が囲碁に適用不可能ということを確認しておこう。これには，二つのことが関係している。一つは，探索空間の規模の差であり，もう一つは，静的評価関数の複雑さの差である。まず，探索空間について，チェスでは，局面あたりの可能手の数は約 40 で、計算機のパワーが 許す限り (平均的に 7,8 の深さ迄，また現在最強と目される DEEP-THOUGHT では 10 手迄) 先読みする。一方，囲碁の場合，局面当たりの平均可能手は 200 手を越えるため，仮に静的評価関数の複雑さがチェスの場合と全く同じとしても，同じ計算パワーのもとでは，高々 3 手先が限界であろう。これでは，戦術的にも，戦略的にも意味のある結果はほとんど期待できない。次にもう一つの理由である，静的評価関数に関して，チェスでは「駒得」が主な評価因子であり極めて簡単な設計にすることができる。一方囲碁では，静的評価関数に相当するものは，形勢判断である。チェスの「駒得」に相当する評価因子を強いて挙げると「確定地」であるが，「確定地」の計算に要する時間は，「駒得」の計算に比べ，著しく長く (ざっと見積もっても 1000 倍は違うだろう)，又，「駒得」が他の位置的な評価因子と比べ常に大きな比重を占めるのに対し (駒得以外の評価因子の大きさは全て考慮しても，高々ポーン一個分にも満たない)，「確定地」は，重要な評価因子ではあるが，他の評価因子 (模様，弱石など) と比べ必ずしも比重が大きいとはいえない。従って，碁の静的評価関数 (形勢判断) は遥かに複雑なものになり，そのため実際には 2 手先の先読みさえ怪しくなる。

  コンピュータチェスは，探索指向といっても，決して全局を通して探索モードで着手決定が行われているわけではなく，序盤は膨大な定跡データベースでカバーされ，終盤もまた，盤上での駒数が 6 個程度になってからは，これ又よく整備された詰めのデータベースが作動しており，中盤の部分のみに探索モードが適用されているにすぎない。すなわち，最も曖昧な状態にある序盤と，最も作戦的思考を必要とする終盤を，知識ベースでバイパスしなければ，とても現在の強さに達しなかっただろうと推測されている。

  一方，囲碁では，序盤における定石は，チェスにおける定石のような全局的なものではなく (三連星，中国流といった，定石に相当する布石があるが，チェスほど網羅されてはいない)，隅及びその隣辺における局所的な知識であり，各隅において，全局的な観点から定石の選択が求められる。囲碁の中盤は，序盤，終盤と比較して，戦略，戦術的にも最も複雑となる時期である。プログラム化が最も困難となる時期でもある。碁の終盤は，先手，後手の概念を用いたヨセの計画の問題が生じる時期でもあるが，着手の評価は局所的な領域の出入りの問題に還元され，中盤，序盤に比べれば，比較的プログラム化がしやすい時期ではある。

  結局，序盤，中盤，終盤と分解して，チェスと比較しても分かるように，囲碁では，人間の碁の模倣にその方法を求めており，そのことに由来する困難さが生じている。

- 人間の碁の模倣に由来するプログラム化の困難さ

プログラム化を前提とした人間の囲碁のモデル化を進めるにあたり，最初に遭遇する困難は，表現の問題である。碁は，一方で感性(感覚)のゲームとも呼ばれている。それは，盤上に出現する不定型な石の配置に対して何らかの評価を下す必要があるからである(例えば，厚い／薄い，軽い／重い，味が良い／悪い，凝り形など)。これらは全て抽象的な概念である。人間が人間に対してこれらの概念を教えることは，なんとか可能である。しかし，その方法はほとんどの場合，繰り返し例を示すことであり，生徒はこれを感覚的に「会得する」しかない。人間は(碁のエキスパート，プロも含めて)，自分がいかなる基準で，またいかなる手続きを経て，次の一手を決めているかを客観的な言葉で表現しようとすると，著しい困難を感じるに違いない。その理由は多くのことが無意識に処理されているためと考えられる。とにかく，表現できないことはプログラム化もできない。即ち，無意識下の作業も含めたモデル化が必要となる。無意識下の作業のモデル化は，推測による仮説に基づく他はない。そしてモデルの正しさを証明するものは，このモデルに基づくプログラムの動作と人間の振舞いとの類似性のみである。

1.5 第五世代プロジェクトにおける本研究の目的

「碁世代」とは，第五世代コンピュータプロジェクトの一環として，ICOT が取り上げた棋士システム開発プロジェクト(CGS)で開発された囲碁システムに付けられた名前である(最初，棋士システムは囲碁と将棋の両方をテーマとすることを考えていたが，そのうち囲碁一本に絞られることになった)。

棋士システム「碁世代」の研究目的は大きく二つに分けられる。一つは知識処理に関する研究，もう一つは新しい計算機パラダイム(並列処理)に関する研究である。即ち，囲碁は AI の核心に迫るテーマを内蔵しているという認識のもとに，ICOT で開発された新しいプログラミング言語や計算機システムがどの程度，人工知能分野の問題解決に敵しているかを評価するためと，もう一つは，人間プレイヤの思考過程をモデル化し，シミュレーションすることで人工知能的問題解決のための新しい方法論が確立されるかもしれないという期待で始められた。

碁世代は，人間の碁を模倣する立場を取っている。但し，人間の模倣といっても，忠実な模倣ではなく機能的模倣を目指している。ハードウェアとして，人間とコンピュータは全く異質なものである以上，両者を動かすアルゴリズムは異なって当然だと考える。機能的模倣が成功したか否かは，結局人間を負かすプログラムを作る以外に客観的な証明の仕方はない。問題はどの程度の技量の人間を基準とするかであるが，一応，5級に設定された。これはアマチュアの囲碁人口密度のピークであるがこの辺りにあるといわれているからである。

2　碁を打つモデル

従来，囲碁のような完全情報二人ゲームのプログラムではアルファ・ベータ枝刈法によるゲーム木探索手法が主として用いられ開発されてきた。特にチェスを中心として研究が進められ，チェスプログラムとしては現在相当な成果を上げている。しかし碁にチェスと同じ探索指向型の手法を適用しようとすると，碁の探索空間の規模がチェスと比べ桁違いに巨大なため，うまくいかないことが判明している。即ち，囲碁のプログラム化のためには，従来にない本質的に新しい方法が求められたのである。

そこで，碁世代のアプローチは人間プレイヤーが通常行っている思考法を出来るだけ忠実にシミュレートしようとすることを目指した。

我々は，人間プレイヤーが碁を打つ一つの思考モデルとして，「3 段階モデル」を提案する。このモデルは，人間が問題局面を与えられて，着手を決定する過程が大体 3 つの段階「状況の把握」「問題 (ケース) の設定」「解決手段の具体化」を経て行なわれることを示す。

2.1　「3 段階モデル」の概要

人間の碁の思考過程は大体，状況の把握 (第 1 段階)，問題 (ケース) の切り出し (第 2 段階)，問題解決手段の具体化 (第 3 段階) の順序で進む (図 2.1 1)。

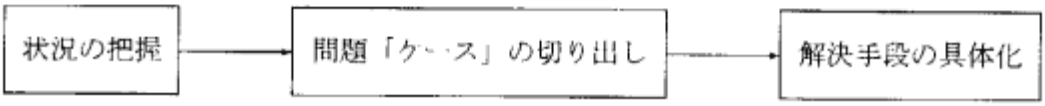


図 2.1–1 : 3 段階モデル

実際には，各段階はそれほど明確に分離しているわけではなく，少しずつ重なりあっている。すなわち，状況把握の途中で既に切り出そうとしている問題の成立を支持する「特徴」の抽出が始まり，問題の切り出しの途中で，解決手段の方向付けなどが開始されるからである。もう少し詳細に言うと，状況認識の途中で，問題の存在することが大体分かる。その問題の性質により状況認識を深める方向と範囲が絞られ，更に問題構造の理解を深めていく。問題構造の理解を深めるということは，解決対策についても，ある程度知る必要があり，問題構造を理解したとき，その問題の解決手段に対して一応の方針を得ている。また，問題の性質が分かることにより，状況認識も進められる (図 2.1–2)。しかし，個々の問題の解決手段を組み合わせて，一つの問題を解くことは第 1 段階の範疇ではない。

1. 状況の把握

   この段階での状況とは，検出すべき問題対象の背景となるべきものを言い，「盤上の石の配置とそれに関わる属性」，「時相 (序盤／中盤／終盤)」，「形勢」，「方針，作戦」がある。

   盤上の石の配置とそれに関わる属性を調べる時，人間は数回に渡って走査している。最初の走査では，原始レベル (各石やそのつながり具合など) の特徴抽出をし，走査を繰り返すことによって，より複雑な特徴 (近接する一塊の石や模様など) 抽出が行なわれる。このように走査を繰り返すことにより，盤面のより重要な部分に焦点を当てて認識作業を絞り込みながら行なっていく。

2. 問題 (ケース) の切り出し

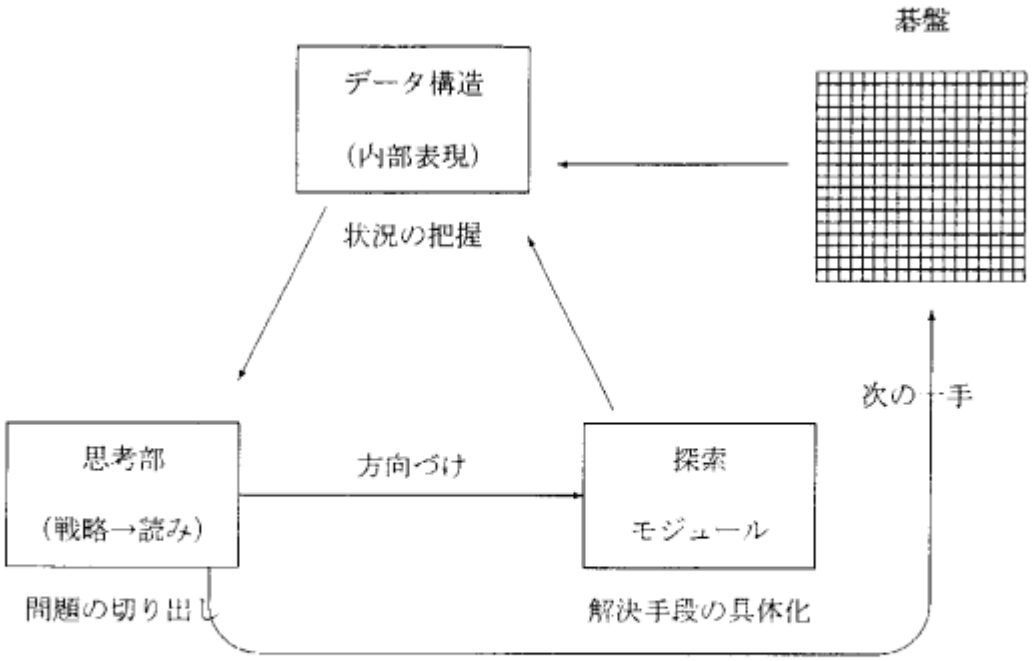


図 2.1-2：3 段階モデルの構造

問題の切り出しでは，戦場の選定，その戦場での戦略，手段としての候補手，その手により成功する可能性や戦果の大きさを見積もることを行なう。「人間は与えられた問題解決の場で，判断の基準として過去に遭遇したケースに関する知識に基づいて行動している」，という仮定を設けた。即ち，問題の切り出しや方針を決定する着眼点の選出は，碁の知識によって設定されたいくつかのケースの観点から行なわれる。

3. 問題解決手段の具体化

問題を検出すると，その問題を解決するための手段を求める。解決手段は，過去に遭遇したケースに関する知識から求められる。そして問題と戦略を求めたら，その実現性を確認するために部分的な「読み」を行なう。また，「読み」によって失敗した戦略同士を組み合わせて，新しい戦略を組み立てたりもしている。

最終的に次の一手を求める時は，いくつかの候補手から最も戦果の大きいものを選んだり，ある戦略に基づいて候補手を選んだりする。

3 逐次版「碁世代」

3.1 概要 (方針，歴史)

碁世代には，逐次版と並列版の2種類がある。逐次版は，人工知能の基本問題に関する研究という目的から開発され，並列版は，並列処理の研究という目的から開発された。ここでは，逐次版碁世代について説明する。

逐次版碁世代は，前章の碁を打つモデルに従って作られた棋士システムである。逐次版碁世代は，中国ルールに従って囲碁の対局を終局まで行うことができる。

1. 目標と方針

   碁世代の目標は，アマチュアの中級レベルの囲碁対局システムの開発とした。

   開発の方針は，チェスなどに代表されるコンピュータゲームプログラムで成功した探索主体の方法では，囲碁プログラムは強くなれないとの観点から，人間プレイヤーの思考方法のシミュレーションを通じて，碁世代を開発することとした。

2. 経緯

   「碁世代」は，第五世代コンピュータプロジェクトの中期 (1985 ～ 88 年) から開発が始められた。1985 年～ 1990 年は基本的な囲碁プログラムの枠組を設計し，逐次型推論マシン PSI 上でアマチュア初級程度のプロトタイプを開発した [実近 88],[実近 91-2]。

   1991 年に入って，候補手の評価値の付け方・着手決定方法についての改良，および各種の知識の整備を続けた [実近 91-1]。総合評価として，逐次版碁世代で 1992 年のコンピュータ囲碁の世界大会に出場した。

   また，第五世代コンピュータプロジェクトの後期 (1989 ～ 92 年) は逐次型推論マシン上での囲碁プログラムを元に，並列型推論マシン Multi-PSI 上で並列版囲碁プログラムを開発した (4章参照)。

3. 開発体制，動作環境

   1985 年に囲碁プログラム開発チーム (CGS-TG : Computer Go System Task Group) が発足してから，構成員の顔ぶれは変わっていったものの，設計者・プログラマそれぞれ約 3,4 人の体制であった。

   また，プロジェクトチーム発足時は，囲碁プログラムの開発経験者及びそれに精通している情報科学関連の研究者による棋士システムワーキンググループの協力も得ていた。なお，本プロジェクトは ICOT と工業技術院電子技術総合研究所との共同研究であった。

   碁世代は，ICOT で開発された逐次型推論マシン PSI (Personal Sequential Infernce Machine) で動く。なお，PSI のオペレーションシステムは SIMPOS である。

   プログラムは ICOT で開発された言語 ESP(Extended Self-contained Prolog) で書かれており，ソースプログラムの総量は約 3 万行である。ESP は，おおよそオブジェクト指向を取り入れた Prolog である。

3.2 構成

逐次版碁世代は，対局システム・知識エディタ・評価用ツールの３つからなる。

対局システムは，中国ルールに従って囲碁の対局を終局まで行うことができる。知識エディタを使って，対局システムに知識を容易に入力することができる。評価用ツールでは，対局システムにおける知識の評価を行うことができる。

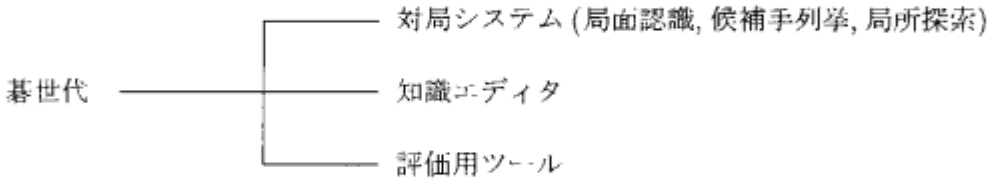


図 3.2-1：碁世代の構成

3.2.1 対局システム

実際に対局を行う為のプログラムである。与えられた局面を認識し次の着手を決定する対局モードと，認識した結果を参照する為の実験解析モードとがある。

1. 対局モード

   碁世代は，中国ルールに従って囲碁の対局を終局まで行なうことができる。

   碁世代の着手決定の手順は，２章の碁を打つモデルの３段階モデルに基づいて設計された。しかし，この３段階モデルは互いに重なりあっている部分もあり (呼びあっている部分もある)，きれいに分けて考えることはできない。そこで，碁世代では図 3.2-2のように３つの段階を経て着手決定を行なう。まず着手により変化した盤面上の石の生死などの局面の認識を行ない (状況の把握)，それに基づいて候補手を列挙し (問題の設定，解決手段の具体化)，その候補手の評価値が最も高い点の座標を次の着手位置とする (解決手段の具体化)。

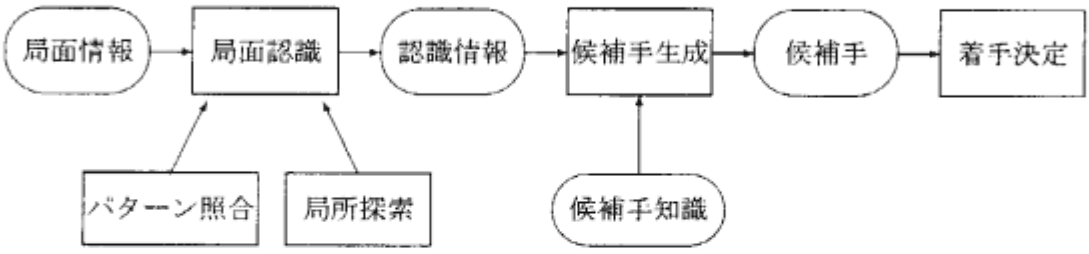


図 3.2-2：碁世代の処理の流れ

2. 実験解析モード

   対局モードにおいて入力待ちの状態の時，及び実験解析ツールから実験解析モードへ移行できる。実験解析モードでは，局面認識によって更新された各種データ構造の内容や，着手決定によって求められた各種候補手や形勢が参照できる。また，各種の局所探索を駆動し，その探索の過程や結果なども見ることができる。

3.2.2 知識エディタ

本システムにおける知識の一部はデータベース化され，対局中に必要に応じて適宜参照される。知識エディタはこれらデータベース化された知識の入力及び管理を容易にする為のツール群である。知識エディタには，ポテンシャルエディタ，定石エディタ，手筋エディタなどが用意されている。

3.2.3 評価用ツール

囲碁対局システムのようなプログラムでは試行錯誤的な改良が余儀ない。評価用ツールは，改良した効果をできる限り正確に評価する為のツール群である。評価用ツールには実験解析ツール，自動テスト機能，棋譜棋力判定ツールなどがある。

実験解析ツールは，ある局面を入力し，その局面の各種データ構造の内容や各種の候補手や形勢を表示する。また，各種の局所探索を駆動し，その探索の過程や結果など見ることができる。

自動テスト機能は，局面認識，局所探索，着手決定や形勢判断に関するいろいろな問題を予め作っておき，それらの問題に対する正答数によって改良の効果を評価する機能である。

棋譜棋力判定ツールは，改良前と改良後のプログラムでどのような着手の差が表われたかを見ることができる。

3.3 局面認識

人間は，盤面を認識する際に，石の配置を単に座標の集合として捉えるのではなく，戦術的に意味のある石の集団として捉えている。その石の集団を認識するとき，人間は一回のパスによって盤面を認識しているのではなく，何回かのパスを通して次第に原始レベル (各石やそのつながり具合など) からより複雑な高次のデータ (近接する一塊の石や模様など) へと順に認識していく。そして，同時に盤面のより重要な部分に焦点を当てて認識作業を絞り込んでいく。

碁世代は，人間プレイヤーのように盤面上の単なる石の配置から，高いレベルのデータ構造を作り出し (図 3.3-1)，それらの属性 (地の大きさ, 形, 包囲度など) を算出する。このような階層的なデータ構造は，人間が盤面を認識する際に細かい部分構造から，ある抽象的な見方をするというプロセスによく似ている。また，このような階層的に盤面を認識することによって，盤上のどの部分が重要な部分かがスムーズに把握できる。

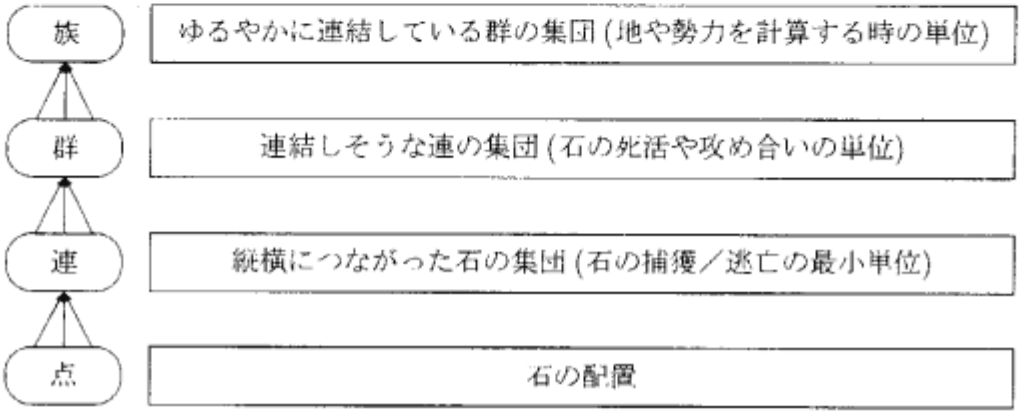


図 3.3-1：意味のある集団形態とデータ構造の関係

3.3.1 局面の表現 (データ構造)

3.3.1.1 点データ

石の配置を認識する為に使われる，碁世代における認識の最小単位である。
盤上の全格子点に対応する点オブジェクトから構成されている。
点データにはいくつかの属性がある。以下に主なものを挙げる。

- 色

  点の色 (黒 / 白 / 空)

- 座標

  点の X 座標と Y 座標で、左上隅が (1,1) である。

- 基準座標および基準座標変換コード

  基準座標 (X,Y) と某準座標への変換方式のコード

碁盤を図 3.3-2のように 8 等分し，図の中の 0 の部分を基準部と呼ぶことにする。

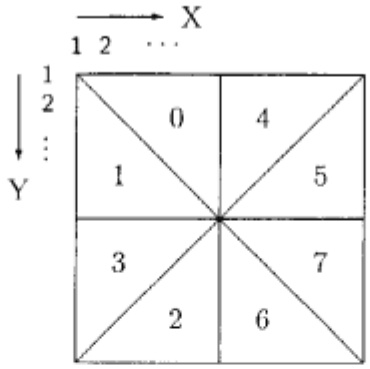


図 3.3-2：基準座標変換コード

基準座標とは，盤上の各点に対して，その点と対称の位置関係にある基準部内の点の座標のことである。

また，図 3.3-2で，8 等分された部分の中の数字は各部分内の点における，基準座標変換コードを示している。盤上の各点の座標を (X,Y) とする時，基準座標変換コードと基準座標との関係を表 3.3-1に示す。

表 3.3-1：基準座標への変換方法

基準座標変換コード	基準座標 X	基準座標 Y
0	X	Y
1	Y	X
2	X	路数 +1−Y
3	路数 +1−Y	X
4	路数 +1−X	Y
5	Y	路数 +1−X
6	路数 +1−X	路数 +1−Y
7	路数 +1−Y	路数 +1−X

- 高さ

  盤端からの距離 (盤端の時が 1)

  $$\text{高さ} = \min(\text{基準座標 X}, \text{基準座標 Y})$$

- 隣接点

  その点に隣接する点の集合

- ダメ点

  隣接点に属する点の内，空の点の集合

- ダメ数

  ダメ点に属する点の数

- ポテンシャル値

  盤上の石 (黒 / 白の点) はその周囲の点に，距離に反比例するような勢力を及ぼすと考えられる。
  ポテンシャル値とは，周囲の石からの勢力を合算したものである (最大 100) 。ただし，死群の石からの勢力は無視する。

  > 1991 年度までの旧碁世代では，模様や形勢を表現するのにこのポテンシャル値がよく使われていたが，最新の碁世代 (以降，現碁世代と呼ぶ) では模様を表現するのに弱領域 (3.3.1.4章) を使うようになったため，あまり重要な意味を持たなくなった。

  ある石によって生じるポテンシャル値と，それを目数に換算した等価目数を図 3.3-3に示す。ただし，隅や辺に近い石では単純な加算ではなく，盤外に味方石があるかのような見方をして，ポテンシャル値を算出している。等価目数関数の算出方法は，ポテンシャル値が 70 以上になったら 1 目と数え，70 未満は式「等価目数関数 = (10/7) × ポテンシャル値」より求める。

  盤上に置かれた各石について，黒石であれば正の，白石であれば負のポテンシャル値をパターンに従い周囲の点に加算する。それらの合計値が各点のポテンシャル値となる。

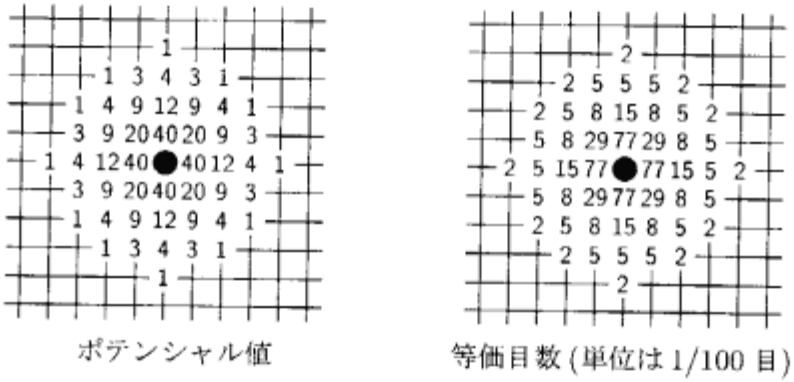


図 3.3-3 : ポテンシャル値と等価目数

- 連

  連に属している点の場合は，その連

- 結線

  結線の端点である場合は，その結線

- 遮断結線

  結線の遮断点に属している場合は，その結線。黒結線 / 白結線のそれぞれ別々に保持している。同色の 2 つ以上の結線の遮断点である時は，より強い結線を優先する。

- 群

  群の領域に属している点の場合は，その群

- 族

  族の領域に属している点の場合は，その族

3.3.1.2 連データ

石のアタリ，ヌキといった状態を認識する為に使われる。石の生き死にの最小の単位である。互いに隣接する同色の石の集合である連オブジェクトから構成されている。
連データにはいくつかの属性がある。以下に主なものを挙げる。

- 色

  連の色 (黒 / 白)

- 点

  連に属する点の集合

- 石数

  連に属する点の数

- ダメ点

  連に隣接する空点の集合

  連に属する点のダメ点の和集合と同じ

- 死活，脱出点，捕獲点

  ダメ数 3 以下の連に対して黒番 / 白番で捕獲探索 (3.4.3.2章) を駆動し，表 3.3–2に従い死活を求める。ダメ数 4 以上の連の死活は「生き」とする。

  > ダメ数が 4 以上の連はほとんどの場合「生き」であること，連の捕獲探索にかかる処理時間が，ダメ 4 になると急に増加すること，により ダメ 3 以下は探索し，4 以上は「生き」とした。

表 3.3–2 : 連の死活

連側手番での捕獲探索結果	連の敵側手番での捕獲探索結果	
	成功	失敗または不明
成功または不明	中立	生
失敗	死	生

連側手番での捕獲探索結果が成功または不明の時の解の手を脱出点とし，連の敵側手番での捕獲探索結果が成功の時の解の手を捕獲点とする。

- 種石 / 非種石

  図 3.3–4のようなキリチガイの状態があり，4 つの石の連の死活が全て生か中立の時，各点の連を種石と呼ぶ。種石以外の連を非種石と呼ぶ。

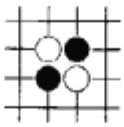


図 3.3-4：キリチガイの状態

3.3.1.3 探索用点，連データ構造

点と連のデータ構造，探索処理向けに改良したものが探索用点，連データ構造である。探索用点，連データ構造によって点，連オブジェクトの更新に要する処理時間及びメモリを大幅に削減できる。

通常の点，連データ構造と探索用の点，連データ構造を1つにまとめることは可能であり，望ましいことであるが別々に管理している。つまり，対局における打着時には両方のデータ構造を同じように更新し，探索における仮想的な打着時には探索用の点，連データ構造のみ更新している。

1. 探索用点，連オブジェクトの属性

   探索用点，連オブジェクトでは，点オブジェクトと連オブジェクトを別々に管理していない。連の属性は，連の石の中で最後に打着された石に対応する点オブジェクトの中に保持されており，最終打着以外の点オブジェクトには，その点が連の最終打着であった時の連の属性が保持されている。また，このような連の中の石の履歴は，親点，子点という属性によって管理されている。

   (a) 色，座標，標準座標，標準座標変換コード，高さ，隣接点，ダメ点，ダメ数
   点オブジェクトと同様

   (b) 親点
   親の点

   (c) 子点
   子の点の集合

   (d) 子数
   子点に属する点の数

   (e) 連石数
   連に属する石の数

   (f) 連ダメ数
   連に属する石に隣接する空点の数

   (g) 連ダメ点
   連ダメ数が3以下の時のみ，連に属する点に隣接する空点の集合
   連のダメ点の管理をダメ数3以下に制限したのは以下の理由による

   - 可変長のデータの管理は処理が重く，また，メモリの無駄使いの原因となる。
   - 現状の探索中に連のダメ点の位置が必要になる処理は，ダメ数が3以下の場合がほとんどである。

ただし，まれにダメ 4 以上の場合でもダメ点の位置が必要になる場合があるので，その場合は子点の属性から算出するメソッドが用意されている。

2. 更新方法

探索用点，連オブジェクトの更新方法を図 3.3–5 を例に説明する。図 3.3–5 の左図は 6 個の白石から成る連を示している。石の中の数字はその順序で打着されたことを示している。

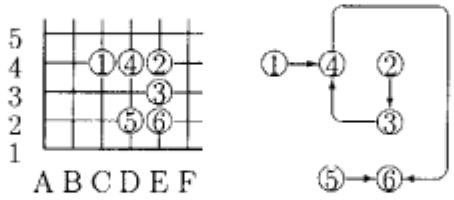


図 3.3–5：探索用点，連オブジェクトの更新方法

また，図 3.3–5 の右図は，各点の親 (点) の様子を現わしている。打着時にその点に隣接する味方石がある時，それぞれの味方石の連の中で最後に打着した石が子 (点) になり，打着した点がそれらの子の親となる。例えば今 6 の石を打着したとすると，1〜4 の石からなる連の中の最後に打着された石である 4 の石と 5 の石が 6 の石の子となり，6 の石は 4 と 5 の石の親となる。

打着した時の探索用点，連オブジェクトの更新処理の内容を以下に示す。

- 打着点の色をセットする。
- 打着点の全ての隣接点のダメ数を 1 減らすとともに，ダメ点から打着点を除く。
- 打着点に隣接する敵石がある場合，その敵石を含む連の中で最後に打着された石の連ダメ数を 1 減らすとともに，連ダメ数が 3 以下の時は連ダメ点から打着点を除く (もしくは再算出する)。
- 打着点に隣接する味方石がある場合，その味方石を含む連の中で最後に打着された石の親点に打着点をセットするとともに，その石を打着点の子点に追加する。
- 打着点の子点の属性から，打着点の連石数，連ダメ数，連ダメ点の属性を算出しセットする。

また，探索のバックトラックの時やアゲ石の処理の場合に必要となる，打着した石をハガす時の更新処理の内容を以下に示す。

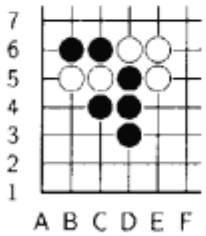
- 打着点の全ての隣接点のダメ数を 1 増やすとともに，ダメ点に打着点を追加する。
- 打着点に隣接する敵石がある場合，その敵石を含む連の中で最後に打着された石の連ダメ数を 1 増やすとともに，連ダメ数がまだ 3 以下の場合は，連ダメ点に打着点を追加する。
- 打着点に隣接する味方石がある場合，その味方石を含む連の中で最後に打着された石の親点をリセットするとともに，打着点の子点をリセットする。
- 打着点の色を空にする。

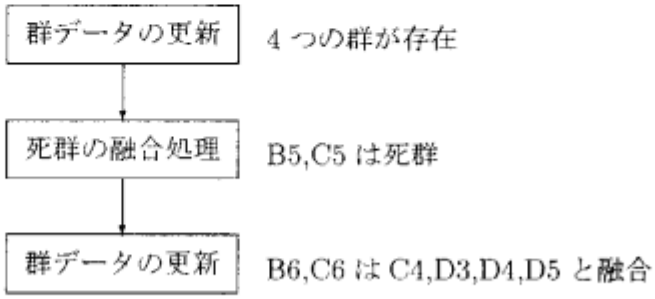
3.3.1.4 群データ

距離が短く，ほとんどつながっていると思えるような石の集団の状態を認識する為に使われる。群は，人間が死活を考える際の基本単位に相当し，戦略の基盤となる最も重要な集団であると考えられる。

碁世代では，強結線 (3.3.1.6章) でつながる同色の石の集合で定義される群オブジェクトから構成されている。

群データ構造の更新は2つのフェーズで行われる。最初のフェーズで，対象となる群の周囲の群が全て生きていると仮定して，対象となる群の認識処理を行なう。全ての群の認識処理が終ったら，死群の隣接敵群を一つの群として融合する。そして，その新しい群につき，もう一度群データの更新を新しいフェーズで行なう。





群データにはいくつかの属性がある。以下に主なものを挙げる。

- 色

  群の色 (黒 / 白)

- 自己石

  領域に属する自己側の石の集合

- 石数

  自己石の数

- 群サイズ

  領域の点と領域の点の外側2路以内の点の数の総和。群の死によって得られる相手側の利得 (自己側の損失) の指標となる。

- 中地

  群の領域内の地の数。群の死活探索 (3.4.3.4章) より求まる。先着中地 (群側の手番でできる中地) と後着中地 (敵側の手番でできる中地) がある。

- 援軍

  弱結線でつながった群

- 弱領域

  現時点では群の領域ではないが，何手かかけると領域になりそうな範囲のこと。(ポテンシャル関数より縦横の関係に重点をおいた勢力圏のようなもの)

  模様の表現に使い，模様に関する候補手の生成や，群の強度の計算に使われる。

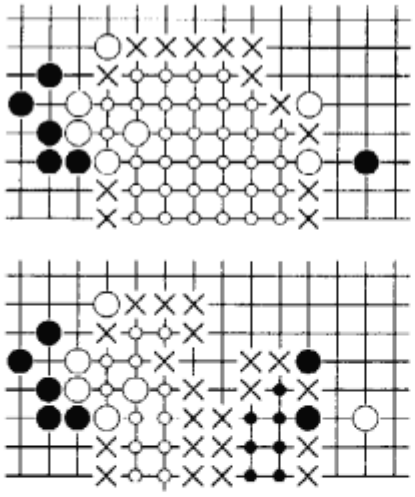
  弱領域は以下のように定義する。

黒の先着中地：8目　　黒の後着中地：2目

図 3.3 6：先着中地と後着中地

- 弱領域は互いに隣接する同色の弱領域点の極大集合である。
- 弱領域点はどの群にも属さない点のうち，味方群までの距離の方が敵群までの距離よりも近い点を言う。
- ただし，味方／敵の群についてはその点から上下左右の各 4 方向にまっすぐ 5 路までの点でそれぞれ最初に到達した群とする。
- 敵群に到達するとは，群もしくは敵群同士をつなぐ結線の点に到達することを言う。
- 味方群に到達するとは，味方群の点に到達することを言う。ただし，盤端に味方石同士の群が五間以内で対峙している時は盤端の 1 路外側に味方群があるものとする。
- また，強い敵群までの距離より近い距離の味方群がある方向の数をその点の勢力と呼ぶ。
- 辺上で敵石と対峙していて，盤の中央方向にも敵石がある場合は弱領域としない。

弱領域の各点の中で内点の点の勢力の和の 1/4 を弱領域の大きさとする。弱領域の内点とは，その隣接点がすべて同色の弱領域または同色の群の点とする。また，弱領域の内点に隣接する同色の群の空点が内点であるとき，その点も内点に加える。



×は弱領域または群の境界

小さい白丸は白の弱領域

×は弱領域または群の境界

小さい白丸は白の弱領域

小さい黒丸は黒の弱領域

図 3.3-7：弱領域の例

- 眼数

領域内の眼の数。群の死活探索 (3.4.3.4章) より求まる。

- 中手

  眼数が自己手番では2眼で相手手番では1眼になる時(1.5眼と呼ぶ)，自己手番で2眼にする / 相手手番で1眼にする手。群の死活探索(3.4.3.4章)より求まる。

- 包囲度

  相手側の石による包囲の度合い。群の生存の条件の1つである脱出のしやすさを求めるのに用いる。包囲度が0の時を特に完全包囲と呼ぶ。

  包囲度は以下のように定義する。

  標的の群に属する各石のダメ点を1次の群ダメ点と呼び，n(1~3)次の群ダメ点の1路外側のダメ点をn+1次の群ダメ点と呼ぶ。ただし，その点が敵遮断点(3.3.1.6章)である場合と，4次の群ダメ点で高さが1の場合は除く。また，その点が敵石または味方の群の石に二度以上接して到達した点である場合も除く。例として図3.3-8のような石の配置における4次以下の群ダメ点を示す。

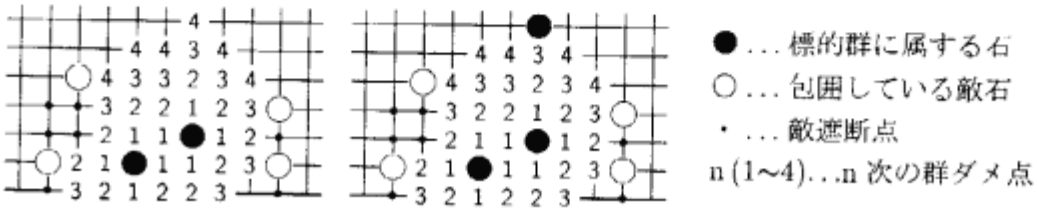


図3.3 8：n次の群ダメ点

4次の群ダメ点を単に群ダメ点と呼び，群ダメ点から包囲度を決定する。群ダメ点をナラビかコスミで互いにつながっている極大集合のグループに分ける。このグループを群ダメ点グループと呼ぶ。2個以上の要素からなる群ダメ点グループについて，その群ダメ点の数の総和を群の包囲度とする。ただし，その群ダメ点が，その点に到達するまでに一度だけ敵または味方の石に接した点である場合は，その点は0.5点だけ加算する。

図3.3-8の左図では全ての群ダメ点は1つの群ダメ点グループに属し，そのうち一番左の群ダメ点は一度白石に接しているので，包囲度は5.5となる。図3.3-8の右図では群ダメ点は2つの群ダメ点グループに属し，包囲度は4.5 (2.5 + 2.0) となる。

- 群ダメ点

  群の包囲度を計算する為の群のダメ点の集合

- 包囲結線

  群を包囲している相手側の結線の集合で，3次以下の群ダメ点に隣接する敵遮断点の属する結線とする。包囲されていても，包囲結線が弱ければ脱出可能となるので，包囲結線について調べておく必要がある。

- 包囲群

  群を包囲している相手側の近傍の群の集合。攻め合い状態などを調べるのに用いる。

- 強度

  群の強さを表すもので，以下のように定義する。

強度 = 3/4 × 先着中地 × 5 + 包囲度 × 2.5　　先着中地 < 3
強度 = 3/4 × 後着中地 × 5 + 包囲度 × 2.5　　後着中地 > 3
強度 = max(後着中地 × 5 + 包囲度 × 2.5, 15)　　後着中地 ≦ 3 ≦ 先着中地

・最低 0 ～ 最高 50 として求める。
・ここでの中地とは，[中地－２目＋弱領域] である。
・弱領域から等距離の味方石が複数ある時は，その数で等分した値を各群に加算する。
・この群の強度は，群の分類によって適当に補正する
　・攻め合い可能群，連絡可能群の強度は最低 5。
　・結線脱出可能群の強度は max(min(脱出後の強度, 15), 5) とする。
　・連絡可能群の強度は，max(連絡先の群の強度 / 2 , 15) を自群の強度に加える。

辺近傍の連絡可能群では，連絡することにより，連絡先の群と一体化するだけでなく，辺に新しく地ができる可能性が高い。そのため，連絡先の群の強度の半分を加算するという補正だけでなく，連絡によって新たにできる地を自群と連絡先群の高さと距離から見積もって，群強度の補正に利用する (詳細は略)。

係数である 5 と 2.5 の定数は以下のようにして決めた。様々の棋譜に表われたいろいろな局面について，その局面の群の強度を人間の感覚で評価した。人間の石の強さに対する感覚を数値で表現する為に，表 3.3–3を目安とした。人間の感覚から求めた強度と前述の式による強度の差が最小になる定数を求めることによって，5 と 2.5 の定数を決定した。

表 3.3 3：石の強さに対する感覚と強度との関係

石の強さに対する感覚	強度
生き	30 ～ 50
ほぼ生き	30
中立	0 ～ 30
死	0

- 緊急度

群の忙しさを表すもので，以下のように定義する。

表 3.3–4：群の緊急度関数

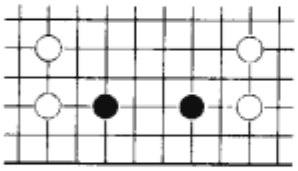
強度	10 未満	15 未満	20 未満	40 未満	40 以上
緊急度	強度 × 8	強度 × 4 + 40	190 − 強度 × 6	130 − 強度 × 3	50 − 強度

特殊弱結線 (3.3.1.6章) の緊急度は，結線の両側から敵にノゾかれている場合は 100，それ以外は 50 とした。

種石の中立連の群 (完全死群は除く) の緊急度は 100 とした。

群の緊急度は，強度から求める 0 ～ 100 までの値で，強度が 15 の時に 100 で，最も緊急度が高い状態とする。すなわち，強度によって群の手入れの緊急度を決定する。緊急度の高い群があれば，その群に対する候補手が打たれやすくなる。

本来は，群の緊急度は，群の強度から直接求めるべきものではない。群の生きるための手段の数などによって決まるものである。我々は，作業時間がなかったため，便宜的に強度から直接緊急度を求めた。



左図の黒の強度：23.0, 緊急度：61.0

先着中地：8 目　　後着中地：2 目

弱領域：1.5 目　　包囲度：6.0

図 3.3–9：群の強度と緊急度

- 種石重要度

  群の死活が周囲に与える波及効果。

  種石とは，その石の死活がその石の周囲の相手側の石の死活に強く影響しているような状態の石のことを言う。即ち，種石が死ぬことによってその周囲の弱かった相手側の石が全て安全になるので，種石の死活に関する候補手の評価値は高くする必要性がある。種石の死活に関する価値を種石重要度と呼び，以下のような関数によってその値を決定した。ここで，T_i 及び S_i はそれぞれ対象としている群の包囲群の強度とサイズである。ただし，強度とは援軍による強度の補正をする前の値のこと。

$$\text{種石重要度} = \frac{1}{15}\sum_i (30 - T_i) \cdot S_i$$

- 手数

  自己石のダメ数の総和。攻め合いの有無のチェックに使われる。

- 分類

  群がどういう状態であるか，生きるためにはどんな手段があるかを示す (3.3.2章)。

3.3.1.5　族データ

模様や厚みといった概念を認識する為に使われる。

ポテンシャル値がある値以上 (黒ならば 7 以上，白ならば −7 以下) で互いに隣接する点の極大集合で定義される族オブジェクトから構成されている。

碁世代の開発当初は，模様や厚みといった概念を認識する為によく使われていたが，群の認識の精度が上がったこと，特に弱領域という概念を取り入れたことにより，現碁世代では殆んど使われていない。

族データにはいくつかの属性がある。以下に主なものを挙げる。

- 色

  族の色 (黒 / 白)

- 領域

  族を構成する点の集合

- サイズ

  領域に属する点の数

- 穴

  族の境界のうち，結線でないところ。模様の境界を表すのに使う。

3.3.1.6　結線データ

連結する可能性のある同色の石と石，または石と盤端との関係を認識する。

石と石(または石と盤端)の距離が長ければ，連結している可能性は低い。そこで結線データは，図3.3-10のように短いパターンのみ用意している。ただし，図の結線パターンで×印のマークのついた位置は味方石でないことを条件とする。

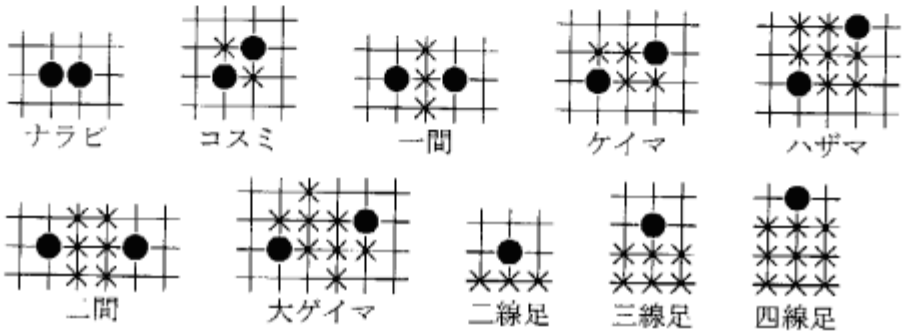


図 3.3 10：結線パターン

結線の認識過程において，距離の長いものは予めパターンで属性を与えておき，距離の短いものは探索を行なうことにより，より精度の高い認識を行なわせている。

また隅においては，表3.3 5のように複数の結線が生成されることがある。

表 3.3 5：隅の結線

X 座標 ＼ Y 座標	4	3	2	1
4	四線足×2	四線足と三線足	四線足と二線足	線足なし
3	三線足と四線足	三線足×2	三線足と二線足	線足なし
2	二線足と四線足	二線足と三線足	二線足×2	線足なし
1	線足なし	線足なし	線足なし	線足なし

1. 連結度

結線の両端点が連結する可能性の度合により，「結線の連結度」を求める。この連結度は，「群データ」作成の材料，及び結線関係の候補手に使われる。

結線の連結度は以下のように決定する。

- ナラビの場合
  常に強結線とする。
- コスミでキリチガイになっている時
  キリチガっている 2 つの敵石の連の死活より表 3.3-6 に従い連結度を決定する。

表 3.3-6：キリチガイのコスミ結線の連結度

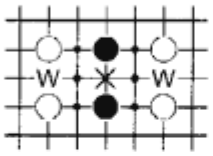
敵石の連の死活	もう一方の敵石の連の死活		
	死	中立	生
死	強結線	強結線	強結線
中立	強結線	強結線	弱結線
生	強結線	弱結線	連結不能結線

- 上記以外のコスミ，一間，ケイマの場合
  黒番 / 白番で連結探索 (3.4.3.3章) を駆動し，探索の結果より表 3.3-7に従い連結度を決定する。

表 3.3-7：コスミ，一間，ケイマ結線の連結度

結線側手番	結線の敵側手番	
	成功	失敗
成功	弱結線	強結線
失敗	連結不能結線	連結不能結線

ただし，結線側手番の探索における失敗とは探索の結果が “ツキヌケ” の時をいい，それ以外は成功とする。また，結線の敵側手番の場合は，探索の結果が “連結” の時を失敗といい，それ以外は成功とする。
ただし，図 3.3-11の一間だけはパターンで連結度を決める。



・は空点
W は白か空点であること
先着: 連結／後着: 先手切り違いとする

図 3.3-11：一間の結線パターン

- 二間，大ゲイマ，ハザマ
  結線パターン (3.3.1.7章) で，かつ切断 / 連結候補手に空点がある時，弱結線とし，切断 / 連結候補手が全て空点でない時，連結不能結線とする。それ以外は強結線とする。
- 二線足，三線足，四線足の場合
  石から盤端へ下ろした垂線上に，強結線または強結線の遮断点 (図 3.3-12) が存在すれば，連結不能結線。存在しなければ，強結線とする。

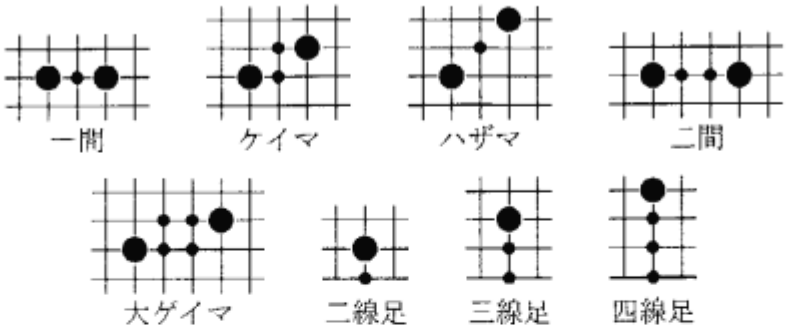


図 3.3-12 : 結線の遮断点

2. 強化済み結線

一つの結線がより短い二つの結線で結ばれている場合，その結線は強化済みとし，結線がないものと同じにする。ただし，短い方の結線が連結不能結線となる場合は，長い方の結線を強化済みとしない。

3. 両ノゾキ

図 3.3-13の一間とハザマ結線のパターンは連結済結線とせず，連結探索 (3.3.1.6章) を駆動することにより，弱結線 / 強結線 / 強化不能結線の判定を行う。

左図のパターンがみつかったら，右図を想定して連結探索を行なう。

図 3.3-13 : 両ノゾキ

判定の結果，弱結線となった時，つないでいる 2 つの結線 (一間の場合は 2 つのコスミ結線，ハザマの場合は 2 つの一間結線) がともに強結線である時は 2 つの結線とも弱結線にする。この時，ノゾキ石や手の位置など弱結線として必要な属性は全て元の一間もしくはハザマの結線の属性と同じとする。

4. 特殊弱結線

実際には連結している結線であるが，弱結線扱いにする結線を設けた。図 3.3-14で，D3-E2 のコスミ結線はツギ (E3) と取り (C2) が見合いで連結しており，本来は強結線となる。従って，C2 へ (白ツギ黒取り) 打つ手は中の白石だけの問題となり，それほど大きい評価値の手にならない。そこで，黒のコスミ結線を弱結線扱いとし，黒をツナグ／キル手の価値を C2 へトル／ツナグ手の位置に出すようにした。

- 定義
  コスミ結線で一方のノゾキ点に中立の敵連の石があり，他方が空点 (この空点を断点と呼ぶ) である時，この結線を特殊弱結線とする。
- 特殊弱結線の扱い
  特殊弱結線は弱結線と同様の扱いとする。従って，群は (他の結線でつながっていなければ) 別々となる。

- 緊急度

  特殊弱結線は，群と同様に緊急度を持つ。ただしこれは，候補手が「先手で効くので早く打とう」という目的のためだけに設けたものであって，忙しい部分という意味ではない。

  緊急度 = 100, 群サイズ = 5 とする。

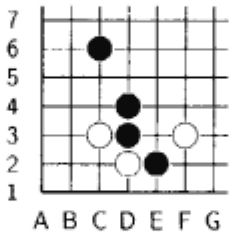


図 3.3-14 : 特殊弱結線の例

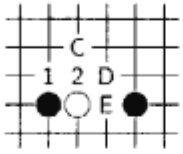
3.3.1.7 結線パターン

結線パターン知識は以下の通り。

●・・・結線の端点
○・・・結線をノゾいている敵石 (ただし, 死連の石でないこと)
×・・・黒白共通の候補手
b・・・黒側の候補手
w・・・白側の候補手
数字・・・黒白共通の条件つき候補手

[二間 a]

白石を A, A に隣接する黒石を B とする



黒番：後手切り違い
白番：先手切り違い

- 1, 2, C, D, E に白石がなく, かつ 1 の点の高さが 2 以上の場合

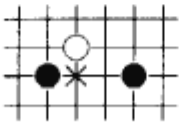
  1・・・A のダメ数≧ 3 かつ B のダメ数≦ 2 の時のみ候補手とする

  2・・・上記以外の時のみ候補手とする

- 上記以外の場合

  1, 2 とも候補手としない

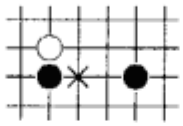
[二間 b]



黒番：連結
白番：先手切り違い

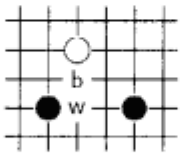
[二間 c]

白石に隣接する黒石を A とする



A のダメ数≧ 4 の場合
　黒番：連結, 白番：先手切り違い
上記以外の場合
　黒番：連結, 白番：後手切り違い

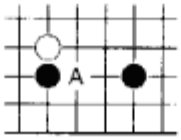
[二間 d]



黒番：連結
白番：先手切り違い

[二間 e]

左側の黒石を B とする

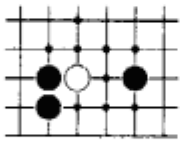


候補手の位置は先着 / 後着とも A の点とする

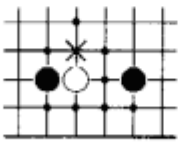
- B のダメ数が 2 以上の場合
  先着：連結, 後着：突き抜け
- B のダメ数が 3 の場合
  先着：連結, 後着：後手切り違い
- B のダメ数が 4 以上の場合
  先着：連結, 後着：連結

[二間 f]

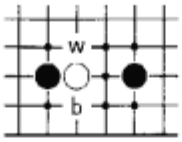
小さい黒丸は空点を意味し, ×は黒白共通の候補手, b/w は黒 / 白の候補手を意味する



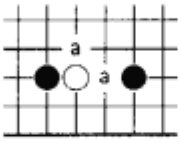
左の黒のダメ数が 3 以上であれば強結線とする



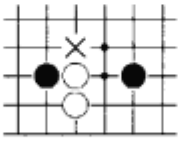
左の黒のダメ数が 3 以上であれば
先着：連結, 後着：突き抜けとする



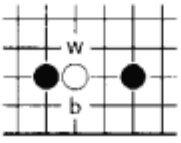
左の黒のダメ数が 3 以上の時
先着：連結, 後着：突き抜けとする



a のどちらかが白であれば連結不能とする



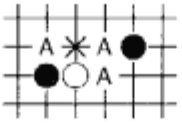
先着：先手切り違い, 後着：突き抜けとする



上記以外の二間直ノゾキパタンはの連結度は
辺近傍でない二間と同じ

[大ゲイマ a]

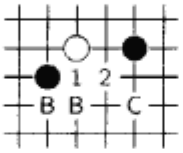
A が全て白石でない時のみ候補手とする



黒番：後手切り違い
白番：突き抜け

[大ゲイマ b]

A が白石でない時のみ候補手とする



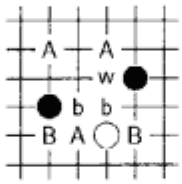
- B に黒石がない, C が黒の場合
  候補手：1
  黒番：連結, 白番：後着切り違い
- B に黒石がない, C が黒でない場合
  候補手：1
  黒番：連結, 白番：突き抜け

- B に黒石がある, C が黒でない場合

  候補手：2

  黒番：連結, 白番：突き抜け

[**大ゲイマ c**]



- B に黒石がある場合

  黒番：連結

  白番：先手切り違い

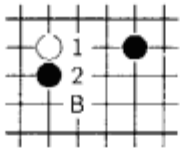
- B に黒石がない場合

  黒番：連結

  白番：突き抜け

[**大ゲイマ d**]

白石に隣接する黒石を A とする.



1 ··· B が黒石の場合のみ候補手とする
2 ··· B が黒石でない場合のみ候補手とする

- A のダメ数≧ 3 の場合

  黒番：連結

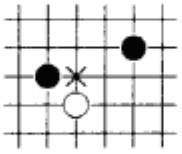
  白番：先手切り違い

- 上記以外の場合

  黒番：連結

  白番：突き抜け

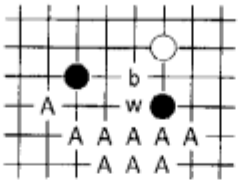
[**大ゲイマ e**]



黒番：連結
白番：先手突き抜け

[**大ゲイマ f**]

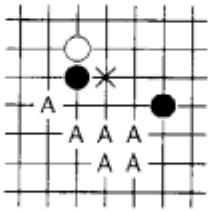
A が黒石でないこと



黒番：連結
白番：先手切り違い

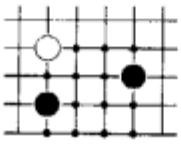
[大ゲイマ g]

A が黒石でないこと



黒番：連結
白番：先手切り違い

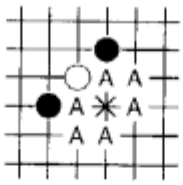
[大ゲイマ h]



強結線とする

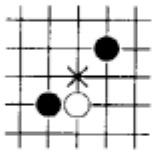
[ハザマ a]

A が全て白石でない場合のみ候補手となる



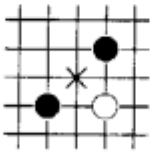
黒番：後手切り違い
白番：先手切り違い

[ハザマ b]



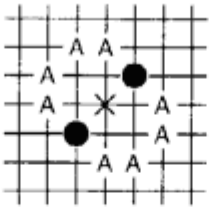
黒番：先手切り違い
白番：先手切り違い

[ハザマ c]



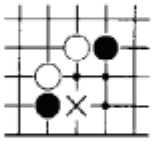
黒番：連結
白番：突き抜け

[ハザマ d]



Ａに白石がある場合
　黒番：連結，白番：先手切り違い
上記以外の場合
　黒番：連結，白番：後手切り違い

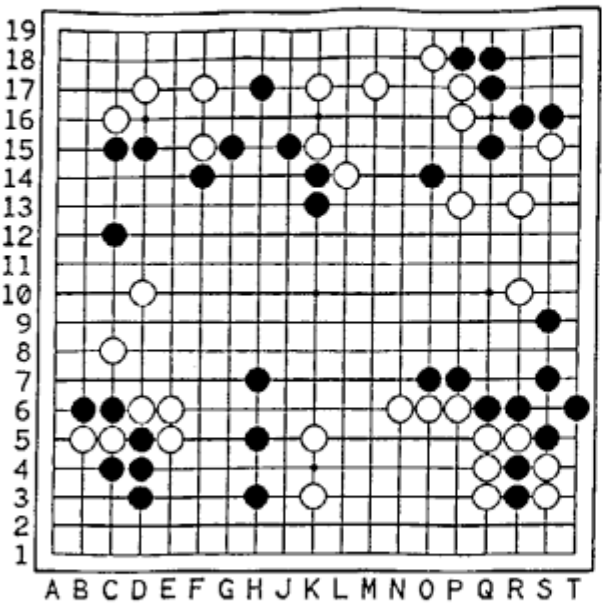
[**ハザマｅ**]



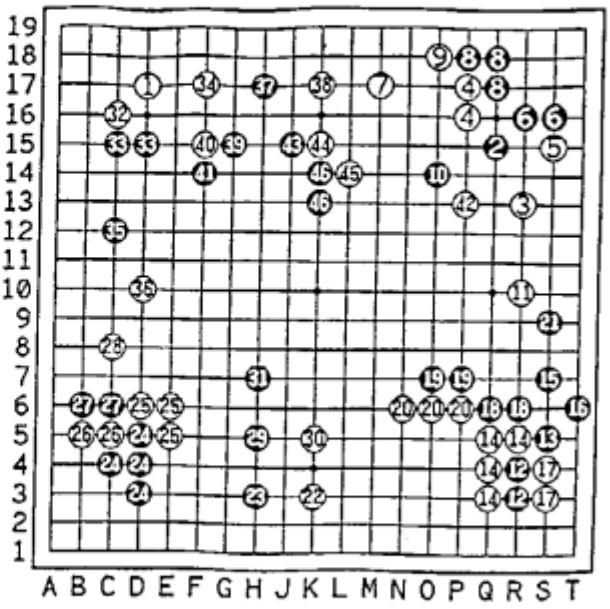
黒番：連結
白番：突き抜け

3.3.1.8　データ例

ある局面における点，連，群，結線データを示す。

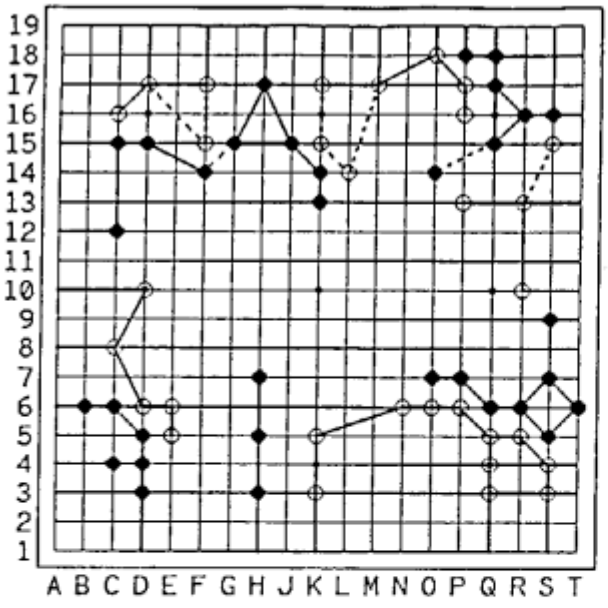


点データ



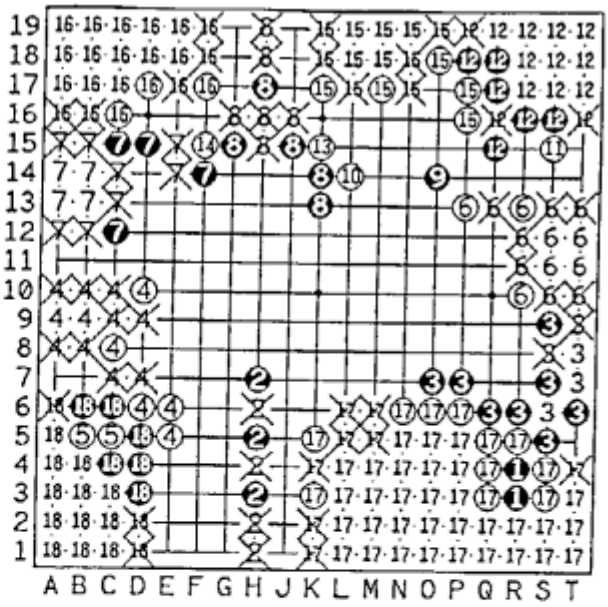
同じ番号は同じ群を表す。

連データ



実線は強結線，破線は弱結線を表す。

結線データ



同じ番号は同じ群を表す。

群データ

図 3.3-15：ある局面のデータ

3.3.1.9 課題

碁世代における課題，及び作業時間の都合により作成できなかったものについて述べる。

1. 隅で交差する線足

図 3.3–16のように隅で線足が交差した場合，交差した部分はどちらの石の群になるべきか。どちらの群であるとも言うべきでないのか。

碁世代では，この部分についての考慮が欠けている。

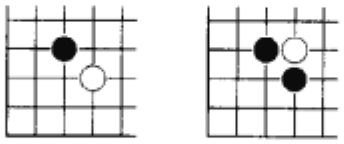


図 3.3–16：隅で交差する線足

2. 群 (先着群と後着群)

人間が見ると一つの群のように考えているが，碁世代の定義では群と認められないものがある。図 3.3–17のように，黒のコスミ結線がノゾかれているなどして弱結線になっているとき，黒は二つの群となっている。

自分の番ならばつながるが，相手の番だと切られるようなものについての認識が著しく異なっている。

これに対して，「群の定義を強結線でなく弱結線にする」や，「先着の群と後着の群と二つのデータ構造を作る」などの意見があったが，決着がつかなかった。

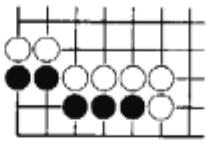


図 3.3–17：群 (先着群と後着群)

3. 模様の緊急度

模様手の価値を評価するためには，ある領域の周辺に味方が打った場合と敵が打った場合の出入りを求めるのがもっとも正確である。この評価の処理量を節約するために，どの領域の周辺に着目すべきかの指針として，領域の緊急度の概念を設けることを検討した。しかしながら，作業時間の割には効果が少なそうだとの見込みから，インプリメントには至らなかった。

模様のうちで緊急度が高いと言える条件は，以下の２つである。

- その模様を囲う／減らす手の位置がが簡単に決まる。(一義性)
- １手で荒せる，または囲える面積が大きい。(模様効率)

それらを計るために，以下を用いることが検討された。

- 一義性：
  - 領域の穴，またはノゾかれている結線の，数と長さ
    (数が少なく，短い方が一義性が高い)
  - 領域の穴の近くにいる敵石の位置，穴からの距離
    (穴の近くに敵石がいる方が一義性が高い)
- 模様効率：
  - 領域の大きさ，形
    (模様が大きく，正方形に近いほど効率がいい)

3.3.2 群の分類

群は，戦略を考える際の基準となるデータである。従って，各群が忙しいかすぐに手を入れなくてもよいか，どんな手を打つべきか，どんな手が打てるかなどがわかると，着手を生成する時の指針となる。

碁世代では，各種の属性により，群を「完全死群」「中地安全群」「群ダメ脱出可能群」「切り違い可能群」「連絡可能群」「死活中立群」「単独生き群」「単独死に群」「攻め合い勝ち群」「攻め合い負け群」「攻め合い中立群」「セキ群」「攻め合い可能群」「手無し群」に分類する。

以下に、群の分類のアルゴリズムを示す。

1．群の石がすべて死連 → 完全死群，exit
2．コスミの弱結線がない，中地が11以上 → 中地安全群，exit
3．有効な群ダメ脱出手あり → 群ダメ脱出可能
4．援軍があり連絡可能 → 連絡可能
5．援軍があり切り違い可能 → 切り違い可能
6．遠援軍があり → 連絡可能
7．包囲度が９未満，強度４５以上 → 単独生群，exit
8．包囲度が９以上，ヒラキ手あり，強度４５以上 → 単独生群，exit
9．包囲度が９未満 または 包囲度が９以上かつヒラキ手あり，強度４５未満
ならば死活探索をし，その結果から
後着１００％生き → 単独生群，exit
先着０％生き → 単独死群
それ以外 → 死活中立群
10．攻め合い状態（他の有効な生きる手がない群が隣接）で 包囲度０かつ，
先着生き確率＝０の隣接する群がある ならば攻め合い探索をし，その結果から
後着勝ち → 攻合勝ち群
先着負け → 攻合負け群
セキ → セキ群
それ以外 → 攻合中立群
11．有効な生存手が存在しない（中地安全群，群ダメ脱出可能，結線脱出可能，
連絡可能，単独生き，死活中立，攻め合い状態のいずれでもない）かつ，
包囲群の中に補正前強度が［ 被包囲群の脱出後強度＋８ ］未満の群がある
→ 攻合可能群

12．有効な結線脱出手あり，脱出後の強度が7.5以上，群の包囲群の中に補正前強度が
［ 包囲群の脱出後強度＋８ ］ 未満の群がある
→ 結線脱出可能
13．手数３０以下，強度が０，連絡以外に手がない → 完全死群，exit
14．強度１以上，手がない → 手無し群，exit

3.3.3 注意の焦点化

人間は，盤面を認識する際に，必ずしも碁盤全体を見てはいない。盤面の中から，忙しいところを瞬時に見つけ出し，その限られた部分だけに着目して，次の一手を作っている。そのように一部分だけに着目する主な目的は，考慮時間の節約である(考えたすべてのことが記憶できないという，記憶容量の問題もある)。

人間は石の集団を認識するとき，人間は一回のパスによって盤面を認識しているのではなく，何回かのパスを通して次第に原始レベル(各石やそのつながり具合など)からより複雑な高次のデータ(近接する一塊の石や模様など)へと順に認識していく。そして，同時に盤面のより重要な部分に焦点を当てて認識作業を絞り込んでいく。しかしながら，対局中に盤面を認識する場合はこのような方式ではなく，前回の着手時の盤面認識結果を記憶しておいて，着手によって変化した部分だけを認識し直して，記憶された部分と合わせて，認識作業を終える。

> 人間は，盤面のどの部分に着手をしているか調べたところ，ほとんどの着手は，相手着手の近傍に打たれていることがわかった。また，一手打つことによって状態が変化する群は，ほとんどが着手の近傍だけである(シチョウアタリのような例外もある)。人間は，前回の着手までに最も忙しいところと，今相手に着手された部分の近傍のみに絞って，次の一手を考えている。

そこで碁世代では，局面認識作業は，着手の近傍だけに絞って行なうことにした。そして，アタリまたは石が盤面から取り除かれた時だけは，群が死ぬことにより局面全体が大きく変化すると考えられるので，その時だけ盤面全体の認識作業を行なう。それ以外の場合は，前に認識した結果を保持しておき(群に関する候補手も記憶しておく)，それを利用して着手の生成を行なう。

着手の近傍とは，着手によって生じた群とその群の隣接群，及び着手位置の大ゲイマ範囲に存在する群とした。

また，考慮時間の節約のため，対局中は相手着手の後だけ，局面の認識作業を行ない，碁世代の着手後には何も作業をしない。

[今後の改良ポイント]

人間は，着目する場所を絞りこんだ後，その部分に対してだけ熟考している。それに対し碁世代は，盤面全体の認識作業時も，着手の近傍だけの認識作業をする時も，全く同じ認識方法を用いている。本来は絞り込んだ部分に対しては，より精密な認識作業を行なうのが良いのだが，マシンの性能から，これ以上多くの処理を取り入れることはできなかった。

しかしながら，もし処理時間があれば以下のような処理を取り入れたかった。

- 詰碁のように，精度の高い認識作業
- 候補手が本当に目的を達成しているか，候補手によって不利益が生じるかの確認
- 先手で打つことのできる部分への着手した場合の群の状態の認識

3.4 局所探索

3.4.1 局所探索の意義・活用方法

局面認識において各種対象の状態を認識する方法として，パターン的な知識を利用することと探索によって求めることが考えられる。一般に前者はコストは小さいが正確さに欠け，反対に後者は正確であるがコストが大きい。碁世代では，通常はパターン的な知識を使って処理し，一定時間内処理できる程問題が煮詰まってくれば探索によって求める方法をとっている。

例えば，ある結線の連結度を調べる際に，ナラビ・コスミ・一間・ケイマ結線については探索を使っているが，それよりも二点間の距離が大きい二間・大ゲイマ・ハザマ結線についてはパターンで処理している。

碁世代では，表 3.4-1のような 8 種類の局所探索が用意されている。

表 3.4-1 : 局所探索の種類

種類	標的	問題
シチョウ	ダメ数 2 以下の連	標的連がシチョウか
捕獲	ダメ数 3 以下の連	標的連が捕獲可能か
連結	短い結線	標的結線が連結可能か
死活	中小群	標的群の生死
眼数	サイズ 4 以下の地	標的地に 1 眼できるか
攻め合い	攻め合い状態の群	標的群の生死
多重標的連捕獲	ダメ数 2 以下の連	いずれかの連が捕獲可能か
詰碁	完全包囲の群	標的群の生死

人間は探索結果及び道筋を覚えておき，次の着手以降に再び同じ部分を探索し直さないようにしている。また，探索の結果失敗した道筋を組み合わせて新たな作戦を立てるためにも，探索結果及び道筋を記憶している。しかし，碁世代では探索結果及び道筋を数手に渡って記憶はしていない。毎回探索し直した方が，(1) 正確な判断ができるだろうと思われること，(2) 作業時間がなかったことにより，このインプリメントは行なわれなかった。

3.4.2 探索アルゴリズム

本システムで用意されている全ての局所探索は，完全に評価できる局面まで探索を行う完全探索 (読み切り探索) であり，探索のアルゴリズムはアルファベータ法を使っている。

また，手順前後が頻繁に生じる探索では，一度探索した局面と結果を記憶しておき，同じ局面が発生した時にその結果を使うことで無駄な探索をしないようにしている。

3.4.2.1 探索における局面記憶の効果

探索における局面記憶は，探索の途中で生じた局面と評価値を記憶しておき，同じ局面が再び現れた時に記憶しておいた評価値を使うことによって，無駄な探索を減らす為に行われる。

碁世代のいろいろな局所探索に局面記憶を導入してみたところ，詰碁や眼数などの探索では，探索の手数を大幅に削減できた。また，シチョウ，捕獲，連結といった探索では性能に大きな変化が見られなかった為，現在局面記憶は行っていない。

以下では，詰碁探索を例にして，探索における局面記憶の方法について述べる。

- ハッシュテーブル

  局面の石の配置をキーとし，その局面の繰り上がり評価値をデータとするハッシュテーブル上に局面を記憶する。
  手番が攻め番 / 守り番のそれぞれの場合について，別のテーブル (ハッシュインデクスは，現在各 5,000 エントリー用意している) に記憶する。

- キー

  盤面の石の配置 (盤上の格子点の黒 / 白 / 空の状態を該当する 2bit 上に表現) とコウの情報からなるストリングデータをキーとする。

- ハッシュ関数

  標的とする群の近傍にある数点の石の配置を数値化した値を，ハッシュインデクスエントリー数で割った剰余をとる。

また，記憶する局面として，評価値が繰り上がってきた時だけでなく，子ノードに進む時の局面も記憶しておくことによって以下のような効果が得られる。

碁では，劫以外に何手も掛けて元の局面に戻ること (長生や循環劫など) が稀に起こる。局面記憶を採用する以前の探索では，このような同形再現を認識することは難しかった。この為，一定の深さを越えても終端局面に達しない場合に，そのような同形再現が生じたと認識していた。子ノードへ進む時の局面も記憶することによって，そのような同形再現を即座に認識できるようになる。探索の深さによる間接的な認識と比較して，より正確でかつ無駄な探索を減らす効果が得られる。

3.4.2.2 キラーヒューリスティック

キラーヒューリスティックとは「現局面に似た局面で以前調べたことがある場合，その局面における最良手 (キラー手と呼ぶ) は，現局面において最良手になりやすい」という考え方であり，既にチェスなどのプログラムで，その効果が認められている。

碁世代では，詰碁探索を例にキラーヒューリスティックの効果を測定したが，特に効果が認められなかったので，キラーヒューリスティックは使っていない。

以下に，詰碁を例にしたキラーヒューリスティックの実験について述べる。

チェスプログラムでは，様々なキラー手が提案されているが，碁世代で実験したキラー手は以下の 2 通りである。

- 直前に探索された兄弟ノードにおける最良手

- 直前に探索された子ノードにおける最良手

前者は，チェスプログラムなどでも採用されている代表的なキラー手である。チェスを例に説明してみよう。相手側がある手を打った局面において，例えば相手側のビショップをただ取りにする手を発見し，その手によって相手側の手を反駁した場合，相手側が元の局面で他の手を打った局面においても，なおビショップをただ取りできる手が残っているなら，まずその手を探索するという考え方である。

この考え方が囲碁においても有効であるかどうかを予測するのは難しい。「2 の一」のように位置だけに依存するような急所が残されているような局面についてはある程度有効かもしれないが，多くの場合，碁の急所は相手の打着によってその都度変化すると考えられるからである。

後者のキラー手は，チェスプログラムなどの例にはない。多分初めての試みだと思われる。この手は，ある手を打ってみてダメだったら，その手を反駁した相手側の手の位置に打ってみるというもので，人間が通常使っているヒューリスティックである。この考え方は「敵の急所は我が急所」という囲碁の格言にも通じるものがある。

3.4.3 各種の局所探索ルーチン

3.4.3.1 シチョウ探索

シチョウ探索は，アタリの連続によって標的とする連が捕獲可能かどうかを探索するもので，以下の用途に使われる。

- 捕獲などの探索における終端局面判定や着手の生成の為に
- 各種パターン知識の条件として

1. 標的対象

   - 攻め番でダメ数2以下の連
   - 守り番でダメ数1以下の連

2. 終端局面

   局面が以下のいずれかの条件を満たす時，終端局面と判定し，その終端局面の評価値を返す。手数の制限は三劫などの循環劫対策である。

   - 守り番で標的連のダメ数が2以上の時 → 生き
   - 攻め番で標的連のダメ数が1の時 → 死に
   - 攻め番で標的連のダメ数が3以上の時 → 生き
   - 探索の深さが100手を越えた時 → 不明
   - 探索の総手数が300手を越えた時 → 不明

   なお，「不明」は「生き」とみなす。

3. 着手生成

   以下の着手について，評価値の大きい順に探索する。評価値はその連の捕獲しやすさを考慮して決められている。

   - 守り番の時
     - 標的連の包囲連をヌク手がある時その全ての手
     - 標的連のダメ点
   - 攻め番の時
     - 標的連のダメ点

   また，評価値が非常に大きいと考えられる手は強制手とし，その局面では強制手以外は読まない。

   - 強制手
     - 攻め番で，守り側の打着によって標的連のダメ数が4以上になる手がある時，その手

3.4.3.2 捕獲探索

捕獲探索は，標的とする連が捕獲可能かどうかを探索するもので，以下の用途に使われる。

- 連の死活判定の為
- 連結などの探索における終端局面判定や着手の生成の為
- 各種パターン知識の条件として
- 最終着手決定時のフィルタとして

1. 標的対象

   - ダメ数 3 以下の連

2. 終端局面

   局面が以下のいずれかの条件を満たす時，終端局面と判定し，その終端局面の評価値を返す。

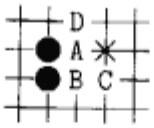
   - 守り番で標的連のダメ数が 4 以上の時 → 生き
   - 攻め番で標的連のダメ数が 1 の時 → 死に
   - 攻め番で標的連のダメ数が 2 で，シチョウ探索によって標的連が捕獲可能の時 → 生き
   - 攻め番で標的連のダメ数が 4 以上の時 → 生き
   - 探索の深さが 20 手を越えた時 → 不明
   - 探索の総手数が 200 手を越えた時 → 不明

   なお，「不明」は「生き」とみなす。

3. 着手生成

   以下の着手について，評価値の大きい順に探索する。評価値はその連の捕獲しやすさを考慮して決められている。

   - 守り番の時
     - ダメ点
       標的連のダメ点 (ただし，その点への守り側打着によって標的連のダメ数が 1 の時は除く)。
       一間点
       標的連のダメ数が 2 で，標的連 (黒石) に対して図 3.4–1 のような一間 (×) の手がある時，その手
     - コスミ点
       標的連に対してコスミの手
     - 包囲連ヌク手
       アタリの包囲連をヌク手
     - アタリでキリを入れる手
       包囲連に対して，アタリでキリを入れる手がある時，その手 (ただし，その点への守り側の打着によってできる連のダメ数が 1 の時は除く)



A.×は空点でBは空点か白でCかDが黒であること

図 3.4-1：標的連に対する一間点

- 包囲連シチョウで捕る手
  高さが2で，石数が1で，ダメ数が2の包囲連でシチョウ探索で捕獲可能の時，その包囲連をシチョウで捕る手
- パスの手
  以上の処理で生成された全ての着手の最大着手評価値が負の時，パスの手

- 攻め番の時
  - ダメ点
  - 一間点
  - コスミ点
  - 包囲連ツグ手
    高さ2で，石数1で，アタリの包囲連のダメ点の高さが1の時，そのアタリの包囲連をツグ手
  - 包囲連シチョウから逃げる手
    アタリの包囲連をツグ手がシチョウ探索で捕られない時，そのアタリの包囲連をツグ手
  - 元ツギの手
    標的連のダメをツメる手が，打ってもアタリになる時，打ってもアタリにならないようにする手

また，評価値が非常に大きいと考えられる手は強制手とし，その局面では強制手以外は読まない。

- 強制手
  - 守り番の時
    * 標的連のダメ数が1の時，そのダメ点もしくはアタリの包囲連をヌク手
    * 守り側の打着によって標的連のダメ数が4以上になる手がある時，その手
    * サイズ2以上でアタリの包囲連がある時その包囲連をヌク手
    * 2つ以上の包囲連に対して両アタリする手がある時，その手
    * サイズ2以上でダメ数2の包囲連をシチョウ探索で捕獲可能の時，その捕獲手
  - 攻め番の時
    * 守り側の打着によって標的連のダメ数が5以上になる手がある時，その手

4. 課題

- 1級から10級程度の捕獲問題を数多く実行させて，誤った回答の原因を探った。以下に誤った答を出した原因を示す。

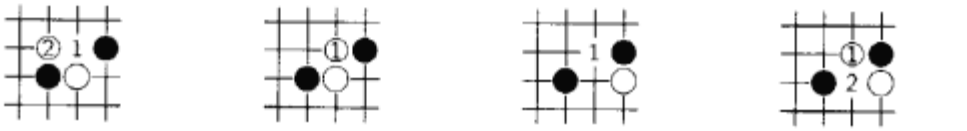
- 劫の手を単に打着禁止手として処理している為, 劫にせず捕獲できる問題であっても, 劫になる手の方を正解とする場合がある
- ダメ数3以上の隣接敵連に対する攻め側の手が生成されない
- 標的連とコスミでつながっている石や, タケフでつながっている石のダメをツメるような攻め側の手が生成されない

誤答を出した原因を取り除くような改良を加えることで正解率は高くなるが, 反対にそのような改良によって全体の探索時間が増える。改良にあたっては充分注意が必要である。

- 碁世代の棋力の目標をアマチュア中級と設定した時, どの程度の問題が解ければよいかについて実験した。その結果, ダメ3の連の捕獲がだいたい正しくできれば良いことがわかった。また標的をダメ4の連の捕獲に拡張して実験したところ, 処理時間の割には棋力が上がらなかった。

3.4.3.3 連結探索

連結探索は, 標的とする結線が連結可能かどうか探索するもので, 連結可能でない場合には, 「切り違い」か「突き抜け」かを求める。また, 「切り違い」の場合は, 先に切りの手が打てる場合を「先着切り違い」, 切りの手が後手の場合を「後着切り違い」とした (図 3.4-2)。



黒番：後着切り違い　白番：突き抜け　黒番：連結　白番：先着切り違い

図 3.4-2：切り違いの分類

1. 標的対象

   - 線足以外の全ての結線

2. 終端局面

   局面が以下のいずれかの条件を満たす時, 終端局面と判定し, その終端局面の評価値を返す。

   - 攻め番で端点が一つの連結連になった時 → 連結
   - 両端点の連結連の距離 (互いの連結連の一番近い2点の距離) が $\sqrt{5}$ (ケイマ) で, 結線と垂直に敵石が2つ並んだ時 → 突き抜け
   - 両端点の連結連の距離が $\sqrt{4}$ (一間) で, 結線と垂直に敵石が3つ以上並んだ時 → 突き抜け
   - 両端点の連結連の距離が $\sqrt{2}$ でキリチガイパタンの時
     - 攻め番
       (a) どちらかの端点の連が敵手番で捕獲可能 → 突き抜け

(b) それ以外で，両方のキリ石が敵手番で捕獲可能，またはどちらかのキリ石が味方手番で脱出不能 → 連結

(c) 上記以外 → 切り違い

– 守り番

(a) どちらかの端点の連が味方手番で脱出不能 → 突き抜け

(b) それ以外で，どちらかののキリ石が敵手番で捕獲可能 → 連結

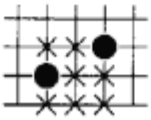
(c) 上記以外 → 切り違い

- 探索の深さが 20 手を越えた時 → 不明
- 探索の総手数が 300 手を越えた時 → 不明

3. 着手生成

強制手がない時，以下の着手について結線の両端点の中心に近いものから順に候補手を生成する。

- 端点を斜辺とする四角形内 (一間，二間の時はその左右一路まで) のすべての点
- 上記の範囲内にアタリの敵連がある時，そのダメ点
- ケイマ結線で，端点の高さが 2 と 3 の時は以下の図における×の点



4. 強制手

両端点の連結連の距離が一間，ケイマ，キリチガイの時，図 3.4–3に示す・点が空点ならその点を強制手とする。

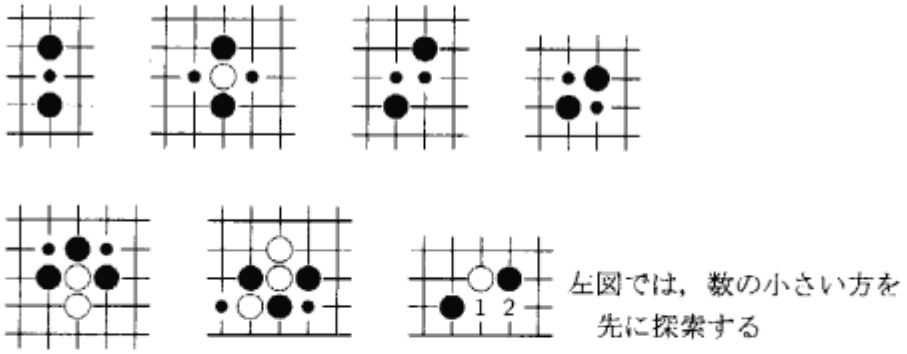


図 3.4–3 : 強制手

5. 課題

現基世代では短い結線についてのみ，連結探索を起動している。今後はより長い結線についても連結探索を起動させた方がよい。ただし，現在の結線探索は終端局面から連の捕獲探索を呼んでいるため (切り違った連が捕獲できれば突き抜けとなる)，処理時間が多い。大きい結線の連結度を探索によって処理する場合は，強制手や終端局面のヒューリスティックなどを取り入れて処理時間を減らす努力が必要である。

3.4.3.4 死活探索

死活探索は，標的となる群の生存確率を求めるものである。生存確率は，各群について先着の場合と後着の場合の2通り算出する。

死活探索モジュールを作成するにあたり，必要な処理とそれに要する時間について概算した。必要な処理としては，フトコロ手，眼形手，包囲連の捕獲手，包囲網を破る手などの候補手の作成と，終端条件としての中手知識を利用した生き／死にのパターンとの照合処理，候補手作成や終端条件のための連の生き／死に，結線の連結／切りの判定とフトコロの大きさの更新が考えられた。それらの処理を導入したときの1手分の探索に必要な処理時間は，碁世代の全体の処理時間の約半分くらいと見積もられた。従って，探索の候補手を1局面に2手くらい，深さを5手くらいに制限したとしても，死活探索の導入によって対局時間が数十倍に増えることが明らかになった。

これに対する打開策として「強制手を増やして候補手を1手に絞り，ほとんど一直線の読みにすればせいぜい数倍程度の対局時間の伸びに抑えられるのではないか」という提案がなされ，それならば「一直線の読みで作られる最終局面の図であれば，1手ずつ局面データを更新して作成しなくとも似たような図はできるのではないか，それなら処理時間はそれ程要らない」ということになり，その方針で作ることが決まった。

即ち，「死活判定ルーチン」は「死活ケース」の仕様を一直線の読みにした時に得られるであろう最終局面の図を1手ずつ更新せずに近似的にいっきに作成するようにしたものである。

1. 標的対象

   - 「群の石がすべて死連」，「またはコスミの弱結線がなく中地が11以上の群」以外の群

2. アルゴリズム

   領域がまだ固まっておらず，領域の大きさが増減する余地のある場合は，フトコロ候補手を利用して領域が固まるまでの想定図を作り，その想定領域の大きさにより死活を予測する。領域がある程度固まっている場合には眼形知識や一眼検出探索を利用して眼数を求め死活を予測する。ただし，包囲している敵連の中に中立の連がある時などはそのことも考慮する。

   群の生存確率の求め方は以下の手順で行なう。

   (a) フトコロ，辺点，外フトコロの各候補手を使って，先着／後着の場合に想定される領域の形を求め，その領域内の空点の内点をカウントすることにより，先着／後着中地を算出する。

   (b) 先着と後着の中地から，それぞれ生きると思われる確率を求める。

      i. 中立の包囲連がある場合

         - 先着生き確率

           後着中地が1目以上の時100%とし，以外は50%とする。

         - 後着生き確率

先着中地による生き確率とする。ここで中地による生き確率とは以下の表による。

中地	2以下	3	4	5	6	7	8以上
生き確率	0%	17%	33%	50%	67%	83%	100%

ただし，求めた先着生き確率が後着生き確率より大きくならなければ，ii. 以降の処理を行う。

ii. 固い群の場合

群の眼数を算出し，算出した眼数により以下の結果を返す。

1. 2眼以上の時
   先着，後着　100% 生き
2. 1.5眼の時
   先着　100% 生き
   後着　0% 生き
3. 1眼の時
   先着，後着　0% 生き

ただし，ここで固い群とは以下の条件を全て満たす時をいう。

- 先着中地 — 後着中地 ≦ 1
- 境界結線が全て一間 / ケイマ / 二線足
- ツトコロ候補手の先着の手が全て1線
- 二線の味方基点の下に敵基点があり，その味方基点が2の1でも2の2でもないツトコロ候補手がない

iii. 上記以外の場合

先着 / 後着中地による生き確率を先着 / 後着の生き確率とする。
ただし，先着生き確率が0% の場合 ii. と同様に群の眼数により先着 / 後着の生き確率を求める。

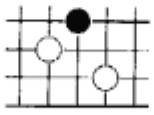
3. ツトコロ候補手

群の辺に沿ったツトコロを拡大／縮小する手である。これらの手は対峙する自群と敵群の基点の石をキーとするパターンとして与えておく。

パターンは，二間対峙以内，高さ4以下を基点とするものについて用意してある (図3.4-4)。

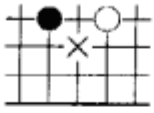
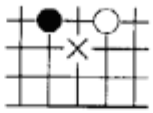
パターンは，対象とする群の基点からの敵基点の位置が重要なものから並べてあり，最初にマッチしたパターンを候補手として使う。これにより敵石が複数存在しても，対峙する石の関係に最も影響する石にだけ着目することになる。

ただし，以下の図のように基点があってもツトコロ候補手を作成しない場合もある。



(a) ツトコロ候補手の先手 / 後手

- 先着の場合


黒からの拡大候補手

白からの縮小候補手

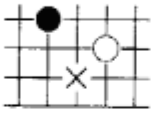
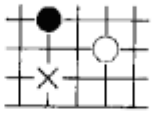
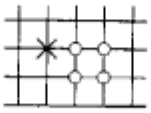

黒からの拡大候補手

白からの縮小候補手

図 3.4-4：フトコロ候補手パターンの例

- 手が 3 線の場合

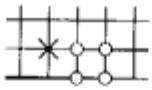
  候補手を打った時にできる連のダメ数が 3 以上で以下の点 (小さい白丸) に敵石がない時先手とし, 以外は後手とする。



  ただし、上図で × が候補手の位置で, 左側に味方石, 右側に敵石があるものとする。

- 手が 2 線の場合

  チェックする敵石の位置が以下の図のように変わるのみで, 後は 3 線の場合と同様。



- 手が 1 線の場合

  常に後手とする

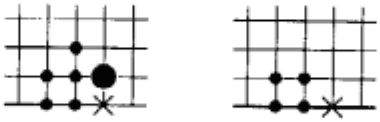
• 後着の場合

- 手が 3 線もしくは 2 線の場合

  先手の場合と同様

- 手が 1 線の場合

  その手の上に敵石があれば左下図の小さい黒丸, 敵石がなければ右下図の点に敵石がなければ先手, 以外は後手とする。

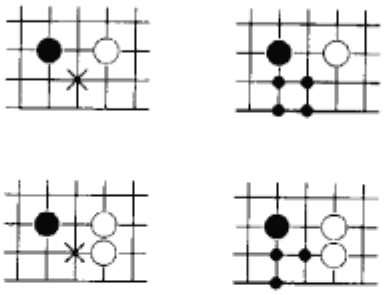


(b) フトコロ候補手の先着 / 後着想定図

• 先着の場合

  辺に沿って味方基点から候補手の点を結んだ線から下の点全てとする。
  ただし, 後手の場合は候補手の下の点は除外する。

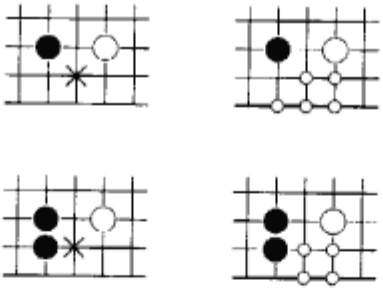
例えば以下の図で上段は先手, 下段は後手の例である。左側の図でコスム手を打った場合にできると思われる領域の想定図は右側の図の小さい黒丸になる。



- 後着の場合

辺に沿って相手基点から候補手の点を結んだ線から下の点全てとする。ただし, 先手の場合は候補手の 1 路先の以下の点も加える。

ただし, 上図で×は候補手の位置 小さい白丸は加える点である。
以下の図の例で上段は先手, 下段は後手の例である。左側の図でコスム手を打たれた場合に侵略されると思われる領域の想定図は右側の図の小さい白になる。



(c) フトコロ候補手の出入り値

先着想定図によってできると思われる領域の点の内空点で自群の点でもない点の数を Vs とし, 後着想定図によって侵略されると思われる領域の点の内空点でかつ自群の点の数を Vk とする時, Vs + Vk をフトコロ候補手の出入り値とする。

4. 辺点候補手

(a) 辺点候補手の先手 / 後手

- 先着の場合

6 間以上もしくは異色対峙で相手が 4 線の時先手とし, 以外を後手とする。つまり, ヒライた後に, なおヒラキまたはケイマにスベリの余地のある場合を先手にしている。

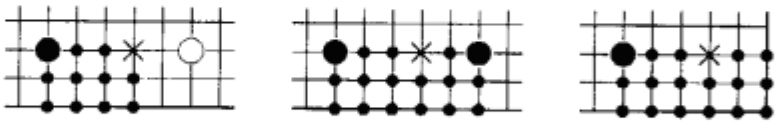
- 後着の場合

  常に後手とする

(b) 辺点候補手の先着 / 後着想定図

- 先着の場合

  5 間以内の同色対峙もしくは盤端対峙の時, 味方基点から対峙点までの 3 線以下の点全てとし, それ以外は味方基点から候補手までの 3 線以下の点全てとなる。
  つまり以下の図の例のように通常ヒラいた手までが想定される領域だが, その先に結線ができそうならそれも含めるという考えである。



- 後着の場合

  なし。
  拡大を防止する手であって, 侵略される領域は特にない。

(c) 辺点候補手の出入り値

先着想定図で求めた領域の幅の 3 倍とする。

5. 外フトコロ候補手

外フトコロ手は群の境界結線にケイマや大ゲイマの結線がある場合に生成する。

(a) 外フトコロ手の先手 / 後手

- 先着の場合

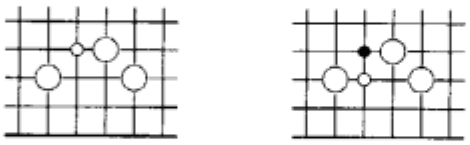
  後手とする

- 後着の場合

  先手とする

(b) 外フトコロ手の先着 / 後着想定図

ともに外側の遮断点とする。
例えば以下のようなケイマの境界結線がある時, 先着であれば左図まで広げられるが, 後着であれば右図までは入られると考える。



(c) 外フトコロ手の出入り値

1 目とする

6. 先着 / 後着の中地の算出

フトコロ, 辺点, 外フトコロの各候補手を使って, 先着 / 後着の場合に想定される領域の形を求め, その領域内の空点の内点をカウントすることにより, 先着 / 後着中地を算出する。

- 先着 / 後着の想定領域算出

  群の領域に対して，フトコロに関する手を以下の順に打った場合にでき上がる先着 / 後着の想定領域を作成する。

  (a) 先着の場合
  - 先着で先手の手を全て打つ
  - 先着で後手の最大出入り値の手を打つ
  - 後着で先手の手を全て打つ
  - 後着で後手の最大出入り値の手を打つ
  - 残った両後手の手を出入り値の大きい手から先着 / 後着の順に交互に打っていく

  (b) 後着の場合
  - 後着で先手の手を全て打つ
  - 後着で後手の最大出入り値の手を打つ
  - 先着で先手の手を全て打つ
  - 先着で後手の最大出入り値の手を打つ
  - 残った両後手の手を出入り値の大きい手から後着 / 先着の順に交互に打っていく

  ただし，辺点の手は後手でも他の先手の手に優先して打つことにする。
  また，候補手を打つことによる領域の変化は「先着の場合候補手の先着想定図の点を自群の領域にすることで，後着の場合候補手の後着想定図の点を自群の領域でなくする」ことに相当する。

- 先着 / 後着中地の算出

  先着 / 後着の想定領域の点の内，空点の内点をカウントしたものが先着 / 後着中地となる。

7. 眼数の算出

群の眼数の算出方法は以下の通り。

(a) 群の内点より眼形を求め，各眼形の眼数合計を群の眼数とする。
ここで眼形とは，群の内点の内互いに隣接する内点の極大集合をいう。

(b) 2 目以下の眼形の眼数

先着 / 後着の一眼検出探索 (3.4.3.5章) の結果により求める。

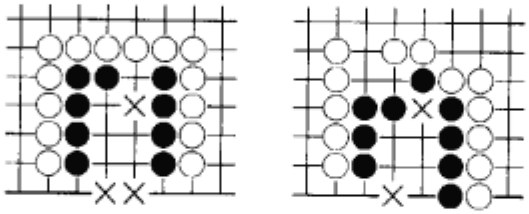
先着	後着	
	成功 (非眼)	失敗 (1 眼)
成功 (1 眼)	0.5 眼	1 眼
失敗 (非眼)	0 眼	0 眼

(c) 3 目以上の眼形の眼数

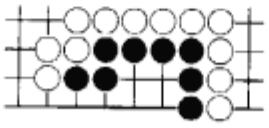
次数コードによる中手知識を利用して眼形の眼数を求める。
ただし，以下のような点がある時は各点につきその点を含めた場合と除いた場合の 2 通りの眼形を求め，その内の最低の眼数とする。即ち，以下の条件を満たす点が眼形の中に N 個あったとすると，2^N 通りの眼形についてそれぞれ眼数を求め，その内の最低の眼数を採用する。

- 1線の空の境界点に隣接する1線の点
- コスミ範囲に敵石のある空の境界点に隣接する点
- ダメ2の自群の連のダメ点でもう1つのダメ点が自群の内点でない場合

従って，以下の図で内点はそれぞれ8目と7目であるが，最悪×の点に全て黒を打った図を考えて5目中手となり1.5眼となる。



ただし，守り側に有利な(大きな)眼形において，眼数が1.5眼のときは，攻め側に有利な眼形において眼数が1.0眼でも，1.5眼と認識する。これは，以下の図のように「眼型手を打った後の形から，また眼型手が出る可能性がある」ためである。



サイズ4の眼形パターンからは1.0眼だが
サイズ5の眼形パターンからは1.5眼なので，
最終的に1.5眼とする。

8. 次数コードによる中手知識

眼形の次数コードとは，眼形内の各点の次数の組合せのことである。次数とは，その点の隣接点の内，同じ眼形内にある点の数のことである。急所とは，石の死活にとって重要な点のことで，次数によってほぼ一意的に決まる(ほとんどの場合，急所は次数の高い点になる)。

我々は，眼形が小さい場合，形が違っても性質の同じものが多数存在することから，眼形でなく次数コードによる分類方法を採用した。そこで，このサイズ，次数コード，急所，眼数を組み合わせた中手知識を用意し，これを利用して固い群の生死を判定する。

なお隅，辺では，サイズが同じでも，急所の位置や数が異なる。表3.4-2，表3.4-3，図3.4-4に，中央におけるサイズ4の中手知識を示す。

9. 課題

碁世代の死活探索はかなり精度が悪い。碁世代の棋力を下げている大きな要因の一つである。

この原因と対処案として以下のものが挙げられる(本当の原因は限られた時間内に処理することによる処理量の制限)。

- 対象の設定：群 → 弱結線でつながった味方群
- 一直線の読みによって作る領域の想定図の精度(特にフトコロ候補手)
- 眼数探索の駆動条件
- 欠け眼の可能性を考慮しない敵の中立連による生存確率の設定
- 中央にできる地を考えていない(辺に沿った地のみ考慮している)

表 3.4-2：眼形サイズと次数コード

サイズ	次数コード	急所数	眼数	備考
2 以下	−	−	1.0 眼	
3	(2,1,1)	1	表 3.4-3	
4	(2,2,1,1)	2	表 3.4-3	図 A
	(3,1,1,1)	1	表 3.4 3	図 B
	(2,2,2,2)	0	1.0 眼	図 C
5	(3,2,1,1,1)	2	表 3.4-3	図 D
	(3,2,2,2,1)	1	表 3.4-3	図 E
	(4,1,1,1,1)	1	表 3.4-3	図 F
	(2,2,2,1,1)	−	2.0 眼	図 G
6	(4,2,1,1,1,1)	2	表 3.4 3	
	(3,3,2,2,1,1)	2	表 3.4 3	
	(4,2,2,2,1,1)	1	表 3.4-3	
	(3,3,2,2,2,2)	2	表 3.4-3	
	その他	−	2.0 眼	
7	(4,3,2,2,1,1,1)	3	表 3.4-3	
	(4,2,2,2,2,1,1)	4	表 3.4-3	
	(4,2,2,2,2,2,2)	3	表 3.4-3	
	(4,3,2,2,2,2,1)	2	表 3.4-3	
	その他	−	2.0 眼	

- 援軍や脱出による利きを考えていない
- 求まった眼型を単純に加算している (ダメ詰まりなど単純に加算できない場合があるから)

特に，一番目に挙げられた，対象の設定に問題が多い。碁世代は，群を対象にしているが，人間は弱結線でつながった味方群まで範囲を広げて考えている。これにより，作成可能な群の領域の大きさが著しく異なる。

表 3.4-3：急所数と眼数

急所数	急所内の敵石数	眼数
N	N	1.0
	N - 1	1.5
	N - 2 以下	2.0

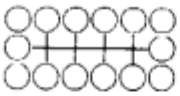
図 A  次数コード (2,2,1,1)，急所数 2

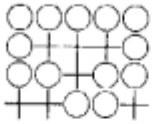
図 B  次数コード (3,1,1,1)，急所数 1

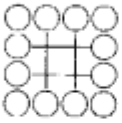
図 C  次数コード (2,2,2,2)，急所数 0

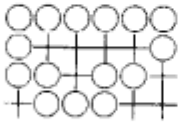
図 D  次数コード (3,2,1,1,1)，急所数 2

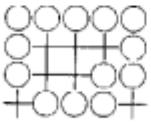
図 E  次数コード (3,2,2,2,1)，急所数 1

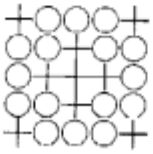
図 F  次数コード (4,1,1,1,1)，急所数 1

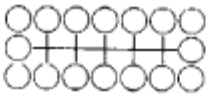
図 G  次数コード (2,2,2,1,1)

図 3.4 5：眼形と次数コード例

3.4.3.5 眼数探索

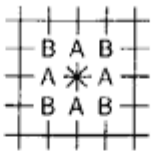
眼数探索は，標的となる地の点に完全一眼ができるかどうか探索するもので，群の死活探索から呼ばれ，眼数計算の為に使われる。

1. 標的対象

   - 隣接する群の内点

2. 終端局面

   局面が以下のいずれかの条件を満たす時，終端局面と判定し，その終端局面の評価値を返す。ただし，眼形対象領域とは，対象となる地の点とその地の点に隣接し，かつ単純な欠眼にならない全ての空点とし，完全一眼とは，「ダメ 2 以上の一つの連で囲まれた領域」または「眼形外の群の内点を共有するダメ 2 以上の連で囲まれた領域」とする。また，単純な欠け目にならない空点とは，下図の × の点で敵石が A の点には 1 つもなく，B の点にはたかだか 1 個の時とする。ただし，敵石はダメ 3 以上かダメ 2 の場合はシチョウで捕獲されないことを条件とする。



- 守り番で眼形対象領域に完全一眼ができた時 → 1 眼
- 攻め番で眼形対象領域に完全一眼ができた時 → 1 眼
- 守り番で眼形対象領域に単純な欠け目にならない空点がなくなった時 → 0 眼
- 攻め番で眼形対象領域に単純な欠け目にならない空点がなくなった時 → 0 眼
- 探索の深さが 20 手を越えた時 → 不明
- 探索の総手数が 10,000 手を越えた時 → 不明

なお，「不明」は「生き」とみなす。

3. 着手生成

以下の着手について着手評価値の大きい順に探索する。ただし，複数の着手が同じ点になる時は 1 つとし，着手評価値は各着手評価値の内最大の評価値とする。また，同じ評価値が複数ある時は，盤上の位置がより高い方の手を優先し，高さも同じ場合はより中央の手を優先する。(評価値は探索駆動時に算出しておき，探索中は評価値の再計算をしない)。

- 守り番
  - コスミ範囲の手
    眼形対象領域の点からコスミ範囲にある全ての空点
    敵石を抜く手
    眼形対象領域の点からコスミ範囲にある全てのダメ 1 の敵石をヌク手
- 攻め番
  - コスミ範囲の手
  - 外点
    眼形対象領域の点からコスミ範囲にある全ての空点に隣接する空点
  - 敵石を抜く手

3.4.3.6 攻め合い探索

攻め合い探索は，攻め合い中の群の攻め合いの優劣を判定するもので，手数差と候補手位置を求める。

1. 初期処理

(a) 型判定

問題をキリ違い型と同心円型の 2 つの型に分ける。

キリ違い型とは双方の群の石を含むキリ違いパタンが 1 つ以上存在する場合とする。同心円型はそれ以外の場合とする。

(b) 幹連と枝葉連判定

キリ違い型の場合は, 双方の群の石でキリ違いになっている石の連をそれぞれの群の幹連とする。もし複数ある場合はダメ数最小のものを幹連とする。

同心円型ではそれぞれの群の連の確定ダメ数最小のものを幹連とする。ただし, 確定ダメ数とは, 連のダメ点に虎口点がなければ連のダメ数とし, 虎口点があればその点への味方打着によってできる連のダメ数 −1 と元の連のダメ数の内大きい方とする。

幹連以外の群の連を枝葉連とする。

2. 終端局面

局面が以下のいずれかの条件を満たす時, 終端局面と判定し, その終端局面の評価値を返す。

- 眼形手がなく双方の群が 1 つの連となった時 (初期処理で求めた幹連及び枝葉連の内ヌカれていない連が全て同じ連になった時)

  評価値 = 手数差 (後述)

- 眼形手がなく手番と反対側の幹連がダメ数 3 以下でその手番で捕獲可能の時

  評価値 = 100

* 探索の深さが 30 手を越えた時

  評価値 = 0(不明)

- 探索の手数が 100 手を越えた時

  評価値 = 0(不明)

なお, 「不明」は「生き」とみなす。

3. 手数差の算出

双方の群が 1 つの連となった時, 以下のように手数差を算出する。

(a) 眼数の算出

双方の群の眼数を算出する。眼数とは眼形の数で, 眼形は群の石 (幹連) で囲まれた閉領域をいう。

(b) 双方の群の眼数が 2 以上の時

評価値 = 0

(c) 自群の眼数が 2 以上で, 敵の眼数が 1 以下の時

評価値 = 100

(d) 自群の眼数が 1 以下で, 敵の眼数が 2 以上の時

評価値 = - 100

(e) 双方の群の眼数が 1 以下の時

- 双方の手数算出

  双方の手数を算出する。手数は眼形のある群の場合以下の式で求められる。

  手数 = 外ダメ手数 + 手筋手数 + 内ダメ手数 + 眼形手数

  また, 眼形のない場合は以下の式となる。

  手数 = 外ダメ手数 + 手筋手数

- 外ダメ手数
  外ダメ手数は外ダメの数とする。ここで，外ダメとは双方の幹連のダメの内，相手のダメと共通しているダメ (内ダメと呼ぶ) と眼形に含まれるダメを取り除いた点のことである。
- 手筋手数
  以下のものがある.

  元ツギ手筋
  外ダメの中に虎口点 (その点に相手が打ってもすぐ取り返せるような点) がある場合, 各虎口点から元ツギ数を求め, 元ツギ数より以下のように元ツギ手数を求める。
  * 対象としている群 (幹連) に眼形がある時
    元ツギ手数 = 全ての虎口点の元ツギ数合計
  * 上記以外で内点がある時
    元ツギ手数 = 全ての虎口点の元ツギ数合計
    − max(最大の元ツギ数, 内点数 × 2)
  * 上記いずれでもない時
    元ツギ手数 = 全ての虎口点の元ツギ数合計
    − 最大の元ツギ数

  また, 各虎口点の元ツギ数は以下のように求める。
  * 虎口点へ敵が打った時のダメも外ダメに含まれる時
    2 つの外ダメを 1 セットで以下の元ツギ数を求める。
    虎口点に隣接する相手石に着目する (標的連と呼ぶ)。標的連のダメ数が 4 以上になるまで, 標的連に隣接する敵石を抜く手もしくは標的連のダメの内 2 つの外ダメ以外のダメ点へ打っていき, その打 った手数を元ツギ数とする。
    ただし, この処理で求めた元ツギ数は, 元ツギ手数計算の処理の中で最大元ツギ数を求める時には除外する必要がある (必ず打たねばならない元ツギ手)。
  * 上記以外の時
    1 手目を虎口点へ敵が打った時のダメの位置とし, その打った石を標的連とし, 標的連のダメ数が 3 以上になるまで, 標的連に隣接する敵石を抜く手もしくは標的連のダメの内最初の虎口点以外のダメ点へ打っていき, その打った手数を元ツギ数とする。

  − 両バネ手筋
  パス 1 では本手筋のチェックは行わずに手数差を求める。手数差が 1 手負けもしくは 1 手勝ちの時, 以下のように両バネ手筋を求める。即ち手数差が 1 手負けの時は味方群の石を基点とする両バネパタンが存在する時は 1 手勝ちに変える。手数差が 1 手勝ちの時は相手群の石を基点とする両バネパタンが存在する時は 1 手負けに変える。
  両バネパタンとは対象とする群の石が 2 線に 3 箇以上並んでいて, その両側に敵石があり, その 2 つの敵石への 1 線へのハネの手がどちらも既に味方石か先手で打てる状態をいう。ここで先手で打てるとは,
  その手への味方石を置いた仮想局面において, 相手石を味方手番で捕獲可能の状態をいう。

- 内ダメ手数
  内ダメ手数は内ダメの数とする。
- 眼形手数
  眼形手数は眼形の大きさをN, 眼形の中の敵石の数をMとする時, 以下の式で与えられる。

$$\begin{aligned} \text{眼形手数} &= N(N-3)/2+3-M \quad (N \geqq 4 \text{ の時}) \\ &= N-M \qquad\qquad\qquad\quad (N \leqq 3 \text{ の時}) \end{aligned}$$

- 手数差
  手数差は以下の式で求められる。

$$\begin{aligned} \text{手数差} &= \max(\text{自分手数} - \text{相手手数} + 1 - \text{内ダメ緩衝巾}) \\ &\qquad\qquad\qquad\qquad (\text{自分手数} \geqq \text{相手手数の時}) \\ &= \min(\text{自分手数} - \text{相手手数} + \text{内ダメ緩衝巾}) \\ &\qquad\qquad\qquad\qquad (\text{自分手数} < \text{相手手数の時}) \end{aligned}$$

  ここで, 内ダメ緩衝巾とは以下の式で求める。

$$\begin{aligned} \text{内ダメ緩衝巾} &= \max(\text{内ダメの数} - 1, 0) \\ &\qquad\qquad (\text{双方の眼数が同じ場合}) \\ &= 0 \qquad\qquad (\text{上記以外}) \end{aligned}$$

4. 着手生成

終端局面以外の時は, 以下の優先順に着手について探索する。

- 眼形手
  眼形手とは, 眼持ち手と眼作り手の総称。眼形度の大きい手を優先する。
  (a) 眼持ち手
  眼形 (群の石 (幹連) で囲まれた閉領域) 内の点を除く幹連のダメ点の内そこへ打つと眼形ができる手。
  (b) 眼作り手
  眼作り手とは双方の群の包囲内の空点について以下のように眼形度を求めた時, 眼形度が9.5以上となる点とする。
  眼形度とはその点からコスミ範囲にある9個の点について以下の貢献度を求め, 貢献度から眼形度を算出する。貢献度 P_i をその位置により以下で表わすこととする。

$$\begin{matrix} P_4 & P_3 & P_2 \\ P_5 & P_0 & P_1 \\ P_6 & P_7 & P_8 \end{matrix}$$

$$\text{眼形度} = (X_1+X_3+X_5+X_7)+\frac{1}{2}(X_2+X_4+X_6+X_8)-X_0$$

  ただし,

$$X_i = \begin{cases} P_i+1 & (P_i \neq \times, i>0) \\ P_i & (P_i \neq \times, i=0) \\ 0 & (P_i = \times) \end{cases}$$

また, 貢献度 P_i は以下のようにして求める。

$$
\begin{aligned}
P_i = \quad & \times \quad (\text{その点が空点でない}) \\
& \times \quad (\text{その点の隣接点に敵石がある}) \\
& \times \quad (\text{一線の点でその点の斜めの点に敵石がある}) \\
& \times \quad (\text{その点の斜めの点に敵石が 2 つ以上ある}) \\
& Nb_1 + \tfrac{1}{2}(Nb_2 - Nw_2)
\end{aligned}
$$

ここで, Nb_1 はその点の隣接点の内の味方石または盤外の数とし, Nb_2 はその点の斜めの点の内の味方石または盤外の数, また Nw_2 はその点の斜めの点の内の敵石の数とする。

- ダメ出入最大点

  双方の幹連のダメ点の内, 自分の幹連のダメ数 − 相手幹連のダメ数の出入値 (黒 / 白の石を置いた局面との差) が最大となる点

3.4.3.7 多重標的連捕獲探索

多重標的捕獲探索は, 複数の連を標的とし, いずれかの連が捕獲可能か探索するものである。ただし, この多重標的捕獲探索は多重の標的 / 意図を扱う探索の 1 つの簡単な実験段階であり, 対局モード中に呼ばれることはない。実験解析モードでのみ駆動できる。

この多重標的連捕獲探索は使っていないが, 複数の連を対象に, どちらかの連が捕獲できるかを調べ, それを候補手とする候補手知識もある (3.5.3.4章, 連絡／分断ケース, 多重捕獲キリ／ツナギ候補手参照)。

1. 標的対象

   任意の複数の同色連を対象とする。ただし, 実験解析モードから呼び出す時は, 操作を簡単にする為, ある 1 つの連を指定することによって, その連に隣接する全ての敵連を標的対象にするようにしている。

2. 終端局面と着手生成

   以下のように局面が終端かどうかを判定する。終端局面である場合はその終端局面の評価値を返し, 終端局面でない場合は着手を生成する。

   - 守り番でアタリ (ダメ数が 1) の標的連が複数ある時
     - 各アタリの標的連をツグ手 (アタリの標的連のダメ点とアタリの標的連に隣接するアタリの敵連のダメ点の和集合) の積集合が空集合の時 → 失敗
     - 上記以外の時
       各アタリの標的連をツグ手の積集合を着手とする。
   - 守り番でアタリの標的連が 1 つの時

     アタリの標的連に対する守り番のシチョウ探索の結果が失敗 (逃げられない) の時 → 失敗
     - 上記以外の時
       シチョウ探索で成功した手を着手とする。
   - 守り番で上記いずれでもない時 → 成功

- 攻め番でアタリの標的連が１つ以上ある場合 → 成功
- 攻め番でダメ数２の標的連が１つもない時 →失敗
- 攻め番で上記いずれでもない時
  全てのダメ数２の標的連のダメ点を着手とする。

3. 性能

図 3.4-6 の３つの多重標的捕獲探索問題について多重標的捕獲探索の性能を測定した。×印のマークのついた白石が標的となっている連を表わしている。全て黒番で標的連のいずれかが捕獲できるかどうかの問題であり，答えは全て捕獲可能である。

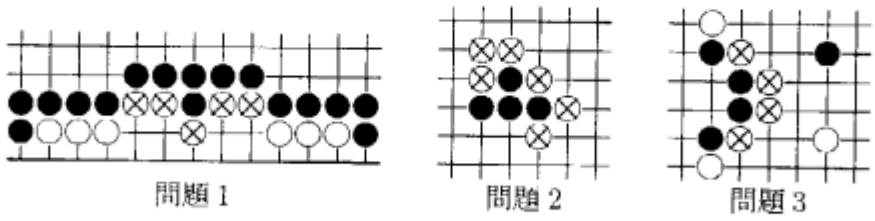


図 3.4-6： 多重標的探索の問題

問題１はアタリでキリをいれてから両アタリに持っていく問題で，問題２はアタリの連続から最後にシチョウにする問題である。問題３は問題２と同様アタリの連続から最後にシチョウにする問題であるが，問題１と２が１手打ったことによってできたダメ数２の敵連に対して次々アタリに掛けていくのに対し，離れた別々のアタリを組み合わせてシチョウに持っていく問題になっている。

3.4.3.8 詰碁探索

詰め碁探索は，標的とする完全包囲の群の死活を探索するもので，標的群に対する生 / 死 / 劫 (取り番)/ 劫 (立て番)(図 3.4-7参照) の結果及び，着手位置を求める。ただし，この詰碁探索は試作段階であり，対局モード中に呼ばれることはない。実験解析モードでのみ駆動できる。

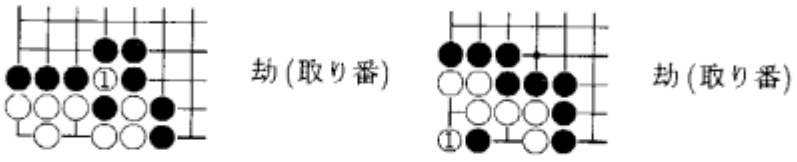


図 3.4-7： 劫の判定

1. 標的対象

- 完全包囲の群

2. 劫のための特殊処理

詰め碁では結果が劫になる問題がしばしばある。結果が劫になる問題とは，探索の過程で劫が生じ，劫の勝ち負けで生死の結果が決まる場合のことである。劫勝ちを条件として成功す

る場合より無条件に成功する方が当然望ましいので，劫勝ちを条件とする以外成功する手順のない時のみ，劫という結果を返す必要がある。この為，以下のような劫の為の特殊処理を行う。

- 劫の手の着手は最後に生成する。
- 直前の守り側の手が劫の手の時のみ，攻め側にもパスの着手を生成し，その直後の守り側の着手ではパスの手を生成しない。
- 劫を着手した場合，その局面に対する繰り上がり評価値が「成功」の時は劫にする。

なお，碁世代では，ヨセ劫や両劫の判定まではしていない。「隅の曲がり四目」は死ではなく劫と結果が返る。

3. 終端局面評価関数

局面が以下のいずれかの条件を満たす時，終端局面と判定し，その終端局面の評価値を返す。ただし，要石とは，標的群の自己石の内要石度の最も高い石の連のことであり，探索駆動時に求めておく。要石度とは以下の式で定義する。

$$\text{要石度} = \text{その石の連のダメ数} - \text{その石と標的群の重心との間の距離}$$

ここで，標的群の重心とは，標的群の自己石の座標を平均したものとする。

- 攻め番で要石のダメ数が 1 の時 → 成功
- 探索の深さが 30 手を越えた時 → 不明
- 探索の総手数が 10,000 手を越えた時 → 不明

なお，「不明」は「生き」とみなす。

4. 着手生成

システムが自動的に求めることも，人間が任意に入力することもできる。また，システムが求める場合の着手の範囲の決め方について，最適な方法はまだない。

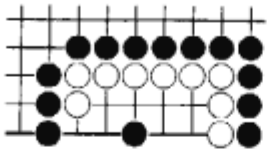
過去の実験から，最初に標的を設定した時点での，群の周囲 (例えばコスミ範囲) だけを着手の範囲とするだけでも，処理時間が大きくなりすぎることがあり，対局システムには向かない。

探索における着手の優先順位については，探索開始時に一度決めて手をソートする方法を取った。探索の枝ごとに優先順位をその都度決め直す方が探索手数は減るが，それに要する処理時間は，減らした枝分の処理時間より大きくなることが多かった。

5. 課題

- 探索範囲

  対象を群としているが，実際の対局で生じる死活問題は，近傍の群までを一まとめとして考えた方が良い場合も多い。また，一見囲ったように見える敵石が脱出可能な場合，その敵石及びその周辺は群の領域にならず，探索範囲からはずれてしまうことがある。これらは，群の認識と合わせて検討しなければならない。



1線の黒1子は脱出可能なので，その周囲は白の領域でない

- 連絡・脱出などの利き

  完全包囲の群を対象にしているとはいえ，二手連打すると援軍と連絡したり脱出ができる場合，包囲側は手抜きができなくなる(被包囲側からは利かしとなる)。このような利かしの手に対し，包囲側はどう応じれば良いか(手が抜けるかどうか，手が抜けないときはどこに打つか)について調べるには，多くの処理時間を要する。現在は利きについては何も考慮していない。

- 終端ノードの底上げ

  より早く詰碁探索を終らせるため，探索木の終端ノードの底上げについて考慮したことがある。これは，対象となる石が捕獲できるかを終端とするのではなく，二眼できる形を予めたくさん覚えておいて，それとマッチングしたら終端ノードと認識するというものである。しかしこれは，終端ノードの数の増え方が探索手数の減り方より大きいこと，ダメ詰まりなどの条件により同じ形でも生死が変わることなどの理由により，実現が難しいことがわかった。

3.4.4 探索とパターン処理の相補性

探索はあらゆる手の組合せを調べるので，処理時間がかかる。パターン処理は，照合するだけなので処理時間は短いが，周囲の状況をどの程度詳しく見るかによって精度が大幅に変わる。また，人間がパターンを列挙したり分類する設計の時間が大きい。

碁世代では，石(連)の捕獲や結線の認識以外では，なるべく探索処理を使わないようにした。それでも碁世代は，探索に関する処理が全体の約半分を占める。

しかし，パターン処理では精度が悪く，それが形勢に大きく影響すると思われる群の死活処理では，パターン処理と探索の両方を取り入れた融合型になっている。群のフトコロの大きさはパターンとの照合で想定し，個々の眼形は眼数探索によって眼ができるかどうかを調べている。

今後，飛躍的なマシンの性能アップがないならば，碁世代のように探索とパターン処理の使いわけが必要である。

3.5 候補手 (プラン) 生成

3.5.1 ケースに基づく候補手 (プラン)

人間は与えられた問題解決の場で判断の基準として，過去に遭遇した典型的状況に関する知識に基づき行動している。即ち，方針を決定する着眼点の選出は，碁の知識によって設定されたいくつかのケース (典型的状況) の観点から行なわれる。「碁世代」ではこの典型的状況についてその特徴を定義し，それにより現在どの様な典型的状況が生じているか判定出来るようにした。そして，その典型的状況が検出されたとき，これを処理する手段として候補手を用意した。碁世代では，局面認識の処理によって得られたいろいろな対象の状態に応じて表 3.5–1 のような 13 種類の候補手知識から候補手が生成される。

表 3.5–1 : 候補手知識

名称		内容
弱群手 (群の攻防)	死活	領域を増やす / 減らす手
	攻め合い	攻め合いの手
	包囲 / 脱出	群を包囲する / 逃げる手
	結線包囲 / 脱出	包囲結線を食い破る手
	連絡 / 分断	群同士の連絡 / 分断の手
定石		序盤，隅における標準的な手筋 (650 型)
辺点		ヒラキ・ツメ・割り打ち等の辺上の手
ダメ点		ハネ・ノビ・オシ等の石の競り合いの手
族の分離 / 連絡		族同士の分離 / 連絡の手
模様手		模様の拡大 / 縮小の手
結線の切断 / 連結		石をつなぐ / 切る手
打ち込み		打ち込みと防御
ヨセ		ヨセの手

また，人間は単独の目的から候補手を生成するだけでなく，複数の候補手を組み合わせて戦略を立てている。碁世代では，個々の候補手知識の整備がまだまだ稚拙なこと，処理時間が大きくなりすぎてしまうだろうこと，から組み合わせて戦略を立てることまでは行なっていない。

3.5.2 候補手評価

各候補手に対して，その手の価値を示す評価値を求める。

候補手はその効果によって 1 次要素と 2 次要素について評価する。評価値は，目数に換算して算出する。

評価基準

一次要素：領域の大きさ，勢力の出入りなど，目的の達成度から算出

(a) 実利 (弱群に対して) → 群のサイズ × 2 × 生存確率の出入り

(b) 勢力 (領域に関して) → 各点の地になりやすさ (弱領域のサイズ) の出入り

二次要素：着手後に発生する新しい味 (敵への攻め，味方の欠陥)

生存確率は全てまず強度を求め，その強度に対して以下の表 3.5–2から生存確率を求める.

表 3.5-2：強度と生存確率

強度	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
生存確率	0	0	1	1	2	2	3	4	6	8	10	14	20	28	38	50

強度	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30 以上
生存確率	62	72	80	86	90	92	94	96	97	98	98	99	99	100	100

設計段階では，評価基準には三次要素まで考えられたが，実現には至らなかった。参考までに，三次要素の内容を記す。

三次要素： 着手を実行することで失われる可能性（競合する候補手など）

3.5.3 各種のケース

3.5.3.1 定石ケース

「定石」とは，盤上の四隅における序盤の攻防の最善手の手筋である。「定石」の手が発生する状況とは，盤上のある隅が，定石手筋途中の局面 (石が全くない時も含む) であるときである。

なお碁世代では，定石は 1 つの木構造をしたファイル上に納められており，知識エディタツールの 1 つである定石エディタによって作成される。碁世代には，約 650 変化図 (定石木のリーフの数) の定石データが登録されている。

1. 定石進行中の管理

   定石とは最善手の集まりなので，定石手順中では定石以外の手を打たない方が望ましい。碁世代で「定石進行中」とは，各四隅の石の配置が定石木のリーフ以外のノードの石の配置と同じで，かつ定石スコープ内に他の石のない場合を言い，それ以外を「定石はずれ」の状態という。いったん「定石はずれ」となったらその隅については定石候補手は生成されない。

   定石スコープとは各四隅の $n \times n$ の点の範囲で，n は以下の式によって求める。

   $$n = \text{路数}/2 - 1$$

2. 候補手評価値

   定石候補手の評価値は一次要素のみを考慮している。

   各ノードには評価値に対する基礎点のデータが登録されている。また，その時の局面に応じてその基礎点を微調整する為の情報が登録されている場合もあり，その場合は調整条件文と調整点が登録されている。候補手の評価値は該当ノードの基礎点と調整点によって以下のように求められる。

   $$\text{評価値} = \text{基礎点} + \text{調整点}$$

   調整点は調整条件文が真の時だけ加算するもので，その値は正負どちらの場合もある。調整条件文は，現在局面の任意の点のポテンシャル値を指定した定数と比較する論理演算子のみが許されている (図 3.5-1)。また，ノードが定石スコープの外の石のコスミ範囲にある場合は，特に調整条件文を記述しなくとも，そのノードの評価値は無条件に 0 としている。なお，碁世代における定石候補手の最高値は 30 点に抑えている。

```
[[black(16,4),230,0],                                    % 星 (16,4) : 評価値 23.0
 [[white(17,6),200,[chkpot,[16,10,greater,30],-1]],      % 小ゲイマ カカリ *(1)
  [[black(17,8),230,0],                                  % 一間ハサミ : 23.0
   [[white(17,3),250,0],                                 % 三々 : 25.0
    [[black(16,3),290,[chkpot,[10,4,greater,30],2]],     % 抑え *(2)
     [[white(17,4),300,0],                               %
      [[black(16,5),300,0],                              %
       [[white(18,6),300,0],                             %
        [[black(15,7),300,0]]],                          %
```

(1) 評価値 20.0 だが，座標 (16,10) におけるポテンシャル値が 30 以上ならば評価値 - 1.0
(2) 評価値 29.0 だが，座標 (10,4) におけるポテンシャル値が 30 以上ならば評価値 + 2.0

図 3.5-1 : 定石データ

3. 定石候補手例

定石候補手の例を図 3.5-2に示す。

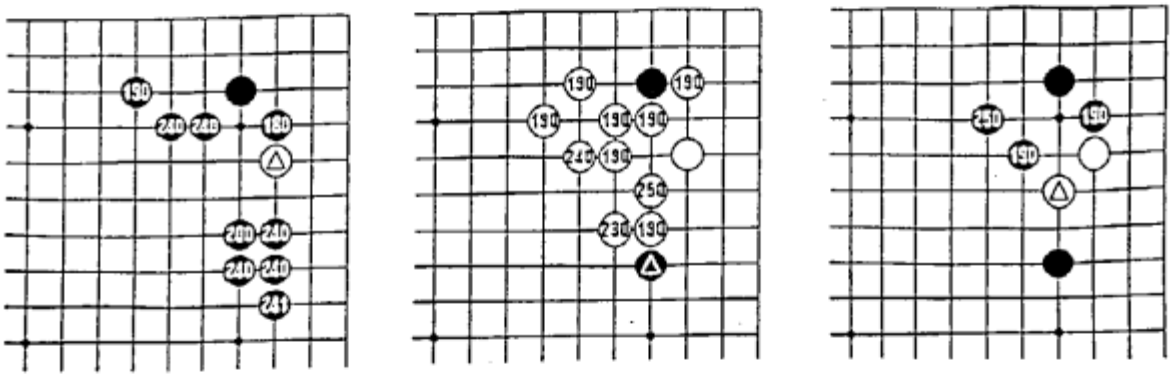


図 3.5-2 : ある定石木のスナップショット

[今後の改良ポイント]

1. 定石範囲の指定

碁世代では，どんな定石を進行中でも同じ大きさの定石スコープを使用している。そのため，人間が定石進行中と思っていても碁世代は定石終了と思ったり，定石スコープ外で起きた攻防が定石スコープ内まで広がってきても，その攻防に関する手が定石以外の手だからと打たなかったりする。

定石ごとに異なる大きさ・形状の定石スコープを使った方が良い。

2. 大型定石

碁世代では，定石は隅の攻防に関する手筋のみ登録されている。これを全局的な規模に広げ，「三連星」「秀策流」「中国流」などを定石として登録してしまう，ということも可能である。

3. 評価値の設定

定石候補手の評価値を正しく設定するのは，困難である。「置き石1つ10目」とよく言われるが，誰もそれを証明したものはいない。正しい評価値は誰もわからない。また周囲の状況によっても評価値は変化するという性質を持っている。他の候補手の評価値とバランスを保つように作成しなければならない。

3.5.3.2 辺点ケース

辺点手とは，辺上の三間以上で対峙する2石の間の点に打つ手のことである。囲碁用語で言うところの「布石のうち定石手を除いたもの」である。

1. 候補手位置

3線もしくは4線の石が，辺上でL間 ($3 \leqq L \leqq 12$) で対峙する場合，以下の3点を考慮して辺点候補手を生成する。ただし，対峙する2石の間には1つの石もあってはならない。また，対峙する石の群の強度は両方とも5以上の時のみ候補手を生成する。

- 対峙する石 (群) の水平方向 (盤端に沿う方向) の距離

  通常は対峙する石の中央を候補手位置とする。Lが偶数の場合は，対峙している2石のうち弱い方へ寄せる。ただし，四間で異色対峙している場合は石の強弱によらず，黒の候補手は黒石から二間の位置 (白石から一間の位置) ／白の候補手は白石から二間の位置 (黒石から一間の位置) とする (好形になるようにする)。

- 対峙する石 (群) の高さ

  対峙している石の高さからバランスを考えて位置を決める。

  異色対峙：　3線を候補手位置とする。
  同色対峙で，同色側の手 (守り)：
  　　3線で対峙するときのみ3線。
  　　それ以外は，4線を候補手位置とする。
  同色対峙で，異色側の手 (割打ち)：
  　　3線を候補手位置とする。

- 盤の中央からのノゾキ

  同色対峙で，対峙L間内の5線もしくは6線に敵石がある場合は，その敵石の2路下の位置に補正する。また，その敵石と対峙しているどちらかの石と辺方向に1路しか離れていない場合は，さらに1路中央よりの位置に補正する。

2. 評価値

辺点候補手の評価値は一次要素のみを考慮している。

評価値は，対峙する距離が大きいほど値が高く，最高点は定石候補手とほぼ同じ値に設定している (表 3.5–3)。ただし，同色対峙で対峙L間内の5線もしくは6線に敵石がある場合は，求めた評価値をさらに2倍する。

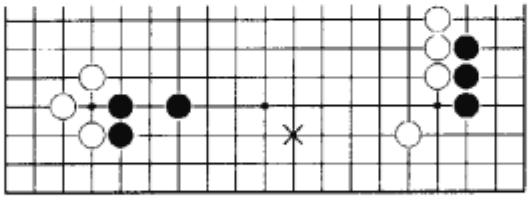
3. 辺点候補手例

辺点候補手の例を図3.5–3に示す。

[今後の改良ポイント]

表 3.5-3：辺点候補手の評価値

異色対峙										
L(距離)	3	4	5	6	7	8	9	10	11	12
評価値	18.0	18.5	19.0	20.0	20.1	20.2	21.0	21.1	21.2	21.3
同色対峙										
L(距離)	3	4	5	6	7	8	9	10	11	12
評価値	4.0	6.0	19.0	20.0	20.1	20.2	21.0	21.1	21.2	21.3



×：黒の辺点候補手

図 3.5-3：辺点の例

1. 厚みの考慮

候補手位置の算出では，対峙する石の「厚み」という概念を取り入れて，候補手位置・評価値に反映させるとよい。囲碁用語に「二立三析」というのがある。これは，「縦に2つ同色の石があるところからの辺への候補手位置は三間ビラキが良い」という意味である。そこで，適当なヒラキ幅を求めるために，ヒラキの基となる石の強さを調べるため，その周囲 (上下3～6線までの各点) についての味方石 / 相手石の有無によって，理想的なヒラキ幅を求める方法を考案した。しかしながら，この「厚み」の概念を反映させる修正は作業時間の割に，効果が上がらないとの見込みから，インプリメントには至らなかった。

2. 二次要素の考慮

「ツメ」を打った後には，さらにすべって領域を小さくするという攻めを続行できる。また，「打ち込み」の余地ができることもある。このように，辺点にも二次要素は存在する。これも設計時には考慮されたが，インプリメントには至らなかった。

3. 理想形

囲碁は四角の頂点に同じ色の石を置くと地ができやすい。そのため，「鶴翼形」という理想的な布石の配置がある。このような理想的な配置ができる場合，評価値を補正するためのボーナス点が考えられる。

3.5.3.3 ダメ点ケース

黒 / 白の石が互いに近接する場合，それぞれの石の勢力に関する攻防手として，ダメ点候補手を生成する。囲碁用語で言うハネやノビ / オシの手になる。

1. 候補手位置

図 3.5–4 のような黒 / 白の石の配置がある場合の × の位置で，(a) をナラビダメ，(b) を共有ダメと呼ぶ。

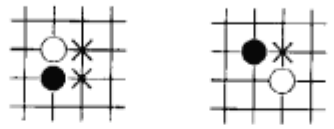


図 3.5–4 : ダメ点候補手パターン

2. 評価値

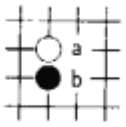
ダメ点候補手の評価値は一次要素のみを考慮している。

$$\text{評価値} = \text{ダメ点基礎点} + \text{ダメ増分度} \times \text{愚形係数} + \text{ナラビダメボーナス点}$$

「弱い石が強くなる ≒ ダメ数が増える」と仮定し，ダメが少ない味方の連のダメ数が多くなるように，相手の連のダメ数が少なくるように，手の価値が高くなる関数である。

また、味方の連のダメ数が多く、相手のダメ数が少ない場合は、ナラビダメ点が共有ダメ点より評価値が高くなる。逆に、味方の連のダメ数が少なく、相手のダメ数が多い場合は、共有ダメ点がナラビダメ点より評価値が高くなる。

- ダメ点基礎点 $v(n) = v(\ \{1,2,3,4,5,6\text{ 以上}\}\ ) = \ \{4,2,1.75,1.5,1.25,1\}$
- ダメ増分度 $\delta = \min(\text{ダメ点打着後のダメ数} - \text{打着前のダメ数}, 2)$
  ここでのダメ数とは，ナラビダメ：敵石のダメ数，共有ダメ：自石のダメ数のこと
- 愚形係数 $g(\ \{\text{ダメ点手が愚形 (アキ三角，団子) になるとき, それ以外}\}\ ) = \ \{0,1\}$
- ナラビダメボーナス点 = 4 または 1
  - 位置

    ナラビダメ点は図 (黒番) の a(ハネ)，b(ノビ) の 2 つの候補手を生成する。

    黒石の連のダメ数を Df，隣接する敵石の連のダメ数を De，黒が a に打った時にできる連のダメ数を Da，白が b に打った時にできる連のダメ数を Db とし，
    * min(Df,Da) ≥ min(De,Db) ならば a の点にナラビダメボーナス点をつける
    * min(Df,Da) < min(De,Db) ならば b の点にナラビダメボーナス点をつける
  - 条件

    対象としている石の高さが 2 線以上で，かつ以下のいずれかのとき 4 点。
    * Df ≧ 2 かつ Db ≦ 2 の場合
    * (Df = De = 3) かつ (Da = Db = 3 または 4) かつ 隣接する黒／白の石数がともに (Da-1) の場合

    それ以外は 1 点。

3.5.3.4 弱群ケース

群に対する候補手である。候補手は5種類用意されている。

1. 包囲／脱出ケース

ある程度囲まれていて(中地安全群でない)，まだ完全には包囲されていない(群ダメグループの群ダメ点の数が2以上)群は，脱出が可能であると判断して，包囲／脱出候補手を生成する。

(a) 候補手位置

包囲候補手は2種類のケース(一手封鎖型，二手以上の封鎖型)に分けて生成する。一手で封鎖できる場合は，多少弱点があってもとにかく封鎖するが，一手で封鎖ができない場合は，確実に群を追いやる(包囲側に弱点ができない)ような位置に包囲候補手を生成する。

以下，包囲候補手位置生成のアルゴリズムを示す。

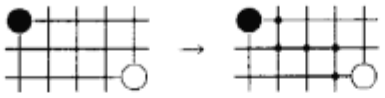
i. 二間ビラキの余地がある群の場合，二間ビラキを包囲候補手とする。

ii. 上記以外の時，群の4次ダメ点を，群ダメ点グループに分ける。
そのグループと該当群の間に，結線包囲候補手があるときは，群ダメ包囲候補手は生成しない。(包囲結線があるかないかで調べる)

iii. 群ダメ点グループ内の両方の端点を選ぶ。

iv. 包囲連の中でこの端点のそれぞれから最も近い完全死群でない石(直線距離)を求める。この石を包囲の仮基点と呼ぶ。

v. 包囲の仮基点と脱出基点(脱出候補手参照)とを結ぶ直線を引き，その直線と盤上の線の交点から距離1未満(下図の・参照)にある完全死群でない包囲石のうち，最も被包囲群に近い包囲連の中の石を求め，この点を基点と呼ぶ。

(例1)  (例2) 

vi. この基点をキーにして，群ダメ包囲候補手パターンから，包囲候補手を求める。

候補手があれば → exit
候補手がなければ → 以下へ続く。

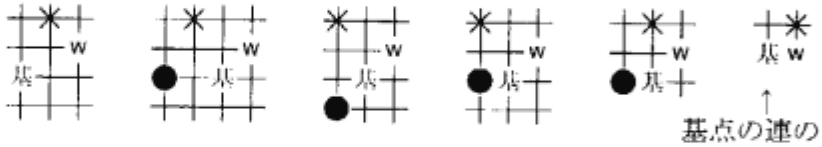
vii. 群ダメ点の数が5以下のとき，
基点からケイマ，一間，コスミ，ナラビの点のうち，2線以上でもっとも群ダメ点の重心に近い点から順に並べ，それぞれにつき以下のチェックを行う。チェックにパスした点がみつかった時点でその位置を，包囲候補手の点とする。
ただし，候補手を生成した包囲連のダメ数が2以下であるとき，候補手の位置は敵石に隣接して，ナラビの点とする。チェックをパスした候補手が愚形を作るときは，次の順位の候補手についてチェックを行っていき，どの手もパスしなかったら愚形となる手を候補手とする。

A. 候補手が包囲連から一間のとき，一間の中間の点または一間の点が敵石に隣接していない。

B. 候補手が包囲連からケイマで，基点のダメ数が4以上の時，ケイマの点が敵石に隣接していない，またはケイマの遮断点に敵石がいない。

C. 候補手が包囲連からケイマで，基点のダメ数が3以下の時，ケイマの点が敵石に隣接していない，またはケイマの遮断点に敵石がいない，またはケイマの遮断点に敵石が隣接していない。

D. ただし，候補手の点(×)と基点(基)と味方石(●)が以下のパターンの時は，wの位置に敵石があっても(図では右方に被包囲群があるとする)，その点を候補手とする。



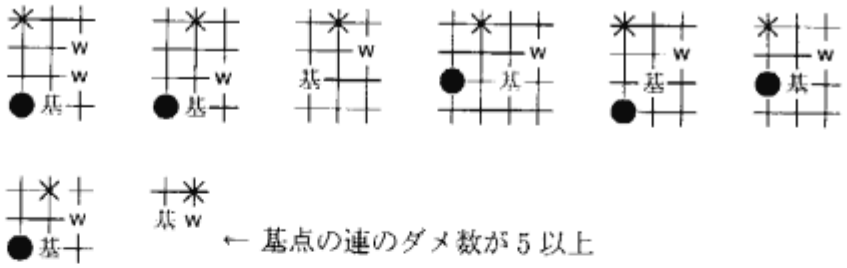
viii. 群ダメ点の数が6以上のとき，

基点から大ゲイマ，二間，ケイマ，一間，コスミ，ナラビの点のうち，2線以上でもっとも群ダメ点の重心に近い点から順に並べ，それぞれにつき以下のチェックを行う。チェックにパスした点がみつかった時点でその位置を，包囲候補手の点とする。

ただし，候補手を生成した包囲連のダメ数が2以下であるとき，候補手の位置は敵石に隣接して，ナラビの点とする。チェックをパスした候補手が愚形を作るときは，次の順位の候補手についてチェックを行っていき，どの手もパスしなかったら愚形となる手を候補手とする。

A. ナラビ，コスミ以外の候補手が，候補手と基点によってできる結線の遮断点または候補手の点が敵石に隣接していない。

B. ただし，候補手の点(×)と基点(基)と味方石(●)が以下のパターンの時は，wの位置に敵石があっても(図では右方に被包囲群があるとする)，その点を候補手とする。



ix. ただし，包囲手が包囲の基点ごとに2つ作られる時は，基点の所属する群の強度が弱い方の基点からの候補手のみとする。群の強度が等しい時は両方の手を生成する。

- 群ダメ包囲候補手パターン

群ダメ包囲候補手パターンを図3.5–5に示す。

それぞれのパターン図では，黒をつなぐ仮想の線の下方に白の該当群がいるものとする。それぞれのパターンについて，数字の小さいものから1つ点を選びだし，以下のチェックを行う。チェックにパスした点がみつかった時点でその位置を，包囲候補手の点とする。

(1) 候補手が敵石に隣接していない。
(2) 候補手の遮断点に敵石がいない。
(3) 候補手と基点が大ゲイマを形成するときは，以下の・に敵石がいない。

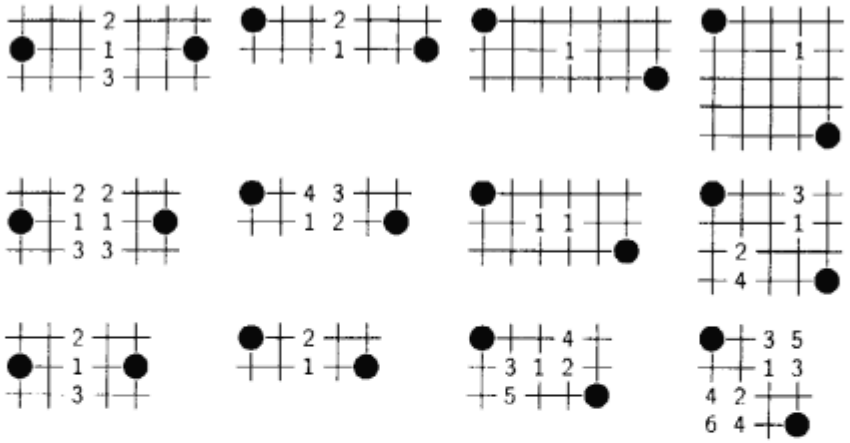
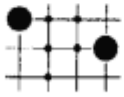


図 3.5–5：群ダメ包囲候補手パターン



以下，脱出候補手位置生成のアルゴリズムを示す。

i. 二間ビラキの余地がある群の場合，二間ビラキを脱出候補手とする。

ii. 上記以外の時，群の 4 次ダメ点を，群ダメ点グループに分ける。
そのグループと該当群の間に，結線脱出候補手があるときは，群ダメ脱出候補手は生成しない。(包囲結線があるかないかで調べる)

iii. 群ダメ点グループ内の群ダメ点の数が 8 未満のとき → 群ダメ点グループ内の群ダメ点の重心を求め，群の中でその重心に最も近い (ユークリッド距離) 石を求め，その点を脱出基点と呼ぶ。
群ダメ点グループ内の群ダメ点の数が 8 以上のとき → 群ダメ点グループ内の両方の端点から 2 個目の 2 点を選ぶ。群の中で，これらの点のそれぞれから最も近い石を求め，その点を脱出基点と呼ぶ。

iv. 基点から二間，一間，コスミ，ナラビの点のうち，2 線以上でもっとも群ダメ点の重心に近い点から順に並べ，それぞれにつき以下のチェックを行う。チェックにパスした点がみつかった時点でその位置を，脱出候補手の点とする。
ただし，候補手を生成した脱出連のダメ数が 2 以下であるとき，候補手の位置は敵石に隣接して，ナラビの点とする。チェックをパスした候補手が愚形を作るときは，次の順位の候補手についてチェックを行っていき，どの手もパスしなかったら愚形となる手を候補手とする。

A. 候補手が一間のとき，一間の中間の点が敵石に隣接していない。

B. 候補手が大ゲイマ，二間，ケイマのとき，ノゾキ点に敵石がいない。

C. 被包囲群が基点を結ぶ線のどちらか一方にしかない時，その候補手の位置は，包囲の基点を結ぶ線より外側または包囲の基点を結ぶ線より内側でも距離が 1 以内にある。

(b) 候補手評価値

- 一次評価値＝群のサイズ× 2 ×生存確率の出入り

- 生存確率の出入り＝脱出手着手後の生存確率－包囲手着手後の生存確率
  着手後の生存確率は，着手後の群の強度を求め，その値から算出する。
- 一次評価値が0の候補手に対しては，包囲／脱出候補手はなかったものとする。
- 二次評価値＝包囲している群の強さの変化＋勢力の出入り＝種石重要度×生存確率の出入り＋勢力の出入り
- 強度が18以上で包囲度が7以上の群に対する包囲／脱出候補手の評価値には，1／2を乗ずる。
- 一間ビラキのある群に対する包囲／脱出候補手の評価値には，1／2を乗ずる。
- 1つの群の弱群手に連絡／分断候補手と包囲／脱出候補手の両方の手があり，包囲／脱出候補手の評価値が連絡／分断候補手の一次評価値(複数ある時は最大のもの)以下である時は，包囲／脱出候補手の一次評価値を0にする。

(c) 包囲／脱出候補手例

包囲／脱出候補手の例を図3.5–6に示す。

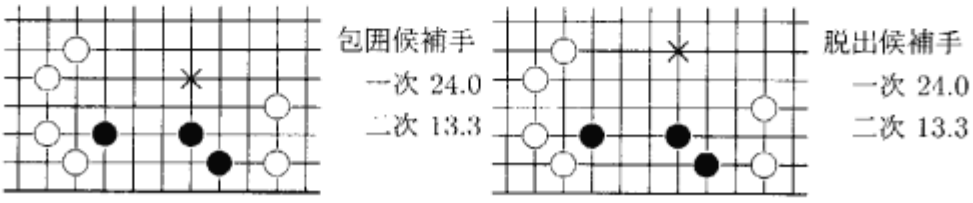


図 3.5 6：包囲／脱出候補手例

[今後の改良ポイント]

- 逃亡の候補手位置として，群ダメの中央に逃げるのが最も安全な逃げ方だが，周囲に与える影響が少ないので，もっと一間で逃げる方がよいかもしれない。
- モタレて逃げるなどという手筋を取り入れた方がよい。
- 結線包囲／結線脱出に比べて評価値が大きいきらいがある。他の弱群候補手の評価値とのバランス調整をもっと検討した方がよい。

2. 結線包囲／結線脱出ケース

結線によって完全に囲まれている群が，包囲している結線のうちの弱結線を破って脱出する手を結線脱出候補手とする。また逆に，そのような弱結線を守る手を結線包囲候補手とする。

(a) 候補手位置

群を包囲している結線のうち，被包囲群側からツキヌケ可能な結線があれば，その結線をツキヌケる位置を候補手位置とする。この候補手位置は，結線の認識時に求められる。

ただし結線脱出パターンであっても，以下の条件のいずれかが満たされていない時は，結線脱出候補手は生成しない。

- 脱出後の強度が7.5以上になる。

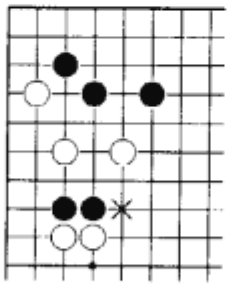
- 包囲群の中に補正前強度が［被包囲群の脱出後強度＋8］未満の群がある。

(b) 候補手評価値

- 一次評価値＝群のサイズ×2×生存確率の出入り
- 生存確率の出入り＝脱出後の生存確率－脱出前の生存確率
- 二次評価値＝包囲している群の強さの変化＝種石重要度×生存確率の出入り

(c) 結線包囲／脱出候補手例

結線包囲／脱出候補手の例を図3.5-7に示す。



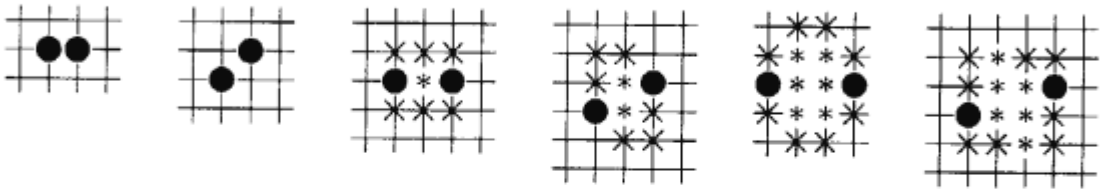
結線包囲／脱出候補手
一次 8.0
二次 15.2

図 3.5-7：結線包囲／脱出候補手例

3. 連絡／分断ケース

弱群の近くに味方の群があれば，その群とつながることによって，群は安全な状態になる。逆に攻め側としては，そのような群につながらせないようにしたい。この目的を達成するのが連絡／分断候補手である。

弱結線でつながっている味方の群を援軍，結線はないが一手打つと同時に2つの強結線ができてつながる味方の群を遠援軍と呼ぶ。ここでいう強結線はノゾキのない結線のことである。遠援軍手および連絡先の点がともに4線以下の時，＊に敵石もしくは敵遮断点(線足の遮断点は除く)があったらノゾキとする。遠援軍手または連絡先の点が5線以下の時，×または＊に敵石があったらノゾキとする。



両アタリのように，一手打てばそれに隣接する連のうちどちらかが捕獲できる場合は，連絡／分断ケースとした。

(a) 候補手位置

- 援軍との連絡／分断候補手は，結線のツナギ／キリの点。
  ただし，弱結線のノゾキ石の強度が0の場合は生成しない。
- 援軍との切り違い候補手は，結線の切り違いの点。
  ただし，弱結線のノゾキ石の強度が0の場合は生成しない。

- 遠援軍との連絡候補手は，遠援軍と連絡する手から生成する。
  連絡手が複数ある場合は，できる2つの強結線の両端点間の距離(ユークリッド距離)の大きい方の距離が最小のものとする。それが複数ある場合は，両端点間の結線の距離が小さい方の距離が最小のものとする。それも複数ある場合は任意の1つとする。
- 距離が大ゲイマ以内にある2つの群の間には遠援軍手を生成しない。
- 遠援軍分断手の位置は，両端点の中間の点とする。
- 多重捕獲切り／ツナギ(または両アタリ)候補手は，コスミ結線の端点の連のダメ数がそれぞれ2と3以下で，キリを入れてツイだ図で，どちらかの連が捕獲可能の時のキリ／ツナギの点。

(b) 候補手評価値

援軍または遠援軍による連結／分断候補手の場合

- 一次評価値＝自群サイズ×2×自群の生存確率の出入り＋連絡先の群のサイズ×2×連絡先の群の生存確率の出入り
  - 生存確率の出入り(双方の群がそれぞれ，中地≧1のとき)＝100－自群の補正前生存確率
  - 生存確率の出入り(上記以外)＝min(2つの群の補正前生存確率の和,100)－自群の補正前生存確率
- 二次評価値＝包囲している群の強さの変化－自群の種石重要度×自群の生存確率の出入り＋連絡先の群の種石重要度×連絡先の群の生存確率の出入り
- 遠援軍の二次評価値＝0
- 援軍に連絡する手の隣接点に援軍より群強度の大きい味方群がいる時はその群と連絡する価値を援軍候補手の評価値とする。

切り違い候補手の場合

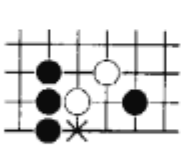
- 一次評価値－連絡／分断候補手と同じ評価値
- 二次評価値＝0

多重捕獲切り／ツナギ(または両アタリ)候補手の場合

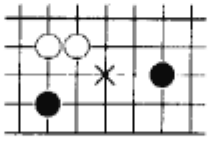
- 一次評価値＝両端点の群サイズの平均×2
- 二次評価値＝両端点の群の種石重要度の平均

(c) 候補手例

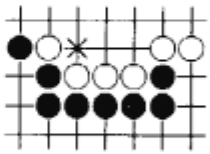
連絡／分断候補手の例を図3.5-8に示す。



援軍連絡／分断候補手



遠援軍連絡／分断候補手



多重捕獲切り／ツナギ候補手

図3.5-8：連絡／分断候補手例

[今後の改良ポイント]

群と群を結ぶことにより新たにできる地について，現在は評価していない。例えば，図 3.5–8の右図では 1,2 線が黒の連絡候補手によって新たにできる地であるが，評価値は各々の群の生存確率を単純に足しているだけで，新たにできる地による生存確率の向上分は考慮していない。このような新たにできる地を求め，評価値に反映させるべきである。碁世代では，この新たにできる地を求めるのに処理時間がかかる (本当に囲った部分が地になるのかを検証する手間が大きい) と思われたので，取り入れられなかった。

4. 死活ケース

群の死活に関する手である。中立の群を完全に殺す／生きるという手だけでなく，群の生きやすさが減少／増加する手も死活候補手とした。

中立連を助ける手，フトコロを拡大／縮小する手，眼を作る／取る手の 3 種類がある。

(a) 候補手位置

死活探索より求まった候補手位置を使用。

ただし，強度が 45 以上の群には，死活候補手を生成しない。

(b) 候補手評価値

中立連を助ける手の場合

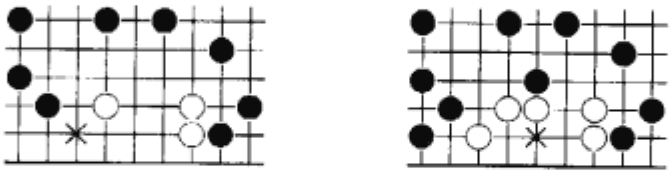
- 一次評価値＝群サイズ× 2 ×生存確率の出入り
- 援軍がいる場合の生存確率の出入り＝援軍の生存確率
- 援軍がいない場合の生存確率の出入り＝死活手着手後の生存確率
- 二次評価値＝包囲している群の強さの変化－種石重要度×生存確率の出入り

フトコロを拡大／縮小する手，眼を作る／取る手の場合

- 一次評価値＝群サイズ× 2 ×生存確率の出入り
- 生存確率の出入り＝生存確率 (先着時強度) －生存確率 (後着時強度) ＝生存確率 (死活手による先着中地× 5 ＋包囲度× 2 ＋援軍による補正) －生存確率 (死活手による後着中地× 5 ＋包囲度× 2 ＋援軍による補正)
- 二次評価値＝包囲している群の強さの変化＝種石重要度×生存確率の出入り

(c) 候補手例

死活候補手の例を図 3.5–9に示す。



フトコロ拡大／縮小候補手　　眼取り／眼作り候補手

図 3.5–9 : 死活候補手例

5. 攻め合いケース

群同士の攻め合いに関する手である。

攻め合い候補手は，群の認識時に起動される「攻め合い探索」結果を用いている。

(a) 候補手位置

攻め合い探索より求まった候補手位置を使用。

(b) 候補手評価値

一次評価値：先着／後着における攻め合い探索の結果から求まる。

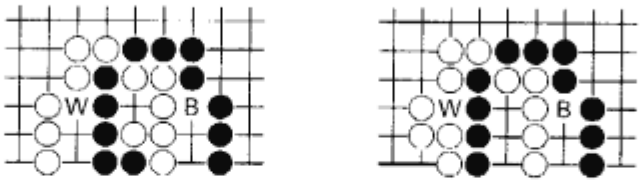
- 後着負け，先着勝ち → 自群のサイズ×２＋敵群のサイズ×２
- 後着負け，先着セキ → 自群のサイズ×２
- 後着負け，先着不明 → 自群のサイズ×２＋敵群のサイズ
- 後着セキ，先着勝ち → 敵群のサイズ×２
- 後着セキ，先着セキ → 0
- 後着セキ，先着不明 → 敵群のサイズ
- 後着不明，先着勝ち → 自群のサイズ＋敵群のサイズ×２
- 後着不明，先着セキ → 自群のサイズ
- 後着不明，先着不明 → 自群のサイズ＋敵群のサイズ

二次評価値＝攻め合っている群以外の周囲の群の強さの変化

- 後着負け，先着勝ち → 周囲の味方群のサイズ×２＋周囲の敵群のサイズ×２
- 後着負け，先着セキ → 周囲の味方群のサイズ×２
- 後着負け，先着不明 → 周囲の味方群のサイズ×２＋周囲の敵群のサイズ
- 後着セキ，先着勝ち → 周囲の敵群のサイズ×２
- 後着セキ，先着セキ → 0
- 後着セキ，先着不明 → 周囲の敵群のサイズ
- 後着不明，先着勝ち → 周囲の味方群のサイズ＋周囲の敵群のサイズ×２
- 後着不明，先着セキ → 周囲の味方群のサイズ
- 後着不明，先着不明 → 周囲の味方群のサイズ＋周囲の敵群のサイズ

(c) 候補手例

攻め合い候補手の例を図 3.5-10に示す。



黒番：勝ち／白番：負け　　黒番：セキ／白番：勝ち

図 3.5-10：攻め合い候補手例

3.5.3.5　族の分離／連絡ケース

盤上のサイズ 30 以上の族に対して分離 / 連絡候補手を生成する。分離 / 連絡候補手とは、その手によって 1 つの族が複数族に分離する (またはその手を防ぐ) ような手のことである。

1. 候補手位置

ポテンシャル分離点の位置とする。ただし、ポテンシャル分離点の算出方法は「馬の背による近似」を使う。

ポテンシャル分離点は，ポテンシャルを標高と考えた時，言わば山脈の馬の背における鞍部の位置になると考えられる。従って，族内の点の中から，その点の周囲のポテンシャルの配置が馬の背の形をしている点を求めることは，ポテンシャル分離点を求める１つの近似となると考えられる。
馬の背の形をした点を求める為に以下の処理を行う。

- 族の各点についてその点を中心に３×３の範囲のポテンシャル分布を求める。
- 図 3.5–11 の左図のポテンシャル分布のように，中心の点の周囲に中心の点より高いポテンシャルの点の集まりと，中心の点より低いポテンシャルの点の集まりが２個ずつ交互に分布している場合，この中心の点は馬の背の形をしていると考える。

図 3.5–11：馬の背の形の点の例

- 図の中央の図と右図では，中心の点のポテンシャル値と等しいポテンシャルの点が周囲に存在している。この場合，正確にはポテンシャルの等しい点の方向の先のポテンシャル分布を考慮しなければ，馬の背の形をしているかどうか判断できないが，現在この２つの例の場合もそのままポテンシャル分離点としている。
- ただし，以下の点は馬の背の形をしていてもポテンシャル分離点からは除外している。
  - 石に隣接している点
  - 高さが２線以下の点

2. 評価値

初期の碁世代では良い候補手であったが，碁世代の改良が行なわれるのにともない，他の候補手の精度が上がったため、この候補手の存在価値自体がなくなってきている。現碁世代では，評価値は１点に満たない。

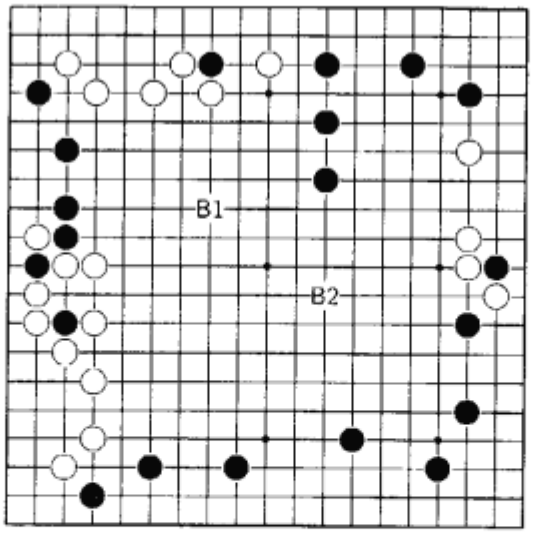
3. 候補手例

族の分離／連絡点の例を図 3.5–12に示す。

3.5.3.6 模様ケース

地模様候補手は模様を地にしようとしたり，逆に地になるのを防ぐような手である。碁世代では，模様を表現するために弱領域という概念を定義した。そこで，弱領域の境界を破る／守る手として地模様候補手と，弱領域を増やす／減らす手として模様接点候補手を作った。

1. 候補手位置



B1：分離候補手 (黒番)

B2：連絡候補手 (黒番)

図 3.5–12：族の分離／連絡点候補手例

- 地模様手 (弱領域の境界を破る／守る手)

  弱領域の境界がノゾキ手番で突き抜け可能な弱結線のとき，その結線を強化 (ツナグ) ／切断 (キル) 手。位置は，結線の認識時に求まる結線をツナギ／ツキヌケの点を使用する。

- 模様接点手 (弱領域を増やす／減らす手)

  互いの弱領域の境界の基点の石の位置と候補手位置をパターンで登録してあるデータから求める (図 3.5–13)。基点と候補手位置の間には結線ができることを想定しているため，距離が離れ過ぎる手や，結線のノゾキ位置に敵側の石がある場合には，候補手は生成しない。

  模様接点候補手は基点となる黒 / 白の石がともに完全死群でない時のみ生成する。

2. 評価値

- 地模様候補手

  一次評価値＝弱領域の出入り (候補手の位置に，味方が打った場合と敵が打った場合の弱領域の差分)

  二次評価値＝ 2 × (包囲している敵の各群の生存確率の出入り × 2 × サイズ)

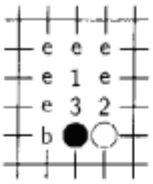
- 模様接点候補手

  一次評価値＝弱領域の出入り (候補手の位置に，味方が打った場合と敵が打った場合の弱領域の差分)

  二次評価値＝ 0

[今後の改良ポイント]

- 中央に地を作る候補手



(1)　bが黒でcと1～3が全て空点の時
　　2(ハネ)：黒のダメ数≧3かつ白のダメ数≦3
　　かつ黒が2へ打った時のダメ数≧3
　　2(ハネ)：黒のダメ数≧4かつ白のダメ数≦4の時
　　1(一間)：上記以外
(2)　(1)以外の時
　　「1(一間)」を「3(ノビ)」に変える以外は(1)と同様

上記のナラビ以外のパターンでは，模様候補手のコスミ範囲の点の内，
黒白の基点以外の点はすべて空点であること

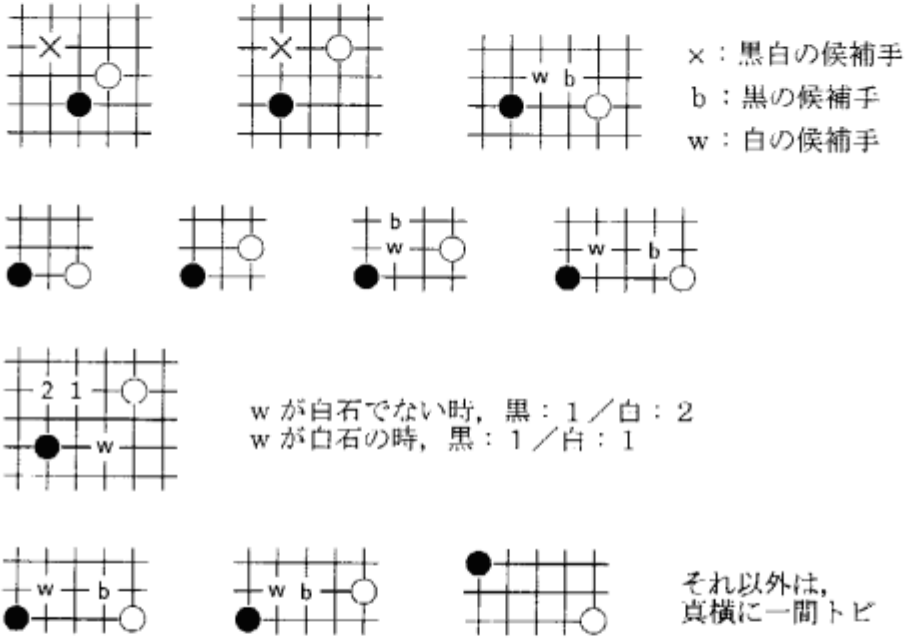


図 3.5-13：模様候補手の例

模様の拡大／縮小候補手は，模様が接近している場合には活躍するが，遠く離れている時には候補手が出ない。そのため碁世代は，敵が近くにいない場合の「模様の拡大」や「消し」ができない。

また，1目の地を作る(囲う)ような小さい地を作る候補手もできない(模様とは言えないかもしれないが)。

3.5.3.7　結線の切断／連結ケース

弱結線を切断／連絡する手。ただし，弱群をつなぐ結線の切断／連絡手，模様の境界結線の切断／連絡手は，別に評価する(3.5.3.4章,3.5.3.6章 参照)。

1. 候補手位置

- 通常の結線の切断／連結候補手

結線の認識時に，大ゲイマ，二間，ハザマ結線の結線種別はパターンデータによって求められる。同時に，もしその結線が弱結線ならば，結線を切る／つなぐ手も求められる。

その他の結線は，探索によって強結線か弱結線か判断される。この場合も，結線を切る／つなぐ手が認識時に求められる。

- 特殊弱結線の切断／連結候補手

  特殊弱結線をノゾいている中立連の捕獲／逃亡の位置。

2. 候補手評価値

- 通常の結線の切断／連結候補手

  群や模様に関係のない結線を，切断／連結する手にはあまり意味がない。碁世代では，最高でも4点としている。

  評価値は，結線のつながり具合によって4段階用意されている (表 3.5–4)。

表 3.5–4：結線候補手の評価値

結線側＼ノゾキ側	後手切違	不明	先手切違	突き抜け
連結	連絡／切違,1 点	連絡／切違,3 点	連絡／切違,3 点	連絡／分断,4 点
後手切り違い	－－	有利な切違,2 点	有利な切違,2 点	切違／分断,3 点
不明	－－	有利な切違,2 点	有利な切違,2 点	切違／分断,3 点
先手切り違い	－－	－－	－－	切違／分断,1 点

- 特殊弱結線候補手
  - 一次評価値 ＝ 中立連のサイズ×2

    二次評価値 ＝ 特殊弱結線の両端点の群の分離評価値×1/2
  - なお，この候補手が同じ位置に複数存在する時は上記の一次／二次評価値を合算する。

3.5.3.8　打ち込みケース

盤上の四隅に対して三々への打ち込み／打ち込み防止の手を生成する。三々以外の一般的な打ち込みについては，辺点候補手生成処理で行われる。

1. 候補手位置

   隅の4×4の範囲に星の位置のみ石があり，星以外の点はすべて空点の場合，三々の位置に打ち込み／打ち込み防止候補手を生成する。

2. 評価値

- 一次評価値＝20
- 二次評価値＝0

3. 打ち込み候補手例

打ち込み／打ち込み防止候補手の例を図 3.5–14に示す。

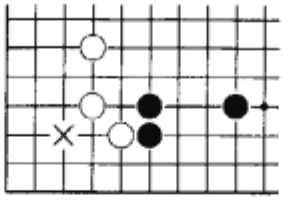


図 3.5-14：打ち込み／打ち込み防止の例

[今後の改良ポイント]

- 碁世代では，打ち込みの手も打ち込み防止の手も同じ位置に生成される。本来は，打ち込み防止の手の位置は，周囲の状況によって星からのサガリやコスミなど変化させる方が良い。
- 星に対しての三々の打ち込みのみ，候補手を生成しているが，小目からのシマリなどに対する打ち込み候補手も用意すべきである。
- 一般に，隅への打ち込みは手筋と考えられる。定石のように手順ごとデータとして与えることが考えられる。

以上の3点は，碁世代設計時に考慮されたが，作業時間が足らなかったため，インプリメントには至らなかった。

3.5.3.9 ヨセケース

「ヨセ」とは，自分方の地の拡大／相手方の地の縮小を行なう手のことである。「ヨセ」の手が発生する状況とは，自分方または相手方に，地や地になりそうな部分があるときである。

碁世代では，辺においては双方の石の配置のパターンで，中央においては地と地を囲う結線の存在から，候補手を生成する。

- 辺のヨセ

  一般に辺のヨセ候補手の位置は，局所的な石の配置からでは正しく求められない。それは，ヨセとして打った石が相手に取られたり，後手になるためにあまり得をしないことがあるからである。碁世代ではそのような問題に対処するために，部分的な先読みを行ない，その結果から，候補手の位置を補正している。部分的な先読みとは，候補手及びそれに対抗する相手方の一般的な対応手を打った局面を想定し，その仮想局面においてある石の生死を判定する捕獲探索である。

- 中央のヨセ

  強結線をノゾク手／膨らむ手，敵石をアタリにする／アタリを防ぐ手の2種類の候補手を生成する。

1. 候補手位置

   - フトコロパターンによるヨセ候補手

辺における高さ4以下で二間以内の異色対峙のすべての組合せについて，先着／後着の候補手パターンを登録した。辺のヨセ候補手位置はこのフトコロパターンより求められる。

ただし，完全死群からのヨセ候補手，完全死群へのヨセ候補手は生成しない。

- ヨセパターンによるヨセ候補手

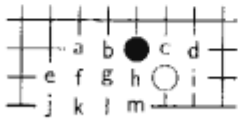
フトコロパターンのうち，ヨセとしてよく使われるものについてはヨセパターンとしても登録した。ある異色対峙がヨセパターンとフトコロパターンの両方登録されている場合は，ヨセパターンが優先される。

ヨセパターンでは，候補手の位置が周囲の状況により細かく設定されているだけでなく，その手が先手か後手まで登録されている。

また，中央のヨセについてはすべてヨセパターンが用意されている。

ただし，完全死群からのヨセ候補手，完全死群へのヨセ候補手は生成しない。

図3.5 15に辺のヨセ候補手を，図3.5–16に中央のヨセ候補手パターンを記す。



・黒の先手／後手

黒 h へ仮打ちした局面で○を黒番で捕獲可能なら先手，以外は後手

・白の先手／後手

－手の位置が h である時

a,b,f,g に黒がないならば先手，以外は後手

－手の位置が g である時

a,b,e,f,g がすべて空点ならば(盤外は空点以外とすること) 先手，以外は後手

－手の位置が l である時

白 l，黒 f，白 k，黒 j，白 h，黒 g，白 m に仮打ちした局面で j の黒石が白番で捕獲可能なら先手，以外は後手

・黒番の位置

h

・白番の位置

－ c,d,i に白石がなければ h

－上記以外で○のダメ数5以上で g が先手となるならば g

－上記以外で l が先手となるならば l

－上記以外は h

・出入り値

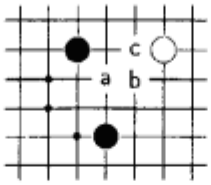
両先手なら9目，片先手なら8目，両後手なら7目

図 3.5 15：辺のヨセ候補手位置の例(2線のオサエ)

- アタリによるヨセ候補手

ダメ点の手が以下のいずれかの条件を満たす時，その点をアタリによるヨセ候補手とする。

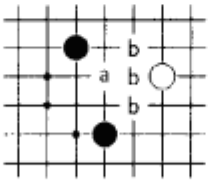
大ゲイマ a



a, b, c は空点であること

黒番：後手, b
白番：先手, a
評価値：4 目

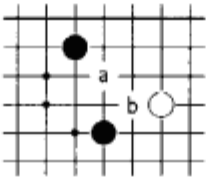
大ゲイマ b



a, b は空点であること

黒番：後手, a
白番：先手, a
評価値：4 目

大ゲイマ c



a, b は空点であること

黒番：後手, a
白番：先手, b
評価値：4 目

図 3.5 16：中央のヨセ候補手位置の例 (大ゲイマ)

条件
(a) その手によって敵石をアタリにできる
(b) その手が敵石からのアタリを防ぐ

2. 評価値

- フトコロパターンによるヨセ候補手
  先番と後番について，フトコロパターンから生成される候補手を打着した時にできる領域を予想し，その出入りから評価値を求める。
- ヨセパターンによるヨセ候補手
  ヨセパターンには，候補手位置と基本評価値がセットになって登録されている。
  ヨセ候補手の評価値は以下の式で求められる。

ヨセ評価値＝基本評価値 × 先手ボーナス係数

ヨセ候補手は，先手である方が後手あるよりも評価値が高くなるべきものである。先手ボーナス係数は，表 3.5–5の通り。

表 3.5 5：先手ボーナス係数

	両先手	片先手	逆ヨセ	両後手
係数	2.0	1.5	1.25	1.0

囲碁において，ヨセの正しい計算方法は確立されていない(大雑把な目安として，「逆ヨセ 2 倍」とか「大きい方から打て」などの格言はあるが，その手が「先手か後手か」がわからないことによる)ので，碁世代では各ヨセパターンについて，標準的な評価値を登録している。

- アタリによるヨセ候補手

  ダメ点の手が以下のいずれかの条件を満たす時，その点をアタリによるヨセ候補手とし，評価値を以下のように与える。

$$\text{評価値} = \max(\text{従来の評価値}, 1) \times k$$

  条件

  (a) その手によって敵石をアタリにできる

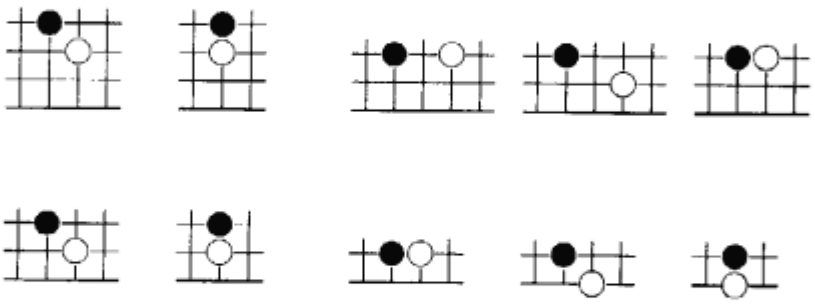
  (b) その手が敵石からのアタリを防ぐ

ここで k は (a),(b) ともに満たす時 2.0, (a) のみの時 1.5, (b) のみの時 1.25 とする。

3. ヨセ候補手パターン

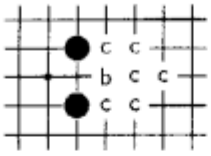
ヨセ候補手の登録パターンを列挙する。ただし，スペースの都合上，候補手位置と基本評価値は略す。

[**辺のヨセパターン**]

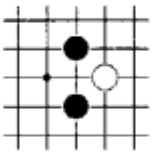


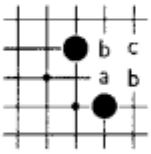
[**中央のヨセパターン**]

強結線が以下のパタンである時ヨセ候補手を生成する。以下のパタンで小さい黒石はそのいずれかが黒の領域であることを示している。

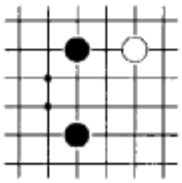


b が空点であること
c は白または空点であること

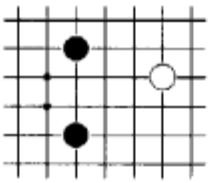




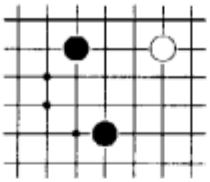
a は空点であること
b のどちらかが白か, b が両方空で
c が白であること



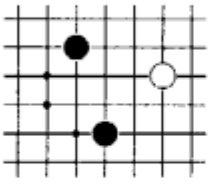
a, b は空点であること



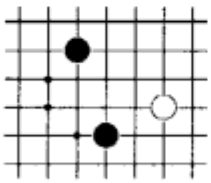
a, b は空点であること



a, b, c は空点であること



a, b は空点であること



a, b は空点であること

[今後の改良ポイント]

- パターンについて

  碁世代に登録されたヨセパターンは少ない。また，各パターンのサイズも小さい。パターンの数を増やすか，部分的な先読みを積極的に取り入れることにより改良すべきである。

- 多重ヨセ，見合い

  一手で二方向をヨセることができる場合がある。逆に，ヨセたつもりが実はヨセていない「見合い」のような場所もある。現在はそのようなことを全く考慮していない。

- 隅のヨセ候補手の評価値

  隅のヨセは，辺の中央のヨセと比べて評価値を変えるべきである。

3.5.4 旧碁世代における候補手生成と評価

旧碁世代における候補手生成と評価の方法について簡単に記述する。旧碁世代のどの部分が悪く，それをどう改良して現在の碁世代になっていったかを述べる。

なお，ここでいう旧碁世代とは，1990 年度版の碁世代である。詳細は参考文献 [実近 91-2] を参照のこと。

3.5.4.1 ケースに基づく候補手

ケースに基づく候補手の生成という方針は，初期の頃から変わっていない。主に変わった部分は，候補手知識の種類とその評価方法である。

表 3.5–6 に 1990 年度版の碁世代の候補手知識を記す。

評価値は，典型的なケースに対する評価値が算出される。周囲の状況に応じて評価値が変化することは少ない。

表 3.5–6：候補手知識

名称	内容
定石	序盤，隅における標準的な手筋 (239 型)
辺点	ヒラキ・ツメ・割り打ち等の辺上の手
打ち込み	打ち込みと防御
捕獲／逃亡	連を取る／逃げる
ダメ点	ハネ・ノビ・オシ等の石の競り合いの手
結線の切断／連結	石をつなぐ / 切る手
群の包囲／脱出	群を包囲する／逃げる手
フトコロ拡大／縮小	領域を増やす / 減らす手
死活，攻め合い	中手の手，攻め合いの手
族の分離／連絡	族同士の分離 / 連絡の手
地模様	模様を地にする
模様接点	模様の拡大／縮小

3.5.4.2 候補手の改良

個々の候補手について，主な改良点を記す。

1. 定石候補手

   定石の登録パターンの増加。

   周囲の状況によって異なる定石を選ぶ。

2. 辺点候補手，打ち込み候補手，ダメ点候補手

   特に大きな改良はない。候補手位置や評価値の整備のみ。

3. 捕獲／逃亡候補手

   死活の対象は，連でなく群と考えるべきなので，対象を連から群に変更した。これにより，評価値をより正確に算出することに成功した。

4. 結線の切断／連結候補手

   「模様を囲う結線」，「群同士がつながることが可能な弱結線」など，結線の意味を考えて評価をすべきであると考え，現在の碁世代では，その意図により各種の結線の切断／連結候補手がある。これにより，評価値を正確に算出できるようになった。

5. 群の包囲／脱出候補手

   候補手の位置について整備した。特に，一手で包囲できる場合と，包囲に二手以上かかる場合とで候補手の求め方が異なる。

   また，弱群候補手としたことにより，評価値が正確になった。

6. フトコロ拡大／縮小候補手

   現在の碁世代の死活候補手の一部 (領域の拡大／縮小) とヨセ候補手に相当する。これも群を攻める／守るのが目的なのか，生死ははっきりしていて単純にヨセの意味だけの手なのかを区別するようにしたため，評価値が正確になった。

7. 死活，攻め合い候補手

   死活候補手は現碁世代の死活候補手の一部 (眼を作る／潰す) に相当する。これも弱群手としたため，評価値が正確になった。また，一眼検出のための処理や，登録した中手パターンの増加などにより，候補手位置も正確になった。

8. 族の分離／連絡候補手

   旧碁世代では有用であったが，現碁世代では「群の連絡／分断候補手」ができたため，必要なくなった。

9. 地模様候補手，模様接点候補手

   現在の碁世代では，二つをまとめて模様候補手とした。模様の表現方法が変わったこと，模様接点候補手パターンの整備により，より正しい手が出るようになった。

3.5.4.3　評価方法の改良

旧碁世代の候補手の評価値は，予め一般的な状況においての評価値を候補手に知識として与えられる。従って，周囲の状況や，局面の進み具合 (序盤・中盤・終盤) と無関係に評価値が出る。そのため，評価値が局面に合っていないことが起きた。

現在の碁世代は，周囲の状況に合わせて評価値が変わる (目的の達成度を使っているから)。また，評価値の基準も統一的に，目数に換算して算出される (模様も目数に換算している)。このようにして，評価値の精度が上がった。

3.6 着手決定

候補手は，各知識が互いに独立に挙げてくるため，異種ケース間の不調和を局所的に調整してより効果を高めたり，不調和を削除する。このような候補手間の調整を行った後で，最も評価値の高い点を打着する。

3.6.1 最終着手決定手順

列挙された候補手より以下の手順で最終的に着手を決定する。

1. 群の緊急度による評価値の補正を行なう。
2. 列挙された候補手に，ボーナス点による評価値の補正を行なう。
3. 各候補手に定石フィルターをかける。
4. 優先順位の高い手の中から，最も評価値の高い手を選択する。
5. 自殺手フィルターをかける。

3.6.2 群の緊急度による評価値の補正

一般に，緊急度が高いと候補手の評価値が高くなるが，現在の碁世代では緊急度の高さが着手の優先順位にあまり反映されていない。そこで，緊急度の高い手をより多く打たせるため，緊急度の高さに応じて，候補手の評価値を上げるというチューニングを行なった。ただし，包囲／脱出候補手の二次評価値には補正を行なわない。

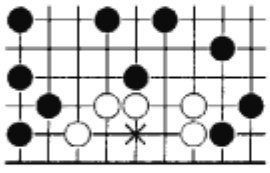
最終評価点 = 評価点 × (緊急度 ／ 50) (緊急度 ≧ 50)
最終評価点 = 評価点 (緊急度 < 50)

さらに，同じ評価値でもサイズが大きい方が一般には急がれない (安全である) ことが多い。そこで，群のサイズも着手の優先順位を考慮する際の，パラメータとした。

候補手の優先順位は以下の通り。

1. 緊急度 ≧ 70 かつ 群サイズ ≧ (105 − 緊急度) の候補手
2. 上記以外の候補手

群の緊急度による評価値の補正の例を図 3.6–1に示す。



死活候補手：19.6
死活手の最終評価値：39.2 = 19.6 × (緊急度 (100) ／ 50)

緊急度 (100) ≧ 70, 群サイズ (14) ≧ 105 − 緊急度 (100)
より 優先順位も最高

図 3.6–1：群の緊急度による評価値の補正の例

3.6.3 ボーナス点による評価値の補正

対局中に候補手の評価値を出す場合，相手の最終着手の近傍にボーナス点を付け，最終着手の近傍の手の評価値を高くした。これは，局面認識の精度が高くない場合，相手の意図が見抜けずに，大事な手を逃すことがあるためである。しかし闇雲に相手の手に従うのではよくない。着手の近傍の結線の連結／切断候補手とダメ点候補手位置に対してだけ，評価値を高めることにした。

- ノゾキの手に反応

  手の位置：最終相手着手に隣接する味方弱結線のノゾキ強化点
  手の評価値：結線の切断 / 連結候補手評価値 ×2.5

- ツケやオシ / ハネなどダメをツメてきた手に反応

  手の位置：最終相手着手に隣接する味方連のダメ点
  手の評価値：ダメ点候補手評価値 ×2.5
  ただしナラビダメ点候補手は，フクラミの手がダメ 2 以下の場合はボーナス点を与えない (アタリされて後手を引くから)。

- カタツキの手に反応

  手の位置：最終相手着手の点のダメ点
  手の評価値：最終相手着手の連のダメ点候補手評価値 ×2.5

3.6.4 各種のフィルター

候補手は，各知識が互いに独立に，ある目的を達成する手を挙げてくるため，他の視点から見ると不都合な点に生成されることがある。このような手を取り除くのがフィルターである。

1. 定石フィルター

   「定石」とは，盤上の四隅における序盤の攻防の最善手の手筋である。従って，盤上の四隅のうち，定石進行中の隅については定石候補手以外の候補手は不要である。そのため，定石フィルターを設け，定石フィルター内に生成された定石候補手以外の候補手はすべて取り除く。

   定石フィルターの範囲は各四隅の $n \times n$ の点の範囲で，n は以下の式によって求める。

   $$n = \text{路数}/2 - 1$$

2. 自殺手フィルター

   最終着手決定で求められた手が，相手番で捕獲可能な場合，その手は無駄になることがある。そのようなことを防ぐため，自殺手フィルターを設けた。

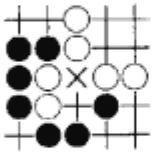
   最終着手決定で求められた次の一手を実際に打ってみて，着手の反対側の手番で，着手した石が捕獲可能かどうかを捕獲探索によって調べる。捕獲可能な時は，その手以外の手で最終着手決定の処理が行なわれる。

   ただし，「ウッテガエシ」や「ホウリコミ」といった捨て石まで，自殺手フィルターで取り除かれては困る。碁世代では，「自殺手であっても，それが捕獲手でかつ捨石となる手

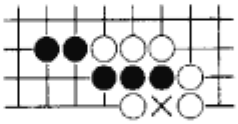
ならば自殺手フィルターによって取り除かない」とした。ここで捕獲手かつ捨石となる手とは，捕獲に必要とする手数を N とし，捕獲手自体が捕獲される手数を n とする時，

$$N \geq n + 2$$

が成り立つならば，その捕獲手は捨石となる手とする。ただし，この時の手数とは探索の過程において，ターゲットの石が抜かれる終端までに要する手数の最大のものとする。



捨て石となる捕獲候補手
3(N) = 1(n) + 2



捨て石にならないのでフィルターにかかる手
3(N) < 5(n) + 2

図 3.6-2：群の緊急度による評価値の補正の例

3.6.5 その他の補正

1 つの弱群に対して，複数の弱群手が生成されたとき，どちらかの手を優先させた方が良い場合がある。

碁世代では，包囲／脱出候補手の手より連絡／分断候補手を優先させるために，「包囲／脱出候補手の一次評価値が連絡／分断候補手の一次評価値以下である時は包囲／脱出候補手の一次評価値を 0 点にする」とした。

他に，脱出候補手と包囲候補手との優先順位についても考慮された。これは，ある包囲されている群が，近傍に敵の群を包囲する手と，自群が脱出する手の両方をつくり出した場合，どちらの手を優先させるかというものである。我々は，包囲されている群の強度と，近傍の敵の群の強度から優先順位を決めるよう設計したが，インプリメントには至らなかった。

3.6.6 旧碁世代との比較

1. 評価値の加算

   1990 年度版の旧碁世代では，同じ点 (座標) に複数の候補手が生成された場合，その点の着手の評価値は各種の候補手の評価値の和としていた。しかし，各種の候補手が全く別々の要素を使って評価しているわけではないので，評価値の加算による不具合も大きかった。現在の碁世代は，異なる意図は二次要素で計算されるため，加算の必要性がなくなった。

2. 着手の優先順位

   1991 年度版の旧碁世代では，候補手の評価値と群の緊急度を組み合わせて，優先順位を決めていた。

   (a) 緊急度 100 ～ 80 の群についての候補手の評価値が 30 目以上の大きさの手
   (b) 緊急度 80 未満の群，それ以外についての候補手で，評価値が 30 目以上の手
   (c) 緊急度 100 ～ 80 の群についての候補手の評価値が 20 目以上の大きさの手
   (d) 緊急度 80 未満の群，それ以外についての候補手で，評価値が 20 目以上の手

(e) 評価値が最も大きい手

この方法では，緊急度や目数が離散的なので，境界値付近での不正確さがあった。

3. 群の緊急度による評価値の補正

1990 年度版の旧碁世代には，この概念がなかった。そのため，序盤も終盤も，危険な群の手も安全な群の手も，評価値に変化がなかった。「群の緊急度」という概念を取り入れたため，緊急度の高い (囲碁用語で忙しい) ものから打つようになった。

4. フィルター

1990 年度版の旧碁世代には，現碁世代の各種のフィルター以外に「無駄手フィルター」が存在した。これは，候補手が単にパターンから出される時，他の視点から不都合な候補手が頻繁に出たためである。しかし現碁世代では，各候補手におけるパターンが整備されたことにより，この「無駄手フィルター」が不要になった。

3.7 評価

逐次版碁世代の個々のモジュールについての評価と課題については，それぞれの章で記述した通りである。ここでは，逐次版碁世代全体の棋力について述べる。

3.7.1 思考方法のシミュレート

人間の思考方法をシミュレートするという方針で碁世代を開発したが，まだ人間の思考方法を全てモデル化したとはいえない。個々の知識を整備する段階で終っている。特に，以下の項目についてはまだ手がつけられていない。

- 問題のありそうな場所の絞り込み，種石・カス石の判断

  群に緊急度という属性を持たせて，候補手を選ぶ場合の絞り込みは実現したが，認識の段階での絞り込みは行なっていない。

- アジ／キキを利用した読み筋の組み立て，先手／後手の判断，複数の標的の設定
- 作戦に沿った一貫性のある着手
- 局面に応じた手筋の活用
- 学習

3.7.2 コンピュータ囲碁の世界大会

逐次版碁世代は，1992年11月11日～11月12日に，日本棋院会館(東京)で行なわれた1992 International Computer Go Congress(國際電腦圍棋賽)に参加した。結果は，表3.7-2の通り，第4位であった。上位6プログラムの実力は接近していて，激しい星の潰し合いが繰り広げられた。碁世代は4勝2敗の成績を残した。碁世代は，世界のトップレベルを持つプログラムであることがわかった。(この大会の棋譜は本論文の付録参照)

3.7.3 コンピュータ囲碁と人間の棋力

強さとは，対局中の全ての手のうち，最も悪い手に依存する。つまり，1手の悪手によって，その他の百数十手の良い手の価値が打ち消されてしまう。コンピュータ囲碁は，一部の探索は人間より深く広く読めるし，パターンで与えられる知識は，絶対に間違えないし，忘れないので，人間より優れている。しかし戦いの最中では「ソッポ」や「おびえた手」など人間よりも劣っていると思われる部分がある。そのため，まだアマチュア中級程度の人間と対戦すると負けてしまうことが多い。問題集により正解率で人間と勝負するならば，アマチュア中級者に勝つことはできても，人間とは異なるバランスを持っているために，対局すると負けるのである。コンピュータ囲碁と人間では，棋力の判定の基準を変えるべきだという意見もコンピュータ囲碁の開発者の中にはある。

3.7.4 総合的に見ての碁世代の棋力

人間の棋力を判定する場合には，一般に2種類の方法がある。一つは，人間同士がプレイして，相対的に強さを調べるもの。もう一つは，問題集をとかせて，得点によって客観的に強さを調べるものである。しかし，問題集での成績が良かったからといって，成績の悪かったプレイヤーに常に勝てるわけではなく，また逆に，対局するといつも勝っているプレイヤーと問題集の成績

を比べると劣っていることもある。また，同じ初段といっても，実際は4子ぐらいの幅があり，人間の棋力の判定は結構いい加減なものである。

碁世代は，個々のモジュールについては，問題集を実行させることによりアマチュア中級以上の力があると判定されるが，対局全体を通じてみると，まだアマチュア中級者には及ばない。しかし，コンピュータ囲碁界では，世界最高レベルの強さを持っている。

強さの判断については客観的な評価ができないが，我々は，「棋士システムのプロジェクトの目標であったアマチュア中級にまでは達してはいないが，ほぼそれに近い強さになった」と考えている。

碁世代には，まだまだ実現されていない機能が多くあり，それらを取り込むこと，現在持っている知識の整備を続けることにより，当初の目標であったアマチュア中級レベルの人間にも勝つことができるようになるだろう。

3.8 開発環境 (ツール)

開発ツールは，囲碁プログラム本体と直接の関係はないが，試行錯誤的な改良を余儀なくされている本システムのような囲碁プログラムにおいては，見かけ以上に重要な部分である。これにより，開発期間を大幅に短縮できたり，また間接的にプログラムの質を向上するのに役立つ。

本システムは，対局モードと実験解析の間を実行時に自在に行き来できるように設計してある。これもデバッグの効率向上をはかる上で重要なことである。

開発ツールは大きく以下の3種類に分類される。

- 実験解析モード

  対局の任意の時点で局面認識や着手決定の処理で得られた結果を参照したり，局所探索を駆動させてその結果や探索過程を見ることができる。また，各種知識で使われているパラメータを一部変更して対局にどのような影響が現れるかを確認することもできる。

- 知識エディタ

  本システム内のデータベース化された知識を編集する為のツール群である。

- 評価用ツール

  改良による効果を正確に評価する為のツール群である。

3.8.1 実験解析モード

これはいわゆるデバッグを行うモードである。対局中に問題が生じたときでもその状態を直接調べることが出来るように対局モードと実験解析モードはオンラインで任意に切り替えが出来る機能を持つことが望ましい。実験解析モードではデータ構造内の全ての対象及びその属性が覗ける peep 機能が必要である。更に問題を生じたサブモジュールの trace 機能も必要である。

実験解析モードでは以下の機能が提供される。

- データ構造表示

  現在の局面に対する各種データ構造及びそれぞれの対象の属性を表示する。

- 候補手表示

  候補手列挙及び候補手調整によって得られたスコアテーブルの内容を表示する。また，個々の候補手がどのような意図から成り，かつそれぞれの意図がどれぐらいの価値を持っているかを等価目数換算した値で表示する。

- 形勢表示

  現在の局面に対する形勢を表示する。

- 探索過程表示

  各種探索モジュールを呼びだしてその探索過程と結果を表示する。また，探索における候補手の優先順位も表示可能である。

- パラメータのチューニング

  対局システムにおける各種囲碁知識の中に使われている様々のパラメータに対する参照/更新の処理を行う。

3.8.2 知識エディタ

本システムにおける知識の一部はデータベース化され，対局中に必要に応じて適宜参照される。知識エディタはこれらデータベース化された知識の入力及び管理を容易にする為のツール群である。これらデータベース化された知識は以下のような知識エディタによって編集される。

- 問題図エディタ

  デバッグの為のいろいろな局面を作成したり，保存／再生したりする。また，それらの局面を問題として自己採点する機能がある。自己採点には，捕獲や詰碁などの探索問題や，「次の一手」などの着手決定の問題，また石の強弱の判定など局面認識の問題が登録できる。また，自己採点中に，内部パラメータを変更しながら解の変化を見ることもできる。

- ポテンシャルエディタ

  ポテンシャルは，石がその周囲の点に及ぼす影響度を求める為のものである。周囲への影響度は基本的に石からの距離に反比例すると考えられるが，辺の近傍の点では特殊な事情があり一律に決められない。そこで，人間の感覚を目安に試行錯誤的に決定することにした。ポテンシャルエディタはそのような試行錯誤に要する手間を少なくする為のツールである。

- 定石エディタ

  囲碁の定石は数が多く，また年々新しい定石も増えている。定石エディタは，定石 (定石ノード) の追加 / 削除，評価値の設定を容易にする為のツールである。

- 手筋エディタ

  囲碁には，局所的な石の配置に対して有効な手を教える手筋と呼ばれる知識が数多く存在する。手筋エディタは，そのような手筋の入力や管理を容易にする為のツールである。ただし，手筋エディタはまだ試作段階であり，実際に手筋エディタで編集された知識を対局システムで利用してはいない。

  手筋エディタによって「適用場所」「キーパターン」「適用条件」「意図」が編集される。

3.8.3 評価用ツール

囲碁対局システムのようなプログラムでは試行錯誤的な改良が余儀なくされている。従って，改良した場合はその効果をできるだけ正確かつ効果的に調べる必要がある。本システムではその為のツールとして以下のものが用意されている。

1. 自動テスト機能

   予め用意しておいた問題を解かせて，その正解率や所用時間を表示する局面編集ツール (後述) 内の機能である。与えることのできる問題には，捕獲や詰碁などの探索問題，群の強度などの局面認識の問題，次の 1 手などの着手決定の問題などがある。

2. 棋譜棋力判定ツール

   予め用意された棋譜の手とコンピュータが考えた手に対して人間がどちらの手が良いかを判断していくことによって，棋譜の手とコンピュータの手との相対的評価を行うツールである。

例えば，改良前のコンピュータ同士の棋譜を取っておき，改良後のプログラムで当ツールを走らせると，改良したことの影響として以前と違う手を打つようになった局面が表示される。各局面について改良前と改良後の手でどちらが良い手であるかを人間が評価していけば，改良の効果が評価できる。ある局面でより良い手を打つように改良しても，その改良の為に別の局面で悪い手を打つようになることはしばしば起きる。そのような状況を発見するのに便利である。

3. 棋譜編集ツール

棋譜を簡単に作成する為のツールである。1手目から順に入力していくのではなく，碁盤の上に数字を書いていく要領で入力していく。1つの棋譜を5分くらいで入力できる。

4. 局面編集ツール

ある局面を簡単に作成する為のツールである。碁盤上に黒 / 白の石を適当に置いていく要領で局面が作成できる。問題を作成する時に便利である。

5. 遠隔対局機能

改良の効果を調べる為には実際に対局させてみることが確実な方法である。遠隔対局機能とは，異なるPSI同士，または他のコンピュータの囲碁ソフトと自動的に対戦させる為の，対局システム内の機能の1つである。前者は古いバージョンと新しいバージョンのシステムを対戦させたり，パラメータを一部変えて対戦させる場合に便利である。後者は通信対局機能を備えた市販の囲碁ソフトと対戦させるのに便利である。

6. 消費時間 / メモリ測定機能

囲碁のようなプログラムでは消費時間を一定時間内に抑えることは重要である。本システムでは一局に要するコンピュータの消費時間を30分以内にすることを目安としている。消費時間を一定時間内に抑える為には，導入する知識の量を制限する必要が出てくる。従って，知識の処理に要する時間を減らすことは，導入できる知識の量を増やせることにつながるので，棋力に関係してくる。当ツールは実際の対局をしながら，個々の処理に要した時間とメモリを測定する対局システム内の機能の1つである。各知識の効果とそれに要する時間とを考え合わせて，改良の目安とするのに便利である。

3.9 開発方法

ゲームプログラムのように数多くのヒューリスティックスを組み合わせてできたプログラムでは，最初に設定した通りの強さになるまでには数多くの試行錯誤が必要である。この碁世代も同様に，何度となく試行錯誤が行なわれて現在の姿になった。そこで，この章では改良の方針について記述する。

碁世代の改良方法は，次のサイクルを踏んでいる。

実行 → 悪い部分の抽出 → 改良 → 実行 → …

この悪い部分の抽出には，2通りの方法があり，(1) 問題集を解かせる，(2) 実際に対局を行ない，強い人間プレーヤーにより悪手を選び出す，である。そして，悪い部分の抽出が終ったら，その部分を改良する優先順位を付け，改良を行なう。そして再び，実行させ全体への影響を見ながら，また悪い部分を抽出する。

- 悪い部分の抽出作業

  幸いにも，碁世代の実力は開発者の実力を大きく下回っているため，対局による悪手の抽出作業には，問題は起きなかった。

  しかし，問題集を使って悪い部分を抽出する方法だと，一度改良するとその悪い手を見つけた問題は不要になってしまう。そのため，数多くの問題を用意しなければならない。また，市販の問題集には同じ目的の問題が数多く含まれているため，問題数の割には悪い部分の抽出が少ないことがある。また，人間を教育する時の上達の仕方と，プログラムの上達の仕方が違うので，思いがけないところに悪いところがあっても，問題集では見つけにくい。

  我々は，普段は対局を通じて悪い部分を抽出し，時々バランスをとるために問題集を使うという方針をとった。

- 改良の優先順位

  プログラムは開発者の棋力 (アマチュアの四〜六段) を大きく下回るために悪ところが数多く挙げられすぎた。そして，改良のための優先順位をつけるのには，コンセンサスがなかなかとれなかった。結局のところ，コンセンサスはとらずに，ある一人が優先順位をつけ，その順番に従って改良作業を行なった。これは，開発者の棋力が碁世代のそれを大きく上回っているために，誰が優先順位をつけても大差ないだろうと判断されたためである。

4 並列版「碁世代」

4.1 概要

碁世代には，逐次版と並列版の2種類がある。逐次版は，人工知能の基本問題に関する研究という目的から開発され，並列版は，並列処理の研究という目的から開発された。ここでは，並列版碁世代について説明する。

並列版碁世代は，逐次版碁世代に並列処理を取り入れることによってできた囲碁プログラムである。並列版碁世代は，中国ルールに従って囲碁の対局を終局まで行うことができる。

1. 研究の位置付け

   囲碁プログラムは大規模な知識処理システムであり，第五世代コンピュータプロジェクトにおける大規模並列推論マシンの性能検証，大規模知識処理システムの並列化方式の研究，負荷分散をはじめとする処理効率向上のための研究を目的としている。

   また囲碁プログラムは，問題解決方式自体に不明な点を多く含むという特徴もあり，探索問題，曖昧性の処理，例外処理，協調問題解決などの人工知能の基本問題に関する新しい問題解決方法の研究題材としても適している。

2. 経緯

   第五世代コンピュータプロジェクトの中期の最終年度(1988年)に，並列詰碁プログラムを試作した[沖 89]。

   並列版碁世代は，逐次版碁世代を元に，第五世代コンピュータプロジェクトの後期(1989～92年)に開発された。

   まず初めに，並列版の最初の実験システムとして，思考部分の一部を並列実行し，それ以外の部分は逐次版碁世代を使うという並列部と逐次部を融合した構成のシステムを作成した[清 91-1]。その上で，仕事の粒度の大きさを考慮した負荷分散方式の実験を行ない，その有効性を確認した。次に，全ての部分が並列言語で書かれる本格的な並列版囲碁プログラムを開発した。その上で，並列知識処理の一つのアプローチとして，リアルタイム性を保ちながら，できる限り多くの処理を実行するための方式「遊軍処理方式」を試作した[清 91-2]。

3. 開発体制，動作環境

   囲碁プログラム開発チーム(CGS-TG : Computer Go System Task Group)の一部のメンバーよりなる開発チームが発足した。設計者・プログラマそれぞれ約1,2人の体制であった。

   並列版碁世代は，ICOTで開発された並列型推論マシンMulti-PSIで動く[瀧 88]。なお，Multi-PSIのオペレーションシステムはPIMOSである。

   プログラムのうち，表示部は逐次版碁世代を使い，その他の部分(対局部)はICOTで開発された並列論理型言語KL1で書かれている。KL1で書かれているソースプログラムの総量は約1万行である。

4.2 構成

並列版碁世代は，中国ルールに従って囲碁の対局を終局まで行うことができる。並列版碁世代は，与えられた局面を認識し次の着手を決定する対局モードと，認識した結果を参照する為の実験解析モードとがある。

1. 対局モード

碁世代は，中国ルールに従って囲碁の対局を終局まで行なうことができる。

碁世代の着手決定の手順は3つの段階からなる(図4.2-1)。まず着手により変化した盤面上の石の生死などの局面の認識を行ない，それに基づいて候補手を列挙し，その候補手の評価値の合計が最も高い点の座標を次の着手位置とする。

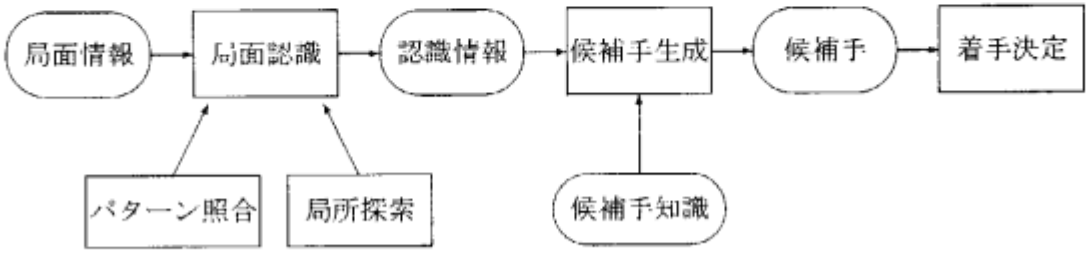


図 4.2-1：碁世代の処理の流れ

2. 実験解析モード

対局モードにおいて入力待ちの状態の時，及び実験解析ツールから実験解析モードへ移行できる。実験解析モードでは，局面認識によって更新された各種データ構造の内容や，着手決定によって求められた各種候補手や形勢が参照できる。また，各種の局所探索を駆動し，その探索の過程や結果なども見ることができる。このとき，プロセッサの台数を変化させて実行することもできる。

4.3 着手生成

並列版碁世代は，1990 年度の逐次版碁世代がベースになっている。局面認識，候補手生成，着手決定の処理のうち，一部を除いて全く同じ処理をしている。1990 年度の逐次版碁世代の詳細な仕様は文献 [実近 91-2] を参照のこと。

4.3.1 局面認識

人間は，盤面を認識する際に，石の配置を単に座標の集合として捉えるのではなく，戦術的に意味のある石の集団として捉えている。碁世代は，人間プレーヤーのように盤面上の単なる石の配置から，高いレベルのデータ構造を作り出し (図 4.3–1)，それらの属性 (地の大きさ, 形, 包囲度など) を算出する。このような階層的なデータ構造は，人間が盤面を認識する際に細かい部分構造から，ある抽象的な見方をするというプロセスによく似ている。また，このような階層的に盤面を認識することによって，盤上のどの部分が重要な部分かがスムーズに把握できる。

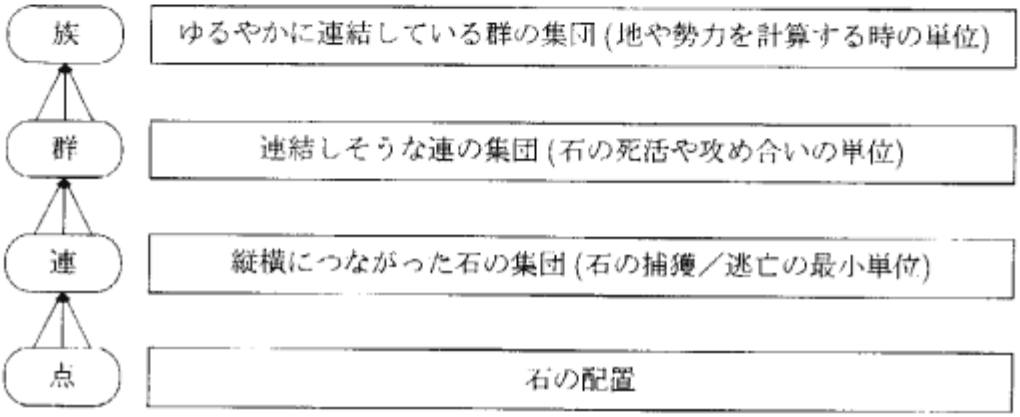


図 4.3–1 : 意味のある集団形態とデータ構造の関係

1. 局面の表現 (データ構造)

   データ構造について，現在の逐次版碁世代と思想は同じである。異なるところは，個々の属性である。

   以下に，逐次版 (3.3.1章) と異なる主な属性をあげる。

   - 連データの死活

     並列マシンでは逐次マシンより計算パワーがあるので，連データの死活探索の対象をダメ 4 以下の連とした (逐次版ではダメ 3 以下を対象)。

   - 援軍 (群データ)

     並列版の群データにおいて，援軍という属性はない。1990 年度の逐次版碁世代における群の分類では，援軍の有無によって分類しなかったためである。

   - 弱領域 (群データ)

     並列版の群データにおいて，弱領域という属性はない。1990 年度の逐次版碁世代では，ポテンシャルを使って模様を表現していたためである。

- 中地 (群データ)

  並列版の群データにおける中地の計算方法は，逐次版と異なる。1990 年度の逐次版碁世代では，フトコロ候補手によりできる地に対して考慮されていなかった。

- 包囲度 (群データ)

  並列版の群データにおける包囲度の計算方法は，逐次版と異なる。1990 年度の逐次版碁世代では，群ダメ点の条件が少し緩めであった。

- 強度 (群データ)

  並列版の群データにおける強度の計算方法は，逐次版と異なる。1990 年度の逐次版碁世代では，先着と後着の違いなどを考慮していなかった。また，群の分類による補正もない。

- 両ノゾキ，特殊弱結線 (結線データ)

  並列版の結線データにおいて，両ノゾキ・特殊弱結線という属性はない。これらは，1991 年度以降に設けられた概念である。

2. 局所探索

並列マシンの計算パワーを活用するため，逐次版より探索の種類を増やした。以下に，逐次版 (3.4章) と異なる主な処理を列挙する。

- 捕獲探索

  対象となる連はダメ 4 以下 (逐次版ではダメ 3 以下) とする。

- 死活探索

  並列版には死活探索はない。1990 年度の逐次版では，フトコロの拡大／縮小による群の死活候補手がなかったためである。

- 眼数探索

  並列版に眼数探索はあるが，実験解析モードでのみ実行が可能である。

- 攻め合い探索

  並列版には攻め合い探索はない。1990 年度の逐次版では，群の攻め合い候補手がなかったためである。

- 詰碁探索

  並列版では詰碁探索による死活候補手が存在する。1990 年度の逐次版では，詰碁探索は実験解析モードからのみ実行が可能であったが，並列マシンの計算パワーを活用するため，並列版では詰碁探索を取り入れている。

  なお，対象は完全包囲かつ中立かつ中地が 10 以下の群。攻め側，守り側の両方の立場から探索を行ない，その結果より以下の表のように分類する。

攻め側＼守り側	生き	劫 (取り番)	劫	死に
生き，不明	生き	同左	同左	同左
劫 (取り番)	中立 3	劫	同左	同左
劫	中立 2	中立 3	劫	同左
死に	中立 1	中立 2	中立 3	死に

4.3.2 候補手生成と評価

人間は与えられた問題解決の場で判断の基準として，過去に遭遇した典型的状況に関する知識に基づき行動している。即ち，方針を決定する着眼点の選出は，碁の知識によって設定されたいくつかのケース (典型的状況) の観点から行なわれる。「碁世代」ではこの典型的状況についてその特徴を定義し，それにより現在どの様な典型的状況が生じているか判定出来るようにした。そして，その典型的状況が検出されたとき，これを処理する手段として候補手を用意した。碁世代では，局面認識の処理によって得られたいろいろな対象の状態に応じて表 4.3-1 の候補手知識から候補手が生成される。

表 4.3-1 : 候補手知識

名称	内容
定石	序盤，隅における標準的な手筋 (239 型)
辺点	ヒラキ・ツメ・割り打ち等の辺上の手
打ち込み	打ち込みと防御
捕獲／逃亡	連を取る／逃げる
多重標的連捕獲／逃亡	弱い連のうちどれか一つを取る／逃げる
詰碁	群を取る／生きる
ダメ点	ハネ・ノビ・オシ等の石の競り合いの手
結線の切断／連結	石をつなぐ / 切る手
群の包囲／脱出	群を包囲する／逃げる手
フトコロ拡大／縮小	領域を増やす / 減らす手
死活，攻め合い	中手の手，攻め合いの手
地模様	模様を地にする
模様接点	模様の拡大／縮小

ほとんどの候補手は，1990 年度逐次版碁世代と同じである。それらの候補手の位置，評価値については文献 [実近 91-2] を参照のこと。ただし，並列版では族の分離／連絡候補手は生成しない。

以下に，1990 年度逐次版碁世代と異なる候補手について記述する。

1. 多重標的連捕獲／逃亡

   - 候補手位置
     多重捕獲探索によってみつかった，中立の連を取る／逃げる手の位置
   - 評価値
     固定点で 20 点

2. 詰碁

   - 候補手位置
     詰碁探索によってみつかった，中立の群を取る／生きる手の位置
   - 評価値
     探索結果より求まった分類から，死活候補手の評価値をもとに以下のように算出する。

- 「生き」「劫」「死に」 → 0 点
- 「中立 1」→ 1 × 死活評価値
- 「中立 2」→ (2/3) × 死活評価値
- 「中立 3」→ (1/3) × 死活評価値

4.3.3 着手決定

候補手は，各知識が互いに独立に挙げてくるため，異種ケース間の不調和を局所的に調整してより効果を高めたり，不調和を削除する。このような候補手間の調整を行った後で，評価値の合計が最も高い点に打着する。

列挙された候補手より以下の手順で最終的に着手を決定する。

1. 列挙された候補手に，連打捕獲ボーナス点による評価値の補正を行なう。
2. 各候補手に定石フィルターをかける。
3. 盤面上の各点に対し，候補手の評価値を合計し，もっとも高い点を選ぶ。
4. 自殺手フィルターをかける。

- 連打捕獲ボーナス点

  生きている連に対して，捕獲側が 2 手連打すると死ぬ連について，連打候補手を出す。連打すると捕獲できるので，捕獲側から見ると，「効き」を打ったことになる。

  - 評価値の補正

    候補手評価値を 2 倍する

- 定石フィルター，自殺手フィルター

  候補手は，各知識が互いに独立に，ある目的を達成する手を挙げてくるため，他の視点から見ると不都合な点に生成されることがある。このような手を取り除くのがフィルターである。

  補正の仕方は 3.6.4 章と同じ。

4.4 並列処理

4.4.1 並列性

「碁世代」の処理のうち，各処理段階で盤面上に多数発生し，独立に実行可能なものがあり，それらは並列に実行可能である。以下に，囲碁プログラムにおいて並列性のあると思われる部分を示す。

- 局面認識における各オブジェクト (連，群，結線) 毎の認識作業
- 局所探索における各オブジェクトの探索。また，その探索自体の並列実行
- 各種の候補手知識毎の候補手列挙と評価値の計算

4.4.2 並列実行

1. 並列処理の対象とした処理

   並列に実行可能な部分の処理の粒が小さいと，通信コストの割合が高くなり，並列処理の効果がでない。そこで本プログラムでは，局面認識・候補手生成の段階において大きな粒の処理である，各種の探索処理を並列に実行させた (表 4.4-1)。

表 4.4-1 : 並列実行する処理

名称	内容
連の捕獲／逃亡	連を取る／逃げる
連打捕獲	捕獲側が連打したときの連の生死
結線の切断／連結	石をつなぐ / 切る手
多重標的連捕獲／逃亡	弱い連のうちどれか一つを取る／逃げる
詰碁	群を取る／生きる

2. 構成

   並列版「碁世代」は，1 つのマネージャプロセスと，多数のワーカプロセスとから構成される。マネージャプロセスは，ワーカプロセスで実行される仕事を作り出すこと，実行された結果から次の着手を決める処理を行う。ワーカプロセスは，マネージャプロセスによって渡された仕事の処理を行う。そのため，各プロセスには碁盤が用意されている。

   プロセッサへの割付けは，マルチ PSI のプロセッサのうちの一つにマネージャプロセスを置き，残りのプロセッサにはワーカプロセスを一つずつ配置した。

3. 負荷分散

   並列プログラムの実行においては，プロセッサ間の負荷を均等にすることが重要である。

   並列版碁世代では，暇なプロセッサを見つけ出して，その暇なプロセッサに対してのみ仕事を送りつける動的負荷分散方式を採用した。これにより均等な負荷分散を実現した。

4. 処理の進み方

   システムが相手の着手を入力すると，まず局面の認識を行なう。この時，処理を粒の大きなもの (連の捕獲探索など) と小さいもの (点データの更新) に分ける。粒の小さな処理は全

てのプロセッサが行ない，粒の大きな処理はワーカープロセッサに割り振り並列に実行させる。実行結果は，マネージャプロセッサの持つ碁盤に登録する。

候補手の生成は，マネージャプロセスでのみ実行する。大きな粒の候補手生成処理は，ワーカープロセッサに割り振られ，結果をマネージャプロセッサの持つ碁盤に登録する。

そして，結果をもとに次の候補手を決定する (図 4.4–1)。

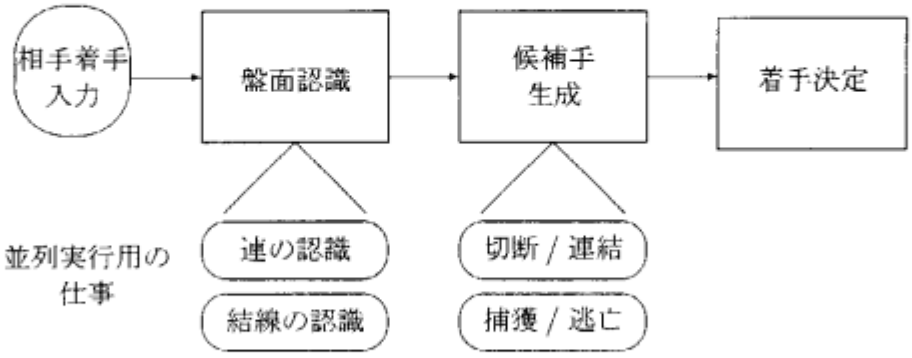


図 4.4 1：並列版「碁世代」における処理の流れ

4.4.3 遊軍処理

囲碁システムは，人間との対局を行なうため，短い時間内に着手を決めなければならないリアルタイム性を要求されるシステムである。我々は，このリアルタイム性を保持しながらシステムを強くすることを目的とした並列知識処理の一つのアプローチとして，「遊軍処理方式」を提案した。

遊軍処理方式とは，強くするためには重要ではあるが，次の着手には間に合わなくてもよい程度の処理を見つけ出し，それに低いプライオリティを付け，重要かつ処理の小さな処理には高いプライオリティを付けてプロセッサに実行させるものである。我々は，前者の仕事を実行するプロセス群を本軍と呼び，後者の仕事を実行するプロセス群を遊軍と名付けた。そして，本軍の処理結果と，本軍の仕事がすべて終った時点までに終っている遊軍の仕事の結果から着手を決定する。

遊軍処理を取り入れたシステムは，本軍と呼ばれるプロセス群と遊軍と呼ばれるプロセス群からなる (図 4.4–2)。本軍，遊軍ともに，一つのマネージャと、マネージャから与えられた仕事を実行するワーカ群からなる。プロセッサ 1 番をマネージャプロセッサとし，本軍マネージャと遊軍マネージャが配置され，残りの各プロセッサには、本軍用ワーカプロセスと遊軍用ワーカプロセスが一つづつ配置される。

- 実行の様子

  着手を入力すると，本軍用マネージャプロセスは盤面認識や候補手列挙などの仕事を作り，本軍用ワーカプロセスに実行させる。遊軍で処理させる仕事は，遊軍用マネージャに送る。遊軍用マネージャは遊軍用の仕事を持っていないワーカプロセスに仕事を送る。ワーカプロセスは送られた仕事を終えると結果をマネージャプロセスに送る (ただし遊軍のプライオリティは低く，本軍用マネージャは全ての遊軍用ワーカプロセスの処理結果を待たないので，一般に遊軍用ワーカが仕事を受けとってから結果を返すまでの間に，何手か対局が進んでいる)。本軍プロセスは遊軍プロセスより高いプライオリティを持っているので，同じ

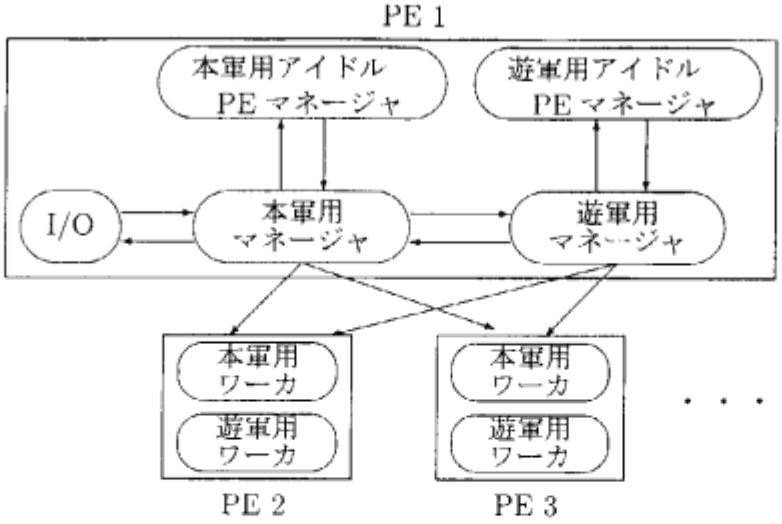


図 4.4-2：システム構成

プロセッサに本軍用の仕事と遊軍用の仕事が与えられた場合，本軍用の仕事が先に実行される。本軍用の全ての仕事が終わると，本軍のマネージャプロセスは，その時までに終了している遊軍用の仕事の実行結果を集め，その結果と本軍によって実行された処理結果から，次の着手を決める。着手決定処理においては遊軍プロセスに投げた仕事の結果を待たないので，次の着手を生み出す時間は，遊軍に実行させる仕事には影響を受けず，遊軍用の仕事が増えてもシステムのリアルタイム性は保たれる。

- 実行のキャンセル

遊軍は次の着手を決める処理 (本軍で行われる) とは独立に仕事を実行する。システムが着手した後，相手の考慮中も遊軍用ワーカプロセスは処理の実行を行う。また，遊軍が実行している仕事のもとになった盤面の部分が，対局が進むことによって変化し，実行中の遊軍の仕事が無意味になることがある。そのため，相手の着手を入力すると，遊軍用マネージャ内にあるまだ送っていない遊軍用の仕事が不要になるかどうかのチェックをする。また，既にワーカプロセスに送った仕事もチェックするために，各ワーカプロセスにも着手を伝える。

- 遊軍処理方式の利点

遊軍処理方式による利点は以下の通り。

- 遊軍プロセスは，次の着手を決める処理とは独立に行われ，本軍の実行を邪魔しない。

  各プロセッサは一つの本軍と遊軍プロセスを持ち，本軍のプロセスは遊軍のプロセスより高い優先順位を持つ。従って，遊軍は同じプロセッサ内の本軍プロセスの処理が終わった時に初めて動き出す。

  また，着手決定処理においては遊軍プロセスに投げた仕事の結果を待たない。故に，次の着手を生み出す時間は，遊軍に実行させる仕事には影響を受けず，遊軍用の仕事が増えてもシステムのリアルタイム性は保たれる。

  プロセッサが多いほど，より賢い手が生み出される。

  遊軍プロセスは，本軍プロセスの実行が終わったプロセッサ上でのみ動く。従って，システムが打着した後には本軍は暇になり遊軍が動き出す。また，並列マシンが多く

のプロセッサを持っていると，局面によっては本軍の仕事がプロセッサの数を下まわることもあり，本軍が割当てられないプロセッサが生じることもある。そのようなプロセッサでは，最初から遊軍の仕事が実行される。

以上のような場合には遊軍の仕事が多く実行されるので，多くの有用な結果が集まり，より賢い手を打つようになる。

4.5 評価

4.5.1 並列処理による効果

表 4.5–1に，並列実行による台数効果を示す。この表から，序盤 (30 手目)・中盤 (90 手目)・終盤 (180 手目) のいずれの局面においても，並列実行によって思考時間が短くなったことがわかる。

表 4.5-1：並列実行による台数効果 (単位: 倍)

	第 13 期棋聖戦挑戦手合第 1 局			第 13 期名人戦挑戦手合第 5 局		
局面	1 PE	4 PE	16 PE	1 PE	4 PE	16 PE
30 手目	1.0	3.3	5.1	1.0	3.2	5.4
90 手目	1.0	3.4	5.3	1.0	3.4	5.6
180 手目	1.0	3.7	7.5	1.0	3.6	5.9

4.5.2 遊軍処理による効果

1. 遊軍処理のオーバーヘッド

遊軍処理のない碁世代と，遊軍処理のある碁世代とを作り，遊軍処理がシステムのリアルタイム性を保つことの確認を行った。この測定は，1991 年度の並列版碁世代で行なったため，遊軍には詰碁ではなく，ダメ 4 の連の捕獲探索を行なわせている。表 4.5–2に，その結果を示す。表から，次の着手を決める時間は遊軍処理により増えていないことがわかる。

表 4.5–2：遊軍処理のない碁世代と，ある碁世代との実行時間の比較 (16 プロセッサ)

	第 13 期棋聖戦挑戦手合第 1 局			第 13 期名人戦挑戦手合第 5 局		
局面	30 手目	90 手目	180 手目	30 手目	90 手目	180 手目
遊軍なし	10.9 秒	12.4 秒	13.1 秒	5.59 秒	7.67 秒	11.7 秒
遊軍あり	10.9 秒	12.5 秒	13.7 秒	5.68 秒	7.72 秒	11.7 秒

2. 遊軍処理のリアルタイム性

図 4.5–1に，各プロセッサ毎の仕事の実行の様子を示す。各プロセッサにおける上の線は本軍の仕事の存在を，下の線は遊軍の仕事の存在を示し，縦線は石が盤に置かれた時 (白：人間, 黒：碁世代) を示す。プロセッサ 1 番にはマネージャプロセスが配置されている。これから，ある時には各プロセッサに遊軍と本軍の両方の仕事が存在することがわかる。また，遊軍は数手にまたがって存在し処理されていることがわかる。さらに，相手が考慮中にも遊軍が処理を行っていることも示されている。なお，この測定も 1991 年度の並列版碁世代で行なったため，遊軍には詰碁ではなく，ダメ 4 の連の捕獲探索を行なわせている。

3. 遊軍処理による棋力の向上

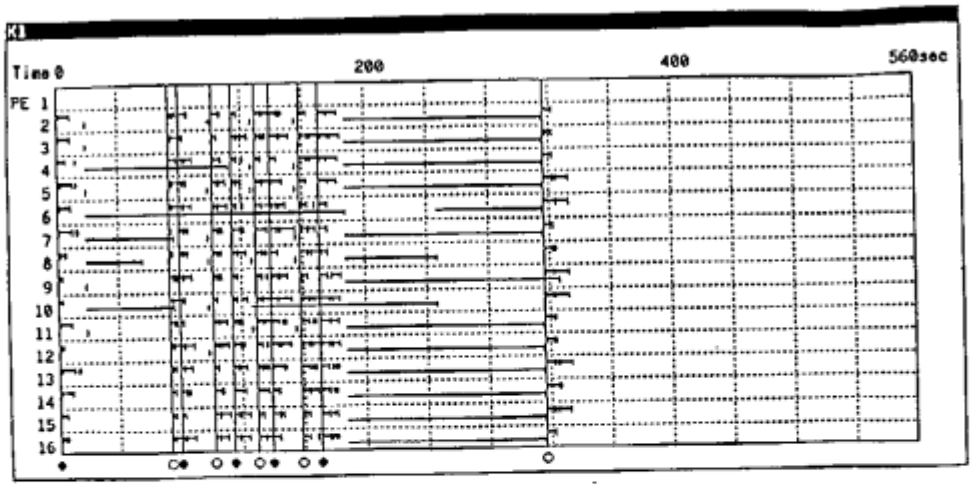


図 4.5–1：各プロセッサにおける実行ログ

碁世代の最も弱いと考えられる死活を遊軍処理の対象としてシステムを作成した。ここでは，死活を探索によって判定することにより，精度の高い判定を下すことを目的とした。

並列版碁世代では，局面認識の段階において大きな粒の処理である，一つながりの石 (連) の認識や，石同士の結び付きを示す結線の認識などを，候補手生成の段階においては，「切断／連結候補手」「捕獲／逃亡候補手」などを本軍にて並列実行させた。また遊軍では，群の死活候補手を並列実行させた (図 4.5–2)。群の死活候補手は，ある局面において沢山現れるものではないが，1 つの死活についての処理時間はとても大きいので，死活探索木の深さ 1 の枝毎にプロセッサに分けて処理している。

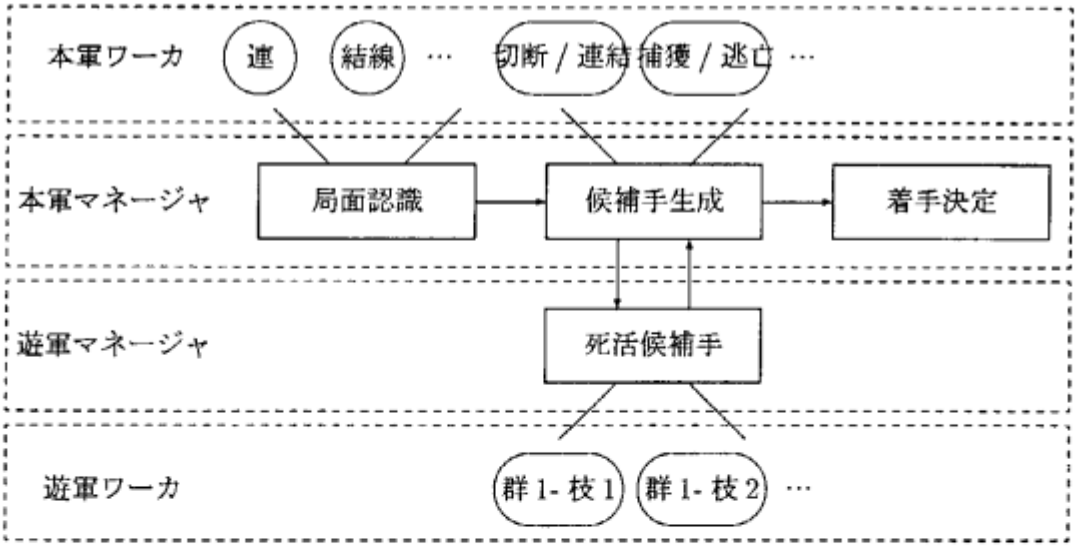
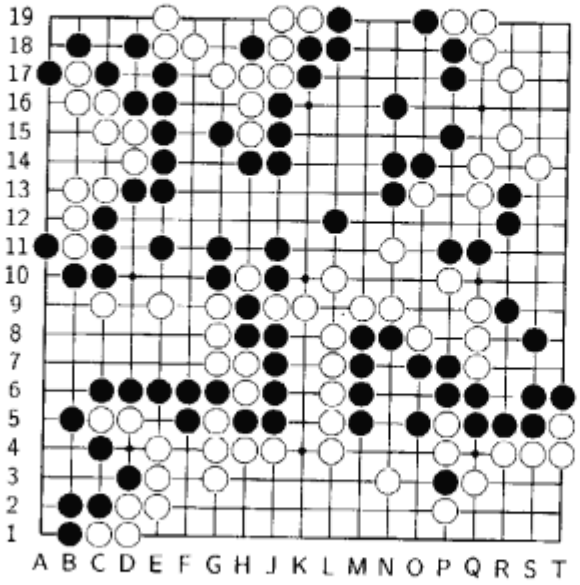


図 4.5–2：並列版碁世代の処理の流れ

並列版碁世代の遊軍に，弱点であった詰碁を処理させた結果，今まで候補手がでなかった大きい石の死活候補手が出るようになった (図 4.5–3)。

ただし一つの候補手を新しく採り入れたからといって，飛躍的な棋力の向上は望めない。また，この並列版碁世代は逐次版碁世代と試合をすると負けてしまう。これは，逐次版の方が，各種の知識の整備・チューニングにかけた時間が多かったからである。並列化によっ

て処理の実行が早くなったり，遊軍処理により効果的に新しい処理を取り込むことができるようになったが，囲碁に関しては，並列化よりも各種の処理のチューニングの方が，棋力の向上の早道のようだ。



死活候補手 1：
B-14, 評価値 36 点

死活候補手 2：
G-19, 評価値 34 点

図 4.5.3：遊軍によって新たに加わった死活候補手

5　現状の評価と今後の課題

逐次版碁世代および並列版碁世代の棋力，個々の知識の評価と課題については，それぞれの章に記した。ここでは，棋士システムプロジェクト全体の評価と今後の課題について述べる。

棋士システム「碁世代」の研究目標は，知識処理に関する研究と並列処理に関する研究である。知識処理に関するテーマは1.2章で述べた通り，探索，曖昧，例外，協調，学習である。並列処理に関するテーマは，囲碁に並列処理を応用すること(または並列実行を利用しての新しい方法を生み出すこと)，ICOT で開発された並列推論マシンの性能検証である。以下，それぞれについて成果と課題について記述する。

- 探索

  囲碁の着手生成において，意図(ゴール)を特定しない，グローバルな局面の評価関数による探索は，可能な手の組合せが多過ぎて実用的でない。そこで検査すべき候補手を，意図と状況に応じてできるだけ絞る努力を行なった。

  囲碁プログラムにおいて探索の用途は多様であり，単に着手決定目的の探索よりも，その前準備としての局面認識のための補助的手段として探索が用いられる場合が多い。従って，さまざまな意図別に探索ルーチンを設計した。このようなシステムでは，探索問題を切り出す機構，探索結果を活用する機構が，個々の探索ルーチンと直接に関連しており，全体として初めて有効に機能する。即ち，個々の探索ルーチン自体の効率を考えることも重要であるが，探索システム全体(探索ルーチン＋探索問題検出機構＋探索結果活用機構)としての効率の方がもっと重要である。このような認識のもとに，設計を行なった。

  探索システムの効率を高めるために，碁世代では，一貫して認識の「段階的深化」の原理に従っている。これは(広義の)探索の過程を多段階に分けて行ない，前段階の結果に基づき，次の段階の探索範囲を絞り込み，その代わり，前段階より深く探索を行なう。即ち，各段階での仕事量をできるだけ一定に保つことにより(利用可能な資源以下に抑えることにより)，探索における組合せ爆発を回避しようとする原理である。碁世代では，さまざまな探索ルーチンを組み込んだ一つの探索システムの中での「段階的深化」を実現した。特に，群の認識を含む局面認識過程は一つの探索システムとみなすことができ，この原理に従っている。また，碁世代では，着手生成の最も一般的な基準となる群の認識処理において，群の緊急度という属性を扱っている。これは，どの群は最も忙しい(早く手を入れるべき)かを数値的に表現するものである。碁世代は，この緊急度を使って，着手決定時に候補手を絞り込むことに成功した。

  あまりに厳格な階層型データ構造は，囲碁のように変化の多い可能性を探るゲームでは，表現効率が悪い。碁世代もこれを考慮して，比較的単純なデータ構造を目指したが，基本的には先読みを行なわない方針で設計したので(単独意図の局所探索は除く)，一般的な探索にはあまり適していない。

  対局中の群の認識では，相手の最終着手によって変化する盤面は，その手の近傍に限定されているため，局所的なデータ更新を行なっており，盤面全体を複数段階に分けて徐々に認識を精密に行なっていく本格的な段階的認識は行なっていない。しかし，現在の碁世代の群の認識の精度はまだまだ悪く，さらに精密に群を認識しなければならないと思われるので，局所的なデータ更新においても，段階的深化の方法をより徹底させる必要がある。

  今後の課題としては，以下のテーマが残されている。

  - 個別の探索ルーチンにも「段階的深化」の原理を導入し，多重意図を扱える探索ルー

チン等を増設するとともに，適用状況の分類により，各ルーチンの精度と効率の向上を図る。

- 適用状況を限定することにより，ある程度グローバルな局面評価の扱える探索を導入する。(例. 布石の手順探索，ヨセの手順探索，多重ケースの融合，異種ケースの融合)

- 曖昧

碁世代では，包囲度，模様の大きさ，厚み，群の強さなど，様々な種類の抽象的な概念を全て数値で表現している。これらの属性値は単位はバラバラであるが，これらを組み合わせて，最終的に候補手の評価値を求めている。その評価値の単位は目数に統一した。

単位の異なるいくつかの属性値から，新しい抽象概念の属性値を決定する「式」を求めるためには，人間の間隔と経験に頼る他はない。そして，その変換式の正しさを確かめるために，多くの実例について，人間の判断を適用し，チューニングする方法をとった。このような経験的方法は，伝統的な数学的分析的方法と比べると不確実であり，万人を納得させる根拠に欠けているようであるが，本来人間の曖昧な感覚に属する領域であり，これに厳密さを要求することは無意味に思える。注意すべきことは，これらの抽象概念に附属する数量化された属性値は，その絶対値はあまり重要ではなく，同じ単位の属性値について比較した時の大小関係が重要なことである。チューニングにおいても，このことを考慮して行なうべきである。

また，抽象概念の中には数量化されない(またはする必要のない)ものも含まれる。表現方法としては，ある性質を満たしている対象とか，あるパターンに整合した対象とかである。碁世代では，これらの抽象概念を部分的に扱っているが，まだ不十分である。また，不完全な知識のまま行動したり，判断したりするという曖昧思考は実現されていない。

今後の課題としては，以下のテーマが残されている。

- 抽象レベルの意図で作戦を立て，これを探索で具体化する方式を実現する。
- 自動チューニング方式を確立する。

- 例外

碁世代にとって例外事象または非常事態とは，局面状況が用意された問題ケースのいずれともうまく整合しない場合や，形勢(局所的，大局的)が大きく変動した場合を言う。現在の碁世代は，作業時間的に余裕がなく，手が回らなかったという理由で，特に例外対策を行なっていない。しかし，碁世代に知的な振舞いを期待するには，この例外対策がかなり大きな比重を占めることになるであろうと思われる。

今後の課題としては，以下のテーマが残されている。

- 問題ケースの種類を増設するとともに，局面状況と問題ケースとに一致度から例外ケースを設定する基準を設ける。
- 大局的形勢 → 形勢判断に応じた着手決定基準を設ける。
- 局所的形勢 → 強制手の連続する先読みを導入する。

- 協調

碁においては，いくつかの味(可能性)を見ながら相手の出方に応じて手段を作って行く場合に，特にこの能力が発揮される。しかし現在の碁世代では，協調に関して検討はしたが，作業時間的に余裕がないこと，個々の候補手知識の整備が先と判断されたことから，まだ実装には至っていない。

- 学習

  手筋エディタや定石エディタなど，知識獲得のためのツールを用意し，エキスパートにより与えられた知識を記憶する「記憶学習」について実装した。しかし，システム自身が主体的に行なう「自動学習」についての処理は行なっていない。まだ検討段階であるが，この必要性は高いと思われる。

- 並列処理と囲碁

  並列マシンを利用することにより，多くの処理が取り入れることができ，そのうえ実行時間が増えないという実験結果から，囲碁に並列処理が利用可能であることが実証された。また，我々は「遊軍処理」という新しい手法を提案し，実験した。しかし，この手法が囲碁のプレイに関して，どれだけ有効か，また他の分野にどれだけ適応できるかについてはこれから議論していかねばならない。

- 並列推論マシンの性能検証

  何をもって性能検証とするか難しいが，囲碁という大規模な知識処理システムが，人間との対局に耐えうる時間内に着手を生成できることから，ICOT が開発した並列推論マシンが実用的なものでると言えるのではないだろうか。

我々がたてた囲碁を打つモデルを全て実現するには至っていない。また，我々はアマチュア高段者の知識を基に碁世代を開発してきたが，まだ人間の囲碁を打つ思考方法についてすべて解明したとは言えない。今後は，まず我々のたてたモデルの実現を目指さなければならない。また我々は，これまで作成してきて，チューニングの重要性をあらためて感じている。

碁世代はまだまだ発展途上のシステムである。

謝辞

本研究の機会をいただいた ICOT の渕一博 所長，内田俊一 研究部長，新田克巳 第 2 研究室室長，横井俊夫 元第 2 研究室室長，吉川貞行 元第 2 研究室室長，吉岡勉 元第 2 研究室室長に感謝します。貴重なるアドバイスを頂いた CGS-WG のメンバーの諸氏に感謝します。並列版碁世代の研究開発に助言を頂いた元第 7 研究室の市吉伸行氏，瀧和男 元第 1 研究室室長，元第 4 研究室の宮崎敏彦氏に感謝します。逐次版碁世代のプログラム作成の一部を担当して頂いた CGS-TG のメンバーの諸氏に感謝します。

文献

[沖 89] 沖 廣明 他：マルチ PSI における詰め碁プログラムの実現と評価, JSPP'89, pp. 351-357, 1989

[実近 88] 実近 憲昭 他：囲碁システム「碁世代の方法」,ICOT-TM 618, 1988

[実近 90] Sanechika,N. : "Go Generation" A Go Playing System, ICOT Technical Report TR-545, 1990

[実近 91-1] Sanechika,N. et al. : Methodology of GOSEDAI, Proceedings of Game Playing System Workshop, pp. 10-14, 1991

[実近 91-2] Sanechika,N., Sei,S., Akaosugi,T., and Taki,K. : The Specifications of "GO Generation", Proceedings of Game Playing System Workshop, pp. 73-155, 1991

[清 91-1] 清 慎一, 沖 廣明, 瀧 和男 : 囲碁対局システム・並列版『碁世代』の試作, LPC'91, pp. 63-71, 1991

[清 91-2] Sei,S., Oki,H., Sanechika,N., Akaosugi,T., and Taki,K. : Experimental Version of Parallel Computer Go-Playing System "GOG", Proceedings of Game Playing System Workshop, pp. 31-37, 1991

[中島 87] Nakashima,H. and Nakajima,K. : Hardware Architecture of the Sequential Inference Machine PSI-II, Proceedings of 4th Symp. on Logic Programming, pp. 104-113, 1987

[瀧 88] Taki,K. : The parallel software research and development tool: Multi-PSI system, Programming of Future Generation Computers, North-Holland, pp. 411-426, 1988

A 1991 Game Playing System Workshop の棋譜

1991 年 11 月 18 日～11 月 20 日に，ICOT(東京) で行なわれた，GAME PLAYING SYSTEM WORKSHOP(棋士システムワークショップ) における囲碁プログラムトーナメントにおける主な棋譜である。

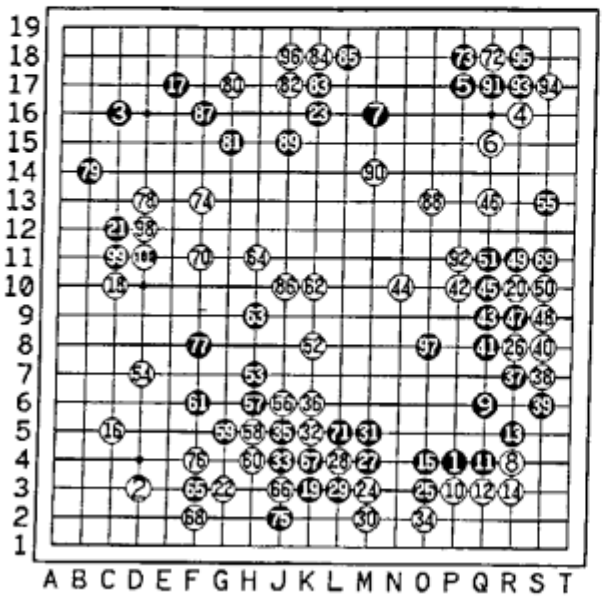
表 A.0–1 : 1991 International Computer Go Congress

Rank	Program	Authors	Country	Result
1	Goliath	Mark Boon	Netherlands	5-0
2	Go Intellect	Ken-Hsun Chen	USA	4-1
3	Star of Poland	Janusz Kraszek	Poland	3-2
4	IGO3	Noriaki Sanechika	Japan	2-3
5	Nemesis	Bruce Wilcox	USA	2-3
6	GOG (碁世代)	ICOT	Japan	2-3
7	Dragon	Shun-Chin Hsu	Taiwan	1-4
8	Explorer	Martin Mueller	Switzerland	1-4

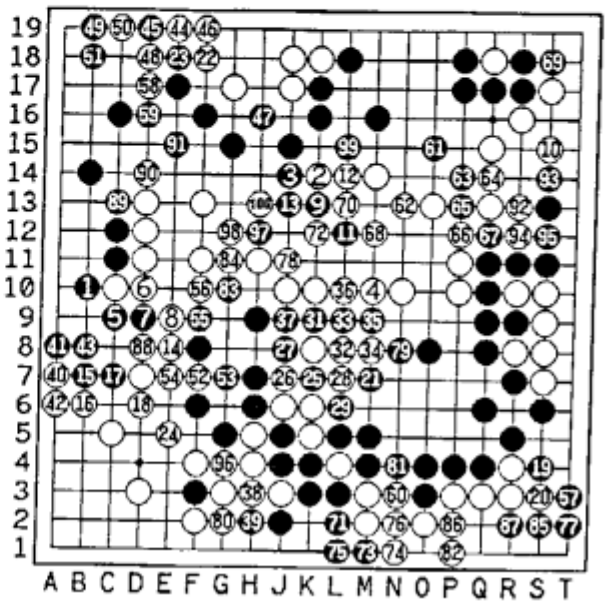
Black : Goliath (winner)

White : GOG

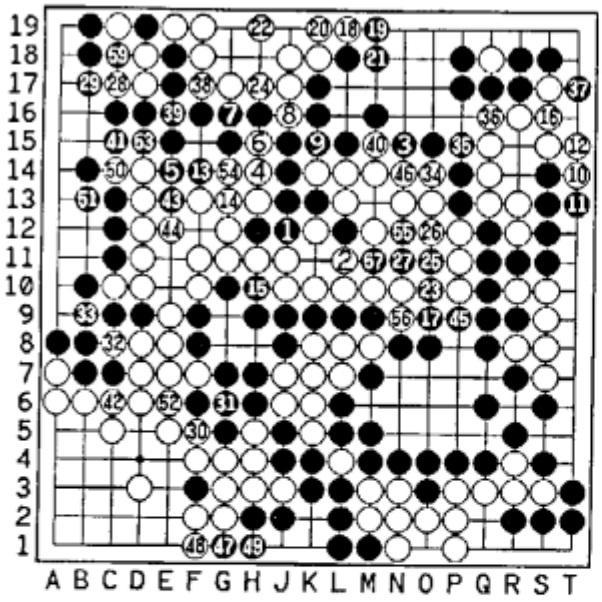
1 ~ 100



101 ~ 201



201 ~ end

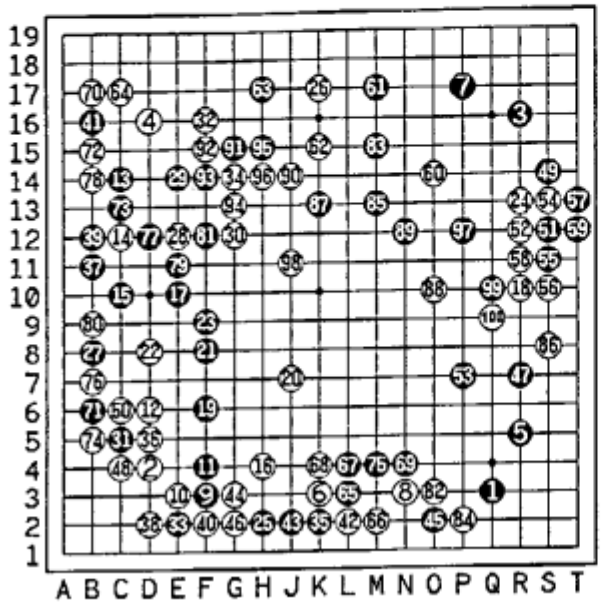


130 (K-7)
258 (pass)
260 (D-19)

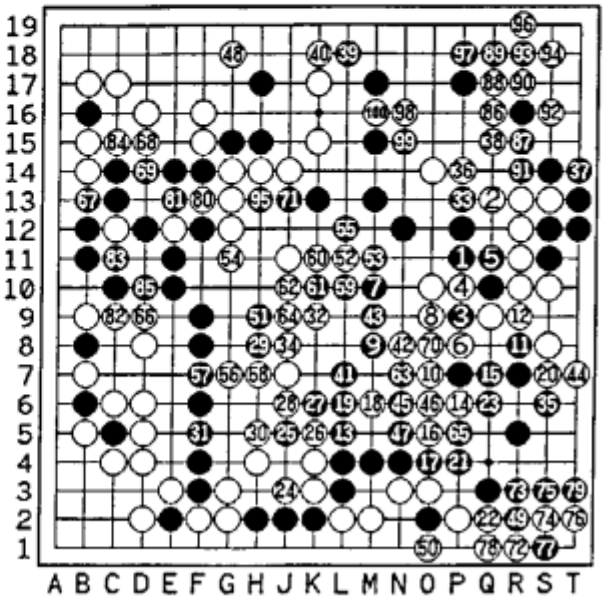
Black : GOG

White : IGO3 (winner)

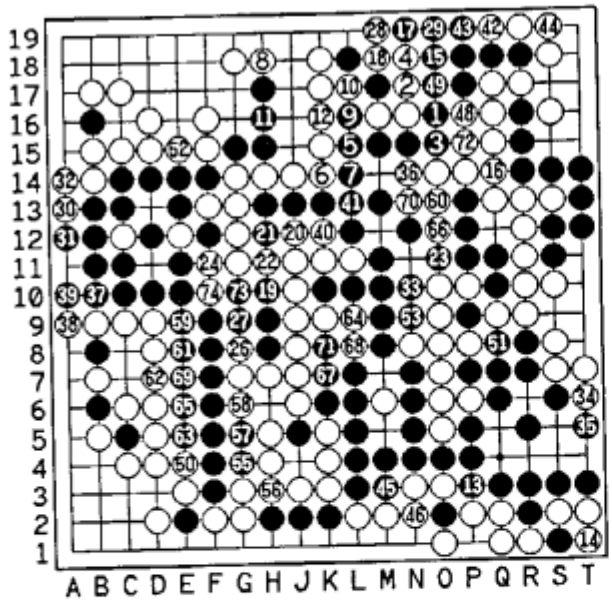
1 ~ 100



101 ~ 201



201 ~ end

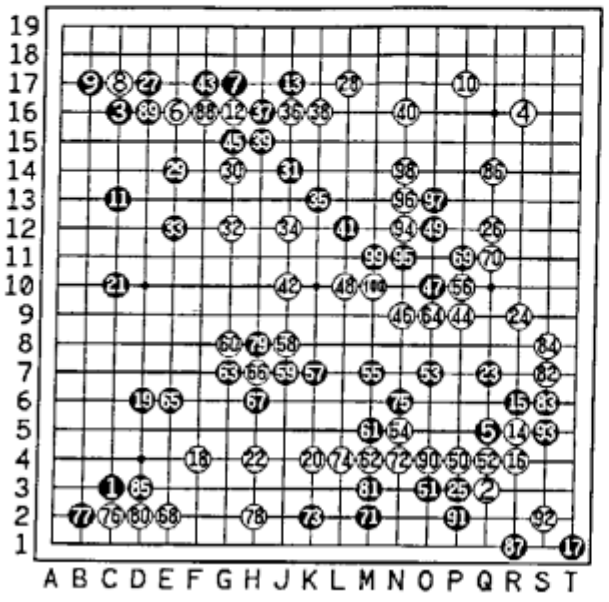


225 (E-12)
247 (S-1)
254 (P-9)

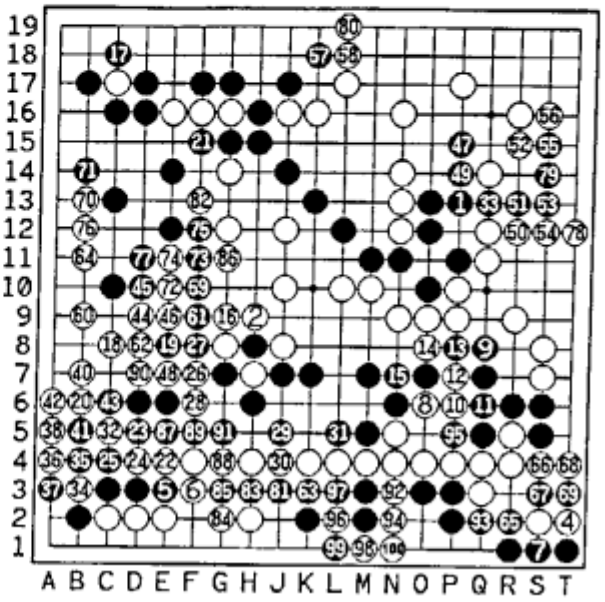
Black : Nemesis (winner)

White : GOG

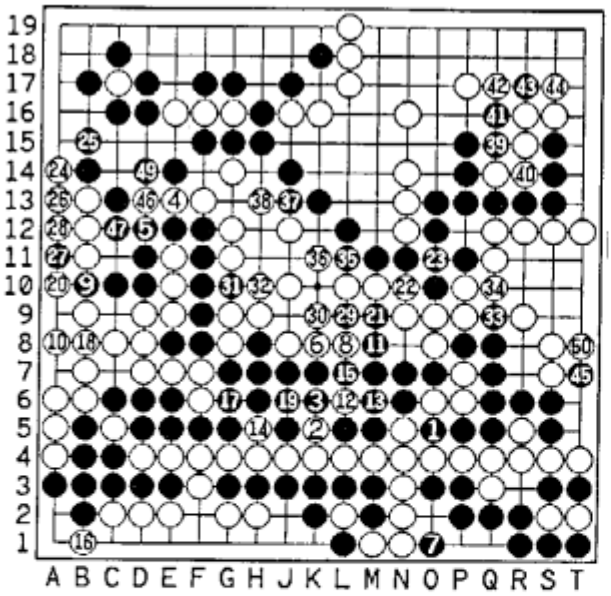
1 ~ 100



101 ~ 201



201 ~ end

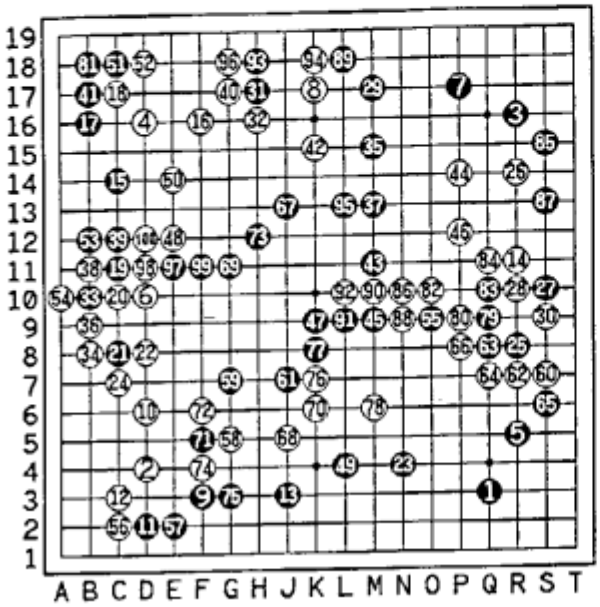


103 (H-7)
139 (B-3)
248 (A-11)

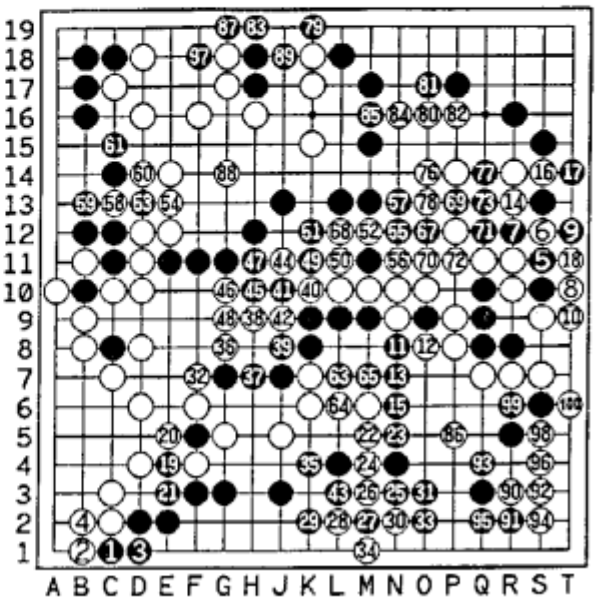
Black : GOG (winner)

White : Dragon

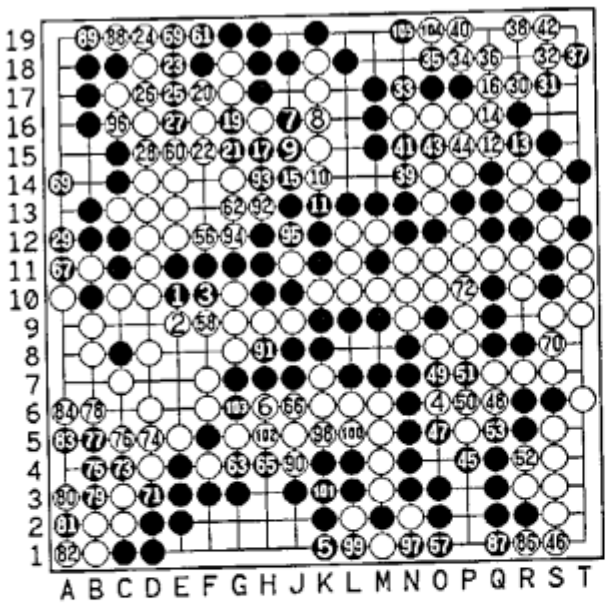
1 ~ 100



101 ~ 201



201 ~ end

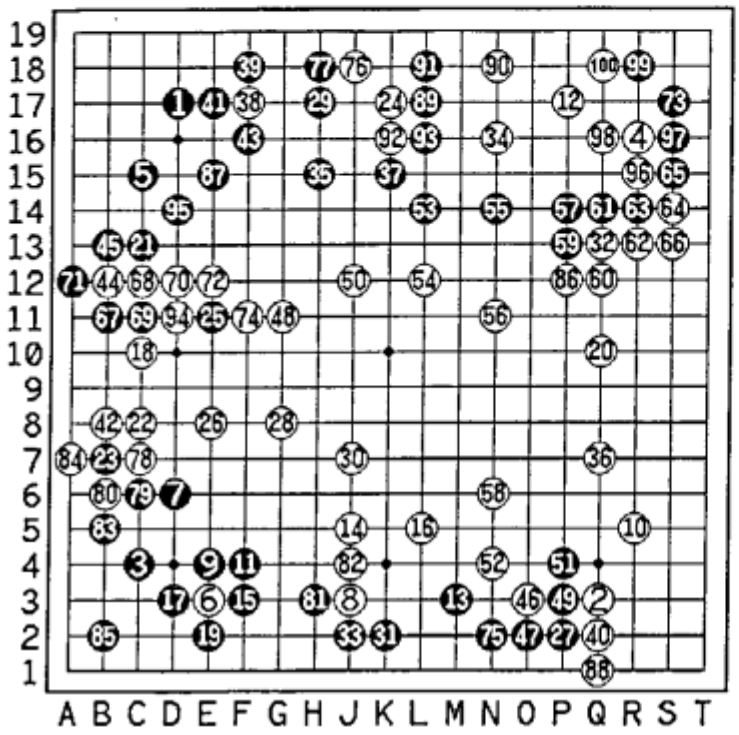


162 (C-13)
166 (D-13)
174 (S-12)
175 (S-11)
218 (M-2)
254 (S-10)
255 (S-12)
264 (F-5)
268 (B-10)
285 (A-2)

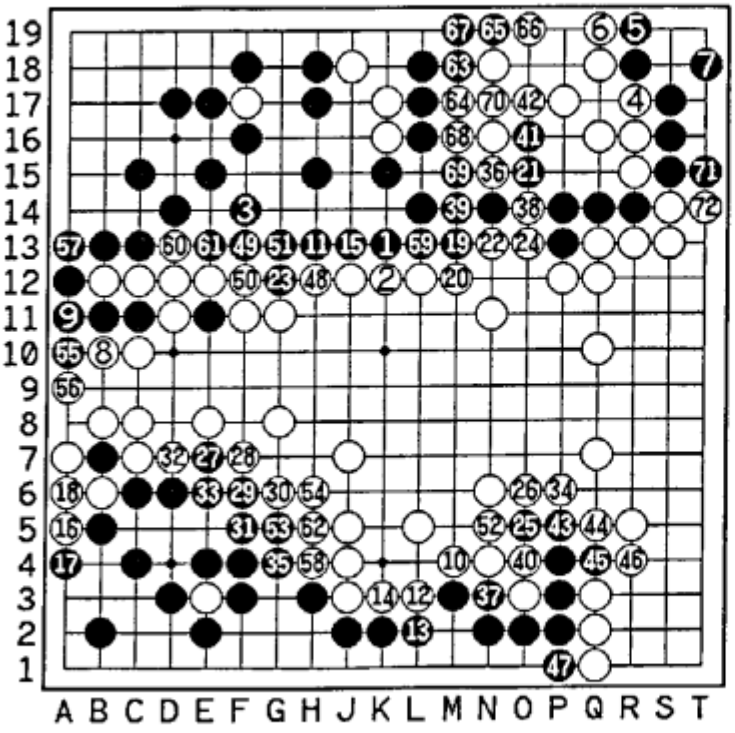
Black : Explorer

White : GOG (winner)

1 ~ 100



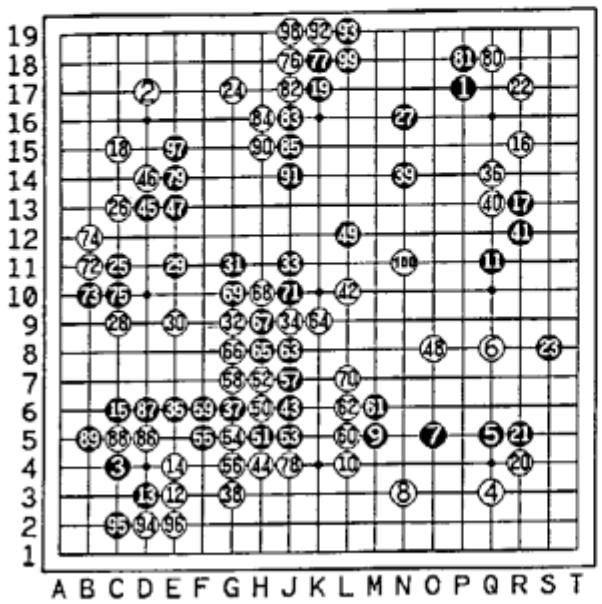
101 ~ end



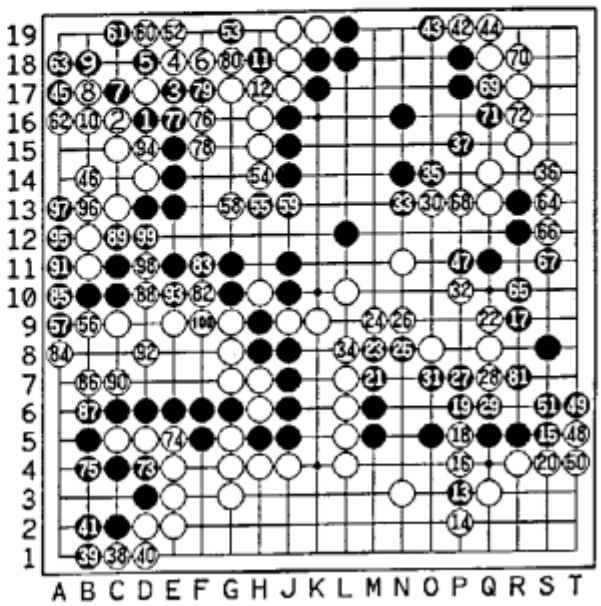
Black : Goliath (winner)

White : Go Intellect

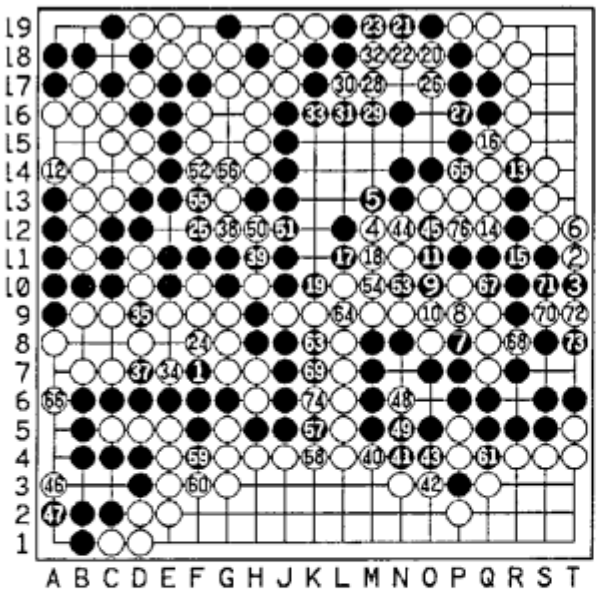
1 ~ 100



101 ~ 201



201 ~ end

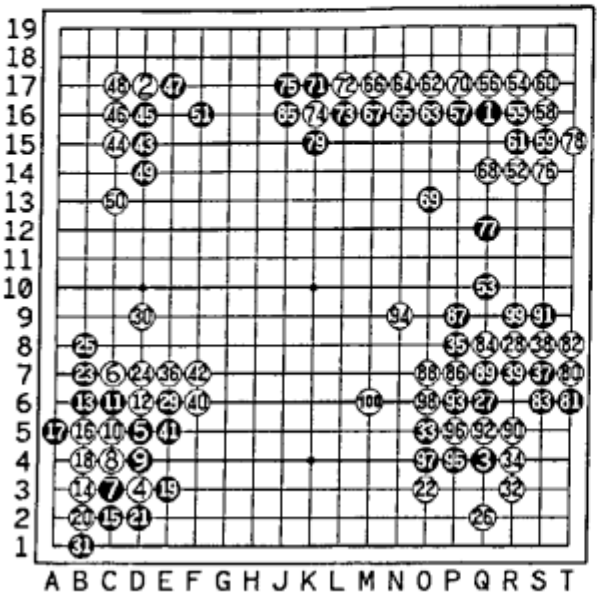


236 (D-10)
262 (P-3)
275 (H-10)

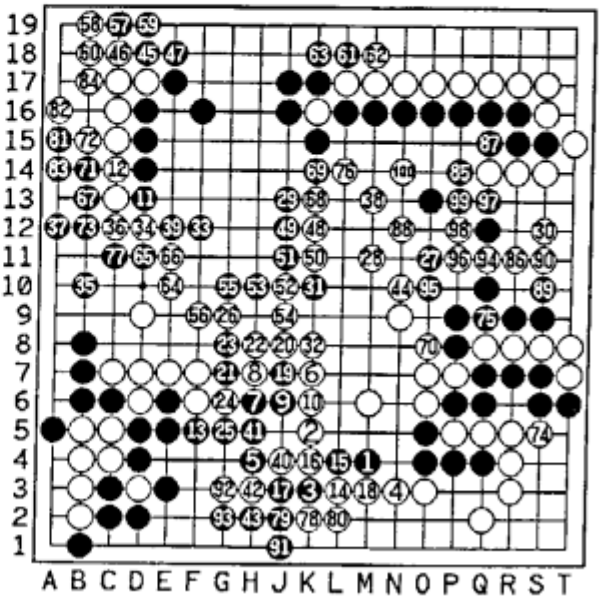
Black : IGO3

White : Goliath (winner)

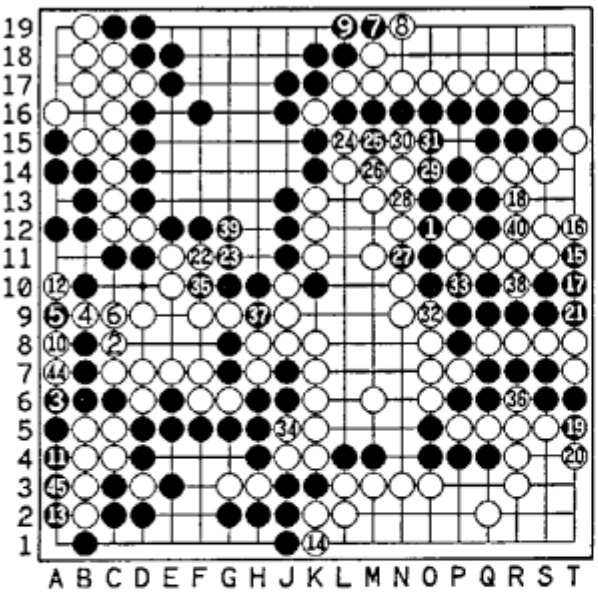
1 ～ 100



101 ～ 201



201 ～ end

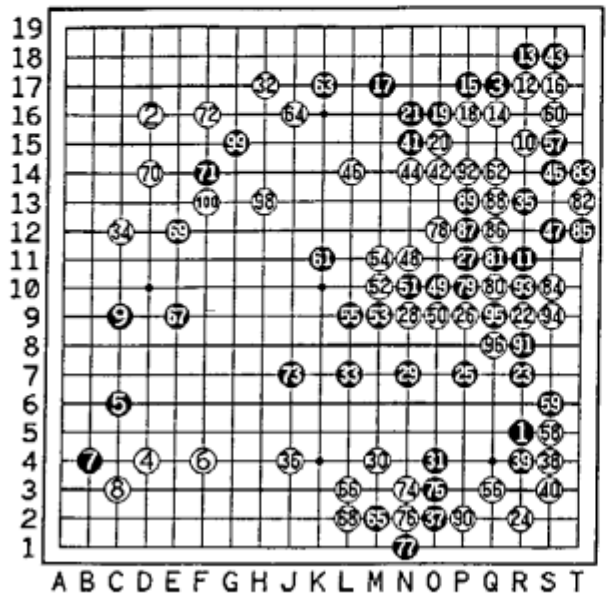


241 (pass)
242 (A-9)
243 (pass)

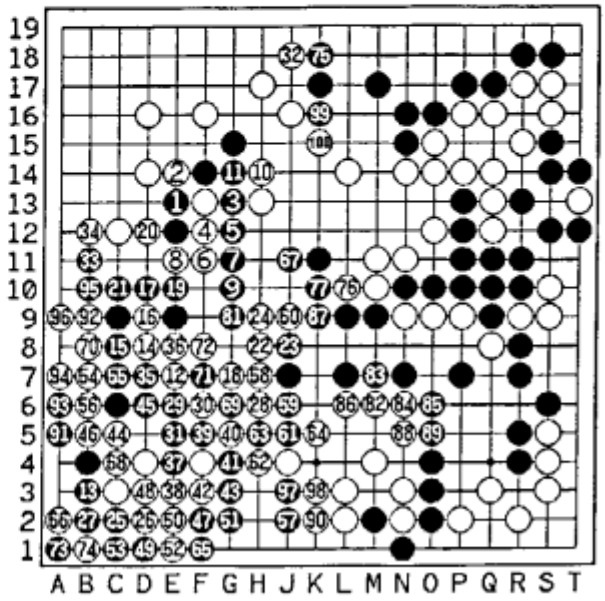
Black : Goliath (winner)

White : Dragon

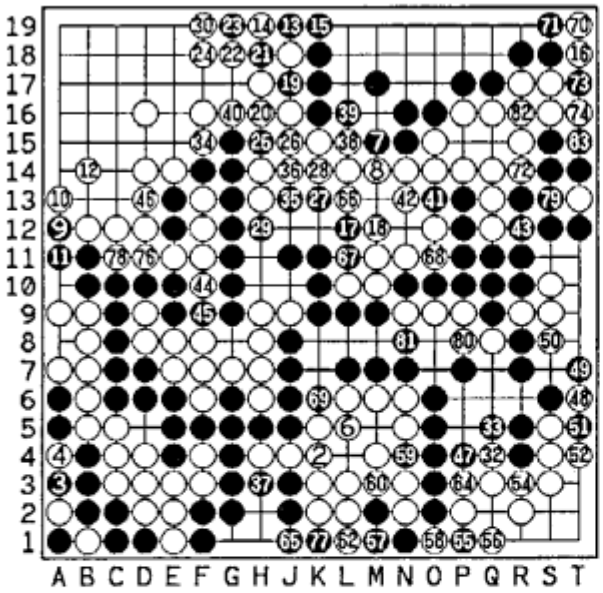
1 ~ 100



101 ~ 201



201 ~ end

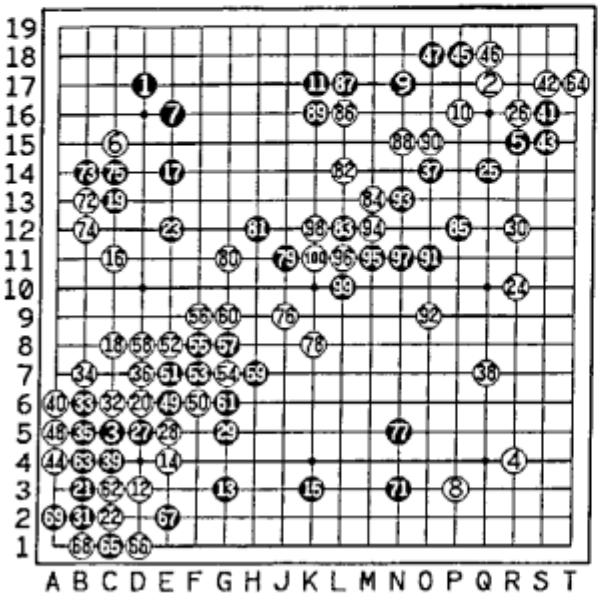


97 (Q-10)
178 (F-6)
179 (G-5)
180 (F-7)
201 (A-1)
205 (A-5)
231 (H-19)
253 (T-6)
261 (P-1)
263 (O-1)
275 (T-18)

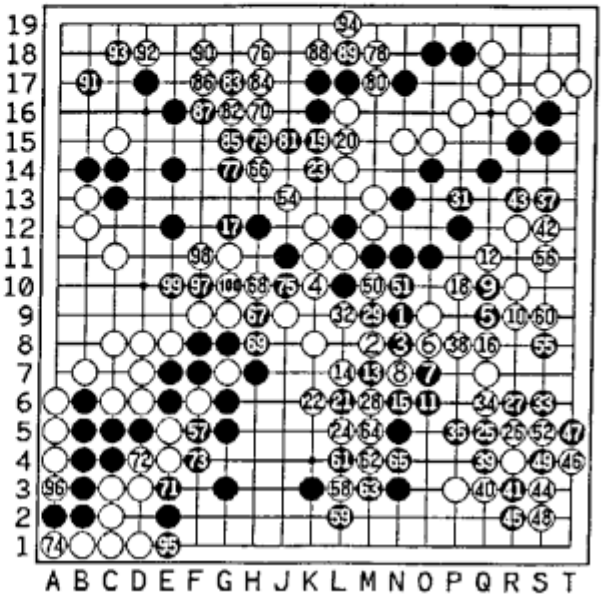
Black : Star of Poland

White : Go Intellect (winner)

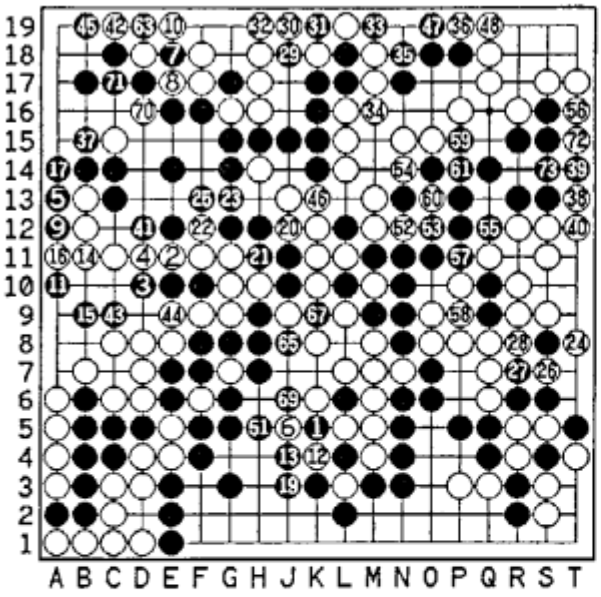
1 ~ 100



101 ~ 201



201 ~ end

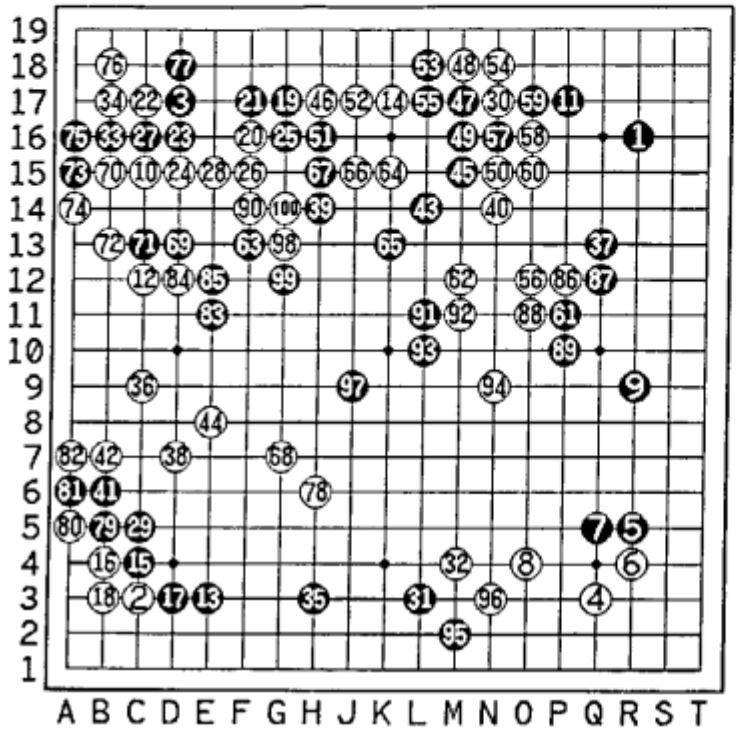


70 (C-1)
130 (N-7)
136 (M-7)
153 (S-4)
218 (L-3)
249 (L-4)
250 (K-5)
262 (N-13)
264 (E-18)
266 (L-3)
268 (L-10)

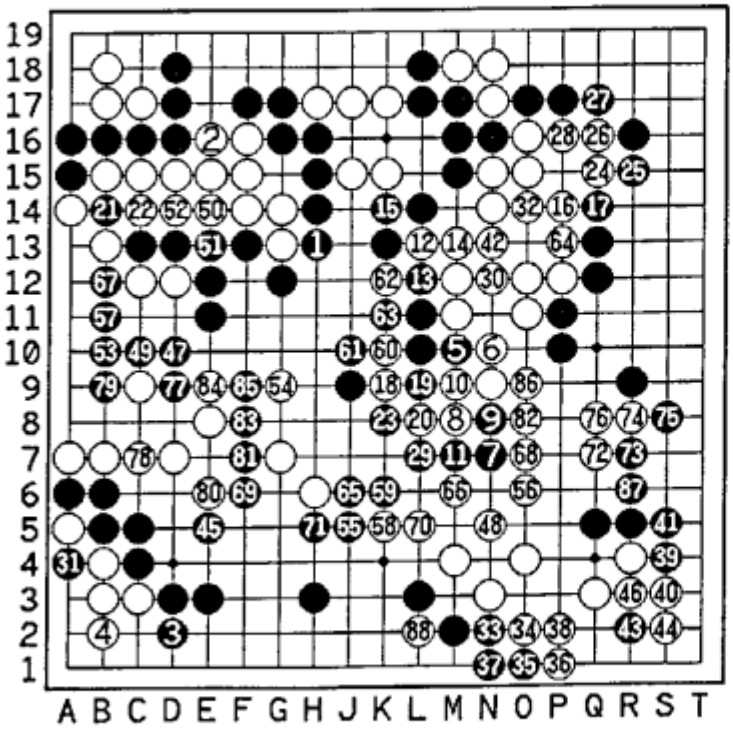
Black : Go Intellect (winner)

White : Nemesis

1 ~ 100



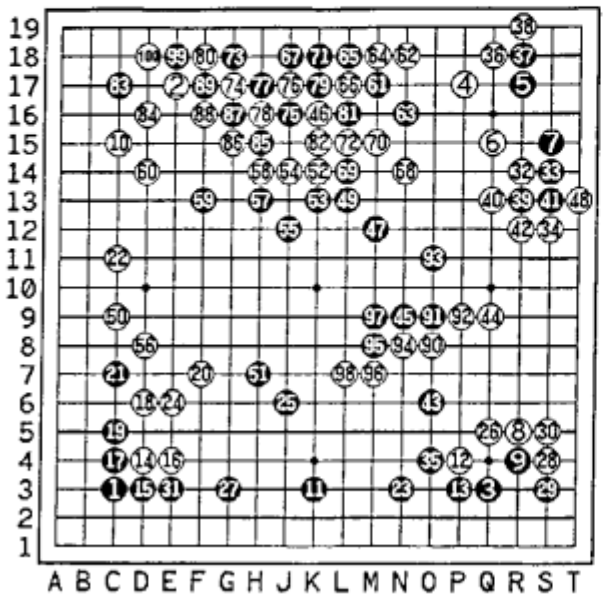
101 ~ end



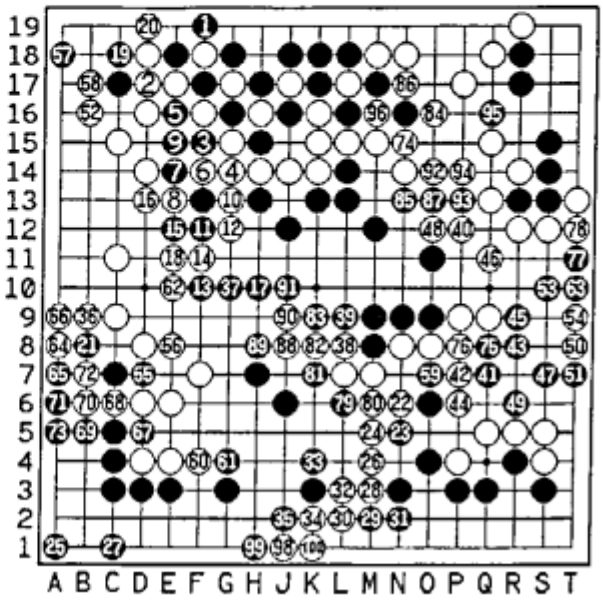
Black : Nemesis

White : Star of Poland (winner)

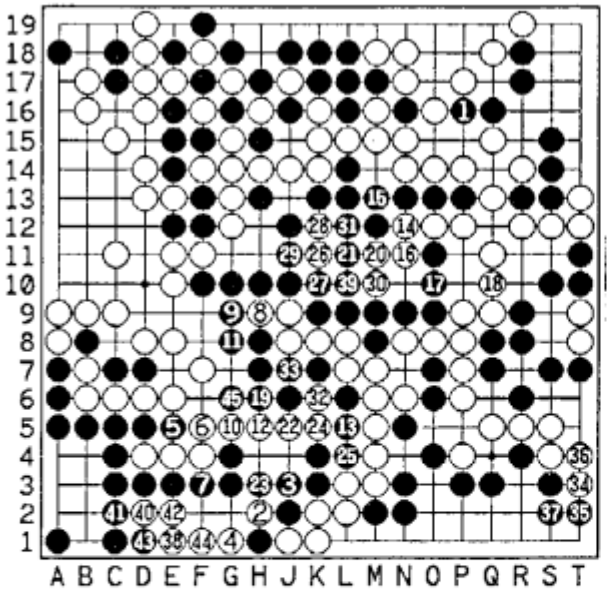
1 ~ 100



101 ~ 201



201 ~ end

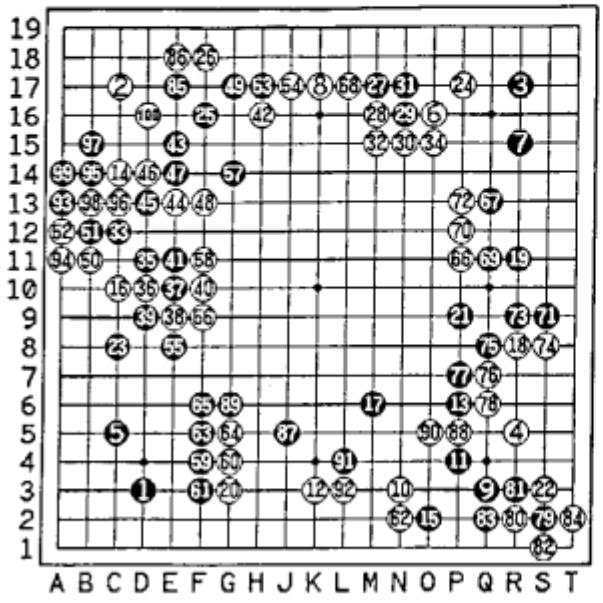


197 (L-17)

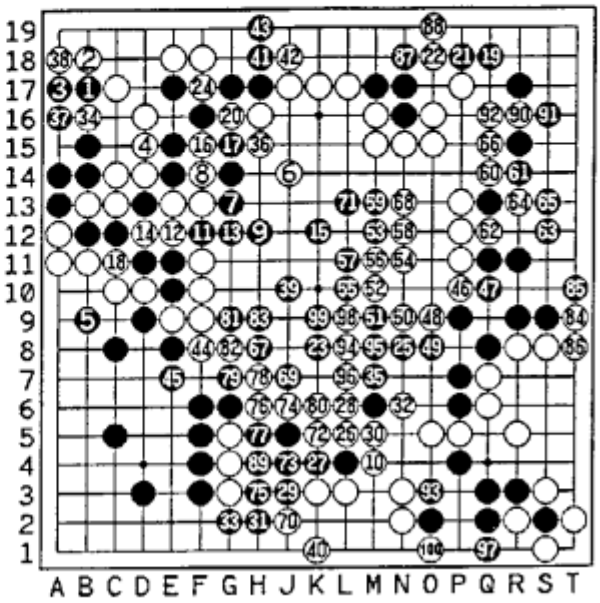
Black : Explorer

White : Star of Poland (winner)

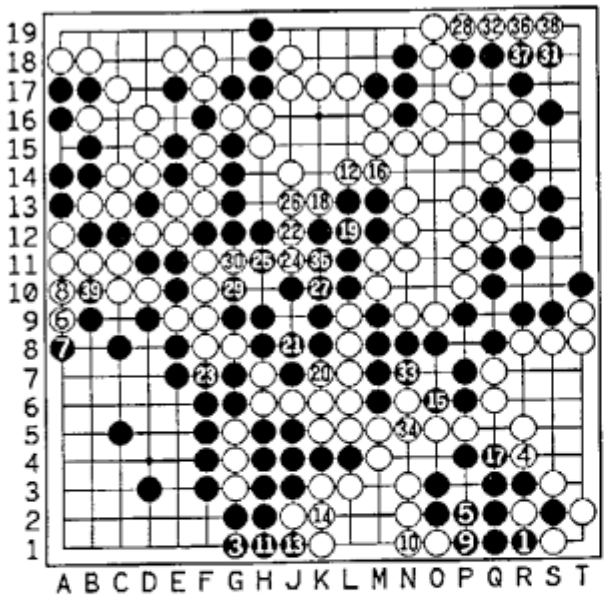
1 ~ 100



101 ~ 201



201 ~ end

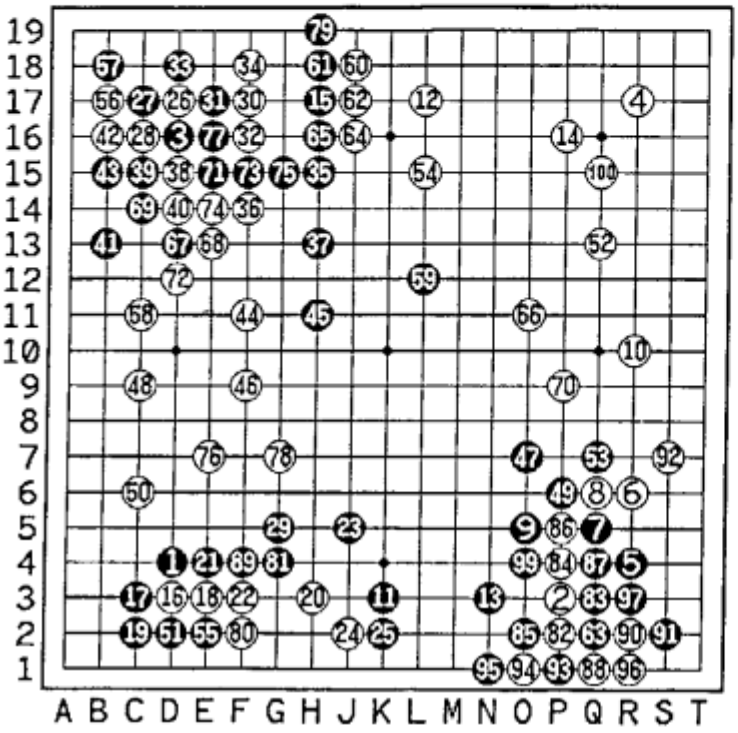


202 (S-2)

Black : Explorer (winner)

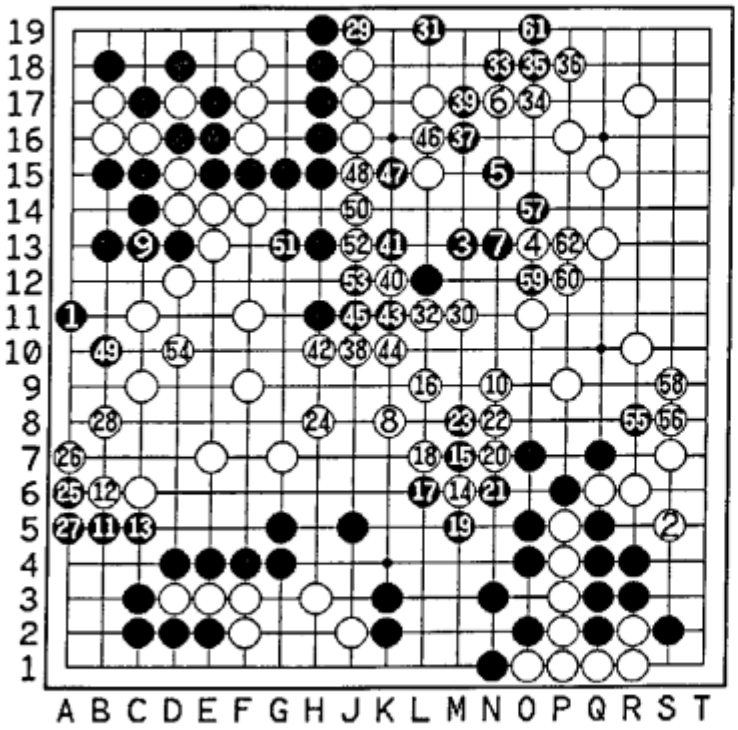
White : IGO3

1 ~ 100



98 (P-1)

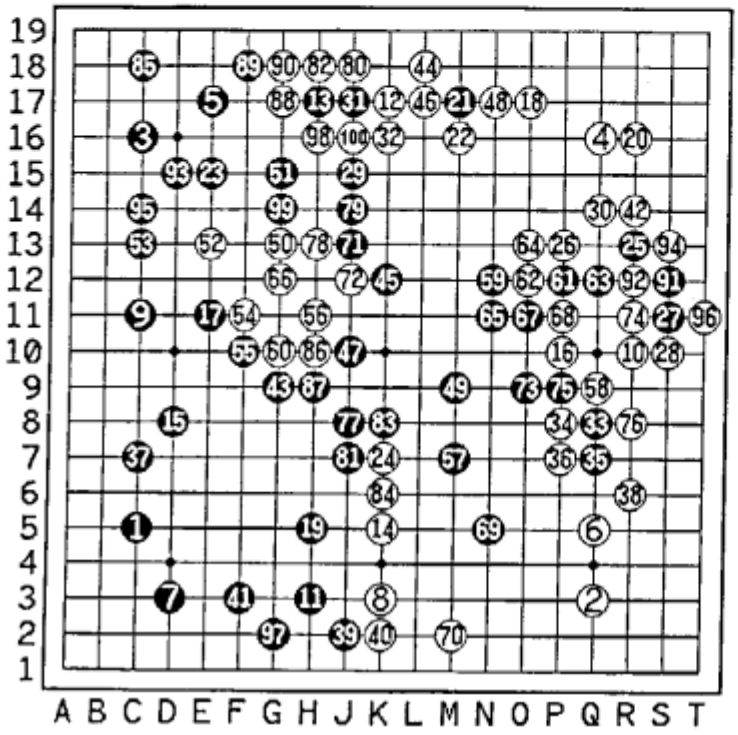
101 ~ end



Black : Explorer

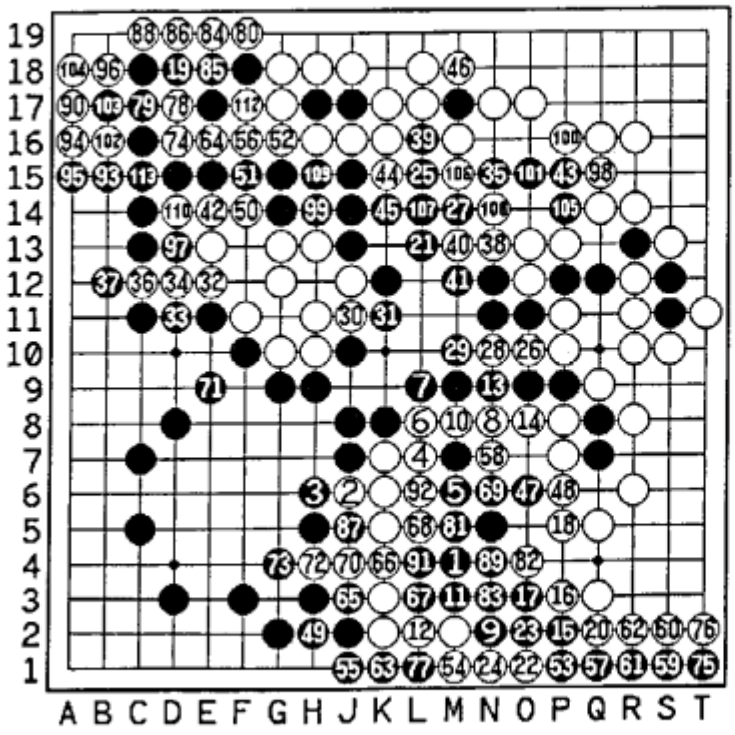
White : Nemesis (winner)

1 ～ 100



211 (pass)

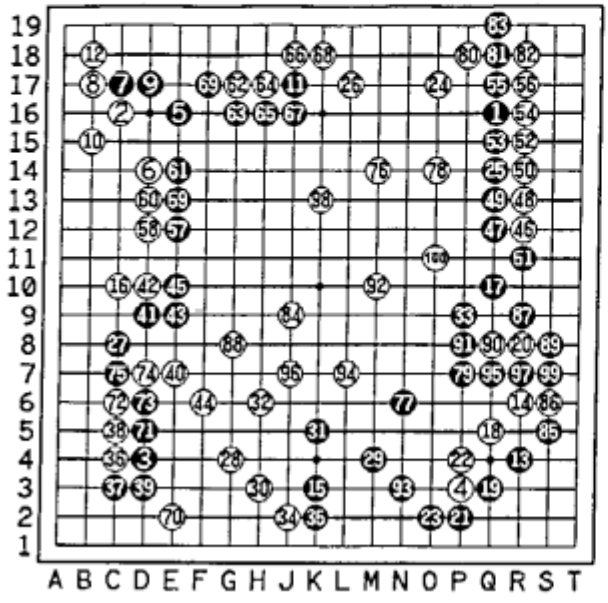
101 ～ end



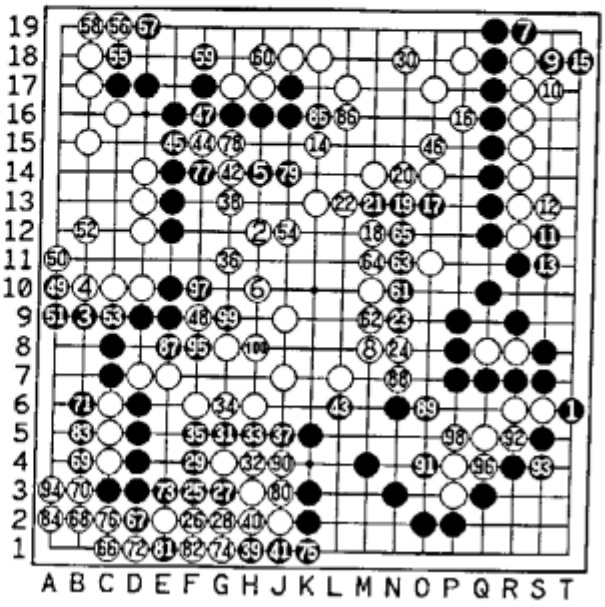
Black : Dragon (winner)

White : Explorer

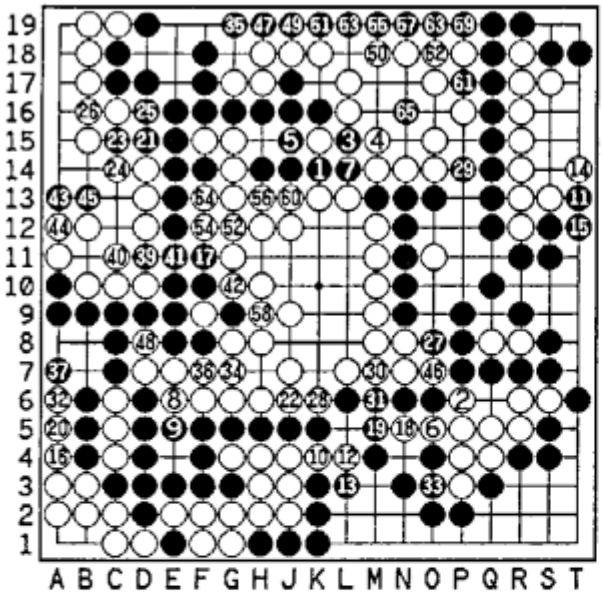
1 ～ 100



101 ～ 201



201 ～ end



238 (E-1)

B 1992 International Computer Go Congress の棋譜

1992 年 11 月 11 日～11 月 12 日に，日本棋院会館 (東京) で行なわれた，1992 International Computer Go Congress(國際電腦圍棋賽) における主な棋譜である。

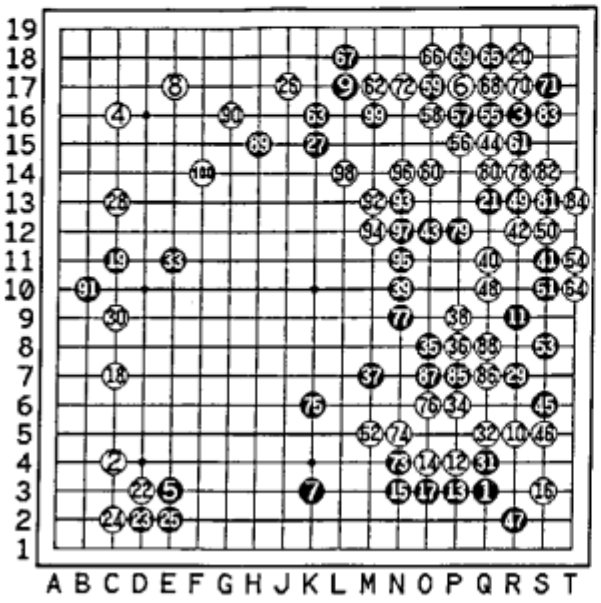
表 B.0–2 : 1992 國際電腦圍棋賽 (International Computer Go Congress)

Rank	Program	Authors	Country	Result
1	Go Intellect	Ken-Hsun Chen	USA	5-1
2	Handtalk	Chen Zhixing	China	4-2
3	Goliath	Mark Boon	Netherlands	4-2
4	GOG (碁世代)	ICOT	Japan	4-2
5	Star of Poland	Janusz Kraszek	Poland	4-2
6	Many Faces of Go	David Fotland	USA	3-3
7	Nemesis	Bruce Wilcox	USA	2-4
8	Dai Hon-in-bou (大本囚坊)	Takeshiro Yoshikawa	Japan	2-4
9	Rex	Hang Joo Kim	Korea	2-4
10	Hwa-Hsia (華夏)	Chiwei Lee, I-hong Kuo	Taiwan	0-6

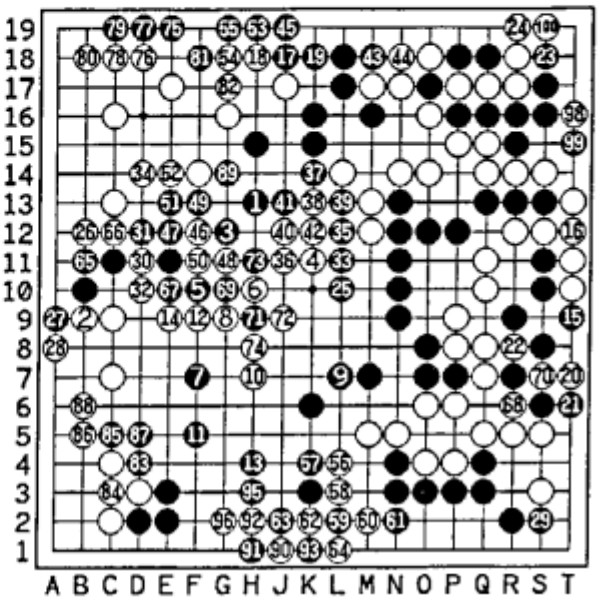
Black : GOG

White : Go Intellect (winner)

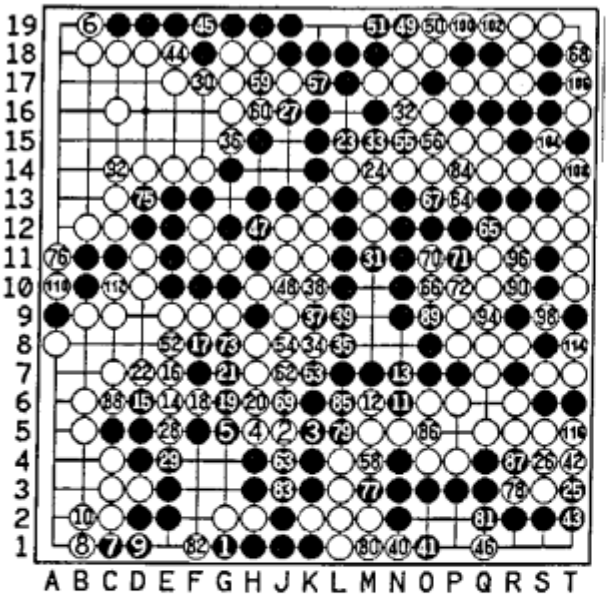
1 ~ 100



101 ~ 201



201 ~ end

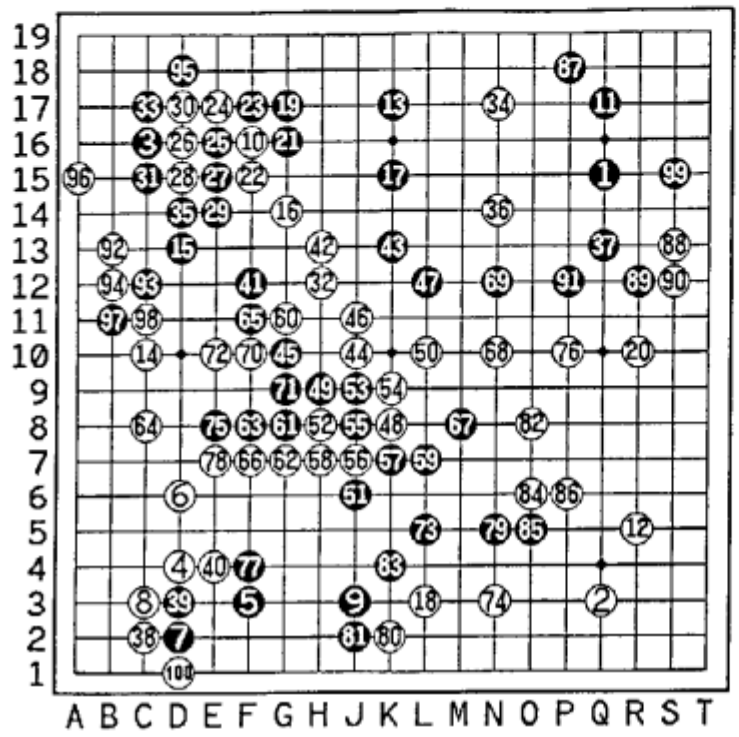


194 (L-2)
197 (J-1)
261 (J-17)
274 (H-9)
291 (pass)
293 (pass)
295 (pass)
297 (pass)
299 (pass)
301 (pass)
303 (pass)
305 (pass)
307 (pass)
309 (pass)
311 (pass)
313 (pass)
315 (pass)

Black : Handtalk

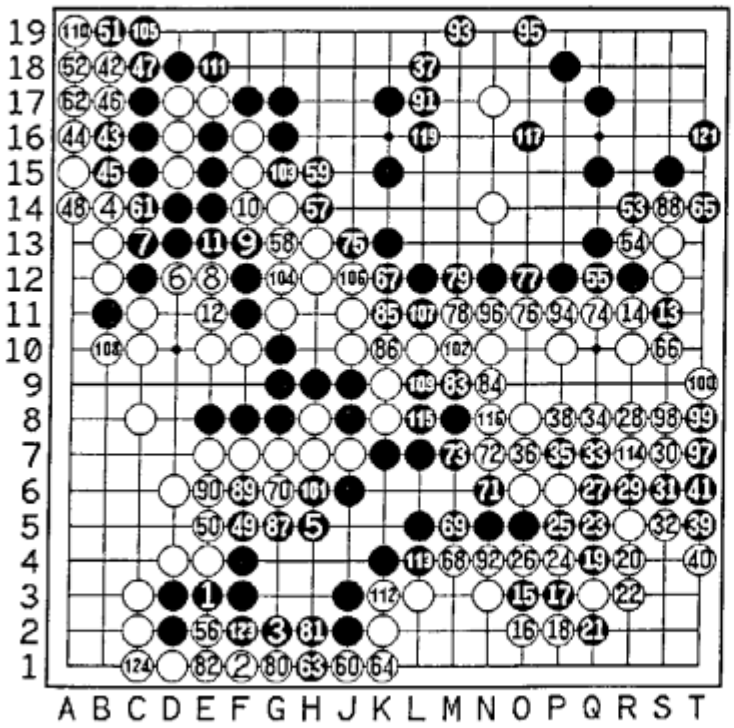
White : GOG (winner)

1 ～ 100



218 (pass)
220 (pass)
222 (pass)

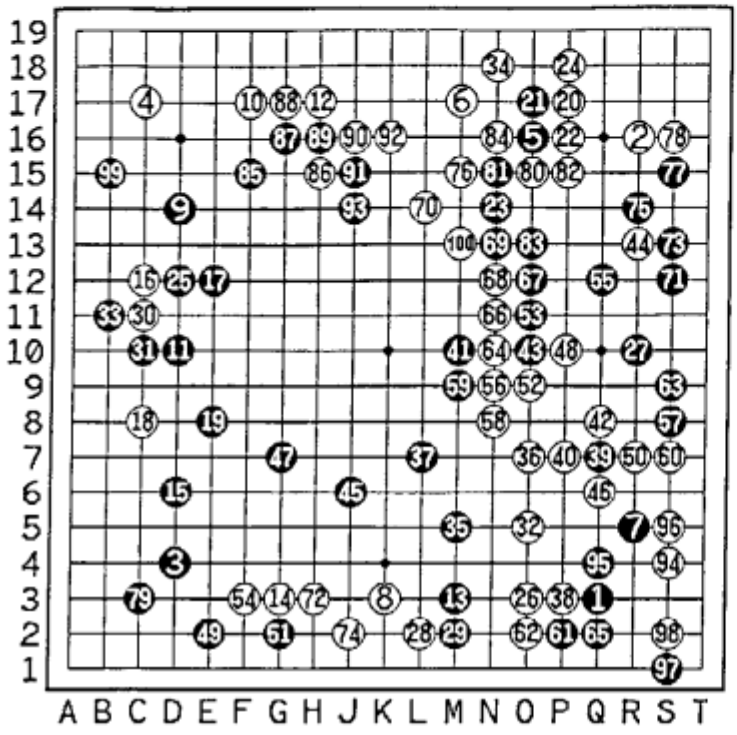
101 ～ end



Black : GOG

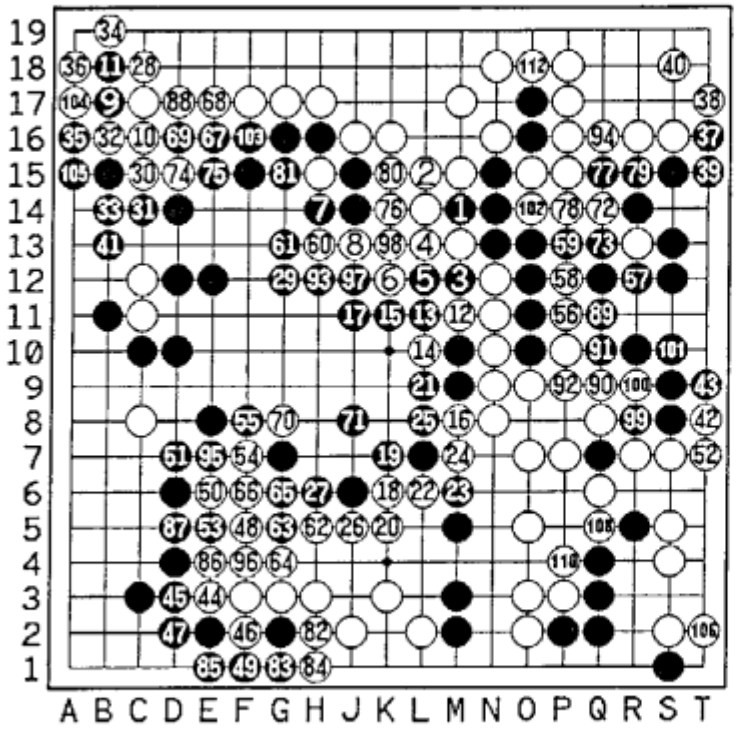
White : Many Faces of Go (winner)

1 ~ 100



207 (pass)
209 (pass)
211 (pass)

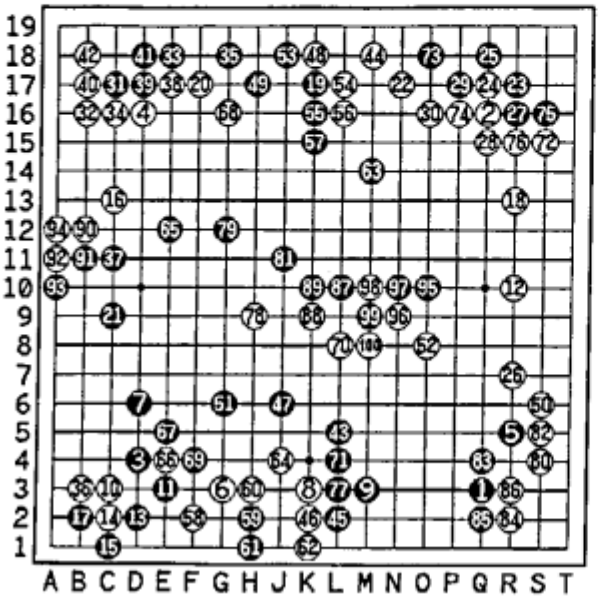
101 ~ end



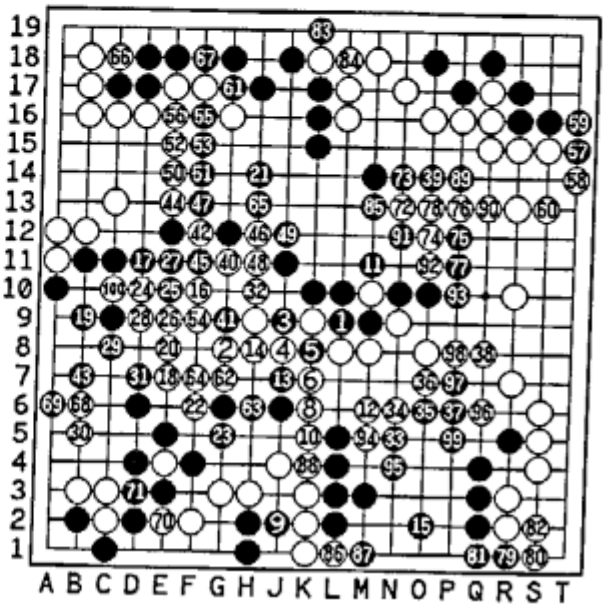
Black : GOG (winner)

White : Nemesis

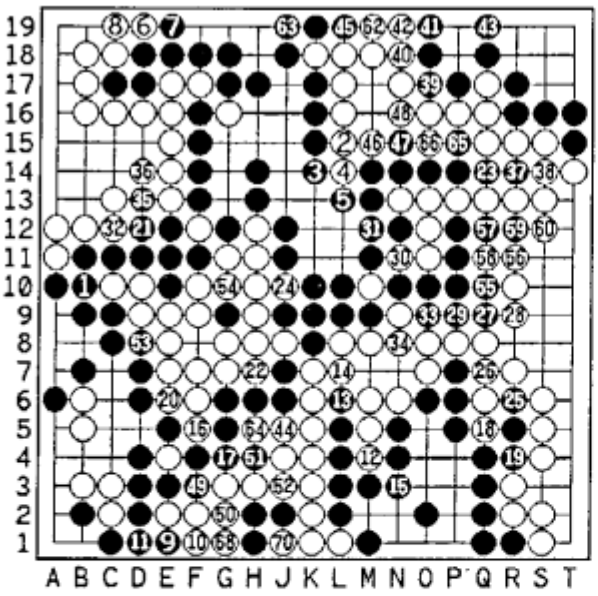
1 ~ 100



101 ~ 201



201 ~ end

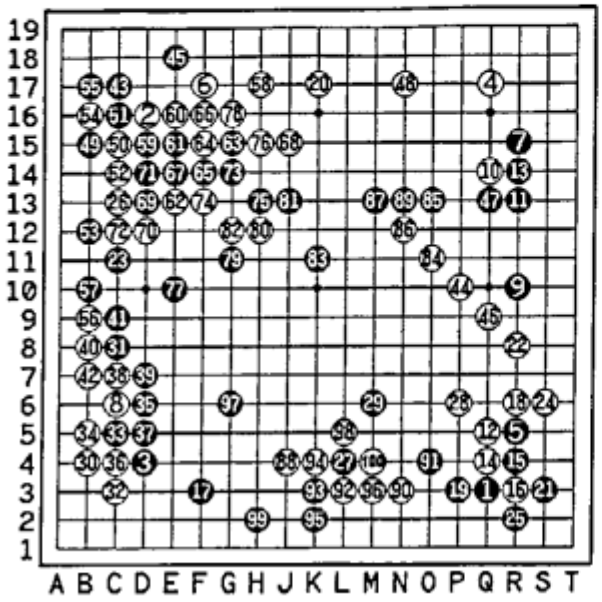


107 (K-9)
261 (M-10)
267 (pass)
269 (pass)

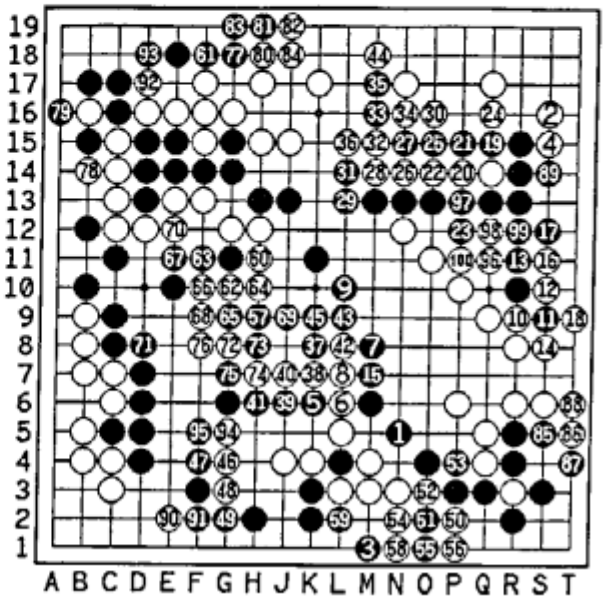
Black : GOG (winner)

White : Dai hon-in-bou

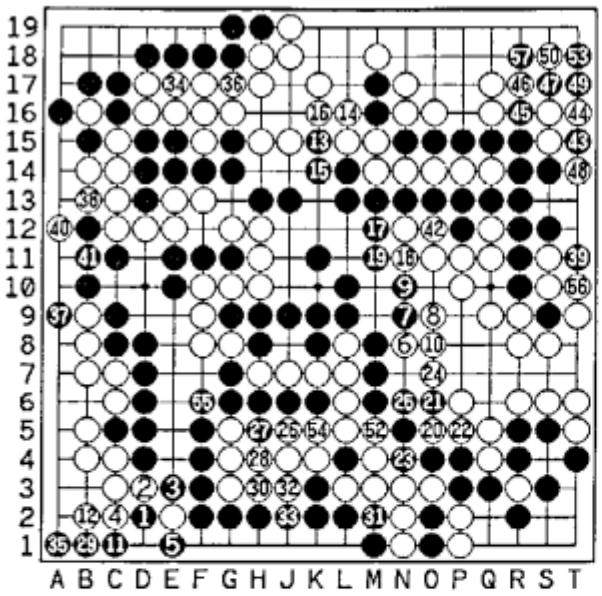
1 ~ 100



101 ~ 201



201 ~ end



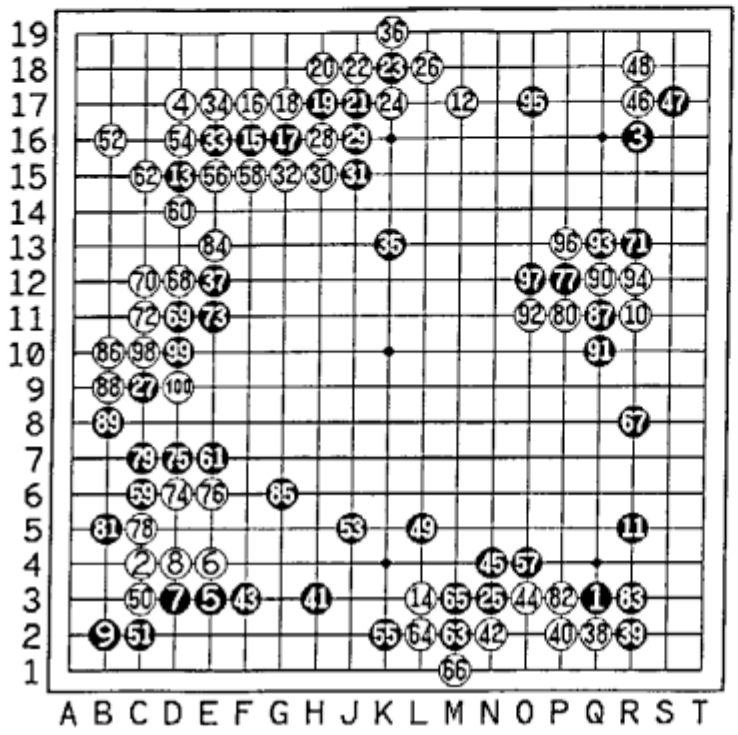
251 (T-15)

Black : GOG (winner)

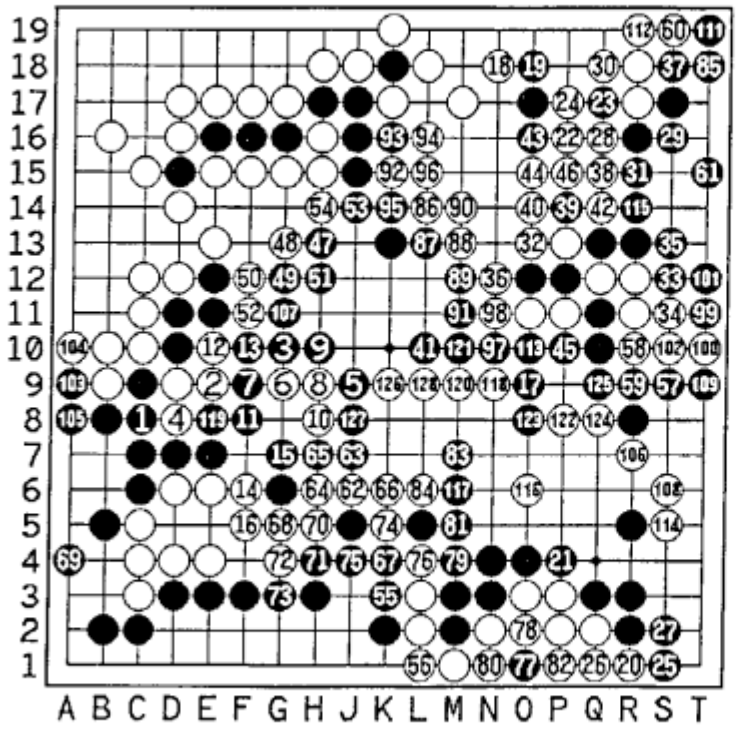
White : Rex

210 (P-12)

1 ~ 100



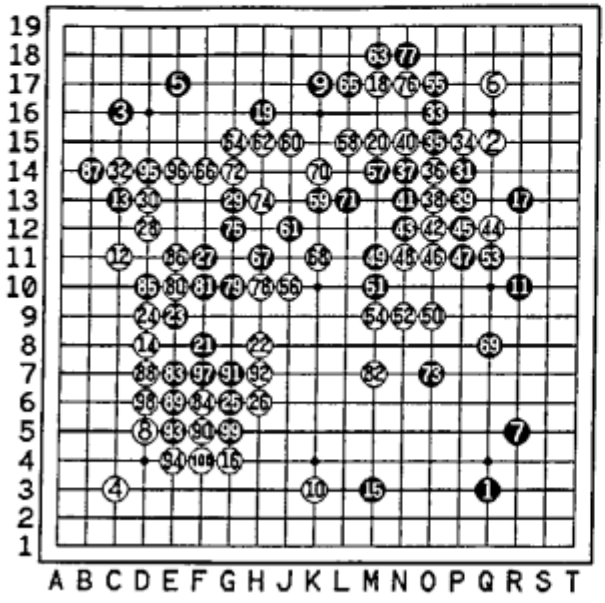
101 ~ end



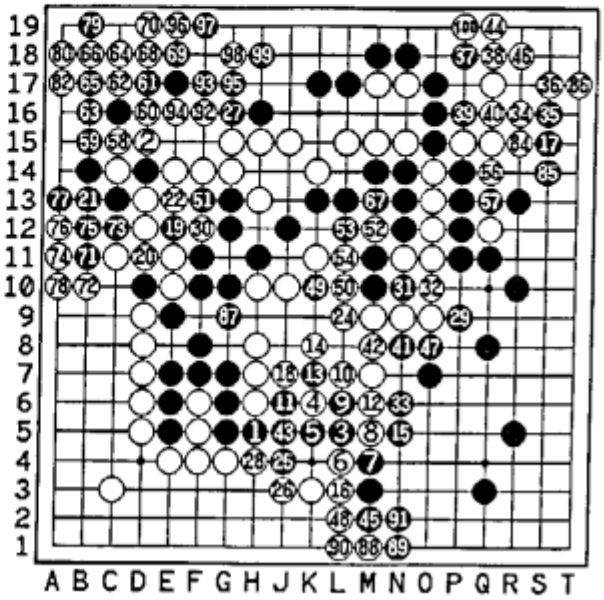
Black : Go Intellect

White : Handtalk (winner)

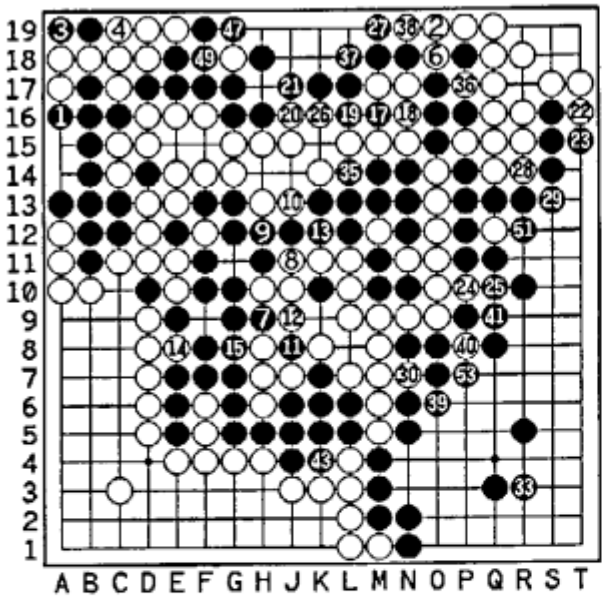
1 ~ 100



101 ~ 201



201 ~ end

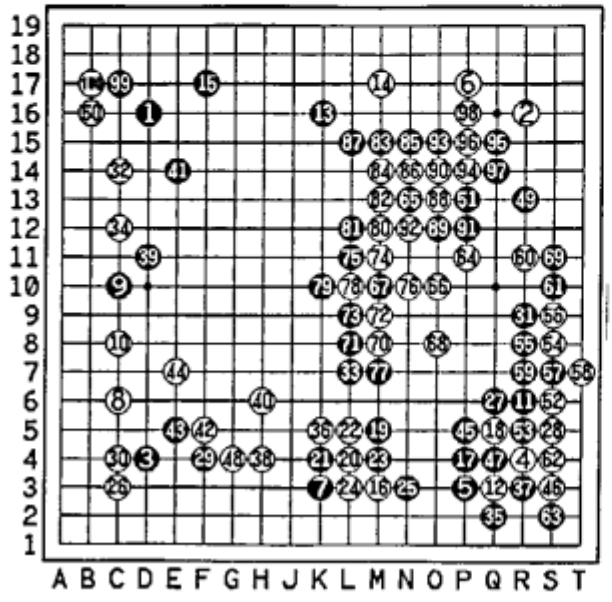


123 (K-6)
155 (E-12)
181 (M-11)
183 (M-10)
205 (A-19)
216 (J-8)
231 (F-12)
232 (pass)
234 (pass)
242 (pass)
244 (pass)
245 (B-19)
246 (pass)
248 (pass)
250 (pass)
252 (pass)

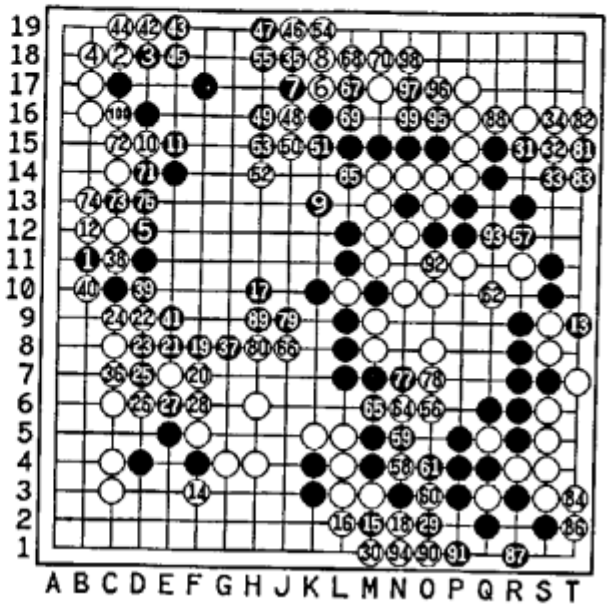
Black : Go Intellect (winner)

White : Star of Poland

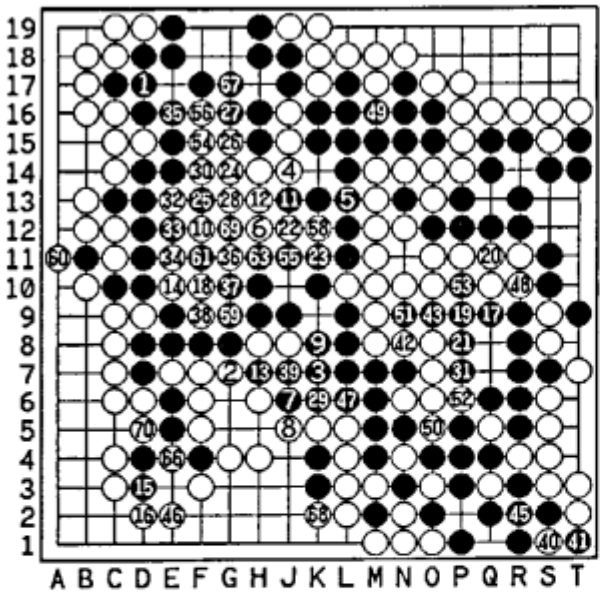
1 ~ 100



101 ~ 201



201 ~ end



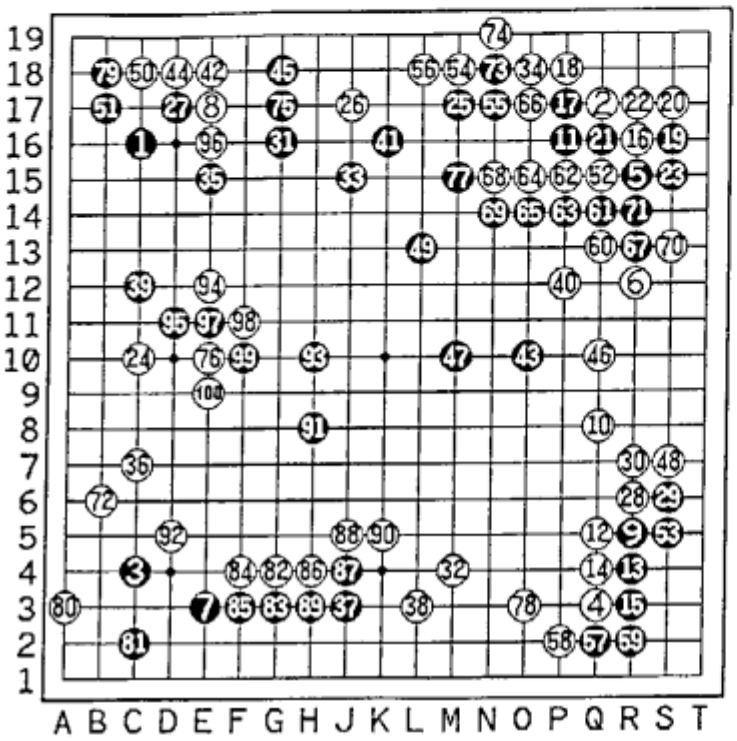
163 (N-3)
176 (M-10)
244 (S-1)
262 (F-10)
264 (E-11)
265 (E-10)
267 (F-11)
271 (F-13)

Black : Go Intellect (winner)

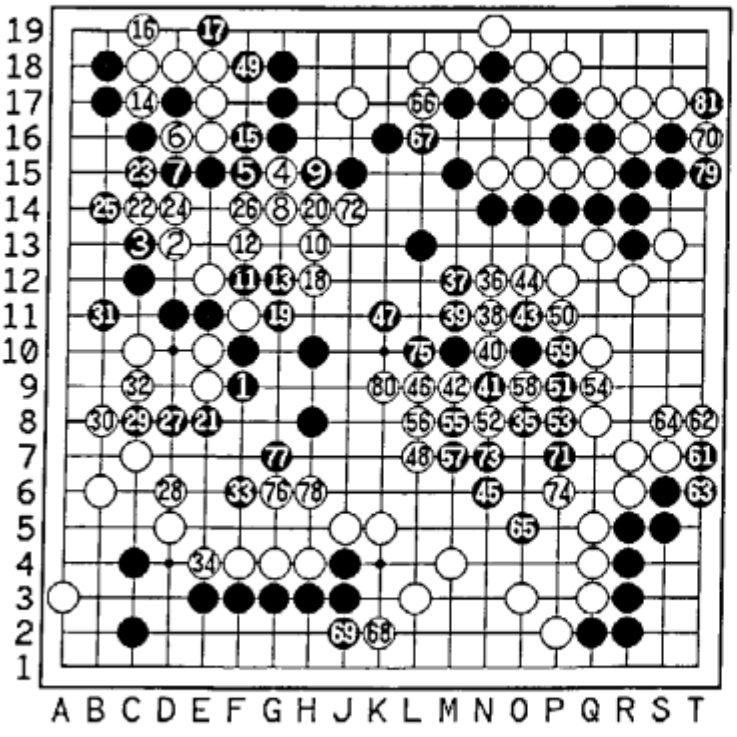
White : Many Faces of Go

160 (N-9)

1 ～ 100



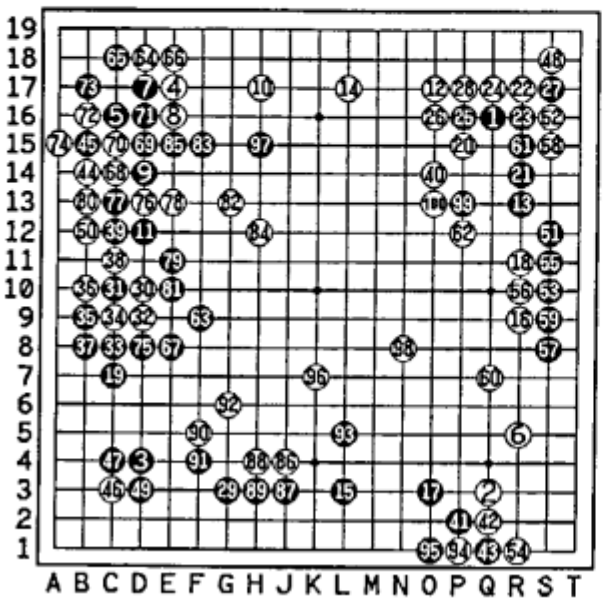
101 ～ end



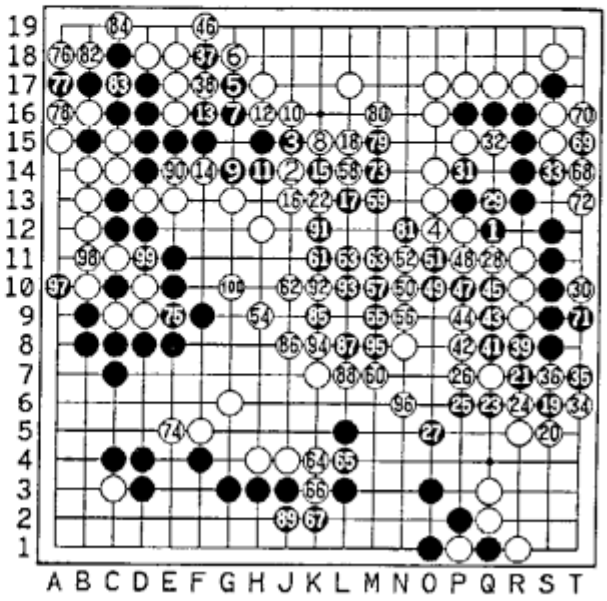
Black : Nemesis

White : Go Intellect (winner)

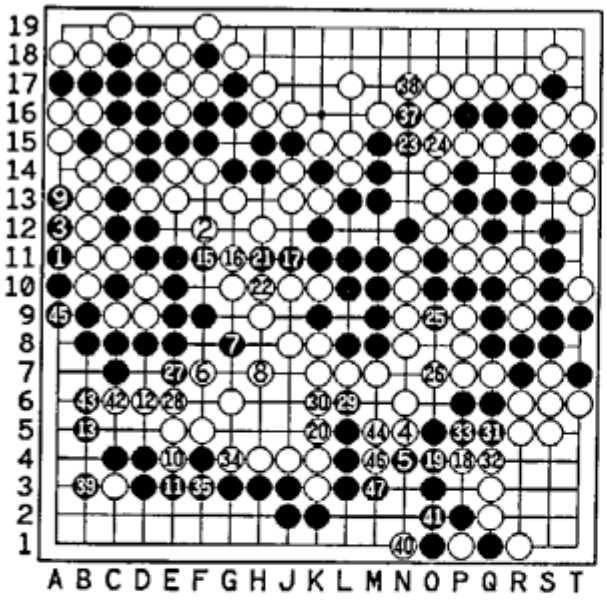
1 ～ 100



101 ～ 201



201 ～ end

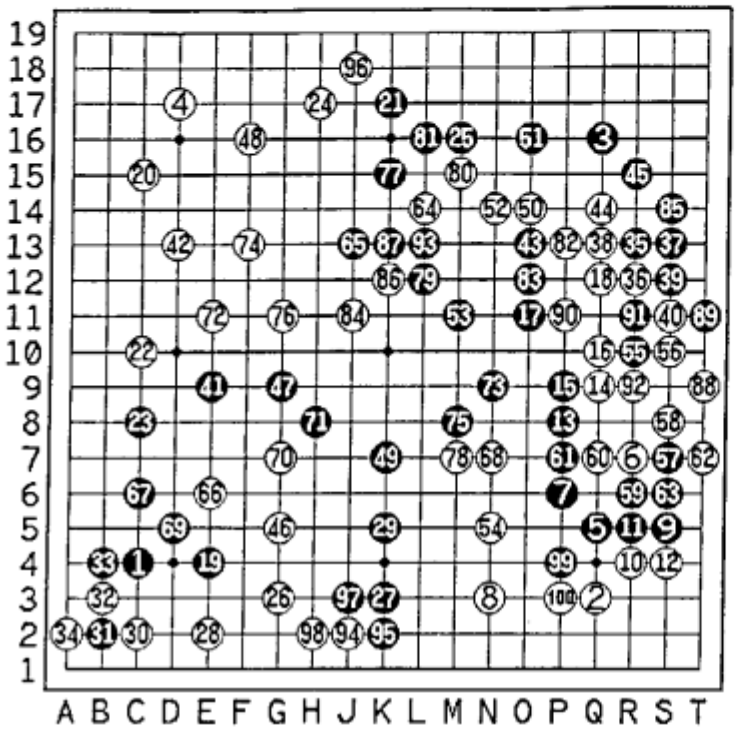


140 (S-6)
214 (C-10)
236 (Q-1)

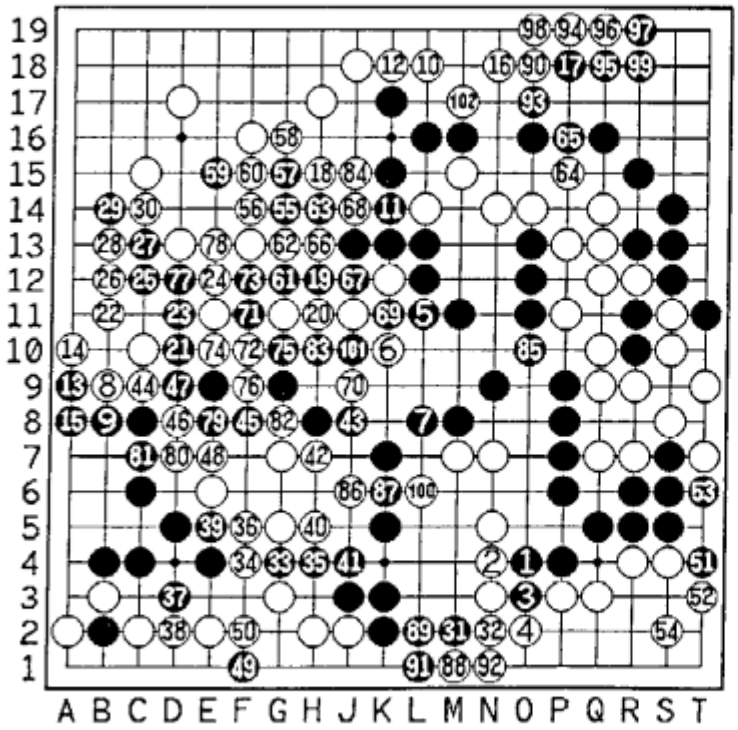
Black : Handtalk

White : Goliath (winner)

1 ~ 100



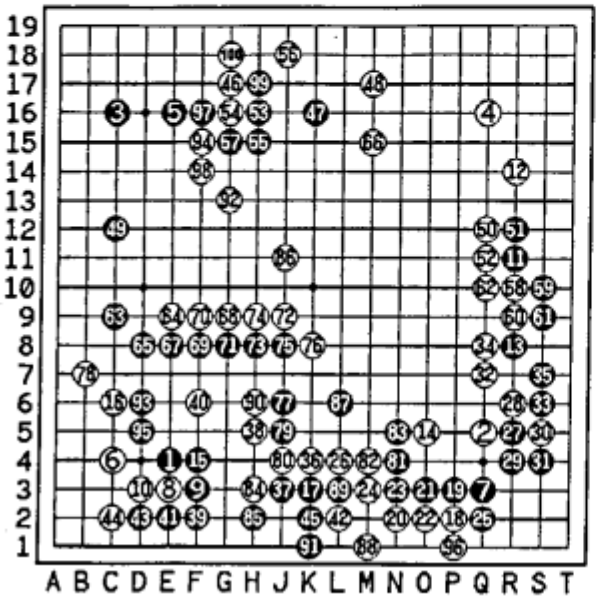
101 ~ end



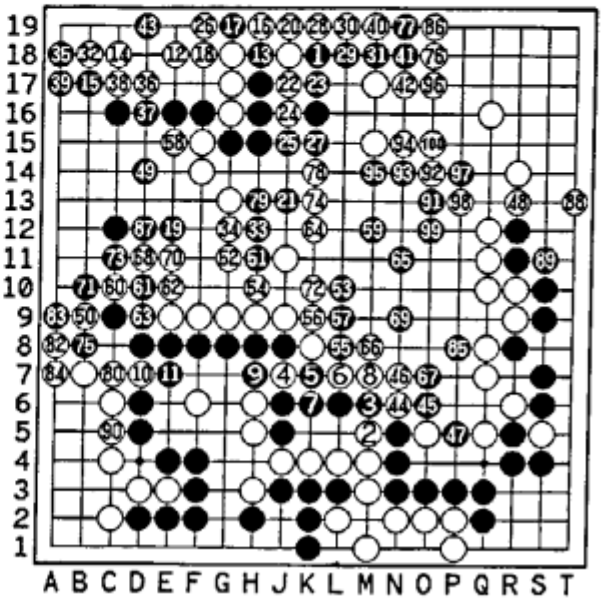
Black : Handtalk (winner)

White : Star of Poland

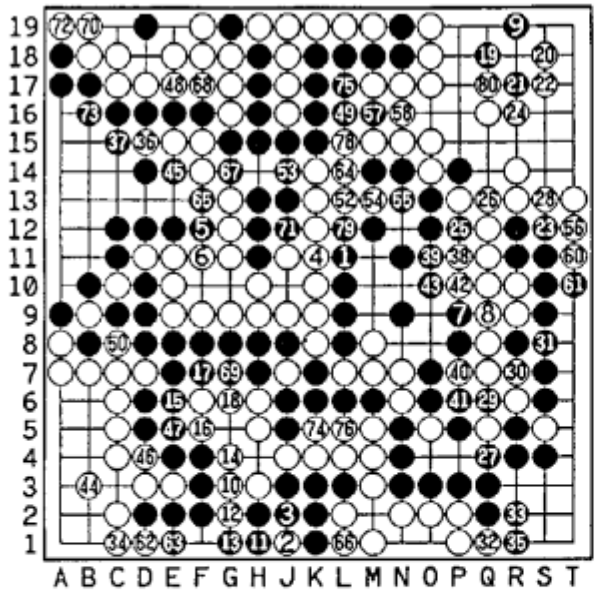
1 ~ 100



101 ~ 201



201 ~ end

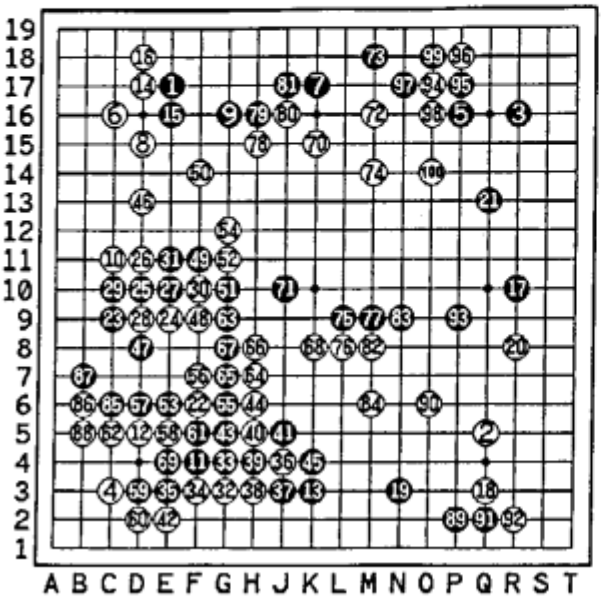


181 (G-19)
251 (B-9)
259 (H-19)
277 (J-7)

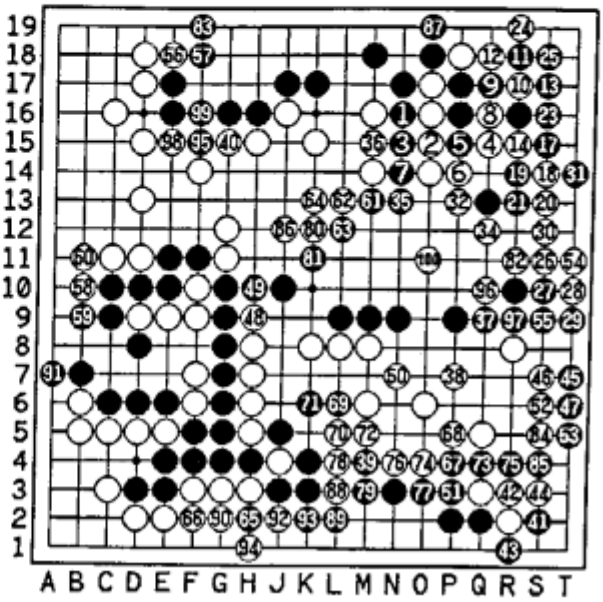
Black : Goliath

White : Star of Poland (winner)

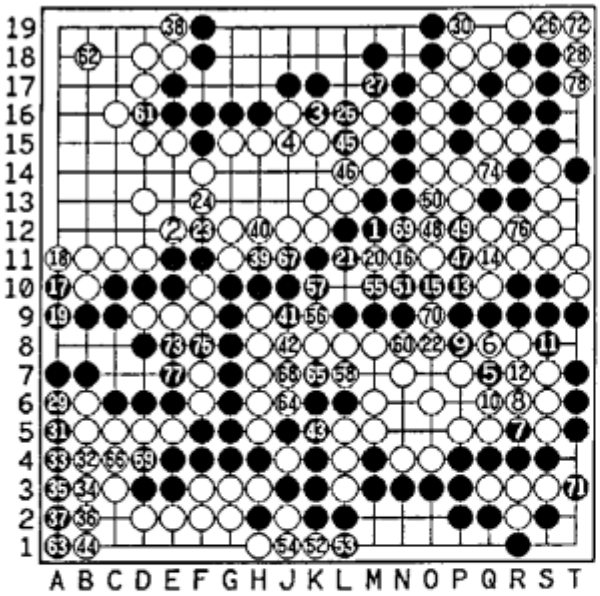
1 ~ 100



101 ~ 201



201 ~ end

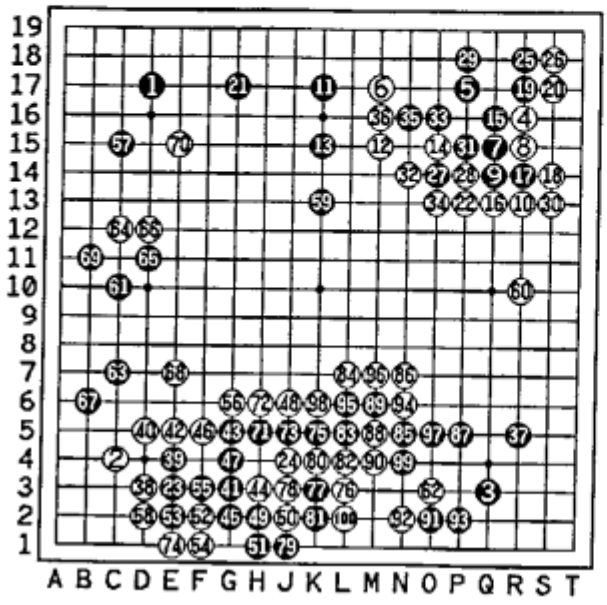


115 (Q-17)
116 (P-17)
122 (R-17)
133 (Q-17)

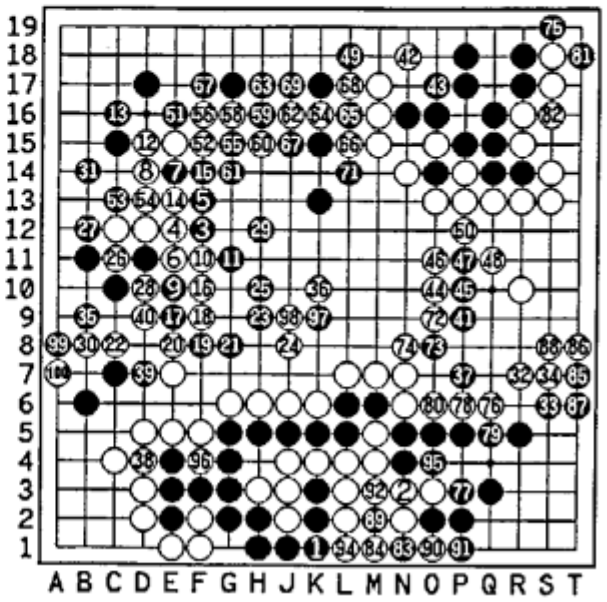
Black : Many Faces of Go

White : Goliath (winner)

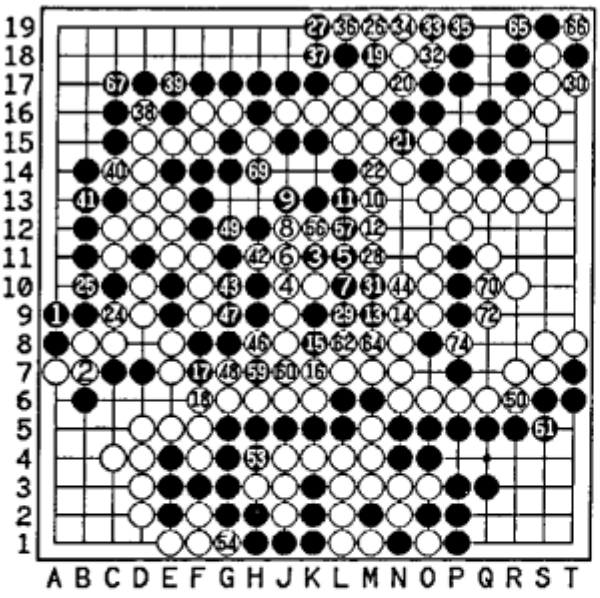
1 ~ 100



101 ~ 201



201 ~ end

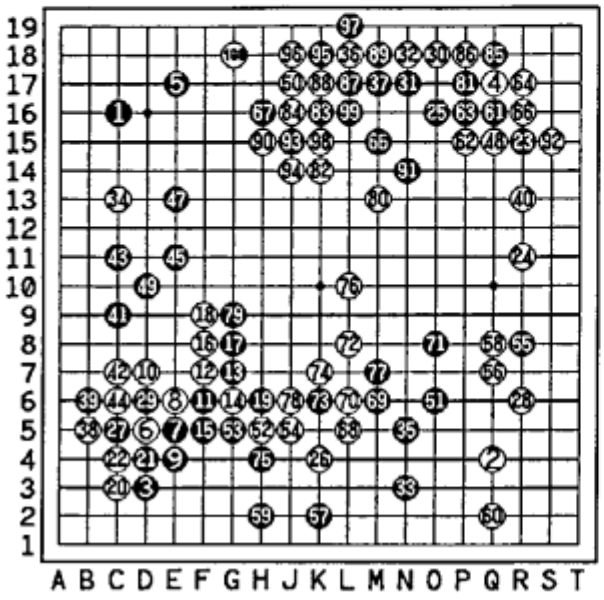


170 (L-16)
193 (N-1)
223 (O-14)
245 (O-1)
252 (O-15)
255 (O-14)
258 (O-15)
261 (O-14)
263 (O-15)
268 (T-18)
271 (pass)
273 (pass)

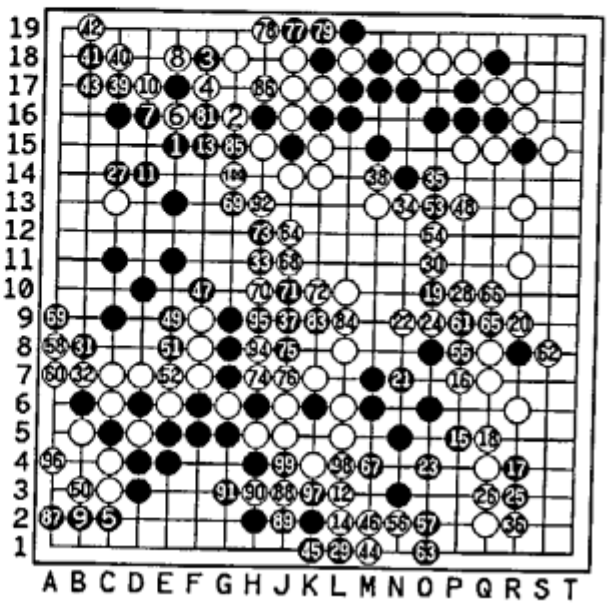
Black : Many Faces of Go

White : Star of Poland (winner)

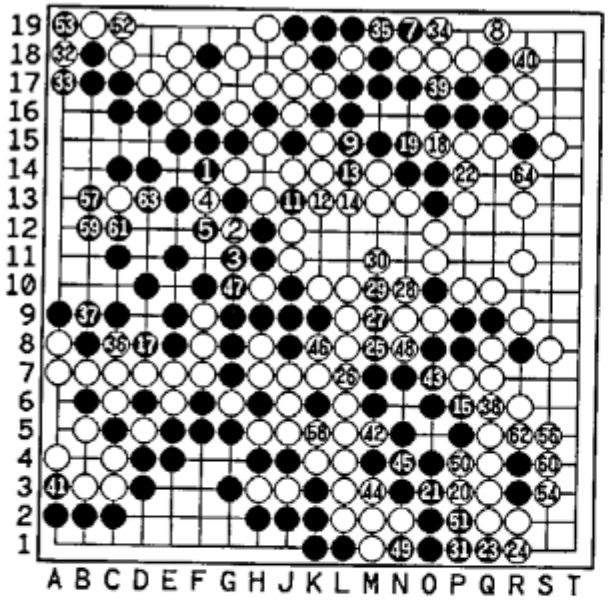
1 ~ 100



101 ~ 201



201 ~ end



46 (D-5)
180 (G-6)
182 (E-17)
193 (H-6)
206 (G-13)
210 (G-6)
216 (H-6)
255 (A-18)