高速利用のための知識の構造化に関する研究

（昭和61年5月31日　原稿受付）

情報工学科（大学院生）　田　原　良　則
永　松　正　博

A study of the structure knowledges for fast use

by Yoshinori TAHARA and Masahiro NAGAMATSU

Abstract

At problem-solving which is one of the main subjects of A. I., many search methods are used to traverse the problem spaces which are usually very large. To make effective and intelligent search, it is necessary to use a great deal of knowledges about the given problem. But if the amount of knowledges becomes great, the accesstime to knowledges increases accordingly. So it is required to develop fast knowledge-access technics.

In this paper"Knowledge-Graph-System"is proposed as a structure of knowledge-base and a fast method of access to it. This system is based on the idea of"A Knowledge which is needed at some moment is included in a small set of knowledges which are determined by the knowledge used in the moment just before". This system works on a directed graph called"Knowledge-Graph". In this graph,each vertex represents a knowledge and each arc is defined from a knowledge which is used at some moment to knowledges which may be used at next moment. Using this knowledge-graph at each moment,we can easily and fast access the knowledge which is required at the moment. Therefore,we can make fast,accurate and intelligent search which are supported by a great deal of knowledges.

1．まえがき

人工知能の研究分野のひとつである問題解決においては，「ゲームのある局面における最善手を見つける」等の問題を解くために，種々の探索手法が用いられる。一般に，このような問題の問題空間は，組み合わせ爆発のため非常に大きく，そのすべてを走査するのは時間がかかりすぎて，現実的ではない[1)2)3)4)5)]。

そこで，発見的手法と呼ばれる方法が用いられる。発見的手法とは，走査している問題空間の中のいくつかの節点を評価関数で評価し，その結果を用いて探索すべき空間を限定し，探索の時間を縮めようというものである。そして，この手法は，探索空間をより多く削減するためには，より精度の高い評価関数を必要とするが，現在までの所，メモリと実行速度の制限により，評価関数は簡単なものしか用いられていない。従って，評価関数の精度が低いので，探索空間はあまり削減できない。

それ故，評価関数として，精度の高いものが要求されるが，そのような評価関数は，大量に知識を持ち，そのアクセスに多量の時間を必要とする（これは，制御の飽

和問題と呼ばれる）ため，探索空間の削減によって節約できる時間に比べて，評価関数の実行に要する時間が大きくなり，結局，探索時間が増加してしまう[4)5)]。

そこで，大量の知識を使う場合でも高速に知識をアクセスするための方法があれば，知識を大量に持つ精度の高い評価関数によって，組み合わせ爆発に打ち勝てる探索アルゴリズムが開発できるであろう。

そこで，本論文では，大量の知識を高速に使うための一手法として，知識グラフ・システムを提案し，また，その有効性を示すために，オセロ・ゲームでの打手決定時に用いられる，探索の一手法であるツリー・サーチにおける評価関数の計算に，これを用いて実験を行った。

2．知識グラフ・システム

離散的に少しずつ変化していく世界を知識グラフ・システムは対象とする。入力は，変化していく世界の状態であり，また，出力は，その各々の時点で必要とされる色々な情報である。

そして，このシステムは，「ある瞬間に必要とされる知識は，その前の瞬間に使われた知識に依存した少量の知識である」という思想に基づいて，知識間に構成される有向グラフを使用して，知識アクセスのための時間を短縮する。

この章では，知識グラフ・システムが，どのようなものであるかを，基本的な部分である知識グラフ・システムⅠと，それに変換の概念に加えてより効率を良くした知識グラフ・システムⅡに分けて説明する。

2.1　知識グラフ・システム

2.1.1　諸概念の定義

(a)　状態集合

対象としている世界の中に現れる状態全ての集合を状態集合と呼ぶことにする。

(b)　抽象的記述

抽象的記述は，状態集合の部分集合を示すための記述であり，状態の特徴を記述する。すなわち，抽象的記述は，ある特徴を持つ状態の集合であり，状態集合の部分集合を示すものである。

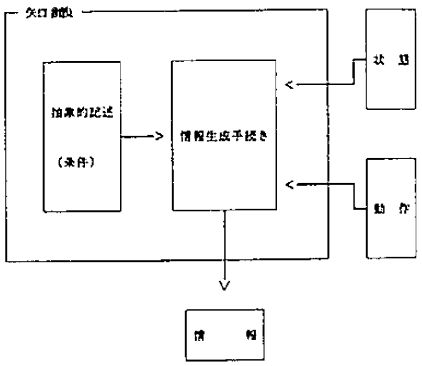
(c)　情報生成手続き

これは，抽象的記述が与えられた時に，状態と知識グラフ・システムの動作によって得られるデータから，情報を生成する手続きである。

(d)　知識，応答

知識は，抽象的記述と，情報生成手続きの２項組である。

変化している世界のある瞬間における状態ｓが，ある特定の知識ｋの抽象的記述ａの示す集合の要素であれば（このことを以下，状態ｓと知識ｋとのマッチングが成功したということにする），抽象的記述ａに対応した情報生成手続きは，図―１に示すように，状態ｓと知識グラフ・システムの動作によって得られるデータから，情報を生成する。そして，生成された情報の集合が，知識グラフ・システムのその時点における応答となる。



図―１　知識

(e)　知識集合

対象としている世界に関して，現在，蓄えられている知識全ての集合を知識集合と呼ぶ。

(f)　知識グラフ

知識グラフは，その節点集合を知識集合とする有向グラフである。知識ｋから知識ｌへの枝が存在するとき，知識ｋは知識ｌの親であるといい，知識ｌは知識ｋの子供であるという。

また，知識グラフの知識の内のいくつかは，初期知識と呼ばれる。初期知識の集合を初期知識集合と呼ぶ。各初期知識から到達可能な集合の和集合は，知識集合であるとする。

知識集合がK，初期知識集合がK_B，枝集合がAである知識グラフを$G=(K, K_B, A)$で表す。

2.1.2　動作アルゴリズム

ここでは，知識グラフ・システムの動作を定める動作関数の定義をする。この関数の入力は，時刻ｔにおける状態S_tと，ポインタP_tり２項組で，出力は，時刻ｔにおける応答R_tと，時刻ｔ＋１におけるポインタ集合P_{t+1}

の2項組である。

以下に，知識グラフG（K，K_B，A）を持つ知識グラフ・システムⅠの動作関数のアルゴリズムをしめす。K，K_B，Aはそれぞれ，知識集合，枝集合，初期知識集合である。また，$P_0=\phi$とする。

1．$R_t=\phi$；$P_{t+1}=\phi$

2．$\forall p\in P_t$なるpについて，以下を実行する。

(1)ポインタpの指す知識を$k\in K$とする。

(2)状態S_tと知識kとのマッチングが，成功したら，

手続きans（k，s_t，YES，R_t，P_{t+1}）を実行する。

成功しなかったら，

知識kの子供の各知識lについて，以下を実行する。

(a)状態s_tと知識kの子供のある知識lとのマッチングが，

成功したら，

手続きans（l，s，NO，R_t，P_{t+1}）を実行する。

3．$\forall b\in K_B$について，以下を実行する。

(1)状態s_tと初期知識bとのマッチングが，成功したら，

手続きans（b，s_t，NO，R_t，P_{t+1}）を実行する。

以下に，アルゴリズム中に出てきた手続きans（k，s，flag，R，P）の引数と動作を示す。

・引数

k∈知識集合；

s∈状態集合；

flag＝YES or NO；これは，着目している知識が変化しなかったことを示すフラグであり，これが知識グラフ・システムの動作によって得られるデータである。

Rは知識グラフ・システムの応答；

Pはポインタ集合；

・動作

(1)知識kの情報生成手続きをf_k，パラメータをX_kとする。

(2)$r=f_k$（x_k，s，flag）；

(3)$R=R\cup\{r\}$；

(4)知識kを指すポインタpを生成する。

(5)$P=P\cup\{p\}$；

次に，（R_t，P_{t+1}）について簡単に説明する。知識グラフ・システムの応答R_t，時刻tにおいて，その知識グラフ・システムを利用しているシステムにとって必要な集合である。そして，P_{t+1}はその応答R_tを作るのに使われた知識を示すポインタの集合である。

また，アルゴリズムの2．(2)において，入力された状態sは，ポインタの指す知識kやその子供の知識lとのマッチングが試みられる。従って，ポインタ集合に含まれるポインタの個数が少ないと，動作関数の計算時間がより短くてすむ。

そこで，知識グラフは，任意の知識とその子供たちの中で，同じ1つの状態とマッチングに成功するものの個数が少なくなるように，構成するのが望ましい。実際，その個数を1つだけであるように，知識グラフを構成することが可能である。なぜならば，今，知識kと知識lそれぞれの抽象的記述a，bの状態sとのマッチングが成功したとする。この時に，kとlを，3つの知識に分けることができる。

1．抽象的記述＝aかつ，b

情報生成手続き＝kとlの情報生成手続きを呼び出して得られる結果を返す手続き

子供への枝　＝kとlの子供への枝

2．抽象的記述　＝aかつ，bでない

情報生成手続き＝kの情報生成手続き

子供への枝　＝kの子供への枝

3．抽象的記述　＝aでないかつ，b

情報生成手続き＝lの情報生成手続き

子供への枝　＝lの子供への枝

2.2　知識グラフ・システムⅡ

ここでは，知識グラフ・システムⅠに変換の概念を加えてより効率を良くした知識グラフ・システムⅡについて簡単に説明する。

まず，変換とは，抽象的記述に対するものである。対象としている世界の状態集合Sに対する任意の抽象的記述を，ある規則に従って，書換えて作られた記述が，やはり，状態集合Sに対する抽象的記述であるならば，このある規則に従った書換えのことを変換と呼ぶ。

そして，1つの知識グラフ・システムに対してこのような変換は，有限個用意され，各変換には，番号が付けてある。

このような変換を抽象的記述に施すことにより，1つの抽象的記述を複数のそれとして，解釈できるようになる。この原理によって，複数の抽象的記述が1つで表現し，知識グラフの節点数を少なくしようというのが，知識グラフ・システムⅡである。

また，これの動作関数は知識グラフ・システムⅠのそれとほぼ同じであり，マッチングを試みる前に，抽象的記述に変換を施すことだけが異なっている。なお，施すべき変換は，各知識に付帯している。

3．知識グラフ・システムの応用

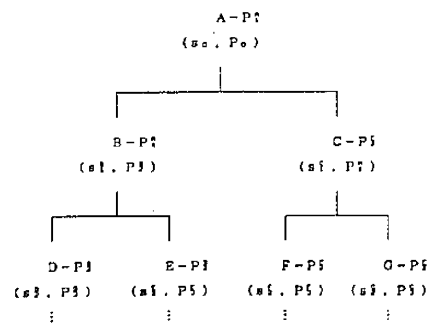
ここでは，ツリー・サーチへの知識グラフ・システムの応用について説明する。

ツリー・サーチにおいて，知識グラフ・システムは評価関数の計算に用いる。従って，情報生成手続きは，評価値と，その他ツリー・サーチの効率を上げるためのなんらかの情報を計算するものとなる。

また，ツリー・サーチにおいては，一般的に，バックトラック等が生じ，評価すべき状態が，世界の変化と同じ順序で出現するとは限らない。しかし，動作関数への入力の1つであるポインタ集合が，知識グラフ・システムの動作状況の記録となっており，ある状態sの評価値を計算させる時は，状態sの示す節点の親の節点を評価した時に出力されたポインタ集合を，状態sと共に入力すればよい。これにより，世界の変化と同じ順序で状態を入力した時と同じ動作を，知識グラフ・システムに行わせることができる。なお，状態sがツリーの根の時に入力すべきポインタ集合は空集合である。

図―2において，各節点の下に書かれている2項組が，その時点での知識グラフ・システムへの入力であり，また，右に書かれているのが，その時点での知識グラフ・システムの出力の1つであるポインタ集合を表す。

例えば，節点Cの示す状態s_1^1に，ある操作を加えることで，節点Fの示す状態s_2^1が作られる。そして，節点Fの評価値等を計算するために，動作関数に与えられる入力は，節点Fの状態s_2^1と，その親である節点Cを評価した時の知識グラフ・システムの出力であるポインタ集合P_2^0である。



図―2　動作関数をもちいたツリー・サーチ

以上を実現するのは，簡単である。なぜなら，ツリー・サーチの実行中に出現した状態sは，ある瞬間には，それがまだ評価される前ならば，必ずどこかに記憶されているはずである。従って，それを記憶した時点で，状態sを示す節点nの親の節点を評価した時の動作関数の入力であるポインタ集合Pを記憶するようにすればよい。これにより，節点nを評価する時にポインタ集合Pを動作関数に与えることができる。

4．評価関数の実現

本研究では，オセロ・ゲームの打手決定アルゴリズムを，ミニ・マックス戦略を若干修正した戦略を用いて作成した。そして，この戦略で用いる評価関数を知識グラフ・システムにより実現した。

4.1　抽象的記述

(a)　ポジション特徴

これは，あるマスの特徴を記述するものであり，(α, β, γ, δ)で表される4次のベクトルである。
その各々の次元は，次のような値をとる。

「1」：条件が成立する。
「0」：条件が成立しない。
「＊」：条件が成立しても，しなくても良い。

また，条件は次元ごとに異なり，次の意味を持つ。

α：打手決定者の駒がある。
β：打手決定者の相手の駒がある。
γ：打手決定者の駒が打てる。
δ：打手決定者の相手の駒が打てる。

(b)　基本特徴

これは，ポジション特徴を8行8列に並べたものである。

(c)　付加特徴

これは，基本特徴と同様に，ポジション特徴を8行8列に並べたものと，どこに何を置くという記述で構成され，どこに何を置いた時に，盤面がどうなるかを示している。なお，「どこに何を置く」という情報は省略しても良い。

(d)　論理式

付加特徴はAからZまでの名前が付けられ，基本特徴は名前として@を持ち，これらの名前は，それの付けられた特徴の示す状態（盤面）の集合に，状態sが含まれ

る（このことを以下，基本特徴が成立する。または，付加特徴が成立するということにする）なら真，そうでないなら偽の値をとる論理変数となる。

そして，これらの論理変数と，２項演算子＋（OR），＊（AND），および，単項演算子－（NOT）で構成される論理式を考える。また，この論理式では，空の時，真の値を取る。

抽象的記述は，以上の基本特徴１つ，26個以下の付加特徴，それに，１つの論理式で構成され，基本特徴が成立し，かつ，論理式の値が真となった時，この抽象的記述と状態ｓとのマッチングは成功する。

4.2　変換

4.1で示した抽象的記述の方法である８行８列の並びの，０°回転，90°回転，180°回転，270°回転，更に裏返して，０°回転，90°回転，180°回転，270°回転という８通りの変換と，打手決定者を反転しない，打手決定者を反転する２通り，合わせて16通りの，抽象的記述に対する変換が考えられる。

そこで，この16通りの変換をT_1から，T_{16}に割当てた。

4.3　情報生成手続きと評価関数

今回用いた情報生成手続きは，次のような４つのデータを持つ。

(a)　ポジション集合

図―３のように，盤面上の各マスを，０から９までに分類する。そして，その０から９までの分類全部の集合の部分集合を，ポジション集合と呼ぶことにする。

例えば，ポジション集合が，分類７と８を含んでいるならば，これは，以下のマスを示している。

分類７：ｃ３，ｆ３，ｃ６，ｆ６，

分類８：ｄ３，ｅ３，ｃ４，ｆ４，ｃ５，ｆ５，ｄ６，ｅ６

	a	b	c	d	e	f	g	h
1	0	1	2	3	3	2	1	0
2	1	4	5	6	6	5	4	1
3	2	5	7	8	8	7	5	2
4	3	6	8	9	9	8	6	3
5	3	6	8	9	9	8	6	3
6	2	5	7	8	8	7	5	2
7	1	4	5	6	6	5	4	1
8	0	1	2	3	3	2	1	0

図－３　盤面上のマスの分類

(b)　Ｂポジション特徴

Ｂポジション特徴とは，4.1の(a)で定義したポジション特徴からβを除いたベクトルである。

(c)　ポジション評価値

これは，ポジション集合の要素に含まれる任意のマスに対する評価値である。

(d)　先読み用パラメータ

これは，ツリー・サーチにおいて，より望ましいと思われる節点より以下をより深い節点まで評価するための非負の整数である。

また，この手続きは次の３つの引数を受け取る。

(a)評価している節点の状態ｓ

(b)マッチングに成功した変換Ｔの番号ｉ

(c)知識グラフ・システムの動作によって得られるフラグｆ

そして，情報ｒを返すが，情報ｒは，２次の数値ベクトル（value，deep）である。以下に，情報生成手続きのアルゴリズムを示す。

１．もし，ｆ＝ＹＥＳなら，deep＝先読み用パラメータ；そうでないなら，deep＝－１；

２．評価用パラメータｖの，基本特徴に対する評価値をvalueに入れる。

３．評価用パラメータｖの，成立した付帯特徴に対する評価値を，valueに加える。

４．ポジション集合をＰ，Ｂポジション特徴をＢ，ポジション評価値をＶとする。

５．∀ｐ∈Ｐなるｐについて，ｐの要素の示す状態ｓのマスがＢの特徴に含まれるなら，Ｖをvalueに加える。

６．∀ｐ∈Ｐなるｐについて，ｐの要素の示す状態要素ｓ要素のマスがＢの打手決定者を反転した特徴に含まれるなら，Ｖをvalueから引く。

７．ｉの示す変換が，打手決定者の反転を含んでいるならば，value＝－valueとする。

次に，評価関数について説明するが，今回作成した評価関数は２つあり，その１つＡ評価関数は今までに説明した知識グラフ・システムを用いる。もう１つＢ評価関数は，以下に示すような単節点グラフを用いて，評価値を計算する。

単節点グラフとは，２つの正の整数ｒ１，ｒ２と4.2で

示したポジション集合，Bポジション特徴，それに，ポジション評価値の5項組である。このr1，r2は，ゲームの始めから何手目であるかを示し，盤面上の駒の個数から3を引いた数である。

今，A評価関数が呼ばれたとすると，Aは，節点nを表す状態をs，節点nの親を評価した時に得られたポインタ集合P_tとして，2次の数値ベクトル（VALUE，DEEP）とP_{t+1}を，次のアルゴリズムで作る。

ただし，VALUEは評価した節点の評価値であり，DEEPは－1でない時は，その節点から深さDEEPまでのツリー・サーチを行うことを示す。

1．（s，P_t）を知識グラフ・システムに与えて，（R，P_{t+1}）を得る。
2．VALUE＝0；
3．DEEP＝－1；
4．∀r∈Rについて，以下を行う。
(1)r＝（v，d）とする。
(2)VALUEにvを加える。
(3)vが正かつ，DEEP＜dなら，DEEP＝dとする。

更に，B評価関数は，評価している節点の状態sが示す盤面上にある駒の個数から3を引いた数をrとし，単節点グラフの集合をOとすると，次のアルゴリズムで，Vを計算する。

1．V＝0とする。
2．∀o∈Oについて，以下を実行する。
(1)o＝（r1，r2，P，B，v）とする。
(2)r1≦r≦2ならば，以下を実行する。
(a)∀p∈Pなるpについて，pの要素の示す，状態sのマスがBの特徴に含まれるなら，eをVに加える。
(b)∀p∈Pなるpについて，pの要素の示す状態sのマスがBの打手決定者を反転した特徴に含まれるなら，eをVから引く。

以上が，評価関数のアルゴリズムであるが，知識の抽象的記述の表現力と，実際に作ることのできた知識の量が十分でなかったため，知識グラフ・システムが評価に全く影響しない節点の個数が多くなった。故に，ミニ・マックス戦略に有効な評価値の計算が出来なくなったので，それを補うためにB評価関数を用意した。

5．実験

5.1　実験方法

実験では，4で示した評価関数を用いたミニ・マックス戦略によって，打手決定するもの同士で，対戦させるのだが，一方は知識グラフ・システムを用いないようにして，知識グラフ・システムの有効性を示す。知識グラフ・システムを用いたアルゴリズムをC，用いない方をMと呼ぶことにした。

実験は，Cを先手として1勝負，Mを先手として1勝負実行することを1組として，3組行った。ただし，全く同じアルゴリズムでは，当然，全く同じ勝負となるので，ミニ・マックス戦略における最大値や最小値の比較の時，もし，値が等しかったら，1/2の確立でその比較が成立するようにした。

5.2　実験結果

表－1に，実験結果を示す。表における項目は，それぞれ，次の意味を持つ。

駒：ゲーム終了時の駒数。（これが多い方の勝）
節点：ツリー・サーチで，評価した節点の個数。
時間：ツリー・サーチに要した時間。（秒）

これらの数値を検討した結果，ツリー・サーチにおいて，評価した節点の個数は，2.3％の増加と，極わずかであるのに対して，勝負を決める駒数は，95％の増加であり知識グラフ・システムの有効性はあきらかである。

また，評価時間が45％増加している理由は，打手決定アルゴリズムCが，節点の評価値を求めるのに，評価関数をA，Bのどちらとも用いたからである。従ってA評価関数（知識グラフ・システム）を用いない打手決定アルゴリズムMにくらべて，時間が増えるのは，当然である。しかし，B評価関数を使わねばならなかったのは，十分な量の知識を使えなかったためであり，知識グラフ・システムそれ自身に原因があるのではない。もし，

表－1　実験結果

	C が 先 手				M が 先 手			
		駒	節点	時間		駒	節点	時間
1	C	40	4460	294	C	45	5105	415
	M	24	6363	319	M	19	4714	267
2	C	47	5400	424	C	55	8888	751
	M	17	6418	344	M	9	5522	276
3	C	42	6711	436	C	25	5516	332
	M	22	5618	261	M	39	6621	344

十分な知識が使えるならば，評価関数Bを知識グラフ・システムは用いる必要がないので，そのような知識がB評価関数程度の計算時間で評価できるならば，時間の増加も，やはり，極わずかなものとなるはずである。

6．あとがき

人工知能の問題において，ツリー・サーチが要する時間を短縮するために，精度の高い評価関数が求められている。そして，評価関数の精度を十分に上げるためにはそれに大量の知識を持たせることが必要であろう。しかし，大量の知識を使うと，評価関数の計算時間が増大してしまい，ツリー・サーチの時間が短縮できない。従って，高速に大量の知識を使うための手法が必要となる。

本論文では，そのような手法として知識グラフ・システムを提案した。

徐々に変化する世界においては，任意の時点において使われる可能性のある知識は，そのシステムの持つ知識全部ではなく，その一部分だけであるといえる。更に，その世界において，任意の時点で使われ得る知識の集合が，その時点の1つ前に使われた知識が決まれば，決めることができるようなものである場合には，前者へ，後者からの枝を用意することによって，知識の集合を有向グラフとして表現することができる。これを知識グラフと呼ぶことにし，それを利用して知識アクセスの時間を短縮する知識グラフ・システムⅠについて，2.1で述べた。

また，抽象的記述を変換という概念によって分類し，変換群により得られる抽象的記述を，少数の抽象的記述で代表するという原理によって，いくつかの知識を1つの知識で代表して，知識グラフの節点数を減少したグラフ・システムⅡを2.2で示した。

そして，知識グラフ・システムⅡを，ツリー・サーチにおいて用いられる評価関数の計算に応用し，それを，オセロ・ゲームの打手決定アルゴリズムとして実現し，コンピューター・シミュレーションによって，その有効性を確認した。

また，オセロ・ゲーム以外の世界においても，知識グラフ・システムにより，知識を有効に構造化することのできる世界は少なくないと思われる。

7．今後の展望

実験のために，知識を作っていて，気がついたのであるが，2.2で述べた知識グラフ・システムⅡは，知識グラフが不必要に大きくなるため，メモリーを多く消費し，知識グラフの効率が悪い。すなわち，変換によって同じになる複数の知識が，知識グラフの中に出てくることがある。

そこで，そのようなことが起きない知識グラフ・システムⅢを実現することが，知識グラフ・システムをより強力にするための第1の課題である。

4.で述べているように，今回作成した知識グラフ・システムの抽象的記述の表現力は十分でない。又，本論文では，その一般的な表現方法について，何も述べていない。しかし，知識表現については，人工知能研究の一分野としてかなり研究されている。従って，今後，知識グラフ・システムをより強力にするためには，現在，知識表現の分野での研究成果を，知識グラフ・システムに応用することが，第2の課題であろう。

参考文献

1) A.L.Samuel:"Some Studies in Machine Learning U sing the G ame of C heckers", in Computers and Thought, E.A.Feigenbaum and J.Feldman (Eds.) MacGraw-Hill (1963)
2) P.H.Winston:"Artificial Intelligence", Addison-Wesley (1977)
3) N.J.Nilsson:"Principles of Artificial Intelligence"Tioga (1980)
4) A.Barr and E.A.Feigenbaum:"The Handbook of Artificial Intelligence", Kaufman (1981)
5) E.Rich:"Artificial Intelligence", McGraw-Hill (1983)
6) 長谷川五郎：'図解オセロ入門'，紅有社（1985）
7) 日本オセロ連盟：'オセロの打ち方'，講談社（1985）