碁盤上の連数最大化問題について

矢野洋平 † 村松正和 ††

電気通信大学情報工学専攻

† youhei@jsb.cs.uec.ac.jp　　†† muramatu@cs.uec.ac.jp

コンピュータで囲碁を考える際に，連 (String) は最も基本的な概念と言って良いだろう．ここでは 19 路盤に存在できる連の最大値を求める問題を考える．本研究の目標は数理計画問題を用いて，19 路盤に存在できる最大の連数を厳密に求める事である．

On the maximum number of strings feasibly put on the n×n Go board

YANO Youhei†　MURAMATSU Masakazui††

Department of Computer Science, the University of Electro-Communications

When we consider the game of Go with computer, string is the most basic concept. However, the maximum number of the strings that can be put on 19×19 board is not known. The purpose of this paper is to calculate this number by using mathematical programing. we call this optimization problem as Max String Problem(MSP).

1　連数最大化問題とは

1.1　はじめに

連とは，**「縦横に繋がった同じ色の石の集合」**である ([1])．別の表現をすると，相手の石によって囲まれたら盤上から取り上げられる石の集合である．19 路盤でルールに合法的に存在できる連数の最大値が $19^2 = 361$ を超えないのは明らかであるが、実際にはいくつ存在できるのかを求めたい．

本研究では，連数最大化問題 (Max Strings Problem，以下 MSP) を非常にシンプルな 0-1 整数計画問題で表現する．一度整数計画問題に定式化すれば，汎用的な整数計画問題を解くプログラム[1]は盛んに開発されているため，これを利用することで解を得る事ができる．本研究でもフリーのソルバである，GLPK [3] を使用した．

ただし整数計画問題は NP 困難な問題であり，大きな問題を解く事は事実上不可能である．本論文では，19 路盤に対する連数最大化問題を解く事は，現在開発されているソルバでは相当困難である事を示す．

1.2　MSP(n)

MSP(n) で，**「n 路盤に存在できる連の数の最大値を求める問題」**又は**「その最大値」**を指すものとする．ただし，黒石と白石は同じ数でなくても良いが，囲碁のルールに反する盤面は認められない．以降の章では，連数の最大値を最適値，最適値を達成する盤面を最適解と呼ぶ．

たとえば MSP(2)=2，MSP(3)=6，MSP(4)=12 であり，その最適解は図 1，2，3 である．

図 1: 2 路盤の最適解

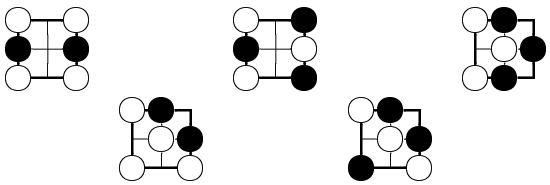

図 2: 3 路盤の最適解

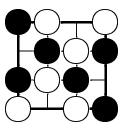

図 3: 4 路盤の最適解

[1] ソルバ (Solver) と呼ばれている．フリーも商用もあるが，商用のソルバの方が高速である

1.3 4路盤の最適性の証明

2路盤，3路盤は手作業で全探索すれば簡単に証明できる．4路盤では以下のように理論的に証明する事ができる．

証明 図4の隅の点線内に存在できる連の最大値は2であり，これ以上置けないのは自明である．点線内以外 (中央の4つ) にはすべて碁石が埋まっている．したがって，MSP(4)=12である．□

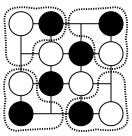

図 4: **点線の中には存在できる連の最大値は 2**

4路盤で使った方法は，5路盤以上の盤面では使えそうも無い．従って，以下では，数理計画の手法を用いてこれを求める事を考える．

本論文の構成は次にようになっている．2章で市松模様の盤面を定義し，最適解の中に市松模様の盤面が存在する事を証明する．3章は市松模様に限定したMSP(n)の整数計画問題への定式化を行い，4章は高速に最適解を求めるための定式化の工夫について述べている．5章は結言であるが，最近MSP(19)が解かれたとの報告があったので，これを紹介する．

2 市松模様

n路盤の盤面パターンは，3^{n^2}通りが存在し，19路盤について単純に数え上げるのは不可能である．

この章では，市松模様という盤面を定義し，少なくとも1つの最適解が市松模様の盤面である事を示す．以降の章では，MSPは市松模様に限定して解くことにする．

2.1 市松模様の盤面の定義

図5のように完全に黒白の石が交互に敷き詰められた盤面から，いくつかの石を取り除いて構成した，囲碁のルールに反しない盤面を**「市松模様の盤面」**と呼ぶことにする．

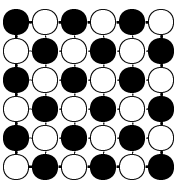

図 5: 敷き詰められた盤面

2.2 市松模様の盤面の定理

定理 1 連数Nの合法な盤面が存在した時，連数Nの市松模様の盤面が少なくとも1つ存在する

この定理より，市松模様の盤面に存在できる連数の最大値はMSPの解と一致するため，市松模様の盤面の最大の連数を求めれば良い．

2.3 定理 1 の証明

補題1，補題2を連結すると主張と一致する．□

補題 1 連数Nの合法な盤面が存在した時，全ての連のサイズが1である連数Nの盤面が少なくとも1つ存在する

補題 1 の証明 連数Nの任意の盤面の中にサイズ2以上の連が存在するとき，全ての連に対して，サイズが1になるまで碁石を取り除いても連数は減少しない．□

具体例をあげたのが，図6，7である．図6の盤面の連数は6である．図7の盤面は図6から石を抜いて，全ての連のサイズを1にした盤面であり，連数は6である．このように，石を抜いても連数は減少することはない．

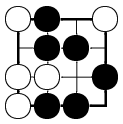

図 6: 元盤面

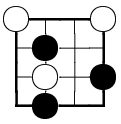

図 7: 除去後

補題 2 全ての連のサイズが1である任意の連数Nの盤面が存在したとき，連数Nの市松模様の盤面が少なくとも1つ存在する

補題 2 の証明 色は問わずに上下左右に隣接した石の集合を大連と定義する．任意の合法的な盤面が存在し，その盤面の全ての石がいくつかの大連に分かれたとする．その盤面の任意の大連について色を反転した盤面も合法な盤面になっている．

全ての連のサイズが1である盤面を大連に分ける．市松模様と色の合っていない大連の色を反転すれば，市松模様の盤面が構成できる．□

具体例を示す．図8はそれぞれの大連に印を付けた例であり，図9が右の大連を反転した図である．このように，それぞれの大連は独立している（干渉しない）ため，大連の色を反転させても合法な盤面のままとなる．

図 8: 元盤面

図 9: 反転後

2.4 市松模様への変形の具体例

実際に連数を減らさずに市松模様に変形する具体例をあげる．

図 6 → 図 10 → 図 11 が市松模様への変形になっている[2] .

最初の変形は，補題 1 の連サイズを 1 にする変形である．次の変形は，定理 1 の市松模様の色にあわせるための反転である．ここでは左上の◎を反転して市松模様にしているが，◎以外の石をすべて反転して市松模様にしてもよい．

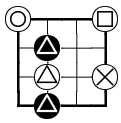

図 10: 除去後

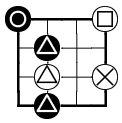

図 11: 市松模様

2.5 盤面の表現方法

市松模様の盤面では，座標で石の色は決まっている．従って，各交点について石が有る or 無いの 2 通りの情報があれば市松模様の盤面を定義できる．市松模様の盤面全体を行列 $\boldsymbol{x}$，それぞれの交点を式 (1) で表現する．

$$x_{i,j} = \begin{cases} 1 & \cdots \quad \text{石がある} \\ 0 & \cdots \quad \text{空点} \end{cases} \tag{1}$$

たとえば，図 11 を行列で表現したのが，図 12 となる．

1	0	0	1
0	1	0	0
0	1	0	1
0	1	0	0

図 12: 行列表現

3 MSP の 0-1 整数計画問題への定式化

市松模様に限定した MSP の行列表現について，現れてはいけないパターンを明確にして，数理計画問題 ([2]) として定式化を行った．

[2] 図 10 は，図 7 の盤面と同じであるが，各大連に印が付けてある．

3.1 現れてはいけないパターンの定式化

市松模様の盤面に限定して考える．行列表現で考えた時，図 13 の 3 つが合法な盤面で現れてはいけないパターンとなる．つまり，このパターンが現れないという制約の下で，1 の個数 (石の個数) を最大化する，という問題を考えればよい事が解る．

	1	
1	1	1
	1	

(a) 中央部分

1	1	
1		

(b) 隅部分

1		
1	1	
1		

(c) 辺部分

図 13: 現れてはいけないパターン

パターン (図 13) を利用して，0-1 整数計画問題として定式化できる．例えば，図 13(a) 中央部分は，

$$x_{i,j} = x_{i-1,j} = x_{i+1,j} = x_{i,j-1} = x_{i,j+1} = 1$$

である時，$x_{i,j}$ の呼吸点が無いため非合法な盤面，という意味である．これは整数計画問題の条件で表現するなら，

$$x_{i,j} + x_{i-1,j} + x_{i+1,j} + x_{i,j-1} + x_{i,j+1} \leq 4$$

となる．

また，端，辺部分も同様に考える事ができる．たとえば，左上の隅の石の呼吸点についてでは，

$$x_{1,1} + x_{1,2} + x_{2,1} \leq 2$$

となる．辺 (ここでは $x_{1,3}$ の呼吸点ついて) では，

$$x_{1,3} + x_{2,3} + x_{1,2} + x_{1,4} \leq 3$$

のように表現できる．

3.2 MSP の定式化

上で述べたことから，市松模様に限定した MSP(n) は式 (2) のように，0-1 整数計画問題と

して定式化できる．

$$
\begin{aligned}
&\textbf{最大化：}\textstyle\sum_{i=1}^{n}\sum_{j=1}^{n} x_{i,j}\\
&\textbf{条件：}\\
&x_{i,j}+x_{i-1,j}+x_{i+1,j}+x_{i,j-1}+x_{i,j+1} \le 4 \quad (2\le i\le n-1, 2\le j\le n-1)\\
&x_{i,j}+x_{i+1,j}+x_{i,j-1}+x_{i,j+1} \le 3 \quad (i=1, 2\le j\le n-1)\\
&x_{i,j}+x_{i,j+1}+x_{i-1,j}+x_{i+1,j} \le 3 \quad (2\le i\le n-1, j=1)\\
&x_{i,j}+x_{i-1,j}+x_{i,j-1}+x_{i,j+1} \le 3 \quad (i=n, 2\le j\le n-1)\\
&x_{i,j}+x_{i,j-1}+x_{i-1,j}+x_{i+1,j} \le 3 \quad (2\le i\le n-1, j=n)\\
&x_{1,1}+x_{1,2}+x_{2,1} \le 2\\
&x_{n,1}+x_{n,2}+x_{n-1,1} \le 2\\
&x_{1,n}+x_{1,n-1}+x_{2,n} \le 2\\
&x_{n,n}+x_{n,n-1}+x_{n-1,n} \le 2\\
&x_{i,j} \le 1 \quad (1\le i,j\le n)\\
&x_{i,j} \ge 0 \quad (1\le i,j\le n)\\
&\textbf{ただし，}x_{i,j}\textbf{は整数}
\end{aligned}
\tag{2}
$$

3.3 数値実験

MSP の定式化 (式 (2)) を，GLPK で解いた．GLPK とは，GNU Linear Programming Kit の略であり，線形計画問題，整数計画問題，混合整数計画問題を解く事が出来るソルバである．

計算機環境は，

OS MacOSX version 10.4.8

CPU Intel Core Duo 2Ghz

メモリ 2GB 667 MHz DDR2

ソルバ GLPK version 4.9

を使用した．

GLPK のデフォルトの設定と，オプションに`--last` を指定した 2 通りの方法で解いた．`--last` オプションは，分枝する変数の順番を降順に固定するオプションである．GLPK のデフォルトはヒューリスティックに順番を決める方法が使用されている．

GLPK で解いた結果が，表 2 である．最適解の盤面は付録として添付した．

n	連数	密度	`--last` (秒)	`default` (秒)
2	2	0.50	0.0	0.0
3	6	0.66	0.0	0.0
4	12	0.75	0.0	0.0
5	18	0.72	0.0	0.0
6	26	0.72	0.0	0.0
7	36	0.73	0.0	0.0
8	49	0.76	0.0	0.0
9	61	0.75	0.0	0.0
10	76	0.75	1.0	1.0
11	92	0.76	0.0	8.0
12	109	0.75	23.0	48.0
13	129	0.76	11.0	88.0
14	149	0.76	4171.0	4788.0
15	169	0.76	3544.0	5547.0

表 1: 結果

14 路盤で時間が爆発的に増えている．`--last` オプションを付けたほうが，すこし高速に解けている．これは条件式がいびつでなく，対称的であるため，ヒューリスティックが巧く働いていないのではないかと予想される．

19 路盤についても`--last` オプションを付けて，実行してみた．丸一日ほど実行したところで，下界 277，上界 282 が得られたが，1 週間や 2 週間では終わりそうもなかった．

4 定式化の工夫

表 2 より，式 (2) のみでは MSP(19) は解けそうも無いので，妥当不等式を追加して高速化することを行った．この式の条件では回転，反転すると同じ盤面にであるような対称な盤面をまったく気にしていない．対称な盤面 (の一部) を除去する妥当不等式を条件式に加えること考えた．

4.1 2 分割

盤面を 2 つの領域 U，D に分割する．(ただし，ここでは簡単のため，盤サイズ n は偶数であり，U，D は同じ大きさとする)

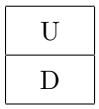

図 14: 2 分割

このとき，領域 U の連数と領域 D の連数についての条件式 (3)

$$\sum_{(i,j)\in U} x_{i,j} \ge \sum_{(i,j)\in D} x_{i,j} \tag{3}$$

を，MSP の定式化 (式 (2)) に追加した問題を考えてみる．

式 (3) の左辺，右辺を $U(\boldsymbol{x})$, $D(\boldsymbol{x})$ と定義する．任意の盤面 B は，

$$U(\boldsymbol{x}) \ge D(\boldsymbol{x}) \tag{4}$$

$$U(\boldsymbol{x}) \le D(\boldsymbol{x}) \tag{5}$$

のどちらかは必ず満たしている．盤面 B が $U(\boldsymbol{x}) \leq D(\boldsymbol{x})$ であった時でも，盤面 B を 180 回転 (もしくは裏返) した盤面 B' は，$U(\boldsymbol{x}) \geq D(\boldsymbol{x})$ を満たしている．従って，式 (3) を見たす最適盤面 B も必ず存在するので，この式を追加しても最適値に変化は無いし，最適解も正しく求まる．

4.2 4 分割

2 分割と同じように考えれば，4 分割も可能である．サイズが偶数の盤面で，領域 A，B，C，D が同じサイズとして，$A(\boldsymbol{x})$, $B(\boldsymbol{x})$, $C(\boldsymbol{x})$, $D(\boldsymbol{x})$ も $U(\boldsymbol{x})$, $D(\boldsymbol{x})$ と同じように，領域内の石の数とする．

A	B
D	C

図 15: 4 分割

4 分割では，

$$A(\boldsymbol{x}) \geq C(\boldsymbol{x}) \tag{6}$$

$$B(\boldsymbol{x}) \geq D(\boldsymbol{x}) \tag{7}$$

この 2 つの式を定式化 (式 (2)) に追加する．任意の盤面 B を対角線で裏返した盤面の中には，式 (6), (7) の両方を満たす盤面 B' が存在する．従って，式 (6), (7) を見たす最適盤面 B^* も必ず存在している．

4.3 奇数サイズの盤面

4.1, 4.2 では，偶数サイズの盤面としてきた．目標とする 19 路盤で盤面の分割を行うためには，真ん中の石の列 (行) の扱いを，少しだけ工夫をする必要がある．方法はいくつかあるが，たとえば，図 16 のように領域を分割して，真ん中の石の列 (行) は使わない方法である．このように分割すれば，偶数サイズの盤面と同じように考えることができる．

もう一つの方法は，図 17 のように，真ん中の石の列 (行) を A，C に含める方法である．この方法でも，偶数サイズと同じ様に考える事ができる．[3]

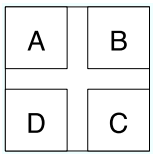

図 16: 使わない

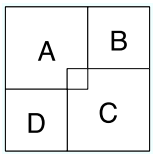

図 17: A，C に含める

4.4 数値実験

MSP の定式化に 4 分割の図 17 の定式化を加えたものを，GLPK で解いた．計算機環境は，同じ環境を使用している．

[3] 2 分割でも，4 分割でも，対角線で領域を分けるという方法もある．対角線で分けると，偶数と奇数を気にしなくてもよい．

N	連数	密度	`--last`(秒)	default(秒)
2	2	0.50	0.0	0.0
3	6	0.66	0.0	0.0
4	12	0.75	0.0	0.0
5	18	0.72	0.0	0.0
6	26	0.72	0.0	0.0
7	36	0.73	0.0	0.0
8	49	0.76	0.0	0.0
9	61	0.75	0.0	0.0
10	76	0.75	0.0	1.0
11	92	0.76	1.0	5.0
12	109	0.75	24.0	39.0
13	129	0.76	12.0	29.0
14	149	0.76	4426.0	2191.0
15	172	0.76	3519.0	3161.0

表 2: 結果

`--last` オプションを付けた場合では，とくに速度向上は見られなかった．しかし，default の場合では，だいたい半分の時間で解く事ができるようになっている．条件式の追加により，対称性が崩れてヒューリスティックが巧く働いているのではないかと思われる．

19 路盤についても実行してみた．前回と同様，丸一日ほど実行したところで，下界 277，上界 281 が得られた．前回の上界は 282 なので，追加した条件式の効果が出ているといえる．丸 3 日ほど動かし続けてみたところでは，上界は 281 であった．

5 おわりに

今回の実験では MSP(19) の下界が 277，上界が 281 という結果が得られた．しかし，この問題に興味をもってくださった方々がおり，実はつい先日，東京農工大学の宮代隆平氏が商用ソルバ CPLEX を用いて，40 日にもなる計算により MSP(19) を解いた，と報告された ([4])．氏によれば最適値は 277 であるという．我々はこの結果を未だ検証していない．

この研究の結果を連情報を配列で持つ場合の配列サイズに使用して貰えれば幸いである．

MSP(19) に関して情報を寄せてくれた宮代隆平氏，いち早く興味をもってくださった山本芳嗣氏にこの場を借りて感謝をいたします．

参考文献

[1] 清慎一, 山下宏, 佐々木宣介. コンピュータ囲碁の入門. コンピュータ囲碁フォーラム (編). 共立出版,2005.

[2] 田村明久, 村松正和. 最適化法. 共立出版,2002.

[3] GLPK, http://www.gnu.org/software/glpk/

[4] 宮代隆平，*private communication.*

A　求まった盤面

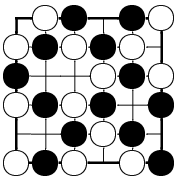

図 18: MSP(6)=26

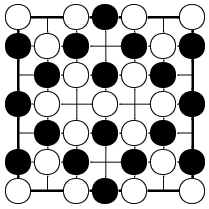

図 19: MSP(7)=36

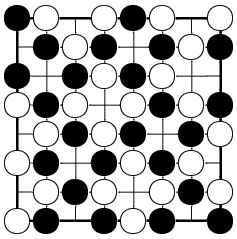

図 20: MSP(8)=49

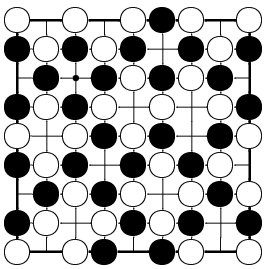

図 21: MSP(9)=61

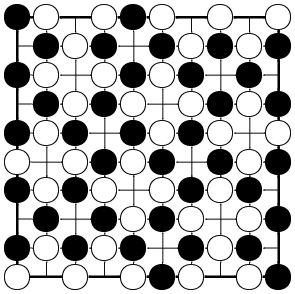

図 22: MSP(10)=76

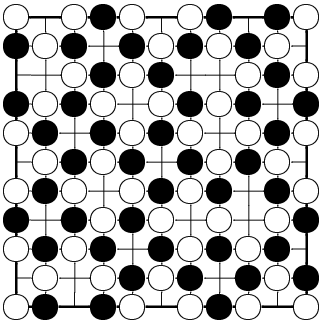

図 23: MSP(11)=92

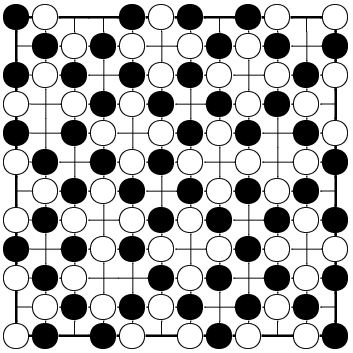

図 24: MSP(12)=109

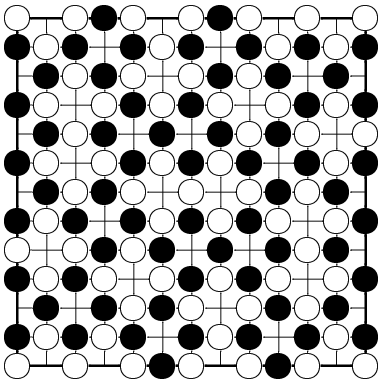

図 25: MSP(13)=129

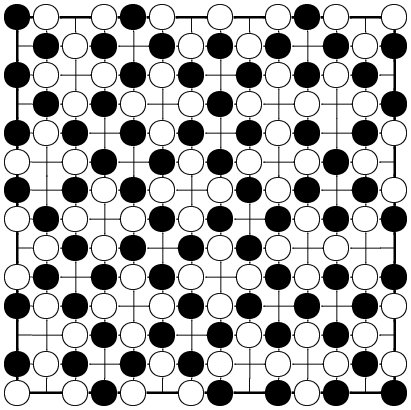

図 26: MSP(14)=149

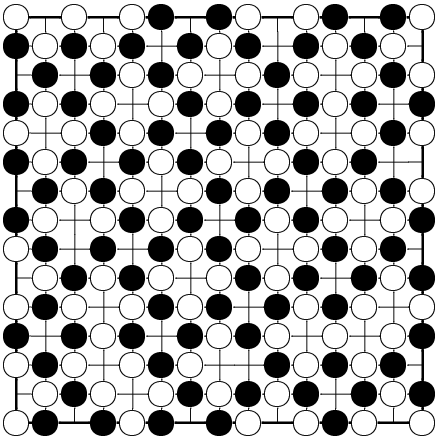

図 27: MSP(15)=172

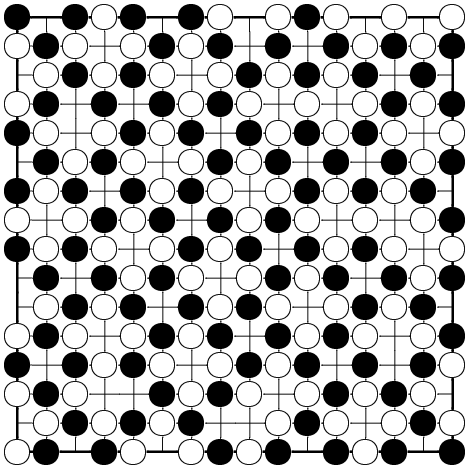

図 28: MSP(16)=196

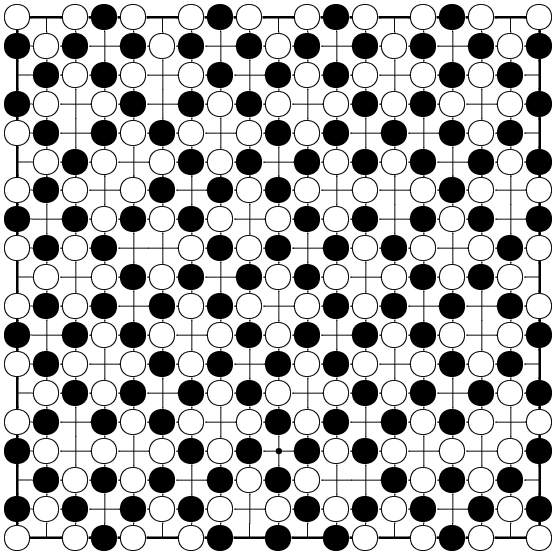

図 29: MSP(19)=277 ([4])