3　ヒューリスティクスによる解法

3.1　解法

検索による方法では，N が大きくなるに従って，検索回数が急激に増加してしまう．次は，検索アルゴリズムを用いない方法で N クイーン問題を解いてみよう．ここでは，問題の条件に対してエネルギー関数 (目的関数，コスト関数) を定義する．解のエネルギーが最小になるようにエネルギー関数 E を設計することによって，最小値を求める問題に帰着することができる．

まず，エネルギー関数 E を設計しよう．マス (i,j) の状態を x_{ij} とし，クイーンが置かれている場合に $x_{ij}=1$，置かれていない場合に $x_{ij}=0$ をとるものとする．ただし，ここでは $x_{ij}=0.4$ のように実数値 $0 \le x_{ij} \le 1$ をとることを許す連続緩和問題を考える．条件 1. を満たすエネルギー関数 E_1 は，$x_{ij}=0,1$ のときに最小になれば良いので，

$$E_1 = \frac{c_1}{2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{ij}\left(1-x_{ij}\right), \tag{1}$$

と表すことができる．但し，c_1 は定数である．全ての x_{ij} が 0 または 1 をとれば $E_1=0$ になることを確認しておこう．同様に，条件 2. と 3. を満たすエネルギー関数 E_2, E_3 は，それぞれ各行または各列にクイーンが 1 つしかないときに最小になれば良いので，

$$E_2 = \frac{c_2}{2} \sum_{i=0}^{N-1} \left(\sum_{j=0}^{N-1} x_{ij} - 1 \right)^2, \tag{2}$$

$$E_3 = \frac{c_3}{2} \sum_{j=0}^{N-1} \left(\sum_{i=0}^{N-1} x_{ij} - 1 \right)^2, \tag{3}$$

と表すことができる．さらに，条件 4. と 5. を満たすエネルギー関数 E_4, E_5 は，

$$E_4 = \frac{c_4}{2} \sum_{n=0}^{2(N-1)} \sum_{\substack{i+j=k+l=n \\ k \neq i}} x_{ij} x_{kl}, \tag{4}$$

$$E_5 = \frac{c_5}{2} \sum_{n=-N+1}^{N-1} \sum_{\substack{i-j=k-l=n \\ k \neq i}} x_{ij} x_{kl}, \tag{5}$$

と表すことができる．ただし、$\sum_{\substack{i+j=k+l=n \\ k \neq i}}$ は、添字 (i,j) と (k,l) の和 $i+j$ と $k+l$ が等しい場合についての和を求める演算を表す。以上より，N クイーン問題のエネルギー関数 E は，

$$E = E_1 + E_2 + E_3 + E_4 + E_5 \tag{6}$$

となる．

E_4, E_5 は次のように表現することも可能である。

$$E_4 = \frac{c_4}{2} \sum_{i,j} \sum_{k,l} \delta_{i+j,k+l} x_{ij} x_{kl}, \tag{7}$$

$$E_5 = \frac{c_4}{2} \sum_{i,j} \sum_{k,l} \delta_{i-j,k-l} x_{ij} x_{kl}, \tag{8}$$

ここで，$\delta_{x,y}$ はクロネッカーのデルタ関数であり，

$$\delta_{x,y} = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{9}$$

である．

エネルギー関数 E が求められたので，この関数の最小値を求めれば解が得られるはずである．そこで，エネルギー関数 E を x_{ij} で偏微分し，その勾配方向を求めておく．勾配方向に沿って下って行けば，**極値**に到達する．それでは，エネルギー関数 E を x_{ij} で偏微分してみよう．

$$\frac{\partial E_1}{\partial x_{ij}} = \frac{c_1}{2} - c_1 x_{ij}, \tag{10}$$

$$\frac{\partial E_2}{\partial x_{ij}} = c_2 \left(\sum_{l=0}^{N-1} x_{il} - 1 \right), \tag{11}$$

$$\frac{\partial E_3}{\partial x_{ij}} = c_3 \left(\sum_{k=0}^{N-1} x_{kj} - 1 \right), \tag{12}$$

$$\frac{\partial E_4}{\partial x_{ij}} = c_4 \sum_{k,l} \delta_{i+j,k+l} x_{kl}, \tag{13}$$

$$\frac{\partial E_5}{\partial x_{ij}} = c_5 \sum_{k,l} \delta_{i-j,k-l} x_{kl}, \tag{14}$$

これより，

$$\frac{\partial E}{\partial x_{ij}} = \frac{\partial E_1}{\partial x_{ij}} + \frac{\partial E_2}{\partial x_{ij}} + \frac{\partial E_3}{\partial x_{ij}} + \frac{\partial E_4}{\partial x_{ij}} + \frac{\partial E_5}{\partial x_{ij}} = 0, \tag{15}$$

となる x_{ij} を求めればよい．

3.2 最急降下法

エネルギー関数 E は x_{ij} に関して，次のような二次形式になっている．

$$E = \sum_{i,j} \sum_{k,l} W_{ijkl} x_{ij} x_{kl} + \sum_{i,j} \theta_{ij} x_{ij}, \tag{16}$$

ただし、$\sum_{i,j} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1}$ を表す。このエネルギー関数の勾配を求めると，

$$\frac{\partial E}{\partial x_{ij}} = \sum_{k,l} W_{ijkl}\, x_{kl} + \theta_{ij}, \tag{17}$$

となる．これより，極値に到達するためには，勾配方向を逆向きに進めばよい．したがって，次の方程式にしたがって状態更新をすれば良い．

$$\frac{du_{ij}(t)}{dt} = -\frac{\partial E}{\partial x_{ij}} \tag{18}$$

$$= -\sum_{k,l} W_{ijkl}\, x_{kl}(t) - \theta_{ij}, \tag{19}$$

$$x_{ij}(t) = F\left(u_{ij}(t)\right), \tag{20}$$

ここで，$u_{ij}(t)$ は第 (i,j) 番目のマスの内部状態を表す．出力関数 F は内部状態 $u_{ij}(t)$ に応じて，出力 $x_{ij}(t)$ を決める関数である．出力関数には以下のような関数が挙げられる．

- ステップ関数

$$F(u) = \begin{cases} 1, & u > 0 \\ 0, & u \leq 0 \end{cases}, \tag{21}$$

- アナログしきい関数

$$F(u) = \begin{cases} u, & u > 0 \\ 0, & u \leq 0 \end{cases}, \tag{22}$$

- 区分線形関数

$$F(u) = \begin{cases} 1, & 1 < u \\ u, & 0 < u \leq 1 \\ 0, & u \leq 0 \end{cases}, \tag{23}$$

- シグモイド関数 (定数 $\beta > 1$)

$$F(u) = \frac{1}{1 + \exp(-\beta u)}, \tag{24}$$

それでは，N クイーン問題の条件を最急降下法に適用してみよう．簡単化のために，$c_1 = c_2 + c_3$ とする．式 (10)〜(14) と式 (17) の係数を比較することによって，係数 W_{ijkl} と θ_{ij} を決定する．従って，

$$W_{ijkl} = c_2\delta_{i,k}(1-\delta_{j,l}) + c_3\delta_{j,l}(1-\delta_{i,k}) + c_4\delta_{i+j,k+l}(1-\delta_{i,k}) + c_5\delta_{i-j,k-l}(1-\delta_{i,k}), \tag{25}$$

$$\theta = -\frac{c_2 + c_3}{2}, \tag{26}$$

となる．

以上より，係数が決まれば，微分方程式 (19) を数値計算することによって，解を求めることができる．微分方程式の求解法にはオイラー法やルンゲクッタ法などのアルゴリズムが知られている．

3.3 メトロポリス法

最急降下法では，勾配方向下向きに状態更新を行うため，エネルギー関数に局所解（ローカルミニマム）が存在すると，最適解が得られない．また，最適解の近傍では，勾配が小さいため，収束に時間がかかる問題がある．そこで，局所解から脱出可能な手法として，メトロポリス法が知られている．

メトロポリス法とは，現在の状態 $\mathbf{x}$ のエネルギー E に対して，次の候補の状態 $\mathbf{x}'$ のエネルギー E' が，減少している場合は，その状態を採用し，増加している場合は，ある確率で採用する方法である．これは，**焼きなまし法**の方法の 1 つである．状態更新によって生じるエネルギーの差を

$$\Delta E = E' - E, \tag{27}$$

とすると，状態更新する確率は，

$$P[\mathbf{x} \to \mathbf{x}'] = \begin{cases} 1 & , \quad \Delta E \leq 0 \\ \exp(-\Delta E/T) & , \quad \Delta E > 0 \end{cases}, \tag{28}$$

と表すことができる (図 4)．ここで，T は温度パラメータである．

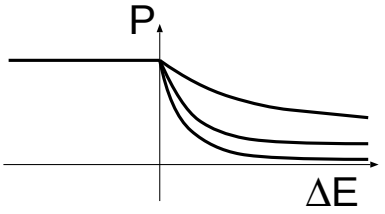

図 4: 遷移確率

3.4 課題 2

1. メトロポリス法を用いて，N クイーン問題を解きなさい．ただし，$c_2 = c_3 = c_4 = c_5 = 1.0$ とする．また，温度パラメータ $T > 0$ を定数で与える．レポートにはプログラムと実行結果を添付しなさい．
2. エネルギー関数 (16) より，各時刻のエネルギーを求めなさい．レポートにはこのグラフをいくつか示しなさい．
3. $N \geq 4$ の場合について，何回か実行し，はじめて解が得られるまでに要した反復回数をグラフにしなさい．横軸に N を，縦軸に反復回数の平均値をとること．
4. 式 (1)〜式 (5) を偏微分して，式 (10)〜式 (14) を導出しなさい．レポートには，導出過程を詳しく記述すること．
5. 探索による解法とメトロポリス法のそれぞれの方法について，両者を比較しながらその利点と欠点を挙げなさい．

3.5 プログラミングのヒント

3.5.1 データ構造

- マスの状態を表す配列を x[N][N] とする．ただし，x[i][j]=1 のときにマス (i,j) にクイーンが置かれているとし，x[i][j]=0 のときに置かれていないとする．
- エネルギー関数の係数を W[i][j][k][l] とする．

3.5.2 関数

○ **デルタ関数**：式 (9) で与えられるクロネッカーのデルタ関数を作成しなさい．

```
int delta( int x, int y ) {  /*** クロネッカーのデルタ関数 ***/
   ...
}
```

○ **エネルギー関数の係数の生成**：係数 W[i][j][k][l] の値は (25) で与えられる．デルタ関数を用いて，全ての W[i][j][k][l] に値を代入する関数を作成しなさい．また，$n = 5, i = j = 2$ の場合における係数の値 (W[2][2][k][l]) を表示し，正しく作れているかを確認しなさい．

```
void make_W( double W[N][N][N][N], int n ) {
   int i,j,k,l;
   /*** W[i][j][k][l]= ... ***/
}
```

○ **エネルギー関数**：式 (16) で与えられるエネルギー関数を作成しなさい．すなわち，状態 $\mathbf{x}$ が与えられたときに，そのときのエネルギーの値を返す関数を作成しなさい．ただし、$\theta_{ij} = 0$ としてもよい。この場合、最小値は 0 になる。

```
double energy( int x[N][N], double W[N][N][N][N], int n ) {
   double E=0.0;
   ...
   return (E);
}
```

○ **ニューロンの出力の表示**：課題 1 と同様に用意すること．

```
void print_queens( int x[N][N], int n ) {
...
}
```

○ **解が求まったかを判定する関数**（使わなくても作成可能）：課題 1 の関数 check() を用いて，与えられた N 個のクイーンの配置が条件を満たしている (返り値 0) か，満たしていない (返り値 1) かを判定する関数を作成しなさい．

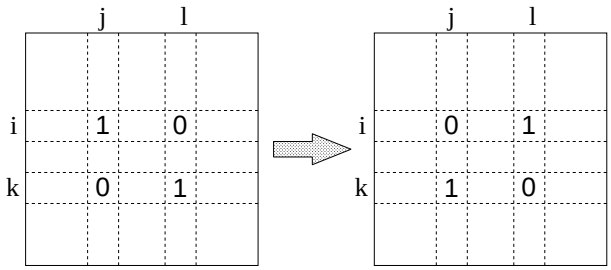

図 5: 近傍の状態の選び方

```
int success( int x[N][N], int n )  {
   int done; /*** 返り値 ***/
   /*** クイーンが置かれている場所の関数 check() の値を調べる.
        全て 0 であれば正解が求まったので，done=0 にする
   ***/
   return (done);
}
```

3.5.3 メイン関数

メトロポリス法のアルゴリズムは次のようになる.

1. 初期状態（初期解）を決める．ただし，各行と各列で，クイーンの数が 1 つとなるように選ぶ．例えば，単位行列でもよい.
2. 温度パラメータを $T > 0$ にする.
3. 現在の状態 $\mathbf{x}$ の近傍の状態を 1 つ選ぶ．これを候補の状態 $\mathbf{x}'$ とする．図 5 のように，$\mathbf{x}'$ は状態 $\mathbf{x}$ の中で 1 となっている 2 つの位置を交換すればよい．このようにすれば，各行と各列のクイーンの数が変化しない.
4. 候補の状態 $\mathbf{x}'$ に対するエネルギー E' を求め，エネルギーの差 $\Delta E = E' - E$ を求める.
   (a) $\Delta E \leq 0$ ならば，$\mathbf{x}'$ を新しい状態 $\mathbf{x}$ に採用する.
   (b) $\Delta E > 0$ ならば，乱数 r を求め，$\exp(-\Delta E/T)$ よりも小さい ($r < \exp(-\Delta E/T)$) ときは，$\mathbf{x}'$ を新しい状態 $\mathbf{x}$ に採用する．大きいときは，状態を更新しない.
5. 新しい状態が最適解になっていれば終了する。そうでなければ，3. より繰り返す.

乱数の発生方法 : 実行毎に異なる結果を必要とするので，乱数を用いる．乱数は C 言語の関数 `drand48()` を用いれば良い．`drand48()` は 0 から 1 の間の実数をランダムに返す関数である．ただし，`drand48()` を使う前に一度，関数 `srand48()` でシード (乱数の種) を与える必要がある.

以上より，main() 関数では以下のような流れになる．

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#define N 20
int main( void ) {
   int n, i,j, k, l;
   int x[N][N], xd[N][N];// 状態 x と次の状態の候補 xd
   double W[N][N][N][N]; // 係数 W
   double Ec, En, dE;    // 現在のエネルギー，候補のエネルギー，エネルギーの差分
   time_t tp;
   time(&tp); srand48(tp);
   /*** クイーン数 n の入力 ***/
   /*** 配列 W の作成 ***/
   /*** 状態 x[i][j] の初期値を与える & エネルギー ***/

   /*** ループ : 終了判定 ※1（解が求まったら終了） ***/
      /*** 近傍の候補の状態を作成 ※2 ***/
      /*** 候補のエネルギーを求める ***/
      /*** エネルギーの差分 ΔE ***/
          /*** ΔE<=0 ならば候補の状態を新しい状態にする ***/
          /*** ΔE>0  ならば確率 exp(-ΔE/T) で新しい状態を採用 ※3 ***/
   /*** ループ終了 ***/
   /*** 解の表示 ***/
   return 0;
}
```

※1. 終了判定は以下のいずれか (または組み合わせ) で行うことができる．

- 一定回数のループを回ったとき
- エネルギーが 0 になったとき（関数 success() の代用としてもよい）

※2. 新しい候補の状態は次のように作成することができる．
i 行目に置かれているクイーンの位置 j を配列 P に保存しておく（P[i]=j）．

1. 新しい候補の状態 xd の初期値は現在の状態 x とする（x のコピーを xd とする）．
2. 交換する位置として，異なる i, k をランダムに選ぶ．
3. 状態 xd の位置 (i,P[i]) と (k,P[k]) のクイーンを取り，交換した位置にクイーンを置き直す．

※3. 新しい候補の状態 xd を状態 x に代入する．このとき，配列 P も更新する．