理工学部 数物・電子情報系学科 情報工学 EP
春学期講義「人工知能」
担当：長尾 智晴
総合研究棟 S401 室（内線 4131）nagao@ynu.ac.jp

【重要】平成 30 年度「人工知能」プログラミング課題レポート

1. 作成するプログラムについて

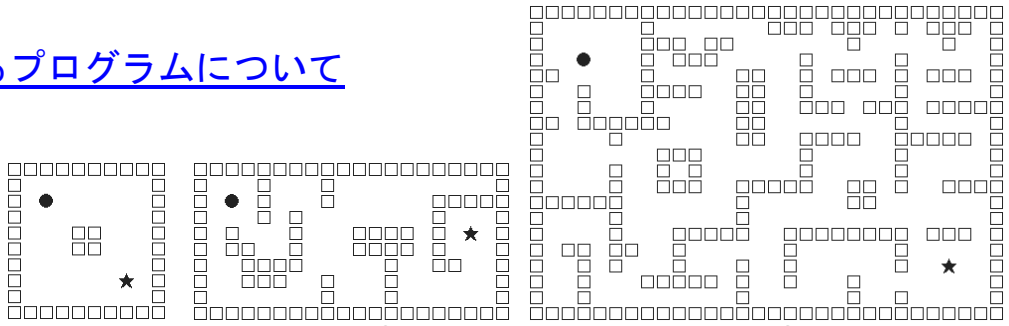


図1 マップの例（それぞれ map**n**.c (**n**=1,2,3)によって定義され，最初にインクルードする）

- **図1**に例示するマップの**スタート地点**（●）に居るエージェントを，壁にぶつからず，かつ 1 歩前の場所には戻らないようにしながら**ゴール地点**（★）まで移動させる即応ルールを遺伝的アルゴリズム（Genetic Algorithm; GA）を使って求めるプログラム（1種．ga.c（インクルードするマップファイルを変更することで全てのマップに対応させる））を作ることが目的である．
- ここで，即応ルールの**入力**は**図2**に示すようにエージェントを中心とする 3×3 のマス目を，壁を 1，壁以外を 0 として 8bit（0～255）で数値化したものとする（方位は絶対方位，場所を示す番号（0～7）はエージェントの向き・進行方向によらない．また，行動**出力**は**図3**に示す**上下左右の 4 通り**（0～3）（**こちらも絶対方位でエージェントの向き・進行方向によらない**）とする．

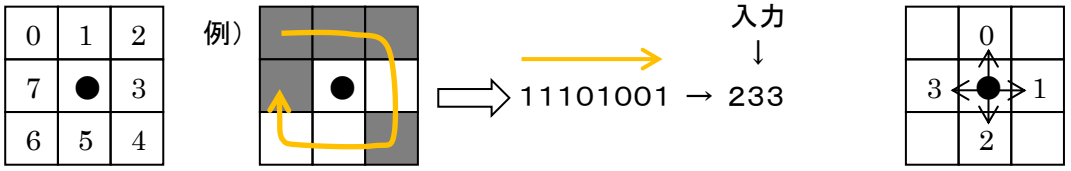


図2 エージェントの周囲の数値化（エージェントの環境入力）（255通り）　図3 行動出力（4通り）

- つまり，ここで**作らなければならない即応ルール**は，エージェントのセンサ入力（0～255）に対して，そのときの適切な行動出力（0～4）を示す**表 1** のような表である．

注：これは正解例ではない．あくまでも一例である．

表 1 求めるべき即応ルール（白い部分にどのような数字を入れれば良いかが問題）

センサ入力値（3×3 近傍を 0~255 に数値化したもの）	0	1	2	3	・・・	253	254	255
行動出力値（0~3 のいずれか）	3	0	3	2	・・・	2	0	1

- 2.で示すサンプルプログラム ga.c においては，次に示す 2 次元配列がこの表に相当する．
  int genotype[POP][IN];　（POP：個体識別番号，IN：入力 0～255 に対する行動出力値）

注：これは正解例ではない．あくまでも一例である．

個体識別番号 POP(=100)	センサ入力値 IN (0～255)							
	0	1	2	3	・・・	253	254	255
0	3	0	3	2	・・・	2	1	3
1	2	3	0	0	・・・	1	2	3
・・・	・・・	・・・	・・・	・・・	・・・	・・・	・・・	・・・
POP-1(=99)	2	3	3	1	・・・	3	2	1

- 壁にぶつかるか，あるいは 1 歩前の場所に戻ってしまったらその時点で終了とする．

2. サンプルプログラム(ga.c)について

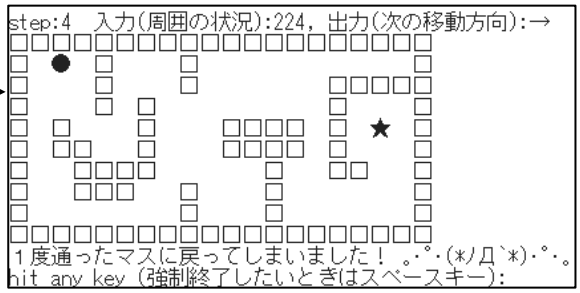
- ゼロから作るのは大変なので,「ひな型」のプログラム(ga.c)が用意されている. このプログラムの中の次の 2 つの関数(今は何も実行しない関数になっている)の中身を完成させれば良い.

void calc_fitness(int steps)	各個体(No.0～POP-1)の適応度(fitness[ ])を計算する.
void determine_next_generation( )	シンプル GA 方式(ルーレット選択)＋エリート保存戦略で次世代の POP 個の個体(の遺伝子型)を決定する. ルーレット選択をして POP 個の個体を決定後, 2つずつランダムに組み合わせて1点交叉させる(交叉の発生確率が定数宣言されている). 次に全個体の全遺伝子に突然変異を実行する(0～3 をランダムに 0～3 に変更する. 生起確率は定数宣言されている). 最後に, No.0 の個体に現世代のエリート(最良個体)を戻す.

3. 期待される回答のプログラムの動作について

- 完成後のプログラムの例 (map2.c をインクルードした場合) の動作は次のようになる (参考までに, 回答例の実行型ファイル ga1.exe, ga2.exe, ga3.exe を提供するので, Windows パソコン上で実行して確認してほしい).

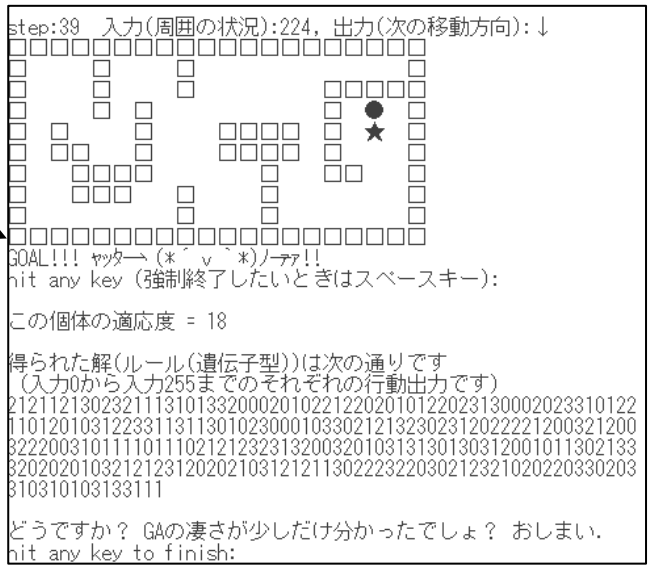
1. 最初に, 初期個体群の No.0 の個体の動きを画面に表示する. 右図では, スタート地点から4ステップ目で1歩前に居た場所に戻って終了している.



2. 次に最大世代数を指定して GA によるルールの最適化を実行する. ここでは第 182 世代で解を得ている.

```
この個体の適応度 = 0
ランダムに作ったので全然ダメですね. . .
===== それではルールをGAで進化させましょう =====
最大世代数(例：30000) = 30000
世代 No.0:最大適応度 = 4
世代 No.100:最大適応度 = 15
世代 No.182:最大適応度 = 18
ゴールに到達するルールが得られたので世代交代を終了しました.
得られた解を示します. hit any key:
```

3. 最後に, 得られた最良個体の動きを何かのキーを押すことで画面に1ステップずつ表示すし, 得られたルール(遺伝子型)を表示する(入力 0～255 に対する出力(0～3)を羅列する).



4. 報告義務について

- インクルードするマップファイルを変更して色々なマップに対する実験ができる **ga.c** の完成版と, 自分で作ったオリジナルのマップ **map4.c** (横と縦のサイズは, いずれも map3.c より大きいことを条件とする)を下記の要領で提出する.
- 提出期限は,

2018 年 8 月 8 日(水) 24:00

(8 月 9 日(木)の午前 0 時)まで とする.

- 必ず, メールの件名を「学籍番号　氏名」(例：1564000　横浜太郎) として下さい.
- メールの中身は次を含むようにすること(添付ファイルを付けずにメール中にテキストで入れて下さい): 添付ファイル付きメールは迷惑メール扱いされる可能性があります. どうぞよろしくお願いします.
  - 自分で改良して完成させた **ga.c** のソースリスト, **map4.c** のソースリスト.
  - 考察
  - 感想(この課題や講義に関する感想・意見など)(採点対象外)
- 考察の内容としては, 例えばマップの壁の位置やエージェントの初期位置を変えたときの挙動や, そのときの GA による最適化に要する世代数の変化, 交叉方法を一様交叉に変えたときの挙動など, 基本的には何でも良い (受講生に任せる) が, ga.c を少し変更したときの挙動についてコメントすることを必要条件とする. 考察の内容と深さに応じてレポートの評点を変える予定である.

5. サンプルプログラム ga.c のソースリスト

```
/*************************************************************/
/* H30 年度情報工学 EP 提供講義「人工知能」レポート課題          */
/* GA（遺伝的アルゴリズム）による即応ルールの最適化(ga.c)     */
/*          担当：長尾智晴（nagao@ynu.ac.jp）TEL.045-339-4131 */
/*                                                           */
/* 提出期限：2018 年 8 月 8 日(水)24:00 までにメールで提出のこと.  */
/*   レポートの内容は次のようにすること（メール中にテキスト  */
/*   として記載すれば良いものとする．別紙の添付は不要.）     */
/*      ・メールの件名：学籍番号　氏名                       */
/*      ・ソースリスト                                       */
/*      ・考察                                               */
/*      ・感想（この課題 or 講義に関するもの（採点対象外））  */
/*************************************************************/
#include<stdio.h>
#include<stdlib.h>       /* for rand(), srand()  */
#include<time.h>         /* for time()           */
#include<conio.h>        /* for getch()          */

/************************ 定数宣言 ***************************/
#define IN         255   /* 入力の種類(8 ビット)              */
#define OUT          4   /* 出力の種類(4 通り)                */
#define POP        100   /* 個体総数                          */
#define CROSSOVER  0.7   /* 交叉率（交叉の発生確率(=70%)）    */
#define MUTATION   0.02  /* 突然変異率（=2%)                  */

/******************* マップの情報の読み込み ******************/
#include"map1.c"         /* int mapdata[WX][WY] などを初期化 */
int map[WY][WX];         /* 作業エリア */

/* 個体の情報 */
int genotype[POP][IN];     /* 遺伝子型 [No.][センサ入力値]    */
    /* 例) genotype[5][0]=1 --> No.5 の個体の周囲がブランク   */
    /*      (=0)のとき行動 1 を選ぶ．右辺は 0～OUT-1(=3)       */
/* 行動出力の移動方向（OUT 通り） */
int move_x[OUT] = { 0, 1, 0, -1 };
int move_y[OUT] = { -1, 0, 1, 0 };
int fitness[POP];           /* 個体の適応度（評価値）        */

/***************** 関数のプロトタイプ宣言 ********************/
void disp_map(void);                    /* マップを表示する   */
void init_population(void);             /* 個体群の初期化     */
void disp_action( int n, int steps );   /* No.n の個体を steps */
                                        /* 回だけ移動させる   */
int  obtain_input( int x, int y );      /* 周囲の状況を数値化 */
void generation( int n );               /* n 回世代交代させる */
/*************************************************************/
/* ★このプログラムは不完全で，下記の関数の中身がありません. */
/* ★下記の２つの関数を作ることが，ここでの課題です.          */
/*************************************************************/
void calc_fitness( int steps );    /* 各個体の適応度を求める */
void determine_next_generation( ); /* 次世代の個体群を求める */
/*************************************************************/


int main(void)
{
    int num,c;

    printf("¥n*** GA(遺伝的アルゴリズム)による即応プランの最適化デモ ***¥n");
    /* 初期個体群の生成 */
    init_population( );
```

```
    /* 初期個体の動きの例示 */
    printf("試しに初期個体 No.0 の動きを見てみましょう¥n");
    disp_action( 0, 100 );     /* No.0 の個体を 100step 動かす    */
    printf("ランダムに作ったので全然ダメですね... ¥n");
    printf("===== それではルールを GA で進化させましょう =====¥n");
    /* 最大世代数の入力 */
    printf("最大世代数(例：30000) = ");
    do{
        scanf("%d",&num);
    }while( num <=0 );
    /* num 回の世代交代で解を求める */
    generation( num );
    /* 終わり */
    printf("¥n どうですか？ GA の凄さが少しだけ分かったでしょ？ ");
    printf("おしまい. ¥n");
    printf("hit any key to finish:");
    c = getch();
    printf("¥n");

    return 0;
}


void disp_map(void)
/* 画面にキャラクタ（全角）でマップを表示する                          */
/* 0:空白(　), 1:壁(■), 2:エージェント(●), 3:Goal(★), 4:軌跡(　) */
{
    int x,y;  /* 制御変数 */

    for(y=0;y<WY;y++){
        for(x=0;x<WX;x++){
            switch ( map[y][x] ){    /* ← y,x の順 */
                case 0: printf("　"); break;  /* ブランク        */
                case 1: printf("□"); break;  /* 壁              */
                case 2: printf("●"); break;  /* エージェント    */
                case 3: printf("★"); break;  /* ゴール          */
                case 4: printf("　"); break;  /* 軌跡(見えない) */
            }
        }
        printf("¥n");
    }
}


void init_population(void)
/* 個体群の遺伝子型をランダムに初期化する */
{
    int i,j;  /* 作業変数 */

    /* 乱数の初期化 */
    srand((unsigned)time(NULL));     /* 時刻で初期化      */
    /* 遺伝子型の初期化 */
    for(i=0;i<POP;i++){              /* i:個体番号        */
        for(j=0;j<IN;j++){           /* 入力に対する出力 */
            genotype[i][j] = rand() % 4;   /* [0,3] */
        }
    }
}


void disp_action( int n, int steps )
/* No.n の個体を steps 回だけ動かす.                              */
/* 途中で壁にぶつかったり, １回居た場所に戻ったり, ゴールに */
```

```
/* 到達したら steps 回に到達していなくても表示を終了する.    */
{
    int x, y, i;          /* 作業変数                        */
    int dx, dy;           /* 移動先の座標                    */
    int input, output;    /* 入力・出力を数値化したもの */
    int fit,finish,kbd;

    /* マップの初期化 */
    for(y=0;y<WY;y++){
        for(x=0;x<WX;x++){
            map[y][x] = mapdata[y][x];
        }
    }
    map[START_Y][START_X] = 2;     /* スタート地点の設定     */
    map[GOAL_Y][GOAL_X]   = 3;     /* ゴール地点の設定       */
    x = START_X;                   /* 現在地のＸ座標の初期化 */
    y = START_Y;                   /* 現在地のＹ座標の初期化 */
    /* 個体 No.n を steps 回だけ動かす */
    i=1;        /* i:ステップ数を表す変数             */
    finish=0;   /* 何等かの原因で終了したら１にする */
    do{
        /* 画面に step, 入力信号, 出力信号 を表示する */
        printf("step:%d  ",i);
        input = obtain_input( x, y );  /* 入力パターンの調査 */
        printf("入力(周囲の状況):%d, ",input);
        output = genotype[n][input];  /* 行動出力 */
        printf("出力(次の移動方向):");
        switch( output ){
            case 0: printf("↑¥n");  break;
            case 1: printf("→¥n");  break;
            case 2: printf("↓¥n");  break;
            case 3: printf("←¥n");
        }
        /* マップを表示する */
        disp_map();
        /* 次の場所を決める */
        dx = x + move_x[output];    /* 次の場所（仮）  */
        dy = y + move_y[output];    /* 次の場所（仮）  */
        if ( dx>0 && dx<WX-1 && dy>0 && dy<WY-1 &&
             ( map[dy][dx] != 1 && map[dy][dx] != 4 ) ){
            /* 次の場所が壁(=1)および通過軌跡(=4)以外なら動かす */
            map[y][x] = 4;  /* 今居る場所を通過軌跡(=4)にする */
            x = dx;         /* 現在地のＸ座標を更新する */
            y = dy;         /* 現在地をＹ座標を更新する */
            map[y][x] = 2;  /* 移動先の数値を直す       */
            /* ゴールに到達したら表示する */
            if ( x == GOAL_X && y == GOAL_Y ){
                printf("GOAL!!! ｬｯﾀｰﾍ(*´ｖ｀*)ﾉｰｱｧ!!¥n");
            }
        } else {
            finish=1;  /* 即終了 */
            if ( map[dy][dx] == 1 ){    /* 壁にぶつかった          */
                printf("壁にぶつかりました！ 。･ﾟ･(*ﾉД`*)･ﾟ･。¥n");
            } else {                    /* １度通ったマスに戻った */
                printf("１度通ったマスに戻ってしまいました！ 。･ﾟ･(*ﾉД`*)･ﾟ･。¥n");
            }
        }
        /* キーボード入力待ちにする */
        printf("hit any key (強制終了したいときはスペースキー):");
        kbd = getch();
                if ( kbd == 32 ) exit(1);
        printf("¥n¥n");
        /* ステップを増やす */
```

```
        i++;
    }while( ( x != GOAL_X || y != GOAL_Y ) && i<=steps && finish==0 );
    /* 最終的な適応度（評価値，ゴールとの現在位置との差で計算）*/
    fit = 1 + abs(GOAL_X - START_X) + abs(GOAL_Y - START_Y ) - abs(GOAL_X - x) - abs(GOAL_Y - y);
    printf("この個体の適応度 = %d¥n",fit);
}


int obtain_input( int x, int y )
/* 座標(x,y)に居るときの周囲を 0～255 で表して返す */
/*  0  1  2      エージェント(A)の周囲 8 マスを左のように 0～7bit と */
/*  7  A  3      みなし，壁があれば 1，なければ 0 として [0, 255] で */
/*  6  5  4      数値化している．                                    */
{
    int shift_x[8] = {-1, 0, 1, 1, 1, 0, -1, -1};  /* X のシフト量 */
    int shift_y[8] = {-1, -1, -1, 0, 1, 1, 1, 0};  /* Y のシフト量 */
    int i,input_value=0;    /* 作業変数 */

    /* エージェントの周囲を 8 ビットで数値化している */
    for(i=0;i<8;i++){
        input_value = input_value * 2;
        if ( map[ y+shift_y[i] ][ x+shift_x[i] ] == 1 ){
            input_value++;
        }
    }
    return input_value;
}


void generation( int n )
/* n 世代だけ世代交代させる */
{
    int i,j,max,maxnum;       /* 作業変数           */
    int maximum;              /* 理論上の最大適応度 */
    int kbd;

    /* 理論上の最大適応度（今回の適応度定義式による）   */
    /*  （現在位置がゴール位置に一致したとき最大になる）*/
    maximum = 1 + abs(GOAL_X - START_X) + abs(GOAL_Y - START_Y );
    /* 初期個体群の適応度を求める(引数は動かす最大 step 数) */
    calc_fitness( 100 );
    /* 世代交代 */
    i=0;  /* i:世代数 */
    do{
        /* 次世代の POP 個の個体を求める */
        determine_next_generation( );
        /* 各個体の適応度を求める(引数は動かす最大 step 数) */
        calc_fitness( 100 );
        /* 最大適応度を求める */
        max = - WX * WY;    /* 仮の値 */
        maxnum = 0;         /* 仮の値 */
        for(j=0;j<POP;j++){    /* 全個体をスキャンして値を更新 */
            if ( fitness[j] > max ){
                max = fitness[j];
                maxnum = j;
            }
        }
        if ( i % 100 == 0 ){    /* 100 世代ごとに画面表示 */
            printf("世代 No.%d:最大適応度 = %d¥n",i,max);
        }
        i++;
    }while( max < maximum && i < n );
    if ( max == maximum ){
```

```
            printf("世代 No.%d:最大適応度 = %d¥n",i,max);
            printf("ゴールに到達するルールが得られたので世代交代を終了しました. ¥n");
        }
        printf("得られた解を示します. hit any key:");
        /* キーボード入力待ちにする */
        kbd = getch();
        printf("¥n¥n");
        printf("最優秀個体 No.%d の適応度 = %d¥n",maxnum,max);
        /* 最優秀個体の動作を表示 */
        disp_action( maxnum, 100 );
        /* 得られたルールの画面への表示 */
        printf("¥n 得られた解(ルール(遺伝子型))は次の通りです¥n");
        printf(" (入力 0 から入力%d までのそれぞれの行動出力です) ¥n",IN);
        for(i=0;i<IN;i++){
            printf("%d",genotype[maxnum][i]);
        }
        printf("¥n");
}


void calc_fitness( int steps )
/* 各個体を steps だけ動かしたときの適応度を求める */
/* 構造的には disp_actoin() とほとんど同じである.  */
{
    /* 全個体を動かして適応度 fitness[0～POP-1] を求めましょう.      */
    /* 毎回, 最初にマップを初期化してから動かします. disp_action を */
    /* コピペしてから直すと楽です.                                    */

    /* スタート地点から, それぞれの位置での入力値を元に, 遺伝子型に */
    /* 書かれた行動規則に従って動かします. 次の場所が, 壁(=1)か, 移 */
    /* 動軌跡(=4), あるいはゴール(=3)になったら, その座標(x,y)から, */
    /* 次式を使って求めます.                                         */
    /* fitness = 1 + abs(GOAL_X - START_X) + abs(GOAL_Y - START_Y ) */
    /*           - abs(GOAL_X - x) - abs(GOAL_Y - y);                */
    /* ただし, fitness < 0 となったら fitness = 0 にすること.        */
}


void determine_next_generation( )
/* シンプルＧＡ方式（ルーレット選択）＋エリート保存で次世代の個体を求める    */
/* ここでは普通にルーレット選択してから, 0 番の個体にエリートを戻すこととする */
{
    /* まずはエリート（最優秀）個体の遺伝子型を elitest[IN] に保存しよう.    */

    /* ルーレット選択は, まずは全ての個体の適応度の和を sum として求め,      */
    /* 次に random() % sum として [0,sum-1]の乱数でルーレットの位置を決め,   */
    /* それが何番目の個体に該当するかを調べましょう. これを POP 回おこなって, */
    /* 新しい個体の遺伝子型 new_genotype[][] に保存しましょう.               */

    /* 次に, 個体を２個ずつ組み合わせて１点交叉させます.                     */
    /* この際, 交叉率の確率で交叉が生じることとします.                       */
    /* 交叉点の位置はランダムに設定し, その位置で親１, 親２の遺伝子型を分割  */
    /* して, 子１：親１の前半＋親２の後半, 子２：親２の前半＋親１の後半　に  */
    /* して新しい遺伝子型を作ります.                                         */

    /* 次に, 突然変異では, 全個体の全遺伝子（0～3）を対象として, 突然変異率  */
    /* の生起確率で, 0～3 にランダムに変更しましょう.                         */

    /* 最後に, 最初に取っておいた現世代のエリート個体の遺伝子側を, No.0 の   */
    /* 遺伝子側に強制的にコピーしましょう. つまり No.0 が常に最大適応度の個体 */
    /* となります.                                                           */

    /* 作業は以上で終了です. 上の作業をするために, この関数内で適宜, 変数を  */
```

```
    /* 宣言して利用して下さい.                                    */
}
```

レポート内容について剽窃が発見された場合は，見せた人，見た人の両方が0点になりますので，コピペは絶対に止めて下さい．自力では解らない場合，友達に尋ねても良いですが，他人のを写すのはダメです．自分で考えましょう．

人によってはプログラミングが苦手な人も居るかも知れません．完成できない人は，途中までの未完成の状態でも結構ですので，レポートは提出するようにし，なぜうまくできなかったか？などを考察するようにして下さい．どうぞよろしく．

もし何か分からないことがあれば，メールで長尾（nagao@ynu.ac.jp）までお気軽にご相談下さい．なお，プログラムのデバッグや問題の解き方などに関するご質問にはお答えかねますので，あらかじめご了解下さい．では，ご健闘をお祈りします．

注意：

MinGW で C と C++のコンパイラをインストールした場合，c=getch( );
の部分に Warning が出るかも知れませんが，無視して下さい．