#define *\_CRT\_SECURE\_NO\_WARNINGS*

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<time.h>

/\* Random function \*/

#define NN 624

#define MM 397

#define MATRIX\_A 0x9908b0df /\* constant vector a \*/

#define UPPER\_MASK 0x80000000 /\* most significant w-r bits \*/

#define LOWER\_MASK 0x7fffffff /\* least significant r bits \*/

#define TEMPERING\_MASK\_B 0x9d2c5680

#define TEMPERING\_MASK\_C 0xefc60000

#define NAMEOUT "K4b075r5Q2"

#define TEMPERING\_SHIFT\_U(y) (y >> 11)

#define TEMPERING\_SHIFT\_S(y) (y << 7)

#define TEMPERING\_SHIFT\_T(y) (y << 15)

#define TEMPERING\_SHIFT\_L(y) (y >> 18)

static unsigned long mt[NN]; /\* the array for the state vector \*/

static int mti = NN + 1; /\* mti==NN+1 means mt[NN] is not initialized \*/

void sgenrand(unsigned long seed) {

int i;

for (i = 0; i < NN; i++) {

mt[i] = seed & 0xffff0000;

seed = 69069 \* seed + 1;

mt[i] |= (seed & 0xffff0000) >> 16;

seed = 69069 \* seed + 1;

}

mti = NN;

}

void lsgenrand(unsigned long seed\_array[]) {

int i;

for (i = 0; i < NN; i++)

mt[i] = seed\_array[i];

mti = NN;

}

double genrand() {

unsigned long y;

static unsigned long mag01[2] = { 0x0, MATRIX\_A };

if (mti >= NN) {

int kk;

if (mti == NN + 1)

sgenrand(4357);

for (kk = 0; kk < NN - MM; kk++) {

y = (mt[kk] & UPPER\_MASK) | (mt[kk + 1] & LOWER\_MASK);

mt[kk] = mt[kk + MM] ^ (y >> 1) ^ mag01[y & 0x1];

}

for (; kk < NN - 1; kk++) {

y = (mt[kk] & UPPER\_MASK) | (mt[kk + 1] & LOWER\_MASK);

mt[kk] = mt[kk + (MM - NN)] ^ (y >> 1) ^ mag01[y & 0x1];

}

y = (mt[NN - 1] & UPPER\_MASK) | (mt[0] & LOWER\_MASK);

mt[NN - 1] = mt[MM - 1] ^ (y >> 1) ^ mag01[y & 0x1];

mti = 0;

}

y = mt[mti++];

y ^= TEMPERING\_SHIFT\_U(y);

y ^= TEMPERING\_SHIFT\_S(y) & TEMPERING\_MASK\_B;

y ^= TEMPERING\_SHIFT\_T(y) & TEMPERING\_MASK\_C;

y ^= TEMPERING\_SHIFT\_L(y);

return y;

}

double randf() {

return ((double)genrand() \* 2.3283064370807974e-10);

}

long randi(unsigned long LIM) {

return ((unsigned long)genrand() % LIM);

}

/\* End of random \*/

#define RANDOMIZE 3145215

#define L 200

#define K 0.1

#define NEINUM 4

#define SIZE (L\*L)

#define MC\_STEPS 200000

#define REC\_STEPS 5000

#define REFRESH\_PRE 100 /\* Control the frequency of refresh screen, 1 for all the time \*/

#define TRY\_TIME 10

double b;

int net[SIZE][4]; /\* Control the frequency of refresh screen, 1 for all the time \*/

int cooperator, defector;

char frequency[100];

double B = 1;

double A = 0.5;

double u = 0.15;

double t = 0.2;

double p[SIZE];

struct Strategy {

int s;

int type;

} stra[SIZE + 1];

/\* Payoff matrix and its update \*/

double payoff\_matrix[2][2] = { { 1, 0 }, { 1, 0 } };

/\* Call update\_matrix(b) after loop for b \*/

#define update\_matrix(b) payoff\_matrix[1][0]=b;

//player x周围的四个邻居

void prod\_neighbors() {

int i, j, x;

for (i = 0; i < L; i++) {

for (j = 0; j < L; j++) {

x = i \* L + j;

net[x][0] = i \* L + ((j - 1 + L) % L); /\*left\*/

net[x][1] = ((i - 1 + L) % L) \* L + j; /\*up\*/

net[x][2] = ((i + 1) % L) \* L + j; /\*down\*/

net[x][3] = i \* L + ((j + 1) % L); /\*right\*/

}

}

}

//对每个player进行策略初始化

void init() {

cooperator = defector = 0;

int i;

for (i = 0; i < SIZE; i++) {

stra[i].s = randi(2); //随机生成0,1

if (randf() < u) {

stra[i].type = 0; //学习策略

}

else {

stra[i].type = 1; //费米策略

}

if (stra[i].s == 0)

cooperator++;

else

defector++;

p[i] = 0.5;

}

}

double payoff(int x) {

int i;

double pay = 0;

for (i = 0; i < NEINUM; i++)

pay += payoff\_matrix[stra[x].s][stra[net[x][i]].s]; //与自己的邻居进行对比，算出收益

return pay;

}

double stimu(int x) {

double s, r;

r = payoff(x) / 4;

s = tanh(B \* (r - A));

return s;

}

void calcul() {

int i, j;

double s;

for (i = 0; i < SIZE; i++) {

s = stimu(i);

if ((s >= 0) && stra[i].s == 0) {

p[i] = p[i] + (1 - p[i]) \* s;

}

else if ((s < 0) && stra[i].s == 0) {

p[i] = p[i] + p[i] \* s;

}

else if ((s >= 0) && stra[i].s == 1) {

p[i] = p[i] - p[i] \* s;

}

else {

p[i] = p[i] - (1 - p[i]) \* s;

}

}

}

void update\_stra() {

int i, j, player;

double pay1, pay2;

for (i = 0; i < SIZE; i++)

{

player = (int)randi(SIZE);

if (stra[player].type == 0)

{

if (randf() <= p[player])

{

stra[player].s = 0;

}

else

{

stra[player].s = 1;

}

}

else {

pay1 = payoff(player);

j = net[player][(int)randi(NEINUM)]; //player的一个邻居

pay2 = payoff(j);

if (stra[player].s != stra[j].s) {

if (randf() < 1 / (1 + exp((pay1 - pay2) / K))) {

stra[player].s = stra[j].s;

}

}

}

/\*static long mm = 1;

if(mm % 50 == 0)

{

cooperator= defector= 0;

for(int k= 0;k< 10000;k++)

{

if (stra[k].s == 0)

{

cooperator++;

}

else

{

defector++;

}

}

double m, xxx, yyy;

m = (double)mm / (SIZE);

xxx = (double)cooperator / SIZE;

yyy = (double)defector / SIZE;

//ZZZ = (double)loner / SIZE;

//cout<<m<<'\t'<<xxx<<'\t'<<endl;

printf("%lf\t%lf\n", m, xxx);

}

mm += 1;

}

\*/

}

return;

}

void update\_data() {

int i;

cooperator = defector = 0;

for (i = 0; i < SIZE; i++) {

if (stra[i].s == 0) {

cooperator++;

}

else {

defector++;

}

}

return;

}

int main()

{

sgenrand(RANDOMIZE);

prod\_neighbors();

int step; int yc; double avgpay; int count;

double fc, afc = 0;

FILE\* fp\_aver = fopen("averageb.txt", "w");

for (u = 1; u > 0.5; u -= 0.05)

{

for (b = 1; b <= 2.05; b += 0.05)

{

sprintf(frequency, "%g\_frequency.txt", b);

FILE\* fp\_freq = fopen(frequency, "w");

update\_matrix(b);

avgpay = 0;

count = 0;

init();

for (step = 1; step < MC\_STEPS; step++)

{

calcul();

update\_stra();

update\_data();

fc = (double)cooperator / SIZE;

if (step > MC\_STEPS - REC\_STEPS - 1) //排除干扰

afc += fc;

fprintf(fp\_freq, "%d\t%lf\n", step, fc);

if (step % REFRESH\_PRE == 0)

printf("\rStep:%d\tC:%lf%% ", step, fc \* 100);

if ((!cooperator) || (!defector))

{ //当只剩下C或只剩下D时终止程序

if (step++ < MC\_STEPS - REC\_STEPS) afc = cooperator ? 1 : 0;

break;

}

if (step >= (MC\_STEPS - 100))

{

for (yc = 0; yc < SIZE; yc++)

{

avgpay += payoff(yc);

}

count++;

}

}

avgpay = avgpay / count;

avgpay = avgpay / 40000;

fprintf(fp\_aver, "%lf\t%lf\t%lf\n", b, u, avgpay);

putchar(10);

fclose(fp\_freq);

}

}

fclose(fp\_aver);

return 0;

}

//int main() {

//

// sgenrand(RANDOMIZE);

// prod\_neighbors();

// int x, step;

// double fc, afc = 0;

//

// FILE\* fp\_aver = fopen("average.txt", "w");

//

// double us[] = { 0, 0.2, 0.5, 0.8, 1 };

//

// for (int i = 0; i <= 4; i += 1) {

// u = us[i];

//

// for (b = 1; b <= 2; b += 0.025) {

//

// update\_matrix(b);

//

// // sprintf(frequency, "%g\_%g\_frequency.txt", u, b);

// // FILE\* fp\_freq = fopen(frequency, "w");

// init();

// char banGraphFlieName[100];

// sprintf(banGraphFlieName, "u=%g\_b=%g\_banGraph.txt", u, b);

// FILE\* fp\_banGraph = fopen(banGraphFlieName, "w");

// for (step = 1; step < MC\_STEPS; step++) {

// calcul();

// update\_stra();

// update\_data();

// // fc = (double) cooperator / SIZE;

// /\*

// if (step % REFRESH\_PRE == 0)

// printf("\rStep:%d\tC:%lf%% ", step, fc \* 100);

// \*/

// // if ((!cooperator) || (!defector)) { //当只剩下C或只剩下D时终止模拟

// // if (step++ < MC\_STEPS - REC\_STEPS)

// // afc = cooperator ? 1 : 0;

// // break;

// // }

// //

// // if (step > MC\_STEPS - REC\_STEPS - 1) //排除干扰

// // afc += fc;

// // fprintf(fp\_freq, "%d\t%lf\n", step, fc);

// }

// //

// // if (step > MC\_STEPS - REC\_STEPS && afc != 1 && afc != 0)//如果跑到了后5000步-求一个平均afc，否则afc为0或1

// // afc /= step + REC\_STEPS - MC\_STEPS;

//

// // fprintf(fp\_aver, "%g\t%g\t%g\n", u, b, afc);

// // printf("\rb: %lf\tavg\_C: %lf%% ", b, afc \* 100);

//

// // if (afc == 0 || afc == 1) //当遇到全为0的b值终止程序

// // break;

// // putchar(10);

//

// for (int j = 0; j < L \* L; j++) {

// if ((j % L) == 0 && j != 0)

// fprintf(fp\_banGraph, "\n");

// fprintf(fp\_banGraph, "%d\t", stra[j].s);

// }

// fclose(fp\_banGraph);

//

// printf("u=%g,b=%g,%s\n", u, b, "输出完毕");

// }

// }

//

// fclose(fp\_aver);

//

// return 0;

//}