Майер Р.В.

ИМИТАЦИОННАЯ МОДЕЛЬ ДЕЯТЕЛЬНОСТИ УЧАЩЕГОСЯ ПРИ РЕШЕНИИ ЗАДАЧИ

При решении задачи учащийся ведет себя как вероятностный автомат (ВА), осуществляющий ту или иную последовательность операций из некоторого множества $O=\{O_1, O_2, ..., O_S\}$. Пусть решение всегда начинается с операции O_1 (чтение условия); последовательность $O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow O_4 \rightarrow O_5$ является правильной. Вероятность выбора операции O_2 после O_1 обозначим p_1, выбора O_{i+1} после O_i — p_i. Решение задачи аналогично поиску выхода из лабиринта. Все операции ВА совершает безошибочно; если же он допускает ошибку, значит выполняет какую–то другую операцию. Вероятность решения задачи с первой попытки равна произведению $p_1 p_2 p_3 ... p_5$. Если алгоритм решения известен, то $p_1 = p_2 = ... = 1$, и ВА ведет себя как детерминированный автомат, достигая результата за минимальное число шагов.

Пусть учащийся не знает, как решается задача, но ему известно, что решение требует не более L шагов. При решении задачи учащийся как бы движется по некоторому пути в лабиринте, каждый раз делая выбор: выполнить новый шаг в данном направлении или начать все сначала. Если конец решения далеко, то желание учащегося идти по выбранному пути уменьшается, так как он не знает, правильно ли он выбрал направление движения или ошибся в самом начале. Он выполняет L шагов в некотором направлении (проходит несколько узлов лабиринта) и, если ему не удается получить ответ (выйти из лабиринта), возвращается к началу пути. Затем повторяет все снова и так k раз. С каждой новой попыткой вероятность того, что учащийся бросит решать задачу, увеличивается. Упорство учащегося характеризуется глубиной поиска L и числом попыток k.

Автомат, моделирующий деятельность учащегося, перед каждым шагом должен “решить”, следует выполнять новое действие или лучше вернуться к началу O_1. Логично предположить, что с каждым i–ым шагом вероятность того, что ВА будет продолжать идти по выбранному пути, уменьшается по экспоненциальному закону: $P = \exp(-ai)$. Перед каждой новой попыткой решить задачу, ВА должен сделать выбор: продолжать решение или отказаться от него. Вероятность каждой следующей попытки тоже уменьшается по закону $Q = \exp(-bj)$, где j — номер попытки. Справедливо утверждение: ВА обязательно решит задачу, если: 1) ВА владеет всеми необходимыми операциями O_1, O_2, ..., O_S; 2) в “памяти” ВА имеется путь от первой O_1 до конечной операции O_k с ненулевыми вероятностями переходов p_i; 3) число шагов, предпринимаемых ВА в каждой попытке, больше длины решения; 4) ВА делает бесконечно большое число попыток; 5) ВА умеет отличать

правильный ответ от неправильного. Рассмотрим алгоритм программы, моделирующей деятельность ученика при решении задачи:

```
i:=1; op[1]:=1; k:=1; shag:=0; R:=true; ПЕЧАТЬ(i,'  ',op[i]);
ПОВТОРЯТЬ { —
  shag:=shag+1; i:=i+1; x:=RND(100)/100;
  ЕСЛИ x<p[op[i–1]] ТО op[i]:=op[i–1]+1 ИНАЧЕ {  m: op1:=RND(30)/10+1;
  ЕСЛИ(op1=op[i–1]+1)ИЛИ(op1=op[i–1]) ТО ИДТИ К m
                                                  ИНАЧЕ op[i]:=op1;  }
  ЕСЛИ op[i]–op[i–1]=1 ТО ПЕЧАТЬ(' ВЕРНО') ИНАЧЕ {
                                          ПЕЧАТЬ (' НЕВЕРНО'); R:=false; }
  Q :=exp(–0.08*shag); x:=RND(100)/100;
  ЕСЛИ x > Q ТО [shag:=0; op[i]:=1; R:=true;
                          k:=k+1; ПЕЧАТЬ('попытка ',k);]    —}
ПОКА ((op[i]=6)И(R=true))ИЛИ(i>150);
ЕСЛИ (op[i]=6)И(R=true) ТО ПЕЧАТЬ('ЗАДАЧА РЕШЕНА ');
ПЕЧАТЬ('ЧИСЛО ШАГОВ ',i,');
```

Используется программа ПР–1. Элементы массива op[i] хранят номер операции, выбранной на i–ом шаге с момента $t = 0$ получения задачи. Номер шага по выбранному пути решения задачи (попытка k) хранится в счетчике shag. ВА, моделирующий ученика, может допустить ошибку и продолжать двигаться в неверном направлении. Так продолжается, пока ВА не примет решение начать новую попытку и вернется к решению задачи. Даже двигаясь в правильном направлении, ВА может остановиться и вернуться к началу задачи. При этом число i выполненных шагов с момента $t = 0$ остается равным некоторому целому числу, а переменная shag принимает значение 1. Правильно выполнив все операции, учащийся осознает, что решил задачу.

Результаты моделирования — на рис. 1.1. По вертикали откладывается номер операции, последняя попытка соответствует правильному пути 1 – 2 – 3 – 4 – 5 – 6. Также была промоделирована деятельность ученика в случае, когда учитель находит его ошибки и подсказывает правильное решение (увеличивает p_i).

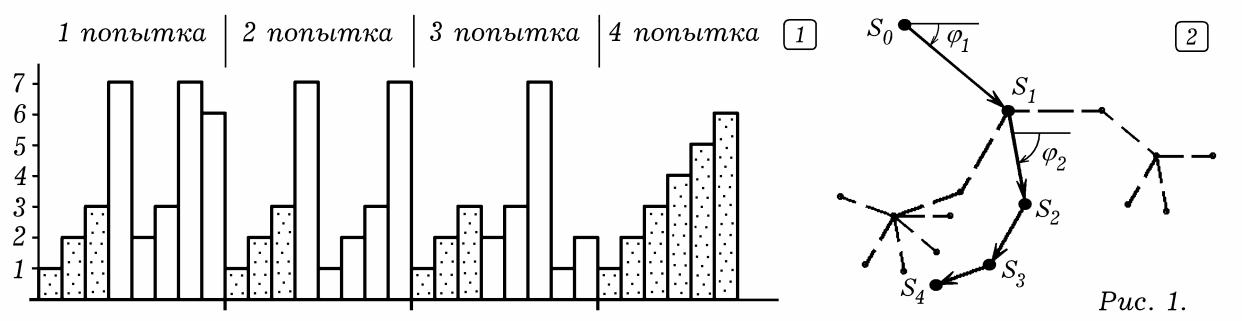

Рис. 1.

Чтобы построить граф решения (рис. 1.2), от начальной точки S_0, соответствующей начальному состоянию, откладывается вектор S_0S_1 длиной

$L_1 = 1$ под углом $\varphi_i = j\Delta\varphi$ к горизонтали, где j – номер операции. Затем от точки S_1 откладывается вектор S_1S_2 длиной $L_2 = 1/2$ под углом $\varphi_i = j\Delta\varphi$ к горизонтали и т.д. В результате перебора всех различных сочетаний и последовательностей операций получается граф, имеющий фрактальную структуру. Правильному решению задачи соответствует некоторый путь из состояния S_0 в состояние S_m, где m — число операций. Понятно, что при заданных параметрах модели (p_i, a и b) число шагов N, совершаемых ВА для решения задачи, — случайная величина. Чтобы определить среднее значение N, использовался метод статистических испытаний. В программу был добавлен цикл, в котором 100–500 раз запускается ВА, решающий задачу. При этом подсчитывалось общее число шагов, число отказов от решения, после чего результаты усреднялись (программа ПР–2).

Программа ПР–1.

```
uses crt,graph;                          { Free Pascal }
const p: array [1..6]of real=(0.6,0.5,0.4,0.7,0.6,0);
var zz,dv,mV,kk,i,k,shag,op1:integer;sluch,ver,S:real;
R:boolean; x,y,op:array[1..1501]of integer; Label m;
BEGIN DV:=Detect; InitGraph(DV,MV,'c:\bp\bgi');
randomize; i:=1; op[1]:=1; k:=1; x[1]:=320; y[1]:=40;
shag:=1;
R:=true; circle(x[1],y[1],3);
Repeat inc(shag); inc(i); sluch:=random(100)/100;
  If sluch<p[op[i-1]] then op[i]:=op[i-1]+1 else
      begin m:
        op1:=round(random(50)/10)+1;
        If (op1=op[i-1]+1)or(op1=op[i-1])
              then goto m else op[i]:=op1; end;
  If op[i]-op[i-1]<>1 then R:=false;
  ver:=exp(-0.1*shag); sluch:=random(100)/100;
  If shag>7 then begin shag:=1; op[i]:=1;
  R:=true; inc(k); x[1]:=320; y[1]:=40;
  end else begin zz:=op[i];
  x[shag]:=x[shag-1]+round(250/shag*cos(2*3.14*zz/15));
  y[shag]:=y[shag-1]+round(250/shag*sin(2*3.14*zz/15));
  line(x[shag-1],y[shag-1],x[shag],y[shag]);
  circle(x[shag],y[shag],2); delay(10);
  rectangle(5*i,450-10,5*(i-1),450-10*op[i]); end;
until ((op[i]=6)and(R=true))or(Keypressed);
inc(kk); shag:=1; x[1]:=320; y[1]:=40;
Repeat until KeyPressed; Closegraph;
END.
```

Программа ПР-2.

```
uses crt;
const p: array [1..5]of real=(0.8,0.85,0.7,0.6,0);
var i,k,k_isp,shag,op1:integer; x,ver,S:real;
R:boolean; op:array[1..9000]of integer; Label m;
BEGIN clrscr; randomize; k_isp:=1; i:=1;
Repeat op[1]:=1; k:=1; shag:=0; R:=true;
Repeat inc(shag); inc(i); x:=random(100)/100;
 If x<p[op[i-1]] then op[i]:=op[i-1]+1 else begin m:
 op1:=round(random(30)/10)+1;
 If (op1=op[i-1]+1)or(op1=op[i-1]) then goto m
                            else op[i]:=op1; end;
 If op[i]-op[i-1]<>1 then R:=false;
{ writeln('ISP ',k_isp,' | ',i,' SHAG ',shag,
        ' OPERACIYA ', op[i],' PUT ',R); delay(500);}
 ver:=exp(-0.1*shag); x:=random(100)/100;
 If x>ver then begin shag:=0; op[i]:=1;
 R:=true; inc(k);
 writeln('VSE SNACHALA. Popitka ',k); delay(500); end;
until ((op[i]=5)and(R=true))or(Keypressed);
If (op[i]=5)and(R=true) then begin
 Writeln('ZADACHA RESHENA ');
 Writeln('VSEGO SHAGOV= ',i,' ISPITANIE ',k_isp,
 'DLINA PUTI',i/k_isp); delay(500); inc(k_isp); end;
until (k_isp>=100)or(Keypressed); Readkey;
END.
```

Литература

1. Харин Ю.С., Малюгин В.И., Кирлица В.П. и др. Основы имитационного и статистического моделирования. Учебное пособие. — Мн.: Дизайн ПРО, 1997. — 288 с.
2. Шеннон Р. Имитационное моделирование систем: искусство и наука. — М.: Мир, 1978. — 302 с.