Yugoslav Journal of Operations Research
10 (2000), Number 1, 99-108

AN ALGORITHM FOR GENERATING RANDOM NUMBERS WITH BINOMIAL DISTRIBUTION

Dorin BOCU

Faculty of Science, Department of Computer Science
"Transilvania" University of Braşov, Romania

Abstract: In this paper we introduce an algorithm for generating random numbers with binomial distribution. The algorithm is based on the idea of a "pipe" between the simulation of a statistical experiment and the algorithm for selection according to utility, often invoked in genetic search problems.

Keywords: Binomial distribution, selection according to utility.

1. PRELIMINARIES

First, we shall introduce several definitions and results that are useful in the main statements of the paper.

Definition 1.1. *We call a system a set of interacting components that operate within a boundary to a certain predefined purpose.*

The boundary filters the types and flow rates of inputs and outputs between the system and its environment. The specification of the boundary defines both the system and the environment of the system.

We denote with S an object, process or phenomenon indicated according to Definition 1.1.

Definition 1.2. *We call the **context of a system** the set of restrictions under which the system can achieve its purpose.*

Eliminating or adding restrictions to the context of a system S affects its knowing, so it requires a new specification effort.

We denote with C_S the context of system S.

Definition 1.3. *We call the observation criterion of a system S by an instance I the set of all different and significant states of system S from the point of view of instance I. We denote with* O_{SI} *the observation criterion of system S by instance I. We remark that* O_{SI} *is a representing model of system S in context* C_S*, according to the perception of instance I.*

Definition 1.4. *We call an* ***experiment*** *the repeated analysis of a system S working in the same context* C_S *and according to the same observation criterion* O_{SI}*.*

We denote with **E** a generic experiment.

Definition 1.5. *We call an elementary random event the result of an unpredictable experiment. If S is a system which is the object of an experiment* **E**, *we call the result of the experiment the state in which system S arrives at the end of experiment* **E**. *The set of distinct events, characteristic of an unpredictable experiment, is called a complete event system.*

We denote with E the complete event system associated to experiment **E**.

Definition 1.6. *We call a* ***random event*** *any proper subset of E.*

We denote random events by *A, B, C, ...*

Definition 1.7. *Let* **E** *be experiment whose complete system event is E. If* $(\exists)p: P(E) \to [0,1]$ *so that p is a measure of the posibility of the occurence of events* $A \subseteq E$ *then we say that p is a* ***probability***.

Definition 1.8. *Let* **E** *be experiment whose complete event system is E. We call the* ***distribution law*** *of the events related to E a mathematical model by means of which we can make probabilistic predictions about the results of experiment* **E**.

Such mathematical models are: probability density, repartition function, characteristic function the random variable. Out of the usual distribution laws we illustrate: uniform distribution, binomial distribution, Poisson distribution, etc.

This paper does not aim to discuss aspects related to the theoretical contents of any of these laws. However, we shall study to obtain an algorithm which allows the simulation of the behaviour of a system with a binomial distribution of events.

Related to the binomial experiment we present the definition below.

Definition 1.9. *We say that a random variable X given by table:*

$$X:\begin{pmatrix} 0 & 1 & 2 & \cdots & n \\ p_0 & p_1 & p_2 & \cdots & p_n \end{pmatrix}$$

has a binomial distribution with parameters n and p if:

$$p_i = P(X = i) = C_n^i p^i (1-p)^{n-i}; i = \overline{0, n}$$

Remark 1.1. *The most simple way to introduce a binomial distribution is by studying the following experiment:*

"Let **E** *be an experiment related to which event A occurs with probability p and event* $\overline{A}$ *occurs with probability* $1-p$. *Given the experiment* $\mathbf{E}^1$ *which is obtained by repeating experiment* **E** *n-times we raise the question: what is the probability that A occurs i-times?". The theoretical probability that the above-mentioned event occurs is specified in Definition 1.9. In this paper we are looking for an algorithm which generates integer numbers belonging to* $\{0,1,\ldots,n\}$ *according to the law of binomial distribution of n and p given parameters.*

Proposition 1.1. *A random variable Y which takes values uniformly distributed in* $[0,1]$ *has the repartition function:*

$$F_Y(x) = \begin{cases} 0; & \text{if } x \in (-\infty, 0] \\ x; & \text{if } x \in [1, \infty) \end{cases}$$

Proof: Because *Y* has the uniform distribution of values in $[0,1]$ then the probability density is:

$$f_Y(x) = \begin{cases} 1; & \text{if } x \in [0,1] \\ 0; & \text{if } x \in \mathbf{R} \setminus [0,1] \end{cases}$$

So:

$$F_Y(x) = \int_0^x f_Y(x)dx = \begin{cases} 0; & \text{if } x \in (-\infty, 0] \\ x; & \text{if } x \in (0,1] \\ 1; & \text{if } x \in (1, \infty) \end{cases}$$

Proposition 1.2. *Let* $k \in (0,1)$ *and experiment* **E** *be defined in this way:*

1) A random number $NRA \in (0,1)$ *is generated according to the uniform distribution law;*

2) If $NRA \le k$ *then we consider that event A is realized;*

3) If $NRA > k$ *then we consider that event A is not realized;*

What is the probability of event A?

Proof:

$$P(A) = P(NRA \le k) = P(0 \le NRA \le k) = P(NRA = 0) + P(0 < NRA \le k).$$

Because *NRA* is uniformly distributed in $[0,1]$, according to Definition 1.1 we have $P(A) = F_{NRA}(k) - F_{NRA}(0) = k$.

The statement of Proposition 1.2 represents the basis of a procedure which models an abstract experiment **E** related to which an event A occurs with probability k. So, the experiment **E** which represents the basis of introducing binomial distribution as a consequence of **E** repeated n-times.

Proposition 1.3. *Let RND be a function which returns randomly and uniformly distributed numbers in* $[0.1]$ *and let S be a system having the space of states* $\Omega = \{0.1, \dots, n\}$. *Let* $F = \{f_0, f_1, \dots, f_n\}$ *be the utility vector related to the states of system S so that* $f_i \in [0.1]\ \forall i \in \{0.1, \dots, n\}$ *and*

$$\sum_{i=0}^{n} f_i = 1 .$$

Then the algorithm:

$$NRA = RND$$

$$IndSel = \min_{0 \le i \le n} \{f_0 + f_1 + \dots + f_{i-1} < NRA \le f_0 + f_1 + \dots + f_i\}$$

is a selection algorithm according to the utility of the states of system S.

Proof: Because RND returns randomly, uniformly distributed numbers in $[0.1]$, according to Proposition 1.1 we have:

$$F_{RND}(x) = \begin{cases} 0; & \text{if } x \in (-\infty, 0] \\ x; & \text{if } x \in (0.1] \\ 1; & \text{if } x \in (1, \infty) \end{cases}$$

So, we have:

$$P(IndSel = i) = P(f_0 + f_1 + \dots + f_{i-1} < NRA \le f_0 + f_1 + \dots + f_i)$$
$$= F_{RND}(f_0 + f_1 + \dots + f_i) - F_{RND}(f_0 + f_1 + \dots + f_{i-1}) = f_i .$$

The demonstration is concluded.

2. ALGORITHM GBDRN

GBDRN is the abbreviation *for Generating Binomially Distributed Random Numbers*.

Theorem 2.1. *The algorithm GBDRN specified by:*

PrelParDB(n, p)
ZeroCom(VUABS)

```
PrelAmplExp(nre)
for i := 1 to nre do
begin
    ConApA:= 0
    for j := 1 to n do
    begin
        NrA := RND
        if NrA ≤ p then ConApA := ConApA +1
    end
    VUABS[ConApA] := VUABS[ConApA] +1
end
VUREL := Convert(VUABS)
NAL := RNDBD
```

It generates randomly, binomially distributed numbers in $\{0,1,\ldots,n\}$.

In the above algorithm:

-PrelParDB(n, p) is a procedure which takes over the parameters of binomial distribution which represent the basis of generating random numbers.

-ZeroCom(VUABS) is a procedure which sets at zero the components of absolute utility vector VUABS.

-PrelAmplExp is a procedure which takes over the size of the generating experiment.

-ConApA is a variable which counts the number of occurences of the hypothetical event A having probability p.

-Convert is a function which transforms absolute utility vector VUABS to relative utility vector VUREL.

-RNDBD is the function which returns a number in $\{0,1,\ldots,n\}$ *that is randomly, binomially distributed by parameters n and p. The selection of the returned number is made according to the utilities associated to them in VUREL.*

Proof: Let $n \in N^*$ and $p \in (0,1)$ be the number taken over by procedure *PrelParDB*. Let *nre* be a number high enough taken over by procedure *PrelAmplExp*.

According to Proposition 1.2 the sequence

```
NrA := RND
if NrA ≤ p then ConApA := ConApA +1
```

simulates experiment **E**.

The sequence

for j := 1 to n do
begin
NrA := RND
if NrA ≤ p then ConApA := ConApA + 1
end

simulates experiment E^1 indicated in Definition 1.9.

After repeating experiment E^1 many times, absolute utility vector *VUABS* will contain the frequency with which *A* occures *i*-times in experiment E^1 for $i \in \{0, 1, \ldots, n\}$.

Applying Proposition 1.3 the result is that the random numbers are generated according to a binomial distribution.

3. BORLAND PASCAL IMPLEMENTATION OF GBDRN

```
unit RNDBD;
    {$IFNDEF CPU87}
    {$N+}
    {$ENDIF}
    {------------------------------------------------------------------------------------------}
    {The unit contains the capabilities necessary to                                            }
    {generate random numbers with a binomial distribution                                       }
    {by parameters <n> and <p> given.                                                           }
    {------------------------------------------------------------------------------------------}
    interface
    uses crt,dos,objects;
    type
    {------------------------------------------------------------------------------------------}
    {Object type which incapsulates the capabilities of                                         }
    {generating random numbers with a binomial                                                  }
    {distribution.                                                                              }
    {------------------------------------------------------------------------------------------}
    PTO_RNDBD = ^TO_RNDBD;
    TO_RNDBD = object
              N     :longint;
              P     :double;
              NRE:  :longint;
              Total :double;
```

```
            PColUAbs : PCollection;
            constructor Init (FN:longint ;FP:double ;FNRE:longint);
            destructor   Done;virtual;
            procedure   GenExpBinom;
            function    RNDBD:longint;
         end;
implementation
type
{------------------------------------------------------------------------------------}
{Object type which specifies a generic element                                      }
{of the collection in which absolute utility is                                     }
{kept.                                                                              }
{------------------------------------------------------------------------------------}
     PTElCol = ^TElCol;
     TElCol = object (Tobject)
            UAbs:longint;
            constructor init(U:longint);
            destructor   done;virtual;
         end;
{------------------------------------------------------------------------------------}
{Constructor TElCol....                                                             }
constructor TElCol.Init(U:longint);
begin
     UAbs:=U;
end;
{------------------------------------------------------------------------------------}
{Destructor TElCol....                                                              }
destructor TElCol.Done;
begin
end;
{------------------------------------------------------------------------------------}
{Constructor TO_RNDBD....                                                           }
constructor TO_RNDBD. Init (FN: longint ; FP: double; FNRE: longint);
begin
     if (FP>=0) and (FP<=1) then
begin
     N:=FN;
     P:=FP;
     NRE:=FNRE;
     PColUAbs:=new (PCollection, init{FN+1, 10));
```

```
        GenExpBinom;
    end
    else
    begin
        gotoxy(1, 24); clreol;
        textcolor(red); textbackground(white+blink);
        write('The parameter <p> is not in [0,1]!!!');
        readkey; halt;
    end;
end;
{---------------------------------------------------------------------------------}
{Destructor TO_RNDBD....                                                          }
destructor TO_RNDBD. Done;
begin
    dispose (PColUAbs, Done);
end;
{---------------------------------------------------------------------------------}
{Procedure simulates a binomial experiment by                                     }
{parameters <n> and <p> given.                                                    }
procedure TO_RNDBD.GenExpBinom;
var
    I, J        :longint;
    ConApA  :longint;
    NRA: double;
    MAN: longint;
    PElCol : PTElCol;
begin
    for I:=0 to N do
    begin
        PColUAbs^ .Insert (new(PTElCol,init (0)));
    end;
    randomize;
    for I:=1 to NRE do
    begin
        ConApA:=0;
        for J:=1 to N do
        begin
            NRA: =random;
            if NRA<=P then inc(ConApA);
        end;
```

```
            PElCol:=PColUAbs^.AT(ConApA);
            MAN:=PElCol^.UAbs;
            inc(MAN);
            PElCol^.UAbs:=MAN;
            PColUAbs^.AtPut(ConApA,PElCol);
        end;
        Total:=0;
        for I:=0 to N do
        begin
            PElCol:=PColUAbs^.At(I);
            Total:=Total+PElCol^.UAbs;
        end;
    end;
    {------------------------------------------------------------------------}
    {Function returns the numbers which can take values                      }
    {from <0> to <n>, binomially distributed.                                }
    function TO_RNDBD.RNDBD: longint;
    var
        I: integer;
        Termen: double;
        GenAl, Suma: double;
        PElCol : PTElCol;
    begin
        GenAl:=random;
        I:=-1
        Suma:=0;
        repeat
            inc(I);
            PElCol:=PColUAbs^.At (I);
            Termen:=PElCol^.UAbs;
            Suma:=Suma+(Termen/Total);
        until Suma>=GenAl;
        RNDBD:=I;
    end;
    end.
```

4. CONCLUSIONS

The prsesent algorithm can be used from any Pascal program which imports unit RNDBD.

The relation between the theoretical probability of binomial repartition by n, p parameters and the frequency with which random numbers are generated by the capabilities of the RNDBD unit will be analyzed in a future paper.

REFERENCES

[1] Bocu, D., "On the short-term behaviour of genetic algorithms", *Yugoslav Journal of Operations Research*, 7 (1997) 55-63.

[2] Bocu, D., "Modelarea matematica a sistemelor cu comportament stochastic", Ph.D. dissertation, Universitatea "Transilvania" Brasov, 1996 (In Romanian).

[3] Onicescu, O., *Probabilitati şi procese aleatoare*, Editura Ştiintifica şi Enciclopedica, 1977.

[4] Sacuiu, I., and Zorilescu, D., *Numere aleatoare-Aplicatii in economie, industrie şi studiul fenomenelor naturale*, Editura Academiei,1978.