УДК: 512.5, 512.6, 004.43

Математические аспекты цифровых технологий: задача о рассеянном пассажире

A.Yu. Shcherbakov

Mathematical Aspects of Digital Technologies: the Problem of The Absent-Minded Passenger

Abstract. *The example of the absent-minded passenger problem illustrates the fundamental difficulty of estimating practical values of probabilistic parameters on the basis of general theorizing and allows us to give more accurate and subtle estimates of various parameters, as well as their practical computable approximation. This article is important for building systems of artificial intelligence and artificial consciousness, as well as for modern didactics of higher education, especially for the disciplines of "applied mathematics" and "theoretical computer science". It is shown that a detailed study of a random process provides more significant applied materials than formally correct theoretical calculations.*

Keywords: *substitution, queue, probability, logistics, artificial consciousness, didactics, teaching mathematics, digital technologies, Monte Carlo methods, cryptography.*

А.Ю. Щербаков
Доктор технических наук, профессор, заведующий кафедрой когнитивно-аналитических и нейро-прикладных технологий Российского государственного социального университета, ведущий научный сотрудник Государственного университета управления.
E-mail: x509@ras.ru

Аннотация. *Пример задачи о рассеянном пассажире иллюстрирует принципиальную сложность оценок практических значений вероятностных параметров на базе общего теоретизирования и позволяет дать более точные и тонкие оценки различных параметров, а также их практическую вычислимую аппроксимацию. Представленное исследование важно для построения систем искусственного интеллекта и искусственного сознания, а также для современной дидактики высшего образования, особенно для дисциплин "прикладная математика" и "теоретическая информатика". Показано, что детальное изучение случайного процесса дает более значимые прикладные материалы, чем формально корректные теоретические выкладки.*

Ключевые слова: *: подстановка, очередь, вероятность, логистика, искусственное сознание, дидактика, преподавание математики, цифровые технологии, методы монте-карло, криптография.*

НЕФОРМАЛЬНОЕ ВВЕДЕНИЕ

Во второй половине XX-го века появилась и стала хорошо известной, хотя и в достаточно узких кругах любителей математических задач, задача про "сумасшедшую старушку".

Ее формулировка следующая. В аэропорту идет посадка в самолет. В очередь выстроились ровно 100 пассажиров. Первой стоит сумасшедшая старушка. Зайдя в салон, она садится на любое случайно выбранное место, которое ей понравилось (возможно, что и на свое). Каждый следующий пассажир, зайдя в салон, садится на свое (обозначенное в билете) место, если оно свободно, и на любое из свободных — в противном случае. Для ложной простоты можно полагать, что пассажир садится на первое из свободных мест. Какова вероятность, что последний в очереди пассажир сядет на свое место (указанное в его билете)?

Более сложный вопрос – сколько в среднем пассажиров оказываются не на своих местах?

Эта задача прошла достаточно длинный путь – сам автор статьи слышал ее формулировку от своего учителя математики **Бориса Соломоновича Гершмана** в 1983-1984гг. Тогда, конечно, никто не говорил об очереди в самолет – шла посадка на большой советский N-местный автобус.

После этого задача в неизменной формулировке прожила достаточно долгую жизнь от рождения в середине 80-х годов прошлого века до выхода в широкие круги в середине 90-х, дальнейшего "фольклорного" статуса в 2000-е. Она остается очень известной и о ней написано несколько интересных препринтов и статей. Надо отметить, что в процессе передачи от математика к математику ее формулировка несколько смягчилась, и вместо "сумасшедшей старушки" в ней стал фигурировать "рассеянный пассажир" (absent-minded passenger).

Борис Соломонович Гершман (р. 1934). Учитель средней школы №144 Северного административного округа. Почетное звание "Заслуженный учитель Российской Федерации", за заслуги в обучении и воспитании учащихся и многолетний добросовестный труд от 5 октября 1995г. Знаменитый организатор и учитель математических классов.

Задача о рассеянном пассажире – хороший способ подумать в целом над глобальным вопросом взаимодействия информационных технологий с математикой, а математики – с практикой, ее решение может пригодится для транспортной и складской логистики, помочь преподавателю математики расширить кругозор своих учеников новыми математическими понятиями, перекинуть мостики не только к построению основ методологии мышления искусственного интеллекта, но и даже к основам криптографии.

Кроме того, эта задача высвечивает нам целую когорту интересных людей, энтузиастов-математиков, которые предлагали ее коллегам и ученикам. Сегодня, в период падения интереса к науке в целом и разрушения последовательно развивавшихся научных школ, самое время вспомнить о своих учителях и безвременно ушедших коллегах.

Дмитрий Дмитриевич Рютов (род. 6 марта 1940, Москва) — советский, российский и американский физик, специалист в области физики плазмы и теоретической физики. Академик Российской академии наук (с 1992 года). Старший научный сотрудник (Senior Visiting Scientist) Ливерморской национальной лаборатории.

Итак, в начале 2000-х задача по-прежнему передавалась преимущественно лично от одного математика к другому, а в 2004-м ее опубликовал по-английски известный коллекционер интересных задач Питер Винклер. Но недавно Константин Кноп обнаружил, что в 1997-м году **Дмитрий Фон-Дер-Флаасс** (или Дмитрий Флаасс), член жюри Всероссийской Олимпиады по математике, рассказал ее другим членам жюри.

В комментариях и рассказах К. Кнопа выяснилось, что сам Фондер-Флаасс (который учился в Новосибирском государственном университете (НГУ) и почти всю жизнь прожил в Новосибирском академгородке) эту задачу узнал, по-видимому, от Андрея Щетникова примерно в 1985-м году, а Щетников - от Игоря Котельникова (все трое в то время были аспирантами НГУ), а Котельников услышал о ней от своего научного руководителя, знаменитого физика **Дмитрия Дмитриевича Рютова**. По его словам, в первоначальной формулировке Д. Рютова вопрос стоял, как в начале нашего введения - "оценить число пассажиров, не попавших на свои места", а формулировка оценки вероятности для последнего пассажира родилась в процессе обсуждения задачи аспирантами.

Дмитрий Германович Фон-дер-Флаасс (8 сентября 1962 — 10 июня 2010) — российский математик и педагог, кандидат физико-математических наук, старший научный сотрудник Института математики Сибирского отделения (СО) РАН, специалист по комбинаторике, популяризатор математики, автор олимпиадных математических задач, член жюри многочисленных математических олимпиад. Обладатель ***числа Эрдёша***[1], равного 1.

В 1975 году, в возрасте 13 лет (т. е. на два года раньше обычного срока), Фон-дер-Флаасс был зачислен в физико-математическую школу при НГУ. Активно участвовал в олимпиадах школьников по математике, будучи постоянным призёром Всесоюзных олимпиад, участвовал в составе сборной школьников СССР на XIX Международной математической олимпиаде в г. Белграде, где получил бронзовую медаль, будучи на 3—4 года младше своих соперников.

После окончания школы Фон-дер-Флаасс остался в Новосибирске, где учился, жил и работал почти всю жизнь. В возрасте 15 лет он поступил на механико-математический факультет НГУ.

Специализировался на кафедре алгебры и математической логики, где под научным руководством профессора В. Д.Мазурова занимался исследованием конечных групп. По этой же тематике он защитил диплом, поступил в аспирантуру НГУ и в 1986 году (в возрасте 23 лет) защитил кандидатскую диссертацию о максимальных подгруппах конечных простых групп. Результаты диссертации вызвали большой интерес у специалистов и явились существенным вкладом в работу по классификации конечных простых групп в то время.

Фон-дер-Флаасс несколько лет преподавал в США и Великобритании, но затем вернулся в Россию, заявив, что единственное место, где он может чувствовать себя комфортно, — Новосибирский Академгородок.

[1] Число Эрдёша (англ. Erdős number) — метод определения кратчайшего пути соавторства по совместным научным публикациям от какого-либо учёного до венгерского математика Пала Эрдёша (1913—1996).

Эрдёш написал за свою жизнь как минимум 1525 статей, что не имеет аналогов среди современных ему математиков и сопоставимо только с числом работ Леонарда Эйлера (более 850). Поскольку большая часть из этих работ была создана в соавторстве, а в математике совместная статья традиционно является скорее исключением, чем правилом, наличие такого большого числа соавторов породило в среде математиков понятие «число Эрдёша».

Это число определяется следующим образом:

- у самого Эрдёша оно равно нулю;
- у непосредственных соавторов Эрдёша это число равно единице (всего 511 человек, включая Д. Фон-дер-Флаасса);
- соавторы людей с числом Эрдёша, равным n (и не имеющие собственного числа Эрдёша меньше или равного n), имеют число Эрдёша n + 1;
- люди, для которых невозможно построить цепочку соавторов к Палу Эрдёшу, имеют число Эрдёша, равное бесконечности.

Впервые это понятие опубликовал Каспер Гоффман — в 1969 году вышла его статья «И какое ваше число Эрдёша?», в которой он описал свои наблюдения сотрудничества Эрдёша с другими учёными.

ФОРМАЛЬНАЯ ПОСТАНОВКА И РЕШЕНИЕ ЗАДАЧИ О РАССЕЯННОМ ПАССАЖИРЕ

Постановка задачи

В [1] сформулированы условие и решение задачи. В автобусе n мест, и все билеты проданы n пассажирам. Первым в автобус заходит Рассеянный Учёный (далее — рассеянный пассажир) и, не посмотрев на билет, занимает первое попавшееся место. Затем пассажиры входят по одному. Если вошедший видит, что его место свободно, он занимает свое место. Если же место занято, то вошедший занимает первое попавшееся свободное место. Найдите вероятность того, что пассажир, вошедший последним, займет место согласно своему билету?

Решение

Пронумеруем всех пассажиров, начиная с рассеянного пассажира, в том порядке, в каком они заходили в автобус. Последний пассажир имеет номер n. Для простоты места пронумеруем так же. Пусть все, кроме последнего пассажира, уже вошли и заняли места. Осталось одно свободное место. Если бы это было второе место, то второй пассажир (или рассеянный пассажир) уже занял бы его. То же верно для мест номерами 3, 4, 5, ..., n – 1. Значит, это место принадлежит либо последнему пассажиру, либо Рассеянному пассажиру.

Ясно, что оно с равными шансами может принадлежать как первому, так и последнему. В первом случае последний пассажир сядет не на своё место, а во втором – на своё. Значит, вероятности обоих событий равны ½.

Основные понятия и процедуры для решения задачи о рассеянном пассажире

Рассмотрим понятие подстановки [2] степени или порядка n — это отображение вектора (0, 1, ..., n-1) на вектор (p_0, p_1, ..., p_{n-1}), где p_i принадлежит множеству целых чисел от 0 до n-1 и p_i не повторяются. Легко заметить, что это отображение биективно (однозначно).

Именно такой конструкцией (подстановкой) целесообразно представить очередь пассажиров. Единственное уточнение – отказаться от 0 и нулевого элемента и рассматривать вектора с элементами от 1 до n. Число n называется степенью подстановки.

Попробуем сформировать код для генерации случайных подстановок.

Фрагмент 1. Процедура генерации случайной подстановки GenPermit

```
#define N_PRM 250
// Ошибки
//-1- превышен размер подстановки
//-2- превышено число попыток
//-3- ошибка при генерации (не совпадает сумма элементов)
int GenPermit(int n, unsigned char *prm)
  {
   int i,j,k,sum1,sum2,tg,c;
   unsigned char p;
   if(n>=N_PRM) return(-1);
   sum1=0;
   for(i=0;i<n;i++) sum1=sum1+i;
   NextRandom16(rnd,rnd_1,rnd);
   prm[0]=rnd[0]%n;
   c=0;
   for(i=1;i<n;i++)
     {
      m:;
      c++;
      if(c>N_PRM*20) return(-2);
      NextRandom16(rnd,rnd_1,rnd);
      p=rnd[0]%n;
      tg=0;
      for(j=0;j<i;j++) if(p==prm[j]) tg=1;
// Выработанного случайного числа среди предыдущих элементов нет
      if(tg==0) prm[i]=p;
      else goto m;
     }
   sum2=0;
   for(i=0;i<n;i++) sum2=sum2+prm[i];
// Проверка контрольной суммы
   if(sum1!=sum2) return(-3);
   for(i=0;i<n;i++) prm[i]=prm[i]+1;

   return(0);
  }
```

Процедура генерации подстановки GenPermit содержит входной параметр n – степень генерируемой подстановки и выходной – массив чисел от 0 до 255, ограниченный по длине значением N_PRM, которое установлено равным 250.

В процедуре используются беззнаковые целые числа от 0 до 255 (длиной один байт). При анализе задачи для очередей, превышающих 256, необходимо использовать массивы целых чисел.

Данная процедура использует процедуру генерации случайного вектора длиной 16 байт NextRandom16(rnd,rnd_1,rnd), о ней мы поговорим ниже.

Для наших целей будем выбирать нулевой элемент случайного массива rnd и для получения элементов от 0 до n будем приводить каждый байт по модулю n и использовать для этого деление с остатком.

Преимущественно в учебных целях генерируем отдельно первый элемент подстановки указанным образом (получаем нулевой случайный байт из генератора случайных чисел и приводим его по модулю n).

Далее нам необходимо получить еще n-1 элемент подстановки таким образом, чтобы элементы не повторялись, чему посвящены два цикла в основном теле процедуры. Легко видеть, что в процедуре используется нежелательная конструкция goto, но она обезопасена счетчиком циклов получения следующего элемента подстановки.

Обратим внимание на момент, принципиально важный для проведения такого рода экспериментов – это обеспечение корректности и надежности вычислений.

В первую очередь – это контроль размера подстановки: конструкция if(n>=N_PRM) return(-1);

Далее, защита от зацикливания процедуры

if(c>N_PRM*10) return(-2);

И проверка на корректную генерацию подстановки sum2=0;

for(i=0;i<n;i++) sum2=sum2+prm[i];

if(sum1!=sum2) return(-3);

Значение sum1 вычислено заранее:

sum1=0;

for(i=0;i<n;i++) sum1=sum1+i;

Однако можно использовать и формулу суммы арифметической прогрессии вместо выполнения дополнительного цикла.

Пусть после выполнения описанной процедуры получена подстановка степени 20 (очередь из 20 пассажиров):

001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020
008 016 014 003 009 019 013 007 015 005 001 011 012 006 010 017 018 020 002 004

Первый пассажир должен разместиться на 8-м месте, второй – на 16-м, двадцатый – на 4-м.

Теперь рассмотрим процедуру, имитирующую посадку с первым рассеянным пассажиром.

Фрагмент 2. Процедура имитации посадки

```
// Процедура имитация посадки
int boarding(int n, unsigned char *qu,int echo)
    {
  unsigned char mm[N_PRM], mm1[N_PRM], pl[N_PRM],p;
     int i,j,k,c,tg;
     for(i=0;i<n;i++) {mm[i]=0;mm1[i]=1;}
//   список мест- это обратная подстановка
     GenPermit_1(n,qu,pl);
     NextRandom16(rnd,rnd_1,rnd);
     p=rnd[0]%n;
//   Первый рассеянный пассажир
     mm[0]=p+1;
     mm1[p]=0;
     if(echo==1)
      {
      printf("Pass %03d Tic %03d Place %03d\n",1,qu[0],p+1);
       getch();
      }
    if(echo==2) AppLogD1(LOGNAME,n,mm);
// Посадка остальных пассажиров, начиная со второго
     for(i=1;i<n;i++)
       {
        tg=0;
        for(j=0;j<i;j++)
          {
           if(mm[j]!=qu[i]) continue;
            else {tg=1;break;}
           }
//       Если место свободно
         if(tg==0)
          {
           mm[i]=qu[i];
           mm1[mm[i]-1]=0;
           if(echo==1){
            printf("Pass %03d Tic %03d Place %03d - Normal\n",i+1,qu[i],mm[i]);
           for(k=0;k<n;k++) printf("%03d ",mm[k]);
           printf("\n");
           getch();}
          }
//       Если место занято
        else
          {
         for(j=0;j<n;j++) if(mm1[j]==1){mm[i]=j+1;
 mm1[j]=0;break;}
           if(echo==1){
            printf("Pass %03d Tic %03d Place %03d - Wrong\n",i+1,qu[i],mm[i]);
           for(k=0;k<n;k++) printf("%03d ",mm[k]);
           printf("\n");
           getch();}
          }
        if(echo==2) AppLogD1(LOGNAME,n,mm);
       }
// Контроль рассадки
     c=0;
     for(i=0;i<n;i++) c=c+(mm[i]-1);
     tg=0;
     for(i=0;i<n;i++) tg=tg+i;
```

```
// Сколько не на своих местах
    c=0;tg=0;
    for(i=0;i<n;i++) if(qu[i]!=mm[i]) c++;

    if(qu[n-1]!=mm[n-1]) tg=1;
    if(tg==1) return(-c);
    else      return(c);
   }
```

Итак, первый рассеянный пассажир вместо своего 8-го места занял 19-е.

На 19-м месте должен разместиться 6-й пассажир в очереди.

```
 1) 019 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
 2) 019 016 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
 3) 019 016 014 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
 4) 019 016 014 003 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
 5) 019 016 014 003 009 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
```

При посадке 6-го пассажира он занимает свободное 1-е место взамен занятого своего и теперь пассажир, который должен был бы сидеть на 1-м месте (11-й в очереди) тоже пересядет.

```
 6) 019 016 014 003 009 001 000 000 000 000 000 000 000 000 000 000 000 000 000 000
 7) 019 016 014 003 009 001 013 000 000 000 000 000 000 000 000 000 000 000 000 000
 8) 019 016 014 003 009 001 013 007 000 000 000 000 000 000 000 000 000 000 000 000
 9) 019 016 014 003 009 001 013 007 015 000 000 000 000 000 000 000 000 000 000 000
10) 019 016 014 003 009 001 013 007 015 005 000 000 000 000 000 000 000 000 000 000
```

11-й пассажир занимает второе свободное место, а долженствующий сидеть на 2-м 19-й пассажир будет вынужден пересесть.

```
11) 019 016 014 003 009 001 013 007 015 005 002 000 000 000 000 000 000 000 000 000
12) 019 016 014 003 009 001 013 007 015 005 002 011 000 000 000 000 000 000 000 000
13) 019 016 014 003 009 001 013 007 015 005 002 011 012 000 000 000 000 000 000 000
14) 019 016 014 003 009 001 013 007 015 005 002 011 012 006 000 000 000 000 000 000
15) 019 016 014 003 009 001 013 007 015 005 002 011 012 006 010 000 000 000 000 000
16) 019 016 014 003 009 001 013 007 015 005 002 011 012 006 010 017 000 000 000 000
17) 019 016 014 003 009 001 013 007 015 005 002 011 012 006 010 017 018 000 000 000
18) 019 016 014 003 009 001 013 007 015 005 002 011 012 006 010 017 018 020 000 000
```

19-й садится на 4-е место, а двадцатый пересаживается на 8-е.

```
19) 019 016 014 003 009 001 013 007 015 005 002 011 012 006 010 017 018 020 004 000
20) 019 016 014 003 009 001 013 007 015 005 002 011 012 006 010 017 018 020 004 008
```

Последний пассажир оказался не на своем месте.

Теперь мы можем оценить методом Монте-Карло число пассажиров не на своих местах и статистическую вероятность непопадания последнего пассажира на свое место.

Напомним, что методы Монте-Карло (ММК) [3] — группа численных методов для изучения различного рода случайных процессов. Суть методов заключается в следующем: процесс описывается математической моделью (в нашем случае генерация случайных подстановок степени n, которая задает длину очереди и количество мест) с использованием генератора случайных величин, модель многократно реализуется на заданных наборах данных (иногда используют термин "модель обсчитывается"), на основе полученных данных вычисляются вероятностные характеристики рассматриваемого процесса.

Фрагмент 3. Оценка числа пассажиров, не сидящих на своих местах

```
for(i=1;i<11;i++)
        {
         pr_n=i*10;
         sum1=0;sum2=0;
         for(j=0;j<100;j++)
           {
            GenPermit(pr_n,prm1);
            rez=boarding(pr_n,prm1,0);
            sum1=sum1+abs(rez);
           if(rez<0) sum2++;
           }
          printf("\nN= %03d Med1 =%f      Med2=%f\
n",pr_n,(float)(sum1/100.),(float)(sum2/100.));
          AppLogD2(LOGNAME,pr_n,sum1,sum2);
         }
```

В данном фрагменте последовательно сто раз вызывается процедура рассадки пассажиров из фрагмента 2 и меняется степень подстановки от 10 до 100 с шагом десять.

Получаем следующие результаты.

Первый эксперимент

N= 010 Med[pass] =2.680000 P[last] = 0.510000
N= 020 Med[pass] =3.710000 P[last] = 0.530000
N= 030 Med[pass] =4.150000 P[last] = 0.470000
N= 040 Med[pass] =4.340000 P[last] = 0.410000
N= 050 Med[pass] =4.520000 P[last] = 0.500000
N= 060 Med[pass] =4.880000 P[last] = 0.400000
N= 070 Med[pass] =4.460000 P[last] = 0.280000
N= 080 Med[pass] =5.050000 P[last] = 0.360000
N= 090 Med[pass] =5.050000 P[last] = 0.320000
N= 100 Med[pass] =5.370000 P[last] = 0.300000

Второй эксперимент

N= 010 Med[pass] =2.910000 P[last] = 0.470000
N= 020 Med[pass] =3.630000 P[last] = 0.530000
N= 030 Med[pass] =3.630000 P[last] = 0.460000
N= 040 Med[pass] =4.300000 P[last] = 0.410000
N= 050 Med[pass] =4.070000 P[last] = 0.440000
N= 060 Med[pass] =4.620000 P[last] = 0.420000
N= 070 Med[pass] =5.020000 P[last] = 0.300000
N= 080 Med[pass] =4.970000 P[last] = 0.410000
N= 090 Med[pass] =5.160000 P[last] = 0.340000
N= 100 Med[pass] =5.070000 P[last] = 0.340000

Можно видеть, что результаты вполне устойчивые – для десяти пассажиров вероятность того, что последний не окажется на своем месте, составляет примерно 0.5 и понижается с ростом числа пассажиров приблизительно до 0.3.

Число пассажиров, не сидящих на своих местах увеличивается от 2-3 (число пассажиров, естественно, целое) для очереди из 10 пассажиров до 5 для очереди из 100 пассажиров.

Как легко заметить, формально корректное математическое решение не очень хорошо выдерживает столкновение с реальностью.

Еще одна небольшая ремарка: процедура имитации посадки возвращает число пассажиров, сидящих не на своих местах, а если не на своем месте находится последний пассажир, то процедура возвращает отрицательное значение.

ПОДХОДЫ К АППРОКСИМАЦИИ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Теперь попробуем изучить свойства последовательностей изменений количества пассажиров, не сидящих на своих местах, и вероятность того, что последний пассажир не оказался на своем месте.

Используем функцию аппроксимации полиномом второй степени. После аппроксимации мы просто можем использовать вычисления по формуле, не производя трудоемких переборных расчетов.

Фрагмент 4. Вычисление коэффициентов полинома второй степени по трем его аргументам и значениям

```
int a3(float *x, float *y, float *abc)
{
 float m,r,n,z,t,v;
 if(x[1]==0.) return(-1);
 m=x[2]-(x[2]*x[2])/x[1];
 r=y[2]-y[1]*(x[2]*x[2])/(x[1]*x[1]);
 n=1-(x[2]*x[2])/(x[1]*x[1]);
 z=x[3]-(x[3]*x[3])/x[1];
 t=1-(x[3]*x[3])/(x[1]*x[1]);
 v=y[3]-y[1]*(x[3]*x[3])/(x[1]*x[1]);
 if(m==0.) return(-2);
 if((m*t-z*n)==0.) return(-3);
 abc[2]=(m*v-r*z)/(m*t-z*n);
 abc[1]=(r-n*abc[2])/m;
 abc[0]=(y[1]-x[1]*abc[1]-abc[2])/(x[1]*x[1]);
 return(0);
 }
```

Как легко видеть, аппроксимация сводится к решению системы трех линейных уравнений с тремя неизвестными и в переменной abc получаем соответственно коэффициенты аппроксимирующего полинома.

Фрагмент 5. Вычисление значения полинома второй степени

```
float p3(float x, float *abc)
{
 float r;
 r=abc[0]*x*x+abc[1]*x+abc[2];
 return(r);
}
```

Вычислим значение по ММК для n=200, аппроксимируем для значений 20, 100 и 200.

Фрагмент 6. Аппроксимация

```
pr_n=200;
        for(j=0;j<100;j++)
           {

            GenPermit(pr_n,prm1);
            rez=boarding(pr_n,prm1,0);
            sum1=sum1+abs(rez);
           if(rez<0) sum2++;
           }
          printf("\nN= %03d Med1 =%f Med2 = %f\n",
pr_n,(float)(sum1/100.),(float)(sum2/100.));
          AppLogD2(LOGNAME,pr_n,sum1,sum2);
          ss[200]=(float)(sum1/100.);pp[200]=(float)
(sum2/100.);
        xx[1]=20.;
        xx[2]=100.;
```

```
        xx[3]=200.;
        yy[1]=ss[20];
        yy[2]=ss[100];
        yy[3]=(float)(sum1/100.);
        r=a3(xx,yy,ab);
        AppLogD3(LOGNAME,ab[0],ab[1],ab[2]);
        for(i=1;i<21;i++)
           AppLogD4(LOGNAME,ss[i*10],p3((float)
(i*10),ab),ss[i*10]-p3((float)(i*10),ab));
        xx[1]=20.;
        xx[2]=100.;
        xx[3]=200.;
        yy[1]=pp[20];
        yy[2]=pp[100];
        yy[3]=(float)(sum2/100.);
        r=a3(xx,yy,ab);
        AppLogD3(LOGNAME,ab[0],ab[1],ab[2]);
        for(i=1;i<21;i++)
        AppLogD4(LOGNAME,pp[i*10],p3((float)
(i*10),ab),pp[i*10]-p3((float)(i*10),ab));
```

Получим следующий результат

```
N= 010 Med[pass] =2.820000 P[last] = 0.420000
N= 020 Med[pass] =3.490000 P[last] = 0.510000
N= 030 Med[pass] =3.960000 P[last] = 0.470000
N= 040 Med[pass] =4.050000 P[last] = 0.410000
N= 050 Med[pass] =4.700000 P[last] = 0.480000
N= 060 Med[pass] =4.570000 P[last] = 0.340000
N= 070 Med[pass] =4.460000 P[last] = 0.300000
N= 080 Med[pass] =4.800000 P[last] = 0.440000
N= 090 Med[pass] =5.260000 P[last] = 0.360000
N= 100 Med[pass] =4.970000 P[last] = 0.350000
N= 110 Med[pass] =5.520000 P[last] = 0.360000
N= 120 Med[pass] =5.560000 P[last] = 0.420000
N= 200 Med[pass] =11.300000 P[last] = 0.690000
```

Весьма интересный результат: гипотеза о том, что вероятность с ростом N снижается, на практике оказывается неверной.

Количество пассажиров не на своих местах определяется значениями следующего аппроксимирующего полинома.

$Med[pass]=0.000249N^2+-0.011367N+3.617776$

Первое — аргумент N, второе значение вычислено ММК, третье — значение аппроксимирующего полинома, четвертое — разность между ними.

```
10 2.820000 3.528999-0.708999
20 3.490000 3.490000 0.000000
30 3.960000 3.500778 0.459222
40 4.050000 3.561335 0.488666
50 4.700000 3.671668 1.028332
60 4.570000 3.831780 0.738220
70 4.460000 4.041669 0.418331
80 4.800000 4.301335 0.498665
90 5.260000 4.610780 0.649220
100 4.970000 4.970002-0.000002
110 5.520000 5.379001 0.140999
120 5.560000 5.837779-0.277779
130 0.000000 6.346333-6.346333
140 0.000000 6.904665-6.904665
150 0.000000 7.512776-7.512776
160 0.000000 8.170664-8.170664
170 0.000000 8.878328-8.878328
180 0.000000 9.635772-9.635772
190 0.000000 10.442992-10.442992
200 11.300000 11.299991 0.0000
```

Аппроксимация вероятности того, что последний пассажир не находится на своем месте. Легенда столбцов аналогична приведенной выше.

$P[last]=0.000030N^2+-0.005600N+0.610000$

```
10 0.420000 0.557000-0.137000
20 0.510000 0.510000 0.000000
30 0.470000 0.469000 0.001000
40 0.410000 0.434000-0.024000
50 0.480000 0.405000 0.075000
60 0.340000 0.382000-0.042000
70 0.300000 0.365000-0.065000
80 0.440000 0.354000 0.086000
90 0.360000 0.349000 0.011000
100 0.350000 0.350000-0.000000
110 0.360000 0.357000 0.003000
120 0.420000 0.370000 0.050000
130 0.000000 0.389000-0.389000
140 0.000000 0.414001-0.414001
150 0.000000 0.445001-0.445001
160 0.000000 0.482001-0.482001
170 0.000000 0.525001-0.525001
180 0.000000 0.574001-0.574001
190 0.000000 0.629001-0.629001
200 0.690000 0.690001-0.000001
```

Легко видеть, что в обоих случаях значения для аргументов 20, 100 и 200 не отличаются от полученных ММК, поскольку используются как базовые точки для аппроксимации.

Алгебраические объекты, связанные с задачей о рассеянном пассажире

Итак, очередь пассажиров описывается подстановкой, а рассадка пассажиров по местам – обратной подстановкой.

Фрагмент 7. Получение обратной подстановки

```
int GenPermit_1( int n, unsigned char *prm,
unsigned char *prm_1)
    {
     int i;
     if(n>=N_PRM) return(-1);
```

```
for(i=0;i<n;i++)prm_1[prm[i]-1]=i+1;
return(0);
}
```

Пусть прямая подстановка степени 20 такова: 002 014 004 015 012 007 017 005 003 019 010 009 020 011 018 001 013 016 006 008

Тогда обратная подстановка (размещение пассажиров на местах) — следующая: 016 001 009 003 008 019 006 020 012 011 014 005 017 002 004 018 007 015 010 013

16-й пассажир разместился на 1-м месте, 1-й – на втором и т.д.

Если подстановки очереди и рассадки перемножить следующей процедурой, то получится единичная подстановка, то есть подстановка, в которой p_i=i.

Фрагмент 8. Умножение подстановок

```
int MulPermit( int n, unsigned char *prm1, unsigned char *prm2, unsigned char *prm_r)
{
 int i;
 if(n>=N_PRM) return(-1);
 for(i=0;i<n;i++) prm_r[i]=prm2[prm1[i]-1];
 return(0);
}
```

Подстановки степени n образуют некоммутативную группу с операцией умножения, описанной фрагментом 5 порядка (порядок – количество элементов в группе) n! (n факториал). Именно в силу большой вариантности различных очередей целесообразно использовать ММК.

Использование подстановок в криптографии

Рассмотрим реализацию базового цикла зашифрования информации для алгоритма шифрования "Кузнечик" [4].

Фрагмент 9

```
// Базовая итерация в алгоритме зашифрования
int funcLSX1(unsigned char* a, unsigned char* b, unsigned char* outdata)
{
    unsigned char temp1[16];
    unsigned char temp2[16];

    funcX(a, b, temp1);
    funcS(temp1, temp2);
    funcL1(temp2, outdata);

    return 0;
}
int funcS(unsigned char* indata, unsigned char* outdata)
{
 int i;
 for(i=0;i<16;i++) outdata[ i] = kPi[indata[ i]];
 return 0;
}
```

Функция X реализует сложение векторов по модулю 2 (этот блок как раз смешивает шифруемый блок (открытый текст) с ключом), функция S- реализация подстановки степени 256 на каждый байт из 16-и байтового блока, а L1 – реализация регистра сдвига. Комбинация преобразования S и L1 реализуют нелинейное размешивающее преобразование.

Генерация случайных чисел также происходит путем использования алгоритма шифрования или хеширования.

Возвращаясь к предисловию, стоит заметить, что Б.С. Гершман как раз и говорил о том, что задачу о посадке в автобус целесообразно было бы решать методами программирования и ММК, а сдерживало это в то время (80-е годы прошлого века) отсутствие качественных и производительных генераторов (датчиков) случайных чисел.

Вернемся к функции генерации случайных чисел, использованной выше.

Фрагмент 10. Функция получения случайного вектора

```
void NextRandom16(unsigned char *prnd,unsigned char *rnd,unsigned char *krnd)
{
 int i;
 unsigned char keys[160];

 ExpandKey(krnd, keys);

 Encrypt_15_1(prnd, rnd, keys);
 for(i=0;i<16;i++) prnd[i]=prnd[i]^rnd[i];
 return;
}
```

Процедура с конкретными аргументами NextRandom16(rnd,rnd_1,rnd) формирует из текущего случайного числа rnd следующее случайное число путем установки текущего случайного числа в качестве ключа шифрования и последующего смешивания полученного (зашифрованного) и предыдущего случайного числа.

НЕКОТОРЫЕ МЫСЛИ ПО ПОВОДУ ИМПЛЕМЕНТАЦИИ ЗАДАЧИ В СИСТЕМЫ ИСКУССТВЕННОГО СОЗНАНИЯ

В настоящее время теория и практика искусственного интеллекта находится на определенном "перепутье".

Некоторые авторы предлагают продолжать строить имитационные модели на основе нейросетей и их обучения, другие переходят к более концептуальным объектам со сложной архитектурой типа искусственного сознания (ИСо) [5].

В частности, для решения описанных задач целесообразным и перспективным представляется механизм искусственного сознания, связанный с формулированием, решением и постановкой задач.

Своего рода тестом Тьюринга для искусственного сознания могла бы быть рассмотренная задача о рассеянном пассажире. Она содержит важные элементы когнитивной логики, делающей работу ИСо конструктивной:

- методы генерации и представления фундаментальных математических объектов (в данном случае, подстановок);

- методы реализации операций с ними, как базовых (умножение и обращение подстановок), так и развернутых (имитация посадки пассажиров или размещения грузов, находящихся в очереди, с учетом ошибок складской логистики, вызванной, как правило, именно человеческим фактором, например, банальным воровством);

- методы Монте-Карло для решения численных задач и получения данных для прикладной аппроксимации;

- методы внутреннего и внешнего контроля полученных результатов (проверки подстановок, защиты от зацикливания, превышения граничных значений параметров).

Сочетание данных механизмов делает ИСо достаточно креативным для решения весьма широкого круга практических задач и, кроме того, для интеграции в учебные процессы в роли "собеседник", "оппонент" и, возможно, "наставник".

Рассмотренная задача важна и для современной дидактики, особенно для дисциплин "прикладная математика" и "теоретическая информатика".

Достаточно важным и новым фактом является то, что детальное изучение случайного процесса методами программирования и моделирования дает более значимые прикладные результаты, чем формально корректные теоретические выкладки.

СПИСОК ЛИТЕРАТУРЫ

1. Проект МЦНМО при участии школы 57, Задача 65310. URL: https://problems.ru/view_problem_details_new.php?id=65310 (дата обращения: 12.01.2024)

2. Подстановки. URL: http://www.algebraical.info/doku.php?id=glossary:group:permutation (дата обращения: 12.01.2024)

3. Метод Монте-Карло. URL: https://ru.wikipedia.org/wiki/Метод_Монте-Карло (дата обращения: 12.01.2024)

4. Биктимиров М.Р., Щербаков А.Ю. Системно-аналитический подход к оптимизации алгоритма криптографического преобразования «Кузнечик» // Научно-техническая информация. Серия 2. Информационные процессы и системы, 2016. № 7. С. 9-10.

5. Щербаков А.Ю., Урядов А.В. Искусственное сознание: техническое задание в философской и естественнонаучной парадигме // Вестник современных цифровых технологий.- 2023. № 17. С. 4-13.