ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XII, 1 (1971)

J. KUCHARCZYK (Wrocław), DANUTA MACHULEC
and J. WĘGIERSKI (Katowice)

CONGESTION STUDIES IN RAILWAY STATIONS AND YARDS BY DIGITAL SIMULATION (I)

0. Introduction. One of the important problems met while considering the construction or enlargement of railway passenger stations and classification yards is how many tracks (platform tracks in passenger stations and reception or departure tracks in classification yards) should be provided to meet the expected operation demand. In the past, rule of thumb based on experience and intelligent guess played a significant role. Quite recently mathematical models were introduced to take into account the probabilistic aspect of the problem. The work of Potthoff, summed up in his book [4], was here, as we believe, the pioneering one. Worth mentioning are also [2], [3], [5], [7] and [10].

This paper presents the results of some of the research on that subject carried out at the Department of Railroad Development in Mining Industrial Districts (Zakład Rozwoju Kolei w Górniczych Okręgach Przemysłowych) in Katowice, a branch of the Research Institute of the Polish State Railways (COBiRTK) in Warsaw. The present, first part of the paper deals with a simulation model designed for a probabilistic study of congestion in the platform tracks of passenger stations and in the reception park of classification yards. After a presentation of the theoretical queueing model which forms the basis for the simulation, the program itself is described and the obtained simulation results briefly sketched by examples. A second, forthcoming paper will describe the simulation program and model used in the case of departure tracks of classification yards.

1. The queueing model. First, let us consider the simplified models of passenger stations and the reception part of classification yards. It is perhaps worth mentioning that the presented models are designed for our purpose only, i.e. for the investigation of congestion under assumed operation and traffic conditions.

Mathematically, a passenger station may be treated as a stochastic service system with as many servers as there are platform tracks. Arriving trains are regarded as customers wanting service. Although in reality trains may arrive from several directions, it is assumed that only one customer may arrive at any moment. In practice the probability of two or more trains arriving at the same moment is zero. The service time of a train is counted from the moment of the demand for the preparation of the train's track-way at the signal-box up to the moment of clearing the track-way by the train leaving the station. This time includes the approach time necessary for an incoming train to reach the platform, the stopping time at the platform, and the time necessary for the train to leave the station. For simplicity it is assumed that arriving customers are served independently, which in reality is not always the case because the movement of one train may hinder those of others. Dropping that assumption would involve the topography of the station track network to be taken into consideration, and this in turn would mean a loss of generality of the model. The independence assumption is not as severe as it may seem to be at the first look. Trains with near arrival times approach most probably from different directions and in most stations their entries may take place independently. Trains arriving from the same direction must have a time lag between them and this is taken into account by a properly chosen interarrival time distribution.

Railroad classification yards lead to a more complicated queueing model. Schematically, a classification yard may be treated as composed of four parts which the railroad cars pass in sequence (Fig. 1). Trains arrive

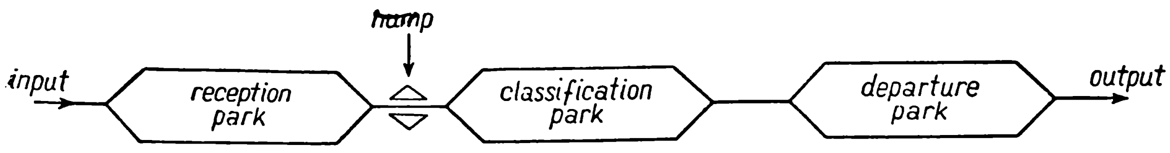

Fig. 1. Schematic representation of a classification yard

in the reception park (part one) where the cars are inspected and some other handling is required. Afterwards they are pushed over a hump (part two) and distributed into the tracks of the classification park (part three) according to their destinations. If sufficient cars in one direction have accumulated they are taken over to the departure park (part four) where a train is formed from them. After some service (as breaking inspection, taking a list of the cars, etc.) the train is ready to leave the yard and leaves it either immediately or after some waiting time, depending upon the traffic situation on the main line. In congestion studies it is not necessary to have a model of the whole yard. For incoming trains the behaviour of the first two parts (i.e. the reception tracks and the hump) is of importance, while for leaving trains a model of part four suffices.

The model which we present in this paper will be a model allowing the simulation of both passenger stations and reception parks in classification yards. Since the behaviour of the model in reception parks of classification yards is more complicated, we shall first consider that system.

Service at the reception end of classification yards takes place in two separate stages. In the first stage there are as many servers as there are reception tracks and service in the channels takes place independently of each other. For the second service stage an additional service channel, the hump, is needed and service takes place simultaneously in the appropriate reception track and at the hump. The reception track is cleared a little earlier than the hump but service of type two may begin only when service of type one is finished in the considered reception track. Moreover, the reception track in question may be occupied by a new customer wanting service of type one only when that part of service of the second stage of the preceding customer which blocks the reception track was finished. Thus, there is no proper waiting room between service stages one and two, and the customers have to wait for service of type two in their first-stage service channels. A queue is allowed before the yard if all reception tracks are occupied by trains either being served or waiting for service.

In passenger stations the situation is simpler, because there is no part two and all service takes place in one stage. In our model it suffices to assume zero length of service of type two for all trains.

Although the simpler situation encountered in passenger stations may be covered by known queueing models, the analytic solution is available for some specific arrival and service time distributions only. Moreover, no queueing model known to us describes the more complicated situation in the reception part of a yard. Because a study of the congestion problems met there was of great importance, we decided to construct a simulation model which would match both situations and which would allow, by a suitable choice of parameters, the investigation of any situation occurring in practice.

2. The simulation model. The simulation model is best described by presenting the Algol program. However, before doing so a short verbal description of the program will be given.

The program incorporates three random variable generators which form the realizations of the random variables for the interarrival and service time distributions. The time intervals between successive train arrivals are generated by the procedure *entry*, and the service times of types 1 and 2 are formed by the procedures *ser1* and *ser2*, respectively. Any distributions, however involved, may be provided there. In our study we actually used the following distributions: deterministic, exponential, Erlangian (with $k = 2$ and $k = 3$), and normal. The procedure form of the generators has practical advantages because during the testing and debugging of the program the procedure may easily be changed to read numbers from tape instead of generating them, thus enabling one to control the execution of the program and the obtained results. Furthermore, reading of numbers from tape may be advisable for awkward distributions, the realizations of which had been calculated earlier and printed on tape. Anyway, we found it convenient to have all distribution generators in form of procedures.

For the generation of uniformly distributed random numbers we used the following procedure:

```
real procedure unif;
  begin real x;
        x: = xp + xn + xn;
        xp: = xn;
        unif: = xn: = if x > 2 then x - 2 else
          if x > 1 then x - 1 else x
  end;
```

This procedures belongs to the family of Fibonacci-type generators and gives pseudo-random numbers from the interval (0,1). *xp* and *xn* are global variables of type **real** which at the beginning may have any different values from the interval (0,1). This generator has been tested on the Elliott 803 and Odra 1204 computers and proved very successful; it will be described more closely in a forthcoming paper by one of the authors.

Random numbers with the exponential and Erlang distributions have been obtained by the use of the following procedure:

```
real procedure erlang (lam, k);
  value lam, k;
  integer k;
  real lam;
  begin integer i; real s;
        s: = 1;
        for i: = 1 step 1 until k do s: = s × unif;
        erlang: = -lam × ln(s)/k
  end;
```

For $k = 1$ this procedure provides exponentially distributed random numbers, for greater k an appropriate Erlang distribution is obtained, *lam* being the mean in both cases.

Generation of deterministic numbers need no comment since it is obvious, and random numbers with a normal distribution were obtained by the use of the generator described in [1].

The actual simulation program works in two modes. First, there is the starting mode at the begin of which all counts are set to zero and the actual simulation starts from a zero state of the system; second, a continuation mode is possible in which the state of the system at the beginning is left unaltered, i.e. it is equal to the state in which some previous simulation was finished. The purpose of this is to allow a new simulation with changed distribution parameters to be performed without the necessity of bringing the system in an "equilibrium". The starting mode is provided for this last mentioned purpose. In addition, any initial state may be read in before starting a simulation.

The proper simulation program itself is given in the Appendix as a procedure with one parameter of type **Boolean.** This parameter, called *roz*, is set to **true** when a starting mode operation is required (the master program will then cause no print-out of the results), and it is set to **false** otherwise (the master program prints then the results of the simulation).

In the sequel we give an explanation of the relevant notation used in the program; this should facilitate an understanding of the program. The input parameter *m* denotes the number of service channels in the first stage, another one, *b*, is the time lag by which in the service of type two the first-stage service channel is cleared earlier than that of the second stage. The duration of the simulation is determined by the input parameter *lpkoniec*, the number of simulated train arrivals. These trains are counted by *liczba*. Several parameters are needed to characterize the relevant distribution generators; an example of them will be given later. All input is made by the master program. The number of occupied service channels and the number of customers in the queue are denoted by *lzt* and *lpk*, respectively. At every simulation time is counted from zero and *moment* gives the moment of the last (previous) change of state in the system.

At the arrival of a customer, labeled *przyp*, the three distribution generators are activated and the relevant numbers sent to *tp*, the moment of the arrival of the current customer, and to *to1*, *to2*, the service times of types 1 and 2, respectively. The moment of arrival of the next train, *tpn*, must also be generated. If an arriving customer finds a free service channel, this is occupied causing *lzt* to be increased by one and the moment of the end of service of type 1 and the duration of service of type 2 to be memorized in two arrays *tz1* and *tto2*, respectively. If, however, all service channels are occupied the customer forms the last queue member, thus an increase of *lpk* by one is made and the relevant numbers *tp*, *to1* and *to2* are stored in the corresponding arrays *tpp*, *tob1* and *tob2*. In this case an alarm print is provided if the queue length exceeds some critical value. In our program the greatest possible number of customers waiting in the queue is set to 50-*m*. At the end of this part of the program a test is performed to decide whether the next relevant event will be either the arrival of a new customer or the beginning or termination of service of type two.

The label *zakob* marks that part of the program which deals with the ending of a service of type 2 and, eventually, with the beginning of a new one. The Boolean variable *obsl* assumes the value **true** if service of type 2 takes place and is set to **false** otherwise, as it was at the very beginning of the simulation. If the service channel of type 1 was just freed and a queue of waiting customers exists, a new customer enters the free channel and occupies it for service of type 1; the queue length is then diminished by one. Then a new customer, if any, is selected for service of type 2 and the moment of its freeing the channel of type 1, *tz2*, is determined. The policy of selection for service is that of selecting the customer whose service of type 1 was (or will be) finished earlier. This does not reflect the actual policy in railroad operations practice, but theoretical arguments were provided to justify our selection policy. A test was made with a version of the program which selected customers for service of type 2 in the order of their arrival to the first-stage service channels. No substantial changes in the behaviour of the system resulted. Our selection policy seems to be especially appropriate under heavy traffic conditions, i.e. such which are interesting in congestion problems.

At any moment of a change of state of the system the variable *liczba*, the number of customers in the system, both waiting and blocking the service channel of type 1, is adjusted and two quantities, *licz*[*liczba*], indicating the number of times the state *liczba* occurred, and *czas*[*liczba*], the time during which the system was up to this moment in state *liczba*, are registered. They are needed for the printout of the results of the simulation. *lzat* counts the number of train stops before the station.

3. The passenger station case. In the case of passenger stations service of type one exists only. Therefore both *b* and *ser2* are set to zero. Interarrival time is assumed exponentially distributed. Though in reality a minimum time lag exists between successive train arrivals, this is so for one arrival direction only, and in the case of a great passenger station with trains arriving from several directions the exponential distribution is appropriate, since even short time intervals between arriving trains are then possible. Service time was found to be distributed according to the Erlang distribution with $k = 2$ plus some constant time.

Table 1 gives a comparison of real (i.e. according to schedule) and simulated behaviour of a system represented by a passenger station with 8 platform tracks. The input parameter is equal to 4.85 minutes. Daily 298 trains arrive at the station. The system was simulated for ten days and the results may be compared with those of reality. The main difference lies in the behaviour at congestion; the scheduled (by the time table) probability of having all service channels occupied is 0.022, while the simulated probability of the system being in the states 8 to 17 equals 0.084, i.e. is four times greater. Having in mind that a time table is never exactly satisfied in reality because of traffic perturbances, the latter probability seems nearer to truth than the former one. The last two columns in Table 1 give an account of the simulated behaviour of the same system when only 7 service channels are available, i.e. when one platform track has been closed for some reasons. The probability of having all channels occupied is then twice as great as before. In both simulations the numbers of trains being halted before the station have been counted and found equal to 261 and 537 trains, respectively. Thus the probability of an incoming train being stopped before the station equals 0.088 for 8 and 0.180 for 7 working service channels. Unfortunately no empirical data were available to confirm this.

The example of Table 1 shows how the simulation program may be used in practice. The behaviour of simulated passenger stations may be simulated under different technological changes, as e.g. in the case of platform track closures. Also the influence of an increase of traffic on the

TABLE 1. Comparison of system behaviour under schedule and in simulation

State of system	8 service channels					7 service channels simulated		
	No. of occurrences		Fraction of time spent			No. of occurrences	Fraction of time spent	
	schedule	simu-lation	schedule	simulation				
0	30	73	0.007	0.023		70	0.023	
1	170	313	0.065	0.080		308	0.078	
2	450	714	0.156	0.156		702	0.153	
3	830	1005	0.201	0.195		967	0.188	
4	1080	1030	0.208	0.169		991	0.163	
5	800	913	0.201	0.138		893	0.137	
6	450	704	0.086	0.094		703	0.092	
7	240	486	0.054	0.061		511	0.068	
8	100	297	0.022	0.038		329	0.040	
9		167		0.018		187	0.022	
10		109		0.013		118	0.013	
11		67		0.007		75	0.010	
12		37		0.004	0.084	46	0.006	0.166
13		22		0.002		27	0.003	
14		12		0.001		14	0.002	
15		4		0.001		9	0.001	
16		3		0.000		6	0.001	
17		1		0.000		1	0.000	

congestion in the station may be likewise investigated. Practical simulation results may be found in a book, written by the third author [11], which is designed for railway operations research workers.

4. The reception part of a classification yard. It was found from traffic statistics that trains arrive in classification yards in accordance with the Erlang distribution with $k = 2$. The service time distribution was found, however, of different kind. As stated before, there are two kinds of service to be performed there. First, the usual handling after arrival and preparing the train to push it over the hump; this was assumed to be normally distributed with mean *nm1* and standard deviation *si1*, and the realization denoted by *obs1*. Second, the pushing-over-the-hump process blocks the reception track as well as the hump. This service time was assumed to be normally distributed, too, with *nm2* and *si2* being the mean and standard deviation, respectively. The whole type 2 service time was then divided into two parts according to the proportion *pr* ; *obs2* denotes the time the reception track is occupied together with the hump, and *b* denotes the additional time necessary to free the hump for another service. The whole has been found to be in accordance with practice, stemming from different train lengths and various sizes of cuts in the pushing-over-the-hump process.

Results of the simulations being interesting for practical purposes may also be found in [11].

5. Conclusion. Against better knowledge, it has proved successful to write a simulation program in Algol. The idea was to write the program first in Algol, to test and debug it, and later on to rewrite it in machine code, possible one with symbolic notation. We had the Elliott 803 computer at our disposal at that time and a simulation of 2000 trains took one hour of computer time, if run in Algol. The Algol translator is not very effective for that computer and because of the need for several hundreds of simulations the Algol program was too much time consuming. The program had to be rewritten in the symbolic machine code AS [6]. However, at that time the Odra 1204 computer was released and a rather effective Algol compiler was written at the University of Wrocław [8], [9]. Our program was run on that computer, primarily only to test the compiler. It appeared that a simulation of 2000 trains took only 5-10 minutes. We then decided to run all simulations in Odra-Algol on that computer. Here existed, too, the possibility of rewriting the program in symbolic machine code, but for economic reasons we decided not to do so.

The coding, testing and debugging of the Algol program, together with discussions about the used model, took about 50 man-hours. Rewriting and testing the machine code version would probably require the same amount of time; this seemed unbearable as that time was sufficient to perform all simulations required.

Our experience allows us then to conclude that a general programming language, such as Algol or Fortran, may be used with success also for simulation purposes, provided the simulation runs do not consume an enormous amount of time. In any case we would advise anyone who has to write a simulation program and who has no access to a special simulation language to write and test his program first in a general language, and eventually only then rewrite it in machine code. Hours of man labour and computer time will then certainly be gained.

Our program, written in the form of a procedure, is reproduced in the Appendix. Also the more complicated simulation program of the departure part of a classification yard was written in Algol; a detailed description will be given in a forthcoming paper.

References

[1] *Algorithm 334*, Comm. ACM 11 (1968), no. 7, 498.

[2] R. Klein, *Über die Gleiszahl in Einfahrgruppen*, ETR-Rangiertechnik no. 19 (1959), p. 15-24.

[3] E. Mühlhans, *Die Bemessung der Gleiszahl in Einfahrgruppen mit Hilfe von Digitalrechnern*, ETR-Archiv f. Eisenbahntechnik, no. 22 (1967), p. 1-19.

[4] G. Potthoff, *Verkehrsströmungslehre*, vol. 1, Transpress, Berlin 1962.

[5] — *Die Bedienungstheorie im Verkehrswesen*, 2nd ed., Transpress, Berlin 1969.

[6] Krystyna Sochacz and J. Szczepkowicz, *A description of the AS-language for the Elliott 803 computer*, Zastosow. Matem. 9 (1968), p. 295-312.

[7] Н. Шабалин, *Применение теории массового обслуживания для расчета устройств станций*, Москва 1968.

[8] J. Szczepkowicz, *Programowanie w Odra-Algolu dla maszyny cyfrowej Odra 1204* (Programming in Odra-Algol for the Odra 1204 computer), Wrocław 1968.

[9] — *On table-driven syntax-checking within an Algol compiler*, Zastosow. Matem. 11 (1969), p. 3-87.

[10] W. Töpfer, *Die Zahl der Gleise in Gleisgruppen*, ETR-Archiv f. Eisenbahntechnik, no. 16 (1962), p. 2-44.

[11] J. Węgierski, *Probabilistic methods in rail transportation design* (in preparation).

DEPT. OF STATISTICS AND OPERATIONS RESEARCH, UNIVERSITY OF WROCŁAW,
AND
RESEARCH INSTITUTE OF THE POLISH STATE RAILWAYS,
DEPT. OF RAILROAD DEVELOPMENT IN MINING INDUSTRIAL DISTRICTS, KATOWICE

Received on 21. 3. 1970

APPENDIX

ALGOL PROGRAM OF THE SIMULATION

```
procedure sygp (roz);
  value roz; Boolean roz;
  begin
    integer lp; real to1, to2, tz1min;
    integer array licz[0: 50]; array czas [0:50];
    comment the procedure uses a lot of global variables, arrays
```

```
            and procedures which are described in the paper and
            must be declared in the master program;
      if roz then
        begin
          obsl: = false; lzt: = lpk: = liczba: = 0; tz2: = −b
        end roz
        else
        begin
          if lzt > 0 then
            begin
              for i: = 1 step 1 until lzt do tz1[i]: = tz1[i]−moment;
              if lkp > 0 then for i: = 1 step 1 until lpk do
                                tpp[i]: = tpp[i]−moment;
            end lzt > 0;
          tz2: = tz2−moment
        end ¬ roz;
      moment: = 0; lzat: = lp: = 0;
      for i: = 0 step 1 until 50 do
        begin czas[i]: = 0; licz[i]: = 0 end;
      tpn: = entry;
przyp: tp: = tpn; lp: = lp+1; if lp > lpkoniec then go to druk;
      to1: = ser1; to2: = ser2; tpn: = tp+entry;
      if lzt < m
        then begin
               lzt: = lzt+1; tz1[lzt]: = tp+to1; tto2[lzt]: = to2
             end
        else begin
              lpk: = lpk+1;, lzat: = lzat+1;
              if lpk+m > 50 then go to alarm;
              tpp[lpk]: = tp; tob1[lpk]: = to1; tob2[lpk]: = to2
            end;
      czas[liczba]: = czas[liczba]+tp−moment;
      licz[liczba]: = licz[liczba]+1;
      moment: = tp; liczba: = liczba+1;
      if obsl then go to nowezd;
zakob: if obsl then
          begin
            if k ≠ lzt then for i: = k+1 step 1 until lzt do
              begin
                tz1[i−1]: = tz1[i]; tto2[i−1]: = tto2[i]
              end;
            obsl: = false; lzt: = lzt−1;
            if lpk > 0 then
            begin
              lzt: = lzt + 1; tz1[lzt]: = tz2 + tob1[1];
              tto2[lzt]: = tob2[1]; lpk: = lpk − 1; lzat: = lzat + lpk;
              if lpk > 0 then for i: = 1 step 1 until lpk do
                begin
                  tpp[i]: = tpp[i + 1]; tob1[i]: = tob1[i + 1];
                  tob2[i]: = tob2[i + 1]
                end i
            end lpk > 0;
            czas[liczba]: = czas[liczba] + tz2 − moment;
            licz[liczba]: = licz[liczba] + 1;
            moment: = tz2; liczba: = liczba − 1
          end obsl;
           if lzt > 0 then
             begin
               tz1min: = ₁₀10;
               for i: = 1 step 1 until lzt do if tz1min > tz1[i] then
                 begin
                   tz1min: = tz1[i]; k: = i
                 end;
               tz2: = tto2[k] + (if tz2 + b ≤ tz1min then tz1min else tz2 + b);
               obsl: = true
             end lzt > 0;
nowezd:   go to if tpn ≥ tz2 ∧ obsl then zakob else przyp;
alarm:    comment alarm print to be inserted here;
druk:     comment insert printout of simulation results here;
         end sygp
```

J. KUCHARCZYK (Wrocław), DANUTA MACHULEC i J. WĘGIERSKI (Katowice)

BADANIE SYMULACYJNE PRACY STACJI KOLEJOWYCH OSOBOWYCH I ROZRZĄDOWYCH (I)

STRESZCZENIE

Praca zawiera opis modelu stacji osobowej i rozrządowej, traktowanych jako system obsługi masowej. W niniejszej, pierwszej części, przedstawiono szczegółowo model stacji osobowej i grupy przyjazdowej stacji rozrządowej i podano występujące w praktyce strumienie wejść i rozkłady czasu trwania obsługi. Ponadto opisany jest przedstawiony w dodatku program symulacji tych systemów wraz z użytymi generatorami wielkości losowych. Praca zawiera także omówienie wyników symulacji oraz wskazówki odnoszące się do techniki programowania.
