

BUSTLE - a bus simulation

John Appleby

University of Newcastle upon Tyne, England.

e-mail: john.appleby@ncl.ac.uk

◆INTRODUCTION◆

“WHY DO BUSES always seem to arrive in threes?” is a common, though not entirely justified, complaint. It’s usually blamed on the traffic, or on the bus drivers, and irregularity of service is a significant disincentive to increased use of public transport.

Yet traffic alone can’t explain it, as gains as well as losses will occur in the schedule, and buses close together would not necessarily get any closer. Passenger arrivals are another source of fluctuations in schedules, owing to the longer time spent at stops. We contend that, whereas traffic effects would not be expected to be cumulative, variations in queue lengths will be. A simple model, neglecting the effects of traffic, can demonstrate an important mechanism for the problem, and a simple simulation shows clearly what happens.

This model and simulation were originally constructed both for general interest and as a demonstration for school students. I wanted to show that there is more to mathematics than algebra, giving a motivation for modelling, statistics (specifically, arrivals in a queue represented as a Poisson process, and the use of random numbers) and computing.

In practice, the simulation has been used with a range of groups from age fourteen to second year undergraduates. The aim has been principally motivational, but different aspects can be developed with groups at different levels. With school students, a familiar issue provided an unfamiliar face of engineering. With Foundation Year engineering students, the modelling and computing aspects have been considered. With second year students, the queue simulation has been of interest.

◆ THE MODEL ◆

We assume that buses start equally spaced on the route and all drive at a constant speed. The effects of traffic are assumed to affect all buses equally during any time step. Variations in effects of traffic will perturb the state in which buses are equally spaced,

but there is no reason to suppose that they will be cumulative. However, the perturbation caused by variations in numbers of passengers is assumed to be much more significant. The eventual outcome is likely to be unaffected by neglect of traffic effects relative to this.

Passenger arrival is assumed to be random and independent, and follows a Poisson process. Thus 0, 1, 2 or more passengers may arrive at each time step at each stop. The rate is low, so 0 is the commonest number in any one step.

The central feature of the model concerns the delay as passengers board at a stop. We model the boarding as follows. Buses always stop, and if there is one passenger or none then no further delay is created. For additional passengers, it is assumed that the stopping time is proportional to the number of passengers waiting. Passengers alight much more quickly, so it is assumed that alighting is unimportant compared with boarding.

No maximum capacity is assumed for the bus, and no special action is taken once a bus catches up another, as the model seeks only to demonstrate the bunching process.

If ‘rapid boarding’ is incorporated, then the delay at a stop is constant, irrespective of the number of passengers. This represents the use of pre-paid tickets, bus conductors, etc.

◆THE SIMULATION◆

x-To illustrate the general ideas, a continuous circular bus route was sufficient, with circles representing buses, stops and passengers (figure 1).

Each segment of road between any two stops is divided into ten sections. During one time step, the following happens:

- any bus not at a stop, nor directly behind another bus, moves forward one section
- passengers may arrive at all stops
- any bus at a stop moves on if it has stayed for at least one time step and if there are no passengers waiting
- any bus at a stop where there are passengers stays

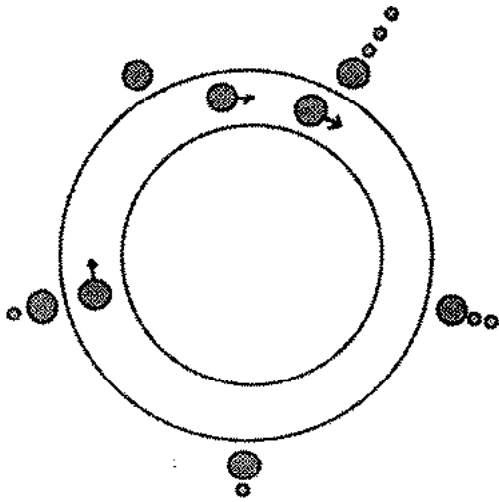


Fig 1. Buses beginning to bunch on the route

there, and one passenger ‘boards’ (i.e. vanishes from the screen)

The number of buses and stops can be varied, and the user can change all the colours too if they wish. (Some students can’t resist having e.g. red buses on a red background, so passengers vanish into thin air!) The speed can also be changed to show, initially, behaviour at stops, then to accelerate the bunching effect.

The other important changes are to vary the passenger arrival probability, and to permit ‘rapid boarding’. When this is enabled, buses stop for exactly one time step at each stop, so that they do not catch each other up. The only irregularity occurs when a passenger arrives at a stop whilst the bus is there, so that it stays for an extra time step.

Passenger Arrival

Passengers arrive according to a Poisson process but the frequency of arrival must be set to give a good simulation under varying conditions, namely different numbers of buses and stops, and also buses evenly spaced or bunched.

The Poisson process gives the probability of arrival of exactly n passengers during one interval as

$$p(n) = \frac{\lambda^n e^{-\lambda}}{n!}$$

where λ is the rate constant (and expected value). For small values of λ , $p(0)$ is close to unity, that is, during most time steps, no passengers will arrive at most stops.

For the simulation of a real bus-route, this rate would

be determined empirically, as well as other parameters of the model. For a simulation aimed at elucidating properties of the model, it must be chosen to give a ‘realistic’ number of passengers at each stop, that is, typically zero to three or four. Moreover, we wish this pattern of arrivals to continue after buses have started to bunch, so that we have neither zero nor too many passengers at stops too often.

After some experimentation and thought, the formula chosen was as follows:

$$\lambda = \lambda_0 \sqrt[3]{(nbus/nstop)}$$

where $nbus$ and $nstop$ are the numbers of buses and stops respectively, and where the coefficient λ_0 can be varied at run-time. The default value is taken to be 0.15. The reasoning behind this formula and this value is as follows.

Consider the case of a single bus and a single stop. During the ten time steps taken to move from the stop to the next arrival there, the number of passengers arriving is given by a Poisson process with rate 10λ (since the sum of Poisson variables is itself Poisson), and so

$$p(0) = e^{-10\lambda}, \quad p(1) = \lambda e^{-10\lambda}$$

$$p(2) = \frac{(10\lambda)^2}{2!} e^{-10\lambda}, \quad \text{etc}$$

For the value $\lambda_0 = 0.15$, this gives

$$p(0) = 0.223, \quad p(1) = 0.335,$$

$$p(2) = 0.251, \quad \text{etc.}, \quad \text{total } :p(0...4) = 0.982$$

so that there will usually be not more than three or four passengers waiting. The theoretical case of equal numbers of buses and stops, which continue to be equally spaced overtime, could follow exactly the same pattern. If there is still one bus, but more stops, then the number of time steps between arrivals at any one stop is increased, and so the rate is scaled inversely on the number of stops.

The case of more buses which may not be equally spaced is more complex. Initially we might want λ increased in proportion. However, if we do this, once the buses begin to bunch together there are too many waiting passengers for an effective simulation. For this reason, the scaling on $\sqrt[3]{(nbus)}$ was chosen, and has proved effective.

The generation of the Poisson data is described in the Appendix. Note that, for reasons of space in the display, the maximum number of passengers waiting at any stop

is limited to six. This limit is rarely reached before the buses are bunched (and therefore badly delayed).

◆RESULTS AND DISCUSSION◆

If two or more buses start travelling, equally spaced along the route, they encounter different numbers of passengers at the stops. If there is more than one waiting, then the bus is slightly delayed. On average, all buses get delayed a little at some stops.

However, suppose a bus is delayed more (by three or more passengers) or repeatedly. There are two consequences. Firstly, because the bus is delayed in its arrival at the next stop, there may be extra passengers there, increasing the delay. Secondly, the bus behind will be closer, and therefore will arrive at stops sooner after the previous bus, and probably collect fewer passengers. It is possible for subsequent arrivals for each bus, by chance, to reverse this process, but it is more likely that the delay will increase for the first bus, whilst the probability of delay for the second recedes. Ultimately, we expect the second bus to catch up with the first.

For more than two buses, the first bus in this analysis might itself catch up the one ahead.

On watching the simulation, the buses tend to adjust their relative positions for some time before the bunching effect is noticeable. Thereafter it proceeds rapidly, until all buses are proceeding in a ‘caterpillar’ fashion. The model doesn’t address the behaviour in this phase.

We can say that the initial state, in which the buses are equally spaced, is an *equilibrium* state, as the mean number of passenger arrivals is the same for all buses. In the simulation, we see the equal spacing being approximately maintained for significant periods. However, the perturbation introduced by a slight delay to one bus tends to grow, leading ultimately to the bunching effect bemoaned by real users. We can say, therefore, that the equilibrium is *unstable*, in that small departures from it tend to grow, rather than decay.

The *BUSTLE* program shows, when a key is pressed to change parameters, the average number of time steps per stop (calculated for one, typical, bus). Eleven steps are required simply to move between stops and to pause for the minimum time. Therefore, for a realistic simulation, we want the number to be in the range 11-14, i.e. between one and four

passengers at each stop. Using the default value of $\lambda_0 = 0.15$, experiment shows that the number of passengers waiting at each stop averages around two (steps per stop around 12), which gives an effective simulation.

With normal boarding, i.e. the bus waits at a stop for one time step if there is 0 or 1 passenger, and for n steps if there are $n > 1$ passengers, bunching may take some time, often ten or more complete circuits of the route, but eventually all will catch up with one.

If the basic passenger arrival rate λ_0 is changed, the bunching effect will also change, but the effects are only obvious and easily explained for extreme values. If a very low rate is used, then it is rare that any bus is delayed at all, and bunching may not happen for a long time. Conversely, if a high rate is used, then all buses are likely to be delayed, so bunching may not occur as quickly as for intermediate values. Note that if a high rate is used, it is also more likely that passengers will arrive whilst a bus is standing at a stop, so that in effect more than six passengers may board.

Another choice that affects the bunching process in this simulation is if the number of stops is high relative to the number of buses, e.g. $n_{bus} = 2$ and $n_{stop} = 9$. Since the passenger arrival rate is scaled in inverse proportion to n_{stop} , delays at any one stop become less common. However, varying the arrival rate would be expected to change this.

Rapid Boarding

With ‘rapid boarding’ there is some minor variation in the pattern of buses. This occurs because a passenger may arrive whilst a bus is at a stop, and the simulation then causes the bus to stay for one additional time step. This effect is random and creates no bunching. This demonstrates that one solution to the problem is to reduce significantly the delay to the schedule caused by a bus queue, by re-introducing bus conductors, or encouraging pre-payment (i.e. use of bus passes and multi-tickets). An alternative is to allow enough slack time in the schedule to absorb minor delays.

◆THE PROGRAM AVAILABILITY◆

The program runs under DOS, and will work under Windows 3.1 or 95/NT. The executable file ‘bustle.exe’ and the graphics driver ‘egavga.bgi’ (© Borland International) are freely available as a self-extracting zip-file from the author’s web page, address as follows:

<http://www.staff.ncl.ac.uk/john.appleby>

The source code, written in Turbo Pascal v5.0, is also provided. A Windows version of the program will probably be produced in the future. The program may be used freely for any educational purpose, and distributed freely under the same conditions, but may not be sold for any purpose.

Since this article was accepted for publication, we have become aware of the publication of a book by Rob Eastway and Jeremy Wynham entitled “Why do buses come in threes?” (Robson Books, 1998). Readers may wish to refer to this for a fuller analysis of this and related problems.

Appendix: Generation of Random Poisson Data

A pseudo-random number r in the range $[0,1)$ is generated by a function provided with the compiler used. To choose the number of passengers, we test repeatedly (up to some convenient limit) for successive numbers of arrivals given by $p(0)$, $p(1)$, etc.:

If $r \leq p(0)$ then 0 passengers arrive.

If $p(0) < r \leq p(1)$ then 1 arrives.

If $p(1) < r \leq p(2)$ then 2 arrive, etc.

For $\lambda_0 = 0.15$, it is highly improbable ($p < 2 \times 10^{-5}$) that more than three passengers will arrive in one time step, so the calculation stops at three.

Pseudo-random number generators of the kind provided within many compilers and packages are usually initialised, or ‘seeded’, to avoid repetition of the same sequence of numbers on each occasion (there are circumstances where exact repetition is desirable, so initialisation is not automatic). The ‘seed’ number is chosen arbitrarily; here we use the system time as its basis, so that each simulation is almost guaranteed to start with a different number.

Such built-in functions are often of the type known as ‘congruential generators’. An example of the kind of formula that might be used is:

Starting with an integer $I_0 = 3142$, generate a sequence of integers in the range $[0..6075]$ as $I_{n+1} = 106I_n + 1283 \pmod{6075}$, and then obtain a number in $[0,1)$ from $r_n = I_n/6075$.

Such formulae are quick, and adequate for purposes such as this simulation, but can suffer from *sequential correlation*, as well as their obvious limitation to a finite sequence of numbers. More sophisticated routines can be implemented when needed for other purposes. For this simulation, there is no need, though it *is* desirable that the sequence should start differently each time.