Západočeská Univerzita v Plzni
Fakulta Aplikovaných Věd

Bus Line Simulation

Filip Jašek

Předmět: KKY/MS1 (Modelování a simulace)
Vyučující: Ing. Hajšman Václav, Ph.D., Ing. Liška Jindřich, Ph.D., Ing. Janeček Petr, Ph.D.
Cvičící: Ing. Fetter Miloš Datum: 7. dubna 2022

1 Úvod a motivace

Pro svou semestrální práci jsem si vybral simulaci autobusové linky. Simulace byla vytvořena za pomoci balíčku "javaSimulation", který usnadňuje modelaci procesů v jazyce Java.

Téma práce bylo zpracováno z pohledu tvorby simulace a nástroje pro provozovatele hromadné dopravy ve městech. Po mírných úpravách např. odlišných časů mezi zastávkami lze použít i pro dálkové autobusové linky. Hlavním smyslem práce však bylo vytvořit simulaci, která by poskytovala informace pro dopravce a dokázala poskytnout odpovědi na požadovanou kapacitu autobusů nebo jejich frekvenci a následnou změnu jízdních řádů.

2 Vypracování

Před započetím programování cílového řešení bylo potřeba ujasnit si, jaké třídy budou pro zvolenou simulaci potřeba. K tomu posloužila následující dekompozice procesu provozu autobusové linky.
Třídy:

- `BusLineSimulation`
- `Bus`
- `BusGenerator`
- `BusStop`
- `Passenger`

2.1 BusLineSimulation

Tato třída slouží jako hlavní třída, v jejímž rámci probíhá samotná simulace. Kromě globálních parametrů pro celou simulaci a její analýzu obsahuje v metodě `actions()` cyklus pro počáteční inicializaci, vytvoření odpovídajícího počtu zastávek a vložení do seznamu `busLine`. Následně aktivuje generátor autobusů s předem definovanými parametry (`bus_capacity`, čas po kterém autobusy vyjíždějí apod.). Po nastavené délce simulace `simPeriod` v minutách a času navíc pro dokončení. Po doběhnutí simulace se do konzole vypíše výstup obsahující informace o průběhu a zakončení.

2.2 Bus

Třída pro reprezentaci autobusu a simulaci přepravy cestujících. Obsahuje informace o své kapacitě (`capacity`), aktuální zastávce (`busstopnumber`) a cestujících v seznamu `passengers`. Obsahuje navíc ještě seznam `getting_off`, do kterého se po naplnění autobusu zařadí cestující, co budou vystupovat na příští zastávce. Tento seznam není vyloženě nutný, ale v simulaci představuje lidi, kteří se postaví ke dveřím, aby co nejrychleji vystoupili.
Autobus při obsluze pracuje ve třech fázích. V první nechá vystoupit cestující, kteří chtějí vystoupit. Ve druhé nechá nastoupit lidi na aktuální zastávce, dokud to kapacita autobusu dovolí a nakonec proběhne zařazení vystupujících do seznamu `getting_off` a následně je autobus přemístěn po určeném čase na další zastávku.

2.3 BusGenerator

Jak již název napovídá, tato třída má na starosti generování autobusů v pravidelných intervalech podle jízdního řádu dokud neuplyne simulační čas. V cyklu vypouští nové autobusy na první zastávku, odkud pokračují na další zastávky v seznamu `busLine`.

2.4 BusStop

Reprezentuje zastávku, kam chodí lidé s cílem přepravit se na jednu z následujících zastávek. Zastávka s určitým nadhledem funguje jako generátor cestujících a byla tak i naprogramována. Po zařazení zastávky do seznamu `busLine` v počáteční inicializaci se spustí vnitřní cyklus, který začne generovat cestující do seznamu `waitingPassengers` reprezentující frontu na zastávce. Tuto frontu obsahuje každá vygenerovaná zastávka a z ní si pak autobus nabírá své cestující.

2.5 Passenger

Jednoduchá třída oddděděná od třídy `Link`, která neobsahuje žádné aktivní procesy. Každý cestující má však jeden parametr, jímž je výstupní zastávka, která mu je však náhodně přiřazena zastávkou. Náhoda je omezena s ohledem na nástupní zastávku a celkový počet zastávek.

3 Realizace

Výstupem simulace je následující text, který informuje o jejím výsledku. V první části se lze dozvědět informace o počtu autobusů, zastávek a cestujících, kteří byli nebo nebyli přepraveni ze zastávek během simulace. Následuje výpis nepřepravených cestujících stále čekajících na autobus na jednotlivých zastávkách.
Dále výstup obsahuje statistické informace o průměrném využití kapacit autobusů mezi jednotlivými zastávkami a časové srovnání proběhlé simulace a očekávaného ideálního času. Pro optimalizaci volby nových parametrů jsou vypsány i procentní využití času pro nástup, výstup a zpoždění.
Pro ukázkovou simulaci byly použity jak následující parametry:

- počet zastávek - 5
- kapacita autobusu - 10
- simulační čas - 12*60 min
- ideální čas mezi zastávkami - 10 min
- interval mezi příjezdy autobusů na první zastávku - 90 min
- generování zpoždění, chyb, délky nástupů/ výstupů atd. - náhodně

```
Simulation output...

GENERAL SIMULATION INFO
-----------------------
Generated buses: 9
Number of bus Stops: 5
Number of generated passengers: 273
Number of travelled passengers: 133
Number of passenger on 1. bus Stops: 8
Number of passenger on 2. bus Stops: 52
Number of passenger on 3. bus Stops: 41
Number of passenger on 4. bus Stops: 39
Number of passenger on 5. bus Stops: 0

SIMULATION STATISTICS
---------------------
Bus capacity:10
Average used bus capacity between 1. and 2. stop is: 70.0 %
Average used bus capacity between 2. and 3. stop is: 91.11111111111111 %
Average used bus capacity between 3. and 4. stop is: 97.77777777777777 %
Average used bus capacity between 4. and 5. stop is: 96.66666666666667 %
Ideal travel time thru busline with 1 min for stopping and boarding passengers: 44.0 min
Average bus travel time: 57.879226488546834 min
Boarding takes: 3.032841858605662 %
Exitting takes: 2.25373651158541 %
Traffic jam takes: 4.974542099515745%
Average time of boarding 1 passenger takes 0.11878547876267986 min
Average time of exiting 1 passenger takes 0.08827073187940108 min
```

Na následujících stranách v příloze na straně 4 je možné prohlédnou dříve popsaný zdrojový kód generující výstup výše.

4 Závěr

V této semestrální práci jsem si vyzkoušel jak se vytváří simulace reálného procesu a začal přemýšlet více o tom, jak by se i jiné procesy daly dekomponovat a simulovat, aby se bez testování dokázaly odhadnout potřebné parametry simulace.
Simulace autobusové linky se povedla přesně podle očekávání a pracuje tak jak má. Analytická vygenerovaná data odpovídají očekávání a dokážou zpřesnit představu o fungování reálného procesu. Jedná se však o velmi zjednodušený simulační model, ale dalo by se ho jednoduše rozšířit o další data nebo náhodné chyby či rozdílné generování lidí podle času během dne apod.
Závěrem považuji semestrální práci za zdařilou, splňující očekávání.

5 Příloha - 1

```
1  import javaSimulation.*;
2  import javaSimulation.Process;
3
4  public class BusLineSimulation extends Process{
5          /**
6          * class representing bus line
7          * by calling constructor of this class you will receive data related to your parameters
8          */
9          int noOfBusStops;//number of bus stops in the busline
10         Head busLine = new Head();//queue contains bus stops
11         Random random = new Random(5);
12         //variables for simulation analysis
13         int generated_passengers;//total number of generated passengers on busstops
14         int travelled_passengers;//number of transported passengers
15         int generated_buses;//number of generated buses in simulation time
16         double[] avg_used_bus_capacity;//average used bus transport capacity between bus stops
17         double realTravelTime;
18         double boardingTime;
19         double exitTime;
20         //variables for changing simulation environment
21         int bus_capacity;//capacity of generated buses
22         double simPeriod = 12*60;//simulation time
23         double idealTravelTime=10;//ideal travel time between two bus stops
24
25         BusLineSimulation(int n, int capacity) {noOfBusStops = n;avg_used_bus_capacity = new double[
               n];bus_capacity=capacity;}
26
27         public void actions() {
28                 //running simulation
29                 for (int i = 1; i <= noOfBusStops; i++)
30                 activate(new BusStop());//generating bus stops based on given parameter
31                 activate(new BusGenerator());//activating bus generator
32                 //hold for simulation time plus some added time to let simulation finish it work
33                 hold(simPeriod+1000000);
34                 report();//generate report about simulation
35         }
36
37         void report() {
38                 System.out.println("\nSimulation output...");
39                 System.out.println("Number of bus Stops: " + busLine.cardinal());
40                 //printing all individual passengers and printing their entry and exit stop
41                 Link stoplink = busLine.first();
42                 BusStop stop;
43                 for (int i = 1; i <= noOfBusStops;i++){
44                         stop = (BusStop)stoplink;
45                         System.out.println(("Number of passenger on "+ i +". bus Stops: " + stop.
                               waitingPassengers.cardinal()));
46                         stoplink = stoplink.suc();
47                 }
48                 //statistics based on fullness of busses
49                 System.out.println("Number of generated passengers: "+generated_passengers);
50                 System.out.println("Number of travelled passengers: "+travelled_passengers);
51                 System.out.println("Generated buses: "+generated_buses);
52                 for(int i=0;i<noOfBusStops-1;i++){
53                         System.out.println("Average used bus capacity between "+ (i+1) + ". and " +
                               (i+2)+ ". stop is: "+ 100*avg_used_bus_capacity[i]/(bus_capacity*
                               generated_buses) + " %");
54                 }
55                 System.out.println("Ideal travel time thru busline with 1 min for stopping and
                       boarding passengers: "+ ((idealTravelTime+1)*(noOfBusStops-1))+" min");
56                 System.out.println("Average bus travel time: " + (realTravelTime/generated_buses)+"
                       min");
57                 System.out.println("Boarding takes: " + (100*boardingTime/realTravelTime) +" %");
58                 System.out.println("Exitting takes: " + (100*exitTime/realTravelTime)+" %");
59                 System.out.println("Traffic jam takes: " + (100*(realTravelTime-((idealTravelTime+1)
                       *generated_buses*noOfBusStops))/realTravelTime)+"%");
60                 System.out.println("Average time of boarding 1 passenger takes "+ (boardingTime/
                       travelled_passengers)+" min");
61                 System.out.println("Average time of exiting 1 passenger takes "+ (exitTime/
                       travelled_passengers)+" min");
62
```

```
63
64
65          }
66
67
68
69          class Bus extends Process{
70                  /**
71                  * class representing bus in bus line simulation
72                  */
73                  public void actions(){
74                          //internal parameters specifying bus
75                          int capacity = bus_capacity;//
76                          int busstopnumber = 1;//actual number of busstop
77                          Head passengers = new Head();//passengers in bus capacity
78                          Head getting_off = new Head();//passengers exiting next stop (waiting near
                                doors)
79                          Link busstop_link = busLine.first();//link onto first bus stop
80                          //variables for helping with processing
81                          BusStop busstop;
82                          Link passengerlink;
83                          Passenger passenger;
84                          //statistics parameters
85                          double inTime;
86                          double totalInTime=0;
87                          double outTime;
88                          double totalOutTime=0;
89                          double start = time();//bus starting time
90                          //cycle what bus do from time its generated till final stop
91                          while(busstopnumber<=noOfBusStops) {
92                                  //exiting
93                                  while(!getting_off.empty()){
94                                          getting_off.first().out();//unboarding / exiting
95                                          outTime = random.uniform(0.0,15.0)/60;//generating random
                                                exit time
96                                          totalOutTime+=outTime;
97                                          hold(outTime);//time that passenger needs to exit bus
98                                  }
99                                  //boarding
100                                 busstop = (BusStop) busstop_link;
101                                 while(passengers.cardinal()<capacity) {
102                                         if(!busstop.waitingPassengers.empty()){
103                                                 passengerlink = busstop.waitingPassengers.first();
104                                                 passenger = (Passenger)passengerlink;
105                                                 passengerlink.out();
106                                                 passenger.into(passengers);
107                                                 travelled_passengers+=1;
108                                                 inTime = random.uniform(0.0,15.0)/60;//generating
                                                        random exit time
109                                                 totalInTime+=inTime;
110                                                 hold(inTime);//one passenger boarding time
111                                                 continue;
112                                         }else {
113                                                 break;
114                                         }
115                                 }
116                                 //counting passengers in bus for statistics
117                                 if(busstopnumber<noOfBusStops){
118                                         avg_used_bus_capacity[busstopnumber-1] += passengers.
                                                cardinal();
119                                 }
120
121                                 //nalezeni pasazeru, co vystupuji nasledujici zastavku
122                                 passengerlink = passengers.first();
123                                 passenger = (Passenger)passengers.first();
124                                 for(int i=0 ;i<passengers.cardinal();i++){
125                                         if(busstopnumber==noOfBusStops-1){
126                                                 passenger.out();
127                                                 passenger.into(getting_off);
128                                                 passengerlink = passengers.first();
129                                                 passenger = (Passenger)passengerlink;
130                                         }
131                                         if(passenger.exitStop==busstopnumber+1){
```

```
132                                             passenger.out();
133                                             passenger.into(getting_off);
134                                             passengerlink = passengers.first();
135                                             passenger = (Passenger)passengerlink;
136                                             continue;
137                                     }
138                                     passengerlink = passengerlink.suc();
139                                     passenger = (Passenger)passengerlink;
140                             }
141                             busstop_link = busstop_link.suc();
142                             busstopnumber += 1;
143                             hold(idealTravelTime+random.uniform(0.0,2.0));//travel time plus
                                    delay
144                     }
145                     //counting complete statistics
146                     boardingTime+=totalInTime;
147                     exitTime+=totalOutTime;
148                     realTravelTime+=time()-start;
149             }
150     }
151
152     class BusGenerator extends Process{
153             /**
154             * class representing some depo from which drive buses in periodic time given by bus
                    line schedule
155             */
156             public void actions(){
157                     //generating busses in fixed interval
158                     while(time()<=simPeriod){
159                             activate(new Bus());
160                             generated_buses+=1;
161                             hold(90);//waiting time between bus departures
162                     }
163             }
164     }
165
166     class Passenger extends Link{
167             /**
168             * class representing passenger with his own exit stop
169             */
170             int exitStop;
171             Passenger(int e){exitStop=e;}
172     }
173
174     class BusStop extends Process{
175             /**
176             * class representing bus stop
177             * passengers appears here so it works like passenger generator
178             */
179             Head waitingPassengers = new Head();//waiting passengers on bus stop
180             int no_of_stop;//process variable holding number of this bus stop
181             public void actions(){
182                     into(busLine);//taken into bus line
183                     no_of_stop = busLine.cardinal();//given number based on position in bus line
184                     if(no_of_stop<noOfBusStops){
185                             //generating passengers on busstop
186                             while (time() < simPeriod) {
187                                     new Passenger(random.randInt(no_of_stop+1,noOfBusStops)).
                                            into(waitingPassengers);//new waiting passenger
188                                     generated_passengers +=1;//statistics of total generated
                                            passengers
189                                     //statistics about generated passengers and their desired
                                            exit stop
190                                     System.out.println("pasazer cekajici na zastavce: "+
                                            no_of_stop+" vystupuje na zastavce: " + ((Passenger)
                                            waitingPassengers.first()).exitStop);
191                                     hold(random.negexp(1/11.0));//hold time defining frequency
                                            of generating passengers
192                             }
193                     }
194
195
196             }
```

```
197             }
198
199             public static void main(String args[]){
200                     activate(new BusLineSimulation(5,10));
201             }
202     }
```