Universitetet i Stavanger

Discrete Simulation of Bar Logistics and Resource Management

PROJECT REPORT

In the **DAT530** course of the master program **Computational Engineering**
at Universitetet i Stavanger

by

René König

Atanu Das

11th November 2022

Student numbers	268127, 268128
Course	DAT530
Supervisor	Prof. Reggie Davidrajuh

Abstract

GPenSIM is used to perform discrete simulations of a bar to gain insight into factors determining the profitability of a night for the bar.

The bar is modelled using a flexible, modular, timed and colored PetriNet, with staff members represented by resources, inhibitor arcs used for capacity constraints and costs calculated after runtime from the number of times transitions have fired.

Different cases representing different scenarios, starting at a good business night, then reducing the number of customers and next implementing different solutions to turn it into a profitable business night again, are set up, run multiple times and analysed.

A combination of reducing the number of employees, removing entrance fees and increasing drink prices is a viable option to turn an unprofitable business night into a profitable one.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

B	Bar
Cap	Capacity
Cust	Customer
Df	Dancefloor
Ent	Entrance
Gen	Generate
IMC	Intermodular Connector
MSF	Main Simulation File
n	Number of […]
NonAlc	Non-Alcoholic
pdf	Petrinet Definition File
Que	Queue
R	Resupply
S	Storage
T	Tables
Toi	Toilet
Wd	Wardrobe

1 Introduction

This project is part of the course DAT530: Discrete Simulation and Performance Analysis at Universitetet i Stavanger. The MATLAB toolbox GPenSIM is used to simulate the logistics and resource management at a bar, which are modelled using a PetriNet.

1.1 Problem Definition

> *"A new omicron variant of the Corona virus is spreading in Norway, and believed to be more resistant to vaccines." [1]*

This recent headline, referring to the newest COVID-19 variant BF.7, is troubling news for many Norwegians. In fact, as shown in Figure 1, experts estimate COVID-19 cases to rise again over the winter. [1], [2]

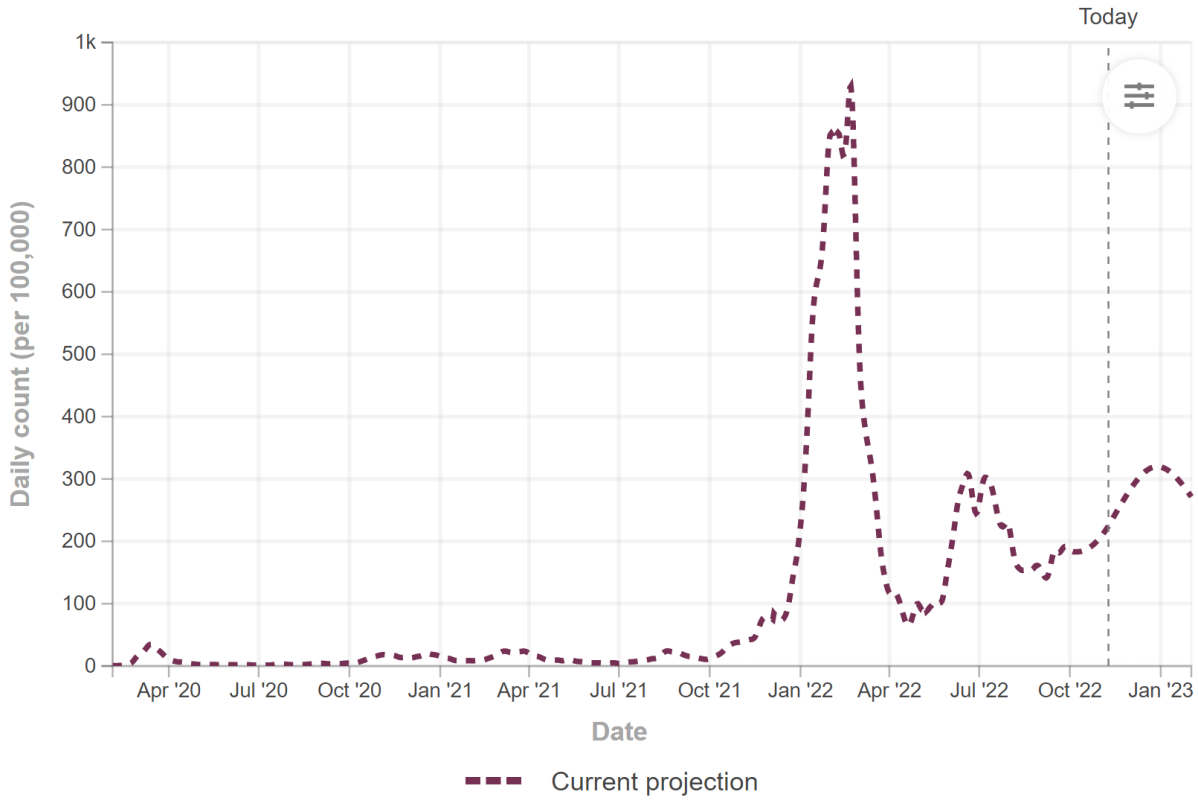

Figure 1: Estimated daily COVID-19 infections in Norway [2]

Owners of bars in Norway have been affected by COVID-19 and related restrictions, such as the complete ban on serving alcohol in bars from December 2021 to January 2022. [3], [4]

The COVID-19 Pandemic however is not the only thing making life difficult for bar owners. Another factor affecting the business of bar owners is the current inflation, with the consumer price index for September 2022 of alcoholic beverages and tobacco rising by 5%, Food and non-alcoholic beverages increasing by 12.1%, Housing, water, electricity, gas and other fuels increasing by 5.8%, Recreation and culture increasing by 5% and Restaurants and hotels increasing by 8.7%, compared to September 2021. [5]

Because of a shortage in CO2 bar owners fear the prices for beer could increase even more drastically over the winter. [6]

To overcome these difficult times, it is important for bar owners to manage their resources and adjust their bar design, pricing and business philosophy accordingly.

However, this is not trivial, as there are a lot of factors influencing the profitability of running a bar, with the results differing from night to night, depending on things like the customer numbers and number of drinks sold. Therefore, a model needs to be built to simulate what is happening at a bar during the night. The model needs to be flexible enough to be adjustable to different scenarios and should allow for different bar designs, as no two bars are the same.

2 Method and Design

A PetriNet model of the bar is first developed graphically using places (circles), transitions (rectangles) and arcs (arrows). This model is then implemented in the GPenSIM toolbox for modelling discrete-event systems for MATLAB, developed by Reggie Davidrajuh. [7]

2.1 Overall Design

The PetriNet model for the bar can be seen as a service center, where customers are moving through and guided through different stations of the service center, where they are potentially served by servers, until they leave the service center. Therefore the focus of the model is on modelling the movement of the customers throughout the bar. The customers are represented as tokens, which are moving from one place to another place via transitions.

2.2 Modular Approach

While the actual bar itself, where the customer's order and receive their drinks, is the primary service, there are other stations in the bar that a customer potentially visits on a night at the bar. These include the entrance, the toilets, the tables and the dancefloor. These stations can be modelled as separate modules, which are connected via an intermodular connector (IMC). Each module has different places and transitions to move customers from one place to another, as well as an in- and output ports.

2.3 Techniques

A timed PetriNet is used to model the different timings involved in the movement of the customers and staff members, such as the time it takes for the security guard to perform the security check on the customer at the entrance, the time it takes the bartender to make a drink, the time it takes the barback to collect empty glasses or resupply ingredients or the time during which a customer is occupied with dancing, sitting at a table or using the toilet.

The different staff members employed at the bar, namely the bartenders, barbacks, bar assistants and security guards, are represented as resources, which are requested by the transitions which they are used by before firing and released again afterwards. If a staff member is not available because they are preoccupied with

another task, the transition requesting the resource does not fire until the resource is released by the other transition. It should be noted that resources used by transitions inside a module cannot be released in the COMMON_POST processor, and therefore need to be released in the modular post processor of that module or the specific post processor of that transition.

Throughout the model inhibitor arcs are used to inhibit transitions from firing when the following place is at its pre-defined capacity.

Colored tokens are used to mark customers which have a jacket and subsequently disposed of their jacket at the wardrobe, to handle the wardrobe business when entering and exiting the bar.

To keep the wardrobe from overfilling with jackets or removing more jackets than are disposed the number of times the transitions for disposing and retrieving a jacket have fired are compared with the capacity of the wardrobe.

Another use of colored tokens is for security guards to mark customers that are causing trouble and guide them to the exit. This is done by preventing tokens with the color Exit from entering any other module.

The distribution of customers to the different modules happens randomly with definable inhibitor factors reducing the probability of customers entering a specific module or exiting the bar. This allows for different customer behaviours to be modelled by adjusting these inhibitor factors.

To calculate the costs and incomes throughout the night the cost or income from firing a transition is multiplied by the number of times that transition has fired throughout the night. Additionally fixed costs and employee costs are also considered on a per-night basis.

3 Implementation

There are five modules: Entrance, Toilet, Bar, Tables, Dancefloor which are connected through the IMC. In Figure 2 the interaction between these modules is shown in the global PetriNet structure.

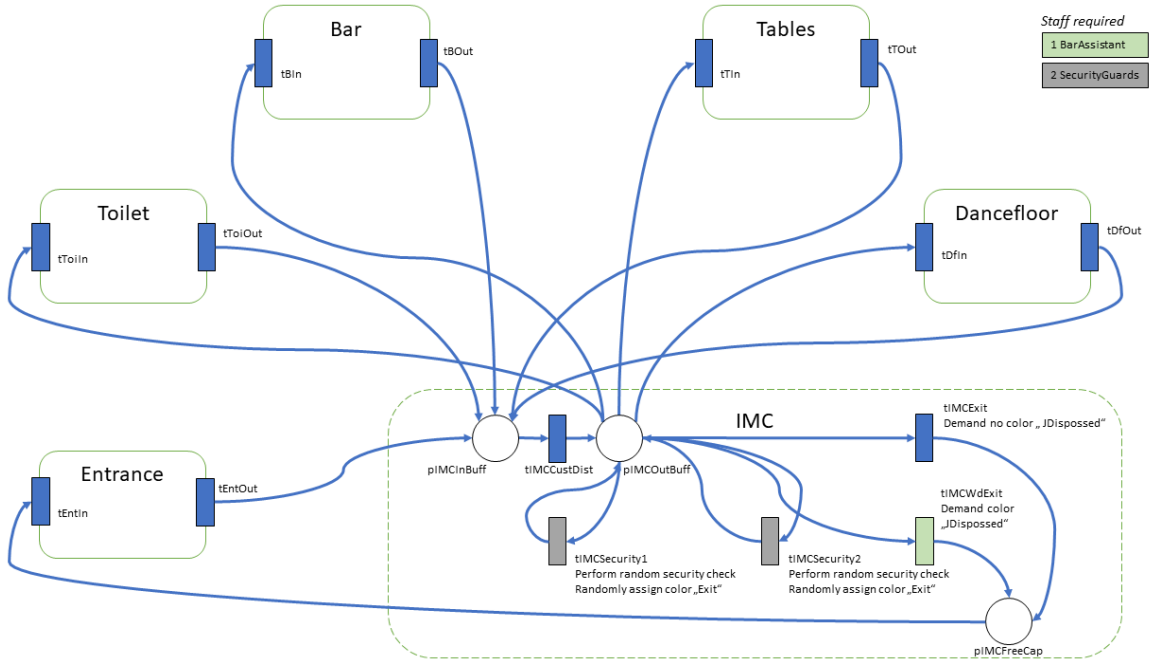

Figure 2: Global PetriNet Structure

Each transition and place name starts with t or p followed by a shorthand, like Ent for Entrance, marking the module it belongs to.

3.1 Modules

Each module has an input and output port through which a customer can enter or exit the module. An exception to this is the entrance module, where the input port is solely used to count the freed-up capacity from people leaving the bar through the exit.

3.1.1 Entrance

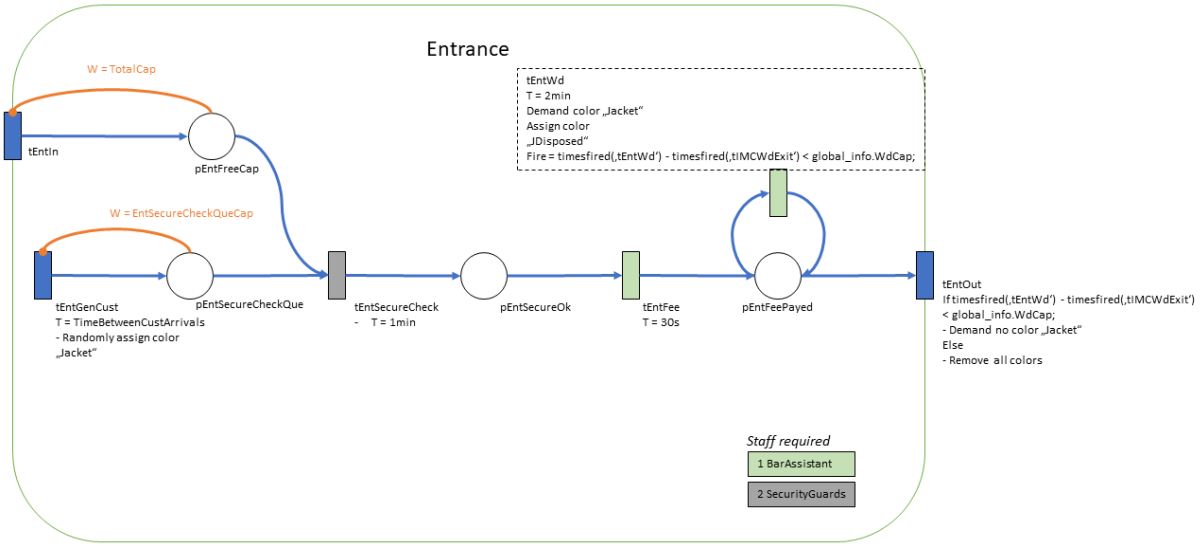

Figure 3: Entrance Module PetriNet Structure

The entrance is the first module in this model. It starts with tEntGenCust which generates customers according to the pre-defined time between customer arrivals. It also randomly assigns some customers the color Jacket, which is implemented in the modular pre-processor.

First, each customer waits in the security check queue (pEntSecureCheckQue).

If the security check queue is at its pre-defined capacity, the respective inhibitor arc disables the transition tEntGenCust.

The place pEntFreeCap starts with a number of tokens equal to the total capacity of the bar. If the bar is full, this place will be empty, disabling the security check transition (tEntSecureCheck). The inhibitor arc prevents this place from overflowing.

The security check needs one minute, during which the security guard checks if the customer is eligible to enter the bar, depending on their age, if they are too drunk or if they are carrying any illegal items.

When passing the security check the customer transitions to the next place pEntSecureOk. The customer then pays the entrance fee. A bar assistant is required to receive the payment which takes about 30 seconds. Once the payment is complete, the customer is in the pEntFeePayed place. If a customer has the color Jacket the next transition is tEntWd where a bar assistant receives their jacket and collect the wardrobe fee, which takes about 2 minutes. During firing the color Jacket is replaced by the color JDisposed. If the wardrobe is full, the customer avoids the wardrobe section and directly enters the bar through the tEntOut transition, where in this case the colors of their token are reset. tEntOut only allows customers without a jacket to enter the bar unless the wardrobe has reached its capacity.

3.1.2 Toilet

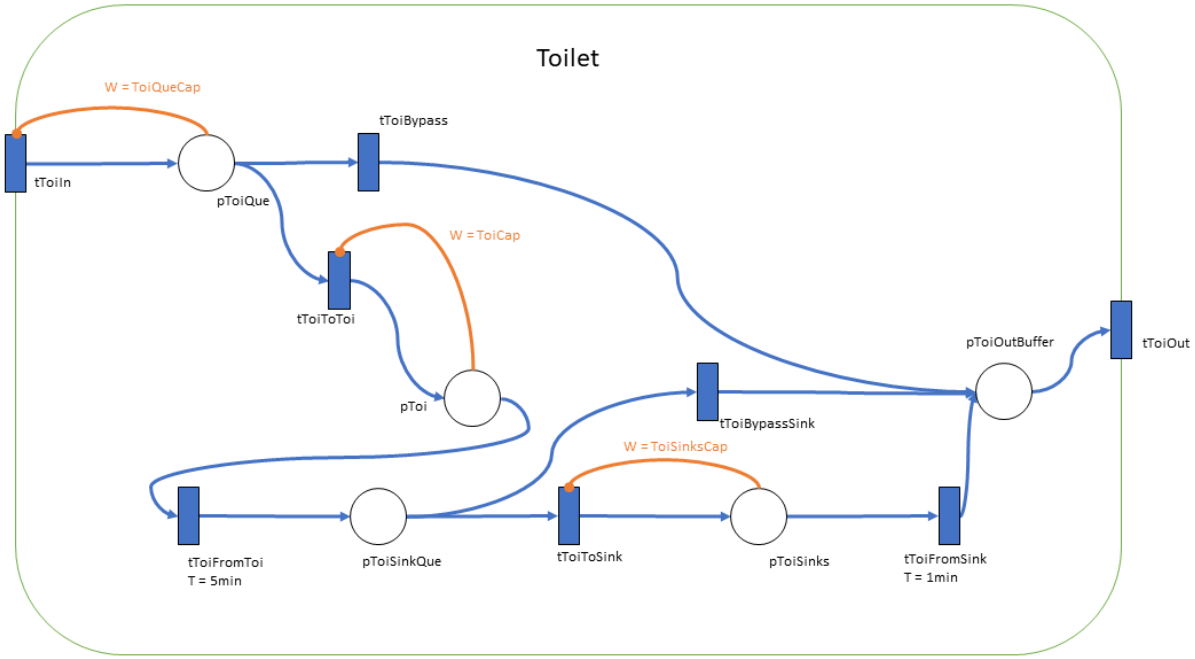

Figure 4: Toilet Module PetriNet Structure

The toilet module starts with the transition tToiIn. If the toilet queue is too long, the inhibitor arc from pToiQue disables the transition tToiIn so the customers cannot enter in the toilet. Customers in the toilet queue go to the toilet through tToiToToi unless the toilet pToi is at its capacity, at which the respective inhibitor arc disables the transition tToiToToi and the customer uses the bypass tToiBypass to directly go to the output buffer pToiOutBuffer. Note that the bypass is only enabled when the toilets have reached their capacity, which is implemented in the pre-processor of the module. The customer leaves the toilet after 5 minutes through the transition tToiFromToi and waits in pToiSinkQue. If the sinks are at their capacity, the respective inhibitor arc disables the transition tToiToSink. At any point the customers can randomly bypass the sink through the transition tToiBypassSink to the output buffer pToiOutBuffer. Otherwise, the customer moves on to pToiSinks to wash their hands through the transition tToiFromSink, which takes 1 minute, and then moves on to pToiOutBuffer, from where the customer leaves the Toilet module through the tToiOut transition.

3.1.3 Bar

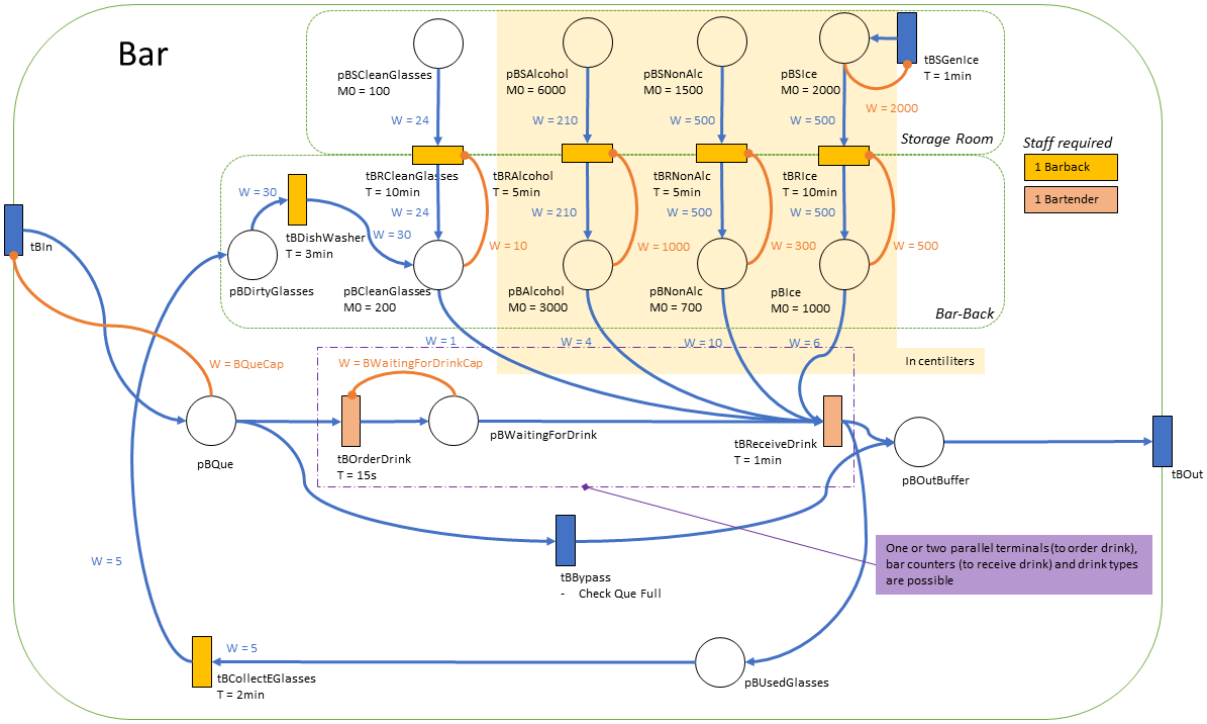

Figure 5: Bar Module PetriNet Structure

In the bar module, there are three types of tokens: Customers, glasses and ingredients. The bar is initially filled up with 200 clean glasses, 3000 centilitres of alcohol, 700 centilitres of non-alcoholic ingredients and 1000 centilitres of ice. In the storage room, there is a backup supply of 100 clean glasses, 6000 centilitres of alcohol, 1500 centilitres of non-alcoholic ingredients and 2000 centilitres of ice in the ice machine. If the ice in the ice machine is less than 2000 centilitres, it starts producing ice at a rate of 1 centilitre/minute until it is filled up to 2000 centilitres again, which is implemented by the respective inhibitor arc.

If the ingredients (or glasses) at the bar are below a certain threshold (for glasses, alcohol, non-alcoholic ingredients and ice, the thresholds are 10, 1000, 300 and 500 centilitres respectively), the barback resupplies the respective ingredient from the storage room. The barback can carry 24 clean glasses or 500 centilitres of ice, which both take 10 minutes, or 210 centilitres of alcohol or 500 centilitres of non-alcoholic ingredients, which both take 5 minutes.

The barback also collects used glasses throughout the bar and puts them into the dishwasher. They can carry 5 used glasses at a time, and it takes them 2 minutes. When 30 dirty glasses have been collected, they will run the dishwasher which takes 3 minutes to wash those glasses, after which they are available at the bar again.

The customers enter the bar module through tBIn. If there are more people in the bar queue than its capacity, the inhibitor arc from place pBQue disables the transition tBIn, so the customers cannot enter the Bar module. Customers in the queue give their drink order to a bartender and pay for their drink, which takes 15 seconds, after which they are waiting to receive their drink in pBWaitingForDrink. If this place reaches its capacity, the inhibitor arc disables the transition tBOrderDrink and the bypass is enabled through the modular pre-processor, allowing customers to bypass the bar through the tBBypass transition, going directly to pBOutBuffer. For customers waiting for their drink the bartender prepares their drink, which takes 1 minute and the customer will receive it through the tBReceiveDrink transition. This transition requires the respective ingredients for the drink to be available in the needed amount at the bar. In this case the recipe for the drinks is 4 centilitres alcohol, 10 centilitres of non-alcoholic ingredients and 6 centilitres of ice, which are served in 1 glass. After receiving their drink, the customer leaves the bar module through the pBOutBuffer place and tBOut transition. Additionally, a token is placed in pBUsedGlasses.

It is possible for the bar to have up to two order terminals to order two different types of drinks and two counters to receive the respective drink. In this case the lanes for drink1 and drink2 run parallel.

3.1.4 Tables

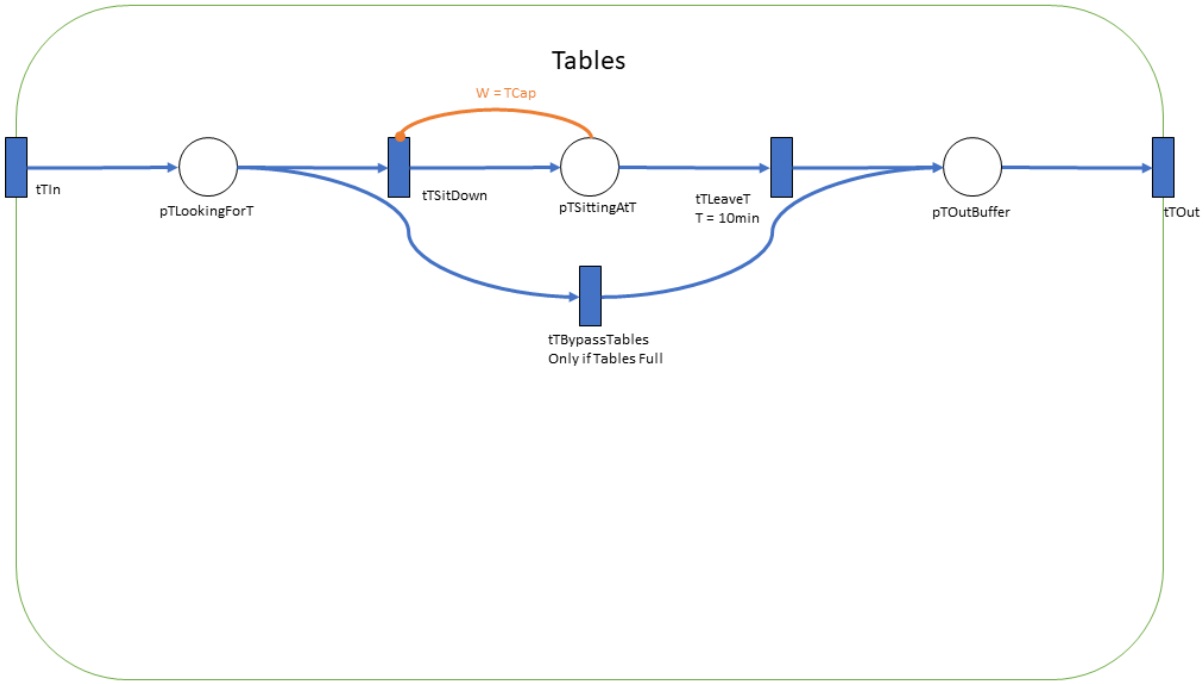

Figure 6: Tables Module PetriNet Structure

The customer enters the tables module through tTin. After entering the module, they are looking for a free place at a table. If there is a free place, they sit down through tTSitDown and sit at the place at pTSittingAtT. If pTSittingAtT is full, the inhibitor arc disables the transition tTSitDown. Simultaneously the bypass tTBypassTables is opened in the pre-processor of the module, allowing the customers to bypass the tables and directly move to the output buffer tTOutBuffer. After sitting at a table for 10 minutes the customer leaves the tables through tTleaveT, pToutBuffer and tTOut.

3.1.5 Dancefloor

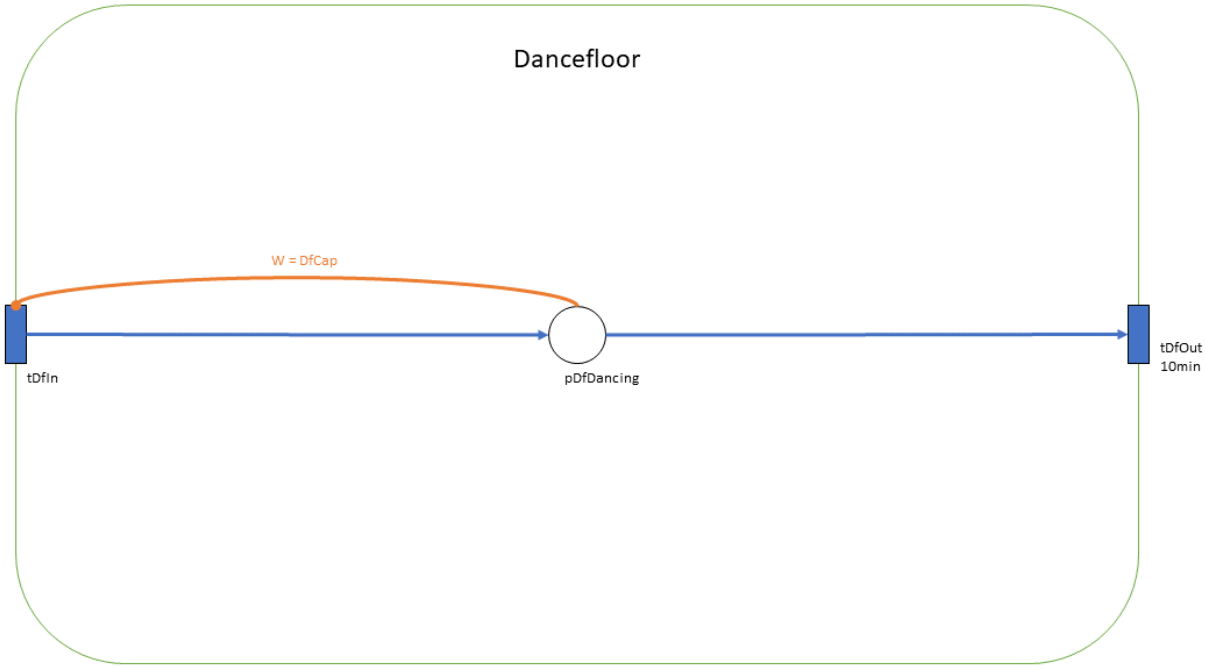

Figure 7: Dancefloor Module PetriNet Structure

The dancefloor module starts with the transition tDfIn, through which the customer enters the dancefloor. They are dancing at pDfDancing for 10 mintutes and leave the place through the transition tDfOut. If the dancefloor is full, the inhibitor arc disables the transition tDfin, so the customers cannot enter the dancefloor.

3.2 Intermodular Connector

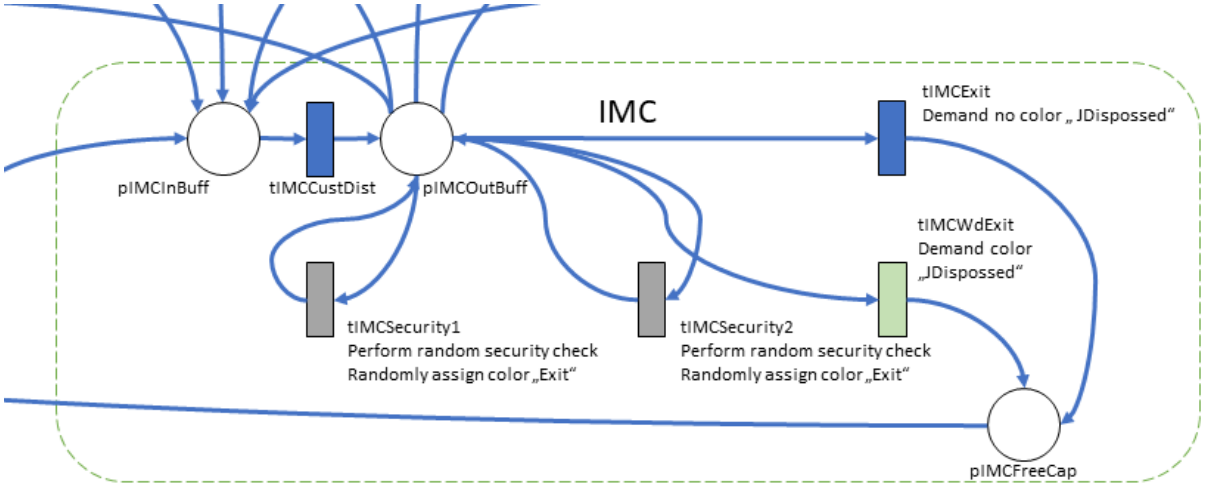

Figure 8: IMC PetriNet Structure

The IMC connects all other modules. Customers come from the output ports of the entrance, toilets, bar, tables or dancefloor module into the IMC input buffer pIMCInBuff. They transition through tIMCCustDist to the IMC output buffer pIMCOutBuff, from which they can enter the modules toilets, bar, tables or dancefloor through their respective input ports. If the customer has not disposed their jacket in the wardrobe when entering the bar, they can also leave the bar through tIMCExit. If the customer disposed their jacket in the wardrobe, they can leave the place through tIMCWdExit where a bar assistant returns them their jacket.

The security guard can also perform random security checks and can potentially throw out troublesome customers. This is achieved through a color Exit being assigned to the troublesome customer, which prevents them from using the input transitions of the modules and leaves them no other option than to exit the bar through one of the exits. This logic is implemented in the common pre-processor. From tIMCExit and tIMCWdExit the freed-up capacity is counted in pIMCFreeCap and passed on to the entrance module, where it is used to determine how many more customers can enter the bar. To control the customer behaviour the random distribution of customers to the different modules or the exit can be adjusted using inhibitor factors, which control the probability of a customer entering a specific module. This is implemented in the common pre-processor, but the inhibitor factors can be changed in the main simulation file (MSF).

3.3 Design Alternatives

Different approaches for handling the wardrobe business have been examined. Initially the wardrobe was supposed to be its own module, as depicted in Figure 9, through which customers would have to pass after entering the bar and before exiting.

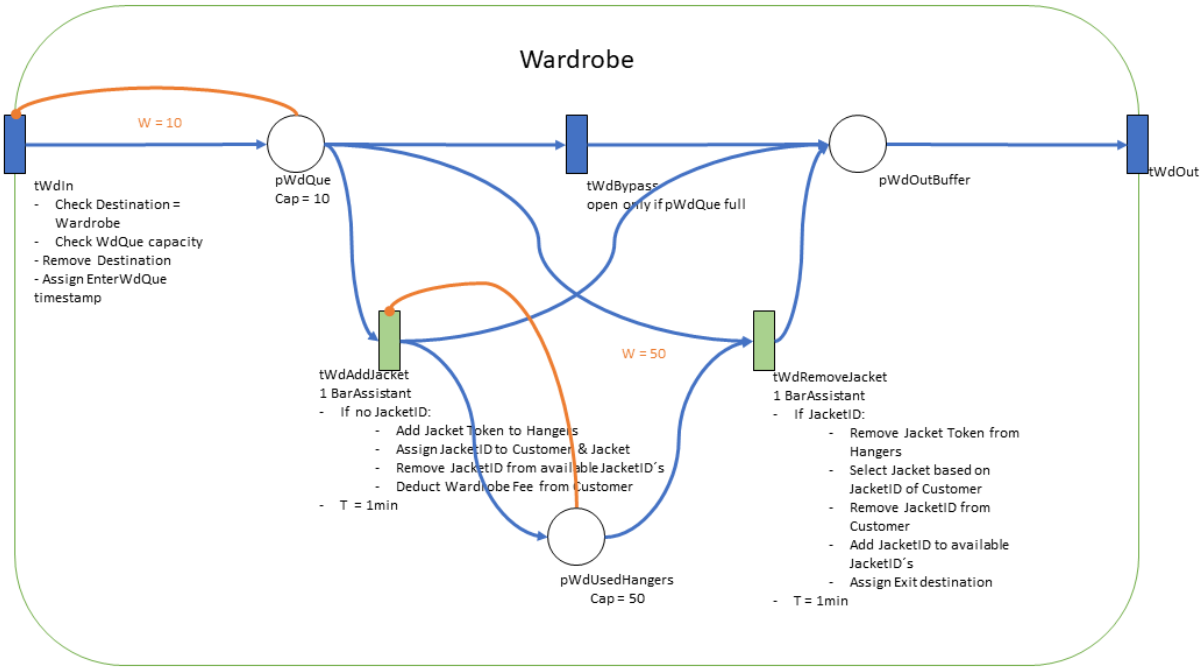

Figure 9: Work in Progress of Discarded Wardrobe Module PetriNet Structure

The tWdAddJacket and tWdRemoveJacket transitions would only take customers depending on whether they had disposed their jacket before or not. This would have been done through a color JDisposed assigned to the customer token by tWdRemoveJacket. Additionally, a color with the JacketID, referring to the hanger on which the jacket would be placed inside the wardrobe, would be assigned to that customer token. The JacketID would be taken from a pool of available JacketIDs in a global variable, that would be updated with every jacket that was added to or removed from the wardrobe. Removing these specific colors from a customer without overwriting other tokens they might posses was more complicated than expected, since GPenSIM does not provide a function to remove specific colors from a token. A similar approach where the available hangers in the wardrobe were to be represented by specific resources failed because of a similar problem, where GPenSIM does not allow the release of specific resources. Therefore these

approaches were disregarded as they were proven to be too complex, causing reliability and performance issues. The benefit of being able to pair jackets and customers and to track individual jackets was not relevant to the overall problem and did therefore not justify the added complexity.

The inbuilt functionality of GPenSIM for calculating cost based on tokens or transition firings (including timing) was not used, since it was not needed for the relatively simple cost calculations, which can all be calculated after running the simulation by counting the number of times a transition has fired, therefore not warranting the additional complexity in the pre-processor files. This also increases performance compared to performing these calculations at each transition firing during runtime.

Using inhibitor arcs to implement capacity constraints also proved to increase performance compared to querying the number of tokens in the following place during transition firing to stop it from firing, as the transition won't be enabled when inhibited and therefore there is no need to perform its pre-processing.

4 Testing, Analysis and Results

Different cases representing different scenarios are implemented and analysed.

4.1 Simple User Manual

To create a scenario, the parameters in the beginning of the main simulation file (MSF) can be adjusted from the base case, which is represented in the following code snippets, to represent a given case. Figure 10 shows the parameters representing the design of the bar, such as the different capacities and the number of terminals to order a drink and counters to receive a drink.

```
%%%Bar Design%%%
global_info.TotalCap = 50;
global_info.EntSecureCheckQueCap = 20;
global_info.WdCap = 25;
global_info.BQueCap = 10;
global_info.BWaitingForDrinkCap = 5;
global_info.DfCap = 20;
global_info.TCap = 20;
global_info.ToiQueCap = 10;
global_info.ToiCap = 8;
global_info.ToiSinksCap = 4;

global_info.BnOrderTerminals = 2;
global_info.BnCounters = 2;
if global_info.BnOrderTerminals == 2
    global_info.BnCounters = 2;
end
```

Figure 10: Code Snippet MSF Bar Design

Note that it is only possible to have one or two order terminals and counters. If there are two order terminals there need to be two counters as well, as each order terminal – counter lane will be serving its own drink type, drink1 and drink2 respectively.

Note that these parameters are assigned to global variables as they are referenced in the PetriNet definition files (pdf) of their respective module.

The recipes for the drinks can be adjusted in the pdf of the bar module by adjusting the weights for each ingredient per drink.

The time it takes the bartender to prepare drink1 and drink2 can be adjusted in the respective firing time of the transition ‘tBReceiveDrink1’ or ‘tBReceiveDrink2’

Note that these weights and firing times need to be adjusted across all the cases of combinations of number of order terminals and counters.

Figure 11 shows the parameters for the number of employees for each of the four employee types.

```
%%%Employee numbers%%%
nBarAssistant = 2;
nBarback = 1;
nBartender = 2;
nSecurityGuards = 4;
```

Figure 11: Code Snippet MSF Employee numbers

Figure 12 shows the parameters for the different costs, divided into variable costs, employee costs and fixed costs.

```
%%%Costs%%%
%%Variable costs (per unit sold)%%
Costs.Ingredients_Drink1 = 25;
Costs.Ingredients_Drink2 = 25;

%%Employees costs (per employee, per hour)%%
Costs.EmployeeBarAssistant = 185;
Costs.EmployeeBarback = 185;
Costs.EmployeeBartender = 185;
Costs.EmployeeSecurity = 200;
EmploymentHours = 6;

%%Fixed costs (per night)%%
Costs.Fixed.Artists = 300*(EmploymentHours-1);
Costs.Fixed.Electricity = 2000;
Costs.Fixed.Water = 350;
Costs.Fixed.Rent = 5000;
Costs.Fixed.MusicFees = 250;
Costs.Fixed.Insurance = 500;
Costs.Fixed.Maintainance = 750;
Costs.Fixed.ToiletPaper = 100;
Costs.Fixed.Soap = 100;
Costs.Fixed.PaperTowels = 100;
```

Figure 12: Code Snippet MSF Costs

The costs for the ingredients of drink1 and drink2 are the combined costs for all the ingredients per drink sold.

The employee costs are per employee and per hour. The employment hours per night are defined here as well.

The fixed costs are calculated per night and include the cost for the artists, which is calculated from an hourly price and assumed to be working one hour less than the

regular employees, as they are not involved in the preparation and clean-up of the bar before and after the night.

Figure 13 shows the parameters to calculate the income, namely the entrance fee, wardrobe fee and prices for drink1 and drink2.

```
%%%Income%%%
EntranceFee = 150;
WardrobeFee = 25;
PriceDrink1 = 125;
PriceDrink2 = 98;
```

Figure 13: Code Snippet MSF Income

Figure 14 shows the parameters to adjust customer and security guard behaviour.

```
%%%Customer Behaviour%%%
TimeBetweenCustArrivals = [0 1 0];
%Inhibitor factors, the higher the number (integer), the less likely a
%customer will visit that part of the bar
global_info.BInhibFact = 1;
global_info.TInhibFact = 3;
global_info.DfInhibFact = 3;
global_info.ToiInhibFact = 3;
global_info.ExitInhibFact = 3;

%%%Security Behaviour%%%
%Inhibitor factors, the higher the number (integer), the less likely a
%security guard will perform that action
global_info.SecurityCheckInhibFact = 50;
global_info.SecurityThrowOutInhibFact = 5;
```

Figure 14: Code Snippet MSF Behaviours

This includes the time between customer arrivals at the entrance of the bar and inhibitor factors to adjust the random distribution of customers to the different bar modules inside the bar. Note that these inhibitor factors must be positive integers and increasing an inhibitor factor lowers the probability of a customer entering the respective module. To increase the probability of a customer entering a specific module the inhibitor factor of this module must be decreased or alternatively the inhibitor factors of all other modules must be increased.

Similarly, the inhibitor factor for the random security checks and throwing out of troublesome customers can be adjusted to in- or decrease the probability of these events.

The initial markings of places of ingredients starting with ‘pB’ or ‘pBS’, as shown in Figure 15, can be changed to adjust the available supplies at the bar or in storage at the beginning of the night respectively.

```
dyn.m0 = {'pEntFreeCap',global_info.TotalCap,... %Customer movement
          'pBCleanGlasses',200,'pBAlcohol',3000,'pBNonAlc',700,'pBIce',1000,... %Initial Bar supplies [glasse
          'pBSCleanGlasses',100,'pBSAlcohol',6000,'pBSNonAlc',1500,'pBSIce',2000}; %Initial Storage supplies
```

Figure 15: Code Snippet MSF Initial Markings

The respective firing times of transitions can be changed to adjust the timing of everything happening at the bar.

The weights of the respective arcs in the bar pdf can be changed to adjust the drink recipes, carrying capacity for each ingredient when resupplying the respective ingredient and handling of glasses.

The weights of the respective inhibitor arcs in the bar pdf can be changed to adjust when each ingredient needs to be refilled.

4.2 Example Run

This sample run represents the base case (Case 1). The parameters are chosen as depicted in section 4.1 Simple User Manual. In this case we are looking at a good night for the bar owner with a high number of customers, as it likely has been experienced on a Saturday night pre-COVID-19.

Figure 16 shows the number of tokens in each place of the entrance module over time. Note that the time continues over 24 instead of restarting from 0.

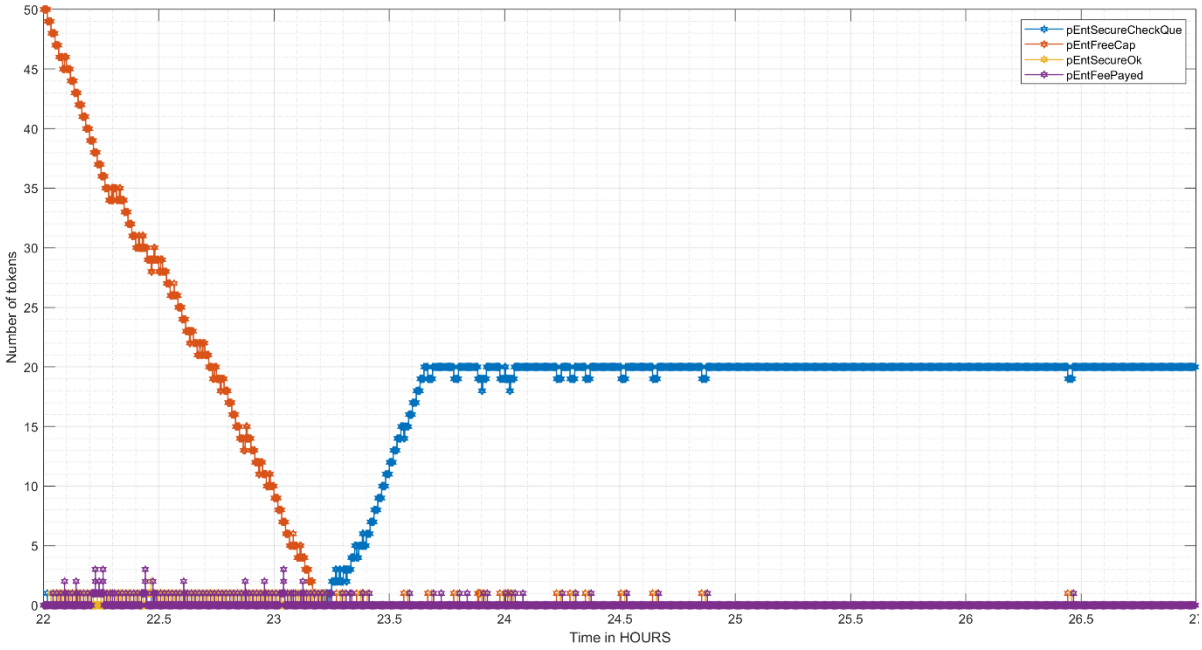

Figure 16: Case 1 Number of Tokens Entrance

In the beginning of the night, as customers are going through the security check and paying the entrance fee to enter the bar, the free capacity decreases. When the maximum capacity of the bar is reached at around 23:15 a queue starts building up in front of the bar before the security check. The security guards are now only performing security checks and letting in new customers, when another customer leaves the bar, freeing up capacity. As a long entrance queue makes the bar unattractive to new customers no new customer enters the queue when it is longer than 20 customers.

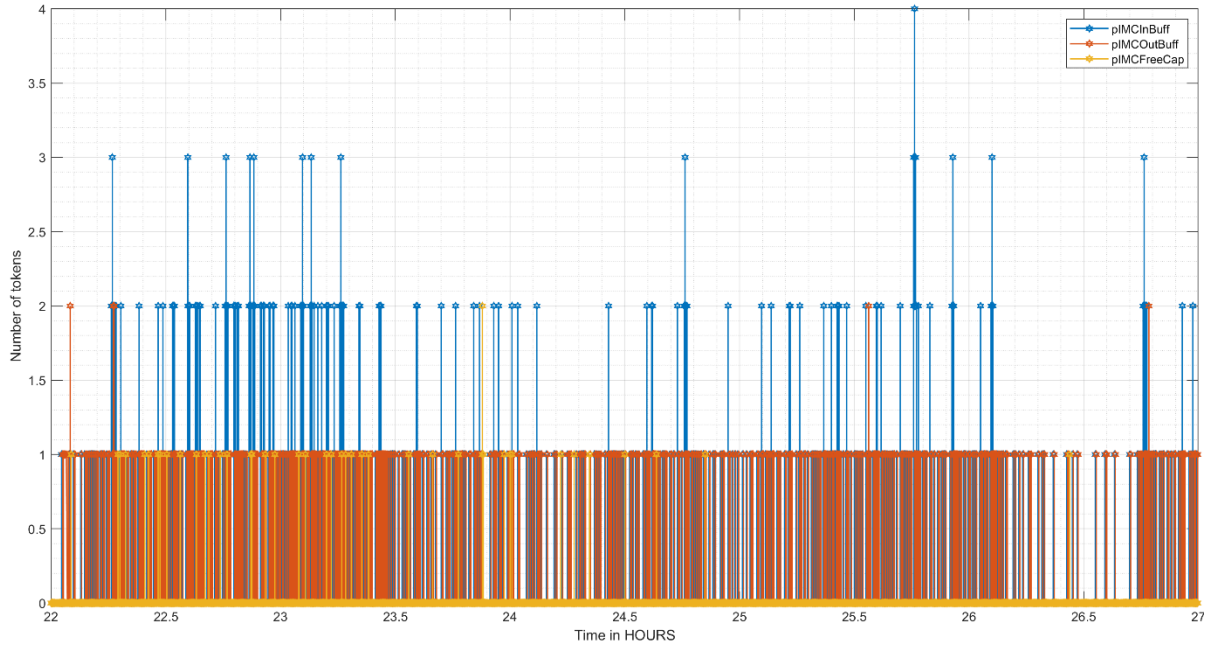

Figure 17: Case 1 Number of Tokens IMC

As the transitions in the IMC and in- and output ports of the modules are virtually instantaneous there is not a lot of built up in the places of the IMC, as can be seen in Figure 17. The only time customer accumulate in the input buffer of the IMC is when customers are exiting multiple different modules at the same time.

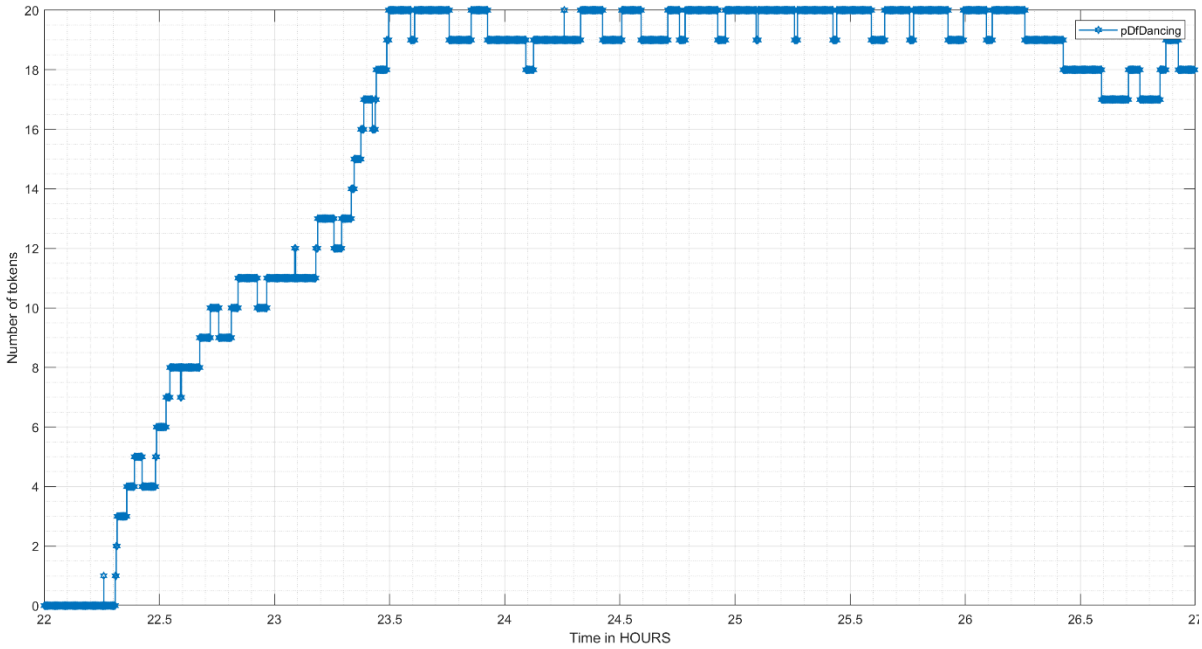

Figure 18: Case 1 Number of Customer Tokens Dancefloor

Figure 18 shows that about 15 minutes after the bar opens customers start accumulating on the dancefloor until it reaches its capacity at around 23:30.

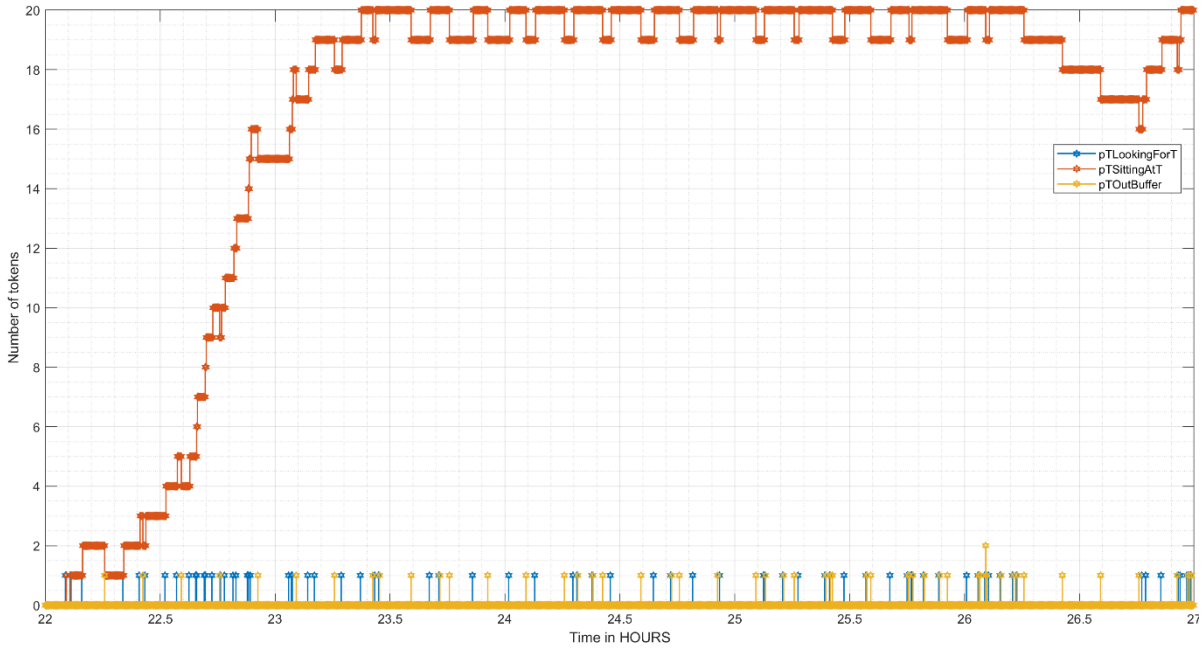

Figure 19: Case 1 Number of Customer Tokens Tables

Figure 19 shows that similarly the tables reach their maximum capacity at around 23:30. Customers looking for a table either sit down, if there is a free place, or leave the module through the out buffer, if the tables are full.

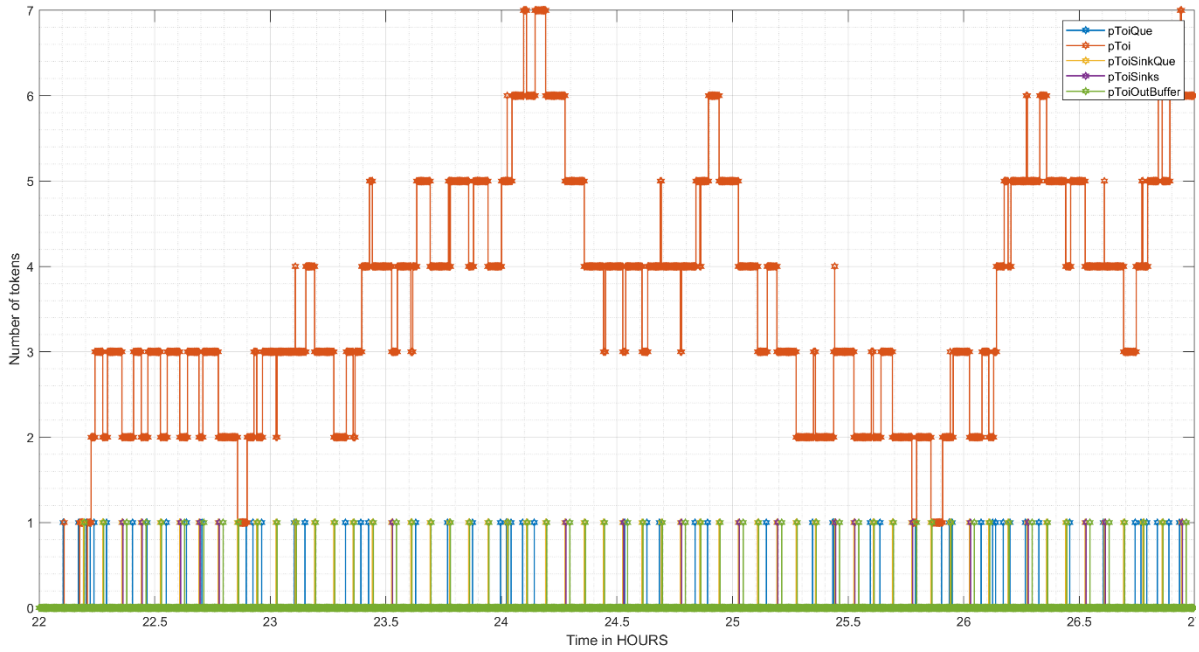

Figure 20: Case 1 Number of Customer Tokens Toilet

Figure 20 shows that some customers are accumulating on the toilets between 23:30 and 01:00 and again after 02:00, but the toilets never reach their capacity limit of 8 customers. Therefore, there is no queue building up in front of the toilets. There is also no queue building up in front of the sinks and never more than one sink in use, as the process time is shorter at the sinks than at the toilet.

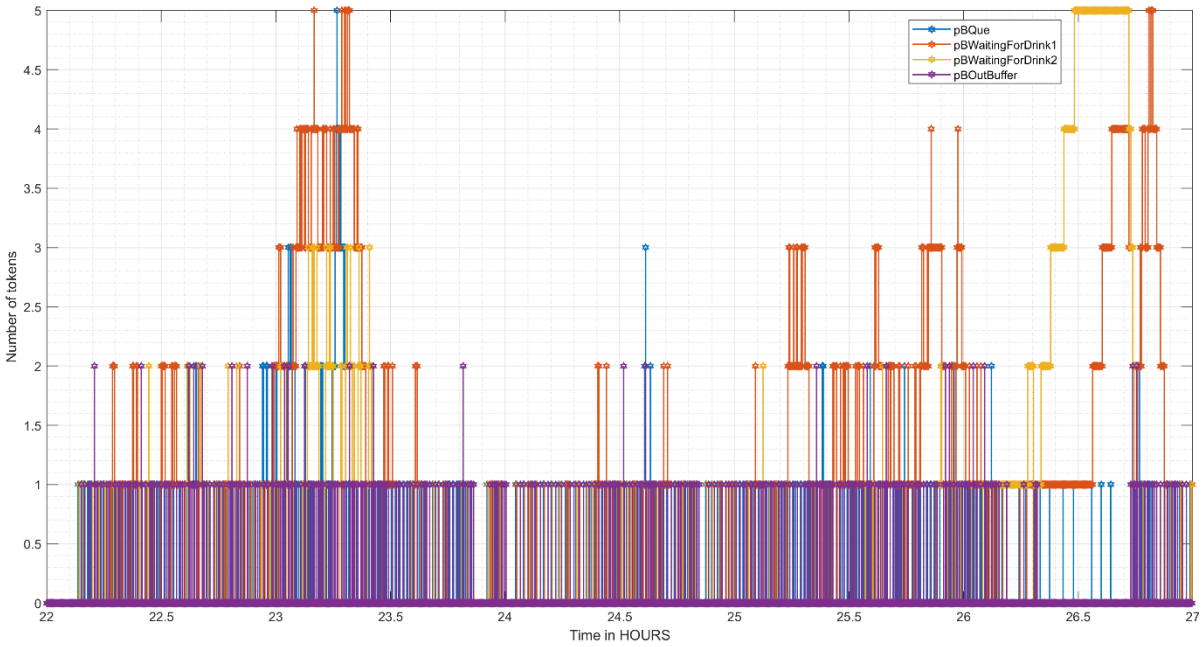

Figure 21: Case 1 Number of Customer Tokens Bar

Figure 21 shows that in the time between 23:00 and 23:30 customers are accumulating in all three queues, waiting to order and receive their respective drinks. The queue for drink1 reaches its maximum capacity of 5 people a few times before the bartenders catch up with their workload, reducing the number of customers waiting again. Towards the end of the night, at around 02:30 (26.5 in the figure), customers have accumulated waiting for drink2, having to wait 15-30 minutes before receiving their drink. The queue has reached its capacity, so customers are diverting to ordering drink1 instead, leading to a built up of the queue for drink1 shortly after.

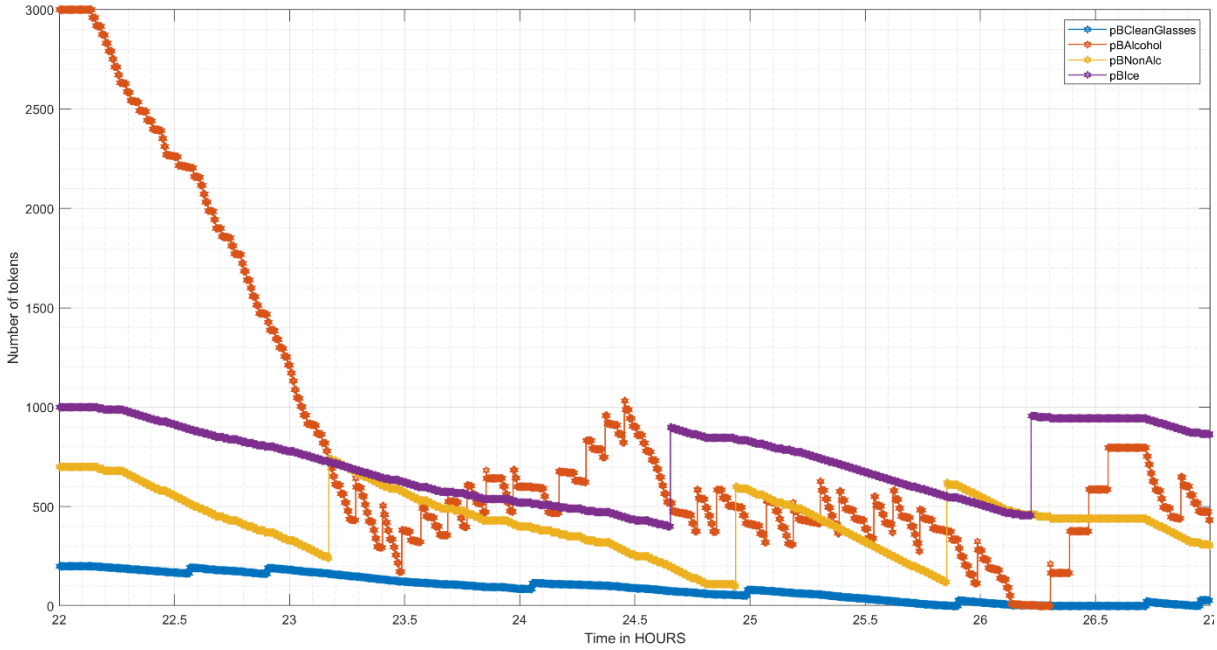

Figure 22: Case 1 Number of Ingredient Tokens Bar

Figure 22 shows that the ingredients available at the bar decrease as customers are receiving their drinks until the ingredients get refilled from the storage by the barback. The highest consumption rate is in the alcohol, as it is used in both drink1 and drink2, unlike Non-alcoholic ingredients and ice, which are only used in drink1. This combined with the relatively low carrying capacity when refilling, because alcohol is usually stored in heavy glass bottles, leads to it having to be refilled a lot more than the other ingredients. During the time when the barback is refilling alcohol they cannot carry out other duties such as collecting empty glasses, which leads to a shortage of glasses towards the end of the night at around 02:15 (26.15 in the figure), which is probably the cause for customers having to wait for their drink, as discussed in regard to Figure 21. This problem could be solved by increasing the carrying capacity when resupplying the alcohol for example using a sack barrow or similar devices, or by employing another barback.

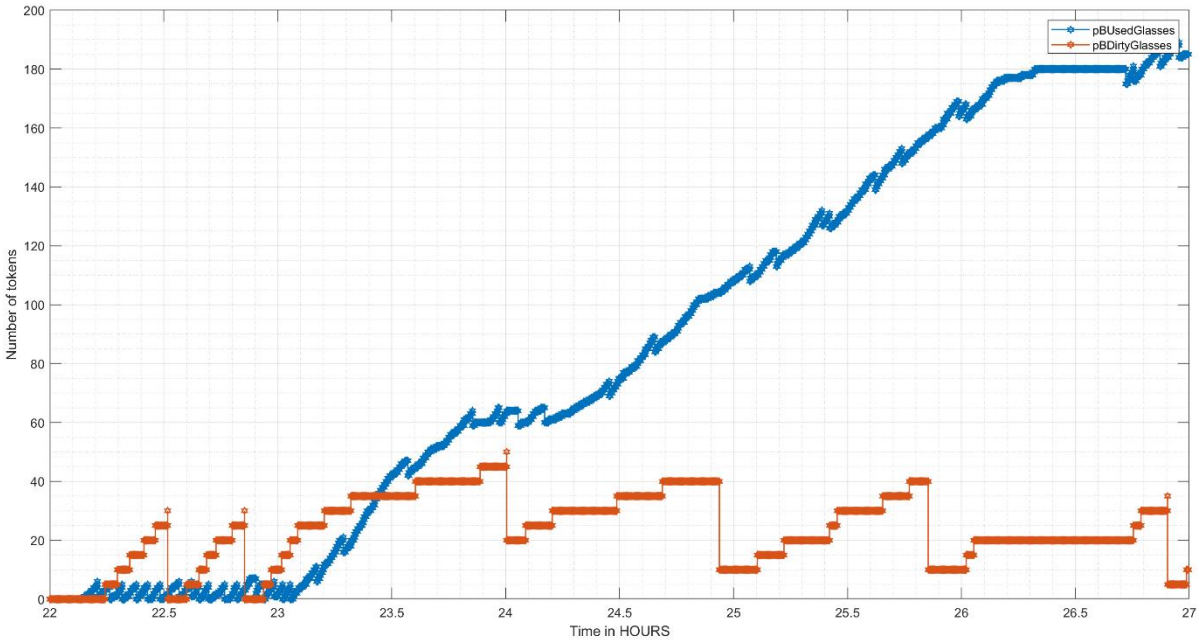

Figure 23: Case 1 Number of Glasses Tokens

Figure 23 shows that in the beginning of the night the barback is keeping up with collecting used glasses and putting the dirty glasses in the dishwasher. Once at least 30 glasses have accumulated in the dishwasher, as for example at 22:30, it is run and the glasses are available at the bar again shortly after, which can be seen in Figure 22. Figure 23 also shows that towards the end of the night the used glasses accumulate throughout the bar and the barback is unable to keep up with filling up and running the dishwasher, as they are busy with resupplying other ingredients, as discussed in regard to Figure 22.

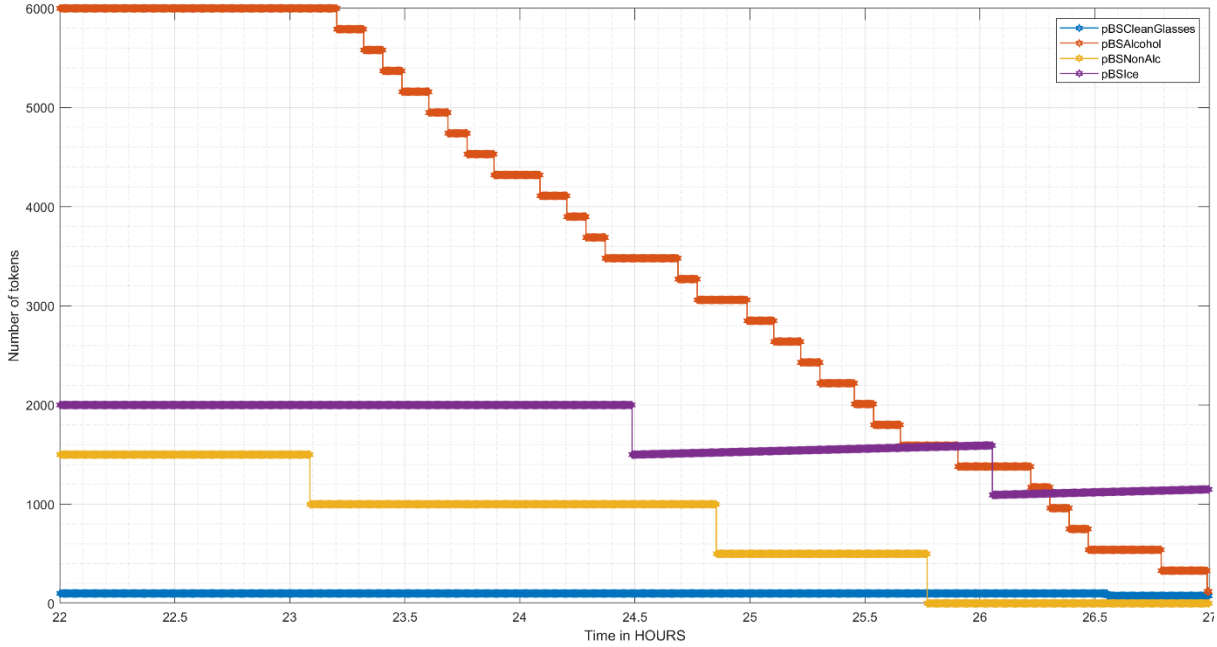

Figure 24: Case 1 Number of Ingredient Tokens Storage

Figure 24 shows the decrease in available resources at the storage room as they are resupplied to the bar throughout the night. It is also visible that as soon as ice is refilled for the first time at 00:30 (24.5 in the figure), freeing up space in the freezer, the ice machine starts generating new ice. Alcoholic as well as non-alcoholic ingredients are running out at the end of the night and need to be resupplied for the next night. Furthermore, we can see, that at around 02:30 (26.5 in the figure) the glasses are being resupplied from the storage room instead of the dishwasher, as the later was not full enough to run at that time, as can be seen in Figure 23.

Figure 25 shows the Gantt chart for resource scheduling. Unfortunately, the time axis is in seconds rather than hours and there is no legend to aid in understanding at which transition a resource was used. However, the latter can be inferred from the insights created by the previous charts and the printout from the `prnschedule` function.

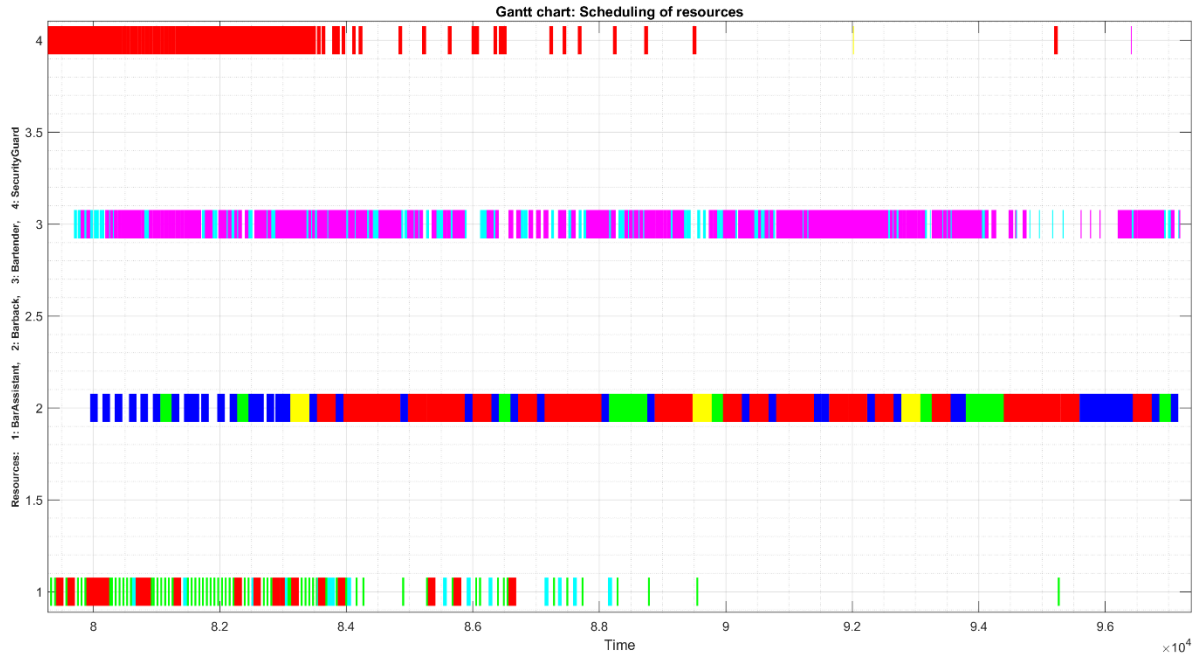

Figure 25: Case 1 Resource Scheduling Gantt Chart

The security guards are primarily occupied with the security check at the entrance (red), which mainly takes place at the beginning of the night, before the capacity of the bar is reached. The random security checks inside the bar (yellow and pink) only happen a few times during the night.

The bartenders are busy almost throughout the whole night taking orders and preparing drinks (light blue and pink), with exception to the time period towards the end of the night where they are waiting on fresh glasses, as discussed before.

The barback is also busy almost throughout the whole night with collecting glasses (blue), running the dishwasher (green), resupplying non-alcoholic ingredients (yellow), alcoholic ingredients (red), ice (not visible) and glasses (not visible).

In the beginning of the night the barback is less busy, as there are still enough ingredients at the bar and not a lot of glasses used to collect.

The bar assistants are primarily busy with collecting entrance fees (green) and receiving jackets (red) in the beginning or the night, as well as returning some jackets to customers later in the night (light blue).

The printout from `prnschedule` shows, that security guards 3 and 4 are rarely used, compared to security guards 1 and 2, which are assumably busy at the entrance:

```
Resource Instance: (SecurityGuard-1):: Used 95 times.  Utilization time: 5520
Resource Instance: (SecurityGuard-2):: Used 95 times.  Utilization time: 5520
Resource Instance: (SecurityGuard-3):: Used 5 times.  Utilization time: 165
Resource Instance: (SecurityGuard-4):: Used 5 times.  Utilization time: 165
```

In contrast to that bartender 1 and 2 are used more equally:

```
Resource Instance: (Bartender-1):: Used 439 times.  Utilization time: 13695
Resource Instance: (Bartender-2):: Used 313 times.  Utilization time: 8880
```

Bar assistant 1 is almost used three times as much as bar assistant 2:

```
Resource Instance: (BarAssistant-1):: Used 96 times.  Utilization time: 4860
Resource Instance: (BarAssistant-2):: Used 33 times.  Utilization time: 1170
```

Using the function `timesfired` we know that in this night 93 customers paid the entrance fee and 18 paid the wardrobe fee, as well as that 189 drinks of type drink1 are sold and 186 drinks of type drink2.
Using this information, the total costs and incomes of the night are calculated, resulting in an overall balance of 25878NOK.

4.3 Parameters used for Testing

The parameters used are based on insight provided by the experience of a bar manager. However, they do not represent a specific existing bar but rather represents a generic bar. Using this generic bar design, different cases are setup reflecting different scenarios to examine the effect of changing different parameters. Table 1-4 show the parameters used for the different cases.

Table 1: Customer Arrivals and Employee Numbers

Case #	TimeBetweenCustArrivals	nBarAssistant	nBarback	nBartender	nSecurityGuards
1	[0 1 0]	2	1	2	4
2	[0 10 0]	2	1	2	4
3	[0 10 0]	1	1	1	2
4	[0 10 0]	1	1	1	2
5	[0 10 0]	1	1	1	2
6	[0 3 0]	1	1	1	2
7	[0 1 0]	2	1	2	4

Table 2: Prices and Capacities

Case #	EntranceFee	WardrobeFee	PriceDrink1	PriceDrink2	TotalCap	DfCap	TCap
1	150	25	125	98	50	20	20
2	150	25	125	98	50	20	20
3	150	25	125	98	50	20	20
4	150	25	150	125	50	20	20
5	150	25	100	75	50	20	20
6	0	25	150	125	50	20	20
7	150	25	125	98	120	50	50

Table 3: Fix Costs

Case #	Electricity	Water	Rent	Insurance	Maintainance
1	2000	350	5000	500	750
2	2000	350	5000	500	750
3	2000	350	5000	500	750
4	2000	350	5000	500	750
5	2000	350	5000	500	750
6	2000	350	5000	500	750
7	4000	500	10000	1000	1500

Table 4: Inhibitor Factors

Case #	BInhibFact	TInhibFact	DfInhibFact	ToiInhibFact	ExitInhibFact
1	1	3	3	3	3
2	1	3	3	3	3
3	1	3	3	3	3
4	1	2	2	2	2
5	1	4	4	4	4
6	1	3	3	3	3
7	1	3	3	3	3

Case 1 represents the base case from section 4.2 Example Run.

Case 2 represents a bad night for the bar owner with a low number of customers, comparable to a Wednesday night during COVID-19 restrictions. This is achieved by increasing the time between customer arrivals from 1min to 10min, meaning that a new customer arrives every 10min instead of every 1min.

Case 3 is based on case 2, but to compensate for the lower number of customers the bar manager decides to reduce the number of employees. Instead of employing 2 bar assistants, 2 bartenders and 4 security guards they only employ 1 bar assistant, 1 bartender and 2 security guards for the night.

Case 4 is based on Case 3, but to increase profit, the bar manager decides to increase the prices for the drinks. However, this also means that the customers are less likely to buy a drink at the bar. This is implemented by decreasing the inhibitor factors for all other parts of the bar and the exit.

Case 5 is the opposite of Case 4. To increase profit, the bar manager decides to decrease the prices for the drinks, making them more attractive for the customers. This is implemented by increasing the inhibitor factors for all other parts of the bar and the exit, increasing the probability of a customer buying a drink instead of going somewhere else.

Case 6 is also based on Case 3 but represents another strategy to increase profit. The bar manager decides to not collect any entrance fee to make their bar more attractive to customers, but at the same time increase the prices of the drinks. Since the customers did not have to pay any entrance fee they are willing to pay a higher price for the drinks, so the inhibitor factors stay the same as in Case 3. Since the bar is now attracting more customers, they arrive at an interval of 3min.

Case 7 is based on Case 1, assuming a good night to be the norm. Therefor the bar manager decides to expand their business and increase the capacities of their bar. This is done by increasing the respective capacity parameters. However, this expansion also means higher fix costs, as the new capacity needs more space, leading to higher rent, insurance, electricity, water and maintenance costs.

4.4 Results and Analysis

Since the behaviour of the customers is quite random, to reflect the variability from one night to another night, multiple runs must be made for a given case to get a statistically significant number of results.

Therefor a special version of the MSF has been created to run a given case multiple times and save the main results (Balance, number of drinks sold and number of entrance- and wardrobe-fees collected) to an excel file.

Each of the seven presented cases has been run at least 30 times.

In the following the balance of the night, number of drinks sold and fees collected are compared across the different cases and, if applicable, some interesting developments are pointed out in an example run of the case.

4.4.1 Case 1

The detailed developments of an example run for this base case are discussed in section 4.2 Example Run.

Figure 26 shows the histogram on the balance of 30 runs of case 1.

Each run of case 1 is completed in less than 3min.

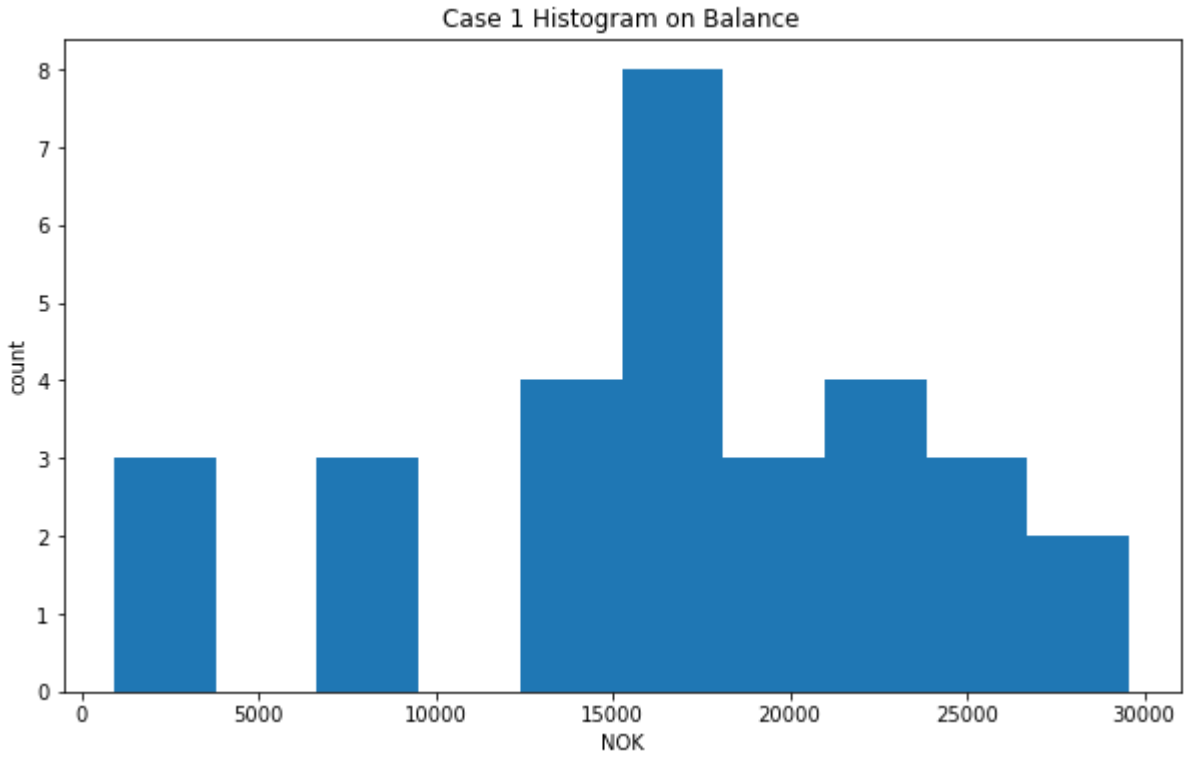

Figure 26: Case 1 Histogram on Balance

The mean balance is 16780.57NOK with a standard deviation of 7167.81NOK.

On average about 138 drinks of type drink1 and 138 drinks of type drink2 are sold and about 90 entrance fees and 20 wardrobe fees collected per night. This means that on average each customer orders about 3.1 drinks per night.

4.4.2 Case 2

Figure 27 shows the number of tokens in each place of the entrance module over time.

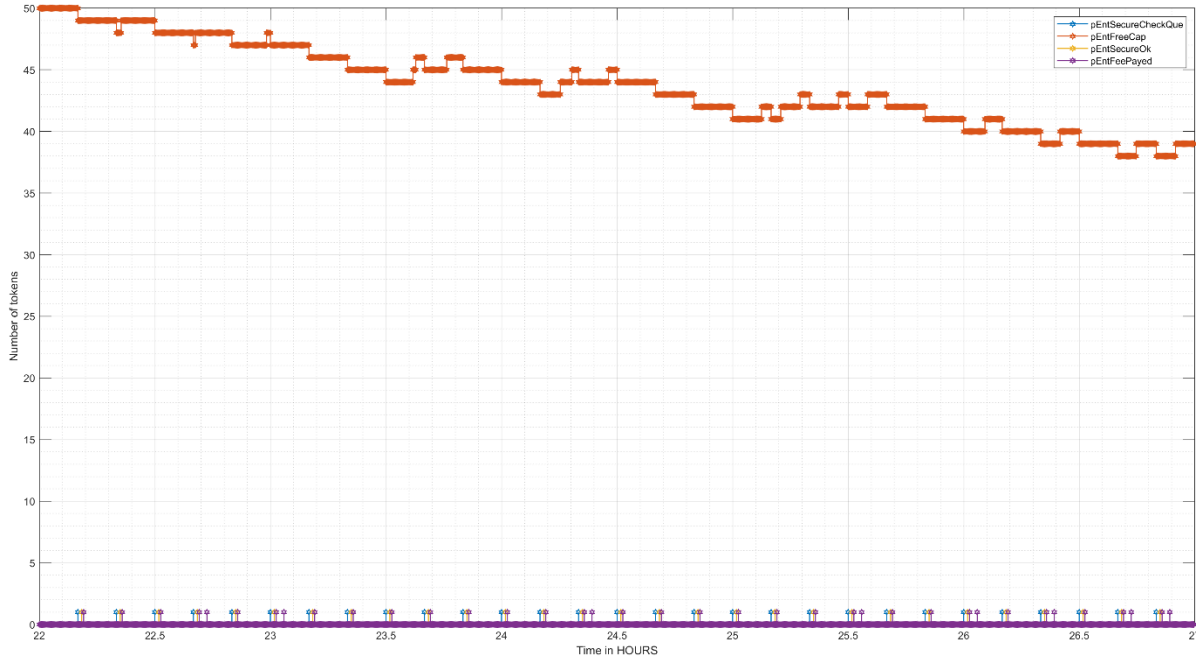

Figure 27: Case 2 Number of Tokens Entrance

With the higher time between arrivals of customers the capacity of the bar is never reached during the night and no queue is building up in front of the security check.

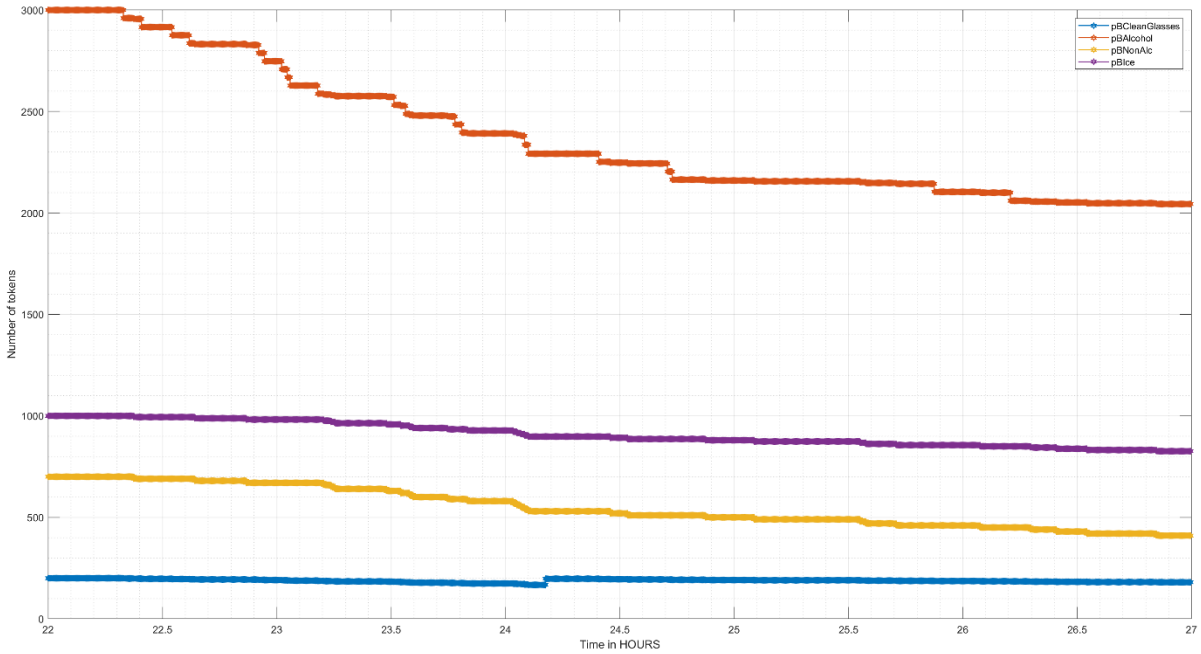

Figure 28: Case 2 Number of Ingredient Tokens Bar

Figure 28 shows, that not a lot of ingredients are used throughout the night and Figure 29 shows that the dishwasher is only run once.

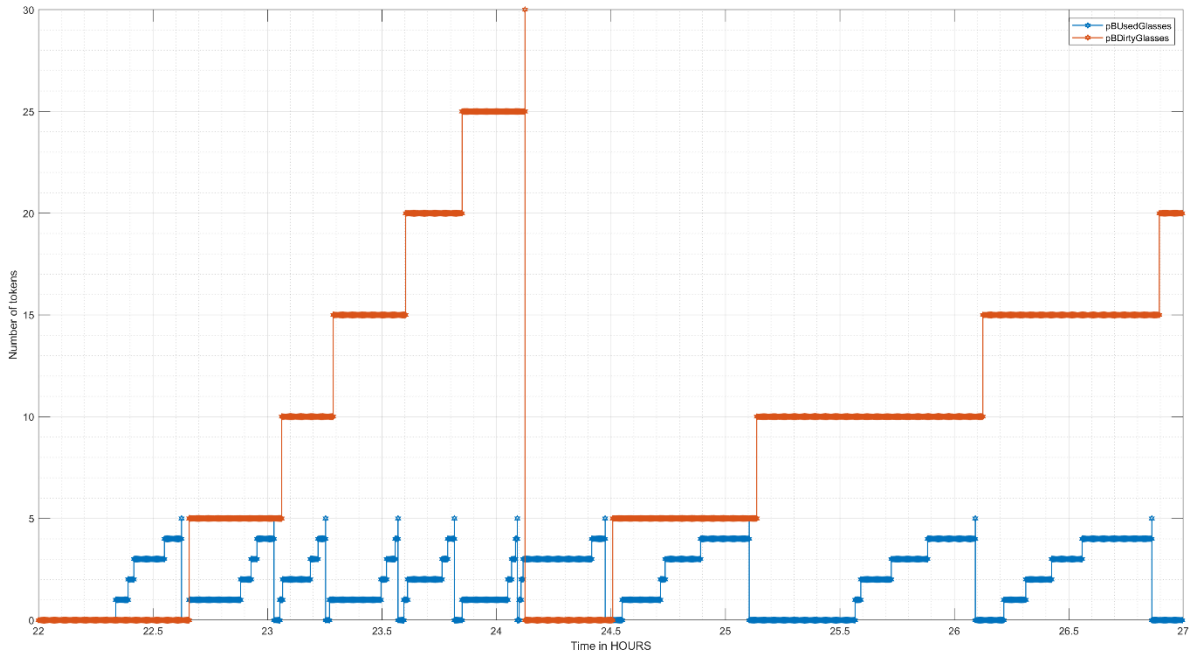

Figure 29: Case 2 Number of Glasses Tokens

Figure 30 shows that none of the employees are very busy throughout the night.

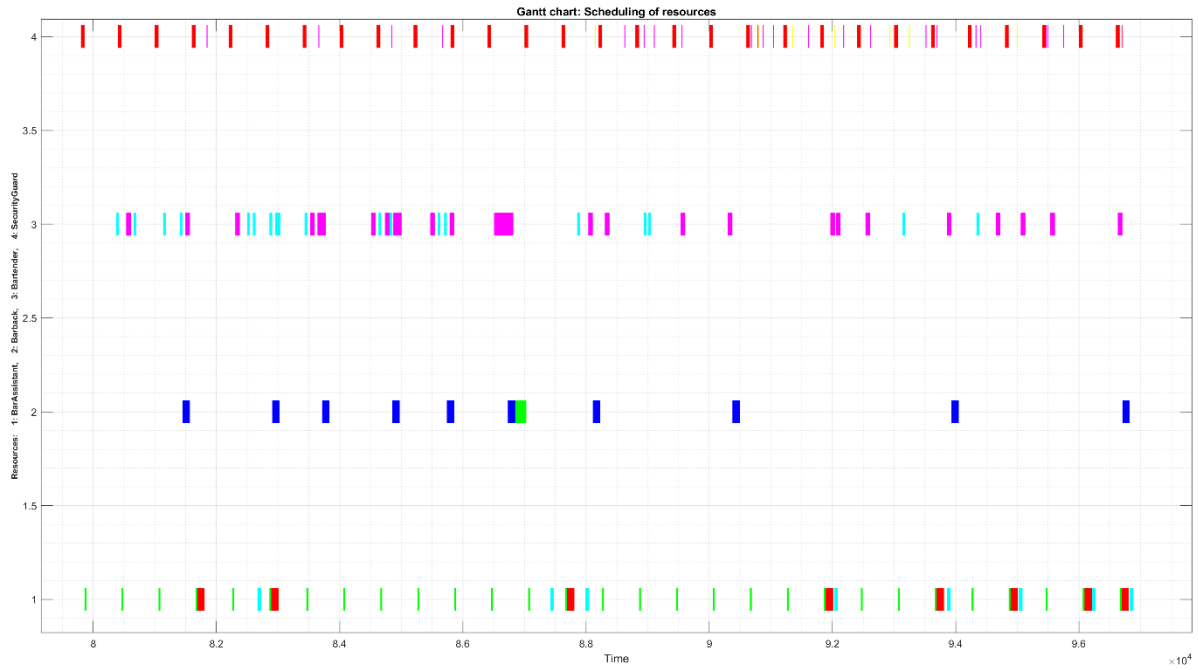

Figure 30: Case 2 Resource Scheduling Gantt Chart

Figure 31 shows the histogram on the balance of 100 runs of case 2.
Each run of case 2 is completed in less than 1min.

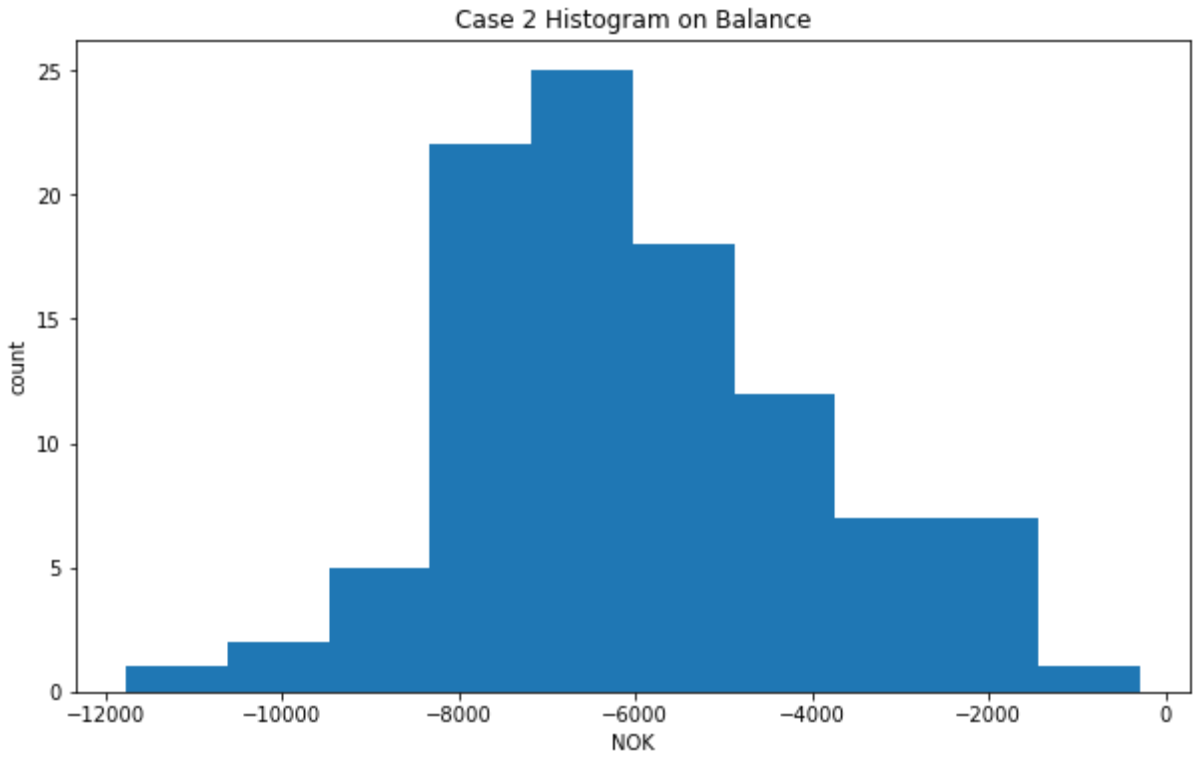

Figure 31: Case 2 Histogram on Balance

The mean balance is -5924.99NOK with a standard deviation of 2108.84NOK.
On average about 63 drinks of type drink1 and 59 drinks of type drink2 are sold and about 29 entrance fees and 5 wardrobe fees collected per night. This means that on average each customer orders about 4.2 drinks per night.
Note that since the number of customers is limited by the length of the night the number of entrance fees collected cannot exceed 29.
Case 2 is clearly less profitable than the base case, because of the lower customer numbers and resulting lower number of drinks sold. Figure 31 shows, that it is pretty much impossible to make profit in this case.

4.4.3 Case 3

Figure 32 shows the histogram on the balance of 30 runs of case 3.
Each run of case 3 is completed in less than 1min.

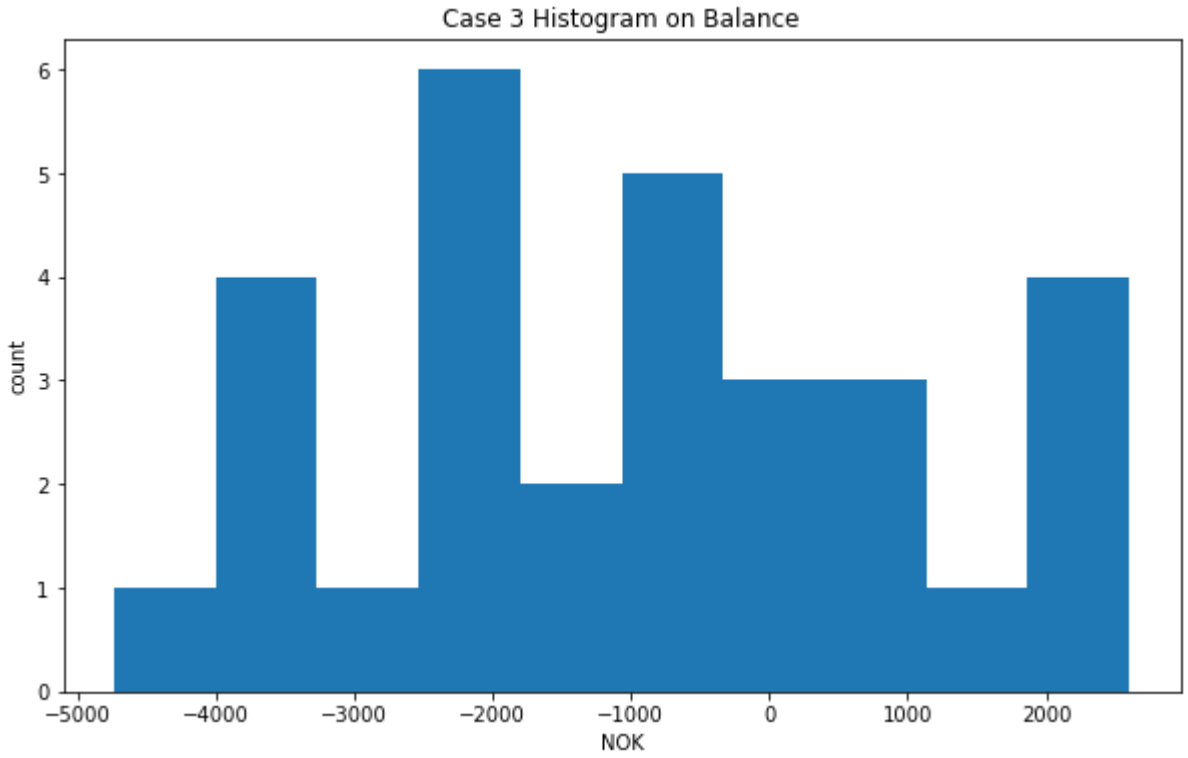

Figure 32: Case 3 Histogram on Balance

The mean balance is -909.57NOK with a standard deviation of 2053.34NOK.
On average about 65 drinks of type drink1 and 62 drinks of type drink2 are sold and about 29 entrance fees and 6 wardrobe fees collected per night. This means that on average each customer orders about 4.4 drinks per night.
Since the number of customers is still limited by the length of the night, the numbers of sold drinks and collected fees does not change much. However, through reducing the cost for employees the overall balance can be increased, even making it possible to make profit from time to time, as can be seen from Figure 32, but on average the balance is still negative.

4.4.4 Case 4

Figure 33 shows the histogram on the balance of 30 runs of case 4.

Each run of case 4 is completed in less than 1min.

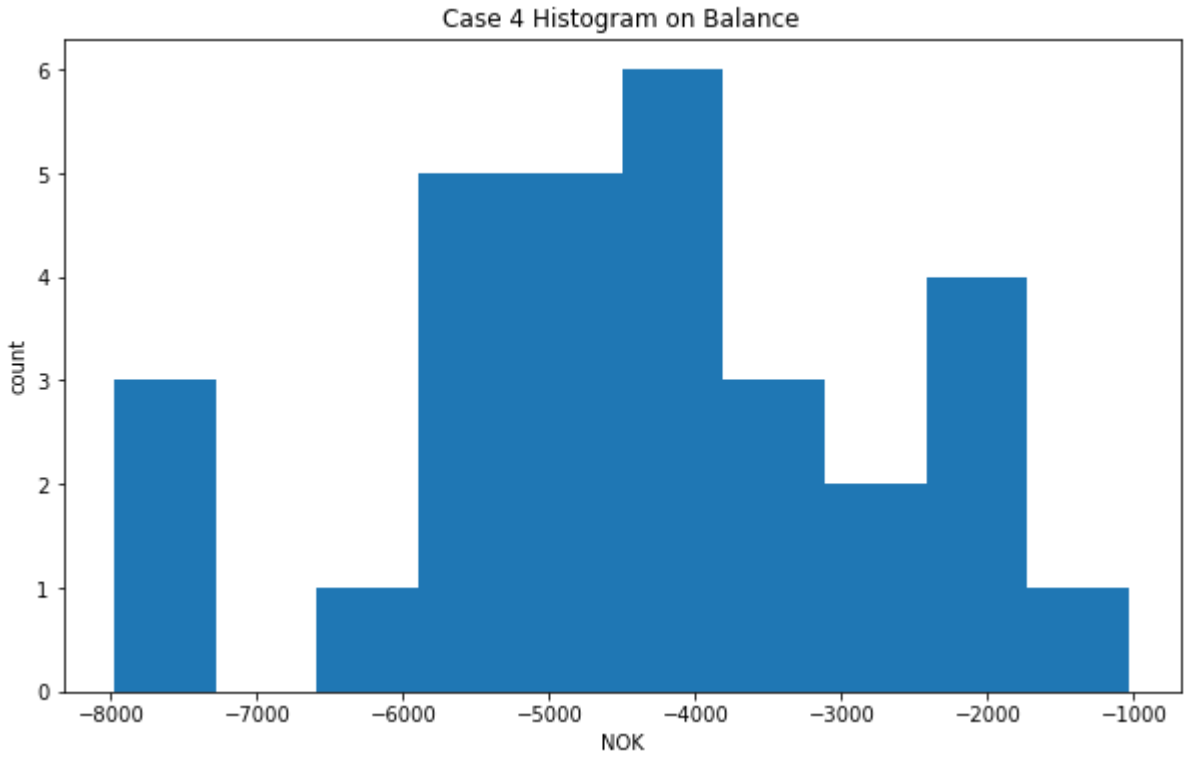

Figure 33: Case 4 Histogram on Balance

The mean balance is -4464.83NOK with a standard deviation of 1673.16NOK.

On average about 34 drinks of type drink1 and 32 drinks of type drink2 are sold and about 29 entrance fees and 6 wardrobe fees collected per night. This means that on average each customer orders about 2.3 drinks per night.

Because the higher drink prices make the drinks less attractive to the customers, less drinks are sold and therefor the average balance is worse than in case 3. This strategy is clearly not profitable.

4.4.5 Case 5

Figure 34 shows the histogram on the balance of 50 runs of case 5.
Each run of case 5 is completed in less than 1min.

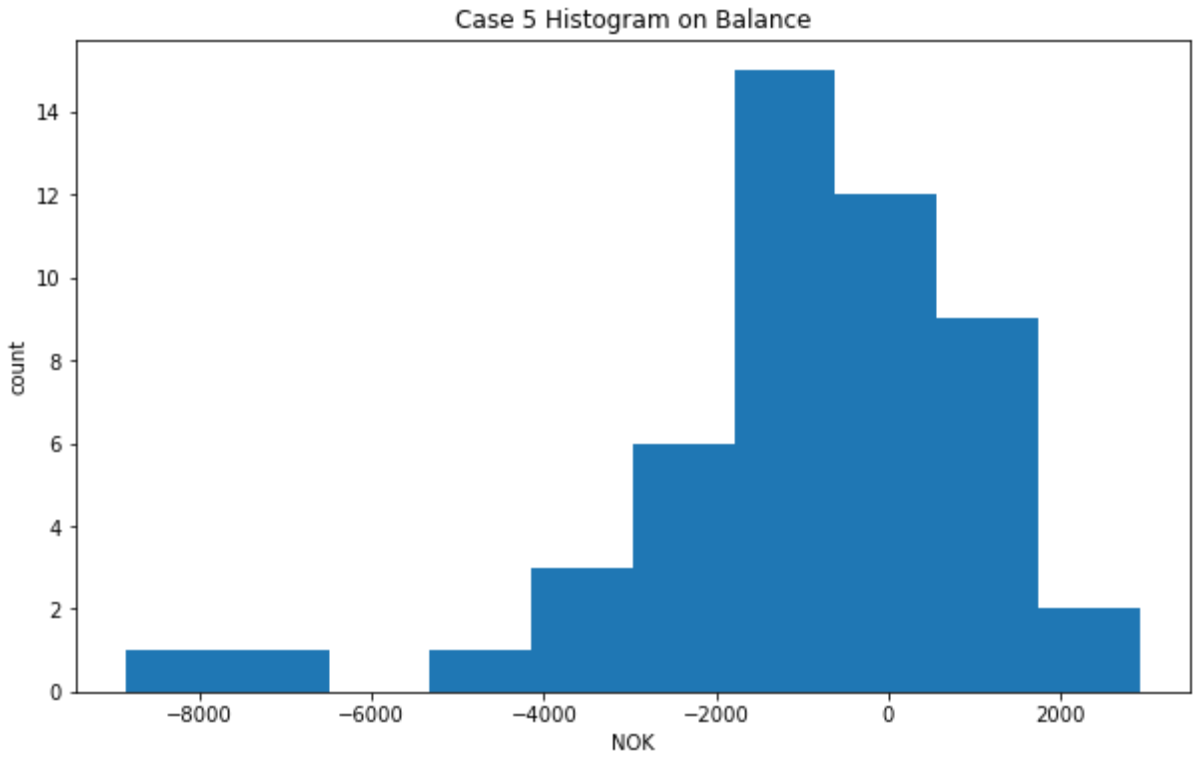

Figure 34: Case 5 Histogram on Balance

The mean balance is -960NOK with a standard deviation of 2066.4NOK.
On average about 89 drinks of type drink1 and 85 drinks of type drink2 are sold and about 29 entrance fees and 6 wardrobe fees collected per night. This means that on average each customer orders about 6 drinks per night.
Because the lower drink prices make the drinks more attractive to the customers, more drinks are sold compared to case 3. The average balance is only slightly worse than in case 3 and the histogram is slightly skewed towards the positive, as can be seen in Figure 34.
This strategy is more profitable than the strategy from case 4 and shows that the drink prices must be carefully balanced to increase the win margin per drink while keeping the drinks attractive to the customers.

4.4.6 Case 6

Figure 35 shows the number of tokens in each place of the entrance module over time.

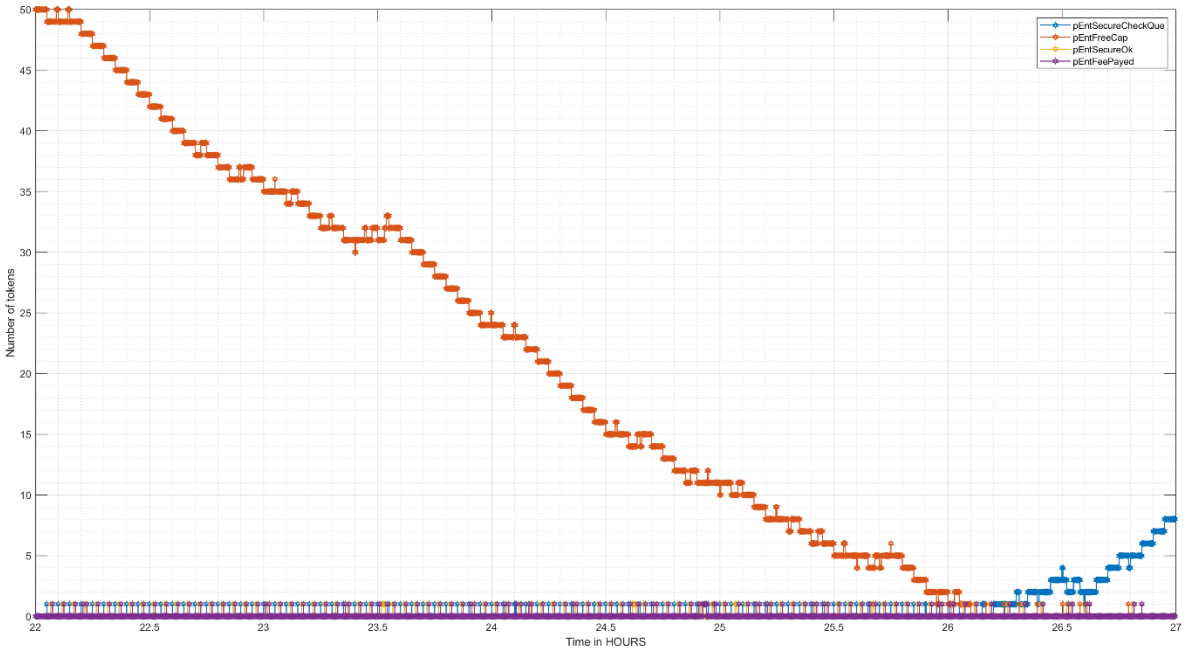

Figure 35: Case 6 Number of Tokens Entrance

Because of the free entrance the bar is more attractive to customers and they arrive in shorter intervals. This leads to the bar reaching its capacity again at around 02:00. Once the capacity is reached a queue starts building in front of the security check.

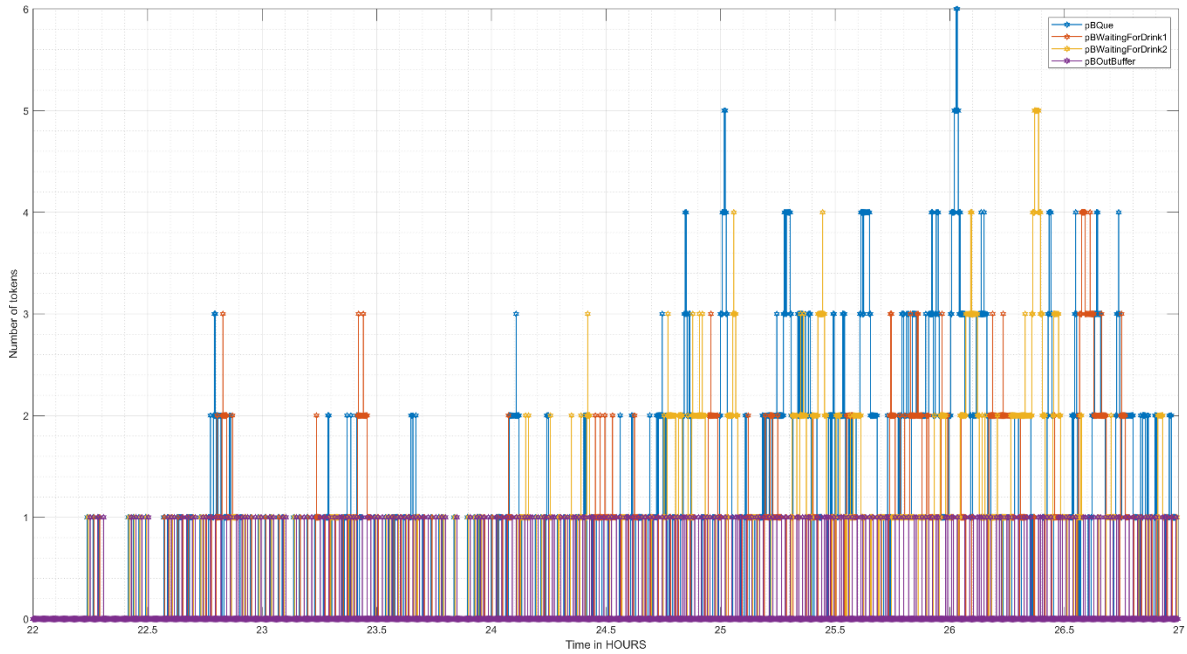

Figure 36: Case 6 Number of Customer Tokens Bar

Figure 36 shows that the single bartender is starting to have difficulties keeping up with the demand for drinks in the later half of the night in this case, leading to queues building for customers waiting to order and receive their drink.

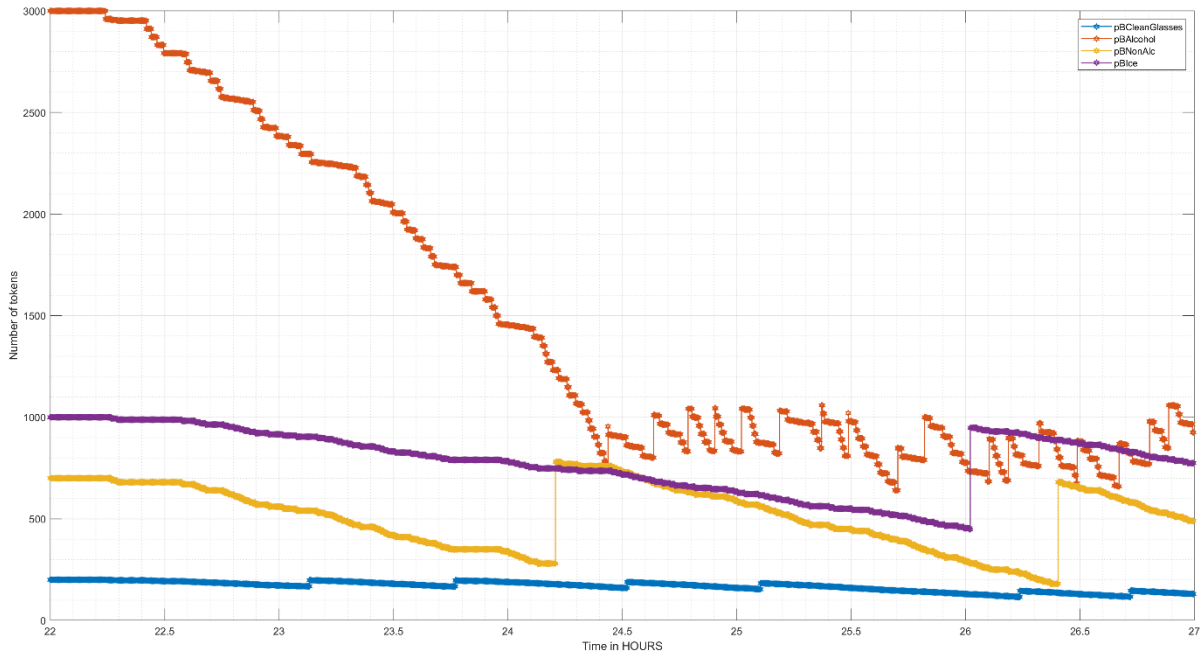

Figure 37: Case 6 Number of Ingredient Tokens Bar

Figure 37 shows that ingredients are being used up at a faster pace again in this case. This leads to the barback having to resupply them from the storage in the later half of the night.

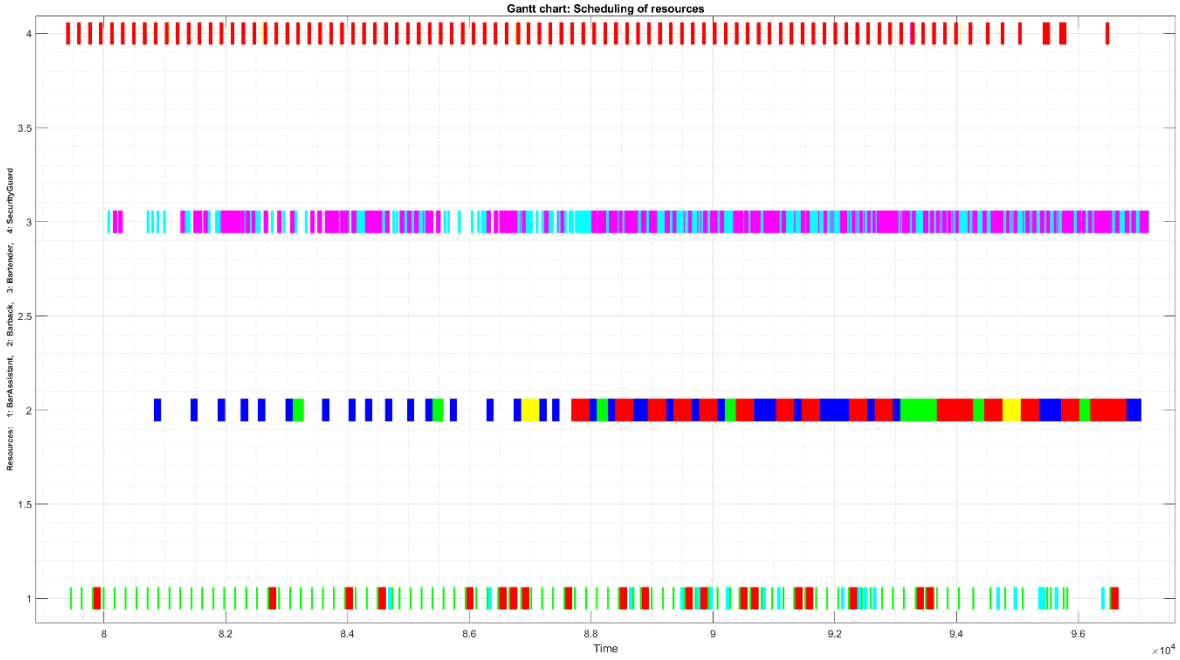

Figure 38: Case 6 Resource Scheduling Gantt Chart

Figure 38 shows that the bartender and barback are busier in this case, compared to cases 2-5, especially in the later half of the night. The security guards and the bar assistants are also busier because of more customers arriving at the entrance.

Figure 39 shows the histogram on the balance of 34 runs of case 6.
Each run of case 6 is completed in less than 2min.

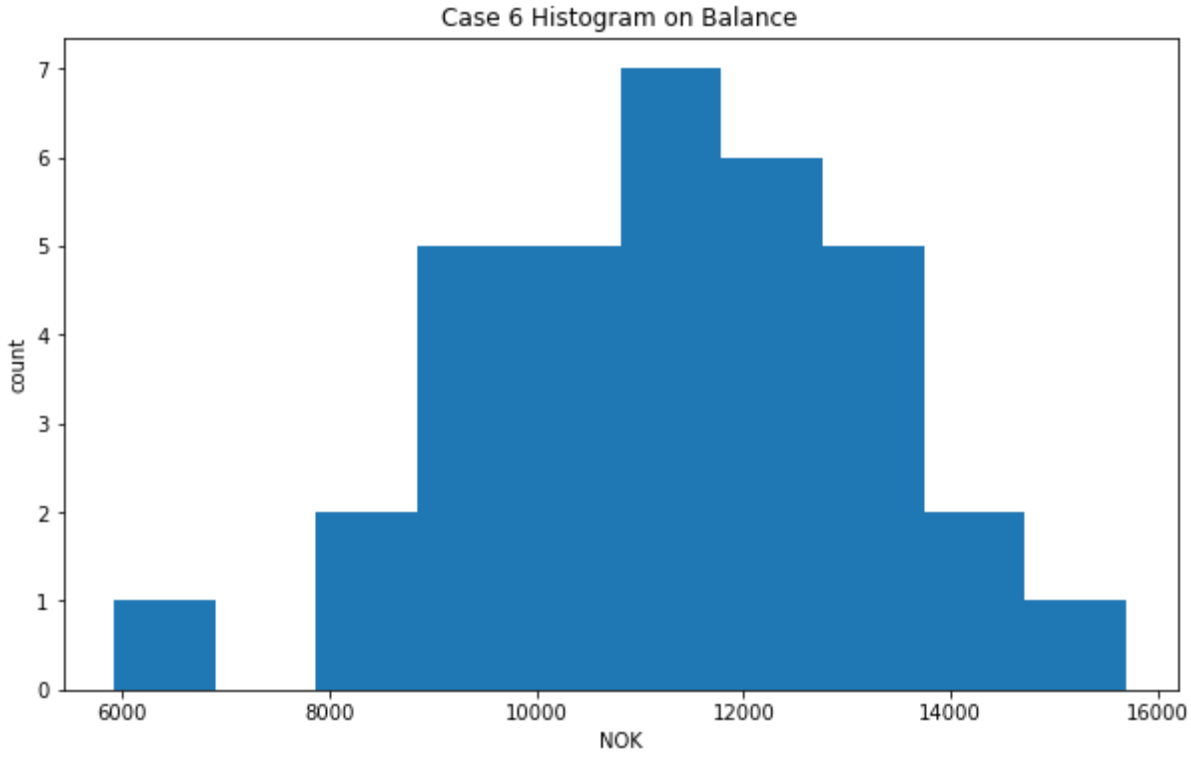

Figure 39: Case 6 Histogram on Balance

The mean balance is 11251.62NOK with a standard deviation of 1974.87NOK.
On average about 123 drinks of type drink1 and 119 drinks of type drink2 are sold and about 86 entrance fees and 17 wardrobe fees collected per night. This means that on average each customer orders about 2.8 drinks per night.
The bar can make consistent profit again, even without the income from the entrance fee, which can be seen in Figure 39.
This shows that the combined business strategies of this case are a viable option to turn an unprofitable night into a profitable night.
The comparison of the previous cases shows that a lot of factors, which are not necessarily independent of each other, influence the result and need to be balanced carefully. It also shows that the focus should be on cutting costs while also making the bar more attractive to customers to increase the customer numbers again. An increase in prices without justification will not help in increasing profitability.

4.4.7 Case 7

Figure 40 shows the histogram on the balance of 30 runs of case 7.
Each run of case 7 is completed in less than 6min.

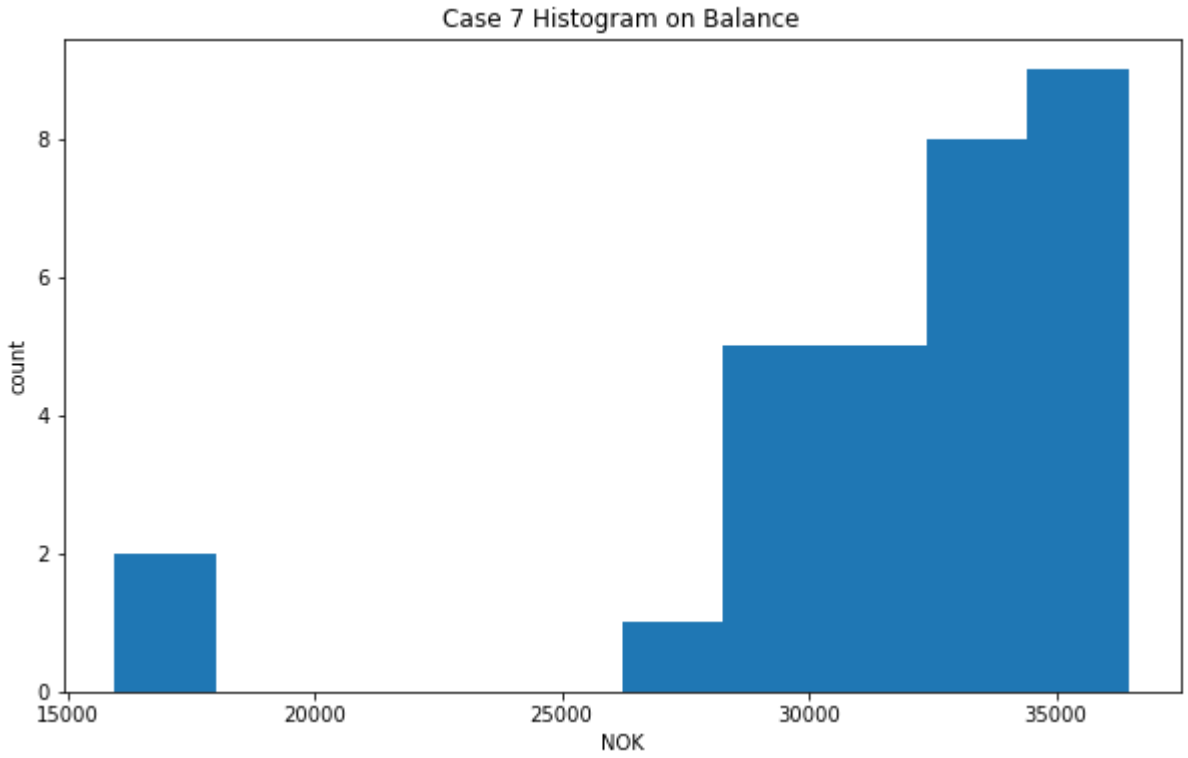

Figure 40: Case 7 Histogram on Balance

The mean balance is 31698.37NOK with a standard deviation of 4776.07NOK.
On average about 177 drinks of type drink1 and 175 drinks of type drink2 are sold and about 197 entrance fees and 38 wardrobe fees collected per night. This means that on average each customer orders about 1.8 drinks per night.
With the increased capacity the bar can make higher profit in a good night, compared to case 1, even with the higher fix costs. However, it should be noted that the risk of not making profit on a bad night is higher. Similar to case 5, Figure 40 shows a histogram that is skewed towards the positive with a few negative outliers. This could indicate to some inhibiting factor preventing further increase of the profit and a bottle-neck factor that could turn the result of the night from good to bad, under the right circumstances. Both factors could be an employee working at their absolute maximum capacity or the bar completely running out of an ingredient.

5. Discussion

The created model is discussed in the following

5.1 Originality of this Work

As demonstrated in section 4 Testing, Analysis and Results, the biggest strength of our model is its flexibility, allowing to represent different scenarios by adjusting a few parameters. Since every parameter of the model is adjustable the number of possible scenarios is virtually infinite. Our model reacts to parameter changes as expected. This allows to get very detailed and useful insights to the things happening at a bar during the night, possibly revealing unexpected dependencies between parameters and results. These insights can be used to find the best business strategies to adapt to a new situation.

The modular approach keeps the model easy to understand and allows it to be expanded with more or completely new modules.

5.2 Relevant Works

During our research we were not able to find a similar model that meets our expectations and could help solve our problem. Different service centers like hospitals [8], Vaccination centers [9] and restaurants [10] have been modelled in the past, but none are applicable to a bar or flexible enough to represent one.

5.3 Limitations of this Work

While we are satisfied with the results of our work, there is still room for improvement. The timing of the transitions is fixed, as GPenSIM does not natively support variable timing. This prevents our model from being more accurate, as real-world timings are seldom constant, especially if humans are involved. Workarounds are possible but increase the complexity or decrease the performance of the model.

While the model is already very modular, it is currently not possible to implement two bar modules which access the same storage room, as the storage room is integrated into the bar module.

The number of possible drink types and their ingredients is currently limited to two types of drinks and one generic alcoholic ingredient, one generic non-alcoholic ingredient, ice and the glass. This limits the flexibility of the model and its ability to accurately represent more specific scenarios. Related to this there are currently no parameters controlling the probability of a customer buying one type of drink over another.

Some of the implemented bypasses may have become redundant with the later introduction of the inhibitor arcs and might hinder the performance of the model, as their pre-processing must be run every time they are enabled.

It is currently not possible to parallel process customers in the toilets, dancefloor or tables, which prevents our model from being more accurate to the real-world situation.

The customer behaviour is quite random and not controlled by customer specific variables, such as the remaining money they are willing to spend during the night or their remaining energy levels. It was investigated to implement this functionality using colors. The limitation of GPenSIM not natively allowing to remove a specific color from a token would lead to the use of workarounds. Implementing these was deemed to increase the complexity of the model too much.

It is currently not possible to track which jacket in the wardrobe belongs to which customer. While this is not necessary for the model to work properly, it would be nice to have.

5.4 Future Work

As mentioned before, there is room for improvement of our model. A few approaches to improve our model in future work are mentioned below.

A satisfying solution for variable timing, which does not compromise performance or increase complexity needs to be found and implemented.

The storage room needs to be made into its own module and an IMC between the bar modules and the storage module needs to be implemented to allow multiple bar modules accessing the same storage room.

More types of alcoholic and non-alcoholic ingredients as well as different types of glasses could be implemented. This would allow the implementation of more types of drinks. A parameter controlling the probability of a customer buying one type of drink over another should also be implemented. This would increase the flexibility of the model and allow it to be more accurate for more specific scenarios

A simple method to control customer behaviour more individually needs to be found and implemented.

A satisfying solution to track which jacket in the wardrobe belongs to which customer could be implemented.

5.5 Learning Experience

While some of the previously mentioned limitations of GPenSIM, such as the missing native support for variable timing, the ability to remove specific colors or release specific resources, have caused frustration during the work on our model, the general learning curve was quite pleasant and the satisfaction of seeing our model work as expected shadows these minor inconveniences.

References

[1] NEWSinENGLISH.no, “New omicron variant spreads,” NEWSinENGLISH.no, 07 November 2022. [Online]. Available: https://www.newsinenglish.no/2022/11/07/updates-here-as-corona-rages-on/. [Accessed 09 November 2022].

[2] Institute for Health Metrics and Evaluation, “COVID-19 Projections (Norway),” 21 October 2022. [Online]. Available: https://covid19.healthdata.org/norway?view=infections-testing&tab=trend&test=infections. [Accessed 09 November 2022].

[3] F. Norwell, “Norway further tightens Covid rules with nationwide alcohol ban in bars,” THE LOCAL, 13 December 2021. [Online]. Available: https://www.thelocal.no/20211213/what-covid-restrictions-could-norways-government-announce-on-monday. [Accessed 09 November 2022].

[4] F. Norwell, “Norway lifts alcohol ban as Covid rules eased,” THE LOCAL, 13 January 2022. [Online]. Available: https://www.thelocal.no/20220113/what-covid-19-rule-changes-could-norways-government-announce-on-thursday. [Accessed 11 November 2022].

[5] Statistisk sentralbyrå, “Consumer price index,” 10 October 2022. [Online]. Available: https://www.ssb.no/en/priser-og-prisindekser/konsumpriser/statistikk/konsumprisindeksen. [Accessed 09 November 2022].

[6] SPANISH NEWS TODAY, “Drinking Spain dry: bar owners fear a winter beer shortage,” 08 November 2022. [Online]. Available: https://spanishnewstoday.com/drinking_spain_dry_bar_owners_fear_a_winter_beer_shortage_1830924-a.html. [Accessed 09 November 2022].

[7] R. Davidrajuh, Modelling Discrete-Event Systems with GPenSIM: An Introduction, Stavanger: Springer, 2017.

[8] J. Melstveit and H. Teppan, “Modelling Stavanger Universitetssykehus’ (SUS) Capacity During SARS-CoV-2,” Universitetet i Stavanger, Stavanger, 2020.

[9] M. Z. Guniem, “Simulation of Mass Vaccination Programs using GPenSIM in Matlab,” Universitetet i Stavanger, Stavanger, 2021.

[10] Z. S. Hussein, “Simulation of Food Restaurant Using Colored Petri Nets,” *Journal of Engineering and Development,* vol. 18, no. 4, pp. 77-88, 2014.

[11] The Norwegian Water Resources and Energy Directorate, “Map Services,” Norwegian Government Agency, [Online]. Available: https://www.nve.no/map-services/. [Accessed 29 May 2022].

[12] Meteoblue, “Simulated historical climate & weather data for Norway,” [Online]. Available: https://www.meteoblue.com/en/weather/historyclimate/climatemodelled/norway_united-states_5004016. [Accessed 29 May 2022].

Appendix

A1: Complete Code for Different Cases (.zip file)

BarV2.m (Case 1)

```
% BarV2

%%%Abrriviations:%%%
%B: Bar
%Cap: Capacity
%Cust: Customer
%Df: Dancefloor
%Ent: Entrance
%Gen: Generate
%IMC: Inter-Modular-Connector
%n: number of [...]
%NonAlc: Non-Alcoholic
%R: Resupply
%S: Storage
%T: Tables
%Toi: Toilets
%Wd: Wardrobe

clear all; clc;
global global_info

%%%Simulation Parameters%%%
global_info.START_AT = [22 0 0]; %OPTION: start simulations
global_info.STOP_AT  = [27 0 0]; %OPTION: stop  simulations
global_info.DELTA_TIME = 15; %in seconds
caseID = 1 %to identify case for multirun

%%%Bar Design%%%
global_info.TotalCap = 50;
global_info.EntSecureCheckQueCap = 20;
global_info.WdCap = 25;
global_info.BQueCap = 10;
global_info.BWaitingForDrinkCap = 5;
global_info.DfCap = 20;
global_info.TCap = 20;
global_info.ToiQueCap = 10;
global_info.ToiCap = 8;
global_info.ToiSinksCap = 4;

global_info.BnOrderTerminals = 2;
global_info.BnCounters = 2;
if global_info.BnOrderTerminals == 2
    global_info.BnCounters = 2;
end

%%%Employee numbers%%%
nBarAssistant = 2;
nBarback = 1;
nBartender = 2;
nSecurityGuards = 4;

%%%Costs%%%
%%Variable costs (per unit sold)%%
Costs.Ingredients_Drink1 = 25;
Costs.Ingredients_Drink2 = 25;
```

```
%%Employees costs (per employee, per hour)%%
Costs.EmployeeBarAssistant = 185;
Costs.EmployeeBarback = 185;
Costs.EmployeeBartender = 185;
Costs.EmployeeSecurity = 200;
EmploymentHours = 6;

%%Fixed costs (per night)%%
Costs.Fixed.Artists = 300*(EmploymentHours-1);
Costs.Fixed.Electricity = 2000;
Costs.Fixed.Water = 350;
Costs.Fixed.Rent = 5000;
Costs.Fixed.MusicFees = 250;
Costs.Fixed.Insurance = 500;
Costs.Fixed.Maintainance = 750;
Costs.Fixed.ToiletPaper = 100;
Costs.Fixed.Soap = 100;
Costs.Fixed.PaperTowels = 100;

%%%Income%%%
EntranceFee = 150;
WardrobeFee = 25;
PriceDrink1 = 125;
PriceDrink2 = 98;

%%%Customer Behaviour%%%
TimeBetweenCustArrivals = [0 1 0];
%Inhibitor factors, the higher the number (integer), the less likely a
%customer will visit that part of the bar
global_info.BInhibFact = 1;
global_info.TInhibFact = 3;
global_info.DfInhibFact = 3;
global_info.ToiInhibFact = 3;
global_info.ExitInhibFact = 3;

%%%Security Behaviour%%%
%Inhibitor factors, the higher the number (integer), the less likely a
%security guard will perform that action
global_info.SecurityCheckInhibFact = 50;
global_info.SecurityThrowOutInhibFact = 5;

%%%Simulation Setup%%%
pns =
pnstruct({'BarV2_Entrance_pdf','BarV2_IMC_pdf','BarV2_Dancefloor_pdf','BarV2_Table
s_pdf','BarV2_Bar_pdf','BarV2_Toilet_pdf'});
dyn.m0 = {'pEntFreeCap',global_info.TotalCap,... %Customer movement
          'pBCleanGlasses',200,'pBAlcohol',3000,'pBNonAlc',700,'pBIce',1000,...
%Initial Bar supplies [glasses and centiliter]
          'pBSCleanGlasses',100,'pBSAlcohol',6000,'pBSNonAlc',1500,'pBSIce',2000};
%Initial Storage supplies [glasses and centiliter]

if global_info.BnOrderTerminals == 1
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink',[0 1 0],'tIMCWdExit',[0 1
```

```
0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer
movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink1',[0 1 0],'tBReceiveDrink2',[0 0
30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0
1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
elseif global_info.BnOrderTerminals == 2
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink',[0 1
0],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1
0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink1',[0 1
0],'tBReceiveDrink2',[0 0 30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2
0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
end

dyn.re =
{'BarAssistant',nBarAssistant,inf,'Barback',nBarback,inf,'Bartender',nBartender,in
f,'SecurityGuard',nSecurityGuards,inf}; %Staff

%%%Run Simulation%%%
pni = initialdynamics(pns, dyn);
results = gpensim(pni);

%%%Print Out results%%%
prnschedule(results)
% prncolormap(results,{'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'})
% prnss(results)

%%Count transition firerings%%
nEntIn = timesfired('tEntIn')
nEntGenCust = timesfired('tEntGenCust')
```

```
nEntSecureCheck = timesfired('tEntSecureCheck')
nEntFeePayed = timesfired('tEntFee')
nWdFeePayed = timesfired('tEntWd')
nEntOut = timesfired('tEntOut')
nBarIn = timesfired('tBIn')
if global_info.BnOrderTerminals == 1
    nDrinksSold = timesfired('tBReceiveDrink')
elseif global_info.BnOrderTerminals == 2
    nDrink1sSold = timesfired('tBReceiveDrink1')
    nDrink2sSold = timesfired('tBReceiveDrink2')
end
nToiIn = timesfired('tToiIn')
nTIn = timesfired('tTIn')
nDfIn = timesfired('tDfIn')
nSecurity1 = timesfired('tIMCSecurity1')
nSecurity2 = timesfired('tIMCSecurity2')
nExit = timesfired('tIMCExit')
nWdExit = timesfired('tIMCWdExit')
nIMC = timesfired('tIMCCustDist')

%%Calculate Costs%%
%Drinks%%
TotCost.Drink1 = nDrink1sSold * Costs.Ingredients_Drink1
TotCost.Drink2 = nDrink2sSold * Costs.Ingredients_Drink2

%Employees%
TotCost.Employees = EmploymentHours*(Costs.EmployeeBarAssistant*nBarAssistant +
Costs.EmployeeBarback*nBarback + Costs.EmployeeBartender*nBartender +
Costs.EmployeeSecurity*nSecurityGuards)

%Fix%
TotCost.Fix = Costs.Fixed.Artists + Costs.Fixed.Electricity + Costs.Fixed.Water +
Costs.Fixed.Rent + Costs.Fixed.MusicFees + Costs.Fixed.Insurance +
Costs.Fixed.Maintainance + Costs.Fixed.ToiletPaper + Costs.Fixed.Soap +
Costs.Fixed.PaperTowels

OverallCost = TotCost.Drink1 + TotCost.Drink2 + TotCost.Employees + TotCost.Fix

%%Calculate Income%%
Income.EntranceFee = EntranceFee*nEntFeePayed
Income.WardrobeFee = WardrobeFee*nWdFeePayed
Income.Drink1 = PriceDrink1*nDrink1sSold
Income.Drink2 = PriceDrink2*nDrink2sSold

TotIncome = Income.EntranceFee + Income.WardrobeFee + Income.Drink1 +
Income.Drink2

Balance = TotIncome - OverallCost

%%%Visualize results%%%
figure(1), plotp(results,
{'pEntSecureCheckQue','pEntFreeCap','pEntSecureOk','pEntFeePayed'});
figure(2), plotp(results, {'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'});
figure(3), plotp(results, {'pDfDancing'});
figure(4), plotp(results, {'pTLookingForT', 'pTSittingAtT', 'pTOutBuffer'});
if global_info.BnOrderTerminals == 1
        figure(5), plotp(results, {'pBQue', 'pBWaitingForDrink', 'pBOutBuffer'});
```

```
elseif global_info.BnOrderTerminals == 2
        figure(5), plotp(results, {'pBQue',
'pBWaitingForDrink1','pBWaitingForDrink2', 'pBOutBuffer'});
end
figure(6), plotp(results, {'pBUsedGlasses','pBDirtyGlasses'});
figure(7), plotp(results, {'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce'});
figure(8), plotp(results, {'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'});
figure(9), plotp(results, {'pToiQue', 'pToi',
'pToiSinkQue','pToiSinks','pToiOutBuffer'});
plotGC(results)
```

BarV2.m (Case 2)

```
% BarV2

%%%Abrriviations:%%%
%B: Bar
%Cap: Capacity
%Cust: Customer
%Df: Dancefloor
%Ent: Entrance
%Gen: Generate
%IMC: Inter-Modular-Connector
%n: number of [...]
%NonAlc: Non-Alcoholic
%R: Resupply
%S: Storage
%T: Tables
%Toi: Toilets
%Wd: Wardrobe

clear all; clc;
global global_info

%%%Simulation Parameters%%%
global_info.START_AT = [22 0 0]; %OPTION: start simulations
global_info.STOP_AT  = [27 0 0]; %OPTION: stop  simulations
global_info.DELTA_TIME = 15; %in seconds
caseID = 2 %to identify case for multirun

%%%Bar Design%%%
global_info.TotalCap = 50;
global_info.EntSecureCheckQueCap = 20;
global_info.WdCap = 25;
global_info.BQueCap = 10;
global_info.BWaitingForDrinkCap = 5;
global_info.DfCap = 20;
global_info.TCap = 20;
global_info.ToiQueCap = 10;
global_info.ToiCap = 8;
global_info.ToiSinksCap = 4;

global_info.BnOrderTerminals = 2;
global_info.BnCounters = 2;
if global_info.BnOrderTerminals == 2
    global_info.BnCounters = 2;
end

%%%Employee numbers%%%
nBarAssistant = 2;
nBarback = 1;
nBartender = 2;
nSecurityGuards = 4;

%%%Costs%%%
%%Variable costs (per unit sold)%%
Costs.Ingredients_Drink1 = 25;
```

```
Costs.Ingredients_Drink2 = 25;

%%Employees costs (per employee, per hour)%%
Costs.EmployeeBarAssistant = 185;
Costs.EmployeeBarback = 185;
Costs.EmployeeBartender = 185;
Costs.EmployeeSecurity = 200;
EmploymentHours = 6;

%%Fixed costs (per night)%%
Costs.Fixed.Artists = 300*(EmploymentHours-1);
Costs.Fixed.Electricity = 2000;
Costs.Fixed.Water = 350;
Costs.Fixed.Rent = 5000;
Costs.Fixed.MusicFees = 250;
Costs.Fixed.Insurance = 500;
Costs.Fixed.Maintainance = 750;
Costs.Fixed.ToiletPaper = 100;
Costs.Fixed.Soap = 100;
Costs.Fixed.PaperTowels = 100;

%%%Income%%%
EntranceFee = 150;
WardrobeFee = 25;
PriceDrink1 = 125;
PriceDrink2 = 98;

%%%Customer Behaviour%%%
TimeBetweenCustArrivals = [0 10 0];
%Inhibitor factors, the higher the number (integer), the less likely a
%customer will visit that part of the bar
global_info.BInhibFact = 1;
global_info.TInhibFact = 3;
global_info.DfInhibFact = 3;
global_info.ToiInhibFact = 3;
global_info.ExitInhibFact = 3;

%%%Security Behaviour%%%
%Inhibitor factors, the higher the number (integer), the less likely a
%security guard will perform that action
global_info.SecurityCheckInhibFact = 50;
global_info.SecurityThrowOutInhibFact = 5;

%%%Simulation Setup%%%
pns =
pnstruct({'BarV2_Entrance_pdf','BarV2_IMC_pdf','BarV2_Dancefloor_pdf','BarV2_Table
s_pdf','BarV2_Bar_pdf','BarV2_Toilet_pdf'});
dyn.m0 = {'pEntFreeCap',global_info.TotalCap,... %Customer movement
          'pBCleanGlasses',200,'pBAlcohol',3000,'pBNonAlc',700,'pBIce',1000,...
%Initial Bar supplies [glasses and centiliter]
          'pBSCleanGlasses',100,'pBSAlcohol',6000,'pBSNonAlc',1500,'pBSIce',2000};
%Initial Storage supplies [glasses and centiliter]

if global_info.BnOrderTerminals == 1
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
```

```
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink',[0 1 0],'tIMCWdExit',[0 1
0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer
movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink1',[0 1 0],'tBReceiveDrink2',[0 0
30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0
1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
elseif global_info.BnOrderTerminals == 2
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink',[0 1
0],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1
0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink1',[0 1
0],'tBReceiveDrink2',[0 0 30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2
0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
end

dyn.re =
{'BarAssistant',nBarAssistant,inf,'Barback',nBarback,inf,'Bartender',nBartender,in
f,'SecurityGuard',nSecurityGuards,inf}; %Staff

%%%Run Simulation%%%
pni = initialdynamics(pns, dyn);
results = gpensim(pni);

%%%Print Out results%%%
prnschedule(results)
% prncolormap(results,{'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'})
% prnss(results)

%%Count transition firerings%%
nEntIn = timesfired('tEntIn')
```

```
nEntGenCust = timesfired('tEntGenCust')
nEntSecureCheck = timesfired('tEntSecureCheck')
nEntFeePayed = timesfired('tEntFee')
nWdFeePayed = timesfired('tEntWd')
nEntOut = timesfired('tEntOut')
nBarIn = timesfired('tBIn')
if global_info.BnOrderTerminals == 1
    nDrinksSold = timesfired('tBReceiveDrink')
elseif global_info.BnOrderTerminals == 2
    nDrink1sSold = timesfired('tBReceiveDrink1')
    nDrink2sSold = timesfired('tBReceiveDrink2')
end
nToiIn = timesfired('tToiIn')
nTIn = timesfired('tTIn')
nDfIn = timesfired('tDfIn')
nSecurity1 = timesfired('tIMCSecurity1')
nSecurity2 = timesfired('tIMCSecurity2')
nExit = timesfired('tIMCExit')
nWdExit = timesfired('tIMCWdExit')
nIMC = timesfired('tIMCCustDist')

%%Calculate Costs%%
%Drinks%%
TotCost.Drink1 = nDrink1sSold * Costs.Ingredients_Drink1
TotCost.Drink2 = nDrink2sSold * Costs.Ingredients_Drink2

%Employees%
TotCost.Employees = EmploymentHours*(Costs.EmployeeBarAssistant*nBarAssistant +
Costs.EmployeeBarback*nBarback + Costs.EmployeeBartender*nBartender +
Costs.EmployeeSecurity*nSecurityGuards)

%Fix%
TotCost.Fix = Costs.Fixed.Artists + Costs.Fixed.Electricity + Costs.Fixed.Water +
Costs.Fixed.Rent + Costs.Fixed.MusicFees + Costs.Fixed.Insurance +
Costs.Fixed.Maintainance + Costs.Fixed.ToiletPaper + Costs.Fixed.Soap +
Costs.Fixed.PaperTowels

OverallCost = TotCost.Drink1 + TotCost.Drink2 + TotCost.Employees + TotCost.Fix

%%Calculate Income%%
Income.EntranceFee = EntranceFee*nEntFeePayed
Income.WardrobeFee = WardrobeFee*nWdFeePayed
Income.Drink1 = PriceDrink1*nDrink1sSold
Income.Drink2 = PriceDrink2*nDrink2sSold

TotIncome = Income.EntranceFee + Income.WardrobeFee + Income.Drink1 +
Income.Drink2

Balance = TotIncome - OverallCost

%%%Visualize results%%%
figure(1), plotp(results,
{'pEntSecureCheckQue','pEntFreeCap','pEntSecureOk','pEntFeePayed'});
figure(2), plotp(results, {'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'});
figure(3), plotp(results, {'pDfDancing'});
figure(4), plotp(results, {'pTLookingForT', 'pTSittingAtT', 'pTOutBuffer'});
if global_info.BnOrderTerminals == 1
```

```
        figure(5), plotp(results, {'pBQue', 'pBWaitingForDrink', 'pBOutBuffer'});
elseif global_info.BnOrderTerminals == 2
        figure(5), plotp(results, {'pBQue',
'pBWaitingForDrink1','pBWaitingForDrink2', 'pBOutBuffer'});
end
figure(6), plotp(results, {'pBUsedGlasses','pBDirtyGlasses'});
figure(7), plotp(results, {'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce'});
figure(8), plotp(results, {'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'});
figure(9), plotp(results, {'pToiQue', 'pToi',
'pToiSinkQue','pToiSinks','pToiOutBuffer'});
plotGC(results)
```

BarV2.m (Case 3)

```
% BarV2

%%%Abrriviations:%%%
%B: Bar
%Cap: Capacity
%Cust: Customer
%Df: Dancefloor
%Ent: Entrance
%Gen: Generate
%IMC: Inter-Modular-Connector
%n: number of [...]
%NonAlc: Non-Alcoholic
%R: Resupply
%S: Storage
%T: Tables
%Toi: Toilets
%Wd: Wardrobe

clear all; clc;
global global_info

%%%Simulation Parameters%%%
global_info.START_AT = [22 0 0]; %OPTION: start simulations
global_info.STOP_AT  = [27 0 0]; %OPTION: stop  simulations
global_info.DELTA_TIME = 15; %in seconds
caseID = 3 %to identify case for multirun

%%%Bar Design%%%
global_info.TotalCap = 50;
global_info.EntSecureCheckQueCap = 20;
global_info.WdCap = 25;
global_info.BQueCap = 10;
global_info.BWaitingForDrinkCap = 5;
global_info.DfCap = 20;
global_info.TCap = 20;
global_info.ToiQueCap = 10;
global_info.ToiCap = 8;
global_info.ToiSinksCap = 4;

global_info.BnOrderTerminals = 2;
global_info.BnCounters = 2;
if global_info.BnOrderTerminals == 2
    global_info.BnCounters = 2;
end

%%%Employee numbers%%%
nBarAssistant = 1;
nBarback = 1;
nBartender = 1;
nSecurityGuards = 2;

%%%Costs%%%
%%Variable costs (per unit sold)%%
Costs.Ingredients_Drink1 = 25;
```

```
Costs.Ingredients_Drink2 = 25;

%%Employees costs (per employee, per hour)%%
Costs.EmployeeBarAssistant = 185;
Costs.EmployeeBarback = 185;
Costs.EmployeeBartender = 185;
Costs.EmployeeSecurity = 200;
EmploymentHours = 6;

%%Fixed costs (per night)%%
Costs.Fixed.Artists = 300*(EmploymentHours-1);
Costs.Fixed.Electricity = 2000;
Costs.Fixed.Water = 350;
Costs.Fixed.Rent = 5000;
Costs.Fixed.MusicFees = 250;
Costs.Fixed.Insurance = 500;
Costs.Fixed.Maintainance = 750;
Costs.Fixed.ToiletPaper = 100;
Costs.Fixed.Soap = 100;
Costs.Fixed.PaperTowels = 100;

%%%Income%%%
EntranceFee = 150;
WardrobeFee = 25;
PriceDrink1 = 125;
PriceDrink2 = 98;

%%%Customer Behaviour%%%
TimeBetweenCustArrivals = [0 10 0];
%Inhibitor factors, the higher the number (integer), the less likely a
%customer will visit that part of the bar
global_info.BInhibFact = 1;
global_info.TInhibFact = 3;
global_info.DfInhibFact = 3;
global_info.ToiInhibFact = 3;
global_info.ExitInhibFact = 3;

%%%Security Behaviour%%%
%Inhibitor factors, the higher the number (integer), the less likely a
%security guard will perform that action
global_info.SecurityCheckInhibFact = 50;
global_info.SecurityThrowOutInhibFact = 5;

%%%Simulation Setup%%%
pns =
pnstruct({'BarV2_Entrance_pdf','BarV2_IMC_pdf','BarV2_Dancefloor_pdf','BarV2_Table
s_pdf','BarV2_Bar_pdf','BarV2_Toilet_pdf'});
dyn.m0 = {'pEntFreeCap',global_info.TotalCap,... %Customer movement
          'pBCleanGlasses',200,'pBAlcohol',3000,'pBNonAlc',700,'pBIce',1000,...
%Initial Bar supplies [glasses and centiliter]
          'pBSCleanGlasses',100,'pBSAlcohol',6000,'pBSNonAlc',1500,'pBSIce',2000};
%Initial Storage supplies [glasses and centiliter]

if global_info.BnOrderTerminals == 1
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
```

```
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink',[0 1 0],'tIMCWdExit',[0 1
0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer
movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink1',[0 1 0],'tBReceiveDrink2',[0 0
30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0
1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
elseif global_info.BnOrderTerminals == 2
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink',[0 1
0],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1
0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink1',[0 1
0],'tBReceiveDrink2',[0 0 30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2
0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
end

dyn.re =
{'BarAssistant',nBarAssistant,inf,'Barback',nBarback,inf,'Bartender',nBartender,in
f,'SecurityGuard',nSecurityGuards,inf}; %Staff

%%%Run Simulation%%%
pni = initialdynamics(pns, dyn);
results = gpensim(pni);

%%%Print Out results%%%
prnschedule(results)
% prncolormap(results,{'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'})
% prnss(results)

%%Count transition firerings%%
nEntIn = timesfired('tEntIn')
```

```
nEntGenCust = timesfired('tEntGenCust')
nEntSecureCheck = timesfired('tEntSecureCheck')
nEntFeePayed = timesfired('tEntFee')
nWdFeePayed = timesfired('tEntWd')
nEntOut = timesfired('tEntOut')
nBarIn = timesfired('tBIn')
if global_info.BnOrderTerminals == 1
    nDrinksSold = timesfired('tBReceiveDrink')
elseif global_info.BnOrderTerminals == 2
    nDrink1sSold = timesfired('tBReceiveDrink1')
    nDrink2sSold = timesfired('tBReceiveDrink2')
end
nToiIn = timesfired('tToiIn')
nTIn = timesfired('tTIn')
nDfIn = timesfired('tDfIn')
nSecurity1 = timesfired('tIMCSecurity1')
nSecurity2 = timesfired('tIMCSecurity2')
nExit = timesfired('tIMCExit')
nWdExit = timesfired('tIMCWdExit')
nIMC = timesfired('tIMCCustDist')

%%Calculate Costs%%
%Drinks%%
TotCost.Drink1 = nDrink1sSold * Costs.Ingredients_Drink1
TotCost.Drink2 = nDrink2sSold * Costs.Ingredients_Drink2

%Employees%
TotCost.Employees = EmploymentHours*(Costs.EmployeeBarAssistant*nBarAssistant +
Costs.EmployeeBarback*nBarback + Costs.EmployeeBartender*nBartender +
Costs.EmployeeSecurity*nSecurityGuards)

%Fix%
TotCost.Fix = Costs.Fixed.Artists + Costs.Fixed.Electricity + Costs.Fixed.Water +
Costs.Fixed.Rent + Costs.Fixed.MusicFees + Costs.Fixed.Insurance +
Costs.Fixed.Maintainance + Costs.Fixed.ToiletPaper + Costs.Fixed.Soap +
Costs.Fixed.PaperTowels

OverallCost = TotCost.Drink1 + TotCost.Drink2 + TotCost.Employees + TotCost.Fix

%%Calculate Income%%
Income.EntranceFee = EntranceFee*nEntFeePayed
Income.WardrobeFee = WardrobeFee*nWdFeePayed
Income.Drink1 = PriceDrink1*nDrink1sSold
Income.Drink2 = PriceDrink2*nDrink2sSold

TotIncome = Income.EntranceFee + Income.WardrobeFee + Income.Drink1 +
Income.Drink2

Balance = TotIncome - OverallCost

%%%Visualize results%%%
figure(1), plotp(results,
{'pEntSecureCheckQue','pEntFreeCap','pEntSecureOk','pEntFeePayed'});
figure(2), plotp(results, {'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'});
figure(3), plotp(results, {'pDfDancing'});
figure(4), plotp(results, {'pTLookingForT', 'pTSittingAtT', 'pTOutBuffer'});
if global_info.BnOrderTerminals == 1
```

```
        figure(5), plotp(results, {'pBQue', 'pBWaitingForDrink', 'pBOutBuffer'});
elseif global_info.BnOrderTerminals == 2
        figure(5), plotp(results, {'pBQue',
'pBWaitingForDrink1','pBWaitingForDrink2', 'pBOutBuffer'});
end
figure(6), plotp(results, {'pBUsedGlasses','pBDirtyGlasses'});
figure(7), plotp(results, {'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce'});
figure(8), plotp(results, {'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'});
figure(9), plotp(results, {'pToiQue', 'pToi',
'pToiSinkQue','pToiSinks','pToiOutBuffer'});
plotGC(results)
```

BarV2.m (Case 4)

```
% BarV2

%%%Abrriviations:%%%
%B: Bar
%Cap: Capacity
%Cust: Customer
%Df: Dancefloor
%Ent: Entrance
%Gen: Generate
%IMC: Inter-Modular-Connector
%n: number of [...]
%NonAlc: Non-Alcoholic
%R: Resupply
%S: Storage
%T: Tables
%Toi: Toilets
%Wd: Wardrobe

clear all; clc;
global global_info

%%%Simulation Parameters%%%
global_info.START_AT = [22 0 0]; %OPTION: start simulations
global_info.STOP_AT  = [27 0 0]; %OPTION: stop  simulations
global_info.DELTA_TIME = 15; %in seconds
caseID = 4 %to identify case for multirun

%%%Bar Design%%%
global_info.TotalCap = 50;
global_info.EntSecureCheckQueCap = 20;
global_info.WdCap = 25;
global_info.BQueCap = 10;
global_info.BWaitingForDrinkCap = 5;
global_info.DfCap = 20;
global_info.TCap = 20;
global_info.ToiQueCap = 10;
global_info.ToiCap = 8;
global_info.ToiSinksCap = 4;

global_info.BnOrderTerminals = 2;
global_info.BnCounters = 2;
if global_info.BnOrderTerminals == 2
    global_info.BnCounters = 2;
end

%%%Employee numbers%%%
nBarAssistant = 1;
nBarback = 1;
nBartender = 1;
nSecurityGuards = 2;

%%%Costs%%%
%%Variable costs (per unit sold)%%
Costs.Ingredients_Drink1 = 25;
```

```
Costs.Ingredients_Drink2 = 25;

%%Employees costs (per employee, per hour)%%
Costs.EmployeeBarAssistant = 185;
Costs.EmployeeBarback = 185;
Costs.EmployeeBartender = 185;
Costs.EmployeeSecurity = 200;
EmploymentHours = 6;

%%Fixed costs (per night)%%
Costs.Fixed.Artists = 300*(EmploymentHours-1);
Costs.Fixed.Electricity = 2000;
Costs.Fixed.Water = 350;
Costs.Fixed.Rent = 5000;
Costs.Fixed.MusicFees = 250;
Costs.Fixed.Insurance = 500;
Costs.Fixed.Maintainance = 750;
Costs.Fixed.ToiletPaper = 100;
Costs.Fixed.Soap = 100;
Costs.Fixed.PaperTowels = 100;

%%%Income%%%
EntranceFee = 150;
WardrobeFee = 25;
PriceDrink1 = 150;
PriceDrink2 = 125;

%%%Customer Behaviour%%%
TimeBetweenCustArrivals = [0 10 0];
%Inhibitor factors, the higher the number (integer), the less likely a
%customer will visit that part of the bar
global_info.BInhibFact = 1;
global_info.TInhibFact = 2;
global_info.DfInhibFact = 2;
global_info.ToiInhibFact = 2;
global_info.ExitInhibFact = 2;

%%%Security Behaviour%%%
%Inhibitor factors, the higher the number (integer), the less likely a
%security guard will perform that action
global_info.SecurityCheckInhibFact = 50;
global_info.SecurityThrowOutInhibFact = 5;

%%%Simulation Setup%%%
pns =
pnstruct({'BarV2_Entrance_pdf','BarV2_IMC_pdf','BarV2_Dancefloor_pdf','BarV2_Table
s_pdf','BarV2_Bar_pdf','BarV2_Toilet_pdf'});
dyn.m0 = {'pEntFreeCap',global_info.TotalCap,... %Customer movement
          'pBCleanGlasses',200,'pBAlcohol',3000,'pBNonAlc',700,'pBIce',1000,...
%Initial Bar supplies [glasses and centiliter]
          'pBSCleanGlasses',100,'pBSAlcohol',6000,'pBSNonAlc',1500,'pBSIce',2000};
%Initial Storage supplies [glasses and centiliter]

if global_info.BnOrderTerminals == 1
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
```

```
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink',[0 1 0],'tIMCWdExit',[0 1
0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer
movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink1',[0 1 0],'tBReceiveDrink2',[0 0
30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0
1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
elseif global_info.BnOrderTerminals == 2
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink',[0 1
0],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1
0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink1',[0 1
0],'tBReceiveDrink2',[0 0 30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2
0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
end

dyn.re =
{'BarAssistant',nBarAssistant,inf,'Barback',nBarback,inf,'Bartender',nBartender,in
f,'SecurityGuard',nSecurityGuards,inf}; %Staff

%%%Run Simulation%%%
pni = initialdynamics(pns, dyn);
results = gpensim(pni);

%%%Print Out results%%%
prnschedule(results)
% prncolormap(results,{'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'})
% prnss(results)

%%Count transition firerings%%
nEntIn = timesfired('tEntIn')
```

```
nEntGenCust = timesfired('tEntGenCust')
nEntSecureCheck = timesfired('tEntSecureCheck')
nEntFeePayed = timesfired('tEntFee')
nWdFeePayed = timesfired('tEntWd')
nEntOut = timesfired('tEntOut')
nBarIn = timesfired('tBIn')
if global_info.BnOrderTerminals == 1
    nDrinksSold = timesfired('tBReceiveDrink')
elseif global_info.BnOrderTerminals == 2
    nDrink1sSold = timesfired('tBReceiveDrink1')
    nDrink2sSold = timesfired('tBReceiveDrink2')
end
nToiIn = timesfired('tToiIn')
nTIn = timesfired('tTIn')
nDfIn = timesfired('tDfIn')
nSecurity1 = timesfired('tIMCSecurity1')
nSecurity2 = timesfired('tIMCSecurity2')
nExit = timesfired('tIMCExit')
nWdExit = timesfired('tIMCWdExit')
nIMC = timesfired('tIMCCustDist')

%%Calculate Costs%%
%Drinks%%
TotCost.Drink1 = nDrink1sSold * Costs.Ingredients_Drink1
TotCost.Drink2 = nDrink2sSold * Costs.Ingredients_Drink2

%Employees%
TotCost.Employees = EmploymentHours*(Costs.EmployeeBarAssistant*nBarAssistant +
Costs.EmployeeBarback*nBarback + Costs.EmployeeBartender*nBartender +
Costs.EmployeeSecurity*nSecurityGuards)

%Fix%
TotCost.Fix = Costs.Fixed.Artists + Costs.Fixed.Electricity + Costs.Fixed.Water +
Costs.Fixed.Rent + Costs.Fixed.MusicFees + Costs.Fixed.Insurance +
Costs.Fixed.Maintainance + Costs.Fixed.ToiletPaper + Costs.Fixed.Soap +
Costs.Fixed.PaperTowels

OverallCost = TotCost.Drink1 + TotCost.Drink2 + TotCost.Employees + TotCost.Fix

%%Calculate Income%%
Income.EntranceFee = EntranceFee*nEntFeePayed
Income.WardrobeFee = WardrobeFee*nWdFeePayed
Income.Drink1 = PriceDrink1*nDrink1sSold
Income.Drink2 = PriceDrink2*nDrink2sSold

TotIncome = Income.EntranceFee + Income.WardrobeFee + Income.Drink1 +
Income.Drink2

Balance = TotIncome - OverallCost

%%%Visualize results%%%
figure(1), plotp(results,
{'pEntSecureCheckQue','pEntFreeCap','pEntSecureOk','pEntFeePayed'});
figure(2), plotp(results, {'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'});
figure(3), plotp(results, {'pDfDancing'});
figure(4), plotp(results, {'pTLookingForT', 'pTSittingAtT', 'pTOutBuffer'});
if global_info.BnOrderTerminals == 1
```

```
        figure(5), plotp(results, {'pBQue', 'pBWaitingForDrink', 'pBOutBuffer'});
elseif global_info.BnOrderTerminals == 2
        figure(5), plotp(results, {'pBQue',
'pBWaitingForDrink1','pBWaitingForDrink2', 'pBOutBuffer'});
end
figure(6), plotp(results, {'pBUsedGlasses','pBDirtyGlasses'});
figure(7), plotp(results, {'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce'});
figure(8), plotp(results, {'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'});
figure(9), plotp(results, {'pToiQue', 'pToi',
'pToiSinkQue','pToiSinks','pToiOutBuffer'});
plotGC(results)
```

BarV2.m (Case 5)

```
% BarV2

%%%Abrriviations:%%%
%B: Bar
%Cap: Capacity
%Cust: Customer
%Df: Dancefloor
%Ent: Entrance
%Gen: Generate
%IMC: Inter-Modular-Connector
%n: number of [...]
%NonAlc: Non-Alcoholic
%R: Resupply
%S: Storage
%T: Tables
%Toi: Toilets
%Wd: Wardrobe

clear all; clc;
global global_info

%%%Simulation Parameters%%%
global_info.START_AT = [22 0 0]; %OPTION: start simulations
global_info.STOP_AT  = [27 0 0]; %OPTION: stop  simulations
global_info.DELTA_TIME = 15; %in seconds
caseID = 5 %to identify case for multirun

%%%Bar Design%%%
global_info.TotalCap = 50;
global_info.EntSecureCheckQueCap = 20;
global_info.WdCap = 25;
global_info.BQueCap = 10;
global_info.BWaitingForDrinkCap = 5;
global_info.DfCap = 20;
global_info.TCap = 20;
global_info.ToiQueCap = 10;
global_info.ToiCap = 8;
global_info.ToiSinksCap = 4;

global_info.BnOrderTerminals = 2;
global_info.BnCounters = 2;
if global_info.BnOrderTerminals == 2
    global_info.BnCounters = 2;
end

%%%Employee numbers%%%
nBarAssistant = 1;
nBarback = 1;
nBartender = 1;
nSecurityGuards = 2;

%%%Costs%%%
%%Variable costs (per unit sold)%%
Costs.Ingredients_Drink1 = 25;
```

```
Costs.Ingredients_Drink2 = 25;

%%Employees costs (per employee, per hour)%%
Costs.EmployeeBarAssistant = 185;
Costs.EmployeeBarback = 185;
Costs.EmployeeBartender = 185;
Costs.EmployeeSecurity = 200;
EmploymentHours = 6;

%%Fixed costs (per night)%%
Costs.Fixed.Artists = 300*(EmploymentHours-1);
Costs.Fixed.Electricity = 2000;
Costs.Fixed.Water = 350;
Costs.Fixed.Rent = 5000;
Costs.Fixed.MusicFees = 250;
Costs.Fixed.Insurance = 500;
Costs.Fixed.Maintainance = 750;
Costs.Fixed.ToiletPaper = 100;
Costs.Fixed.Soap = 100;
Costs.Fixed.PaperTowels = 100;

%%%Income%%%
EntranceFee = 150;
WardrobeFee = 25;
PriceDrink1 = 100;
PriceDrink2 = 75;

%%%Customer Behaviour%%%
TimeBetweenCustArrivals = [0 10 0];
%Inhibitor factors, the higher the number (integer), the less likely a
%customer will visit that part of the bar
global_info.BInhibFact = 1;
global_info.TInhibFact = 4;
global_info.DfInhibFact = 4;
global_info.ToiInhibFact = 4;
global_info.ExitInhibFact = 4;

%%%Security Behaviour%%%
%Inhibitor factors, the higher the number (integer), the less likely a
%security guard will perform that action
global_info.SecurityCheckInhibFact = 50;
global_info.SecurityThrowOutInhibFact = 5;

%%%Simulation Setup%%%
pns =
pnstruct({'BarV2_Entrance_pdf','BarV2_IMC_pdf','BarV2_Dancefloor_pdf','BarV2_Table
s_pdf','BarV2_Bar_pdf','BarV2_Toilet_pdf'});
dyn.m0 = {'pEntFreeCap',global_info.TotalCap,... %Customer movement
          'pBCleanGlasses',200,'pBAlcohol',3000,'pBNonAlc',700,'pBIce',1000,...
%Initial Bar supplies [glasses and centiliter]
          'pBSCleanGlasses',100,'pBSAlcohol',6000,'pBSNonAlc',1500,'pBSIce',2000};
%Initial Storage supplies [glasses and centiliter]

if global_info.BnOrderTerminals == 1
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
```

```
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink',[0 1 0],'tIMCWdExit',[0 1
0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer
movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink1',[0 1 0],'tBReceiveDrink2',[0 0
30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0
1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
elseif global_info.BnOrderTerminals == 2
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink',[0 1
0],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1
0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink1',[0 1
0],'tBReceiveDrink2',[0 0 30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2
0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
end

dyn.re =
{'BarAssistant',nBarAssistant,inf,'Barback',nBarback,inf,'Bartender',nBartender,in
f,'SecurityGuard',nSecurityGuards,inf}; %Staff

%%%Run Simulation%%%
pni = initialdynamics(pns, dyn);
results = gpensim(pni);

%%%Print Out results%%%
prnschedule(results)
% prncolormap(results,{'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'})
% prnss(results)

%%Count transition firerings%%
nEntIn = timesfired('tEntIn')
```

```
nEntGenCust = timesfired('tEntGenCust')
nEntSecureCheck = timesfired('tEntSecureCheck')
nEntFeePayed = timesfired('tEntFee')
nWdFeePayed = timesfired('tEntWd')
nEntOut = timesfired('tEntOut')
nBarIn = timesfired('tBIn')
if global_info.BnOrderTerminals == 1
    nDrinksSold = timesfired('tBReceiveDrink')
elseif global_info.BnOrderTerminals == 2
    nDrink1sSold = timesfired('tBReceiveDrink1')
    nDrink2sSold = timesfired('tBReceiveDrink2')
end
nToiIn = timesfired('tToiIn')
nTIn = timesfired('tTIn')
nDfIn = timesfired('tDfIn')
nSecurity1 = timesfired('tIMCSecurity1')
nSecurity2 = timesfired('tIMCSecurity2')
nExit = timesfired('tIMCExit')
nWdExit = timesfired('tIMCWdExit')
nIMC = timesfired('tIMCCustDist')

%%Calculate Costs%%
%Drinks%%
TotCost.Drink1 = nDrink1sSold * Costs.Ingredients_Drink1
TotCost.Drink2 = nDrink2sSold * Costs.Ingredients_Drink2

%Employees%
TotCost.Employees = EmploymentHours*(Costs.EmployeeBarAssistant*nBarAssistant +
Costs.EmployeeBarback*nBarback + Costs.EmployeeBartender*nBartender +
Costs.EmployeeSecurity*nSecurityGuards)

%Fix%
TotCost.Fix = Costs.Fixed.Artists + Costs.Fixed.Electricity + Costs.Fixed.Water +
Costs.Fixed.Rent + Costs.Fixed.MusicFees + Costs.Fixed.Insurance +
Costs.Fixed.Maintainance + Costs.Fixed.ToiletPaper + Costs.Fixed.Soap +
Costs.Fixed.PaperTowels

OverallCost = TotCost.Drink1 + TotCost.Drink2 + TotCost.Employees + TotCost.Fix

%%Calculate Income%%
Income.EntranceFee = EntranceFee*nEntFeePayed
Income.WardrobeFee = WardrobeFee*nWdFeePayed
Income.Drink1 = PriceDrink1*nDrink1sSold
Income.Drink2 = PriceDrink2*nDrink2sSold

TotIncome = Income.EntranceFee + Income.WardrobeFee + Income.Drink1 +
Income.Drink2

Balance = TotIncome - OverallCost

%%%Visualize results%%%
figure(1), plotp(results,
{'pEntSecureCheckQue','pEntFreeCap','pEntSecureOk','pEntFeePayed'});
figure(2), plotp(results, {'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'});
figure(3), plotp(results, {'pDfDancing'});
figure(4), plotp(results, {'pTLookingForT', 'pTSittingAtT', 'pTOutBuffer'});
if global_info.BnOrderTerminals == 1
```

```
        figure(5), plotp(results, {'pBQue', 'pBWaitingForDrink', 'pBOutBuffer'});
elseif global_info.BnOrderTerminals == 2
        figure(5), plotp(results, {'pBQue',
'pBWaitingForDrink1','pBWaitingForDrink2', 'pBOutBuffer'});
end
figure(6), plotp(results, {'pBUsedGlasses','pBDirtyGlasses'});
figure(7), plotp(results, {'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce'});
figure(8), plotp(results, {'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'});
figure(9), plotp(results, {'pToiQue', 'pToi',
'pToiSinkQue','pToiSinks','pToiOutBuffer'});
plotGC(results)
```

BarV2.m (Case 6)

```
% BarV2

%%%Abrriviations:%%%
%B: Bar
%Cap: Capacity
%Cust: Customer
%Df: Dancefloor
%Ent: Entrance
%Gen: Generate
%IMC: Inter-Modular-Connector
%n: number of [...]
%NonAlc: Non-Alcoholic
%R: Resupply
%S: Storage
%T: Tables
%Toi: Toilets
%Wd: Wardrobe

clear all; clc;
global global_info

%%%Simulation Parameters%%%
global_info.START_AT = [22 0 0]; %OPTION: start simulations
global_info.STOP_AT  = [27 0 0]; %OPTION: stop  simulations
global_info.DELTA_TIME = 15; %in seconds
caseID = 6 %to identify case for multirun

%%%Bar Design%%%
global_info.TotalCap = 50;
global_info.EntSecureCheckQueCap = 20;
global_info.WdCap = 25;
global_info.BQueCap = 10;
global_info.BWaitingForDrinkCap = 5;
global_info.DfCap = 20;
global_info.TCap = 20;
global_info.ToiQueCap = 10;
global_info.ToiCap = 8;
global_info.ToiSinksCap = 4;

global_info.BnOrderTerminals = 2;
global_info.BnCounters = 2;
if global_info.BnOrderTerminals == 2
    global_info.BnCounters = 2;
end

%%%Employee numbers%%%
nBarAssistant = 1;
nBarback = 1;
nBartender = 1;
nSecurityGuards = 2;

%%%Costs%%%
%%Variable costs (per unit sold)%%
Costs.Ingredients_Drink1 = 25;
```

```
Costs.Ingredients_Drink2 = 25;

%%Employees costs (per employee, per hour)%%
Costs.EmployeeBarAssistant = 185;
Costs.EmployeeBarback = 185;
Costs.EmployeeBartender = 185;
Costs.EmployeeSecurity = 200;
EmploymentHours = 6;

%%Fixed costs (per night)%%
Costs.Fixed.Artists = 300*(EmploymentHours-1);
Costs.Fixed.Electricity = 2000;
Costs.Fixed.Water = 350;
Costs.Fixed.Rent = 5000;
Costs.Fixed.MusicFees = 250;
Costs.Fixed.Insurance = 500;
Costs.Fixed.Maintainance = 750;
Costs.Fixed.ToiletPaper = 100;
Costs.Fixed.Soap = 100;
Costs.Fixed.PaperTowels = 100;

%%%Income%%%
EntranceFee = 0;
WardrobeFee = 25;
PriceDrink1 = 150;
PriceDrink2 = 125;

%%%Customer Behaviour%%%
TimeBetweenCustArrivals = [0 3 0];
%Inhibitor factors, the higher the number (integer), the less likely a
%customer will visit that part of the bar
global_info.BInhibFact = 1;
global_info.TInhibFact = 3;
global_info.DfInhibFact = 3;
global_info.ToiInhibFact = 3;
global_info.ExitInhibFact = 3;

%%%Security Behaviour%%%
%Inhibitor factors, the higher the number (integer), the less likely a
%security guard will perform that action
global_info.SecurityCheckInhibFact = 50;
global_info.SecurityThrowOutInhibFact = 5;

%%%Simulation Setup%%%
pns =
pnstruct({'BarV2_Entrance_pdf','BarV2_IMC_pdf','BarV2_Dancefloor_pdf','BarV2_Table
s_pdf','BarV2_Bar_pdf','BarV2_Toilet_pdf'});
dyn.m0 = {'pEntFreeCap',global_info.TotalCap,... %Customer movement
          'pBCleanGlasses',200,'pBAlcohol',3000,'pBNonAlc',700,'pBIce',1000,...
%Initial Bar supplies [glasses and centiliter]
          'pBSCleanGlasses',100,'pBSAlcohol',6000,'pBSNonAlc',1500,'pBSIce',2000};
%Initial Storage supplies [glasses and centiliter]

if global_info.BnOrderTerminals == 1
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
```

```
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink',[0 1 0],'tIMCWdExit',[0 1
0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer
movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink1',[0 1 0],'tBReceiveDrink2',[0 0
30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0
1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
elseif global_info.BnOrderTerminals == 2
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink',[0 1
0],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1
0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink1',[0 1
0],'tBReceiveDrink2',[0 0 30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2
0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
end

dyn.re =
{'BarAssistant',nBarAssistant,inf,'Barback',nBarback,inf,'Bartender',nBartender,in
f,'SecurityGuard',nSecurityGuards,inf}; %Staff

%%%Run Simulation%%%
pni = initialdynamics(pns, dyn);
results = gpensim(pni);

%%%Print Out results%%%
prnschedule(results)
% prncolormap(results,{'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'})
% prnss(results)

%%Count transition firerings%%
nEntIn = timesfired('tEntIn')
```

```
nEntGenCust = timesfired('tEntGenCust')
nEntSecureCheck = timesfired('tEntSecureCheck')
nEntFeePayed = timesfired('tEntFee')
nWdFeePayed = timesfired('tEntWd')
nEntOut = timesfired('tEntOut')
nBarIn = timesfired('tBIn')
if global_info.BnOrderTerminals == 1
    nDrinksSold = timesfired('tBReceiveDrink')
elseif global_info.BnOrderTerminals == 2
    nDrink1sSold = timesfired('tBReceiveDrink1')
    nDrink2sSold = timesfired('tBReceiveDrink2')
end
nToiIn = timesfired('tToiIn')
nTIn = timesfired('tTIn')
nDfIn = timesfired('tDfIn')
nSecurity1 = timesfired('tIMCSecurity1')
nSecurity2 = timesfired('tIMCSecurity2')
nExit = timesfired('tIMCExit')
nWdExit = timesfired('tIMCWdExit')
nIMC = timesfired('tIMCCustDist')

%%Calculate Costs%%
%Drinks%%
TotCost.Drink1 = nDrink1sSold * Costs.Ingredients_Drink1
TotCost.Drink2 = nDrink2sSold * Costs.Ingredients_Drink2

%Employees%
TotCost.Employees = EmploymentHours*(Costs.EmployeeBarAssistant*nBarAssistant +
Costs.EmployeeBarback*nBarback + Costs.EmployeeBartender*nBartender +
Costs.EmployeeSecurity*nSecurityGuards)

%Fix%
TotCost.Fix = Costs.Fixed.Artists + Costs.Fixed.Electricity + Costs.Fixed.Water +
Costs.Fixed.Rent + Costs.Fixed.MusicFees + Costs.Fixed.Insurance +
Costs.Fixed.Maintainance + Costs.Fixed.ToiletPaper + Costs.Fixed.Soap +
Costs.Fixed.PaperTowels

OverallCost = TotCost.Drink1 + TotCost.Drink2 + TotCost.Employees + TotCost.Fix

%%Calculate Income%%
Income.EntranceFee = EntranceFee*nEntFeePayed
Income.WardrobeFee = WardrobeFee*nWdFeePayed
Income.Drink1 = PriceDrink1*nDrink1sSold
Income.Drink2 = PriceDrink2*nDrink2sSold

TotIncome = Income.EntranceFee + Income.WardrobeFee + Income.Drink1 +
Income.Drink2

Balance = TotIncome - OverallCost

%%%Visualize results%%%
figure(1), plotp(results,
{'pEntSecureCheckQue','pEntFreeCap','pEntSecureOk','pEntFeePayed'});
figure(2), plotp(results, {'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'});
figure(3), plotp(results, {'pDfDancing'});
figure(4), plotp(results, {'pTLookingForT', 'pTSittingAtT', 'pTOutBuffer'});
if global_info.BnOrderTerminals == 1
```

```
        figure(5), plotp(results, {'pBQue', 'pBWaitingForDrink', 'pBOutBuffer'});
elseif global_info.BnOrderTerminals == 2
        figure(5), plotp(results, {'pBQue',
'pBWaitingForDrink1','pBWaitingForDrink2', 'pBOutBuffer'});
end
figure(6), plotp(results, {'pBUsedGlasses','pBDirtyGlasses'});
figure(7), plotp(results, {'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce'});
figure(8), plotp(results, {'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'});
figure(9), plotp(results, {'pToiQue', 'pToi',
'pToiSinkQue','pToiSinks','pToiOutBuffer'});
plotGC(results)
```

BarV2.m (Case 7)

```
% BarV2

%%%Abrriviations:%%%
%B: Bar
%Cap: Capacity
%Cust: Customer
%Df: Dancefloor
%Ent: Entrance
%Gen: Generate
%IMC: Inter-Modular-Connector
%n: number of [...]
%NonAlc: Non-Alcoholic
%R: Resupply
%S: Storage
%T: Tables
%Toi: Toilets
%Wd: Wardrobe

clear all; clc;
global global_info

%%%Simulation Parameters%%%
global_info.START_AT = [22 0 0]; %OPTION: start simulations
global_info.STOP_AT  = [27 0 0]; %OPTION: stop  simulations
global_info.DELTA_TIME = 15; %in seconds
caseID = 7 %to identify case for multirun

%%%Bar Design%%%
global_info.TotalCap = 120;
global_info.EntSecureCheckQueCap = 20;
global_info.WdCap = 25;
global_info.BQueCap = 10;
global_info.BWaitingForDrinkCap = 5;
global_info.DfCap = 50;
global_info.TCap = 50;
global_info.ToiQueCap = 10;
global_info.ToiCap = 8;
global_info.ToiSinksCap = 4;

global_info.BnOrderTerminals = 2;
global_info.BnCounters = 2;
if global_info.BnOrderTerminals == 2
    global_info.BnCounters = 2;
end

%%%Employee numbers%%%
nBarAssistant = 2;
nBarback = 1;
nBartender = 2;
nSecurityGuards = 4;

%%%Costs%%%
%%Variable costs (per unit sold)%%
Costs.Ingredients_Drink1 = 25;
```

```
Costs.Ingredients_Drink2 = 25;

%%Employees costs (per employee, per hour)%%
Costs.EmployeeBarAssistant = 185;
Costs.EmployeeBarback = 185;
Costs.EmployeeBartender = 185;
Costs.EmployeeSecurity = 200;
EmploymentHours = 6;

%%Fixed costs (per night)%%
Costs.Fixed.Artists = 300*(EmploymentHours-1);
Costs.Fixed.Electricity = 4000;
Costs.Fixed.Water = 500;
Costs.Fixed.Rent = 10000;
Costs.Fixed.MusicFees = 250;
Costs.Fixed.Insurance = 1000;
Costs.Fixed.Maintainance = 1500;
Costs.Fixed.ToiletPaper = 100;
Costs.Fixed.Soap = 100;
Costs.Fixed.PaperTowels = 100;

%%%Income%%%
EntranceFee = 150;
WardrobeFee = 25;
PriceDrink1 = 125;
PriceDrink2 = 98;

%%%Customer Behaviour%%%
TimeBetweenCustArrivals = [0 1 0];
%Inhibitor factors, the higher the number (integer), the less likely a
%customer will visit that part of the bar
global_info.BInhibFact = 1;
global_info.TInhibFact = 3;
global_info.DfInhibFact = 3;
global_info.ToiInhibFact = 3;
global_info.ExitInhibFact = 3;

%%%Security Behaviour%%%
%Inhibitor factors, the higher the number (integer), the less likely a
%security guard will perform that action
global_info.SecurityCheckInhibFact = 50;
global_info.SecurityThrowOutInhibFact = 5;

%%%Simulation Setup%%%
pns =
pnstruct({'BarV2_Entrance_pdf','BarV2_IMC_pdf','BarV2_Dancefloor_pdf','BarV2_Table
s_pdf','BarV2_Bar_pdf','BarV2_Toilet_pdf'});
dyn.m0 = {'pEntFreeCap',global_info.TotalCap,... %Customer movement
          'pBCleanGlasses',200,'pBAlcohol',3000,'pBNonAlc',700,'pBIce',1000,...
%Initial Bar supplies [glasses and centiliter]
          'pBSCleanGlasses',100,'pBSAlcohol',6000,'pBSNonAlc',1500,'pBSIce',2000};
%Initial Storage supplies [glasses and centiliter]

if global_info.BnOrderTerminals == 1
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
```

```
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink',[0 1 0],'tIMCWdExit',[0 1
0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer
movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink1',[0 1 0],'tBReceiveDrink2',[0 0
30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0
1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
elseif global_info.BnOrderTerminals == 2
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink',[0 1
0],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1
0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink1',[0 1
0],'tBReceiveDrink2',[0 0 30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2
0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer movement
                    'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                    'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
end

dyn.re =
{'BarAssistant',nBarAssistant,inf,'Barback',nBarback,inf,'Bartender',nBartender,in
f,'SecurityGuard',nSecurityGuards,inf}; %Staff

%%%Run Simulation%%%
pni = initialdynamics(pns, dyn);
results = gpensim(pni);

%%%Print Out results%%%
prnschedule(results)
% prncolormap(results,{'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'})
% prnss(results)

%%Count transition firerings%%
nEntIn = timesfired('tEntIn')
```

```
nEntGenCust = timesfired('tEntGenCust')
nEntSecureCheck = timesfired('tEntSecureCheck')
nEntFeePayed = timesfired('tEntFee')
nWdFeePayed = timesfired('tEntWd')
nEntOut = timesfired('tEntOut')
nBarIn = timesfired('tBIn')
if global_info.BnOrderTerminals == 1
    nDrinksSold = timesfired('tBReceiveDrink')
elseif global_info.BnOrderTerminals == 2
    nDrink1sSold = timesfired('tBReceiveDrink1')
    nDrink2sSold = timesfired('tBReceiveDrink2')
end
nToiIn = timesfired('tToiIn')
nTIn = timesfired('tTIn')
nDfIn = timesfired('tDfIn')
nSecurity1 = timesfired('tIMCSecurity1')
nSecurity2 = timesfired('tIMCSecurity2')
nExit = timesfired('tIMCExit')
nWdExit = timesfired('tIMCWdExit')
nIMC = timesfired('tIMCCustDist')

%%Calculate Costs%%
%Drinks%%
TotCost.Drink1 = nDrink1sSold * Costs.Ingredients_Drink1
TotCost.Drink2 = nDrink2sSold * Costs.Ingredients_Drink2

%Employees%
TotCost.Employees = EmploymentHours*(Costs.EmployeeBarAssistant*nBarAssistant +
Costs.EmployeeBarback*nBarback + Costs.EmployeeBartender*nBartender +
Costs.EmployeeSecurity*nSecurityGuards)

%Fix%
TotCost.Fix = Costs.Fixed.Artists + Costs.Fixed.Electricity + Costs.Fixed.Water +
Costs.Fixed.Rent + Costs.Fixed.MusicFees + Costs.Fixed.Insurance +
Costs.Fixed.Maintainance + Costs.Fixed.ToiletPaper + Costs.Fixed.Soap +
Costs.Fixed.PaperTowels

OverallCost = TotCost.Drink1 + TotCost.Drink2 + TotCost.Employees + TotCost.Fix

%%Calculate Income%%
Income.EntranceFee = EntranceFee*nEntFeePayed
Income.WardrobeFee = WardrobeFee*nWdFeePayed
Income.Drink1 = PriceDrink1*nDrink1sSold
Income.Drink2 = PriceDrink2*nDrink2sSold

TotIncome = Income.EntranceFee + Income.WardrobeFee + Income.Drink1 +
Income.Drink2

Balance = TotIncome - OverallCost

%%%Visualize results%%%
figure(1), plotp(results,
{'pEntSecureCheckQue','pEntFreeCap','pEntSecureOk','pEntFeePayed'});
figure(2), plotp(results, {'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'});
figure(3), plotp(results, {'pDfDancing'});
figure(4), plotp(results, {'pTLookingForT', 'pTSittingAtT', 'pTOutBuffer'});
if global_info.BnOrderTerminals == 1
```

```
        figure(5), plotp(results, {'pBQue', 'pBWaitingForDrink', 'pBOutBuffer'});
elseif global_info.BnOrderTerminals == 2
        figure(5), plotp(results, {'pBQue',
'pBWaitingForDrink1','pBWaitingForDrink2', 'pBOutBuffer'});
end
figure(6), plotp(results, {'pBUsedGlasses','pBDirtyGlasses'});
figure(7), plotp(results, {'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce'});
figure(8), plotp(results, {'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'});
figure(9), plotp(results, {'pToiQue', 'pToi',
'pToiSinkQue','pToiSinks','pToiOutBuffer'});
plotGC(results)
```

BarV2.m (MultiRun, currently set up for case 5)

```
% BarV2

%%%Abrriviations:%%%
%B: Bar
%Cap: Capacity
%Cust: Customer
%Df: Dancefloor
%Ent: Entrance
%Gen: Generate
%IMC: Inter-Modular-Connector
%n: number of [...]
%NonAlc: Non-Alcoholic
%R: Resupply
%S: Storage
%T: Tables
%Toi: Toilets
%Wd: Wardrobe

clear all; clc;
global global_info

%%%Simulation Parameters%%%
global_info.START_AT = [22 0 0]; %OPTION: start simulations
global_info.STOP_AT  = [27 0 0]; %OPTION: stop  simulations
global_info.DELTA_TIME = 15; %in seconds
caseID = 5 %to identify case for multirun

%%%Bar Design%%%
global_info.TotalCap = 50;
global_info.EntSecureCheckQueCap = 20;
global_info.WdCap = 25;
global_info.BQueCap = 10;
global_info.BWaitingForDrinkCap = 5;
global_info.DfCap = 20;
global_info.TCap = 20;
global_info.ToiQueCap = 10;
global_info.ToiCap = 8;
global_info.ToiSinksCap = 4;

global_info.BnOrderTerminals = 2;
global_info.BnCounters = 2;
if global_info.BnOrderTerminals == 2
    global_info.BnCounters = 2;
end

%%%Employee numbers%%%
nBarAssistant = 1;
nBarback = 1;
nBartender = 1;
nSecurityGuards = 2;

%%%Costs%%%
%%Variable costs (per unit sold)%%
Costs.Ingredients_Drink1 = 25;
Costs.Ingredients_Drink2 = 25;
```

```
%%Employees costs (per employee, per hour)%%
Costs.EmployeeBarAssistant = 185;
Costs.EmployeeBarback = 185;
Costs.EmployeeBartender = 185;
Costs.EmployeeSecurity = 200;
EmploymentHours = 6;

%%Fixed costs (per night)%%
Costs.Fixed.Artists = 300*(EmploymentHours-1);
Costs.Fixed.Electricity = 2000;
Costs.Fixed.Water = 350;
Costs.Fixed.Rent = 5000;
Costs.Fixed.MusicFees = 250;
Costs.Fixed.Insurance = 500;
Costs.Fixed.Maintainance = 750;
Costs.Fixed.ToiletPaper = 100;
Costs.Fixed.Soap = 100;
Costs.Fixed.PaperTowels = 100;

%%%Income%%%
EntranceFee = 150;
WardrobeFee = 25;
PriceDrink1 = 100;
PriceDrink2 = 75;

%%%Customer Behaviour%%%
TimeBetweenCustArrivals = [0 10 0];
%Inhibitor factors, the higher the number (integer), the less likely a
%customer will visit that part of the bar
global_info.BInhibFact = 1;
global_info.TInhibFact = 4;
global_info.DfInhibFact = 4;
global_info.ToiInhibFact = 4;
global_info.ExitInhibFact = 4;

%%%Security Behaviour%%%
%Inhibitor factors, the higher the number (integer), the less likely a
%security guard will perform that action
global_info.SecurityCheckInhibFact = 50;
global_info.SecurityThrowOutInhibFact = 5;

%%%Simulation Setup%%%
pns =
pnstruct({'BarV2_Entrance_pdf','BarV2_IMC_pdf','BarV2_Dancefloor_pdf','BarV2_Table
s_pdf','BarV2_Bar_pdf','BarV2_Toilet_pdf'});
dyn.m0 = {'pEntFreeCap',global_info.TotalCap,... %Customer movement
          'pBCleanGlasses',200,'pBAlcohol',3000,'pBNonAlc',700,'pBIce',1000,...
%Initial Bar supplies [glasses and centiliter]
          'pBSCleanGlasses',100,'pBSAlcohol',6000,'pBSNonAlc',1500,'pBSIce',2000};
%Initial Storage supplies [glasses and centiliter]

if global_info.BnOrderTerminals == 1
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink',[0 1 0],'tIMCWdExit',[0 1
```

```
0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer
movement
                  'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                  'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink',[0 0 15],'tBReceiveDrink1',[0 1 0],'tBReceiveDrink2',[0 0
30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0
1 0],... %Customer movement
                  'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                  'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
elseif global_info.BnOrderTerminals == 2
    if global_info.BnCounters == 1
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink',[0 1
0],'tIMCWdExit',[0 1 0],'tEntWd',[0 2 0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1
0],... %Customer movement
                  'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                  'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    elseif global_info.BnCounters == 2
        dyn.ft = {'allothers',1,'tEntGenCust',TimeBetweenCustArrivals,
'tEntSecureCheck', [0 1 0], 'tEntFee', [0 0 30],'tDfOut',[0 10 0],'tTLeaveT',[0 10
0],'tBOrderDrink1',[0 0 15],'tBOrderDrink2',[0 0 15],'tBReceiveDrink1',[0 1
0],'tBReceiveDrink2',[0 0 30],'tIMCWdExit',[0 1 0],'tEntWd',[0 2
0],'tToiFromToi',[0 5 0],'tToiFromSink',[0 1 0],... %Customer movement
                  'tBCollectEGlasses',[0 2 0],'tBDishWasher',[0 3 0]... %Glasses
life cycle
                  'tBRCleanGlasses',[0 10 0],'tBRAlcohol',[0 5 0],'tBRNonAlc',[0 5
0],'tBRIce',[0 10 0],'tBSGenIce',[0 1 0]}; %Resupply ingredient
    end
end

dyn.re =
{'BarAssistant',nBarAssistant,inf,'Barback',nBarback,inf,'Bartender',nBartender,in
f,'SecurityGuard',nSecurityGuards,inf}; %Staff

nruns = 19
disp(clock)
disp('Run #')
for ind = 1:nruns
    disp(ind)
    %%%Run Simulation%%%
    pni = initialdynamics(pns, dyn);
    results = gpensim(pni);

    %%Count transition firerings%%
%     nEntIn = timesfired('tEntIn');
%     nEntGenCust = timesfired('tEntGenCust');
```

```
%     nEntSecureCheck = timesfired('tEntSecureCheck');
    nEntFeePayed = timesfired('tEntFee');
    nWdFeePayed = timesfired('tEntWd');
%     nEntOut = timesfired('tEntOut');
%     nBarIn = timesfired('tBIn');
    if global_info.BnOrderTerminals == 1
        nDrinksSold = timesfired('tBReceiveDrink');
    elseif global_info.BnOrderTerminals == 2
        nDrink1sSold = timesfired('tBReceiveDrink1');
        nDrink2sSold = timesfired('tBReceiveDrink2');
    end
%     nToiIn = timesfired('tToiIn');
%     nTIn = timesfired('tTIn');
%     nDfIn = timesfired('tDfIn');
%     nSecurity1 = timesfired('tIMCSecurity1');
%     nSecurity2 = timesfired('tIMCSecurity2');
%     nExit = timesfired('tIMCExit');
%     nWdExit = timesfired('tIMCWdExit');
%     nIMC = timesfired('tIMCCustDist');

    %%Calculate Costs%%
    %Drinks%%
    TotCost.Drink1 = nDrink1sSold * Costs.Ingredients_Drink1;
    TotCost.Drink2 = nDrink2sSold * Costs.Ingredients_Drink2;

    %Employees%
    TotCost.Employees = EmploymentHours*(Costs.EmployeeBarAssistant*nBarAssistant
+ Costs.EmployeeBarback*nBarback + Costs.EmployeeBartender*nBartender +
Costs.EmployeeSecurity*nSecurityGuards);

    %Fix%
    TotCost.Fix = Costs.Fixed.Artists + Costs.Fixed.Electricity +
Costs.Fixed.Water + Costs.Fixed.Rent + Costs.Fixed.MusicFees +
Costs.Fixed.Insurance + Costs.Fixed.Maintainance + Costs.Fixed.ToiletPaper +
Costs.Fixed.Soap + Costs.Fixed.PaperTowels;

    OverallCost = TotCost.Drink1 + TotCost.Drink2 + TotCost.Employees +
TotCost.Fix;

    %%Calculate Income%%
    Income.EntranceFee = EntranceFee*nEntFeePayed;
    Income.WardrobeFee = WardrobeFee*nWdFeePayed;
    Income.Drink1 = PriceDrink1*nDrink1sSold;
    Income.Drink2 = PriceDrink2*nDrink2sSold;

    TotIncome = Income.EntranceFee + Income.WardrobeFee + Income.Drink1 +
Income.Drink2;

    Balance = TotIncome - OverallCost;

    %%%Save results%%%
    %Existing results might be overwritten!
    xlswrite(strcat('results',num2str(caseID)), ind, 'Sheet1',
strcat('A',num2str(ind+1)));
    xlswrite(strcat('results',num2str(caseID)), Balance, 'Sheet1',
strcat('B',num2str(ind+1)));
```

```
    xlswrite(strcat('results',num2str(caseID)), nDrink1sSold, 'Sheet1',
strcat('C',num2str(ind+1)));
    xlswrite(strcat('results',num2str(caseID)), nDrink2sSold, 'Sheet1',
strcat('D',num2str(ind+1)));
    xlswrite(strcat('results',num2str(caseID)), nEntFeePayed, 'Sheet1',
strcat('E',num2str(ind+1)));
    xlswrite(strcat('results',num2str(caseID)), nWdFeePayed, 'Sheet1',
strcat('F',num2str(ind+1)));
    xlswrite(strcat('results',num2str(caseID)), clock, 'Sheet1',
strcat('G',num2str(ind+1)));
end
disp('done')

%%%Print Out results%%%
% prnschedule(results)
% % prncolormap(results,{'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'})
% % prnss(results)
%
% %%%Visualize results%%%
% figure(1), plotp(results,
{'pEntSecureCheckQue','pEntFreeCap','pEntSecureOk','pEntFeePayed'});
% figure(2), plotp(results, {'pIMCInBuff','pIMCOutBuff','pIMCFreeCap'});
% figure(3), plotp(results, {'pDfDancing'});
% figure(4), plotp(results, {'pTLookingForT', 'pTSittingAtT', 'pTOutBuffer'});
% if global_info.BnOrderTerminals == 1
%         figure(5), plotp(results, {'pBQue', 'pBWaitingForDrink',
'pBOutBuffer'});
% elseif global_info.BnOrderTerminals == 2
%         figure(5), plotp(results, {'pBQue',
'pBWaitingForDrink1','pBWaitingForDrink2', 'pBOutBuffer'});
% end
% figure(6), plotp(results, {'pBUsedGlasses','pBDirtyGlasses'});
% figure(7), plotp(results, {'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce'});
% figure(8), plotp(results,
{'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'});
% figure(9), plotp(results, {'pToiQue', 'pToi',
'pToiSinkQue','pToiSinks','pToiOutBuffer'});
% plotGC(results)
```

BarV2_Bar_pdf.m

```
% PDF for Bar Model of BarV2

function [png] = BarV2_Bar_pdf()
global global_info

png.PN_name = 'BarV2_Bar';

if global_info.BnOrderTerminals == 1
    png.set_of_Ps = {'pBQue', 'pBWaitingForDrink', 'pBOutBuffer',... %Customer
movement
            'pBUsedGlasses','pBDirtyGlasses',... %Glasses life cycle
            'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce',... %Ingredients
stored behind the bar
            'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'}; %Ingredients
resupply in storage room

    if global_info.BnCounters == 1
        png.set_of_Ts = {'tBIn', 'tBOrderDrink',
'tBBypass','tBReceiveDrink','tBOut',... %Customer movement
            'tBCollectEGlasses','tBDishWasher'... %Glasses life cycle
            'tBRCleanGlasses','tBRAlcohol','tBRNonAlc','tBRIce','tBSGenIce'};
%Resupply ingredients

        png.set_of_As =
{'tBIn','pBQue',1,'pBQue','tBOrderDrink',1,'tBOrderDrink','pBWaitingForDrink',1,'p
BWaitingForDrink','tBReceiveDrink',1,'tBReceiveDrink','pBOutBuffer',1,'pBQue',
'tBBypass',1,'tBBypass','pBOutBuffer',1,'pBOutBuffer','tBOut',1,... %Customer
movement

'tBReceiveDrink','pBUsedGlasses',1,'pBUsedGlasses','tBCollectEGlasses',5,'tBCollec
tEGlasses','pBDirtyGlasses',5,'pBDirtyGlasses','tBDishWasher',30,'tBDishWasher','p
BCleanGlasses',30,... %Glasses life cycle

'pBCleanGlasses','tBReceiveDrink',1,'pBAlcohol','tBReceiveDrink',4,'pBNonAlc','tBR
eceiveDrink',10,'pBIce','tBReceiveDrink',6,... %Ingredients for generic drink
[glasses and centiliter]

'pBSCleanGlasses','tBRCleanGlasses',24,'tBRCleanGlasses','pBCleanGlasses',24,...
%Resupply glasses from storage
            'pBSAlcohol','tBRAlcohol',210,'tBRAlcohol','pBAlcohol',210,...
%Resupply Alcohol from storage [centiliter]
            'pBSNonAlc','tBRNonAlc',500,'tBRNonAlc','pBNonAlc',500,... %Resupply
Non-Alcohol from storage [centiliter]
            'pBSIce','tBRIce',500,'tBRIce','pBIce',500,... %Resupply Ice from
storage [centiliter]
            'tBSGenIce','pBSIce',1,}; %Generate new Ice

    elseif global_info.BnCounters == 2
        png.set_of_Ts = {'tBIn', 'tBOrderDrink',
'tBBypass','tBReceiveDrink1','tBReceiveDrink2','tBOut',... %Customer movement
            'tBCollectEGlasses','tBDishWasher'... %Glasses life cycle
            'tBRCleanGlasses','tBRAlcohol','tBRNonAlc','tBRIce','tBSGenIce'};
%Resupply ingredients
```

```
        png.set_of_As =
{'tBIn','pBQue',1,'pBQue','tBOrderDrink',1,'tBOrderDrink','pBWaitingForDrink',1,'p
BWaitingForDrink','tBReceiveDrink1',1,'tBReceiveDrink1','pBOutBuffer',1,'pBWaiting
ForDrink','tBReceiveDrink2',1,'tBReceiveDrink2','pBOutBuffer',1,'pBQue',
'tBBypass',1,'tBBypass','pBOutBuffer',1,'pBOutBuffer','tBOut',1,... %Customer
movement

'tBReceiveDrink1','pBUsedGlasses',1,'tBReceiveDrink2','pBUsedGlasses',1,'pBUsedGla
sses','tBCollectEGlasses',5,'tBCollectEGlasses','pBDirtyGlasses',5,'pBDirtyGlasses
','tBDishWasher',30,'tBDishWasher','pBCleanGlasses',30,... %Glasses life cycle

'pBCleanGlasses','tBReceiveDrink1',1,'pBAlcohol','tBReceiveDrink1',4,'pBNonAlc','t
BReceiveDrink1',10,'pBIce','tBReceiveDrink1',6,... %Ingredients for generic drink1
[glasses and centiliter]

'pBCleanGlasses','tBReceiveDrink2',1,'pBAlcohol','tBReceiveDrink2',4,'pBNonAlc','t
BReceiveDrink2',10,'pBIce','tBReceiveDrink2',6,... %Ingredients for generic drink2
[glasses and centiliter]

'pBSCleanGlasses','tBRCleanGlasses',24,'tBRCleanGlasses','pBCleanGlasses',24,...
%Resupply glasses from storage
            'pBSAlcohol','tBRAlcohol',210,'tBRAlcohol','pBAlcohol',210,...
%Resupply Alcohol from storage [centiliter]
            'pBSNonAlc','tBRNonAlc',500,'tBRNonAlc','pBNonAlc',500,... %Resupply
Non-Alcohol from storage [centiliter]
            'pBSIce','tBRIce',500,'tBRIce','pBIce',500,... %Resupply Ice from
storage [centiliter]
            'tBSGenIce','pBSIce',1,}; %Generate new Ice
    end

    png.set_of_Is =
{'pBQue','tBIn',global_info.BQueCap,'pBWaitingForDrink','tBOrderDrink',global_info
.BWaitingForDrinkCap,... %Max capacities customer movemement

'pBCleanGlasses','tBRCleanGlasses',10,'pBAlcohol','tBRAlcohol',1000,'pBNonAlc','tB
RNonAlc',300,'pBIce','tBRIce',500,'pBSIce','tBSGenIce',2000}; %When to refill bar
supplies

elseif global_info.BnOrderTerminals == 2
        png.set_of_Ps = {'pBQue', 'pBWaitingForDrink1','pBWaitingForDrink2',
'pBOutBuffer',... %Customer movement
            'pBUsedGlasses','pBDirtyGlasses',... %Glasses life cycle
            'pBCleanGlasses','pBAlcohol','pBNonAlc','pBIce',... %Ingredients
stored behind the bar
            'pBSCleanGlasses','pBSAlcohol','pBSNonAlc','pBSIce'}; %Ingredients
resupply in storage room

        png.set_of_Ts = {'tBIn', 'tBOrderDrink1','tBOrderDrink2',
'tBBypass','tBReceiveDrink1','tBReceiveDrink2','tBOut',... %Customer movement
            'tBCollectEGlasses','tBDishWasher'... %Glasses life cycle
            'tBRCleanGlasses','tBRAlcohol','tBRNonAlc','tBRIce','tBSGenIce'};
%Resupply ingredients

        png.set_of_As =
{'tBIn','pBQue',1,'pBQue','tBOrderDrink1',1,'tBOrderDrink1','pBWaitingForDrink1',1
,'pBQue','tBOrderDrink2',1,'tBOrderDrink2','pBWaitingForDrink2',1,'pBWaitingForDri
nk1','tBReceiveDrink1',1,'tBReceiveDrink1','pBOutBuffer',1,'pBWaitingForDrink2','t
```

```
BReceiveDrink2',1,'tBReceiveDrink2','pBOutBuffer',1,'pBQue',
'tBBypass',1,'tBBypass','pBOutBuffer',1,'pBOutBuffer','tBOut',1,... %Customer
movement

'tBReceiveDrink1','pBUsedGlasses',1,'tBReceiveDrink2','pBUsedGlasses',1,'pBUsedGla
sses','tBCollectEGlasses',5,'tBCollectEGlasses','pBDirtyGlasses',5,'pBDirtyGlasses
','tBDishWasher',30,'tBDishWasher','pBCleanGlasses',30,... %Glasses life cycle

'pBCleanGlasses','tBReceiveDrink1',1,'pBAlcohol','tBReceiveDrink1',4,'pBNonAlc','t
BReceiveDrink1',10,'pBIce','tBReceiveDrink1',6,... %Ingredients for generic drink1
[glasses and centiliter]

'pBCleanGlasses','tBReceiveDrink2',1,'pBAlcohol','tBReceiveDrink2',40,'pBNonAlc','
tBReceiveDrink2',0,'pBIce','tBReceiveDrink2',0,... %Ingredients for generic drink2
[glasses and centiliter]

'pBSCleanGlasses','tBRCleanGlasses',24,'tBRCleanGlasses','pBCleanGlasses',24,...
%Resupply glasses from storage
            'pBSAlcohol','tBRAlcohol',210,'tBRAlcohol','pBAlcohol',210,...
%Resupply Alcohol from storage [centiliter]
            'pBSNonAlc','tBRNonAlc',500,'tBRNonAlc','pBNonAlc',500,... %Resupply
Non-Alcohol from storage [centiliter]
            'pBSIce','tBRIce',500,'tBRIce','pBIce',500,... %Resupply Ice from
storage [centiliter]
            'tBSGenIce','pBSIce',1,}; %Generate new Ice

    png.set_of_Is =
{'pBQue','tBIn',global_info.BQueCap,'pBWaitingForDrink1','tBOrderDrink1',global_in
fo.BWaitingForDrinkCap,'pBWaitingForDrink2','tBOrderDrink2',global_info.BWaitingFo
rDrinkCap... %Max capacities customer movemement

'pBCleanGlasses','tBRCleanGlasses',10,'pBAlcohol','tBRAlcohol',1000,'pBNonAlc','tB
RNonAlc',300,'pBIce','tBRIce',500,'pBSIce','tBSGenIce',2000}; %When to refill bar
supplies
end

png.set_of_Ports = {'tBIn','tBOut'};
```

BarV2_Dancefloor_pdf.m

```
% PDF for Dancefloor Model of BarV2

function [png] = BarV2_Dancefloor_pdf()

global global_info
png.PN_name = 'BarV2_Dancefloor';

png.set_of_Ps = {'pDfDancing'};
png.set_of_Ts = {'tDfIn','tDfOut'};
png.set_of_As = {'tDfIn','pDfDancing',1,'pDfDancing','tDfOut',1};
png.set_of_Is = {'pDfDancing','tDfIn',global_info.DfCap}; %Max Dancefloor capacity
png.set_of_Ports =  {'tDfIn','tDfOut'};
```

BarV2_Entrance_pdf.m

```
% PDF for Entrance Model of BarV2

function [png] = BarV2_Entrance_pdf()

global global_info;

png.PN_name = 'BarV2_Entrance';

png.set_of_Ps = {'pEntFreeCap', 'pEntSecureCheckQue',
'pEntSecureOk','pEntFeePayed'};
png.set_of_Ts = {'tEntIn', 'tEntOut',
'tEntGenCust','tEntSecureCheck','tEntFee','tEntWd'};
png.set_of_As =
{'tEntIn','pEntFreeCap',1,'pEntFreeCap','tEntSecureCheck',1,'tEntGenCust','pEntSec
ureCheckQue',1,'pEntSecureCheckQue','tEntSecureCheck',1,'tEntSecureCheck','pEntSec
ureOk',1,'pEntSecureOk','tEntFee',1,'tEntFee','pEntFeePayed',1,'pEntFeePayed','tEn
tWd',1,'tEntWd','pEntFeePayed',1,'pEntFeePayed','tEntOut',1};
png.set_of_Is =
{'pEntSecureCheckQue','tEntGenCust',global_info.EntSecureCheckQueCap,'pEntFreeCap'
,'tEntIn',global_info.TotalCap}; %Max entrance and total capacity
png.set_of_Ports = {'tEntIn','tEntOut'};
```

BarV2_IMC_pdf.m

```
% PDF for IMC of BarV2

function [png] = BarV2_IMC_pdf()

png.PN_name = 'BarV2_IMC';

png.set_of_Ps = {'pIMCInBuff', 'pIMCOutBuff', 'pIMCFreeCap'};
png.set_of_Ts = {'tIMCCustDist',
'tIMCExit','tIMCWdExit','tIMCSecurity1','tIMCSecurity2'};
png.set_of_As =
{'tEntOut','pIMCInBuff',1,'pIMCInBuff','tIMCCustDist',1,'tIMCCustDist','pIMCOutBuf
f',1,'pIMCOutBuff','tIMCExit',1,'tIMCExit','pIMCFreeCap',1,'pIMCOutBuff','tIMCWdEx
it',1,'tIMCWdExit','pIMCFreeCap',1,'pIMCFreeCap','tEntIn',1,'pIMCOutBuff','tDfIn',
1,'tDfOut','pIMCInBuff',1,'pIMCOutBuff','tTIn',1,'tTOut','pIMCInBuff',1,'pIMCOutBu
ff','tBIn',1,'tBOut','pIMCInBuff',1,'pIMCOutBuff','tToiIn',1,'tToiOut','pIMCInBuff
',1,'pIMCOutBuff','tIMCSecurity1',1,'pIMCOutBuff','tIMCSecurity2',1,'tIMCSecurity1
','pIMCOutBuff',1,'tIMCSecurity2','pIMCOutBuff',1};
```

BarV2_Tables_pdf.m

```
% PDF for Tables Model of BarV2

function [png] = BarV2_Tables_pdf()

global global_info
png.PN_name = 'BarV2_Tables';

png.set_of_Ps = {'pTLookingForT', 'pTSittingAtT', 'pTOutBuffer'};
png.set_of_Ts = {'tTIn', 'tTSitDown','tTBypassT','tTLeaveT','tTOut'};
png.set_of_As =
{'tTIn','pTLookingForT',1,'pTLookingForT','tTSitDown',1,'tTSitDown','pTSittingAtT'
,1,'pTSittingAtT','tTLeaveT',1,'tTLeaveT','pTOutBuffer',1,'pTLookingForT','tTBypas
sT',1,'tTBypassT','pTOutBuffer',1,'pTOutBuffer','tTOut',1};
png.set_of_Is = {'pTSittingAtT','tTSitDown',global_info.TCap}; %Max Table Capacity
png.set_of_Ports = {'tTIn','tTOut'};
```

BarV2_Toilet_pdf.m

```
% PDF for Toilet Model of BarV2

function [png] = BarV2_Toilet_pdf()
global global_info

png.PN_name = 'BarV2_Toilet';

png.set_of_Ps = {'pToiQue', 'pToi', 'pToiSinkQue','pToiSinks','pToiOutBuffer'};
png.set_of_Ts = {'tToiIn', 'tToiToToi',
'tToiBypass','tToiFromToi','tToiBypassSink','tToiToSink','tToiFromSink','tToiOut'}
;
png.set_of_As =
{'tToiIn','pToiQue',1,'pToiQue','tToiToToi',1,'tToiToToi','pToi',1,'pToi','tToiFro
mToi',1,'tToiFromToi','pToiSinkQue',1,'pToiSinkQue','tToiToSink',1,'tToiToSink','p
ToiSinks',1,'pToiSinks','tToiFromSink',1,'tToiFromSink','pToiOutBuffer',1,'pToiOut
Buffer','tToiOut',1,'pToiSinkQue','tToiBypassSink',1,'tToiBypassSink','pToiOutBuff
er',1,'pToiOutBuffer','tToiOut',1,'pToiOutBuffer','tToiOut',1,'pToiQue','tToiBypas
s',1,'tToiBypass','pToiOutBuffer',1,'pToiOutBuffer','tToiOut',1};
png.set_of_Is =
{'pToiQue','tToiIn',global_info.ToiQueCap,'pToi','tToiToToi',global_info.ToiCap,'p
ToiSinks','tToiToSink',global_info.ToiSinksCap}; %Max Toilette que, toilet and
sink capacities
png.set_of_Ports = {'tToiIn','tToiOut'};
```

COMMON_POST.m

```
function [] = COMMON_POST (transition)
release(); %release all resources after firing
```

COMMON_PRE.m

```
function [fire, transition] = COMMON_PRE (transition)
global global_info

if strcmp(transition.name, 'tIMCExit') %Firing conditions for exiting the bar
below
    if randi([1,global_info.ExitInhibFact])==1 %Random probability of firing,
based on inhibitor factor
        tokIDExit = tokenWOAnyColor('pIMCOutBuff',1,{'JDisposed'}); %Only take
customers who have not disposed their jacket at the wardrobe
        if tokIDExit == 0
            fire = 0;
        else
            transition.selected_tokens = tokIDExit;
            transition.override = 1;
            transition.new_color = {}; %Reset colors
            fire = tokIDExit;
        end
    else
        fire = 0;
    end
elseif strcmp(transition.name, 'tIMCWdExit') %Firing conditions for exiting the
bar through the wardrobe below
    if randi([1,global_info.ExitInhibFact])==1 %Random probability of firing,
based on inhibitor factor
        tokIDWdExit = tokenAnyColor('pIMCOutBuff',1,{'JDisposed'}); %Only take
customers who have disposed their jacket at the wardrobe
        if tokIDWdExit == 0
            fire = 0;
        else
            if timesfired('tEntWd') - timesfired('tIMCWdExit') >0 %Make sure the
number of jackets given out at the wardrobe does not exceed the number of jackets
received
                if requestSR({'BarAssistant',1}) == 1 %Request one bar assistant
to handle the jacket return
                    transition.selected_tokens = tokIDWdExit;
                    transition.override = 1;
                    transition.new_color = {}; %Reset colors
                    fire = tokIDWdExit;
                else
                    fire = 0;
                end
            else
                fire = 0;
            end
        end
    else
        fire = 0;
    end
elseif strcmp(transition.name, 'tIMCSecurity1') || strcmp(transition.name,
'tIMCSecurity2')
    if requestSR({'SecurityGuard',2}) == 1
        if randi([1,global_info.SecurityCheckInhibFact]) == 1
            tokIDSecurityCheck = tokenAny('pIMCOutBuff',1);
            transition.selected_tokens = tokIDSecurityCheck;
            transition.override = 0;
```

```
            if randi([1,global_info.SecurityThrowOutInhibFact])==1 %Randomly throw
out troublesome customers
                transition.new_color = {'Exit'};
                fire = tokIDSecurityCheck;
            else
                fire = tokIDSecurityCheck;
            end
        else
            fire = 0;
        end
    else
        fire = 0;
    end
elseif strcmp(transition.name, 'tToiIn')
    if randi([1,global_info.ToiInhibFact])==1
        tokIDIn = tokenWOAnyColor('pIMCOutBuff',1,{'Exit'}); %Troublesome
customers that have been thrown out by security are not allowed to enter other
parts of the bar und must exit
        if tokIDIn == 0
            fire = 0;
        else
            fire = tokIDIn;
        end
    else
        fire = 0;
    end
elseif strcmp(transition.name, 'tBIn')
    if randi([1,global_info.BInhibFact])==1
        tokIDIn = tokenWOAnyColor('pIMCOutBuff',1,{'Exit'});
        if tokIDIn == 0
            fire = 0;
        else
            fire = tokIDIn;
        end
    else
        fire = 0;
    end
elseif strcmp(transition.name, 'tTIn')
    if randi([1,global_info.TInhibFact])==1
        tokIDIn = tokenWOAnyColor('pIMCOutBuff',1,{'Exit'});
        if tokIDIn == 0
            fire = 0;
        else
            fire = tokIDIn;
        end
    else
        fire = 0;
    end
elseif strcmp(transition.name, 'tDfIn')
    if randi([1,global_info.DfInhibFact])==1
        tokIDIn = tokenWOAnyColor('pIMCOutBuff',1,{'Exit'});
        if tokIDIn == 0
            fire = 0;
        else
            fire = tokIDIn;
        end
    else
```

```
        fire = 0;
    end
else
    fire = 1;
end
```

MOD_BarV2_Bar_POST.m

```
function [] = MOD_BarV2_Bar_POST (transition)
release(); %Release all resources after firing
```

MOD_BarV2_Bar_PRE.m

```
function [fire, transition] = MOD_BarV2_Bar_PRE (transition)
global global_info
fire = 1;

if strcmp(transition.name, 'tBBypass')
    if ntokens('pBQue') <global_info.BQueCap
        fire = 0; %Bypass only enabled when que is full
    end
%%%Request required bar staff respectively%%%
elseif strcmp(transition.name, 'tBOrderDrink')
    fire = requestSR({'Bartender',1});
elseif strcmp(transition.name, 'tBReceiveDrink')
    fire = requestSR({'Bartender',1});
elseif strcmp(transition.name, 'tBOrderDrink1')
    fire = requestSR({'Bartender',1});
elseif strcmp(transition.name, 'tBReceiveDrink1')
    fire = requestSR({'Bartender',1});
elseif strcmp(transition.name, 'tBOrderDrink2')
    fire = requestSR({'Bartender',1});
elseif strcmp(transition.name, 'tBReceiveDrink2')
    fire = requestSR({'Bartender',1});
elseif strcmp(transition.name, 'tBCollectEGlasses')
    fire = requestSR({'Barback',1});
elseif strcmp(transition.name, 'tBDishWasher')
    fire = requestSR({'Barback',1});
elseif strcmp(transition.name, 'tBRCleanGlasses')
    fire = requestSR({'Barback',1});
elseif strcmp(transition.name, 'tBRAlcohol')
    fire = requestSR({'Barback',1});
elseif strcmp(transition.name, 'tBRNonAlc')
    fire = requestSR({'Barback',1});
elseif strcmp(transition.name, 'tBRIce')
    fire = requestSR({'Barback',1});
end
```

MOD_BarV2_Entrance_POST.m

```
function [] = MOD_BarV2_Entrance_POST (transition)
release(); %Release all resources after firing
```

MOD_BarV2_Entrance_PRE.m

```
function [fire, transition] = MOD_BarV2_Entrance_PRE (transition)

global global_info
fire = 1;

%%%Request the required bar staff respectively%%%
if strcmp(transition.name, 'tEntSecureCheck')
    fire = requestSR({'SecurityGuard',2});
elseif strcmp(transition.name, 'tEntFee')
    fire = requestSR({'BarAssistant',1});
elseif strcmp(transition.name, 'tEntWd')
    if timesfired('tEntWd') - timesfired('tIMCWdExit') < global_info.WdCap %Make
sure that the number of jackets in the wardrobe does not exceed it´s capacity
        tokIDAddJ = tokenAnyColor('pEntFeePayed',1,{'Jacket'}); %Only take
customers that have a jacket to dispose
        if tokIDAddJ == 0
            fire=0;
        else
            if requestSR({'BarAssistant',1}) == 1
                transition.selected_tokens = tokIDAddJ;
                transition.override = 1;
                transition.new_color = {'JDisposed'}; %Mark customer to having
their jacket disposed at the wardrobe
                fire = tokIDAddJ;
            else
                fire = 0;
            end
        end
    else
        fire = 0;
    end
elseif strcmp(transition.name, 'tEntOut')
    if timesfired('tEntWd') - timesfired('tIMCWdExit') < global_info.WdCap %Unless
the wardrobe is full, take only customers without a jacket
        tokIDOut = tokenWOAnyColor('pEntFeePayed',1,{'Jacket'});
        if tokIDOut == 0
            fire=0;
        else
            fire = tokIDOut;
        end
    else
        tokIDOut = tokenAny('pEntFeePayed',1); %If wardrobe is full, take all
customers
        if tokIDOut == 0
            fire=0;
        else
            transition.selected_tokens = tokIDOut;
            transition.override = 1;
            transition.new_color = {};
            fire = tokIDOut;
        end
    end
elseif strcmp(transition.name, 'tEntGenCust')
    jacket = randi([1, 5]); %randomly generate customers with or without jacket
    if jacket == 1
```

```
        transition.new_color = 'Jacket';
    end
end
```

MOD_BarV2_Tables_PRE.m

```
function [fire, transition] = MOD_BarV2_Tables_PRE (transition)
global global_info
fire = 1;

if ntokens('pTSittingAtT') <global_info.TCap
    if strcmp(transition.name, 'tTBypassT')
        fire = 0; %Only enable bypass if the tables are full
    end
end
```

MOD_BarV2_Toilet_PRE.m

```
function [fire, transition] = MOD_BarV2_Toilet_PRE (transition)
global global_info
fire = 1;

if ntokens('pToiQue') <global_info.ToiQueCap
    if strcmp(transition.name, 'tToiBypass')
        fire = 0; %Only enable bypass if the toilet que is full
    end
end
```

Python Code for Histograms

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
Case = [0]
ncases = 7
for casenum in range(1,ncases+1):

Case.append(pd.read_excel('./CombinedResults.xlsx',sheet_name=str('Case'+str(casen
um))))
Case[1].describe()
for casenum in range(1,ncases+1):
    print('Case '+str(casenum))
    print('Number of runs:',Case[casenum]['Balance'].count())
    print('Balance mean:',np.round(Case[casenum]['Balance'].mean(),2))
    print('Balance std:',np.round(Case[casenum]['Balance'].std(),2))
    plt.figure(figsize=(10,6))
    plt.hist(Case[casenum]['Balance'])
    plt.xlabel('NOK')
    plt.ylabel('count')
    plt.title('Case '+str(casenum)+' Histogram on Balance')
    plt.show()
```