Simulation Assignment1

Sunny Hsieh (604455), Francisco Matias (624427)

January 2022

Question 1

a

Relevant state variables

Description	Variable
How many booths are busy	nServersBusy
How many people are standing in queue for booths	nServesQueue
How many chairs are taken	nChairsBusy
How many people are waiting for a chair	nChairQueue

Counter variables

Description	Variable
Total amount of people who had to wait for a chair	cumNoAvailableChair
Total time spend in waiting for a chair of all people who waited	cumQueueChair
Total arrivals	arrivals
Total departures	departures

Events

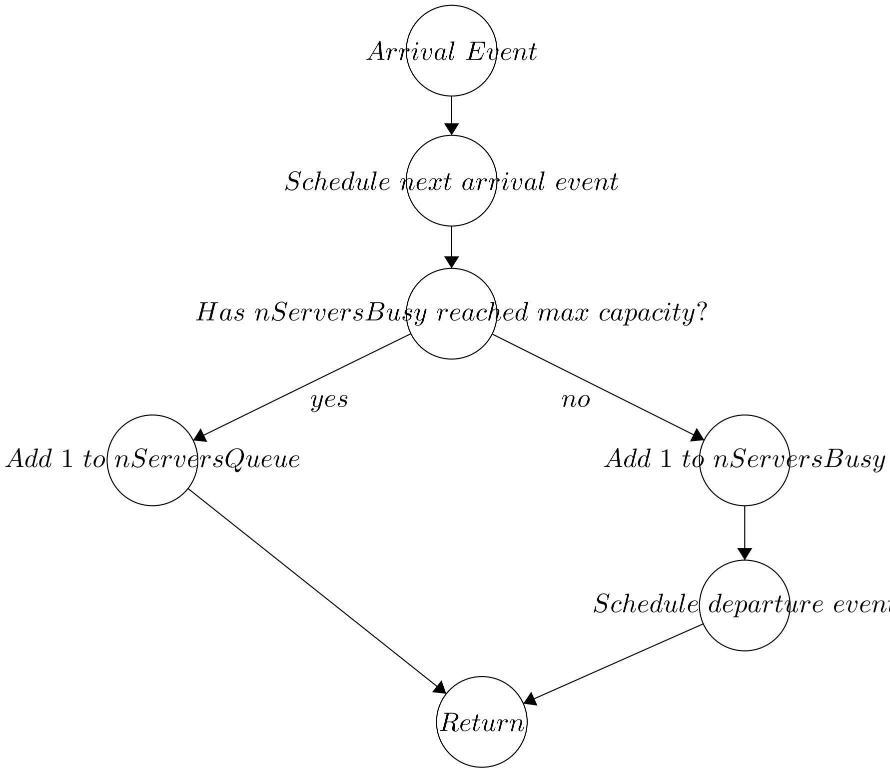
Description	Variable
Arrival time of next person	nextArrivalTime = eventTime + nextInterArrivalTime
Departure time after served by the booth	departure time = eventTime + ServiceDuration
Leaving time after seated in chair	eventTime + chairSittingTime

How all variables are updated and how the simulation is initialized and for each event when it is generated

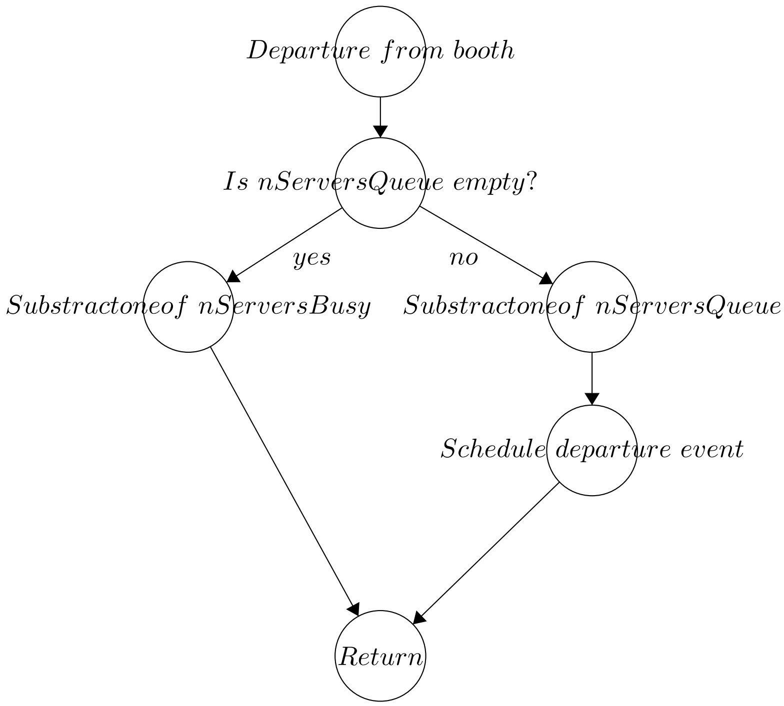
Variable (state)	**How it is updated**
nServersBusy	If someone enters the booth: nServersBusy++ If someone is done with the service time: nServersBusy–
nServersQueue	If nServersBusy = max capacity and someone arrives: nServersQueue++ If nServersBusy <max capacity and nServersQueue >0 : nServersQueue–
nChairsBusy	After someone is served by booth, if still a chair free: nChairBusy++ If someone has seated for 15 min: nChairBusy–
nChairQueue	After someone is served by booth, if no chairs are free: nChairQueue++ If a chair is free and nChairQueue >0: nChairQueue–
Variable (Counter)	**How it is updated**
cumNoAvailableChair	For each person who is added to the nChairQueue: cumNoAvailableChair++
cumQueueChair	Time between t_0 and t_1 times * nChairQueue: (eventTime - getCurrentTime()) * this.nChairQueue
arrivals	Each person enters system: arrivals++
departures	Each person leaves system: departures++

We also provide the flowchart. Note that keep track of the variable arrivals (we want to do so in our code to make it ourselves more structural), even though it is fixed. For departures, we did include this in the flowchart.

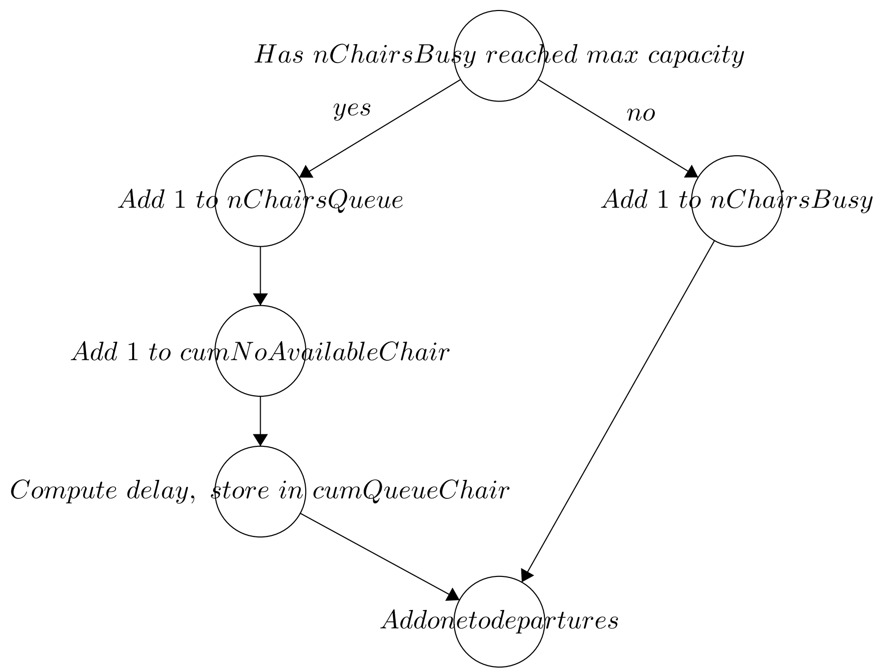
First the flowchart for arrival routine:



and the departure flowchart:



and the flowchart for arriving chairs



The flowchart for departure of chairs is not necessary since you will leave the system when you finish sitting in a chair.

How the required performance measures can be calculated.

The GGD is interested in the following things:

1) How many people they can maximally vaccinate each day while ensuring that the expected probability that someone cannot find an empty chair after getting vaccinated is at most 1%.

The expected probability that someone cannot find an empty chair after getting vaccinated can be calculated by

$$cumNoAvailableChair/arrivals$$

furthermore, the total people who get vaccinated each day is just the variable departures.

2) The expected waiting time it takes before a chair becomes available if someone has to wait.

This can be calculated by

$$cumQueueChair/cumNoAvailableChair$$

Note, that this can't be calculated if cumNoAvailableChair is equal to zero. We did take care of that in an if-else statement. So if cumNoAvailableChair equals zero we will set the expected waiting time to zero as well, else we will use the formula cumQueueChair/cumNoAvailableChair.

3) The expected time the last person leaves for staff scheduling purposes.

This can be get by getting the departure event of the last person. We keep track of when a departure takes place, so when someone leaves the sytem. The last person who leaves the system will be stored in the variable lastPersonLeftTime. Moreover we will calculate the expected time the last person leaves by summing over all lastPersonLeftTime of n simulations and then divide it by n.

b

See code.

c

Results

First we will show our results:

nBooths: 1
Arrivals: 86.081 (0.17810716201534774)
cumNoAvailableChair: 0.0 (0.0)
cumQueueTime: 0.0 (0.0)
if no chair time is: 0.0 (0.0)
last person arrived at: 8.560297742195896 (0.016440967435515507)
last person left at: 9.260854723181493 (0.016909312063363074)
p: 0.0 (0.0)

---

nBooths: 2
Arrivals: 174.406 (0.25244978687829217)
cumNoAvailableChair: 0.0 (0.0)
cumQueueTime: 0.0 (0.0)
if no chair time is: 0.0 (0.0)
last person arrived at: 8.724037615420121 (0.01094275360249096)
last person left at: 9.262713474384858 (0.011357548121133328)
p: 0.0 (0.0)

---

nBooths: 3

Arrivals: 263.206 (0.3143793860901157)
cumNoAvailableChair: 0.0 (0.0)
cumQueueTime: 0.0 (0.0)
if no chair time is: 0.0 (0.0)
last person arrived at: 8.762734835532433 (0.009647849118400496)
last person left at: 9.276720478191013 (0.009973151012327125)
p: 0.0 (0.0)

---

nBooths: 4
Arrivals: 352.157 (0.3617806465750146)
cumNoAvailableChair: 0.03 (0.013611953764441239)
cumQueueTime: 5.990841572149068E-4 (4.750829362568527E-4)
if no chair time is: 1.0132356533185158E-4 (4.538360394136697E-5)
last person arrived at: 8.785187066556711 (0.008864698089180131)
last person left at: 9.286555724180987 (0.009157461437391198)
p: 8.348648253571225E-5 (3.7823296914023824E-5)

---

nBooths: 5
Arrivals: 441.119 (0.4287991461135509)
cumNoAvailableChair: 0.477 (0.05559281933452806)
cumQueueTime: 0.008172042900827552 (0.0012648006339506974)
if no chair time is: 0.0015593238644040187 (1.6945055571838328E-4)
last person arrived at: 8.812614575763032 (0.007517881162120404)
last person left at: 9.30767775839885 (0.008034004936598144)
p: 0.0010697180864703297 (1.2456101730706992E-4)

---

nBooths: 6
Arrivals: 529.864 (0.4506512960045635)
cumNoAvailableChair: 5.139 (0.22269996469302378)
cumQueueTime: 0.0993060128812217 (0.006046441870412483)
if no chair time is: 0.009281499655487404 (3.351372777374808E-4)
last person arrived at: 8.839947937957577 (0.00699829715944735)
last person left at: 9.336568318457061 (0.0076817025756753185)
p: 0.009610483641416727 (4.147678725465407E-4)

---

nBooths: 7
Arrivals: 619.581 (0.46248819912980693)
cumNoAvailableChair: 30.424 (0.6156421356918641)
cumQueueTime: 0.6781798242076335 (0.018654814906027356)
if no chair time is: 0.01962550531904376 (2.84988930118502E-4)
last person arrived at: 8.855782383627044 (0.006343147791874547)
last person left at: 9.347504562800657 (0.00695020476148214)

p: 0.048796272368671324 (9.765250883368628E-4)

---

nBooths: 8
Arrivals: 708.479 (0.5145019643026347)
cumNoAvailableChair: 102.281 (1.1981843283421314)
cumQueueTime: 2.669414836710151 (0.04254694422362355)
if no chair time is: 0.025267524614949074 (2.1253266965732203E-4)
last person arrived at: 8.865465692133103 (0.005985783925646884)
last person left at: 9.374148557115719 (0.006639198886624208)
p: 0.14364046884473433 (0.0016321671443815283)

---

nBooths: 9
Arrivals: 798.172 (0.5517724792601297)
cumNoAvailableChair: 245.643 (1.8220613212673322)
cumQueueTime: 7.613078864252519 (0.0791594906858977)
if no chair time is: 0.0305395295710213 (1.6084762190443495E-4)
last person arrived at: 8.86821856010933 (0.005648160620519521)
last person left at: 9.373788058419395 (0.006564657756970301)
p: 0.3067069024307302 (0.002150844076751915)

---

nBooths: 10
Arrivals: 888.292 (0.5449915153025322)
cumNoAvailableChair: 447.079 (2.083741523735068)
cumQueueTime: 16.692819052243628 (0.1166534137752881)
if no chair time is: 0.03708904581360627 (1.44615703220467E-4)
last person arrived at: 8.88116701880683 (0.00508653253911462)
last person left at: 9.39187662098845 (0.006083253548035795)
p: 0.5022835912154713 (0.0021266048578341165)

Results in a table

Now we will summarize the values we are interested in a table where the values are rounded:

Amount of booths open	Expected people who gets vaccinated	Expected probability someone can't find a chair	Expected waiting time it takes before a chair becomes available if someone has to wait (in seconds)	Expected time the last person leaves (converted in time)
1	86	0	x	17:16
2	174	0	x	17:16
3	263	0	x	17:17
4	352	0.000083	0.36 seconds	17:17
5	441	0.0011	5,6 seconds	17:18
6	530	0.0096	33 seconds	17:20
7	620	0.049	71 seconds	17:21
8	708	0.14	90 seconds	17:22
9	798	0.31	110 seconds	17:22
10	888	0.5	133 seconds	17:23

Note that we put an x for the expected waiting time it takes before a chair becomes available if someone has to wait. That is because there were no people who had to wit at all. In that case you can't tell what the expected wating time is if the event someone has to wait never happens.

Give a recommendation to the GGD regarding the number of booths they can open

Now let's look at what the GGD is interested in:

1) How many people they can maximally vaccinate each day while ensuring that the expected probability that someone cannot find an empty chair after getting vaccinated is at most 1%.

We can look at the results above and see when the expected probability that someone cannot find an empty chair after getting vaccinated exceeds 1%, so 0.01. We notice that that happens when the amount of booths is equal to 7. So our recommendation is to open 6 booths.

2) The expected waiting time it takes before a chair becomes available if someone has to wait.

Since we recommend to open 6 booths, we see that the expected waiting time it takes before a chair becomes available if someone has to wait is ≈ 33 seconds.

3) The expected time the last person leaves for staff scheduling purposes.

For opening 6 booths, the expected time the last person leaves is at $\approx 17:20$, so we would suggest the GGD to schedule their staff until 17 : 30, since 17 : 20 is the expected value so there might be days were when the last person leaves might exceed 17:20. Also, it's common to schedule staff in half hours, so 17 : 00, 17 : 30 or 18 : 00 (or even quarter hours but that would result in the same suggestion) and here 17 : 30 would be the best suggestion based on our simulation.

We summarize our suggestion in the following table

Amount of booths open	**Expected people who gets vaccinated**	**Expected probability someone can't find a chair**	**Expected waiting time it takes before a chair becomes available if someone has to wait (in seconds)**	**Suggested ending time for staff scheduling**
6	530	0.0096	33 seconds	17:30

Question 2

a

Relevant state variables

In this case there are not really state variables but we will list out the variables we defined

Description	**Variable**
The amount of garbage when it will be sensored	sensor
The time between sensored and garbage thrown away	timeDelay
The cost of emptying the container	costContainer
The cost of leaving a garbage next to container	costOutsideBag
The max capacity the container can take	maxCapacity

Counter variables

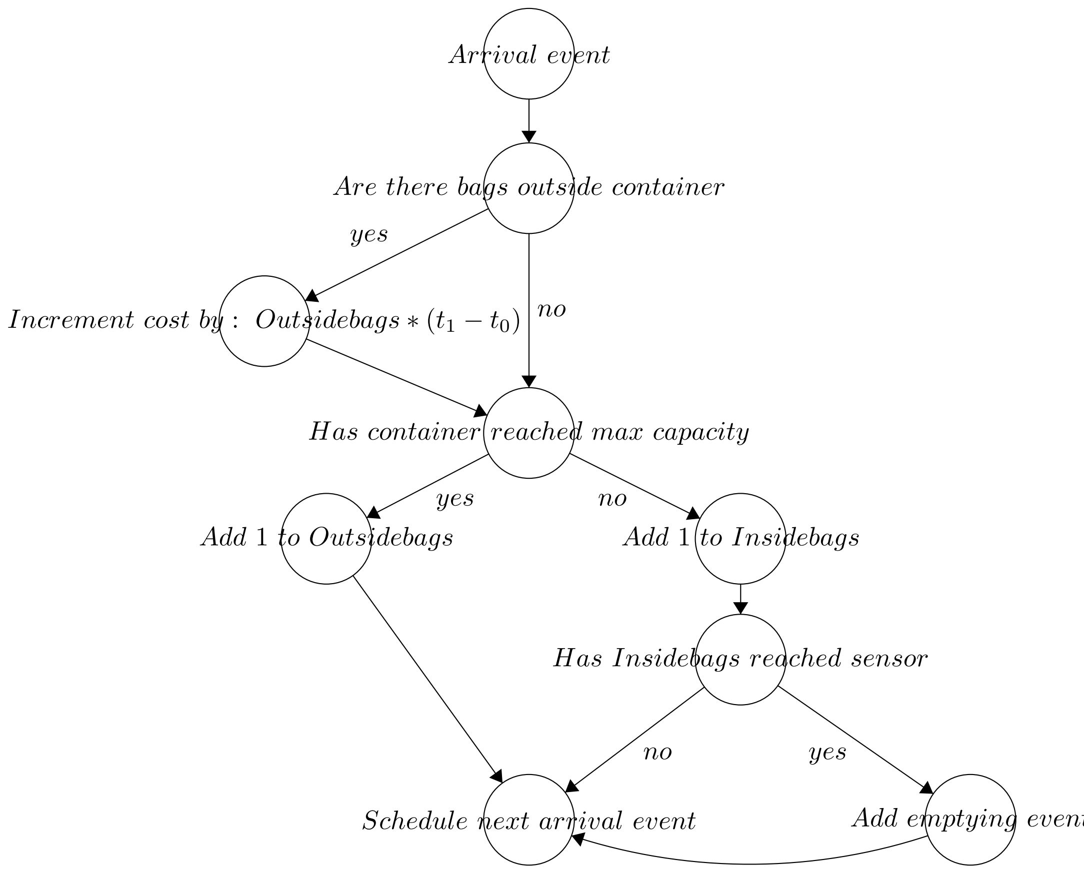
Description	**Variable**
Amount of bags placed outside the container	outsidebags
Amount of bags placed outside the container	insidebags
Amount of costs	costs
The expired time	time

Events

Event	**Description**
nextArrivalTime,eventTime plus nextInterArrivalTime	Arrival of next person with garbage
Emptying container, when sensored plus delay it takes to empty	eventTime + timeDelay

How all variables are updated and how the simulation is initialized and for each event when it is generated

The flowchart is as follows



How the required performance measures can be calculated.

We are interested in the minimizing the total costs. So we need to measure the total cost. We kept track of the cost by increasing it when there were bags placed outside the container. So let's take the next arrival event as t_1 and current time as t_0, then the cost during timeframe $t_0 - t_1$ can be calculated by

$$amountofbagsoutisde * (t_1 - t_0)$$

Furthermore, everytime the event of emptying the container take place we will add a costs of one hundred.

At the end we can directly call a method "getCosts()" and the same for the time horizon by calling "time.getValue()". So we get yearly costs by:

$$\frac{Totalcosts}{Totaltime} * 24 * 365$$

since total time is in hours.

b

See code.

c

Results

First we will show our results:

SensorLevel: 850
Cost: 100.0 (0.0)
Time till clean up (days): 19.72413854917633 (0.020172670083989538)
Yearly Cost: 1852.4579602605443 (1.89442108737607)

SensorLevel: 855
Cost: 100.0 (0.0)
Time till clean up (days): 19.82278910599495 (0.019653983502327027)
Yearly Cost: 1843.1196935625944 (1.823660713026007)

SensorLevel: 860
Cost: 100.0 (0.0)
Time till clean up (days): 19.922253580040362 (0.01959842101867757)
Yearly Cost: 1833.8899742382196 (1.8005517558082773)

SensorLevel: 865
Cost: 100.00007029827822 (7.029827822198437E-5)
Time till clean up (days): 20.049031240941552 (0.020663333540301757)
Yearly Cost: 1822.467532703719 (1.8758585127298382)

SensorLevel: 870
Cost: 100.0 (0.0)
Time till clean up (days): 20.136402205652775 (0.02080497290746698)
Yearly Cost: 1814.5721112608803 (1.8760113363110644)

SensorLevel: 875

Cost: 100.0092718217418 (0.008632256394634817)
Time till clean up (days): 20.24488642947786 (0.020259019027212725)
Yearly Cost: 1804.8992081237077 (1.816900405878881)

SensorLevel: 880
Cost: 100.04726453111635 (0.027872409987659347)
Time till clean up (days): 20.34834112764828 (0.020400084743211293)
Yearly Cost: 1796.4112969280109 (1.874700559275902)

SensorLevel: 885
Cost: 100.0529398759023 (0.015886155545662 76)
Time till clean up (days): 20.44634404336648 (0.020684446760249376)
Yearly Cost: 1787.9362996404714 (1.8355167457228232)

SensorLevel: 890
Cost: 100.26865742993593 (0.0544336129583069)
Time till clean up (days): 20.562209791337615 (0.02034600290586601)
Yearly Cost: 1781.6416973243572 (2.0437965644246052)

SensorLevel: 895
Cost: 100.78999595912641 (0.10651206965475239)
Time till clean up (days): 20.65679355057234 (0.02036236125034619)
Yearly Cost: 1782.6206161908815 (2.558419062266434)

SensorLevel: 900
Cost: 102.4219715968462 (0.22315465718583813)
Time till clean up (days): 20.76956161433393 (0.020302686451893925)
Yearly Cost: 1801.660403676477 (4.283959961802886)

SensorLevel: 905
Cost: 105.43107865749344 (0.3485659363018773)
Time till clean up (days): 20.865443659499043 (0.020634008697948403)
Yearly Cost: 1846.1206080158888 (6.368549190667895)

SensorLevel: 910
Cost: 110.8455515043826 (0.5166297829471982)
Time till clean up (days): 20.968220619017096 (0.02059620746229525)
Yearly Cost: 1931.4922988308442 (9.254378832894025)

SensorLevel: 915
Cost: 120.82404796184176 (0.7663392656864846)
Time till clean up (days): 21.0724708954701 (0.020642174841040507)
Yearly Cost: 2094.5014583060683 (13.391841738755607)

SensorLevel: 920
Cost: 133.61016523759108 (1.0140722759663252)
Time till clean up (days): 21.170833435345664 (0.02037502627401915)
Yearly Cost: 2305.3248251134714 (17.594380107243385)

SensorLevel: 925
Cost: 153.2758778575519 (1.2398399949217624)
Time till clean up (days): 21.286031849395624 (0.021099757274223217)
Yearly Cost: 2631.8987831186128 (21.64756011054805)

SensorLevel: 930
Cost: 178.15269489306203 (1.5039534250047688)
Time till clean up (days): 21.396490789759497 (0.020518596435655583)
Yearly Cost: 3041.968439266714 (25.84456287440885)

SensorLevel: 935
Cost: 205.1549030302967 (1.7576013415395193)
Time till clean up (days): 21.493699693617025 (0.02117063440643346)
Yearly Cost: 3486.8033148255963 (30.025466725137115)

SensorLevel: 940
Cost: 239.65105316612463 (2.061494300006474)
Time till clean up (days): 21.595961808452884 (0.02065865581873402)
Yearly Cost: 4055.678971160477 (35.303270387125764)

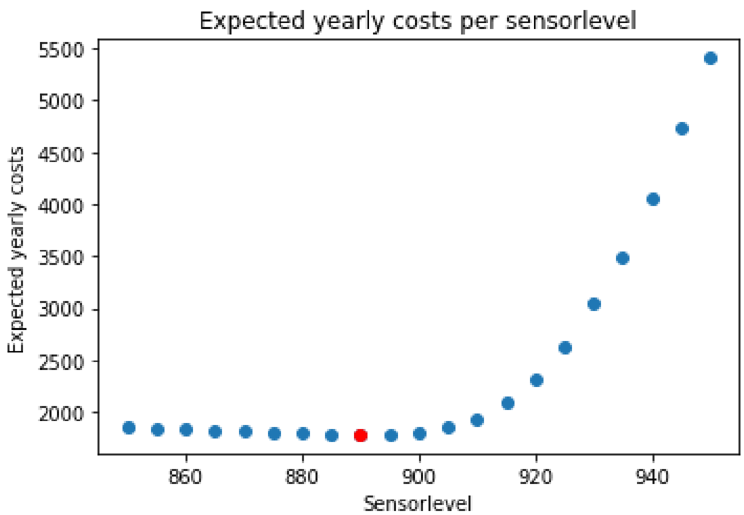
SensorLevel: 945
Cost: 281.1762698938438 (2.2164091353883846)
Time till clean up (days): 21.70310668733565 (0.021467961923982183)
Yearly Cost: 4733.266396874624 (37.59804555866769)

SensorLevel: 950
Cost: 322.7142870885767 (2.3859948614246074)
Time till clean up (days): 21.802017242640304 (0.0215818725279138)

Yearly Cost: 5408.009014127841 (40.24578917688158)

Give a recommendation to the municipality regarding the level at which the sensor should be placed based on your results

We plottted the expected yearly costs per sensorlevel and marked the minimal point in red.



We see that that is the case for sensorlevel 890 with expected yearly costs of ≈ 1782 euros. So we would suggest to set the sensorlevel at 890.