Agent based models

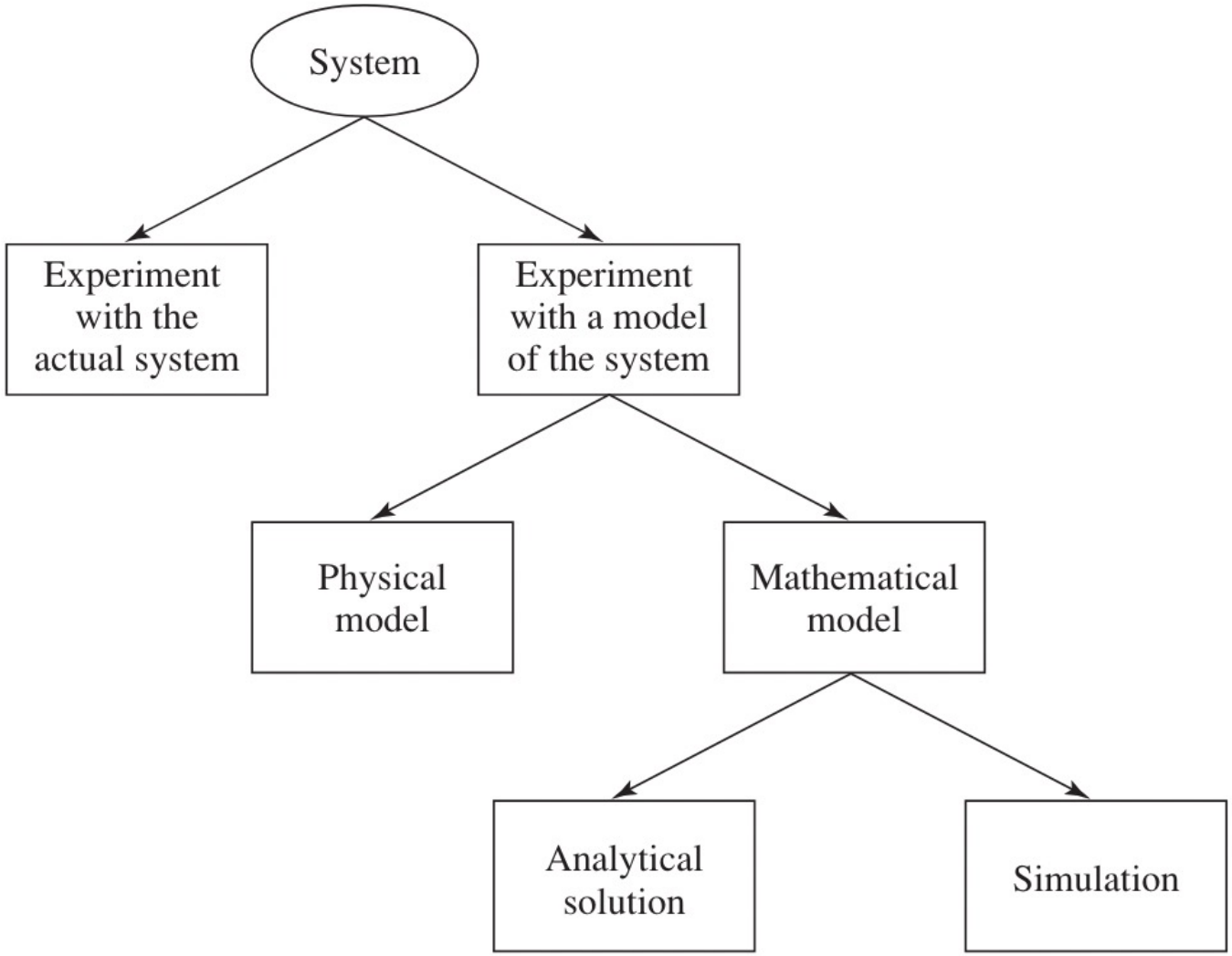
References

- Simulation Modeling and Analysis 5th edition
- Anylogic in 3 days

Itinerary

- introduction to simulation
- agent-based modeling
- emergent behaviors
- input modeling
- using simulations for decision making
- MESA demo

The nature of simulation



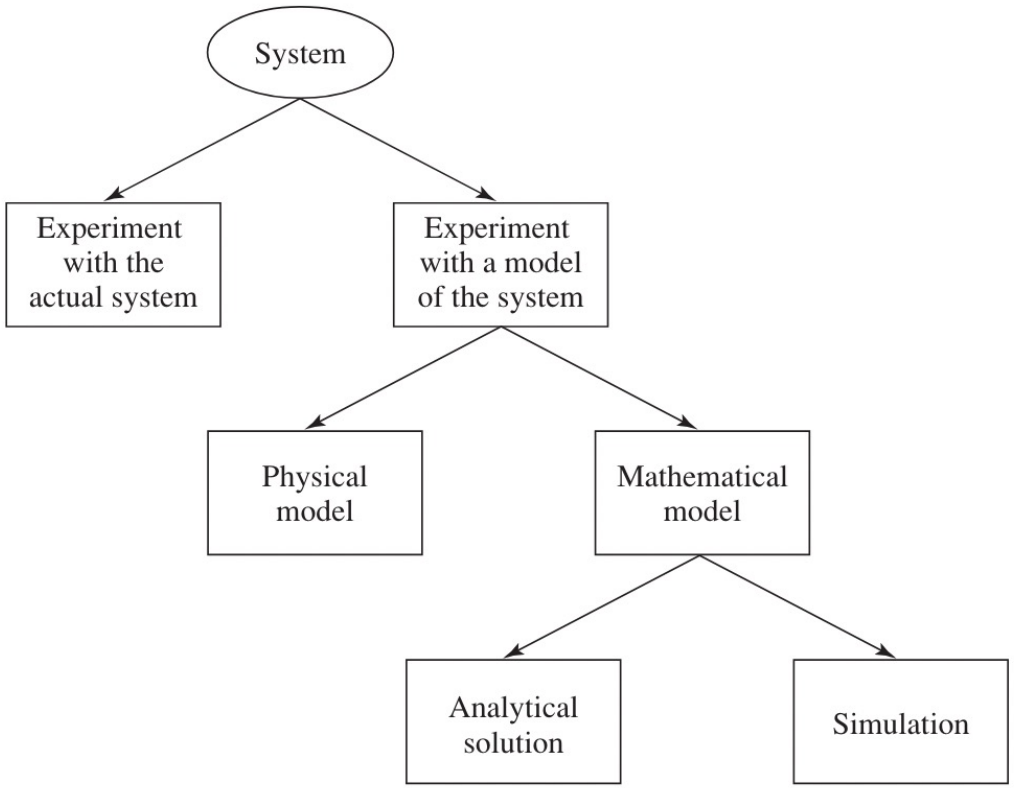
Simulation is the use of computational techniques to study or understand systems or real-world facilities or processes.

examples of systems are:

- arrival of customers to a bank
- departure of fans from a sports arena

What are some reasons for modeling the system with a simulation rather then just using experiments with the actual system?

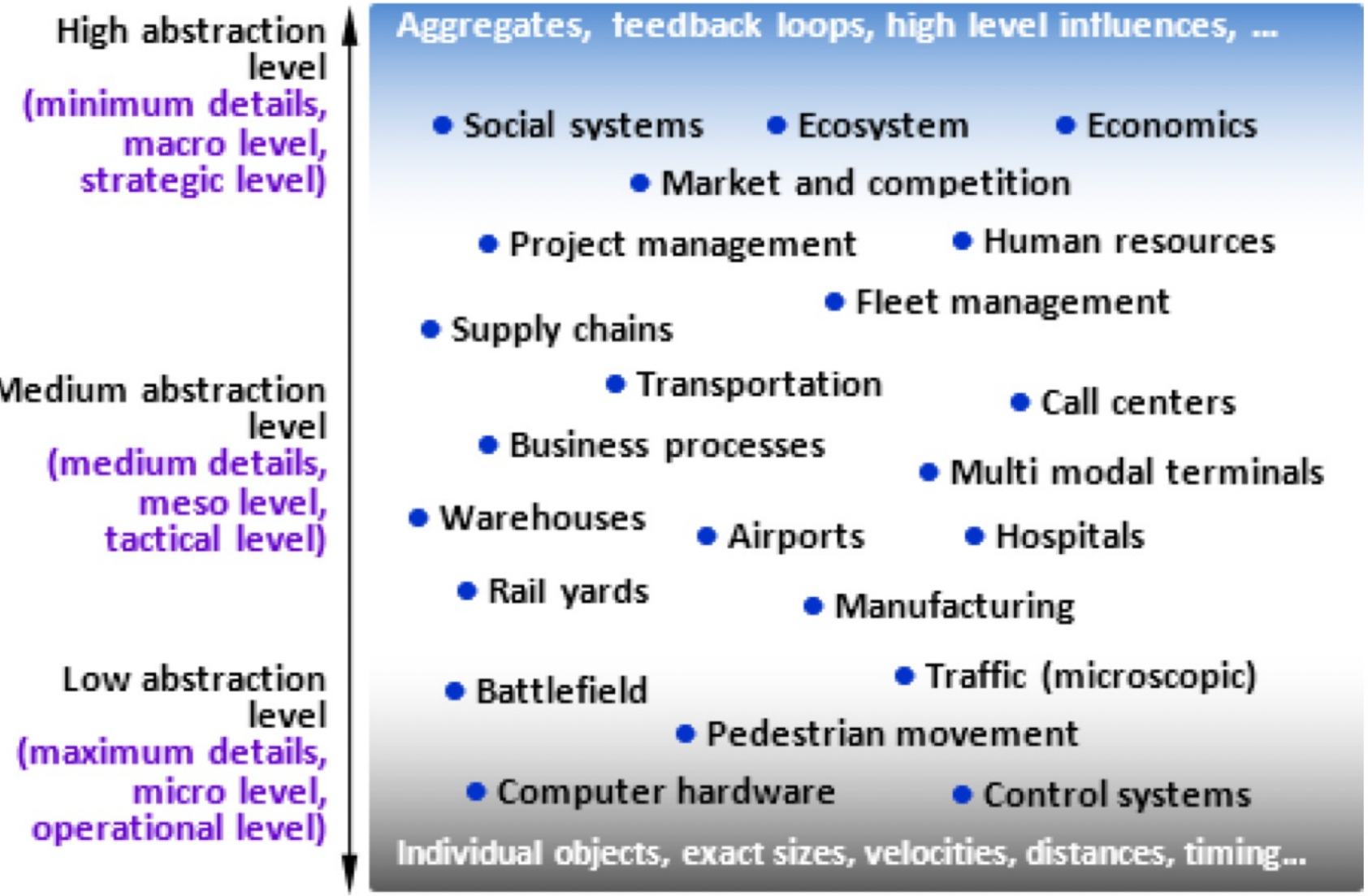
The nature of simulation



What are some reasons for modeling the system with a simulation rather then just using experiments with the actual system?

- real life experimentation is expensive!
- The changes might be disruptive, you might be putting unnecessary stress on the system.
- the system might not even exist yet to draw data from

Using simulation, we can understand and learn more about the behavior and structure of the system.
The simulation can act as a less complex version of the real world that allow us to test multiple conditions and explore the system's response to various conditions
(i.e we can preform stress testing with out hurting anyone)



simulations estimate the real-world interactions in the system

The complexity of the application of the simulations determines the level of abstraction that is needed in the model.

at the bottom are physical level models that are highly detailed representations of the real world.

At the top are macro level model where more abstraction is allowed

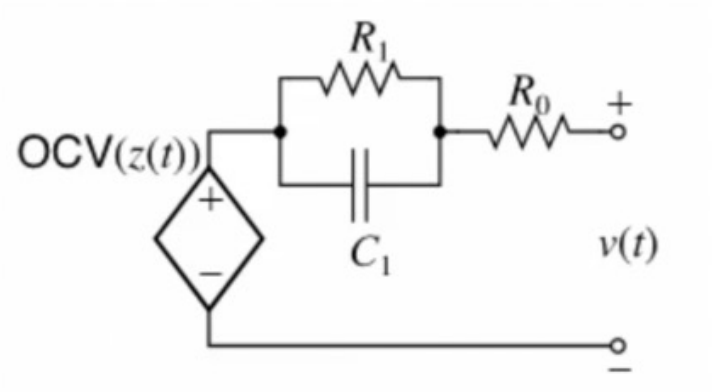
This is an example of a low-level abstraction model

Equivalent Circuit Design Model

We will build a model-based battery simulation tool to estimate the SOC and voltage of a battery in simulation time

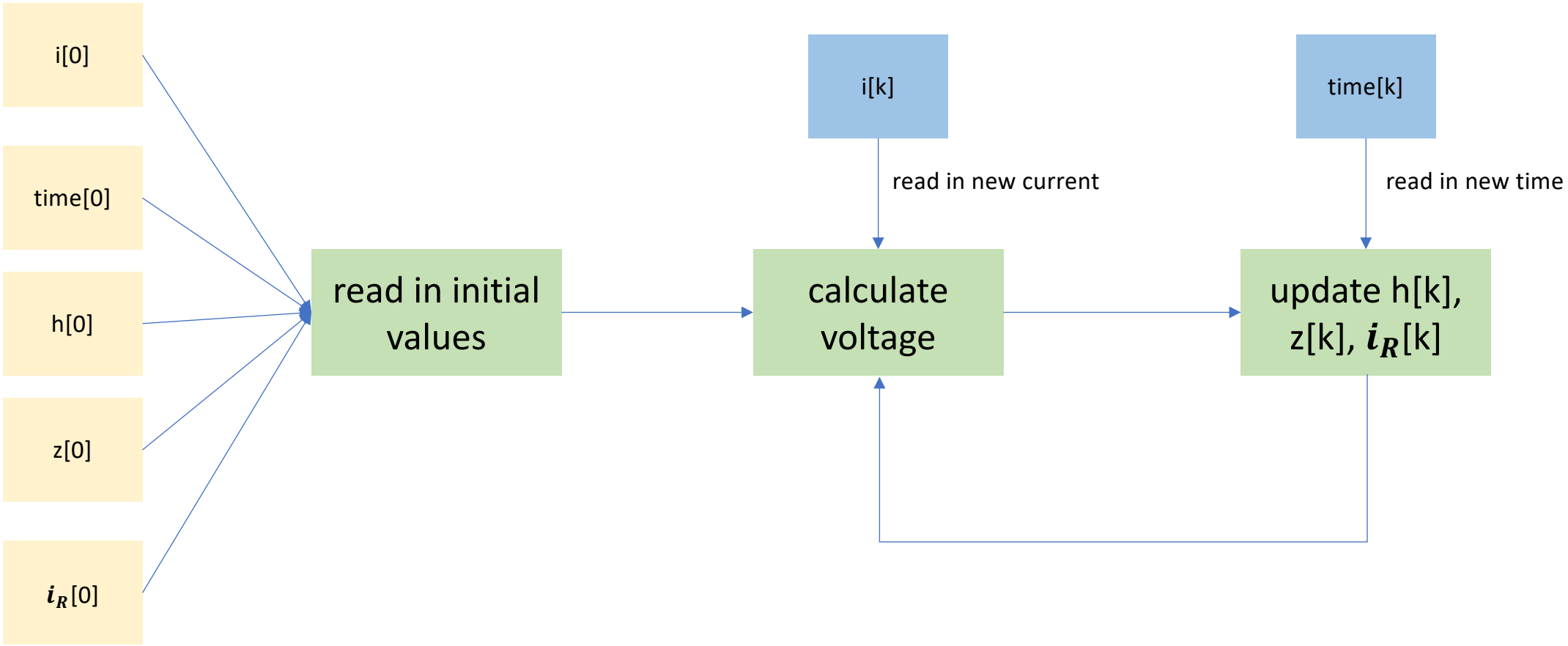
A model-based approach was chosen because:

- It provides a robust performance because of its closed loop feedback mechanism that allows for parameter estimations to be reevaluated
- Relatively easy implementation, uses a series of partial differential equations

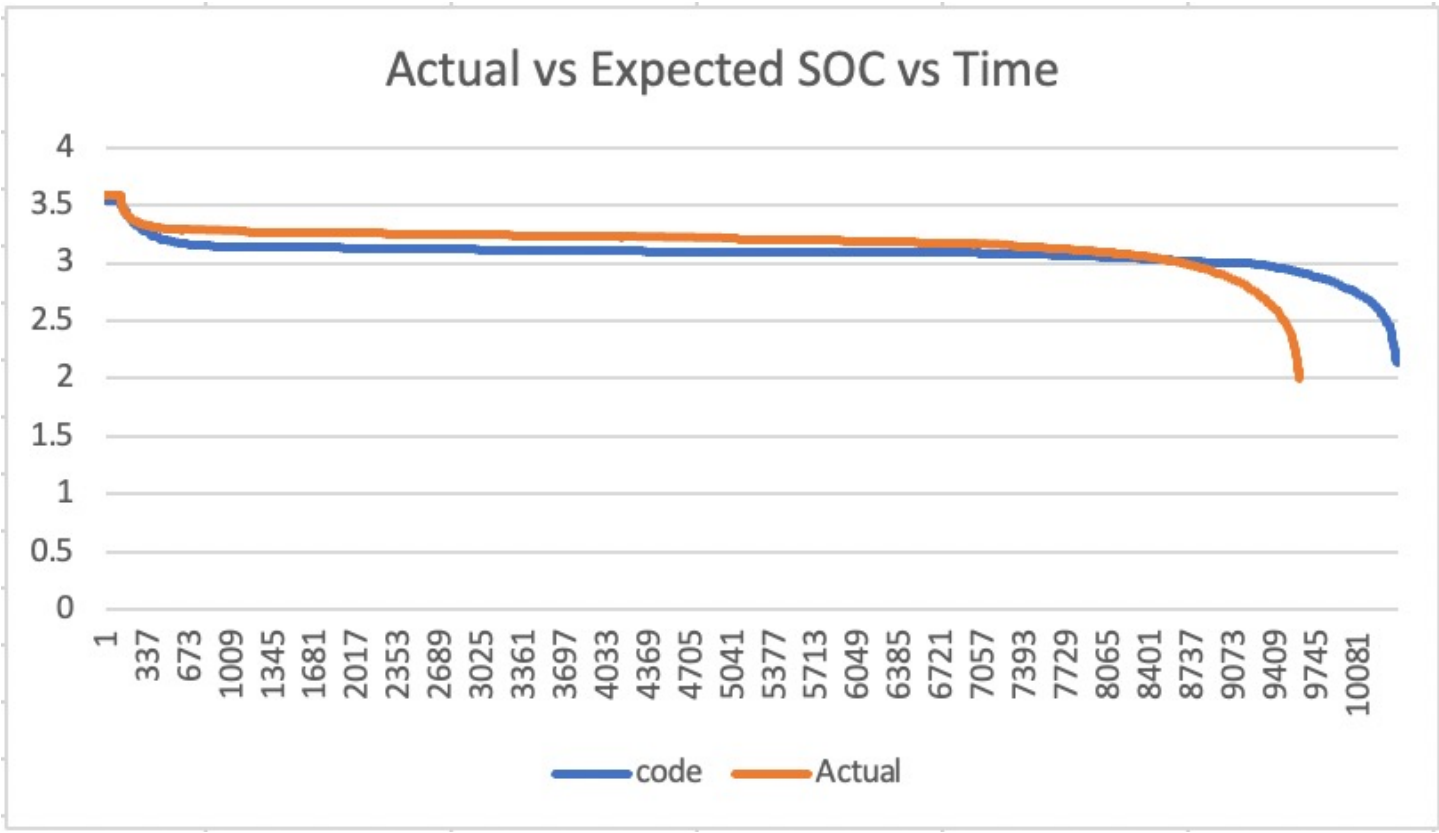


$$\begin{bmatrix} z[k+1] \\ i_R[k+1] \\ h[k+1] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & A_{RC} & 0 \\ 0 & 0 & A_H[k] \end{bmatrix} \begin{bmatrix} z[k] \\ i_R[k] \\ h[k] \end{bmatrix} + \begin{bmatrix} -\frac{\eta[k]\Delta t}{Q} & 0 \\ B_{RC} & 0 \\ 0 & (A_H[k]-1) \end{bmatrix} \begin{bmatrix} i[k] \\ \text{sgn}(i[k]) \end{bmatrix}.$$

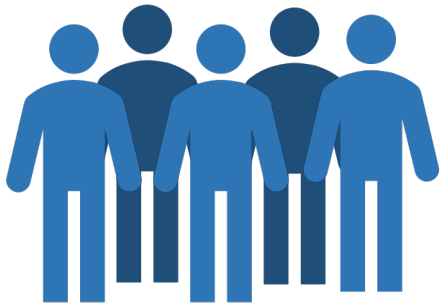
The steps used to code this model-based approach are based on the lecture series Dr. Gregory L. Plett of Colorado University


i[0]
time[0]
h[0]
z[0]
i_R[0]
read in initial values
i[k]
read in new current
calculate voltage
time[k]
read in new time
update h[k], z[k], i_R[k]

Code validation



This is an example of a medium-level abstraction model

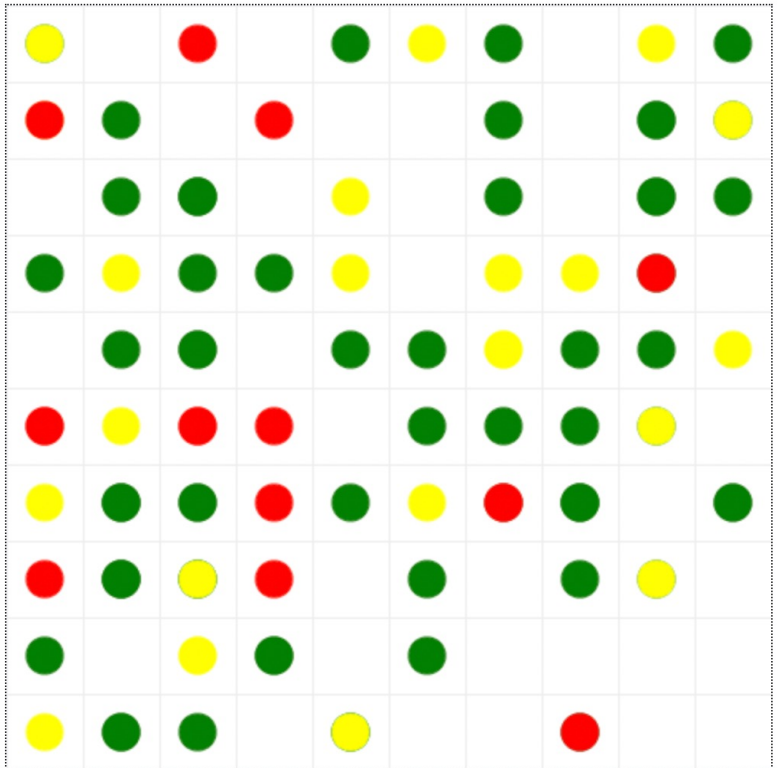


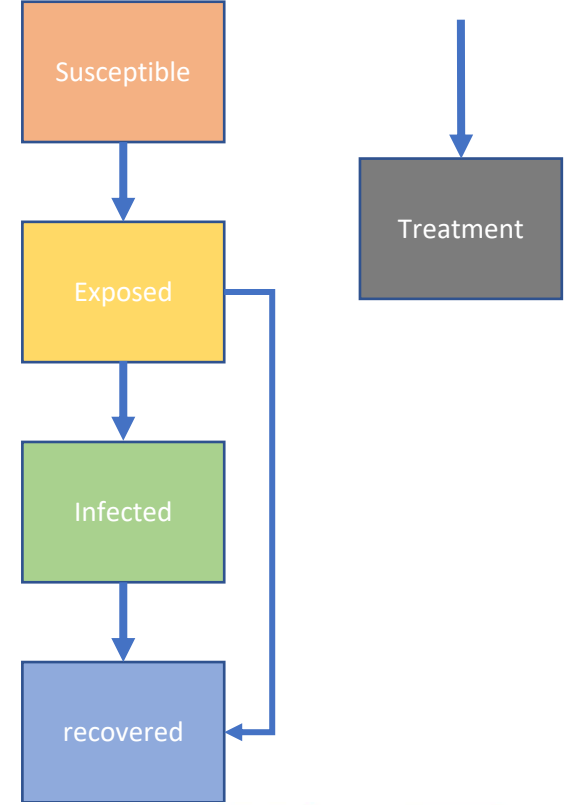
Agent Characteristics examples

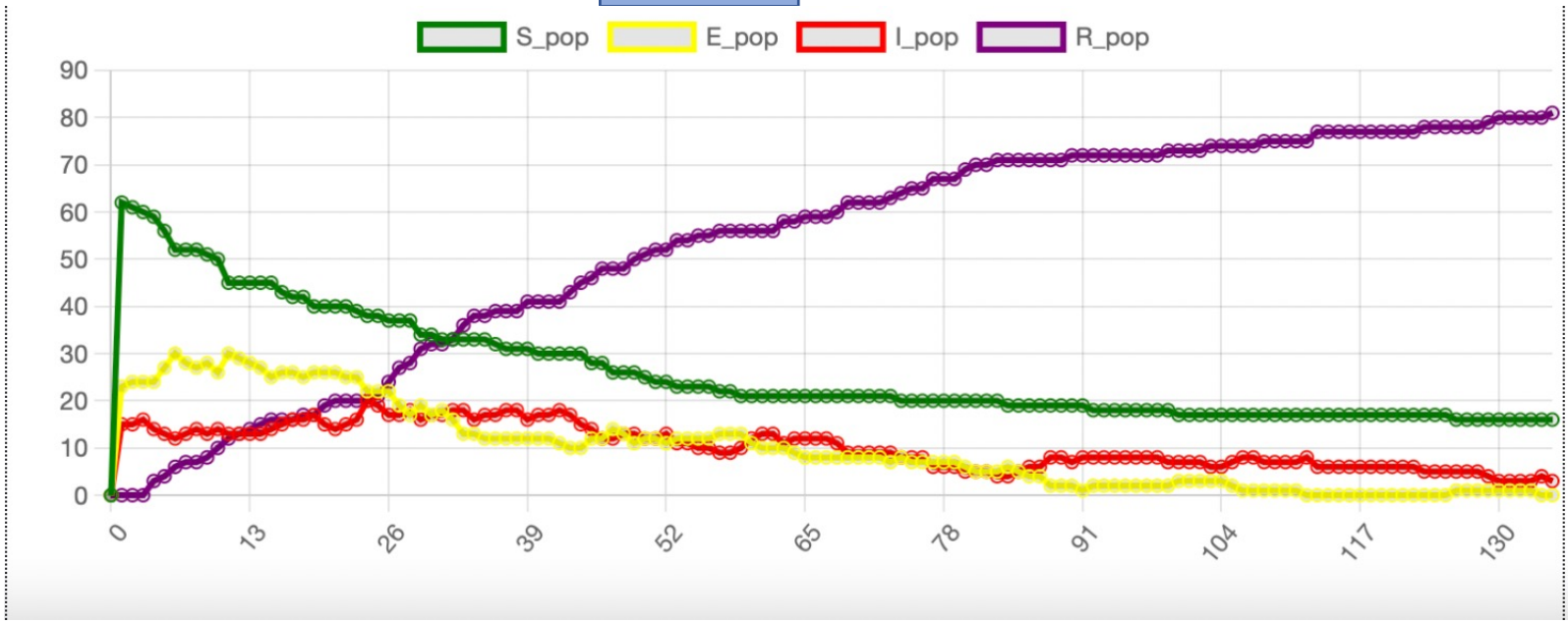
- Weak Immune System (1,0)
- disease characteristic

Frames Per Second 1 20

Current Step: 0







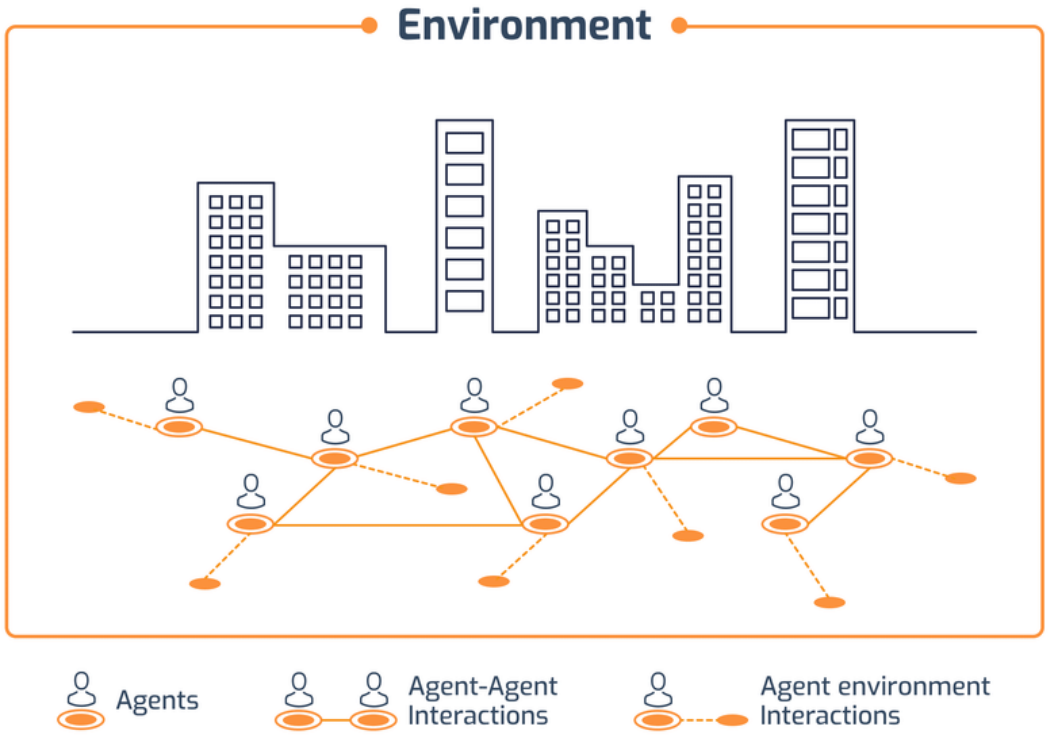
Agent based model

An agent-based model (ABS) is a computational model for simulating the actions and interactions of autonomous agents in order to understand the behavior of a system overall

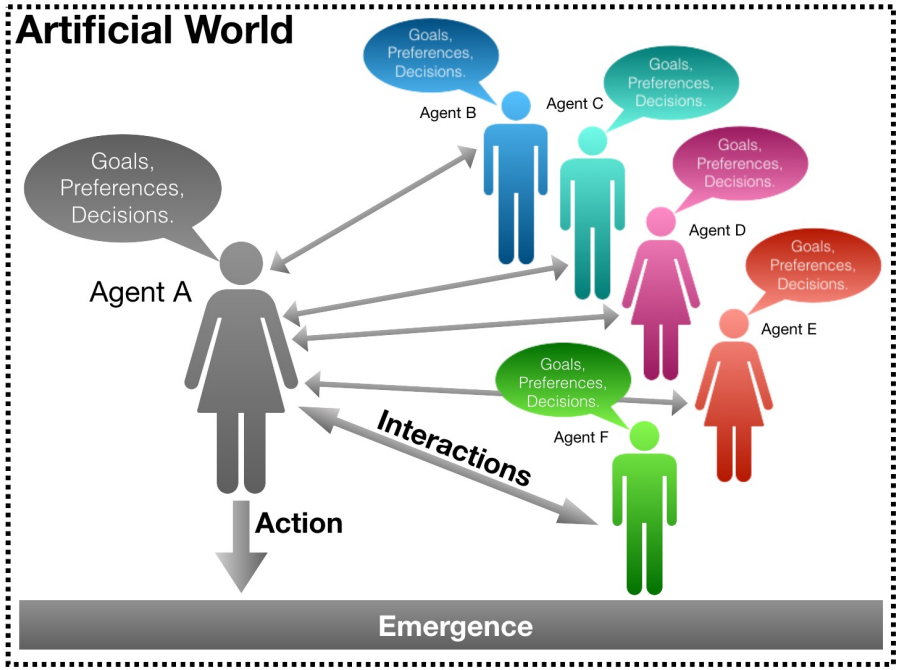
In ABS, modelers take a a bottom-up approach to modeling, the emphasis is on describing the behavior and interaction of the individual agents.
an agent's behavior can be described by:

- its own personal parameters
- interactions with other agents
- interactions with the environment

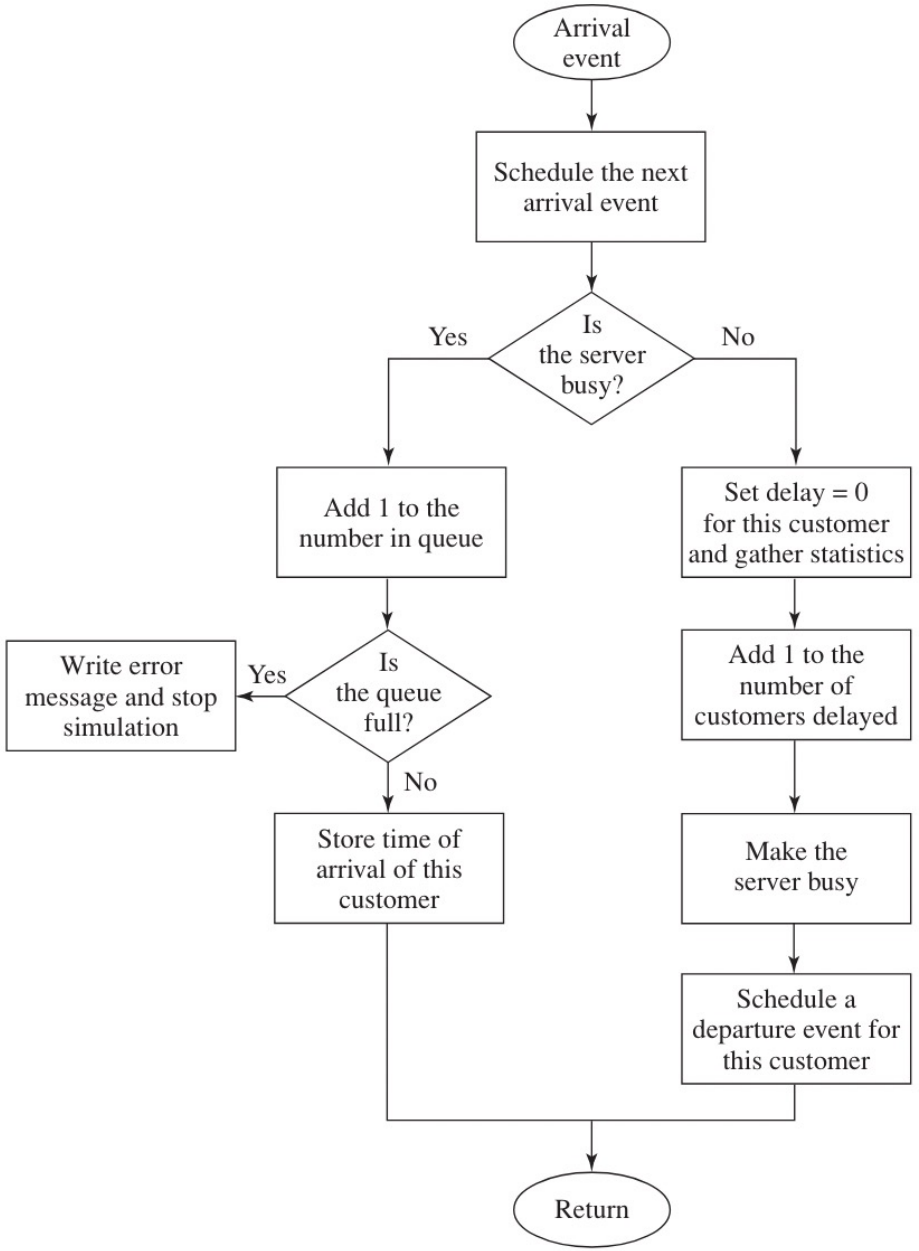
Because of the modular nature of ABM, they are usually implemented in object-oriented programs like Java or Python



Agents define the model



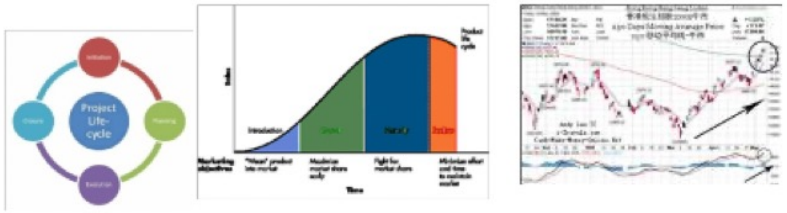
- An agent is an autonomous "entity" that can sense its environment, including other agents, and use this information in making decisions.
- Agents have attributes and a set of basic if/then rules that determine their behaviors.
- They may also learn (gain a better understanding of the status of other agents and their environment) and adapt their behaviors (change their decision rules) over time
  - which will require them to have some form of memory



People in different roles:
consumers, citizens, employees, patients, doctors, clients, soldiers, …

Equipment, vehicles:
trucks, cars, cranes, aircrafts, rail cars, machines, …

Non-material things:
projects, products, innovations, ideas, investments …



Organizations:
companies, political parties, countries, …

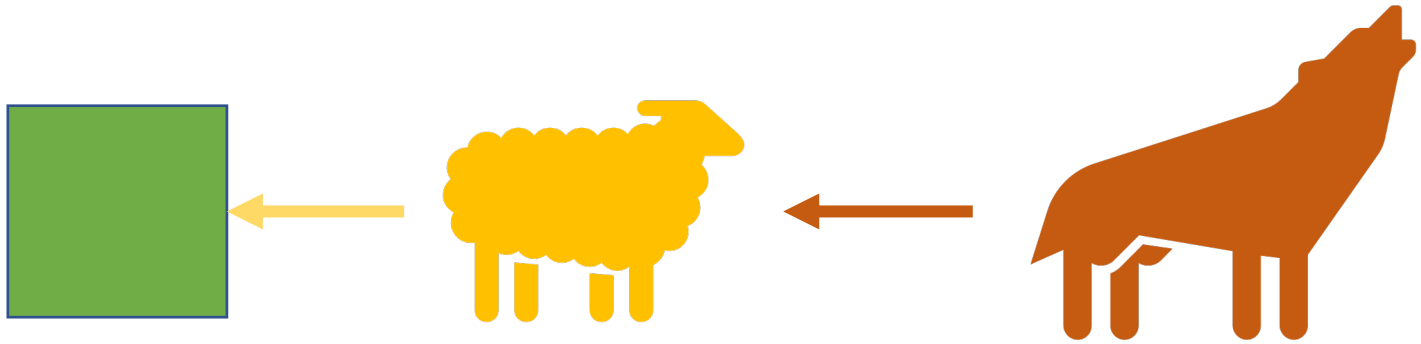
who can be an agent? this is point of debate…..
but in essence anything!

- agents don’t have to live in discreet space
- agents don’t have to be people
- agents can be passive
- you can have multiple types of agents
- there are agent based models where agents don’t interact

predator prey model

let's go through The simplest ecological **system that** can be constructed from interacting species,
The **predator prey model**

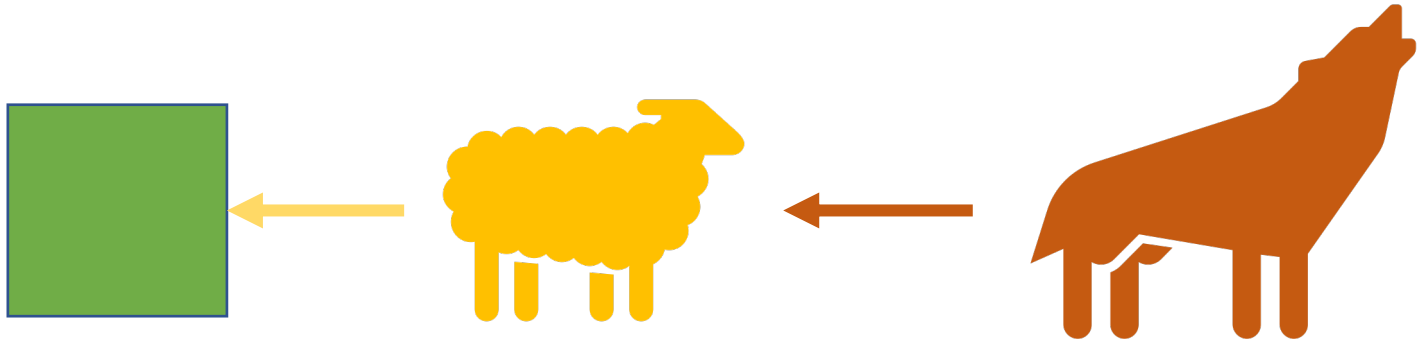
in this system we look at three agent types; grass, sheep, and wolves
and we are interested in how their population level changes as they interact in an ecosystem



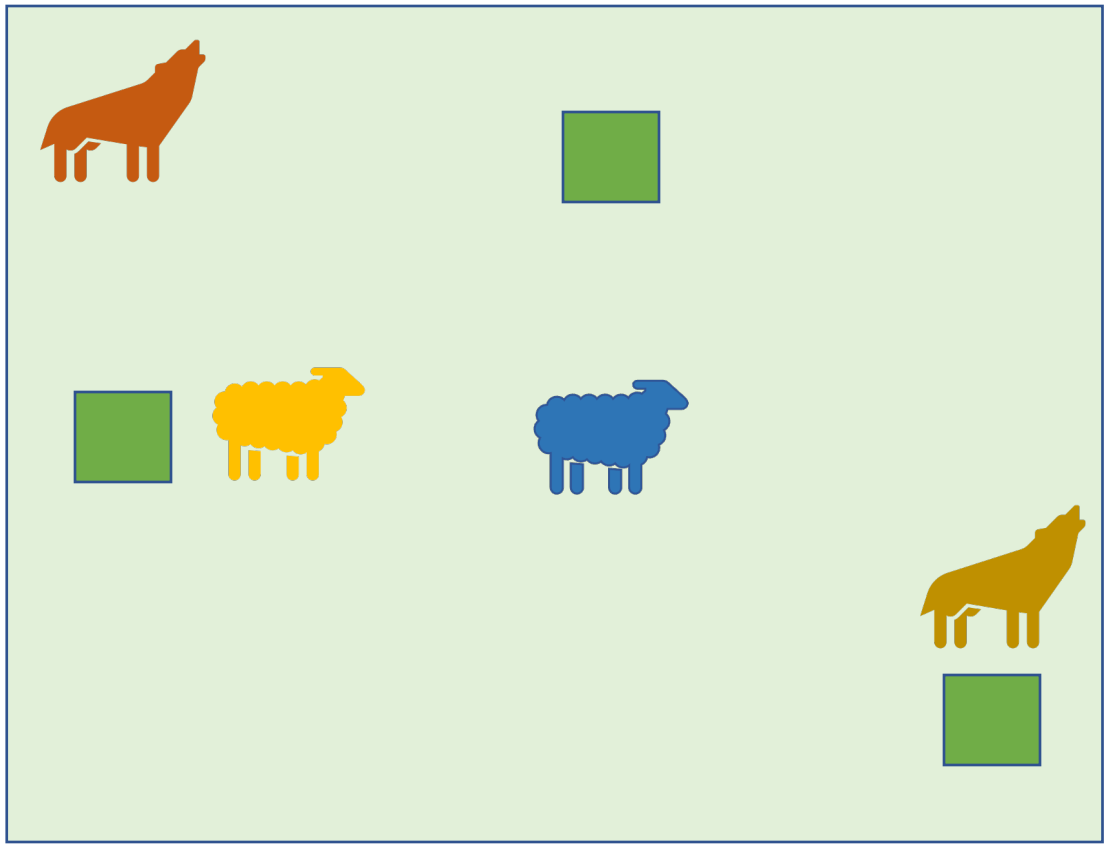
predator prey model

we will take a bottom-up approach and describe each agent's:

- parameters
- interactions with self
- interactions with others



	grass	sheep	wolf
parameters	• regrow time left • alive?	• energy level	• energy level
interactions with self	• regrow if not alive	• lose 1 energy every time it moves • 0 energy means death • produce offspring with a probability of .04	• lose 1 energy every time it moves • 0 energy means death • produce offspring with a probability of .05
interactions with others	• can be eaten by sheep	• can be eaten by wolf on same tile • can eat grass and gain 4 energy	• can eat sheep and gain 20 energy



energy level

2

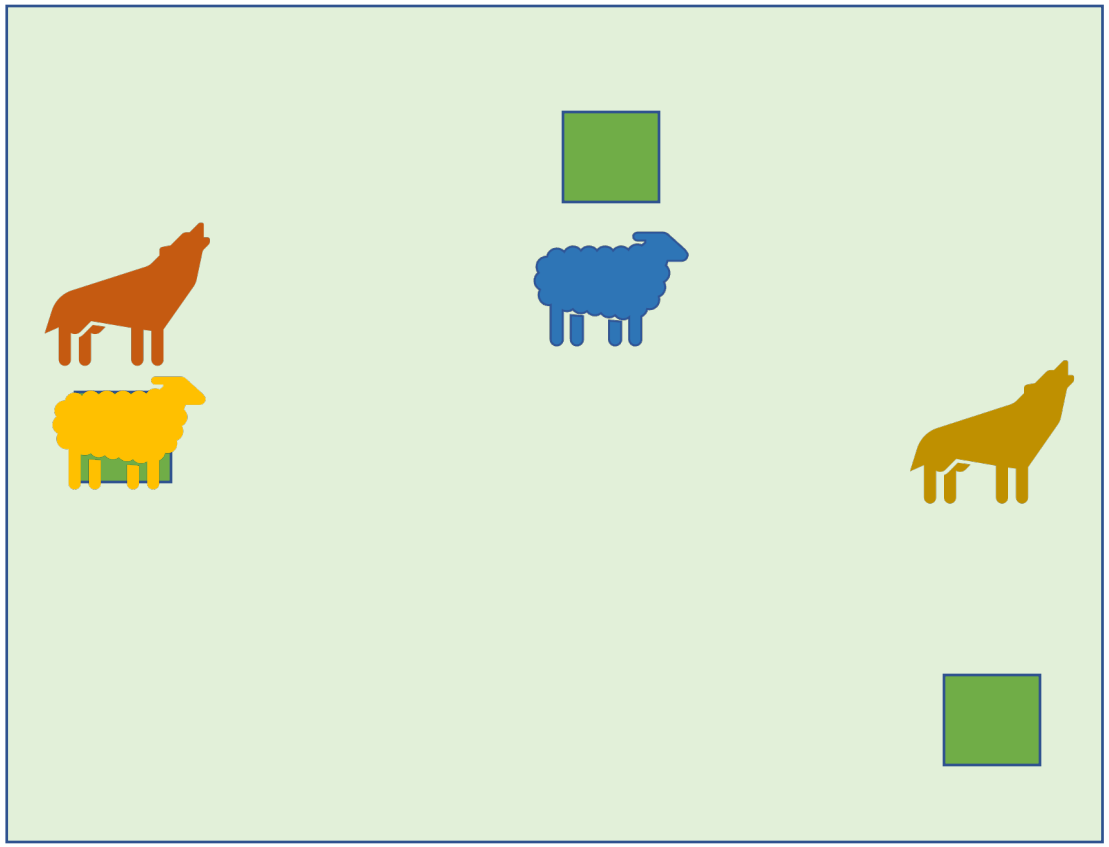
4

2

5

time unit 1

movement



energy level

2 – 1 = 1

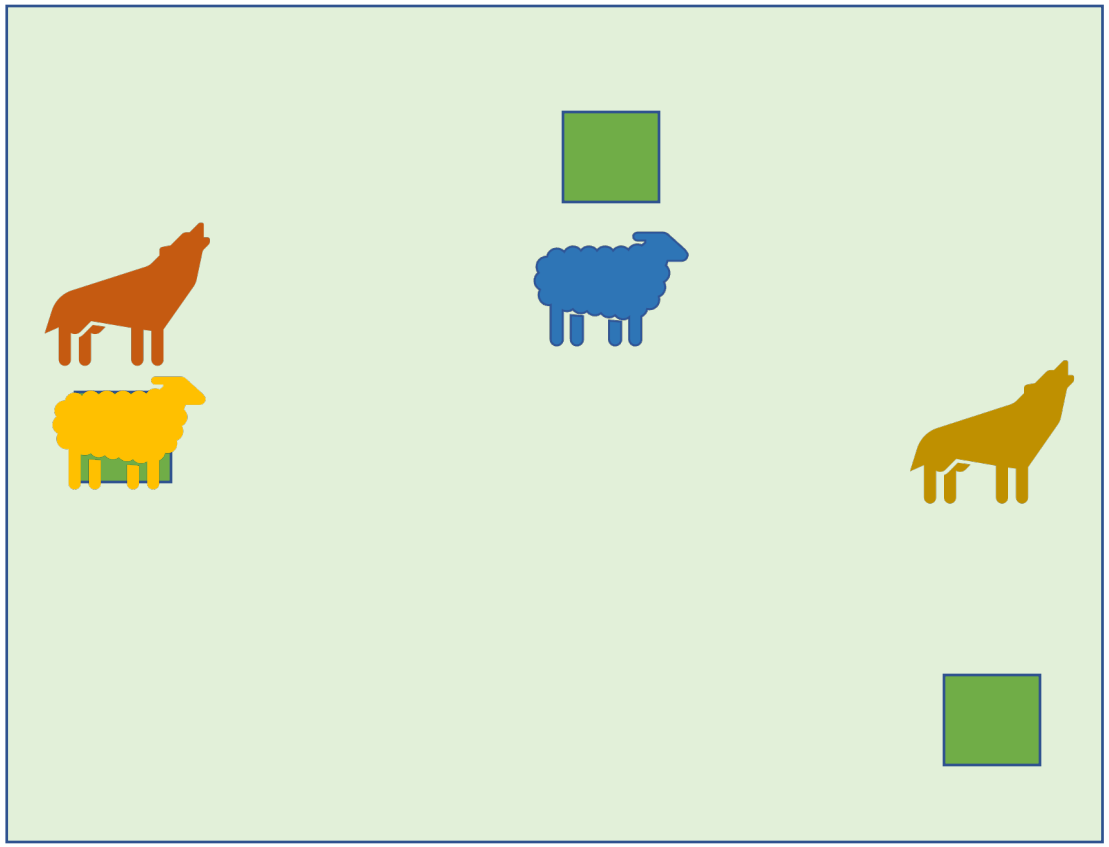
4 – 1 =3

2 – 1 = 1

5 – 1 = 4

time unit 1

check energy



energy level

1

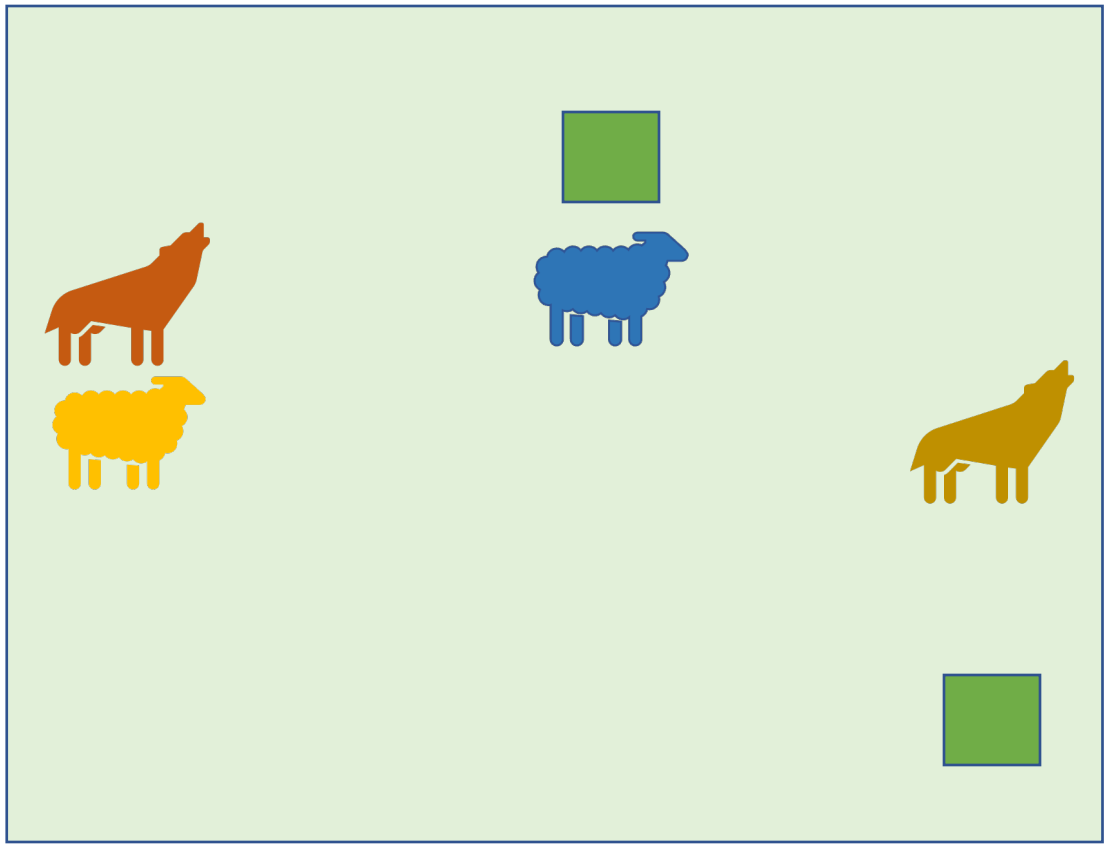
3

1

4

time unit 1

eat



1 + 4 = 5

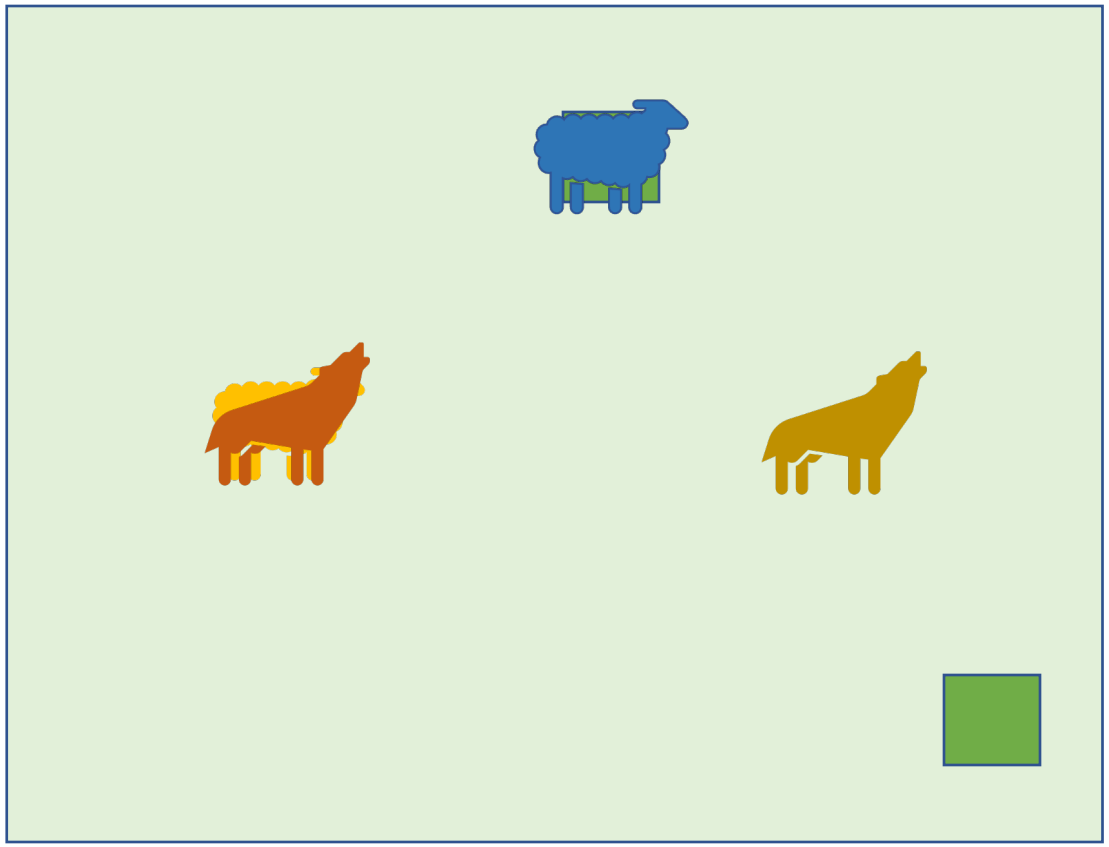
3

1

4

time unit 2

movement



5 – 1=4

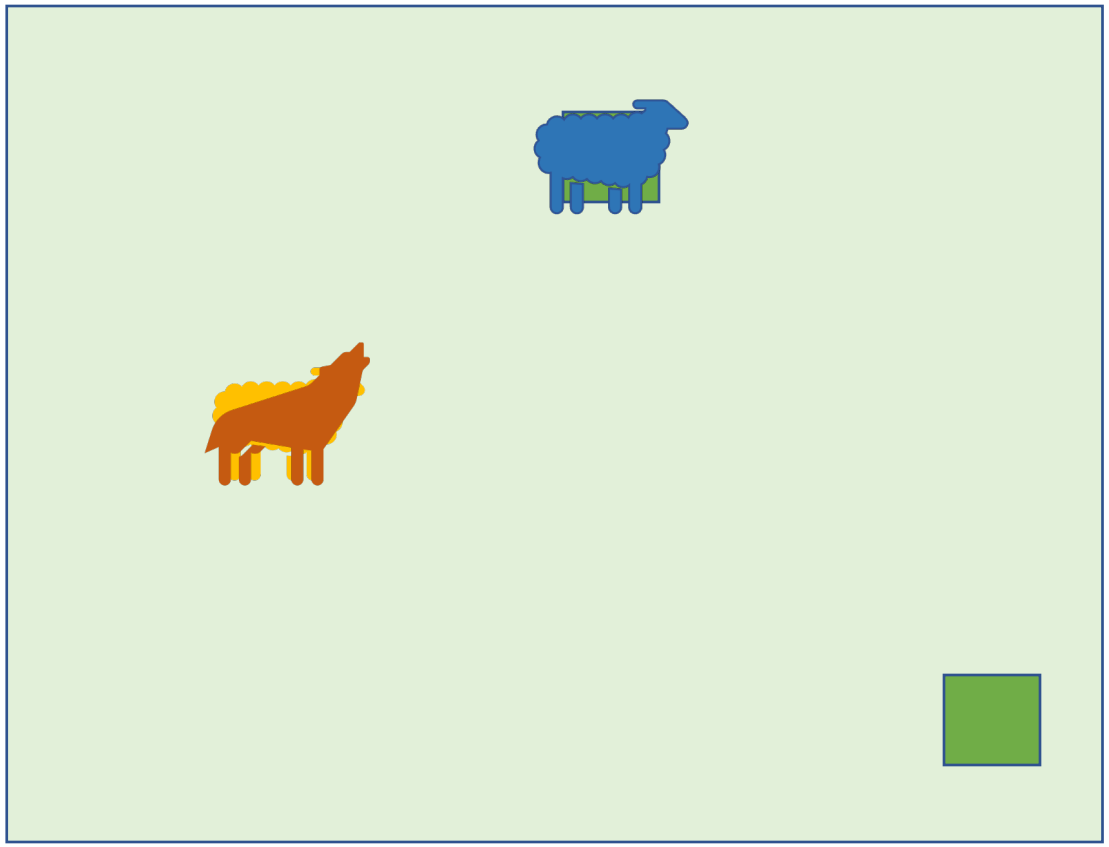
3 – 1 = 2

1 – 1 = 0

4 – 1 = 3

time unit 2

check energy



4

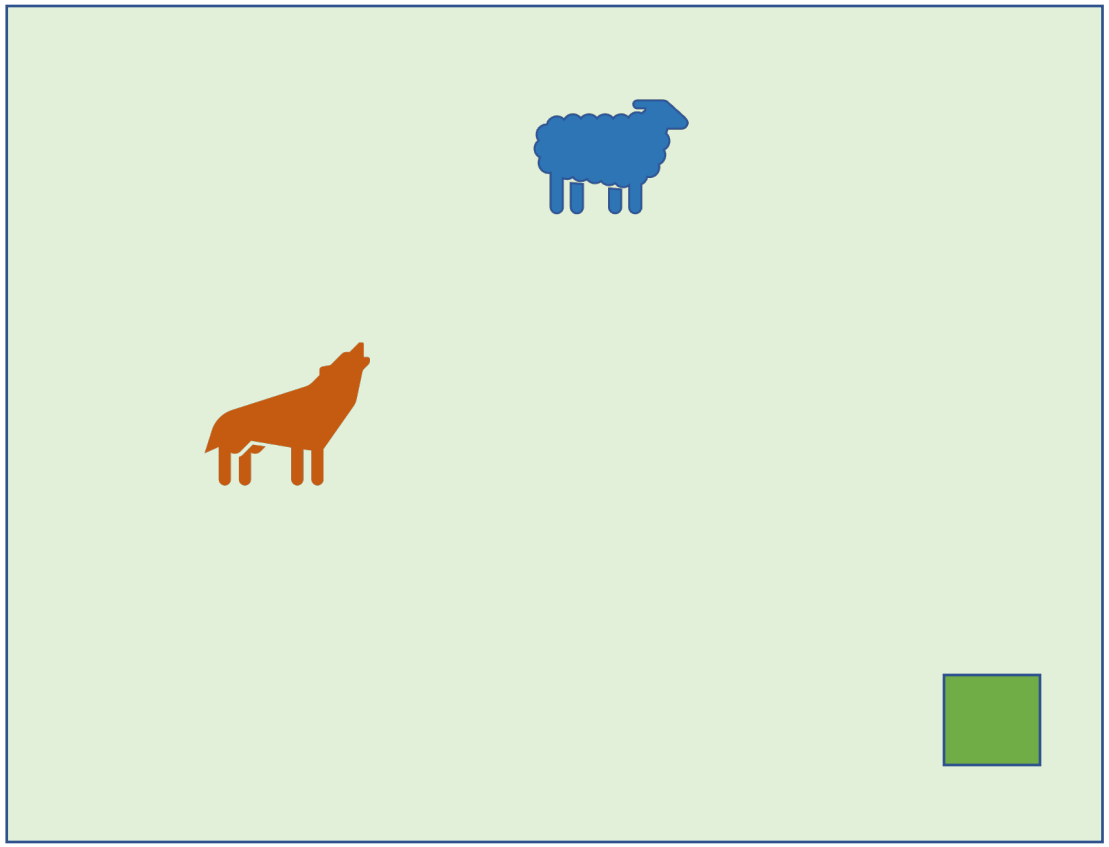
2

0

3

time unit 2

eat

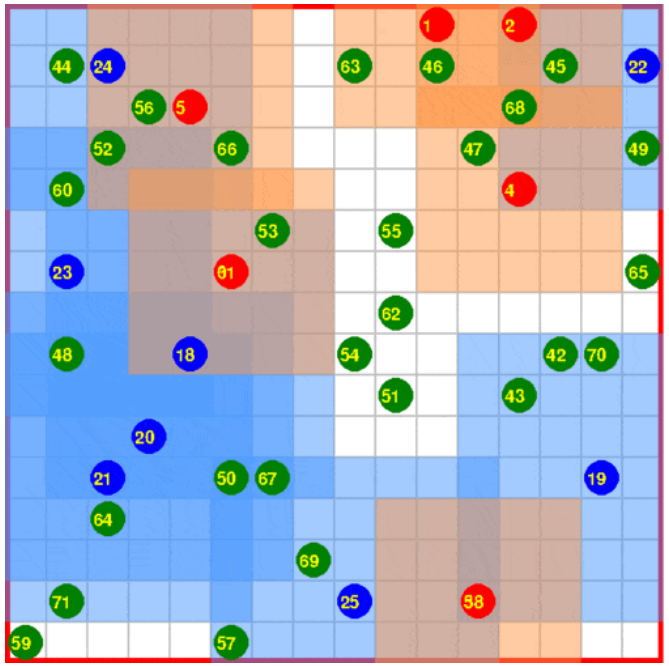


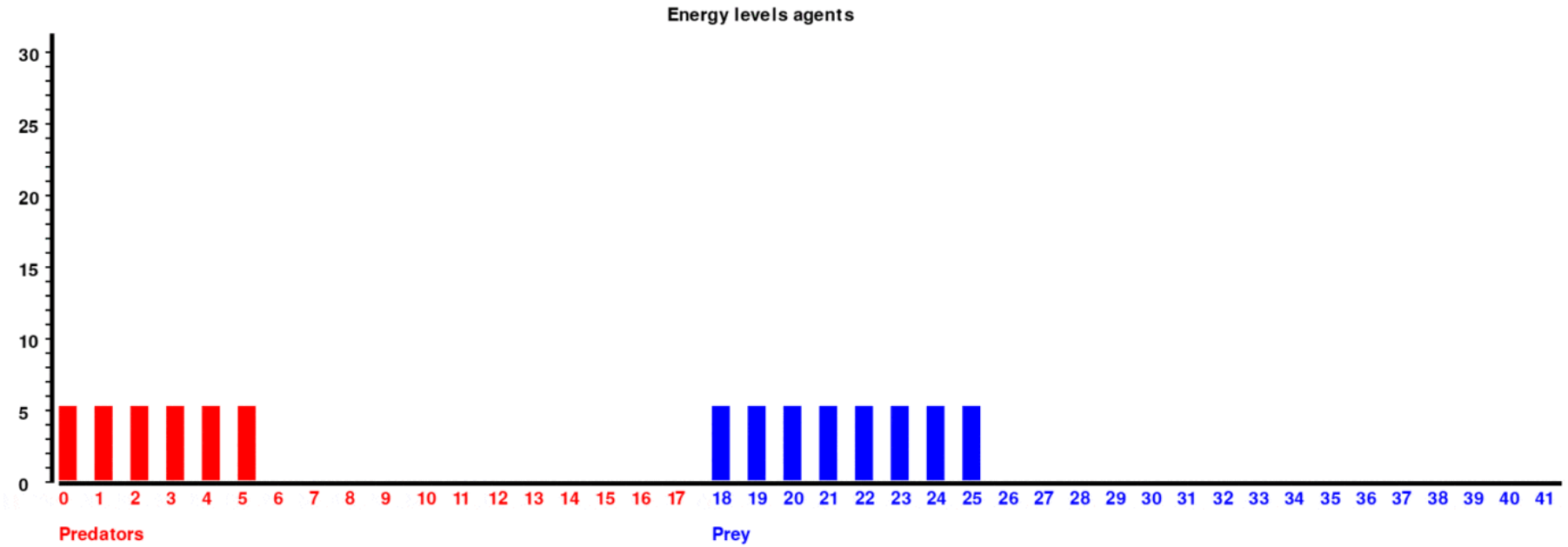
dead

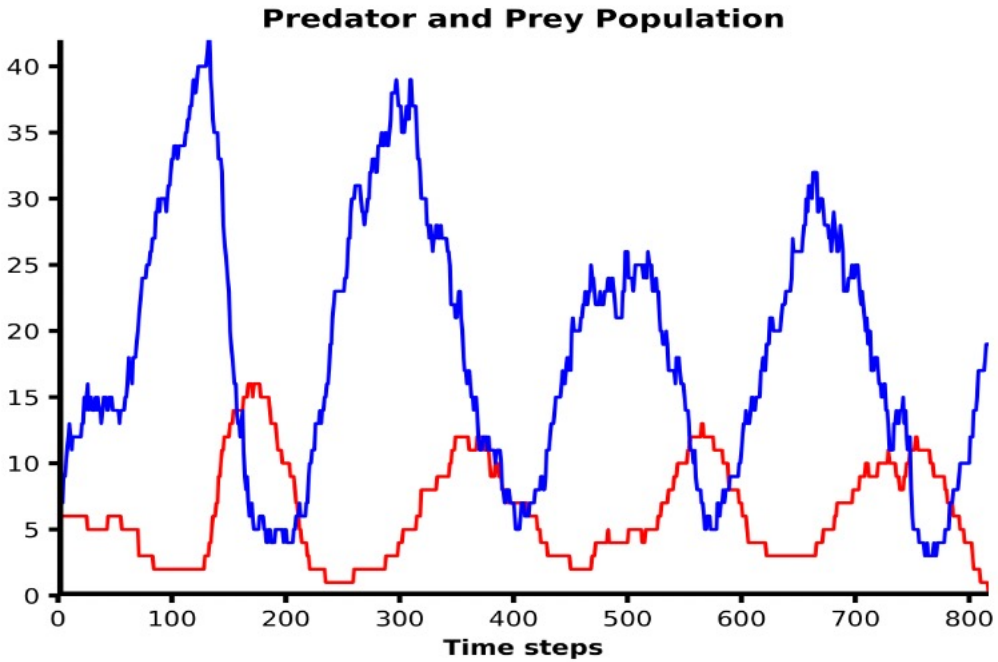
2 + 4 = 6

dead

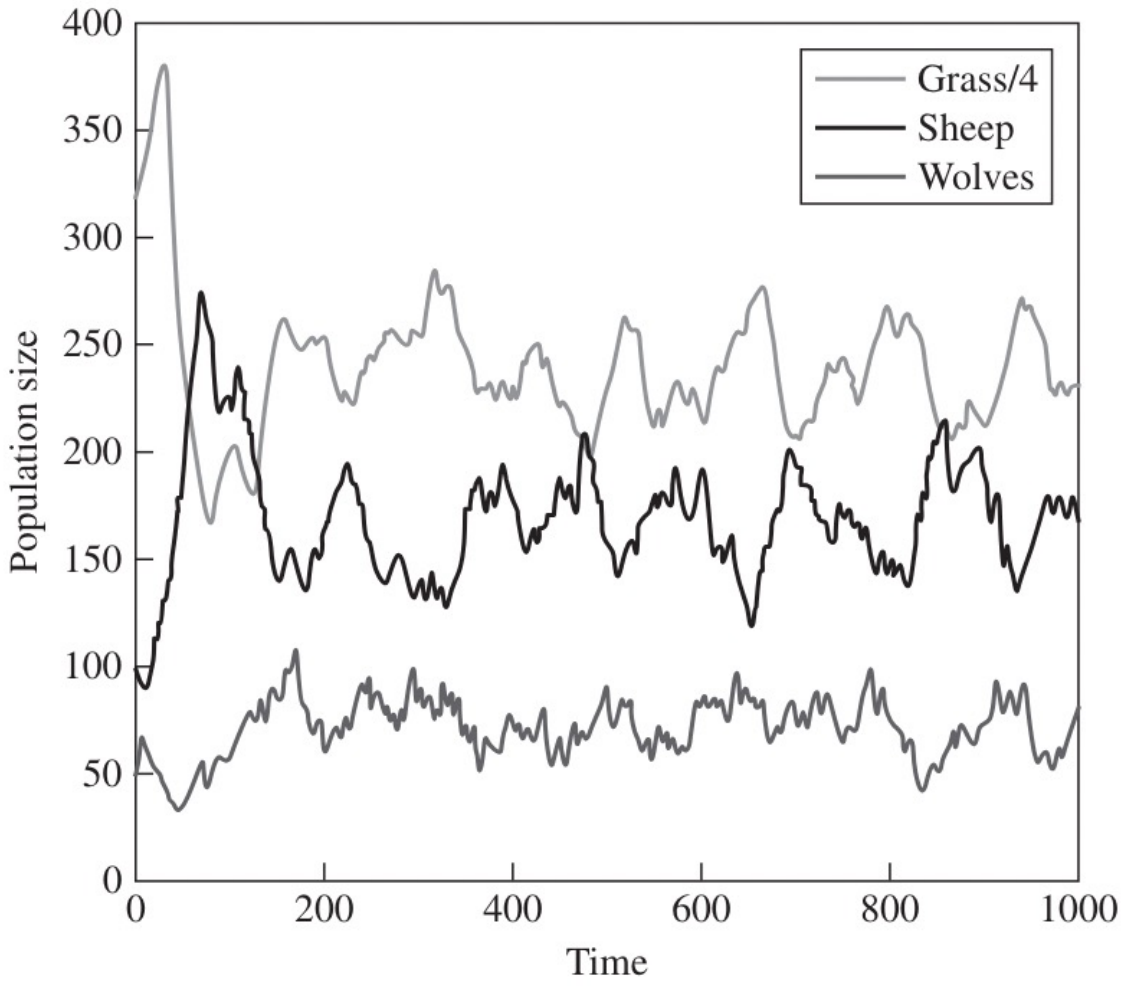
3 + 20 = 23







what observations can you make about the sheep and wolf population?



lead time

eventual equilibrium

• The initial output of a simulation is very sensitive to initial conditions, leading to the strange and chaotic response in the beginning. This is known as the lead time or warm up period

after this “warm-up” period, the distributions of simulation output do not depend on initial conditions

when analyzing the results or making decisions, we usually want to delete this “warm-up” stage

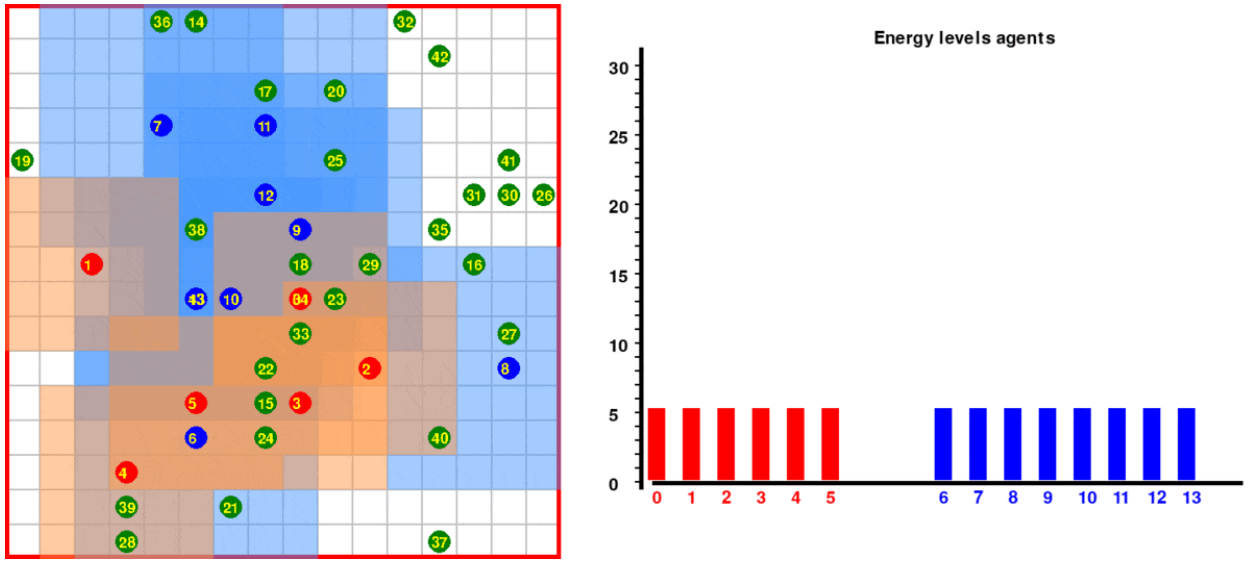
How to determine warm-up (data-deletion) length?

o A graphical procedure

o Pilot runs and production runs

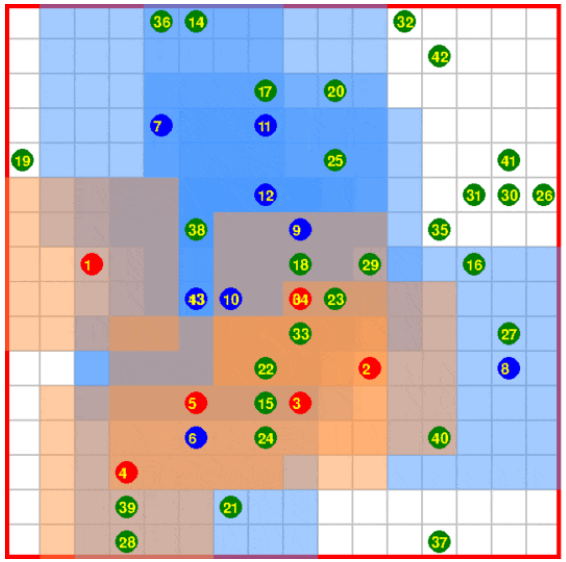
Emergent behavior in agent based model

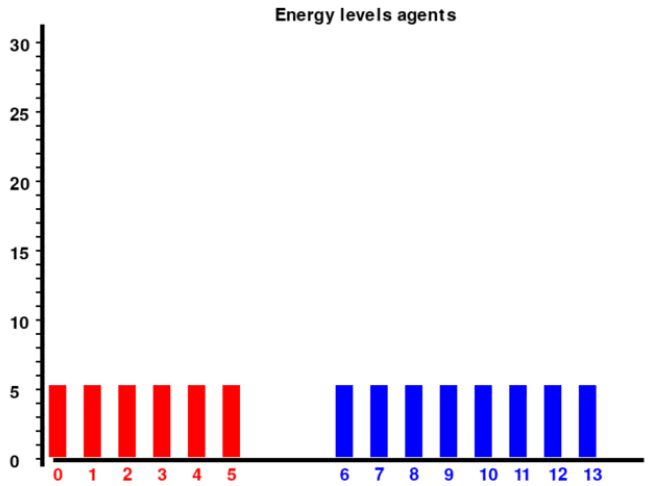
- One of the most interesting results of agent based simulations are the elaborate behaviors that can emerge from simple rules
- In some ABS the interactions of the “low-level” agents over time result in emergent behavior of the system, which is not deducible from the characteristics of the individual agents

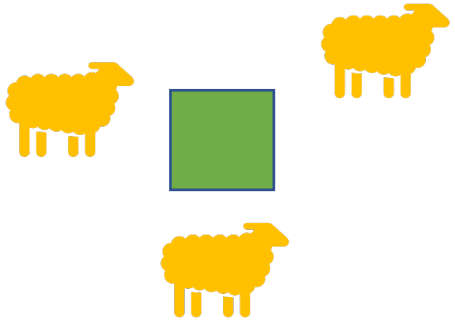


Emergent behavior in agent based model

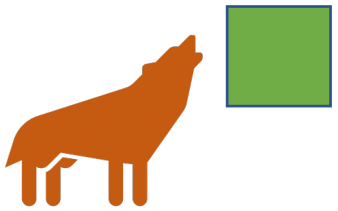
- One of the most interesting results of agent based simulations are the elaborate behaviors that can emerge from simple rules
- In some ABS the interactions of the “low-level” agents over time result in emergent behavior of the system, which is not deducible from the characteristics of the individual agents



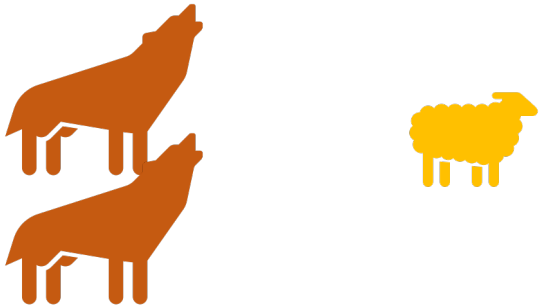




sheep eat grass when ever they can



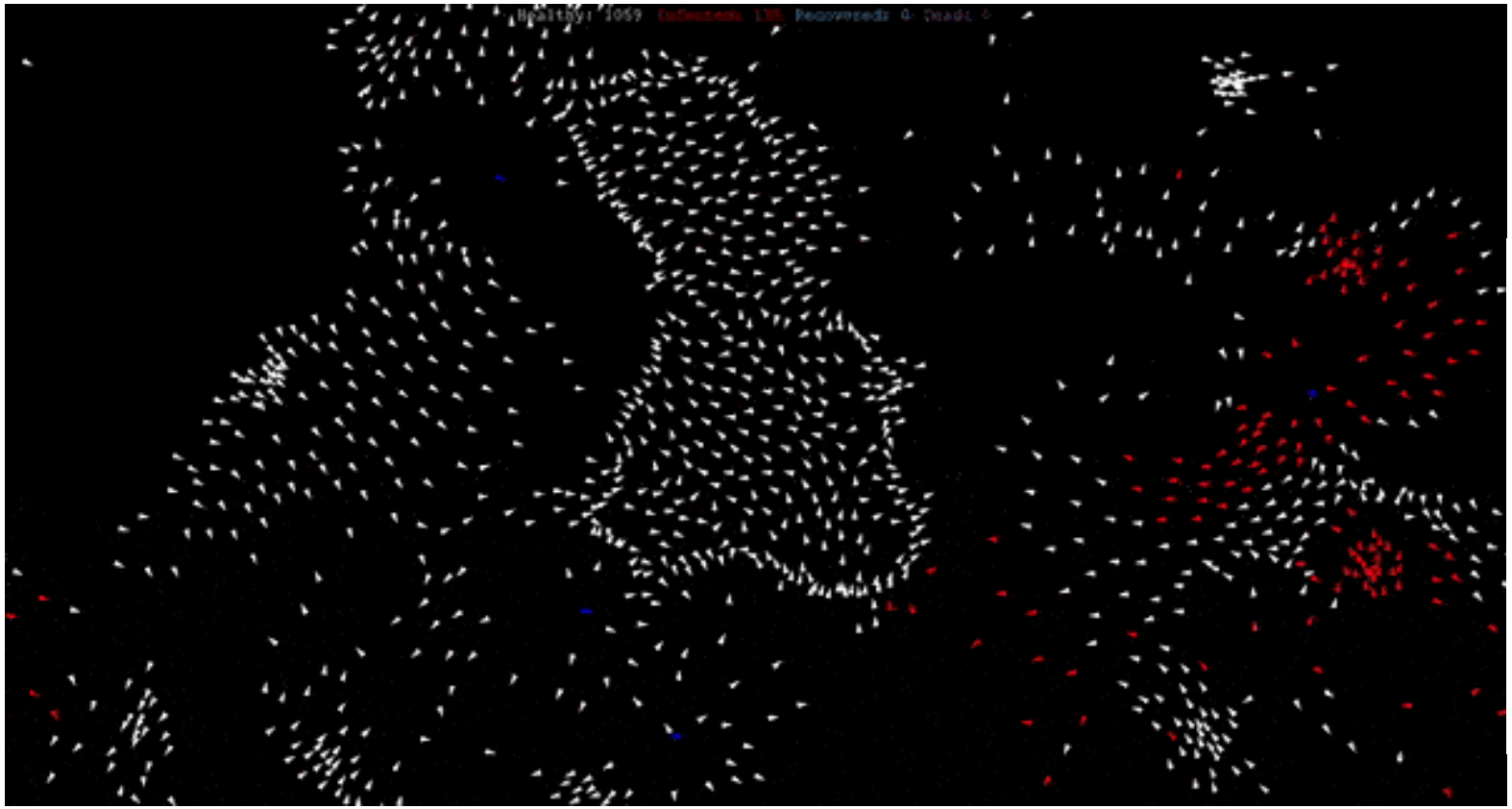
wolves will camp around grass waiting for sheep



wolves will hunt sheep in groups

one of the most famous examples of emergence in ABS

- EXAMPLE 13.2. Consider a version of the “Boids” bird-flocking model [Reynolds (1987)] where agents (Boids) use the following three simple rules:
- Cohesion—each agent steers toward the average position of “local” (or nearby) agents
- Separation—each agent steers to avoid crowding local agents
- Alignment—each agent steers toward the average heading of local agents



```
PROCEDURE move_all_boids_to_new_positions()

        Vector v1, v2, v3
        Boid b

        FOR EACH BOID b
                v1 = rule1(b)
                v2 = rule2(b)
                v3 = rule3(b)

                b.velocity = b.velocity + v1 + v2 + v3
                b.position = b.position + b.velocity
        END

END PROCEDURE
```

Even with the agents just applying these three simple rules:

- cohesion
- separation
- alignment

we see leaderless flocks develop amongst the boids, which is considered by many to be an example of emergent behavior.

Activity

Design an agent-based model using a bottom-up approach

- customers arriving to a bank
- a disease spreading in a population
- predator prey model

- Determine what your agent will be
  - determine what parameters you want to track for each agent
  - determine what interactions they will have with other agents
  - determine what interaction they will have with the environment
- Determine what decisions each agent in your model might make, and what possible emergent behaviors you might see

Input modeling

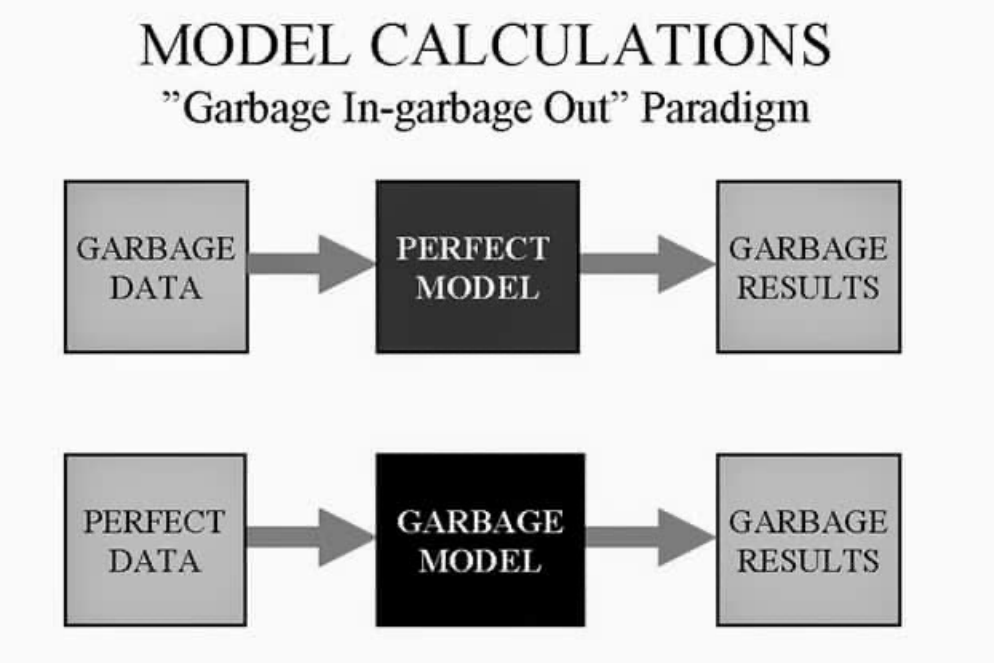


TABLE 6.1
Sources of randomness for common simulation applications

Type of system	Sources of randomness
Manufacturing	Processing times, machine times to failure, machine repair times
Defense-related	Arrival times and payloads of missiles or airplanes, outcome of an engagement, miss distances for munitions
Communications	Interarrival times of messages, message types, message lengths
Transportation	Ship-loading times, interarrival times of customers to a subway

- Simulations require some form of probability distribution to model the random nature of the world
- why can't we just use the mean?
- The "right" input modeling is where good simulation starts from
  - regardless of how good your data is, if your model is trash, the results will match

common definitions

what is an experiment?
- a process with an unknown outcome

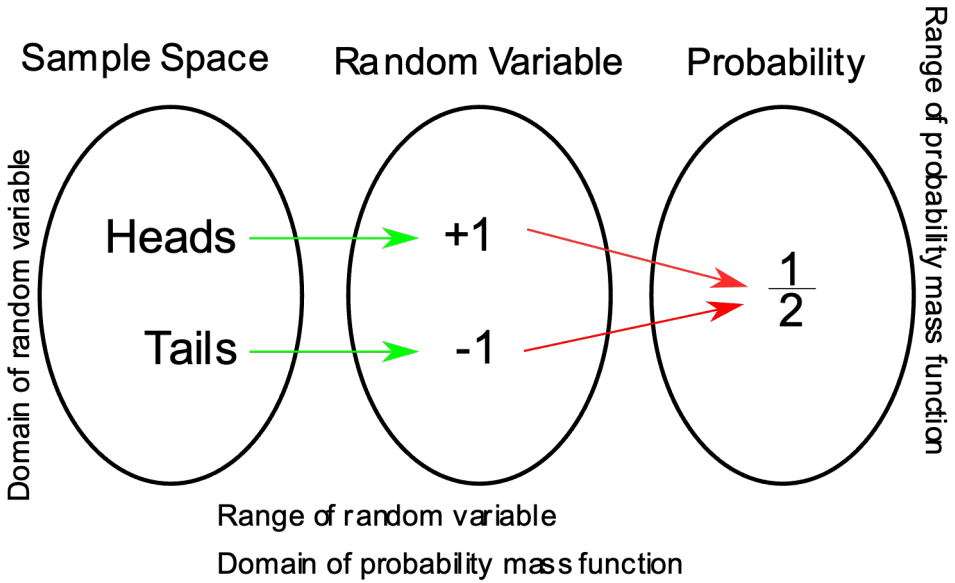
what is the sample space?
- the possible outcomes of an experiment

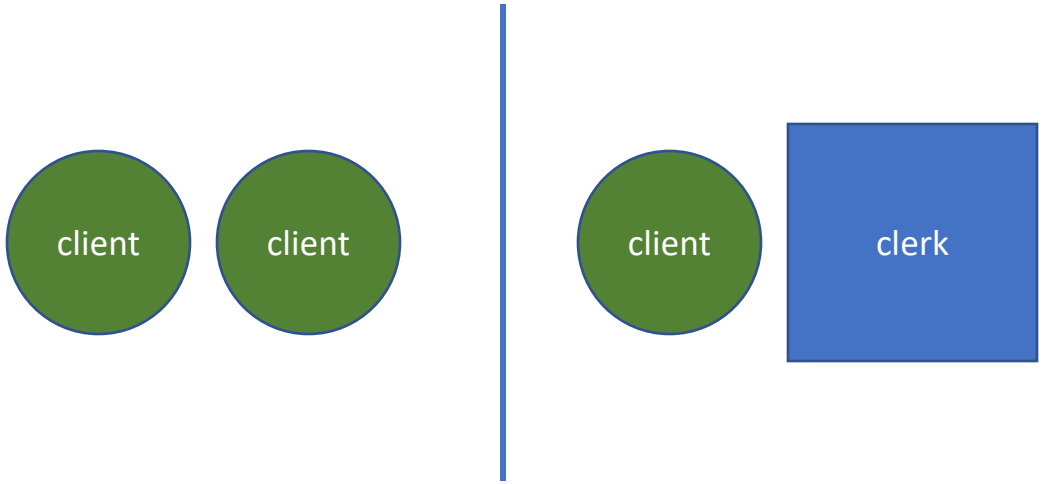
what is a random variable?
- the output of an experiment

what is a distribution function?
- rule used to define the probability associated with an event

what is a discreet distribution?
- when the sample space is made up of a countable number of points

what is a continuous distribution?
- when the sample space is made up of an infinite number of points





Simulation results can be quite sensitive to input probability distributions

For example, lets look at an experiment where we are trying to determine the average delay for customers in queue.
In this experiment the time it take for a clerk to help a client, the random variable, can cause a great difference in delay.

TABLE 6.2
Simulation results for the five service-time distributions (in minutes where appropriate)

Service-time distribution	Average delay in queue	Average number in queue	Proportion of delays ≥20
Exponential	6.71	6.78	0.064
Gamma	4.54	4.60	0.019
Weibull	4.36	4.41	0.013
Lognormal	7.19	7.30	0.078
Normal	6.04	6.13	0.045

how we determine the service time is very important!

How do we determine the input probability distribution?

If it is possible to collect data on an input random variable of interest, we can use one of the following approaches

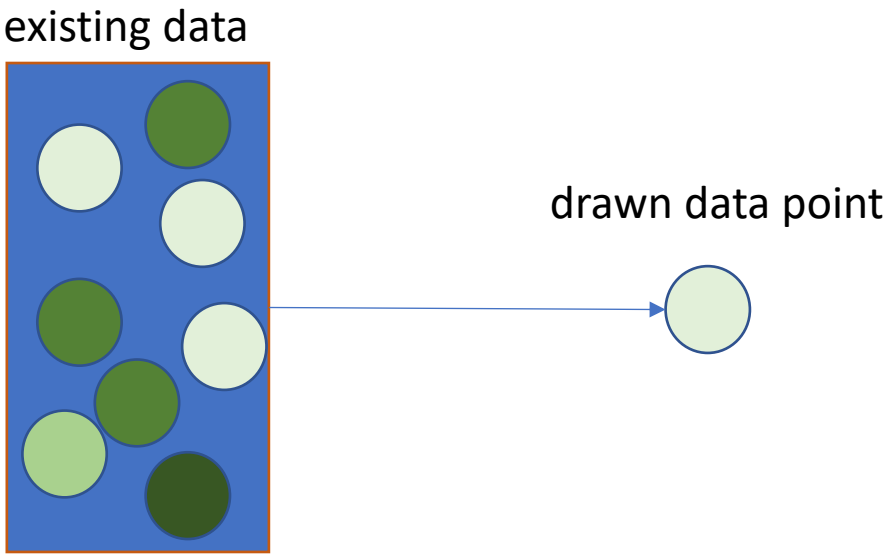
- The data values themselves are used directly in the simulation. This is sometimes called a trace-driven simulation.
  - For example, if the data represent service times, then one of the data values is used whenever a service time is needed in the simulation.
- The data values themselves are used to define an empirical distribution function.
  - If these data represent service times, we would sample from this distribution when a service time is needed in the simulation.
- “fit” the data to a theoretical distribution form
  - for example, exponential or Poisson
  - If a particular theoretical distribution is a good model for the service-time data, then we would sample from this distribution when a service time is needed in the simulation.

what if you have no data? ☹

- This is not a rare situation, data is hard to get if it is proprietary or if its sensitive information
- No standard “good” approach to dealing with lack of data, but some options are:
  - Use expert input
  - estimate the range and mode
  - think of Physical reason to choose a particular distribution
    - Inter-arrival time: Exponential
    - Number of “random” events in an interval: Poisson
    - Sum of independent random events: normal
    - Product of independent random events: lognormal

trace-driven approach

draw data from existing data in the simulation



Pros:

- useful for model validation. Can compare the expected and actual data easily
- makes system comparison easy

Con:

- data can quickly become unwieldy to store
- can only reproduce situation that have happened historically (there is seldom enough data to paint the nuance of real world clearly

Empirical distribution approach

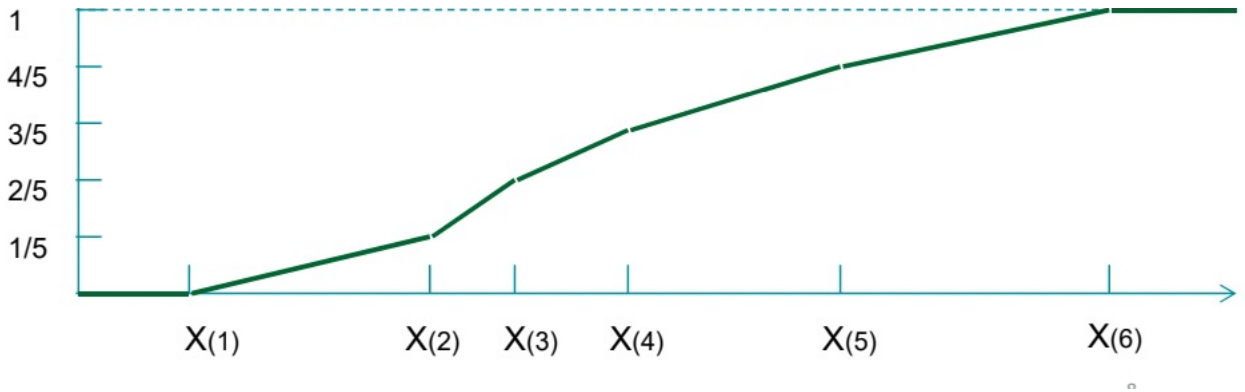
create distribution from data to specify a probability distribution

Pros:

- better then trace driven
- great when there doesn't exist a good theoretical distribution to use

Con:

- can not extrapolate beyond the range of the data
- difficult to perform sensitivity test



use sorted data directly to create a piecewise linear CDF to draw random variates from

Parametric distribution approach

determining a parametric distribution that can represent the data

Pros:

- smooths out data irregularities
- greater predictive power, allows for outliers not seen in the data
- fast
- no need to store data, just need distribution parameters

Con:

- not always possible to find

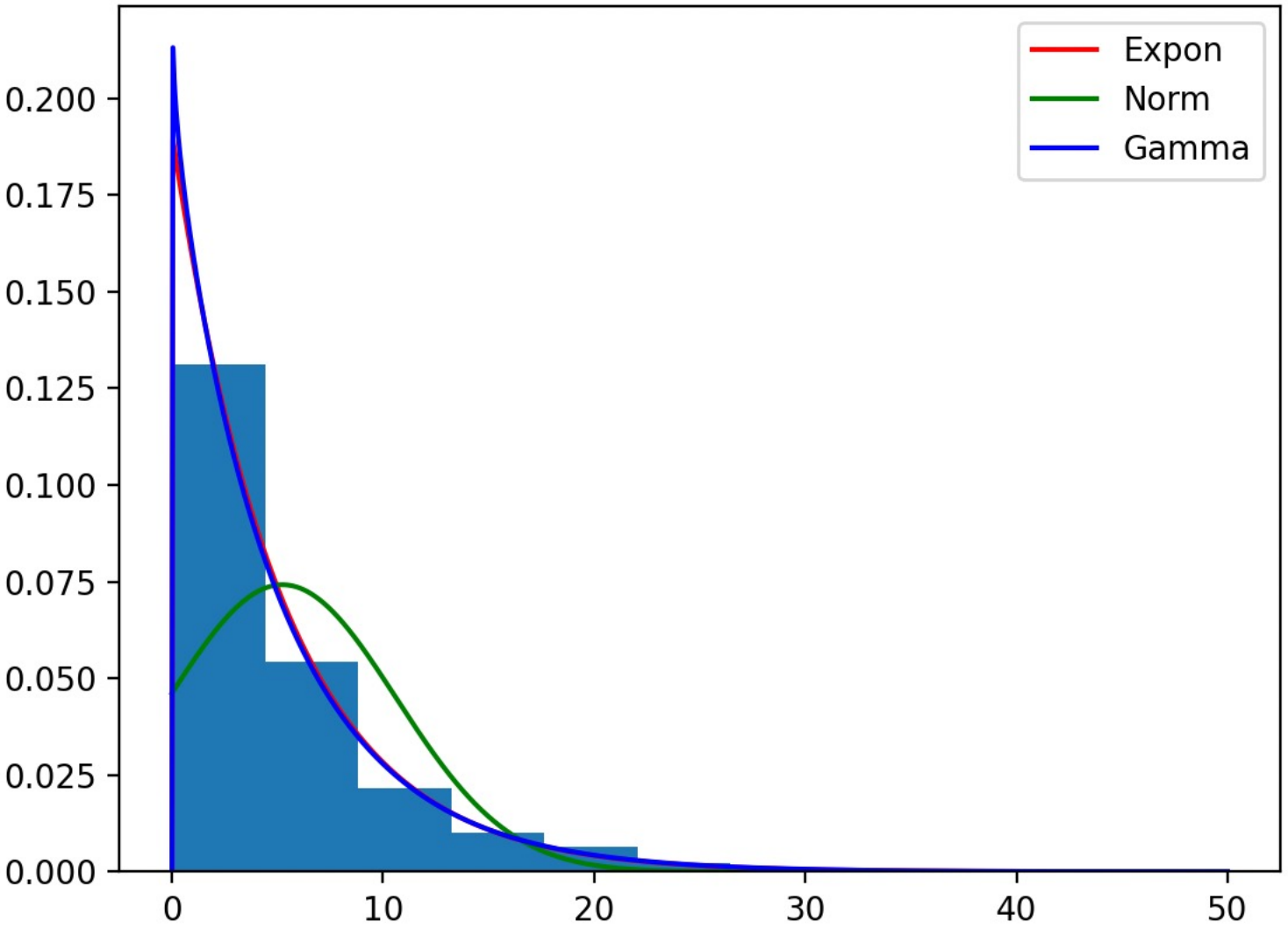
- **U(a,b): a = min X_i, b = max X_i**
- **Expo(λ):** $1/\bar{X}(n)$
- **Gamma: numerical solution, but easily available by method of moment**
- **Weibull: numerical solution**
- **N(μ, σ^2):** $\bar{X}(n), \frac{n-1}{n}S^2(n)$
- **LN(μ, σ^2):** $\frac{\sum_{i=1}^{n} \log X_i}{n}$, $\frac{\sum_{i=1}^{n} (\log X_i - \hat{\mu})^2}{n}$
- **Beta: numerical solution**
- **Discrete U(a,b): a = min X_i, b = max X_i**
- **Bernoulli(*p*):** $\bar{X}(n)$
- **Poisson(λ):** $\bar{X}(n)$

steps to hypothesize parametric family

Three steps to Hypothesizing families of distributions, given data independence

- What distributions may be appropriate based on shape?
  - Do we have any physical or historical reason to choose some?
- Estimate the parameters for candidate probability distributions
- Determine the “quality” of fitted distributions
  - Graphical heuristic procedures
  - Statistical goodness-of-fit tests

How do we determine quality of fit?



Several methods, there are many more

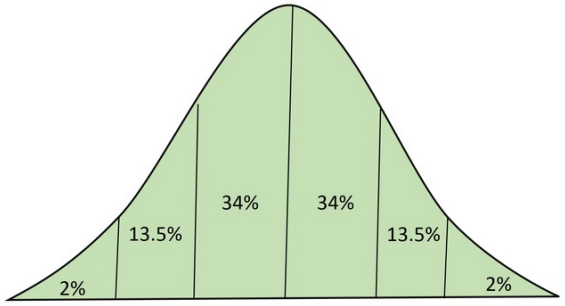
visual quality of fit tests

- looking at the histogram and theoretical distribution
- Q-Q plot

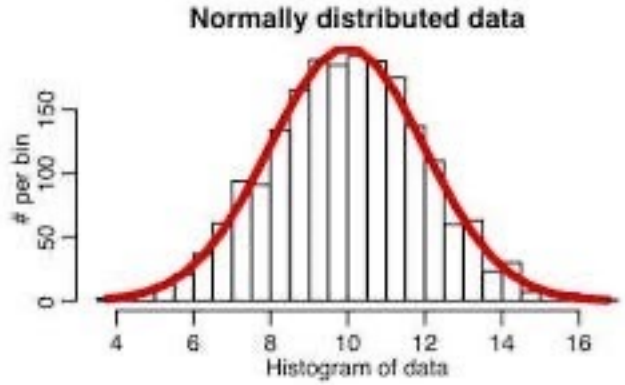
goodness of fit tests

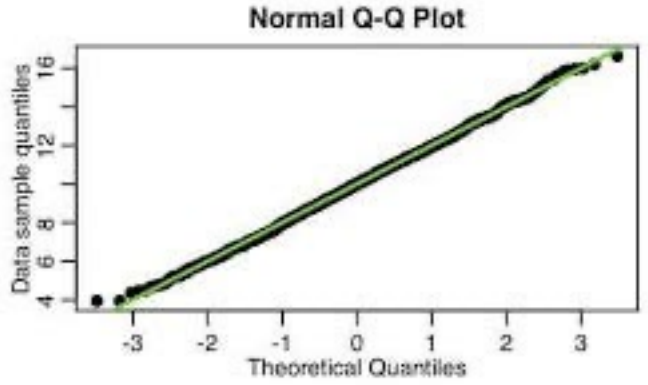
- chi-square tests
- Kolmogorov-Smirnov Tests

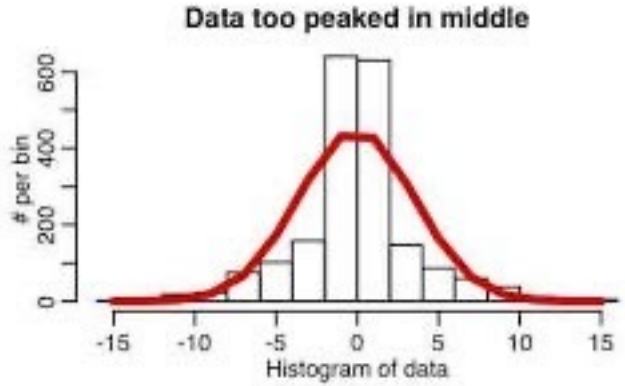
Q-Q plot

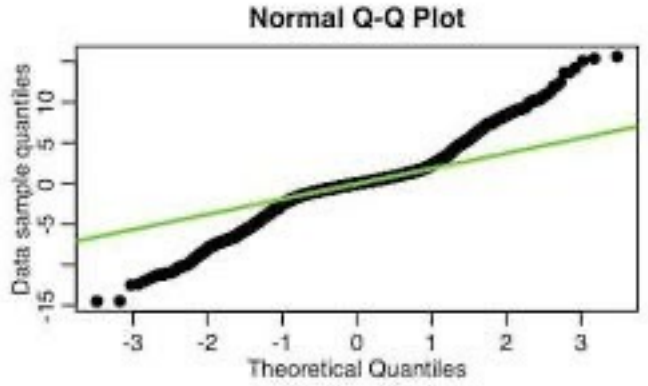


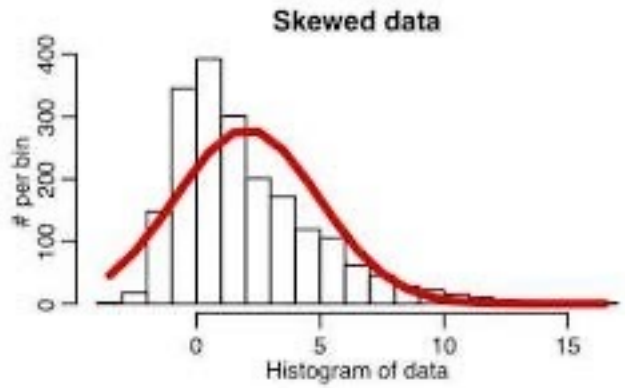
- visual tool to determine the quality of fit by looking at how the partitions (quantiles) in the data match up
  - is the 50% mark around the same place?
- plots the Empirical distribution quantiles with respect to theoretical distribution quantiles
- If the theoretical distribution and true underlying distribution are the same, then the probability distributions will match and the Q–Q plot will be approximately linear with an intercept of 0 and a slope of 1.

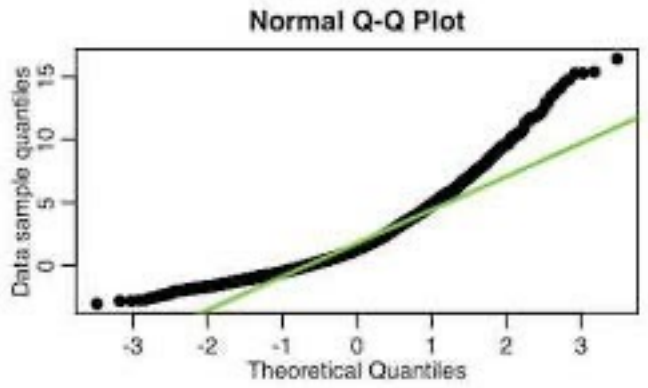




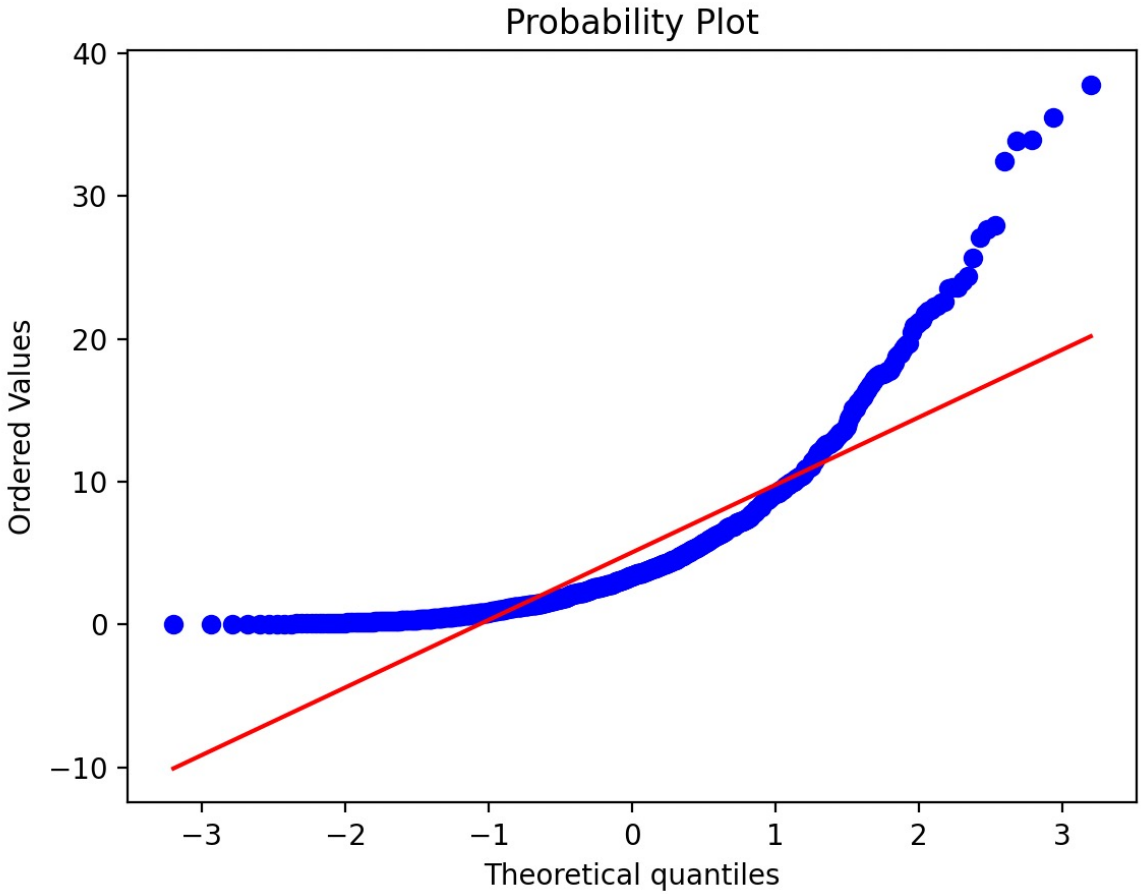




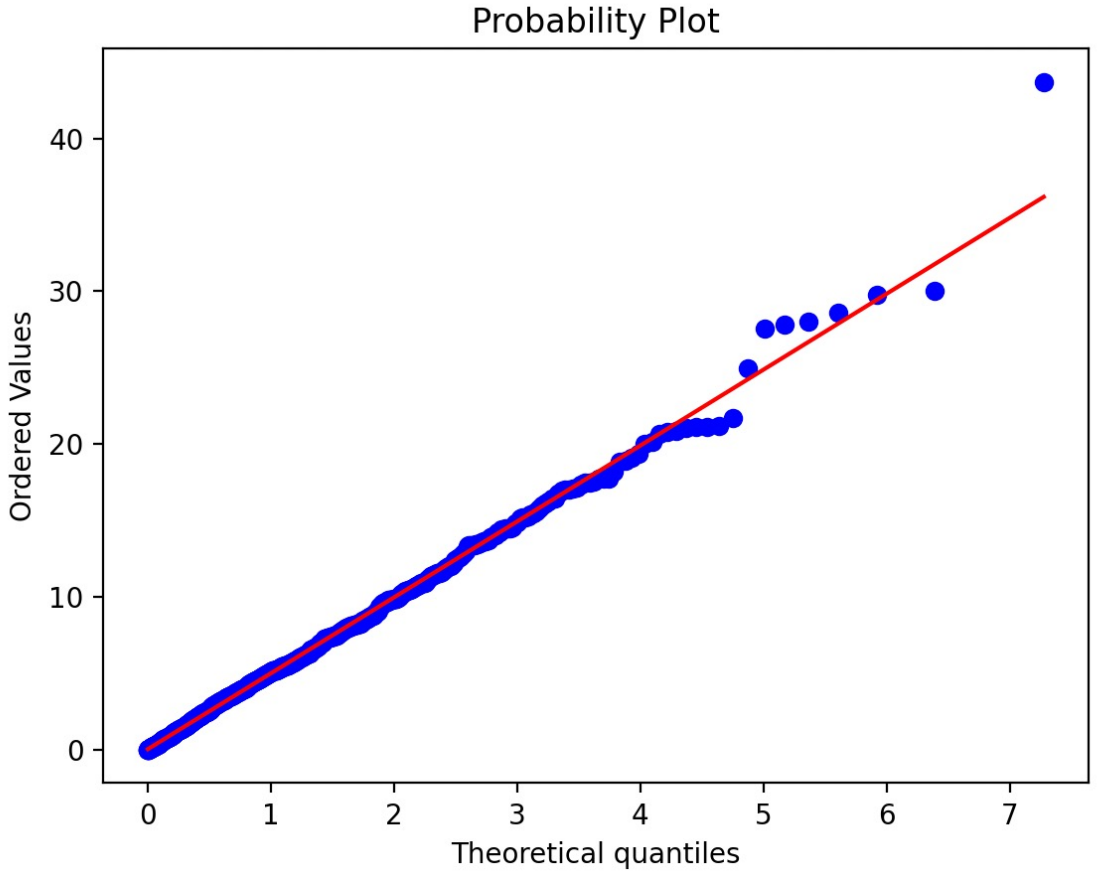




normal qq plot



exponential qq plot



Even if F^(x) is the correct distribution, there will be departures from linearity for small to moderate sample sizes

Determining goodness of fit

Can use statistical tests to formally evaluate if the data set is an independent sample from the fitted distribution.

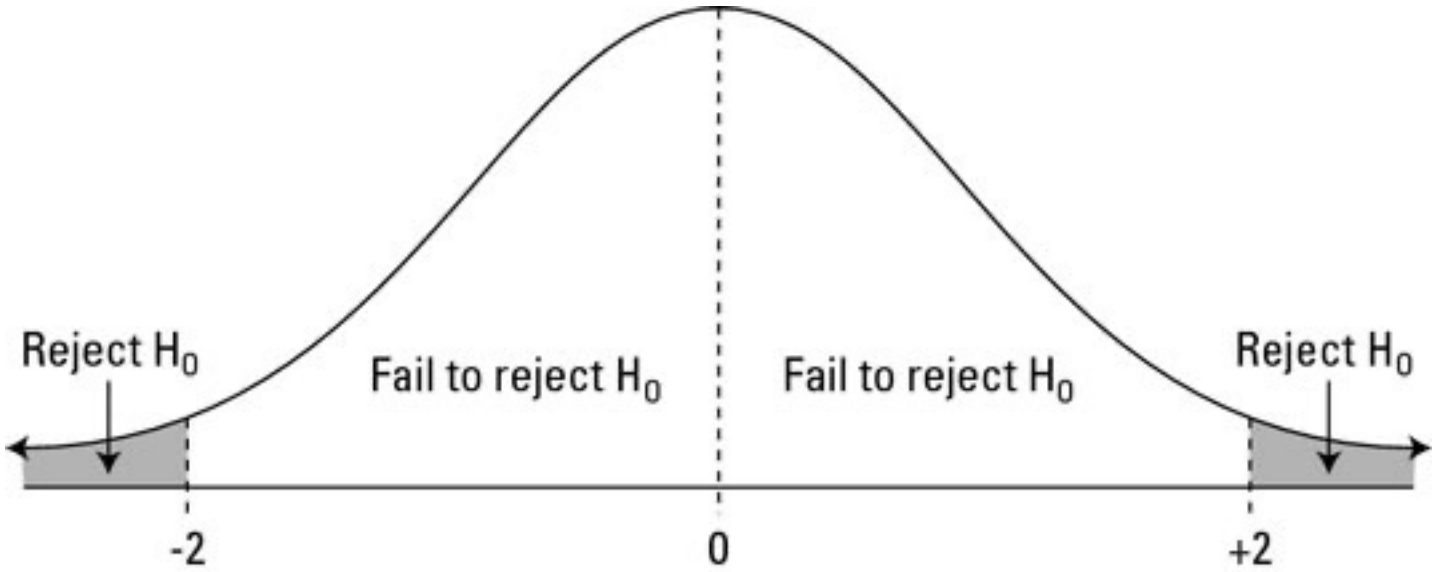
Using the null hypothesis, H0, the claim being studied, we can analyze the claim that the points's are IID random variables with distribution function .

There are two results that you can get from a hypothesis test:

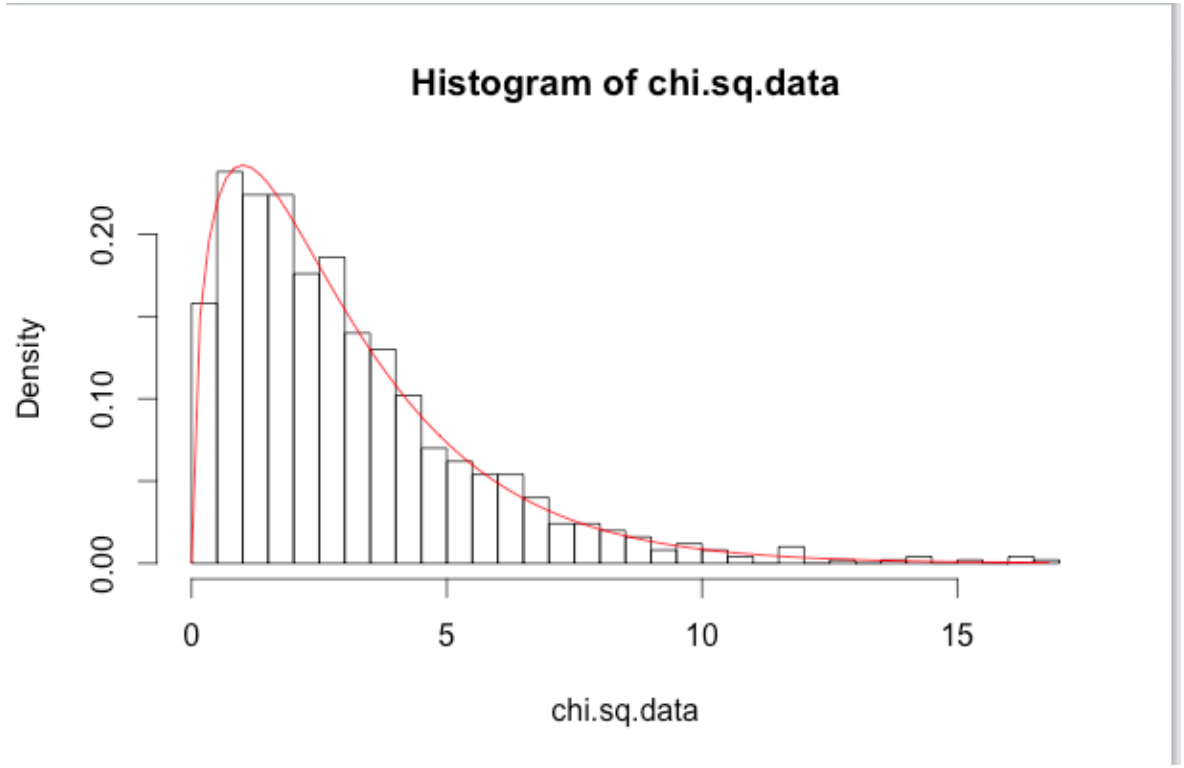
- Reject the null hypothesis if the probability, the p value is too low
- Fail to reject the null hypothesis when the p value is above a specific thresh-hold

The p-value is this case the probability that the data comes from a specific distribution.

- if the probability that the data is normally distributed is .00013%, we should reject the null hypothesis that the data is normally distributed
- if the probability that the data is exponentially distributed is .93%, we should fail to reject the null hypothesis that the data is exponentially distributed



Chi square test



a chi-square test a statistical hypothesis test that compares a histogram of the data with the fitted density or mass function

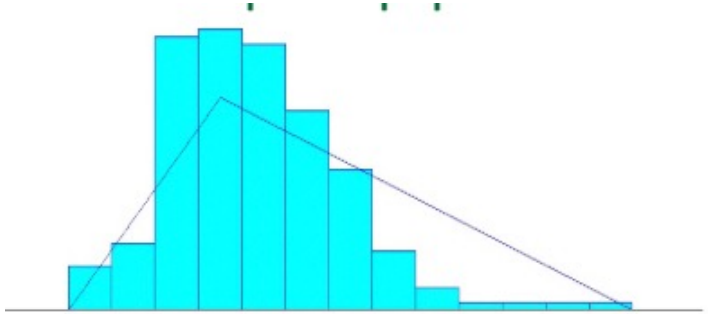
$$\chi^2 = \sum_{j=1}^{k} \frac{(N_j - np_j)^2}{np_j}$$

Because we are making the comparison to a histogram, we must first divide the entire range of the fitted distribution into k adjacent intervals

this step can be an issue because there is a clear lack of prescription for interval selection.

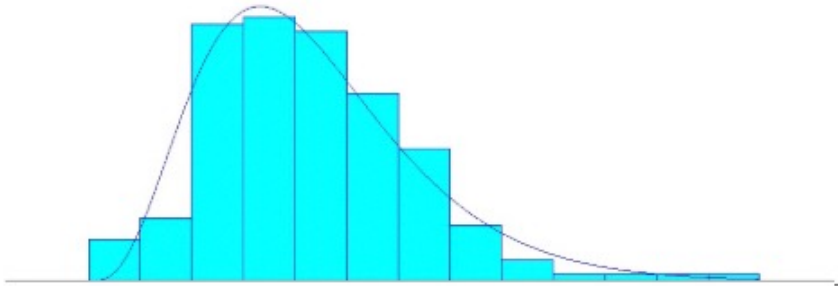
- The value you get back from this test is rather sensitive to the choice interval size

The chi-square test nevertheless remains in wide use, because it can be applied to any hypothesized distribution.



Distribution: Triangular
Expression: TRIA(3, 5.69, 13)
Square Error: 0.024551

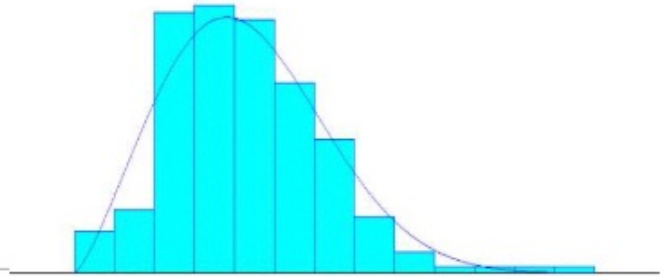
Chi Square Test
- Number of intervals = 10
- Degrees of freedom = 8
- Test Statistic = 60.6
- Corresponding p-value < 0.005



Distribution: Gamma
Expression: 3 + GAMM(0.775, 4.29)
Square Error: 0.003873

Chi Square Test
- Number of intervals = 7
- Degrees of freedom = 4
- Test Statistic = 4.68
- Corresponding p-value = 0.337

Kolmogorov-Smirnov Test



Distribution: Weibull
Expression: 3 + WEIB(3.75, 2.3)
Square Error: 0.004426

Chi Square Test
- Number of intervals = 7
- Degrees of freedom = 4
- Test Statistic = 5.97
- Corresponding p-value = 0.213

Kolmogorov-Smirnov Test

MASON

Kolmogorov-Smirnov Tests

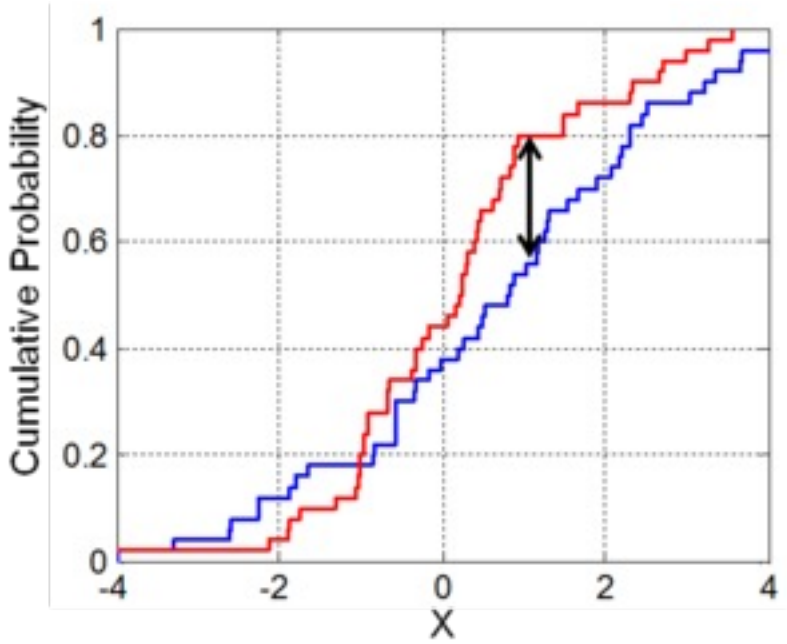
Kolmogorov Smirnov (K-S) also tests for goodness of fit,
It compare an empirical distribution function with the distribution function of the hypothesized distribution.

PROS

- Unlike the chi square test, K-S tests do not require us to group the data in any way, so no information is lost; this also eliminates the troublesome problem of interval specification.
- Finally, K-S tests tend to be more powerful than chi-square tests against many alternative distributions

CONS

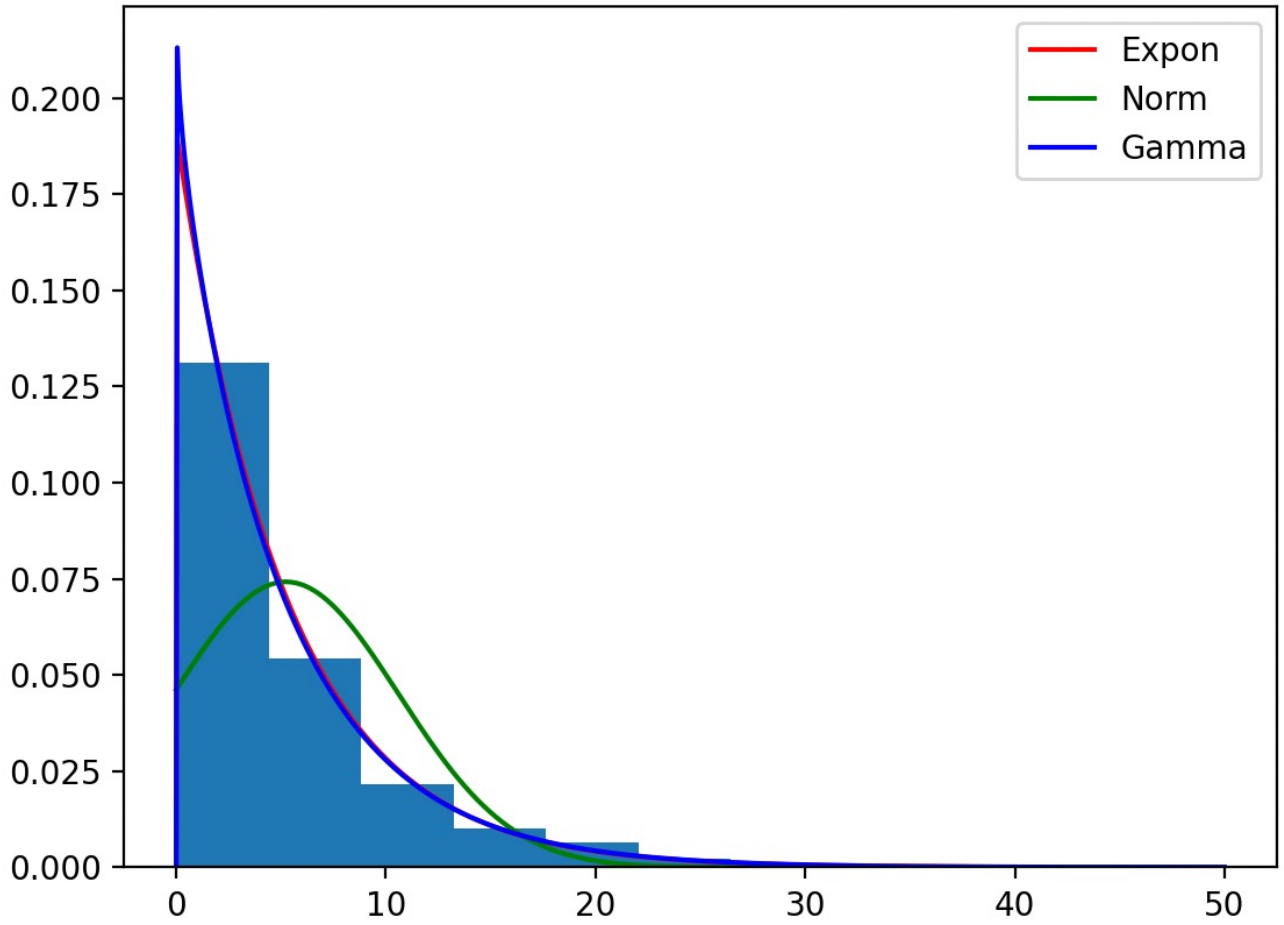
- hard to apply to discreet distributions
- If the parameters of a distribution are estimated from data, we can only look at a limited number of distributions



looks at the max difference between the theoretical and empirical distributions

$$\text{K-S test statistic} = \sup_x \left| F_a(x) - F_b(x) \right|$$

sup – supremum (basically the max difference between the two graphs)



exponential
K-S test
statistic=0.02197015
pvalue=0.711327684368124

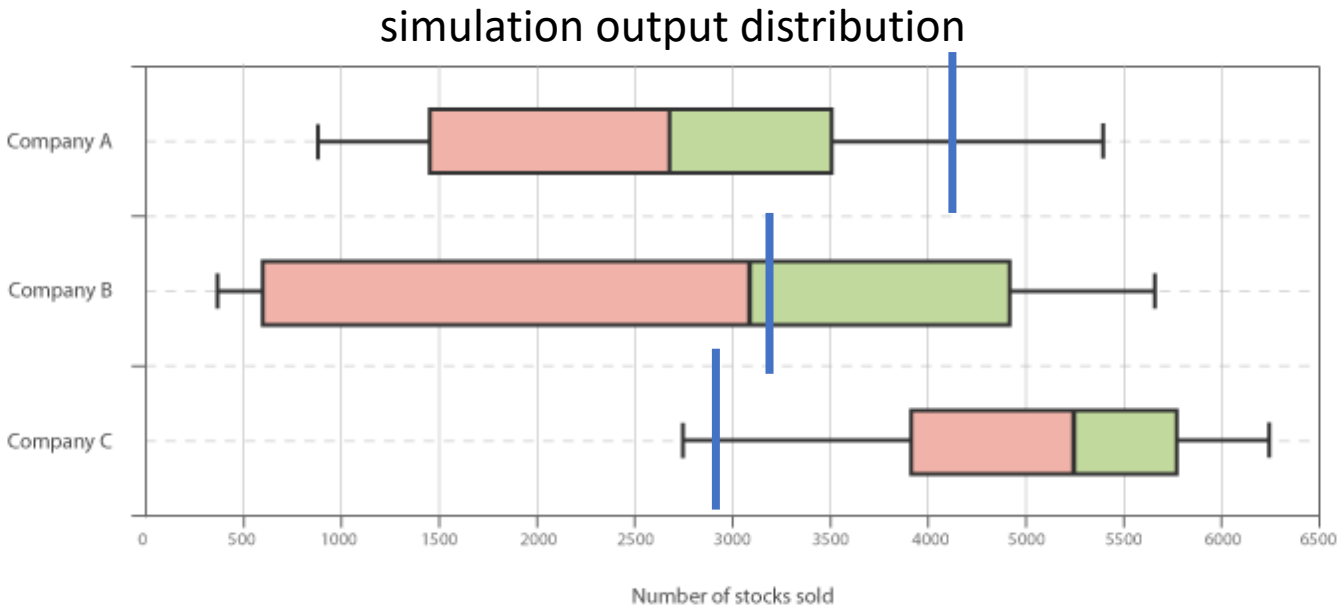
Normal
K-S test
statistic=0.1544289526028424
pvalue=2.723222970239228e-21

gamma
K-S test
statistic=0.028312253401895 65
pvalue=0.3919833432821260 3

output analysis

cool you have a model, now what?

- How do we use the simulation model we made to make a decision?
- Because of the stochastic nature of simulations: we will want to run multiple replications of a simulation in order to get a better idea the output distribution



if we were to make a decision based on only one of the simulation, what company would we say is best?

multiple replications is needed to make a more precise estimate of performance

In order to obtain a point estimate of the mean and standard deviation
we can take n independent replications of the simulation to get x1,x2,……..xn iid random variable
and find the mean X(n)

we can assume that as n, the number of replications, increases. we will approach the true mean and standard deviation

EXAMPLE 9.15. For the inventory system of Sec. 1.5 and Example 9.8, suppose that we want to obtain a point estimate and an approximate 95 percent confidence interval for the expected average cost over the 120-month planning horizon, which is given by

$$E(X) = E\left(\frac{\sum_{i=1}^{120} C_i}{120}\right)$$

We made 10 independent replications and obtained the following X_j's:

129.35	127.11	124.03	122.13	120.44
118.39	130.17	129.77	125.52	133.75

which resulted in

$$\overline{X}(10) = 126.07, \qquad S^2(10) = 23.55$$

and the 95 percent confidence interval

$$126.07 \pm 3.47 \qquad \text{or, alternatively,} \qquad [122.60, 129.54]$$

Comparing model

- The real utility of computer simulation often lies in the comparison of different system design/configuration alternatives without the need of time/resource-consuming or impossible physical experiments
- Making recommendations based on simulation results can be error prone, using proper output analysis tools for the job is important

Comparing two designs (systems)

A paired t-test is used to compare two population means where you have two samples in which observations in one sample can be paired with observations in the other sample.

this is another null hypothesis test

$$t_{calc} = \frac{\bar{d}}{s_d / \sqrt{n}}$$

t Table

cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$	$t_{.9995}$
one-tail	0.50	0.25	0.20	0.15	0.10	0.05	0.025	0.01	0.005	0.001	0.0005
two-tails	1.00	0.50	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.002	0.001
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073
16	0.000	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921	3.686	4.015
17	0.000	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898	3.646	3.965
18	0.000	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878	3.610	3.922
19	0.000	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.579	3.883
20	0.000	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.552	3.850
21	0.000	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.527	3.819
22	0.000	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.505	3.792
23	0.000	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.485	3.768
24	0.000	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.467	3.745
25	0.000	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.450	3.725
26	0.000	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.435	3.707
27	0.000	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.421	3.690
28	0.000	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.408	3.674
29	0.000	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.396	3.659
30	0.000	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.385	3.646
40	0.000	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704	3.307	3.551
60	0.000	0.679	0.848	1.045	1.296	1.671	2.000	2.390	2.660	3.232	3.460
80	0.000	0.678	0.846	1.043	1.292	1.664	1.990	2.374	2.639	3.195	3.416
100	0.000	0.677	0.845	1.042	1.290	1.660	1.984	2.364	2.626	3.174	3.390
1000	0.000	0.675	0.842	1.037	1.282	1.646	1.962	2.330	2.581	3.098	3.300
z	0.000	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576	3.090	3.291
	0%	50%	60%	70%	80%	90%	95%	98%	99%	99.8%	99.9%
	Confidence Level										

Common random number

- One of the most well known and used variance-reduction technique

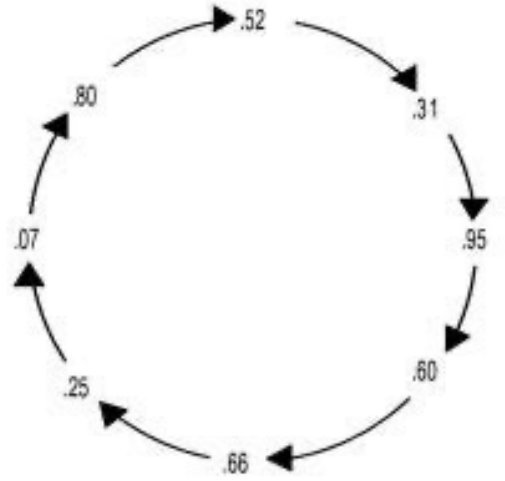
- random numbers in a simulation are generated from a random number stream
  - a stream uses a starting seed number to recursively generate a series of numbers
  - each starting seed generates a unique stream
  - controlled randomness

Dedicating a specific stream for each source of randomness allows us to more sufficiently compare models under the same circumstance

in python you can set the random seed using the following

```
import random

random.seed(10)
print(random.random())
```



Example of a Random Stream Cycle with a Very Short Period

random number equation

$$X_{n+1} = [a * X_n] \pmod{m}$$

where: $a = 16{,}807$, and $m = 2^{31} - 1$

How do we efficiently compare multiple systems and find the best one?

more complicated

- Pairwise comparison tests
- compare to a standard system
- Heuristics
- ranking and selection techniques

Ranking & Selecting Techniques

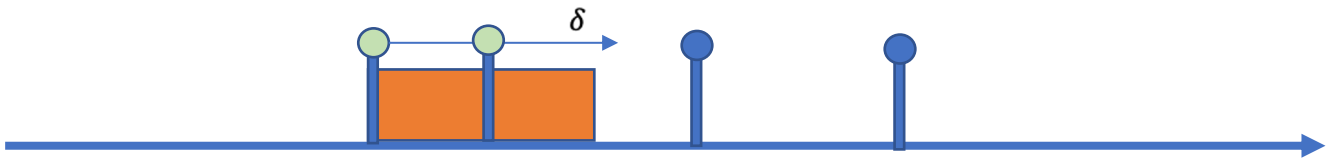
This refers to the process of choosing the best of two or more processes (designs) according to a performance measure of some sort

Indifference Zone Techniques

This is a frequentist technique, meaning the probability of correct solution is with respect to the repeated simulations. Indifference zone techniques guarantees that we choose the best solution with a certain probability.

In cases where we have solutions that are very close to each other, we define a indifference parameter delta, such that any solution with in delta of the best solution is acceptable

indifference zone ranking and selection guarantees that we will either select the best value or one with a performance measure within δ with a certain probability



Indifference zone method - Rinott's procedure

Two stage procedure – two rounds of simulations

Basic procedure

- Determine the initial number of simulation (n_0), the number of designs (k), and the probability of selecting the best solution (1-α)

- Look up the rinnot constant: h(k,n_0, 1-α)

- start with an initial count of n_0 iid simulations for each design

- Using the sample variances of the n_0 simulations for each design, the rinnot constant, and the desired indifference zone compute how many more simulations are required

$$N_i = \frac{h^2 S(x)^2}{\delta^2}$$

- Take $(N_i - n_0)^+$ additional simulations for each design

- Compute the means of each

$$\bar{Y}_i = \sum_{j=1}^{N_i} Y_{ij}/N_i$$

- Select the best design

Simple Example

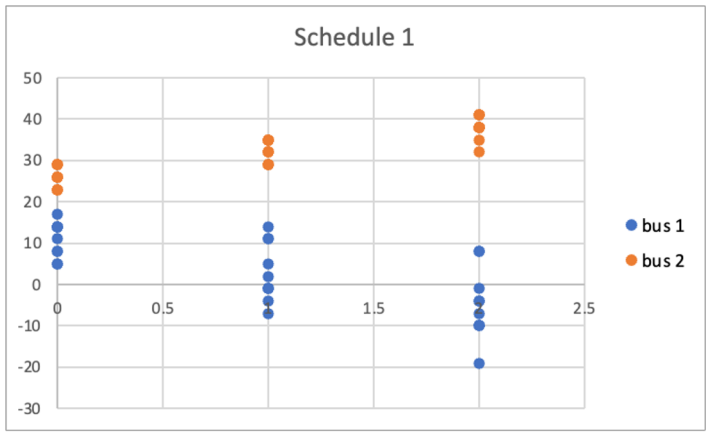
- Step 1 Determine the initial number of simulation (n_0), the number of designs (k), and the probability of selecting the best solution (1-α)

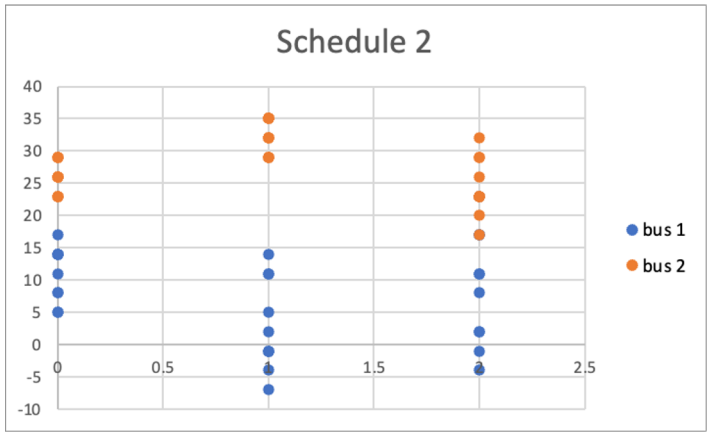
k = 8 designs
1-α = .95
n_0 = 10

- Step 2 Look up the rinnot constant: h(k,n_0, 1-α)

h = 4.2287

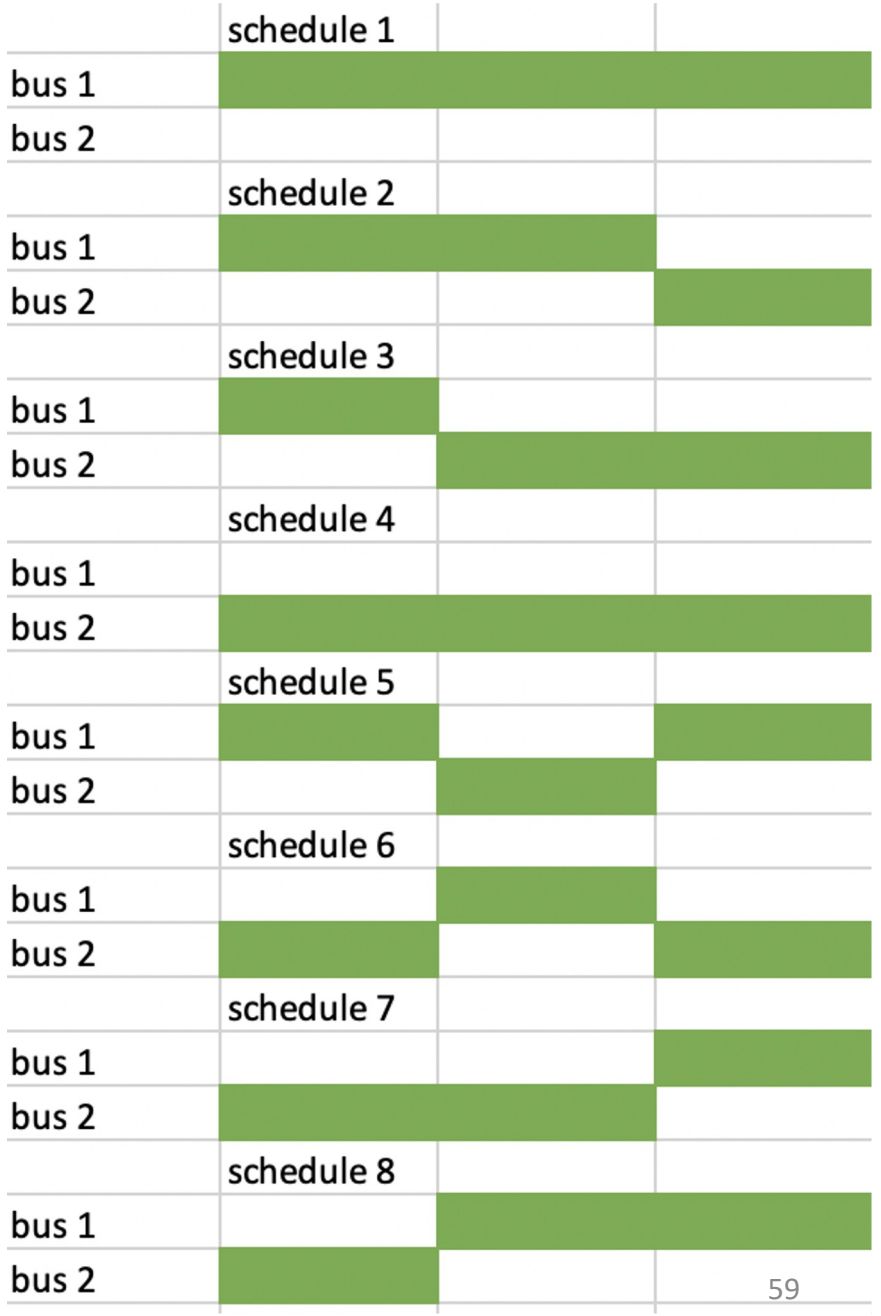
- Step 3 Start with an initial count of n_0 iid simulations for each design

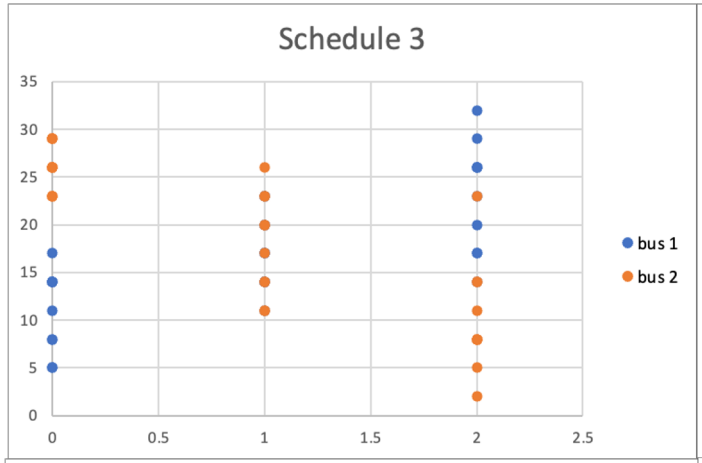


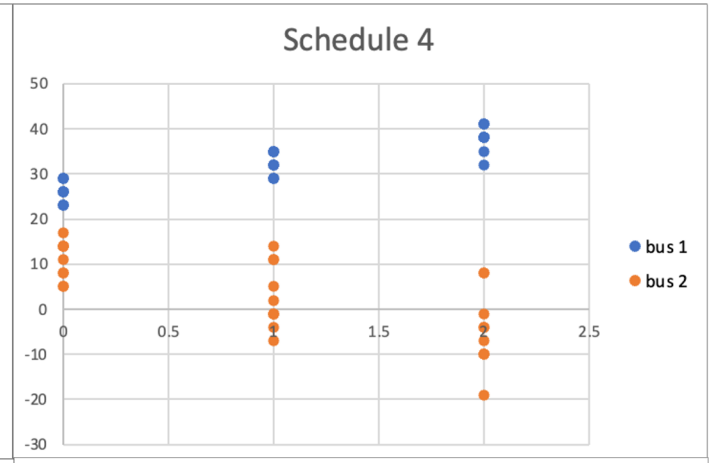


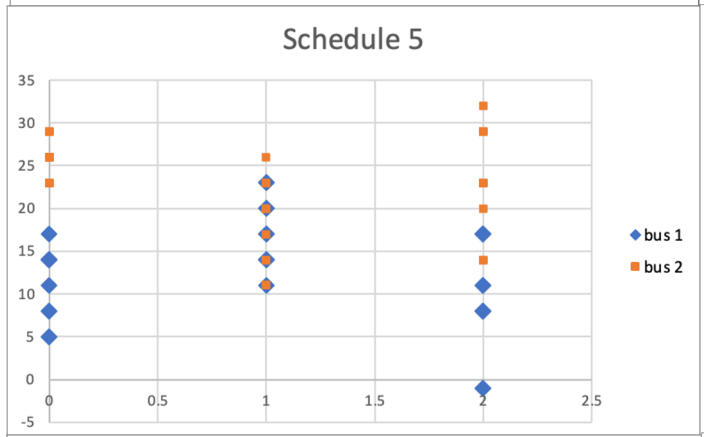
Sample output for 10 simulations for design 1 and 2

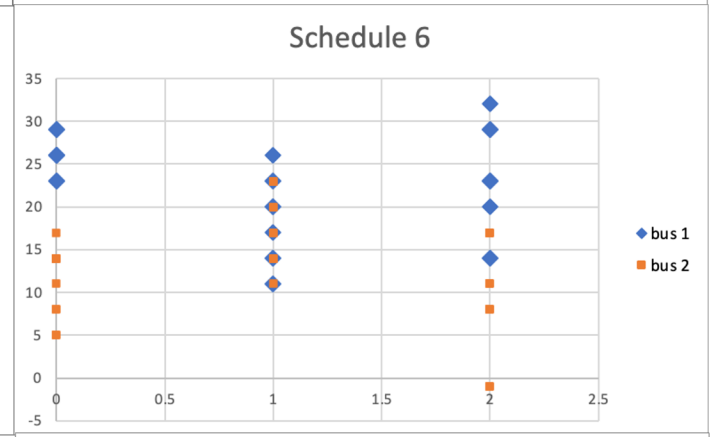
Sample Bus Schedule: 2 buses, 3 shifts, 1 bus in use at a time

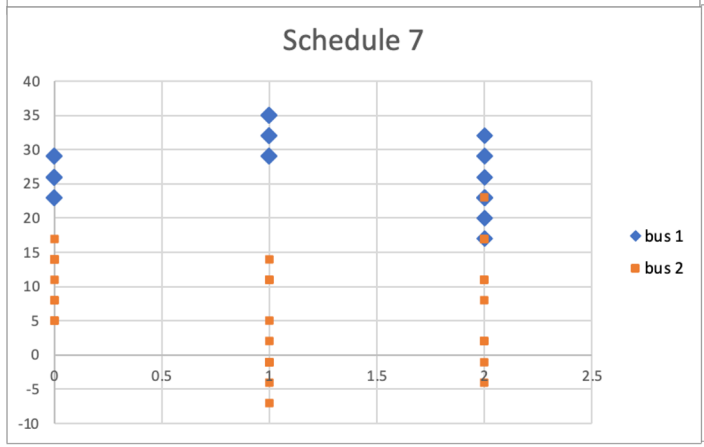


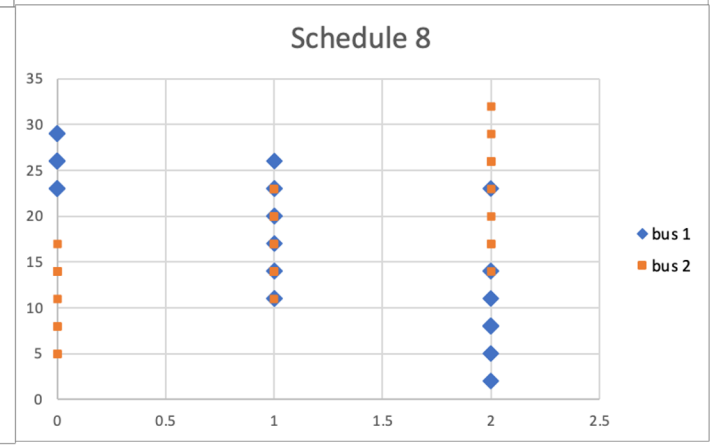

Schedule 3
bus 1
bus 2

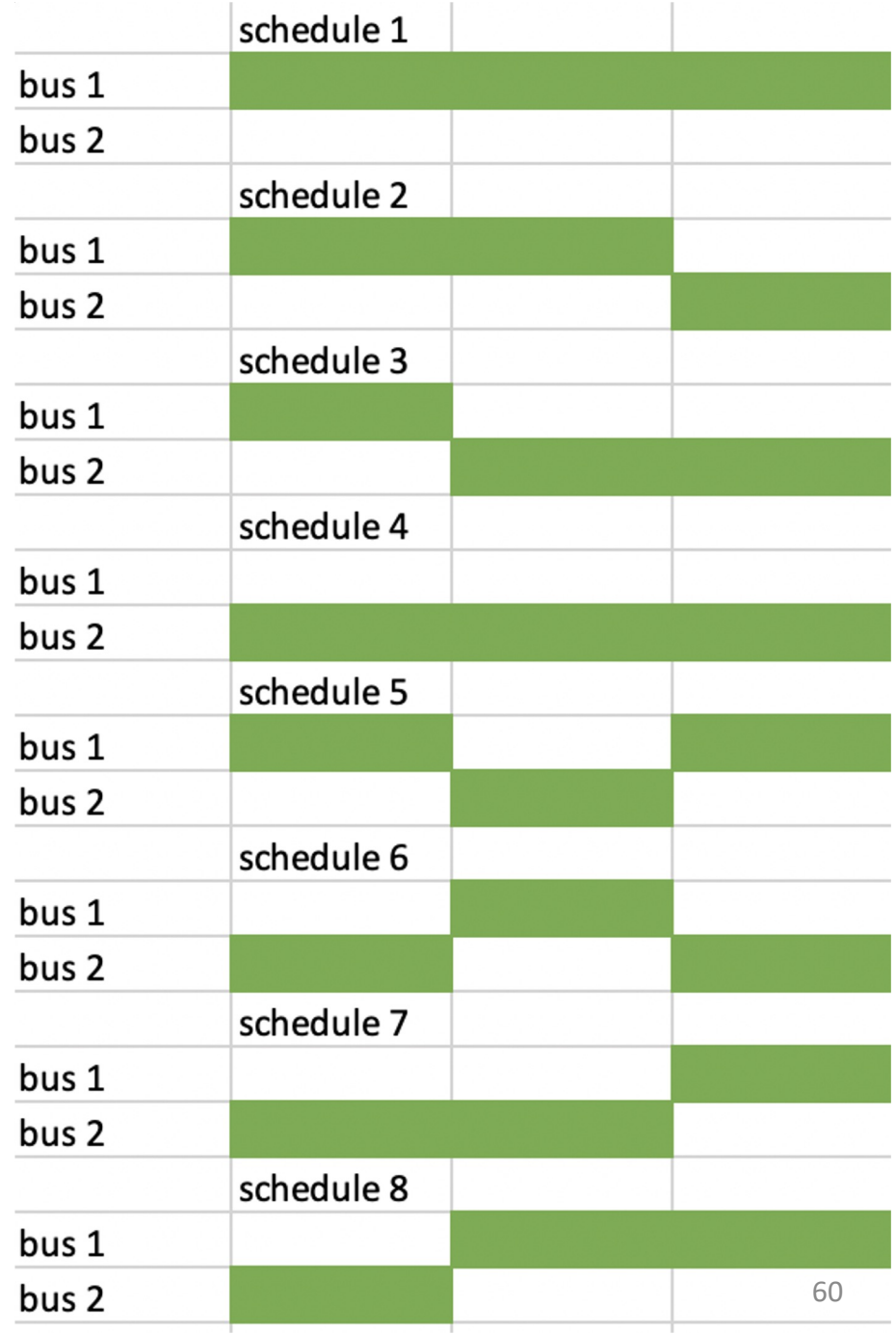

Schedule 4
bus 1
bus 2


Schedule 5
bus 1
bus 2


Schedule 6
bus 1
bus 2


Schedule 7
bus 1
bus 2


Schedule 8
bus 1
bus 2

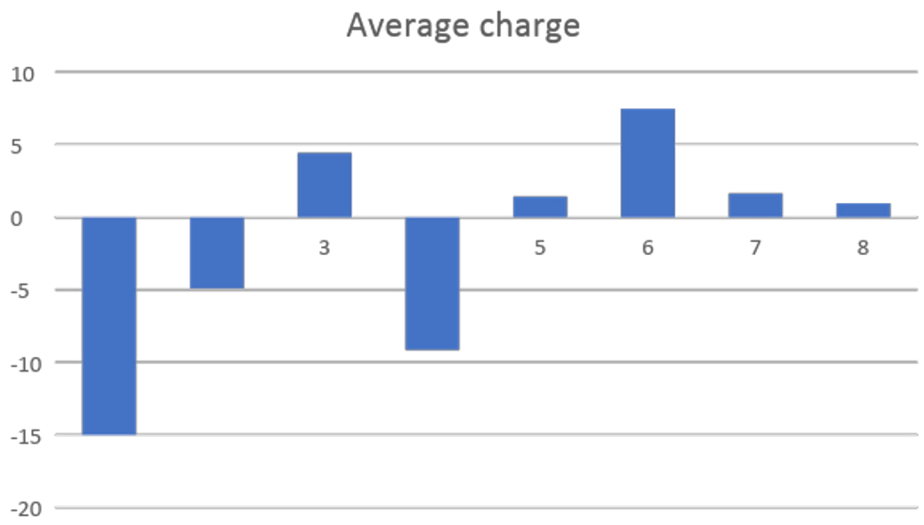

schedule 1
bus 1
bus 2
schedule 2
bus 1
bus 2
schedule 3
bus 1
bus 2
schedule 4
bus 1
bus 2
schedule 5
bus 1
bus 2
schedule 6
bus 1
bus 2
schedule 7
bus 1
bus 2
schedule 8
bus 1
bus 2

- Step 4: Using the sample variances of the n_0 simulations for each design, the Rinnot constant, and the desired indifference zone, compute how many simulations are required:

variances: [54.9, 59.6, 61.6, 126.5, 96.0, 118.4, 45.6, 77.6]
N_i: [246, 267, 276, 566, 430, 530, 204, 347]

  - Take $(N_i - n_0)^+$ additional simulations for each design
  - Compute the means of each
  - Select the best design

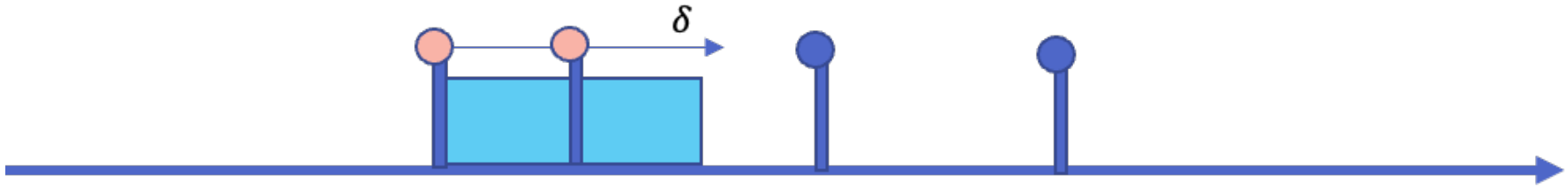
Final Means: [-15.02, -4.89, 4.44, -9.16, 1.44, **7.47**, 1.65, 0.95]



- **Schedule 6** has the highest positive mean, thus the best schedule.

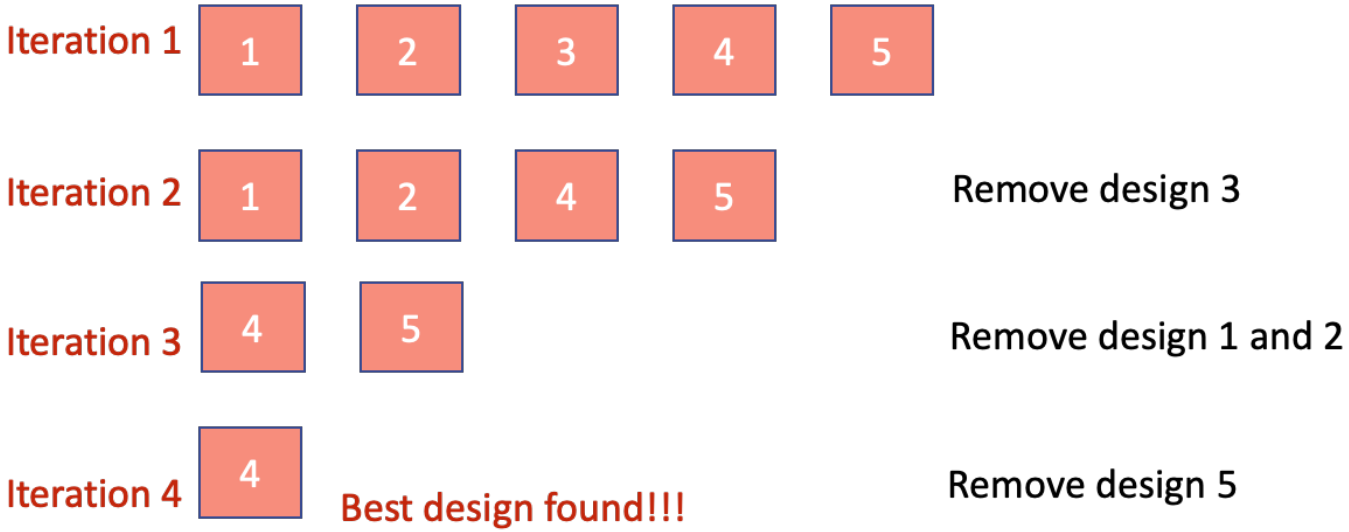
Picking the Best Design

With indifference zone procedures we define a value δ that defines an indifference zone around th
Such that we are indifferent to picking another design within δ of it



Elimination strategy – subset selection

Iteratively replicate, eliminate, replicate, …. Designs until only 1 is left
Gain efficiency in the number of replications needed by eliminating clearly bad designs.



Picking the Best Design: Subset Selection

Relation between Brownian motion and Rank and Select procedure

The subset selection procedure we are using uses the Brownian motion property of scaling and dr

$$\sigma\mathcal{B}\left(\mathrm{t};\frac{\delta}{\sigma}\right) = \sigma\mathcal{B}(\mathrm{t}) + \delta t$$

Let ,

$D_x(r) = \sum_{j=1}^{r} Y_j(k) - Y_j(x)$ The sum of the differences between design k and design x

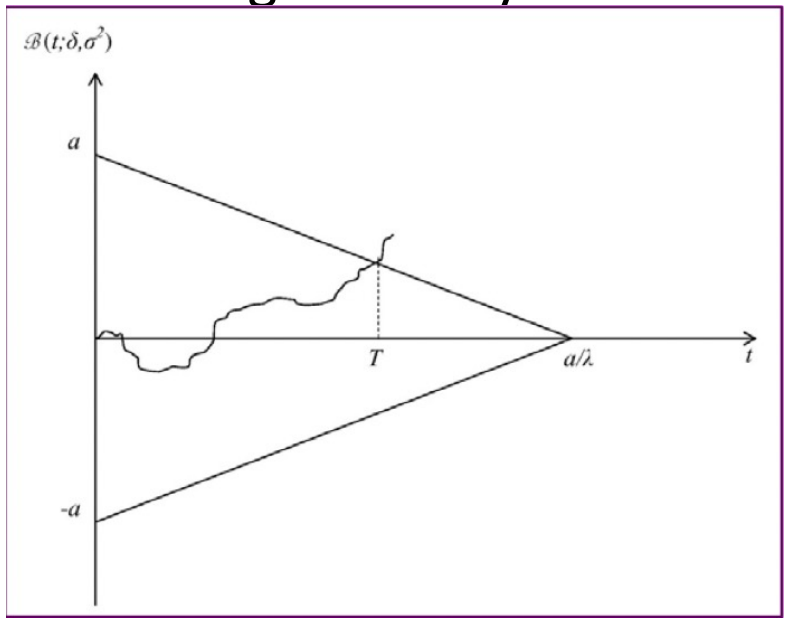
$\sigma^2{}_{kx} = Var(Y_j(k) - Y_j(x))$ The variance of the difference between design k and x

$\delta_{kx} = \mu(k) - \mu(x)$ The difference of the means of design k and x

Brownian motion can be thought of as a continuous analog for summing normally distributed rand

$$\{D_x(r); r = 1,2, \dots\} \approx \{\sigma\mathcal{B}\left(\mathrm{t};\frac{\delta}{\sigma}\right); r = 1,2, \dots.\}$$

We can use this concept to find the probability of exiting the region in the wrong direction in the Paulson's procedure



Picking the Best Design:
Indifference Zone Method- Paulson Procedure

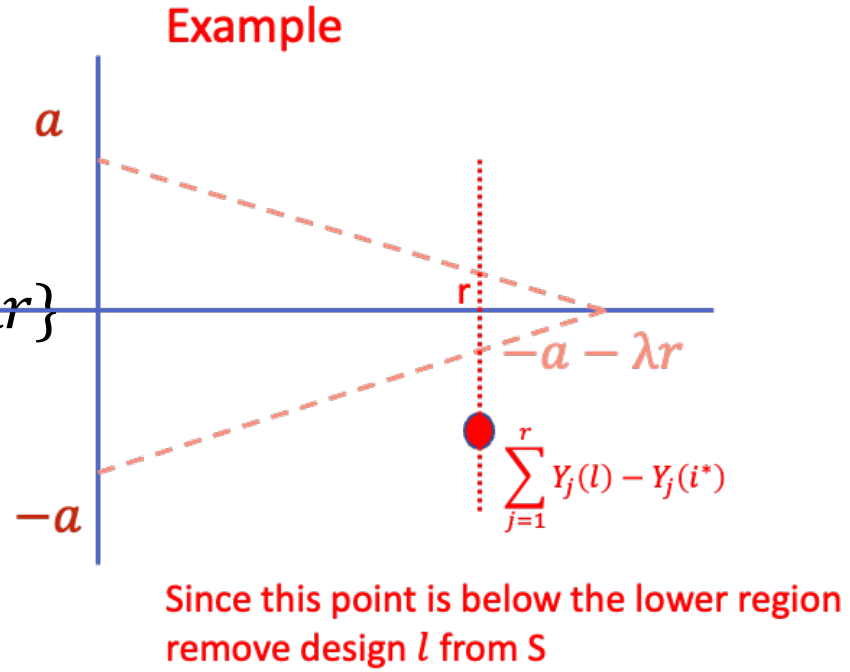
First fully sequential ranking and selecting procedure

1. Create a set S that contains {1,2,…,k} designs, choose a $\lambda \in (0, \delta)$, set $\alpha = \frac{\sigma^2}{\delta - \lambda} \ln\left(\frac{k-1}{\alpha}\right)$

2. Set r+=1, Simulate $Y_r(x), \forall x \in S$

3. Mark designs for elimination if

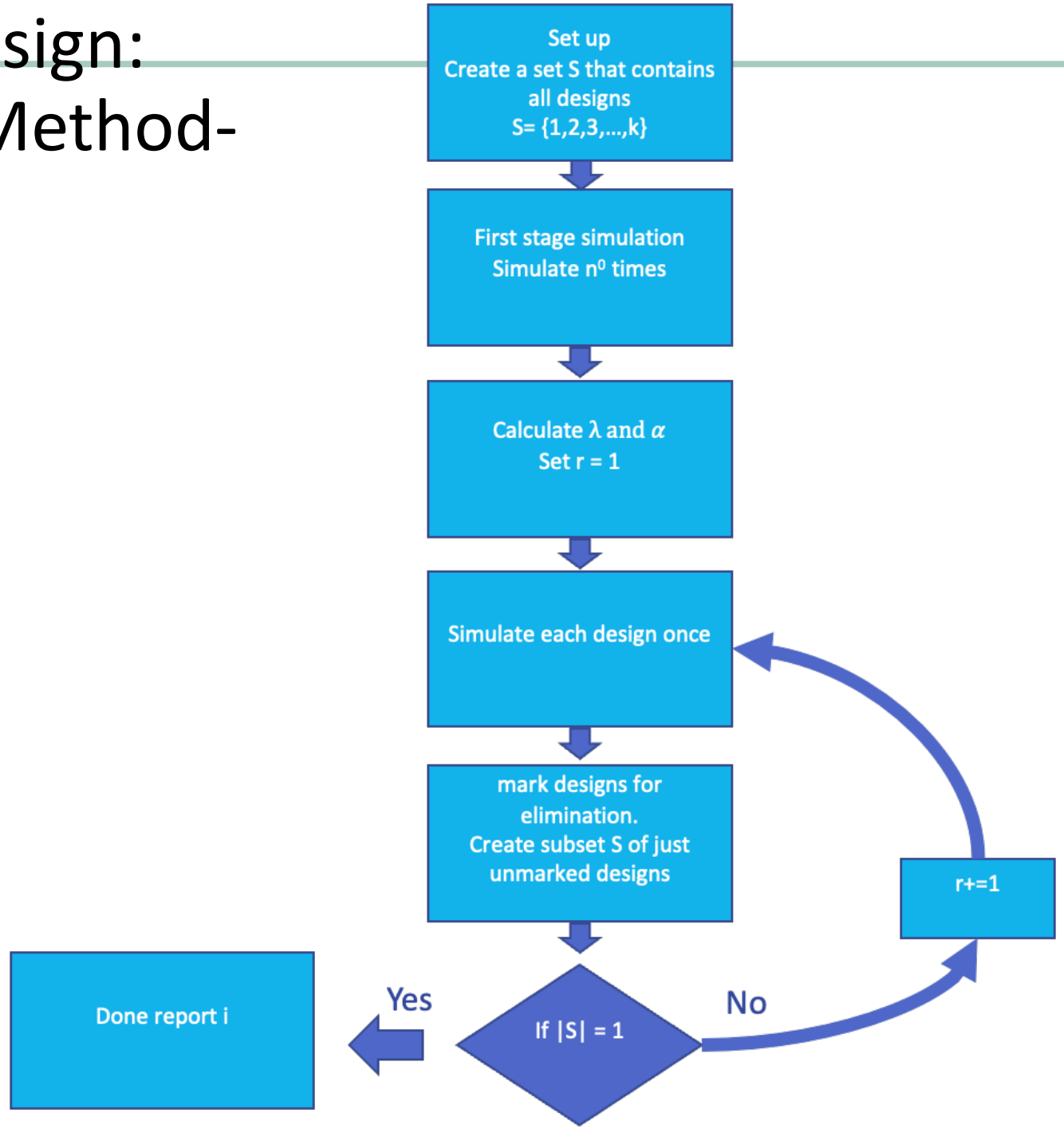
$$\min_{i \in S} \left\{ \sum_{j=1}^{r} Y_j(l) - Y_j(i) \right\} < \min\{0, -a - \lambda r\}$$

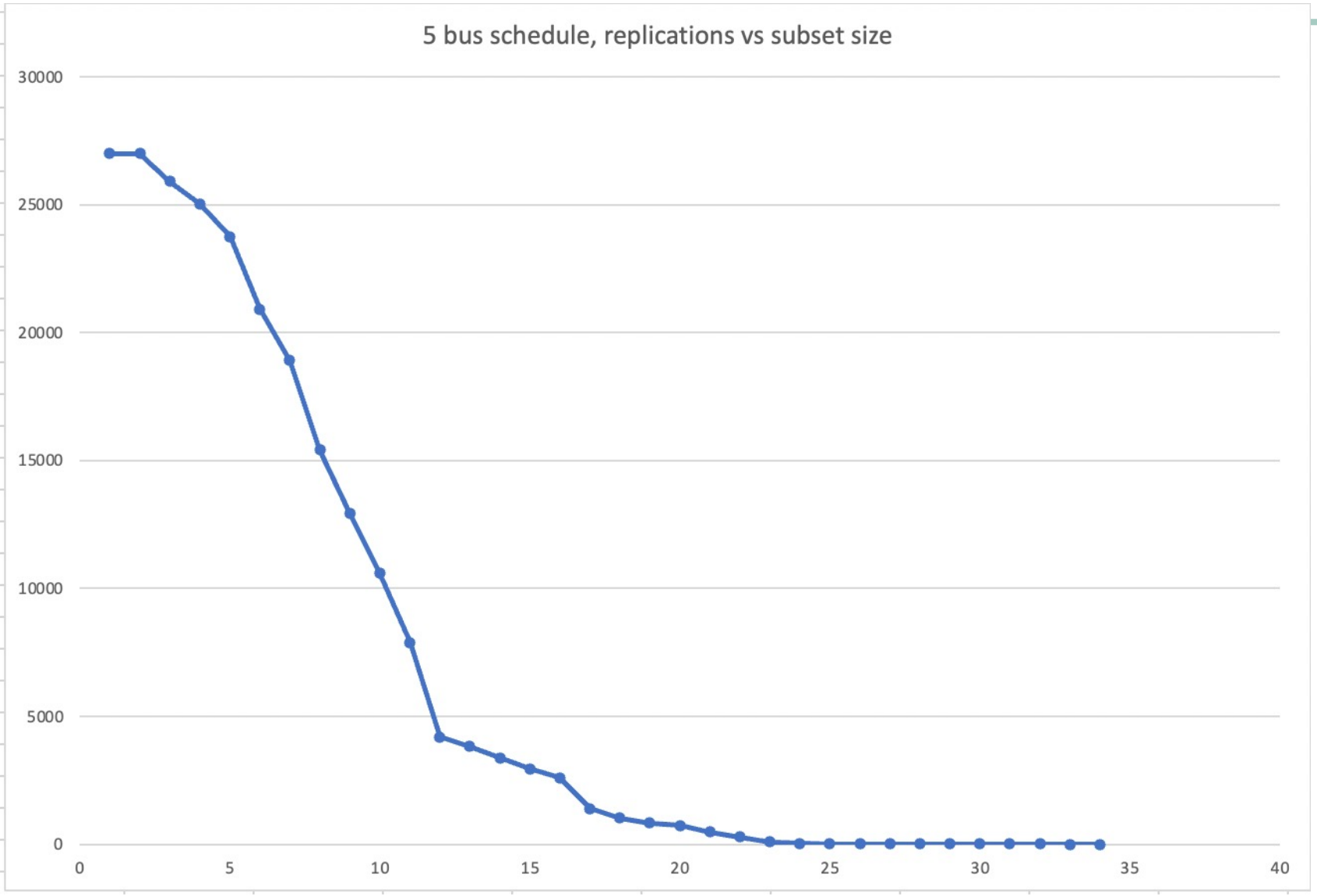


4. Remove all marked designs for elimination

5. If |S| = 1, finish. Else return to step 2

Picking the Best Design: Indifference Zone Method- Paulson Procedure





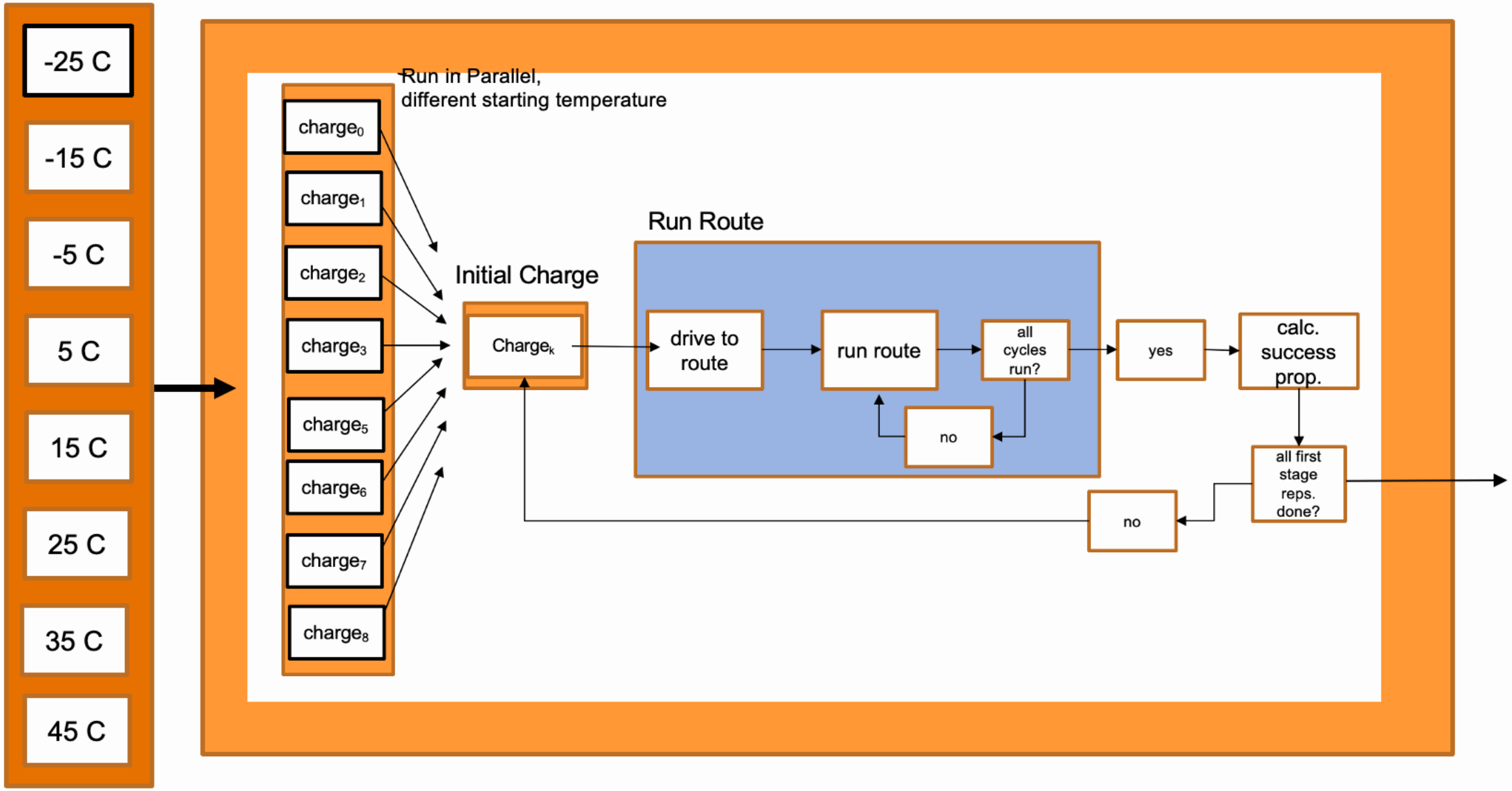
Best schedule found
[[(0, 1, 0, 2, 1),
(2, 0, 1, 1, 0),
(1, 1, 0, 0, 2)]]

Indifference zone: 1

78 mins. to converge

EXTRA SLIDES

Bus Pseudo Schedule Simulation



Simulation Behavior Example

Goal: Identify the lowest charge at which an electric bus can start a specific route for each tem

Simulation Task: Assuming the output from all the replications is normally distributed, determine the probability the proportion of time that the voltage is below 0.1

Metro Data Example:

- Temperature: -15 degrees °C
- Charge: The lowest charge such that *P(X<=.1) < 99.5%* is a charge of 28.75%

											AVERAGE	STDEV.	P(X<.1)	
0	0	0	0	0	0	0	0	0.010265	0.052181		0.006245	0.01646	1	31.25
0	0.064157	0	0	0	0	0	0	0	0		0.006416	0.020288	0.999998	30
0	0	0	0	0	0	0	0.005133	0.064157	0		0.006929	0.020173	0.999998	28.75
0.047904	0	0	0.015398	0	0	0	0.045338	0	0.142857		0.02515	0.045547	0.949847	27.5
0	0	0.15911	0	0.000855	0	0.09923	0	0.011976	0		0.027117	0.055731	0.904523	26.25
0	0.190761	0.188195	0.197605	0.045338	0.044482	0.005133	0.029085	0	0.126604		0.08272	0.083907	0.581581	25
0.167665	0.254919	0	0.174508	0.158255	0	0.098375	0.242943	0.134303	0		0.123097	0.096495	0.405414	23.75
0.088965	0.088109	0.186484	0.118905	0.017109	0.131737	0	0.185629	0.058169	0.013687		0.088879	0.06772	0.565219	22.5

Sample results from 10 replications of the Metro Route simulation.

Data Summary

West Campus Data	
Starting Temp	Min. Charge
-25	45
-15	45
-5	45
5	45
15	45
25	45
35	45

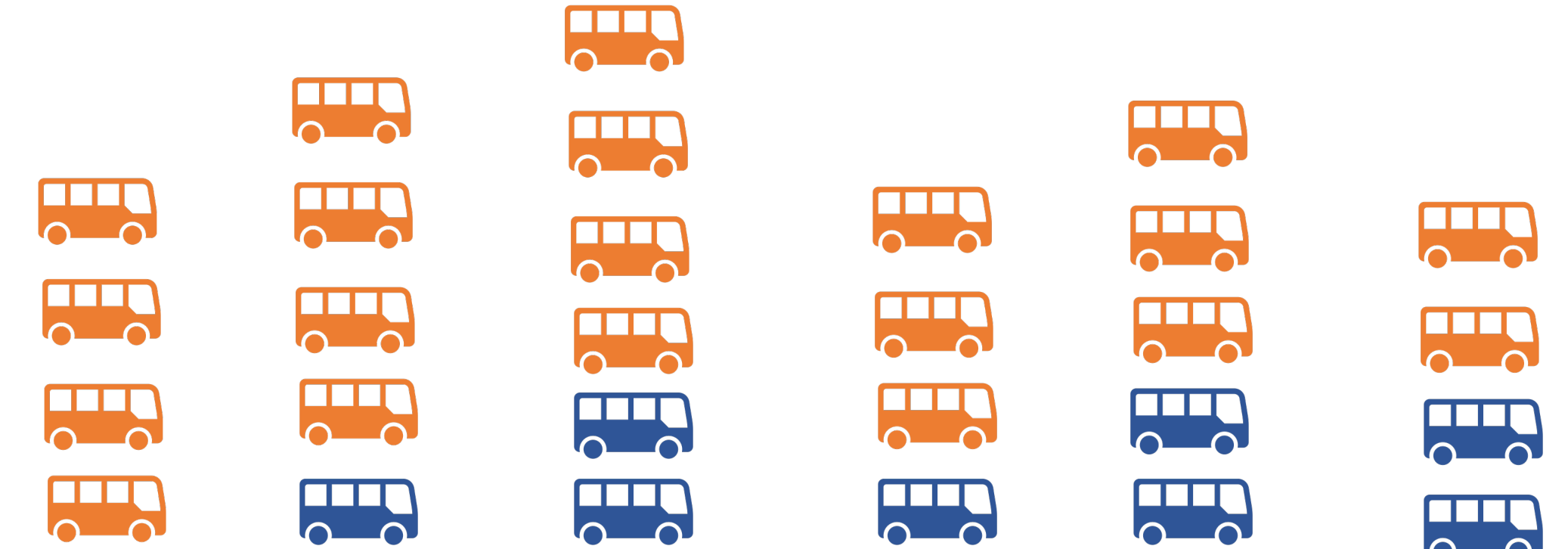
Metro Data	
Starting Temp	Min. Charge
-25	31.25
-15	28.75
-5	30
5	30
15	31.25
25	28.75
35	30

Temperature does not have a significant effect on the minimum charge in this model.
This could be due to multiple factors:

- Battery cooling is not being considered
- Effect of power from battery being deviated to power AC or heating is not being consid

Bus Fleet Simulation Designs

The following fleet designs were evaluated:



4 Electric Buses
0 Diesel Bus

4 Electric Buses
1 Diesel Bus

4 Electric Buses
2 Diesel Bus

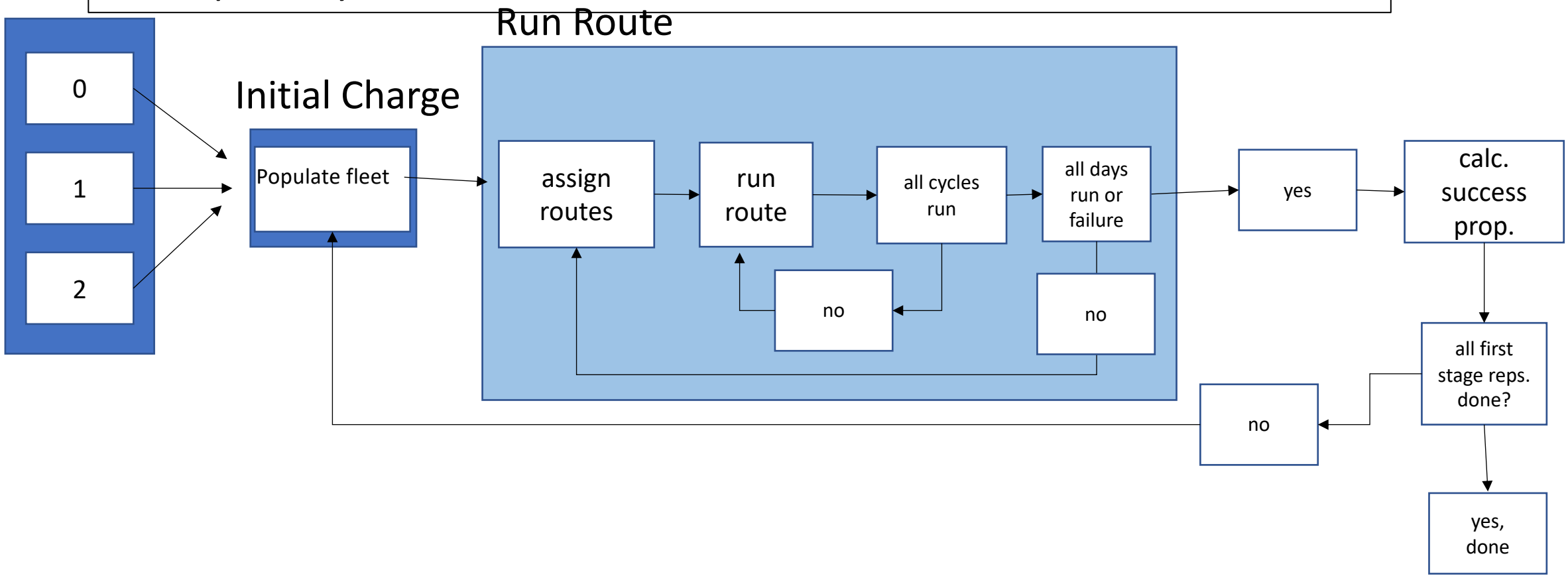
3 Electric Buses
1 Diesel Bus

3 Electric Buses
2 Diesel Bus

2 Electric Buses
2 Diesel Bus

Bus Fleet Simulation Designs

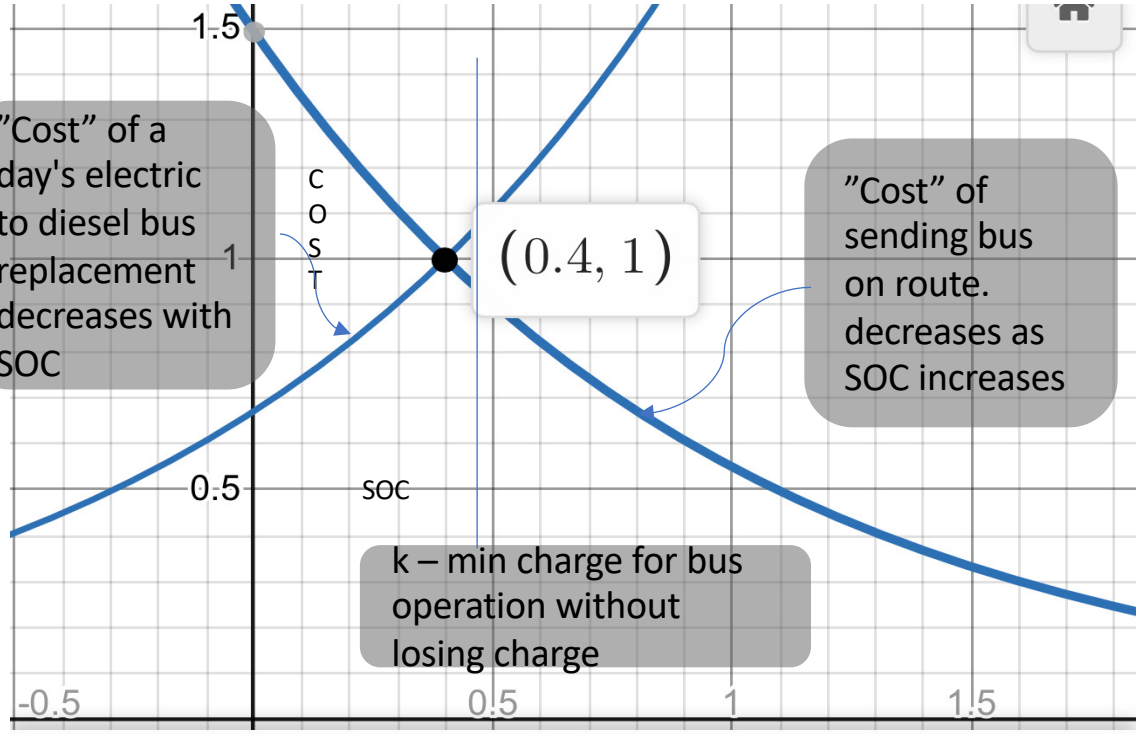
The simulation was run in parallel for different numbers of diesel buses, and separately for different numbers of electric buses.



Order Assignment Optimization

At the start of each day, busses are assigned according to the model's integer optimization problem.

- Assign buses according to current charge
- Single stage problem -> doesn't consider optimal choice based on potential future scenarios
- Future work suggestion – model bus as a multistage problem



$$min \sum_i \exp\left(\frac{charge}{100} - k\right) X_{i0} + \exp\left(-\frac{ch}{1}\right.$$

such that $+ \exp\left(\frac{charge}{100} - p\right) X_{i1} + \exp\left(-\frac{charge}{100} + \right.$

$X_{i0} + X_{i1} + X_{i2} + X_{i3} = 1 \qquad for\ i \in set\ of$

- all busses assigned

$$\sum_i X_{i0} \leq metro\_Busses$$

- assign at most metro_Busses to the route

$$\sum_i X_{i1} \leq westcampus\_Busses$$

- assign at most westcampus_busses to the ro

$$\sum_i X_{i2} + X_{i3} \leq diesel\_Busses$$

- assign at most diesel_busses to the route

$$X_{i2} + X_{i3} \leq 1 \qquad for\ i \in set\ of\ busses$$

Simulation Data

Four E-Bus System

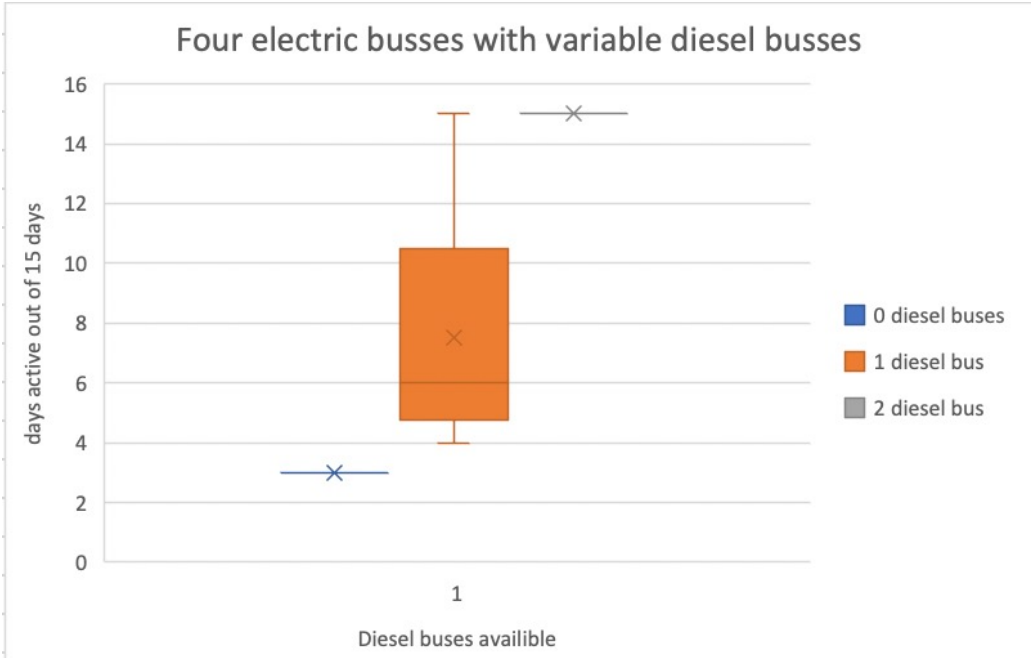
- Fleet only guaranteed to work with 2 backup diesel buses
- 1 backup diesel bus led to unpredictable results

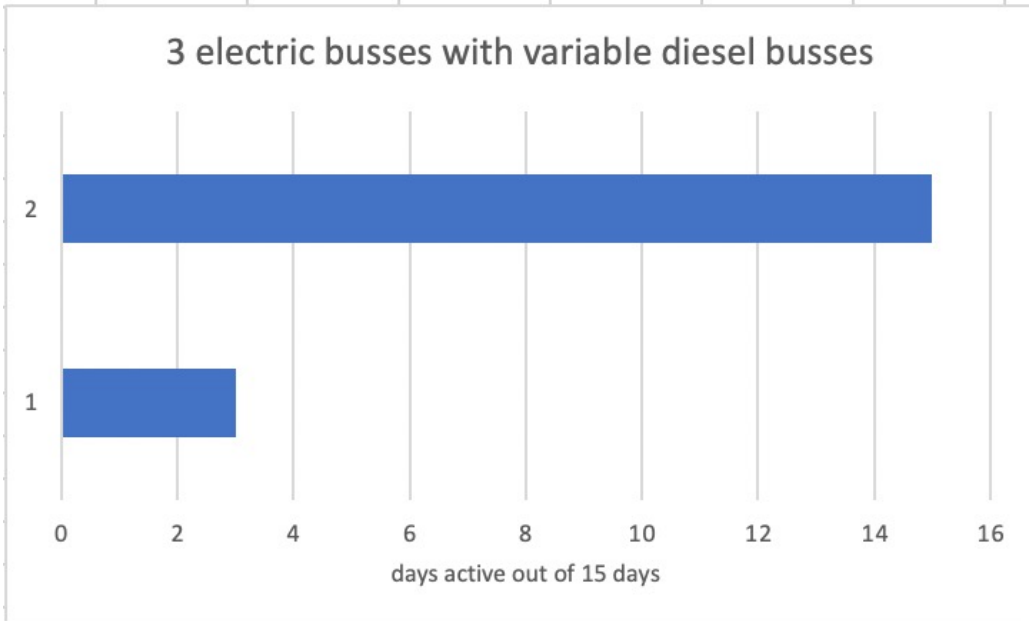
Three E-Bus System

- Fleet only guaranteed to work with 2 backup diesel buses
- 1 backup diesel bus led to guaranteed failure after 3 days

Two E-Bus System

- No tested fleet designs could stay active beyond 3 days





Simulation Behavior Example

Goal: Identify the lowest charge at which an electric bus can start a specific route for each tem

Simulation Task: Assuming the output from all the replications is normally distributed, determine the probability the proportion of time that the voltage is below 0.1

Metro Data Example:

- Temperature: -15 degrees °C
- Charge: The lowest charge such that *P(X<=.1) < 99.5%* is a charge of 28.75%

											AVERAGE	STDEV.	P(X<.1)	
0	0	0	0	0	0	0	0	0.010265	0.052181		0.006245	0.01646	1	31.25
0	0.064157	0	0	0	0	0	0	0	0		0.006416	0.020288	0.999998	30
0	0	0	0	0	0	0	0.005133	0.064157	0		0.006929	0.020173	0.999998	28.75
0.047904	0	0	0.015398	0	0	0	0.045338	0	0.142857		0.02515	0.045547	0.949847	27.5
0	0	0.15911	0	0.000855	0	0.09923	0	0.011976	0		0.027117	0.055731	0.904523	26.25
0	0.190761	0.188195	0.197605	0.045338	0.044482	0.005133	0.029085	0	0.126604		0.08272	0.083907	0.581581	25
0.167665	0.254919	0	0.174508	0.158255	0	0.098375	0.242943	0.134303	0		0.123097	0.096495	0.405414	23.75
0.088965	0.088109	0.186484	0.118905	0.017109	0.131737	0	0.185629	0.058169	0.013687		0.088879	0.06772	0.565219	22.5

Sample results from 10 replications of the Metro Route simulation.

Current Employee Resources

Clerks

Roles:

- In charge of processing order requests as they come in
- Validate orders
- Create pull forms for pickers
- Sort items after they come in
- check for errors
- bag items and pack them for shipping

Pickers

Roles:

- Operate MHEs
- go to each location on the pull form and collect the necessary items
- bring back-ordered items to be sorted and packed by the clerks

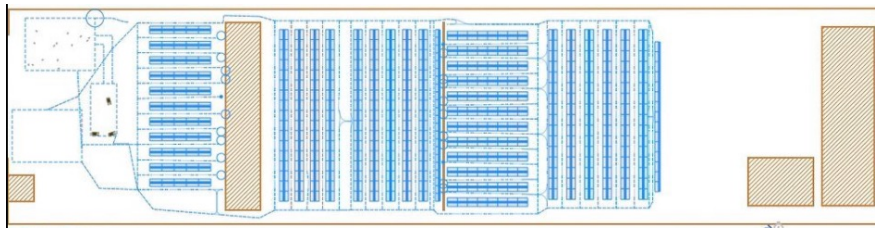
- Currently clerk's role acts as a limiting factor
- Proposed models reduce the amount of time and the queue at this step to increase the order through-put rate

Itinerary

We will build three models:

AS-IS model

- This allows us to validate the logic of model with actual data from the warehouse
- Additionally, it allows us to determine by how much the proposed changes improve the time and utilization rates

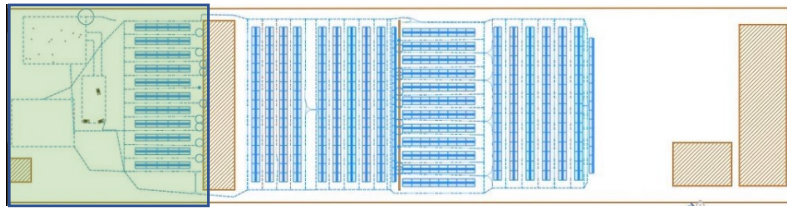


Proposed Layout model

- This model changes the physical layout of the warehouse so that all NSN lines are kept in

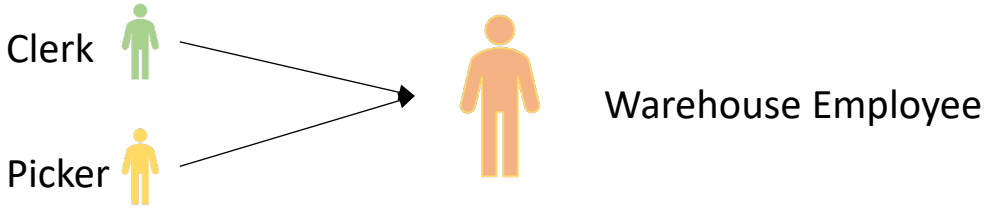
A bay, the first section closest to the shipping area

- Additionally, all NSNs are put in set locations according to their predicted demand value



Combined Resource model

- This model creates a new employee resource that combines the tasks of the clerks and the pickers so that pickers can help with clerk roles during surges or if a large amount of backlog occurs

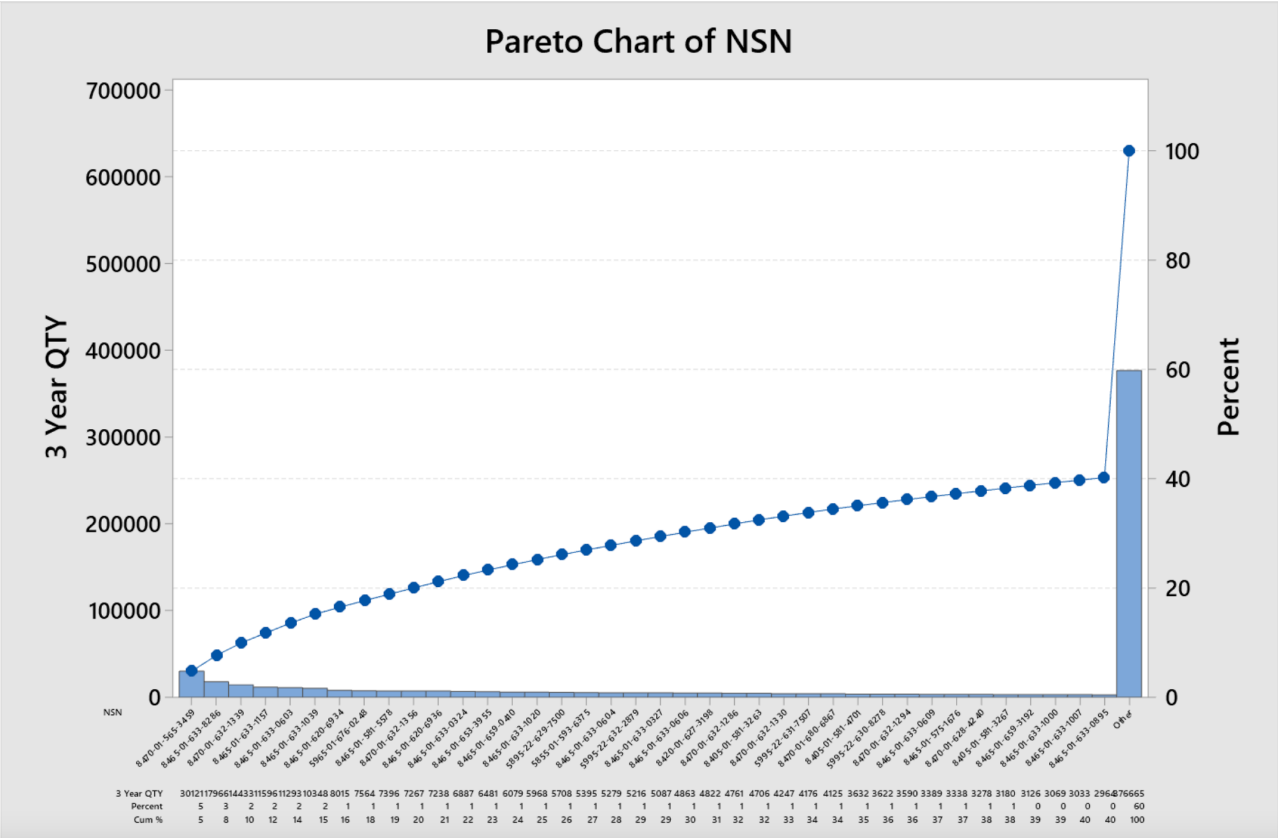


Creating Storage Allocation and Order Lists

- Noticed discrepancy in demand for individual NSNs
- **Majority of pulls from the warehouse came from the top 34 NSN lines**
- Suggests time can be saved by intentional storage allocation

Discrepancy due to multiple factors:

- Shelf life of individual NSNs, ex: ballistic plates and helmets
- Preference for bulk orders (order for multiple people at once) to save on time



Order Content Generation

To generate order forms that consider this difference in demand we will use inverse transformation sampling to generate order request contents.

This will require the following assumptions:

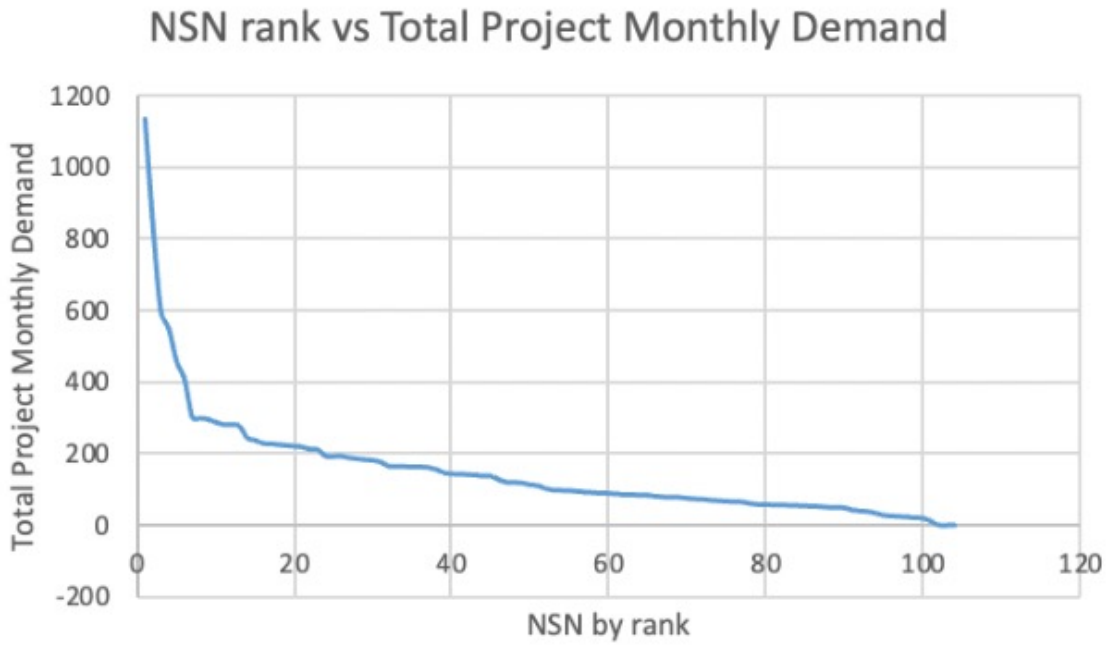
- Assume that the items being ordered are proportional to the demand

- Assume that all NSN demands, $x1$, $x2$, ..., $x4300$ can be assigned a rank number based on that demand

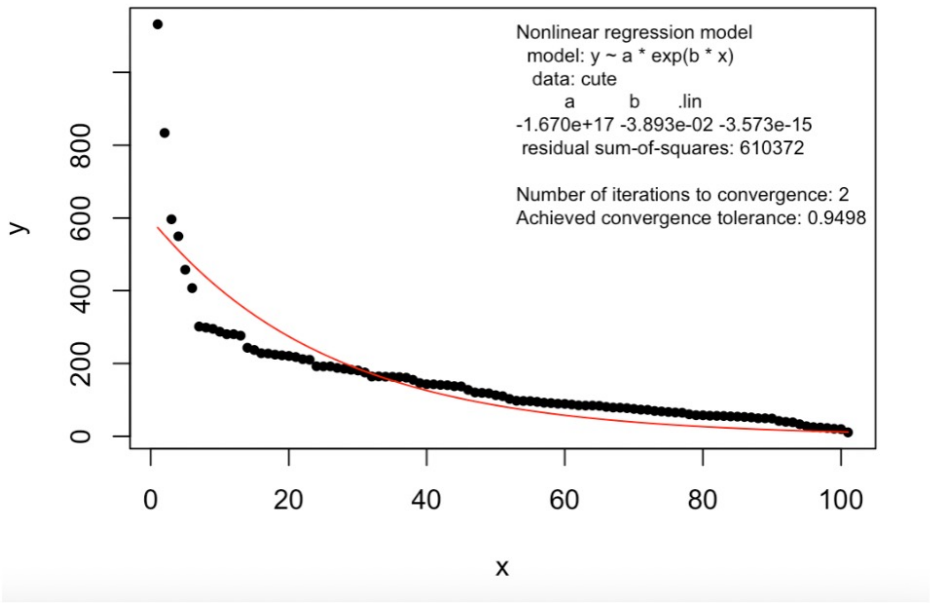
such that $f(xi) < f(xj) <$.....

- Assume that the demand of the remaining NSN lines not given follow the same parametric curve.

This will allow us to extrapolate the monthly demand of NSNs not given



Order Content Generation



Inverse Transformation Random Sampling Procedure

The following parametric curve was fit to the data

$$y = -1.67E17 * e(-.03893*x)$$

using the following PDF and CDF we can use inverse transformation sampling

$$f(x) = \begin{cases} 0 & x < 1 \\ \frac{1}{(4.127061E+18)} - 1.670E17\ e^{(-.03892x)} & 1 \le x \le 4300 \\ 0 & else \end{cases}$$

$$F(x) = \begin{cases} 0 & x < 1 \\ 2.42303E - 19(4.29085E18\ e^{(-.03892x)} & 1 \le x \le 4300 \\ 0 & else \end{cases}$$

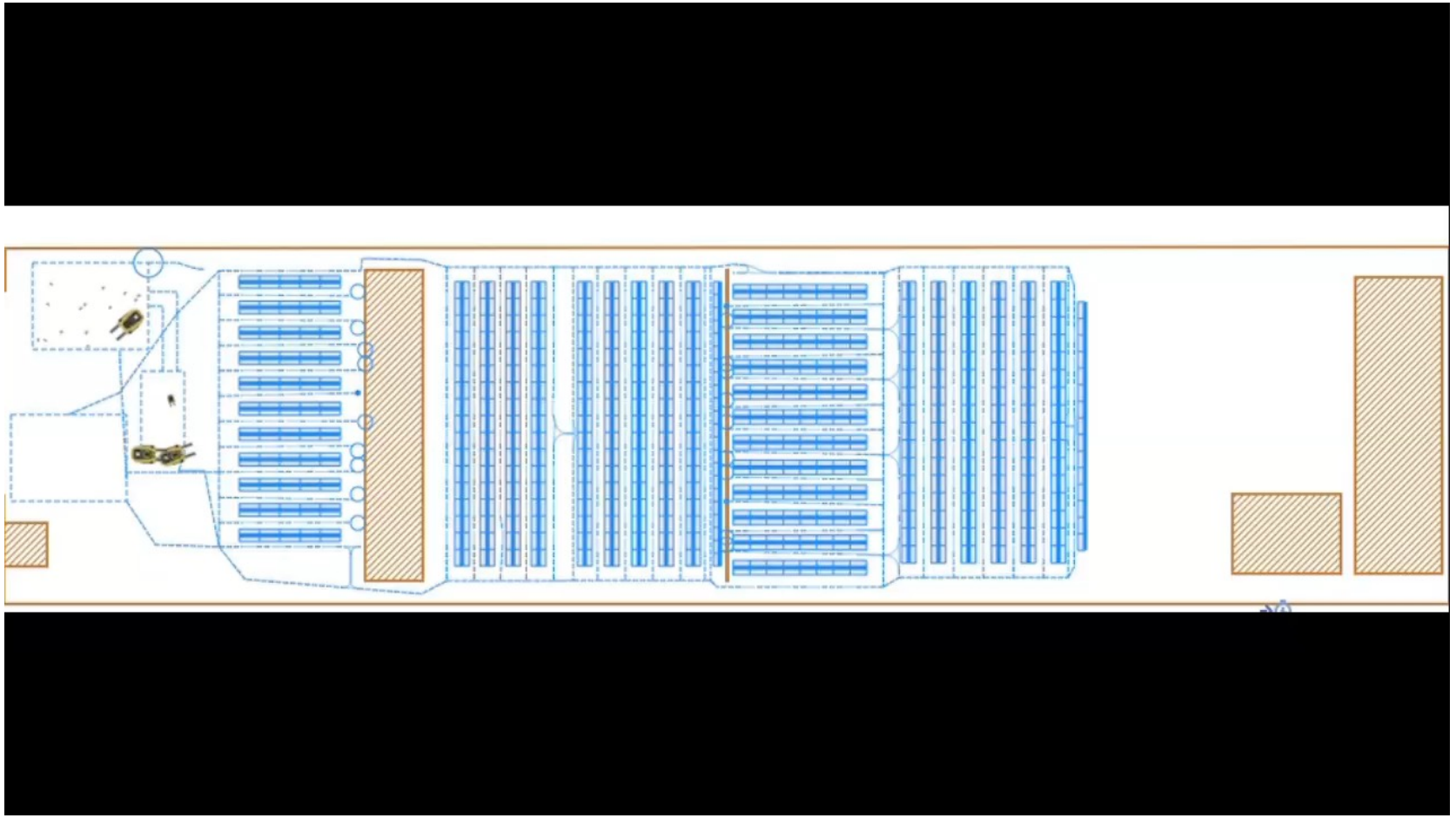
Now we can use inverse transformation sampling

- Generate y~$U(0,1)$
- plug y into $F^{-1}(y) = -25.69373\ ln(.96182\ y)$
- round down value to get the corresponding NSN realization

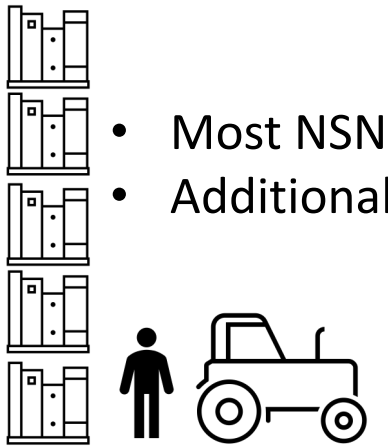
Using this procedure, we can generate order contents that represent the uneven demand for NSNs

AS-IS Model Video

- MHE's travel the entire warehouse to pick up items from their order list
- Pickers may have to travel to multiple Bays to fulfil a single order
- Popular items are allocated randomly to warehouse locations



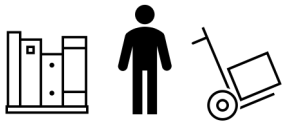
Current state



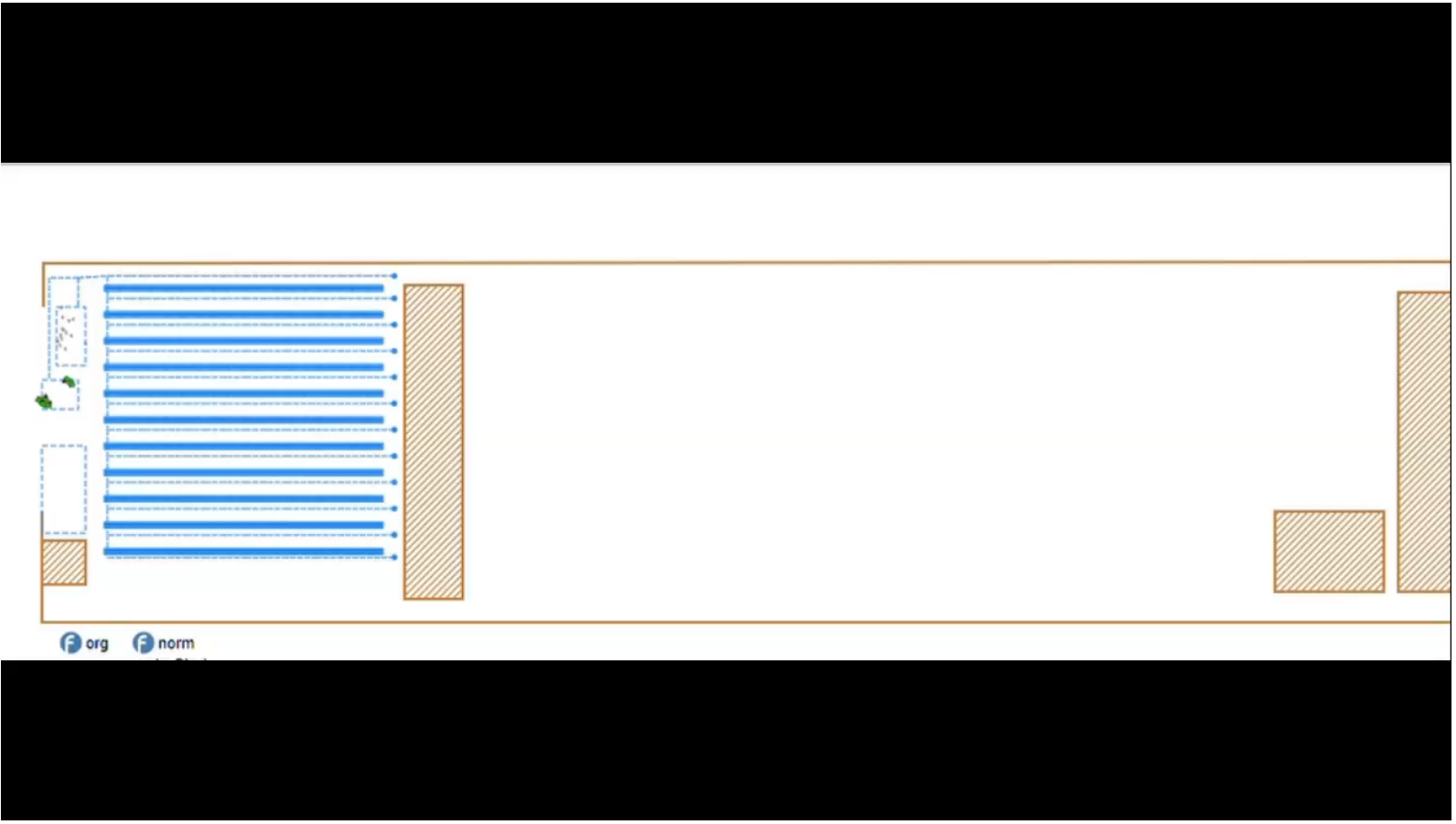
- Most NSNs unreachable without MHE
- Additionally, NSNs are randomly distributed

Proposed Layout

- All NSNs are reachable by a person with a cart, quicker pull
- NSNs are allocated according to predicted demand, there is a set location for each item reducing confusion and **NILs**

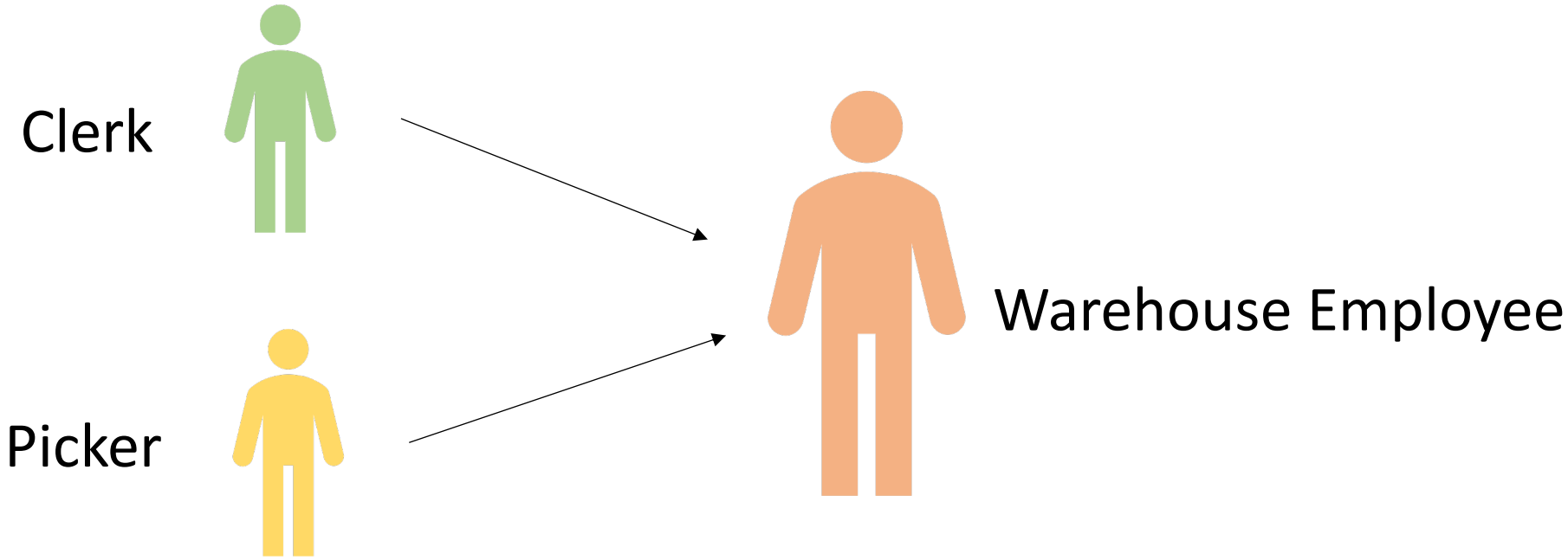


Proposed Optimized Simulation Video



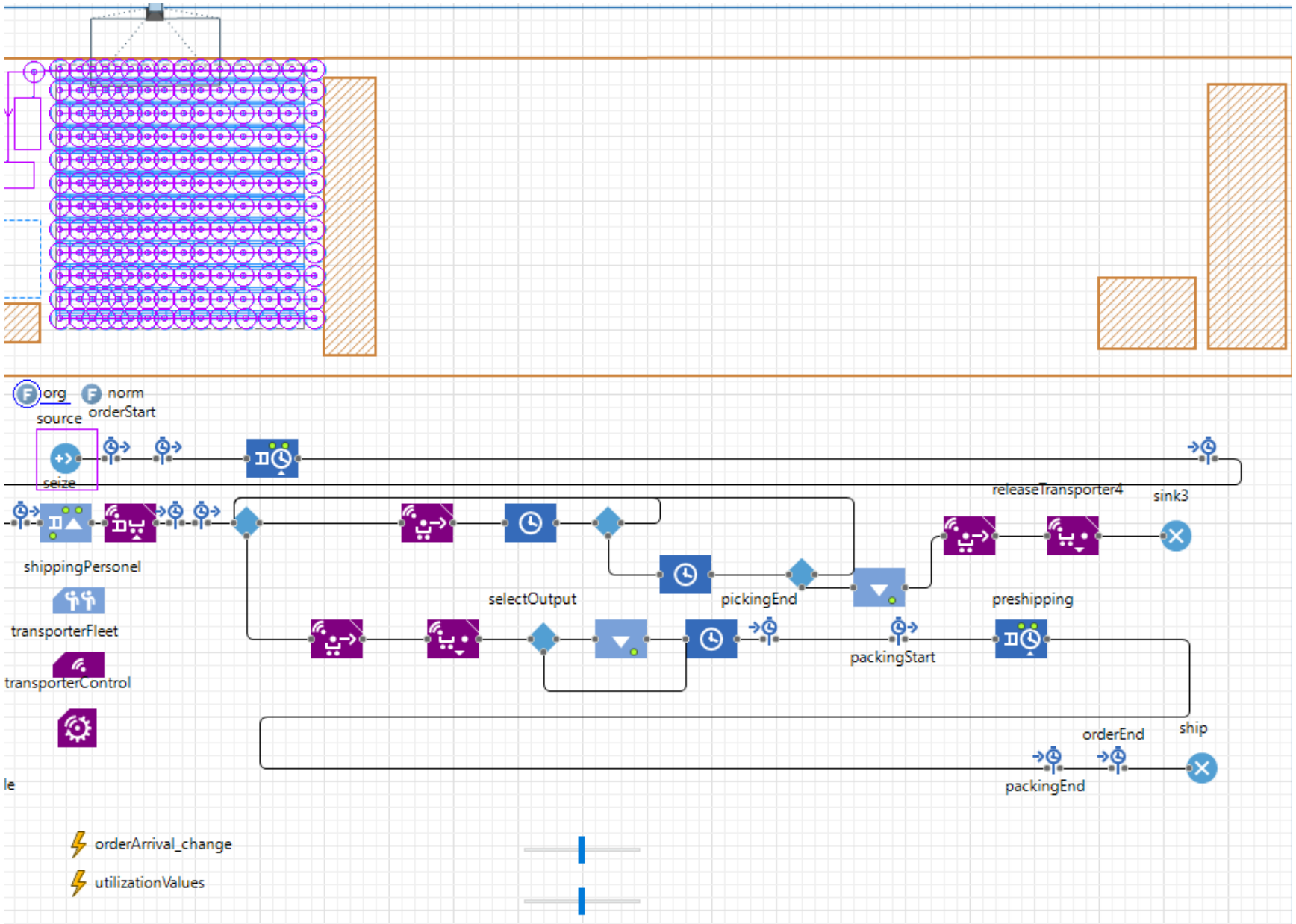
- Most in demand items are placed closest to the shipping area
- Pickers mostly concentrated in the left most section of the warehouse, the area closest to the shipping area
- Popular items are allocated set warehouse locations based on demand

GMU Combined Resource model



- Same warehouse layout as proposed before, but employees' roles are combined to increase efficiency and throughput

GMU Proposed Model

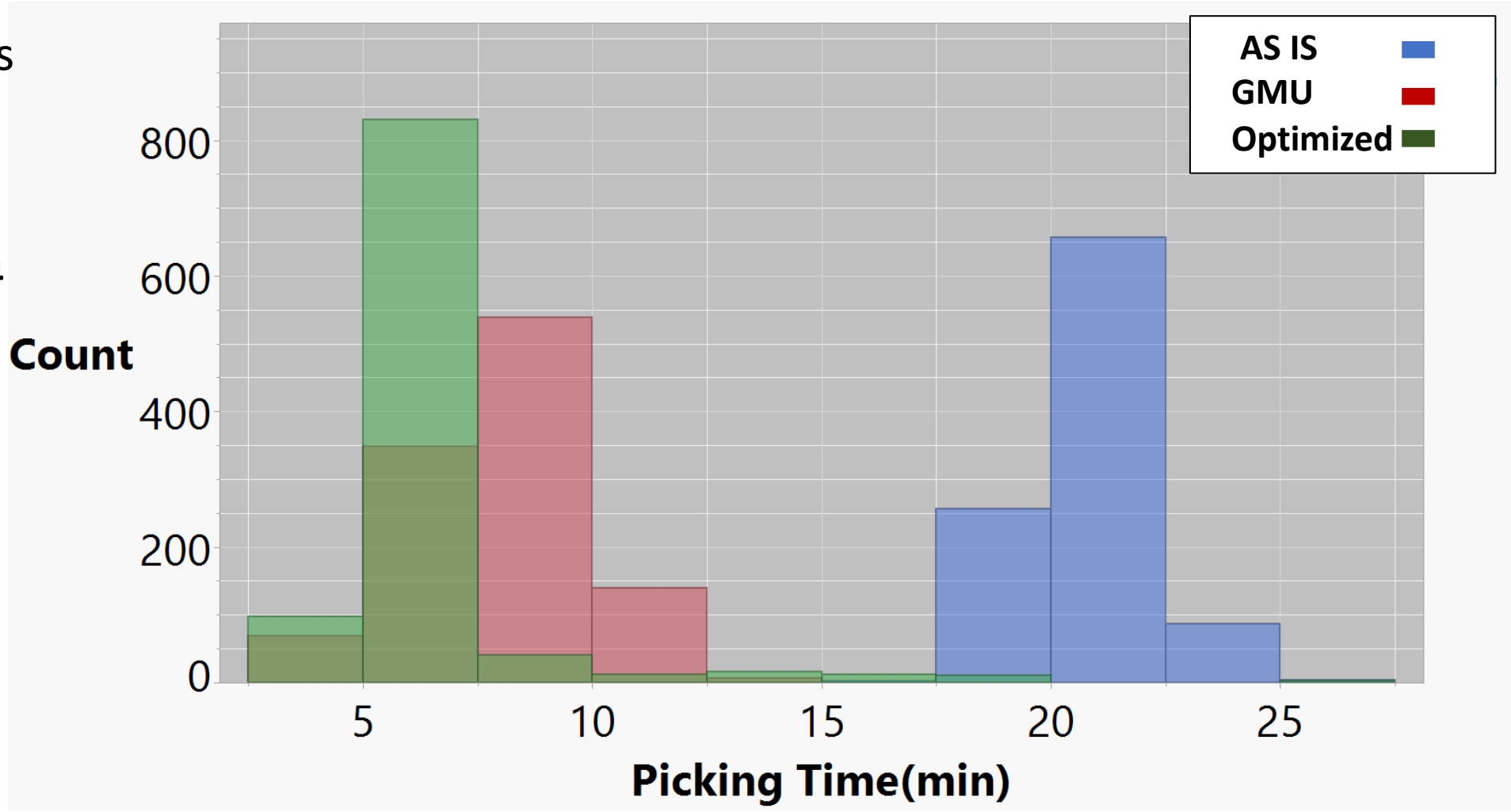


Picking Time Distributions

Average picking times in minutes:

- AS-IS model = **20.2**
- Optimized model = **6.4**
- GMU model = **8**

Decrease of 13.8 min. or 68% from AS -IS to the Optimized model

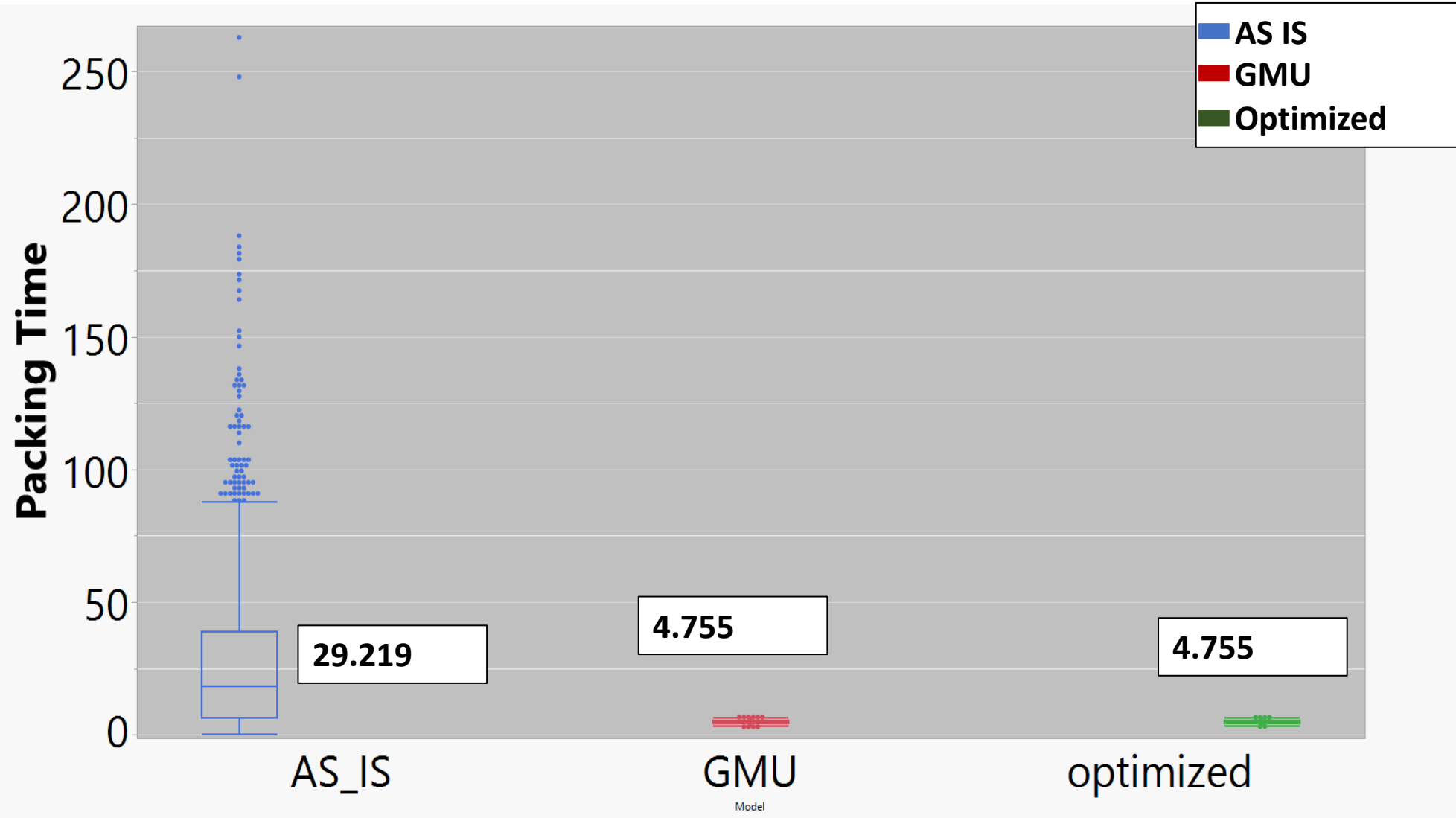


Packing Time

Average packing times in minutes:

- AS-IS model= **29.2**
- GMU, Optimized model = **4.8**

- **Decrease of 24.47 min. or *83.7%***

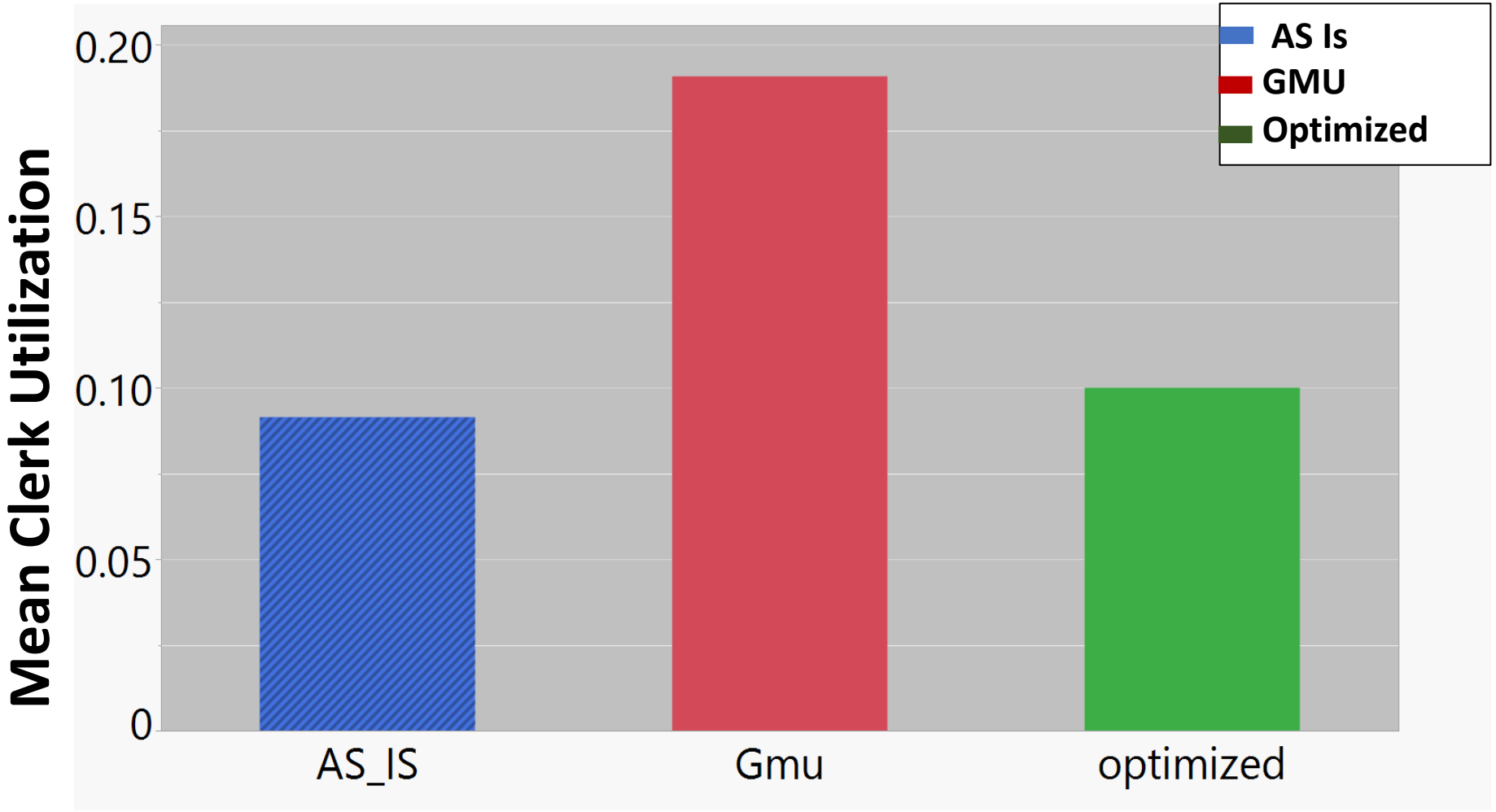


Mean Personnel Utilization

Average Utilization Rates:

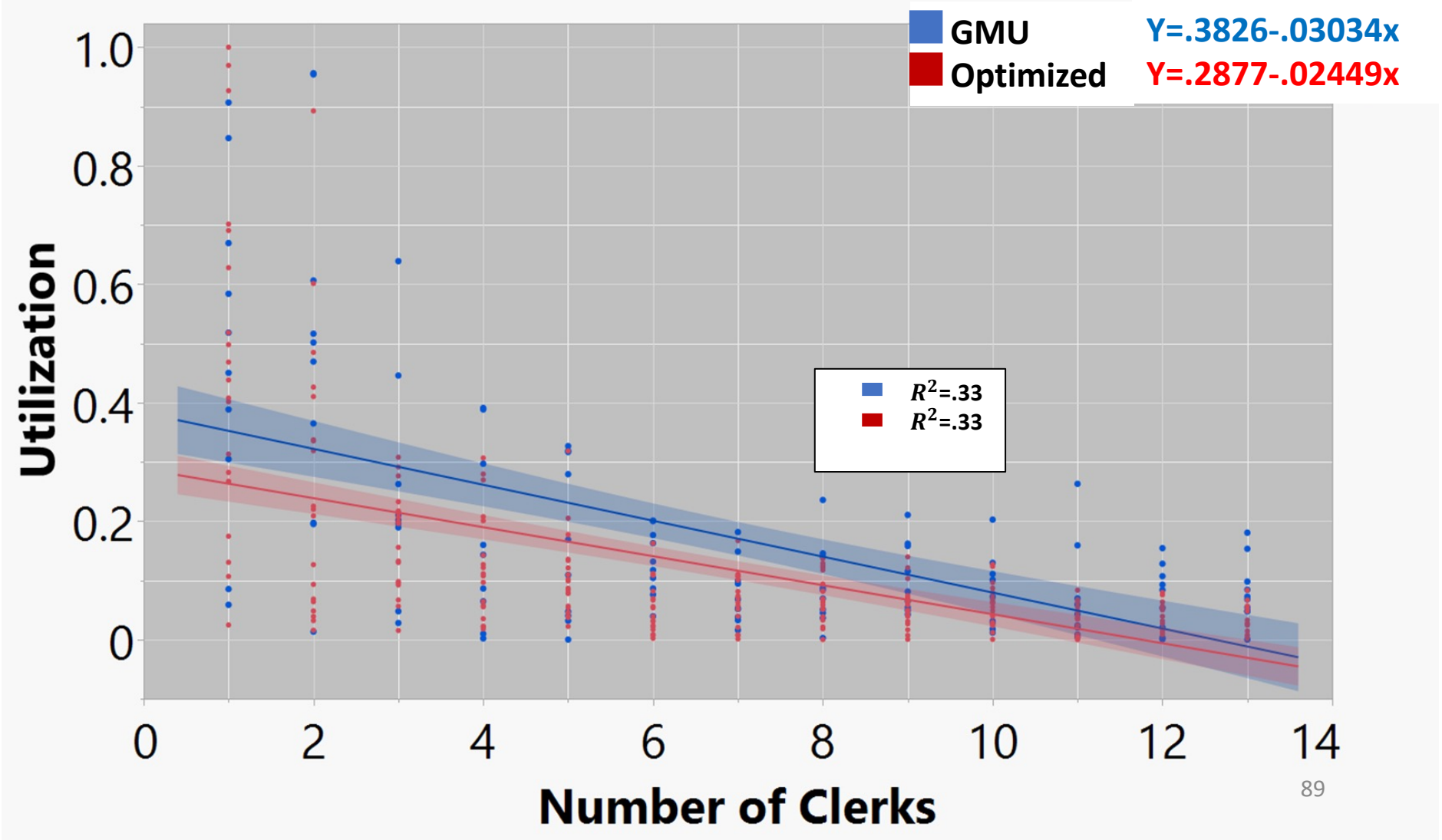
- AS-IS model= 0.09
- Optimized model = **0.1**
- GMU Model = 0.19

- **Increase of about .1**

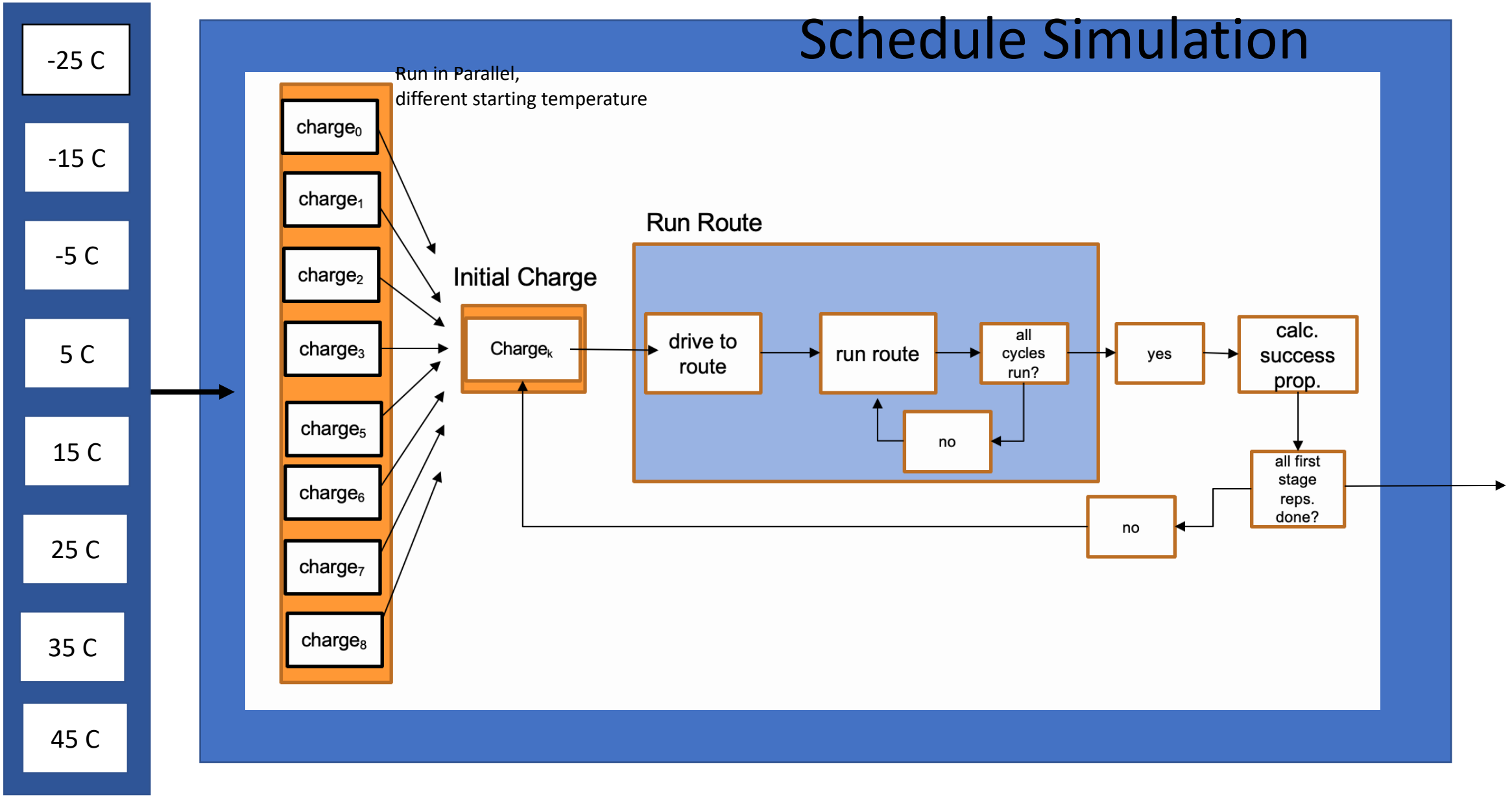


Sensitivity Analysis Number of Clerks vs Utilization

- Linear Regression model shows as the number of clerks increase the utilization rate decreases
- This effect increases for the GMU Proposed model



Bus Pseudo Schedule Simulation



CODE

```
wrkbk = openpyxl.load_workbook("battery_data.xlsx")
ws13 = wrkbk["Sheet17"]
ws8 = wrkbk["Sheet9"]
t0 = 0
R1 = [0.067057,0.02827586,0.008625677,0.007526372,0.006613802,0.005748813,0.003615515,0.003037664] #RC resistance
RC1 = [1.53,1.817227457,2.227546757,1.763904666,3.364588851,3.932676439,0.806026086,0.58073472] #RC resistance and capacit
R0 = [0.1203,0.058017975,0.029720972,0.016399955,0.01142771,0.008955544,0.007253926,0.00695822] #Resitancce
eta = [0.9933,0.968988303,0.98929249,0.995382636,0.995259434,0.994442137,0.993660319,0.992396564] #efficiency factor
gamma = [45.90491,61.54935912,62.37300504,1.00000002,1.000000019,1.000000019,1.000000019,1.000000019] #rate of decay
Q = [2.0559,2.057191539,2.062977475,2.046479553,2.051807947,2.049513705,2.049865862,2.043193788] #nominal capacity
M0 =[0.002613,0.002741896,0.001755915,0.004622616,0.001676118,0.000773851,0.000927482,0.002263506] #instantaneous hysteres
M = [0.194074,0.094699628,0.096311905,0.186225957,0.174526022,0.177104584,0.156279569,0.091961094] #hysteresis

i0 = 0 #current
ir0 =0 #current in RC sub circut
v0 = 3.599443197 #voltage
h0 = 0 #hysteriesis
z0 = 99.7 #inital SOC
s0 = 0 #indicator variable for instantaneous change in hysteresis voltage
sgni =0 #sign od current
```

Excel files read in

- Bus schedules and currents data
- SOC vs OCV data

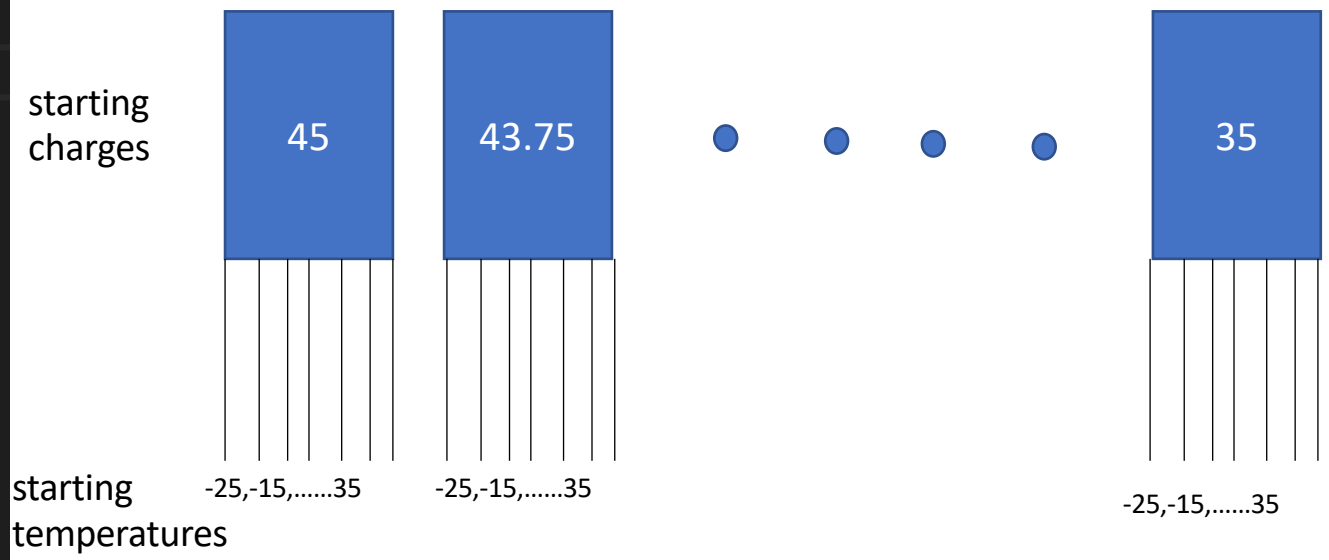
- estimated battery parameters stored as a list for each temperature
- initial values are initiated

```
def main(i):
    global cnt
    valList = []
    for j in range(firstStageReps):
        batt = Battery(cnt)
        rou = Route(batt,i)
        dog = rou.runningOn(batt)
        valList.append(dog)
    print("end")
    return valList
cnt = 0
#rat=main(3)
propList = []

for _ in range(7):
    if __name__ == '__main__':
        numbers = [1,2,3,4,5,6,7,8,9]
        with multiprocessing.Pool() as pool:
            results = pool.map(main,numbers)
        print(results)
        propList.append(results)
    cnt +=1
print(propList)
```

The code tests different starting charges at different temperature

this is done in parallel to save on time



Bus Cycle Loop –metro route

```
def runningOn(self,battery):
    timeList = [0,300,0,0,1800,0,0,0,0,3600]
    for i in range(16):
        cell = (ws13.cell(row=i+2,column = 1)).value
        deltaTime = cell + cell*random.triangular(-.2,.5,0)#change in time
        self.runRoute(i+2,battery,deltaTime)

    cell = (ws13.cell(row=18,column = 1)).value
    deltaTime = cell + cell*random.triangular(-.2,.5,0)#change in time
    self.runRoute(18,battery,deltaTime)
    for j in range(18):
        time = 0
        k =0
        for i in range(64):
            cell = (ws13.cell(row=i+19,column = 1)).value
            deltaTime = cell + cell*random.triangular(-.2,.5,0)#change in time
            if cell >= 60:
                self.runRoute(i+19,battery,deltaTime)
                time += deltaTime
            else:
                dt = deltaTime
                if timeList[k] > 0:
                    if timeList[k] - time >0:
                        dt = timeList[k] - time
                self.runRoute(i+19,battery,dt)
                time += dt
                k+=1
    voltLIST.append(self.volt)
    dog = ValueTest(self.volt)
    return dog
```

Drive bus to the start of the route:
delta time is read in from the excel sheet

- $deltatime_k = deltatime_k + deltatime_k * triangle(-0.2,0.5,0)$

update battery value

run route 18 time:
if bus arrives early to a stop, wait until it is time to go
update battery value

Update Battery Values

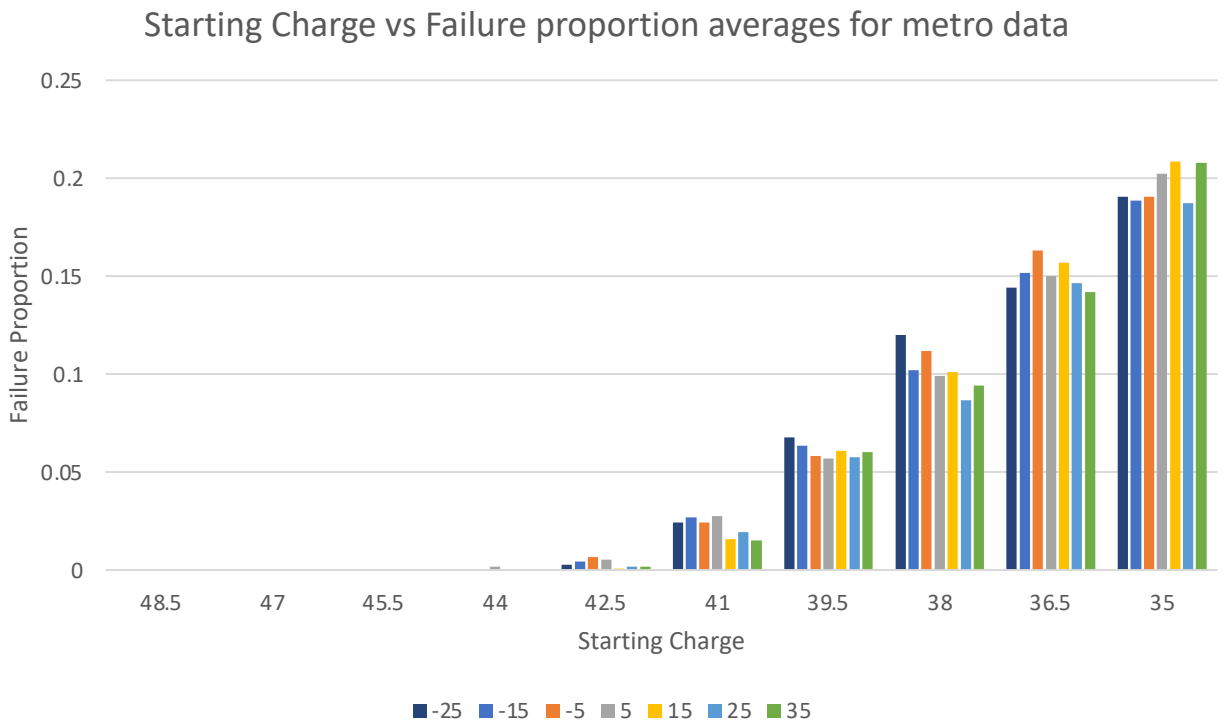
```python
class Battery:
    def __init__(self,temp):
        self.temp = temp
    def f(self):
        print("cat")
        return "hello world"
    def IR(self,deltaT): #current
        cat = math.exp(-deltaT/RC1[self.temp])
        dog = 1-cat
        return [cat,dog]
    def H(self,i0,deltaT): #hysteresis
        cat = eta[self.temp]*i0*gamma[self.temp]*(deltaT/3600)
        dog = abs(cat/(Q[self.temp]))
        bird = math.exp(-dog)
        return [bird]
    def Z(self,deltaT): #SOC
        cat = (-eta[self.temp]*(deltaT/3600))/(Q[self.temp])
        return [1,cat]
    def S(self,i0,s0): #sign
        if abs(i0) >0:
            return np.sign(i0)
        return s0
    def OCV(self,z0): #OCV
        for i in range(2,ws8.max_row+1):
            cell = (ws8.cell(row=i,column = 1)).value
            if z0 >= cell:
                cat = (ws8.cell(row=i,column = 2)).value
                return cat
        cat = (ws8.cell(row=ws8.max_row,column = 2)).value
        return cat
```

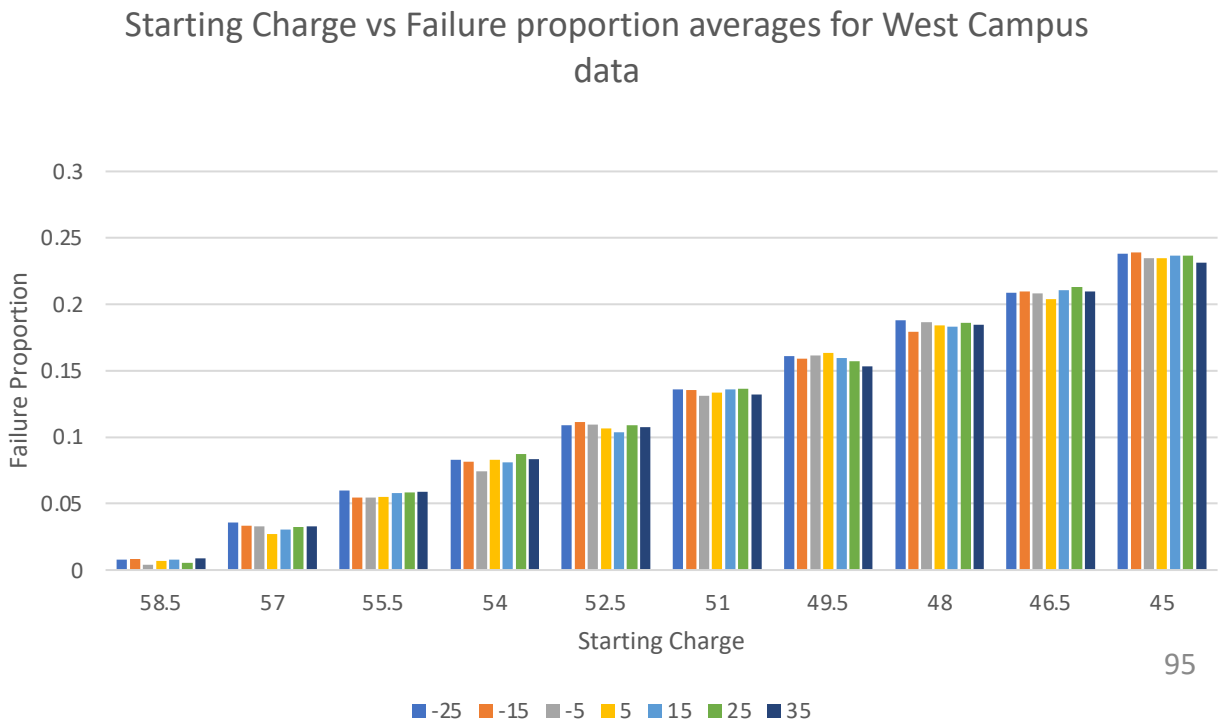
```python
def runRoute(self,i,battery,time):
    z0 = self.charge
    v0 = self.voltage
    deltaTime = time #change in time
    i0 = -(ws13.cell(row=i,column = 2)).value
    if i0 >0:
        red = random.triangular(-.1,.1,0)
        i0 = i0 + i0*red
    ir = battery.IR(deltaTime) #get values for new current for change in time
    h = battery.H(i0,deltaTime)  #get values for new current for change in time
    z = battery.Z(deltaTime) #get values for new SOC for change in time
    ocv = battery.OCV(z0/100) #get OCV for the prior SOC
    s0 = battery.S(i0,self.s0) #get sign of prior current
    self.s0 = s0
    v0 = ocv + M0[battery.temp]*self.s0 + M[battery.temp]*self.h0 - R1[battery.temp]*self.ir0 - R0[battery.temp]*i0 #calculate voltage for prior values
    self.voltage = v0
    print(v0)
    self.volt.append(v0)
    if v0 < 0 or v0 >5:
        print("cat")
    h0 = self.h0*h[0] + self.s0*(h[0]-1)
    self.h0 = h0
    ir0 = ir[0]*self.ir0 + ir[1]*i0
    self.ir0 = ir0
    #z0 = z0 + z[1]*i0
    z0 = (z0/100 + z[1]*i0)*100
    if z0 >= 100:
        z0 = 100
    self.charge = z0
    self.ch.append(z0)
```

```
def ValueTest(v):
    belowThreshold = 0
    cnt =0
    i = 0
    for i in range(len(v)):
        if v[i] <= 3:
            cnt += 1
    belowThreshold= cnt/len(v)
    return belowThreshold
```

The optimality of each design is evaluated by determining how many times the voltage drops below 3 Volts

- This code returns the proportion of times the battery is below 3 Volts
- This value can be used to compare designs





Output

The proportion of time under the floor voltage for different temperatures for the metro rout

-25 degrees												
starting charge	rep 1	rep 2	rep 3	rep 4	rep 5	rep 6	rep 7	rep 8	rep 9	rep 10		average
48.5	0	0	0	0	0	0	0	0	0	0		0
47	0	0	0	0	0	0	0	0	0	0		0
45.5	0	0	0	0	0	0	0	0	0	0		0
44	0	0	0	0	0	0	0	0	0	0		0
42.5	0	0	0	0	0.00672143	0	0	0	0	0.01941748		0.00261389
41	0.05377147	0.00522778	0.04929052	0.05675878	0.01344287	0	0.0029873	0.00448096	0.04929052	0.00896191		0.02442121
39.5	0.04705004	0.09260642	0.09634055	0.09932786	0.06945482	0.08289768	0.01643017	0.05302465	0.04480956	0.07617625		0.0678118
38	0.07617625	0.10231516	0.09335325	0.16131441	0.10828977	0.12546677	0.14712472	0.1120239	0.11127707	0.16355489		0.12008962
36.5	0.15758028	0.16056759	0.13965646	0.1351755	0.16131441	0.10978342	0.13592233	0.09559373	0.21433906	0.13293503		0.14428678
35	0.19417476	0.16056759	0.21359223	0.2300224	0.16504854	0.16206124	0.17849141	0.19342793	0.17401046	0.23525019		0.19066468
-15 degrees												
starting charge	rep 1	rep 2	rep 3	rep 4	rep 5	rep 6	rep 7	rep 8	rep 9	rep 10		average
48.5	0	0	0	0	0	0	0	0	0	0		0
47	0	0	0	0	0	0	0	0	0	0		0
45.5	0	0	0	0	0	0	0	0	0	0		0
44	0	0	0	0	0	0	0	0	0	0		0
42.5	0.01120239	0	0.017177	0	0.00522778	0.01045556	0	0	0	0		0.00440627
41	0.06721434	0.02464526	0.01344287	0.02763256	0.00672143	0.05601195	0.01643017	0.02763256	0.02464526	0.00448096		0.02688574
39.5	0.05003734	0.01418969	0.05675878	0.07020164	0.07020164	0.07991038	0.08812547	0.034354	0.08065721	0.0918596		0.06362957
38	0.08887229	0.12696042	0.08215086	0.09111277	0.16280807	0.10530246	0.07617625	0.13069455	0.06646751	0.09036594		0.10209111
36.5	0.16056759	0.17176998	0.13442868	0.12770724	0.17102315	0.10828977	0.16206124	0.16504854	0.15608663	0.15907394		0.15160568
35	0.18147872	0.18147872	0.22180732	0.18820015	0.21135176	0.18745332	0.19641524	0.16206124	0.17401046	0.18222554		0.18864824

*not shown is the data from -5 C, 5 C, 15 C, 25 C, and 35 C

goal; want to find the lowest charge we can start each bus for each route and each temperature

Assuming the output from all the replications is normally distributed,
we can determine the probability that the proportion of time that the voltage is below the thres
example from metro data for temp -25 degrees C, the lowest charge such that the P(X<=.1) is less
is a charge of 41%, thus we should not send out a bus below 41 for the metro route at temp -25

-25 degrees starting charge	rep 1	rep 2	rep 3	rep 4	rep 5	rep 6	rep 7	rep 8	rep 9	rep 10	Average	standard. Dev.	P(x<=.1)	Starting Charge
48.5	0	0	0	0	0	0	0	0	0	0	0	0	1	48.5
47	0	0	0	0	0	0	0	0	0	0	0	0	1	47
45.5	0	0	0	0	0	0	0	0	0	0	0	0	1	45.5
44	0	0	0	0	0	0	0	0	0	0	0	0	1	44
42.5	0	0	0	0	0.00672143	0	0	0	0	0.01941748	0.00261389	0.006270671	1	42.5
41	0.05377147	0.00522778	0.04929052	0.05675878	0.01344287	0	0.0029873	0.00448096	0.04929052	0.00896191	0.02442121	0.024325213	0.99905506	41
39.5	0.04705004	0.09260642	0.09634055	0.09932786	0.06945482	0.08289768	0.01643017	0.05302465	0.04480956	0.07617625	0.0678118	0.026958936	0.88375544	39.5
38	0.07617625	0.10231516	0.09335325	0.16131441	0.10828977	0.12546677	0.14712472	0.1120239	0.11127707	0.16355489	0.12008962	0.029039487	0.24453036	38
36.5	0.15758028	0.16056759	0.13965646	0.1351755	0.16131441	0.10978342	0.13592233	0.09559373	0.21433906	0.13293503	0.14428678	0.032495847	0.08646579	36.5
35	0.19417476	0.16056759	0.21359223	0.2300224	0.16504854	0.16206124	0.17849141	0.19342793	0.17401046	0.23525019	0.19066468	0.027665174	0.00052419	35