Queuing Theory

Nuno Antunes Ribeiro

Assistant Professor

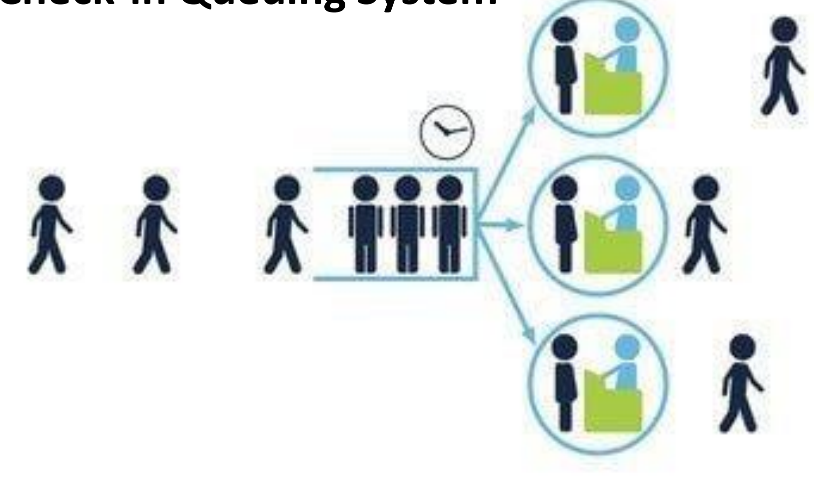
SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Engineering Systems and Design

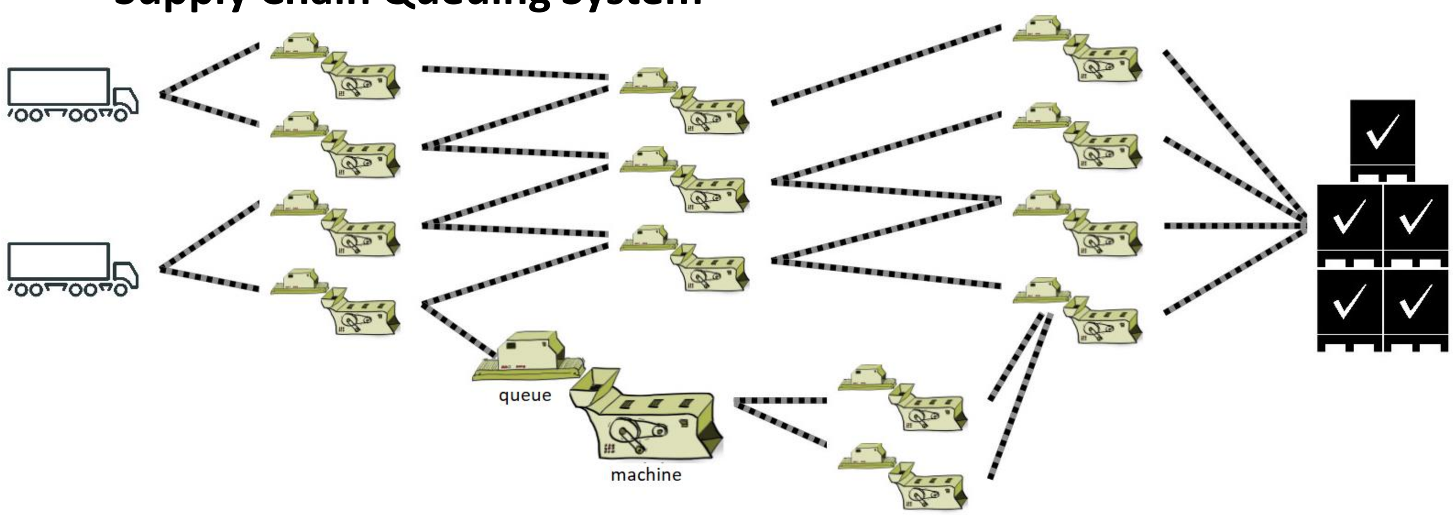
Queuing Systems

- A system having a **service facility** at which units of some kind arrive for service; whenever there are more units in the system than the service facility can handle simultaneously, a **queue** (or waiting line) develops.
- In simple terms, a queuing system consists of a **demand source**, a **queue** and a service facility with one or more identical parallel **servers**
- A queuing network is a set of interconnected queuing systems

Check-in Queuing System

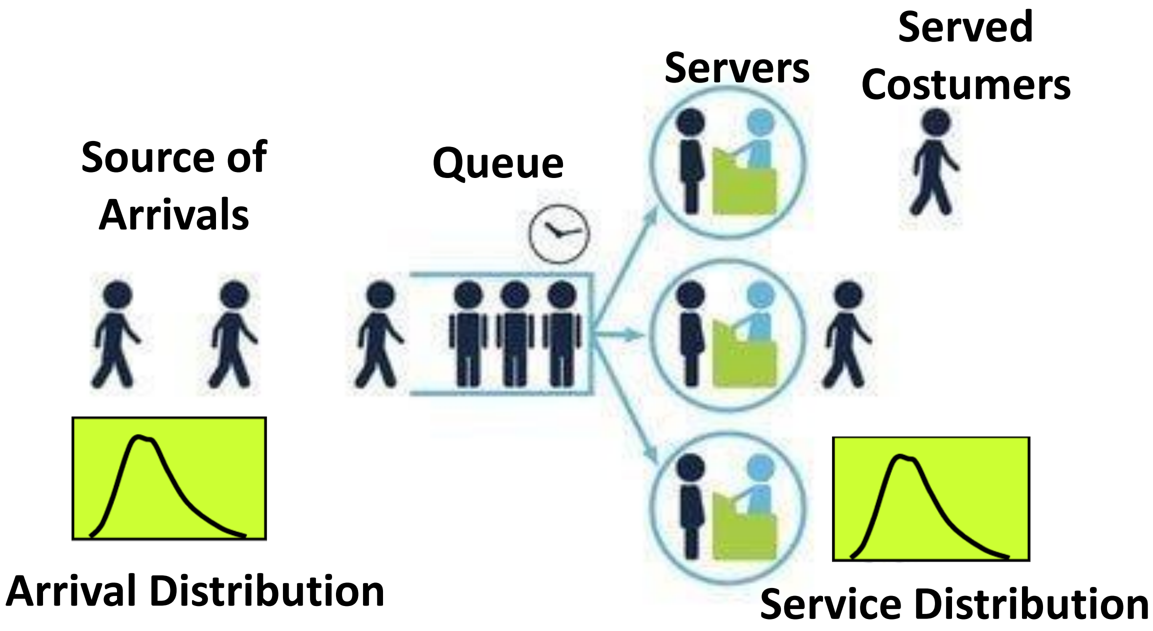


Supply Chain Queuing System



Queuing Theory

- Queuing Theory is concerned with the behavior of waiting lines (delays/congestion)
- Fundamental parameters of a queuing system:
  - Demand Rate
  - Service Rate
  - Probability distribution of demand inter-arrival times
  - Probability distribution of service times
  - Queue discipline (FCFS, SIRO, priorities, etc).



Kendal Notation

What is a M/M/1 queueing system?

A / S / c / K / P / QD

A: inter-arrival time distribution
- M: Exponential (M stands for memoryless/Markovian)
- D: Deterministic
- E_k: kth-order Erlang distribution
- G: General distribution

S: service time distribution

c: number of servers

K: total system size (∞) maximum number of customers allowed in the system

P: population size (∞) size of the population from which the customers come from

QD: Queue discipline (FIFO)

Little's Law

$\boldsymbol{L} = \textit{expected number of users in queueing system (those in queue plus those receiving service)}$

$\boldsymbol{L_q} = \textit{expected number of users in queue}$

$\boldsymbol{W} = \textit{expected time in queing system per user (waiting time plus service time)}$

$\boldsymbol{W_q} = \textit{expected time in queue per user}$

$$W = W_q + 1/\mu \qquad L = L_q + \lambda/\mu$$

$$L_q = \lambda W_q \qquad L = \lambda W$$

- **Obtain one of the performance measures, the other three can be computed**

Important Result from Queueing Theory

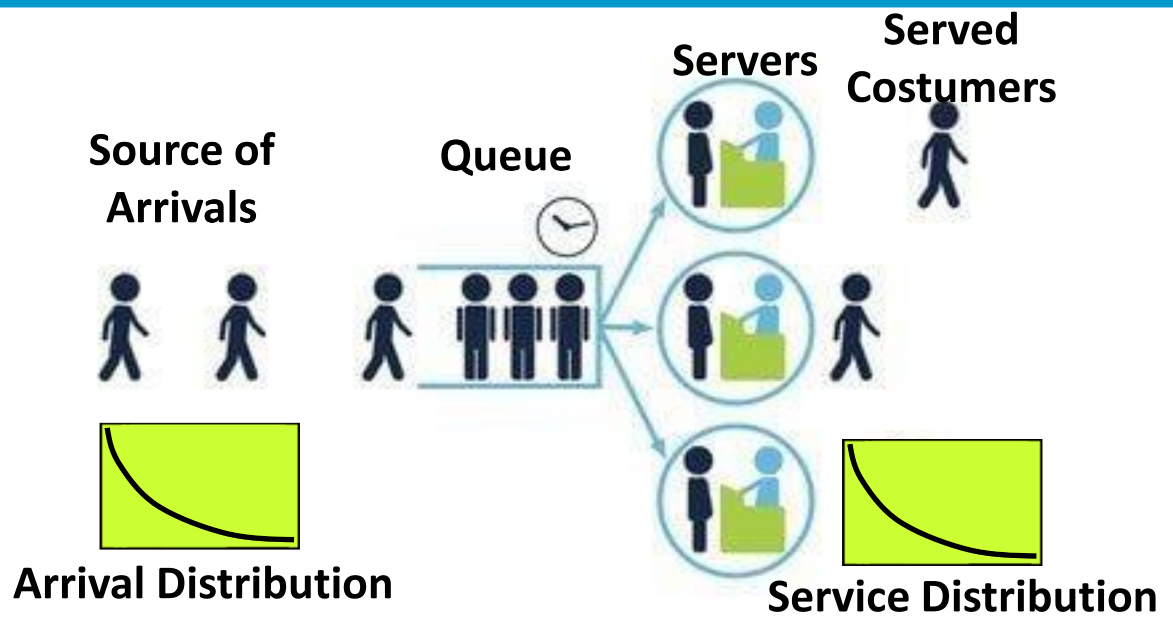
- MMS Queueing System

$$I = \frac{\lambda}{\mu}$$ **Intensity**

$$\rho = \frac{I}{s}$$ **Utilization Ratio**

$$P_0 = \frac{1}{\sum_{n=0}^{s-1} \frac{I^n}{n!} + \frac{I^s}{S!\,(1-\rho)}}$$ **Probability that there are 0 customers in the system**

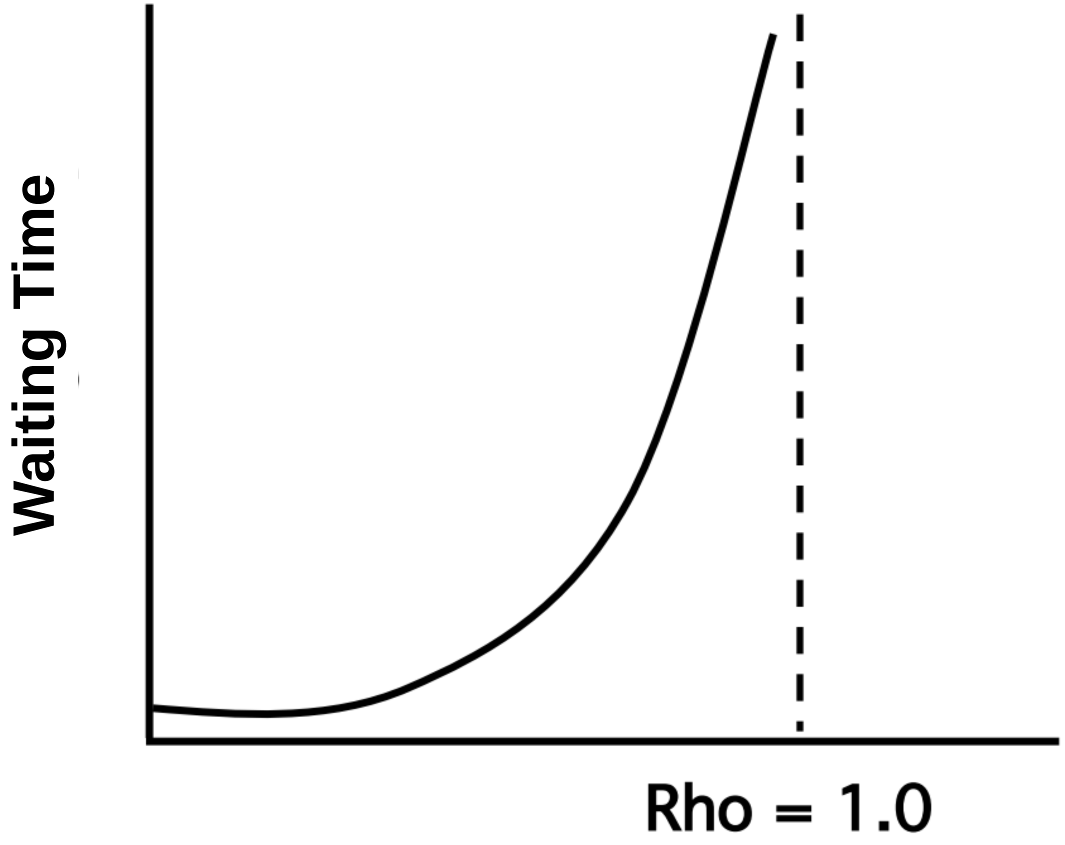
$$L_q = \frac{P_0 I^s \rho}{S!\,(1-\rho)^2}$$ **Mean number of customers in the queue**



Steady-state Conditions

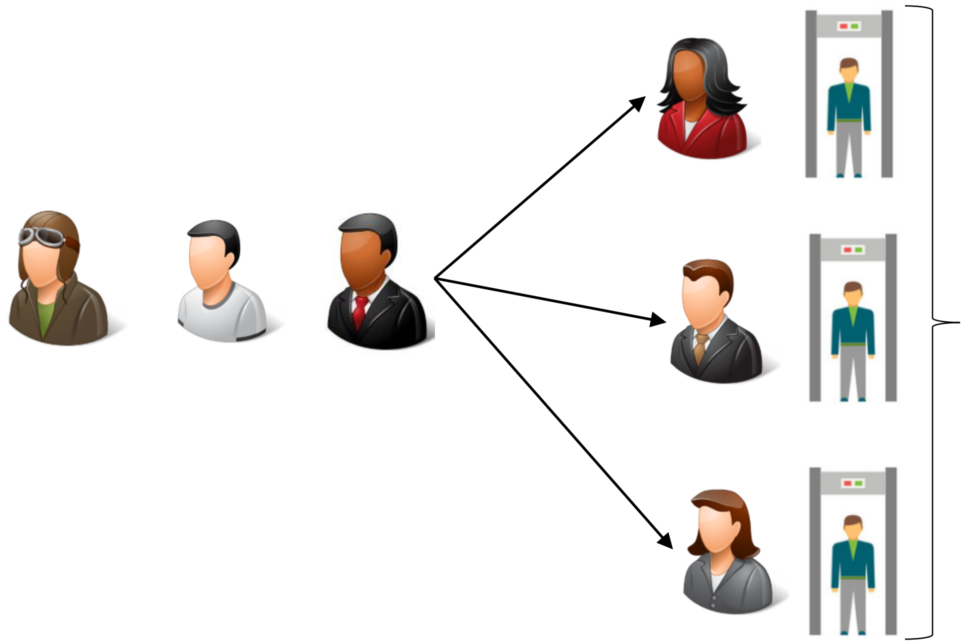
Steady-state Conditions

- Rho = ratio of demand rate vs service rate
- As loads on system increase, average waiting time increases exponentially
- Practical capacity = less than throughput capacity due to excessive delays
- Note that graph is for **steady state conditions**



Example – Vaccination Stalls

- We aim to compute the minimum number of stalls required in a vaccination centre*.
- The service rate per stall is about 30 services per hour
- **Minimum number of stalls to open?**

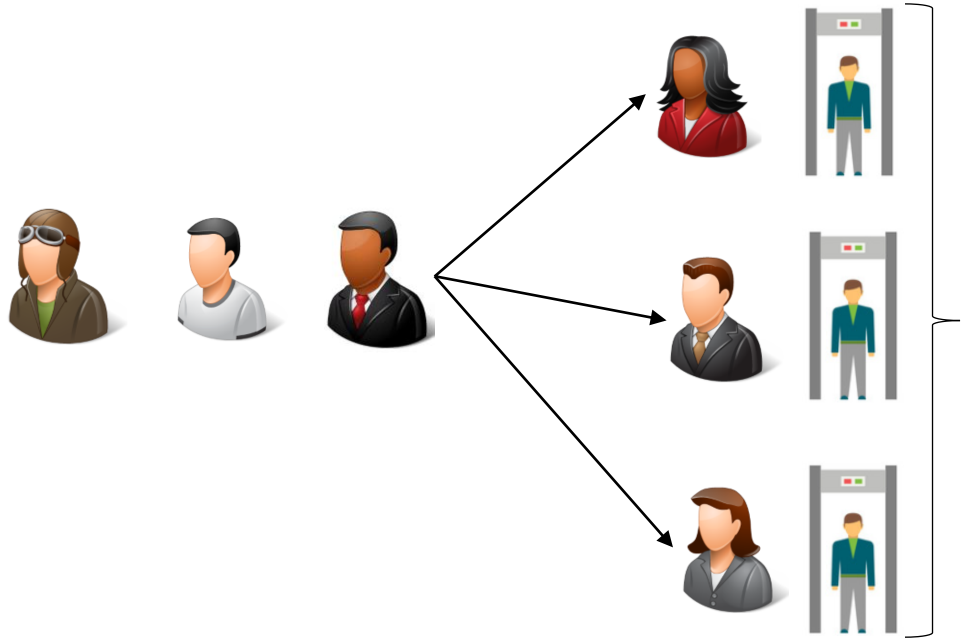


Time	Demand
04:00	0
05:00	0
06:00	40
07:00	320
08:00	1120
09:00	2280
10:00	2480
11:00	2480
12:00	2160
13:00	1880
14:00	2240
15:00	2440
16:00	2760
17:00	3200
18:00	2600
19:00	1680
20:00	960
21:00	320
22:00	40
23:00	0

*I used exactly the same example for airport security checkpoints in 40.321 Airport Systems Modeling and Simulation

Example – Vaccination Stalls

- We aim to compute the minimum number of stalls required in a vaccination centre*.
- The service rate per stall is about 30 services per hou
- **Minimum number of stalls to open?**



Under Steady State Conditions

Easy Answer

$$\frac{\textit{Demand Rate}}{\textit{Service Rate}}$$

$$\frac{3200}{30} = 106.66 \textit{ servers}$$

Time	Demand
04:00	0
05:00	0
06:00	40
07:00	320
08:00	1120
09:00	2280
10:00	2480
11:00	2480
12:00	2160
13:00	1880
14:00	2240
15:00	2440
16:00	2760
17:00	3200
18:00	2600
19:00	1680
20:00	960
21:00	320
22:00	40
23:00	0

*I used exactly the same example for airport security checkpoints in 40.321 Airport Systems Modeling and Simulation

Example – Steady State Results

Time	Dem.	Min Check-in	Q Model
			Expected Time in System (min)
04:00	0	0	0
05:00	0	0	0
06:00	40	2	3.6
07:00	320	11	7.32
08:00	1120	38	4.63
09:00	2280	77	3.74
10:00	2480	83	7.74
11:00	2480	83	7.74
12:00	2160	73	3.73
13:00	1880	63	7.7
14:00	2240	75	7.72
15:00	2440	82	4.74
16:00	2760	93	3.76
17:00	3200	107	7.77
18:00	2600	87	7.74
19:00	1680	57	3.7
20:00	960	33	3.61
21:00	320	11	7.32
22:00	40	2	3.6
23:00	0	0	0

$$I = \frac{\lambda}{\mu}$$

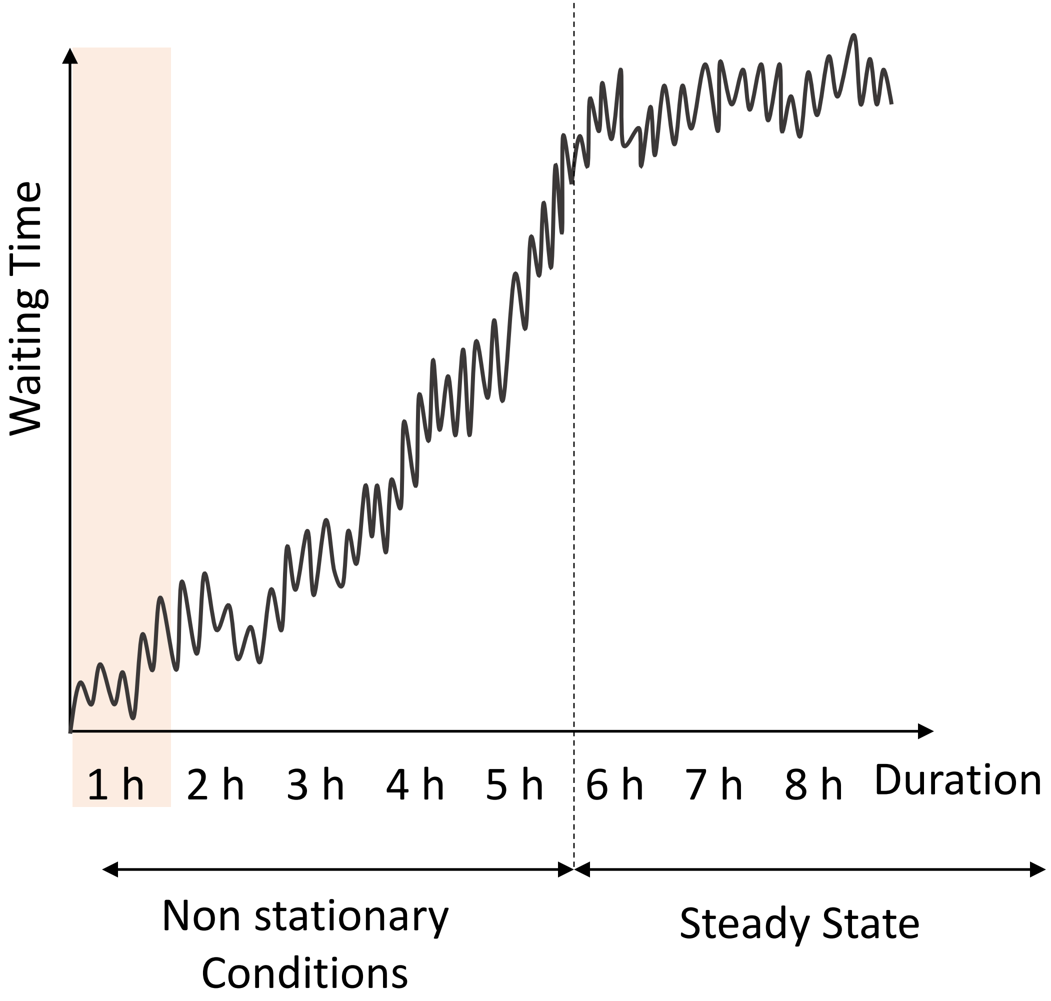
$$\rho = \frac{I}{s}$$

$$P_0 = \frac{1}{\sum_{n=0}^{s-1} \frac{I^n}{n!} + \frac{I^s}{S!\,(1-\rho)}}$$

$$L_q = \frac{P_0 I^s \rho}{S!\,(1-\rho)^2}$$

Non-stationary Conditions

- However, it is important to recognize that queues in many systems:
  - Build up over time (non-stationary state)
  - Demand patterns are not constant over the day
  - First arrivals get no delay, later arrivals join growing queue



Non-Stationary Queuing Systems

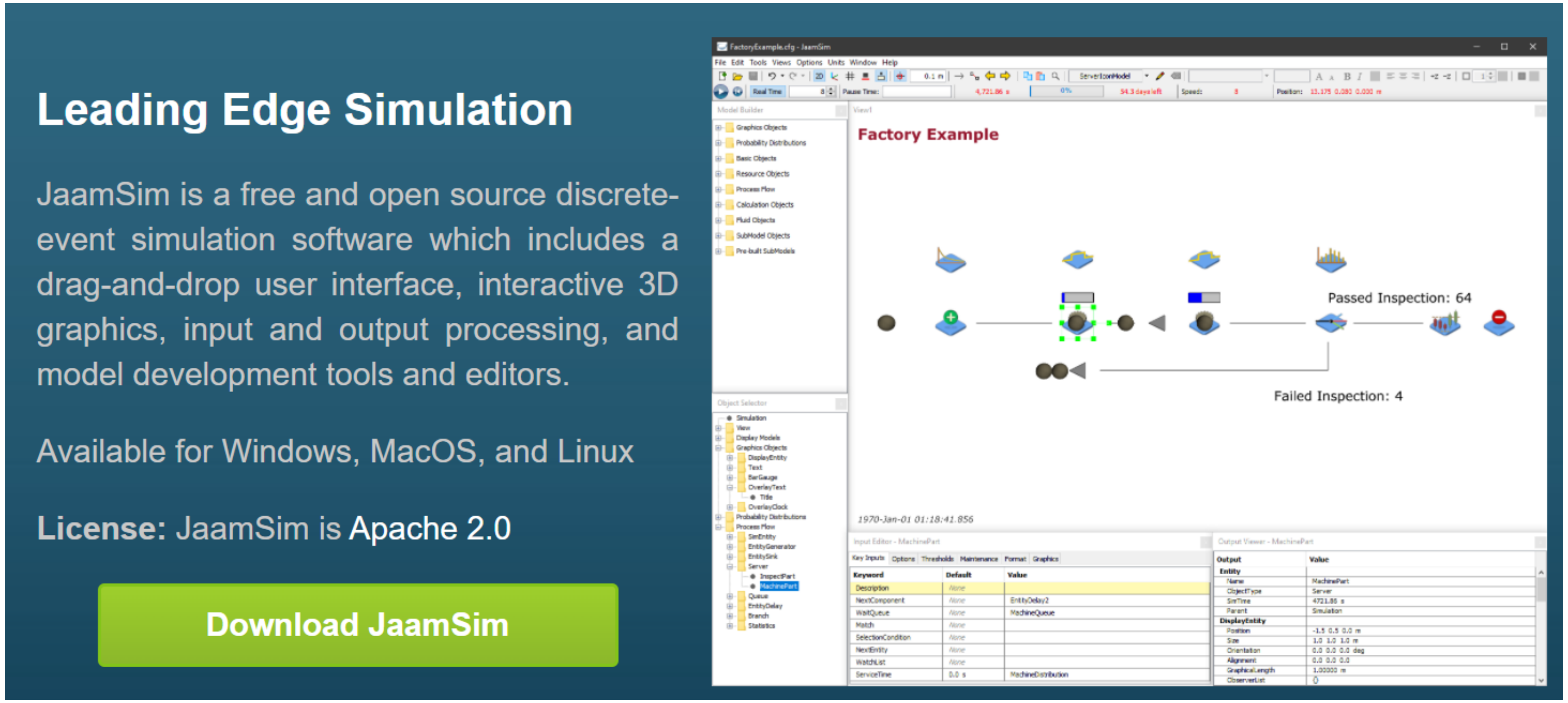
- Many service and production systems operate under dynamic conditions. These systems are often named as **non-stationary queueing systems**, as steady state conditions are never achieved.
- A characteristic trait of these systems is that the **demand rate may exceed the service capacity** at certain periods of the day - temporal overloading.
- During overloaded periods queues build up - overloaded periods must be followed by periods of low demand to ensure that queues return to acceptable levels.
- Complex simulations models are often utilized to analyse and optimize the performance of these systems. However, optimization is generally difficult and time consuming due to the large number of variables that can be adjusted by decision-makers

Non-Stationary Queuing Systems

- Examples of non-stationary queueing systems can be found everywhere: **aviation systems** (check-in, security checkpoints, flight scheduling); **healthcare systems** (resource and staff allocation), **transportation systems** (crew and fleet allocation), **logistic systems** (delivery management), **manufacturing systems** (production management), **computer systems** (server allocation), etc.
- COVID vaccination centres are a recent example of a time-dependent, non-stationary queueing system **– demand and capacity vary considerably across different periods of the day –** health officials need to manage the number of slots to make available per hour (demand rate control); and the number of staff required in the vaccination centres (service rate control); by considering the typical demand patterns (e.g. most people prefers to be vaccinated early or later in the day).

Simulation Models

- Steady-state equations are not valid in non-stationary queues
- We can use simulation models to mimic queues and optimize service and demand rates



Source: https://jaamsim.com/
Tutorial: https://www.youtube.com/watch?v=8DhFtfxZV0A

Discrete- Event Simulation – Steady State

Input Editor - ExponentialDistribution1

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	None	
CustomOutputList	None	
UnitType	None	TimeUnit
RandomSeed	None	1
MinValue	0.0 h	0 s
MaxValue	Infinity h	
Mean	2.77777777777...	1.125 s

3200 Pax/hour= 1 Pax every 1.125 sec

Input Editor - ExponentialDistribution2

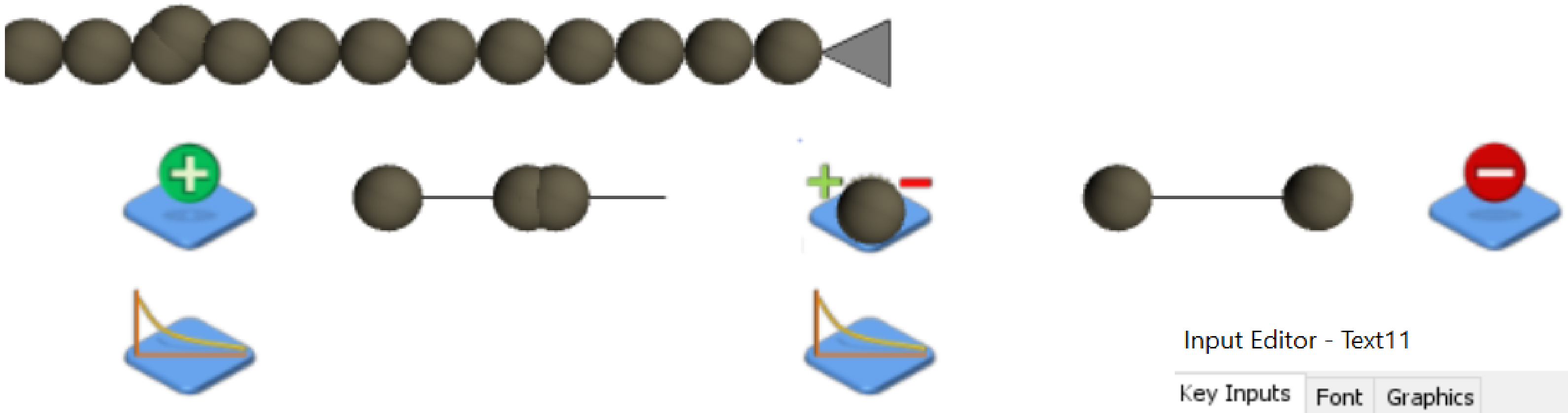
Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	None	
CustomOutputList	None	
UnitType	None	TimeUnit
RandomSeed	None	2
MinValue	0.0 h	0 s
MaxValue	Infinity h	
Mean	2.77777777777...	120 s

30 Pax/hour= 1 Pax every 120 sec

Discrete- Event Simulation – Steady State

Avg. Time in the System (min) **8.43**



Input Editor - Text11

Key Inputs | Font | Graphics

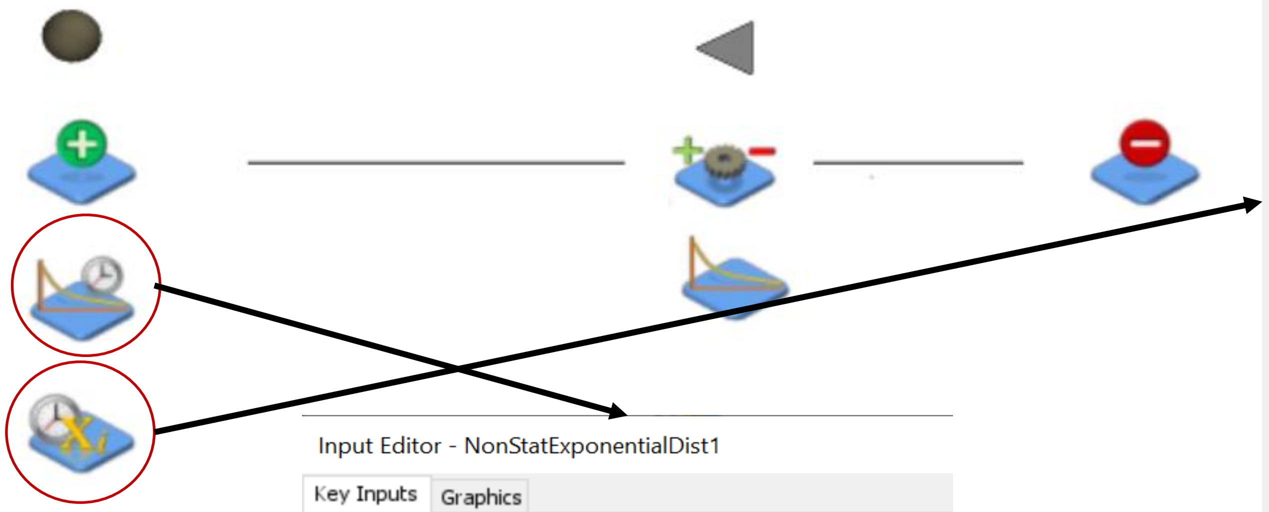
Keyword	Default	Value
AttributeDefinitionList	None	
CustomOutputList	None	
Format	%s	%.1f
UnitType	None	
Unit	None	
DataSource	None	[Queue1].Queue…
FailText	Input Error	

Example – Steady State Results

Time	Dem.	Min Check-in	Q Model	JaamSim
			Expected Time in System (min)	Expected Time in System (min)
04:00	0	0	0	0
05:00	0	0	0	0
06:00	40	2	3.6	3.59
07:00	320	11	7.32	7.82
08:00	1120	38	4.63	4.88
09:00	2280	77	3.74	3.88
10:00	2480	83	7.74	8.09
11:00	2480	83	7.74	8.09
12:00	2160	73	3.73	3.85
13:00	1880	63	7.7	8.55
14:00	2240	75	7.72	8.73
15:00	2440	82	4.74	5.12
16:00	2760	93	3.76	3.98
17:00	3200	107	7.77	**8.44**
18:00	2600	87	7.74	8.27
19:00	1680	57	3.7	3.83
20:00	960	33	3.61	3.67
21:00	320	11	7.32	7.82
22:00	40	2	3.6	3.59
23:00	0	0	0	0

Discrete- Event Simulation – NSS Conditions

Avg. Time in the System (min) **NaN**



Input Editor - NonStatExponentialDist1

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	None	
CustomOutputList	None	
RandomSeed	None	3
MinValue	0.0 h	0 h
MaxValue	Infinity h	
ExpectedArrivals	None	TimeSeries1

'' { } | this Sim null Entity

```
{ 0 h 0 }
{ 07 h 40 }
{ 08 h 360 }
{ 09 h 1480 }
{ 10 h 3760 }
{ 11 h 6240 }
{ 12 h 8720 }
{ 13 h 10880 }
{ 14 h 12760 }
{ 15 h 15000 }
{ 16 h 17440 }
{ 17 h 20200 }
{ 18 h 23400 }
{ 19 h 26000 }
{ 20 h 27680 }
{ 21 h 28640 }
{ 22 h 28960 }
{ 23 h 29000 }
{ 24 h 29000 }
```

NS Conditions - Results

Input Editor - EntityProcessor1

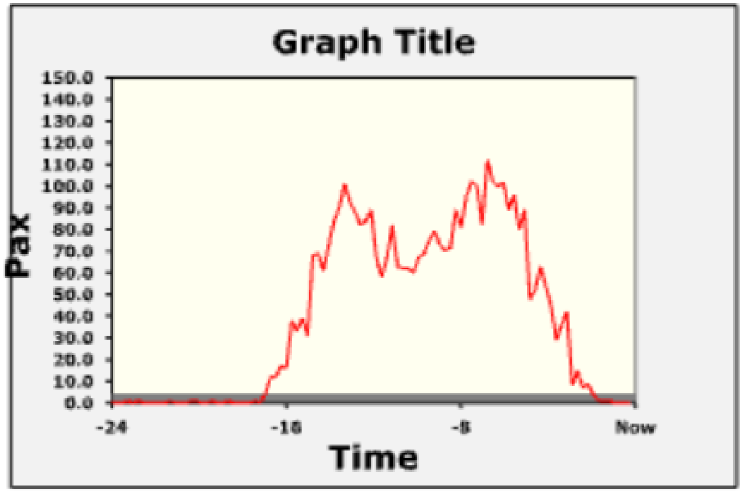
Key Inputs | Thresholds | Maintenance | Format | Graphics

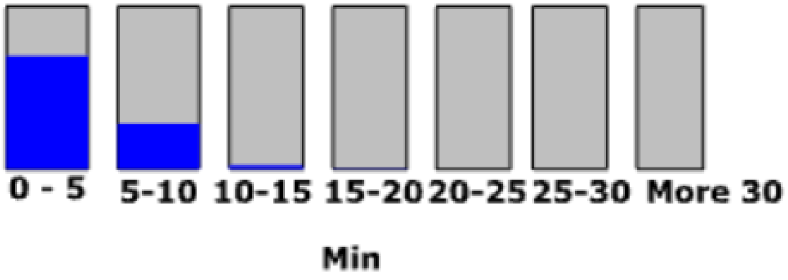
Keyword	Default	Value
Trace	FALSE	
AttributeDefinitionList	None	
CustomOutputList	None	
NextComponent	None	EntityConveyor2
StateAssignment	None	
WaitQueue	None	Queue1
Match	None	
ResourceList	None	
NumberOfUnits	{ 1.0 }	
Capacity	1.0	10000
ServiceTime	0.0 h	ExponentialDistribution2

Open an Infinite Number of Servers

Avg. Time in the System (min) 2.13

Max. Time in the System (min) 30.99





NS Conditions - Results

Input Editor - EntityProcessor1

Key Inputs | Thresholds | Maintenance | Format | Graphics

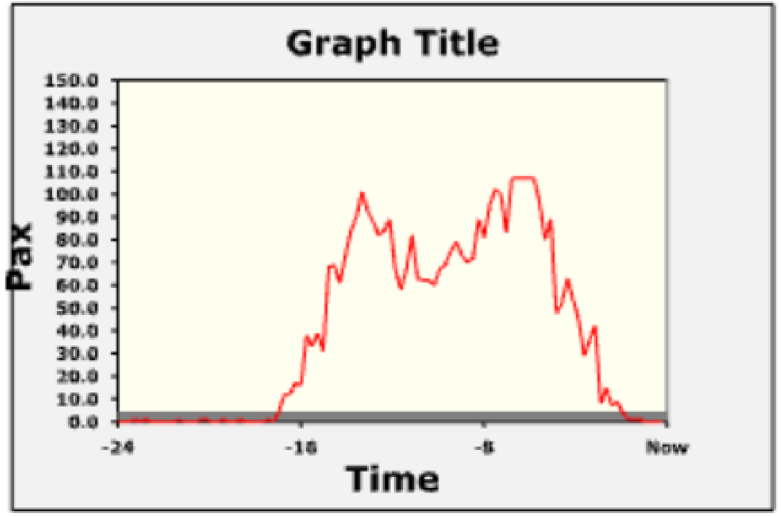
Keyword	Default	Value
Trace	FALSE	
AttributeDefinitionList	None	
CustomOutputList	None	
NextComponent	None	EntityConveyor2
StateAssignment	None	
WaitQueue	None	Queue1
Match	None	
ResourceList	None	
NumberOfUnits	{ 1.0 }	
Capacity	1.0	107
ServiceTime	0.0 h	ExponentialDistribution2

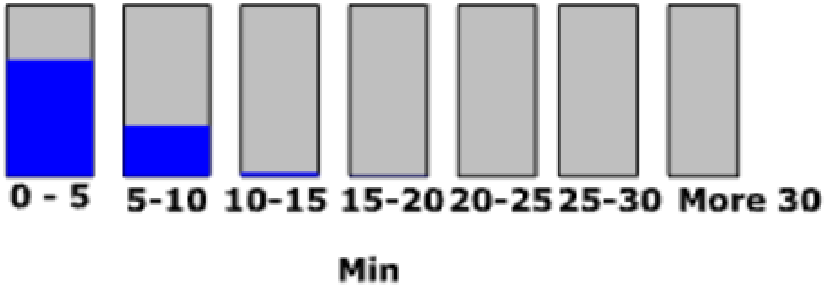
Open 107 serves across the entire day

Avg. Time in the System (min) 2.20

Max. Time in the System (min) 30.99

Under steady state conditions, the model predicts 8.88 mins of avg. Time in the system





NS Conditions - Results

Input Editor - EntityProcessor1

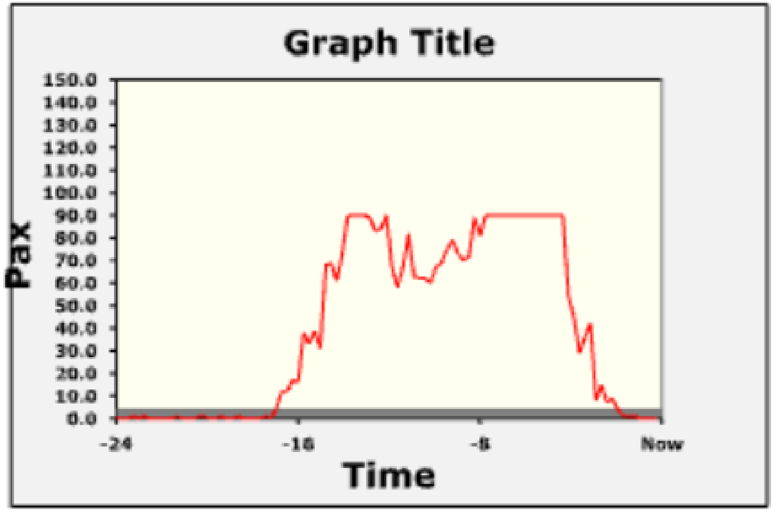
Key Inputs | Thresholds | Maintenance | Format | Graphics

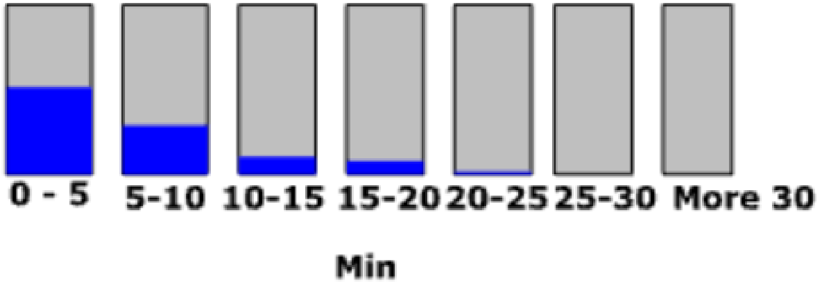
Keyword	Default	Value
Trace	FALSE	
AttributeDefinitionList	None	
CustomOutputList	None	
NextComponent	None	EntityConveyor2
StateAssignment	None	
WaitQueue	None	Queue1
Match	None	
ResourceList	None	
NumberOfUnits	{ 1.0 }	
Capacity	1.0	90
ServiceTime	0.0 h	ExponentialDistribution2

Open 90 serves across the entire day

Avg. Time in the System (min) 4.35

Max. Time in the System (min) 40.08





NS Conditions - Results

Input Editor - EntityProcessor1

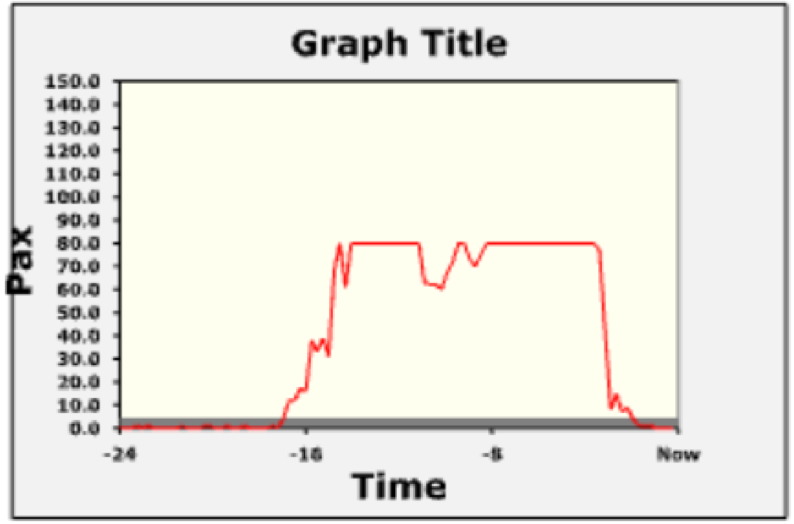
Key Inputs | Thresholds | Maintenance | Format | Graphics

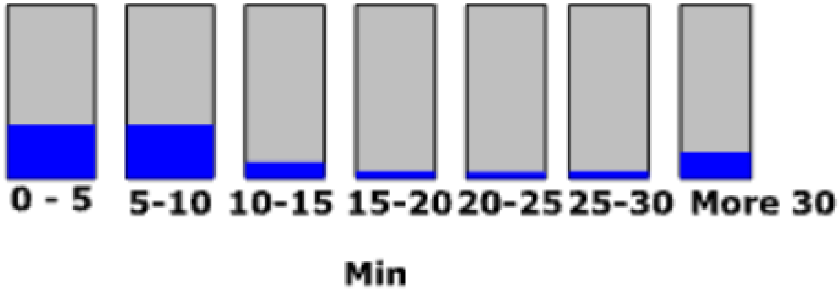
Keyword	Default	Value
Trace	FALSE	
AttributeDefinitionList	None	
CustomOutputList	None	
NextComponent	None	EntityConveyor2
StateAssignment	None	
WaitQueue	None	Queue1
Match	None	
ResourceList	None	
NumberOfUnits	{ 1.0 }	
Capacity	1.0	80
ServiceTime	0.0 h	ExponentialDistribution2

Open 80 serves across the entire day

Avg. Time in the System (min) 10.52

Max. Time in the System (min) 63.06





NSS Conditions - Results

Input Editor - EntityProcessor1

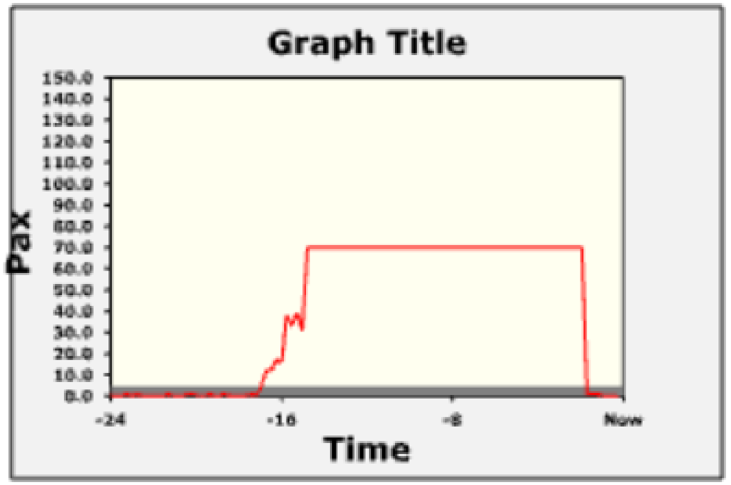
Key Inputs | Thresholds | Maintenance | Format | Graphics

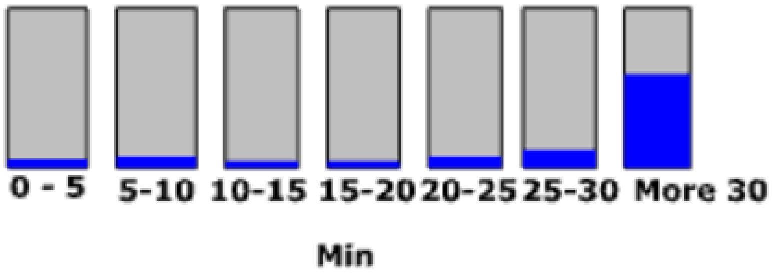
Keyword	Default	Value
Trace	FALSE	
AttributeDefinitionList	None	
CustomOutputList	None	
NextComponent	None	EntityConveyor2
StateAssignment	None	
WaitQueue	None	Queue1
Match	None	
ResourceList	None	
NumberOfUnits	{ 1.0 }	
Capacity	1.0	70
ServiceTime	0.0 h	ExponentialDistribution2

Open 70 serves across the entire day

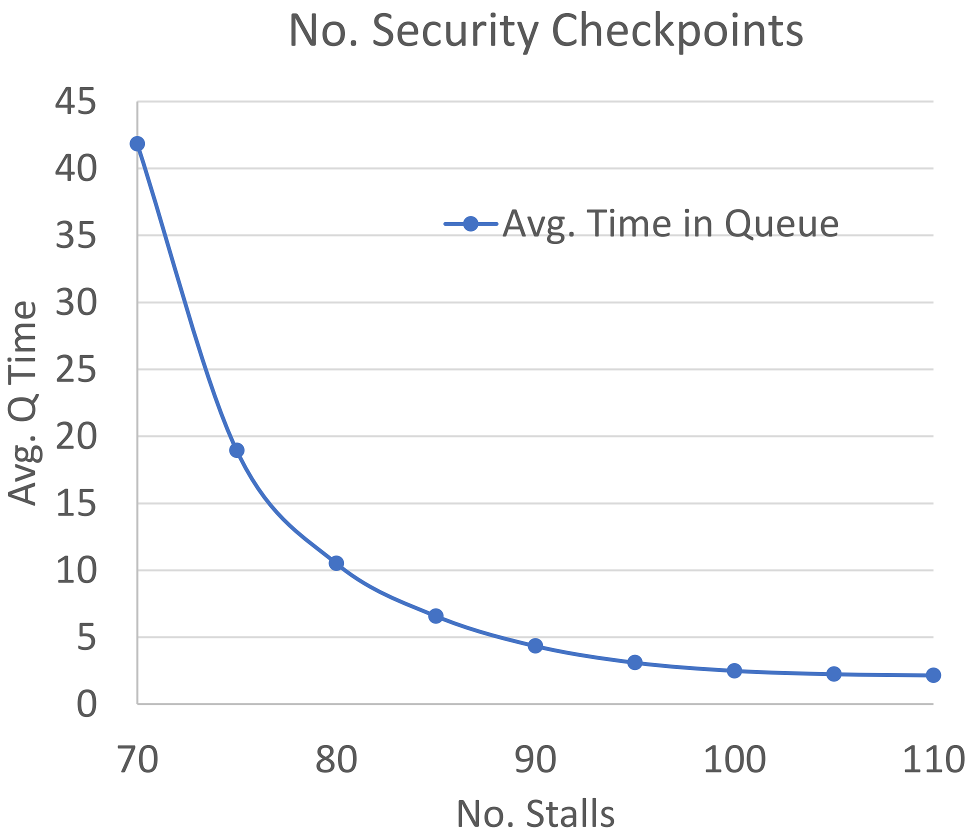
Avg. Time in the System (min) 41.85

Max. Time in the System (min) 136.28





NS Conditions - Results



- This analysis only shows part of the optimization process of non-stationary systems
  - What about having a variable number of servers across the day (no need to have 100 stalls open during the entire day)
  - What about controlling the arrival demand by imposing slot limits (such as in vaccination centres, slot times are assigned to people)?

Multi-Objective Problem aiming to optimize 3 main objectives: level of service (e.g. minimize waiting time) ; demand acceptance rate (minimize demand displacement) ; service costs (minimize the number of servers to open per hour)

Capacity Management in Non-Stationary Queuing Systems – NSGA II

Nuno Antunes Ribeiro

Assistant Professor

Singapore University of Technology and Design

Engineering Systems and Design

Capacity Management

Capacity management is the field of research that aims to optimize infrastructure operations while having "just enough" resources required to run applications and services without interruptions in desired performance.

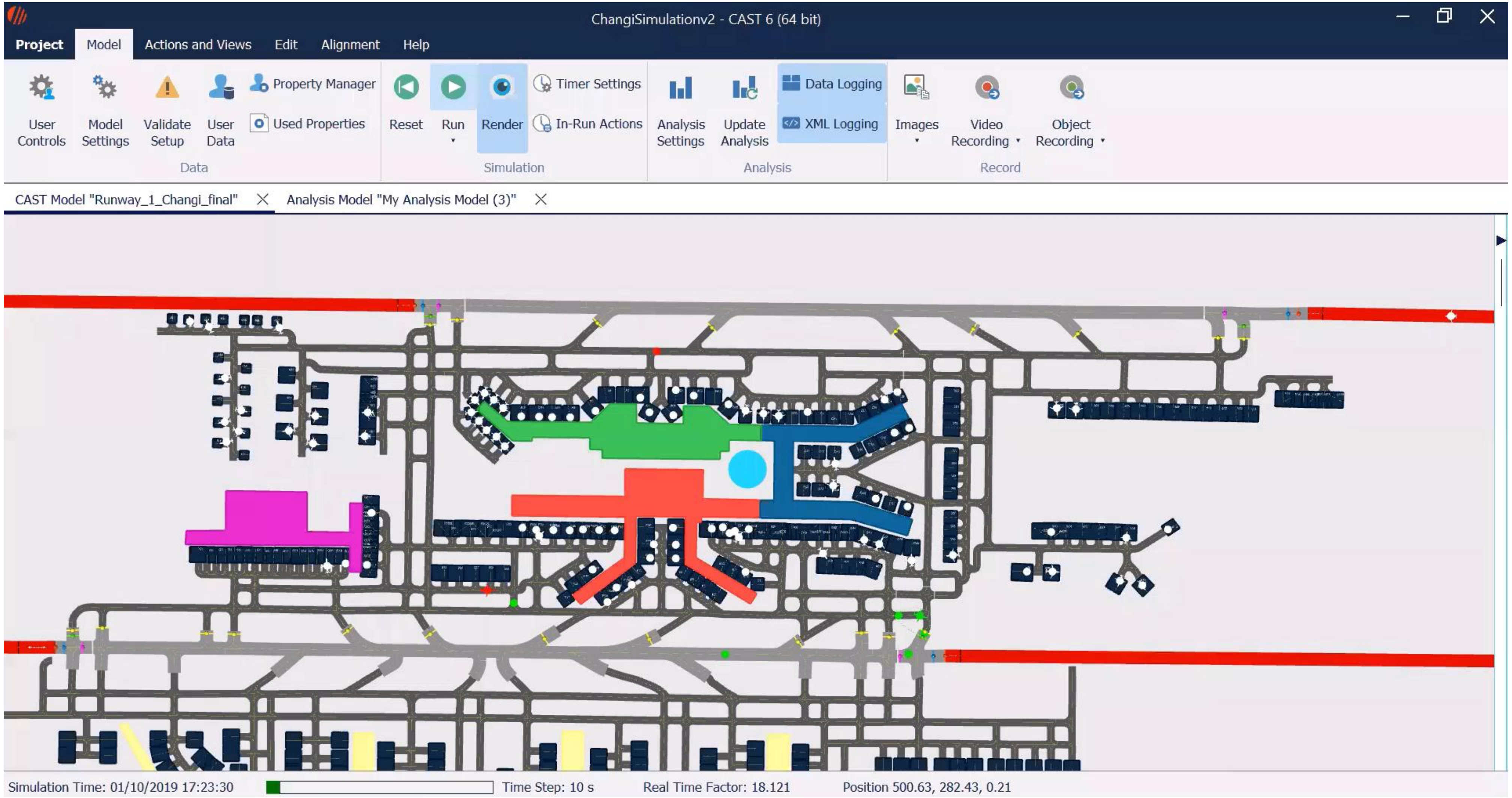
Two main capacity management strategies are implement:

- Increasing service capacity – By investing on resource capacity – more staff; more machines, more infrastructure, etc.
- Efficiently distributing demand – By imposing limits on scheduling – slot scheduling; demand rate control, etc.

Airport Slot Allocation Case Study

- Airport infrastructure Capacity is fixed by the number of runways in the airport – for instance Changi Airport runway system have a capacity of around 10 arrival flights every 15 minutes.
- Slot allocation is used to efficiently distribute demand across the day
- **Question:** How many slots to make available per hour given airport capacity constraints (i.e. no. of runways) and airline’s slot requests (i.e. slot times requested by the airlines to operate their flights)?
  - Two main objectives to optimize:
    - Minimize expected fight delays in the airport
    - Minimize slot displacement to the airlines
  - Decisions Variable
    - Number of slots to make available per hour

Airport Simulation



Declared Capacity Tool

Legend:

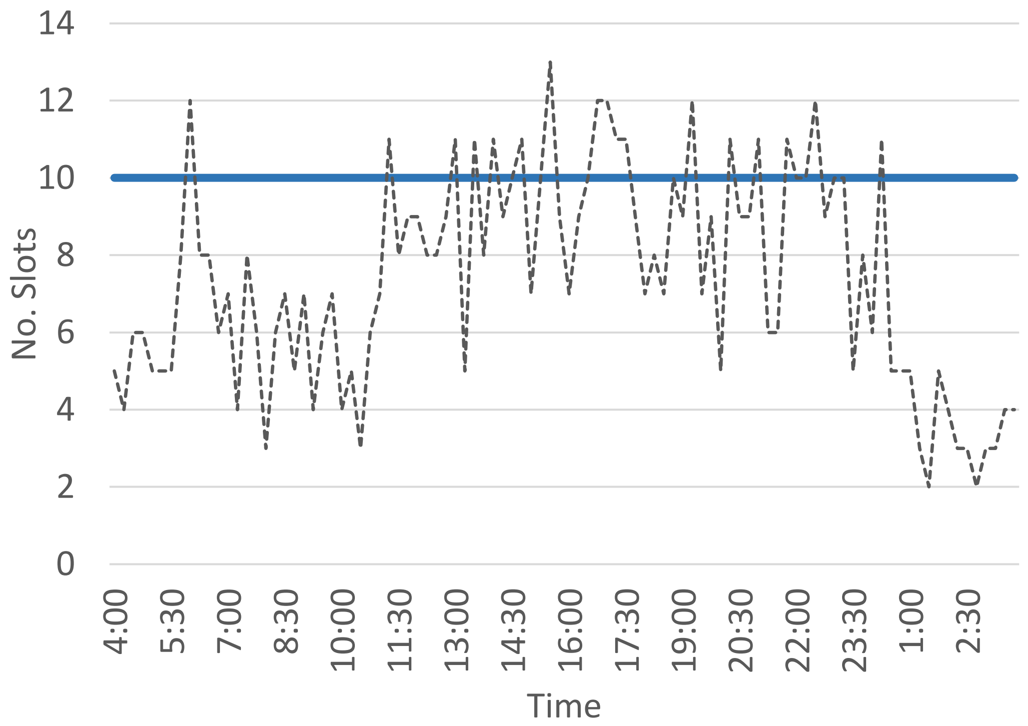
Requested demand

Allocated demand

Slot Limit

Expected average delays

- **Outputs**

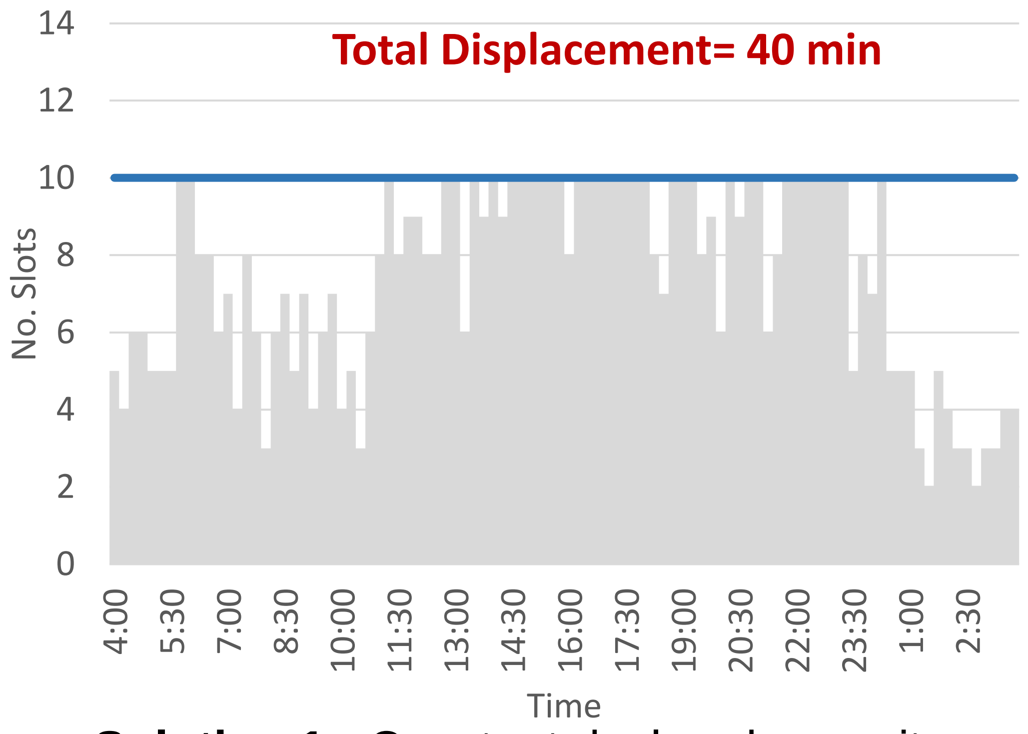


Solution 1 - Constant declared capacity = 10 mov/hour (CAAS current capacity)

Declared Capacity Tool

Legend:
- Requested demand
- Allocated demand
- Slot Limit
- Expected average delays

- **Outputs**



Solution 1 - Constant declared capacity = 10 mov/hour (CAAS current capacity)

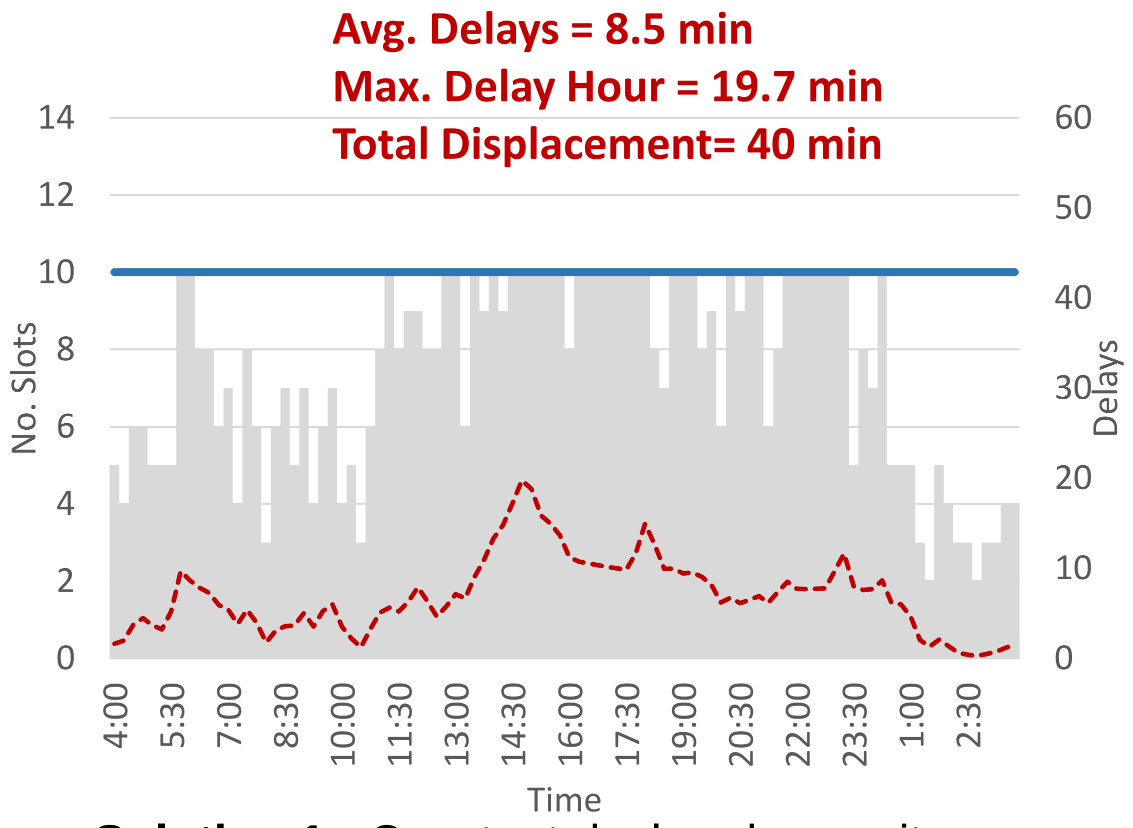
Slot Allocation Model

Declared Capacity Tool

Legend:

- Requested demand
- Allocated demand
- Slot Limit
- Expected average delays

- **Outputs**

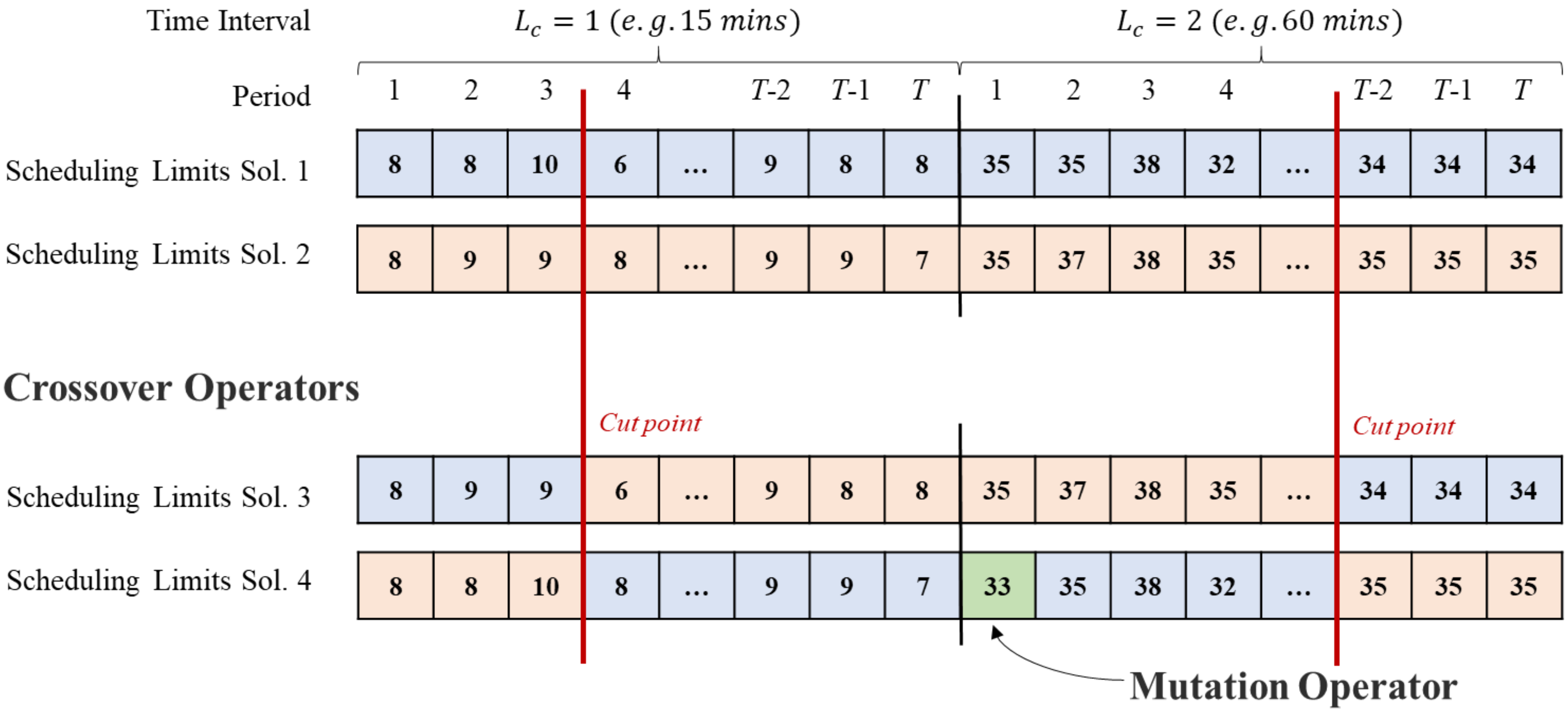


Solution 1 - Constant declared capacity = 10 mov/hour (CAAS current capacity)

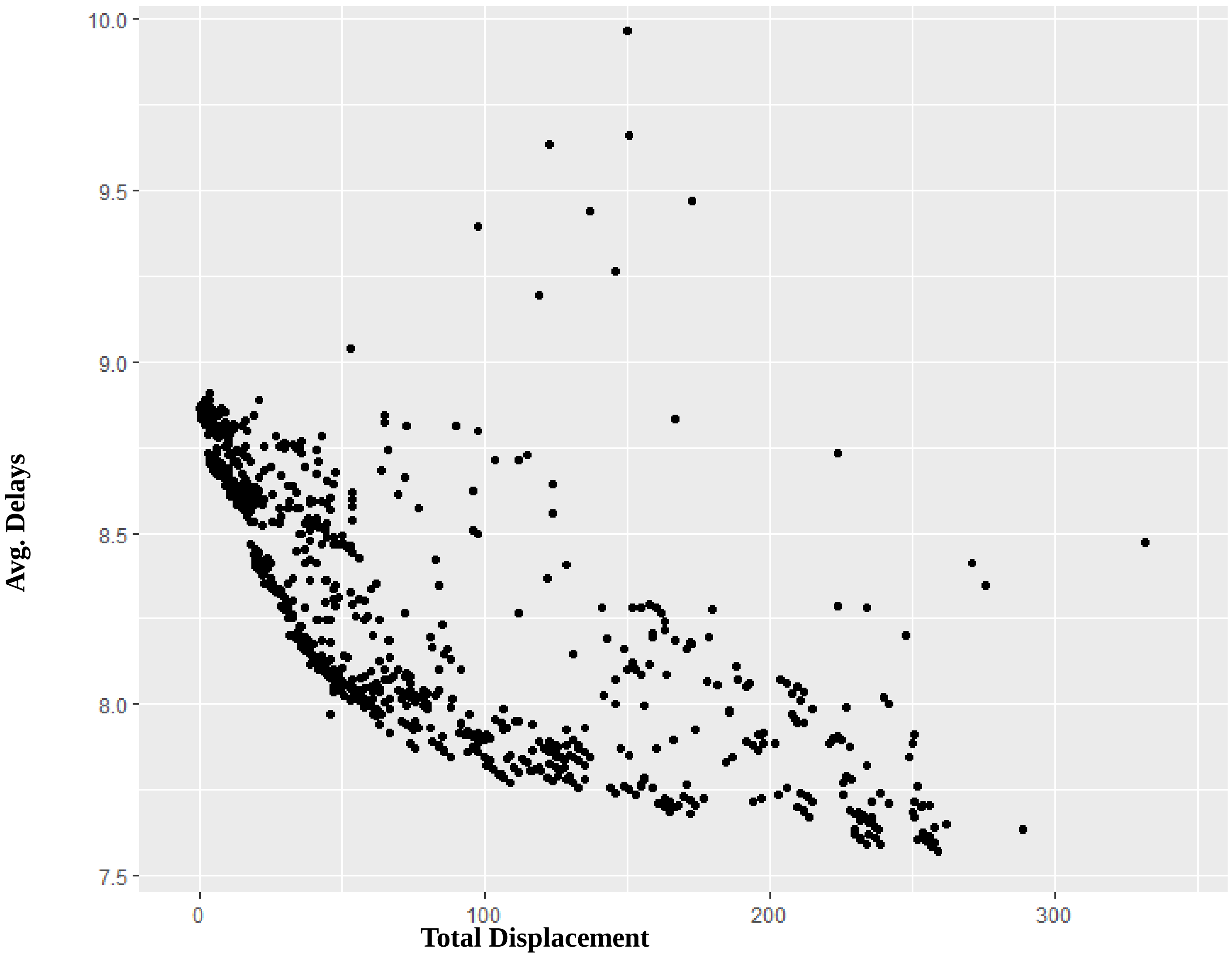
Simulation Model

Genetic Algorithm

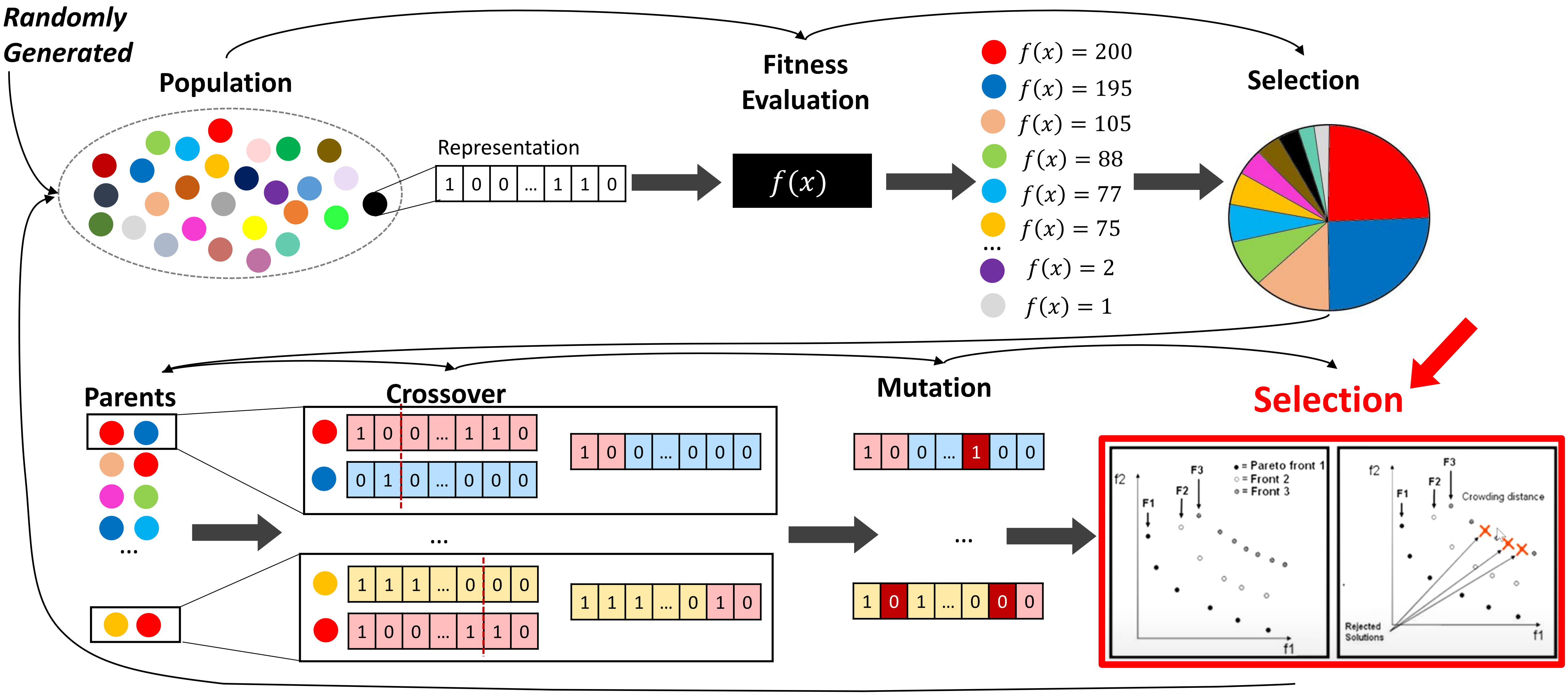
Chromosome encoding



NSGA-II – After 1 hour of Computation

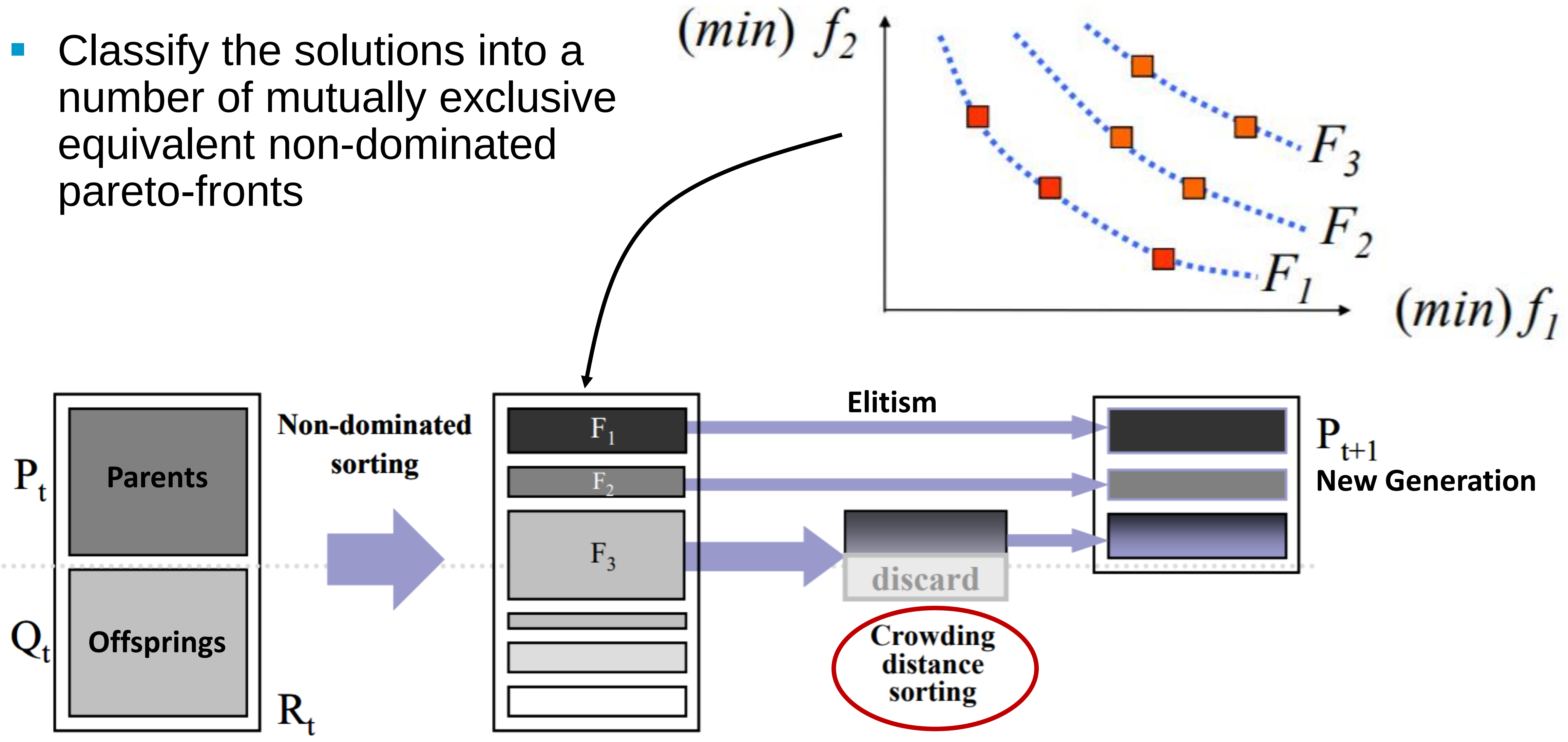


Recall: NSGA-II

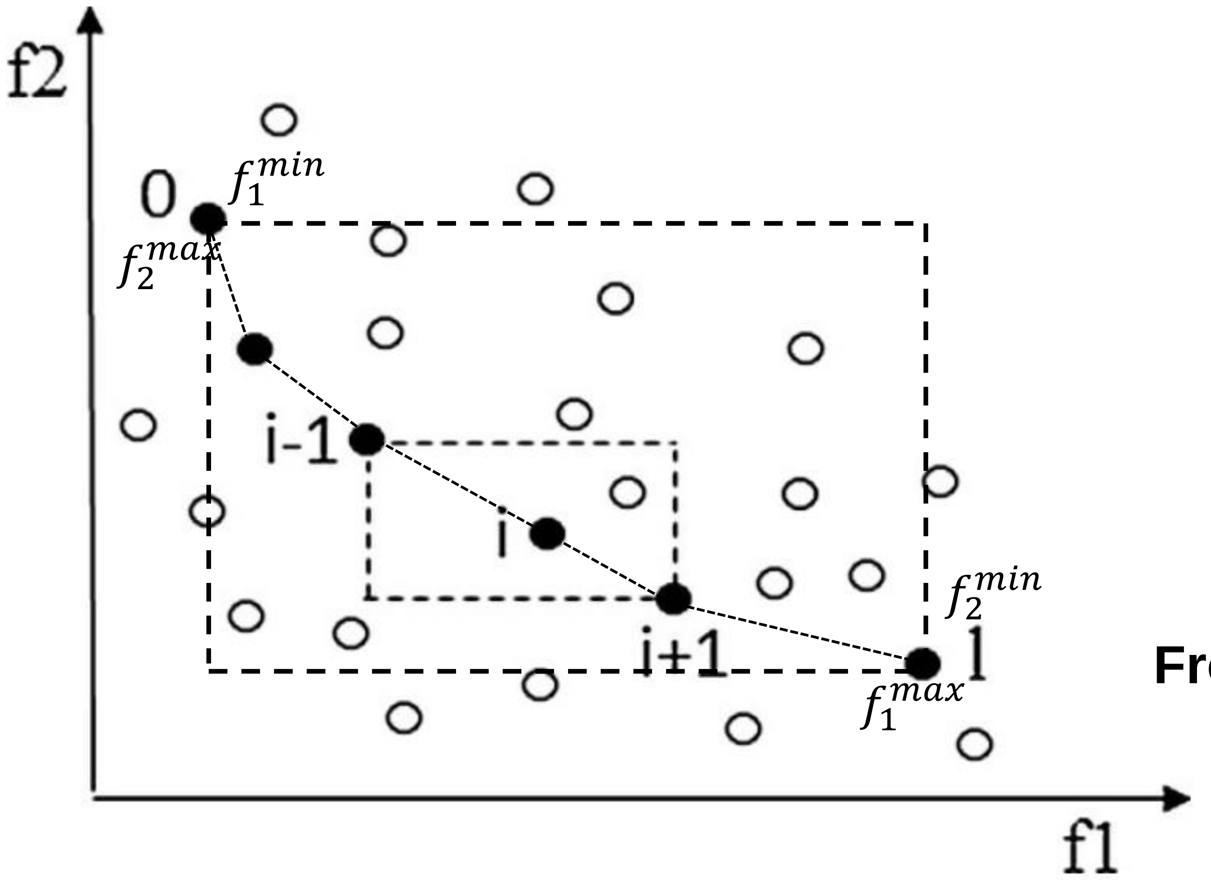


Recall: Non-Dominated Sorting

- Classify the solutions into a number of mutually exclusive equivalent non-dominated pareto-fronts



Recall: Crowding Distance



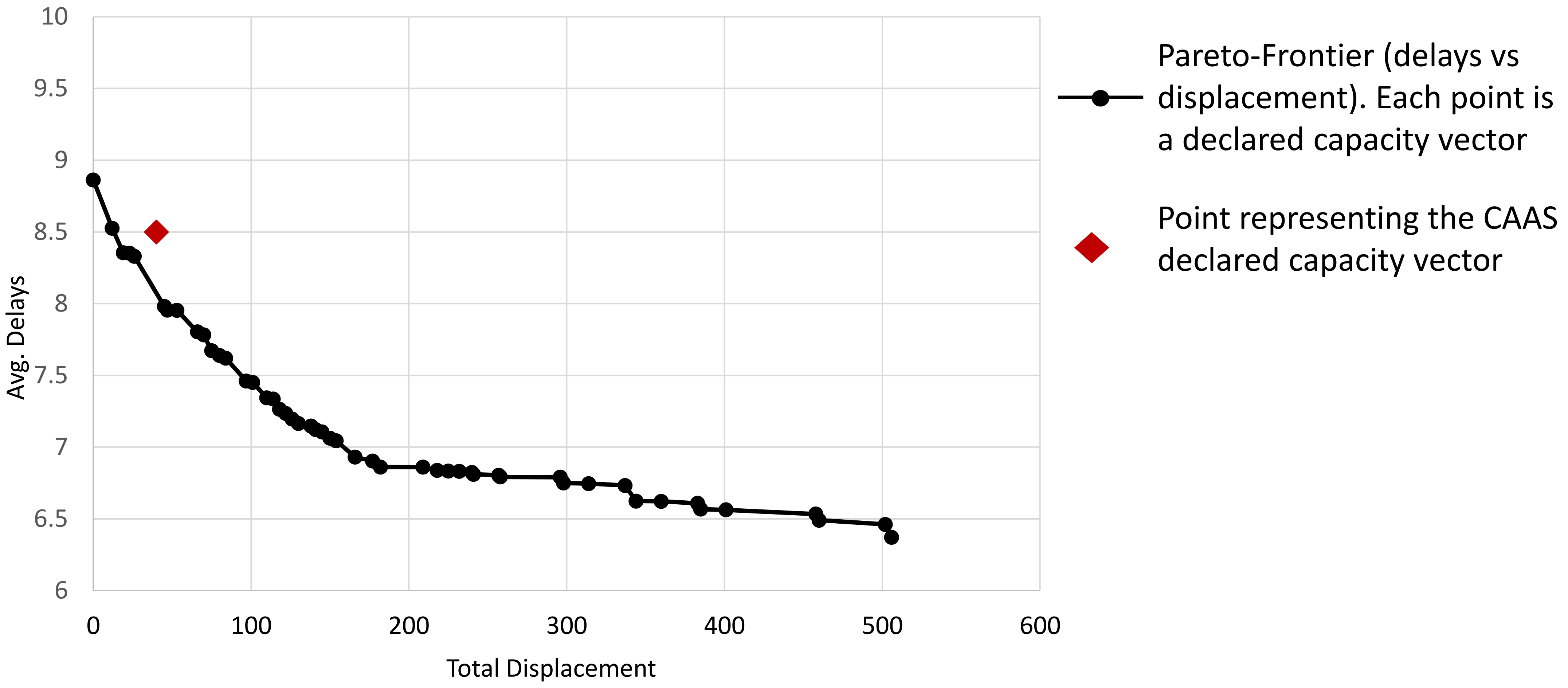
Front n

$$CD = \frac{f_1^{i+1} - f_1^{i-1}}{f_1^{max} - f_1^{min}} + \frac{f_2^{i+1} - f_2^{i-1}}{f_2^{max} - f_2^{min}}$$

$$CD = \sum_{M} \frac{f_m^{i+1} - f_m^{i-1}}{f_m^{max} - f_m^{min}}$$

M – Set of Objectives

NSGA-II – Pareto-Frontier



NSGA-II – Optimal Solutions

6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	0
10	8	6	7	8	11	12	10	10	11	11	12	11	9	10	13	10	10	10
9	7	6	7	7	9	9	9	10	11	10	11	11	10	10	10	10	10	10
10	7	6	7	7	9	10	10	9	10	10	10	10	10	10	11	12	10	10
10	7	6	7	8	11	12	10	10	11	11	11	11	9	10	10	10	10	10
10	8	6	7	8	11	10	10	10	10	10	10	10	9	10	9	10	10	10
10	7	6	7	8	10	10	10	10	11	11	11	11	9	10	10	10	10	6
9	7	6	7	8	9	9	9	10	10	10	11	10	9	10	10	10	10	10
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	8	6	7	7	9	12	9	9	11	9	10	10	9	9	10	11	11	10
10	7	6	7	8	9	9	9	9	9	10	10	11	9	10	10	10	10	10
10	7	6	7	8	9	9	10	10	10	10	10	10	9	10	10	9	13	6
10	7	6	7	8	9	9	9	9	10	10	10	10	10	9	9	9	9	9
9	7	6	7	8	9	9	9	10	10	10	11	10	9	9	9	9	9	9
10	8	6	7	8	9	9	9	10	10	10	10	10	9	9	9	9	9	9
10	7	6	7	8	9	9	9	10	10	10	10	10	9	9	9	9	9	9
9	7	6	7	8	9	9	9	10	10	10	10	10	9	9	9	9	9	9
10	7	6	7	8	11	8	9	10	10	10	10	10	9	9	9	9	9	9
9	7	6	7	8	9	9	9	10	10	10	10	10	9	9	9	9	10	6
10	7	6	7	8	11	8	9	9	10	9	9	9	9	9	9	12	10	6
9	7	6	7	7	9	9	9	9	9	9	9	9	9	9	9	9	9	9
10	7	6	7	8	10	9	9	10	10	10	10	10	9	10	9	10	10	9
10	7	6	7	8	10	8	9	9	10	9	9	9	9	9	9	9	10	6
10	7	6	7	7	9	9	9	9	9	9	9	9	9	9	9	9	10	6
10	7	6	7	8	9	9	9	9	9	9	9	9	9	9	9	9	10	6
9	7	6	7	7	9	9	9	9	9	9	9	9	9	9	9	9	9	6
9	7	6	7	8	9	9	9	9	9	9	9	9	9	9	9	9	9	6
10	7	6	7	8	10	8	9	9	9	9	9	9	9	9	9	9	9	6
10	8	6	7	8	9	8	10	10	10	9	9	9	9	9	9	10	10	6
9	7	6	7	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9
10	7	6	7	8	11	8	9	10	10	10	9	9	9	10	9	12	10	6
10	7	6	7	8	9	9	9	9	9	9	9	9	9	9	9	9	9	8
10	7	6	7	8	9	9	9	9	9	9	9	9	9	9	9	9	9	6

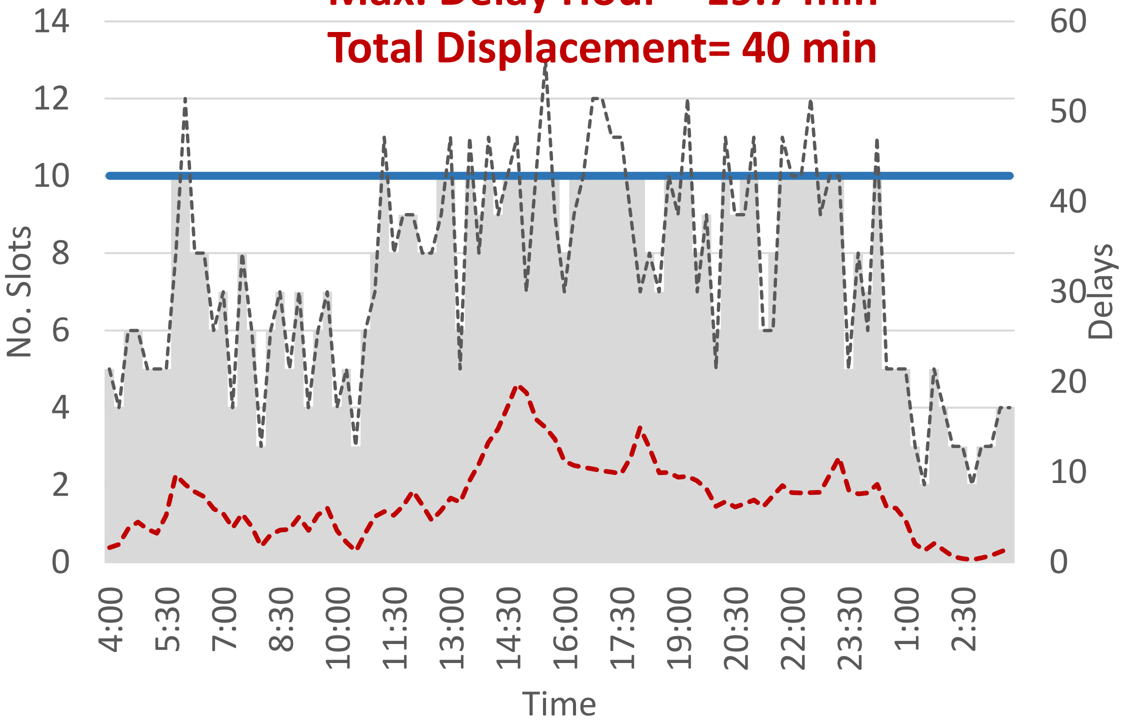
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	0
31	26	25	24	30	33	40	36	38	41	41	43	40	36	38	43	40	37	24
31	25	25	24	30	36	36	37	38	40	42	43	39	36	38	40	40	35	24
31	25	25	24	30	33	40	38	38	40	40	40	40	36	37	42	43	35	24
31	25	25	24	30	33	41	36	38	40	40	40	39	37	39	38	39	39	24
31	26	25	24	30	39	36	38	36	39	39	40	40	38	38	39	40	38	24
31	25	25	24	30	36	39	34	36	40	40	40	39	37	39	38	39	39	24
31	25	25	26	30	35	36	37	38	36	40	41	39	37	39	38	39	39	24
38	30	30	30	30	38	38	38	38	35	38	41	38	38	38	38	38	38	30
31	26	25	24	30	35	39	36	38	40	38	40	40	36	37	38	39	39	27
30	25	26	24	30	36	36	36	35	38	38	38	40	38	37	38	40	40	24
31	25	25	24	30	36	37	39	35	38	38	38	40	38	37	38	40	40	24
31	25	25	26	30	36	36	34	36	39	40	40	36	39	36	36	36	36	33
31	25	25	30	35	36	35	36	36	36	38	39	36	36	36	36	36	32	35
31	26	25	30	35	36	35	36	36	36	38	39	36	36	36	36	36	32	35
31	25	25	30	35	36	35	36	36	36	38	39	36	36	36	36	36	32	35
31	25	25	30	32	36	35	36	36	36	38	39	36	36	36	36	36	32	35
31	25	25	30	32	36	35	36	36	36	38	39	36	36	36	36	36	32	35
31	25	26	31	33	36	35	32	36	36	36	38	38	34	36	34	39	40	28
31	25	27	31	33	36	35	32	36	37	36	36	36	35	36	34	39	37	28
31	26	25	24	30	35	36	34	36	36	36	36	36	36	36	36	36	36	24
31	25	27	31	33	36	35	32	36	36	36	36	36	35	36	34	36	36	35
31	25	27	31	33	36	35	32	36	37	36	36	36	35	36	34	39	40	28
31	25	26	28	33	36	35	32	36	36	36	36	36	35	36	34	39	40	28
31	25	26	28	33	36	35	32	36	36	36	36	36	35	36	34	39	40	28
31	25	26	28	33	36	35	32	36	36	36	36	35	36	36	34	36	36	30
31	25	26	31	33	36	32	32	36	36	36	36	36	34	36	34	36	36	30
31	25	26	31	33	36	32	32	36	36	36	36	36	34	36	34	36	36	30
28	26	25	24	30	35	32	32	36	36	36	36	32	34	36	34	40	40	26
31	25	26	31	33	34	32	34	36	36	36	30	32	34	36	34	36	36	30
31	25	26	31	33	36	32	32	36	36	36	30	32	34	36	34	39	37	30
31	25	26	31	33	34	32	34	36	36	36	30	32	34	36	34	36	36	30
31	25	26	31	33	36	32	34	36	36	36	30	32	34	36	34	36	36	30

NSGA-II – Optimal Solutions

Legend:
- Requested demand
- Allocated demand
- Declared capacity
- Expected average delays

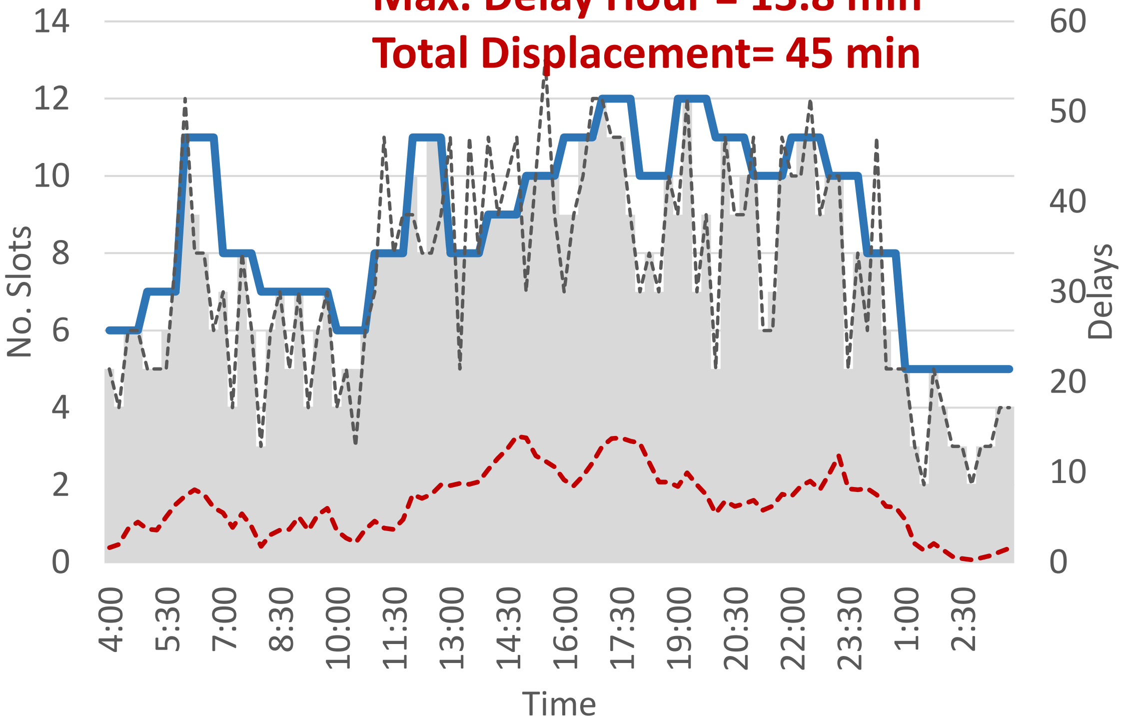
- **Outputs**

Avg. Delays = 8.5 min
Max. Delay Hour = 19.7 min
Total Displacement= 40 min



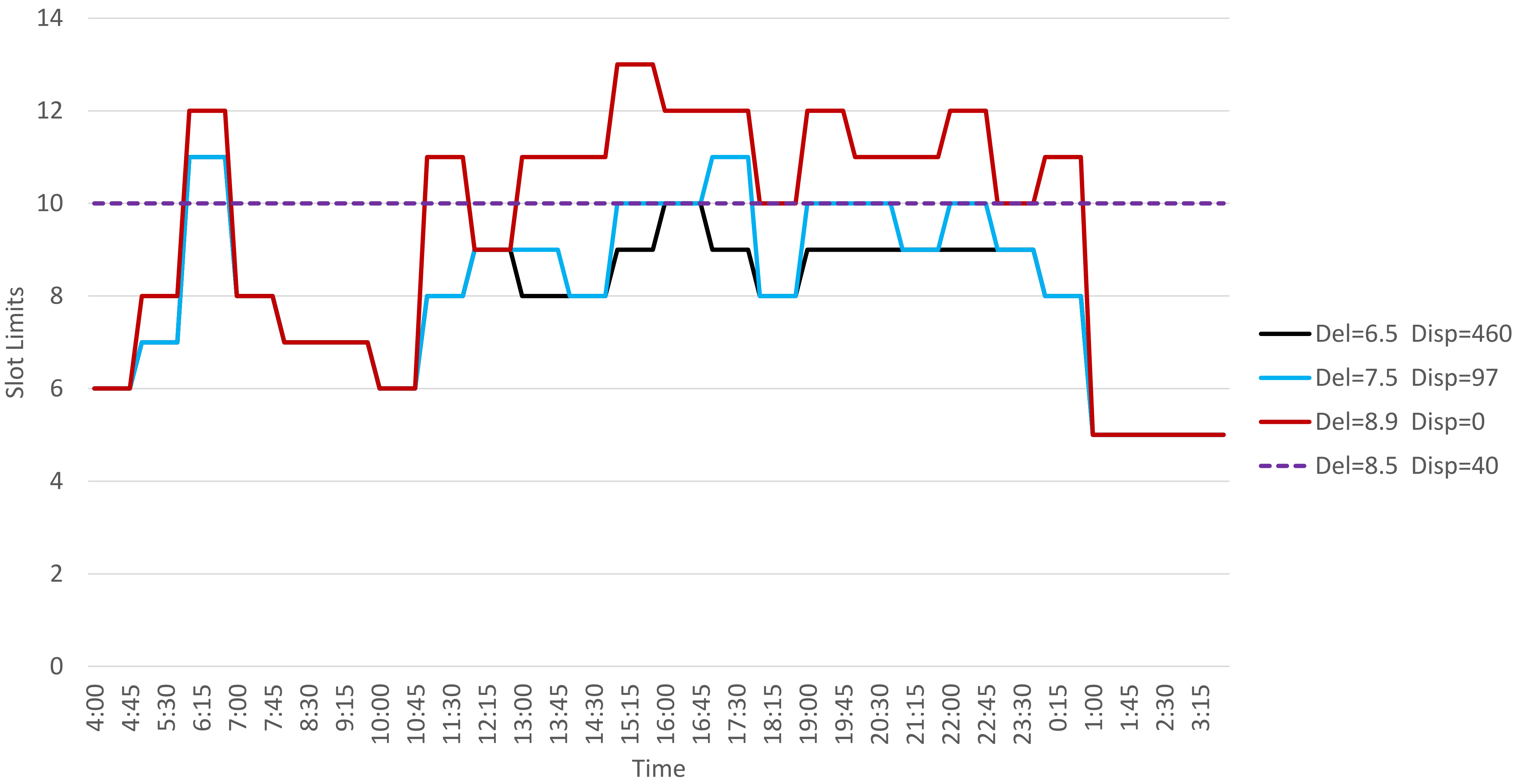
Solution 1 - Constant slot limit = 10 mov/hour

Avg. Delays = 8.0 min
Max. Delay Hour = 13.8 min
Total Displacement= 45 min

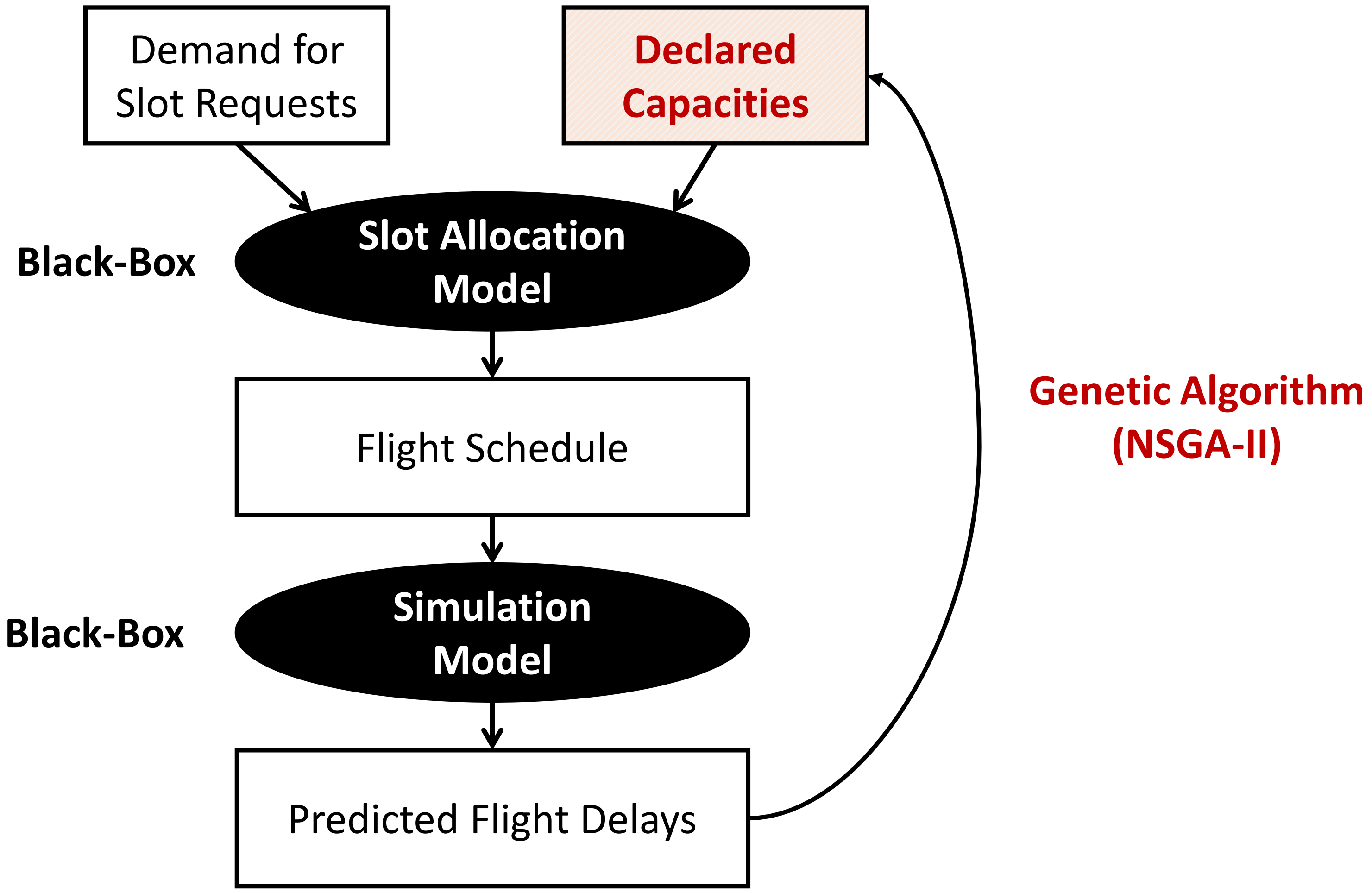


Solution 2 - Variable slot limit - optimized through NSGA-II Algorithm

NSGA-II – Optimal Solutions



Declared Capacity Algorithm



Meta-Modeling

Nuno Antunes Ribeiro

Assistant Professor

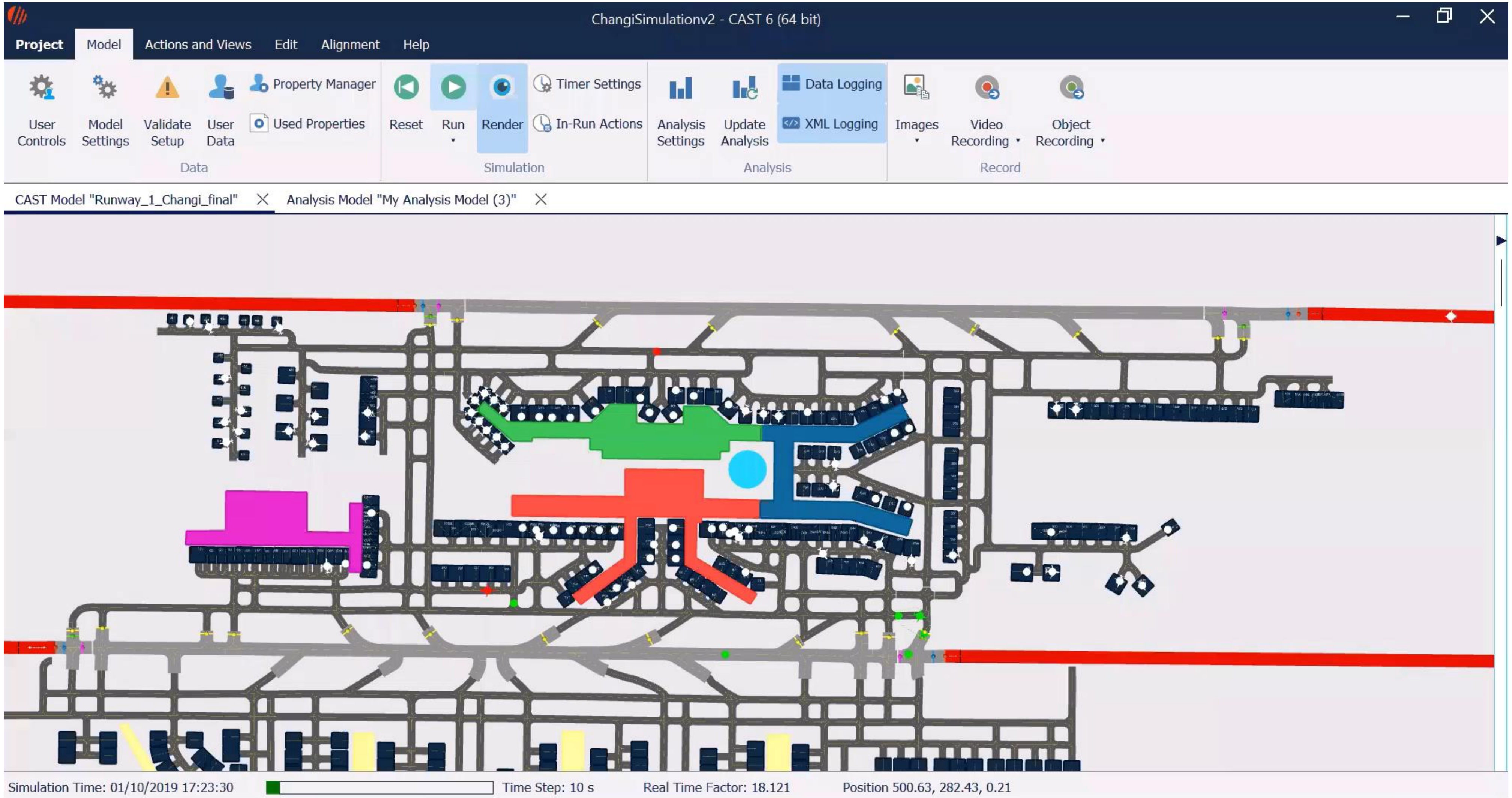
Singapore University of Technology and Design

Engineering Systems and Design

Meta-Models

- It is well known that most of the time, in metaheuristics, the time-intensive part is the evaluation of the objective function.
- In many problems, the objective function is quite costly to compute (e.g. simulations).
- The alternative to reduce this complexity is to approximate the objective function and then replace the original objective function by its approximation function.
- This approach is known as meta-modeling

Airport Simulation



1 Hour of Computation to run 1 month of schedules – This invalidates the iterative process we aim to apply by using metaheuristics approaches

Meta-modeling Techniques

- Many meta-modeling techniques may be employed for expensive objective functions. They are based on constructing an approximate model from a properly selected sample of solutions:
  - Analytical Approximations
  - Machine Learning Models
  - Neural Networks
  - Relaxed Simplified Model (e.g. ignore some constraints)
  - Model Decomposition

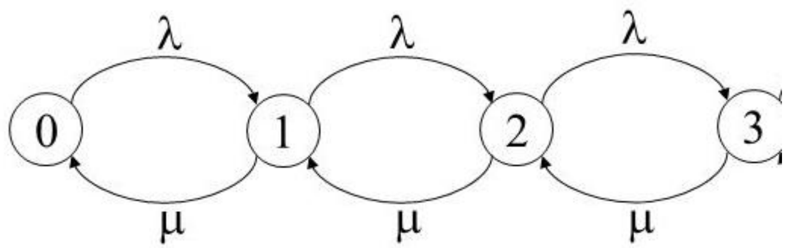
- There is a trade-off between the complexity of the model and its accuracy.

Predicting Flight Delays

Macroscopic Models

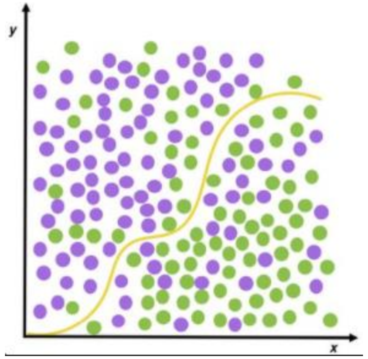
Markov-Chains + Queuing Theory

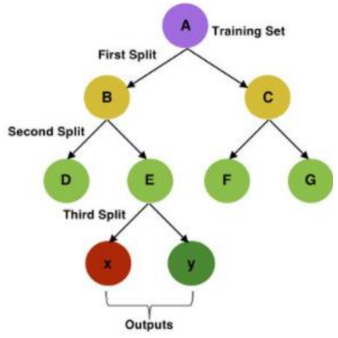
Delays are approximated using mathematical equations derived from Markov-chain theory



Machine Learning Models

Analyses historical data of flight delays to make predictions





CAST Simulation Tool

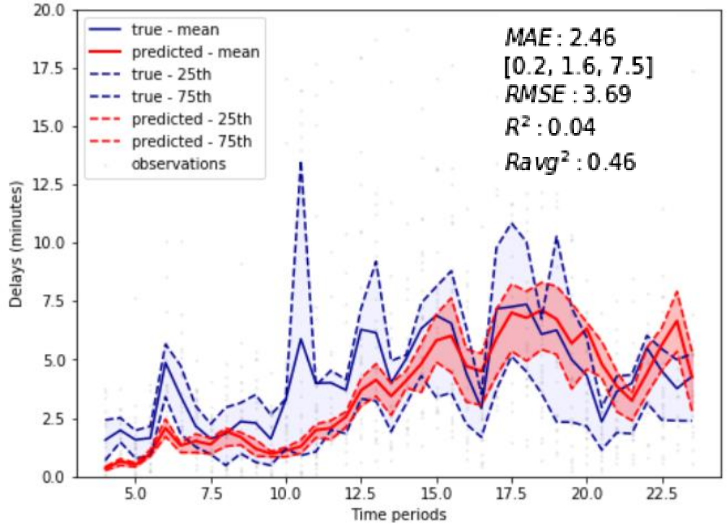
Flight operations are simulated in very detail using agent-based simulation

Microscopic Models

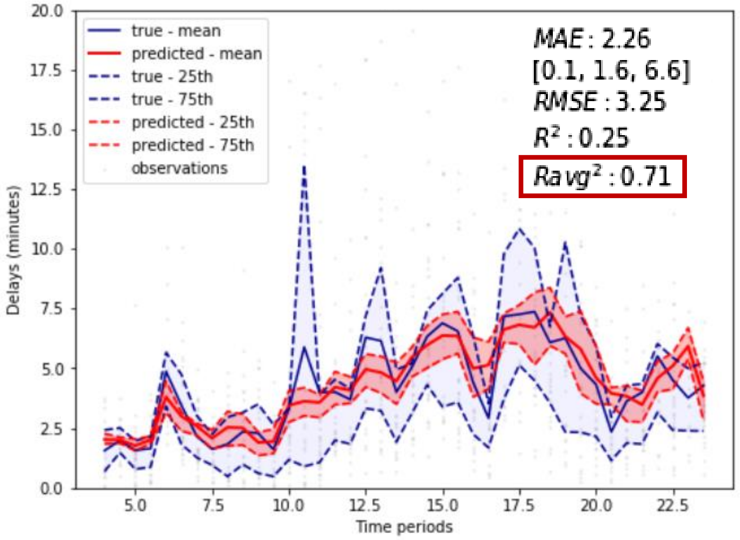
Predicting Flight Delays

- **We develop a random forest model to predict airport local delays by leveraging historic data from flight operations and meteorological conditions.**
- **Explanatory variables include: congestion indicators (no. arrivals, no. departures, congestion index, etc.), weather related variables (lightning count, wind speed, wind direction), queuing model predictions, time-of-the day dummy variables, etc.**

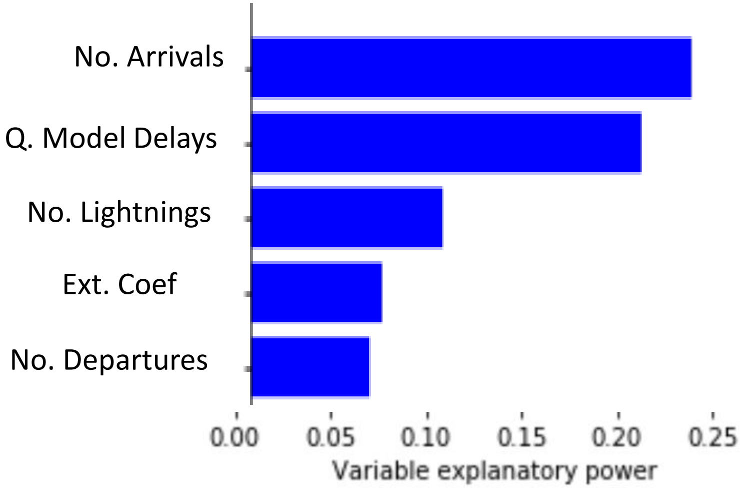
Queueing Model - Prediction



Random Forest - Prediction



Top 5 Explanatory Variables



Predicting Flight Delays

CAST Simulation Model