U.S. DEPARTMENT OF ENERGY

SMARTMOBILITY

Systems and Modeling for Accelerated Research in Transportation

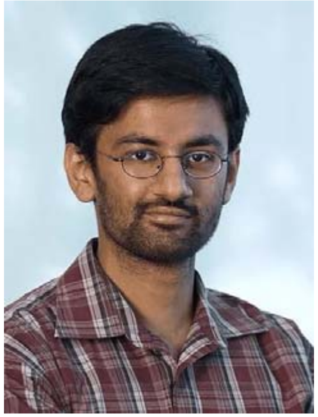
# Agent-based Simulation of the Ride-hailing Market

Presenter: Rashid Waraich (Lawrence Berkeley National Lab)
Development Team: Colin Sheppard, Sid Feygin, Andrew Campbell, Michael Zilske, Conveyal, 7 Summits LLC
Advisory Team: Anand Gopal, Tom Wenzel

MATSim User Meeting, Atlanta
June 23, 2018

# Key BEAM Contributors

Colin Sheppard


Rashid Waraich

Sid Feygin

Michael Zilske

Andrew Campbell

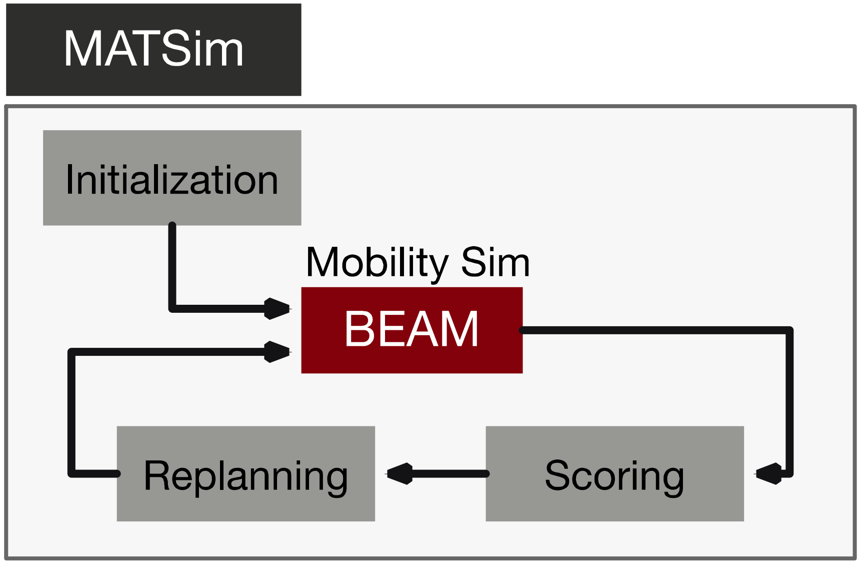
conveyal

7 Summits IT AG
taking you to the top

https://github.com/LBNL-UCB-STI/beam

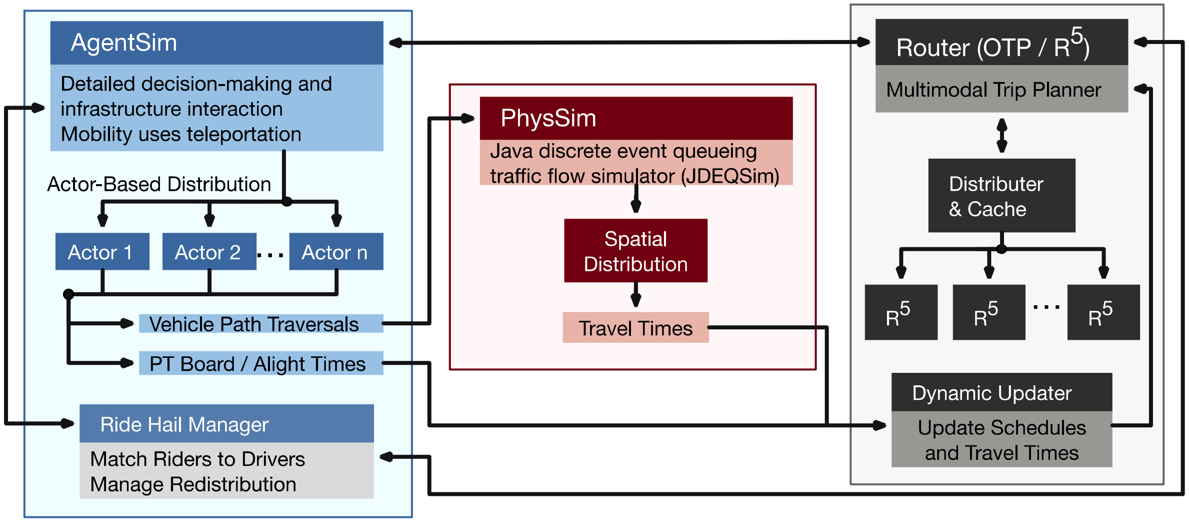
# What is BEAM?

- The Modeling Framework for **B**ehavior, **E**nergy, **A**utonomy, and **M**obility”
- Primarily a new mobsim engine for MATSim
- Introduces new approach to parallel execution to the mobsim
- Maintains as much compatibility with MATSim as possible
  - All standard MATSim events are thrown
  - Runs from MATSim inputs and configuration data along with some new inputs



# What is BEAM?

## BEAM Mobility Sim



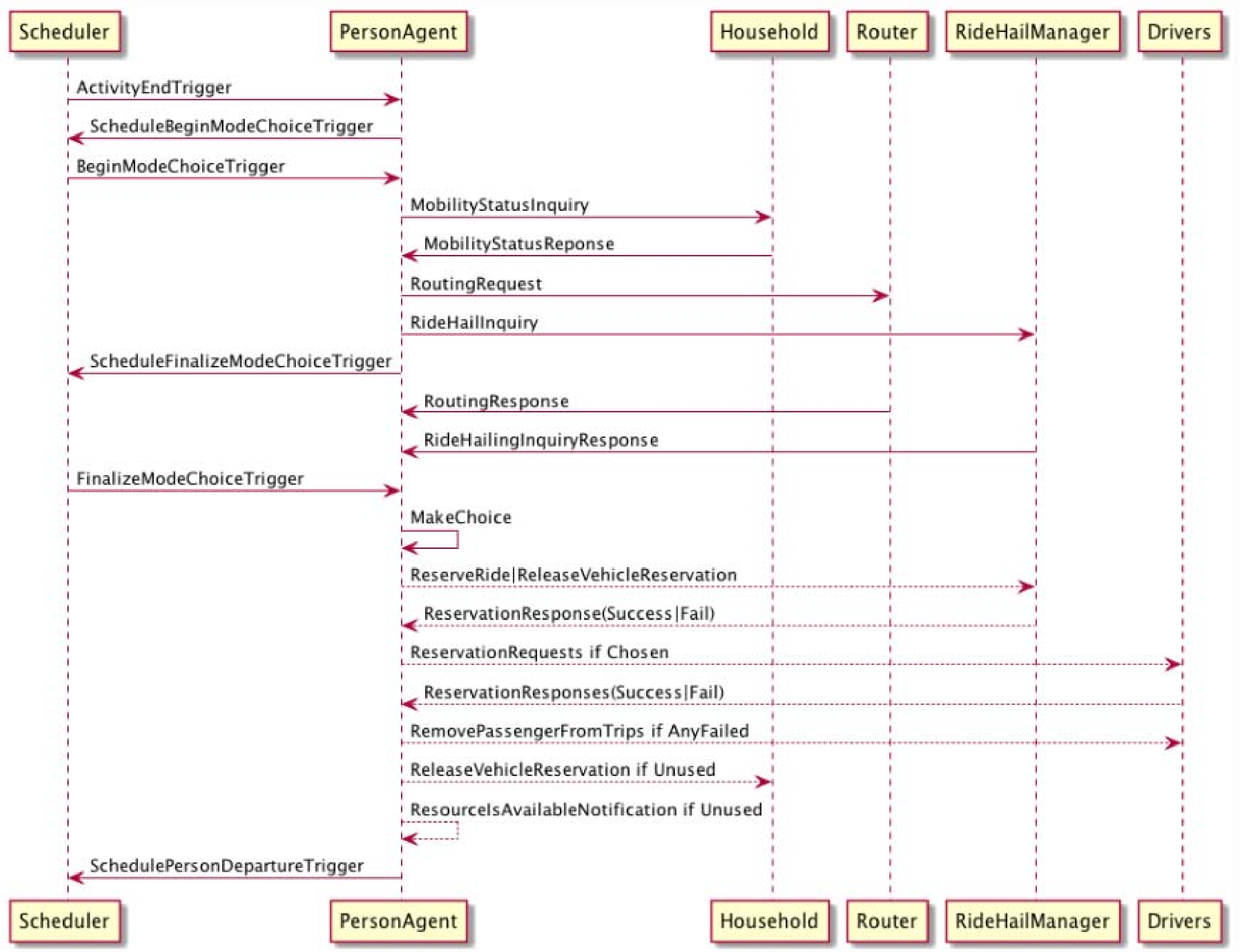
- BEAM re-envisions the MATSim Mobility Simulation
- Inspired somewhat by Pieter Fourie's work on PSim and replanning on AWS cluster and Discrete Event Queue time window, which we experimented with parallel JDEQSim in 2009.
- Decouples agent behavior & resource acquisition from traffic flow

# Ride Hailing

- Use actor model of computation instead of threads
  - No locking needed, instead use messages to communicate among actors
  - naturally fits agent modelling approach so that we can easily model agents as actors

- Goal
  - Focus on autonomous ride hail vehicle fleet
  - Rebalancing of vehicles (increase utilization of fleet)
  - Modelling surge pricing

# Chooses Mode



Scheduler, PersonAgent, Household, Router, RideHailManager, Drivers

ActivityEndTrigger

ScheduleBeginModeChoiceTrigger

BeginModeChoiceTrigger

MobilityStatusInquiry

MobilityStatusReponse

RoutingRequest

RideHailInquiry

ScheduleFinalizeModeChoiceTrigger

RoutingResponse

RideHailingInquiryResponse

FinalizeModeChoiceTrigger

MakeChoice

ReserveRide|ReleaseVehicleReservation

ReservationResponse(Success|Fail)

ReservationRequests if Chosen

ReservationResponses(Success|Fail)

RemovePassengerFromTrips if AnyFailed

ReleaseVehicleReservation if Unused

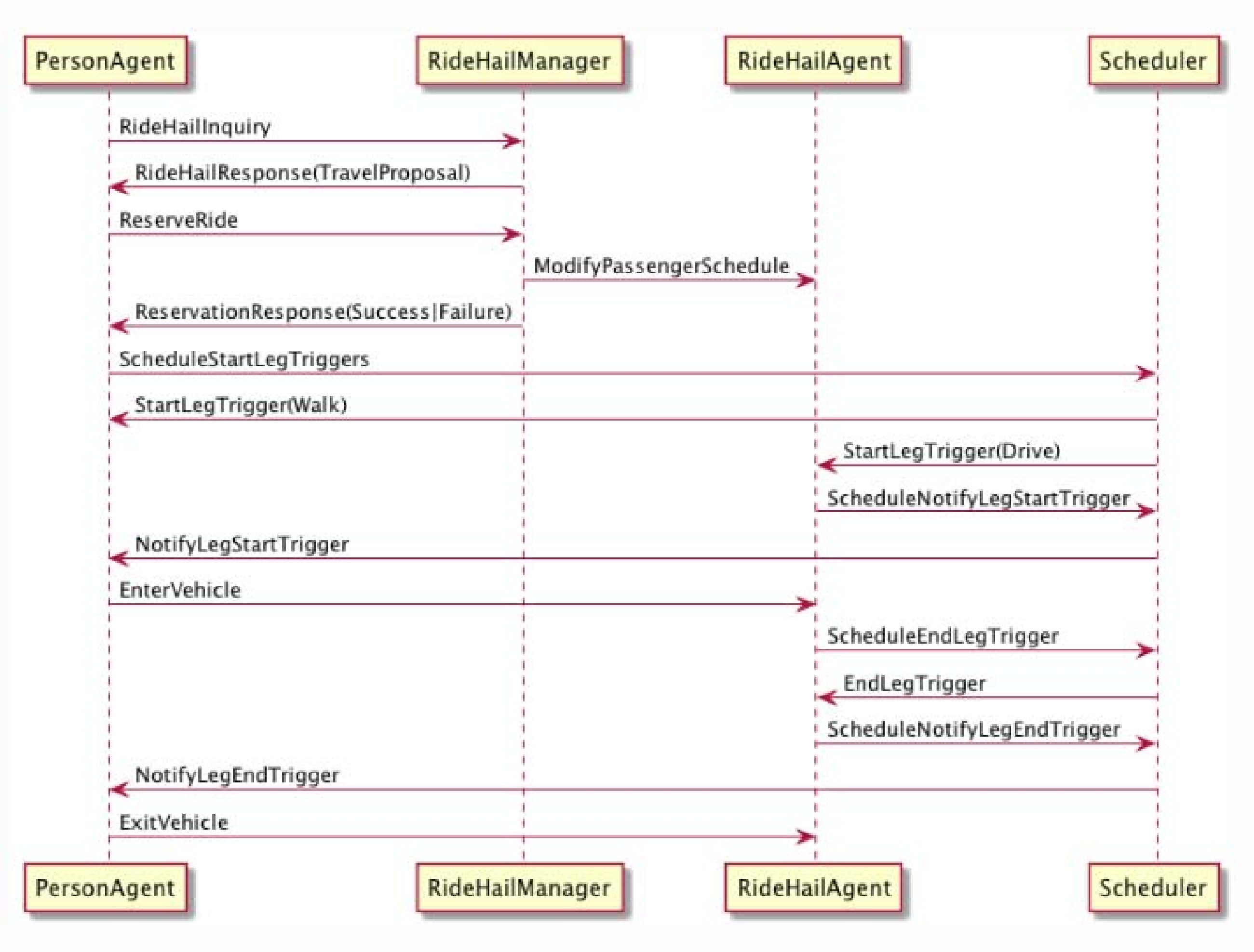
ResourceIsAvailableNotification if Unused

SchedulePersonDepartureTrigger

Scheduler, PersonAgent, Household, Router, RideHailManager, Drivers

**Modes:**
Walk
car (if vehicle available)
Bike (if vehicle available)
walk_transit
drive_transit
rideHail
ride_hail transit
etc.

# RideHailManager Communication Protocol

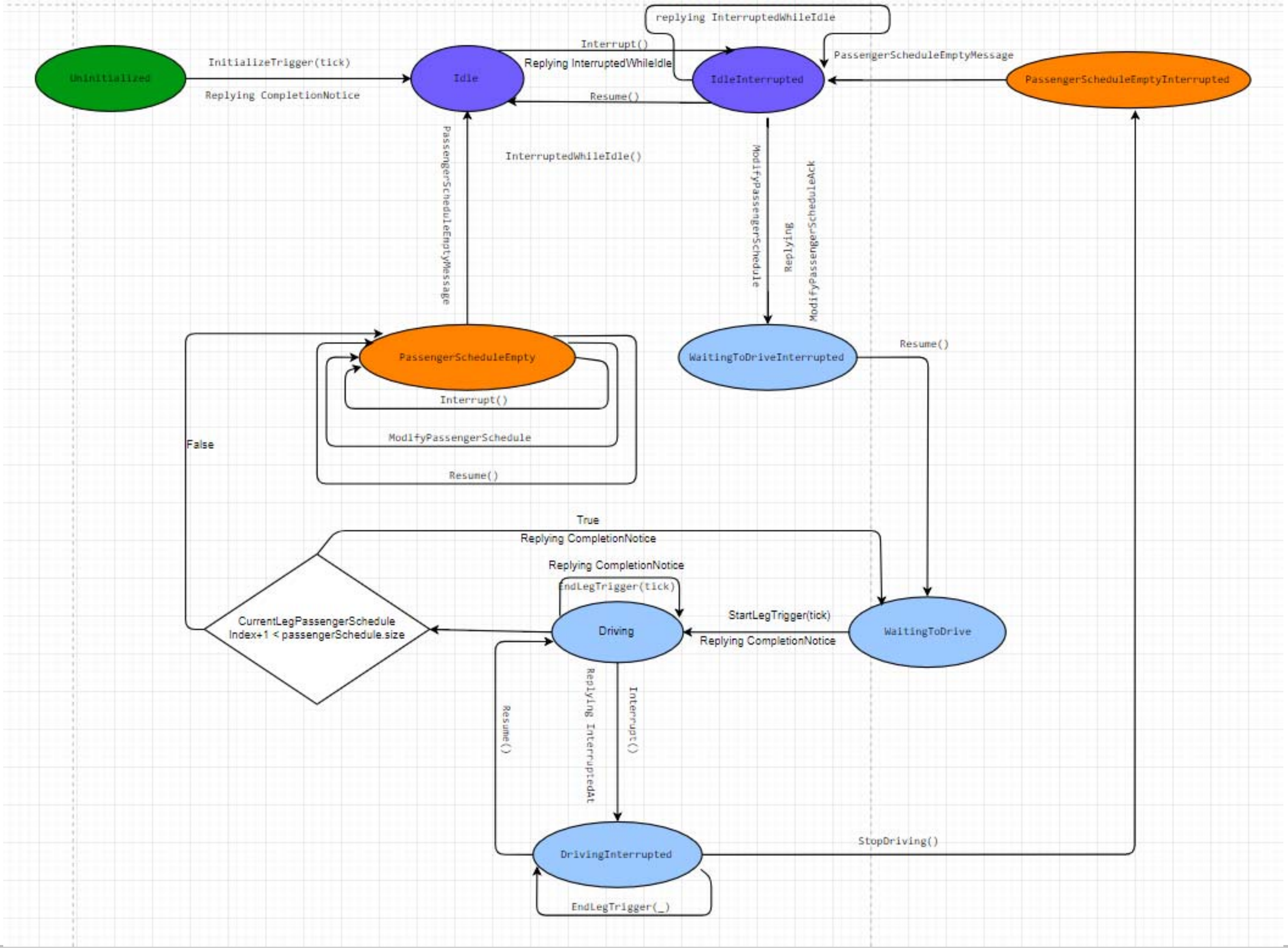


PersonAgent
RideHailManager
RideHailAgent
Scheduler

RideHailInquiry
RideHailResponse(TravelProposal)
ReserveRide
ModifyPassengerSchedule
ReservationResponse(Success|Failure)
ScheduleStartLegTriggers
StartLegTrigger(Walk)
StartLegTrigger(Drive)
ScheduleNotifyLegStartTrigger
NotifyLegStartTrigger
EnterVehicle
ScheduleEndLegTrigger
EndLegTrigger
ScheduleNotifyLegEndTrigger
NotifyLegEndTrigger
ExitVehicle

PersonAgent
RideHailManager
RideHailAgent
Scheduler

# RideHailAgent is a Finite State Machine

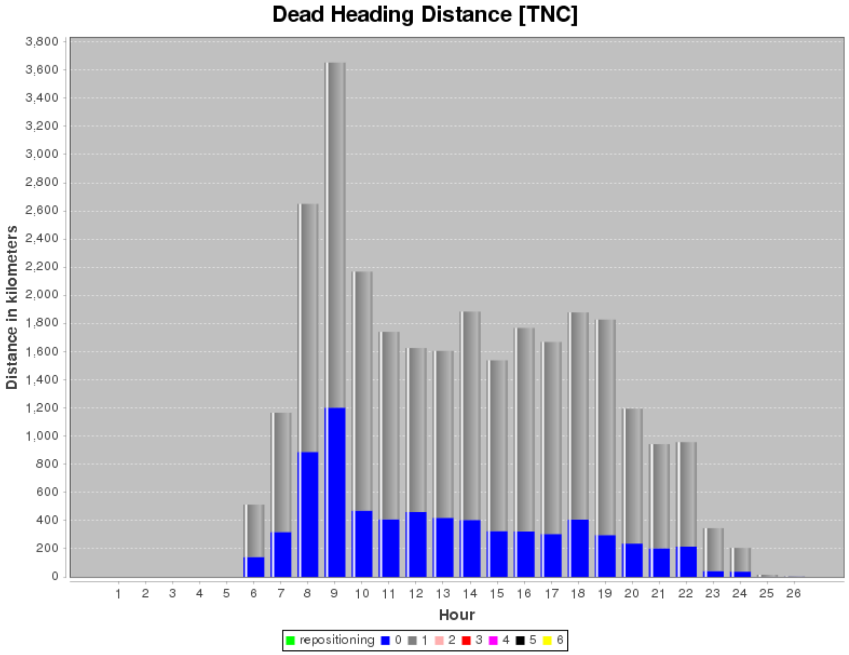

Uninitialized
InitializeTrigger(tick)
Replying CompletionNotice
Idle
Interrupt()
Replying InterruptedWhileIdle
replying InterruptedWhileIdle
IdleInterrupted
Resume()
PassengerScheduleEmptyMessage
PassengerScheduleEmptyInterrupted
PassengerScheduleEmptyMessage
InterruptedWhileIdle()
ModifyPassengerSchedule
Replying
ModifyPassengerScheduleAck
PassengerScheduleEmpty
Interrupt()
ModifyPassengerSchedule
Resume()
WaitingToDriveInterrupted
Resume()
False
True
Replying CompletionNotice
Replying CompletionNotice
EndLegTrigger(tick)
CurrentLegPassengerSchedule Index+1 < passengerSchedule.size
Driving
StartLegTrigger(tick)
Replying CompletionNotice
WaitingToDrive
Resume()
Replying InterruptedAt
Interrupt()
DrivingInterrupted
StopDriving()
EndLegTrigger(_)

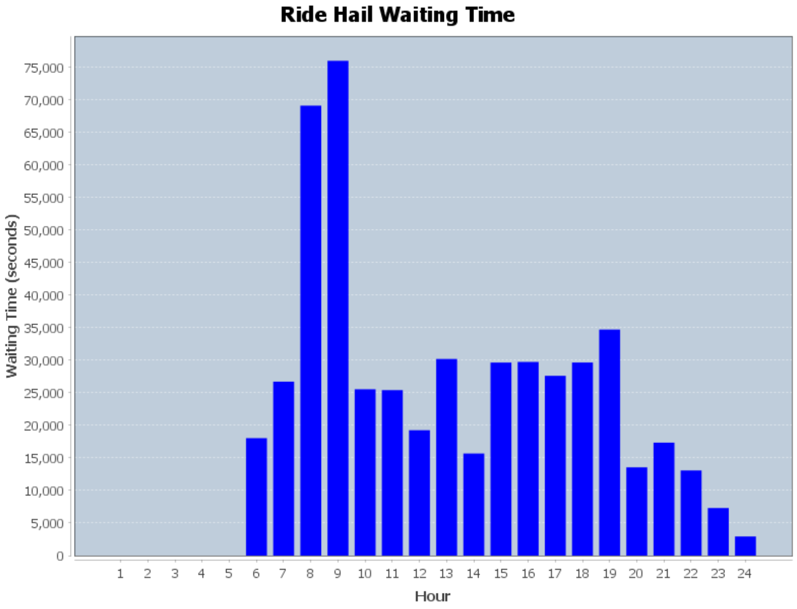
U.S. DEPARTMENT OF ENERGY SMARTMOBILITY
Systems and Modeling for Accelerated Research in Transportation

# Experiment results

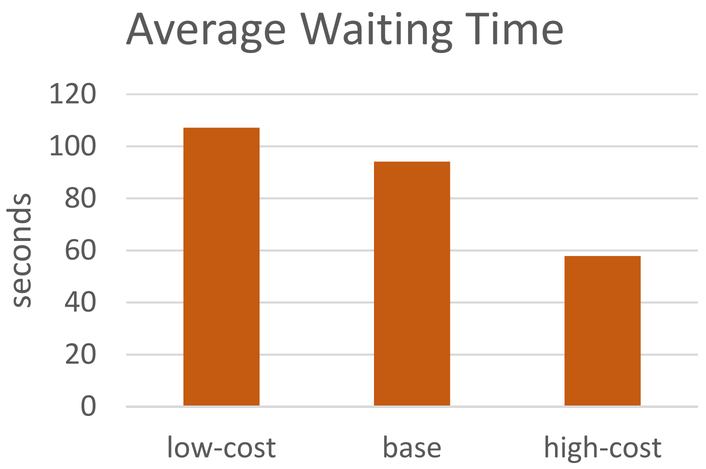
- We have various parameters
  - sf scenario (dummy scenario for testing) with 10k agents
  - Cost per mile (default: $0.75/mile)
  - Fleet size: 5% of population
  - Initial vehicle location: in 10km radius around home locations of agents

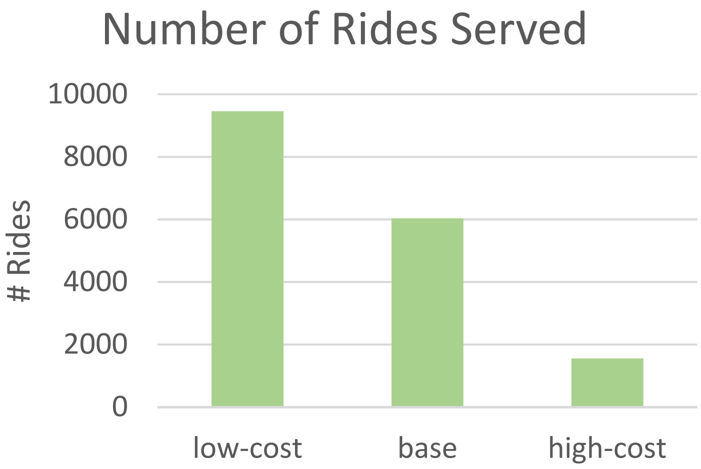
# Typical run output with default Ride hailing strategy



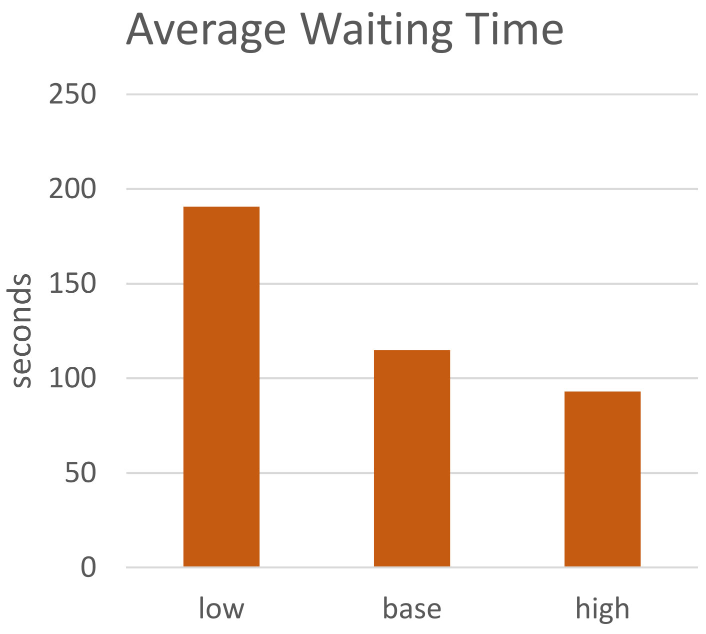


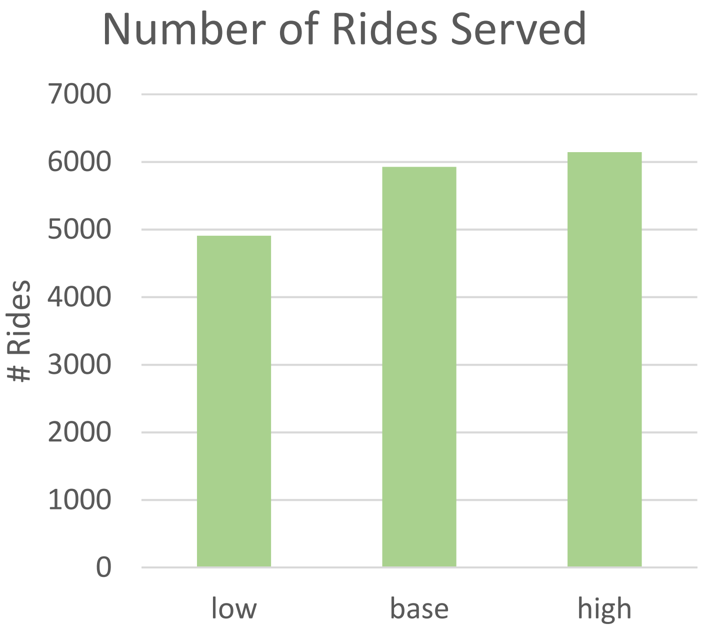
# Sensitivity: Cost (Cost Per Mile)





# Sensitivity Analysis: Number of Ride Hail Agents





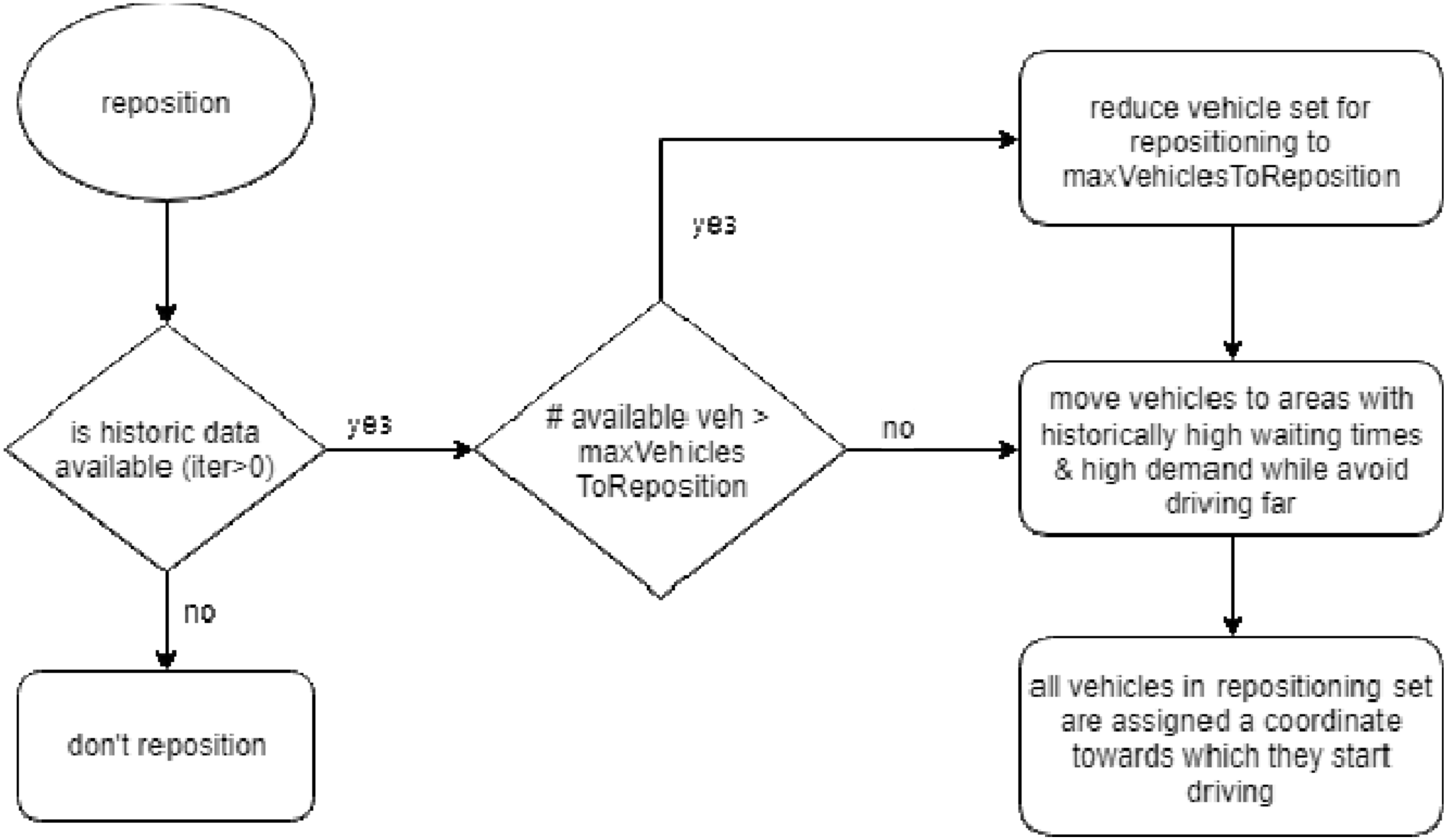
# Rebalancing/Repositioning

- How to better utilize the fleet and decrease waiting times?
- Heuristic: Move vehicles from low demand to high demand areas

- Reposition max. e.g. 1% of vehicles every 5min

- Important to note: Vehicles which are being repositioned can still be reserved at any time

# Rebalancing Algorithm: Overview


reposition
is historic data available (iter>0)
yes
no
don't reposition
# available veh > maxVehicles ToReposition
yes
no
reduce vehicle set for repositioning to maxVehiclesToReposition
move vehicles to areas with historically high waiting times & high demand while avoid driving far
all vehicles in repositioning set are assigned a coordinate towards which they start driving

# Rebalancing Algorithm: Historic Data we Collect

- Collecting in each iteration Traffic Analysis Zone (TAZ) level data for defined time interval, e.g. 30min:
  1. number of ride hail requests
  2. number of idling vehicles
  3. sum of waiting times
  4. number of ending activities

Avoid oscillations: use average of last 2 iterations

# Selecting Idling Vehicles for Repositioning

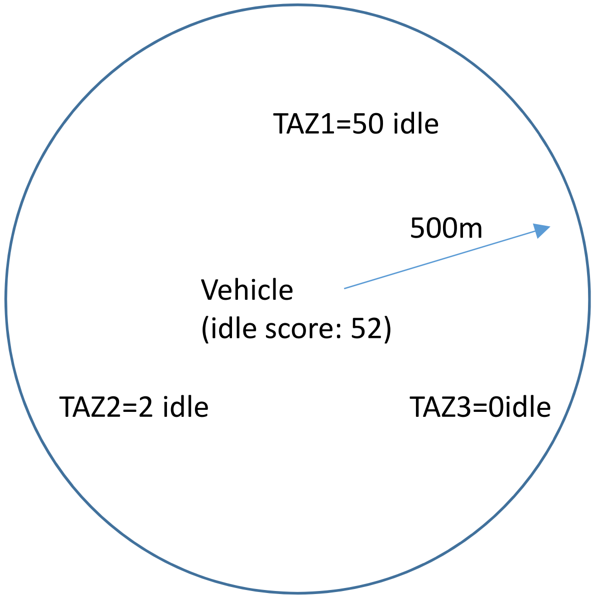
1.) Prioritize vehicles which are not currently repositioning

2.) Identify good candidates for repositioning based on number of idle vehicle historically in area

-> for each TAZ in 500 meters from a vehicle, we find out how many idle vehicles it had within the next x minutes (e.g. 20min)

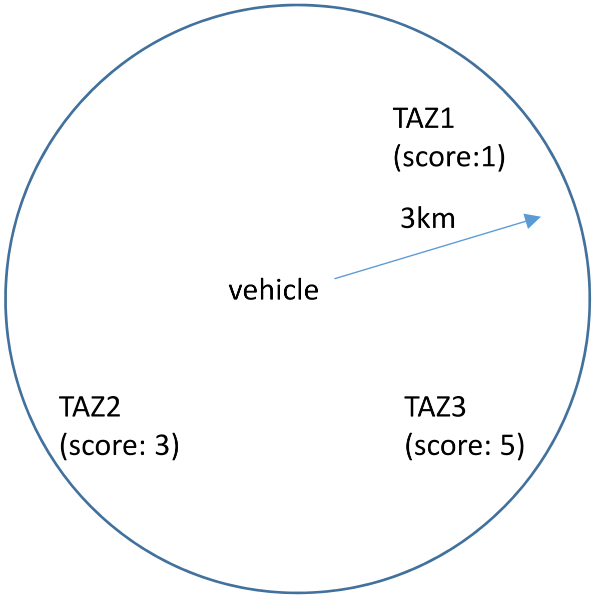
-> convert idle score per vehicle into selection probability of vehicle for repositioning

-> sample maxNumberOfVehiclesToReposition vehicles



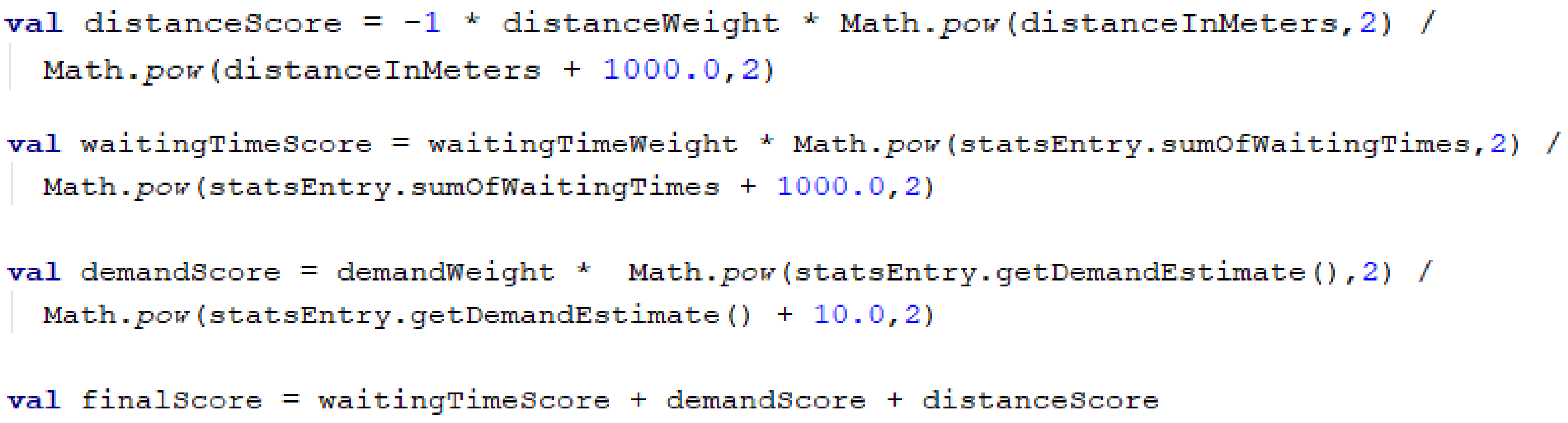
# Where to Move Selected Vehicles for Repositioning?

- Estimate historic demand: historic ride hail demand and activity combined
  - Activities contain potential customers
  - Past ride demand tells us of past rideHail customers
- For each vehicle we have selected for repositioning, we try to find a TAZ in radius r (e.g. 3km), which is best to move to in terms of:
  - Score=f(WaitingTime, demandEstimate, distanceToVehicle) – look again into future (e.g. 20min)
  - Convert score into probability and assign TAZ to move



# Scoring Function for Repositioning

- Scored function used:



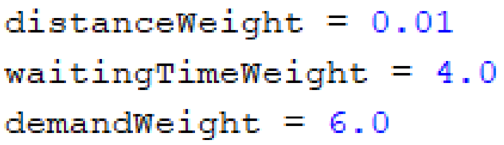
```
val distanceScore = -1 * distanceWeight * Math.pow(distanceInMeters,2) /
  Math.pow(distanceInMeters + 1000.0,2)

val waitingTimeScore = waitingTimeWeight * Math.pow(statsEntry.sumOfWaitingTimes,2) /
  Math.pow(statsEntry.sumOfWaitingTimes + 1000.0,2)

val demandScore = demandWeight *  Math.pow(statsEntry.getDemandEstimate(),2) /
  Math.pow(statsEntry.getDemandEstimate() + 10.0,2)

val finalScore = waitingTimeScore + demandScore + distanceScore
```

with



```
distanceWeight = 0.01
waitingTimeWeight = 4.0
demandWeight = 6.0
```

# Ride Hail Rebalancing

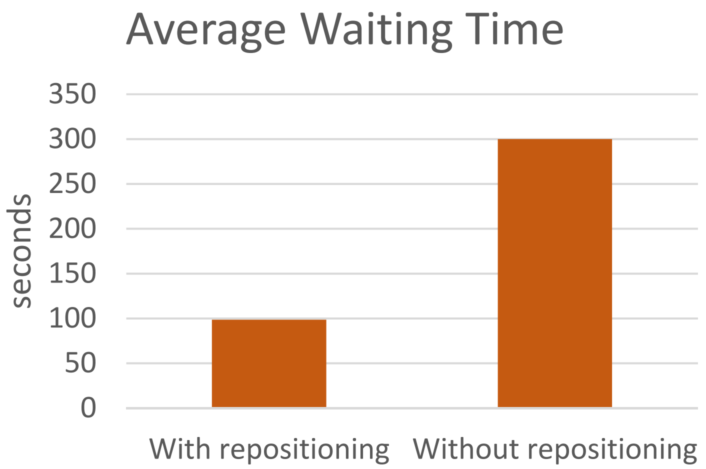
- How to allow exploration?
  - E.g. if we don't have any areas with good scores? Allow expanding circle so at least a certain demand percentage covered (+ max. radius)

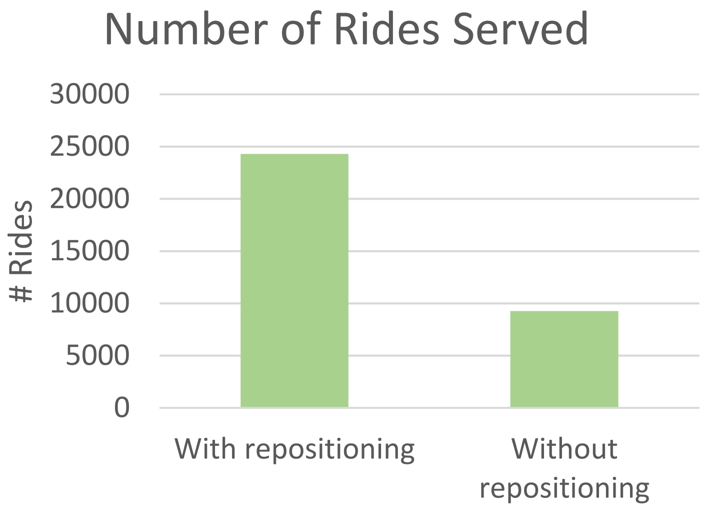
- More aggressive: keep top N scores

- real implementation: group vehicles by TAZ to avoid calculating same multiple times.

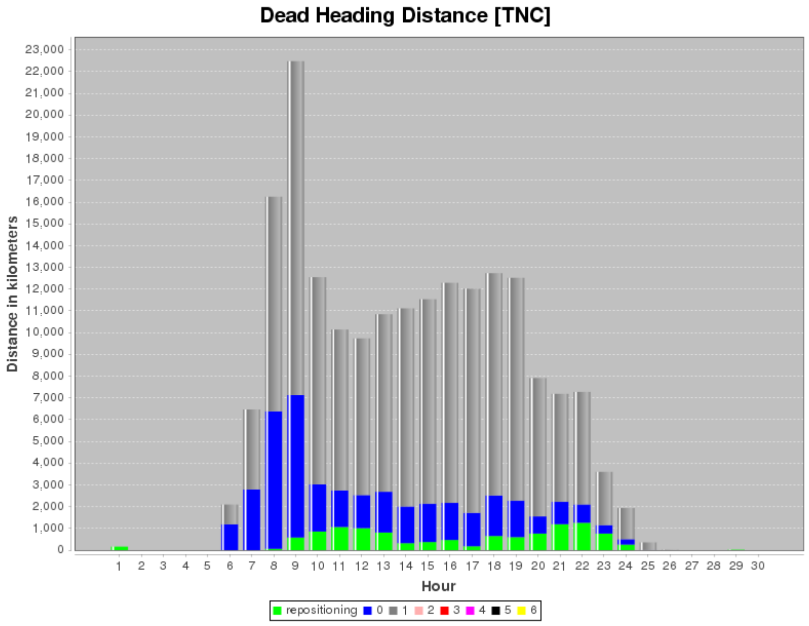
- Experiments
  - Initial location: All vehicles at one corner of activity plane

# With Repositioning Results

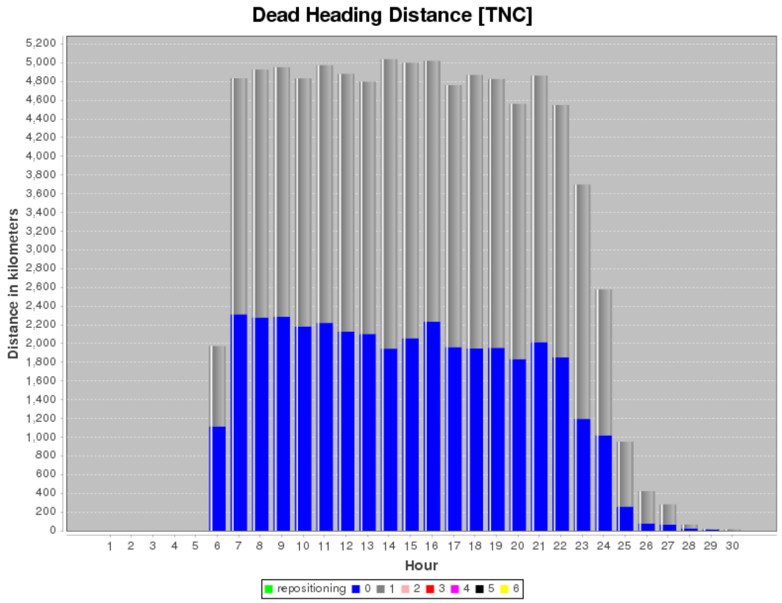




# Can we improve deadheading?



Dead Heading Distance [TNC]



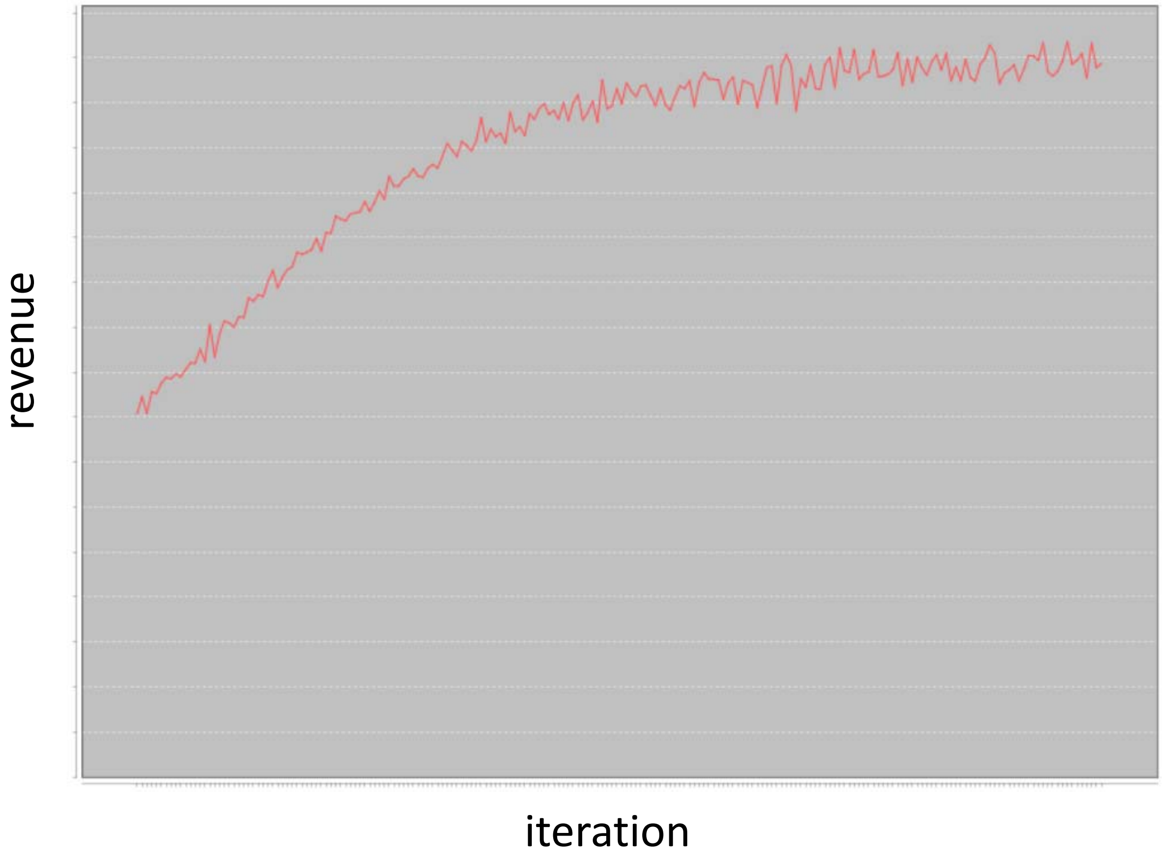
Dead Heading Distance [TNC]

# Surge Pricing

- Purpose
  - Bring in new drivers onto road
  - Reduce demand
- We have modelled a one sided one where supply is fixed autonomous fleet, but by changing price we reduce demand
- Surge pricing algorithm
  - Maintain for each TAZ and time interval info (e.g. hourly) about revenue and price level
  - Initial iteration start at price level 1.0
  - Increase or decrease price randomly in iteration 1 by 0.1
  - If revenue increased for TAZ and timeBin, keeping changing price level in same direction by 0.1, otherwise opposite direction
  - Possible to provide minimum price level

# Surge Pricing

- Ride Hailing Revenue: 200 Iterations



# Future Work

- Implement multiple passenger pickup (pooling)
- Human drivers entering/exiting market

# QUESTIONS?