Redstone Project High-Level:

Satellite Network Routing Algorithm Design using MDP (Markov Decision Process)

Hongseok Kim
Undergraduate Student in
UT Austin Aerospace Engineering

Importance of this Research

Limitations of previous methods:

- Algorithms like Dijkstra's worked effectively for stationary networks but failed in satellite constellations due to their rapidly changing network structure.
- Traditional algorithms couldn't track or adapt to the dynamic nature of satellite networks, making them less suitable for this application.

New capabilities:

- There exist multiple researches which try to apply MDP to satellite network, but they didn't show explicit framework of flowchart from scratch.
- This project has developed a MATLAB library with MDP algorithms and satellite network configurations, allowing for customization and adaptation to different constellation concepts.

Benefits:

- The availability of the MATLAB library on GitHub enables researchers to modify and tailor the basic code for a variety of satellite constellation structures.

Research Configuration

Problem Formulation	Theoretical Background	Methodology	Simulation Setup	Result and Discussion
Satellite Network Configuration	MDP (Markov Decision Process) • Policy Iteration	Using MATLAB satellite communication toolbox framework	2 orbits with RAAN difference (15 ~ 90), 24 SATs for each	• **Simulation 1 Result** • Orbit Visualization • Network Graph
Routing Algorithm for Single Packet		Routing Path Generation using MDP + Backward Induction	Configuration of MDP with given reward Structure and **Simulation 1**	• State value change over time by given MDP structure • Cumulative Reward, Final Reward, State/Action Value
Collision Avoidance Algorithm Design for Multiple Packets	Backward Induction • Dynamic Programming • Value iteration	Modifying Routing Algorithm • Sequential Method • Cooperative Method • Congestion Penalty	**Simulation 2 & 3** • 20 Packets, 4 Destinations • 100 Packets, 5 destinations	• **Simulation 2**: Compare performance of 3 methods • **Simulation 3**: Performance of Penalty Method • Computational Complexity

Problem Formulation

Satellite Network Configuration

- Define optimal orbital parameters and number of satellites to ensure continuous communication.
- Define interconnectivity among satellites for data relay and transmission
- Create a network architecture that adapts to satellite positions and contact conditions

Routing Algorithm for Single Packet

- Determine the optimal pathway for data transmission considering starting satellite, starting time and destination
- Account for the dynamic nature of satellite positions and shifting network topologies
- Develop a solution that allows a single data packet to constituently identify an optimal pathway in real-time

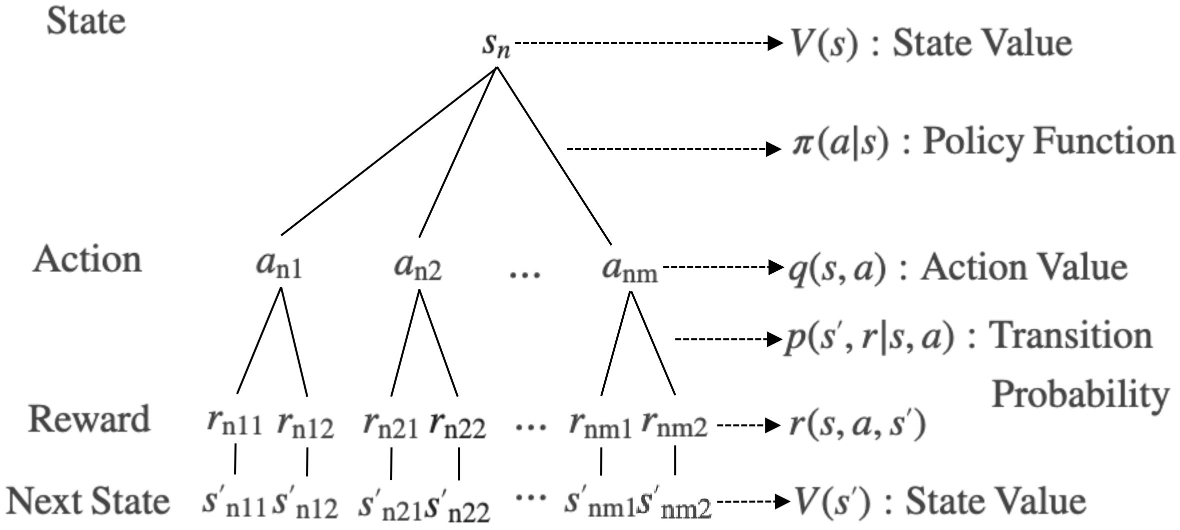
Collision Avoidance Algorithm for Multiple Packets

- Address challenges of routing multiple data packets simultaneously within the same network structure
- Minimize the risk of communication bottlenecks by preventing packet convergence on the same satellite
- Design cooperative or sequential routing strategies to avoid collisions and ensure efficient transmission

Theoretical Background [1/3]

Markov Decision Process

$M = (S, A, T, R)$



Known, Input Value
- State
- Action
- Transition probability
- Reward

Unknown, Output Value
- State Value
- Action Value
- Policy Function

Theoretical Background [2/3]

Policy Iteration Process in MDP

Policy Iteration (Using iterative policy evaluation)

1. Initialization

$V(s) \in R$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$

2. Policy Evaluation

Repeat

$\Delta \leftarrow 0$

For each $s \in S$

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

3. Policy Improvement

$policy - stable \leftarrow$ true

For each $s \in S$

updated $\leftarrow \pi(s)$

$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

If $a \neq \pi(s)$, then $policy - stable \leftarrow$ false

If policy $-$ stable, then stop and return V and π;

elso go to 2

Reference: Sutton, Richard S., and Andrew G. Barto. "Reinforcement learning: an introduction, 2nd edn. Adaptive computation and machine learning." (2018).

Initialize a policy

- Start with an arbitrary policy (a mapping from states to actions).

Policy Evaluation:

- Evaluate the value function for the current policy by iteratively solving the Bellman equation.
- Continue until the value function stabilizes (converges), indicating that it accurately reflects the long-term rewards under the policy.

Policy Improvement:

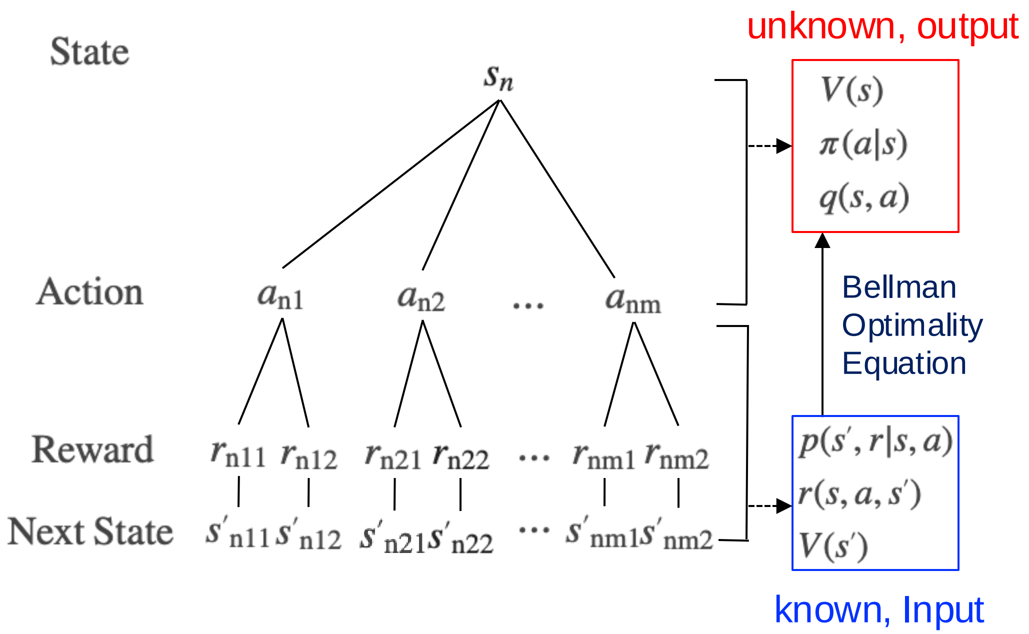
- Use the current value function to improve the policy.
- For each state, choose the action that maximizes the expected value based on the current value function.

Repeat

- Alternate between policy evaluation and policy improvement until the policy converges (i.e., no further improvements can be made).
- Once the policy stops changing, the optimal policy has been found.

Theoretical Background [3/3]

Backward Induction in MDP



State Value : $V(s) = \max_a q(s,a)$

Policy Function : $\pi(s|a) = \arg\max_a q(s,a,t)$

Action Value : $q(s,a) = \sum_{s',r} p(s',r|s,a)[r(s,a,s') + \gamma V(s')]$

Key Idea

- Assume we know the state value function of certain timestep
- Then we can know the State Value, Policy Function and Action Value in previous timestep by using Bellman Optimality Equation

Methodology [1/4]

Satellite Network Configuration

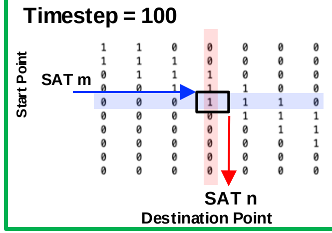
Satellites Configuration

Orbit Parameter Design
Constellation Design

Ground Station Config

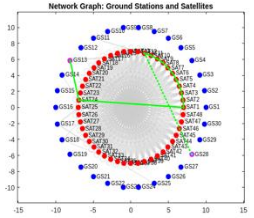
Lat/Long/Altitude Position
Elevation Angle Condition

Network Access Availability Using Contact Matrix



- 1 at [m,n]: Data can transit SAT/GS m to SAT/GS n
- row m = 0: Cannot Escape from SAT m
- column n = 0: Cannot approach to SAT n

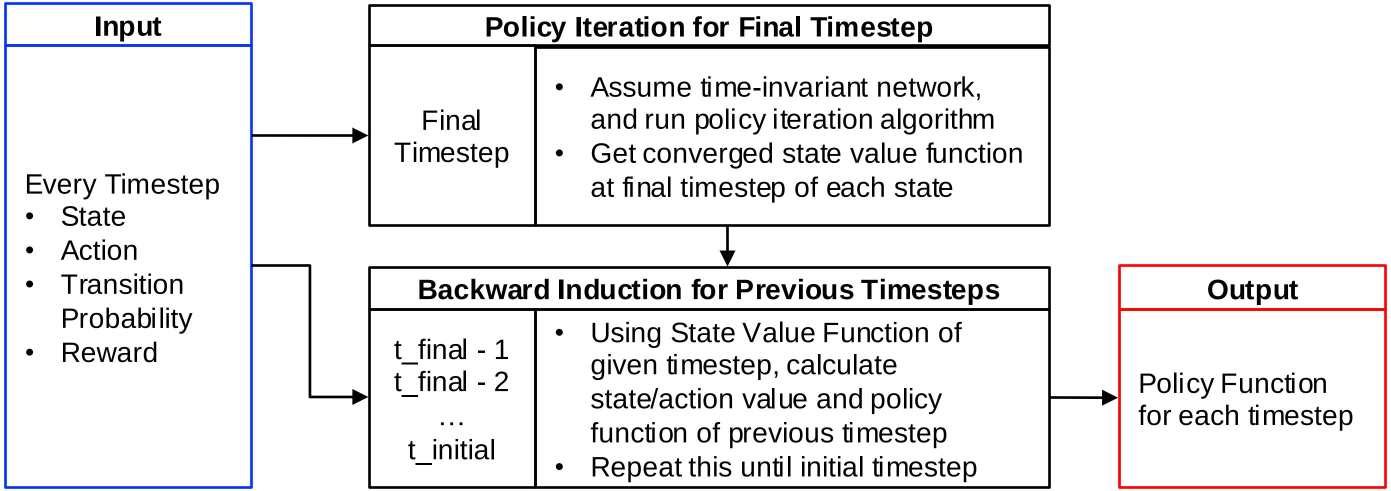
Network Graph



- Using Satellite Communication toolbox in MATLAB, we can generate access information between satellites and GS based on visibility between two objects.
- Using this information, we made contact matrix compose of 0 and 1, and network graph

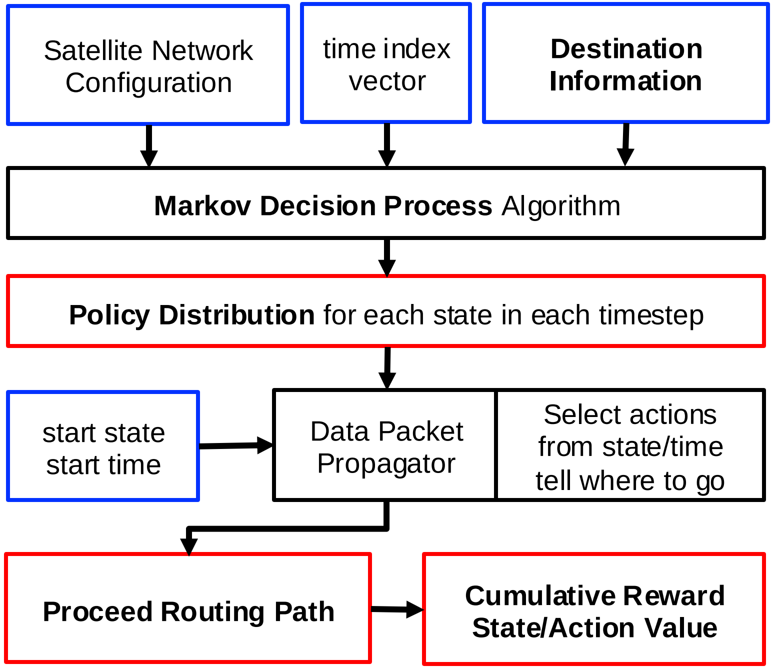
Methodology [2/4]

Routing Algorithm for single packet: Combining Policy Iteration and Backward Induction



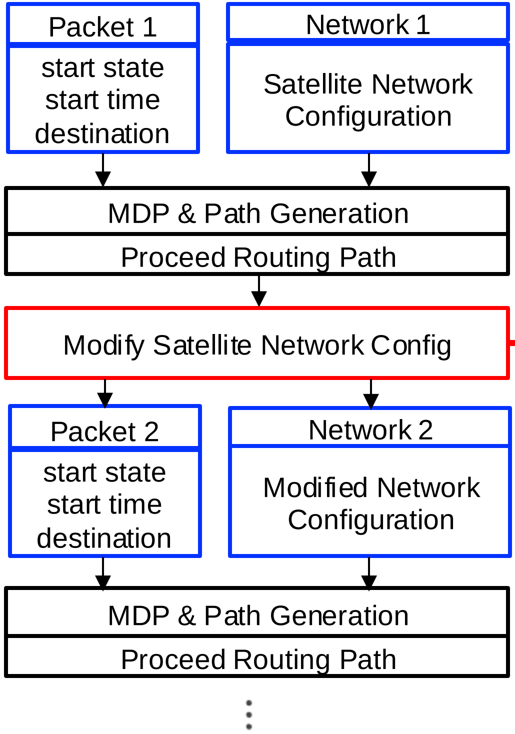
Methodology [3/4]

Routing Algorithm for single packet: Data Packet Routing Algorithm

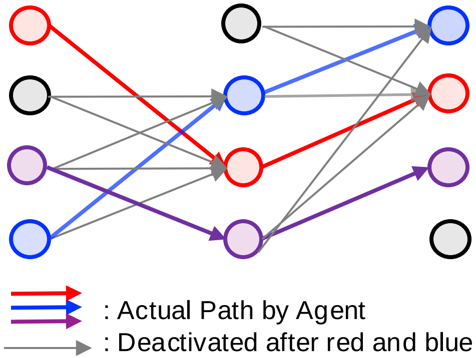


Methodology [4/5]

Collision Avoidance Algorithm for multiple packets: Sequential Approach



Sequence: > >



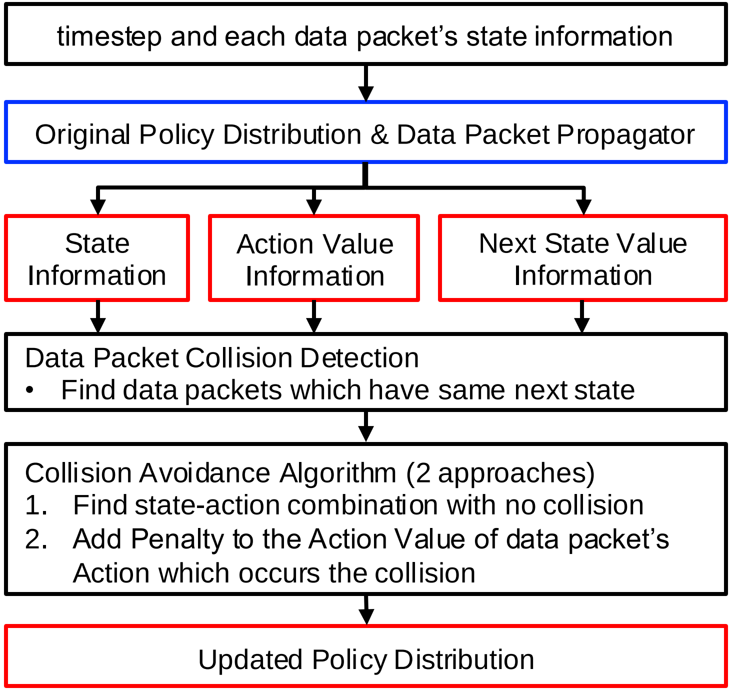

```
sat_to_sat(:,state_index,time_index-1) = 0;
```

To avoid collision, after each agent's pathway propagation, deactivate every edge approaching to the node in which the agent stayed at given time

Methodology [5/5]

Collision Avoidance Algorithm for multiple packets

Cooperative and Penalty Approach



Cooperative Approach

parse vector of active states $S_{t_i} = \{s_1, s_2, s_3, s_4\}$ at t_i

parse vector of available **actions** $a_i = \{a_{j1}, a_{j2}, a_{j3}, a_{j4}\}$

parse corresponding **action values** $q_i = \{q_{j1}, q_{j2}, q_{j3}, q_{j4}\}$

create set of action – action value vector from given state

$\Rightarrow$ in state $s_j : \{(a_{j1}, q_{j1}), (a_{j2}, q_{j2}), (a_{j3}, q_{j3}), (a_{j4}, q_{j4})\}$

create matrix of states including these informations

$$\Rightarrow \begin{bmatrix} s_1 & s_2 & s_3 & s_4 \\ (a_{11}, q_{11}) & (a_{21}, q_{21}) & (a_{31}, q_{31}) & (a_{41}, q_{41}) \\ (a_{12}, q_{12}) & (a_{22}, q_{22}) & (a_{32}, q_{32}) & (a_{42}, q_{42}) \\ (a_{13}, q_{13}) & (a_{22}, q_{22}) & (a_{33}, q_{33}) & (a_{43}, q_{43}) \\ (a_{14}, q_{14}) & (a_{24}, q_{24}) & (a_{34}, q_{34}) & (a_{44}, q_{44}) \end{bmatrix}$$

Caution : number of actions of each state are different

parse 1 action value from each active state

$$\Rightarrow \begin{bmatrix} & s_1 & s_2 & s_3 & s_4 & \Sigma q \\ \text{case 1} & q_{13} & q_{24} & q_{32} & q_{41} & Q_1 \\ \text{case 2} & q_{14} & q_{21} & q_{33} & q_{42} & Q_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \text{case256} & q_{12} & q_{23} & q_{34} & q_{43} & Q_{256} \end{bmatrix} (Q_1 \geq Q_2 \geq \ldots Q_{256})$$

for case 1 : 256

if case k does not have collision

take case k : $\{(a_{13}, q_{13}), (a_{21}, q_{21}), (a_{34}, q_{34}), (a_{41}, q_{41})\}$

$\Rightarrow$ collision resolved

Penalty Approach

create dataset of current states including these informations

$$\text{SAS}_t = \begin{bmatrix} s_1 - \pi_1 & s_2 - \pi_2 & s_3 - \pi_3 & s_4 - \pi_4 \\ (a_{11}, q_{11}, s'_{11}) & (a_{21}, q_{21}, s'_{21}) & (a_{31}, q_{31}, s'_{31}) & (a_{41}, q_{41}, s'_{41}) \\ (a_{12}, q_{12}, s'_{12}) & (a_{22}, q_{22}, s'_{22}) & (a_{32}, q_{32}, s'_{32}) & (a_{42}, q_{42}, s'_{42}) \\ (a_{13}, q_{13}, s'_{13}) & (a_{22}, q_{22}, s'_{23}) & (a_{33}, q_{33}, s'_{33}) & (a_{43}, q_{43}, s'_{43}) \\ (a_{14}, q_{14}, s'_{14}) & (a_{24}, q_{24}, s'_{24}) & (a_{34}, q_{34}, s'_{34}) & (a_{44}, q_{44}, s'_{44}) \end{bmatrix}$$

while true (infinite loop)

parse current policy distribution for each active state

policy $= [\pi_1(a_{11}|s_1) = 1 \quad \pi_2(a_{21}|s_2) = 1 \quad \pi_3(a_{31}|s_3) = 1 \quad \pi_4(a_{41}|s_4) = 1]$

caution : number of actions of each state are different

create dataset of next state **applying current policy** π for each state

$\rightarrow$ Newly Generated States may be included in this case

$$\text{SAS}_{t+1} = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 \\ s'_{11} & s'_{21} & s'_{31} & s'_{41} & s'_{51} \end{bmatrix} (s_5 \text{ is newly generated packet at } t+1)$$

for active states $S_{t_i} = \{s_1, s_2, s_3, s_4\}$

find number of colliding states for all allction for current state

$s'_{j1} \cdots s'_{j4} \in \{s'_{21}, s'_{31}, s'_{41}, s'_{51}\}$

update action values for each action by applying penalty from collision

if s'_{j1} has k collisions $\Rightarrow q'_{j1} = q_{j1} - k\alpha$ (α : Collision Factor)

update policy by finding maximum action values updated

if $\max\{q'_{j1}, q'_{j2}, q'_{j3}, q'_{j4}\} = q'_{j4} \Rightarrow \pi'_j(a'_{j4}|s) = 1$

create policy_updated $= [\pi'_1 \quad \pi'_2 \quad \pi'_3 \quad \pi'_4]$

if policy_updated == policy

then **break** $\rightarrow$ policy_updated is equilibrium policy for all active agents

else policy = policy_updated $\rightarrow$ **return** to while loop

Simulation Setup [1/3]

Satellite Network Configuration

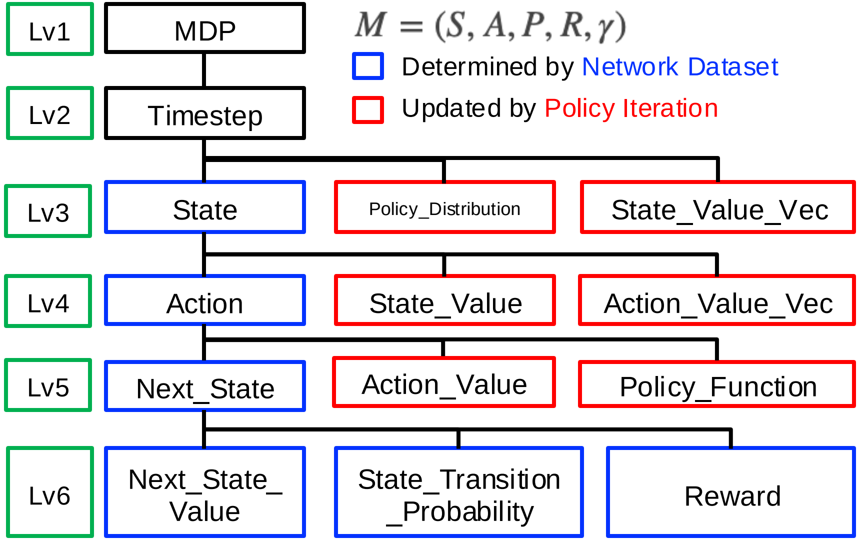
1. Two Orbits design: Inclination = 98deg
2. 15deg, 45deg, 60deg,90deg RAAN difference
3. Each orbit: 24 satellites
4. Simulation duration: 1hr
5. Simulation Timestep: 15 seconds -> 400 timesteps
6. Orbit Propagator: SGP4
7. 30 ground station points

Two LEO orbits Constellation Design	
Epoch	March 21, 2024, 00:00:00
Orbit Altitude	500km
Eccentricity	1e-5 (Circular)
Inclination Ang.	97.4022 deg
RAAN	RAAN1: 90deg, RAAN2 : [105, 135, 150, 180]deg
AOP	0 deg
True Anomaly	24 Satellites on each orbit (Total 48) True Anomaly [0,15,30, … , 345]
Orbit Propagator	SGP4
Duration	1 hour, 15 seconds timestep (240 timesteps)
Ground Station	30 ground stations around the globe

Simulation Setup [2/3]

Routing Algorithm for single packet: Data Packet Routing Algorithm

MDP Structure



Transition Probability / Reward Structure

Transition Probability	Simulation 1,3	0.8
	Simulation 2	1
Reward Structure	Transition in Same Orbit	-1
	Transition in Different Orbit	-15
	Staying in Current State	-5
	Failure of Transition	-30
	Approach to Destination in Same Orbit	+100
	Approach to Destination in Different Orbit	+75

Simulation Setup [3/3]

Collision Avoidance Algorithm for multiple packets

	Simulation 1	Simulation 2	Simulation 3
ose	Visualize Orbit and state value change Check routing duration	Compare 3 collision avoidance algorithms	Simulate Large-scale data routing using penalty method
ber of Packets	48	20	100
State	Determined [1,2,3, ..., 48]	Random [1 - 48]	Random [1 - 48]
Timestep	Same timestep: 100	Random [100 – 120]	Random [100 – 120]
lation Horizon	Timestep 100 – 130 (total 450 seconds)		
ber of nations	1	4	5
ithm Used	Default (Data Packets don't influence each other)	Default Sequential Cooperative Penalty	Default Penalty

	Simulation 1	Simulation
Purpose	Visualize Orbit and state value change and check routing duration	Compare collision avoida algorithms
Number of Packets	48	20
Transition Probability	Success: 0.8 Fail: 0.2	Success: 1 Fail: 0
Start State	Determined [1,2, ..., 48]	Random fro - 48]
Start Timestep	Same timestep: 100	Random [10 120]
Simulation Horizon	Timestep 100 – 130 (tota	
Number of Destinations	1	4
Algorithm Used	Default - Packets don't influence each other	Default / Sequential/ Cooperative Penalty

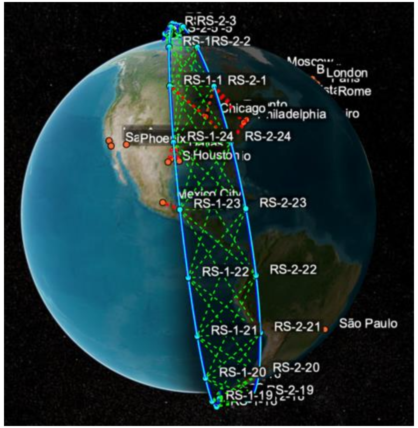
Result [1/3]

Satellite Network Configuration

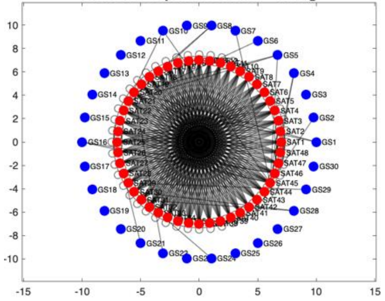
- **Result 1**: Orbit visualization and network graph for different RAAN difference
- Purpose for visualization: Visualize the unique characteristics of satellite constellation – the network structure changes rapidly by changing connectivity

Result [1/3]

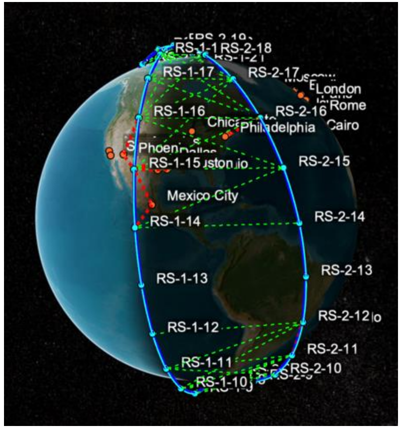
RAAN diff: 15 deg



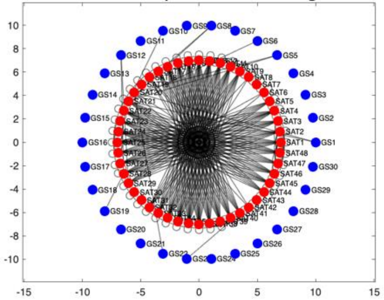
Network Graph: RAAN diff. = 15deg



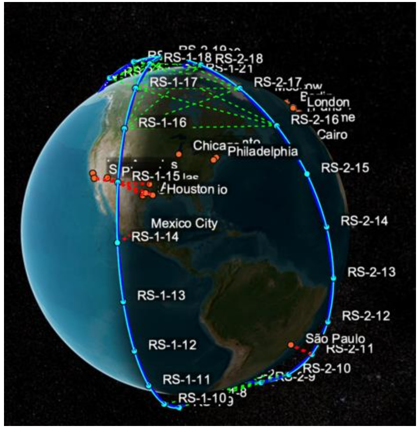
RAAN diff: 45 deg



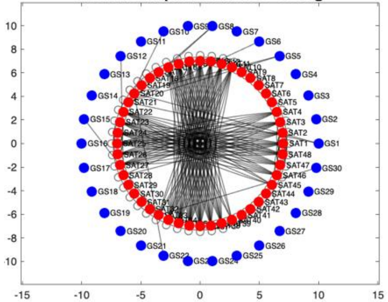
Network Graph: RAAN diff. = 45deg



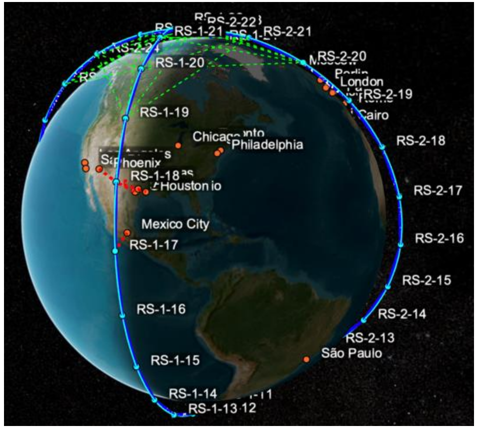
RAAN diff: 60 deg



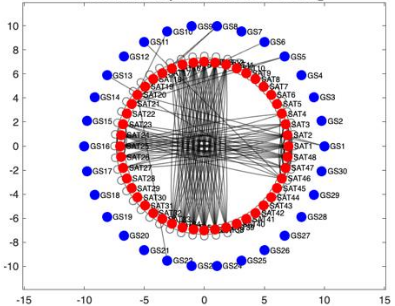
Network Graph: RAAN diff. = 60deg



RAAN diff: 90 deg

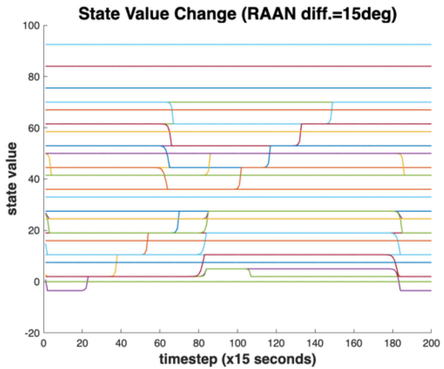


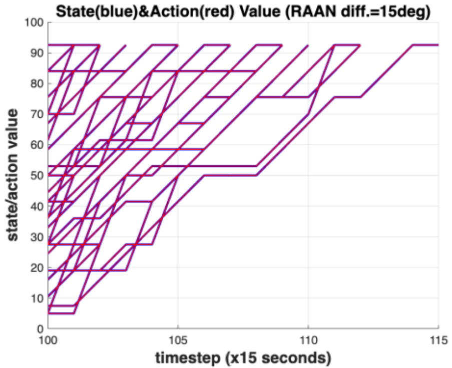
Network Graph: RAAN diff. = 90deg



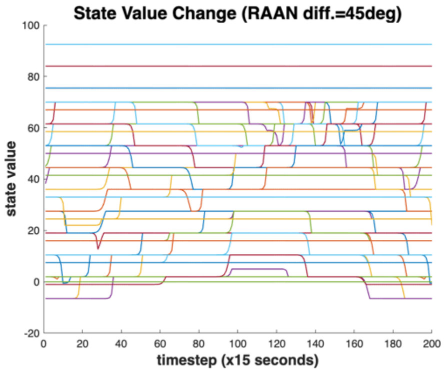
Result [2/3]

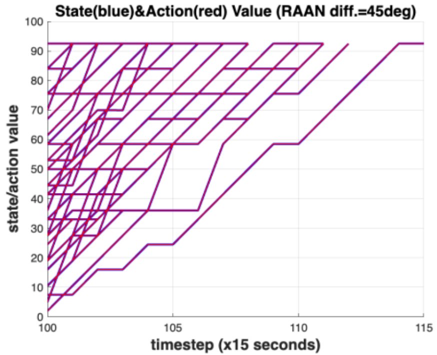
RAAN diff: 15 deg



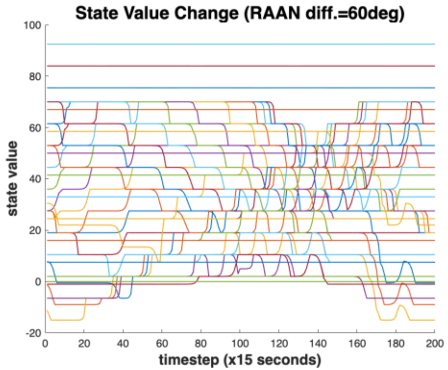


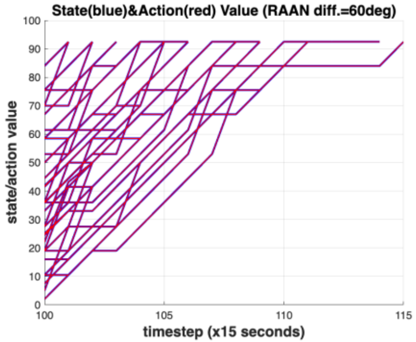
RAAN diff: 45 deg



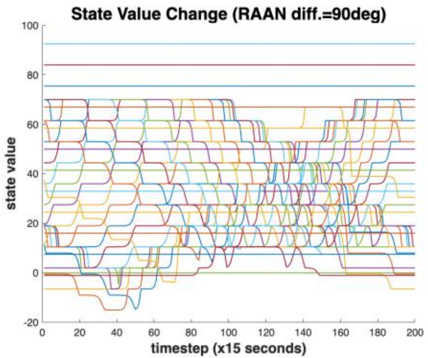


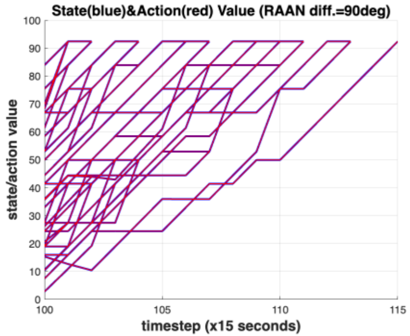
RAAN diff: 60 deg





RAAN diff: 90 deg





Result [2/3]

Routing Algorithm for single packet: Data Packet Routing Algorithm

Destination = 40

A. **Result 2: Default graph – No collision avoidance algorithm**
   i. Graph of state-action value for each agent
   ii. Show that even network change rapidly most of data reaches the destination (Simulating 48 packets, 1 destination)
B. **Result 3: State value change over time**
   i. Given destination, Run MDP in given timestep vector
   ii. Visualize the change of state value by changing RAAN difference
   iii. Show that increasing RAAN difference makes rapid change of state value, which indicates rapid network condition change

Result [3/3]

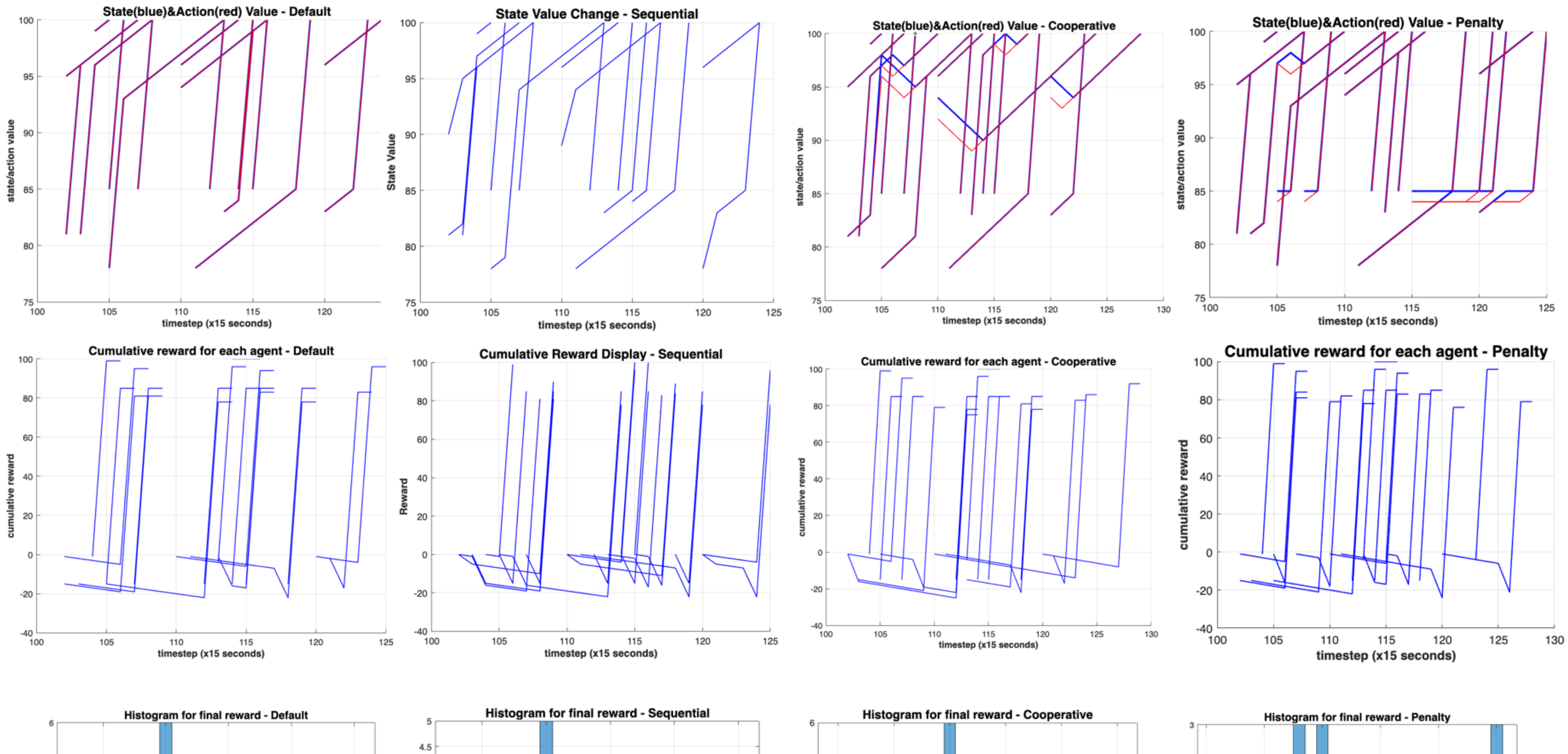
Collision Avoidance Algorithm for multiple packets

A. **Result 4:** Default, Sequential, Cooperative and Penalty comparison
   i. To minimalize time-variant effect, take 15-degree RAAN difference
   ii. Same simulation input: 20 agents, 4 destinations
      1. Cumulative Reward graph
      2. Histogram for final reward: Best to compare the performance of different methodology
      3. State value and Action value graph
B. **Result 5:** Congestion Penalty vs Default: 100 users, 5 destinations
   i. Compare Congestion Status
   ii. Cumulative Reward Histogram / State – Action value graph

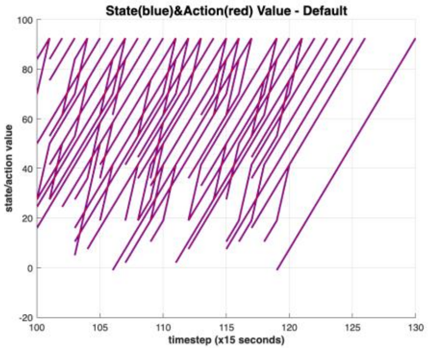
```
>> start_state'

ans =

    40    44     7    44    31     5    14    27    46    47     8    47    46    24    39     7    21    44    39    47

>> destination_values

destination_values =

    45     2    41    32

>> destination_state

destination_state =

    41    45    41    45     2    45    45    32    41     2    32    45     2     2    32    32    45     2     2    41

>> start_time

start_time =

   114   115   105   114   113   103   102   110   120   107   112   104   115   105   110   114   118   120   111   102

>> agents_input

agents_input =

   114    40    41
   115    44    45
   105     7    41
   114    44    45
   113    31     2
   103     5    45
   102    14    45
   110    27    32
   120    46    41
   107    47     2
   112     8    32
   104    47    45
   115    46     2
   105    24     2
   110    39    32
   114     7    32
   118    21    45
   120    44     2
   111    39     2
   102    47    41
```

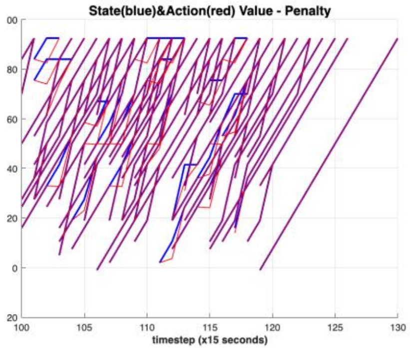
Result [3/4]

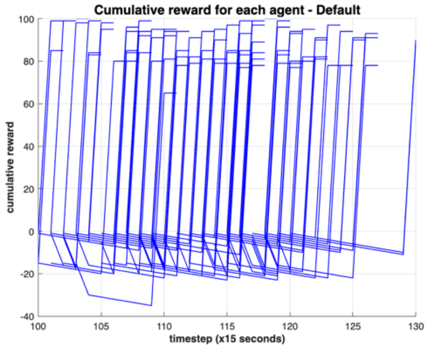
RAAN diff = 15 deg

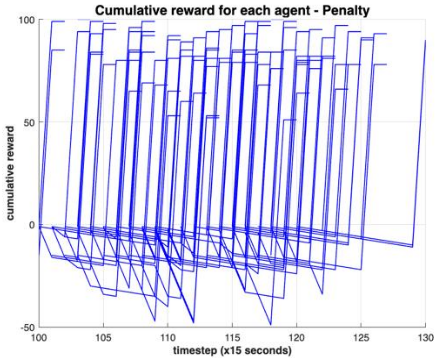


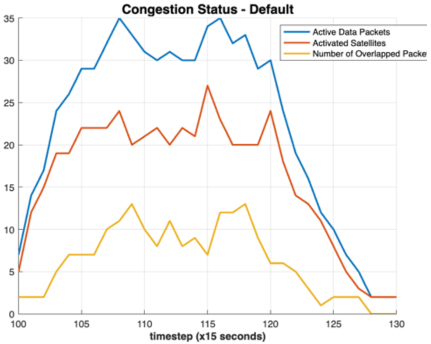
Result [4/4]

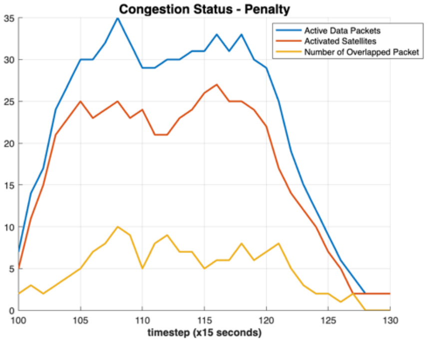


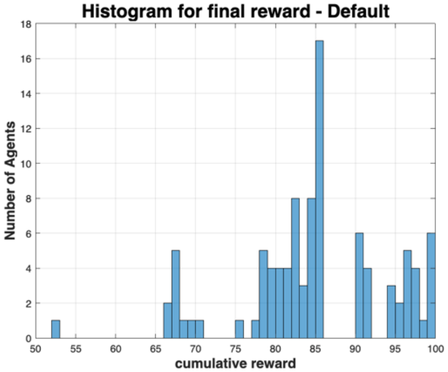


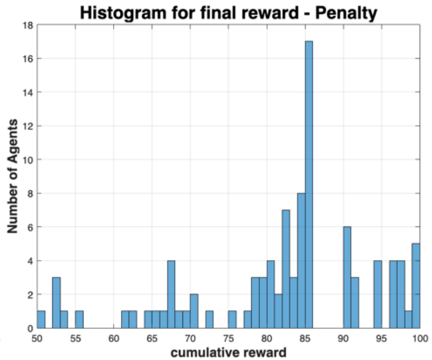












`congestion_factor_sum = 192`

`congestion_factor_sum = 150`

Discussion Points

MDP Calculation

```
Total Number of Destinations: 5
--------------------------------------
Running MDP 1 / 5 , Destination 3
simulation set up complete!
Value Iteration: 100
Value Iteration: 200
Value Iteration: 300
Value Iteration: 400
Policy: 1 -> Value Iteration: 412
Policy: 2 -> Value Iteration: 15
Policy: 3 -> Value Iteration: 8
Policy: 4 -> Value Iteration: 7
Policy: 5 -> Value Iteration: 12
Policy: 6 -> Value Iteration: 5
Policy: 7 -> Value Iteration: 6
Policy: 8 -> Value Iteration: 6
Policy: 9 -> Value Iteration: 2
--------------------------------------
```

In Cooperative Method

```
--------------------------------------
collision occured at time index 113
No. of Active Agents: 7, No. of Agent-Action Cases: 16384
Optimal Action Values:
83.00 100.00 80.00 99.00 91.00 100.00 100.00
Sum: 653.00
Updated Action Values at case index 11:
83.00 100.00 80.00 99.00 89.00 100.00 100.00
Sum: 651.00
Collision Resolved!
--------------------------------------
```

In Penal

```
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
Policy Updated Comple
```

Conclusion and Future Plan

The University of Texas at Austin
Cockrell School of Engineering