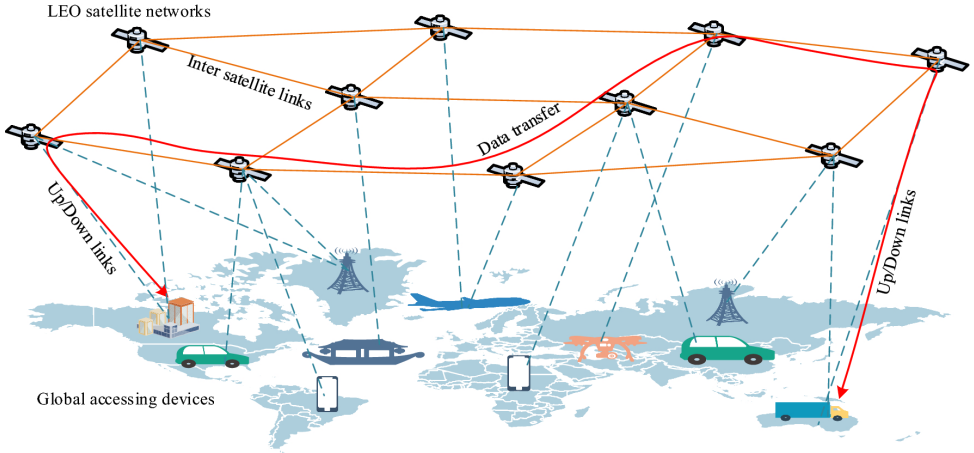
The University of Texas at Austin
Aerospace Engineering
and Engineering Mechanics
Cockrell School of Engineering

The Redstone Project

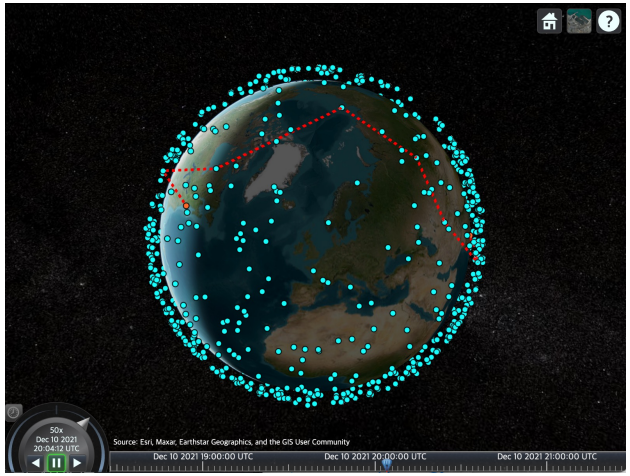
Satellite Network Control System Design using MDP with Cooperative Game

Hongseok Kim
Oden Institute for Computational Engineering and Sciences
08/03/2024

Problem Statement



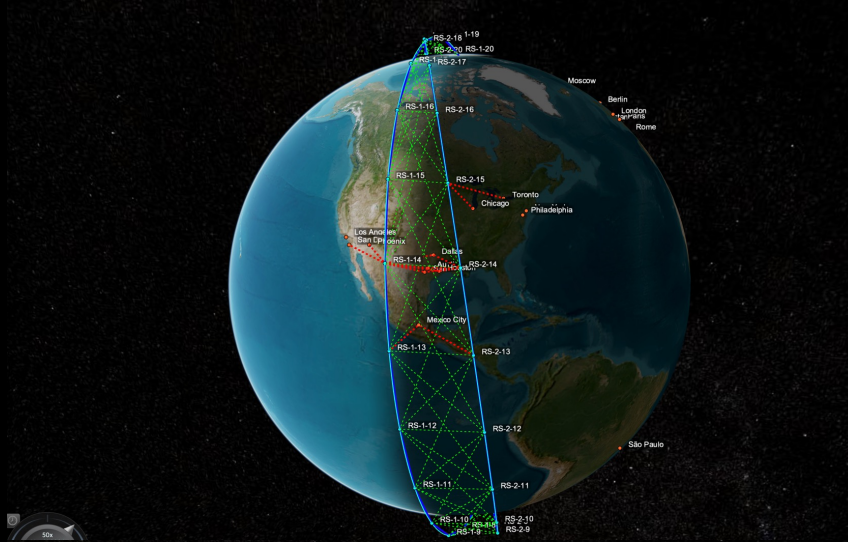
Zhao, N., Long, X. & Wang, J. A multi-constraint optimal routing algorithm in LEO satellite networks. Wireless Netw (2021). https://doi.org/10.1007/s11276-021-02674-3

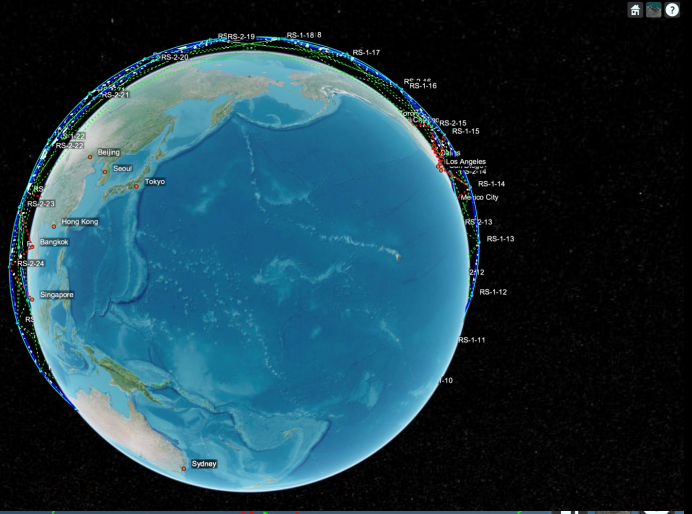


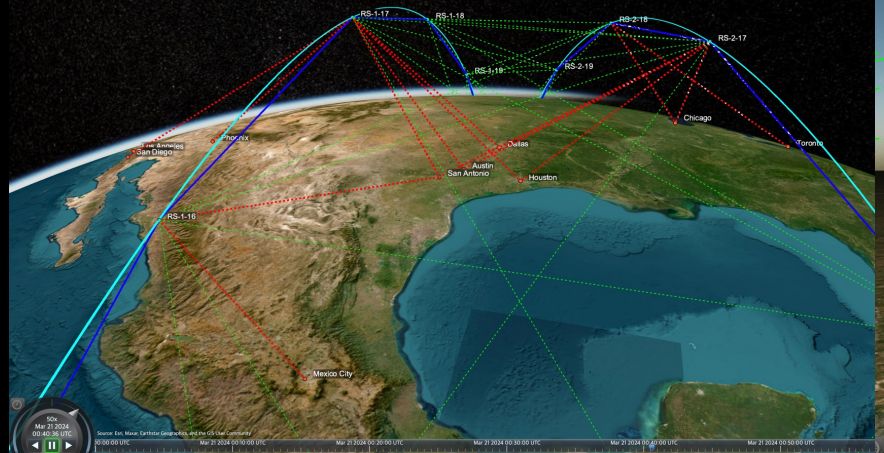
https://www.mathworks.com/help/satcom/ug/multihop-path-select-through-sat-constellation.html

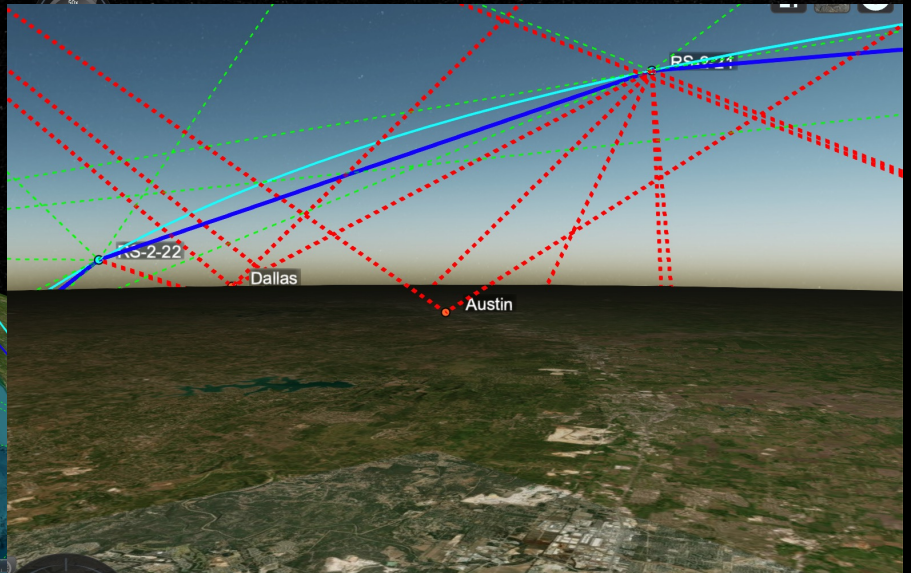
I. Introduction

- Recently, ride-share launch vehicle service led lower cost of satellite operation, which made possible for design large-constellation for various missions.
- STARLINK currently operates 3000 satellites and OneWeb operates 700 satellite at LEO for low latency internet service globally.
- Planet Lab operates 300 satellites and South Korea ADD plans 100 satellites constellation at LEO for Earth Observation Mission, which minimizes re-visit time.
- Therefore, the demand for controlling constellation via autonomous network control sequence going to be higher, adapting optimized mission operation concept generated by machine learning solution.
- This project proposes the structure of satellite network control system and mission operation protocols in given operation requirements with simulation results.


RS-2-18
RS-1-19
RS-2-20
RS-1-20
RS-1
RS-2-17
Moscow
Berlin
London
Paris
Rome
RS-1-16
RS-2-16
RS-1-15
RS-2-15
Toronto
Chicago
Philadelphia
Los Angeles
San Diego
Phoenix
Dallas
Austin
Houston
RS-1-14
RS-2-14
Mexico City
RS-1-13
RS-2-13
RS-1-12
RS-2-12
São Paulo
RS-1-11
RS-2-11
RS-1-10
RS-2-10
RS-1-9
RS-2-9


RS-2-19
RS-1-18
RS-2-20
RS-1-17
RS-2-21
RS-1-16
RS-2-15
RS-1-15
Dallas
Los Angeles
RS-1-14
Mexico City
RS-2-13
RS-1-13
RS-2-12
RS-1-12
RS-1-11
RS-1-10
Beijing
Seoul
Tokyo
RS-2-23
Hong Kong
Bangkok
RS-2-24
Singapore
Sydney


RS-1-17
RS-1-18
RS-2-18
RS-2-17
RS-1-19
RS-2-19
Chicago
Toronto
Los Angeles
San Diego
Phoenix
Dallas
Austin
San Antonio
Houston
RS-1-16
Mexico City


RS-2-22
Dallas
Austin

II. Framework of the Project [1/3]

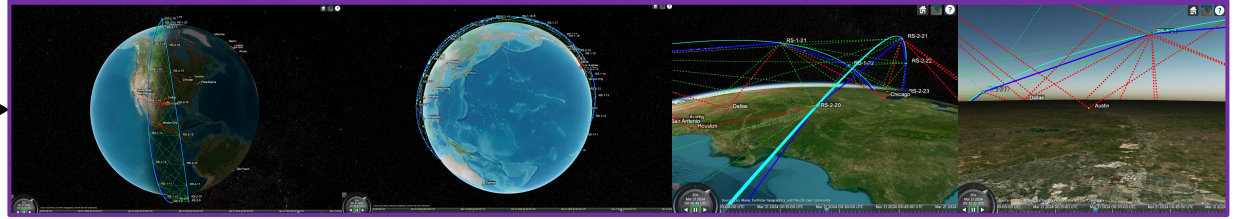
Satellites Configuration

Orbit Parameter Design
Constellation Design

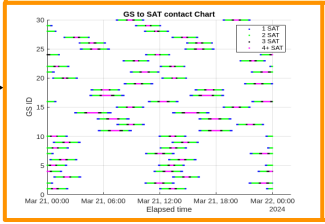
Ground Station Config

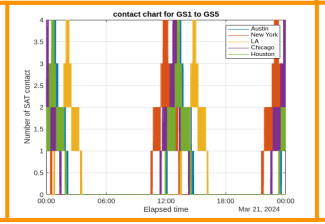
Lat/Long/Altitude Position
Elevation Angle Condition

Visualization

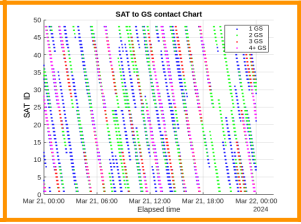


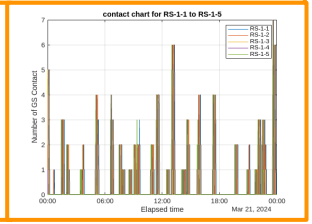
GS to SAT Contact Chart





SAT to GS Contact Chart

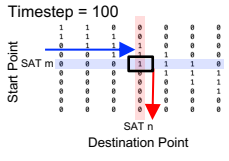




Contact 3D Matrix

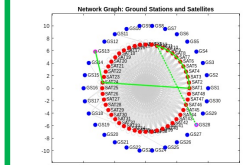
GS_to_SAT
SAT_to_SAT
SAT_to_GS

Network Access Availability Using Dataset



- There Exist 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



II. Framework of the Project [2/3]

Network Dataset

SAT_to_SAT_
Contact_3D_Matrix

Key Equations

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

$$V(s) = \max_a q(s,a)$$

$$\pi(s) = \arg\max_a q(s,a)$$

Mechanism Diagram

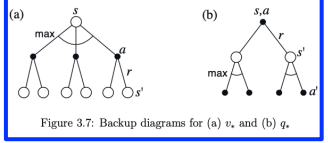


Figure 3.7: Backup diagrams for (a) v_* and (b) q_*.

MDP Structure Configurations

Lv1	MDP	$M = (S, A, P, R, \gamma)$	
Lv2	Timestep	Determined by Network Dataset	
Lv3	State	Policy_Distribution	State_Value_Vec
Lv4	Action	State_Value	Action_Value_Vec
Lv5	Next_State	Action_Value	Policy_Function
Lv6	Next_State_Value	State_Transition_Probability	Reward

Updated by Policy Iteration

Policy Evaluation

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)

Policy Improvement

3. Policy Improvement
$policy - stable \leftarrow$ true
For each $s \in S$
$a \leftarrow \pi(s)$
$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$
If $a \neq \pi(s)$, then $policy - stable \leftarrow$ false
If policy − stable, then stop and return V and π; also go to 2

II. Framework of the Project [3/3]

Time Variant , Multi Agent MDP Configuration

$$q(s,a,t) = \sum_{s',r} p(s',r|s,a)[r(s,a,s',t+1) + \gamma V(s',t+1)]$$

Indicates network change between t and t+1

$$V(s,t) = \max_a q(s,a,t)$$

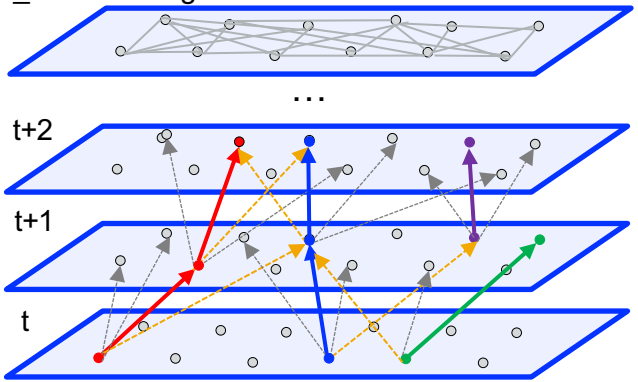
$$\pi(s,t) = \arg\max_a q(s,a,t)$$

Input info from t : $p(s',r|s,a)$ ← **From Contact Matrix**

Input info from $t+1$: $V(s',t+1), r(s,a,s',t+1)$

Output info for t : $q(s,a,t),\ V(s,t),\ \pi(s,t)$

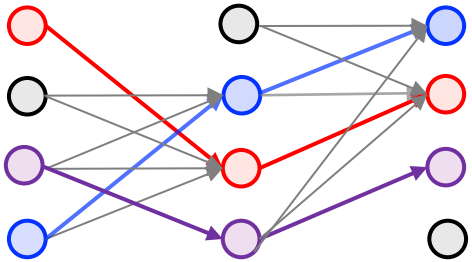
t_final: converged network structure



Data Packet Collision Avoidance Algorithm

Sequential Approach (Perfect Hierarchy)

Sequence: > >



: Actual Path by Agent
: Deactivated after red and blue

```
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

Parallel Approach (Cooperative Game)

Packet Collision Avoidance Algorithm – Cooperative Game

this exmple : **active** 4 **states**, **each state have** 4 **available actions**

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

⇒ 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

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

for case 1 : 256

if case k does not have collision

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

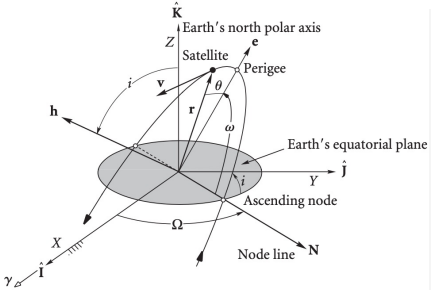
⇒ collision resolved

III. Satellite Constellation Scenario Configuration [1/3]

3.1 Satellites Configuration

- Design Orbit: Kepler Orbit Elements

```
% Intial Orbit Settings

Altitude = 500; % (km)
Eccentricity = 1e-5;
Inclinaton = 97.4022; % (deg)
RAAN = 90;  % (deg)
AOP = 0; % (deg)
% Number of satellites
number_of_SATs = 24;
```



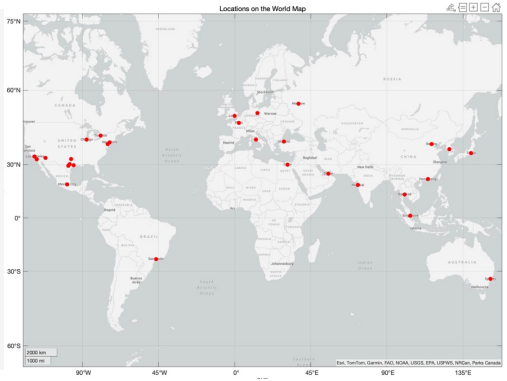
- Allocate Satellite in each orbit

```
% Satellite Characteristics - This will entail -1, -2, ... for
% constallation
satellite_name = "RS-1";

% Orbit Propagator
Orbit_Propagator = 'SGP4';

orbit_information.SMA = Altitude*1000 + 6378000;
orbit_information.ecc = Eccentricity;
orbit_information.inc = Inclinaton;
orbit_information.RAAN = RAAN;
orbit_information.AOP = AOP;

% Generate First Stream of Satellites
satellites_orbit_1 = single_orbit_constellation(scenario, ...
                                                orbit_information, ...
                                                number_of_SATs, ...
                                                Orbit_Propagator, ...
                                                satellite_name);
```

3.2 Ground Stations Configuration

- 3D Geographical Locations:
  Latitude/Longitude/Altitude

```
gs_info = ["Austin", 30.2672, -97.7431, 15;
           "New York", 40.7128, -74.0060, 10;
           "Los Angeles", 34.0522, -118.4837, 71;
           "Chicago", 41.8781, -87.6298, 181;
           "Houston", 29.7604, -95.3698, 30;
           "Phoenix", 33.4484, -112.0740, 331;
           "Philadelphia", 39.9526, -75.1652, 12;
           "San Antonio", 29.4481, -98.4936, 198;
           "San Diego", 32.7157, -117.1611, 22;
           "Dallas", 32.7767, -96.7970, 137;
           "London", 51.5074, -0.1278, 35;
           "Tokyo", 35.6895, 139.6917, 6;
           "Paris", 48.8566, 2.3522, 35;
           "Moscow", 55.7558, 37.6176, 155;
           "Sydney", -33.8688, 151.2093, 19;
           "Toronto", 43.6511, -79.3830, 76;
           "Berlin", 52.5200, 13.4050, 34;
           "Rome", 41.9028, 12.4964, 17;
           "Seoul", 37.5665, 126.9780, 23;
           "Mumbai", 19.0760, 72.8777, 14;
           "Beijing", 39.9042, 116.4074, 43;
           "Mexico City", 19.4326, -99.1332, 2240;
```



- Elevation Angle Condition
  (GS – SAT connectivity condition)

```
gs = groundStation(scenario,"Name",gs_info(:,1), ...
                            "Latitude",str2double(gs_info(:,2)), ...
                            "Longitude", str2double(gs_info(:,3)), ...
                            "Altitude", str2double(gs_info(:,4)), ...
                            "MinElevationAngle", 15);
```

III. Satellite Constellation Scenario Configuration [2/3]

3.3 Orbit Simulation Configuration

- Epoch / Timestep / Duration Initialization

```
% Initial Epoch
Year = 2024;
Month = 3;
Day = 21;
Hour = 0;
Minute = 0;
Seconds = 0;

% Duration and Steptime
duration = hours(24);
steptime = 15; % seconds
startTime = datetime(Year,Month,Day,Hour,Minute,Seconds);
stopTime = startTime + duration;
scenario = satelliteScenario(startTime,stopTime,steptime);
```

- SAT-to-SAT Access Matrix Generation

III. Create 3D inter-satellite contact (Diff orbit) matrix for each time step

```
sat1_to_sat2_contact_3d_matrix = zeros(24,24,length(time_vector));
sat2_to_sat1_contact_3d_matrix = zeros(24,24,length(time_vector));

for i = 1:24
    sat1_to_sat2_access_status = accessStatus(access(satellites_orbit_1(i),satellites_orbit_2));
    sat1_to_sat2_contact_3d_matrix(i,:,:) = sat1_to_sat2_access_status;
    sat2_to_sat1_contact_3d_matrix(:,i,:)  = sat1_to_sat2_access_status;
end
```

IV.Create 3D inter-satellite contact (Same orbit) matrix for each time step

```
sat1_to_sat1_contact_3d_matrix = zeros(24,24,length(time_vector));

     sat1_to_sat1_contact_3d_matrix(1,1,:) = 1;
     sat1_to_sat1_contact_3d_matrix(2,1,:) = 1;
     sat1_to_sat1_contact_3d_matrix(1,2,:) = 1;
     sat1_to_sat1_contact_3d_matrix(24,1,:) = 1;
     sat1_to_sat1_contact_3d_matrix(1,24,:) = 1;
     sat1_to_sat1_contact_3d_matrix(24,23,:) = 1;
     sat1_to_sat1_contact_3d_matrix(23,24,:) = 1;
     sat1_to_sat1_contact_3d_matrix(24,24,:) = 1;

for i = 2:23
   sat1_to_sat1_contact_3d_matrix(i,i,:) = 1;
   sat1_to_sat1_contact_3d_matrix(i+1,i,:) = 1;
   sat1_to_sat1_contact_3d_matrix(i,i+1,:) = 1;
   sat1_to_sat1_contact_3d_matrix(i,i-1,:) = 1;
   sat1_to_sat1_contact_3d_matrix(i-1,i,:) = 1;
end

sat2_to_sat2_contact_3d_matrix = sat1_to_sat1_contact_3d_matrix;
sat_to_sat_contact_3d_matrix = [sat1_to_sat1_contact_3d_matrix, sat2_to_sat1_contact_3d_matrix;
                                sat1_to_sat2_contact_3d_matrix, sat2_to_sat2_contact_3d_matrix];
```

- GS – to - SAT Access Matrix Generation

II. Creat 3D ground-to-satellite contact matrix for each time step

```
number_of_gs =length(gs);
number_of_SATs = length(satellites);
time_vector = scenario.StartTime:seconds(scenario.SampleTime):scenario.StopTime;

gs_to_sat_contact_3d_matrix = zeros(number_of_gs, number_of_SATs, length(time_vector));

for i = 1:number_of_gs
    gs_to_sat_access_satatus = accessStatus(access(gs(i), satellites));
    gs_to_sat_contact_3d_matrix(i,:,:) = gs_to_sat_access_satatus;
end
```

- Resultant Contact Table

gs_to_sat_access_table = *273×8 table*

	Source	Target	IntervalNumber	StartTime	EndTime	Duration
1	"Austin"	"RS-1-1"	1	2024-03-21 00:05:15	2024-03-21 00:11:00	345
2	"Austin"	"RS-1-2"	1	2024-03-21 00:01:30	2024-03-21 00:07:00	330
3	"Austin"	"RS-1-3"	1	2024-03-21 00:00:00	2024-03-21 00:03:00	180
4	"Austin"	"RS-1-12"	1	2024-03-21 00:57:00	2024-03-21 01:00:00	180
5	"Austin"	"RS-1-13"	1	2024-03-21 00:53:00	2024-03-21 00:57:15	255
6	"Austin"	"RS-1-14"	1	2024-03-21 00:48:45	2024-03-21 00:53:30	285
7	"Austin"	"RS-1-15"	1	2024-03-21 00:44:45	2024-03-21 00:49:45	300
8	"Austin"	"RS-1-16"	1	2024-03-21 00:40:45	2024-03-21 00:45:45	300
9	"Austin"	"RS-1-17"	1	2024-03-21 00:36:45	2024-03-21 00:42:00	315

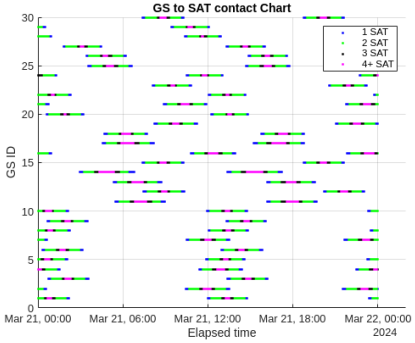
sat_to_gs_access_table = *273×8 table*

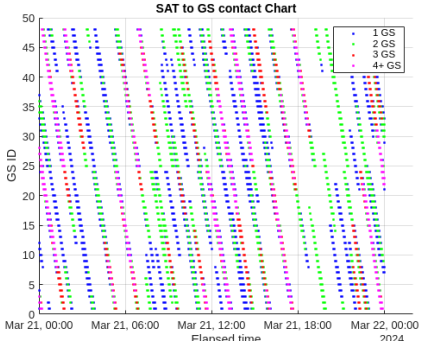
	Source	Target	IntervalNumber	StartTime	EndTime	Duration
1	"RS-1-1"	"Austin"	1	2024-03-21 00:05:15	2024-03-21 00:11:00	345
2	"RS-1-1"	"Chicago"	1	2024-03-21 00:08:45	2024-03-21 00:13:00	255
3	"RS-1-1"	"Houston"	1	2024-03-21 00:05:00	2024-03-21 00:10:45	345
4	"RS-1-1"	"San Antonio"	1	2024-03-21 00:05:15	2024-03-21 00:10:45	330
5	"RS-1-1"	"Dallas"	1	2024-03-21 00:06:00	2024-03-21 00:11:30	330
6	"RS-1-1"	"Mumbai"	1	2024-03-21 00:41:15	2024-03-21 00:44:15	180
7	"RS-1-1"	"Mexico City"	1	2024-03-21 00:03:00	2024-03-21 00:08:00	300
8	"RS-1-2"	"Austin"	1	2024-03-21 00:01:30	2024-03-21 00:07:00	330
9	"RS-1-2"	"Chicago"	1	2024-03-21 00:04:30	2024-03-21 00:09:15	285

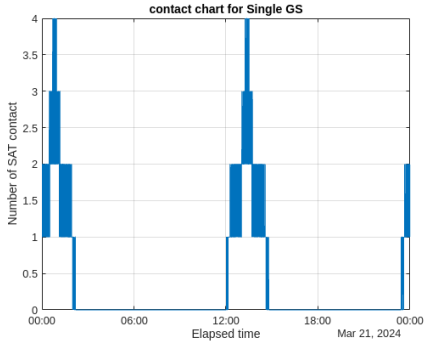
III. Satellite Constellation Scenario Configuration [3/3]

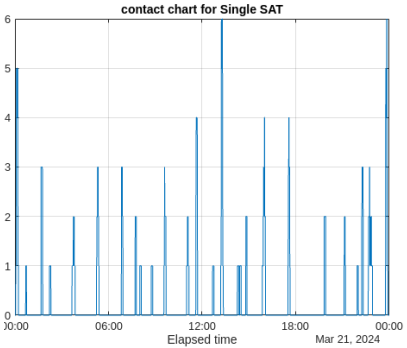
3.4 Orbit Simulation result

- GS→ SAT / SAT-> GS contact chart generation

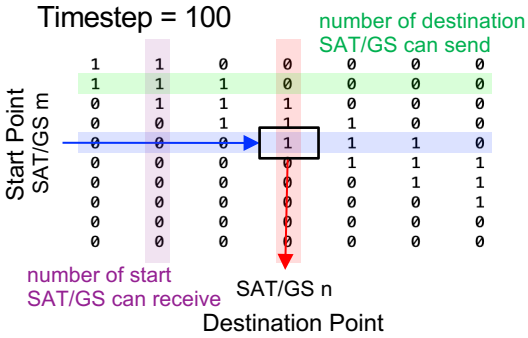




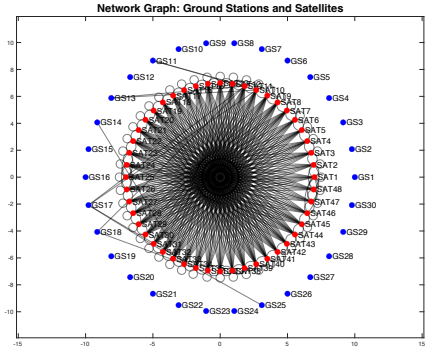


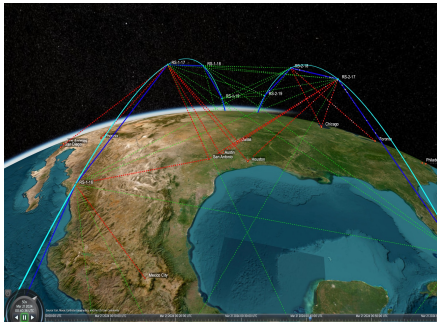


- Contact 3D Matrix -> network graph generation



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

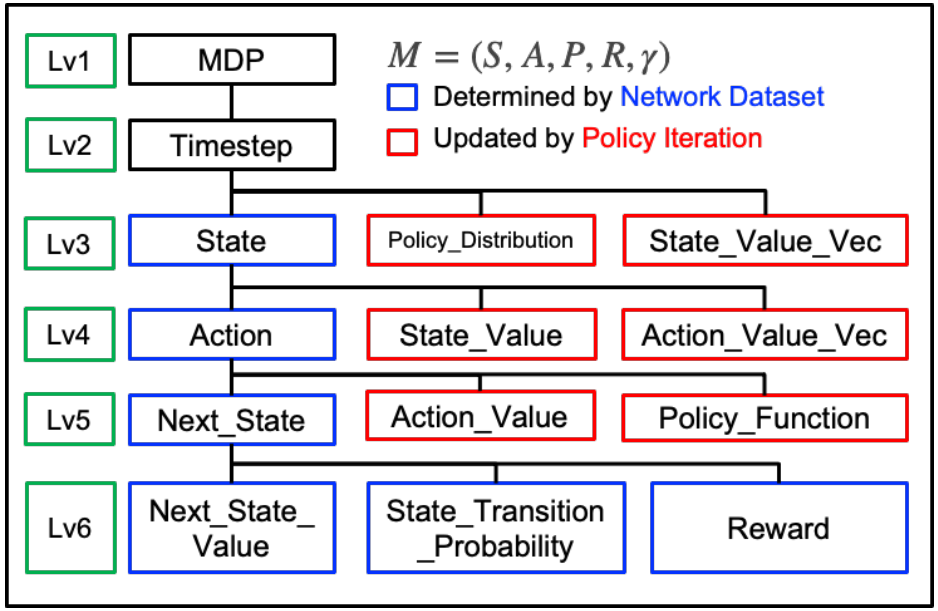




IV. MDP-DP structure Configuration [1/3]

4.1 Network Dataset to MDP Structure Definition

- In creating MDP framework environment, we need 5 necessary variable space $M = (S, A, P, R, \gamma)$and policy $\pi(a \mid s)$
- Since we are simulating the network structure, we define action as transiting one data packet from state to state
- Caution: The satellites and ground stations themselves are not the states. The phenomenon that each active data packet is located at each agent are states.



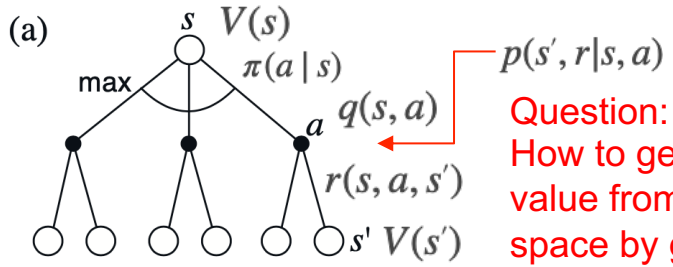
S : Finite **Set** of States A : Finite **Set** of Actions γ : Discount **Factor**

P : State Trasition Probability **Matrix** R : Reward **Function**

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

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

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



Question:
How to get action value from state-action space by given policy?

IV. MDP-DP structure Configuration [2/3]

4.2 Using Dynamic Programming (Policy Iteration) for solving 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; else go to 2

Reward System by next state by different cases

Case 1: Regular transition

- Currently, we have two sub-orbits of the satellite system: Orbit 1 and Orbit 2.
- Transiting between different orbit may take more cost, which means less reward.

1-1: From orbit 1 to orbit 1: -1

1-2: From orbit 1 to orbit 2: -15

1-3: From orbit 2 to orbit 1: -1

1-4: From orbit 2 to orbit 2: -15

1-5: Returning to starting state: -15

Case 2: Next State = end state

2-1: From orbit 1 to orbit 1: 100

2-2: From orbit 1 to orbit 2: 50

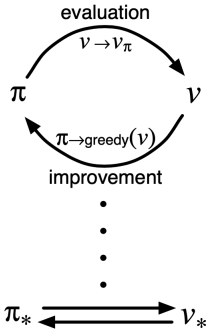
2-3: From orbit 2 to orbit 1: 100

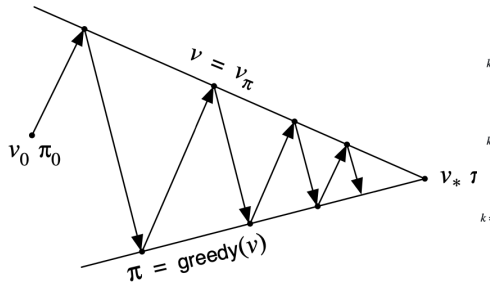
2-4: From orbit 2 to orbit 2: 50

2-5: Returning to starting state: -15

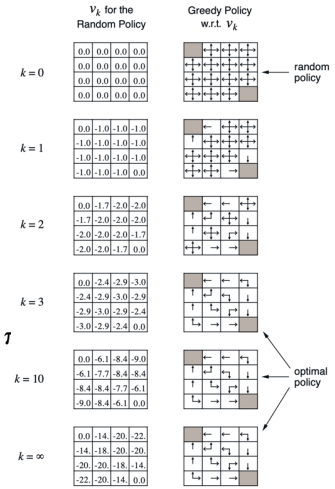
Case 3: Starting State = end state

3-1: Returing to starting state: 0





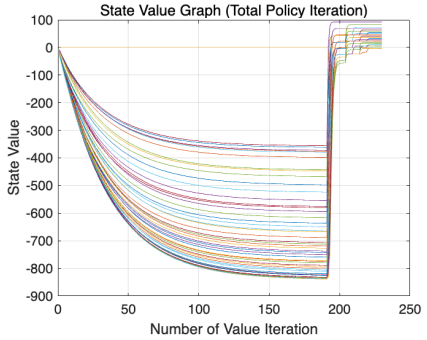

```
simulation set up complete!
Policy: 1 -> Value Iteration: 466
Policy: 2 -> Value Iteration: 14
Policy: 3 -> Value Iteration: 5
Policy: 4 -> Value Iteration: 5
Policy: 5 -> Value Iteration: 8
Policy: 6 -> Value Iteration: 10
Policy: 7 -> Value Iteration: 9
Policy: 8 -> Value Iteration: 1
Policy: 9 -> Value Iteration: 1
Policy: 10 -> Value Iteration: 1
Policy: 11 -> Value Iteration: 1
```

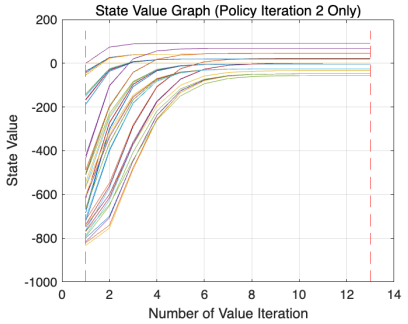


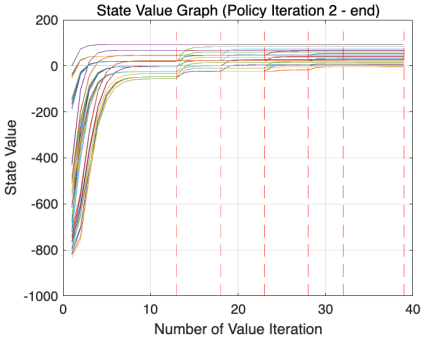
states_index	state_value
37	92.269
39	92.269
36	82.64
40	82.64
41	73.131
35	73.131
11	68.675
12	68.675
13	68.675
14	68.675
15	68.675
16	68.675
17	68.675
18	68.675
42	63.74
34	63.74
10	59.34
19	59.34
43	54.467
33	54.467
9	50.121
20	50.121
32	45.375
44	45.375
21	41.017
8	41.017
31	36.33
45	36.33
22	32.027
7	32.027
46	27.398
24	27.398
30	27.398
23	23.149
6	23.149
47	18.578
29	18.578
5	14.381
48	9.8675
28	9.8675
4	5.7196
25	4.6129
26	4.6129
1	4.6129
27	1.2656
38	0
3	−2.8527
2	−3.9234

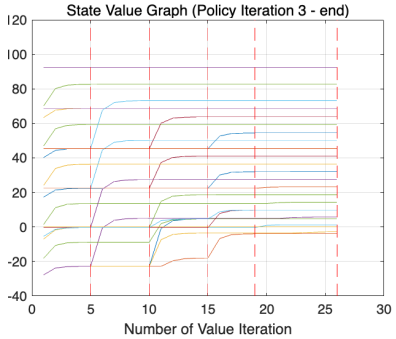
IV. MDP-DP structure Configuration [3/3]

4.3 State Value Change by Value/Policy Iteration

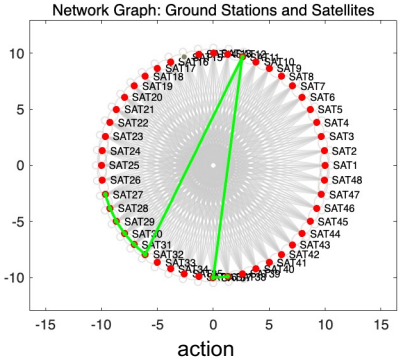


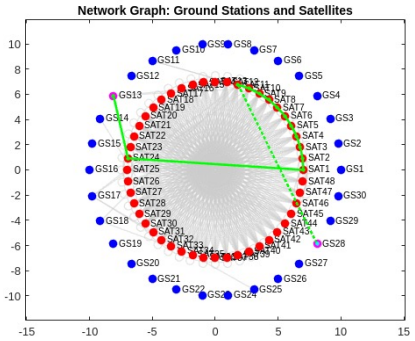


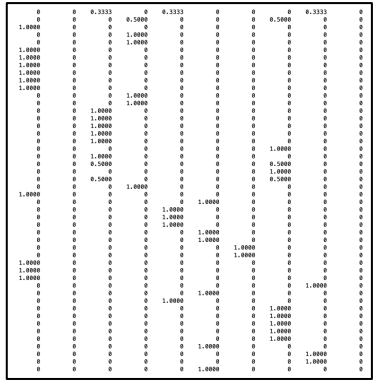




4.4 Generated Network Path (Time Invariant







← Policy Distribution

Simulation Result

state_list	cumulative_reward_list	state_value_list
3	0	2
4	−1	10.5
5	−2	19
31	−17	41.5
32	−18	50
33	−19	58.5
34	−20	67
35	−21	75.5
36	−22	84
37	−23	92.5
38	77	0

V. Time Variant – Multi Agent MDP Configuration [1/2]

5.1 Basic Mechanism – time propagation

$$q(s,a,t) = \sum_{s',r} p(s',r|s,a)[r(s,a,s',t+1) + \gamma V(s',t+1)]$$

$$V(s,t) = \max_a q(s,a,t)$$

$$\pi(s,t) = \arg\max_a q(s,a,t)$$

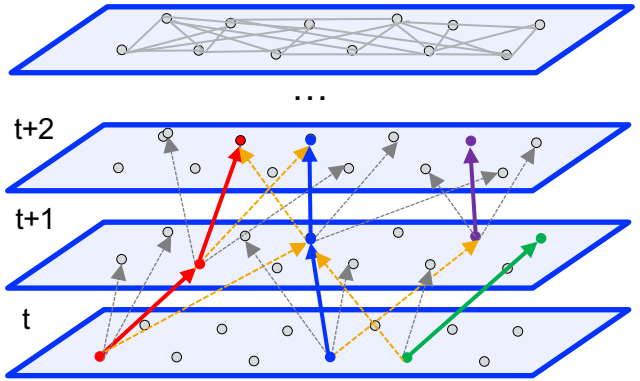
Indicates network change between t and t+1

Input info from t : $p(s',r|s,a)$ ← **From Contact Matrix**

Input info from $t+1$: $V(s',t+1), r(s,a,s',t+1)$

Output info for t : $q(s,a,t),\ V(s,t),\ \pi(s,t)$

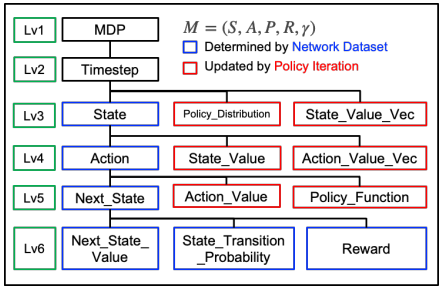
t final: converged network structure



Steps for Time Variant MDP – DP

Step 1 : Policy Iteration for $T = T_0 + \Delta T$

Output info for the input : $V(s, T_0 + \Delta T)$



Step 2 : State Value Propagation from $T = T_0 + \Delta T$ to $T = T_0$

$$q(s,a,t) = \sum_{s',r} p(s',r|s,a)(r(s') + \gamma V(s',t+1)) \rightarrow \text{Level 4}$$

$$V(s,t) = \max_a \sum_{s',r} p(s',r|s,a)(r(s') + \gamma V(s',t+1)) \rightarrow \text{Level 3}$$

Input info from $T = t$: $p(s',r|s,a) \Rightarrow$ From Satellite to Satellite Contact Matrix at $T = t$

– We have to define action from $s(t)$ to $s'(t+1)$ and corresponding rewards

Input info from $T = t+1$: $V(s',t+1) \Rightarrow$ From State value (level 3) at $T = t+1$

Output info for $T = t$: $V(s,t), q(s,a,t) \Rightarrow$ Reuse this information to calculate $V(s,t-1), q(s,a,t-1)$

Step 3 : Policy Determination Based of State Value

For each $T = t$,

$$\pi(s,t) = \underset{a}{\text{argmax}} \sum_{s',r} p(s',r|s,a)(r(s') + \gamma V(s',t+1))$$

If there is n multiple actions which have same action value, $\pi = \frac{1}{n}$ for each action

Step 4 : Test the simulation

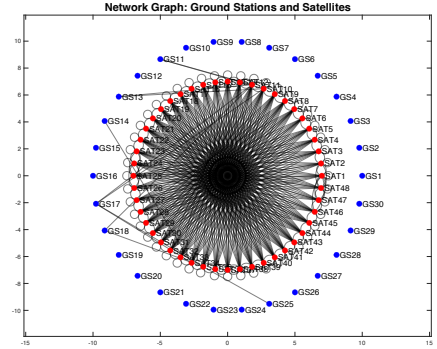
Input : S_n at $T = T_0$

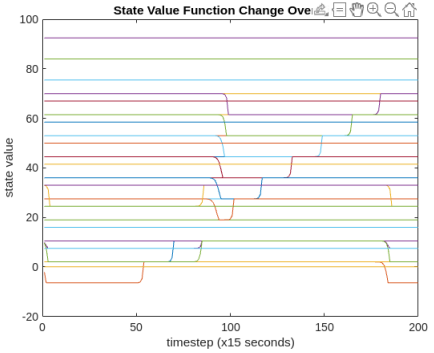
Output : S_m at $T = T_0 + \Delta T$

Or any time, any input can be possible, check wheter the output is intended destination

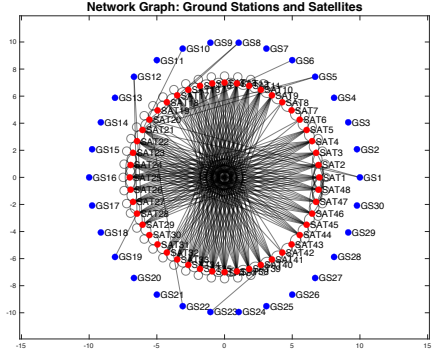
V. Time Variant – Multi Agent MDP Configuration [2/2]

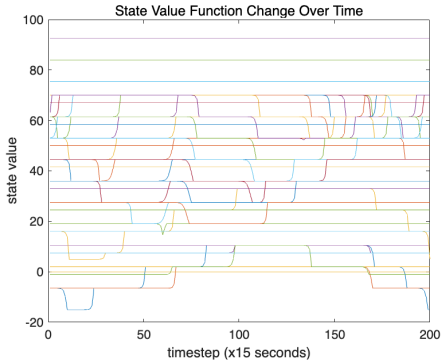
5.2 State Value Distribution Change over time: reflects network change



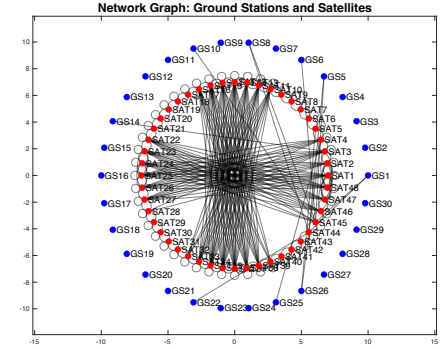


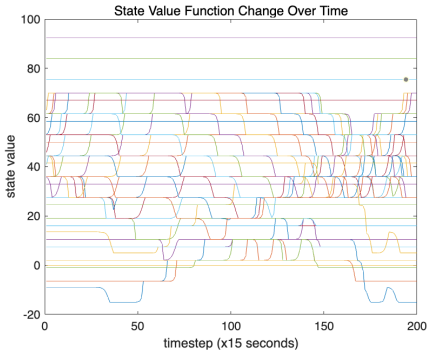
RAAN difference = 15 deg



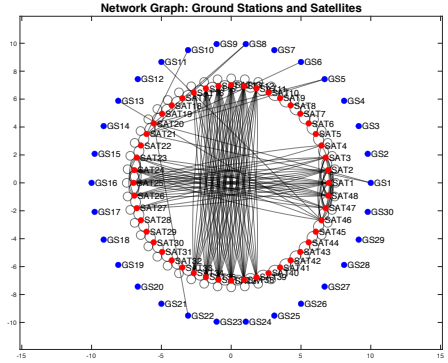


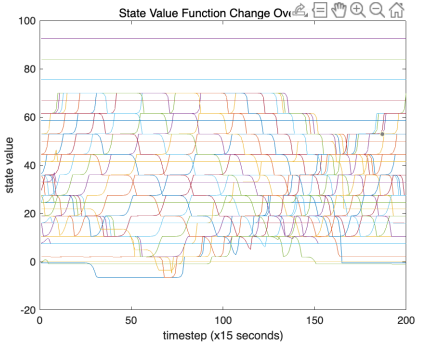
RAAN difference = 45 deg





RAAN difference = 60 deg





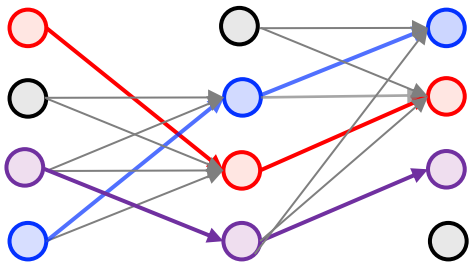
RAAN difference = 90 deg

VI. Collision Avoidance Algorithm [1/4]

6.1 Sequential Approach (Perfect Hierarchy)

- Algorithm Explanation, Code Configuration

Sequence: ○ > ○ > ○

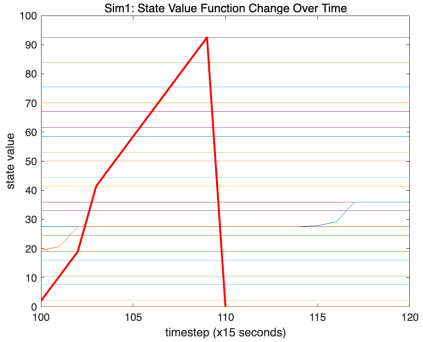


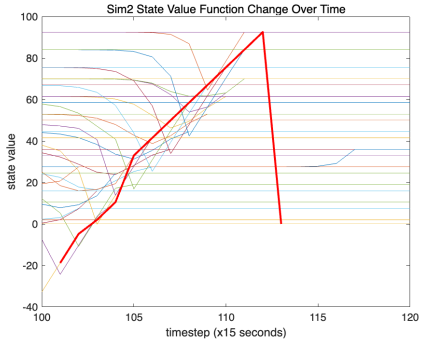
: Actual Path by Agent
: Deactivated after red and blue

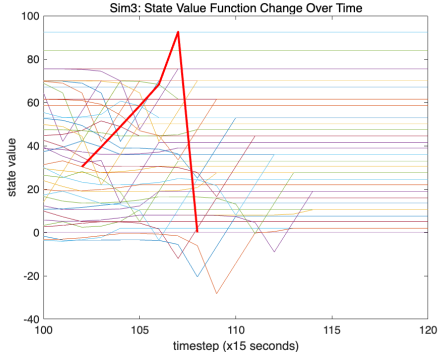
```
sat_to_sat(:,state_index,time_index-1) = 0;
```

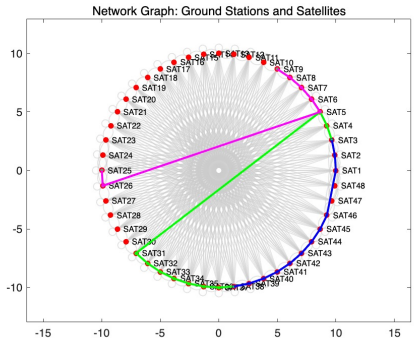
To avoid collision, after each agent's pathway propagation, deactivate every edge approaching to the node in which the agent stayed at given time

- State Value change by every path









VI. Collision Avoidance Algorithm [2/4]

6.2 Parallel Approach (Cooperative Game)

Packet Collision Avoidance Algorithm – Cooperative Game

this exmple : active 4 **states, each state have** 4 **available actions**

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

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

for case $1 : 256$

if case k does not have collision

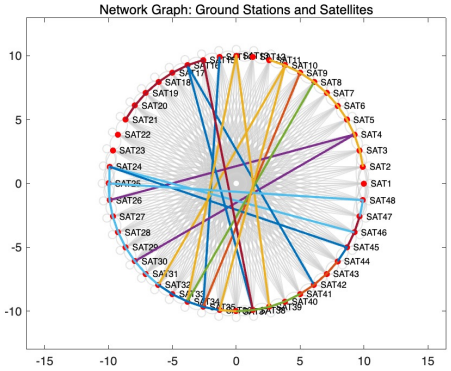
take case k : $\{(a_{13}, q_{13}), (a_{21}, q_{21}), (a_{34}, q_{34}), (a_{41}, q_{41})\}$

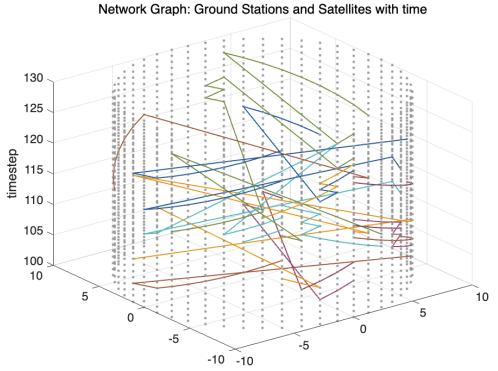
$\Rightarrow$ collision resolved

```
simulation set up complete!
Policy: 1 -> Value Iteration: 419
Policy: 2 -> Value Iteration: 14
Policy: 3 -> Value Iteration: 5
Policy: 4 -> Value Iteration: 5
Policy: 5 -> Value Iteration: 5
Policy: 6 -> Value Iteration: 10
Policy: 7 -> Value Iteration: 9
Policy: 8 -> Value Iteration: 1
Policy: 9 -> Value Iteration: 1
```

```
collision occured at time index 109
collision resolved at time index 109  at case index 5
collision occured at time index 111
collision resolved at time index 111  at case index 59
collision occured at time index 112
collision resolved at time index 112  at case index 2
collision occured at time index 113
collision resolved at time index 113  at case index 5
```

100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
102	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	45
103	0	0	0	0	0	47	9	0	0	0	0	0	0	0	0	0	0	0	0	24
104	0	0	0	0	0	46	35	0	0	0	0	7	0	0	0	0	0	0	0	26
105	0	0	14	0	0	45	36	0	0	0	0	8	0	44	0	0	0	0	0	27
106	0	0	35	0	0	44	37	0	0	0	0	34	0	43	0	0	0	0	0	28
107	0	0	34	0	0	43	0	0	0	24	0	35	0	42	0	0	0	0	0	29
108	0	0	33	0	0	42	0	0	0	45	0	36	0	41	0	0	0	0	0	30
109	0	0	0	0	0	41	0	0	0	44	0	37	0	40	0	0	0	0	0	31
110	0	0	0	0	0	40	0	47	0	43	0	0	0	41	39	0	0	0	0	32
111	0	0	0	0	0	39	0	48	0	42	0	0	0	40	38	0	0	0	41	33
112	0	0	0	0	0	38	0	25	0	16	39	0	0	41	37	0	0	0	40	0
113	0	0	0	0	46	37	0	26	0	38	13	0	0	40	36	0	0	0	39	0
114	31	0	0	27	45	0	0	4	0	37	34	0	0	39	35	47	0	0	38	0
115	32	5	0	28	44	0	0	30	0	0	33	0	21	38	34	48	0	0	37	0
116	33	6	0	29	43	0	0	31	0	0	0	0	20	37	35	25	0	0	0	0
117	0	7	0	30	42	0	0	32	0	0	0	0	19	0	34	26	0	0	0	0
118	0	8	0	31	41	0	0	33	0	0	0	0	18	0	35	27	32	0	0	0
119	0	9	0	32	40	0	0	0	0	0	0	0	17	0	34	28	33	0	0	0
120	0	35	0	0	39	0	0	46	0	0	0	0	16	0	33	29	34	2	0	0
121	0	36	0	11	38	0	0	24	0	0	0	0	15	0	0	30	35	3	0	0
122	0	35	0	10	37	0	0	26	0	0	0	0	38	0	0	31	36	4	0	0
123	0	36	0	11	0	0	0	27	0	0	0	0	37	0	0	32	35	5	0	0
124	0	37	0	10	0	0	0	28	0	0	0	0	0	0	0	33	36	6	0	0
125	0	0	0	36	0	0	0	29	0	0	0	0	0	0	0	0	37	7	0	0
126	0	0	0	37	0	0	0	30	0	0	0	0	0	0	0	0	0	8	0	0
127	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	9	0	0
128	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	10	0	0
129	0	0	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	36	0	0
130	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0



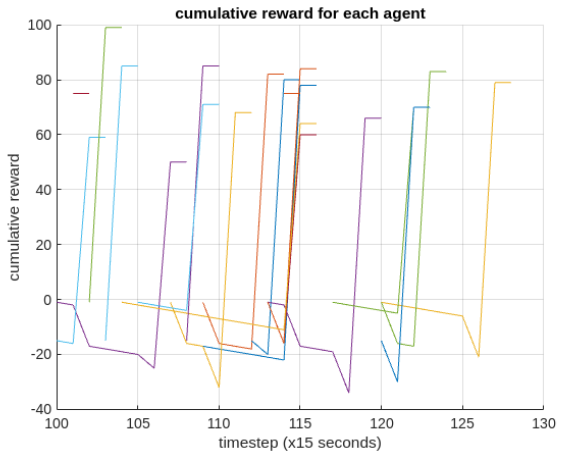


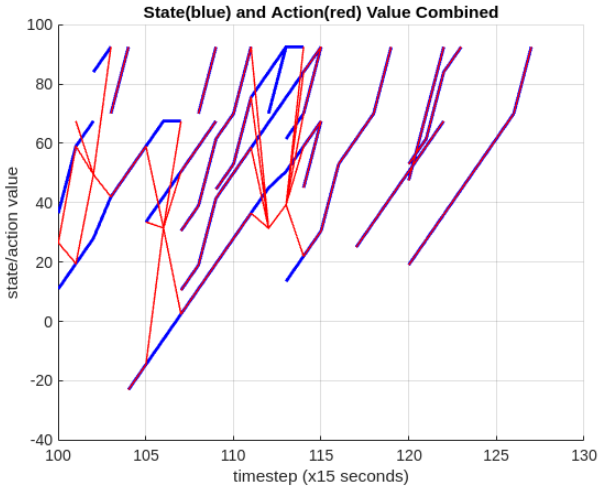
VI. Collision Avoidance Algorithm [3/4]

6.3 Comparison between two methodology

- Pros and Cons
- Quantified Performance Comparison
- state Value / Action Value distribution by each agent
- Total Reward change over time by each agent

```
----------------------------------------
collision occured at time index 102
No. of Active Agents: 4, No. of Agent-Action Cases: 64
Optimal Action Values:
84.00 0.00 28.00 67.50
Sum: 179.50
Updated Action Values at case index 2:
84.00 0.00 21.20 67.50
Sum: 172.70
Collision Resolved!

----------------------------------------
collision occured at time index 106
No. of Active Agents: 3, No. of Agent-Action Cases: 64
Optimal Action Values:
42.00 -6.00 67.50
Sum: 103.50
Updated Action Values at case index 5:
42.00 -6.00 57.50
Sum: 93.50
Collision Resolved!

----------------------------------------
```





VI. Collision Avoidance Algorithm [4/4]

6.4 Using Congestion Game

Packet Collision Avoidance Algorithm – Congestion Game

this exmple : **active** 4 **states**, **each state have** 4 **available actions**

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** $q_i = \{q_{j1}, q_{j2}, q_{j3}, q_{j4}\}$

parse corresponding **next state values** $s'_i = \{s'_{j1}, s'_{j2}, s'_{j3}, s'_{j4}\}$

create set of action – action value – next state vector from given state

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

create dataset of current states including these informations

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

while true (infinite loop)

parse current policy distribution for each active state

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

*c*aution : 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

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

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

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

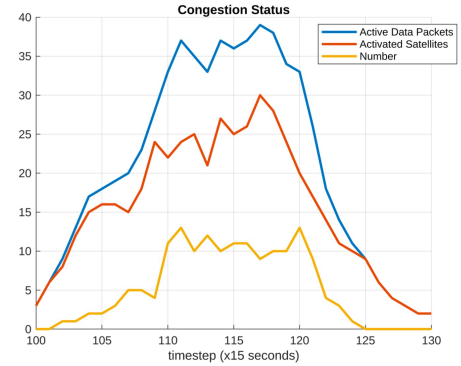
if $\max\{q'_{j1}, q'_{j2}, q'_{j3}, q'_{j4}\} = q'_{j4} \Rightarrow \pi'_j(a'_{j4}|s) = 1$

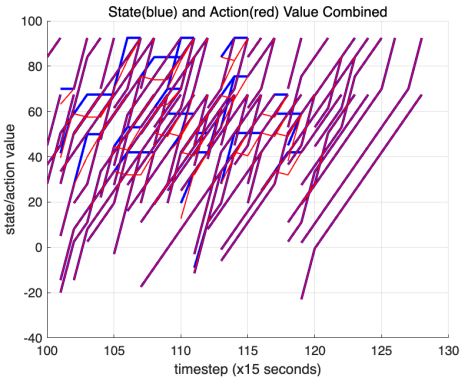
create policy_updated = $\left[\pi'_1 \quad \pi'_2 \quad \pi'_3 \quad \pi'_4\right]$

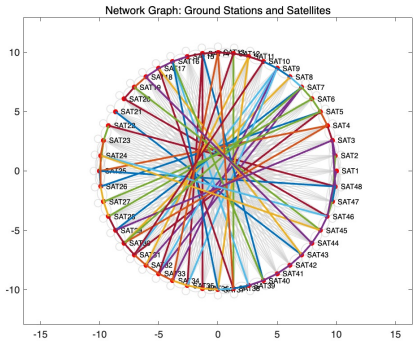
if policy_updated == policy

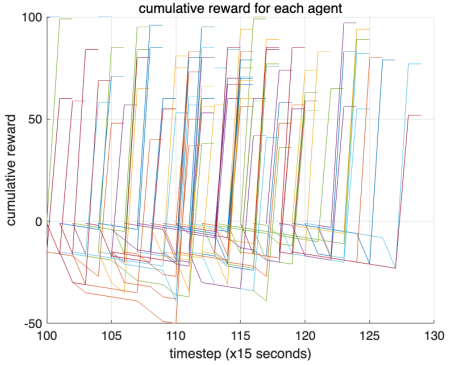
then **break** $\rightarrow$ policy_updated is equilibrium policy for all active agents

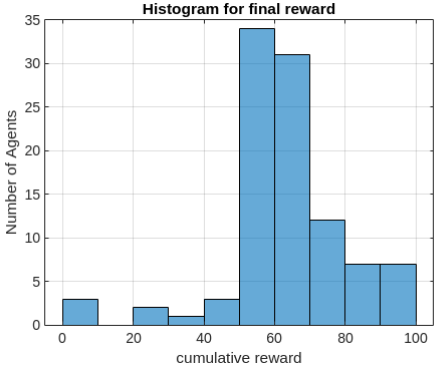
else policy = policy_updated $\rightarrow$ **return** to while loop

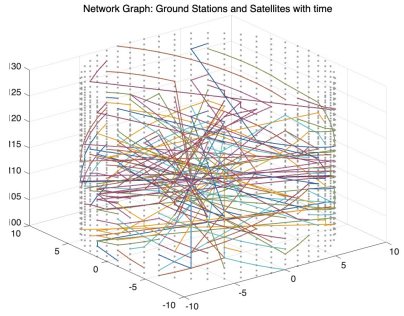












VII. Conclusion, Discussion and Future Topics

“Connecting the Dots with the Line of Genuine Insight”

The University of Texas at Austin
Cockrell School of Engineering