A BRIEF OVERVIEW OF THIS REPOSITORY

> IMPORTANT NOTE :

The paper used both capital 'M' (M+1 = number of rates) and lowercase 'm' (number of channels the jammer can sweep simultaneously). I am using lower case 'm' for the former (# rates) and lowercase 'n' for the latter.

analysis.py

intended to replicate figures presented in the paper

→ runs optimize_game from optimize.py

|returns|

model.py

model : Model of the game containing parameters, transition matrices, etc.

f: optimal transmit strategy

y: optimal jammer strategy

→ runs many Simulations to determine average performance

simulation.py

optimize.py

* optimize_game (function) : Finds the Nash equilibrium (NE) for a set of parameters and returns the strategies & game model

→ runs find_equilibrium

→ uses scipy.optimize.minimize to find the minimum of objective_function

"memfunc"

NOTE class MemoryFunctions is used to save computation time

$$\sum_x V_1(x) + V_2(x)$$

AKA memfunc.get(0, ...)

best_transmitter_value

best_jammer_value

AKA memfunc.get(1, ...)

GOAL: minimize Eqn. (22) AKA objective-function

- best_transmitter_value AKA memfunc.get(0, ...)

$$V_1(x) = \max\Big(\overset{A\times S}{R(x)}\, \overset{S\times 1}{\vec{y}} + \delta\, \overset{A\times S}{T(x, V_1)}\, \overset{S\times 1}{\vec{y}} \Big)$$

$$\overset{\leftarrow S \rightarrow}{\left[\ \ \ \right]}_{A} \left[\ \right]_{S}$$

(1)

Let A be the size of the action space of the transmitter, $(A = 2(M+1))$, and S be the size of the strategy of the jammer $(S = M+1)$.

$R(x)$ AKA model.reward_matrices[x]

get_reward_matrix(x)
(MODEL.PY)

$$= \begin{bmatrix} r(x, a_0, P_{J_0}) & \cdots & r(x, a_0, P_{J_m}) \\ \vdots & \ddots & \vdots \\ r(x, a_{2m+1}, P_{J_0}) & \cdots & r(x, a_{2m+1}, P_{J_m}) \end{bmatrix}$$

where $r(x, a, p)$ AKA transmitter_rewards

get_immediate_transmitter_reward
(MODEL.PY)

$$= \sum_{x'} U(\cdot, a_1, a_2, x')\, P(x' | x, a_1, a_2)$$

$U(\cdot, a_1, a_2, x')$

AKA transmitter_payoffs

get_immediate_transmitter_payoffs
(MODEL.PY)

= Equation 8

$P(x' | x, a_1, a_2)$

AKA transition_probabilities

get_transition_probabilities
(MODEL.PY)

= Eqns. 9, 11, and 12

- best_jammer_value AKA memfunc.get(1, ...)

$$V_2(x) = \max\Big(-\overset{1\times A}{f(x)}\, \overset{A\times S}{R(x)} + \delta\, \overset{1\times A}{f(x)}\, \overset{A\times S}{T(x, V_2)} \Big)$$