1. Blue팀 전투 Flow chart

Fail

Success

No

Yes

전투 종료

Red팀의 병력이 0인가?

사격

표적획득

2. Simulation Code (Python)

import random as rd  
  
global B\_tank  
global B\_antitank  
global R\_tank  
global R\_antitank  
global War  
  
B\_tank=[1,1,1,1,1] # 1  
B\_antitank=[1,1,1] # 2  
R\_tank=[1,1,1,1,1,1] # 3  
R\_antitank=[1,1,1,1] # 4  
War=[0,B\_tank,B\_antitank,R\_tank,R\_antitank]  
Hit\_table=[[0,0,0.4,0.5],[0,0,0.2,0],[0.3,0.3,0,0],[0.1,0,0,0]]  
event=[]  
# element of event : [time,[attack group, attack member], [targeted group, targeted member]]  
def add(event,x):  
 for i in range(len(event)):  
 if x[0]<event[i][0]:  
 event.insert(i,x)  
 return event  
 event.append(x)  
 return event  
  
def target(a):  
 if a==1:  
 A=sum(R\_tank)  
 B=sum(R\_antitank)  
 C=A+B  
 rand=int(C\*rd.random())  
 if rand <= A-1:  
 for i in range(len(R\_tank)):  
 if R\_tank[i]==1:  
 if rand==0:  
 return [3, i]  
 else:  
 rand -= 1  
 else:  
 rand -= A  
 for i in range(len(R\_antitank)):  
 if R\_antitank[i]==1:  
 if rand==0:  
 return [4,i]  
 else:  
 rand -= 1  
 elif a==2:  
 A=sum(R\_tank)  
 rand = int(A \* rd.random())  
 for i in range(len(R\_tank)):  
 if R\_tank[i] == 1:  
 if rand == 0:  
 return [3, i]  
 else:  
 rand -= 1  
 elif a==3:  
 A=sum(B\_tank)  
 B=sum(B\_antitank)  
 C=A+B  
 rand=int(C\*rd.random())  
 if rand <= A-1:  
 for i in range(len(B\_tank)):  
 if B\_tank[i]==1:  
 if rand==0:  
 return [1, i]  
 else:  
 rand -= 1  
 else:  
 rand -= A  
 for i in range(len(B\_antitank)):  
 if B\_antitank[i]==1:  
 if rand==0:  
 return [2,i]  
 else:  
 rand -= 1  
 elif a==4:  
 A=sum(B\_tank)  
 rand = int(A \* rd.random())  
 for i in range(len(B\_tank)):  
 if B\_tank[i] == 1:  
 if rand == 0:  
 return [1, i]  
 else:  
 rand -= 1  
 return [0,0]  
  
def targeting\_time(a):  
 rand=rd.random()  
 if a==1:  
 if rand<0.5:  
 return 3+(2\*rand)\*\*(0.5)  
 else:  
 return 5-(2-2\*rand)\*\*(0.5)  
 elif a==2:  
 if rand<0.5:  
 return 2+3\*(2\*rand)\*\*(0.5)  
 else:  
 return 8-3\*(2-2\*rand)\*\*(0.5)  
 elif a==3:  
 if rand<0.5:  
 return 2+2\*(2\*rand)\*\*(0.5)  
 else:  
 return 6-2\*(2-2\*rand)\*\*(0.5)  
 elif a==4:  
 if rand<0.5:  
 return 4+(2\*rand)\*\*(0.5)  
 else:  
 return 6-(2-2\*rand)\*\*(0.5)  
  
def hitting\_time(a):  
 if a == 1:  
 return 2\*rd.random()+1  
 elif a==3:  
 return 2\*rd.random()+2  
  
  
# initial part  
for i in range(len(B\_tank)):  
 event.append([0,[1,i],[0,0]])  
for i in range(len(B\_antitank)):  
 event.append([0,[2,i],[0,0]])  
for i in range(len(R\_tank)):  
 event.append([0,[3,i],[0,0]])  
for i in range(len(R\_antitank)):  
 event.append([0,[4,i],[0,0]])  
  
  
# simulation part  
while True:  
 if War[event[0][1][0]][event[0][1][1]] == 1:  
 if event[0][2][0] == 0:  
 event = add(event, [event[0][0] + targeting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 prob = Hit\_table[event[0][1][0]-1][event[0][2][0]-1]  
 rand = rd.random()  
 if War[event[0][2][0]][event[0][2][1]]==0:  
 if event[0][1][0]==1 or event[0][1][0]==3:  
 event = add(event, [event[0][0] + hitting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 event = add(event, [event[0][0] + targeting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 if rand < prob:  
 War[event[0][2][0]][event[0][2][1]] = 0  
 print(sum(B\_tank),sum(B\_antitank),sum(R\_tank),sum(R\_antitank))  
 if event[0][1][0]==1 or event[0][1][0]==3:  
 event = add(event, [event[0][0] + hitting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 event = add(event, [event[0][0] + targeting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 if event[0][1][0]==1 or event[0][1][0]==3:  
 event = add(event, [event[0][0] + hitting\_time(event[0][1][0]), event[0][1], event[0][2]])  
 else:  
 event = add(event, [event[0][0] + targeting\_time(event[0][1][0]), event[0][1], event[0][2]])  
 event.pop(0)  
  
 if sum(B\_tank)+sum(B\_antitank)==0:  
 print("Red is win")  
 break  
 elif sum(R\_tank)+sum(R\_antitank)==0:  
 print("Blue is win")  
 break  
  
 if len(event)==0:  
 break

1. Variable

global B\_tank  
global B\_antitank  
global R\_tank  
global R\_antitank  
global War  
  
B\_tank=[1,1,1,1,1] # 1  
B\_antitank=[1,1,1] # 2  
R\_tank=[1,1,1,1,1,1] # 3  
R\_antitank=[1,1,1,1] # 4  
War=[0,B\_tank,B\_antitank,R\_tank,R\_antitank]  
Hit\_table=[[0,0,0.4,0.5],[0,0,0.2,0],[0.3,0.3,0,0],[0.1,0,0,0]]  
event=[]

B\_tank, B\_antitank, R\_tank, R\_antitank 의 각 숫자들이 1이면 병력의 생존, 0이면 병력의 사망을 뜻한다.

Hit\_table은 명중률의 table이다.

1. Function definition

def add(event,x):  
 for i in range(len(event)):  
 if x[0]<event[i][0]:  
 event.insert(i,x)  
 return event  
 event.append(x)  
 return event

표적을 정하거나 사격하는 event를 event라는 이름의 list에 시간 순서대로 배열한다.

def target(a):  
 if a==1:  
 A=sum(R\_tank)  
 B=sum(R\_antitank)  
 C=A+B  
 rand=int(C\*rd.random())  
 if rand <= A-1:  
 for i in range(len(R\_tank)):  
 if R\_tank[i]==1:  
 if rand==0:  
 return [3, i]  
 else:  
 rand -= 1  
 else:  
 rand -= A  
 for i in range(len(R\_antitank)):  
 if R\_antitank[i]==1:  
 if rand==0:  
 return [4,i]  
 else:  
 rand -= 1  
 elif a==2:  
 A=sum(R\_tank)  
 rand = int(A \* rd.random())  
 for i in range(len(R\_tank)):  
 if R\_tank[i] == 1:  
 if rand == 0:  
 return [3, i]  
 else:  
 rand -= 1  
 elif a==3:  
 A=sum(B\_tank)  
 B=sum(B\_antitank)  
 C=A+B  
 rand=int(C\*rd.random())  
 if rand <= A-1:  
 for i in range(len(B\_tank)):  
 if B\_tank[i]==1:  
 if rand==0:  
 return [1, i]  
 else:  
 rand -= 1  
 else:  
 rand -= A  
 for i in range(len(B\_antitank)):  
 if B\_antitank[i]==1:  
 if rand==0:  
 return [2,i]  
 else:  
 rand -= 1  
 elif a==4:  
 A=sum(B\_tank)  
 rand = int(A \* rd.random())  
 for i in range(len(B\_tank)):  
 if B\_tank[i] == 1:  
 if rand == 0:  
 return [1, i]  
 else:  
 rand -= 1  
 return [0,0]

target을 결정하는 system이다. 남아있는 상대의 잔존병력을 uniform distribution에 따라 선택한다. Return값인 [a,b]는 a group의 b번째 대상이다.

def targeting\_time(a):  
 rand=rd.random()  
 if a==1:  
 if rand<0.5:  
 return 3+(2\*rand)\*\*(0.5)  
 else:  
 return 5-(2-2\*rand)\*\*(0.5)  
 elif a==2:  
 if rand<0.5:  
 return 2+3\*(2\*rand)\*\*(0.5)  
 else:  
 return 8-3\*(2-2\*rand)\*\*(0.5)  
 elif a==3:  
 if rand<0.5:  
 return 2+2\*(2\*rand)\*\*(0.5)  
 else:  
 return 6-2\*(2-2\*rand)\*\*(0.5)  
 elif a==4:  
 if rand<0.5:  
 return 4+(2\*rand)\*\*(0.5)  
 else:  
 return 6-(2-2\*rand)\*\*(0.5)

양측 무기체계의 M, C 값에 따른 표적획득시간 및 사격시간이다. pdf를 통해 cdf를 구하고 그것의 inverse function을 이용하여 난수를 발생시킨다.

def hitting\_time(a):  
 if a == 1:  
 return 2\*rd.random()+1  
 elif a==3:  
 return 2\*rd.random()+2

uniform distribution에 따른 양측 전차의 사격시간이다.

1. Simulation

# initial part  
for i in range(len(B\_tank)):  
 event.append([0,[1,i],[0,0]])  
for i in range(len(B\_antitank)):  
 event.append([0,[2,i],[0,0]])  
for i in range(len(R\_tank)):  
 event.append([0,[3,i],[0,0]])  
for i in range(len(R\_antitank)):  
 event.append([0,[4,i],[0,0]])

Simulation code를 실행시키기 위한 사전작업이다. Event list의 element는 [time,[a,b],[c,d]이렇게 구성되며 time은 시간, a group의 b번째 구성원이 c group의 d번째 구성원을 공격한다는 의미이며(a와 c는 Blue tank, Blue antitank, Red tank, Red antitank가 각각 1,2,3,4의 값을 가짐) 표적의 값인 c가 0일경우 표적획득 과정을 수행하게 coding이 되어 있어 사전에 0값으로 지정했다.

# simulation part  
while True:  
 if War[event[0][1][0]][event[0][1][1]] == 1:  
 if event[0][2][0] == 0:  
 event = add(event, [event[0][0] + targeting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 prob = Hit\_table[event[0][1][0]-1][event[0][2][0]-1]  
 rand = rd.random()  
 if War[event[0][2][0]][event[0][2][1]]==0:  
 if event[0][1][0]==1 or event[0][1][0]==3:  
 event = add(event, [event[0][0] + hitting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 event = add(event, [event[0][0] + targeting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 if rand < prob:  
 War[event[0][2][0]][event[0][2][1]] = 0  
 if event[0][1][0]==1 or event[0][1][0]==3:  
 event = add(event, [event[0][0] + hitting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 event = add(event, [event[0][0] + targeting\_time(event[0][1][0]), event[0][1], target(event[0][1][0])])  
 else:  
 if event[0][1][0]==1 or event[0][1][0]==3:  
 event = add(event, [event[0][0] + hitting\_time(event[0][1][0]), event[0][1], event[0][2]])  
 else:  
 event = add(event, [event[0][0] + targeting\_time(event[0][1][0]), event[0][1], event[0][2]])  
 event.pop(0)

if War[event[0][1][0]][event[0][1][1]] == 1: 공격대상이 생존중일때만 event를 실행한다.

if event[0][2][0] == 0: 표적의 group값이 0일경우, add function을 이용하여 표적을 획득한다. 이때 시간은 targeting\_time, 표적 선정은 target함수를 이용한다.

Else: random.random() 함수를 이용하여 명중률인 Hit\_table에 따라 명중, 비명중을 결정한다. 명중 시 War list에서 명중 대상의 값을 0으로 지정하며 새로운 표적을 획득한다. 비 명중시, 다시 사격하는 event를 추가한다.

if sum(B\_tank)+sum(B\_antitank)==0:  
 print("Red is win")  
 break  
elif sum(R\_tank)+sum(R\_antitank)==0:  
 print("Blue is win")  
 break  
  
if len(event)==0:  
 break

한쪽진영의 병력의 합이0가 되거나 모든 event를 실행하면 simulation이 종료된다. (마지막 줄은 code에 이상이 있을 때만 실행된다.)

1. Result

print(sum(B\_tank),sum(B\_antitank),sum(R\_tank),sum(R\_antitank))

사격이 명중할 때 마다 잔존병력을 Blue tank, Blue antitank, Red tank, Red antitank 순으로 출력한다. Simulation 결과는 아래와 같다.

5 3 5 4

5 3 5 3

5 3 5 2

4 3 5 2

4 3 4 2

4 3 3 2

4 3 2 2

4 3 2 1

4 3 1 1

4 3 0 1

4 3 0 0

Blue is win

Excel을 이용하여 결과 graph를 그려보았다.

