Special Topics Advanced Intelligent Systems

Project Report

Name : Pooja Devarajan

Student Id : 1001553456

COALITION FORMATION

DESCRIPTION

The approach that I have used is entirely based on fairness where my program determines the Shapley values and calculates the maximum payoff the agent can get using a sub-coalition and states the difference.

EXPLANATION

- ➢ The necessary variables and lists are declared.
- ➢ User input : Number of agents and their resources.
- ➢ The number of agents is stored in a list format as here

Ex : For no of agents = 3 : [1,2,3]
For no of agents = 6 : [1,2,3,4,5,6]

Snippet :

```
for j in range(1,n+1):
        agent_list.append(agent)
        agent=agent+1
```

agent_list is the list containing all the agents.

- ➢ The resources are also stored in the same format so that they can be mapped with their respective agent.
- ➢ Now a grand coalition where distribution is based on fairness is calculated by calling the shapley() function for every agent.

Code Sample

```
for num in range(0,n):
        current_player=agent_list[num]
        shapley_result.append(shapley(current_player))
```

- ➢ We then calculate the maximum payoff an agent can get by forming a sub-coalition.
- ➢ Then the incentive was given as a difference between this maximum and the payoff e gets from grand coalition

Methods used in the program

- shapley()
- max_sub_coalition()
    - combination()

Shapley()

- The current agent's number is passed by value to the function so that the shapley value is calculated for that particular agent.
- A temporary variable to store the calculation for every step of the shapley formula is used.
- The for loop for **permute** goes through the various permutations of how a grand coalition can be found.
    - [(1, 2, 3, 4), (1, 2, 4, 3), (1, 3, 2, 4), (1, 3, 4, 2), (1, 4, 2, 3), (1, 4, 3, 2), (2, 1, 3, 4), (2, 1, 4, 3), (2, 3, 1, 4), (2, 3, 4, 1), (2, 4, 1, 3), (2, 4, 3, 1), (3, 1, 2, 4), (3, 1, 4, 2), (3, 2, 1, 4), (3, 2, 4, 1), (3, 4, 1, 2), (3, 4, 2, 1), (4, 1, 2, 3), (4, 1, 3, 2), (4, 2, 1, 3), (4, 2, 3, 1), (4, 3, 1, 2), (4, 3, 2, 1)]
- Then we loop through each permutation to get the average payoff an agent generates by coming into the coalition at any point of time.

```
#Shapley value calculation for one player at a time
def shapley(current_player):
    temp=0
    temp_array=[]
    for i in range(0,n):
        temp_array.append(0)
    for x in permute:
        current_list=x
        set_s=0
        total_res=0
        #print(current_list)
        for y in range(0,n):
            temp_res=int(res[int(current_list[y])-1])
            if int(current_player)==int(current_list[y]):
                if (total_res*1.0/sum_resource*1.0)*100*1.0<51:
                    if(((total_res+temp_res)*1.0/sum_resource*1.0)*100*1.0)>50:
                        temp=factorial(set_s)*factorial(n-set_s-1)*10000
                        temp_array[y]=temp_array[y]+temp

            else:
                set_s = set_s+1
                total_res = total_res+temp_res
    for i in range(0,n):
         temp_array[i]=int(temp_array[i]/(factorial(i)*factorial(n-i-1)))
    result=0
    result=sum(temp_array)/fact
    return result
```

- The payoff of the agent by entering in each of the positions in a grand coalition is calculated by looping through each permutation that can be formed.
- Shapley formula $$\phi_i(N,v) = \frac{1}{|N|!} \sum_{S \subseteq N \setminus \{i\}} |S|!(|N| - |S| - 1)! \Big[v(S \cup \{i\}) - v(S) \Big].$$

- This formula has been used in the code as shown.
- **Max_sub_coalition()**
- This function has been used to generate the maximum payoff each player can receive by forming sub-coalitions.
- **Combination() function** is called to generate the different sets of sub-coalitions that can be formed.
    - **Example :**
    - No of agents = 4
    - [(1,), (2,), (3,), (4,), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4), (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4)]
    - **#Finding the combinations**
    - def combination(agent_list):
        - return [c for i in range(1,len(agent_list)) for c in combinations(agent_list,i)]
- The maximum an agent can get by forming any of the above sub-coalitions is found.
- I have modified the shapley formula to calculate the maximum sub-coalition payoff by applying the concept of fairness.
- Thus, we loop through every combination using, **for x in combination_agent:** for every player.
- Once we do this we go through the combination list to find the current_agent and calculate his payoff by forming that coalition.

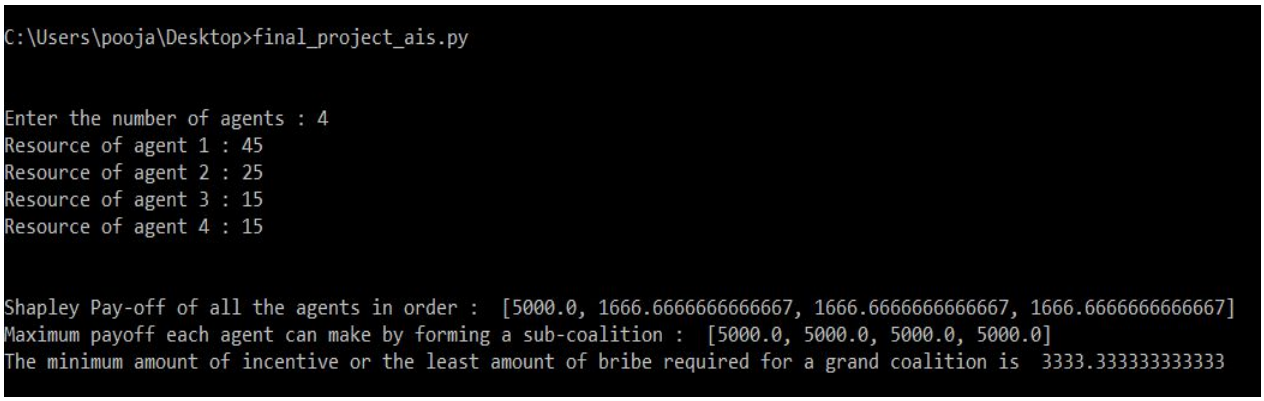
```
#calculating the maximum pay-off an agent can get from a sub-coalition
def max_sub_coalition(current_player_subcoalition):
    result=0
    for x in combination_agent:
        current_list=x
        set_s=0
        total_res=0
        temp_length=len(list(x))
        temp=0
        flag=0
        for i in range(0,temp_length):
            if int(current_player_subcoalition)==int(current_list[i]):
                flag=1
        if flag==1:
            for y in range(0,temp_length):
                temp_res=int(res[int(current_list[y])-1])
                if int(current_player_subcoalition)==int(current_list[y]):
                    if (total_res*1.0/sum_resource*1.0)*100*1.0<51:
                        if(((total_res+temp_res)*1.0/sum_resource*1.0)*100*1.0)>50 or current_player_subcoalition<=n-3 or n<=3 :
                            temp=temp+(factorial(set_s)*factorial(temp_length-set_s-1)*10000)
                else:
                    set_s = set_s+1
                    total_res = total_res+temp_res
                temp_result=temp/factorial(temp_length)
                if temp_result>result:
                    result=temp_result
    if current_player_subcoalition<=n-3:
        return result/(n-2)
    elif n<=3:
        return result/2
    else:
        return result
```

- temp_length has the temporary combination's length and is used as the N of the shapley formula.
- Hence the loop is only run for one combination at a time for a particular agent and every time a higher payoff is found the final result is replaced with the maximum thus giving the maximum payoff each player could get by forming a different sub-coalition.
- Now, the final **result** is returned as payoff

➢ Now after getting the respective utilities for each players respective to the coalition formed I've calculated an incentive based on the difference between the two utilities based on fairness.

RESULT

The output by running the code is given as a screenshot.



References:

http://www.masfoundations.org/mas.pdf

https://www.cs.ubc.ca/~kevinlb/teaching/cs532l%20-%202007-8/lectures/lect23.pdf

https://math.stackexchange.com/questions/111580/shapley-value-formula?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa