White Elephant Gift Exchange Problem

Junye Li & Matt Kretchmar
Department of Mathematics and Computer Science
Denison University
Granville, OH 43023
Email: li_j9@denison.edu, kretchmar@denison.edu

***Abstract*—White Elephant Gift Exchange (WEGE) is a party game that people play during Christmas holiday in Western civilizations. Our goal is to find the optimal strategy for playing the game and the average gift value for each player. The main approach is to use Dynamic Programming to compute the value of the game and find the optimal policy.**

I. Definition of the Game

White Elephant Gift Exchange is played widely in the West by different rules. Normally, there are n players playing this game and each player will bring a wrapped gift. Each player can recognize their own gift. Before the game starts, each player will be assigned a number which indicates the round that person goes first. For example, if someone gets 1, then he/she will go first in the first round. In each turn, a player will have two choice. He/She can either open a random gift or steal another player's gift. A round starts with the player who has been assigned the corresponding number goes first. The player will have two kind of choice. If the player choose to steal, then it will create a cycle until someone open an random gift. Therefore, each round will have many turns.
In order to avoid an infinite cycle, we assume that each gift can only be stolen once in each round. The round ends when someone chooses to open a wrapped gift.
Because we want to quantify the value, we make some modifications to the rules of the game.

1) There are n people playing game.
2) Each person brings one wrapped gift.
   a) The gifts have values 1, 2, 3, ... n
   b) Each person knows the values of the gifts but does not know which wrapped gift is which value.
   c) Each person has the same valuation for each gift.
3) There are n rounds in the game.
4) People are assigned start numbers: 1 to n.
   Person k goes first in round k.
5) Round:
   - Round 1: Person 1 must randomly select wrapped gift, since there are no gifts to steal.
   - Round 2: Person 2 can either steal Person 1's gift or open another random wrapped gift.
     If Person 2 steals Person 1's gift, then Person 1 must now open another wrapped gift.
   - Round k: Person k can either steal a gift of Person 1 to k-1 or select another random wrapped gift.
     If Person k steals Person m's gift(where $0 < m < k$), Person m can select a random wrapped gift or steal another people's gift, which hasn't been stolen in this round.
     A round will continue until someone chooses to open a wrapped gift.
6) In order to avoid infinite cycle, each gift can only been stolen once in each round.

II. State and Action Spaces

White Elephant Gift Exchange consists of series of states and actions. A state represents the status of each player in the game. Namely, the state indicates which player has which gift in the game. An action is a transition between two states. In general, each player will face two kinds of action. The player can either open a new gift or steal another player's gift.

A. State Space

The state represents the status of each player in the game. $[0, 0, 0, 0]$ is the start state for a game with four players where none of them has a gift. In our notation, we use 0 to represent the corresponding player does not have a gift. The state $[1, 4, 0, 0]$ represents the situation where Player 1 has a gift with value 1 and Player 2 has a gift with value 4. Also, both Player 3 and 4 do not have a gift yet. In the next round, Player 3 is the next person to play first, because the first zero is in the position for Player 3. Player 3 will choose to steal gift 4, because it has highest value. The next state will then be $[1, 0, 4*, 0]$. We use the * to show that the gift has been stolen in this round and can no longer be stolen in this round. Therefore, this state represents the situation where Player 1 has gift 1, Player 2 has no gift and Player 3 has gift 4 which cannot be stolen again in this round. Now, it is the Player 2's turn, since their gift is stolen.
The terminal states are a permutation of n numbers. In other words, there are no zeros and *s in the state. The terminal state means the end of the game and no more actions can be taken.
Now, we consider the size of state space. We count the number of states in round k by grouping them into two types: the intermediate states in round k and the states that round k ends.Let's consider when n=3,k=2. There are three intermediate states: $[0, 1*, 0], [0, 2*, 0], [0, 3*, 0]$. In end of Round 2, there are six states:$[1, 2, 0], [1, 3, 0], [2, 1, 0], [2, 3, 0], [3, 1, 0], [3, 2, 0]$.
Let's firstly consider the ending state. We suppose there are

n players playing this game and we are now playing in k round. In the end of kth round, all of k players have one gift of different value. Therefore, we will randomly choose k numbers form 1 to n in the order. Then, there are P_k^n kinds of way. Here $k = 2, n = 3$ and then $P_2^3 = 6$ which matches that there are 6 ending states.
Now Let's consider the intermediate states. If round k does not end, one player will have no gift and only $k-1$ gifts has been opened. We know that one player does not have gift, because his/her gift has been stolen. Also, we know that Player k, who is the first one to take action in round k, cannot be stolen. Then, $k-1$ players' gifts are possible to be stolen and there are P_{k-1}^n ways to choose $k-1$ gifts from n gifts in the order. Also, we assume that there are j gifts cannot be stolen in this round. As we know, the round will end until someone open a new gift. Therefore, if in the round, some gifts are stolen, Player k must steal gift from other player. If we have more than one gift has been stolen in this round, we have C_{k-2}^{j-1} number of arrangements that j gifts have been stolen.

Size of possible state space for a round:
Suppose $0 < k \leq n$, then the formula of the size of possible state space in round k is: $P_k^n + \sum_{j=1}^{k-1}(k-1)P_{k-1}^n C_{j-1}^{k-2}$

- P_k^n: In the end of each round, the number of possible state, because we choose an order that which represent k gift are open.
- the summation represent the total number of intermediate state that with * in the round k, where j represents the number of *
- $(k-1)$:the possible spots to pick 0, because the k th position represent Player k who will go first in round k and it is impossible that he does not have gift. Therefore, $k-1$ players are possible to have no gift.
- P_{k-1}^n: pick $k-1$ number of gifts from n gifts for k people (one person has 0 for now)
- C_{k-2}^{j-1}: the number of locations for *
  - $k-2$: possible places to put a *, because Player k always get a * and there are at most k-1 player get a *, because if k players have *, then all of k players have a gift but all * should been removed.
  - $j-1$: because last person always get a *, so we need to find j-1 location for other *s.

For example, if there are 3 players and it is round 2, then $k = 2$ and $n = 3$. Since it is in round 2, only Player 1's gift can be stolen. Then there are 1 position to put 0. If Player 1's gift has been stolen, only Player 2 will get the gift. Player 2 will have 3 possible gift. Similarly, since only player can steal gift is Player 2, we will have 1 position to put the *. Therefore, there are three possible states: $[0, 1*, 0], [0, 2*, 0], [0, 3*, 0]$. Then, in the end of the round, Player 1 and 2 will have 2 gifts which are randomly choose from 3 gifts. These states are $[1, 2, 0], [1, 3, 0], [2, 1, 0], [2, 3, 0], [3, 1, 0]$, and $[3, 2, 0]$. Therefore, if there are 3 players in round 2, there are 9 possible states in total.

Number of players	Size of state space
1	2
2	7
3	43
4	405
5	4971
6	72643

Table 1. Size of state space

From the table above, we can see that with the number of players increases,the size of state space increase exponentially.

B. Action Space

Action is the transition that happens between two states. In general, each player will face two actions. The player can either open a new gift or steal another player's gift. The steal action will only have one outcome. However, the open action is a stochastic event, because all the gift are wrapped and players do not know the value of the game. Therefore, besides action, we will also want to know the action outcome. Action outcome is the outcome after a player makes some action. For example, if Player 1 has gift 2 and Player 2 and 3 do not have gift, which means we are now at state $[2, 0, 0]$, then Player 2 needs to take an action. Player 2 can either open a random gift or steal gift 1 from Player 1. Then, the next state is $[0, 2*, 0]$. If Player 2 choose to open a random gift, then it will have two possible random outcome–Player 2 can get gift 1 or gift 3. If Player 2 open the gift 1, the next state is $[2, 1, 0]$. Otherwise, the next state is $[2, 3, 0]$. Therefore, there are two possible actions and three possible action outcomes.
Notation:
$O(p, g)$: person p opens gift g
$S(p, g, q)$: person p steals gift g from person q.

C. Trajectory

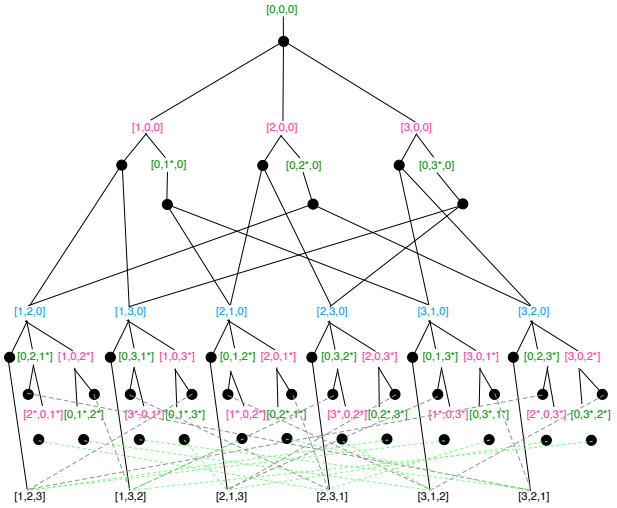
1) Definition:sequence of states and actions that moves the game from the start state to end state :$[0, 0]\overrightarrow{}_{a)} O(1, 1)\overrightarrow{}_{b)} [1, 0]\overrightarrow{}_{c)} S(2, 1, 1)\overrightarrow{}_{d)} [0, 1*]\overrightarrow{}_{e)} O(1, 2)\overrightarrow{}_{f)}$ $[2, 1]$, which represents
   a) Player 1 opens the gift 1.
   b) Player 1 haw gift 1.
   c) Player 2 steals the gift 1 from Player 1.
   d) Player 1 has no gift and Player 2 has gift 1 which cannot be stolen in this round.
   e) Player 1 opens the gift 1.
   f) Player 1 has gift 2 and Player 2 has gift 1.
2) Size of trajectories

Number of players	Size of Trajectory
1	1
2	4
3	60
4	3840
5	1248000
6	2441088000

Table 2. Size of Trajectory

We can see from the table that the size of trajectories goes faster than size of state space with the number of players increase.

III. Tree Diagram



As the figure shows above, it simulates all the possible trajectories that three players are playing in the game. The black dot in the tree diagram means the player is randomly opening a gift. Also, we use the color to distinguish different players. The green state means Player 1 is making an action, pink is Player 2 and blue is Player 3.

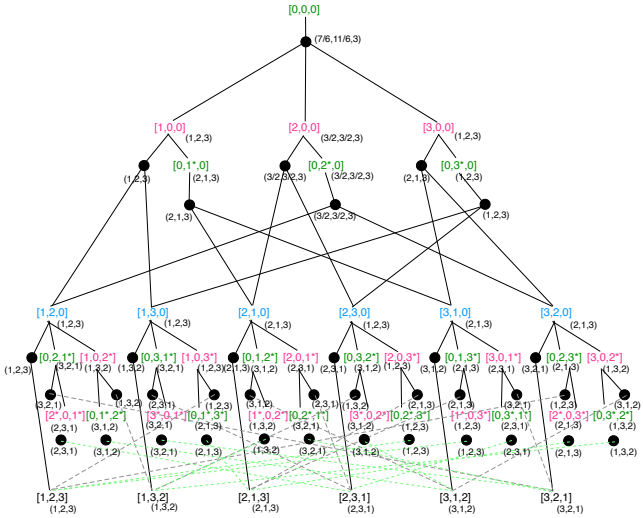
The game starts at the start state $[0,0,0]$ and Player 1 has no choice and he can only open a random gift. Then, he will have 3 possible state, since he has $\frac{1}{3}$ chance to get gift 1,2,3. We suppose that he get gift 1. Then, it ends with state $[1,0,0]$. Player 2 now needs to make actions. He will have two choices: open a random gift or steal gift 1 from Player 1. We assume Player one a random gift and get gift 2. Then, Player 3 is going to make action and he can steal player1 or 2's gift or open a random gift. We assume Player 3 open a gift 3. Therefore, we end up with the state $[1,2,3]$

IV. Value of the state

The aim of this work is to calculate the value and the optimal policy for each state. We firstly use Dynamic Programming to get the accurate value for each state. However, we find that with the number of player increasing, the time it takes to calculate the value becomes longer. Then, we use Reinforcement Learning to get the value of each state.

A. Definition of value

The value of a state represents expected value you would see that follow some policies. The value of state can help us know the reward that we could get by taking the corresponding action. In the game, we assume the value of the sate will depends on the the value of the terminal state. In the other words, we will do not have the intermediate reward.



The tree graph above shows the game with 3 players in total.We calculate the value of each state. The value of the state store the value of state for n players.

$[1,2,3]$ is a terminal state and it means Player 1 get gift 1, Player 2 get gift 2 and Player 3 get gift 3. So, it is obvious that Player 1 will get value 1,Player 2 will get value 2,and Player 3 will get value 3.

$[0,2*,3]*$ is a state that Player 2 has gift 2 and Player 3 has gift 3. None of their gift can be stolen in this round. Therefore, Player 1 can only open the gift 1. Then, the value of this state will be the value of gift that Player 1 get is 1, Player 2 is 2 and Player 3 is 2.

$[0,2*,0]$ is a state the Player 2 has gift 2 and gift 2 cannot be stolen in this round. So, the only choice Player 1 has is to open a random gift. Then, there are two possible outcomes and when we calculate the value of this state, we will use the expected value of two possible next state. If Player 1 open the gift 1, the next state is $[1,2,0]$ with value $(1,2,3)$. Or Player 1 open the gift 3, the next state is $[3,2,0]$ with value $(2,1,3)$. Therefore, the value of this state is $(1.5,1.5,3)$.

$[1,0,0]$ is a state that Player 1 has gift 1 and Player 2 needs to take an action. Generally speaking, Player 2 will have 2 kinds of action. Player 2 can steal Player 1's gift and then the next state is $[0,1*,0]$. From the tree diagram, we can get that the value of the state is $(2,1,3)$. Otherwise, Player 2 can choose to open a random gift. Then, Player 2 has 50 percent chance to get gift 2 and 50 percent chance to get gift 3. From the plot, we can know that the value of the state $[1,2,0]$ is $(1,2,3)$ and that of $[1,3,0]$ is $(1,2,3)$. Then, Player 2 is faced with a maximization choice, because he wants to take an action that can maximize the value that he can get in the future. Comparing the value of two kinds of action, Player 2 will choose to open a random gift.

From the plot, we can see that Player 3 will get the highest value gift.

B. Definition of Policy

In order to learn the value of each state, we firstly should know what is the policy. Policy is a function maps state to action. Domain of this function is state space and range of

this function is action space. In our paper, we define the policy function and find the optimal function that gives us the best policy.

The best policy in our paper is the policy that will maximize the value of the state. For example, if we are at the state $[1, 0, 0]$, from the plot, we can know that the best policy is Player 2 open a random gift.

C. Dynamic Programming

In *Reinforcement Learning: An Introduction*, it defines Dynamic Programming as operate in sweeps through the state set, performing a full backup operation on each state. [1]

We use the Dynamic Programming to get the accurate value of each state. Firstly, it is clear that the terminal states will have reward which is the same value as their state. Then, we go backwards to update value of each state. If it is an open action, we will calculate the average value of next states. If it has two possible action, we need choose the action that has the higher value. However, Dynamic Programming has a very obvious drawback is the algorithm needs to calculate the value of every state. Therefore, if the size of state becomes very large, the time that algorithm run will become very long.

However, we find that if we want to calculate the value of each state with 8 players, it will take our few hours to run. Also, from Table 2, we can see the number grows very fast. Therefore, we need to find an alternative way to find the value of each state.

V. Future Work

In our paper, we have used the Dynamic Programming to find the value of the state and the optimal policy for each state. We also find that with the number of players increase, the time it takes to find the value becomes longer. If the number of players exceeds 10, it becomes impossible for us to track. Then, Reinforcement Learning will become an alternative way to help us find the optimal policy and the value.

A. Reinforcement Learning

Dynamic Programming provides an idea way for us to calculate the value of the state. However, it is not an efficient way to calculate. Then, we try another way to get the approximation of the value. Temporal Difference Learning and Eligibility Trace learns from the experience and update the value.

1) Temporal Difference Learning
   Temporal Difference learning is a combination of Monte Carlo ideas and dynamic programming (DP) ideas.[1]

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$;
  until S is terminal

Figure 1. Q-Learning [1]

The figure above shows the algorithm of Q-learning. At the first time we playing this game, it will randomly choose the first action. Then, it will observe the intermediate reward if we take that action. In our game, we will do not have intermediate reward. Then, it will update the value of the state with given action by adding the temporal difference. Temporal Difference is the difference between the maximize value of next state and the value of the current state with the given action.

2) Eligibility Trace
   Although Q-Learning can help us to get the approximation of the value, we think the algorithm is not efficient enough.

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
  $E(s,a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
  Initialize S, A
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $A^* \leftarrow \arg\max_a Q(S',a)$ (if A' ties for the max, then $A^* \leftarrow A'$)
    $\delta \leftarrow R + \gamma Q(S', A^*) - Q(S,A)$
    $E(S,A) \leftarrow E(S,A) + 1$ (accumulating traces)
    or $E(S,A) \leftarrow (1-\alpha)E(S,A) + 1$ (dutch traces)
    or $E(S,A) \leftarrow 1$ (replacing traces)
    For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta E(s,a)$
      If $A' = A^*$, then $E(s,a) \leftarrow \gamma\lambda E(s,a)$
                  else $E(s,a) \leftarrow 0$
    $S \leftarrow S'$; $A \leftarrow A'$
  until S is terminal

Figure 1. Eligibility Trace [1]

Acknowledgments

Thank you! This project was funded by the Laurie Bukovac Hodgson David Hodgson Endowed Fund. Endowment at Denison University, with additional support from Hodgson

References

[1] Sutton, R. S., Barto, A. G. (2018). Reinforcement learning: An introduction. Cambridge, MA: The MIT Press.

[2] White elephant gift exchange. (2020, June 01). Retrieved July 23, 2020, from https://en.wikipedia.org/wiki/White_elephant_gift_exchange

[3] Introduction to Dynamic Programming 1 Tutorials Notes: Algorithms. (n.d.). Retrieved July 23, 2020, from https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/tutorial