Finding Optimal Play of Chopsticks Hand Game Using Best First Search Algorithm and Memoization

Raden Rifqi Rahman - 13520166
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520166@std.stei.itb.ac.id

***Abstract*—Chopsticks is a turn-based game using both of player's hands to play. It is played by tapping opponent's player hand to increase their hand value or fingers. Players start with one finger on both hands. The goal of the game is to make both opponent's hands "dead". Chopsticks has many variants, one of which is rollover in which the dead hand is defined by having exactly five fingers after being tapped. There are 225 functionally distinct positions in this variation. Using best first search algorithm and memoization, we can evaluate every reachable position in the game either as drawable, winning, losing, or lost. By evaluating all positions in the game, we gain the knowledge whether a player can force a win, or it is always a draw by perfect play. Additionally, we discover the optimal play of the game at any given position. Our evaluation algorithm shows that out of 225 functionally distinct positions, only 207 of them are reachable. As for these 207 reachable positions, 100 of them are drawable, 72 are winning, 21 are losing, and 14 are lost by perfect play by both players.**

Keywords—Chopsticks, best first search, memoization.

I. INTRODUCTION

Chopsticks is a turn-based game using both of players' hands to play. According to [1], it is originated from Japan and is commonly played by two players. However, it has spread to other countries outside Japan and can be played by more than two players. Chopsticks is played using players' fingers indicating the *value*–or *chopsticks*–they have in each of their hands. Therefore, if a hand has three fingers raised, the hand is said to have a value of 3. As consequences, a hand cannot have a value of more than 5. More strictly, the set of rules enforces any hand to not have a value of more than 4.

There are no commonly agreed official rules to the game. However, a various description of chopsticks described by children conducted in a study [2] agrees that chopsticks is played by tapping one's hand to another player hands. Upon tapping, the tapped player's hand value is added by the value of the tapper's hand. For example, if player *A* with a hand value of 2 taps player *B* with a hand value of 1, then player *B*'s hand will have a value of 3 while player *A*'s remains 1. Another common rule is that a player starts with value 1 on both of their hands, meaning that both players raised one finger on both hands at the start of the game, as defined in [1]–[3]. Fig. 1 shows the starting hands of all players in the game.

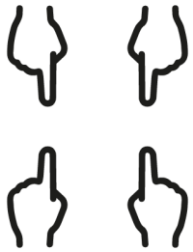

Fig. 1. Chopsticks starting hands.

Since there are no official rules, several variants exist in this game. Despite having many variants, the goal remains the same. A player wins if the other players are out of the game. A player is out of the game if both of their hands are "dead". A hand is said to be dead if it does not have any fingers raised. While the various sources approve the same definition of dead hand, reference [2] shows a different scenario of when a hand is considered dead as opposed to [1] and [3]. One of the children [2] claims that a hand remains in the game if its value is more than 5. For instance, if a hand with a value of 4 taps a hand with a value of 2, then it should have a value of 6. Since no hand can have more than 5 fingers raised, the value of 6 is then deducted by 5, leaving only 1 value left on the hand. Thus, four fingers hand tapping two fingers hand results in a one finger hand. In addition, a hand will be considered dead if it has an *exact* value of 5. In contrast to this claim, [1] and [3] explain that having more than five fingers will make the hand dead regardless of any leftover fingers. Despite the differences, [1] and [3] acknowledge the claim in [2] as a variant called *leftovers* or *rollover*. Furthermore, both rules confirm that no hand can have five fingers raised at any time, meaning that no hand will ever have more value than 4.

Aside of tapping other player's hand, a turn in chopsticks can be played by *moving* a finger or more to another hand. As well as the dead hand rule, the various sources define the moving rule differently. According to [1], a player may "tap their own hands together" and distribute the total fingers of both hands differently as they want. This means that two hands with two and four fingers may be rearranged as three and three fingers. Nonetheless, as demonstrated in [2], player can "split" their fingers to both of their hands if only they have one dead

hand in the game, resulting to "bringing back" the dead hand alive.

The set of rules and variants may seem rather complicated to new players who never played the game before. Having said that, the game is quite simple and straightforward. Mathematically, there are 625 possible positions with only 225 being functionally distinct positions. This begs the question that whether we could force a win in the game either by going first or second with an optimal play, or not.

II. METHODS

A. Best First Search Algorithm

Best First Search is one of various graph traversal algorithm available. This algorithm utilizes a heuristic function to find a specific node of state-space graph quickly. Pearl [4] described a *heuristic evaluation function f*(*n*) to numerically estimate the promise of a node, which "may depend on the description of *n*, the description of the goal, the information gathered by the search up to that point, and most importantly, on any extra knowledge about the problem domain".

In our case, a node *n* represents a specific state of the game, whereas the goal of the search is to find the optimal play of the game. However, it is not so clear of how we can find an optimal play. We may define an optimal play as playing the best possible move at any given time. Nonetheless, searching a specific node within the state-space graph of the game indicates searching for a specific possible state of the game, while what we are searching for is the move instead of the state. Hence, we shall later see our approach to find the optimal play using best first search algorithm.

B. Memoization

Memoization is the concept where a function is made so that it remembers the result of its previous computation [5]. Norvig [5] states that the basic idea of memoization is to store the previously computed input and result pairs into a table. With a table storing all previously computed input and result, referred to as a *memo*, we can easily retrieve the result of a function with previously computed input.

C. Optimal Play Approach

A state-space graph is defined as all the reachable *states* from the starting position after a sequence of moves played in the game. We refer to a state as such reachable position. For simplicity, we will notate a state in the format of AB-XY with A and B being the number of fingers in the left hand and right hand of the *player to move*, respectively, and X and Y being the number of fingers the other player has. As an example, the position shown in Fig. 1 is notated as 11-11. In addition, we also refer a state or position as a *hand* or *hands*. Since the first two numbers denote the hands of the player to move, both pairs AB and XY are always swapped after a move.

Each reachable state in the game must be connected with at least one other state in one way or another. While a state may not lead to any other state, there still must be another state which leads to such state. In other words, a state may not have a *child* state, but always have a *parent* state. We refer to state *A* as a child of state *B* if there is any legal move can be played from state *B* that results in state *A*. Consequently, we refer to state *B* as the parent of state *A* in the same scenario.

All states within the state-space graph of chopsticks are connected with one another. As mentioned earlier, a state results in another state if we play a move. We refer to a move as any playable action according to the rules. This includes tapping other player's hand or *moving* a finger or more to another hand. To avoid confusion, we refer to the fingers *moving* action as a *transfer*. We may also call the tapping action as an *attack*. Recall that in [2], a player is said to "split" their fingers to bring dead hand back alive. Hence, we will also call a transfer to a dead hand as a *split* move to be more specific. In consequence, we may also call a transfer resulting in a dead hand as a *merge* move.

We need to find the state-space graph of the game to use best first search algorithm to find the optimal play. Therefore, we will first have to expand all reachable states from 11-11. However, due to all the variations and differences in rules, there may be different state-space graphs for different variants and rules. While the usage of the algorithm should be similar, we will specifically choose the rollover variation with all kinds of transfer allowed—including split and merge—for only two playing players. Thus, there are 8 different moves available in this variation, four of which are attack moves whereas the other four are transfer moves.

For convenience, we will refer to those attack moves as left-to-left (LTL), left-to-right (LTR), right-to-left (RTL), and right-to-right (RTR) depending on which attacker hand attacks which attacked hand. Moreover, we refer to the transfer moves as 1-finger-leftside (L), 1-finger-rightside (R), 2-fingers-leftside (L2), and 2-fingers-rightside (R2) depending on how many fingers are transferred and to which side those fingers are transferred. It is important to note that there is no 3-fingers transfer since it is only possible to do so with a 04 hand. However, transferring 3 fingers in a 04 hand will result in 13 hand which is equivalent to transferring only 1 finger as shown in Fig. 2(a) and 2(b). Similarly, there is no 4-fingers transfer since doing so with 04 hand will result in 40 hand which is the mirror of the initial hand. Because the rules state that player can only distribute their fingers differently, the 4-fingers transfer will always be an illegal move. We refer to these mirrored hands as *functionally indistinct* hands whereas two different hands which are not a mirror of each other are *functionally distinct*.

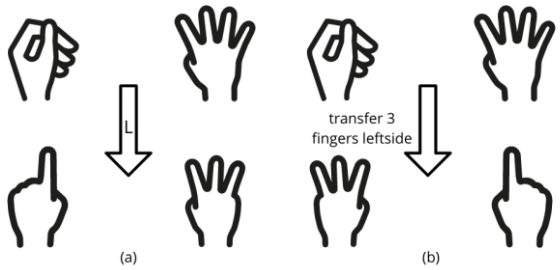

Fig. 2. Identical transfer moves. (a) Transfer 1 finger leftside on 04 hands. (b) Transfer 3 fingers leftside on 04 hands

As a result, we need to set a constraint of when each of the moves can be played. As for attack moves, they can only be played if both the attacker and attacked hands do not have a value of 0. Nonetheless, we add an extra constraint that RTL and RTR moves are not possible if both attacker player hands have an equal value because they result in the same state as LTL and LTR moves, respectively. Similarly, LTR and RTR moves are not possible if both attacked player hands have an equal value for similar reason. Consider the first state of the game. All attack moves result in the same state if they are played on the first state of the game. Regardless of which hand attack which hand, any attack move played at 11-11 state will result in 12-11 state. Therefore, we only allow LTL and disallow LTR, RTL, and RTR to keep every state functionally distinct in the state-space graph. As for transfer moves, it is obvious that R2 is only legal for 22 hand and L2 is only legal for 04 hand. Additionally, we set R and L as legal for as long as the transferring hand has more value than 0 and the transferred hand has less value than 4 and the resulting hand is functionally distinct with the initial hand.

With the state-space graph being expanded to completion, we still have to define the heuristic evaluation function $f(n)$ to choose one of the nodes or states n to prioritize the expansion. To define the function, we introduce a new variable called the *evaluation* of each state. In contrast with Pearl's [4] description of f, the *evaluation* is not necessarily numeric, but instead an enumerated value. We classify the evaluation of a state as either one of the following.

- Winning
- Losing
- Lost
- Drawn
- Unknown

A state is evaluated as *winning* if the player to move in such state can force a win regardless of what the opponent plays afterwards. Accordingly, a state is evaluated as *losing* if the opponent at such state can force a win with an optimal play regardless of what move played by the current player to move. A state is evaluated as *lost* if the player to move has both hands dead, meaning the game has ended. In contrast with previous evaluations, a state is evaluated as *drawn* if both players can play indefinitely with optimal play, meaning that neither player can force a win. Lastly, a state is evaluated as *unknown* if we cannot determine the real evaluation of the state yet. The unknown evaluation is the default evaluation for every state in the game, with an exception to those of lost states. Having these enumerated values, we then define f as the priority of an evaluated state to be expanded.

The evaluation function f we define earlier is the key to utilize best first search algorithm. Using this function, we are finally able to determine which node is the "best" to expand at any time. Still, we need to determine the priority of those enumerated values, which evaluation should be prioritized, and which should not be. In our approach, we order those values priority from the highest to lowest as being lost, losing, drawn, winning, and unknown.

While it seems like a random order, there is a reasoning behind this order. Before any expansion or evaluation, we know that a certain state must be evaluated as lost, regardless of what are their children states or parent states. A position is lost if both hands are dead, meaning the only states that are lost are in the form of 00-XY with X and Y being any possible number as depicted in Fig. 3. Thus, we already know which states are lost, whereas the rest remains unknown. Furthermore, we knew that the parents of lost and losing states must be winning states, regardless of what other children the parent has. For example, let *A* be a lost or losing state. If there is any state *B* such that there is a move from *B* which results in *A*, then the player can play the move and force their opponent to be in a lost or losing state, therefore losing the game. In consequence, the player having to move at state *B* is winning, thus evaluates state *B* as winning.

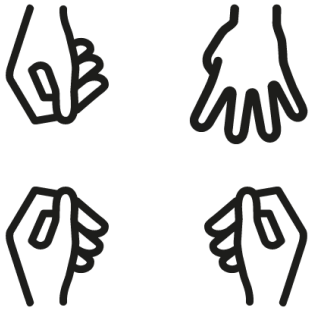

Fig. 3. Hands represented by 00-04 lost position.

In contrast with lost and losing states, drawn states guarantee their parents to be evaluated as drawn if and only if the parent does not have any children which is losing or lost. As the parents of lost and losing states are winning, it does not matter if they have a drawn child, they are still winning. Nevertheless, if none of the children are lost or losing, the parents of drawn states must be drawn. The reason being that if the parent is not winning, then it can force a draw by going to the drawn state. For instance, let *A* be a state which does not have any losing children, but has a drawn child *B*. Then the player to move at state *A* can force a draw by playing a move which results in state *B*.

Winning states are the second least prioritized in our case with the reason being that they require more calculation than the previous three. Aside of winning and drawn, the parents of winning states can also be losing, or even still be unknown. They are losing if and only if all their children are winning and are unknown if all previous conditions are not met. It is possible because if, say, a state has only winning children, then it must be losing because there is nothing the player to move can do to avoid a winning state for their opponent.

Lastly, the least prioritized evaluation is the unknown evaluation. As opposed to all the other evaluations, we do not expand unknown states to evaluate their parents. Instead, we evaluate the state so that it is no longer unknown. The idea of this approach is that we can not guarantee that all the unknown states will be evaluated after we expand their children since expanding winning states does not ensure their parents to not

be unknown. Therefore, for the unknown states remaining, we evaluate them as if they were a parent of other states, meaning that we check if they have a lost or losing child, or if they have a drawn child, and so on. However, evaluating the states instead of expanding them introduces inconsistency. From the previous evaluations and expansions, we may have evaluated a state as being drawn because it has a drawn child and not a lost or losing child. But we didn't know if it has an unknown child to be evaluated afterwards. That being the case, the unknown child may later be evaluated as losing. As a result, we incorrectly evaluate the state as drawn where it should be winning instead.

To address this issue, we need to reevaluate and re-expand all parents of every state we expand if the evaluation of the state is changed. Although it seems counter-intuitive and contradicts the purpose of heuristic evaluation function in [4], the reevaluation is necessary to obtain correct and complete result of the evaluations. Consequently, we need to memoize every state evaluation in order to determine whether their new evaluation differs from the old one. To simplify the evaluation, we do not use a table to memoize the previous evaluation, but rather "attach" the current evaluation of a state to the state itself. In addition to this simplification, we enforce the evaluation algorithm to reevaluate unknown states to not remain unknown and be at least drawn instead because they will be reevaluated regardless of what their actual evaluation is. Thus, the pseudocode to evaluate every state in the game is as follows.

```
build the state-space graph of Chopsticks,
  store in S
let Q be a priority queue with priority
  function f as defined
for each lost state P in S, enqueue P to Q
while Q is not empty:
  dequeue Q, store in R
  for each parent of R as P:
    reevaluate P
    if the evaluation of P is changed, enqueue
      P to Q
```

III. Results and Discussion

After applying the algorithm to evaluate all different possible states, we end up with a state-space graph containing all states which are evaluated as either lost, losing, winning, or drawn. Out of 225 functionally distinct positions we mentioned earlier, there are only 207 reachable positions in this variation of the game. This means that 18 of those positions are impossible to reach due to having no connection to other states. Hence, since the state-space graph connects every state with one another with their respective move, we obtain the information of which state results from which state. As a result, we can infer the optimal play by choosing the best possible move at any state.

A. Drawn States

Out of 207 reachable positions, our evaluations shows that there are 100 drawn positions in the game, as shown in Table I.

TABLE I. Chopsticks Drawn Positions

Drawn Positions				
02-02	02-11	02-12	02-14	02-22
02-23	02-33	02-34	02-44	03-13
03-14	03-22	03-24	03-44	04-02
04-03	04-11	04-14	04-22	04-23
04-24	04-33	04-34	04-44	11-03
11-11	11-13	11-22	11-23	11-24
11-44	12-11	12-12	12-23	12-24
12-34	12-44	13-11	13-13	13-22
13-23	13-24	13-33	13-34	13-44
14-02	14-11	14-12	14-13	14-14
14-22	14-23	14-24	14-33	14-34
14-44	22-02	22-11	22-12	22-14
22-22	22-23	22-33	22-34	22-44
23-11	23-14	23-22	23-24	23-34
23-44	24-14	24-22	24-23	24-24
24-33	24-34	24-44	33-04	33-11
33-22	33-24	33-33	33-34	33-44
34-14	34-22	34-24	34-33	34-34
44-03	44-04	44-12	44-13	44-14
44-22	44-23	44-24	44-33	44-34

We see that none of the drawn states player to play hands are 01, unlike the other hands. Likewise, none of the opponent hands are 01 either. On the other hand, we also find that 11-11, the starting hands, are drawn. That being so, we know that with the perfect play, both players can force a draw which in the context of chopsticks is to play indefinitely.

B. Winning States

There are 107 remaining non-drawn states in the game. We observe that most of these 107 states are winning. The evaluation result indicates that 72 out of 107 remaining states are winning, as shown in Table II.

TABLE II. Chopsticks Winning Positions

Winning Positions					
01-04	01-14	02-01	02-03	02-04	02-13
02-24	03-01	03-02	03-03	03-12	04-01
04-04	04-12	04-13	11-02	11-04	11-12
11-14	11-34	12-01	12-03	12-04	12-13
12-14	13-01	13-02	13-03	13-04	13-12
13-14	14-01	14-03	14-04	22-01	22-03
22-04	22-13	22-24	23-01	23-02	23-03
23-04	23-12	23-13	23-23	23-33	24-01
24-02	24-03	24-04	24-11	24-12	24-13
33-01	33-02	33-03	33-12	33-13	33-14
33-23	34-01	34-02	34-03	34-04	34-11
34-12	34-13	34-23	44-01	44-02	44-11

In contrast with drawn states, we observe that most of the 01 hands starts to show up. Out of 15 functionally distinct AB-01 hands, there are 12 which are winning. On the contrary, only 2 of 15 01-XY hands are winning.

C. Losing States

Our evaluation shows that there are 21 losing positions in the game out of the remaining 35 positions. Table III lists all 21 positions evaluated as losing.

TABLE III. CHOPSTICKS LOSING POSITIONS

Losing Positions						
01-01	01-02	01-03	01-11	01-12	01-13	01-22
01-23	01-24	01-33	01-34	01-44	03-04	03-11
03-23	03-33	03-34	11-33	12-02	12-22	12-33

It is noticeable that most of the losing hands are in the form of 01-XY, being more than half of all losing hands. Furthermore, it is also evident that 17 out of 21 losing hands has a dead hand. With that said, we can infer that there is quite a chance of losing when we have a dead hand at any time. Aside of losing positions, the winning positions also confirm this inference. As we observed, 41 of 72 positions states show the opponent hands having an empty hand.

D. Lost States

There are 15 functionally distinct hands for each player. Because lost states are in the form of 00-XY for any possible X and Y, there should be 15 lost positions in the game. Nonetheless, 00-00 position is impossible to reach, resulting in only 14 lost positions available, as shown in Table IV.

TABLE IV. CHOPSTICKS LOST POSITIONS

Lost Positions						
00-01	00-02	00-03	00-04	00-11	00-12	00-13
00-14	00-22	00-23	00-24	00-33	00-34	00-44

E. Optimal Play

By evaluating all reachable different states, we are finally able to figure out the optimal play for our variation of chopsticks. That being said, there are 207 different positions and 8 different moves which add up to roughly more than a thousand of position-move combinations, thus listing and remembering all the different combinations is basically impractical for casual player. Therefore, we need to simplify the best strategy or the optimal play of the game. Generally, the optimal play of the game can be classified by which classification of position we are at, as follows.

- If we are at a drawn position, play any move that results in another drawn position. This strategy keeps the game going indefinitely and avoids losing the game.
- If we are at a winning position, play any move that results in a losing position. This means we are forcing the opponent to lose the game regardless of what move they play afterwards.
- If we are at a losing position, play a move that results in a winning position such that it has the least losing children states or positions. Although it is hard, this strategy increases the chances of our opponent to play the wrong move.
- If we are at a lost position, there is nothing to be done since the game has already ended and we have lost the game.

Using the strategies above, we should be able to force a draw if we play the correct move every time. Nonetheless, we should also be able to not lose easily when we are in a losing position. Still, remembering hundreds or thousands of positions or moves is not an easy task. Therefore, we study the positions further to analyze whether there is a pattern in the positions.

Our previous strategy determines of what move to play at a given position. However, we can put a different perspective of the optimal play. Instead of determining the move to *play* at any position, we could also determine what moves to *avoid* at any position. As we observed, most of the 01-XY hands are losing as well as most of the AB-01 hands are winning. With the exceptions of 01-01 and 00-01 hands, all AB-01 hands are winning. Likewise, all 01-XY hands are losing with the exceptions of 01-04 and 01-14 hands. This discovery gives us a knowledge that we should avoid having 01 hands except if we can convert the position to a losing 01-01 position for our opponent.

Another discovery we should mention is that 11-02 position is winning even though it can be reached as early as only one move played into the game. On the first position of the game which is 11-11, we can merge our hands to go into 11-02 position. We must avoid playing the merge move at the beginning of the game since it will cost us to lose the game instantly.

By now, we discover what are the moves or positions to avoid. Yet, we still can get more knowledge from the winning positions. Table II shows us that the most winning hands are 23 and 34 each of which has 8 different winning position combinations. Moreover, any position with these hands is at least drawn by evaluation as shown in Table I and Table II. Consequently, in any given position, we should try to have these hands at any moment. However, we need to keep in mind that we cannot control our opponent moves to lead us to having such hands.

With more analysis of the positions and evaluations above, we obtain more simplified optimal strategy in addition to our obvious strategy as follows.

- Avoid merging both hands at the beginning of the game, i.e., avoid going from 11-11 position into 11-02 position.
- Avoid having 01 hands and try to force our opponent to have them instead. The only winning position with 01

hands is if we can convert the position to a 01-01 position.

- Any move is good to play at any position with 23 or 34 hands as we cannot lose having those hands.

IV. Conclusion

Chopsticks is a turn-based game using both of players' hands to play. Chopsticks can be played by 2 or more players. There are a lot of variants of chopsticks, one of which is the rollover variant which deduct the value by 5 for any hand having more value than 5. For this specific variation of chopstick with only 2 players, the game is drawn by the optimal play, i.e., both players can play indefinitely and without losing the game.

The optimal play of the rollover variant is to avoid losing positions and go for drawn or even winning position if it is possible. There are 207 reachable functionally distinct positions which consist of 100 drawn positions, 72 winning positions, 21 losing positions, and 14 lost positions. To play the game optimally, player must play any move that results to a drawn position in a drawn position, play any move that results to a losing position in a winning position, or play a move that has the least losing chances in a losing position. In addition to this obvious strategy, player needs to avoid merging both hands at the first move, avoid having 01 hands and force them to the opponent instead, and try to reach 23 and 34 hands.

Video Link at Youtube

A more detailed explanation of the methods, approaches, and results is available to watch at https://youtu.be/FPrqtfIU_sM.

Acknowledgment

First and foremost, I would like to praise and thank God, the Almighty, who has granted me countless blessing, knowledge, and opportunity to finish this paper. I would also like to express my gratitude to all lecturers of IF2211 Strategi Algoritma who encourage me to write this paper at the first place. Last but not least, I would also like to thank my family who supports me throughout my life.

References

[1] "Japanese games – Chopsticks (hand game)." s_A_k_U_r_A's zasshi. Accessed on: May 07, 2022. [Online]. Available: https://sakuraentorizasshi.wordpress.com/2008/10/16/japanese-games-chopsticks-hand-game/

[2] Childhood, Tradition & Change. "Chopsticks - Miscellaneous Physical Play." Childhood, Tradition and Change. Accessed on: May 07, 2022. [Online]. Available: https://ctac.esrc.unimelb.edu.au/biogs/E000224b.htm

[3] Activity Village. "Chopsticks Game." ActivityVillage.co.uk. Accessed on: May 07, 2022. [Online]. Available: https://www.activityvillage.co.uk/chopsticks-game .

[4] J. Pearl, "Basic Heuristic-Search Procedures," in *Heuristics: intelligent search strategies for computer problem solving*, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984, ch. 2, sec. 3, pp. 46–56.

[5] P. Norvig, "Techniques for Automatic Memoization with Applications to Context-Free Parsing," Comput. Linguistics, vol. 17, no. 1, pp. 91–98, Mar. 1991, doi: 10.5555/971738.971743.

Statement

I certify that this paper is my own work and is not plagiarized, based on my personal study and research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication.

Bandung, May 20, 2022

13520166 Raden Rifqi Rahman