The Application of Finite Combinatorics and Number Partition Theory in Determining Valid Poker Hand Combinations in Air Poker Game from Usogui

Raffi Fauzi Hermawan - 13525054
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: raffifauzihermawan009@gmail.com , 13525054@std.stei.itb.ac.id

***Abstract*—This paper presents a formal computational approach to determining valid poker hand combinations in Air Poker, a poker variant from the manga *Usogui* in which players choose steel cards whose engraved values represent the sum of the ranks of five cards drawn from a shared deck. The core problem is modeled as a bounded integer partition problem given a target value, find all five-card subsets from the remaining deck whose sum of ranks equals that target, and identify the optimal hand among them. The algorithm implements an exhaustive branch-and-bound search over combinations, using lower and upper bound pruning to avoid redundant enumeration. The search is verified as exhaustive, as the total number of valid combinations across all targets in a new 52-card deck equals the total number of possible 5-card combinations from that deck. Replication experiments using historical matches from the manga confirmed an exact match in four out of five rounds, with the only mismatch analytically explained by the algorithm's greedy nature. Further distribution-controlled experiments confirmed that the valid targets are strictly bounded within a closed interval and form a symmetric distribution, with a peak occurring at the midpoint of the range.**

Keywords—combinatorics, number partition, restricted integer partition, poker, Air Poker, Usogui, branch-and-bound, depth-first search

I. Introduction

Air Poker is high stake variation of the traditional poker game introduced in thriller manga *Usogui.* In this game, two players are submerged in a pressurized water tank and must bet using currency that takes form of a tank that takes form of a coin, that is also their only oxygen reserves, while also deducing valid poker hand from the deck. Unlike traditional poker where players are given five cards from the 52-card deck, *Air Poker*, forces player to dynamically construct card combinations by assigning specific integer values to several partitions that determine both the rank and suit of the cards. Not only does this game focus on the bluffing and psychological resilience, but also rapid mental calculation, making it a game that demands thinking capabilities.

The complexity of the game is governed by strict constraints, including the requirement that the card value must be derived from unique combination of the available card without repetition. Although the game is based on the basic framework of poker hierarchy, evaluating the distribution of possible combinations involves complex knowledge on restricted sets of integers, which is not easily understood intuitively or calculated quickly under time pressure.

This paper presents an alternative formal approach of verifying and optimizing hand combinations using restricted combinatorics and number partition theory as structural frameworks. By decomposing the universal game state and constructing restricted integer partition, we model how raw digital sequence is weighted according to its mathematical structural complexity, which reflects the combinatorial difficulty of determining specific card combinations under various constraints. The partition based is intended to provide a more rigorous, transparent, and algorithmic method for evaluating game options and strategic paths within the operational rules of Air Poker.

II. Theoretical Framework

A. Combinatorics

Combinatorics is a branch of math to count the sum of every possible objects arrangements or selection without the need to enumerate every single possible scenario.

a. Basic Principles of Counting

The basic principle of combinatorics is based on the two principles of counting, the rule of product and the rule of sum. The rule of product means that if there is a way of doing something and b ways of doing another thing, there are a x b way of performing both actions.

On the other hand, the rule of sum means that if we have a way of doing something and b ways of doing another thing and we can't do both at the same time, then there are a + b ways of choosing one of the actions.

b. Principle of Inclusion and Exclusion

Principle of Inclusion and Exclusion (PIE) is a counting technique in which generalized the familiar method of obtaining the number of elements in union or two finite sets, expressed as:

$$|AUB| = |A| + |B| - |A \cap B|$$

c. Permutations

A permutation is the number of different arrangements of objects. A permutation of r elements from n elements is the number of possible arrangements of r elements selected from n elements, where r ≤ n, such that, in each possible arrangement, no two elements are the same.

$$P(n,r) = n(n-1)(n-2)...(n-(r-1)) = \frac{n!}{(n-r)!}$$

Where :

- n = total number of items
- r = number of items selected in order

d. Combinations

A special type of permutation is a combination. While in a permutation the order of appearance is taken into account, in a combination the order of appearance is ignored. A combination of r elements from n elements, or C(n, r), is the number of unordered selections of r elements taken from n elements.

$$\frac{n(n-1)(n-2)...(n-(r-1))}{r!} = \frac{n!}{r!(n-r)!} = C(n,r)$$

Where :

- n = total number of items
- r = number of items selected

e. Combinations with Repetition

Combinations with repetition is a type of combination which allows you to choose objects from a set where order does not matter and the same item can be chosen more than once,

$$C(n+r-1, r) = C(n+r-1, n-1)$$

Where :

- n = total number of items
- r = number of items selected

f. Number Partition

Number partition theory is a way of writing a non-negative integer *n* as a sum of positive integer. For example, the integer 3 can be partitioned as 3, 2+1, and 1 + 1 + 1.

B. Poker

Poker is a game of card that dates to the sixteenth century with the initial name "Pochen" that eventually developed into the French version "Poque" and then brought to New Orleans to be refined further as "Poker" that we know today.

a. Cards

A standard poker deck consists of 52 cards, which are divided into four suits: hearts, spades, clubs, and diamonds. Each suit comprises numerical cards ranging from 2 to 10, three face cards: Jack (J), Queen (Q), and King (K), and the Ace (A). Individually, the cards are ranked from lowest to highest power: 2 through 10, J, Q, and K, whilst Ace on the other hand can be the lowest or the highest power depending on the other cards on the hand.

b. Hands

Based on the available cards, players can form a 'Hand,' which consists of a 5-card combination. Because a hand can be constructed from numerous distinct card combinations, they are strictly evaluated based on a standard hierarchical ranking. The various hands are ranked from highest to lowest value as follows:

TABLE I: Poker Hand Value

No.	Hand	Requirement
1.	Royal Flush	Five cards of the same suit in sequential order: A, K, Q, J, and 10.
2.	Straight Flush	Five cards of the same suit in sequential order, excluding the royal flush.
3.	Four of a Kind	Four cards of the same rank, accompanied by one unrelated card.
4.	Full House	Three cards of one rank, combined with two cards of another rank.
5.	Flush	Five cards of the same suit that are not in sequential order.
6.	Straight	Five cards in sequential order containing more than one suit.
7.	Three of a Kind	Three cards of the same rank, accompanied by two unrelated cards.
8.	Two Pairs	Two cards of one rank, combined with two cards of another rank and one unrelated card.
9.	One Pair	Two cards of the same rank, accompanied by three unrelated cards.
10.	High Hand	Any combination of five cards that does not meet any of the requirements listed above.

C. Air Poker

The "Air Poker" is a betting game played on the climax of the "Protoporos" arc from the manga "Usogui" developed by referee Shion Nawa and Takumi Manabe in which Madarame Baku, a *Kakerou* member, accompanied by Hal, and Vincent Lalo, the leader of the organization *Ideal,* accompanied by Fukurou, played to gain the right to play "Surpassing the Leader" as the requirement to take over the *Kakerou* organization. It is a variation of the usual poker game, but with a few tweaks. The author is going to only focus on the card

game itself and isn't going to include other details such as the betting system.

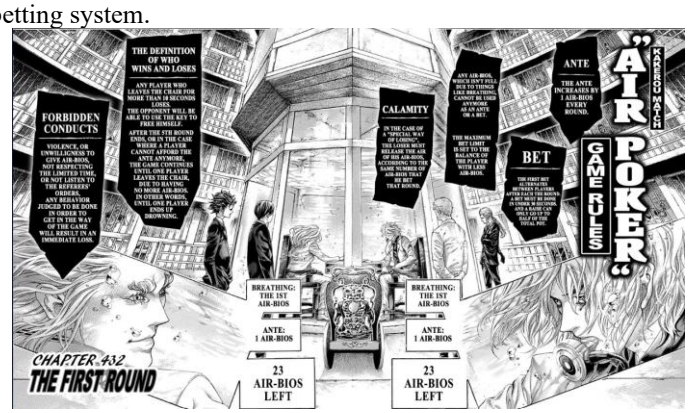

Fig. 1: Air Poker
source: Usogui Chapter 432

a. Cards

Each player is given 5 cards each. But, instead of the usual poker card, the players are given steel cards with number engraved in them, ranging from the minimum of 6 and the maximum of 64.

Fig. 2: Steel Card
source: Usogui Chapter 431

b. Gameplay

Each player placed their desired card on the table. Then, the betting phase begins which the author wouldn't dwell on. After that, the result on which hand won is revealed. The winner is chosen based on these rules

1. The number inscribed on the card represents the sum of the value of the cards used to make a particular set of poker hands. For example, the lowest possible hand combinations are the number "6" which can be made from four aces and one 2 which would make a four-of-a-kind hand. Notice that the steel card couldn't be 5 or lower and 64 or higher since we only have four aces.
2. The hand created to fit a value must be created from the remaining card in the deck. Based on the previous explanation, we can't use any aces anymore on the remaining rounds after we played the card "6". The usual number of cards discarded are 10, but in special cases in which both hands need the same card, that overlapping card is only discarded once. So, if there are two overlapping cards from each hand, then only 8 cards are going to be discarded.

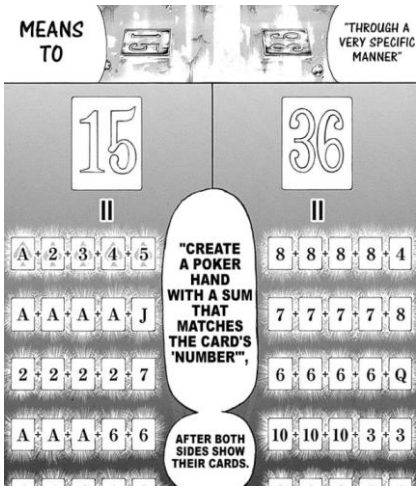

Fig. 3: Gameplay Explanation
source: Usogui Chapter 437

c. How the Hand is Made

This game is not only played by two players, but four players which are going to be divided into two types for each team, the "Lower" player which are going to choose the steel cards, and the "Upper" player which are going to choose the hand that is possible to make based on the number engraved on the steel card that the lower player played, which usually are the best hand possible from the remaining deck.

Fig. 4: Two Players for Each Team
source: Usogui Chapter 450

III. METHOD

A. Limitation

The author is going to limit the scope of this experiment so that we can focus on the main theme of the experiment, which is the usage of combinatorics to determine valid hand. Because of that, the author is going to state some limitations:

- The author is going to only focus on the card game itself and are going to exclude the betting system entirely
- The hand made that are going to be made from each steel card are always going to be the best hand

possible. In the manga, there's a time where the upper player doesn't make the best possible hand to ensure the trump hand are still available in the next round.

- Each player is going to favor different suits to minimize the overlapping card. For example, player A is going to prioritize heart, spade, diamond, and then club whilst player B is going the exact opposite.

B. *Formal Problem Statement*

The game state of Air Poker is governed by a dynamic universal deck D, modeled as a finite multiset. Initially, |D| = 52, where each card c in D is an ordered pair:

$$c = (\text{rank}, \text{suit})$$

Such that:

- $\text{rank} \in \{1, 2, \ldots, 13\}$ (where Ace = 1)
- $\text{suit} \in \{\text{Heart}, \text{Spade}, \text{Diamond}, \text{Club}\}$

```c
typedef struct {
    int rank; /* 1-13 (or 14 in special case) */
    int suit; /* 1-4 */
} Card;

typedef struct {
    Card combo[HAND_SIZE];
    HandRank rank;
    int  found;
    long count;
} SearchState;
```

Fig. 5: Data Structure Representing the Card, the Search Result, and the Optimal Hand (Combo)

source: Author's archive

```c
Card deck[DECK_SIZE];
int  deck_n = 0;
for (int r = 1; r <= NUM_RANKS; r++)
    for (int s = 0; s < NUM_SUITS; s++) {
        deck[deck_n].rank = r;
        deck[deck_n].suit = s;
        deck_n++;
    }
```

Fig. 6: Initialization of the Universal Deck D

source: Author's archive

A steel-card target value t is valid if and only if there exists a 5-element subset H ⊆ D (|H| = 5) where H denotes a single poker hand, satisfying the summation property:

$$\sum_{c \in H} \text{rank}(c) = t$$

Where:

- c = Card
- H = Hand
- rank(c) = Rank of a Card
- t = Value of Steel Card

Let S_t be the set of all valid 5-card combinations remaining in the deck that sum to t:

$$S_t = \{H \subseteq D \mid |H| = 5 \wedge \textstyle\sum_{c \in H} \text{rank}(c) = t\}$$

The set S_t corresponds exactly to the set of restricted integer partitions of t into exactly 5 parts. This formulation marks the intersection of number partition theory (bounding the parts to the interval [1, 13]) and combinations with repetition, where the maximum frequency of any part-value x is dynamically bounded by the number of cards of rank x remaining in the deck. From this set S_t, the algorithm computes two explicit outputs:

1. The total combinatorial density, given by the cardinality $|S_t|$.
2. The optimal hand to play, defined as the maximum element under a total poker hand ordering

```
target 35: 125064 possible hands found
```

Fig. 7: Program output showing the cardinality |St| = 125,064 for target value t = 35

source: Author's archive

C. *Search Algorithm*

To evaluate S_t, the program implements a branch-and-bound depth-first search (DFS) over combinations rather than a brute-force permutation traversal. This directly instantiates the combinatorial r-from-n selection principles

1. Traversal Strategy

   The remaining deck array is sorted in ascending order by rank. At each recursive depth level n (where $0 \leq n \leq 5$), the search loop only considers candidate cards from index start onward. This ensures that every 5-card combination is generated exactly once, suppressing identical hands with altered item orderings.

```c
static void backtrack(int start, Card chosen[], int n_chosen,
                      int current_sum, int target, SearchState *state) {

    /* If 5 hands is already chosen (got a complete 5 card hand)
       Checks if sum is equal to target, if it is then
       1) count every valid hand
       2) evaluate the hand via hand_rank
       3) keep in state if it beats current best  via hr_cmp
    */
    int remaining = HAND_SIZE - n_chosen;
    if (remaining == 0) {
        if (current_sum == target) {
            state->count++;
            HandRank hr = hand_rank(chosen);
            if (!state->found || hr_cmp(&hr, &state->rank) > 0) {
                state->rank  = hr;
                state->found = 1;
                memcpy(state->combo, chosen, HAND_SIZE * sizeof(Card));
            }
        }
        return;
    }

    /* If card in deck is fewer than the hand needed, return */
    int avail = sorted_n - start;
    if (avail < remaining) return;
```

Fig. 8: Base case of backtrack()

source: Author's archive

2. Mathematical Pruning Bounds

The algorithm evaluates the structural properties of numbers to prune dead branches before completing a hand to maximize efficiency:

- Lower Bound Pruning
  If the running cumulative sum plus the maximum possible contribution from the remaining slots cannot reach t, the branch is skipped:
  $$\text{current\_sum} + 13 \times (5 - n) < t$$

- Upper Bound Pruning
  If the running cumulative sum plus the minimum possible contribution from the remaining slots already overshoots t, the remaining loop at that depth level is broken;
  $$\text{current\_sum} + 1 \times (5 - n) > t$$

```
for (int i = start; i <= sorted_n - remaining; i++) {
    Card card = sorted_deck[i];
    int new_sum = current_sum + card.rank;

    /* Pruning to maximize efficiency by cutting unnecessary backtrack
       1) Skips a card whose rank is too low to reach the target
       2) Stops the loop if overshoot the minimum future picks
    */
    if (new_sum + 13 * (remaining - 1) < target) continue;
    if (new_sum + 1  * (remaining - 1) > target) break;

    chosen[n_chosen] = card;
    backtrack(i + 1, chosen, n_chosen + 1, new_sum, target, state);
}
```

Fig. 9: Pruning loop of backtrack()

source: Author's archive

Because the deck is pre-sorted, this upper bound termination guarantees that all subsequent cards in the loop will also overshoot, reducing unnecessary iterations.

```
SearchState find_best_combo(Card deck[], int n, int target, const int suit_order[NUM_SUITS]) {
    sorted_n = n;
    memcpy(sorted_deck, deck, n * sizeof(Card));
    memcpy(suit_pref, suit_order, sizeof(suit_pref));
    qsort(sorted_deck, n, sizeof(Card), card_order_pref);

    SearchState state; memset(&state, 0, sizeof(state));
    Card chosen[HAND_SIZE];

    /* Backtrack starting by index, sum, and cards chosen 0 */
    /* Returns best poker hand in chosen, sum of all valid card in target, and actual best hand
    backtrack(0, chosen, 0, 0, target, &state);
    return state;
}
```

Fig. 10: find_best_combo()

source: Author's archive

As a correctness verification, summing the generated partitions over all reachable targets yields the total combinations of a standard deck:

$$\sum_{t=6}^{64} |S_t| = C(52,5) = 2{,}598{,}960$$

```
target 61: 240 possible hands found
target 62: 92 possible hands found
target 63: 28 possible hands found
target 64: 4 possible hands found
Total: 2598960
```

Fig. 11: verifying the algorithm enumerates every 5-card combination exactly once

source: Author's archive

D. *Hand Evaluation*

Every complete combination in S_t is mapped to a deterministic, 6-element comparison vector:

$$\text{HandRank} = \langle V_0, V_1, V_2, V_3, V_4, V_5 \rangle$$

Where:

a. $V_0 \in [0,9]$ represents the structural poker category
b. V_1 through V_5 hold descending rank values to act as tiebreakers

HandRank

Royal Flush - Category = 9, High Card = 14 (Ace's High)

V0	V1	V2	V3	V4	V5
9	14	0	0	0	0

Four of a Kind - Category = 7, Quad rank = 4, Side Rank = 4

V0	V1	V2	V3	V4	V5
7	8	4	0	0	0

Full House - Category = 6, Triplet Rank = 10, Pair Rank = 7

V0	V1	V2	V3	V4	V5
6	10	7	0	0	0

High Card - Category = 1, 5 Ranks Listed Descending for Tiebraker

V0	V1	V2	V3	V4	V5
0	13	10	8	6	2

Fig. 12: HandRank Visualization

source: Author's archive

E. *The Cyclic Straight Edge Case*

Fixing the Ace at a linear encoding value of 1 introduces a mathematical anomaly. A low straight (A-2-3-4-5) naturally satisfies standard contiguity checks (unique[0] − unique[4] = = 4) However, the high straight (10-J-Q-K-A) becomes non-contiguous under a linear model, appearing as the disjoint set {13, 12, 11, 10, 1}. To handle this cyclic relationship, the program implements an explicit pattern-matching override. When this exact set is detected, the hand is recognized as a valid straight and its high-card value is explicitly remapped:

$$V_1 = 14$$

This ensures the high straight outranks a King-high straight (13-12-11-10-9) and triggers a Royal Flush category designation ($V_0 = 9$) if a flush is also present.

```
/* Check for royal flush */
if (!is_straight && nu == 5 &&
    unique[0] == 13 && unique[1] == 12 && unique[2] == 11 &&
    unique[3] == 10 && unique[4] == 1) {
    is_straight = 1;
    straight_high = 14;
}
```

Fig. 13: Straight Flush Configuration

source: Author's archive

F. Usogui Specific Rule Formulation

1. Suit Preference

Since the suit of a card does not affect the hierarchy of poker hand rankings, except for flushes, a given target value t often yields several different sets of cards that result in identical maximum HandRank vectors. To resolve these equivalence classes deterministically, this program defines a pre-sorted hierarchy of opposing card suits:

- Player A: Heart > Spade > Diamond > Club
- Player B: Club > Diamond > Spade > Heart

When performing a search, the system applies a strict inequality check (> instead of ≥) against the best current card combination. This ensures that the first optimal combination found is retained. By placing Player A and Player B at opposite ends of the card type spectrum, the overlap between the physical cards they select can be minimized.

```
/* Minimizing the overlapping card */
const int SUIT_ORDER_A[NUM_SUITS] = { 0, 1, 2, 3 }; /* Heart first */
const int SUIT_ORDER_B[NUM_SUITS] = { 3, 2, 1, 0 }; /* Club first */
```

Fig. 14: Player Suit Preference

source: Author's archive

2. Overlap Discard Rule

Cards utilized in a round are permanently removed from the deck multiset. If a card is shared between both selected hands, it cannot be double discarded. The operation is modeled as a literal set union ($H_A \cup H_B$) using exact card equality (matching both rank and suit). Applying the principle of inclusion-exclusion, the discard count is formally defined as:

$$\text{Discard Count} = |H_A \cup H_B| = |H_A| + |H_B| - |H_A \cap H_B| = 10 - |H_A \cap H_B|$$

```
/* Getting the Overlapping Cards */
int collect_union(Card h1[HAND_SIZE], Card h2[HAND_SIZE], Card uni[2 * HAND_SIZE]) {
    int n = 0;
    for (int i = 0; i < HAND_SIZE; i++) uni[n++] = h1[i];
    for (int i = 0; i < HAND_SIZE; i++) {
        int dup = 0;
        for (int j = 0; j < HAND_SIZE; j++)
            if (card_eq(h1[j], h2[i])) { dup = 1; break; }
        if (!dup) uni[n++] = h2[i];
    }
    return n;
}
```

Fig. 15: Custom Card Union Counter

source: Author's archive

IV. EXPERIMENT

A. Setup and Replication

To evaluate the mathematical accuracy and validity of the developed algorithm, a replication experiment was conducted using the historical sequence of steel card values played in the Usogui manga match between Madarame Baku and Vincent Lalo. The simulation began with a standard deck of 52 new cards. In accordance with the scope of the experiment, the deck of cards was reused from one round to the next without being reset, and the algorithm was strictly optimized to generate the highest possible poker hand for each target value. The round-by-round programmatic outputs were recorded and compared directly against the canonical narrative outcomes:

```
--- Round 1 --- [52 cards left]
Enter number for Hand A: 36
Enter number for Hand B: 15

Hand A (target 36): 124156 possible hands found
  Best: Four of a Kind   4-Heart-, 8-Heart-, 8-Spade-, 8-Diamond-, 8-Club

Hand B (target 15): 5536 possible hands found
  Best: Straight Flush   A-Club-, 2-Club-, 3-Club-, 4-Club-, 5-Club-

>> Hand B wins the round! (Straight Flush beats Four of a Kind)

10 cards discarded this round (normally 10, minus 0 for overlap).
(42 cards remain in the deck)
```

Fig. 16: Round 1

source: Author's archive

```
--- Round 2 --- [42 cards left]
Enter number for Hand A: 39
Enter number for Hand B: 8

Hand A (target 39): 39813 possible hands found
  Best: Four of a Kind   3-Heart-, 9-Heart-, 9-Spade-, 9-Diamond-, 9-Club-

Hand B (target 8): 12 possible hands found
  Best: Full House       A-Diamond-, A-Spade-, 2-Diamond-, 2-Spade-, 2-Heart

>> Hand A wins the round! (Four of a Kind beats Full House)

10 cards discarded this round (normally 10, minus 0 for overlap).
(32 cards remain in the deck)
```

Fig. 17: Round 2

source: Author's archive

```
--- Round 3 --- [32 cards left]
Enter number for Hand A: 45
Enter number for Hand B: 47

Hand A (target 45): 10052 possible hands found
  Best: Four of a Kind   A-Heart-, J-Heart-, J-Spade-, J-Diamond-, J-Club-

Hand B (target 47): 8814 possible hands found
  Best: Royal Flush      A-Heart-, 10-Heart-, J-Heart-, Q-Heart-, K-Heart-

>> Hand B wins the round! (Royal Flush beats Four of a Kind)

2 cards appeared in both hands, so they are only discarded once:
  - A-Heart
  - J-Heart

8 cards discarded this round (normally 10, minus 2 for overlap).
(24 cards remain in the deck)
```

Fig. 18: Round 3

source: Author's archive

```
--- Round 4 --- [24 cards left]
Enter number for Hand A: 26
Enter number for Hand B: 63

Hand A (target 26): 462 possible hands found
  Best: Full House       4-Spade-, 4-Diamond-, 6-Heart-, 6-Spade-, 6-Diamond

Hand B (target 63): 3 possible hands found
  Best: Full House       Q-Club-, Q-Diamond-, K-Club-, K-Diamond-, K-Spade-

>> Hand B wins the round! (Full House beats Full House)

10 cards discarded this round (normally 10, minus 0 for overlap).
(14 cards remain in the deck)
```

Fig. 19: Round 4

source: Author's archive

```
--- Round 5 --- [14 cards left]
Enter number for Hand A: 25
Enter number for Hand B: 44

Hand A (target 25): 26 possible hands found
  Best: Three of a Kind  3-Spade-, 5-Heart-, 5-Spade-, 5-Diamond-, 7-Heart-

Hand B (target 44): 42 possible hands found
  Best: Full House       7-Club-, 7-Diamond-, 10-Club-, 10-Diamond-, 10-Spade

>> Hand B wins the round! (Full House beats Three of a Kind)

10 cards discarded this round (normally 10, minus 0 for overlap).
(4 cards remain in the deck)

Not enough cards left in the deck to play another round. Game over.
```

Fig. 20: Round 5

source: Author's archive

TABLE II: Cards Used in Usogui

Round	Baku's Card Value (Hand A)	Computed Best (A)	Lalo's Card Value (Hand B)	Computed Best (B)	Winner's Hand	Match?
1	36	Four of a Kind (8s+4)	15	Straight Flush A–5	Lalo's Straight Flush A–5	Exact Match
2	39	Four of a Kind (9s+3)	8	Full House (A, A,2,2,2)	Baku's Four of a Kind of 9s	Exact Match
3	45	Four of a Kind (J's+A)	47	Royal Flush	Lalo's Royal Flush	Exact Match
4	26	Full House (4,4,6,6,6)	63	Full House (3K,2Q)	3 Kings and 2 Queens	Exact Match
5	25	Three of a Kind (3,5,5,5,7)	44	Full House (7,7,10,10,10)	Baku's Straight Flush	Model Can't Produce

B. *Analytical Findings*

The mismatch of the 5th round is caused by the imperfection of the card selection made by Baku specifically to preserve the straight flush for round 5, whereas author's simulation always output the most optimal hand available from the remaining deck. The consequence of the optimal play in the 4th round results in the cards needed for the round 5's straight flush are already used.

Because this simulation strictly applies absolute mathematical optimization at each discrete step, the algorithm uses the resources required during Round 4 to maximize immediate utility. As a result, the specific cards needed to form a Straight Flush with a target of 25 in Round 5 are not available in the depleted deck, thereby reducing Baku's best possible hand to merely Three of a Kind. This experiment provides independent, quantitative confirmation of a claim that was only qualitatively demonstrated in the original manga: under a relentless, short-sighted optimal selection system, the path to a superior hand in the future becomes mathematically blocked.

C. *Controlled Algorithm Experiment*

1. Distribution Experiment

To verify the finite number partition model, a comprehensive programmatic search was conducted of all possible integer targets in an intact 52-card deck. The algorithm evaluates each integer value t from 1 to 64 by inserting it one by one into the search framework to calculate the cardinality of the exact subset $|S_t|$.

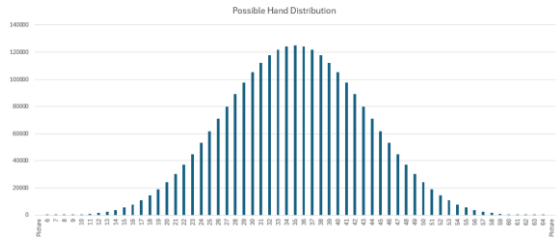

Fig. 21: Possible Hand Distribution

source: Author's archive

The empirical distribution forms a perfectly symmetrical curve bounded strictly within the interval [6,64]. Targets outside of this closed interval (such as $t \leq 5$ or $t \geq 65$) results in exactly 0 valid hand combinations. The complete distribution is provided on the table below:

TABLE III: Possible Card Combinations on Each Target

t	S_t	t	S_t	t	S_t
≤ 5	0	26	70892	47	44892
6	4	27	80076	48	37188
7	28	28	88988	49	30296
8	92	29	97608	50	24136
9	240	30	105344	51	18852
10	484	31	112172	52	14364
11	920	32	117620	53	10728
12	1552	33	121728	54	7788
13	2492	34	124156	55	5536
14	3772	35	125064	56	3772
15	5536	36	124156	57	2492
16	7788	37	121728	58	1552
17	10728	38	117620	59	920
18	14364	39	112172	60	484
19	18852	40	105344	61	240
20	24136	41	97608	62	92
21	30296	42	88988	63	28
22	37188	43	80076	64	4
23	44892	44	70892	≥ 65	0
24	53108	45	61892	Total	2598960
25	61892	46	53108		

2. Boundary Test

Target values at the extreme parameters of the distribution were tested on a fresh deck to validate the physical limitations:

- For $t \leq 5$ and $t \geq 65$, the program returned exactly 0 valid hands, confirming these values are mathematically impossible with 5 cards.

```
target 1: 0 possible hands found
target 2: 0 possible hands found
target 3: 0 possible hands found
target 4: 0 possible hands found
target 5: 0 possible hands found
```

Fig. 22: Total Possible Hand for Input Less than 6

source: Author's archive

```
target 65: 0 possible hands found
target 66: 0 possible hands found
target 67: 0 possible hands found
```

Fig. 23: Total Possible Hand for Higher than 64

source: Author's archive

- For $t = 6$ the algorithm identified exactly 4 combinations. Every variation generated a Four of a Kind composed of four Aces and a 2 (1+1+1+1+2 = 6). The four combinations differed solely by the suit of the remaining 2-ranked cards, providing a direct combinatorial proof of the game's lower limit.

```
target 6: 4 possible hands found
```

Fig. 24: Total Possible Hand for Input 6

source: Author's archive

- For $t = 64$, the algorithm symmetrically identified exactly 4 combinations, each yielding a Four of a Kind composed of four Kings and a Queen (13+13+13+13+12 = 64), establishing the upper limit.

```
target 64: 4 possible hands found
```

Fig. 25: Total Possible Hand for Input 64

source: Author's archive

3. Isolated Overlap Verification

Since historical matches may not automatically trigger every edge case, three controlled testing environments were designed on the new deck to validate the pruning formula based on the inclusion-exclusion principle $(10 - |H_A \cap H_B|)$.

1. 4 Overlapping Cards

By inputting 6 on both hands, we can guarantee that both hands got the same four cards (Aces), whilst the last card is different since both players prioritize different suits, with player A choosing a 2 of Heart while player B choosing 2 of Spade

```
Hand A (target 6): 4 possible hands found
  Best: Four of a Kind   A-Heart-, A-Spade-, A-Diamond-, A-Club-, 2-Heart-

Hand B (target 6): 4 possible hands found
  Best: Four of a Kind   A-Club-, A-Diamond-, A-Spade-, A-Heart-, 2-Club-

>> It's a tie! Both hands are Four of a Kind.

4 cards appeared in both hands, so they are only discarded once:
  - A-Heart
  - A-Spade
  - A-Diamond
  - A-Club

6 cards discarded this round (normally 10, minus 4 for overlap).
(46 cards remain in the deck)
```

Fig. 26: 4 Overlapping Cards

source: Author's archive

2. 2 Overlapping Cards

By inputting 6 on hand A and 9 on hand B, we can guarantee that hand A is a four of a kind of Aces accompanied by a 2 of heart while B got four of a kind of two accompanied by an Ace of club. Because of that, the Ace of club and 2 of heart will overlap

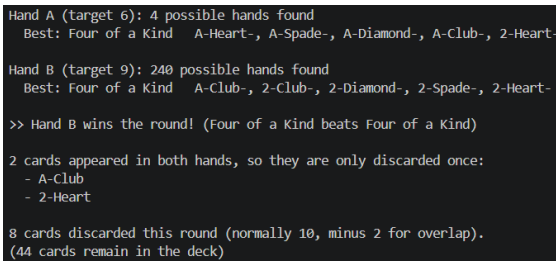

Fig. 27: 2 Overlapping Cards
source: Author's archive

3. 1 Overlapping Card

By inputting 6 on hand A and 15 on hand B, we can guarantee that hand A is four of a kind of Aces accompanied by a 2 of heart, while B got straight flush of club. Because of that, the Ace of club is going to overlap

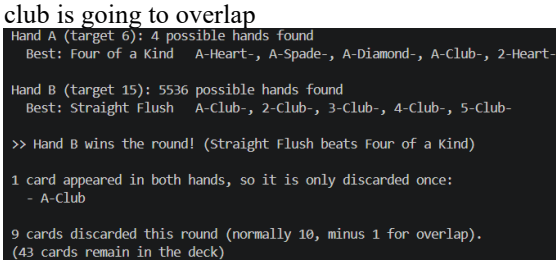

Fig. 28: 1 Overlapping Cards
source: Author's archive

V. Conclusion

This paper has shown that the card selection problem in Air Poker can be rigorously formalized using finite combinatorics and the theory of number partitions. By modeling the deck of cards as a finite multiset and each steel card value as a target for a finite integer partition, the set of valid card combinations (St) for any given target t can be computed exhaustively. The implemented depth-first search with the branch-and-bound method computes each element of St exactly once without reordering, which is confirmed by a completeness check showing that the size $|S_t|$ for all targets is equal to C(52,5) = 2,598,960.

Replication experiments on canonical Usogui games yielded exact matches in four out of five rounds. The single discrepancy in Round 5 is not an error in the model, but rather a confirmation of it: Baku's suboptimal play in Round 4 was an intentional strategic sacrifice to save cards for a future straight flush. Since the algorithm always selects the globally optimal card combination at each step, it consumes the cards set aside in Round 4, making a straight flush mathematically impossible to achieve in Round 5. These results quantitatively validate the strategic claim that was previously demonstrated only narratively in the manga.

Controlled distribution experiments prove that the valid steel card range [6, 64] is not an arbitrary game design choice, but rather a strict mathematical bound derived from the deck's structure: the minimum achievable total with five distinct cards is 1+1+1+1+2 = 6 (four Aces and one Two), and the maximum is 13+13+13+13+12 = 64 (four Kings and one Queen). The perfectly symmetric distribution around t = 35 further reflects the rank-complement bijection inherent in a standard deck of cards. The overlapping discard rule formalized using the inclusion-exclusion principle as $|H_A \cup H_B| = |H_A| + |H_B| - |H_A \cap H_B| = 10 - |H_A \cap H_B|$, has been verified in all three controlled overlap cases.

These results demonstrate that this partition-based model is mathematically sound and can be practically applied to verify game states in Air Poker. Future research could extend this model to incorporate multi-round betting systems and strategic planning, potentially using dynamic programming to model optimal long-term card reservations, rather than round-by-round greedy optimization.

VI. Appendix

a) Source Code: https://github.com/wequra/13525054_MatematikaDiskrit

b) Video: https://youtu.be/w7Bt4H9-GZ4

VII. Acknowledgement

First of all, the author offers praise and thanks to Allah for His blessing and grace, which allows the author to finish the paper on time. Secondly, the author would also like to thank Dr. Ir. Rinaldi Munir, M.T., for the guidance and materials provided in the IF1220 discrete mathematics class. Third, the author would also thank Toshio Sako as the author of Usogui for making such an amazing manga that piqued my interest in analyzing every single game that is played. Lastly, the author wishes to sincerely thank the reader of this paper.

References

[1] R. Munir, "Kombinatorika (Bagian 1),". [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/18-Kombinatorika-Bagian1-2026.pdf [Accessed: 15-Jun-2026].

[2] R. Munir, "Kombinatorika (Bagian 2)," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/19-Kombinatorika-Bagian2-2026.pdf [Accessed: 15-Jun-2026].

[3] "Basics of Poker," Bicycle Cards, The United States Playing Card Company. [Online]. Available: https://bicyclecards.com/how-to-play/basics-of-poker [Accessed: 15-Jun-2026].

[4] "Air Poker," Usogui Wiki, Fandom. [Online]. Available: https://usogui.fandom.com/wiki/Air_Poker [Accessed: 15-Jun-2026].

Statement

I hereby declare that this paper is my own original work; it is not an adaptation or translation of another person's work, nor is it plagiarized.

Bandung, 18 June 2026

Raffi Fauzi Hermawan
13525054