Graph-Based Shortest Path Optimization for Booster Pack Card Selection in Balatro

Jingglang Galih Rinenggan - 13524095
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: jingglang@gmail.com , 13524095@std.stei.itb.ac.id

***Abstract*—This paper explores the application of graph theory and Dijkstra's algorithm to optimize card selection in the roguelike deckbuilder game Balatro, which is inspired by poker mechanics. By modeling the relationships between cards, traits, synergies, and jokers as a weighted, layered directed acyclic graph, we formalize the decision-making process for choosing the most synergistic card from a booster pack. The proposed approach quantifies synergy potential using a custom weight function and leverages Dijkstra's algorithm to identify the optimal card that maximizes benefits with the player's current jokers. Experimental results demonstrate the effectiveness of this method in enhancing strategic gameplay, providing a systematic framework for analyzing complex interactions in modern card games.**

Keywords—Poker, Video Games, Deckbuilding Video Games, Roguelike Video Games, Graph Theory, Shortest Path, Dijkstra's Algorithm

I. INTRODUCTION

Poker has long captivated players and researchers alike, not only as a game of chance but also as a rich domain for strategic thinking, probability, and mathematical reasoning. Its enduring popularity stems from the intricate balance between randomness and skill, where players must constantly evaluate probabilities, read opponents, and make optimal decisions under uncertainty. Over the years, poker has evolved into numerous variants, each introducing new layers of complexity and strategic depth, further cementing its status as a classic game of both luck and logic. Beyond its traditional forms, the core mechanics of poker, such as hand evaluation, betting strategies, and risk management have inspired the development of many other card games.

One notable example is Balatro, a video game by LocalThunk, a modern roguelike deckbuilder that creatively integrates poker hand mechanics with elements of deck construction, resource management, and procedural progression, and has been widely regarded as an innovative example of modern game design. In Balatro, players are challenged not only to form strong poker hands but also to adapt their strategies by acquiring new cards, jokers, and modifiers that interact in complex ways. This fusion of poker fundamentals with innovative game design provides a fertile ground for exploring advanced decision-making processes, synergy optimization, and the application of mathematical models to gameplay.

The combination of poker mechanics and deckbuilding in Balatro offers a unique opportunity to analyze how traditional concepts of probability and graph theory can be leveraged to enhance strategic choices. By modeling the relationships between cards, traits, synergies, and jokers as a graph, and applying algorithms such as Dijkstra's, we can systematically approach the problem of optimal card selection. This research aims to formalize and solve such decision-making challenges, demonstrating how mathematical tools can be applied to modern card games to maximize player outcomes and enrich the overall gaming experience.

II. THEORETICAL FOUNDATION

A. *Poker Mechanics*

Poker is a popular game of cards that combines elements of chance and reasoning. The game has various variants, but the basic mechanics involve players being dealt a hand of cards and making bets based on the strength of their hand. The objective is to win chips or money from other players by either having the best hand at showdown or convincing other players to fold their hands. A poker card game typically consists of a standard 52-card deck, where each card has a rank (Ace, 2-10, Jack, Queen, King) and a suit (Hearts, Diamonds, Clubs, Spades) that determines its value and category. The most relevant mechanics for this research are the poker hands, which are combinations of cards that determine the strength of a player's hand [1]. The standard poker hands, ranked from highest to lowest, are as follows:

1. **Royal Flush**: Ace, King, Queen, Jack, 10 of the same suit
2. **Straight Flush**: Five consecutive cards of the same suit
3. **Four of a Kind**: Four cards of the same rank
4. **Full House**: Three of a kind and a pair
5. **Flush**: Five cards of the same suit, not in sequence
6. **Straight**: Five consecutive cards of different suits
7. **Three of a Kind**: Three cards of the same rank
8. **Two Pair**: Two cards of one rank and two cards of another rank
9. **Pair**: Two cards of the same rank

10. **High Card**: The highest card in the hand if no other hand is made

Over the years, poker has evolved into various forms, with each variant having its own set of rules and strategies. It even has inspired many other card games, including the unique game Balatro, which incorporates poker mechanics into its gameplay.

B. *Balatro*

Balatro is a roguelike deckbuilder inspired by poker created by LocalThunk, where players build their deck and utilize various jokers, tarot cards, and other modifiers to create powerful combinations [2]. The game challenges players to reach increasingly difficult score thresholds by playing poker hands, while adapting their strategy to the evolving deck and available synergies. Its blend of familiar poker mechanics with innovative card interactions and strategic planning sets it apart from traditional card games.

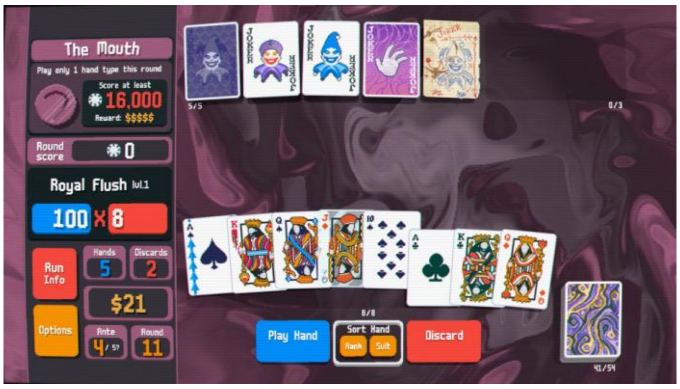


Fig. 1. Playing a blind in Balatro

Players start with the default 52 card poker deck with no jokers. Players then choose to play a blind or skip the next blind. When they play a blind, each blind requires the player to attain a certain score threshold, which is determined by the current blind level. The player can play cards from their hand to form poker hands, which are then evaluated based on the accumulation of each card's underlying chips, which by default is determined by its rank. The amount of chips can be further modified by jokers, tarot cards, and other modifiers that can be acquired throughout the game. Else than chips, there is also a multiplier value that is determined by the poker hand played or joker effects, which is then applied to the total chips accumulated at the end of the blind. The player can also discard up to 5 cards from their hand to draw new cards from the deck, which can be used to form new poker hands when the current hand isn't sufficient enough to reach an optimal score. The player can do those two actions a certain number of times shown by the UI. If the player's hand runs out before reaching the score threshold, they lose. If the player reaches the score threshold, they win the blind and attain money, which can be used to buy new cards from the shop or booster packs [3].

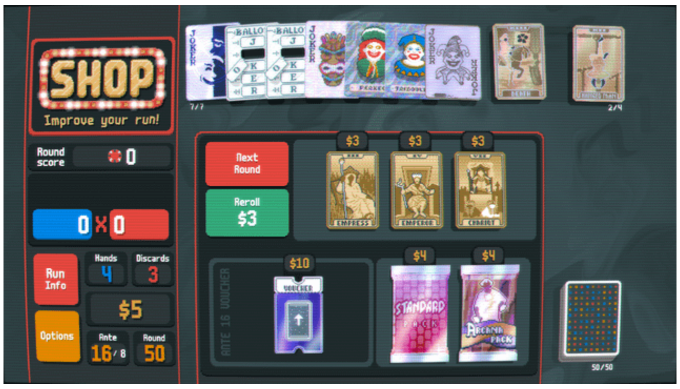


Fig. 2. The shop interface in Balatro

Between blinds, players can choose to buy new cards from the shop or booster packs, which contain a selection of cards that can enhance their deck. The player can also acquire jokers, which are special cards that provide unique benefits or abilities when certain conditions are met. Jokers can be used to modify the player's hand, increase the score, or provide other strategic advantages [3].

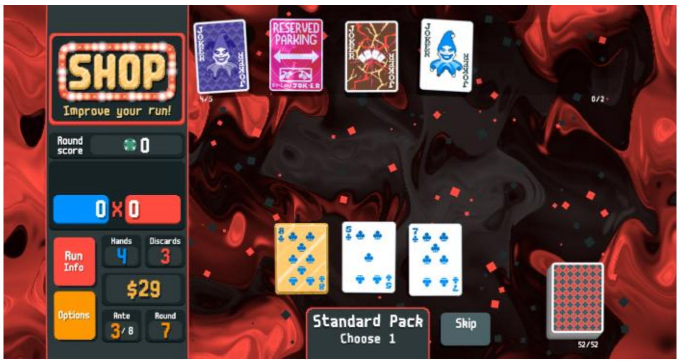


Fig. 3. Standard booster pack selection interface in Balatro

The selection of cards from the booster pack is a crucial aspect of the game, as it allows players to adapt their strategy and enhance their deck with new cards that complement their existing jokers and synergies. The challenge lies in determining which card from the booster pack best complements the player's current jokers for the player to survive the next blinds as it increasingly gets harder.

C. *Graph Theory*

1) *Definition*

Graphs are commonly used to represent discrete objects and relations between objects. Graph G is defined as a pair (V, E), where V is a set of vertices, $V = \{v_1, v_2, ..., v_n\}$, representing the objects, and E is a set of edges, E $= \{e_1, e_2, ..., e_m\}$, representing the relationships between the objects. Each edge connects two vertices, indicating a relationship or interaction between them [4].

Based on wether a graph contains multiple edges between the same pair of vertices or not, it can be classified as a simple graph or a non-simple graph. A simple graph is a graph that does not contain multiple edges between the same pair of vertices, while a non-simple graph allows multiple edges between the same pair of vertices. In this research, we will use a simple graph to represent the relationships between cards, traits, synergies, and jokers in Balatro [4].

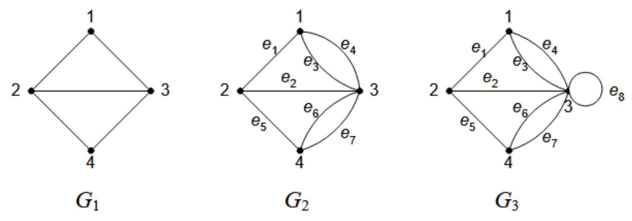


Fig. 4. Illustration of a simple graph (G_1) and non-simple graphs (G_2 and G_3)

(Taken from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf)

In the image above, G_1 is a graph with $V = \{ 1, 2, 3, 4 \}$ and $E = \{ (1, 2), (1, 3), (2, 3), (2, 4), (3, 4) \}$, it is a simple graph since it doesn't have multiple edges between the same pair of vertices whereas G_2 and $G_3$are non-simple graphs because both of them have multiple edges between the same pair of vertices, e_3 and e_4 are both connecting v_1 and v_3 on both graphs.

A graph can also be directed or undirected. A directed graph is a graph where the edges have a direction, indicating a one-way relationship between the vertices. An undirected graph is a graph where the edges do not have a direction, indicating a two-way relationship between the vertices [4].

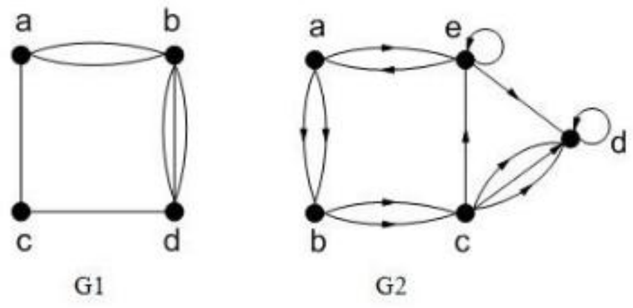


Fig. 5. Illustration of an undirected graph (G_1) and directed graph (G_2)

(Taken from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf)

In the image above, $G_1$is a non-simple undirected graphs as it provides no visual indicator of a direction on its edges, which means its edges doesn't dictate a direction between the vertex it connects to. $G_2$is a non-simple directed graph where every edge has a direction on which vertex it's directed to.

2) *Terminology*

There are a several important terms in graph theory that are relevant to this research. First of all, a vertex is a node in the graph that represents an object or entity, such as a card, trait, synergy, or joker. An edge is a connection between two vertices that represents a relationship or interaction between them. The degree of a vertex is the number of edges connected to it, indicating how many relationships it has with other vertices [4].

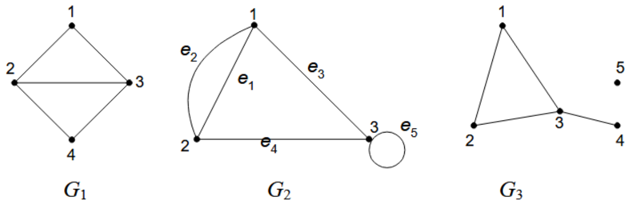


Fig. 6. Illustration of three graphs

(Taken from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf)

In the image above, G_2has an edge $e_3 = (1, 3)$, where one of its vertex, 1, has a degree of 3 because it has 3 edges.

Other than that, there are also path which is a sequence of edges that connects two vertices, and the length of a path is the number of edges in the path. A cycle is a path that starts and ends at the same vertex, forming a loop.

3) *Directed Acyclic Graphs (DAGs)*

A directed acyclic graph (DAG) is a directed graph that does not contain any cycles, meaning that there is no way to start at a vertex and return to it by following the directed edges. DAGs are particularly useful for representing hierarchical structures or dependencies, where the relationships between the vertices have a clear direction and do not loop back on themselves [5].

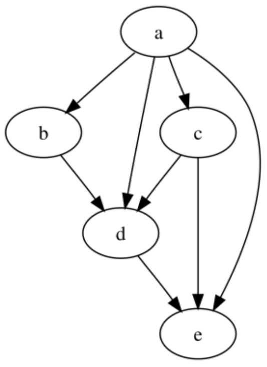


Fig. 7. Illustration of a directed acyclic graph

(Taken from https://commons.wikimedia.org/wiki/File:Tred-G.png)

4) *Weighted Graphs*

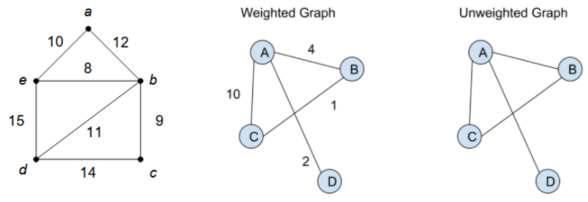


Fig. 8. Illustration of a weighted graph and unweighted graph

(Taken from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf)

A weighted graph is a graph where each edge has a weight or cost associated with it, representing the value or importance of the relationship between the vertices. The weight can be any numerical value, such as a distance, cost, or benefit [4].

D. *Dijkstra's Algorithm*

Dijkstra's algorithm is an algorithm for finding the shortest path in a weighted graph. It was proposed by Edsger W. Dijkstra in 1956 and is widely used in various applications, such as routing and navigation systems [6]. The algorithm works by iteratively selecting the vertex with the minimum distance from the source vertex and updating the distances of its neighboring vertices based on the weights of the edges connecting them [7]. In laymen's terms, it finds the shortest path from a starting vertex to all other vertices in the graph, ensuring that the path has the minimum total weight. The following steps outline the algorithm:

1. Initialize the distance of the source vertex to 0 and all other vertices to infinity.
2. Push the source vertex into a priority queue or min-heap with its distance.
3. While the priority queue is not empty:
4. Pop the vertex with the minimum distance from the priority queue.
5. For each neighboring vertex of the popped vertex, if the distance to the neighboring vertex through the popped vertex is less than its current distance, update its distance and push it into the priority queue.
6. Repeat steps 3-5 until all vertices have been processed or the priority queue is empty.

III. ANALYSIS

A. *Problem Formalization*

As a highly strategic card game, Balatro requires players to make optimal decisions based on the current state of the game. One of the key challenges is to determine the optimal card to choose from a booster pack while taking into account the jokers currently in the player's possession. This decision-making process requires dynamic evaluation of the potential benefits each card can provide based on its traits (rank and suit), jokers in posession, and the deck composition.

Each joker activates its benefits upon meeting specific conditions. For example, a greedy joker will give +3 mult for every diamond card played in hand, or a walkie talkie will give +10 chips and +4 mult for every 10 or 4 card played in hand. We can call this condition a "synergy", which is a specific interaction between the jokers and the cards in the player's hand. The goal is to evaluate which booster card best complements the current jokers, maximizing synergy potential and score outcomes. We can assume that the player's intended strategy is complementing their decks with new cards that enhance the synergy with their jokers. With this in mind, we can solve the problem of selecting the optimal card from the booster pack by considering the synergy between the jokers and the cards available.

This problem can be modeled as a weighted, layered directed acyclic graph, where one set of vertices represents the cards in the booster pack and the other set represents the jokers. Both vertices are connected by two vertices, one for the rank and one for the suit of the card. Each weight on the edges represents how costly it is to select a card based on the synergy with the jokers. Cost, here, is defined as the negative of the potential benefit that can be gained from selecting a card with a strategy that complements the jokers. The goal is to find the card with the minimum cost, which corresponds to the maximum potential benefit. This can be achieved using Dijkstra's algorithm, which efficiently finds the shortest path in a weighted graph. For each card in the booster pack, we compute the shortest path to every reachable joker node, where path cost inversely reflects synergy potential. The card with the lowest total path cost across all jokers is selected as optimal.

B. *Graph Design*

This problem can be modeled as a weighted, layered directed acyclic graph with four types of vertices: cards (V_1), traits (V_2), synergies (V_3), and jokers (V_4). Cards are defined as the cards available in the booster pack, modeled as a struct with fields for rank and suit. Traits are the attributes of these cards, such as their rank (e.g., 2, 3, ..., ace), suit (hearts, diamonds, clubs, spades), and additional composite traits related to some synergies (face cards, even rank, odd rank). Synergies represent the interactions between the jokers and the cards, which can be defined as specific conditions that trigger benefits when certain cards are played in conjunction with jokers, these can be a specific trait or a poker hand. For the simplicity of this research, we will only include the jokers with synergies that are related to a card's trait or hand. Jokers that are not related to a card's trait or hand, such as Gros Michel which gives a +15 mult and 1 in 6 chance of being destroyed regardless of the played card, will not be included in the graph. Jokers that synergize with traits or hands but whose activation conditions are unpredictable or hands, such as To Do List which gives $5 if poker hand is a randomly determined hands that changes every end of round, will also not be included in the graph. Jokers are special cards that provide unique benefits or abilities when certain conditions are met. With that in mind, here are the following jokers and corresponding synergies that will be included in this research [8]:

Joker	Synergy
Greedy Joker	Diamond suit

Lusty Joker	Heart suit
Wrathful Joker	Spade suit
Gluttonous Joker	Club suit
Jolly Joker	Pair
Zany Joker	Three of a Kind
Mad Joker	Four of a Kind
Crazy Joker	Straight
Droll Joker	Flush
Sly Joker	Pair
Wily Joker	Three of a Kind
Clever Joker	Four of a Kind
Devious Joker	Straight
Crafty Joker	Flush
Even Steven	Even rank
Odd Todd	Odd rank
Scholar	Ace rank
Scary Face	Face cards
Walkie Talkie	10 rank
Walkie Talkie	4 rank
Smiley Face	Face cards
Square Joker	Pair
Square Joker	Two Pair
Square Joker	Three of a Kind
Square Joker	Four of a Kind
Spare Trousers	Two Pair
Flower Pot	Diamond suit
Flower Pot	Club suit
Flower Pot	Heart suit
Flower Pot	Spade suit
The Duo	Pair
The Trio	Three of a Kind
The Family	Four of a Kind
The Order	Straight
The Tribe	Flush

Fig. 9. A table of jokers and its synergies

Edges are defined as follows: each card in V_1 is connected to its corresponding traits in V_2 (such as rank and suit) as E_1; traits in V_2 are linked to relevant synergies in V_3 that describe possible interactions as E_2; and each synergy in V_3 is connected to the jokers in V_4 that can activate or benefit from that synergy as E_3. This layered structure allows for a clear representation of the relationships and dependencies between cards, their traits, potential synergies, and the jokers, facilitating the analysis of optimal card selection based on synergy maximization.

C. *Weight Function*

The first edge E_1 $(V_1 \rightarrow V_2)$ that connects each card to its traits doesn't have a weight, as it simply represents the inherent properties of the card, or alternatively defined with a constant weight of 1.

The second edge E_2 $(V_2 \rightarrow V_3)$ that connects traits to synergies is weighted based on how well the traits align with the synergies provided by the jokers. There are two important factors to consider when determining the weight of this edge: how valuable the synergy is and how much is the current deck composition complementing the synergy. The weight of E_2 is defined as:

$$\text{Weight} = \frac{1}{(\text{Synergy Base Value}) \times (\frac{\text{Cards Currently}}{\text{Cards Required}})}$$

Where E_2 represents the edge from traits to synergies. This formula ensures that the edge weight decreases as the synergy becomes more valuable and as the deck composition better supports the synergy, making lower weights correspond to stronger synergies.

Where:

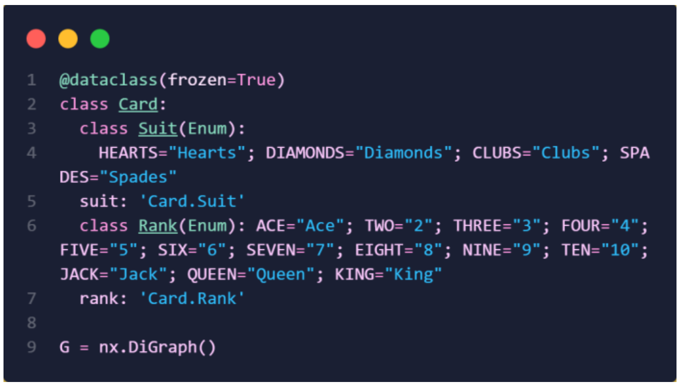
- **Synergy Base Value** is a constant representing the inherent benefit of the synergy (e.g., the bonus multiplier or chips provided by the joker for that trait).
- **Cards Currently** is the number of cards in the current deck that possess the relevant trait.
- **Cards Required** is the minimum number of cards needed to activate the synergy.

This formula ensures that the weight decreases (indicating a stronger synergy and thus a more favorable edge) as the synergy becomes more valuable and as the deck composition better supports the synergy.

The third edge E_3 $(V_3 \rightarrow V_4)$ that connects synergies to jokers is weighted with a boolean value, where 1 indicates the connected joker is in the player's possession, and 0 indicates that the joker is not in possession. This ensures that only jokers that are currently available to the player are considered in the analysis, allowing for a more accurate evaluation of potential synergies.

D. *Implementation*

We can implement the graph structure and Dijkstra's algorithm in Python to find the optimal card from the booster pack. The implementation will involve defining the graph with the vertices and edges as described, applying the weight function to calculate edge weights, and then using Dijkstra's algorithm to find the shortest path from each card to the jokers. This implementation leverages networkx library for graph representation and manipulation and matplotlib for visualization.

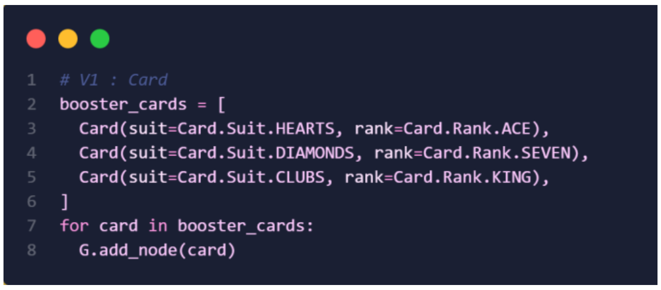


```python
@dataclass(frozen=True)
class Card:
  class Suit(Enum):
    HEARTS="Hearts"; DIAMONDS="Diamonds"; CLUBS="Clubs"; SPADES="Spades"
  suit: 'Card.Suit'
  class Rank(Enum): ACE="Ace"; TWO="2"; THREE="3"; FOUR="4"; FIVE="5"; SIX="6"; SEVEN="7"; EIGHT="8"; NINE="9"; TEN="10"; JACK="Jack"; QUEEN="Queen"; KING="King"
  rank: 'Card.Rank'

G = nx.DiGraph()
```

Fig. 10. Card struct

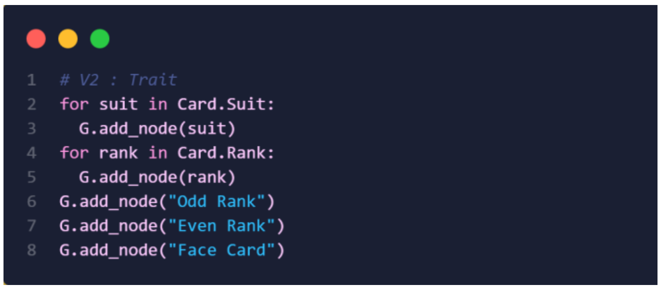
First, we define the Card class with its suit and rank as enumerations. Then, we create a directed graph G using NetworkX.



```python
# V1 : Card
booster_cards = [
  Card(suit=Card.Suit.HEARTS, rank=Card.Rank.ACE),
  Card(suit=Card.Suit.DIAMONDS, rank=Card.Rank.SEVEN),
  Card(suit=Card.Suit.CLUBS, rank=Card.Rank.KING),
]
for card in booster_cards:
  G.add_node(card)
```

Fig. 11. First vertex layer (V_1) construction code

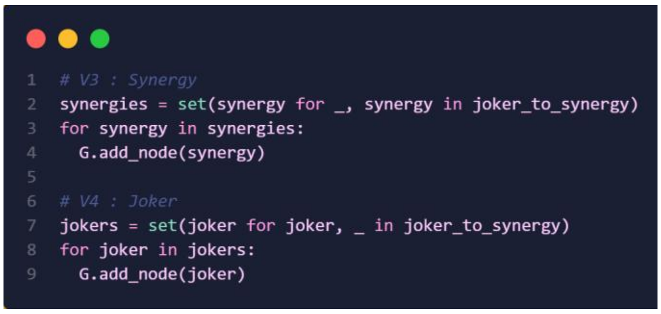
Next, we define the booster cards and add them as nodes in the graph. For the sake of simplicity, we will use a small set of booster cards.



```python
# V2 : Trait
for suit in Card.Suit:
  G.add_node(suit)
for rank in Card.Rank:
  G.add_node(rank)
G.add_node("Odd Rank")
G.add_node("Even Rank")
G.add_node("Face Card")
```

Fig. 12. Second vertex layer (V_2) construction code

Then, we add the traits as nodes in the graph, including suits, ranks, and additional composite traits like "Odd Rank", "Even Rank", and "Face Card".

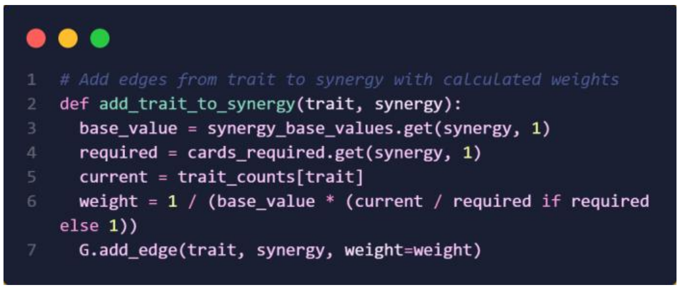


```python
# V3 : Synergy
synergies = set(synergy for _, synergy in joker_to_synergy)
for synergy in synergies:
  G.add_node(synergy)

# V4 : Joker
jokers = set(joker for joker, _ in joker_to_synergy)
for joker in jokers:
  G.add_node(joker)
```

Fig. 13. Third (V_3) and Fourth (V_4) vertex layer construction code

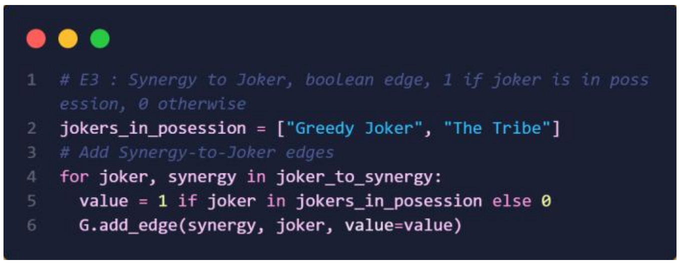
Next, we define the synergies based on the jokers and add them as nodes in the graph. The joker_to_synergy mapping which is based on Fig. 9 will be used to determine which synergies are relevant. We also add the jokers as nodes in the graph.



```python
# Add edges from trait to synergy with calculated weights
def add_trait_to_synergy(trait, synergy):
  base_value = synergy_base_values.get(synergy, 1)
  required = cards_required.get(synergy, 1)
  current = trait_counts[trait]
  weight = 1 / (base_value * (current / required if required else 1))
  G.add_edge(trait, synergy, weight=weight)
```

Fig. 14. Trait to synergy edge construction code

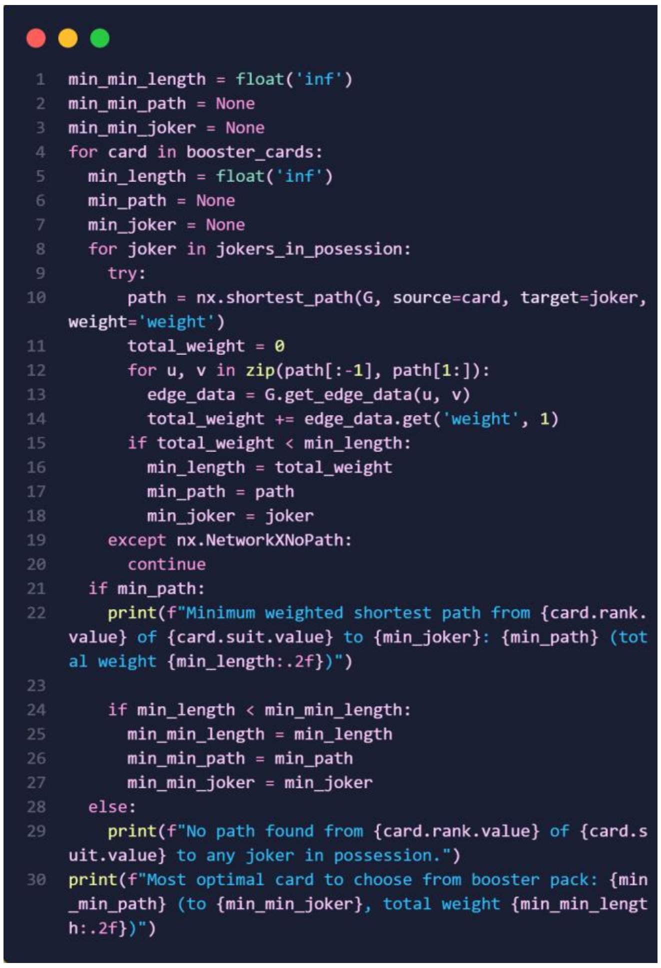
For each trait, we add edges to the synergies with calculated weights based on the synergy base values and the current deck composition. Where synergy_base_values is a dictionary mapping each synergy to its base value, and cards_required is a dictionary mapping each synergy to the number of cards required to activate it. The trait_counts dictionary keeps track of how many cards in the current deck have each trait. For the sake of simplicity, we will assume that the current deck composition is the starting 52 cards of a standard deck.



```python
# E3 : Synergy to Joker, boolean edge, 1 if joker is in possession, 0 otherwise
jokers_in_posession = ["Greedy Joker", "The Tribe"]
# Add Synergy-to-Joker edges
for joker, synergy in joker_to_synergy:
  value = 1 if joker in jokers_in_posession else 0
  G.add_edge(synergy, joker, value=value)
```

Fig. 15. Joker to synergy edge construction code

Next, we add edges from the synergies to the jokers, with a boolean value indicating whether the joker is in possession or not. For this example, we will assume that the player has two jokers in possession: "Greedy Joker" and "The Tribe".



```python
min_min_length = float('inf')
min_min_path = None
min_min_joker = None
for card in booster_cards:
  min_length = float('inf')
  min_path = None
  min_joker = None
  for joker in jokers_in_posession:
    try:
      path = nx.shortest_path(G, source=card, target=joker, weight='weight')
      total_weight = 0
      for u, v in zip(path[:-1], path[1:]):
        edge_data = G.get_edge_data(u, v)
        total_weight += edge_data.get('weight', 1)
      if total_weight < min_length:
        min_length = total_weight
        min_path = path
        min_joker = joker
    except nx.NetworkXNoPath:
      continue
  if min_path:
    print(f"Minimum weighted shortest path from {card.rank.value} of {card.suit.value} to {min_joker}: {min_path} (total weight {min_length:.2f})")

    if min_length < min_min_length:
      min_min_length = min_length
      min_min_path = min_path
      min_min_joker = min_joker
  else:
    print(f"No path found from {card.rank.value} of {card.suit.value} to any joker in possession.")
print(f"Most optimal card to choose from booster pack: {min_min_path} (to {min_min_joker}, total weight {min_min_length:.2f})")
```

Fig. 16. Minimum path determination code

Finally, we iterate through each card in the booster pack, finding the shortest path to each joker in possession using Dijkstra's algorithm. We compute the total weight along the path and keep track of the minimum weight and corresponding path. The card with the minimum total weight is selected as the optimal card to choose from the booster pack.

IV. Results

Taking the example of the booster pack containing three cards: Ace of Hearts, 7 of Diamonds, and King of Clubs, and assuming the player has two jokers in possession (Greedy Joker and The Tribe), the algorithm will output the most optimal card to choose based on the synergies with the jokers, which is determined by the minimum total weight of the paths from the cards to the jokers. The output will indicate the optimal card, the path taken, and the total weight, reflecting the synergy potential with the jokers. Leveraging matplotlib, we can visualize the graph structure, showing the relationships between cards, traits, synergies, and jokers, along with the weights of the edges and get this beautiful graph:

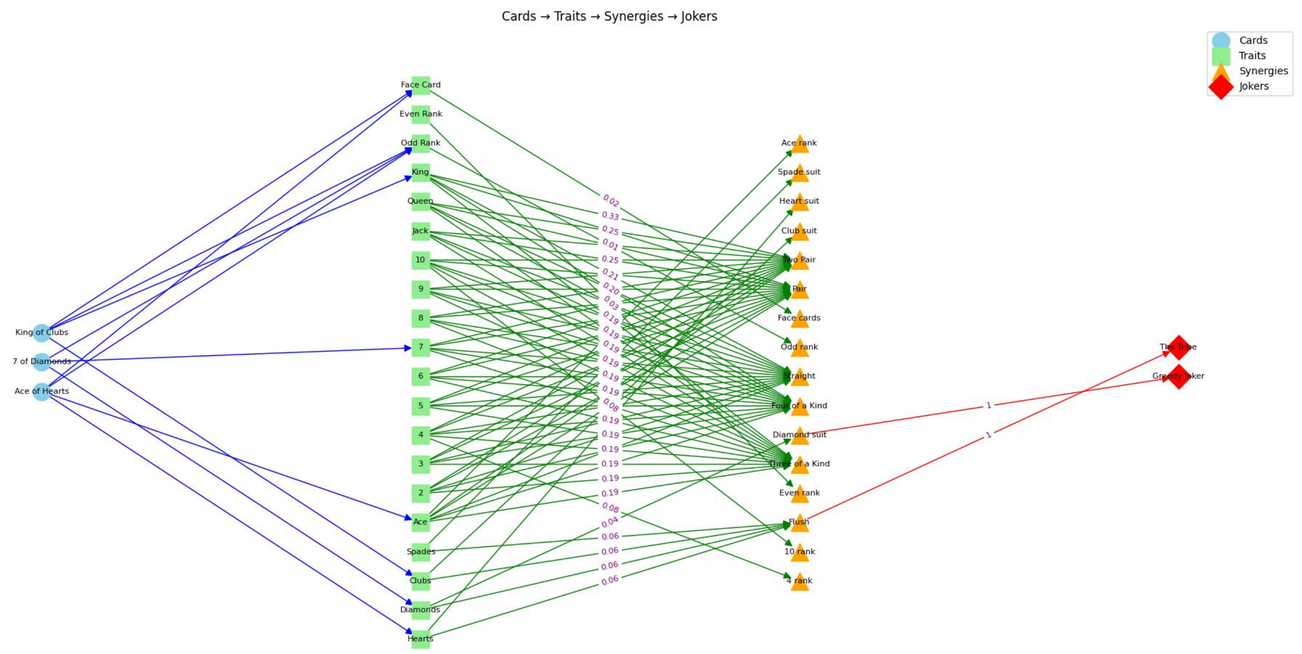
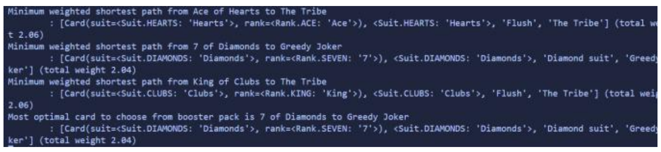


Fig. 17. Graph model visualization



```
Minimum weighted shortest path from Ace of Hearts to The Tribe
        : [Card(suit=<Suit.HEARTS: 'Hearts'>, rank=<Rank.ACE: 'Ace'>), <Suit.HEARTS: 'Hearts'>, 'Flush', 'The Tribe'] (total we
t 2.06)
Minimum weighted shortest path from 7 of Diamonds to Greedy Joker
        : [Card(suit=<Suit.DIAMONDS: 'Diamonds'>, rank=<Rank.SEVEN: '7'>), <Suit.DIAMONDS: 'Diamonds'>, 'Diamond suit', 'Greedy
ker'] (total weight 2.04)
Minimum weighted shortest path from King of Clubs to The Tribe
        : [Card(suit=<Suit.CLUBS: 'Clubs'>, rank=<Rank.KING: 'King'>), <Suit.CLUBS: 'Clubs'>, 'Flush', 'The Tribe'] (total wei
2.06)
Most optimal card to choose from booster pack is 7 of Diamonds to Greedy Joker
        : [Card(suit=<Suit.DIAMONDS: 'Diamonds'>, rank=<Rank.SEVEN: '7'>), <Suit.DIAMONDS: 'Diamonds'>, 'Diamond suit', 'Greedy
ker'] (total weight 2.04)
```

Fig. 18. Program console output

The most optimal card to choose is 7 of Diamonds, which has the lowest total weight of 2.04 to Greedy Joker, indicating that it best complements the current jokers in possession and maximizes synergy potential. This algorithm proves to be effective because the Greedy Joker provides a +3 multiplier for every diamond card played, making the 7 of Diamonds a valuable addition to the player's deck.

V. SUMMARY

In this research, we have successfully modeled the problem of selecting the optimal card from a booster pack in Balatro as a weighted, layered directed acyclic graph. By leveraging Dijkstra's algorithm, we can efficiently determine the card that best complements the player's jokers based on synergy potential. The implementation demonstrates how graph theory can be applied to strategic decision-making in unique card games, providing a systematic approach to optimize gameplay outcomes. Although with its limitation of only considering jokers with synergies related to traits or hands, this approach can be extended to include more complex interactions and additional game mechanics in the future, such as considering for enhanced cards like glass cards or bonus cards, or upgraded hand levels that can change the synergy dynamics. The results show that the algorithm can effectively identify the optimal card to choose, enhancing the player's strategic options and overall gameplay experience in the hit game Balatro.

VI. APPENDIX

As an additional resource, below is a GitHub gist link containing the code used to simulate this model in Python.
https://gist.github.com/Cikang44/6d628a01e69abc461372a5c589fc2c7f

References

[1] "Poker Hands Rankings," World Series of Poker, Accessed: June 20, 2025. [Online]. Available: https://www.wsop.com/poker-hands

[2] LocalThunk, "Balatro," Steam, Accessed June 20, 2025. [Online]. Available: https://store.steampowered.com/app/2379780/Balatro/

[3] G. Morley, "Balatro cast a magic spell that made me like math," Polygon, Accessed: June 20, 2025. [Online]. Available: https://www.polygon.com/reviews/24084564/balatro-review-poker-game-deck-building-roguelite

[4] Munir, Rinaldi. http://informatika.stei.itb.ac.id/~rinaldi.munir/

[5] K. Thulasiraman and M. N. S. Swamy, *Graphs: Theory and Algorithms*, John Wiley & Sons, 1992, ch. 5.7, pp. 118.

[6] "ARMAC," *Unsung Heroes in Dutch Computing History*, 2007. Archived from the original on November 13, 2013.

[7] K. Mehlhorn and P. Sanders, *Algorithms and Data Structures: The Basic Toolbox*, Springer, 2008, ch. 10, "Shortest Paths." DOI: 10.1007/978-3-540-77978-0.

[8] "Every Balatro Joker," *Balatro Wiki (IGN)*, accessed June 20 2025. Available: https://www.ign.com/wikis/balatro/Every_Balatro_Joker

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Juni 2025

[illegible]

Jingglang Galih Rinenggan
13524095