Application of Heuristic Search To Solve The Set Packing Problem In Assigning Cookie Run: Kingdom Mining Teams

Billy Ontoseno Irawan - 13524121
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: billy.ontoseno@gmail.com , 13524121@std.stei.itb.ac.id

***Abstract*— In contemporary mobile gaming, strategic decision-making with randomized constraints is getting increasingly prevalent. Cookie Run: Kingdom, a popular role-playing game developed by Devsisters, features a mining mini-game where players assign a limited pool of unique in-game characters commonly referred as Cookies to mine various types of ore veins. Each ore vein comes with a randomized requirement based on traits or combat roles. This assignment process presents a non-trivial optimization problem in which players must select optimal teams of up to four Cookies per vein, without reusing any single Cookie across up to six concurrent veins. Additionally, certain traits give performance bonuses when matched to specific vein types. This assignment problem closely resembles the set packing problem, a well-known combinatorial optimization problem where the goal is to select the largest possible collection of disjoint subsets from a universe such that each subset satisfies specific criteria. Given its NP-complete nature, exploring the solution to this problem requires the use of heuristic search.**

***Keywords*—Cookie Run: Kingdom, resource allocation, set packing, heuristic search**

I. Introduction

In recent years, the proliferation of mobile games has transformed not only entertainment consumption patterns but also the computational structures embedded in game design. One emerging trend is the incorporation of strategic team-building with randomized constraints that mirror classic combinatorial optimization challenges. *Cookie Run: Kingdom* by Devsisters is one such example of this design ethos. Among its multiple gameplay layers, the mining mini-game stands out for its intricate character allocation system, which requires players to match "Cookies" to ore veins under a variety of randomized and categorical constraints such as personality traits and combat roles.

This system of constrained optimization in mobile gaming mirrors the set packing problem, a canonical NP-complete problem in computational theory. In this context, players must form mutually exclusive subsets (teams of up to four Cookies) across multiple mining sites, ensuring that no Cookie is reused while also maximizing effectiveness under the randomized vein requirements. The inclusion of traits that enhance performance introduces additional nonlinearities, making brute-force or deterministic solutions computationally infeasible. Similar challenges appear in scheduling, resource allocation, and task assignment in operations research, where approximate solutions are often pursued using heuristic or metaheuristic methods [1], [2].

The need to balance performance bonuses from trait matching with the exclusive allocation constraint introduces a multi-objective optimization challenge. Players must not only maximize mining yield but also navigate the probability distribution of future vein requirements, leading to the development of greedy heuristics, rule-based pruning, and probabilistic modelling within the player community. This mirrors real-world scenarios in logistics and scheduling, where similar decision-making frameworks are employed to handle conflicting resource constraints under uncertainty [3].

From a research perspective, the problem posed by *Cookie Run: Kingdom* provides a sandbox for investigating heuristic search strategies under bounded rationality. As more games introduce AI-like complexity in decision-making under randomized constraints, there is growing interest in how players intuitively solve these problems, often approximating solutions to NP-complete problems through domain-specific heuristics. This gamified form of constrained optimization offers a practical and engaging entry point for exploring algorithmic thinking and the effectiveness of human-in-the-loop computation models [4].

II. Theoretical Framework

A. Set Theory

A set is a collection of distinct, unordered objects called elements or members. Elements within a set must be different from each other and are not necessarily related. If x is a member of set A, it is written as $x \in A$; otherwise, $x \notin A$. Specifying the members of a set can be done through either listing each member one by one (also known as roster notation) or using the set builder notation $\{x \mid \Phi(x)\}$ where Φ is a logical formula that evaluates to true for x.

A collection of sets can be called disjoint if any two distinct sets of the collection have no member in common. Formally, sets A and B are disjoint if $A \cap B = \emptyset$.

A set A is a subset of set B if every element of A is also an element of B. If there is at least one element present in B but not in A, then A is a proper subset of B. Formally, the definition of a subset can be expressed as

$$A \subseteq B \Leftrightarrow \forall x(x \in A \rightarrow x \in B) \quad (1)$$

To differentiate between proper subset and subset, a proper subset is denoted as $A \subset B$.

The power set of a set A, denoted *P(A)* or alternatively 2^{A}, is the set containing all subsets of set A including the empty set and set A itself [5].

B. *Set Packing Problem*

Set-packing problems arise in applications where there are strong constraints on an allowable partition. The key feature of packing problems is that no elements are permitted to be covered by more than one selected subset [6]. As an example, consider the following universe of elements U = {1,2,3,4,5} and subsets:

- S_1 = {1,2}
- S_2 = {2,3}
- S_3 = {4,5}
- S_4 = {3,5}

The goal is to select a collection of subsets from S_1, S_2, S_3, and S_4 such that no element in U is repeated across any of the selected subsets. Some feasible solutions are selecting {S_1, S_3}, {S_1, S_4}, and {S_2, S_3} since these subsets are pairwise disjoint. However, if given a constraint where the elements must be >1, then the only valid solution would be {S_2, S_3}

C. *Heuristic Search*

Heuristic search is a class of algorithms designed to find solutions in large or complex state spaces by using domain-informed estimates (heuristics) to guide the search process. Unlike exhaustive methods, heuristic approaches prioritize exploration toward promising regions of the solution space, thereby improving computational efficiency at the potential cost of optimality.

A heuristic function *h(n)* estimates the cost from a node *n* to the nearest goal. A heuristic function is said to be admissible if it never overestimates the cost of reaching the goal, i.e. the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path [7].

Heuristic search encompasses several algorithmic families, classified by their operational constraints [8]:

- Best-First Search (BFS) prioritizes nodes with the lowest heuristic value. Greedy variants ignore path cost, relying on *f(n) = h(n)*.
- A* combines path cost *g(n)* and heuristic *h(n)*, expanding nodes with the lowest *f(n) = g(n)+h(n)*. It is optimal if *h* is admissible.
- Iterative Deepening A* (IDA*) recursively applies depth-limited A* with increasing thresholds, offering memory efficiency.
- Real-Time Search Algorithms like LRTA* interleave planning and acting, suitable for applications with time constraints or limited memory.
- Anytime Heuristics produce suboptimal solutions quickly and refine them given more time (e.g., Anytime Repairing A*)

D. *Cookie Run: Kingdom*

Cookie Run: Kingdom is the eighth game in the Cookie Run series developed and published by South Korean studio Devsisters, releasing worldwide on January 21st, 2021. A departure from the runner gameplay of the original, it features both real-time battle strategy and city-building, with a wide cast of unique Cookies and a customizable kingdom [9].

Fig. 1. Cookie Run: Kingdom
(source: https://www.devsisters.com/en/games/Cookierun-kingdom)

The game's gameplay loop revolves around collecting and upgrading characters and their equipment to successfully battle various enemies. Players may acquire new Cookies through the gacha system using in-game currency known as Cookie Cutters. Every Cookie belongs to one out of eight types (also colloquially referred to as classes or roles) that determines their function in battle, whether it be damage-dealing, buffing/debuffing, or healing [10]. Upgrading characters and equipment requires specific resources, which can be obtained by progressing through the main story, participating in time-limited events, completing side quests, and more.

E. *Into The Depths*

Into the Depths is a Kingdom facility introduced in the *The Flame Awakens* update (v6.3). It is where players can access the Mines and unlock various rewards earned through venturing the mines or mining ore veins. Mining ore veins rewards Crystals, Choco Chalk, Coins, Kingdom EXP, Topping Tart components, Fossils with various rewards, and Agar Cubes that contain powerful Soulstones or Soulprisms [11]. Prior to mining ore veins, the player must dispatch up to four Cookies to venture the mines. Upon successful completion of this expedition, the player is rewarded with ore vein cards, among other potential items. A maximum of six ore vein cards may be processed simultaneously, with the time required for completion increasing in accordance with the rarity of each ore vein.

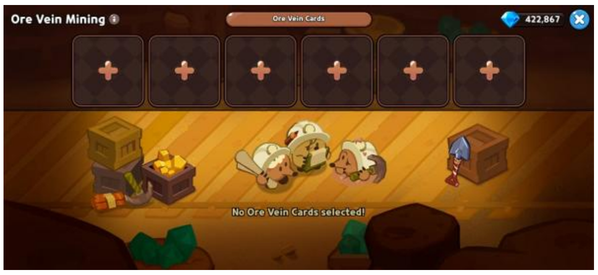


Fig. 2. Ore Vein Mining UI (image by author)

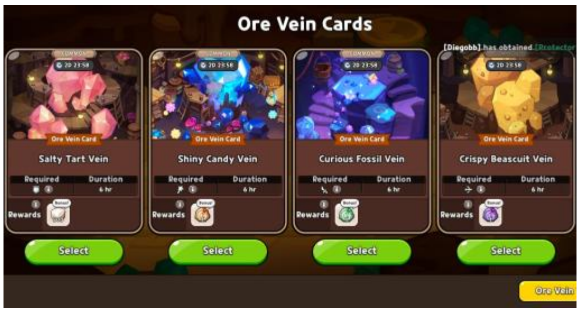


Fig. 3. Ore Vein Cards (image by author)

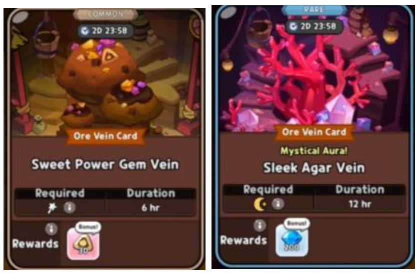


Fig. 4. Sweet Power Gem Vein and Sleek Agar Vein (image by author)

There are six distinct types of ore veins (refer to Fig. 3 and Fig. 4), each classified into one of three rarity levels: common, rare, or epic. The rewards obtained from these ore veins vary depending on their type, while the time required to mine them increases with their rarity. In order to mine an ore vein, the player must assign up to four Cookies that meet the specified requirements; either a designated role for common ore veins or a specific trait for rare and epic ore veins.

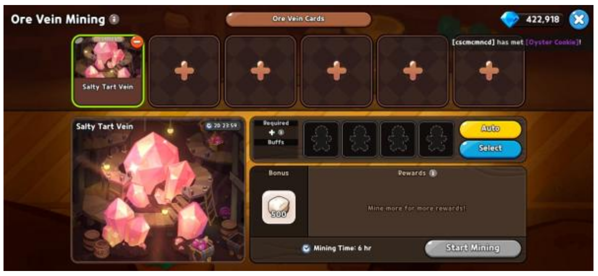
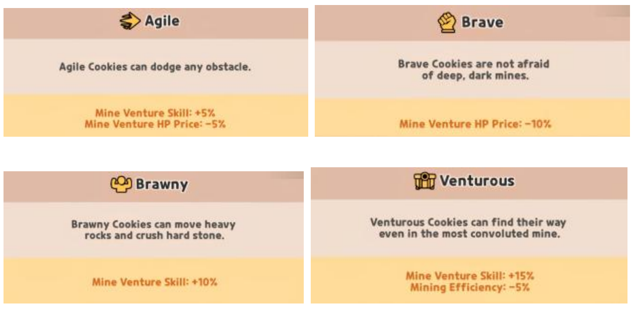


Fig. 5. An example of a yet to be mined ore vein (image by author)

F. *Cookie Traits*

Each Cookie possesses two predetermined traits, with the exception of Adventurer Cookie and Agar-Agar Cookie. Certain traits enhance effectiveness in the Mine Venture mode, while others improve the ability to mine Ore Veins. Some traits provide benefits to both modes simultaneously. Additionally, specific traits may grant a buff in one mode while imposing a debuff in the other. A comprehensive list of these traits grouped by their buffs is provided below:



Agile

Agile Cookies can dodge any obstacle.

Mine Venture Skill: +5%
Mine Venture HP Price: −5%

Brave

Brave Cookies are not afraid of deep, dark mines.

Mine Venture HP Price: −10%

Brawny

Brawny Cookies can move heavy rocks and crush hard stone.

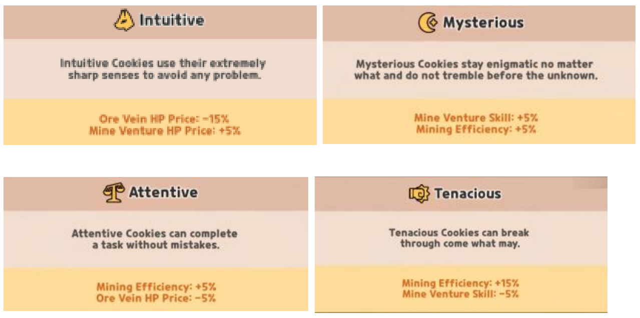
Mine Venture Skill: +10%

Venturous

Venturous Cookies can find their way even in the most convoluted mine.

Mine Venture Skill: +15%
Mining Efficiency: −5%

Fig. 6. Exclusive Mine Venture Traits (image by author)



Intuitive

Intuitive Cookies use their extremely sharp senses to avoid any problem.

Ore Vein HP Price: −15%
Mine Venture HP Price: +5%

Mysterious

Mysterious Cookies stay enigmatic no matter what and do not tremble before the unknown.

Mine Venture Skill: +5%
Mining Efficiency: +5%

Attentive

Attentive Cookies can complete a task without mistakes.

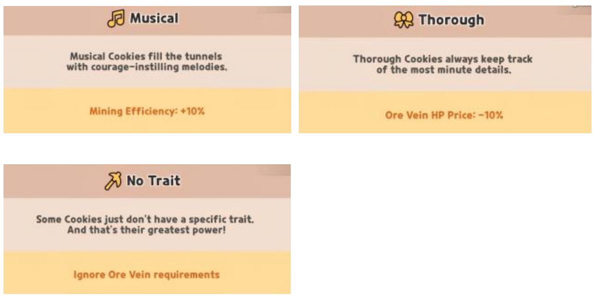
Mining Efficiency: +5%
Ore Vein HP Price: −5%

Tenacious

Tenacious Cookies can break through come what may.

Mining Efficiency: +15%
Mine Venture Skill: −5%

Fig. 7. Mine Venture and Ore Vein Mining Traits (image by author)



Musical

Musical Cookies fill the tunnels with courage-instilling melodies.

Mining Efficiency: +10%

Thorough

Thorough Cookies always keep track of the most minute details.

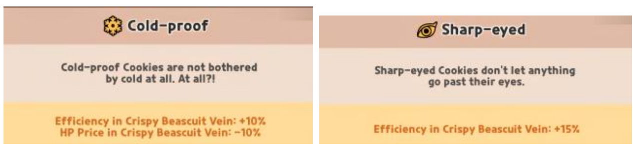
Ore Vein HP Price: −10%

No Trait

Some Cookies just don't have a specific trait. And that's their greatest power!

Ignore Ore Vein requirements

Fig. 8. Exclusive Ore Vein Mining Traits (image by author)



Cold-proof

Cold-proof Cookies are not bothered by cold at all. At all?!

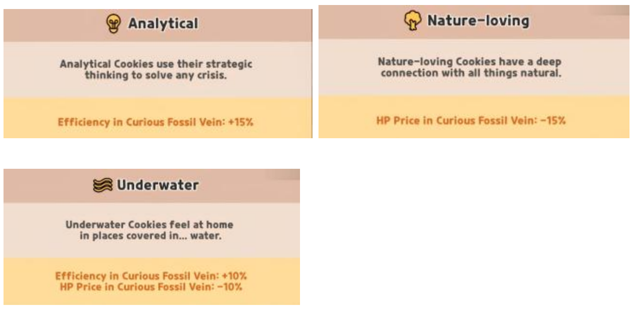
Efficiency in Crispy Beascuit Vein: +10%
HP Price in Crispy Beascuit Vein: −10%

Sharp-eyed

Sharp-eyed Cookies don't let anything go past their eyes.

Efficiency in Crispy Beascuit Vein: +15%

Fig. 9. Crispy Beascuit Vein Traits



Analytical

Analytical Cookies use their strategic thinking to solve any crisis.

Efficiency in Curious Fossil Vein: +15%

Nature-loving

Nature-loving Cookies have a deep connection with all things natural.

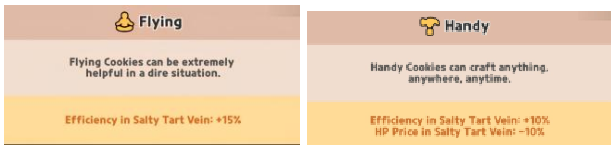
HP Price in Curious Fossil Vein: -15%

Underwater

Underwater Cookies feel at home in places covered in... water.

Efficiency in Curious Fossil Vein: +10%
HP Price in Curious Fossil Vein: -10%

Fig. 10. Curious Fossil Vein Traits (image by author)



Flying

Flying Cookies can be extremely helpful in a dire situation.

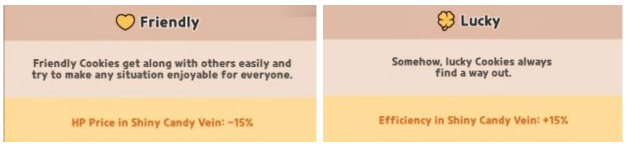
Efficiency in Salty Tart Vein: +15%

Handy

Handy Cookies can craft anything, anywhere, anytime.

Efficiency in Salty Tart Vein: +10%
HP Price in Salty Tart Vein: -10%

Fig. 11. Salty Tart Vein Traits (image by author)



Friendly

Friendly Cookies get along with others easily and try to make any situation enjoyable for everyone.

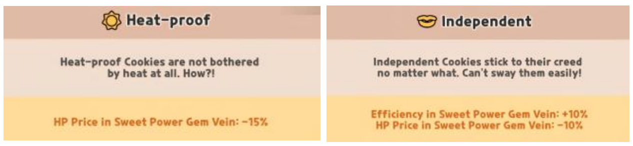
HP Price in Shiny Candy Vein: -15%

Lucky

Somehow, lucky Cookies always find a way out.

Efficiency in Shiny Candy Vein: +15%

Fig. 12. Shiny Candy Vein Traits (image by author)



Heat-proof

Heat-proof Cookies are not bothered by heat at all. How?!

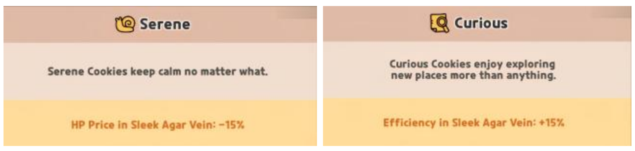
HP Price in Sweet Power Gem Vein: -15%

Independent

Independent Cookies stick to their creed no matter what. Can't sway them easily!

Efficiency in Sweet Power Gem Vein: +10%
HP Price in Sweet Power Gem Vein: -10%

Fig. 13. Sweet Power Gem Vein Traits (image by author)



Serene

Serene Cookies keep calm no matter what.

HP Price in Sleek Agar Vein: -15%

Curious

Curious Cookies enjoy exploring new places more than anything.

Efficiency in Sleek Agar Vein: +15%

Fig. 14. Sleek Agar Vein Traits (image by author)

III. Mathematical Model

The cookie-vein assignment problem can be modeled as a structured set packing integer linear program (ILP). Let C denote the set of cookies and V the set of veins. For each vein $v \in V$ define

$$T_v \subseteq 2^C \tag{2}$$

as the collection of valid cookie subsets that meet vein v's requirement, each of size at most 4. A valid subset $t \in T_v$ must consist of cookies that satisfy either the required trait or role for vein v. Each such subset is assigned a score $s_{v,t}$ calculated by summing a base score and bonuses derived from trait matches, including vein-specific bonuses and universal bonuses.

The decision variable $x_{v,t} \in \{0,1\}$ indicates whether subset t is selected for vein v. The objective function maximizes the total score across all veins:

$$max \sum_{v \in V} \sum_{t \in T_v} s_{v,t} \cdot x_{v,t} \tag{3}$$

Subject to the constraints that each vein can receive at most one subset:

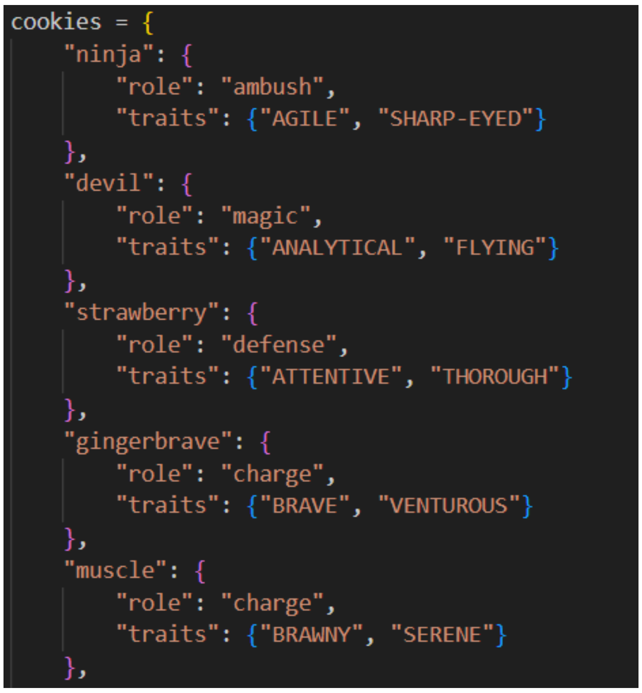
$$\sum_{t \in T_v} x_{v,t} \leq 1 \quad \forall v \in V \tag{4}$$

and that each cookie appears in at most one selected subset:

$$\sum_{v \in V} \sum_{t \in T_v : c \in t} x_{v,t} \leq 1 \quad \forall c \in C \tag{5}$$

IV. Heuristic Search Algorithm

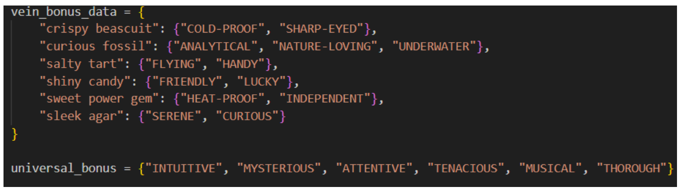
Let the set of Cookies be represented as a dictionary, where each key is a cookie's name and the associated value contains two fields: role (a string) and traits (a set of strings)



```
cookies = {
    "ninja": {
        "role": "ambush",
        "traits": {"AGILE", "SHARP-EYED"}
    },
    "devil": {
        "role": "magic",
        "traits": {"ANALYTICAL", "FLYING"}
    },
    "strawberry": {
        "role": "defense",
        "traits": {"ATTENTIVE", "THOROUGH"}
    },
    "gingerbrave": {
        "role": "charge",
        "traits": {"BRAVE", "VENTUROUS"}
    },
    "muscle": {
        "role": "charge",
        "traits": {"BRAWNY", "SERENE"}
    },
```

Fig. 15. A snippet of the representation of the set of Cookies (image by author)

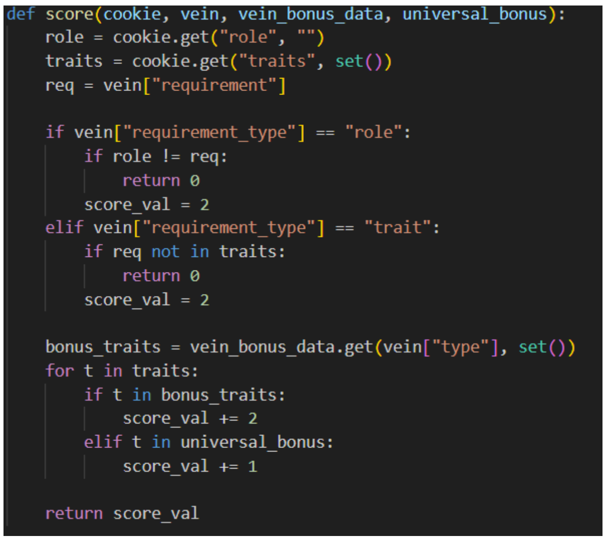
The dictionary of veins in this model represents different types of mining environments as opposed to the set of to-be mined veins themselves, as those are stored in a separate array after user input. Each vein is identified by a name and is associated with a set of vein-specific bonus traits, which grant additional score when matched by a cookie's traits. Additionally, there is a defined set of universal bonus traits that provide a moderate scoring boost across all vein types.



```python
vein_bonus_data = {
    "crispy beascuit": {"COLD-PROOF", "SHARP-EYED"},
    "curious fossil": {"ANALYTICAL", "NATURE-LOVING", "UNDERWATER"},
    "salty tart": {"FLYING", "HANDY"},
    "shiny candy": {"FRIENDLY", "LUCKY"},
    "sweet power gem": {"HEAT-PROOF", "INDEPENDENT"},
    "sleek agar": {"SERENE", "CURIOUS"}
}

universal_bonus = {"INTUITIVE", "MYSTERIOUS", "ATTENTIVE", "TENACIOUS", "MUSICAL", "THOROUGH"}
```

Fig.16. Dictionary of Vein Types (image by author)

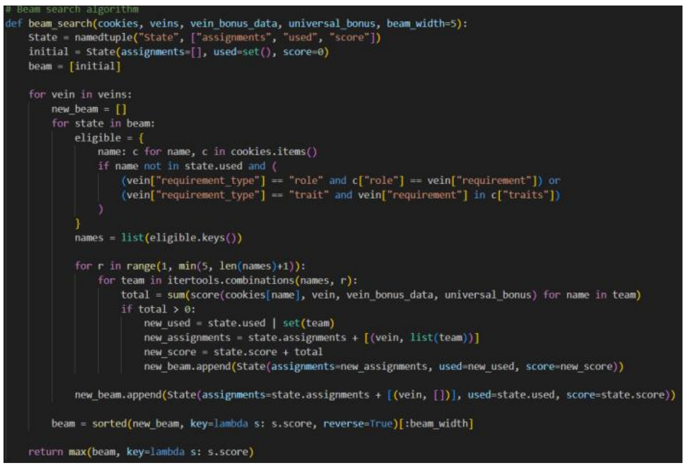
For simplicity, the score $s_{v,t}$ is calculated as follows: +2 points are awarded if a cookie matches the vein's required role or trait, +2 points are given for each trait that matches the vein-specific bonus traits (as outlined in Fig.9. through Fig. 14), and +1 point for every other trait that is not among the exclusive mine venture traits (refer to Fig.6) as these do not contribute to the mining efficiency score. Although Venturous technically imposes a -5% mining efficiency debuff, this is consistently offset by the presence of a corresponding vein-specific trait in all cookies that possess it, effectively neutralizing its penalty.



```python
def score(cookie, vein, vein_bonus_data, universal_bonus):
    role = cookie.get("role", "")
    traits = cookie.get("traits", set())
    req = vein["requirement"]

    if vein["requirement_type"] == "role":
        if role != req:
            return 0
        score_val = 2
    elif vein["requirement_type"] == "trait":
        if req not in traits:
            return 0
        score_val = 2

    bonus_traits = vein_bonus_data.get(vein["type"], set())
    for t in traits:
        if t in bonus_traits:
            score_val += 2
        elif t in universal_bonus:
            score_val += 1

    return score_val
```

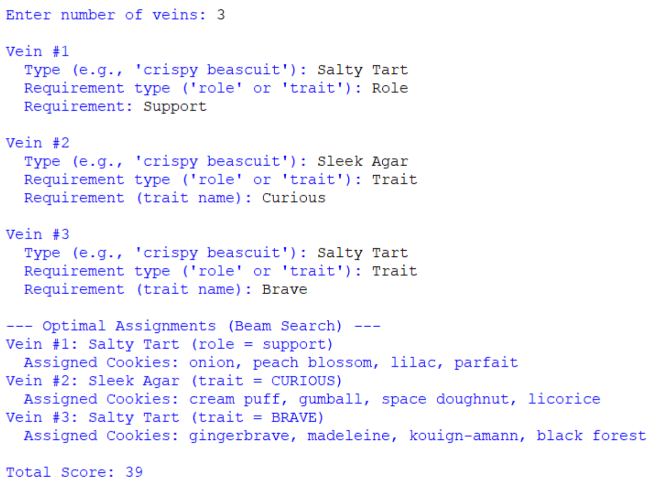
Fig.17. Scoring function (image by author)

Beam search is chosen for this problem because it offers an effective balance between solution quality and computational efficiency. Unlike greedy algorithms that make irreversible local decisions that can lead to suboptimal global outcomes, beam search maintains multiple partial solutions at each step, thereby enabling limited backtracking. This is particularly valuable in this context where early assignments can constrain or enable higher-value combinations in later veins. While exact methods like integer linear programming (ILP) can guarantee optimality, they become computationally expensive as the number of veins and cookies increases. By limiting the number of candidate paths explored at each depth (via the beam width), beam search scales more gracefully while still capturing globally beneficial patterns that greedy approaches often miss. This makes it especially suitable for structured set packing problems where disjoint assignments and multi-dimensional scoring need to be reconciled efficiently.



```python
# Beam search algorithm
def beam_search(cookies, veins, vein_bonus_data, universal_bonus, beam_width=5):
    State = namedtuple("State", ["assignments", "used", "score"])
    initial = State(assignments=[], used=set(), score=0)
    beam = [initial]

    for vein in veins:
        new_beam = []
        for state in beam:
            eligible = {
                name: c for name, c in cookies.items()
                if name not in state.used and (
                    (vein["requirement_type"] == "role" and c["role"] == vein["requirement"]) or
                    (vein["requirement_type"] == "trait" and vein["requirement"] in c["traits"])
                )
            }
            names = list(eligible.keys())

            for r in range(1, min(5, len(names)+1)):
                for team in itertools.combinations(names, r):
                    total = sum(score(cookies[name], vein, vein_bonus_data, universal_bonus) for name in team)
                    if total > 0:
                        new_used = state.used | set(team)
                        new_assignments = state.assignments + [(vein, list(team))]
                        new_score = state.score + total
                        new_beam.append(State(assignments=new_assignments, used=new_used, score=new_score))

            new_beam.append(State(assignments=state.assignments + [(vein, [])], used=state.used, score=state.score))

        beam = sorted(new_beam, key=lambda s: s.score, reverse=True)[:beam_width]

    return max(beam, key=lambda s: s.score)
```

Fig. 18. Beam Search Algorithm (image by author)



```
Enter number of veins: 3

Vein #1
  Type (e.g., 'crispy beascuit'): Salty Tart
  Requirement type ('role' or 'trait'): Role
  Requirement: Support

Vein #2
  Type (e.g., 'crispy beascuit'): Sleek Agar
  Requirement type ('role' or 'trait'): Trait
  Requirement (trait name): Curious

Vein #3
  Type (e.g., 'crispy beascuit'): Salty Tart
  Requirement type ('role' or 'trait'): Trait
  Requirement (trait name): Brave

--- Optimal Assignments (Beam Search) ---
Vein #1: Salty Tart (role = support)
  Assigned Cookies: onion, peach blossom, lilac, parfait
Vein #2: Sleek Agar (trait = CURIOUS)
  Assigned Cookies: cream puff, gumball, space doughnut, licorice
Vein #3: Salty Tart (trait = BRAVE)
  Assigned Cookies: gingerbrave, madeleine, kouign-amann, black forest

Total Score: 39
```

Fig.19. Example Test Case (image by author)

V. Conclusion

The mining mini-game in *Cookie Run: Kingdom* presents a compelling instance of a combinatorial optimization problem analogous to the set packing problem. By modeling the cookie assignment task using set theory and applying a heuristic search algorithm, this paper demonstrates how computational techniques can be effectively applied to game mechanics that simulate real-world optimization challenges. The beam search algorithm, with its balance between performance and computational feasibility, proves to be a suitable approach.

Beyond its immediate application, this paper highlights how complex decision-making problems in entertainment platforms can serve as engaging case studies for heuristic search and constrained optimization strategies in discrete mathematics and computer science education.

VI. Appendix

The source code of the program used in this paper can be found at GitHub: https://github.com/BillyOWasTaken/crk-veinminer

References

[1] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 5th ed. New York, NY, USA: Springer, 2016.

[2] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Chichester, UK: Wiley, 1990.

[3] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Hoboken, NJ, USA: Wiley, 2009.

[4] H. A. Simon, *Models of Bounded Rationality*, Vol. 1. Cambridge, MA, USA: MIT Press, 1997.

[5] R. Munir, “Program Studi Teknik Informatika STEI-ITB,” 2025. Accessed: Jun. 17, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025-2/02-Himpunan(2025)-1.pdf

[6] S. S. Skiena, *The Algorithm Design Manual*. London Springer London, 2008.

[7] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. London, UK: Pearson, 2021

[8] S. Edelkamp and S. Schrödl, *Heuristic Search: Theory and Applications*. Amsterdam ; Boston: Morgan Kaufmann, C, 2011.

[9] Community, “Cookie Run: Kingdom Wiki,” February 1, 2022. Accessed: Jun. 18, 2025 [Online]. Available: https://Cookierunkingdom.fandom.com/wiki/Cookie_Run:_Kingdom_Wiki

[10] Community, “Cookie Type,” December 23, 2023. Accessed: Jun. 18, 2025 [Online]. Available: https://Cookierunkingdom.fandom.com/wiki/Cookie_Type

[11] Community, “Into the Depths,” April 10, 2025. Accessed: Jun. 18, 2025 [Online]. Available: https://Cookierunkingdom.fandom.com/wiki/Into_the_Depths

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

billy

Billy Ontoseno Irawan
13524121