Demacia's Greedy Tactic

An Implementation of Greedy Algorithm to Play *Legends of Runeterra*

Billy Julius - 13519094
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : billyjulius01@gmail.com

***Abstract*—Games are now part of the digital world, being one of the most used type of application. There are many strategies used when playing a game, especially one that utilizes strategy, such as card game. Sometimes, we play a game based on solely our intuitive, while on other times, we would like to write things out of our mind and make our strategy clear and visible. One of the most famous algorithms which can be used is greedy algorithm. In short, greedy algorithm relies on taking actions based on what is best for the current situation. This paper shows how a greedy algorithm can be implemented as a strategy to play the famous card game, *Legends of Runeterra*.**

Keywords—Games, Strategy, Legends of Runeterra, Greedy Algorithm

I. INTRODUCTION

Games have always been a part of the society, since ancient times. Several well-known games, which some are still played today, are boxing, chariot racing, wrestling, discus throw, jumping, horse racing, and javelin throwing [1]. This shows how games are basically ways for people to have fun since the far past.

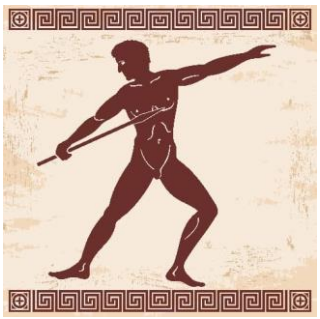

Figure 1. Javelin Throwing as An Ancient Game (Source: Top 9 Popular Games of Ancient Greece, Saugat Adhikari[1])

Nowadays, games have improved and developed much. One of the concrete and visible proofs of this is the existence of digital games. People are able to play and interact with each other even when they are not physically close, using the internet as a medium to transfer data from each device. This kind of games are made by programmers who determines how computers react when given certain input. One of the most famous genre of digital games is collectible card game.

Collectible card game (CCG) is a kind of game which revolves around a set of card collection. Player may collect their own cards and use them when playing with others. The main idea of CCG is how each player build their own deck creatively and use it to beat their opponent. Usually, CCG type of games is played based on turn where players play on their own turn and wait for a turn for their opponent to play.

One of the most played CCG is *Legends of Runeterra* (LoR). LoR is a free-to-play digital collectible card game developed by Riot Games. Over the past year, *Legends of Runeterra* has become one of the most growing digital collectible card game. This has been proven by more and more twitch streamers streaming LoR as their favorite game [2]. Riot Games has also been very active updating the game by expanding collections and fixing bugs, showing how lively the game is.

Figure 2. Legends of Runeterra Poster (Source: https://playruneterra.com/en-sg/)

In this paper, the author describes one of greedy algorithm implementation to play *Legends of Runeterra* (LoR). Greedy Algorithm has been one of the most well-known algorithms to solve various problems in computer science knowledge, such as activity selection problem, integer knapsack problem, fractional knapsack problem, and many more.

Playing LoR is a challenging but also exciting task. In order to outplay the opponent, one must observe the situation thoroughly, expecting and predicting every single move the opponent may make and play a card accordingly. There are many tactics and strategy throughout the *Runeterra* world, but in this paper, the author will describe and explain one greedy strategy using *Demacia* region which is known for its overwhelming troops' statistics.

II. Theoretical Framework

As mentioned before, before going to the actual greedy approach, we shall learn about Greedy Algorithm. In addition, this section will also describe the general rules and gameplay of *Legends of Runeterra*.

A. *Greedy Algorithm*

Greedy Algorithm solves its problem entirely by taking what is best in each step. The decision in each step is made hoping that the local optima will also be global optima [3].

There are some components making up the Greedy Algorithm [3].

1) *Candidate Set*
   The candidate set is the "source" set containing candidates to be picked for the greedy solution. Each candidate is differed by specific identification based on what problem the greedy algorithm is being used on. Later, a candidate will be picked from this candidate set, using the selection function, making progress towards the solution.
2) *Solution Set*
   The solution set will contain current chosen candidates which will be checked, later on, by solution function whether the set has already met what the problem demands. Each candidate picked from the candidate set will be added to this solution set, thus making the solution set subset of the candidate set.
3) *Solution Function*
   The solution function acts as a checker whether the current solution set is already a viable solution to the problem. The algorithm shall stop when this function returns a true statement.
4) *Selection Function*
   The selection function is used whenever we need to choose a candidate from the candidate set. This function determines how we pick each candidate to be added to the solution set, having checked by the feasibility function beforehand. This component is the main idea of how the greedy algorithm works, in this case, choosing over a candidate.
   As mentioned before the greedy algorithm hopes for the local optima to also be the global optima. This will not always be the case. Through this function, the candidate is chosen on the current state, in other words, showing no interest on future choices. On the contrary, sometimes, to reach a globally optima, we may need to take a non-local optimum candidate. That is why the greedy algorithm may produce a solution different from the best solution.
5) *Feasibiliy Function*
   The feasibility function is called after the selection function determines which candidate would be added to the solution set. This function will check whether the candidate is feasible by using some specific constraints. The constraints differ among each problem and are usually described by the problem. When the candidate does not meet the requirements stated by this function, other candidate shall be chosen.
6) *Objective Function*
   The objective function refers to the overall goal of the greedy algorithm, maximization or minimization. This functions states the problem and what needs to be optimized.

To sum things up, greedy algorithm searches for a subset of the candidate set, known as the solutions set, whereas the solution set needs to meet several specific criteria and that solution set is optimized by the objective function [3].

B. *Legends of Runeterra*

Legends of Runeterra holds many complicated mechanics and thus will be explained in this section.

1) *Cards*
   In a card game, cards are where all revolves around. They are used to execute a *move*, progressing the game when played. Each card has their own unique characteristic and it does take some time to remember each one.
   There are 4 types of cards: *Champions*, *Followers*, *Spells*, and *Landmarks*. Champions are cards that are supposed to be more valuable than the others as they hold great effects. A champion is leveled up when it meets certain unique condition. When they level up, their presence in the game board increase significantly and usually act as a game changer. As drawback, a deck can only consist of maximum 6 champion cards.

Figure 3. Garen as A Champion Card
(Source: https://lor.mobalytics.gg/cards/01DE012)

Followers are units, besides champions, that can be played and summoned to the board. Each unit, including champions, has two statistic characteristics:

Power and *Health*. Power determines how many damage a unit deal when battling. Health, on the other hand, determines how many damage a unit can hold when receiving damage before it dies. It costs one turn to play a unit.

Figure 4. Cithria The Bold as A Follower Card (Source: https://lor.mobalytics.gg/cards/01DE051)

Spells are cards that can be casted in various speed: *Burst*, *Fast*, *Focus*, and *Slow*. Burst spells allows the player to play another card after the spell revolves. It can also be casted in response to other slower spells or when a battle is taking place. Fast spells can be casted in response but costs a turn to revolve its effect. Focus spells can't be casted in response but it allows the player to move again after its effect has revolved. Slow spells, acting as the lowest rank in the spell speed hierarchy, can't be casted in response while also costs one turn to revolve its effect. Each spell effect is balanced to its speed to maintain a comfortable and just gameplay.

Figure 5. For Demacia! as A Spell Card (Source: https://lor.mobalytics.gg/cards/01DE035)

Landmarks is a new one, they were added in a recent update. Landmarks can be played onto the board but have no power and health. Some of them do have *countdowns* – a number of turns needed for their effect to revolve. When their effect has been revolved, landmarks are destroyed, leaving the board immediately. There are also some landmarks having no countdown, meaning their effect revolves unlimitedly, such as *The Grand Plaza* from *Demacia*.

Figure 6. The Grand Plaza as A Landmark Card (Source: https://lor.mobalytics.gg/cards/03DE010)

2) *Keywords*

There are literally a lot of keywords existing in *Legends of Runeterra* mechanics. They act as an explanation of how a card behaves specially in certain condition.

In general, there are 4 types of keywords: *Actions*, *Conditions*, *Triggers*, and *Static Keywords* [4]. Actions are keywords that represent specific procedures that a player or card can do. Action keywords include *advance*, *blade dance*, *capture*, *discard*, *drain*, *draw*, *invoke*, *nab*, *obliterate*, *predict*, *rally*, *recall*, *reforge*, *revive*, and *toss*.

Conditions are keywords that refer to the game state. Condition keywords include *attack*, *behold*, *block*, *death*, *deep*, *enlightened*, *level up*, *play*, *see*, *slay*, *strike*, *strongest*, *summon*, and *weakest*.

Triggers are keywords describing what a card can do when a certain trigger is done. Trigger keywords include *allegiance*, *attack*, *aura*, *block*, *countdown*, *daybreak*, *imbue*, *last breath*, *nexus strike*, *nightfall*, *play*, *plunder*, *reputation*, *round end*, *round start*, *strike*, *summon*, and *support*.

Static keywords are distinct and require no other card to revolve its effect. Static keywords are *attune*, *augment*, *barrier*, *challenger*, *deep*, *elusive*, *ephemeral*, *fearsome*, *fleeting*, *frostbite*, *fury*, *immobile (can't attack, can't block)*, *lifesteal*, *overwhelm*, *quick attack (double attack)*, *regeneration*, *scout*, *silence*, *spellsheild*, *stun*, *tough*, and *vulnerable*.

To understand the game thoroughly, one must understand each keyword existing in a card completely well. This is done to predict what cards the opponent is holding and what they can do with their cards.

3) *Regions*

The *runeterra* world currently consists of 9 regions: *Bilgewater*, *Demacia*, *Freljord*, *Ionia*, *Noxus*, *Piltover & Zaun*, *Shadow Isles*, *Shurima*, and *Targon*. Regions state what cards are available for a deck to use. One deck can only use cards from up to 2 regions. Each region has their own unique characteristics and therefore describe their deck gameplay style. For example, *Demacia* is known for its solid units and later on threats to overwhelm the board presence. *Freljord*, on the other hand, is frequently used in a *control*

manner, meaning to hold enemies' attack and dropping high cost units later on in the late game, outvaluing the opponent.

4) *Decks*

A player goes to the play with one deck, consisting of 40 cards. Each player can build their own deck creatively and play the game on their own style. Deck-building is also a skill when it comes to card game.

5) *Play*

The play consists of each player taking turns having the *attack token*. The *attack token* allows one player to attack and threatening the enemy *nexus*. Nexus is what defines the player's life, when it is destroyed, a player lose the game. In the beginning, each player starts with a nexus of 20 health points.

When a player attacks, a battle revolves. The opposing side can block each attacking unit with their own unit. An unblocked attack goes through and hit the nexus, damaging it based on the unit's power.

Each turn, a player draws a card from their own deck to the hand. They would also get *mana* based on how many turns have passed. Any unused mana will be converted to *spell mana* (up to three) which can only be consumed to cast spells.

The play will continue until one of these conditions are met: one of the player's nexus has been reduced to zero, one of the player's deck is out of cards, or the play has reached 40 rounds thus resulting in a draw.

There are still many complicated mechanics that is not explained in this paper. For the strategy purpose, the author will only explain relevant keywords and mechanics toward the gameplay. For further and better understanding, playing the game directly may be worth a try.

III. A Greedy Approach to Play Legends of Runeterra

In this section, we shall map each of greedy algorithm component to the gameplay of *Legends of Runeterra*.

A. Choosing Cards to Play

1) *Candidate Set*

Cards the player is holding will be treated as the candidate set. This is based on the choice the player makes deciding which card to play on each turn. We also know that each turn the player would draw one card from their own deck. As a result, the candidate set will be constantly changing – decreasing in size when a card is played and increasing when the player draws one or more cards.

2) *Solution Set*

The solution set shall refer to what card(s) will be played on certain round. This means that each round will have different solution set. The main idea is to find what card(s) to play on one round.

3) *Solution Function*

The solution function shall be omitted in this greedy strategy case. In other words, all possibility of solution set is accepted. This is based on the game mechanic where players are free to play whatever they desire is the right move.

4) *Selection Function*

As an implementation of greedy algorithm, we will prioritize cards which when played, the board statistics will increase at the maximum amount, this includes the power and health of units existing in the board.

The selection process will be processed by iterating each card in hand. After all cards have been evaluated on how many board statistics it will increase, the player will choose the card that increase the statistics in a maximum way. Therefore, we are able to overwhelm our enemy in board presence. If there exists a tie in calculation process, we shall pick cards that has the more power. If it is still a tie, we shall pick cards with the minimum cost. If the tie still stands, we shall pick the latest card checked.

In addition, when the board is full (6 of our units is still alive), we will still continue our calculation by also minding the decreasing statistics when the unit is replaced by a new unit. If it can increase the statistics, the card will still be played.

More accurately, we does not select one card to play, but in what order a card shall be checked by the feasibility function and then be added to the solution set if it is available to do so. That way, we allow multiple cards to be played in one round, assuming the current mana amount is sufficient.

5) *Feasibility Function*

The feasibility function shall check whether the card's cost is lower than the available mana. As the spell cards are able to use spell mana, the mana count differs between spell cards and other cards. As an example, on turn 5 with 3 spell mana, a player is able to play an 8-cost spell card but can only play a 5-cost unit card. Still, the selection function stands the same, picking the card which improves board statistics the most.

6) *Objective Function*

The objective function is defined as the maximum amount the board statistics can be improved by playing a set of cards.

B. Attacking

When it is our turn to attack, we'll be attacking with all our units available. This makes our gameplay style a bit towards the *aggro* type, where our focus is to quickly reducing our opponent's nexus as fast as possible by developing the board and threatening with big statistics.

C. Blocking

When blocking, we'll only block using units that will still survive the block. If it is possible, we would also try to kill the attacker while keeping our units alive. Although there is the possibility where *combat tricks* (units' statistics are increased when a battle is revolving), we would still continue using this tactic. This is done in order to simplify the algorithm made and also in the end, we would still need to extract that combat trick out of the opponent's hand.

But, we shall always block whenever the attack threatens our nexus to be destroyed. This will prevent us to lose when we still have units on board but block nothing.

IV. An Example Game Scenario Match

In this section, we will be discussing on an example game scenario match. We shall see and look at each round, explaining how we come to the decision move.

A. Round 1

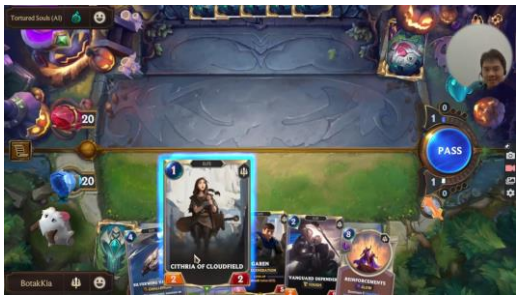

Figure 6. Round 1 of Example Scenario
(Source: Screenshot taken from the game)

In Round 1, our hands consists of 5 cards. As we iterate our hand, we find that the best card that can be played and improve our board presence is the 8-cost spell card *Reinforcements*. But when we check using the feasibility function, we find that our mana is not enough to play that. This also goes for the 5-cost *Garen*, 4-cost *Silverwing Vanguard*, and 2-cost *Vanguard Defender*. When we check the 1-cost *Cithria of Cloudfield*, we find that we are able to play this card, or in other word, adding it to the solution set. At last, we attack and end the round, having *Cithria of Cloudfield* on the board.

B. Round 2

Figure 7. Round 2 of Example Scenario
(Source: Screenshot taken from the game)

In Round 2, the opponent starts by developing his board, playing a 2-cost unit with 3 power and 2 health, with *Fearsome* keyword. In our turn, as we skip our same process in round 1, selecting cards that are not playable because of insufficient mana, we find that we are left with 2 playable cards: *Vanguard Defender* and *Battlesmith*. As these two cards would improve the board statistics by the same value (4), we play the last checked card, *Battlesmith*. The opponent attacks, we skip the block as we are not able to, then end the turn.

C. Round 3

Figure 8. Round 3 of Example Scenario
(Source: Screenshot taken from the game)

As we progress to Round 3, we start by playing the 2-cost *Vanguard Defender* as it is the only playable card from our hand. Later, we attack we all three of our units. Although our *Battlesmith* died being blocked by enemy's unit, we are able to push 2 damage to enemy's nexus. Then, the round ends.

D. Round 4

Figure 9. Round 4 of Example Scenario
(Source: Screenshot taken from the game)

In Round 4, the opponent starts by playing a 1-power-4-health, or as we call a 1-4-unit. As we iterate our hand, we find that we are able to play 2 cards: *Vanguard Silverwing* and *Vanguard Lookout*. As we prioritize the most improvement to our board statistics, we then play *Vanguard Silverwing*.

The enemy then attacks, dealing 2 damage to our nexus as we block one of their attacker. In the end, the enemy's board is filled with 2 units : a 1-4-unit and a 4-3-unit. The round then ends.

E. Round 5

Figure 10. Round 5 of Example Scenario
(Source: Screenshot taken from the game)

The match progresses to Round 5. If we look at our hand, we find that the best card that can be played to increase the board statistics is *Garen* (10). We then played *Garen* and the opponent responds with developing a 5-5-unit. We then attack using all of our units, challenging the 5-5-unit onto the *Vanguard Silverwing* to prevent it killing our *Garen.*

F. Round 6

Figure 11. Round 6 of Example Scenario
(Source: Screenshot taken from the game)

In Round 6, the opponent starts by playing *Kallista*. We then respond by playing *Vanguard Redeemer* as, again, it is the best and playable card available. The opponent keeps developing their board by playing a 1-1-unit. After that, notice that we are still be able to expand our board presence by playing *Vanguard Lookoout*, we then play him.

Figure 12. Round 6 of Example Scenario
(Source: Screenshot taken from the game)

Later on, the opponent attacks and we shall block as showed above. As mentioned before, we block by making sure that our unit still survives and also try to kill the enemies as many as possible. After that, the round ends.

G. Round 7

Figure 13. Round 7 of Example Scenario
(Source: Screenshot taken from the game)

As we progress to Round 7, we start by playing the *Reinforcements*, making our board really *wide*. The opponent responds by removing our *Vanguard Redeemer* using their spell and developing a 3-2-unit. We then played our 0-cost spell to summon even more *Dauntless Vanguard*, then attack fully with our units.

Figure 14. Round 7 of Example Scenario
(Source: Screenshot taken from the game)

The battle revolves as showed above. In the end, we are able to push for 9 damage to the enemy's nexus. After the attack, the round ends.

H. Round 8

Figure 15. Round 8 of Example Scenario
(Source: Screenshot taken from the game)

Progressing to Round 8, the enemy starts with playing *Thresh*, a 3-6-champion. We then responds by playing *Vanguard Bannerman*. This action is based on the most board statistics improving card.

Figure 16. Round 8 of Example Scenario
(Source: Screenshot taken from the game)

The enemy then attacks and we block as showed above. This block is needed because our nexus would be destroyed if we provide no blocking. Then, as we were able to level up *Garen* and granted the *rally*, we attack as showed below.

Figure 17. Round 8 of Example Scenario
(Source: Screenshot taken from the game)

The match ends and we win the game by destroying our opponent's nexus.

Figure 18. Victory Screen
(Source: Screenshot taken from the game)

Through this example, we have seen how the greedy algorithm be implemented when playing a card game *Legends of Runeterra*. For a better explanation, the author suggests to watch the video linked to this paper.

V. Conclusion

As a widely used algorithm, greedy comes in many kind of implementation. Through this paper, we have seen one of its many implementation, being a theoretical framework for a card game strategy. Though it may not be the best strategy available for the game itself, it still stood a chance when the author performed a demonstration.

To conclude things up, this kind of strategy is still far from being a decent strategy to use in real match against real players. As much as it needs reworks, this strategy can still be improved by combining it with another algorithm for deciding the best move available to pick. There are still many aspects to be looked at and repair. Therefore, the author would suggest anyone willing to contribute to this development of strategy polishing its base theory and widening its usage in the actual gameplay.

Lastly, from this paper, we are shown that greedy algorithm has many implementations in real life. Not restricted to only gameplay strategy, greedy algorithm may contribute to finding solutions for real world problems. In fact, we may be using it without even realizing.

Video Link at Youtube

The following link leads to a video showing the example game scenario live demonstration.

https://youtu.be/wNgTi7V6bNc

Acknowledgment

The author would like to thank Dr. Nur Ulfa Maulidevi, ST., M.Sc. as his lecturer for the Algorithm Strategy course, as well as other lecturers for the past guidance and knowledge. In addition, the author would also like to thank his family and friends for the supports given. This paper is completed by all of the support and guidance given. Lastly, the author would like to apologize if there is any misused words or lack of details in this paper.

References

[1] S. Adhikari, "Top 9 Popular Games of Ancient Greece", https://www.ancienthistorylists.com/greek-history/top-9-games-of-ancient-greece/, accessed 11 May 2021

[2] David, "Legends of Runeterra", https://sullygnome.com/game/Legends_of_Runeterra/365/detailedfollowergrowth, accessed 10 May 2021.

[3] R. Munir, "Algoritma Greedy", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf, accessed 10 May 2021.

[4] https://leagueoflegends.fandom.com/wiki/Keywords_(Legends_of_Runeterra), accessed 11 May 2021

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.

Medan, 11 Mei 2021

Billy Julius