Greedy Algorithm on Bots' Offensive and Item Mechanism in DOTA 2

Nadia Mareta Putri Leiden - 13520007
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520007@std.stei.itb.ac.id

***Abstract*—*Greedy* has been used on many fields before and one of the infamous fields is game development. Therefore, Greedy is possible to be implemented in games that require intelligent bots as the main idea is the bots will be divided into each level and each level of the bots will implement different strategies of Greedy. Finally, this concept is useful in differentiating the difficulty of each level in playing along with bots in DOTA 2. (*Abstract*)**

Keywords—game, greedy, algorithm, dota

I. INTRODUCTION

DOTA 2 is a MOBA (Multiplayer Online Battle Arena) game which focuses on how the player strategize through utilizing a wide range of heroes and in-game supporting items that can be crafted and the main goal would be how the player is able to destroy the enemies' team "fountain" which resembles a monument that each playing team has to protect. Each team will consist of five players, divided into two factions: Radiant and Dire. Moreover, there will be a series of towers which are required to be destroyed sequentially before the player could finally attack the fountain.

There will be Tier 4, Tier 3, Tier 2 and Tier 1 Towers on three lanes (usually being referred as Top, Middle, and Bottom lanes) that the user will need to destroy respectively and the higher the tier the stronger the tower will be.

Fig. 1. Tower Tier 4,3,2, and 1

The author would like to limit the scope of the game mode into the concept of All Pick mode, the standard DOTA 2 game mode where user can pick the desired hero (or character), build an item to increase each characters' power and level up their character to a maximum level of 30 on each game session. Moreover, the greedy will solely focus on the bot response to the player's choice of movement.

To add the realistic dimension to the game, the bots or enemies will have their own level of difficulties. The higher the level of the bot the more intelligent it will be. Thus, this is where *greedy* plays its role as one of the alternative solutions programmers can create a harder enemy—as the player decides to choose their desired level of difficulty. There are four level of difficulties based on the newest update which consist of Easy, Medium, Hard, and the highest difficulty is Unfair.

As DOTA 2 is a fairly complex game, the chosen greedy strategy has to be paired with an analysis of the current game condition. There should be multiple *greedy* on each condition based on the enemy distance, tower condition and many other variables which have to be taken into consideration. The use of *greedy* strategy is also limited and could not be implemented in all bots' movements; However, it will be implemented partially in some situations that require some sort of "automation" decision-making process. One example is as following:

Fig. 2. Usable Items

As the readers can see from the figure above, there are five usable items that the player can use. Through implementing Greedy, the bot will be able to make a decision on which item order to be used during the war of each faction. This will be further elaborated in the next section.

II. Theoretical Foundations of Greedy

A. Definition of Greedy

Greedy is one of the popular methods in optimization of problem-solving and it is closely tied to optimization problems. Moreover, optimization problems consist of two elements: Maximization and Minimization. *Greedy* focuses on what is happening at the moment without considering what will happen in the future. Therefore, its idea is to choose the "best" solution that is available at that time (local optimum solution) and through this local optimum solution it is expected to find the global optimum solution. However, the global optimum solution is not always guaranteed to find the best solution available and that is one of the main disadvantages of using Greedy compared to the traditional approach such as: Brute Force Algorithm and Exhaustive Search. In order to prove that the algorithm based on the concept of *greedy*, then we will need to prove it through a complex mathematical equation.

Using Brute Force Algorithm or Exhaustive Search is much more accurate in terms of finding the optimum solution; However, the process is much less efficient compared to *Greedy*, as it will have to check each one of the possible solutions and will be time-consuming and inefficient for the memory storage if we are working with a large set of data. Brute Force Algorithm and Exhaustive Search work preferably with a small set of data, otherwise we can use other alternative algorithms such as *greedy*.

B. Elements of Greedy

There are six elements which are crucial in our process of finding the right solution through *greedy* such as following:

1. Candidate set: the set of elements that will be selected then included on the solutions.
2. Solution function: to make sure whether a solution is reached or not reached yet.
3. Objective function: to determine whether to maximize or minimize the solution
4. Solution set: the set of elements that will be chosen as a solution.
5. Feasibility function: to determine whether the element can be included as a solution based on the initial condition
6. Selection function: to filter the elements from candidate set and to choose the best candidate that will be included in the optimum solution.

It is required to fulfill all of the elements above. Therefore, to demonstrate these elements we will be using one case example of *Greedy*.

C. Case Example of Using Greedy

Below is the example of using *Greedy*. However, it is also possible to be solved using Exhaustive Search and Brute-Force Algorithm by generating all of the solutions. It is a good idea to use Exhaustive Search or Brute Force Algorithm as the data is relatively small.

Suppose we own coins of 25, 20, 10, 5 and we are expected to pay the groceries which cost 50 coins. Hence, another question arose, how can we minimize the number of coins that needs to be paid with a total of 50 coins?

First, we must identify the elements first. From this example we can construct the elements such as following:

1. Candidate set: set of coins which will be taken from 25, 20, 10, or 5
2. Solution function: determine whether the solution has reached the main objective with a total of 60 coins
3. Objective function: minimize the total coins used.
4. Solution set: set of coins taken from candidate set.
5. Feasibility function: check whether the current coins on solution set exceeded our objective or not.
6. Selection function: take the highest nominal of coin.

After we have identified the elements needed to run through our *greedy*, then we can start analyzing the process. There are two main things that we have to pay attention in this case: the *Objective function* and *Selection function*. Our *Objective function* is that we want to minimize the total coins used, which is understandable that we have to keep in mind (we will check it later after we have finished iterating). Then, our *Selection function* is to take the highest nominal of coin which means: every time we progress, we will have to take the highest coin nominal, 25 in this case and as long as it still passes the *Feasibility function*, then it is safe to take 25. However, if we sum it all together and we need a smaller element, then we will take 20 (10, 5 and so on until our objective is reached).

Now we will start to demonstrate how it works:

1. Solution set (S)= {}
2. Choose the highest nominal with our *Selection function*, so the solution set will be S = {25}, check with *Feasibility function*. The sum of S ≤ 60, then it is safe to proceed.
3. Choose the highest nominal with our *Selection function*, so the solution set will be S = {25, 25}, check with *Feasibility function*. The sum of S ≤ 60, then it is safe to proceed.
4. Choose the highest nominal with our *Selection function*, so the solution set will be S = {25,25,25}, check with *Feasibility function*. The sum of S > 60, then it is not safe to proceed.
5. Take the second highest nominal and replace our last element with our *Selection function*, S = {25, 25, 20}. Check with *Feasibility function*. The sum of S > 60, then it is not safe to proceed.
6. Take the next nominal and replace our last element with our *Selection function*, S = {25, 25,

10}. Check with *Feasibility function*. The sum of S ≤ 60, then it is safe to proceed.

7. We will check it with our *Solution function*: S is 60, so it fulfills our objective.

Thus, from the process above we will be using three coins that consist of our Solution set (S) = {25, 25, 10}. Fortunately, in this case, our *greedy* generates the most efficient global optimum solution. However, with the same problem, we may utilize a different *Selection function* and it might not fulfill our *Objective function*.

III. Problem Decomposition

There are many aspects in DOTA 2 that can be implemented with *greedy*. However, there will be a limitation on what we will be focusing on in terms of our *greedy* strategy. We will take the concept that the bots have already designed to defend towers and to push lanes—all designed respectively to their difficulty level, but they do not know which strategy to be used when there are enemies nearby and what item order to be crafted as the game progresses. For a more detailed pre-conditions, the author will elaborate on the next paragraph.

We will take a condition as below. These are the conditions to be addressed on the author's *greedy*:

1. The bots do not have a specific calculated strategy on their priority target during a war (which enemy to be attacked).
2. The bots do not have a specific strategy on how to use skill order or attack to be used during a war.
3. The bots do not have a specific strategy on which item order to be activated during a war.

Hence, from the problem decomposition above, the author will create three *greedy*. Most of the *greedy* strategy focuses on the offensive side mainly how the bots will respond during a "war" both on using items and skills.

Fig. 3. A team preparing for a war, all of the team members are present in one spot (5 characters in spot)

However, to create a clearer perspective the author would also like to list all of the pre-conditions on how the bot will operate outside of the three points above (or what the bots have already designed to do without the *greedy*):

1. The bots will already have an in-built lane formation intelligence respectively based on their difficulty level. The higher the difficulty level, the better the formation will be. (Lane formation is player's positioning on the middle lane, top lane, and bottom lane).
2. The bots will already have an in-built tower defense mechanism respectively based on their difficulty level. The higher the difficulty level, the more responsive the bots will be.
3. The bots will have the same map sight as the player (will be enhanced if they put wards).
4. The bots will already have in-built self-defense mechanism (when to retreat or to keep attacking the enemy) respectively based on their difficulty level. The higher the difficulty level, then the bot will be much more intelligent to avoid dangerous situations.
5. The bots will already have in-built tower pushing intelligence respectively on their difficulty level.
6. The bots will already have an idea of which skill to be leveled up first.
7. The bots will already have a mechanism on how to reorganize neutral stash. Hence, the stash *greedy* mechanism will only focus to a non-neutral item.
8. The bots will already have an in-built intelligence on item build and which item in stash to be sold.
9. The bots will already have an in-built basic level of attack and item mechanism based on their difficulty level.

Based on the three *greedy* that will need to be created most of the points require some sort of "priority list" where the author will have to determine the priority level of each item and skill.

IV. Problem Analysis

Before the author would like to address the conditions, the author will create a Priority Table (this has been mentioned in the previous chapter referred as a "priority list") the Priority Table will be used as a reference on whether to minimize or maximize the optimization later. However, not all cases will be using the Priority Table. Each problem will require different kind of priority calculation which will be divided into following categories:

1. *Greedy* by Target Priority

   The priority level will be determined based on the radius and the health amount of the player. It will be calculated through this formula:

$$Target\ Priority = \frac{Health}{Radius} \quad (1)$$

Thus, in order to make a detailed *greedy* strategy, the author would like to describe the *greedy* properties as following:

a) Candidate set: The enemies within a maximum of the hero's current vision range with a maximum of 1000 radius from the bots' vision range perspective accumulated from nearby teammate vision range.

b) Solution function: check whether there is still any enemy at sight that has not yet being added to solution set.

c) Objective function: maximize the number of effective targets and damage dealt to the enemy team until no enemy is at sight.

d) Solution set: the set of enemy team's heroes sorted from the highest priority.

e) Feasibility function: Adding new target will not exceed maximum target which can be hit from depending from the skill or basic attack type.

f) Selection function: choose the highest enemy priority within the radius.

Fig. 4. Accumulated Vision from Friendly Objects

2. *Greedy* by Skill Priority

Skill Priority will be used to determine which skill order the bots will use during war. This aspect will be highly dependent on Priority Table and will only apply on Active Skills. Furthermore, the priority will be calculated through this formula:

$$Skill\ Priority = \frac{PriorityPoint + Duration}{Mana} \quad (2)$$

Priority Point will be obtained from Priority Table, Mana Point will be taken from the skill description on how much mana will be consumed, and the Duration is taken from how long the skill will affect its caster and target. Hence, if it is not a skill that is categorized as a buff or de-buff that will last some time (including channeling skills), then the Duration will be set into 0.

TABLE I. EFFECT DESCRIPTION PRIORITY TABLE

No	Effect Description		
	Classification	***Effect Name***	***Priority Point per Effect***
1	Disables and Silence	Cyclone, Disabling Orders, Fear, Hide, Hex, Hypnosis, Root, Leash, Stun, Sleep Taunt, Forced Movement / Knockback, Silence	+4
2	Movement	Move Speed Slow, Flying Movement, Phased / Tree Walk / Unobstructed Movement	+3
3	Buffs/Debuffs	Attack Speed Slow, Disarm, Ethereal, Mute, Trap, Blind, Break, Attack Immunity	+2
4	Other	Other properties which are not yet mentioned	+1

[a.] Data taken from DOTA 2 Database and Fandom Wikipedia

On each effect attributed on the skill, the point will be accumulated. For example, if a skill has a stun and slowing effect, then the total Skill Priority point will be 3 + 2 with a total of 5 Skill Priority points.

Below are the *greedy* properties of how will the bots use the skill during the war:

a) Candidate set: All of the active skills converted to Skill Priority point.

b) Solution function: check whether there is still enough mana to perform skills or if all skills are on cooldown.

c) Objective function: maximize the number skills used along with the damage dealt to enemy team.

d) Solution set: the set of skill orders based on the Skill Priority point.

e) Feasibility function: Adding new skill will not exceed the bots' mana bar and are not under cooldown.

f) Selection function: choose the highest Skill Priority point.

To put a disclaimer, this skill priority *greedy* strategy will only be implemented after *Greedy* by Target Priority has been activated prior to casting abilities (or skills). The author would like to give an example on how will this *greedy* strategy work in a real in-game environment.

The author will choose Shadow Fiend as the example because this hero has an almost identical skill with the same effect and the same mana consumption that might need an exception on this *greedy* strategy.

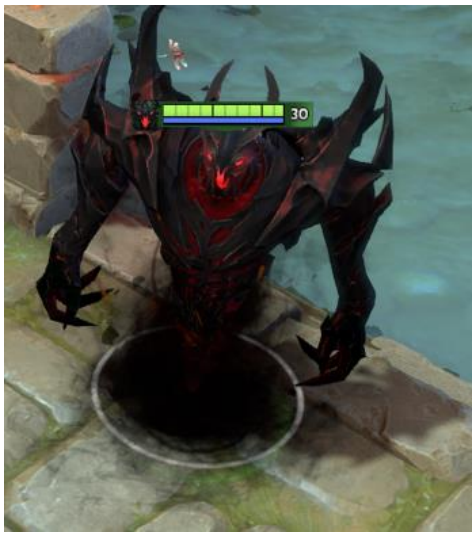
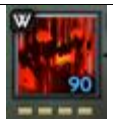
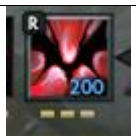
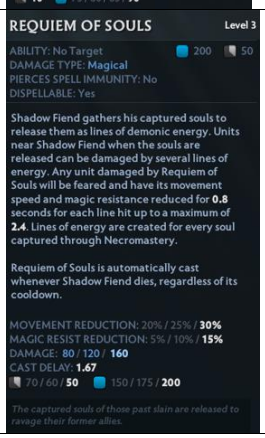


Fig. 5. Shadow Fiend in DOTA 2

TABLE II. CASE STUDY SHADOW FIEND

No	Skill Name		
	Skill	***Description***	***Skill Priority***
1			1/90 = 0.01
2			1/90 = 0.01
3			1/90 = 0.01
4			5 + (2.4)/200 = 0.037

Shadow Fiend has a total skill of 6 skills. However, we will only put 4 skills onto our calculation as the other 2 skills are passive skills. As we can see from the Table II, Shadow Fiend has three identical skills which is "Shadowraze", only differentiated by its range. Then how will this work in this *greedy* strategy?

The author has mentioned before that we will be using *Greedy* by Target Priority prior to this specific *greedy* strategy, so the bot will be using the skill which is nearer to our target priority based on its skill range—that is the case with Shadow Fiend. Each skill will have a cooldown and with this *greedy* strategy after the bot has casted one skill (after skill delay or animation has finished) it will directly shift to search for the next skill to be casted.

Fig. 6. Shadow Fiend Base Level 30 Stats

Now we will start to demonstrate how it works:

1. Solution set (S)= {}, Candidate set = {0.01, 0.01, 0.01, 0.037}
2. Choose the highest priority with our *Selection function*, so the solution set will be S = {0.037}, check with *Feasibility function*. The sum of mana ≤ 1215 because it is the first skill to be casted then it is not under cooldown; it is safe to proceed.
3. Choose the highest priority with our *Selection function*, so the solution set will be S = {0.037, 0.037}, check with *Feasibility function*. The sum of mana ≤ 1215, with a cast delay of 1.76s which means that it is still under cooldown (the cooldown varies with level, but in this case is 200 seconds) then it is not safe to proceed.
4. Choose the highest priority with our *Selection function*, so the solution set will be S = {0.037, 0.01}, check with *Feasibility function*. The sum of mana ≤ 1215, not under cooldown. Then it is safe to proceed.
5. Continue until all of the skills are on cooldown or the character mana is not enough to perform any active abilities.

3. *Greedy* by Item Usage Priority

After buying certain item, then each active (or usable) items will have certain priority levels to be used during the war. The formula is pretty similar in terms of concept with the other two *greedy* strategies.

Thus, this will be the formula the author will use to calculate the Priority Points:

$$Item\ Priority = \frac{PriorityPoint + Duration}{Mana} \quad (3)$$

$$Item\ Priority = Priority\ Point + Duration \quad (4)$$

Formula number (3) is used on basic cases. However, formula number (4) will be used on more than one infinity cases which will be elaborated further below.

TABLE III. ITEM DESCRIPTION PRIORITY TABLE

No	Effect Description		
	Classification / Specific Item	***Effect Name***	***Priority Point per Effect***
1	Black King Bar	Spell Immunity	+100
2	Buffs	Lifesteal, Attack Immune, Attack Speed Slow, Disarm, Ethereal, Silence, Mute, Trap, Blind, Break, Spell Immunity	+4
3	Disables	Cyclone, Disabling Orders, Fear, Hide, Hex, Hypnosis, Root, Leash, Stun, Sleep Taunt, Forced Movement / Knockback	+3
4	Movement and Illusion	Move Speed Slow, Illusion	+2
5	Others	Flying Movement, Phased / Tree Walk / Unobstructed Movement and other unmentioned effects	+1
6	Blink Dagger, Travelling Boots	Will not be included	+0

[b.] Data taken from DOTA 2 Database and Fandom Wikipedia

Black King Bar will be given Priority Point per Effect of 100, which means that Black King Bar will have the highest priority in most cases. The idea is that the bot will first trigger Black King Bar (unless it is under cooldown) then will proceed to the next item afterwards.

Below are the *greedy* properties of how will the bots use the items during the war:

a) Candidate set: All of the active items converted to Item Priority points.

b) Solution function: check whether there is still enough mana to use the item or if all items are on cooldown.

c) Objective function: maximize the number items used along with the damage dealt to enemy team.

d) Solution set: the set of items orders based on the Item Priority point.

e) Feasibility function: Adding new item will not exceed the bots' mana bar and are not under cooldown.

f) Selection function: choose the highest Item Priority point.

Fig. 7. Item Used by Shadow Fiend

TABLE IV. CASE STUDY SHADOW FIEND'S ITEM

No	Item Name		
	Item	*Description*	*Item Priority*
1	Abyssal Blade		3+2/75 = 0.067
2	Black King Bar		100+9/0 = Infinity. Take 109.
3	Satanic		4+6/0 = Infinity. Take 10.
4	Manta Style		20+2/125 = 0.176

As we can see that the Item Priority point of *Satanic* and *Black King Bar* is on the same level which is infinity. In this case how can we determine which item to be prioritized? In the case of more than one item with infinity Item Priority points, we will see from the total of Priority Point + Duration and compare it.

Now that we have identified all of the perquisites, we will start to demonstrate how it works:

1. Solution set (S)= {}, Candidate set = {0.067, 109, 10, 0.176}
2. Choose the highest priority with our *Selection function*, so the solution set will be S = {109}, check with *Feasibility function*. The sum of mana ≤ 1215 because it is the first item to be casted then it is not under cooldown; it is safe to proceed.
3. Choose the highest priority with our *Selection function*, so the solution set will be S = {109, 109}, check with *Feasibility function*. The sum of mana ≤ 12, still under cooldown (in this case is 75 seconds) then it is not safe to proceed.
4. Choose the highest priority with our *Selection function*, so the solution set will be S = {109, 10}, check with *Feasibility function*. The sum of mana ≤ 1215, not under cooldown. Then it is safe to proceed.
5. Continue until all of the skills are on cooldown or the character mana is not enough to perform any active abilities.

It works under the same condition as *Greedy* by Skill Priority, where after the bot have finished triggering the active items and reap its effects, it will automatically shift to the next items in Candidate set. Hence, if the item is still under cooldown it will shift to the next candidate with lower priority.

Then we have finished describing all of the possible *greedy* strategies, then we will have the final verdict:

1. On Easy mode, there will be no *greedy* standard applied. Which means that the bots attack and item mechanism will be implemented at basic level.
2. On Medium mode, we will implement *Greedy* by Target Priority. In this level, the bot will understand which target to attack first, but not as advanced as using items with a specific order.
3. On Hard mode, we will implement *Greedy* by Target Priority and *Greedy* by Item Priority. Here we will start adding patterns on how the bots will use the item in a specific order.
4. On Unfair mode, we will implement all of the *greedy* strategies above. One of the most important offensive mechanisms is *Greedy* by Skill Priority that will increase how the bot will use skill order during war.

V. CONCLUSION

There are many other algorithms to be implemented in determining what kind of strategy the programmers intend to create as it is only one of many other alternatives ideas that

might be implemented in DOTA 2. However, the use of *greedy* strategies is relatively easy to be implemented and to maintain.

VI. Video Link at Youtube

Link youtube: https://youtu.be/taWk4_aT7uU

VII. Acknowledgement

The paper is finished because of God's grace and His blessing despite under pressure of examinations these past weeks. Furthermore, the author is honored to be given such opportunity by Ibu Masayu Leylia Khodra to understand Algorithm Strategies that further can be implemented in many real-life working environments. Furthermore, DOTA 2 communities has been very helpful in providing in-game data that can help the author to analyze the situation and to further gather the information in pursuit of finding the best *greedy* strategies. However, the paper is far from the idea of perfect. Thus, any criticism will be much appreciated.

References

[1] Munir, Rinaldi. 2021. Algoritma Greedy (Bagian I). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

[2] Munir, Rinaldi. 2021. Algoritma Greedy (Bagian II). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

[3] Munir, Rinaldi. 2021. Algoritma Greedy (Bagian III). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian3.pdf

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, 23 Mei 2022

Nadia Mareta Putri Leiden
13520007