Implementation of Greedy Algorithm for Designing Simple AI of Turn-Based Tactical Game with Tile System

Adin Baskoro Pratomo 13513058
Program Sarjana Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
adinbaskoro@s.itb.ac.id

Abstract—Greedy approach is an algorithm which characteristic is making locally optimum, hoping it will lead to globally optimum solution. Artificial Intelligence (AI) for tactical turn-based game with tile system can be made using greedy approach. By using greedy approach, the AI will move according to the most optimum move at that time, which will lead to best move at that time.

***Keywords*—Greedy, Artificial Intelligence, Tactical Turn-Based, Game, Tile**

I. Introduction

Tactical game is one of the most favorite video game genre today. It usually involves great background story, beautiful art, awesome background music, and good replay value which motivates people to play it over and over again.

Most of tactical game implements turn-based movement system for every action. This system is chosen because it's easier for player to understand, and actually increase the strategy needed to win the game. Many people think the best part of tactical game is thinking about strategy needed to win in shortest turn possible.

Movement system in tactical game varies from game to game. One of that system is tile system. In tile system, character or object moves one tile at one time. This makes tile system unique, because the objects can only move at four directions; up, left, down, and right, unlike another system which allows the player to move the character more freely.

Artificial Intelligence in game has been used for years. Tactical turn-based game also use AI to move the character, so human player can play even when he only plays alone. The AI in tactical turn-based RPG will manage what the character will do, how far they should move, what action they should take, etc. They can even assist player on beating their enemies.

Greedy Algorithm is an algorithm that is usually used for optimization problem. It doesn't always yield most optimum result. however it usually produce good result. Another plus of greedy algorithm, it approximates the optimum result in a reasonable time.

The AI in turn-based tactical game can also be categorized as optimization problem, since it needs to do things in efficient way, for it to win.

II. Theory

A. Greedy Algorithm

Greedy algorithm, as its name suggests, it makes a "greedy" choice on each step. It grabs best local option, hoping it will yield globally optimum solution. There are On Each step, the choice made must be:

1. Feasible
   The choice made must satisfy the problem needs and constraint.
2. Locally optimal
   It must be the best choice among other choice at that step.
3. Irrevocable
   Once chosen, it can't be changed on subsequent steps of the algorithm.

In general, greedy algorithm has five components:

1. Candidate Set
2. Selection Function
3. Feasibility Function
4. Objective Function
5. Solution Function

Candidate set is a set from which a solution is created. Selection function is a function which choose best candidate to be added to solution. Feasibility function determines whether the candidate can be added to solution. Objective function assigns value to solution, and solution function to indicate whether we have discovered the solution.

Greedy algorithm is usually used in optimization problem, where most optimum solution is needed. However, it should be noted that greedy algorithm doesn't always yield optimum result. That's because greedy algorithm is 'short sighted' and 'non recoverable'. Greedy always make local best choice, and can't go back after a choice in a step has been chosen.

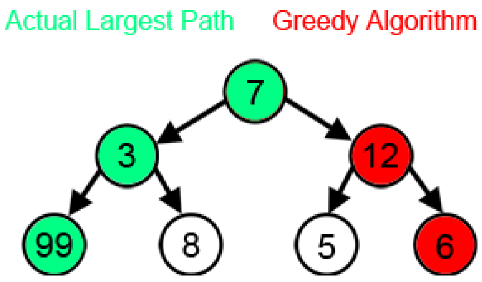


Figure 1. Greedy making wrong choice (source: Wikipedia)

One popular problem in greedy is "making changes" problem. The problem is making a change of a given amount using the least number of coins possible. The informal algorithm is to start with nothing, then at every stage without passing the given amount, add the largest to the coins already chosen. The formal algorithm is:

```
MAKE-CHANGE (n)
        C ← {100, 25, 10, 5, 1}
// constant.
        Sol ←
{};
// set that will hold the solution set.
        Sum ← 0 sum of item in
solution set
        WHILE sum not = n
            x = largest item in set C
such that sum + x ≤ n
            IF no such item THEN
                RETURN    "No
Solution"
            S ← S {value of x}
            sum ← sum + x
        RETURN S
```

For example, making a change for 2.89 have solution of 2 dollars, 3 quarters, 1 dime, and 4 pennies. From the solution itself we can se the algorithm is greedy, because at every stage it picks largest coin without reconsidering it. It's also never changed its mind, in the sense that once a coins in a solution set, it will remains there.

As mentioned above, greedy doesn't always yield optimum solution. It may even produce unique worst case solution. In "Making Changes" problem, where you must match certain value, with the least number of coins of given denominations. Given the coin denominations are 25-cent, 10-cent, and 4-cent coins. Greedy algorithm wouldn't be able to make 41-cent from those coins. Greedy will first select 25-cent, 10-cent, and then 4 cent yielding result of 39-cent, thus making it impossible to make 41, since the smallest value it can made is 39 + 4 or 43-cent.

Even so, for some NP-complete, greedy is still frequently used, because it's easy to think and to implement. It also give a good approximation to the optimum solution, rather than bruteforce method where the time needed to solve a problem can be very unreasonable.

For the problem that has been proven yielding optimum solution when solved using greedy algorithm, greedy becomes method of choice, because it faster from many other method, especially dynamic programming. Greedy is mainly used in minimum spanning tree algorithm, be it Prim algorithm or Kruskal algorithm. It's also used to find optimum Huffman tree.

B. Turn-based Tactical Game

Turn-based tactical game is one type of video game genre, in which the players are each given turn to play, and limited move count. Usually those games are also a role playing game, where the players play a certain character and follow a certain story. As opposed with their non-tactical game counterparts, tactical game usually lack of exploration, and level grinding. Level grinding is an action where player battles same enemy over, and over again to gain as much level as they can. In tactical game, usually there's no level grinding, as the emphasis there is battle strategy.

Battle in turn-based games has specific winning conditions, defeating all enemies and defeating the leader are few example of those. Players can also equip their character with weapon and armor, train them and change their class depending on the game.

Tactical turn-based game originally came from Japan. It's mainly known as "Simulation RPG" (シミュレーション RPG). The earliest Japanese role playing game, The Dragon and Princess by KOEI used a combat system where, following a random encounter, the game transitions to a separate, graphical, overhead battle screen, where a tactical turn-based combat system is used.

Figure 2. Dragon and Princess by KOEI (source: giantbomb.com)

Few years later, many similar tactical games arrived. Such as Fire Emblem series and Bokosuka Wars.

Bokosuka Wars was responsible for laying the foundation of tactical games as we know now. It revolves around the king, who must recruit soldier to defeat his enemy.

However, the genre didn't become very well known until Nintendo released and published the war game RPG, Fire Emblem: Ankoku Ryū to Hikari no Tsurugi, developed by Intelligent Systems for NES. Fire Emblem was an archetype of the genre, from which the gameplay is still used until today, even though some elements are influenced by earlier games. By combining many aspects of earlier tactic games and role-playing games, Nintendo created a hit, which spawned sequels and imitators. It introduces feature, which the character aren't just a mere pawn. Instead, they are unique, and if they lose their hitpoint in a battle, they will remain dead forever. In newer games multiple endings are possible, depending on which characters are alive or dead.

Fire Emblem becomes well known in western only after the release of the release of Fire Emblem: Rekka no Ken for the Game Boy Advance, many years later, that the series was introduced to Western gamers, who until then are more familiar with games that is influenced by Fire Emblem itself, such as Disgaea and Ogre series.

Advance Wars series is also a famous tactical game. It has feature in which the player can build their army from start, rather than using given unit from start. This has spawned many tactical game fans, because it's considered more tactical as there is also resource management strategy needed to win the game.

Movement system in turn-based tactical games is usually tile based, where a character can only move within adjacent tiles. Mostly, the tile shape used are square, as it's simpler and easier to understand. However, more recent games employ another shape, like hexagon or octagon to increase complexity.

The tile are also not limited to 2D. Most games have 3D field, in which the tiles each has their own height. This also add more complexity.

Newer games also use full 3D or 3D isometric map view, rather than 2D top view map. It's considered better to look at, as it shows the terrain height differences and many other things.

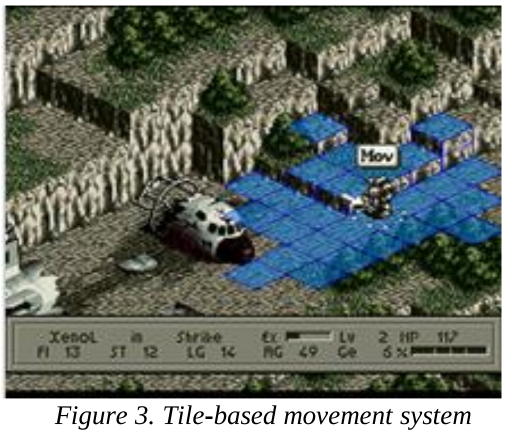


Figure 3. Tile-based movement system (source: Wikipedia)

In square tile-based movement system, player can only move to adjacent tile, according to move point they have that turn. For example, if their move point is 4, they can move 4 tiles away from their original position. They can move in 4 direction; front, left, back, and right. After they used all their move point, the turn ends and switched to the enemy's turn.

In a turn, player usually can move around, searching best place to attack. They can also attack the enemy if possible, or casting spells on them. Some games also have special action such as persuading the enemy to join ally's force.

The game ended after a certain condition has been completed. After that, player can be rewarded in experience points, or given items.

III. Implementation

A. Greedy and Tactical Turn-based system AI

Tactical Turn-based system AI can be approached using greedy algorithm. That's because in tactical turn-based game, the unit must make choice that is effective in order to win the battle. For example, a unit needs to move towards enemy with shortest path possible, so it can attack the enemy. Another example is the choice of moving and attacking. The AI will choose whether to attack or to move, depend on the distance of the AI with the player. If the AI is adjacent with the player, AI will attack the player, and then end his turn. If not, AI will try to chase the player, hoping it can attack the player.

B. Tile-based Movement System

Tile-based movement system can be approached by greedy because to reach a certain place, every movements counts. To reach certain place with shortest path, we need to consider best path for each movement. For example, if the AI is at (0,0) and player is at (0,5), AI will go to right, rather than down, assuming increased y-axis means movement to right. If in one turn AI can move for 3 tiles, it will first move right to (0,1), right again to (0,2) and then right again to (0,3), as it's the most efficient path to reach the player.

The informal algorithm is, at every stage, count the distance to target. And then go to the tile where the distance is lower than the original distance.

Here, the elements of greedy algorithm are:

1. Candidate Set : move (up, down, right, left)
2. Selection Function : nearest distance to target
3. Feasibility Function : number of movements. AI can't move anymore if the number of movement exceed the limit.
4. Objective Function : the distance to target
5. Solution Set : All the move done

The formal algorithm is:

```
 while ((moveleft > 0)){
     if (distancetoplayer(left) <
distance) {
     gotoleft
   else if (distancetoplayer(right) <
distance) {
     gotoright

 if (distancetoplayer(top) <
distance) {
     gototop
```

```
 else if (distancetoplayer(down) <
distance) {
     gotodown
 moveleft--
 }
```

the AI will consider which path it must take by counting the distance of adjacent tile. Also it can only move for a certain number, so if the move number left is zero, it can't move anymore until the end of player's turn, thus ending the AI's turn

C. Action choices

Action choices can also be approached using greedy algorithm. That's because at every step, the AI must choose whether to move or attack, in order to victory. The AI designed here will attack if its distance with player is equal to one, since it can only do melee attack . if the distance is greater than one, it will move using tile move algorithm in order to reach the player in shortest way possible.

The informal algorithm is, if the distance to player is greater than one, move, else, attack.

Here, the elements of greedy algorithm are:

1. Candidate Set : move, attack
2. Selection Function : distance to target
3. Feasibility Function : number of movements. AI can't move or attack anymore if the number of movement exceed the limit.
4. Objective Function : the distance to target
5. Solution Set : All the move done

The formal algorithm is:

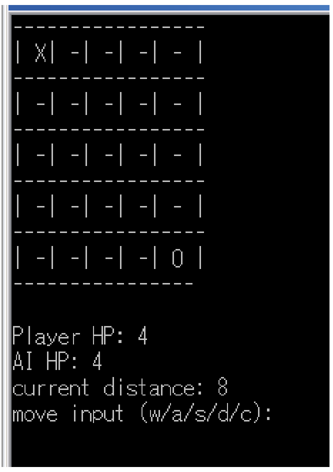
```
if (distance == 1){
     attack
}
Else {
while moveleft > 0 and distance > 1{
     move
     moveleft--

     if (distance == 1){
     attack
     moveleft = 0;
 }
```

The AI will move until player is in adjacent tile with AI. If in the middle of the movement, the AI meet player, and there are move numbers left, the AI can directly attack, then ending its turn.

D. Implementation in Program

I made a simple game program to implement the AI and to test it. The program is written in C++. The program has only 2 players, human and AI, and 5 x 5 tile. AI unit is the 'X' symbol, and human unit is the'O' symbol.   Each player are given 4 hit points, and can only move 3 tile each turn. If any player attack, they deal 1 damage, and their turn end immediately after attacking. The game ended if any of those two's hit point become zero.



```
---------------
| X| -| -| -| - |
---------------
| -| -| -| -| - |
---------------
| -| -| -| -| - |
---------------
| -| -| -| -| - |
---------------
| -| -| -| -| O |
---------------


Player HP: 4
AI HP: 4
current distance: 8
move input (w/a/s/d/c):
```

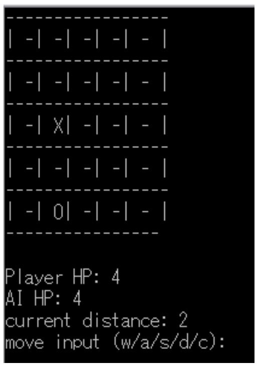
Figure 4. Game Interface

From the screenshot of the interface, you can see that initially the players are separated. They are given 4 hit points, and there are indicator of how far they are separated apart. Here, the human player can input the move they want, 'w' for moving to top, 'a' for moving to left, 's' for moving to bottom, 'd' for moving to right, and 'c' to attack, After human player finishes his turn, the AI will immediately move

It's now human player's turn. They can do anything they want to, even attack. However, the attack won't reach, thus rendering it useless, but it will still immediately ends the human player's turn.

The player can move 3 times, one tile each time. During that time, AI won't move, as it's not its turn yet. After completing those 3 moves, the AI's turn will start, then AI will immediately move to optimum position for attacking.

By following the movement algorithm, AI can calculate position between two units, and then decide which path to take in order to reach human player as fast as possible.



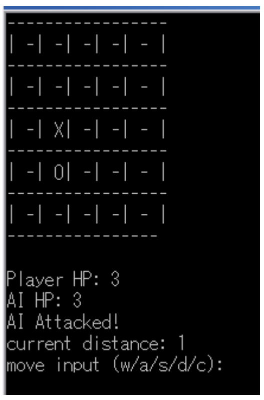
```
----------------
| -| -| -| -| - |
----------------
| -| -| -| -| - |
----------------
| -| X| -| -| - |
----------------
| -| -| -| -| - |
----------------
| -| O| -| -| - |
----------------

Player HP: 4
AI HP: 4
current distance: 2
move input (w/a/s/d/c):
```

Figure 5. AI responds to human player movement

From the figure above, it can be seen that the AI will move to the most optimum distance it can, making the distance difference become 2. However, it can’t attack yet, so it doesn’t attack.

If the distance between them are equal to one, both of them can attack. However, they can only attack on their own turn, and after attacking, their turn ends. If any of their hit points become zero, game is over, and player left is the winner.

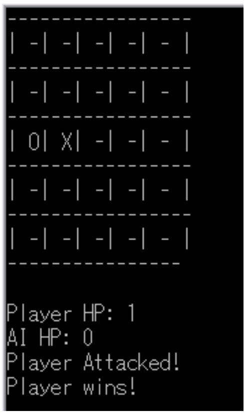


```
----------------
| -| -| -| -| - |
----------------
| -| -| -| -| - |
----------------
| -| X| -| -| - |
----------------
| -| O| -| -| - |
----------------
| -| -| -| -| - |
----------------

Player HP: 3
AI HP: 3
AI Attacked!
current distance: 1
move input (w/a/s/d/c):
```

Figure 6. AI can attack if the distance = 1

As mentioned above, the game is over if one of the player’s hit point become zero. The game will keep going on until the winning condition are fulfilled. The AI will keep attacking or moving until the game is over.



```
----------------
| -| -| -| -| - |
----------------
| -| -| -| -| - |
----------------
| O| X| -| -| - |
----------------
| -| -| -| -| - |
----------------
| -| -| -| -| - |
----------------

Player HP: 1
AI HP: 0
Player Attacked!
Player wins!
```

Figure 6. Human player wins the game

IV. Analysis

By running the program, it can be seen that the AI is following the player, by moving to most efficient choices at that time. Once it comes near player, the AI can attack player, thus dealing damage to the player.

The AI here is very simple, as it only consider its distance to the player, and doesn’t consider another factor, such as player’s HP or his own HP. Even so, this gives a good approximation on how the AI of tactical turn based game works. By adding more consideration factor, the AI will be better, and wiser on determining what to do next.

The consideration factor can be added if we expand the action set. The AI designed her can only move, and attack. Moreover, it can only attack another unit within 1 tile of range. By expanding the action set; adding ranged attack, for example, we can make more sophisticated AI which is smarter.

By adding terrain type, the consideration factor can also be added furthermore. For example, the character may not move into the water, as it lowers its attack and movements. Or by adding height to terrain, the character will avoid higher terrain as it slows down the character.

V. Conclusion

The AI for tactical turn-based game with tile system can be approached with greedy algorithm. The greedy algorithm will check best move at that time, hoping it will lead to victory of the battle. The AI use greedy for deciding on where to move, and deciding whether to move or attack.

However the AI designed here is way too simple to be implemented in a game. To improve it, we can add more move set, such as standby, using spell, long range attack, etc. We can also add “fitness function”. Fitness function is to decide the value of certain action on a certain terrain, or a certain situation. For example, a character may heal his ally that has less than 5% HP remaining, rather than attacking enemy in front of him. It also can be opposite, attacking enemy instead of healing his ally. Fitness function is counted for every action, and then the action with highest fitness function value will be performed.

VI. Acknowledgment

I would like to thank my mentor, Ir Rinaldi Munir M.T., and Dr. Nur Ulfa Maulidevi, S.T., M.Sc as it was due their guidance that this paper can be written, Square Enix for their game “Final Fantasy Tactis Advance”, Atlus for “Ogre Tactics”, and Nintendo for “Advance Wars” which have greatly inspired me to write this paper.

REFERENCES

[1] Levitin,Anany, "Introduction to The Design and Analysis of Algorithm," 3rd ed, Pearson.

[2] "Greedy Introduction" [Online] Available: http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Greedy/greedyIntro.htm [Accessed: 4-May-2015]

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, 4 Mei 2015

[illegible]

Adin Baskoro Pratomo
13513058