CentraleSupélec

MSc AI
Foundation of AI

AI challenge
Vampires VS Werewolves

Jean-Philippe Poli

Introduction

The goal of this project is to develop an Artificial Intelligence to play the Vampire VS Werewolves game. The set of rules is defined in the remainder of this document.

The algorithms described in the different modules can inspire you, but you can also find cutting-edge algorithms. At least, you have to take into account the opponent's moves and not to be a reflex agent.

Evaluation

Your project will be evaluated regarding different criteria: quality of AI, strategy. In addition, your presentation the day of the tournament, your report and the results of tournament will be taken into account.

Source code

You can write your project in any programming language, regarding that your AI will be executed on CTI machines. I will not evaluate the quality of your source code.

It is forbidden to use libraries that implement AI, trees, graphs, etc.

Your report will describe your algorithms and will refer to a file, a function that implements it.

Tournament

The day of the tournament, you will present to the other students your AI in five minutes. The goal is to make them stress and believe their own AI will fail.

This very day, AI will face the others: on each map, you will play both species consecutively. The first three AI will have respectively, 3, 2, 1 bonus points.

I will provide you a server. Your AI must communicate with it. The server will observe your AIs in order to watch if they are cheating or not.

If your AI fails the day of the tournament (cannot communicate with server, cannot be launched automatically, etc), your maximum grade will be 10/20. If you fail to develop an AI, the grade will be 0.

Before the day of the tournament, you will post your source code on edunao.

Rules

In a far far away land, mortal creatures lived a peaceful life. At nightfall, they helplessly witness a fierce struggle between two supernatural species: Vampires and Werewolves. Each night, humans have to stay in their houses.

You will be one of the two species. Your goal: being the dominant species. To increase the number of creatures, you can change humans into your own species: a vampire's bite changes humans into vampires, a werewolf's scratch turns them into werewolves. To change a group of humans, the creatures have to be at least as numerous as them. The creatures can kill each other: they have to be 1.5 time more than their enemies.

The game ends at daylight or when one of the species totally disappears: the winner is the dominant species at this time.

Universe representation

The universe is a n × m grid. Some cases are humans' houses. The universe is totally observable.

Initial state

Two cells are selected to be the starting positions of each specie. All the creatures of the same specie are in the same cell.

Actions

Vampires and werewolves can move across the grid. At each turn, the player as to make at least one move. This consists in moving the totality or just a part of the creatures from one cell to an adjacent cell. The player can make move each group of its monsters.

Figure 1 represents, for a given cell, the possible moves.

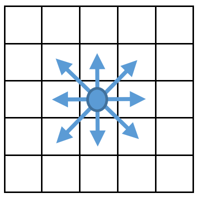
Figure 1: legal moves

When the adjacent cell contains humans, you have to be as numerous as they are to convert them. If you are not, a random battle will start (see section Random battle). If the adjacent cell contains monsters, you have to be 1.5 time more numerous as they are to kill them, otherwise: if they are 1.5 time more than you are, they will kill all the attackers, else a random battle will start.

The next table summarizes the rules:

Rule 1	At least one movement by ply
Rule 2	You can only move your specie groups
Rule 3	You need to have enough creatures on a cell to perform all the moves from this cell
Rule 4	You can move in 8 directions (unless on borders)
Rule 5	In the same ply, a cell cannot be target and source of moves
Rule 6	You have to move at least one creature

Random battle

In the case the player moves to a cell containing enemies or humans and that the number is not correct, a random battle starts.

The end of the battle is given by the following probabilities. Let E1 and E2 be the number of the two species (opponents or humans) and E1 be the number of monsters attacking :

- If E1=E2, then P=0,5 ;
- If E1<E2, then P is given by the linear relation passing through (0 ;0) and (E2 ;0,5), thus $P = \frac{E1}{2\times E2}$;
- Otherwise P is given by the linear relation passing through (CxE2 ;1) and (E2 ;0,5) thus $P = \frac{E1}{E2} - 0{,}5$.

The probability that E1 wins is given by P.

However, the struggle has been tough and some of the attacking monsters died:

- If the attackers win, each of them has a probability P to stay alive. Moreover, if they win over humans, each human has a probability P to survive and to be converted;
- If the attackers loose, each enemy has a probability 1-P to stay alive.

Server communication protocol

To play to the game, you have to communicate with the game server. You will have to connect via a TCP socket and a permanent connexion. The URL and the port are dynamically set and must be command line parameters.

Each packet is composed of a command name coded on 3 bytes, then more bytes for parameters. The command name is given in ASCII.

Communication server → player

SET	Give informations on the grid
HUM	Indicates houses
HME	Indicates your starting position
MAP	Indicates the content of the initial board
UPD	Indicates the modifications during the opponent's turn
END	Indicates that the game ended
BYE	Indicates that you can shut down the communication

Command SET

3 bytes	1 byte	1 byte
SET	n	m

Where *n* and *m* are unsigned integers coded on 1 byte (byte) and indicate respectively the number of rows and columns of the grid.

Command HUM

3 bytes	1 byte	n x 2 bytes
HUM	n	coordinates

where *n* is the number of houses. The list of coordinates of the houses are given as a two unsigned integers coded on 1 byte to represent respectively *x*, *y* the coordinates. The (0;0) cell is at the top left of the grid. Indices are zero based.

Command HME

3 bytes	1 byte	1 byte
HME	x	y

Coordinates of your initial position.

Command UPD

3 bytes	1 byte	n x 5 bytes
UPD	n	List of updates

N is the number of 5-tuples that indicate the changes on the grid. A 5-tuple is composed of 5 bytes:

- X coordinate
- Y coordinate
- Number of humans
- Number of Vampires
- Number of Werewolves.

The update contains your last moves + the opponent's moves.

Command MAP

MAP is like UPD but received once at the beginning of the game.

Commands END and BYE

END and BYE have no arguments. When you receive END, clear the current game and prepare to a new game. When you receive BYE, the connection is shut down.

Communication player → server

NME	Indicates the name of the AI
MOV	Moves creatures from one cell to another

Command NME

3 bytes	1 byte	t bytes
NME	t	Name

t represents the length of the name in characters, encoded in ASCII.

Command MOV

3 bytes	1 byte	n x 5 bytes
MOV	n	List of moves

This packet indicates you want to perform n moves, with 0<=n<=number of groups, each move is characterized by 5 bytes:

2 bytes	1 byte	2 bytes
Source coordinates	Number of creatures	Target coordinates

Course of a game

Your AI must connect to the server. When the connection is established, send NME. In return, the server will answer with SET, HUM, HME, and MAP.

When both players are connected, they receive at each turn UPD. When you receive this command, you have 2 seconds to play and send back a MOV command. Thus, the first player will receive MAP, then UPD without changes.

Time limit

At each turn, you have 2 seconds to play. If no answer is received, the server calls the other player. A game will finish after a maximum time or a maximum number of plys.

Using the GUI

Configuration file

In the configuration file, you have to fill in the fields:

- Ip : server's IP address
- Port : port that must be listened .

Other fields are optional:

- Trace = True allows to write a log
- Map allows to specify a map file

Command line parameters normalization

Your software will be automatically called. The caller will put at the end of the command line, the ip and the port to connect. Please develop your software regarding that convention!

For example :

Python myscript.py –myparams 123.123.3.5 6666

Java myexe.java –myparams 123.123.3.5 6666

myexe.exe –myparams 123.123.3.5 6666