Code Description of Team Sashimi

at the 2nd International AIWolf Competition

October 31, 2020

1 Environment

- Language: Java
- Classpath: `jp.ac.tsukuba.s.s2020602.SashimiRoleAssignPlayer`
- The ZIP file of the source code contains the JAR file: `sashimiAgent.jar`.

2 Overview

The working principles of our agent are inspired by game theory, especially the Bayesian game, which is incomplete with information about the characteristics (often called "types") of other players.

We first consider all combinations of the player roles in the game, and express how probable a player believes they are as a probability distribution over the combinations. Such a probability distribution is called the player's "belief". Our agent updates its belief based on the other players' messages. Our agents also get a predetermined utility at the end of each day of the game, depending on who was eliminated. Based on the belief and the expected utility, our agent decides the broadcasting message and vote so as to obtain the highest utility in the game. Furthermore, for the sake of simplicity of the utility estimation, our agent assumes that all players vote according to the same predetermined method.

3 Agent for 5 player game

As mentioned in the previous section, our agent estimates the belief of each player, including itself, as a probability distribution on the combination of all roles in the game. It is assumed that each player i updates its belief according to the following rules.

- Based on the facts that i knows, such as the i's role and the result of the divination that i actually got, the probability of an impossible combination of roles is set to 0.
- Ignore statements that contradict the facts that i knows.
- Messages that are consistent with the facts that i knows are taken honestly as facts, and only the combinations of applicable roles are kept.
- If the messages that are not ignored are inconsistent with each other, the possibilities are equally divided.

In addition, it is assumed that each player receives each message and decides to vote according to the latest updated belief. Our agent also votes in the same way. On the other hand, when our agent

determines the content of a message, it evaluates the revision of other players' beliefs by issuing the message and anticipate the resulting votes of other players then determines the message with the highest expected utility.

4 Agent for 15 player game

The algorithm determining the behavior of the agent for the 15 player game is simplified to reduce the computational cost.

First, as an alternative to the belief in a 5 player game, we express the degree to which each player is presumed to have a role as a natural number. For example, as the belief of player i, the degree to which j ($\neq i$) is presumed to be a seer or a werewolf is expressed as 1 and 2, respectively. In the initial state, these values in the beliefs of all players are equally set to 0 and are increased or decreased by 1 depending on the content of the message received during the progress of the game.

In addition, it is assumed that each player evaluates the weighted sum of the degree of each role for each player other than itself, and selects the player who seems to have the highest degree of contribution to the opponent's team as the voting destination.

When evaluating the expected utility from a message, one predicts how that statement will change each player's belief and voting destinations in the most probable combination of the roles. The degree of contribution to the villagers team is used to determine the voting destination of the werewolf and the escort destination of the bodyguard, as in the case of determining the voting destination.