Sistemas Multiagente:
Diseño y desarrollo de un agente negociador para la ANAC2019 (Werewolf)

Ramon Ruiz Dolz
Javier Iranzo Sánchez

Febrero 2019

Contents

1 Introducción

En el marco del trabajo de la asignatura de Sistemas Multi-Agente hemos decidido realizar el estudio de una plataforma incorporada de forma novedosa este año en la ANAC2019[1]. En este trabajo se realiza tanto el análisis y estudio de la plataforma AIWolf[2] como una pequeña implementación de un agente capaz de realizar las acciones básicas. AIWolf es una plataforma multiagente que permite jugar al juego del *Werewolf* mediante agentes autónomos inteligentes. Este dominio es realmente interesante desde el punto de vista de la investigación, puesto que varios campos confluyen en él. Al ser un juego de información incompleta eleva la dificultad a la hora de trabajar con el dominio. Además, por su naturaleza, este juego requiere de interacción social como pueda ser la argumentación, el diálogo o la persuasión.

Este trabajo podría considerarse como una introducción a un proyecto de mayor envergadura mediante el cual pretendemos implementar un agente inteligente competitivo en esta plataforma y enviarlo a la competición. El propósito de este trabajo es, mediante el análisis de la plataforma, comprender el funcionamiento de esta y ser capaces de realizar implementaciones sencillas en este entorno.

La estructura del trabajo es la siguiente. En la section 2 se detalla el dominio sobre el que se ha realizado este trabajo, se explican las reglas y se definen los roles existentes. En la section 3 se presenta la plataforma sobre la que se van a lanzar los agentes. Se presenta su arquitectura, se describe en que consiste el proceso de desarrollo de un agente y se explica como lanzar una ejecución. En la section 4 se realiza una breve descripción de las estrategias implementadas para el *BetaAgent* desarrollado en este trabajo para cada uno de los distintos roles existentes en el juego. Finalmente, en la section 5 se analizan las conclusiones alcanzadas mediante este trabajo así como se plantean distintas lineas de trabajo a seguir desarrollando en un futuro.

2 Dominio de desarrollo

2.1 Reglas del juego

Werewolf es un juego que pertenece al arquetipo de los juegos de roles ocultos. Estos juegos tienen en común el hecho de que se les asigna un rol a cada jugador (normalmente de manera aleatoria) al comienzo de la partida, y este rol condiciona las habilidades, funcionamiento y objetivos del jugador durante la partida. Adicionalmente, los jugadores no tienen información perfecta sobre los roles de el resto de jugadores, por lo que una parte importante de una estrategia ganadora consisten en averiguar información sobre el resto de jugadores a la vez que se mantienen ocultos nuestros objetivos del resto de jugadores.

[1] `http://web.tuat.ac.jp/~katfuji/ANAC2019/#werewolf`
[2] `http://aiwolf.org/en/`

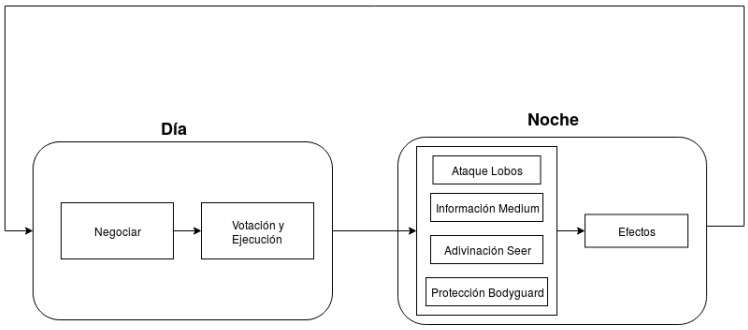

Figure 1: Esquema del desarrollo de una partida de Werewolf

En Werewolf existen dos grandes equipos: una mayoría desinformada, los *villagers*, que desconocen por completo el rol del resto de jugadores, y los *werewolves* u hombres lobos, que conocen la identidad del resto de *werewolves*. El objetivo del cada equipo es acabar con el equipo contrario.

2.2 Desarrollo de la partida

Una partida de Werewolf consiste en una serie de turnos de Día y Noche que se van alternando uno detrás de otro. Durante el Día, se realiza una votación para decidir que persona será ejecutada ese día, en un intento de los aldeanos de combatir el problema de los ataques de hombres lobo. Durante este proceso, los jugadores pueden comunicarse entre ellos. Así pues, el objetivo de los aldeanos es coordinarse para poder ejecutar a un lobo durante el día, ya que es la única manera que tienen de acabar con los lobos, mientras que los lobos tienen que intentar evitar ser descubiertos para que los aldeanos se vayan matando entre ellos, causando desconfianza entre los aldeanos. Al finalizar este proceso de deliberación, aquel jugador que haya recibido un mayor número de votos es ejecutado y eliminado de la partida.

El turno de Noche sigue al turno de Día. Durante la noche, los lobos salen de caza y atacan la aldea. Los lobos se ponen de acuerdo entre ellos sobre que jugador atacar, y el aldeano seleccionado muere y es eliminado del juego. El proceso de ataque de los lobos sucede todos los turnos, y es su mejor herramienta para ir eliminando a los aldeanos de manera garantizada.

Adicionalmente, durante la fase de Noche, otros roles especiales también actúan. Evidentemente, esto solo sucede cuando alguno de esos roles están presentes en la partida.

Es importante destacar que no se revela el rol de los jugadores muertos, lo que añade una mayor incertidumbre a la partida. [3]

El esquema de desarrollo de una partida se muestra de manera gráfica en la Figura 1.

[3]Existen variantes del juego en las que si que se revela esta información.

2.3 Roles especiales

En la versión de Werewolf a tratar, existen 4 roles especiales a parte de los básicos Villager y Werewolf, que son los siguientes:

- **Seer**: Durante la noche, el Seer puede escoger a otro jugador para averiguar más información sobre él. El Seer recibe únicamente el equipo al que pertenece el jugador seleccionado (es decir, aldeano u hombre lobo), pero no recibe información sobre su rol. Este es el único rol que permite a los aldeanos obtener información fidedigna sobre la identidad de los jugadores, y representa una de las mayores bazas de los aldeanos.

- **Medium**: El Medium es informado sobre si la persona que fue ejecutada durante el día era un hombre lobo o no. Este rol permite tener una idea de como está evolucionando la partida, e incluso puede servir para comprobar si existen patrones en las votaciones que nos hagan sospechar de algún "aldeano" en particular.

- **Bodyguard**: El Bodyguard tiene la capacidad de proteger a otro jugador del ataque de los lobos. Durante la noche, el Bodyguard selecciona a un jugador, que recibe inmunidad a los ataques de los lobos durante esa noche. No hay restricción sobre la selección, y el Bodyguard puede incluso seleccionarse a si mismo. El Bodyguard tiene que intentar ir un paso por delante de los lobos y proteger a sus objetivos a los aldeanos importantes.

- **Possessed**: El jugador Possessed no tiene habilidades especiales, pero su condición de victoria es la contraria que un aldeano normal: El jugador Possessed gana la partida si ganan los lobos, por lo que tendrá que conspirar contra el resto de aldeanos si quiere salir victorioso. La identidad de este jugador no es revelado a los lobos al principio de la partida.

La presencia de estos roles especiales añade complejidad al juego y permite desarrollar una mayor serie de estrategias.

3 Plataforma AIWolf

La plataforma multiagente AIWolf es una herramienta desarrollada por Fujio Toriumi, Hirotaka Osawa y Atom Sonoda entre otros investigadores, de las universidades de Tokyo y Tsukuba entre otras instituciones. El proyecto AIWolf[4] nace con el propósito de desarrollar agentes inteligentes capaces de coordinarse, cooperar y competir en un entorno con información incompleta. A diferencia de la mayoría de juegos con los que se han conseguido grandes hitos en el mundo de la inteligencia artificial, los juegos con información incompleta son una área todavía a explorar. El juego del Werewolf es realmente interesante desde el punto de vista de la inteligencia artificial puesto que obliga a los jugadores a interactuar entre sí, argumentar, persuadir y juzgar la información recibida. Siendo todos temas candentes en el mundo de la investigación actual.

[4] http://aiwolf.org/en/

En los siguientes apartados, siguiendo la información publicada en [1], se describen los principales elementos de AIWolf como son el servidor y los clientes. Además también se explica el funcionamiento de una ejecución del juego en esta plataforma.

3.1 Arquitectura de la plataforma

Debido a su naturaleza, la plataforma tiene una arquitectura cliente-servidor. Esta permite realizar la comunicación mediante el protocolo TCP/IP o bien mediante una API interna.

3.1.1 Servidor

En AIWolf, el servidor desempeña un papel principal en el desarrollo de las ejecuciones de las partidas. El servidor espera activo a que se conecten todos los clientes necesarios para iniciar la partida, y en ese momento inicia el juego. Todo lo correspondiente con el desarrollo de la partida, las normas del juego y la gestión de los agentes es llevado a cabo por el servidor de AIWolf. Concretamente el servidor pide a los clientes que comuniquen sus acciones y estos responden cuando se les requiere.

3.1.2 Cliente

Cada cliente en AIWolf consiste en un agente jugador. A diferencia del servidor que siempre es el mismo, puesto que las normas del juego o el desarrollo de la partida es invariable. En AIWolf cada cliente puede ser diferente. Cada agente puede ser desarrollado con estrategias diferentes para cada rol. De hecho, el propósito de la competición es desarrollar un agente capaz de interactuar y cooperar de forma efectiva con otros agentes distintos existentes en la partida.

3.2 Desarrollo de un agente en AIWolf

En este trabajo se ha implementado un agente *BetaAgent* a partir del código de ejemplo *SampleAgent*[5]. Siguiendo la estructura de ejemplo, este agente esta estructurado con una clase *BetaBasePlayer* la cual tiene implementados todos los métodos y acciones básicos para cualquier rol. Esta clase es heredada tanto por el *BetaVillager* como por el *BetaWerewolf* que sobreescriben los métodos necesarios con tal de actuar de forma coherente con los roles Villager y Werewolf respectivamente. Además, *BetaVillager* también es heredada por cada uno de los roles pertenecientes al conjunto de Villagers existente, *BetaSeer*, *BetaMedium*, *BetaBodyguard* y *BetaPossessed*. La implementación de todas estas clases correspondientes a los distintos roles esta explicada con mayor detalle en la section 4 donde se explica la estrategia implementada para cada rol.

[5] https://github.com/aiwolf/AIWolfClient/tree/0.4.x/src/org/aiwolf/sample/player

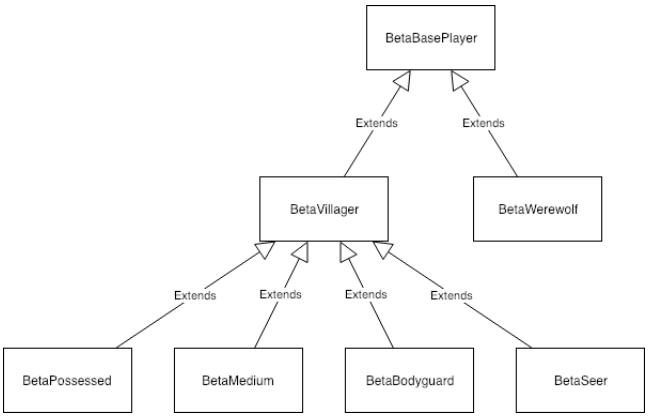

Figure 2: Diagrama de herencias entre clases de un agente.

3.3 Comunicación entre agentes

La interacción es algo fundamental en un juego de estas características. Para comunicarse, los agentes utilizan un protocolo comunicativo propio de la plataforma AIWolf. Este protocolo de comunicación está inspirado en la comunicación entre usuarios en las versiones de Werewolf-BBS (*Bulletin Board System*). Es importante remarcar que el protocolo de comunicación actual de la plataforma todavía esta en un estado prematuro y, por lo tanto, únicamente permite realizar declaraciones sin justificación (tal como cabría esperar en un entorno argumentativo). Es por esto, que la reflexión o la evaluación sobre si creer o no creer una información determinada, es realizada a nivel interno tras escuchar los distintos mensajes de otros agentes. No existe una fase de diálogo directo como tal en la que un agente pueda atacar o apoyar los mensajes lanzados por otros agentes. En esta versión de la plataforma, cada agente dispone de 10 protocolos de comunicación distintos:

- **estimate(Agent, Role)** Mediante este protocolo, un agente determinado es capaz de expresar sus sospechas sobre el rol jugado por otro agente dentro de la partida.
- **comingout(Agent, Role)** Este protocolo permite a un agente revelar su rol. Esta información no tiene porque ser cierta.
- **divined(Agent, Species)** Mediante este protocolo, un agente (Seer o no Seer) comunica el resultado obtenido tras realizar una adivinación.
- **identified(Agent, Species)** Similar al anterior, mediante este protocolo un agente (presuntamente Medium) informa del equipo al que pertenece el agente ejecutado.
- **guarded(Agent)** Este es el protocolo mediante el cual, el supuesto Bodyguard informa que ha protegido a un agente determinado.

- **vote(Agent)** Protocolo para comunicar la elección del voto para la siguiente ejecución.
- **agree(day, id)** Protocolo para mostrar acuerdo con un agente determinado un día en concreto.
- **disagree(day, id)** Justo al contrario que en el anterior caso, este es el protocolo mediante el cual se comunica el desacuerdo con un agente determinado un día en concreto.
- **skip()** Este protocolo permite a un agente saltarse su turno para hablar y esperar hasta el próximo turno.
- **over()** Similar al anterior, haciendo uso de este protocolo el agente salta el turno actual y acepta finalizar la conversación.

Todos estos protocolos permiten a cada agente expresar sus opiniones o engañar a los demás agentes. La plataforma AIWolf proporciona tanto el generador como el analizador de mensajes.

3.4 Ejecución

Existen tres opciones diferentes para ejecutar una partida en la plataforma. AIWolf dispone de un script *AutoStarter.sh* que lanza automáticamente tanto el servidor como los clientes conectados a este siguiendo la configuración especificada en los ficheros de configuración correspondientes. De esta forma es posible lanzar la plataforma con la configuración deseada únicamente modificando estos ficheros. Otra opción es lanzar servidor y cliente por separado. En este caso, en primer lugar se lanza el servidor. Para ello se dispone de un script *StartServer.sh* que abre una interfaz de usuario en la cual se pueden configurar el puerto de conexión, el número de jugadores en la partida y el registro (*log*) de la partida. Con estos parámetros ajustados según nuestras preferencias, se puede activar el servidor para que espere conexiones de clientes tal y como muestra la Figure 3.

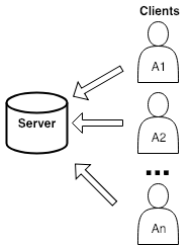

Figure 3: Conexión de los clientes al servidor.

Por otra parte, a la hora de lanzar los clientes existen dos posibilidades. Los agentes cliente pueden ser lanzados desde la terminal o desde una interfaz

de usuario. Para ello disponemos de dos scripts diferentes *StartClient.sh* y *StartGUIClient.sh* respectivamente. Mientras que para lanzar el agente desde la terminal es necesario especificar todos los parámetros necesarios para realizar la conexión correctamente, mediante el uso de la interfaz es posible de realizar este ajuste con mayor comodidad.

Cuando todos los usuarios se hayan conectado correctamente al servidor, puede comenzar el juego. Es importante remarcar que cada agente, tal y como se explicará en la siguiente sección, es capaz de jugar con cada rol existente en el juego. Es por esto que, tal y como se muestra en la Figure 4, previo al inicio de la partida se realiza una asignación de roles a cada agente. Para ello cada agente contiene un fichero Java preparado para instanciar cada agente con un rol determinado en función del resultado de la asignación de roles.

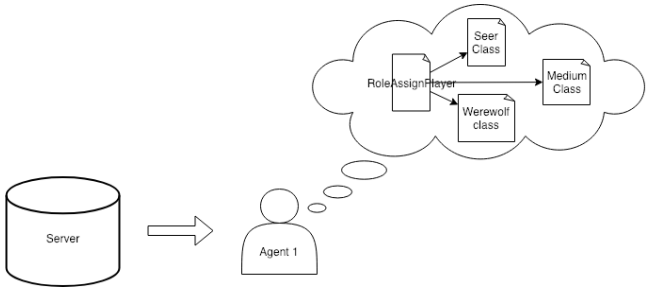

Figure 4: Asignación de roles a cada agente.

En la Figure 5 se puede observar la interfaz de AIWolf al inicializar una partida. En esta partida, compuesta por cinco jugadores, se han asignado los siguientes roles: Sample2 es el *Seer*, Sample1 es uno de los *Villagers*, Sample3 juega como *Werewolf*, BetaAgent es el otro *Villager* y Sample4 es el *Possessed*.

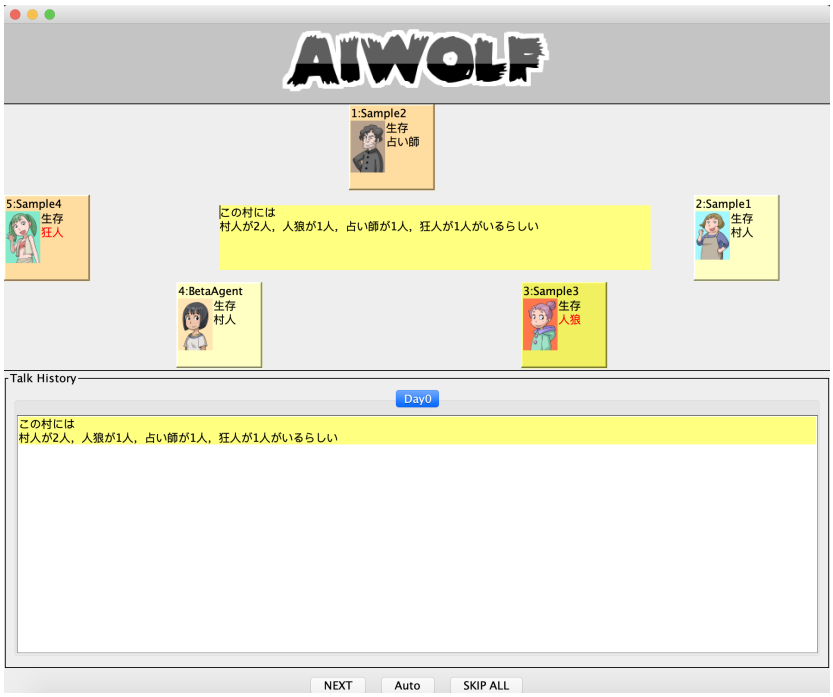

Figure 5: Pantalla inicial de una partida en AIWolf con cinco usuarios.

Una vez los roles hayan sido asignados a cada agente, es posible iniciar una partida. Una partida en AIWolf es esencialmente igual a una partida tal y como se ha explicado en la section 2. Tal y como se aprecia en el diagrama de flujo de la Figure 6, cada rol realiza una serie de acciones específicas. Estas acciones serán las que definan la estrategia a seguir por cada agente. Además de esto, el servidor lleva un control de que cada acción sea realizada en un momento adecuado de la partida. De esta forma es posible conseguir un funcionamiento coherente por parte de los agentes en el contexto del juego.

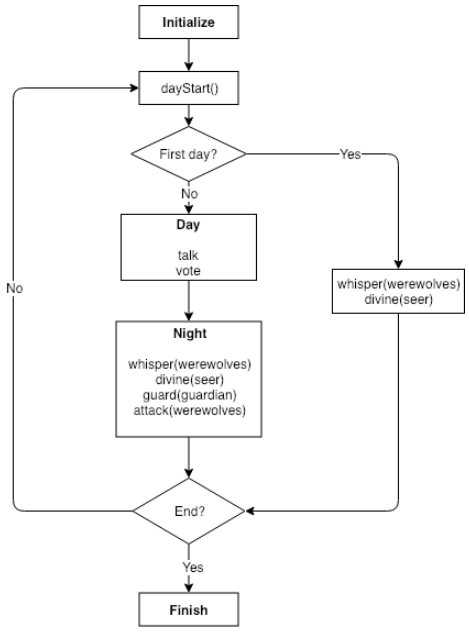

Figure 6: Diagrama de flujo de una partida en AIWolf.

La partida finaliza cuando existe un ganador. En el momento en el que $werewolves = \emptyset$ o bien $villagers = \emptyset$ el juego termina mostrando en pantalla el equipo ganador. Al finalizar una partida, es posible volver a jugar otra con la misma asignación de roles o bién reiniciar el servidor para realizar una rotación.

4 Estrategias implementadas para nuestro agente

En la plataforma AIWolf, un agente representa a un jugador del hombre lobo. Por lo tanto, cada agente debe ser capaz de simular un comportamiento razonable para cada una de las situaciones que se pueden presentar dentro del marco de este juego. Como se ha explicado en la sección anterior, hasta que no empieza la partida no se sabe que rol va a desempeñar cada agente. Esto se debe a que el servidor asigna aleatoriamente un rol a cada agente jugador. Es por esto que cada agente estará preparado para actuar conforme a cada uno de los roles existentes en la partida.

A continuación se detallan las múltiples estrategias definidas para nuestro agente *BetaAgent* para cada uno de los roles asignables. Puesto que se ha trabajado con una nueva plataforma no estudiada durante el curso y debido a la escasa documentación existente entorno a esta, cabe destacar que el propósito de este trabajo no se centra en la implementación de estrategias competitivas, si no en la comprensión de la arquitectura y el funcionamiento de la propia plataforma en sí. Por lo tanto, algunas de estas estrategias pueden parecer

triviales. Además, para la estructura básica de los agentes se ha mantenido el código proporcionado por los desarrolladores de la plataforma. Concretamente se han modificado los métodos relacionados con la comunicación entre agentes y la votación del agente a ejecutar. Con estas modificaciones se ha conseguido generalizar el comportamiento de nuestro agente, puesto que el agente de ejemplo es un agente enfocado a jugar siempre contra él mismo y en una configuración de partida determinada. A continuación se describen las estrategias que hemos considerado para cada rol.

4.1 Estrategia Villager

En primer lugar, se ha modificado la filosofía del Villager. En el código de ejemplo, únicamente mienten los Werewolves y los Possessed. Sin embargo esto es poco realista, puesto que en una partida puede mentir cualquier jugador independientemente del bando asignado. Nuestra aproximación para el rol Villager, puesto que no tiene ninguna habilidad especial, consiste en revelarse como Seer. De esta forma tratará de atraer a los lobos en la siguiente noche y así, los roles realmente relevantes dispondrán de un mayor tiempo para adivinar la identidad de los Werewoves. Esta estrategia se basa intuitivamente, en la mejor elección para los Werewolves que consiste en quitarse de encima al único rol que es capaz de descubrirlos, el Seer. Por lo tanto, en la fase de hablar, nuestro Villager revelará su identidad como Seer, únicamente en el caso de que no haya otro Seer revelado previamente.

Por otra parte, en la fase de votación, nuestro Villager realizará varias comprobaciones. En primer lugar, en el caso de encontrar una adivinación que indique que él mismo es el Werewolf, se anotará el agente que ha lanzado la adivinación falsa como agente del equipo Werewolf. Otra forma de hallar sospechosos consiste en comprobar incoherencias entre la lista de adivinaciones y la lista de identificaciones. En el caso de existir dos identificaciones incoherentes, si existe alguna adivinación que verifique alguna de ambas propuestas, se añade el agente creador de la propuesta contraria a la lista de sospechosos. Esta incoherencia también se comprueba en adivinaciones. Finalmente, si no encuentra sospechoso vota al lobo de forma aleatoria.

4.2 Estrategia Seer

Las principales modificaciones en nuestro agente Seer consisten en la modificación de las respuestas a sucesos en la partida. En primer lugar, nuestro agente Seer no acusará de Werewolf a otro agente que se revele como Seer, sencillamente lo mantendrá en la lista de sospechosos. Sin embargo, en el caso de existir incoherencias entre sus adivinaciones y las adivinaciones promulgadas por otro agente, entonces sí añadirá al otro agente a la lista de Werewolves (aunque posteriormente comprobará si este es Possessed). También se ha modificado la revelación del rol. En nuestra estrategia el agente Seer real únicamente revelará su rol en el caso de adivinar a un Werewolf con el objetivo de persuadir a los demás agentes en sus votaciones.

4.3 Estrategia Medium

La estrategia del Medium es realmente sencilla. Si identifica que un Werewolf ha muerto en el turno anterior, lo comunica a los demás. Por otra parte, en lo que respecta a elegir candidato para la ejecución, de forma similar al Villager o al Seer, comprueba que no exista ningún tipo de incoherencias entre las adivinaciones que se han ido lanzando y las identificaciones de los muertos que realiza el propio Medium. En caso de existir alguna incoherencia se añade el sospechoso como candidato a ser ejecutado. En caso de no tener ningún candidato vota aleatoriamente entre los agentes vivos.

4.4 Estrategia Bodyguard

La estrategia del agente Bodyguard es fundamentalmente la misma que la del agente Villager pero con una habilidad especial de protección. Siguiendo la filosofía de los demás roles, nuestro Bodyguard se encargará de escoger un agente a proteger durante la noche con el siguiente orden de preferencias. En caso de existir algún Seer revelado, este será candidato a ser protegido. En caso de no existir ningún Seer se tratará de proteger a un Medium y si tampoco hay Mediums en este caso se protegerá a un Villager aleatorio. De esta forma se trata de prolongar la vida de los Seer ya que son los agentes capaces de descubrir a los Werewolves.

4.5 Estrategia Werewolf

El Werewolf es un rol ligeramente diferente. Puesto que es el objetivo de todos los demás agentes, al comenzar la partida debe escoger un rol falso. Este rol es elegido aleatoriamente entre Villager, Seer y Medium al inicializar la partida con tal de realizar acciones coherentes durante el transcurso de esta. Además, para encontrar a sus aliados humanos, los Possessed, lleva un control de incoherencias en las adivinaciones puesto que ningún rol "benévolo" se dedica a lanzar adivinaciones falsas. Detectando los agentes Possessed es posible evitar malgastar una votación o un asesinato en un agente que es aliado a su equipo. Además el Werewolf dispone de un método para añadir información falsa a la partida consecuentemente con el rol falso que haya escogido.

En cuanto a las votaciones, el Werewolf tratará de votar a aquellos agentes que hayan revelado el mismo rol que el rol falso escogido por él. Con esta estrategia se pretende conservar la coartada del Werewolf. Además, el Werewolf también lleva la cuenta de los roles que se van revelando. En caso de estar falsificando el rol de Seer o Medium y alguien revelar su rol como uno de estos, el Werewolf pasará a actuar como un simple Villager con tal de evitar levantar sospechas. Finalmente, el principal cambio viene dado en la estrategia de ataque por la noche. Nuestros Werewolfs matarán a los Villagers siguiendo un orden de preferencia. En primer lugar marcaran su objetivo en los Seer, puesto que son los únicos que pueden descubrirles. Una vez asesinados el siguiente objetivo serán los mediums que pueden hallar incoherencias entre la información falsa

y los sucesos reales. Los siguientes objetivos serán los Bodyguard, Villager y Possessed en este orden.

4.6 Estrategia Possessed

Finalmente, la estrategia seguida por los Possessed es similar a la de los Werewolves. Un agente Possessed jugará con un rol falso y tratará de confundir al resto de los agentes introduciendo información falsa en las conversaciones.

5 Conclusión y trabajo futuro

A lo largo de esta memoria se ha explicado el dominio en el cual se ha trabajado, así como el funcionamiento de la plataforma multi-agente empleada y un ejemplo de ejecución de esta. También se ha descrito la implementación de las estrategias para el agente desarrollado. Por lo tanto, hemos cumplido todos los objetivos planteados al inicio del trabajo, tanto el estudio de la plataforma, como la implementación de un pequeño agente inteligente.

Tal y como se ha descrito en la introducción, este trabajo podría verse como una introducción a un proyecto de mayor envergadura. Nuestras ideas en un futuro consisten en la implementación de un agente competitivo capaz de alcanzar la final de la liga de *Werewolf* en la ANAC2019. Este juego puede verse como un problema de clasificación de agentes. Si se consigue desarrollar un agente capaz de predecir correctamente el rol de otro agente a partir de pocas acciones, es mucho más probable que nuestro agente se anticipe a los demás participantes. Es por esto, que de cara a un futuro cercano nos gustaría implementar algún tipo de agente capaz de predecir los roles de los rivales a partir de un modelo estadístico previamente estimado.

References

[1] Fujio Toriumi, Hirotaka Osawa, Michimasa Inaba, Daisuke Katagami, Kosuke Shinoda, and Hitoshi Matsubara. Ai wolf contest—development of game ai using collective intelligence—. In *Computer Games*, pages 101–115. Springer, 2016.