T08: Jogo "Werewolves of Miller's Hollow"

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Agentes e Inteligência Artificial Distribuída

Grupo T08_2:
Hélder Manuel Mouro Antunes - up201406163
Inês Filipa Noronha Meneses Gomes Proença - up201404228
João Filipe Pereira da Costa - up201403967

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

10 de Dezembro de 2017

Conteúdo

1 Objetivo

1.1 Descrição do cenário

O jogo "Werewolves of Miller's Hollow"é um jogo social de dedução e deceção. O jogo passa-se numa pequena aldeia cujo nome é Millers's Hollow, na qual alguns dos habitantes são, na realidade, lobisomens. Cada jogador toma o papel de um habitante de Millers's Hollow, podendo ser tanto um aldeão como um lobisomem, e tenta sobreviver e eliminar a equipa adversária [1].

O jogo desenrola-se através da repetição de duas fases, dia e noite. Durante o dia, os jogadores discutem com o objetivo de identificar quem são os lobisomens. Neste período, a negociação consiste em trocar informações, que podem ser verdadeiras ou falsas. Por fim, os jogadores votam em alguém que acreditam ser um lobisomem. No caso dos lobisomens, o voto é "fingido"uma vez que não lhes interessa votar na própria equipa. Se algum jogador for selecionado ao fim da votação (sem empate), é executado. Durante a noite, os lobisomens elegem um jogador para ser morto. Nesta fase, os aldeões adormecem, exceto aqueles com papéis especiais que podem ativar as suas habilidades, a fim de tentar recolher qualquer informação sobre as identidades dos lobisomens, matar ou salvar outros jogadores. Esses papéis especiais dependem da versão do jogo que está a ser jogado. Logo que os jogadores são mortos, o seu papel é revelado aos restantes.

Antes de cada jogo, um dos jogadores assume o papel de moderador, não participando diretamente no jogo, mas controlando o seu desenrolar. O jogo termina quando uma das equipas perde todos os seus membros, ou seja, quando todos os lobisomens ou todos os aldeões forem eliminados. O jogo pode terminar mais cedo, dependendo das regras, quando o número de lobisomens for igual ao número de aldeões.

1.2 Objetivos do trabalho

O objetivo do trabalho é a implementação do jogo "Werewolfs of Miller's Hollow"utilizando agentes BDI, cada um dos quais simula um jogador.

Os agentes serão capazes de:

- comunicar entre si partilhando as suas suspeitas e alegações, tanto verdadeiras como falsas;
- deduzir e concluir os papeis dos outros agentes e
- tentar aumentar as suas chances de sobreviver e ganhar.

A distribuição dos agentes é configurável pelo utilizador, que pode alterar o número e tipo de cada categoria de agentes.

2 Especificação

No total, serão definidos 5 agentes. Haverá um agente orquestrador do jogo designado por master. Os restantes representarão personagens reais do jogo e serão chamados por *villager*, *werewolf*, *diviner* e *doctor*. Estes agentes serão de 3 tipos: aleatórios, estratégicos e BDI. Os agentes aleatórios tomarão as suas decisões de forma aleatória pelo que apenas será explicado o comportamento dos agentes estratégicos e BDI.

2.1 Agente *master*

O objetivo do agente *master* é orquestrar todas as fases do jogo, mediando a comunicação entre os agentes que representam as personagens reais do jogo. Mais particularmente este agente:

- Cria os outros agentes para estes entrarem no jogo
- Envia informação aos outros agentes sobre o ambiente (fase do dia, resultados das votações)
- Recebe informação dos outros agentes sobre as decisões de voto
- Informa o *diviner* do papel real de um jogador da sua escolha por noite
- Salva, se necessário, o jogador escolhido pelo *doctor*

Na figura 1, podes-se ver a interação entre este agente e os outros ao iniciar o jogo. As restantes interações serão ilustradas pelas figuras das subsecções seguintes para simplificar a sua compreensão.

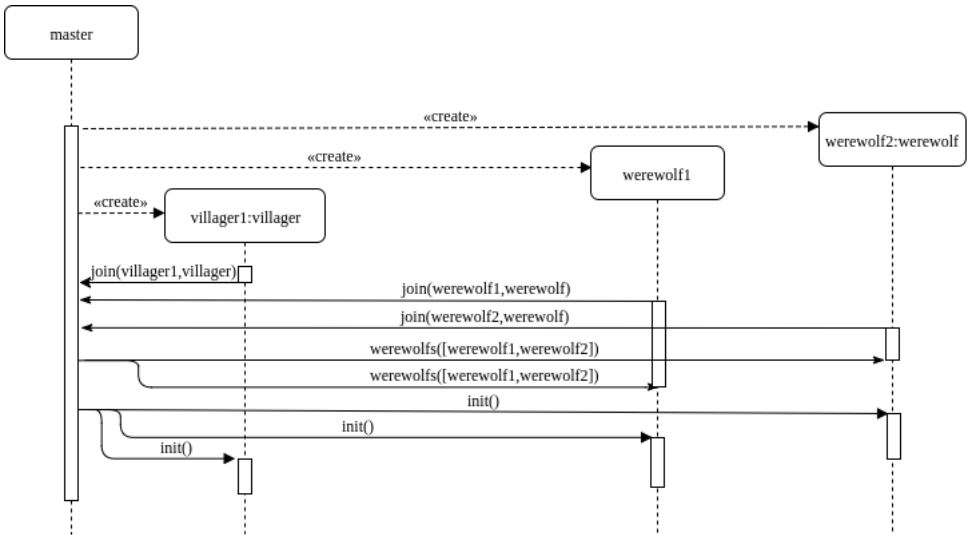

Figura 1: Diagrama de interação do agente master para iniciar o jogo

2.2 Agente *villager*

Este agente estando a simular um jogador com o papel de um aldeão comum só poderá comunicar durante o dia com o intuito de descobrir e, posteriormente, votar para eliminar os lobisomens.

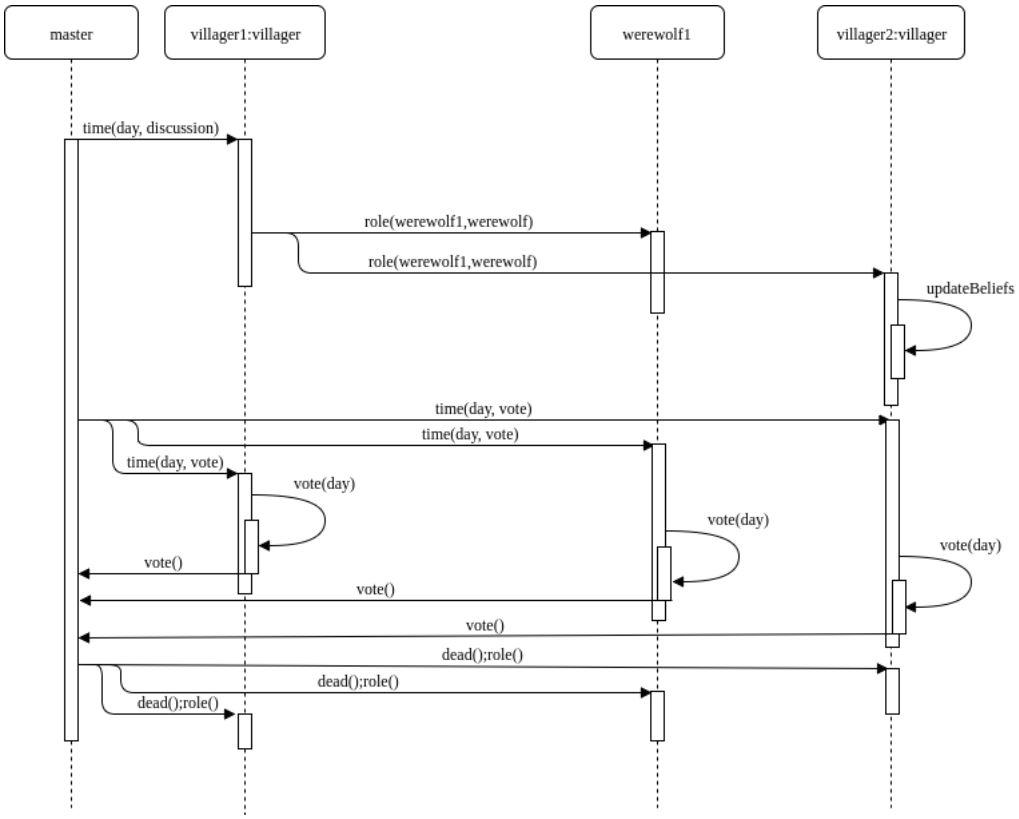

Figura 2: Diagrama de interação do agente *villager* durante um dia

2.2.1 Modo estratégico

Visto que os lobisomens conhecem a verdadeira identidade de todos os jogadores e que, por isso, votarão certamente nos aldeões, a probabilidade de um lobisomem ser pouco votado é superior. Dessa forma, este agente votará num dos jogadores vivos menos votados até ao momento. Na primeira votação, por ausência de informação, o agente votará aleatoriamente.

2.2.2 Modo BDI

Inicialmente, para este agente, o valor da crença de que um qualquer jogador seja lobisomem (suspeita) é zero, sendo que, na primeira ronda, este agente votará aleatoriamente. Nas rondas seguintes, votará no jogador sobre o qual a sua suspeita é maior.

O valor da crença é calculado durante o período de discussão com base na confiança que o agente tem na fonte de informação. Dessa forma, aumenta a suspeita se o fator de confiança for positivo e diminui caso contrário.

No final de cada votação, ao saber o papel verdadeiro do agente que morreu, o fator de confiança nos agentes que manifestaram a sua suspeita sobre qual seria o papel do morto é atualizado. Por outras palavras, se o agente se enganou o fator de confiança desce e o contrário se o agente estava correto.

2.3 Agente *werewolf*

Este agente simula um jogador com o papel de lobisomem. Durante o dia deve comportar-se como um aldeão, tentando não ser descoberto e influenciar as votações a seu favor. Durante a noite, juntamente com os outros agentes com o papel de lobisomem, inicia uma fase de discussão e uma votação privadas, com o objetivo de escolher um aldeão para matar.

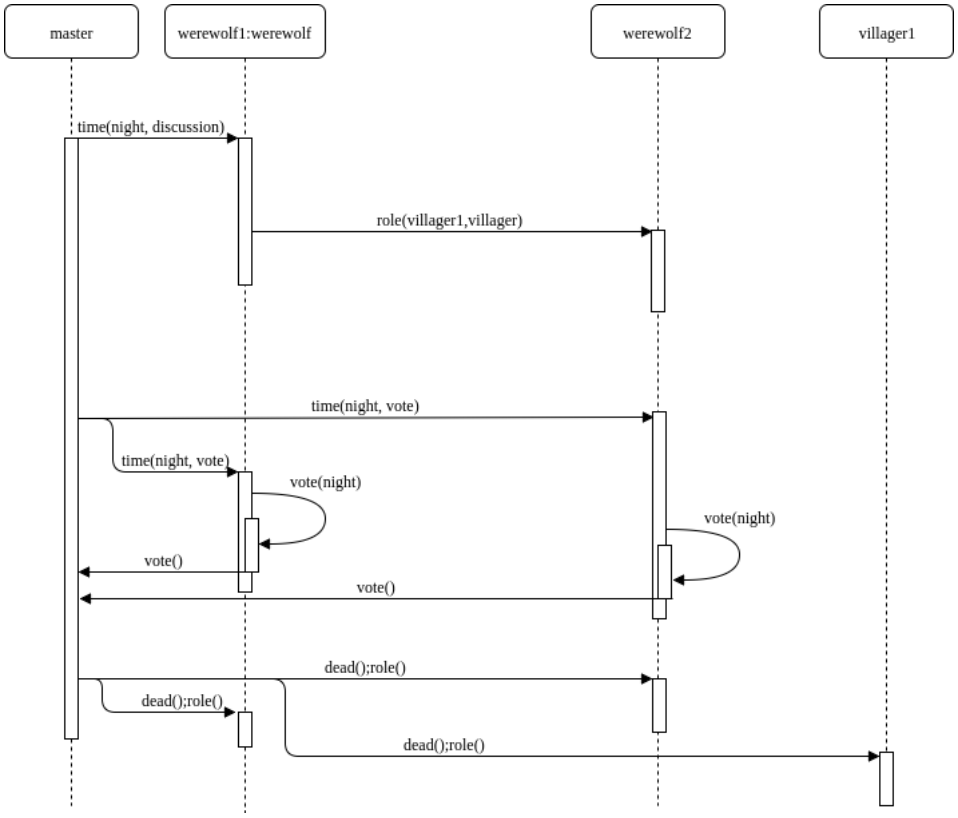

Figura 3: Diagrama de interação do agente werewolf durante a noite

2.3.1 Modo estratégico

Neste modo, o agente vota identifica todos agentes que votaram num lobisomem na ultima ronda, acusando e votando aleatoriamente num dos agentes

dessa lista.

2.3.2 Modo BDI

No modo puramente BDI, o lobisomem baseia-se nas crenças calculadas durante as fases de discussão acerca dos papeis de cada agente. Este calculo é semelhante ao utilizado pelo aldeão, com a nuance de que inicialmente o agente acredita que todos os outros agentes, que o agente Master não indicar que é são lobisomens, são aldeões.

O agente, durante o dia escolhe para votar e indica que este é um lobisomem. Durante a noite o processo é semelhante, no entanto, a acusação é feita com o papel que o agente realmente acredita que o alvo seja.

Este alvo é escolhido da seguinte forma:

- *Diviner* tem prioridade sobre todos os outros agentes
- *Doctor* tem prioridade sobre os *Villagers*
- agente cuja crença de ser determinado papel seja superior.

2.4 Agente *diviner*

Este agente é um aldeão com a capacidade de, uma vez a cada noite, poder descobrir o verdadeiro de um jogador. Durante o dia, comporta-se como um aldeão normal, optando por jogar com a mesma estratégia um agente *villager* no mesmo modo.

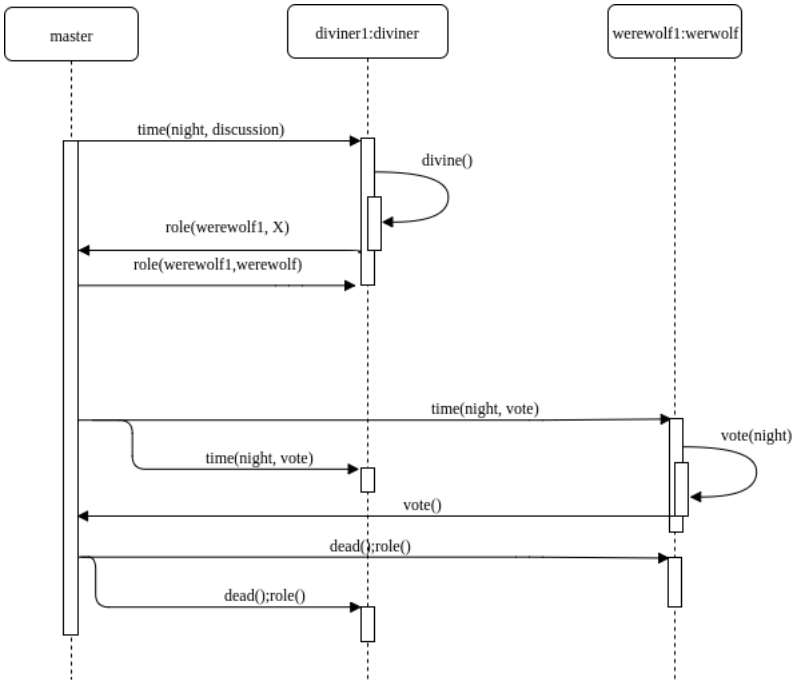

Figura 4: Diagrama de interação do agente *diviner* durante uma noite

2.4.1 Modo estratégico

Durante a noite, este agente vai procurar saber a identidade do jogador do qual mais suspeita. A maior suspeita irá recair sob os agentes menos votados durante o dia, cujo papel o *diviner* ainda não confirmou com as suas habilidades.

2.4.2 Modo BDI

Tal como no modo estratégico, durante a noite, este agente irá confirmar as suas suspeitas sob o papel de um jogador. A diferença é que estas suspeitas serão também baseadas nas acusações ouvidas durante o dia.

2.5 Agente *doctor*

Um doutor tem a capacidade de uma vez por noite, antes na votação dos lobisomens, curar um jogador. Esta habilidade previne o jogador escolhido pelo doutor de morrer durante a noite mesmo for escolhido pelos lobisomens.

Durante o dia, comporta-se como um aldeão normal, optando por jogar como um agente *villager*.

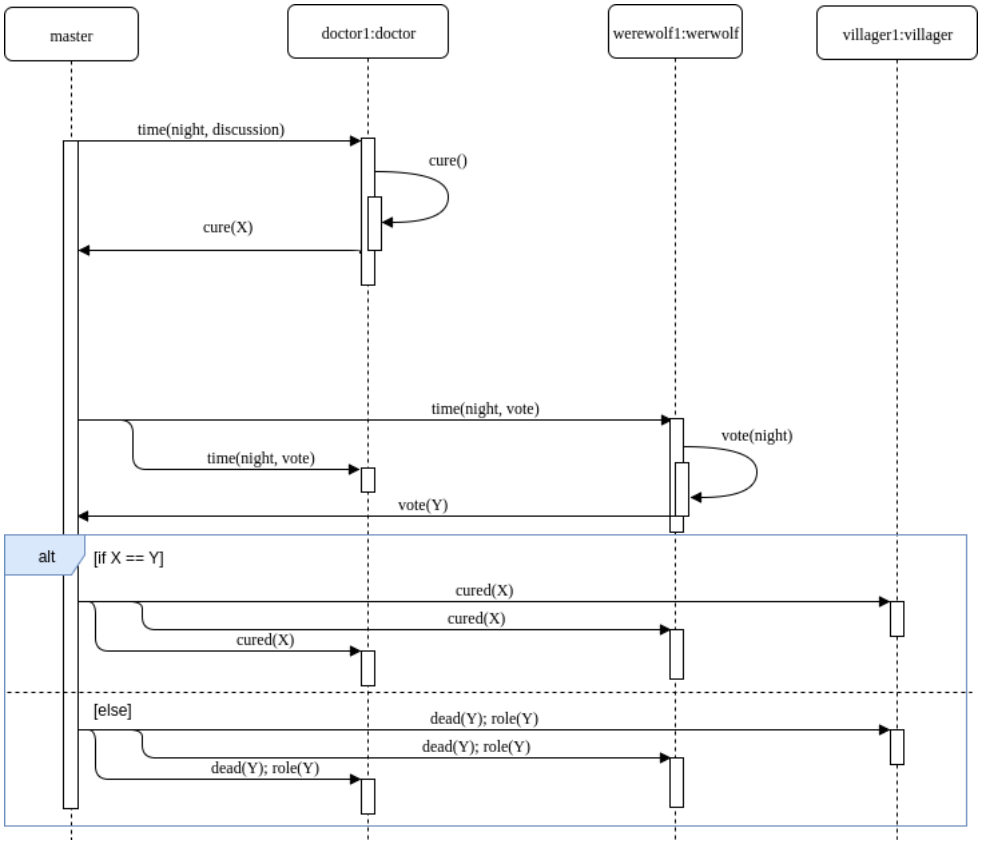

Figura 5: Diagrama de interação do agente *doctor* durante uma noite

2.5.1 Modo estratégico

Durante a noite, o agente *doctor* irá tentar proteger os aldeões que acredita que têm um maior risco de morrer.

Assim, não sabendo à partida quem é aldeão ou lobisomem, este agente irá curar o agente mais votado até ao momento (este agente terá maior probabilidade de ser um aldeão, uma vez que os lobisomens só votam em aldeões e os aldeões, não tendo conhecimento da identidade dos jogadores, tanto podem votar em aldeões como em lobisomens).

A partir do momento em que um lobisomem morre e o seu papel é revelado, o doutor irá preferir curar o jogador ainda vivo com mais votos dados pelos lobisomens conhecidos.

2.5.2 Modo BDI

No modo puramente BDI, o doutor irá curar o agente que menos acredita ser lobisomem. Estas crenças terão origem nas acusações partilhadas durante o dia e no grau de confiança que o agente terá nelas.

3 Desenvolvimento

3.1 Plataforma/ferramenta utilizada

Tendo em conta que o trabalho consiste na implementação de um jogo de negociação cooperativa [2] e, consequentemente, se baseia, na sua maioria, no diálogo entre os jogadores, como descrito na secção 1.1, a arquitetura BDI parece ser a mais indicada para estes agentes. Desse modo, decidiu-se utilizar a ferramenta Jason de modo a implementar os agentes BDI de forma mais fácil e com melhor qualidade.

Jason é um interpretador para uma versão extendida de AgentSpeak implementado em Java [3]. Com este interpretador *open-source*, é possível o desenvolvimento de agentes de arquitetura BDI com comunicação inter-agentes baseada na fala.

Para além de interpretar a linguagem AgentSpeak original, Jason permite conhecimento sobre a origem das crenças e a possibilidade de lidar com conhecimento de um mundo aberto. Dessa forma, os agentes BDI criados com esta ferramenta conseguem facilmente ter a informação necessária para raciocinar sobre a possibilidade do conhecimento que recebe de outros não corresponder à realidade em que se insere. Além disso, a funcionalidade do "mind-inspector"torna-se também muito interessante para perceber se os agentes estão a agir corretamente, de acordo com o seu conhecimento do mundo, especialmente no decorrer do desenvolvimento do trabalho como ferramenta de *debug*.

O projecto foi desenvolvido em sistema operativo Linux com recurso aos IDEs Eclipse Oxygen e jEdit da ferramenta Jason.

3.2 Estrutura da aplicação

O código relativo à simulação do jogo encontra-se na pasta /jason_simulation. Nela estão contidos os ficheiros:

- de configuração (WerewolfsGame.mas2j e WerewolfsGameTest.mas2j)
- dos agentes (/src/jasonAgents/*)
- do ambiente (/env/*).

O código relativo ao teste dos vários agentes encontra-se na pasta /tester. Aí é criado um servidor http que cria várias instâncias do jogo comunicando com o ambiente de cada jogo (WerewolfsGameEnvTest.java é a classe que representa o ambiente cada jogo de teste).

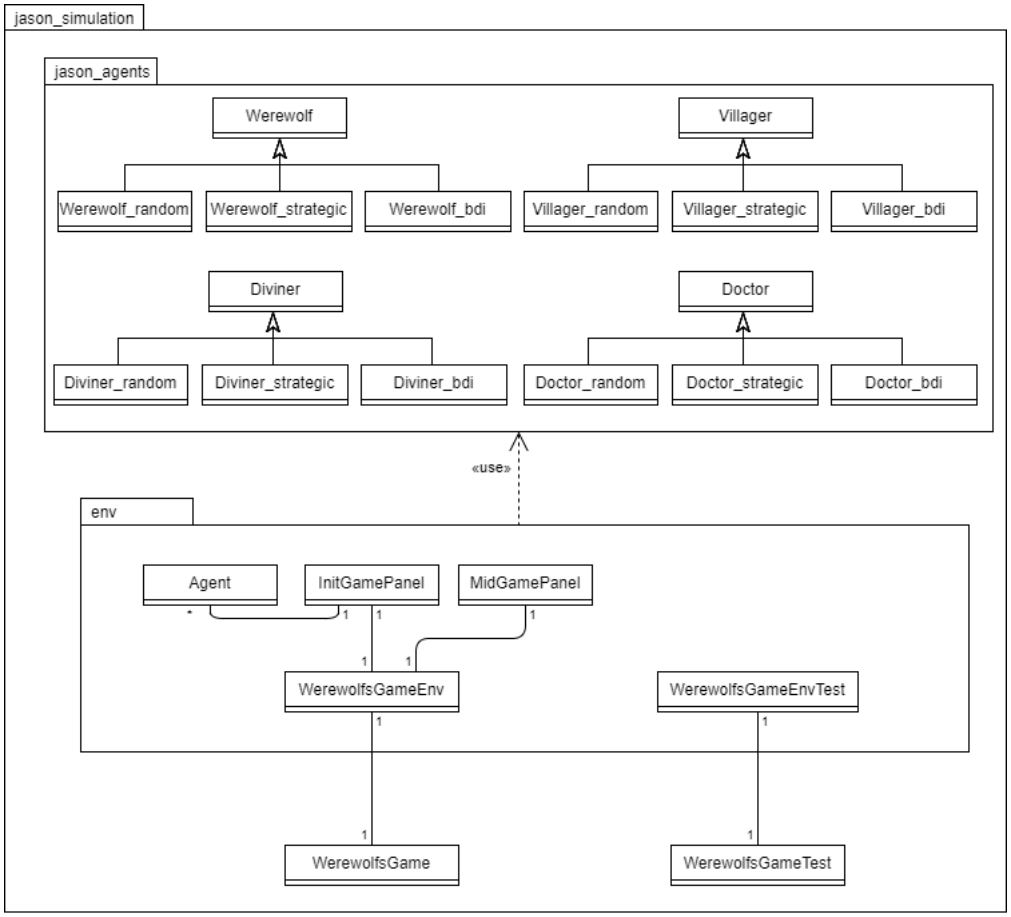

Figura 6: Diagrama de classes do package jason_simulation

3.3 Detalhes relevantes

3.3.1 Protocolo de Comunicação

Todos os agentes seguem um protocolo de comunicação semelhante, sendo que algumas das mensagens podem ser especificas para determinados papeis dos agentes.

O agente master é responsável pelo controlo do jogo, enviando por isso vários tipos mensagens:

- init(List) - mensagem que contem a lista dos agentes que representam personagens, enviada no inicio do jogo para que cada agente saiba quais são os outros jogadores.
- werewolf(WerewolfList) - mensagem que contem a lista dos agentes que representam lobisomens, enviada no inicio do jogo para que cada lobisomem saiba quais são os outros lobisomens.

- time(DayNight, VoteDiscussion) - mensagem enviada no inicio de cada fase de voto/discussão para indicar o inicio dessa fase.
- dead(Agent) - mensagem que indica que um agente está morto.
- role(Agent, Role) - mensagem que indica que um agente tem determinado papel. Esta mensagem é enviada, quando um agente morre ou quando um *diviner* tenta adivinhar o papel desse agente.

Cada agente que representa as personagens deve ser capaz de interpretar as mensagens referidas (com a exceção de werewolf(WerewolfList) que é apenas para os agentes com o papel de lobisomem). Alem disso devem ser capazes de enviar as seguintes mensagens:

- vote(Agent) - mensagem que representa um voto na fase de votação. Cada agente vivo deve enviar exatamente uma destas mensagens durante a votação diurna. O mesmo acontece á noite para os lobisomens.
- role(Agent, Role) - mensagem que representa uma acusação acerca de um papel. A mensagem é semelhante á enviada pelo master pelo que cada agente deve ser capaz de distinguir uma confirmação(feita pelo master) de uma acusação (feita por outros agentes).

A interpretação destas mensagens não é necessária, mas é recomendada pois é a única fonte de informação que cada os agentes tem um dos outros.

Para além destas mensagens, os *diviners* e os *doctors* devem ser capazes de enviar as seguintes mensagens ao master, respetivamente:

- divine(Agent) - mensagem que representa a intenção de divinar o papel de outro agente.
- cure(Agent) - mensagem que representa a intenção de curar um agente.

3.3.2 Confiança

Os agentes BDI partilham de modelo de confiança, utilizado para decidir como interpretar as acusações feitas pelos outros agentes. Cada agente que usa este modelo guarda uma crença trust(Agente,Certos,Errados,Confiança) acerca de cada um dos outros agentes (excluindo o master).

Sempre que o agente recebe uma acusação feita por outra agente, esta acusação é guardada para ser avaliada posteriormente, quando o agente master a confirmar ou desmentir. Nesse momento, o agente atualiza a sua crença trust, aumentando o numero de certos ou errados, e recalculando a confiança. A acusação em causa é descartada.

Algorithm 1: Modelo de Confiança

Data:
P - Agente em causa
X - Agente que faz uma acusação
Acusação - Acusação feita por X
Certos - Número de acusações feitas anteriormente por X que foram confirmadas pelo master
Errados - Número de acusações feitas anteriormente por X que foram refutadas pelo master
Confiança - valor da confiança que A tem em X

trust(X,Certos,Errados,Confiança) - crença que guarda a informação relativa á confiança em X
confirma(Acusação) - master confirma a Acusação
refuta(Acusação) - master refuta a Acusação

case *confirma(Acusação)* **do**
    Certos' = Certos + 1;
    Errados' = Errados;
    Confiança' = Certos'/(Certos'+Errados');
    atualiza trust(X,Certos,Errados,Confiança) para
     trust(X,Certos',Errados',Confiança') ;
end
case *refuta(Acusação)* **do**
    Certos' = Certos;
    Errados' = Errados + 1;
    Confiança' = Certos'/(Certos'+Errados');
    atualiza trust(X,Certos,Errados,Confiança) para
     trust(X,Certos',Errados',Confiança') ;
end

Inicialmente, os valores Errados e Certos são inicializados a 0, enquanto que a confiança é inicializada a 0.5, representando a crença que metade das vezes um agente acusa outro agente está a dizer a verdade.

No caso dos lobisomens, existe a pequena nuance que a confiança é inicializada a 1.0 relativamente aos outros lobisomens.

3.3.3 Suspeita

Além da confiança, os agentes BDI guardam também uma suspeita sobre cada um dos outros agentes (excluindo o master), referentes ao seu papel. Esta crença tem o formato suspect(role(X,Papel),Suspeita). Quando uma acusação é feita, o agente atualiza a suspeita baseando-se no valor de confiança relativo ao agente que faz acusação, tal como referido na secção anterior.

Algorithm 2: Modelo de Suspeita

Data:
P - Agente em causa
X - Agente que faz a acusação
Y - Agente que é acusado
PapelAcusado - Papel que X acusa Y de ser
PapelSuspeitado- Papel que P suspeita Y de ser

Confiança(X) - valor da confiança que P tem em X
Suspeita(Papel) - valor da suspeita de Y ser Papel

acusa(Y, PapelAcusado)[X] - X acusa Y de ser PapelAcusado
suspect(role(Y,PapelSuspeitado),Suspeita) - crença que guarda a informação relativa á suspeita em Y, sendo Suspeita equivalente a Suspeita(PapelSuspeitado)

case *acusa(Y, PapelAcusado)[X]* **do**
  if *PapelAcusado == PapelSuspeitado* **then**
    Suspeita' = Suspeita + (1-Suspeita) * Confiança(X);
    atualiza suspect(role(Y,PapelSuspeitado),Suspeita) para
     suspect(role(Y,PapelSuspeitado),Suspeita');
  else
    seja Suspeita(PapelAcusado) = Confiança(X);
    if *Suspeita(PapelAcusado) ¡= Suspeita(PapelSuspeitado)* **then**
      Suspeita' = Suspeita(PapelSuspeitado) -
       Suspeita(PapelAcusado);
      atualiza suspect(role(Y,PapelSuspeitado),Suspeita) para
       suspect(role(Y,PapelSuspeitado),Suspeita');
    else
      Suspeita' = Suspeita(PapelAcusado) -
       Suspeita(PapelSuspeitado);
      atualiza suspect(role(Y,PapelSuspeitado),Suspeita) para
       suspect(role(Y,PapelAcusado),Suspeita');
    end
  end
end

Através deste modelo, cada agente tem a si associado uma suspeita de que é um determinado papel, sendo esta suspeita representa por um valor que contido entre -1(certeza que não é o papel) e 1(certeza que é o papel).

4 Experiências

Decidiu-se realizar vários jogos com composições diferentes de jogadores para atestar, se como o imaginado, os agentes estratégicos e bdi teriam mais probabilidade de ganhar o jogo do que os aleatórios, e que, os agentes bdi teriam mais sucesso que os estratégicos. Com este objetivo, elaboramos um programa que lê as configurações e número de testes de um ficheiro de texto e retorna um documento *html* com os resultados formatados de forma legível ao utilizador. No ficheiro de configuração, colocamos seis testes com configurações diferentes de jogadores que correm 25 vezes cada para extrair a percentagem de vitórias de cada equipa, lobisomens e aldeões.

4.1 Teste 0 - Todos aleatórios

Este teste tem o papel de "grupo de controlo"da experiência. Como todos os agentes são aleatórios, nenhum tipo de raciocínio está inerente á sua votação. Assim sendo, resultados demonstrariam que, nestas condições, os agentes com maior conhecimento inicial (lobisomens) têm maior hipótese de ganhar.

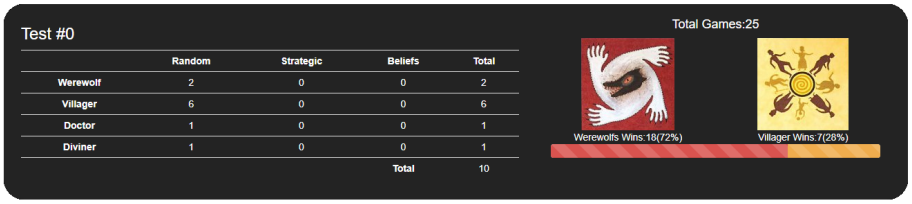

Figura 7: Resultados do teste 0

4.2 Teste 1 - Lobisomens aleatórios vs Aldeia estratégica

Neste teste, os lobisomens continuam completamente descoordenados, mas os agentes que representam os aldeões já conseguem tirar alguma informação das votações passadas, ganhando alguma inteligência. Deste modo, esperava-se que a percentagem de jogos em que os aldeões vencessem aumentasse significativamente.

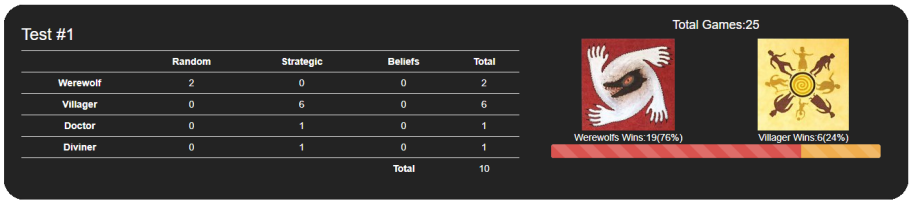

Figura 8: Resultados do teste 1

4.3 Teste 2 - Lobisomens aleatórios vs Aldeia BDI

Neste teste, os lobisomens ainda votam aleatoriamente, mas os agentes que representam os aldeões já conseguem tirar alguma informação das acusações partilhadas, tendo assim alguma inteligência. Deste modo, esperava-se que a percentagem de jogos em que os lobisomens perdesse aumentasse significativamente em relação ao teste 0 e ficasse parecida com a do teste 1.

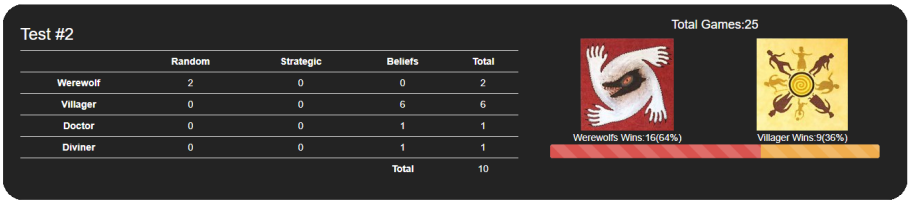

Figura 9: Resultados do teste 2

4.4 Teste 3 - Lobisomens estratégico vs Aldeia aleatória

Esta configuração contém aldeões completamente descoordenados e lobisomens que conseguem tirar alguma informação das votações passadas, ganhando, desta forma, alguma inteligência. Por estes motivos, seria esperado que a percentagem de vitórias dos lobisomens subisse em relação a todos os testes anteriores.

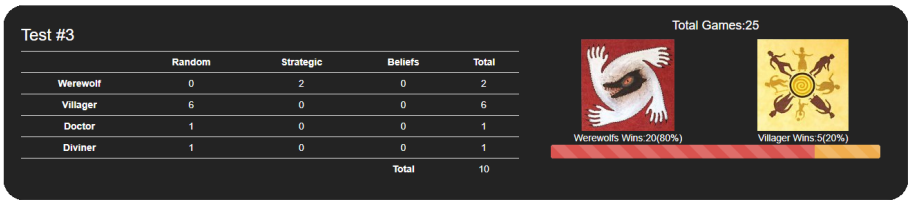

Figura 10: Resultados do teste 3

4.5 Teste 4 - Lobisomens BDI vs Aldeia aleatória

Nesta configuração existem aldeões ainda aleatórios e lobisomens que conseguem ganhar algum conhecimento a partir das acusações, utilizando algum raciocínio para votar. Consequentemente, seria esperado que a percentagem de vitórias dos lobisomens subisse em relação a todos os testes 0 a 2 e fosse semelhante à do teste 3.

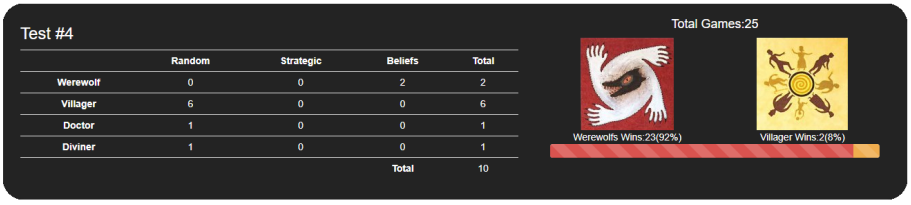

Figura 11: Resultados do teste 4

4.6 Teste 5 - Lobisomens estratégicos vs Aldeia BDI

Neste teste, ambas as equipas possuem algum tipo de inteligência. Os lobisomens retiram informação dos votos e os aldeões raciocinam com base nas acusações que ouvem. Assim sendo, esta configuração de jogadores permite tirar conclusões sobre qual tipo de jogador tem mais chances de vencer.

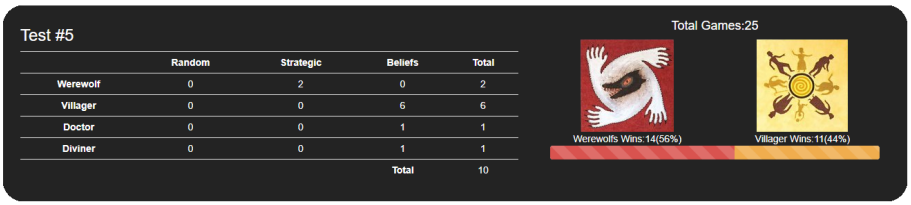

Figura 12: Resultados do teste 5

4.7 Teste 6 - Lobisomens BDI vs Aldeia estratégicos

Este teste tem a configuração inversa do anterior e tem como objetivo consolidar as conclusões retiradas do mesmo.

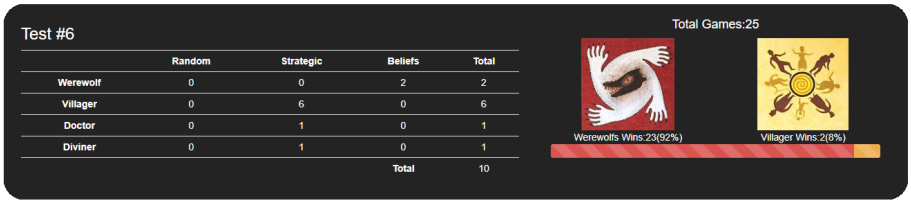

Figura 13: Resultados do teste 6

5 Conclusões

Através da análise dos testes podemos observar que, tal como esperado, os agentes BDI demonstram um performance superior aos agentes estratégicos e aleatórios.

Os agentes estratégicos, no entanto, demonstram uma performance inferior ao esperado, semelhante à dos agentes aleatórios, tal como se pode observar através da comparação dos testes 0 e 1. Apesar de estes resultados em parte poderem estar relacionados com a pequena amostra dos testes, atribuímos a causa principal à heurística utilizada.

De um modo mais geral, também chegou-se à conclusão de que este jogo é facilmente simulado com sistemas multi-agentes, mas, conseguir resultados semelhantes ao jogo entre humanos é mais complexo pois, especialmente nas rondas iniciais, existem decisões pouco racionais de voto.

6 Melhoramentos

As limitações de tempo, disponibilidade e recursos levaram a várias limitações no desenvolvimento do projeto, pelo que existem bastantes melhorias que poderiam ser implementadas:

- o protocolo de comunicação apenas aceita votos e acusações. Seria possível evoluir o protocolo de modo a permitir também a comunicação de suspeitas e mensagens privadas entre agentes.
- seria interessante a inclusão de outros papeis muitas vezes utilizados no jogo, como o vigilante.
- os agentes estratégicos demonstraram resultados abaixo do esperado pelo que uma alteração da heurística utilizada seria recomendado.
- os agentes BDI apenas guardam a suspeita de um agente ser um determinado papel, no entanto, seria possível permitir que estes agentes guardassem a suspeita de cada agente ser cada tipo de papel, permitindo assim suspeitar que um agente seja um de dois papeis, por exemplo.
- os agentes estratégicos são usam informação retirar dos votos da ronda anterior, enquanto que os agentes BDI apenas retiram informação retirada das acusações feitas por outros agentes. A criação de agentes que utilizassem ambas as fontes de informação em simultâneo seria uma experiência interessante.
- a inclusão de um agente que permitisse um utilizador humano jogar seria uma possibilidade.

7 Elementos do grupo

O trabalho foi bem dividido por todos os elementos do grupo. Dessa forma, a percentagem de trabalho efetivo de cada elemento do grupo corresponde a 33%.

8 Recursos

Referências

[1] "Werewolves of miller's hollow," 2016. Accessed in 30-10-2017.

[2] J. Marinheiro and H. L. Cardoso, "A generic agent architecture for cooperative multi-agent games.," in *ICAART (1)*, pp. 107–118, 2017.

[3] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*, vol. 8. John Wiley & Sons, 2007.

[4] "Getting started with jason," 2017. Accessed in 15-10-2017.

[5] "(bdi) hello world," 2017. Accessed in 15-10-2017.

A Manual do utilizador

Na pasta /jar_files encontram-se todos os programas compilados relevantes para correr uma simulação e executar uma pilha de testes.

Para correr uma simulação, ir para o diretório /jar_files e executar:

```
$ java -jar werewolfsGame.jar
```

Para correr uma pilha de testes, ir para o diretório /jar_files e executar:

```
$ java -jar Tester.jar test_werewolfs.txt werewolfsGameTest.jar
```

O primeiro argumento é o ficheiro de testes e o segundo é o programa compilado da simulação de jogo sem interface gráfica. Quando todos os testes tiverem terminado é possível analisá-los em detalhe abrindo o ficheiro statistics.html.

A.1 Screenshots

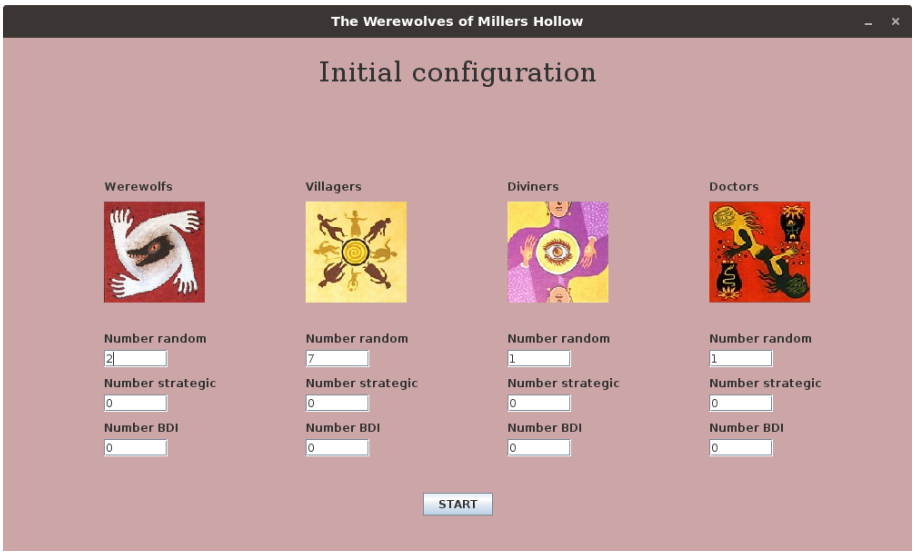

Figura 14: Configuração do jogo

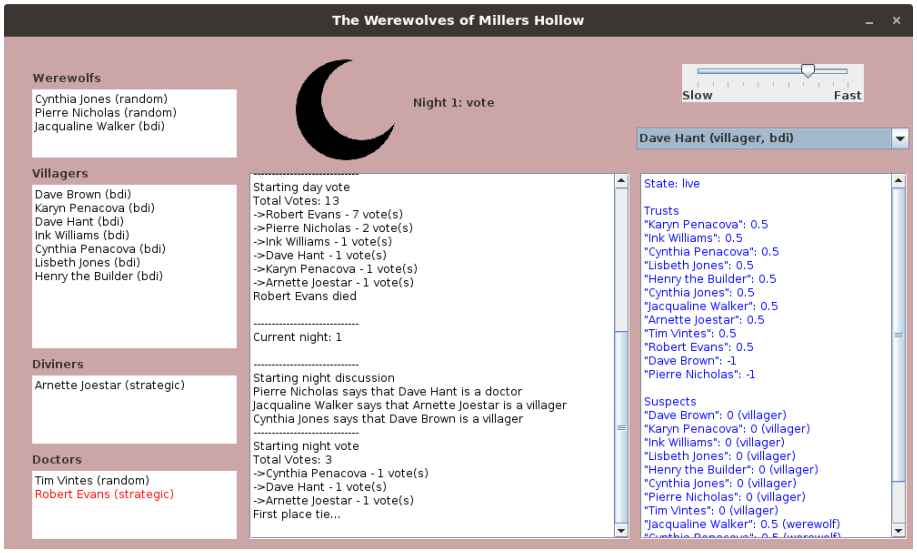

Figura 15: Simulação do jogo

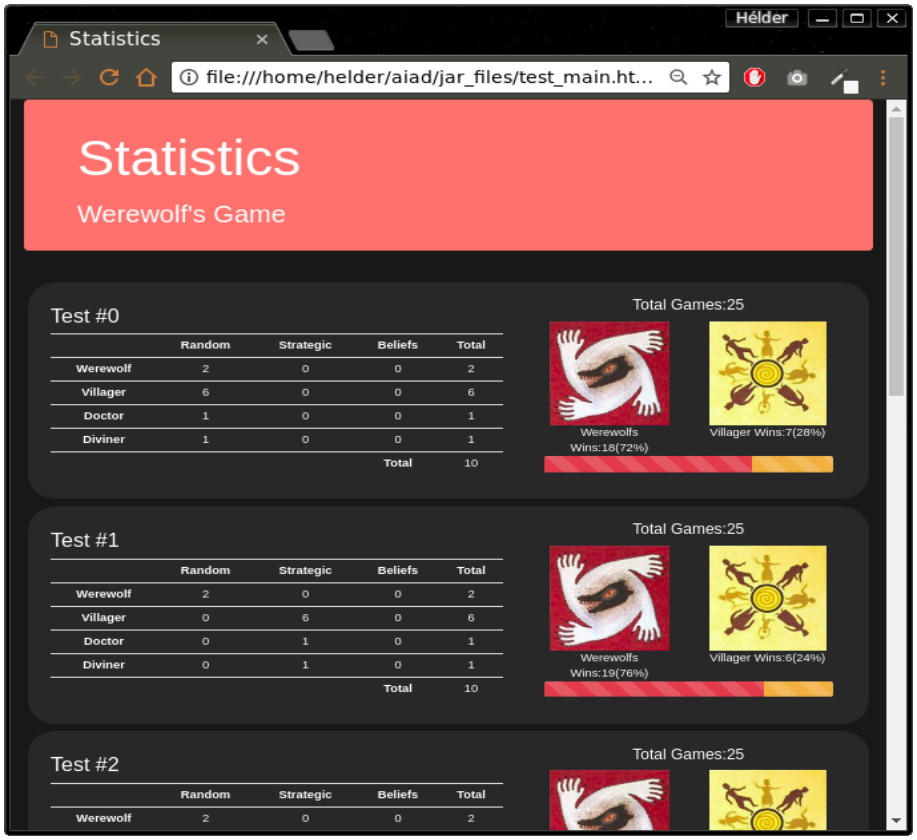

Figura 16: Resultados de uma pilha de testes