Implementación de Risk con algoritmo de Minimax.

Prieto, Estefanía[1]
Galicia, Fernando[1]
Galván, Antonio[1]
[1]Facultad de Ciencias, U.N.A.M.

estefaniaprieto@ciencias.unam.mx
fernandogamen@ciencias.unam.mx
g.antonio@ciencias.unam.mx

21-Noviembre-2014

Abstract

A diferencia de los juegos de apuesta, donde el jugador se pregunta *"¿Cuál es la mejor jugada para ganar un juego?"* y así poder ser el dueño de un premio (generalmente un incentivo monetario), es bien sabido en la teoría de juegos la motiva escenarios tales cómo el ajedrez, go, gato, etc. No existe tal pregunta, si no, ésta se replantea una expresión de la forma *"¿Existe una mejor forma de jugar en tal escenario?"*.
Por lo cuál se propone un modelo de Inteligencia Artificial para una versión acotada del juego **Risk** basado en minimax, con base en estrategias muy complicadas implementadas por un experto, hasta muy básicas diseadas por un novato en el juego.

1 Introducción.

Hacer acá la introducción de minimax.

2 Juegos con información perfecta. [MODIFICAR A INFORMARCIÓN IMPERFECTA Y ARGUMENTAR EL POR QUE PODEMOS ADAPTARLO ASÍ.]

Explicar por que catalogamos al **RISK** cómo un juego de información perfecta y por que hemos

elegido esta implementación.

3 Risk acotado.

Tal y cómo se plantea en el juego original (***veáse [2]***) el objetivo del juego continua siendo la dominación total de un territorio dado, de tal forma que el juego queda concluido cuándo todos los territorios quedan bajo la dominación de un jugador.
En esta implementación acotaremos la cantidad de continentes, es decir, el desarrollo sera unicamente en un solo continente, también la cantidad de dados se ve acotada a unicamente dos dados y restringido a dos jugadores.

Sin embargo mantendremos las demás condiciones iniciales con respecto a las tropas y al equivalente de tropas en cada territorio, es decir:

* Cada unidad representa una *Armada*.

* Cada *Caballería* representa 5 unidades.

* Cada *Artillería* representa 10 unidades.

Teniendo ya esto definido, entonces, cada jugador tendrá un ejercito inicial de 35 tropas.

4 Descripción del agente.

Aquí es donde describimos el comportamiento del agente.

5 Función de evaluación e implementación.

El tablero de juego estará representado por medio de una gráfica, en la cual, cada nodo será el representante de un país, éstos comparten frontera con aquellos nodos en los cuales existe una arista que los conecta.

Para introducir la función de evaluación primero observaremos el siguiente conjunto de definiciones:

Definición: Definimos a la función *"tropas"* denotadas como $t(i,j)$ de la siguiente manera:

$t(i,j)$:= *Número de tropas del i-ésimo jugador en el j-ésimo nodo*.

Definición: Definimos al conjunto de nodos pertenecientes a la gráfica y los denotamos cómo ν y al conjunto de nodos del i-ésimo jugador cómo ν_i

Y nos permitiremos sobrecargar la notación definiendo la función auxiliar $J(\nu)$ que tiene cómo objetivo indicar el jugador asociado al nodo ν.

Con estas observaciones nos permitimos definir ahora a la función de evaluación cómo sigue:

$$F_{eval} := \begin{cases} \infty \textit{ Max resulta ser ganador.} \\ -\infty \textit{ Min resulta ser ganador.} \\ ((\sum_{i=1}^{\nu} t(i,j)) \\ +c_1 + c_2 + \nu_i - \nu_j) \qquad \textit{E.O.C.} \end{cases}$$

Donde definimos a la función c_1 de la siguiente manera:

Pseudocódigo 1 Definición de la función c_1

Salida: Determinar la imagen de la función c_1

1: $cuenta = 0$
2: **para todo** $\nu' \in \nu_1$ **hacer**
3: **para todo** $u \in$ *Vecinos*(ν') **hacer**
4: **si** $u \in \nu_2$ **entonces**
5: $cuenta = cuenta + 1$
6: **termina si**
7: **termina para todo**
8: **si** $cuenta \geq 1$ **entonces**
9: -1
10: **si no**
11: 1
12: **termina si**
13: **termina para todo**
14: **devolver** $cuenta$

El algoritmo 1 muestra el comportamiento de de c_1 sea cual quiera que sea.

Continuamos por definir la función c_2 así de esta manera observemos que:

Pseudocódigo 2 Definición de la función c_2

Salida: Determinar la imagen de la función c_2

1: $cuenta = 0$
2: **para todo** $\nu \in$ *Vecinos*(ν) **hacer**
3: **si** $J(\nu) \neq \emptyset$ **entonces**
4: $J()$
5: **termina si**
6: **termina para todo**

Pseudocódigo 3 Genera los posibles movimientos del jugador

Salida: una gráfica de los posibles movimientos del jugador

1: **para todo** $\nu \in V_{jugador}$ **hacer**
2: **si** $\nu\ tieneVecinoVacio()$ **entonces**
3: $v.avanza(g.node)$
4: **devolver** g'
5: **si no**
6: $posiblesMovimientos(Jugador2, g)$
7: **termina si**
8: **termina para todo**

6 Especificaciones del programa.

La implementación concreta del proyecto se ha realizado en el lenguaje de programación ***Java*** [4] que se ha optado por que *Escribir ventajas de java y por que hemos optado por él*

Una vez aclarado esto, introduciremos la representación del territorio por medio de un archivo llamado *"Territorio.xml"* en el cual obtenemos las ventajas de que éste nos brinda una estructura la cual nos permite adaptar información de manera independiente al manejo de ésta [3]. Así el territorio del juego en el que cada país tendrá una etiqueta que lo represente y dentro de ésta estarán especificados los atributos de cada país.

De esta, hemos seccionado a los países en una pequea base de datos. Así que usaremos la interfaz *JAXP* [1]

7 Conclusiones.

Informar las conclusiones que hemos encontrado en nuestra implementación.

References

[1] Jaxp reference implementation. `https://jaxp.java.net/`. 07/Noviembre/2014.

[2] Parker Brothers. Risk the world conquerior game. `http://www.hasbro.com/common/instruct/risk.pdf`. 27/Octubre/2014.

[3] Borland Software Corporation. *XML Developer's Guide*. 1997.

[4] ORACLE. Descarga gratuita de java. `https://java.com/es/download/index.jsp`. 06/Noviembre/2014.