Universidad Carlos III de Madrid

Escuela politécnica superior

Ingeniería Informática

Proyecto Fin de Carrera

Estudio y aplicación de algoritmos de búsqueda al juego del Risk

Autor: Álvaro Torralba Arias de Reyna

Tutor: Carlos Linares López

Quiero dedicar este trabajo a aquellos que más felices

hace que lo haya terminado:

A mi madre.

A mi padre.

A mi abuela, Esperanza.

A mi novia, Elian.

Agradecimientos

Quiero agradecer el apoyo a todos aquellos que, directa o indirectamente, me han ayudado en este proyecto.

A toda mi familia por su apoyo. A mi madre, por escucharme siempre que lo necesito y dedicarme todo su esfuerzo. A mi padre, por estar ahí y darme siempre buenos consejos. También a mi abuela, Esperanza, por cuidarme siempre con todo su cariño y preocuparse tanto por mí.

A mi novia, Elian, por estar siempre a mi lado y apoyarme cuando lo he necesitado.

A mi tutor Carlos Linares López, por ayudarme a mejorar la calidad del trabajo con sus consejos y correcciones.

A Iñaki Reta Sabarrós por enseñarme LaTeXy resolver siempre todas mis dudas.

A José Tomás Nogales Nieves por sus consejos sobre los casos de uso.

A todos mis amigos por su apoyo en los tiempos de descanso.

A todos mis compañeros de prácticas por todo el trabajo que hemos compartido a lo largo de la carrera, en especial a Pablo Lázaro Grande Benito.

A todos los que han estado a mi lado compartiendo el lugar de trabajo mientras desarrollaba el proyecto.

A todos los profesores que he tenido, desde mi infancia hasta hoy, por sus enseñanzas, sin las cuáles no podría haber realizado este proyecto.

Muchas gracias a todos.

Álvaro Torralba Arias de Reyna

Madrid, 1 de Octubre de 2009.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Area de aplicación

La Inteligencia Artificial (IA), es un área de la informática que intenta *emular* comportamientos inteligentes en sistemas informáticos. Con el objetivo de avanzar en esta línea es común realizar sistemas que resuelvan juegos.

Los juegos son interesantes porque tienen un *dominio acotado* es decir, sus reglas están bien definidas y delimitan con total exactitud qué se puede hacer y qué no. Se ejecutan en un *entorno controlado* por lo que no pueden ocurrir eventos imprevistos durante una partida. Permiten una *evaluación objetiva* del rendimiento del sistema, midiendo el porcentaje de victorias/derrotas.

Un ejemplo bastante clásico en este sentido son los juegos de ajedrez que se han desarrollado durante varias décadas y, a día de hoy, ya se puede considerar que han alcanzado el nivel de un gran maestro, por lo que en los últimos años se ha avanzado un nivel más, afrontando juegos de mayor dificultad.

Este proyecto se sitúa en este marco al pretender desarrollar una IA para el juego del Risk. Este juego resulta muy interesante por las siguientes características:

- Aleatoriedad en las acciones: el resultado de algunas acciones de los jugadores es no determinista, lo que dificulta el cálculo del estado resultante.

- Varios jugadores: pueden jugar de 2 a 6 jugadores por lo que hay que tener en cuenta las estrategias de cada uno de ellos y hace que dañar a otro jugador pueda no ser bueno (puede beneficiar más a un tercer jugador).
- Factor de ramificación muy alto: En cada turno el jugador debe decidir las acciones a tomar y, dado que puede realizar cualquier número de ataques que desee, el número de opciones resulta inmanejable.

Estas características unidas hacen inviables las técnicas de búsqueda utilizadas en el ajedrez ya que, el factor de ramificación es demasiado extenso y las situaciones con más de dos jugadores dificultan la poda del árbol de búsqueda.

1.2. Estructura del documento

Este documento se estructura en varios capítulos, a lo largo de los cuáles se explicará el desarrollo del proyecto y las pruebas y resultados analizados.

Para comenzar se realiza un análisis del *estado de la cuestión*, describiendo detalladamente el dominio de estudio del trabajo (el juego Risk) y los trabajos realizados anteriormente a este proyecto.

A continuación se describen los *objetivos* del proyecto que se desea realizar, tanto desde el punto de vista de la IA como del desarrollo software.

Después se explica el *trabajo realizado*, teniendo en cuenta el desarrollo de los algoritmos utilizados y todas las fases del desarrollo software: análisis, diseño e implementación.

Posteriormente se realizan diversas pruebas y análisis para medir los *resultados* del sistema implementado de la forma más objetiva posible.

Finalmente se incluyen las *conclusiones* obtenidas del agente y sus resultados

Por último se dedica un capítulo a comentar las *líneas futuras* que podrían seguirse para continuar esta línea de trabajo y mejorar los resultados.

Capítulo 2

Estado de la cuestión

Este capítulo se centra en describir el dominio del Risk y los estudios y aplicaciones realizadas anteriormente.

2.1. Descripción

En primer lugar se explica en qué consiste el juego del Risk y las reglas concretas que se utilizarán entre las múltiples versiones existentes.

2.1.1. Introducción al juego del Risk

El Risk es un juego de mesa de dos a seis jugadores que competirán por la conquista del mundo. Consta de un tablero que representa al mundo subdividido en países y los jugadores representan a potencias mundiales que controlan los países mediante fuerzas militares.

Cada partida se organiza en turnos, en los que el jugador al que le toca actuar refuerza sus países añadiendo tropas y ataca a otros países para intentar conquistarlos. El juego termina cuando un jugador domina el mundo, es decir, posee todos los países.

Fue creado en 1950 por Albert Lamorisse como "La Conquette Du Monde" y gracias al éxito obtenido fueron adquiridos los derechos por la empresa estadounidense Parker

Brothers quienes cambiaron su nombre a "Risk". Posteriormente se crearon multitud de versiones que modificaban tanto las reglas como el mapa de juego:

1. Castle Risk (1986)
2. Risk: Édition Napoléon (1999)
3. Risk: 2210 A.D. (2001)
4. Risk: the Lord of the Rings (2002)
5. Risk: the Lord of the Rings: Trilogy Edition (2003)
6. Risk Godstorm (2004)
7. Risk: Star Wars: Clone Wars Edition (2005)
8. Risk: Star Wars Original Trilogy Edition (2006)
9. Risk Junior: Narnia (2006)
10. Risk: The Transformers Edition (2007)
11. Risk: Black OPS (2008)

Actualmente los derechos están en posesión de Hasbro [1].

2.1.2. Reglas del juego

Dado que hay varias versiones de las reglas del juego, se describen a continuación las utilizadas en este proyecto.

A la hora de seleccionar las reglas, se ha tenido en cuenta:

1. Adaptabilidad a diferentes mapas: el mapa de juego no tendrá por qué ser el clásico mapa del mundo sino que podrá ser cualquier mapa que cumpla con las condiciones establecidas en las reglas.
2. Las cartas no deben ser el elemento más determinante de la partida, sino que debe valer más la ventaja estratégica.

[1] Risk en Hasbro: http://www.hasbro.com/risk/

Elementos del juego

Los elementos con los que interactúan los jugadores durante la partida son:

Tablero El tablero de juego representa un mapa del mundo a conquistar y se mantiene durante toda la partida. Se compone de países o territorios y continentes.

Países Los países son las casillas del tablero. Existen conexiones entre éllos, de forma que algunos países son vecinos o adyacentes entre sí. El conjunto de todos los países del tablero forma un grafo no dirigido conexo, de modo que siempre hay un camino entre dos países cualesquiera.

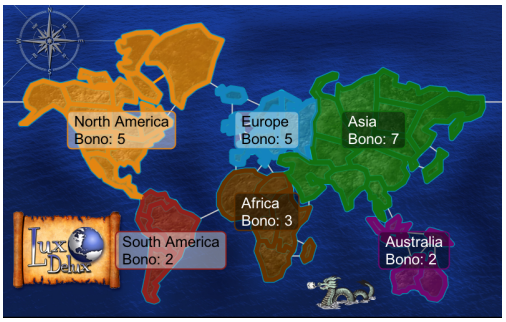
Además, cada país está en posesión de un único jugador y aquel jugador que posea todos los países del tablero ganará la partida. Para poseer un país, un jugador debe tener soldados suyos en él.

Continentes Un continente es un conjunto de países que se puede definir como un *subgrafo conexo del tablero de juego*. Es decir, todos los países pertenecientes al mismo continente tienen un camino entre sí sin pasar por territorios no pertenecientes al continente. Cada país pertenece a un único continente.

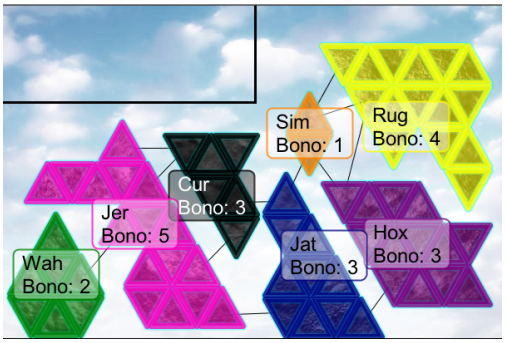
Todo continente tiene una bonificación (número de soldados) de modo que si un jugador posee todos los países de ese continente al inicio de su turno recibirá esa cantidad de soldados extra. Esta bonificación puede ser cualquier número entero, por lo que se permiten el cero y valores negativos.

En la figura 2.1 se muestran algunos ejemplos de tableros.

Soldados Los soldados están situados en los países y son las unidades controladas por los jugadores. En cada país debe haber al menos un soldado. Los soldados pueden moverse/atacar a territorios adyacentes al que están.



(a) Clásico



(b) Hexinfinity

(c) USA war zone

(d) Siege Lux

Figura 2.1: Ejemplo de tableros posibles

Cartas Las cartas sirven para premiar a los jugadores que realizan ataques durante su turno, para evitar situaciones de bloqueo en las que ningún jugador ataca.

Existen dos tipos de cartas, las cartas normales y los comodines. Las cartas normales se componen de una figura o tipo de carta (soldado, caballo o cañón) y de un país, habiendo una carta de este tipo por cada país. Los comodines pueden valer como una carta de cualquier tipo y no tienen un país asociado. En total, hay dos comodines en la baraja.

Fases del turno

El turno de un jugador se divide en varias fases, como muestra la figura 2.2.

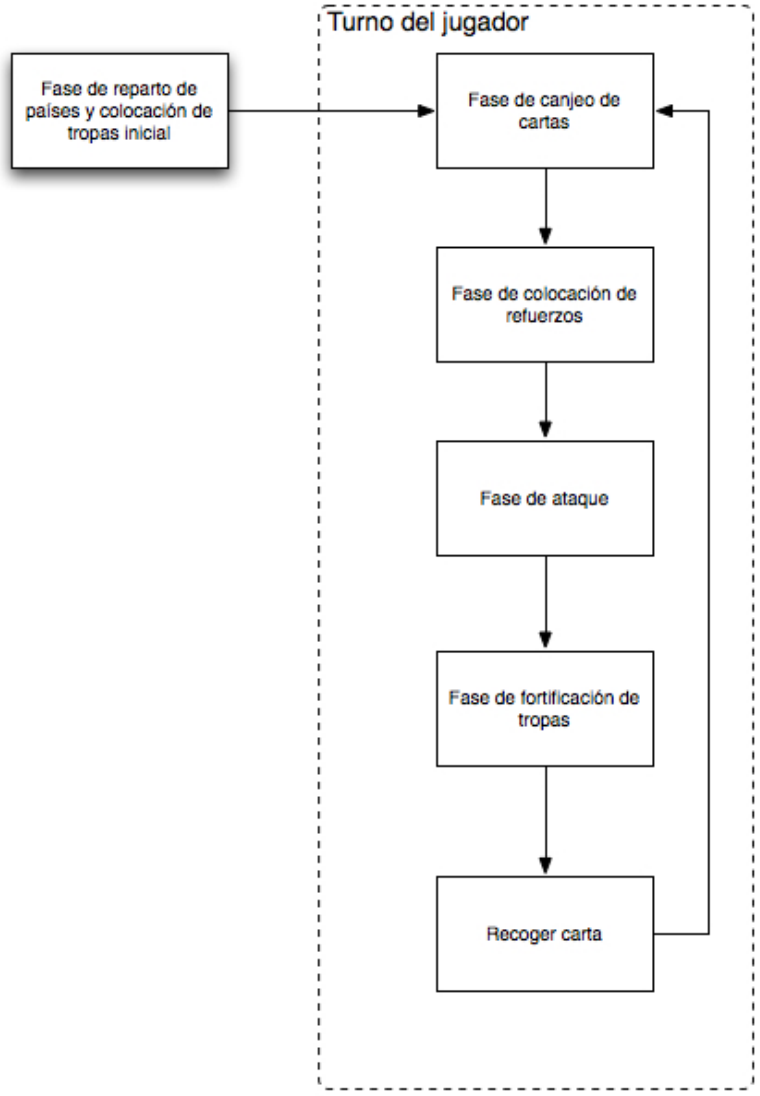


Figura 2.2: Esquema de las fases de una partida del Risk

Reparto de países y colocación de tropas

Al comienzo de la partida lo primero es repartir los países entre todos los jugadores de la partida. En principio, todos los jugadores reciben el mismo número de países pero si el número de países no es divisible entre el número de jugadores, algunos jugadores reciben un país más.

A continuación, se reparten los soldados entre los países. El número de soldados que

Número de jugadores	Soldados para cada jugador
2	40
3	35
4	30
5	25
6	20

Tabla 2.1: Número de soldados iniciales para los jugadores

tiene inicialmente cada jugador es el que viene en la tabla 2.1. Si el número es inferior al número de países que corresponde al jugador, tendrá un soldado por país. En caso contrario, los soldados se reparten uniformemente entre los países.

Fase de canjeo de cartas

En la fase de canjeo de cartas el jugador puede descartar tres cartas que posea para obtener más soldados. No todas las combinaciones de cartas pueden canjearse, sino que, para poder hacer este cambio, es necesario que las cartas estén emparejadas. Tres cartas están emparejadas si son todas del mismo tipo (tres cartas de soldado, de caballo o de cañón) o todas de tipos distintos (una carta de soldado otra de caballo y otra de cañón). Un comodín puede valer por cualquier tipo de carta.

El número base de soldados que recibe un jugador cuando canjea cartas es siempre de 5. Estos soldados se situarán en el país que escoja el jugador, junto con el resto de soldados recibidos al inicio del turno. Además, si el jugador posee alguno de los países presentes en sus cartas, pondrá 2 soldados extra en ese país. Los comodines, al no estar asociados a ningún país, no pueden aportar nunca soldados extra de este modo.

En caso de que el jugador tenga cinco cartas o más está obligado a canjearlas hasta tener menos de cinco. En caso de que tenga tres o cuatro cartas podrá elegir canjearlas o no.

Fase de colocación de refuerzos

En la fase de colocación de refuerzos el jugador sitúa las tropas recibidas en sus países. El número de tropas que recibe un jugador es la suma de tres factores:

1. $max(\lfloor \frac{NumPaisesJugador}{3} \rfloor, 3)$
2. Refuerzo por dominar continentes (depende del mapa de juego concreto).
3. Canjeo de cartas.

Fase de ataque

En la fase de ataque el jugador decide los ataques a realizar. Por cada ataque que decida se siguen los siguientes pasos:

1. Selección del país atacante (con al menos dos soldados).
2. Selección del país defensor (debe ser vecino del país atacante).
3. Decidir el resultado del ataque (aleatoriamente).

El número de atacantes es $S_A = min(3, Ataq - 1)$ y el número de defensores es $S_D = min(2, Def)$, siendo $Ataq$ y Def el número de soldados en el país atacante y defensor respectivamente.

A continuación, cada jugador tira tantos dados como soldados suyos participan en el ataque. Se recogen los dados más altos de cada jugador y se emparejan entre sí, de modo que se obtienen una o dos parejas de dados. Por cada pareja de dados morirá un soldado del jugador que obtenga menor puntuación en su dado y del atacante en caso de empate.

Si el soldado defensor pierde todos los soldados del país atacado entonces ese país pasa a ser del jugador atacante. El jugador atacante escoge cuántos soldados pasar al país conquistado, en el intervalo $[S_A, Ataq - 1]$.

Si un jugador es eliminado, todas sus cartas pasan al jugador atacante. En este caso, si el número de cartas del jugador atacante es mayor o igual a cinco, deberá canjear sus cartas del mismo modo que en la fase de canjeo de cartas.

Fase de fortificación de tropas

Una vez realizados los ataques el jugador puede realizar movimientos de tropas entre países adyacentes, es decir, puede pasar todos los soldados que quiera de cada país a alguno de sus vecinos.

Sin embargo, ningún país puede quedarse sin soldados por lo que para que el último soldado pueda pasar a un país vecino, éste deberá recibir al menos un soldado mediante una fortificación.

Recoger carta

Si el jugador conquistó al menos un país durante este turno entonces recoge una carta aleatoriamente.

2.2. Recopilación del conocimiento de expertos

El Risk es bastante popular y existen muchos aficionados que gracias a su experiencia con el juego han mejorado su estrategia. Es importante recoger este conocimiento ya que puede resultar de utilidad para diseñar el agente.

Este conocimiento no tiene porqué ser utilizado de forma directa en el proceso de diseño del agente, pero debe tenerse en cuenta para que los algoritmos que incorpore el agente den como resultado comportamientos compatibles con las reglas obtenidas.

Una fuente muy interesante de conocimiento de expertos es la página web "total diplomacy" [2] donde hay algunos consejos de expertos que marcan las claves del juego.

Factores importantes a la hora de escoger un plan en un turno son:

[2] Total diplomacy: http://www.totaldiplomacy.com/

1. Siempre que exista la posibilidad de eliminar por completo a otro jugador, hay que tenerlo en cuenta para obtener sus cartas. Sin embargo, en caso de que no haya altas probabilidades de eliminarlo por completo, no es bueno debilitarlo demasiado dejando que otros jugadores lo eliminen y obtengan sus cartas.

2. Cuando eres muy débil es conveniente dificultar la eliminación completa, por ejemplo, concentrando todas tus fuerzas en un país.

3. Intenta asegurarte poder atacar a cualquier jugador para matarlo con el menor coste si se debilita demasiado.

4. Intenta minimizar el número de fronteras con otros jugadores. Escoge la mejor dirección para expandirte y mira como puede beneficiar los movimientos de tus tropas.

5. Intenta adivinar los planes de los rivales para evitar sus planes o que no te ataquen.

Estrategias que se deben combinar a lo largo de una partida:

1. Expandirse: Para dominar algún continente y aumentar el número de armadas que se obtienen en un turno.

2. Escoger continentes: Es importante escoger correctamente el/los continente/s a dominar. En partidas con pocos jugadores, serán preferibles continentes grandes que den más tropas, y en partidas con más jugadores será más sencillo asegurar continentes pequeños aunque den menos tropas por turno. También hay que tener en cuenta los continentes que atacarán los oponentes.

3. Asegurar territorios: Una vez que ya se dispone de un buen territorio (un continente conquistado, por ejemplo) es importante defenderlo correctamente en las fronteras para que nadie se atreva a atacarlo.

4. Quitar el bonus a los rivales: Atacando un país de los continentes que dominan o conquistando muchos países.

5. Eliminar jugadores débiles: Para arrebatarles sus cartas. Es importante encontrar el punto exacto en el cuál merecen la pena las pérdidas para obtener las cartas.

2.3. Aplicaciones existentes del juego del Risk

Se han encontrado diferentes juegos de ordenador que permiten jugar partidas de Risk en modo local o a través de Internet.

2.3.1. Lux Delux

El juego Lux Delux [3] es una aplicación implementada en lenguaje Java. Permite enfrentarse a jugadores a través de Internet y también a agentes de forma local. Permite la adición de agentes en lenguaje Java.

Sigue activa en Septiembre de 2009 y se realizan actualizaciones periódicamente.

2.3.2. Dominate Game

El juego Dominate Game [4] está en una página Web. Permite enfrentarse a través de Internet a otros usuarios y a agentes implementados por otros usuarios que pueden ser publicados en la página web. Permite la adición de agentes en lenguaje Tcl.

Sigue activa en Septiembre de 2009 y se realizan actualizaciones periódicamente.

2.3.3. Ksirk

Ksirk [5] es otra aplicación que permite jugar al Risk a varios jugadores a través de Internet. Incorpora una inteligencia artificial básica, aunque no da soporte para la inclusión de nuevos agentes de forma sencilla.

La última actualización fue en mayo de 2008.

[3] Lux Delux: http://sillysoft.net/lux/
[4] Dominate Game: http://www.dominategame.com
[5] Ksirk: http://gna.org/projects/ksirk

2.4. Estudios teóricos realizados anteriormente

Estos trabajos aportan información teórica acerca del dominio del Risk, aunque no hayan culminado con una implementación práctica. Hay varios tipos:

- Cuestiones matemáticas: cálculos probabilísticos útiles.
- Valoración de la posición: información referente a cómo valorar una posición.
- Algoritmos de búsqueda: aplicación de algoritmos de búsqueda en el Risk.
- Conocimiento de expertos: conocimientos generales de jugadores.

2.4.1. Cuestiones matemáticas

Como algunas acciones de los jugadores tienen consecuencias aleatorias, deben contemplarse las diversas posibilidades y su probabilidad. Los estudios recogidos aquí analizan los cálculos que pueden resolver este problema.

Resultado de un ataque

El jugador atacante puede realizar un ataque simple con $[1, 3]$ soldados y el defensor deberá defenderse con $[1, 2]$ soldados.

Por lo tanto, se define un ataque como un par $[S_A, S_D]$, siendo S_A el número de atacantes y S_D el número de defensores, por lo que hay seis casos para los que resulta necesario calcular la probabilidad de sus distintos resultados posibles.

Como para cada soldado se lanza un dado, el número de posibles resultados es $6^{S_A+S_D}$. Para calcular la probabilidad de victoria, se enumeran todos los casos posibles. Así, en el caso [1,1] el número de posibilidades es de 36, en 15 de las cuales ganará el atacante y en 21 el defensor, por lo que las probabilidades de cada suceso son $\frac{15}{36} = 41,67\,\%$ y $\frac{21}{36} = 58,33\,\%$, respectivamente.

	$S_D = 2$		$S_D = 1$	
	Soldados perdidos	Probabilidad	Soldados perdidos	Probabilidad
$S_A = 3$	Atq -2	$\frac{2275}{7776} = 29,26\,\%$	Atq -1	$\frac{441}{1296} = 34,03\,\%$
	Def -2	$\frac{2890}{7776} = 37,17\,\%$	Def -1	$\frac{855}{1296} = 65,97\,\%$
	Ambos -1	$\frac{2611}{7776} = 33,58\,\%$		
$S_A = 2$	Atq -2	$\frac{581}{1296} = 44,83\,\%$	Atq -1	$\frac{91}{216} = 42,13\,\%$
	Def -2	$\frac{295}{1296} = 22,76\,\%$	Def -1	$\frac{125}{216} = 57,87\,\%$
	Ambos -1	$\frac{420}{1296} = 32,41\,\%$		
$S_A = 1$	Atq -1	$\frac{161}{216} = 74,54\,\%$	Atq -1	$\frac{21}{36} = 58,33\,\%$
	Atq -1	$\frac{55}{216} = 25,46\,\%$	Def -1	$\frac{15}{36} = 41,67\,\%$

Tabla 2.2: Probabilidad de los posibles resultados de un ataque simple

En la tabla 2.2 se muestran los posibles resultados de un ataque simple para cada uno de los casos mencionados, calculados mediante un recuento de casos similar al mencionado para el caso [1,1].

Resultado del ataque continuado a un país

Para calcular el resultado final cuando un jugador ataca un país repetidas veces hasta conquistarlo o quedarse sin soldados suficientes para atacarlo, puede utilizarse la ecuación 2.1, extraída de la definición de probabilidad condicionada como se indica en el trabajo de Vaccaro [VG04].

$$P(A \cap B) = P(A)P(B|A) \tag{2.1}$$

Básicamente, plantea calcular todas las posibilidades tras la realización de todos los ataques. La probabilidad de cada uno de los casos finales es igual a la probabilidad de los casos previos $P(A)$ por la probabilidad de que ocurra el estado final a partir de los casos previos $P(B|A)$.

Además, se pueden almacenar las funciones de densidad calculadas para acelerar el

proceso de cálculo, reutilizando los cálculos intermedios.

Cálculo simplificado del resultado de un ataque continuado

En la FAQ de Owen Lyne[6], aunque no se explica cómo se obtuvo, se muestra la ecuación 2.2 que calcula el número medio de atacantes que se perderán en el ataque para conquistar un país con D soldados defensores, suponiendo que el atacante tiene suficientes soldados para lanzar siempre tres dados.

$$A = 0{,}8534144D - 0{,}2213413(1 - (-0{,}525359)^D) \tag{2.2}$$

Owen Lyne extrae dos conclusiones de esta ecuación:

- El incremento de soldados en países con menos soldados provoca más pérdidas en sus enemigos.
- Es ligeramente mejor repartir los soldados en números pares que en impares.

Probabilidad del canjeo de cartas

Owen Lyne establece en su FAQ las probabilidades que hay en el juego clásico de obtener un conjunto de cartas que sea canjeable.

Los factores a tener en cuenta para calcular la probabilidad de canjear cartas son:

- Las cartas que ya tenemos no pueden volver a salir.
- Si alguien no ha cambiado cartas podemos suponer que tiene más probabilidades de que esas tres cartas no encajen.

Calculando la probabilidad de que se tenga cada posible conjunto de cartas se obtienen las probabilidades de la tabla 2.3.

[6]FAQ sobre Risk de Owen Lyne: http://www.kent.ac.uk/IMS/personal/odl/riskfaq.htm

Situación actual	Situación del canjeo	Probabilidad
Tenemos 0 cartas	Cuando tengamos 3 cartas	$42,28\,\%$
Tenemos 0 cartas	Cuando tengamos 4 cartas	$81,70\,\%$
Tenemos 3 cartas no canjeables	Cuando tengamos 4 cartas	$68,29\,\%$
Tenemos 1 comodín	Enemigo canjea con 3 cartas	$38,06\,\%$
Tenemos 1 comodín	Enemigo canjea con 4 cartas	$79,87\,\%$
Tenemos 2 comodín (todos)	Enemigo canjea con 3 cartas	$33,41\,\%$
Tenemos 2 comodín (todos)	Enemigo canjea con 4 cartas	$77,80\,\%$

Tabla 2.3: Probabilidad de poder canjear cartas en cada caso

2.4.2. Valoración de la posición

La valoración de una posición es una tarea crítica en la mayoría de implementaciones de una Inteligencia Artificial para cualquier dominio.

Hay muchos aspectos que pueden valorarse dentro de una posición. El principal es la valoración de los jugadores, que sirve para determinar qué posición es más favorable para el jugador. También puede obtenerse, y resulta de gran interés en este proyecto, la valoración de un país, es decir la ventaja que ofrece a un jugador controlar dicho país.

Valoración de los jugadores

La valoración de los jugadores consiste en, dada una determinada posición, asignar un valor numérico a cada jugador que mida lo fuerte que es: una aproximación a sus probabilidades de ganar la partida.

La única fuente encontrada al respecto es el trabajo de Vaccaro acerca de la aplicación de algoritmos de búsqueda en el dominio del Risk [VG05].

La valoración consiste en una media ponderada cuyos factores aparecen en la tabla 2.4.

Objetivo	Cálculo	Peso
Maximizar soldados	$\frac{SoldadosJugador}{SoldadosTotales}$	0.45
Maximizar refuerzos esperados	$\frac{RefuerzosJugador}{RefuerzosTotales}$	0.33
Minimizar fronteras defensivas	$1 - \frac{\frac{FronterasJugador}{PaisesJugador}}{\frac{FronterasTotales}{PaisesTotales}}$	0.02
Maximizar fuerza de fronteras	$\frac{SoldadosJugadorFrontera}{SoldadosJugador}$	0.05
Maximizar soporte logístico	$\frac{\frac{SoldadosJugadorFrontera}{SoldadosJugadorApoyo}}{\frac{SoldadosTotalesFrontera}{SoldadosTotalesApoyo}}$	0.15

Tabla 2.4: Factores para la valoración de los jugadores

Valoración de países

La valoración de los países para un jugador puede utilizarse tanto para saber qué país es más interesante conquistar o defender como para predecir qué países atacarán o defenderán los jugadores rivales.

Tal como se plantea en el trabajo de Fredrik Olsson [Ols05], no todos los países valen lo mismo para un jugador, sino que algunos son mejores que otros. Además, este valor cambia a lo largo de la partida.

En primer lugar, la ecuación 2.3 define la valoración del continente V_{sv}, basada en su valor de bonus de soldados (V_b), su tamaño en número de países (V_s) y su número de fronteras(V_{nb}).

$$V_{sv} = \frac{V_b}{V_s V_{nb}} \tag{2.3}$$

La valoración particular para un país depende de múltiples factores, cada uno de ellos ponderado por un peso determinado. La ecuación 2.4 determina el valor del país V_p a partir de los factores de la tabla 2.5.

$$V_p = V_{sv} + V_{fn} + V_{fnu} + V_{en} + V_{enu} + V_{cb} + V_b(V_{cp} + V_{oc} + V_{eoc}) \tag{2.4}$$

Abreviatura	Factor	Peso
V_{sv}	Valor del continente definido en la ecuación 2.3.	70
V_{fn}	Número de países vecinos propios.	1.2
V_{fnu}	Número de soldados en países vecinos propios.	0.05
V_{en}	Número de países vecinos enemigos.	-0.3
V_{enu}	Número de soldados en países vecinos enemigos.	-0.03
V_{cb}	Número de continentes con que el país es frontera.	0.5
V_b	Bonus del continente.	1
V_{cp}	Cuanto dominio se tiene sobre el continente.	1
V_{oc}	Toma valor 1 si se posee todo el continente excepto este país y 0 en caso contrario.	20
V_{eoc}	Toma valor 1 si un enemigo controla todo el continente del país y 0 en caso contrario.	4

Tabla 2.5: Factores para la valoración de los jugadores

2.4.3. Estudio sobre la aplicación de algoritmos de búsqueda

Una posible aproximación es realizar una búsqueda del mejor plan, considerando un plan como el conjunto de ataques a realizar durante un turno.

El número de planes posibles crece exponencialmente respecto al número de países y soldados que hay en la partida, por lo que es imposible utilizar fuerza bruta para revisar todas las posibilidades.

En este sentido, Vaccaro, J.M. y Guest, C.C. realizaron un análisis de diversos algoritmos aplicados a la búsqueda de soluciones en los movimientos finales del Risk [VG05].

En este análisis se realizaron pruebas con diversos algoritmos bajo las mismas condiciones: la misma función de evaluación de planes y el mismo tiempo utilizado para la búsqueda.

Un plan es considerado como una lista de acciones donde la primera acción es el posicionamiento inicial de tropas y luego hay acciones de ataque y acciones de traslado de soldados.

Los casos de prueba eran situaciones generadas de forma aleatoria de finales de partida con 8 jugadores. Los finales de partida son considerados de tal forma que un jugador tiene el 60 % de tropas finales del tablero y los siete jugadores restantes se reparten el otro 40 %. Las cartas tienen un valor muy elevado, por lo que es importante eliminar jugadores para quitarles sus cartas antes de que puedan canjearlas. De este modo, el jugador aventajado tiene que intentar buscar un plan en el que pueda eliminar al resto de jugadores maximizando sus probabilidades de éxito.

En todos los casos (excepto en el aleatorio), la búsqueda se guía por la valoración de los planes evaluados, sin utilizar ninguna heurística alternativa, por lo que se buscarán planes entre aquellos que han dado mejores resultados.

En concreto, se probaron siete algoritmos diferentes:

- Aleatorio
- Primero en profundidad
- Primero en amplitud
- Escalada
- Mejor primero
- Recocido simulado
- Computación evolutiva

En el 85 % de los casos el algoritmo que mejores resultados logró fue la computación evolutiva, mientras que el segundo mejor era el recocido simulado. El resto de algoritmos obtuvieron resultados muy inferiores.

2.5. Soluciones realizadas anteriormente

Aquí se estudian las implementaciones de agentes de Risk encontradas para analizar las técnicas aplicadas y los resultados que han obtenido.

2.5.1. Sistemas expertos basados en reglas

Los sistemas expertos trabajan sobre un modelo del conocimiento de un experto en la materia. Existen bastantes ejemplos de sistemas expertos que juegan al Risk y la mayoría de ellos utilizan modelos basados en reglas sencillas y estrategias preconcebidas. Un ejemplo claro de este tipo de técnicas aplicadas al Risk son los agentes incorporados en el juego Lux Delux [7].

Algunas estrategias predefinidas son:

- Dañar enemigo.
- Matar enemigo.
- Conquistar continente.
- Minimizar fronteras defensivas.
- Defender continente.

Al inicio del turno se selecciona la estrategia con reglas básicas del tipo:

1. "si algún jugador es muy débil ⇒ matar enemigo"
2. "si algún jugador es muy fuerte ⇒ dañar enemigo"
3. "si no se cumplen las reglas anteriores ⇒ Minimizar fronteras defensivas"

Esto hace que tengan un comportamiento demasiado definido y poco flexible respecto a la situación en el tablero, ya que tiene las siguientes limitaciones:

- En muchas ocasiones las reglas se equivocan al seleccionar cuál es la mejor estrategia en la posición.
- Algunas estrategias son contraproducentes. Aunque defender las fronteras de los continentes es generalmente bueno, si el continente no está amenazado los soldados

[7] Lux Delux: http://sillysoft.net/lux/

que lo defienden están fuera de la zona de combate y son inútiles. Dedicar todo el esfuerzo a dañar al jugador más fuerte puede simplemente "regalar" la partida al segundo jugador más fuerte.

- Este esquema no permite fácilmente combinar varias estrategias de forma que se dañe a un jugador al mismo tiempo que se conquista un continente.

En cuanto a los resultados, estos agentes son buenos aunque bastante mejorables. Son muy rápidos y tienen comportamientos razonables pero son incapaces de identificar la mejor estrategia en cada posición.

2.5.2. Sistemas basados en algoritmos evolutivos

Los algoritmos evolutivos realizan una búsqueda optimizando los mejores planes encontrados. Un ejemplo de esta aproximación es el trabajo de Vaccaro [VG04] en el que se tienen en cuenta los siguientes detalles:

1. Guardar información acerca del resultado de aplicar cada plan para acelerar los cálculos de los nuevos planes.
2. Replanificación: durante la ejecución del plan si su probabilidad de éxito se reduce demasiado.

En cuanto a los resultados obtenidos, el tiempo que consume el algoritmo es excesivo para aplicarse en la práctica (20 horas por partida) y no se da ninguna medida del rendimiento contra otras implementaciones.

Por lo tanto, la principal conclusión es que los algoritmos evolutivos para lograr planes son lentos y, aunque podría optimizarse este tiempo no es el interés de este proyecto profundizar en este tipo de técnicas.

2.5.3. Sistemas basados en una arquitectura multiagente

Las arquitecturas multiagente se basan en la división de problemas complejos en problemas más sencillos. En vez de tener un agente que seleccione el mejor plan, se tienen

muchos agentes que cooperan entre sí.

Un ejemplo de aplicación de esta técnica en el dominio del Risk es MARS, desarrollado en la tesis de máster de Fredrik Olsson [Ols05]. Hay un agente por cada país del jugador y un moderador que regula las comunicaciones entre ellos. La decisión de dónde poner las tropas y realizar los ataques se toma mediante un proceso de negociación en el que cada país puja por el ataque que quiere hacer y, el que es capaz de pujar más, realiza su ataque. De este modo, se examinan los ataques interesantes y se escogen los que más valoración tienen.

Los resultados obtenidos con la aplicación de este algoritmo son bastante satisfactorios ya que ha sido comparado con diversos sistemas expertos incluidos en Lux Delux (véase la sección 2.5.1, página 20) mejorando los resultados de todos ellos.

2.6. Conclusiones del estado de la cuestión

El juego del Risk es un desafío notable para el estado de la cuestión actual en Inteligencia Artificial.

El juego tiene más de 50 años de antigüedad y ha recibido bastante atención durante años. Debido a esto, las reglas han ido cambiando y hay muchas versiones. En la actualidad hay muchos aficionados en todo el mundo. Hay múltiples juegos de ordenador que permiten jugar por Internet.

También ha recibido atención de la comunidad científica que, sin embargo, no ha conseguido implementaciones eficientes o con un comportamiento que alcance el calificativo de inteligente. La mayoría de agentes realizan su turno tomando decisiones independientes, sin hacer un plan al inicio del turno. Otros funcionan a través de estrategias predefinidas.

En este proyecto se sigue la línea de búsqueda del mejor plan, aunque no parece haber implementaciones prácticas que jueguen partidas completas.

Conclusiones de la aproximación basada en búsqueda

La aplicación de técnicas de búsqueda tuvo como principal resultado que lo mejor era utilizar algoritmos evolutivos, mientras que las técnicas de búsqueda clásicas funcionaban bastante peor. Sin embargo, el tipo de situaciones escogidas para evaluar estos algoritmos no es relevante para este proyecto:

- No sigue las mismas reglas que este proyecto: hay ocho jugadores y el valor de las cartas es muy elevado.
- No es frecuente en las partidas de Risk: ocho jugadores vivos y que uno tenga el 60 % de las tropas. Además, los países están repartidos de forma aleatoria y en un final de partida real los jugadores tendrían sus tropas agrupadas.
- No es tan relevante encontrar el plan óptimo en este tipo de situaciones debido a la ventaja que tiene el jugador.

Por otro lado, los algoritmos clásicos podrían funcionar mejor si fuesen guiados por heurísticas distintas de la valoración del plan, especialmente en las situaciones de finales de partida en las que fueron probados.

En conclusión, este proyecto continua la aproximación de técnicas de búsqueda centrándose en obtener heurísticas que mejoren los resultados en las técnicas clásicas.

Plataforma para el desarrollo de agentes de Risk

Llama la atención que todos los trabajos buscados en el estado de la cuestión son desarrollos muy particulares. Existen algunos trabajos de aplicaciones de juego de Risk que proporcionan APIs de creación de agentes como Lux Delux o Dominate Game (véase la sección 2.3, página 12) pero tienen algunas limitaciones:

1. Dependen del juego de Risk concreto, por lo que los agentes desarrollados para Lux Delux no pueden integrarse fácilmente en Dominate Game y viceversa.

2. No dan un gran soporte a la reutilización, aunque pueden reutilizarse algunos comportamientos de agentes heredando de ellos.

3. Todo el trabajo en algoritmos implementados no puede reutilizarse para otros dominios, por ejemplo los algoritmos de búsqueda o los relativos a grafos.

Por lo tanto, el agente deberá ser desarrollado desde cero.

Capítulo 3

Objetivos

El objetivo final del proyecto es implementar un agente [1] de Risk, llamado **Ender**. En este capítulo se describen los objetivos concretos del sistema, orientados hacia este objetivo general.

En primer lugar, en la motivación del proyecto se recapitula lo visto en el estado de la cuestión. Después, en los objetivos del proyecto se describen los objetivos que se pretenden alcanzar.

3.1. Motivación del proyecto

Este proyecto trata de cubrir dos huecos dejados por los trabajos anteriores:

1. Hacer un agente de Risk práctico y completo, continuando la línea de búsqueda del mejor plan.

2. Proporcionar una base sobre la que crear agentes de Risk.

[1] El término agente, dentro de este proyecto, hace referencia a un programa o sistema que juega a un juego (como el Risk). No tiene que tener obligatoriamente las características propias de un sistema multiagentes como autonomía, comunicación, etc.

3.2. Objetivos del proyecto

El **objetivo principal** es *diseñar e implementar un agente de Risk* e integrarlo en una aplicación para probarlo jugando partidas contra humanos y otros agentes cumpliendo:

- **Independencia del programa** Podrá integrarse en otras aplicaciones de Risk.
- **Independencia del mapa de juego** Podrá jugar partidas en cualquier tipo de mapa que cumpla las reglas del juego.
- **Heurísticas objetivas** Las heurísticas utilizadas se justificarán a partir de conceptos matemáticos o las reglas del juego.
- **Tiempo razonable** Cada turno durará un tiempo razonable: entre 60 y 300 segundos.

El **segundo objetivo** es construir el agente sobre una *plataforma que pueda ser utilizada en futuros proyectos* de juegos por turnos que cumpla:

- **Mantenibilidad** Soportará la posibilidad de ampliación y modificación de los agentes.
- **Reutilización** Se favorecerá la reutilización de todas las partes del sistema.

Capítulo 4

Desarrollo

En este capítulo se describe todo el trabajo realizado en el proyecto.

Se comienza con el *análisis del sistema* en el que se refinan los objetivos. Se sigue con el *diseño del sistema*, en el que se decide cómo llevar a cabo el trabajo descrito en el análisis.

A continuación, se comienza la descripción de los algoritmos y heurísticas que componen el agente de Risk. Para esto, primero se realizan *estudios de carácter teórico* en los que se obtiene conocimiento que será utilizado posteriormente. El *desarrollo del algoritmo* explica el algoritmo de búsqueda utilizado y en el *desarrollo de las heurísticas* se profundiza en las heurísticas utilizadas para guiar el algoritmo de búsqueda. Después, en la sección de *descartes del algoritmo* se detallan todas las ideas que fueron valoradas pero no llegaron a utilizarse.

Por último, se incluyen los *manuales de usuario y referencia*, para los futuros usuarios y desarrolladores de la aplicación, respectivamente.

4.1. Análisis del sistema

En primer lugar se aporta la descripción de los requisitos del sistema que se refinará con los casos de uso de la aplicación.

4.1.1. Requisitos del sistema

Los requisitos del sistema se dividen en tres *categorías* atendiendo al objetivo del sistema al que afectan:

1. Arquitectura de algoritmos (ALG): La base sobre la que se realizará el agente.
2. Agente de Risk (AGE): La IA que jugará al Risk.
3. Pruebas (PRU): Las pruebas que se realizarán sobre el agente.

Además, los requisitos pueden ser de dos tipos: funcional (F) si expresa una funcionalidad que debe tener el sistema o no funcional (NF) si es una restricción al diseño o implementación del sistema.

De cada requisito se especifica:

1. **Identificador**: identifica de forma unívoca cada uno de los requisitos. Consta de tres partes separadas por guiones: categoría, tipo y número.
2. **Nombre**: resume el requisito.
3. **Descripción**: explicación con detalle del requisito.

ALG-F-1 Configuración mediante parámetros de configuración
Los agentes podrán configurarse modificando ciertos parámetros.

ALG-F-2 Parámetros de configuración desde fichero
Al jugar una partida la configuración de parámetros se leerá de un fichero.

ALG-NF-1 Reutilización de partes del agente
Se podrán reutilizar partes, del agente dentro del mismo o en otros.

AGE-F-1 Agente de Risk
Se realizará un agente que juegue al Risk

AGE-F-2 Integración en aplicación
El agente será integrado en una aplicación que le permita jugar contra otros agentes artificiales o humanos.

AGE-F-3 Variedad de reglas
El agente podrá jugar con diversas reglas. En concreto, deberá ser independiente del tablero y del número de soldados recibidos por el canjeo de cartas.

AGE-NF-1 Tiempo limitado por turno
El agente deberá realizar su turno en un tiempo máximo de dos minutos. Se aplicará una política de "mejor esfuerzo", por lo que esta restricción es flexible.

AGE-NF-2 Heurísticas objetivas
Todas las heurísticas utilizadas deberán estar explicadas y justificadas.

PRU-F-1 Pruebas de rendimiento
Se realizarán pruebas para medir el rendimiento del agente. Recogerán el número de victorias obtenidas en varias partidas.

PRU-F-2 Análisis de tiempos
Se analizará el tiempo que tarda el agente en realizar un turno, detallando el tiempo que tarda en realizar cada cálculo.

PRU-NF-1 Pruebas en varios tableros
Las pruebas se realizarán para al menos dos tableros diferentes.

PRU-NF-2 Pruebas con varios jugadores
Se realizarán pruebas con un número variable de oponentes entre dos y cuatro.

4.1.2. Casos de uso

Los casos de uso permiten refinar la definición del sistema dejando claras las funcionalidades que tendrá que proporcionar a los usuarios.

Identificación de los actores

Los actores son los elementos externos que interactúan con la aplicación:

- **Juego**: actor abstracto que representa cualquier juego por turnos.
- **Juego de Risk**: aplicación que utiliza el agente para jugar partidas.
- **Investigador** realiza pruebas sobre los agentes.

El diagrama de casos de uso de la figura 4.1 muestra todos los casos de uso de la aplicación.

Diagrama de casos de uso

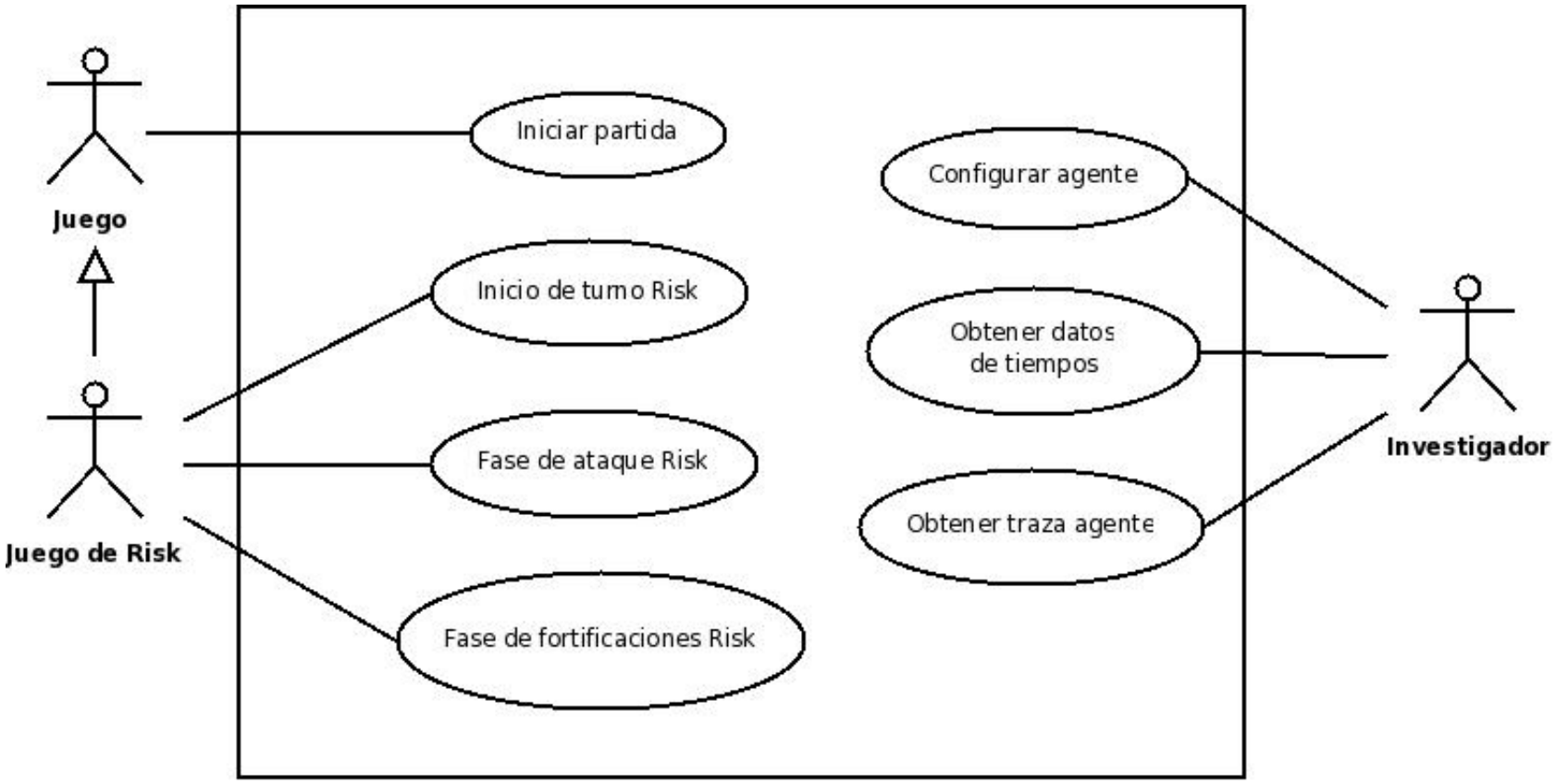


Figura 4.1: Diagrama de casos de uso

Nombre	Caso 1: Iniciar partida
Descripción	Se registra al agente jugador en una partida
Actores	Juego
Precondiciones	Ninguna
Postcondiciones	Partida iniciada
Secuencia	1. El juego crea un nuevo agente 2. El agente se inicializa 3. El juego envía las reglas de la partida, incluyendo el tiempo máximo para realizar el turno 4. El agente se configura con los parámetros adecuados 5. El juego inicia la partida
Secuencia alternativa	Ninguna

Nombre	Caso 2: Inicio turno Risk
Descripción	Se realizan las primeras acciones del turno: canjeo de cartas y posicionamiento de refuerzos
Actores	Juego de Risk
Precondiciones	Partida iniciada
Postcondiciones	Turno iniciado
Secuencia	1. El juego indica al agente el comienzo de turno y le informa del estado actual de la partida 2. El agente inicializa sus estructuras de datos 3. El juego inicia la fase de canjeo de cartas 4. El agente puede canjear cartas o no 5. El juego inicia la fase de posicionamiento de refuerzos 6. El agente coloca los refuerzos
Secuencia alternativa	Paso 3: Si el agente no tiene cartas suficientes, no se inicia la fase de canjeo de cartas

Nombre	Caso 3: Fase de ataque Risk
Descripción	Se realiza la fase de ataque
Actores	Juego de Risk
Precondiciones	Turno iniciado
Postcondiciones	Fase de ataques realizada
Secuencia	1. El juego indica el comienzo de la fase de ataques 2. El agente pide al juego que realice un ataque o finaliza la fase de ataques 3. Si el agente realizó un ataque el juego lo ejecuta 4. Vuelta al paso 2
Secuencia alternativa	Paso 3: Si al realizar el ataque se conquista un país: 1. El juego indica la conquista realizada 2. El agente ordena al juego cuantos soldados trasladar al país conquistado

Nombre	Caso 4: Fase de fortificaciones Risk
Descripción	Se realiza la fase de fortificación
Actores	Juego de Risk
Precondiciones	Fase de ataques realizada
Postcondiciones	Turno finalizado
Secuencia	1. El juego inicia la fase de fortificaciones 2. El agente realiza todas las fortificaciones que desee
Secuencia alternativa	Ninguna

Nombre	Caso 5: Configurar agente
Descripción	Configura el agente para jugar partidas
Actores	Investigador
Precondiciones	Ninguna
Postcondiciones	Ninguna
Secuencia	1. El investigador selecciona los parámetros de configuración que desea y los sitúa en un fichero de configuración 2. El agente utilizará esos parámetros durante la partida
Secuencia alternativa	Ninguna

Nombre	Caso 6: Obtener datos de tiempos
Descripción	Se realiza un informe con datos acerca de los resultados de las partidas
Actores	Investigador
Precondiciones	Partida jugada
Postcondiciones	Ninguna
Secuencia	1. El investigador solicita los datos de tiempo del algoritmo que desee 2. El sistema proporciona el tiempo consumido por cada ejecución del algoritmo subdividido en el tiempo de las partes que lo componen
Secuencia alternativa	Ninguna

Nombre	Caso 7: Obtener traza
Descripción	Se obtiene la traza del agente, para poder solucionar errores o analizar la toma de decisiones que lleva a cabo
Actores	Investigador
Precondiciones	Partida jugada
Postcondiciones	Ninguna
Secuencia	1. El investigador solicita la traza 2. El sistema proporciona la traza del agente en un fichero de log
Secuencia alternativa	Ninguna

4.2. Diseño

El diseño del sistema decide y detalla su esqueleto. En primer lugar, se da la arquitectura de la aplicación en la que se divide en subsistemas, descritos en profundidad en el diseño detallado.

Por último, se estudian las diferentes aplicaciones de juegos de Risk recogidas en el estado de la cuestión para escoger cuál se utilizará y cómo se enlaza al agente.

4.2.1. Arquitectura de la aplicación

En esta sección se describen los módulos en las que se descompone el sistema.

Patrones arquitectónicos

En primer lugar, se examinan los patrones estructurales de arquitectura más comunes. El más clásico es el *modelo vista controlador* (MVC). Sin embargo, en este proyecto no tiene sentido aplicarlo porque tanto de la vista como del controlador van a encargarse los juegos de Risk en los que se integre el sistema.

Los patrones de *arquitectura en pizarra*, *arquitectura multiagente*, *sistemas de producción basados en reglas*, etc. son útiles para desarrollar sistemas de IA pero lo que se pretende es hacer una base para el desarrollo de agentes, por lo que no tiene sentido aplicarlos.

En conclusión, lo mejor es aplicar un modelo de descomposición y no utilizar ningún patrón estructural.

Diseño de la arquitectura

El sistema se subdivide teniendo en cuenta los siguientes criterios:

1. Encapsular la comunicación con los actores.
2. Separar las partes dependientes del juego concreto (Risk).

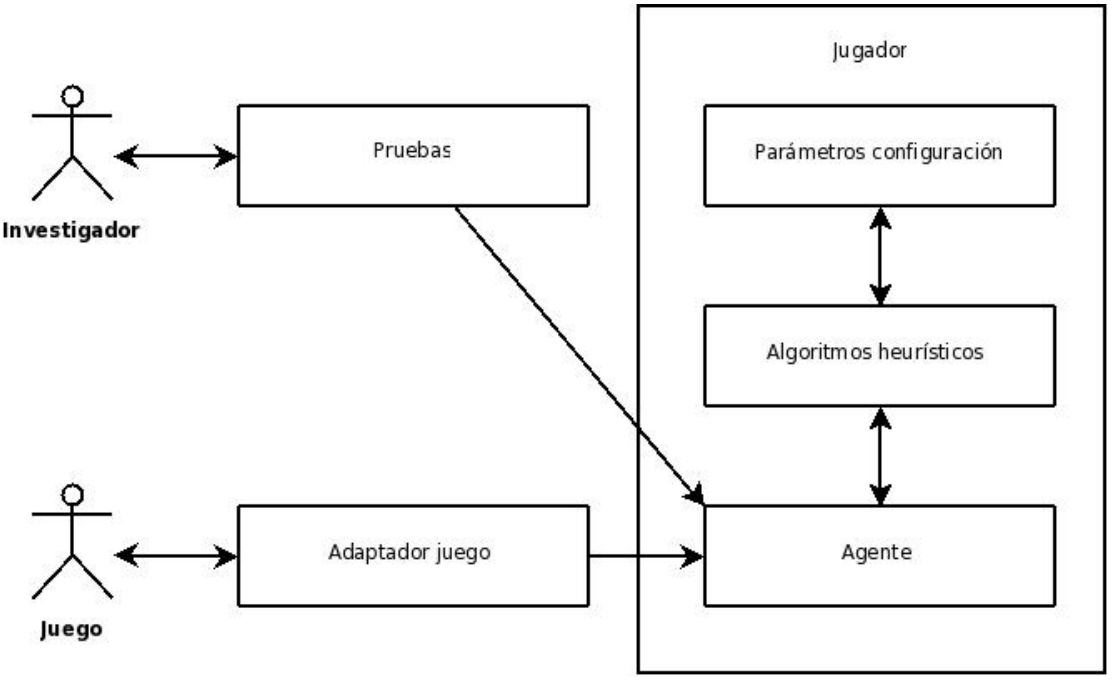


Figura 4.2: Arquitectura de la aplicación

En la figura 4.2 se puede apreciar la división realizada junto a los actores que interactúan con la aplicación y se puede apreciar que se relacionan con subsistemas diferentes.

Agente realiza los turnos durante la partida utilizando los algoritmos heurísticos.

Parámetros de configuración da una base a los algoritmos que componen el agente y permite personalizar su comportamiento mediante parámetros.

Algoritmos heurísticos implementa los algoritmos que componen el agente.

Pruebas recoge los datos solicitados por el investigador y los guarda en ficheros.

Adaptador integra los agentes en aplicaciones de juegos.

4.2.2. Diseño detallado

Aquí se detallan las partes más importantes de cada subsistema con ayuda de diagramas de clases UML.

Subsistema de agente

La clase central del sistema es `AgenteRisk`, mostrada en la figura 4.3.

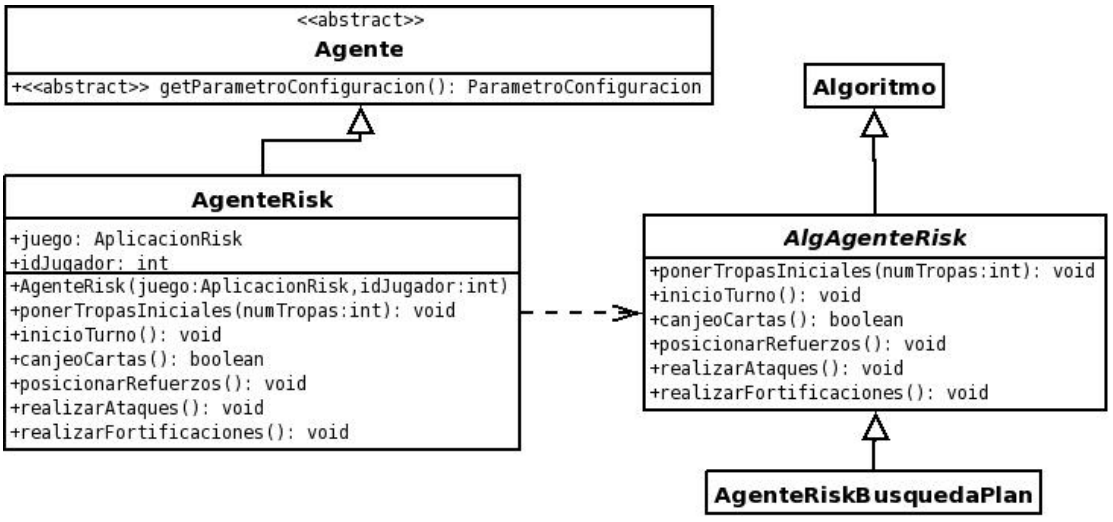


Figura 4.3: Diagrama de clases del agente

Esta clase hereda de `Agente` para que algunas funcionalidades del sistema sean independientes del dominio. Por otro lado, para que pueda haber varias implementaciones de agentes de Risk, este agente no tiene apenas funcionalidad y todas sus llamadas las deriva al algoritmo `AlgAgenteRisk`.

Este algoritmo puede tener diversas implementaciones y `AgenteRiskBusquedaPlan` es la que se va a desarrollar en este proyecto.

Por otro lado el agente necesita utilizar una representación del estado de la partida y, aunque las aplicaciones externas ya la tienen definida, para que el sistema sea independiente se utilizará la que se muestra en la figura 4.4.

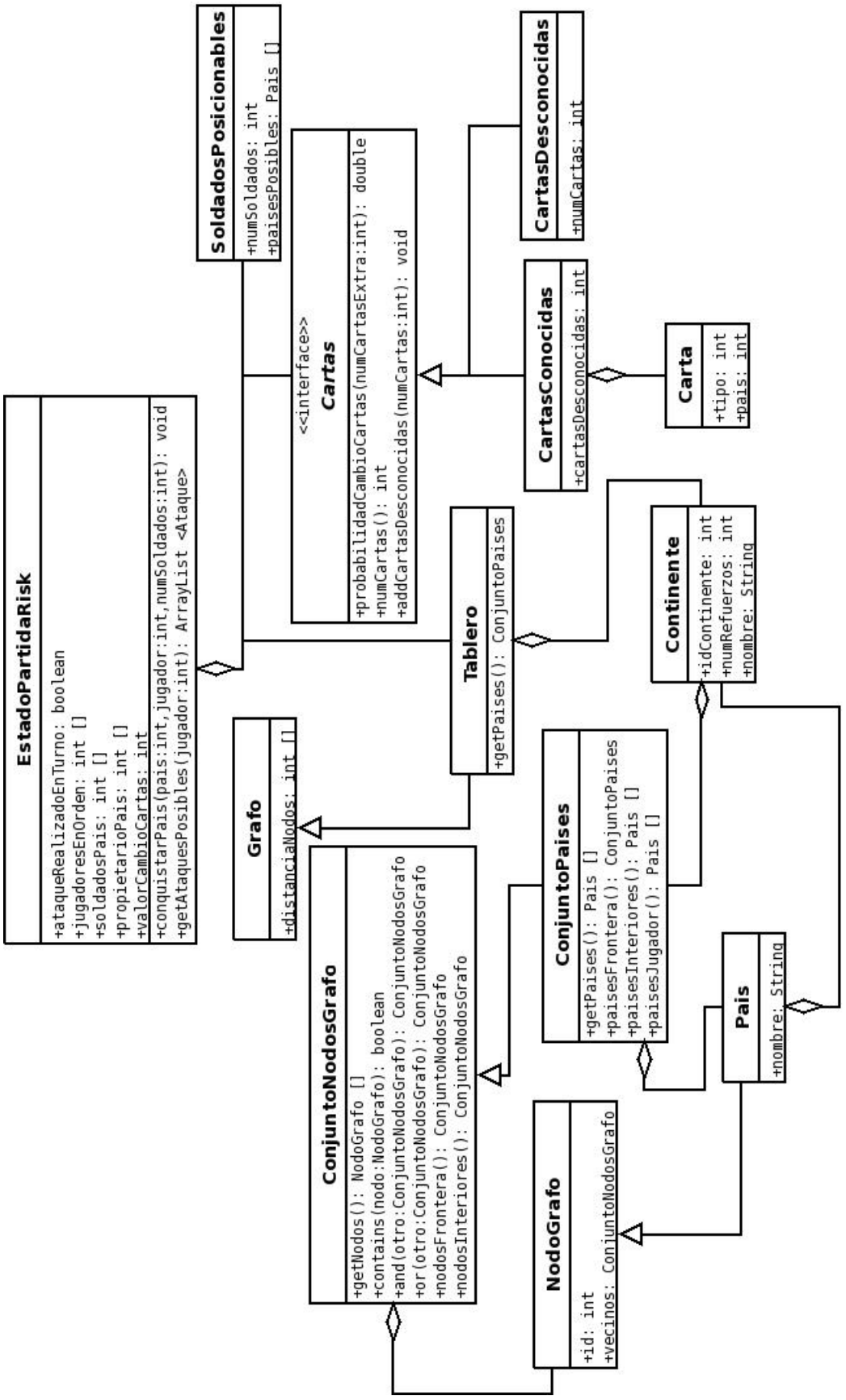


Figura 4.4: Diagrama de clases del estado de la partida

Subsistema de parámetros de configuración

Este subsistema permite la división del agente en algoritmos y su configuración mediante parámetros.

La clase `Algoritmo` representa a un cálculo heurístico que forma parte del agente. Los algoritmos pueden tener varias implementaciones y configurarse mediante parámetros de configuración. Su salida no es objetiva sino un valor estimado o aproximado. Si un algoritmo calculara la salida exacta en un tiempo adecuado no necesitaría de múltiples implementaciones o configuración alguna.

Los parámetros de configuración pueden ser de varios tipos:

- `PConfigDouble`: Un número real.
- `PConfigEntero`: Un número entero.
- `PConfigEnum`: Puede tomar cualquier valor entre varios definidos.
- `PConfigAlgoritmo`: Un algoritmo configurado.

De este modo, el investigador puede seleccionar los parámetros de configuración mediante un fichero de configuración, controlando todo el comportamiento del agente.

En la figura 4.5 se muestran las principales clases de este módulo de la aplicación.

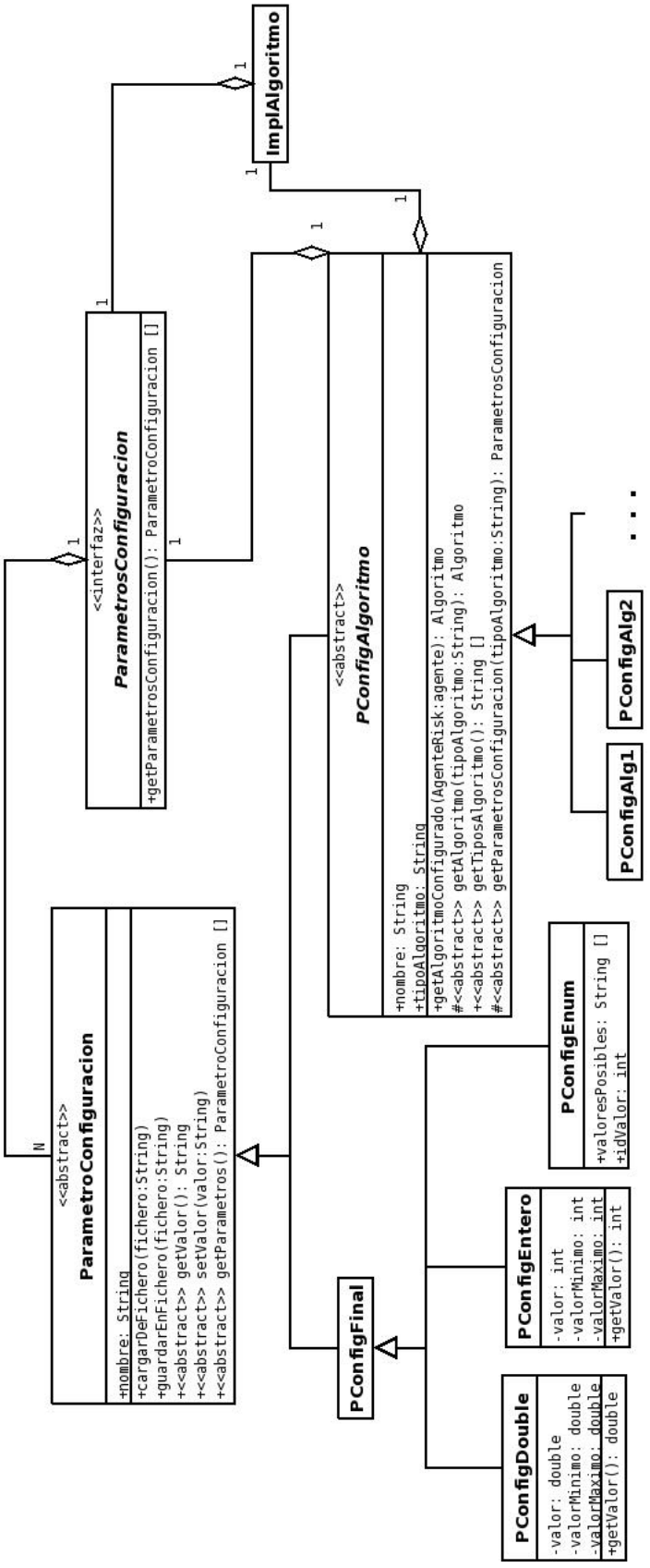


Figura 4.5: Diagrama de clases de los parámetros de configuración

Para modelar el árbol de parámetros se ha adaptado el patrón *composite*, en el que `ParametroAlgoritmo` tiene una agregación de `ParametroConfiguracion` pero a través de `ParametrosConfiguracion`.

La triple relación de agregación entre las clases `ImplAlgoritmo`, `PConfigAlgoritmo` y `ParametrosConfiguracion` es necesaria para manejar los parámetros de configuración de forma independiente a los algoritmos. Así, se puede crear todo el árbol de parámetros de configuración y trabajar con él, sin crear instancias de la implementación del algoritmo hasta que se necesiten.

El parámetro `PConfigAlgoritmo` sirve para seleccionar la implementación que se utilizará de ese algoritmo. Para esto, sus instancias del tipo `PConfigAlg1` contienen un listado con los nombres de todas las implementaciones de `Alg1` e implementan los métodos de creación del algoritmo y sus parámetros dado su nombre.

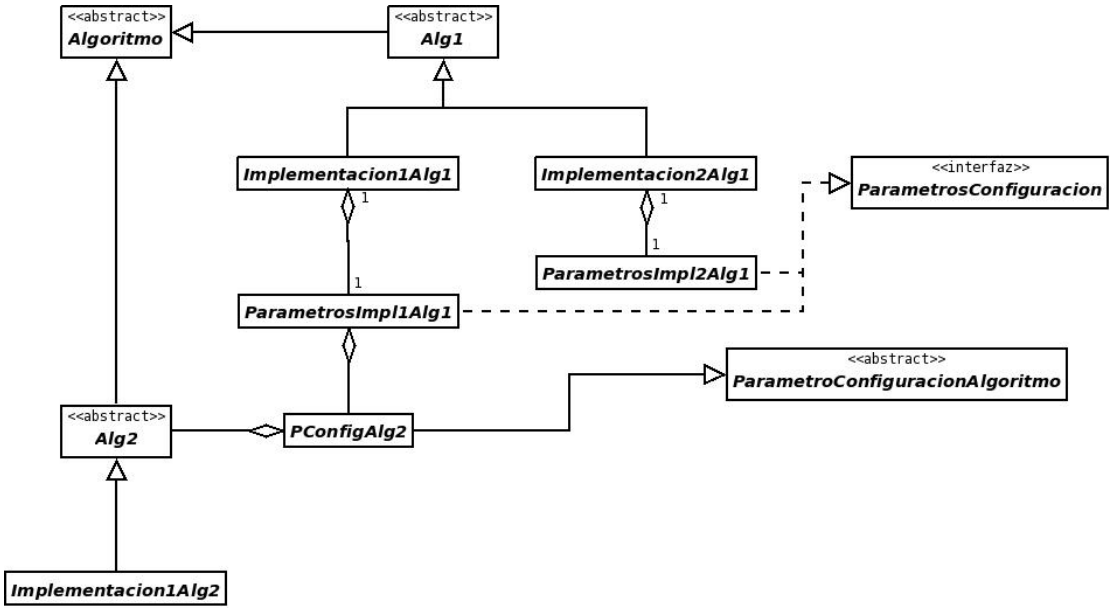


Figura 4.6: Diagrama de clases de la estructura de algoritmos

Las implementaciones particulares de un algoritmo pueden utilizar otros algoritmos para hacer parte de sus tareas. De este modo se crea un árbol de algoritmos en el que la raíz se corresponde con el algoritmo que utiliza el agente. De este modo, un algoritmo puede utilizar otros algoritmos sin conocer la implementación concreta. La figura 4.6 muestra la relación entre las clases que definen un algoritmo y sus implementaciones.

Para definir un nuevo algoritmo debe definirse:

- `AlgNombre`: clase abstracta que hereda de `Algoritmo` y declara el/los método/s de forma abstracta.
- `PConfigNombre`: Hereda de `ParametroConfiguracionAlgoritmo` y permite obtener instancias de las implementaciones y parámetros del algoritmo.
- `Impl1`: Una posible implementación del algoritmo.
- `ParametrosConfiguracionImpl1` (Opcional): Los parámetros de configuración de la implementación del algoritmo, en caso de que los necesite.

Subsistema de algoritmos heurísticos

El subsistema de algoritmos heurísticos, incluye la implementación del algoritmo del agente. Aún no se conocen los algoritmos en los que se subdivide este agente, pero conviene dar un ejemplo de cómo aplicar la estructura de algoritmos para implementar el agente permitiendo ampliarlo y reutilizarlo posteriormente.

La figura 4.7 muestra la estructura de algoritmos que permiten realizar la búsqueda del mejor plan con la posibilidad de añadir en el futuro nuevos tipos de plan y algoritmos de búsqueda de forma independiente.

Como todo se configura mediante los algoritmos, para permitir varios tipos de plan debe crearse una clase `AlgPlan` que permita obtener los algoritmos que manejan ese tipo de plan. `AlgBocetoPlan` permite obtener la versión del boceto de plan de los algoritmos `AlgCrearPlan`, `AlgAnalisisPlan`, etc. Estos algoritmos no aparecen en el diagrama por cuestión de espacio pero evidentemente, `AlgCrearBocetoPlan` hereda de `AlgCrearPlan` y lo mismo para el resto de algoritmos relativos a los planes.

También pueden darse varias implementaciones de cualquiera de estos algoritmos y combinarlas entre sí. Por ejemplo, puede cambiarse el análisis del plan sin afectar a la generación de sus hijos y viceversa.

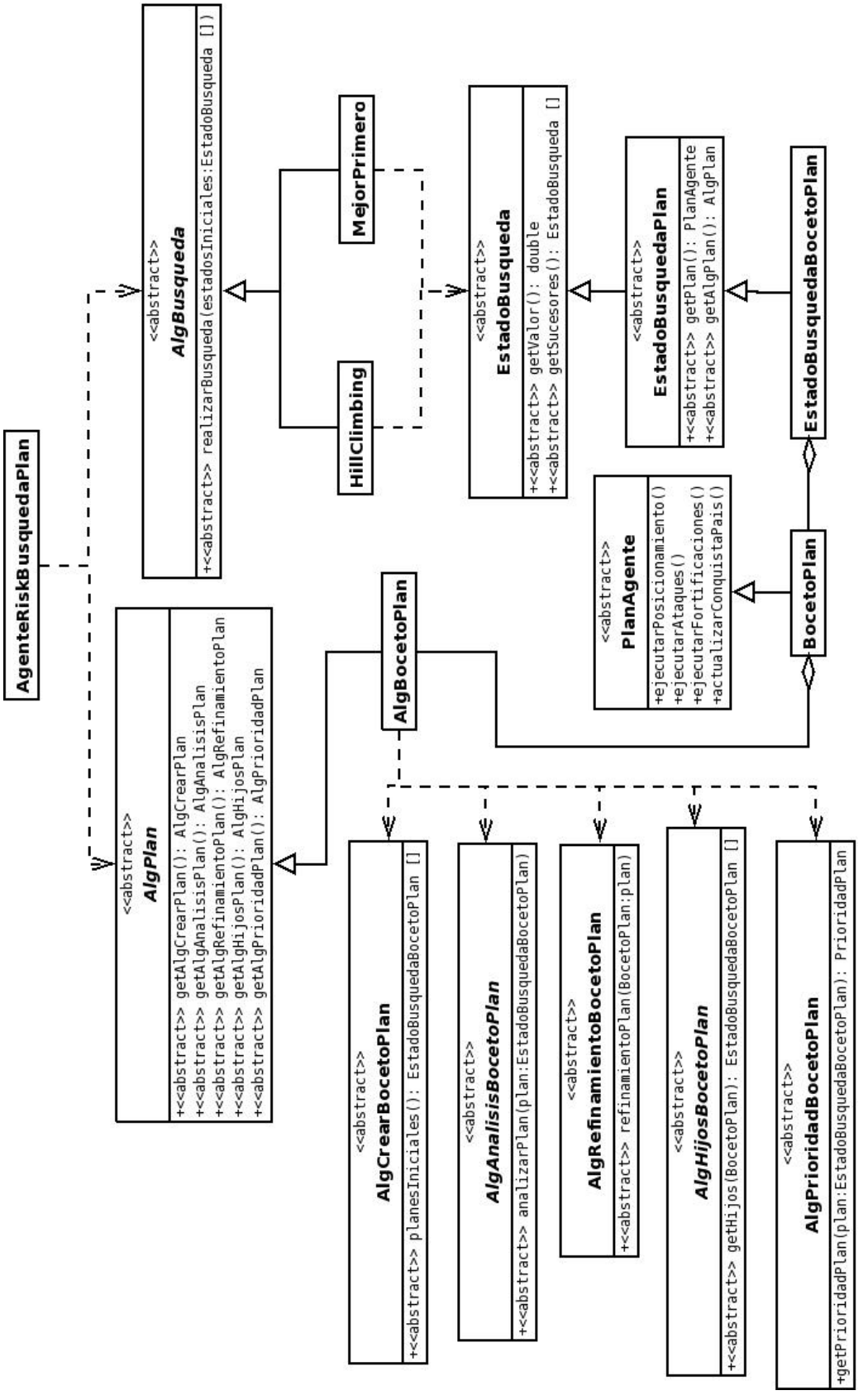


Figura 4.7: Diagrama de clases del agente de búsqueda

Por otro lado, para que los algoritmos de búsqueda puedan reutilizarse deben trabajar sobre `EstadoBusqueda` que no tiene relación con los planes. `EstadoBusquedaPlan` es utilizado por los algoritmos de plan que desconocen el tipo de plan (`AlgCrearPlan`, `AlgAnalisisPlan`, etc.) y `EstadoBusquedaBocetoPlan` contiene el `BocetoPlan`.

Aunque no se muestran en el diagrama, `AlgBocetoPlan` tiene otros parámetros de configuración que son útiles a `BocetoPlan` que accede a ellos mediante la agregación que relaciona ambas clases.

Subsistema de pruebas

La figura 4.8 muestra las clases que se encargan de recoger los datos necesarios durante la ejecución de los algoritmos para poder guardarlos en fichero.

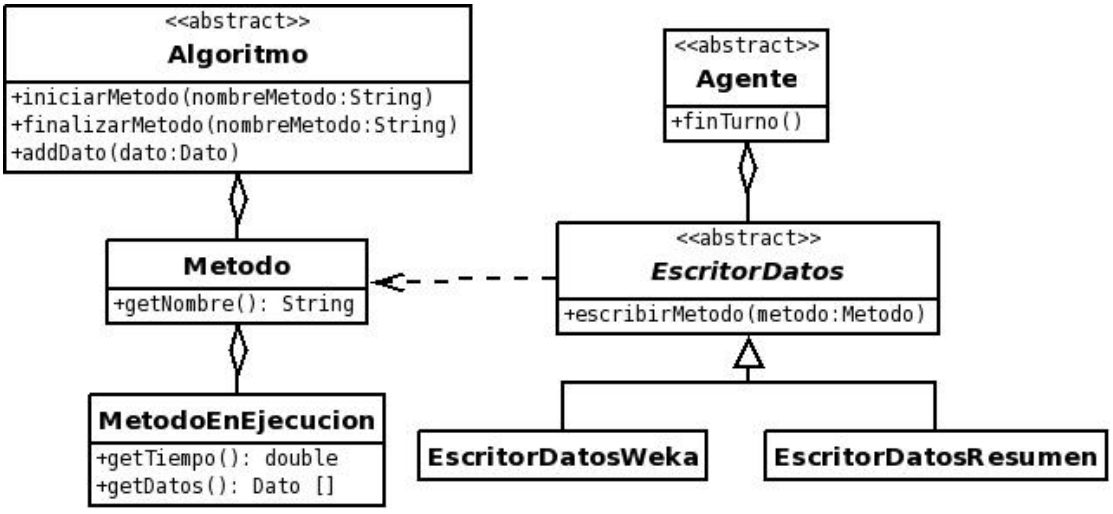


Figura 4.8: Diagrama de clases de las pruebas

Quieren recogerse datos de los algoritmos más importantes. Además, es posible que un algoritmo pueda dividirse en varias partes. Por todo esto, se definen los métodos como las partes de un algoritmo para las cuales se va a recoger su tiempo. La clase `MetodoEnEjecucion` representa una ejecución del método.

Solamente se guardarán datos de aquellos algoritmos que durante su ejecución llamen a los métodos `iniciarMetodo` y `finalizarMetodo`. Opcionalmente podrán llamar al

método `addDato` para incluir otros datos interesantes como el número de nodos explorados en la búsqueda.

Al final de cada turno, es necesario guardar los datos en fichero. Para que el formato de salida sea independiente de los datos recogidos, el agente tiene una o más instancias de `EscritorDatos` que implementarán varias formas de presentar los datos de cada método.

En cuanto al log de la aplicación se utiliza el paquete `java.util.logging` de Java que puede configurarse para mostrar los mensajes deseados por pantalla o escribirlos en fichero.

Subsistema adaptador

El módulo de adaptador se encarga de acoplar los agentes con aplicaciones de juegos externas. En este proyecto se ha escogido Lux Delux (véase la sección 2.3.1, página 12), debido a que es sencillo integrar el agente, los agentes que tiene por defecto son adecuados para hacer pruebas y su interfaz es bastante amigable.

Para incluir el agente en Lux Delux, debe implementarse una clase que herede de `LuxAgent` definiendo sus metodos abstractos.

La figura 4.9 muestra la comunicación entre las clases de Lux Delux y el agente:

- `AgenteRisk`: Recibe la `AplicacionRisk` y realiza sobre ella las acciones que dicta el algoritmo.
- `AplicacionRisk`: Clase abstracta que representa un juego de Risk.
- `Ender`: Implementa los métodos de `LuxAgent` con llamadas a los correspondientes de `AgenteRisk`. Al inicio de la partida crea una instancia de `AplicacionRiskLux` y se la entrega al agente. Al inicio de cada turno y tras cada ataque ordena a dicha instancia que actualice su estado.
- `AplicacionRiskLux`: Implementa los métodos de `AplicacionRisk`, traduciendo las órdenes recibidas por el agente al `LuxDelux.Board` y mantiene el estado de la partida actualizado en todo momento.

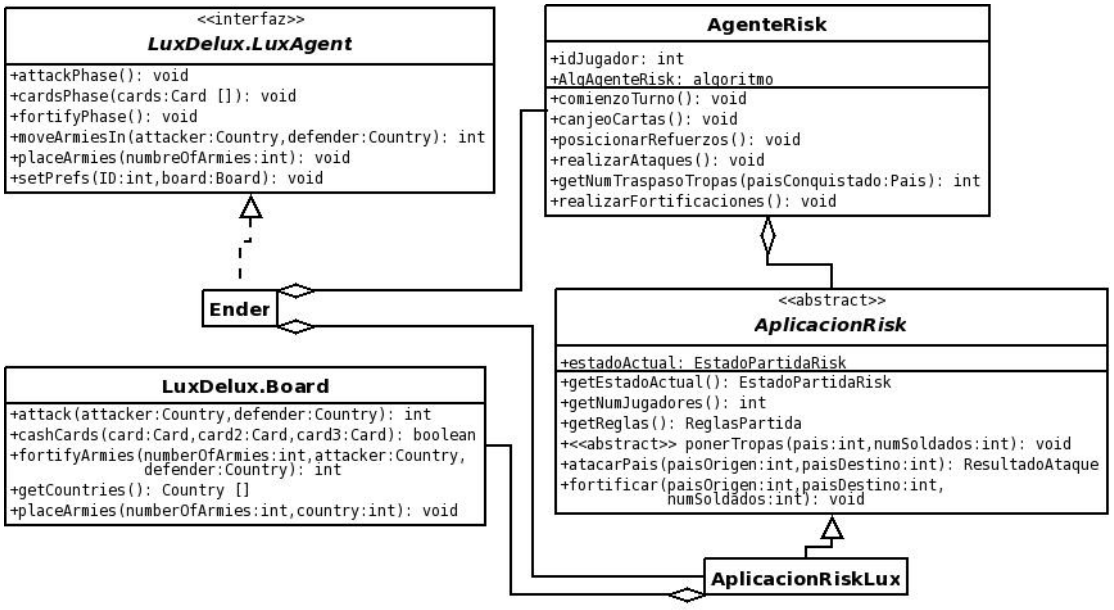


Figura 4.9: Diagrama de clases de integración en Lux Delux

La figura 4.10 muestra un diagrama de secuencia que explica cómo interactuan estas clases. Como se puede ver, en todos los casos el Lux Delux interacciona con Ender quien ordena actualizarse a `AplicacionRisk` (por medio del `LuxDelux.Board`) y llama a `AgenteRisk` para que realice la fase correspondiente del turno. Cuando el agente desea realizar alguna acción se lo indica a la `AplicacionRisk`.

Para más información sobre cómo adaptar el agente de Risk a otros juegos consultar el manual de referencia (véase la sección 4.7.6, página 160).

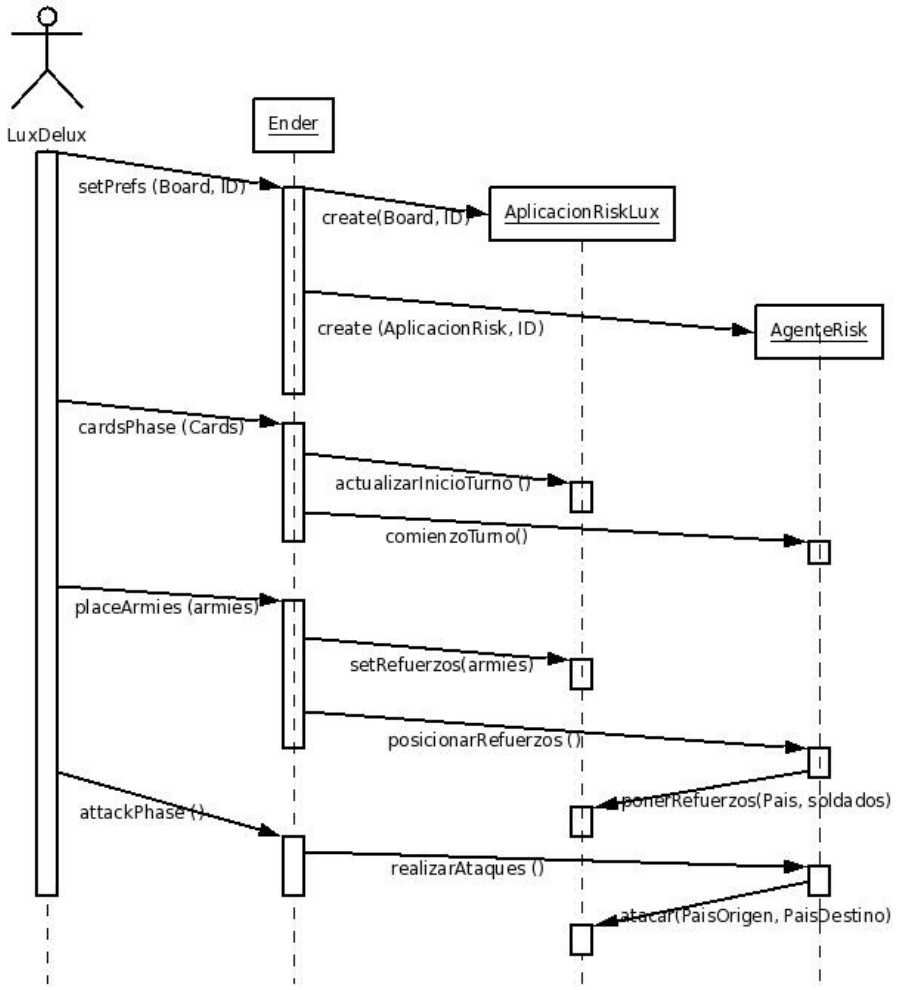


Figura 4.10: Diagrama de secuencia de integración en Lux Delux

4.3. Estudios teóricos del dominio del Risk

Antes de comenzar con el diseño del agente, es necesario comprender el dominio sobre el que se trabaja. Por esto, se comienza estudiando el Risk desde una perspectiva teórica, extrayendo conocimientos que puedan resultar de utilidad en el desarrollo del algoritmo.

4.3.1. Dependencia de las fases del turno

En el estado de la cuestión se comprobó que la mayoría de IAs de Risk asumen que las decisiones que el jugador toma durante su turno (canjeo de cartas, posicionamiento de

refuerzos, ataques a realizar y fortificaciones) pueden hacerse de forma independiente. Además, deciden si realizar (o no) un ataque sin planificar el resto.

El objetivo de este apartado es discutir si esa suposición es válida y qué implicaciones tiene en el rendimiento del agente.

Es evidente que existe una relación entre las distintas decisiones que toma el jugador a lo largo del turno ya que, por ejemplo, los ataques que pueden realizarse durante la fase de ataque están determinados por cómo se sitúen los soldados durante la fase de posicionamiento de tropas y ésta depende de si se canjearon cartas. Lo importante es si, sin tener en cuenta de una forma directa esta relación, es posible lograr un comportamiento inteligente escogiendo las mejores acciones en cada una de las fases.

Por ejemplo, supongamos una situación en la que puede conquistarse uno entre dos continentes, situando todos sus soldados en el escogido. La decisión de qué continente conquistar, que se ejecutará en la fase de ataque, se está tomando indirectamente en la fase de posicionamiento de refuerzos. Si se sitúan los soldados en el sitio correcto (por ejemplo, se reparten siempre cerca del continente que se considere mejor), el resultado de las acciones independientes será correcto.

Demostración de la dependencia de las fases

Para demostrar que tomar estas decisiones de forma independiente no es una simplificación válida, basta con describir una situación que cumpla:

1. Políticas independientes incorrectas. No existe ninguna combinación de políticas independientes (para cada decisión) que sea consistente (pueda funcionar también en el resto de situaciones) y dé como resultado el comportamiento que mejora las probabilidades de victoria del jugador.

2. Políticas dependientes correctas. Existe un modo de obtener cuál es el mejor comportamiento teniendo en cuenta el plan para el resto del turno.

3. Situación típica del juego. Aparece frecuentemente en partidas reales.

Hay varias situaciones en el juego del Risk que cumplen estas características. La más clara de ellas es una situación en la que es posible eliminar a un jugador.

Si durante la fase de posicionamiento de refuerzos no se reparten los soldados para mejorar las probabilidades de éxito del ataque, éste no puede llevarse a cabo, por lo que hay que poner los soldados cerca del jugador débil para poder eliminarlo. ¿Puede una política de reparto de refuerzos situar los soldados para eliminar a un jugador sin planificar los ataques que va a realizar?

Es cierto que podría elaborarse una política que, en caso de que hubiese un jugador muy débil, intentase colocar los soldados lo más cerca posible de dicho jugador. Sin embargo, dicha política no cumpliría con la condición de ser consistente, ya que también podrían darse situaciones en las que el jugador débil no es alcanzable (porque hay que pasar por un país de otro jugador con muchos soldados por ejemplo). Dado que la política de reparto de soldados no planifica los ataques para matar al jugador débil, no puede diferenciarse cuando el jugador débil es alcanzable o no, por lo que se cumple la primera característica.

Además, el motivo por el cuál no se pueden diferenciar estas situaciones es, precisamente, porque no se han planificado los ataques necesarios para eliminarlo. Si se decidiesen los ataques antes de situar las tropas, podría hacerse una mejor política de posicionamiento, por lo que la segunda característica también se cumple.

En el caso de la tercera característica, se puede generalizar la situación planteada a cualquier otra que sea similar pero variando el objetivo del jugador. Posibles objetivos de un jugador pueden ser eliminar un jugador, conquistar un continente, etc. Para determinar si un objetivo es factible debe analizarse cómo se alcanzará y la situación es idéntica a la descrita.

Queda demostrado que no es posible realizar una buena política para todas las fases de forma independiente.

En el caso de los ataques (decidir el siguiente ataque sin planificar el resto) ocurre lo mismo, aún en mayor medida. Para saber si un ataque es interesante, es necesario considerar el resto de ataques que se planea realizar en el mismo turno. Por ejemplo,

atacar a un jugador débil puede ser bueno si se planea eliminarlo completamente, pero podría ser muy malo si no se logra (se debilita más y otro jugador rival podrá eliminarlo y obtener sus cartas). En ese caso es imposible determinar si se realiza el primer ataque sin considerar si después se van a realizar el resto de ataques para eliminar al jugador.

Conclusiones sobre la dependencia entre las fases del turno

Dado que todas las fases del turno están relacionadas, habrá que tomar decisiones acerca de lo que se va a hacer en cada una de ellas antes de comenzar a ejecutar ninguna acción.

Se define un plan como un "elemento de información" que se diseña al comienzo del turno y permanece presente durante todas las fases del mismo, indicando al jugador qué debe hacer y cómo hacerlo. *Ejecutar el plan* es traducir el plan a acciones concretas sobre el tablero de juego.

4.3.2. Probabilidad del canjeo de cartas

Es importante conocer la probabilidad de poder canjear cartas en el próximo turno, tanto propia como de los oponentes.

El cálculo a realizar será la probabilidad de canjeo de cartas dadas X conocidas e Y desconocidas. Obviamente $X \in [0, 3]$ puesto que con 4 cartas en el siguiente turno podrán canjearse seguro e $Y \in [1, 4]$.

En el estado de la cuestión, se recogieron las probabilidades ya calculadas para el juego clásico, en el que hay 42 países (cartas normales) y 2 comodines. Sin embargo, dado que este cálculo depende del número de comodines y cartas normales y éste a su vez depende del número de países del tablero (hay una carta normal por cada país), al variar el tablero hay que recalcular las probabilidades.

Para obtener la probabilidad de canjeo, lo más simple es obtener todas las posibles combinaciones que pueden darse en las Y cartas. Calcular su probabilidad según las cartas conocidas y el número de cartas totales y sumar la probabilidad de todas las combinaciones de aquellas que puedan canjearse.

Según aumenta el valor de Y, el número de posibilidades N aumenta exponencialmente. Como hay cuatro tipos de cartas, $N = 4^Y$. Por ejemplo, las combinaciones para $Y = 2$ serán:

$P(sold, sold)$	$P(sold, cab)$	$P(sold, canon)$	$P(sold, com)$
$P(cab, sold)$	$P(cab, cab)$	$P(cab, canon)$	$P(cab, com)$
$P(canon, sold)$	$P(canon, cab)$	$P(canon, canon)$	$P(canon, com)$
$P(com, sold)$	$P(com, cab)$	$P(com, canon)$	$P(com, com)$

Como el valor máximo de Y es 4, en el peor caso, el número máximo de combinaciones será $4^4 = 256$, por lo que el cálculo de todas las combinaciones posibles es, en este caso, viable.

Ejemplo

$X = 1$ soldado y 1 caballo. $Y = 1$ carta desconocida.

Número de comodines = 1.

Número de países = Número de cartas normales = 51.

Solución

Cartas desconocidas restantes = 52 - 2 = 50

$P(sold) = \frac{16}{50}$, $P(cab) = \frac{16}{50}$, $P(canon) = \frac{17}{50}$, $P(com) = \frac{1}{50}$

Posibilidades:

- 2 soldados y 1 caballo $\Rightarrow \frac{16}{50}$ (no canjeable)
- 1 soldado y 2 caballos $\Rightarrow \frac{16}{50}$ (no canjeable)
- 1 soldado, 1 caballo y 1 cañon $\Rightarrow \frac{17}{50}$ (canjeable)
- 1 soldado, 1 caballo y 1 comodín $\Rightarrow \frac{1}{50}$ (canjeable)

$P(canjear) = \frac{17}{50} + \frac{1}{50} = \frac{18}{50}$

4.3.3. Definiciones relativas a los ataques

Los ataques son la acción más importante ya que un jugador que nunca ataque no puede ganar. Los ataques que un jugador puede realizar en un turno se agrupan en dos tipos de estructuras: listas y árboles.

Lista de ataques Sucesión de varios ataques totales. El país origen de cada ataque debe coincidir con el país destino del ataque anterior.

Se define el país origen de la lista como el país origen de su primer ataque y el país destino de la lista como el país destino de su último ataque.

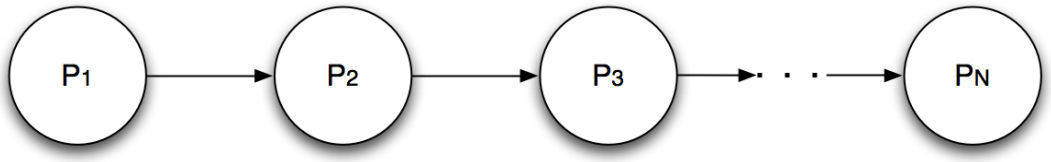


Figura 4.11: Lista de ataques genérica

La figura 4.11 muestra una lista de ataques genérica en la que el jugador propietario del país P_1 conquista los países P_2 a P_n.

El país desde el que se realiza cada ataque es conocido ya que P_i siempre es atacado desde P_{i-1}. El orden de los ataques también esta determinado por la propia estructura de la lista ya que para conquistar P_i es necesario haber conquistado P_{i-1}.

Árbol de ataques Conjunto de ataques organizados en una estructura de árbol.

La figura 4.12 muestra un árbol de ataques genérico, en la que el propietario de la raíz P_0 conquista los otros países del árbol. Cada nodo del árbol es atacado desde su nodo padre, es decir, P_{kj} es atacado desde P_k.

No hay ningún orden implícito o predeterminado en el que se tengan que realizar los ataques del árbol aunque, evidentemente, antes de realizar un ataque tendrá que haberse

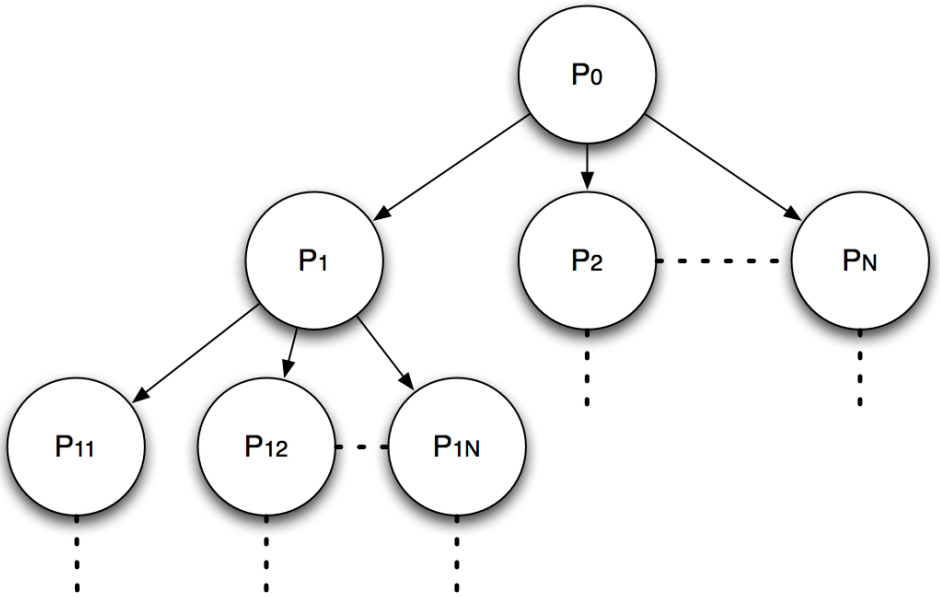


Figura 4.12: Árbol de ataques genérico

conquistado su origen. En el ejemplo, P_1 y P_2 pueden realizarse indistintamente pero P_{11} no podrá hacerse antes que P_1.

Al conquistar un país situado en un nodo intermedio que tenga nodos hermanos, el árbol se subdivide en dos, siendo el país conquistado la raíz del nuevo árbol. La figura 4.13 muestra un ejemplo del resultado de conquistar P_1.

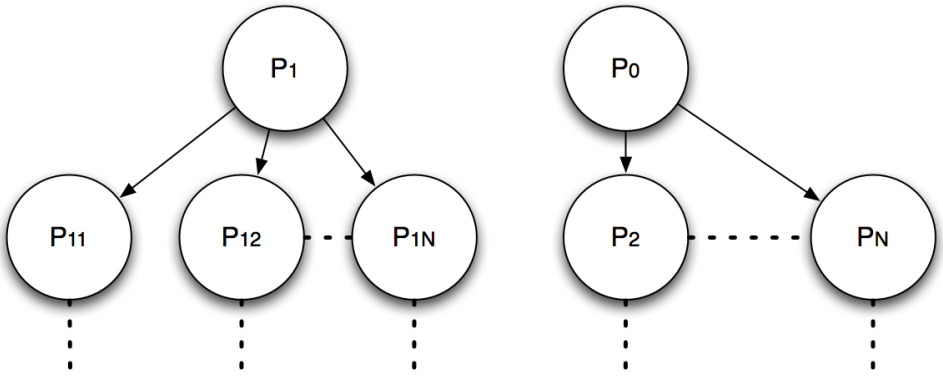


Figura 4.13: División de un árbol de ataques

Resultado de los ataques

Los estudios recogidos en el estado de la cuestión sólo buscaban calcular la situación final tras los ataques. Aquí se extiende el propósito de estos estudios, considerando desconocido el número exacto de soldados atacantes. Esto se debe a que, aunque se conoce el número de soldados en el país atacante, no todos tienen necesariamente que emplearse en el ataque, ya que algunos pueden quedarse en dicho país defendiéndolo.

Hay tres tipos de soldados:

- Inactivos: no participan en los ataques.
- Muertos: mueren al realizar los ataques.
- Supervivientes: participan en los ataques y sobreviven.

Para conocer la situación tras los ataques es necesario saber, de los soldados iniciales, cuántos participan en los ataques y, de éstos, cuántos mueren o sobreviven.

Coste

El *coste del ataque* determina el número medio de soldados que morirán al realizar un ataque. El número medio de soldados que quedarán tras la realización de un determinado ataque será precisamente el número de soldados que se tenía antes de realizarlo menos su coste.

Factor de riesgo

El *factor de riesgo* es el número de soldados que deben participar en el ataque para garantizar el éxito con una certeza razonable.

Este cálculo limita los planes evaluados por el agente, podando aquellos que no dispongan de soldados suficientes para garantizar la probabilidad de éxito deseada. Además, determina el número mínimo de soldados a trasladar a cada nodo del árbol de ataques durante la ejecución del plan.

Dado que el éxito del ataque nunca puede ser asegurado al 100 % de probabilidad (requeriría infinitos soldados), hay que determinar el *riesgo* que el agente está dispuesto a correr. Por lo tanto, se define como parámetro para el cálculo del factor de riesgo, el porcentaje de probabilidad de éxito de los ataques P. Este parámetro es de gran importancia para el correcto funcionamiento del agente: un valor demasiado bajo provocaría que fracasen muchos planes y demasiado alto limitaría la capacidad del agente para realizar ataques.

Factor de riesgo mínimo Como el factor de riesgo debe cumplirse para cada nodo del árbol, independientemente de la cantidad asumida para un nodo del árbol como su factor de riesgo, éste siempre tendrá que tener una cantidad mínima.

Los nodos no terminales deben asegurar que hay suficientes soldados para realizar sus ataques y pasar los soldados necesarios a sus nodos hijos. Además, deberá quedar al menos soldado en el país conquistado por lo que el factor de riesgo mínimo es el que muestra la ecuación 4.1.

$$FRmin = 1 + \sum_{i \in hijos} \lceil Coste_i \rceil + \lceil FR_i \rceil \tag{4.1}$$

En los nodos terminales deben pasar los soldados empleados en el último ataque, entre uno y tres. Por defecto, será siempre tres, ya que siempre que haya soldados suficientes se realizará el ataque con tres soldados. Sin embargo, en casos en los que no hay soldados suficientes para realizar el último ataque con tres soldados, puede asumirse el riesgo y reducir el *FRmin* a dos soldados. Esto sólo es permisible si el ataque no tiene hermanos.

Por lo tanto el factor de riesgo queda finalmente como $FR = \lceil max(FRmin, FRcalculado) \rceil$ donde *FRcalculado* es la aproximación estudiada en la sección 4.3.6, pagina 64.

Probabilidad de éxito

La probabilidad de éxito de un ataque mide la probabilidad de que el atacante conquiste los ataques planificados, para lo que necesita conocer el número de soldados atacantes.

4.3.4. Algoritmo de simulación de ataques

El algoritmo de simulación de ataques realiza una simulación contemplando las diferentes posibilidades que pueden ocurrir en cada acción y analizando su probabilidad.

Para esto necesita conocer todas las acciones que realizará el jugador, por lo que se supone una lista de ataques, que se continua hasta que no quedan tropas. Siempre se trasladan todos los soldados posibles para realizar el siguiente ataque.

El resultado que se pretende calcular es un estado no determinista que incluye los estados que pueden darse y su probabilidad. Estas posibilidades constan del número de supervivientes atacantes y defensores. Son estados finales los que tienen un atacante o cero defernsores.

Para realizar esta simulación se considera el estado actual con una probabilidad del 100 % y se van generando los estados que pueden darse a partir de los ya calculados, hasta llegar a un conjunto de estados finales. La probabilidad de cada estado se calcula como la probabilidad de su estado padre (el que lo genera) multiplicada por la probabilidad de transición desde el estado padre a ese estado (véase la sección 2.4.1, página 14).

El número de soldados atacantes inicial debe estar fijado de antemano para poder realizar la simulación y es desconocido a priori, ya que uno de los objetivos del cálculo es obtener el factor de riesgo. La solución es probar con diferentes soldados atacantes, recoger el resultado de cada caso y analizar su convergencia.

Se define un estado como una tupla (nº de soldados atacantes vivos (N), nº de soldados defensores vivos ($E_1 - E_n$), *Probabilidad*), y se aplica el algoritmo mostrado en la figura 4.14

El algoritmo anterior deja todas las posibles opciones con su probabilidad. A partir de ellas, se pueden calcular muchos datos de interés, como se muestra en la tabla 4.1.

Sin embargo, el algoritmo, añade tres elementos a la lista por cada uno que elimina, por lo que el número de nodos explorados crece exponencialmente a medida que aumenta el número de soldados propios o del enemigo. De hecho, resulta totalmente inviable realizar el cálculo para $N = E = 20$.

```
Inicializa la lista de posibilidades con (N, [E_1 - E_n], 1)

Mientras quedan estados en la lista que cumplan: N > 1 o E_n > 0
  Se coge el primer estado de la lista y se elimina de ella
  Se realiza un ataque simple sobre ese estado, para dar nuevos estados
  P(nuevo estado) = P(resultado ataque) × P(estado)
  Se insertan los nuevos estados en la lista

En la lista quedan los posibles estados finales con su probabilidad
```

Figura 4.14: Algoritmo de simulación de ataques

Objetivo	Cálculo
Probabilidad de éxito o victoria	Sumar la probabilidad de todos los estados finales en los que el número de defensores es 0.
Número medio de soldados restantes	Sumar el número de soldados atacantes de cada estado final multiplicados por su probabilidad.
Soldados que sobrevivirán con probabilidad P	Ordenar los estados por número de soldados y recorrerlos acumulando su probabilidad hasta llegar a P.
Coste	Restar el número medio de soldados restantes al número de soldados atacantes inicial.
Factor de riesgo a un $P\,\%$	Se lanza el algoritmo con N atacantes y se calcula su probabilidad de éxito. Si es mayor o igual a P, N es la solución y, en caso contrario, se lanza de nuevo con $N+1$.

Tabla 4.1: Valores calculables con el algoritmo de simulación de ataques

Se puede realizar una optimización si se observa que hay simetrías en los estados explorados, es decir, muchos son idénticos aunque se alcanzan de distinta forma. Por ejemplo, para pasar de (N, E) a $(N-2, E-2)$ puede hacerse de tres formas distintas: empatando dos veces, ganando primero un jugador y luego el otro o al revés.

Antes de la inserción en la lista comprueba si contiene un estado igual y, si es así, le suma la probabilidad en vez de añadir el estado. Además, se puede mantener la lista ordenada para evitar que se expandan estados que puedan aparecer más adelante, generados por otro estado de la lista. Dado que los nuevos estados se generan mediante ataques y estos siempre eliminan soldados. El orden debe ser descendente respecto al número de soldados totales, $N + E$.

Con esta mejora, si se tienen N atacantes y M defensores, como cada ataque elimina dos soldados, el número de niveles expandidos será $T = \frac{N+M}{2}$. En el nivel N_i se habrán eliminado $2i - 1$ soldados y como estos se pueden repartir entre atacantes y defensores habrá como mucho ese número de estados [1]. Sumando la serie de los soldados en cada nivel se obtiene que el número de estados expandidos por el algoritmo es $E \leq T^2$. Por lo tanto $E \leq \frac{(N+M)^2}{4}$, por lo que la complejidad del algoritmo optimizado es polinómica respecto al número de soldados totales.

En la figura 4.15 se muestra el rendimiento obtenido con diferente número de soldados atacantes para el cálculo del coste y probabilidad de éxito en la conquista de dos países con tres soldados.

Se puede observar la mejora en la complejidad temporal del algoritmo respecto al número de atacantes. El efecto es similar al aumentar los defensores.

La figura 4.16 muestra como el tiempo de la simulación optimizada crece de forma polinomial respecto al número de soldados.

Sin embargo, a pesar de las optimizaciones, el algoritmo sigue siendo excesivamente lento para su implementación en el agente. El coste, factor de riesgo y probabilidad del ataque deben ser calculados muchas veces en cada turno, por lo que se necesita calcularlos en un

[1] En caso de que no hubiese estados finales sería exactamente ese número. Aunque el resultado no sea exacto sirve de cota superior para el número de estados expandidos.

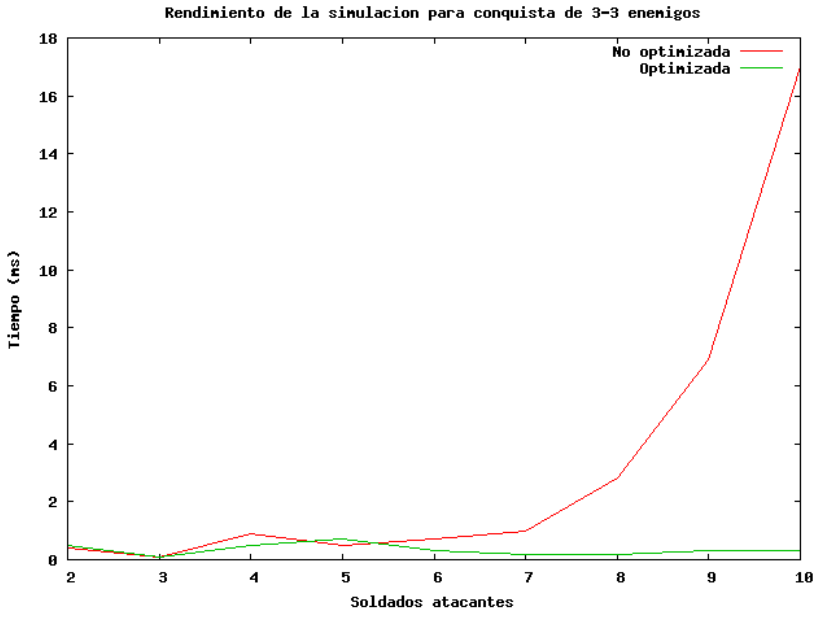


Figura 4.15: Rendimiento de simulación

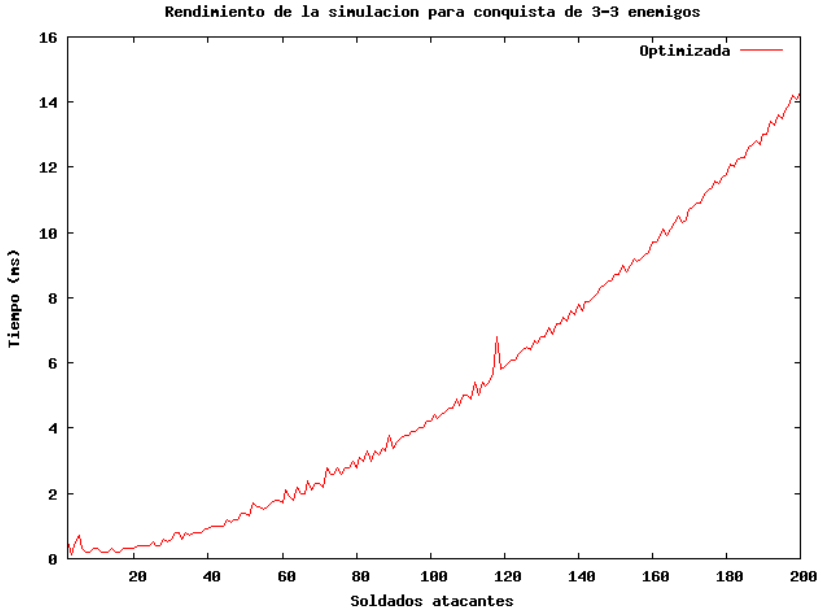


Figura 4.16: Rendimiento de la simulación optimizada

tiempo constante para que el rendimiento del agente no se vea afectado por el número de soldados.

Por esto, en las próximas secciones se intentarán aproximar los datos mediante una función matemática.

4.3.5. Aproximación del coste

El coste depende tanto del número de defensores como de atacantes, ya que debe ser menor que el número de atacantes. Además, a medida que aumenta el número de atacantes, el coste converge hacia un valor determinado, puesto que para eliminar a N enemigos, siempre habrá que ganar, al menos, $\lceil \frac{N}{2} \rceil$ tiradas (en cada tirada se pueden matar hasta 2 soldados defensores) y la probabilidad de perder en dichas tiradas es independiente del número de atacantes, siempre que se tengan suficientes atacantes.

El valor que debe aproximarse es al que converge el coste medio cuando el número de atacantes tiende a infinito. El caso de que el número de atacantes sea menor se descarta ya que gracias al factor de riesgo se garantiza que habrá suficientes soldados.

Coste del ataque a un país

Para el cálculo del coste de un ataque se puede utilizar la expresión ya detallada en el estado de la cuestión (véase la sección 2.2, página 15). Esta expresión permite aproximar el valor al que converge el coste de un ataque si se tienen soldados suficientes.

Hay que comprobar el error cometido por esta expresión, comparando sus resultados con los del algoritmo de simulación de ataques. La figura 4.17 muestra como la función se aproxima perfectamente al límite cuando el número de atacantes tiende a infinito en un caso con 100 defensores. La exactitud es absoluta, teniendo en cuenta que la precisión requerida es un número entero y se aproxima en varios decimales.

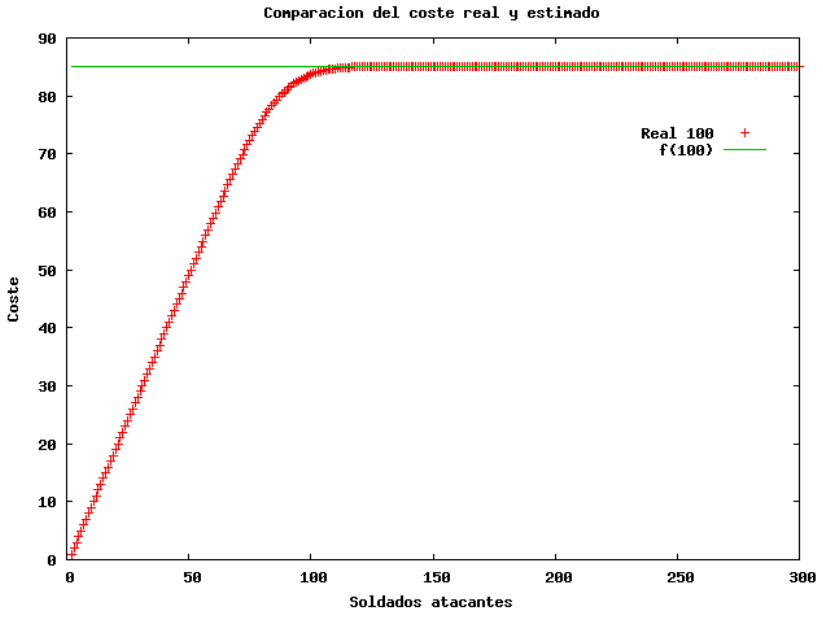


Figura 4.17: Comparativa del coste real y estimado en ataques simples

Coste de una lista de ataques

El coste de una lista de ataques es la suma del coste de sus ataques puesto que los ataques a cada país son independientes cuando hay suficientes soldados atacantes.

Para comprobar esto, de nuevo se analizan datos proporcionados por el simulador de ataques, obteniendo el coste de la cadena de ataque completa, para cadenas de uno a veinte países, en las que hay entre uno y cinco soldados en cada país.

Como se puede apreciar en la figura 4.18, el resultado es lineal y, precisando más, es igual al número de países multiplicado por el coste de conquistar cada país.

Coste de un árbol de ataques

En el caso de un árbol de ataques, al igual que en el caso de la lista de ataques, se puede comprobar que el coste de cada ataque es independiente del resto de ataques (siempre que el número de atacantes sea suficientemente alto).

No es necesario aportar datos empíricos puesto que el caso es exactamente el mismo que

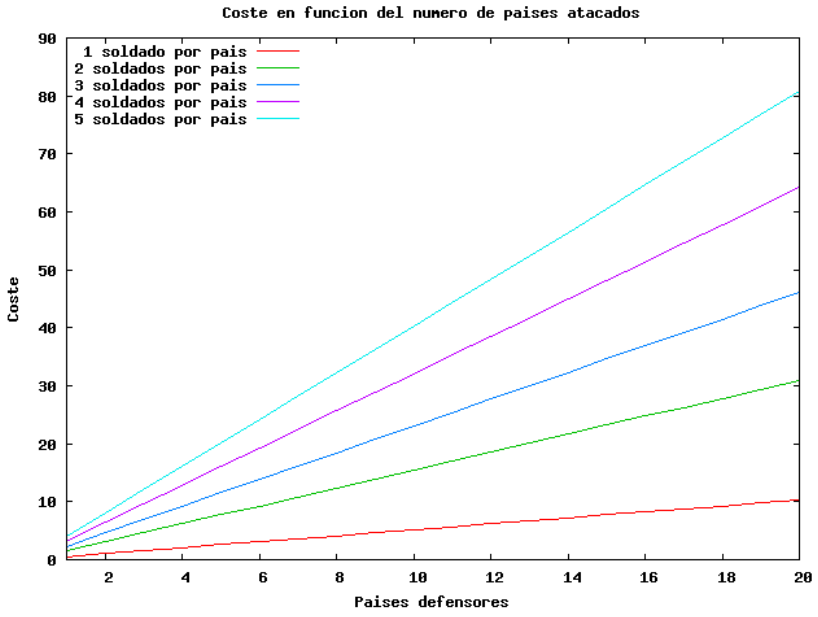


Figura 4.18: Coste de una lista de países

en una lista de ataques: al ser todos los costes de los ataques independientes entre sí, no influye si están situados en forma de árbol o de lista.

4.3.6. Aproximación del factor de riesgo

El objetivo es recoger datos mediante el algoritmo de simulación de ataques y aproximar la función que los genera, obteniendo una expresión matemática que calcule el factor de riesgo en cualquier situación y para cualquier porcentaje de éxito. Esta aproximación se realizará probando varias funciones ajustadas con el algoritmo de Marquardt-Levenberg, incluido en el software gnuplot[2].

Hay dos variables de entrada al problema: los enemigos E y el porcentaje de éxito $Prob$. Este porcentaje de éxito estará siempre en el intervalo entre 70 % y 99,99 %, puesto que valores inferiores se consideran de demasiado riesgo y nunca puede llegarse al 100 % de probabilidad garantizada.

[2]Página oficial de gnuplot: http://www.gnuplot.info/

Factor de riesgo del ataque a un país

Se ataca un país E con *Enem* soldados.

En la figura 4.19 se muestran datos recogidos con el algoritmo de simulación de ataques, con *Enem* en el intervalo [1, 200], para varios valores de *Prob*: 0.8, 0.85, 0.9 y 0.95. Como se puede ver todas tienen una forma muy similar, por lo que deben responder a la misma función variando sus parámetros según el porcentaje de éxito escogido.

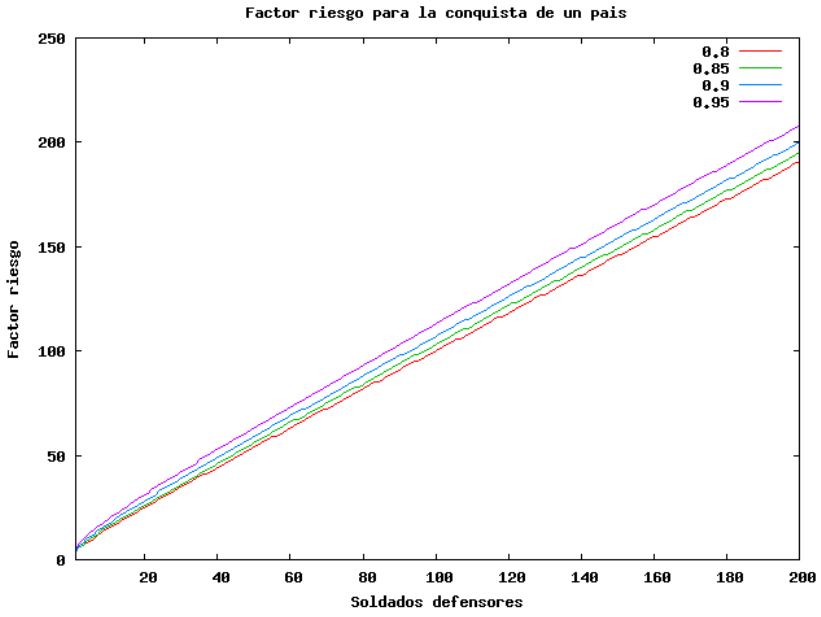


Figura 4.19: Factor de riesgo de un ataque simple

En la figura 4.20 se puede apreciar el ajuste de dos funciones para dos porcentajes de éxito diferente, 80 % y 97,5 %. Para ambos porcentajes la función polinómica, $Ax^B + C$, se ajusta a los datos reales con muy poco error. El factor de riesgo se calcula redondeando al número entero más cercano el resultado de la ecuación 4.2.

$$FR = AEnem^B + C \tag{4.2}$$

Es necesario ajustar los parámetros A, B y C para cada posible *Prob*. Para esto, se recogen datos de los parámetros ajustados para todas las probabilidades de éxito desde 70 % hasta el 99,5 % por cada 0,5 %.

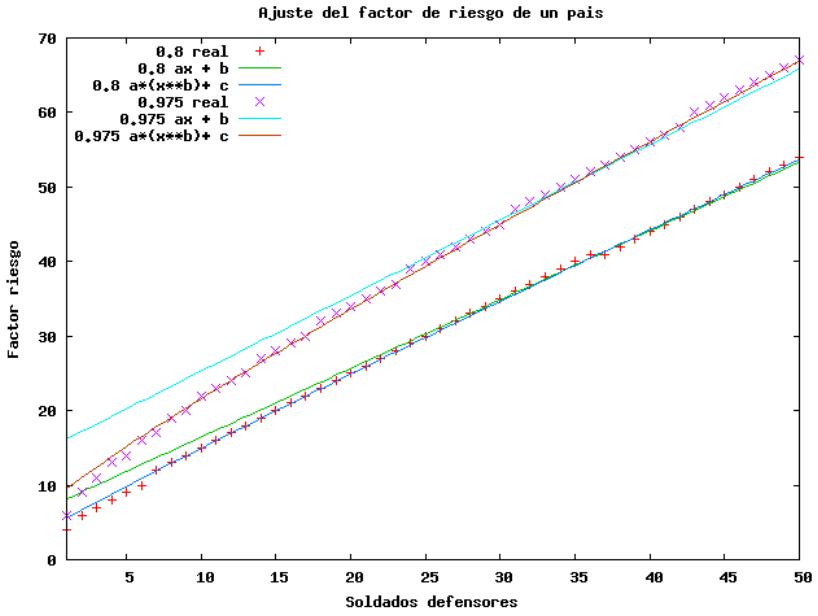


Figura 4.20: Ajuste de la función del factor de riesgo de un ataque simple

Ajuste del parámetro A

En la figura 4.21 se ve que la función que mejor aproxima el parámetro A es:

$$A = A_a Prob^{A_b} + A_c Prob^{A_d} + A_e$$

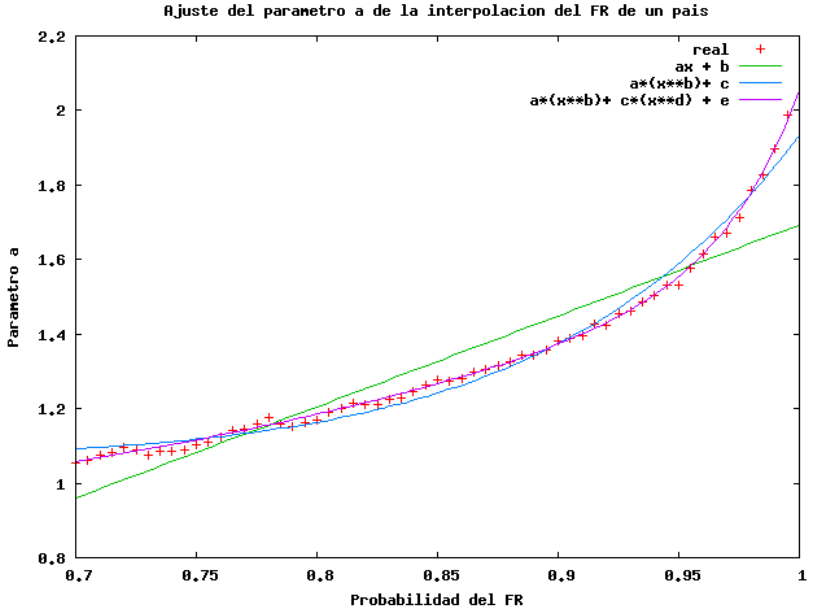


Figura 4.21: Ajuste del parámetro A

El valor aproximado para los parámetros es:

$$A_a = 0{,}745410873070051$$
$$A_b = 3{,}36684031830359$$
$$A_c = 0{,}475093382664162$$
$$A_d = 31{,}5426035438591$$
$$A_e = 0{,}833041811510867$$

Ajuste del parámetro B

En la figura 4.22 se ve que la función que mejor aproxima el parámetro B es:

$$B = B_a Prob^{B_b} + B_c Prob^{B_d} + B_e$$

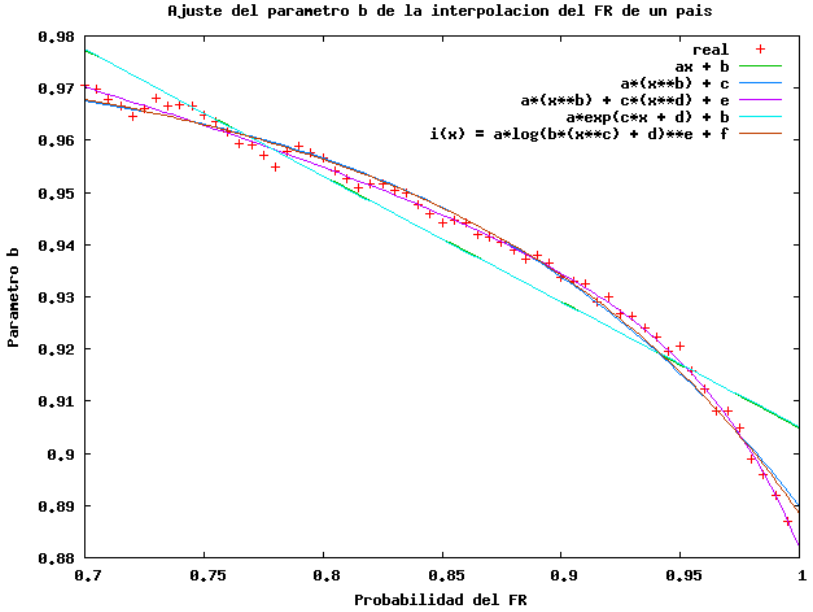


Figura 4.22: Ajuste del parámetro B

El valor aproximado para los parámetros es:

$$\begin{aligned} B_a &= -0{,}0337228504119188 \\ B_b &= 25{,}5437722917476 \\ B_c &= -0{,}0952485451838192 \\ B_d &= 2{,}37673938021666 \\ B_e &= 1{,}01100067181552 \end{aligned}$$

Ajuste del parámetro C

En la figura 4.23 se ve que la función que mejor aproxima el parámetro C es:

$$C = C_a Prob^{C_b} + C_c Prob^{C_d} + C_e$$

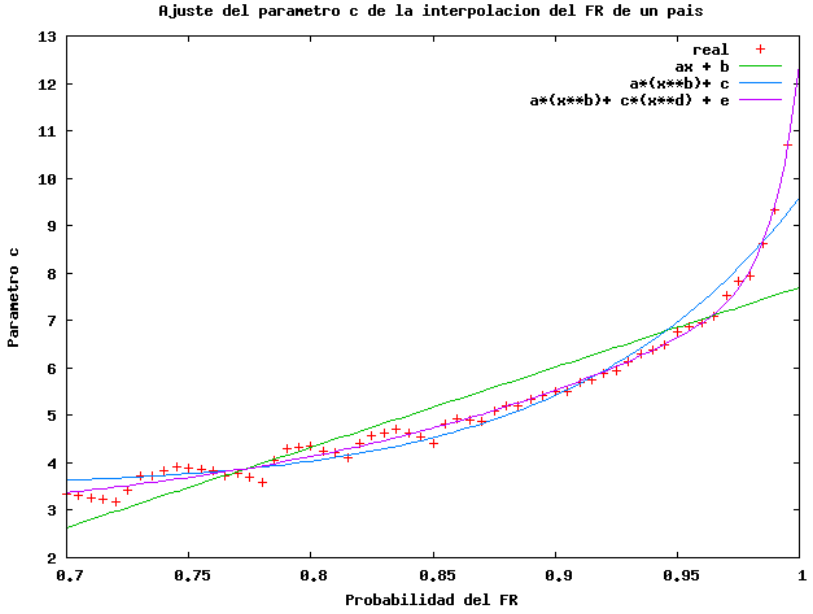


Figura 4.23: Ajuste del parámetro C

El valor aproximado para los parámetros es:

$$\begin{aligned} C_a &= 4{,}44933344559434 \\ C_b &= 91{,}1868445461335 \\ C_c &= 5{,}20837579169743 \\ C_d &= 5{,}9455021683503 \\ C_e &= 2{,}74960596558285 \end{aligned}$$

País con un único soldado

La función anteriormente descrita aproxima correctamente el factor de riesgo pero en el caso de un país con un único soldado es inexacta. Esto se debe a que es un caso especial en el que las tiradas del defensor se realizan siempre con un solo dado.

La solución es aproximarlo con otra función que sólo se aplicará en este caso. El número de soldados es siempre uno, por lo que la nueva función depende sólo de la probabilidad de éxito.

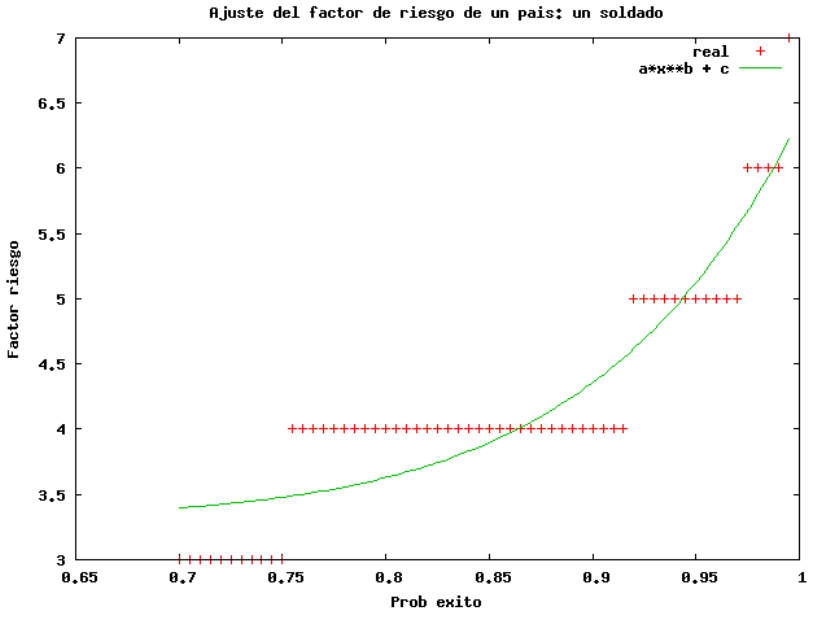


Figura 4.24: Factor de riesgo de un país con un soldado

La figura 4.24 muestra el ajuste con una función polinomial. Se puede comprobar que aplicando un redondeo el resultado es prácticamente exacto para cualquier probabilidad de éxito, por lo que se utiliza la ecuación 4.3.

$$FR = aProb^b + c \tag{4.3}$$

El valor aproximado para los parámetros es:

$$\begin{aligned} a &= 3{,}05795296862387 \\ b &= 10{,}232514709372 \\ c &= 3{,}31699071576838 \end{aligned}$$

Error de la aproximación

Antes de utilizar la función interpolada hay que comprobar que el error que comete en la aproximación es asumible.

En el caso de un solo soldado el error es nulo en la mayoría de los casos, como muestra la gráfica 4.25. El error de un soldado en el límite de los cambios es causado por el redondeo y no tiene ninguna relevancia. El único fallo de la función es en porcentajes de riesgo cercanos a uno, en concreto en el intervalo $[0{,}99 - 1)$, por lo que esta función es totalmente válida para porcentajes entre 70 % y 99 %.

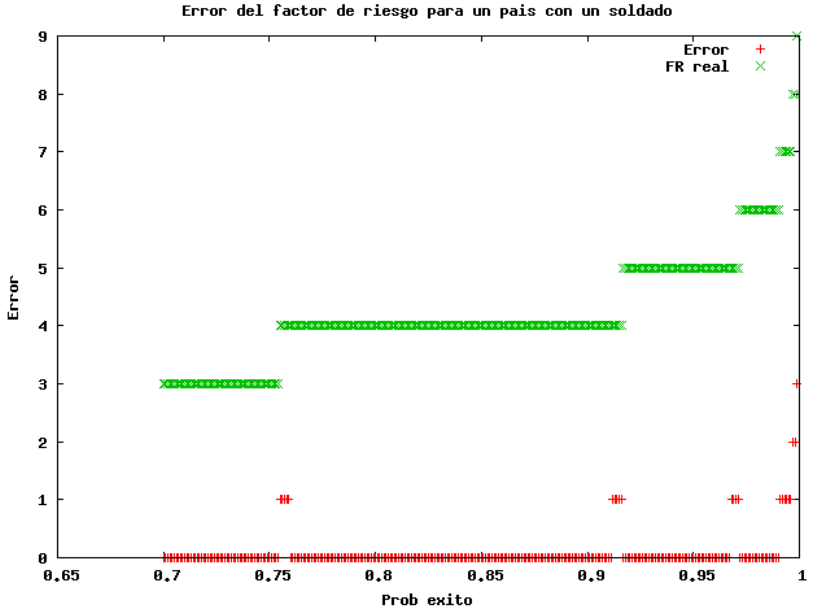


Figura 4.25: Error del factor de riesgo de un país con un soldado

En el resto de casos, la figura 4.26 muestra el error en los datos recogidos.

Como se puede ver, hay un error generalizado de un soldado cuando el número de soldados es bajo. Esto es importante, puesto que ocurre para cualquier porcentaje de riesgo para los casos más probables (no suele haber más de 10 soldados por país). Aun así, el error es en todos los casos añadir un soldado más, lo cuál hará que el agente tome un juego ligeramente menos arriesgado y resulta asumible.

El resto de errores pueden despreciarse, ya que los de ± 1 soldado son debidos al redondeo y no tienen importancia. Los errores de $+2$ ó $+3$ soldados sólo ocurren para porcentajes

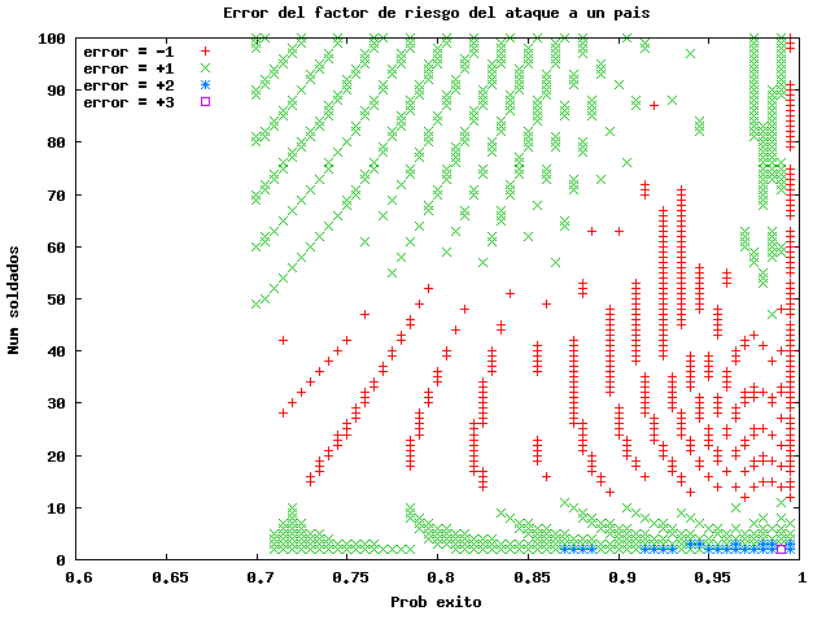


Figura 4.26: Error del factor de riesgo de un país

de riesgo bastantes altos y en casos muy aislados.

Factor de riesgo de una lista de ataques

Una lista de ataques está compuesta de N ataques cada uno con E_i soldados enemigos.

Para calcular su factor de riesgo hay dos opciones:

1. Dada una lista determinar qué ocurre si se añade otro país, al igual que en el coste de un lista se sumaban los costes independientes.

2. Determinar el factor de riesgo para toda la lista directamente, sin partir del cálculo del factor de riesgo para cada ataque que la compone.

El factor de riesgo viene determinado por la probabilidad de victoria del ataque continuado a todos los enemigos. Dado que dicha probabilidad se calcula como la probabilidad encadenada de diversas tiradas de dado, en todos los casos en los que se realicen las mismas tiradas, el factor de riesgo será el mismo.

En el caso de N soldados repartidos entre 2 países hay N - 1 posibles repartos: $[N - m, m], \forall m \in [1, N - 1]$, por lo que se realizarán tiradas hasta matar a $N - m$ soldados y despues a m soldados. Al variar m solamente pasan tiradas de un país a otro sin variar su número. Por lo tanto, *no influye el reparto de los soldados*, excepto en el caso en que el número de soldados de un país vale 1, ya que cambian las tiradas realizadas, que se puede despreciar.

Sin embargo, sí influye el número de países, ya que tendrán que realizarse más tiradas y habrá más tiradas de tipo (3, 1).

Por lo tanto, *el factor de riesgo de una lista depende del número de soldados y de países*, por lo que hay que hacer el cálculo para toda la lista.

Aproximación lineal del factor de riesgo de una lista

El factor de riesgo de una lista se calcula a partir del número de soldados N_s, el número de países N_p y el porcentaje de riesgo X. Como el factor de riesgo de un país es el mismo caso que una lista de un país, se puede partir de este cálculo. La gráfica 4.27 muestra datos de la evolución del factor de riesgo al añadir países.

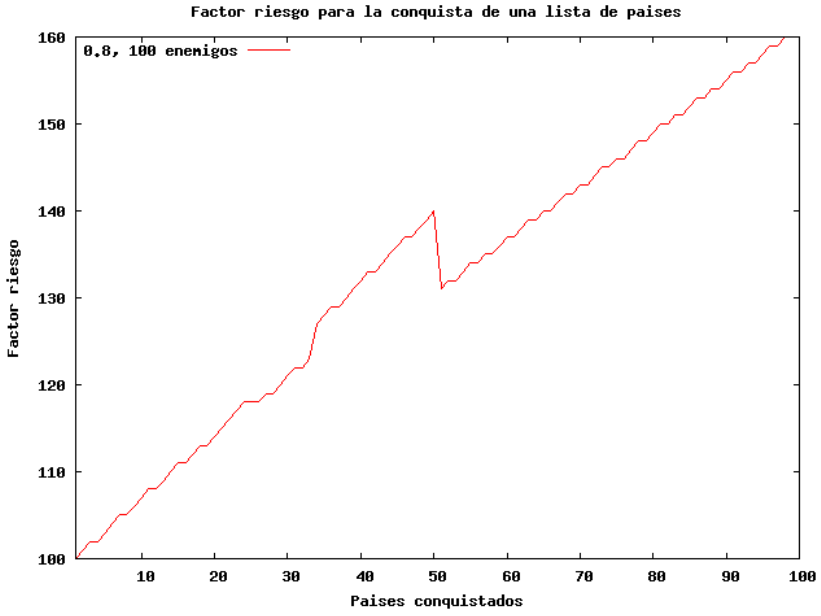


Figura 4.27: Factor de riesgo de la lista respecto a su número de países

Se puede ver que el incremento parece lineal, aunque hay saltos y variaciones en la

pendiente en $N_p = \frac{N_s}{2}, \frac{N_s}{3}, \ldots$ cada vez más pequeños. Las ondulaciones en la función se deben al redondeo al número entero más cercano.

La figura 4.28 muestra el ajuste con la ecuación lineal 4.4.

$$FRlista = FRpais(N_s, X) + A * (N_p - 1) \tag{4.4}$$

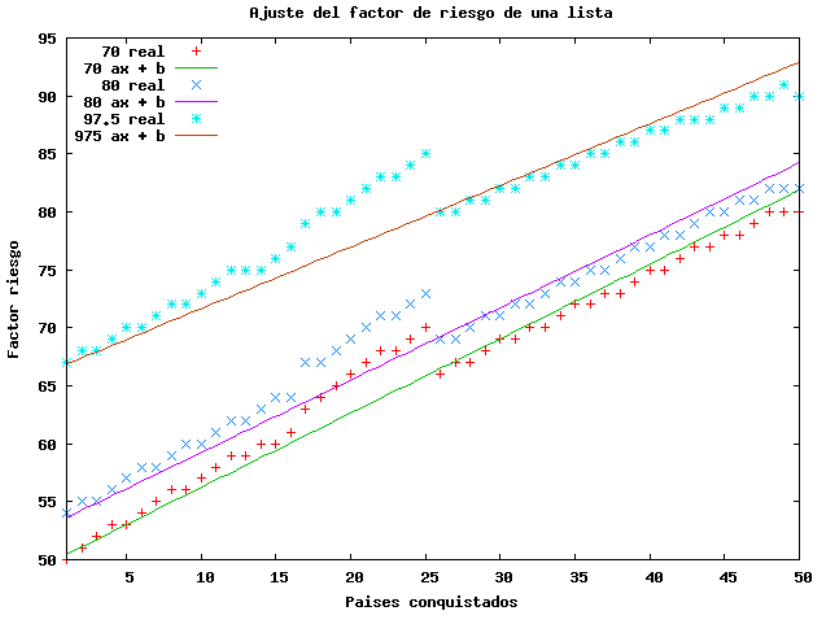


Figura 4.28: Ajuste del factor de riesgo de una lista

Ajuste de la pendiente A

En la figura 4.29 se ve que la función que mejor aproxima A es:

$$A = \frac{A_1}{x + A_2} + A_3$$

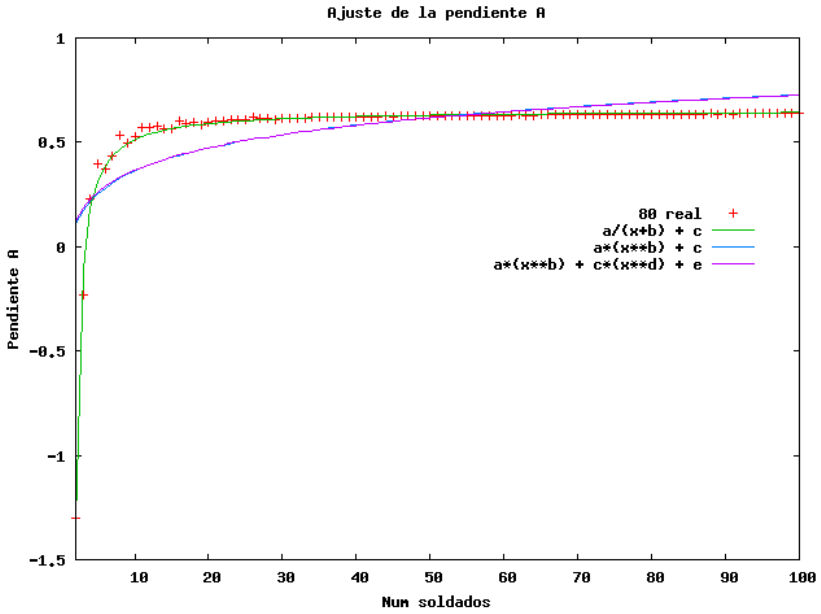


Figura 4.29: Ajuste de la pendiente A del factor de riesgo de una lista

Ajuste del parámetro A_1

En la figura 4.30 se ve el ajuste del parámetro A_1 con la función:

$$A_1 = A_{1a}x^{A_{1b}} + A_{1c}$$

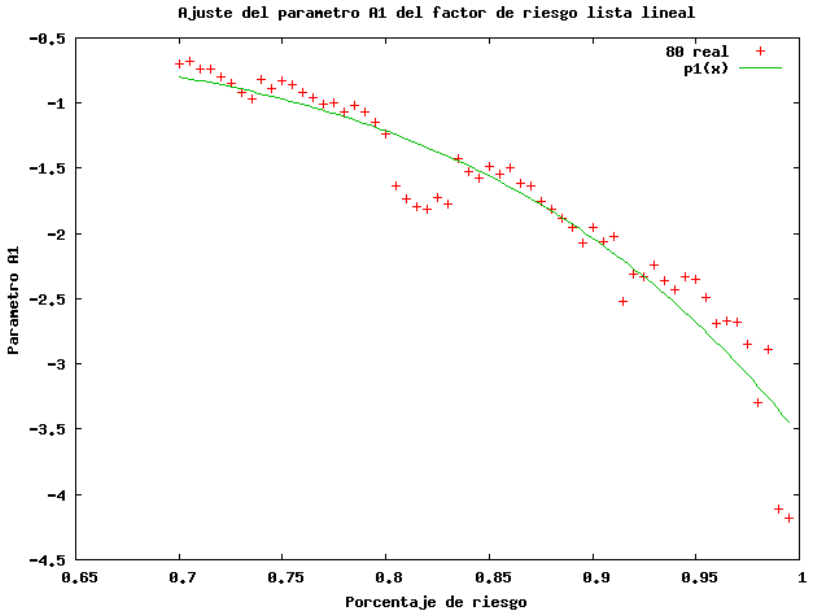


Figura 4.30: Ajuste del parámetro A_1 del factor de riesgo de una lista

El valor aproximado para los parámetros es:

$$\begin{aligned} A_{1a} &= -3{,}04800015484555 \\ A_{1b} &= 6{,}49722555838492 \\ A_{1c} &= -0{,}497511301028481 \end{aligned}$$

Ajuste del parámetro A_2

En la figura 4.31 se ve el ajuste del parámetro A_2 con la función:

$$A_2 = A_{2a}x^{A_{2b}} + A_{2c}$$

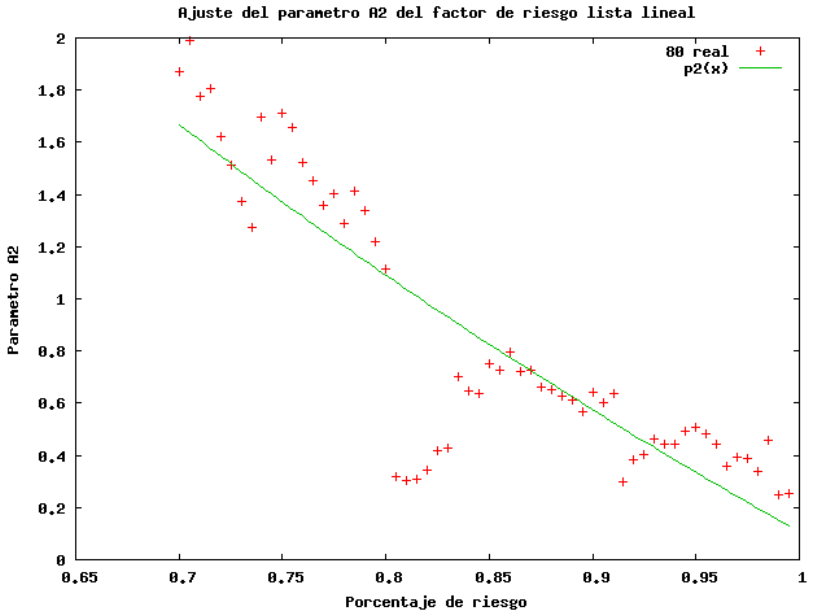


Figura 4.31: Ajuste del parámetro A_2 del factor de riesgo de una lista

El valor aproximado para los parámetros es:

$$\begin{aligned} A_{2a} &= -40{,}5788618811924 \\ A_{2b} &= 0{,}110129652120635 \\ A_{2c} &= 40{,}6848646543789 \end{aligned}$$

Ajuste del parámetro A_3

En la figura 4.32 se ve el ajuste del parámetro A_3 con la función:

$$A_3 = A_{3a}x^{A_{3b}} + A_{3c}$$

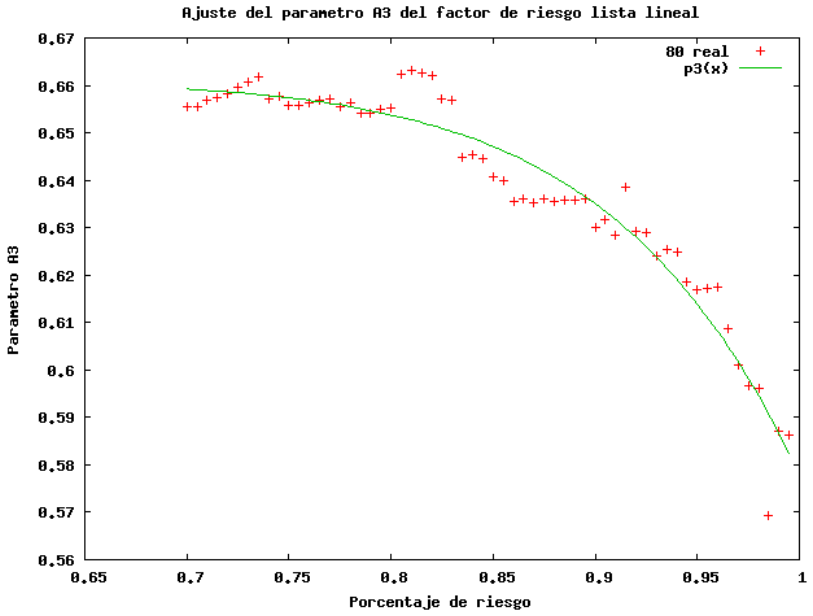


Figura 4.32: Ajuste del parámetro A_3 del factor de riesgo de una lista

El valor aproximado para los parámetros es:

$$\begin{aligned} A_{3a} &= -0{,}0828621653765624 \\ A_{3b} &= 11{,}0936056856326 \\ A_{3c} &= 0{,}660784405274512 \end{aligned}$$

Error de la aproximación del factor de riesgo de una lista

Se comparan los resultados obtenidos por la función interpolada con los datos reales y se mide el error de la aproximación.

En casos normales el error es bastante reducido. En la la gráfica 4.33 se muestra el error cometido para listas de 3 a 100 soldados repartidos entre 3 países con un porcentaje de riesgo del 80 %. Este error está entre -1 y 2 soldados pero en la mayoría de los casos es 0 ó -1

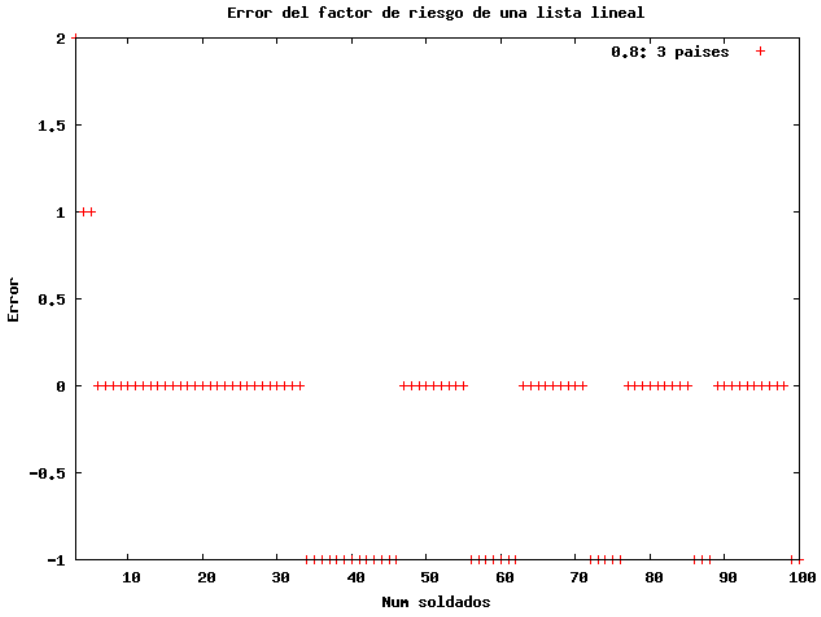


Figura 4.33: Error del factor de riesgo lista lineal. Caso típico

En otros casos con más países y un porcentaje de riesgo superior a 90 % la función interpolada funciona bastante peor. En la gráfica 4.34 se puede ver que en el caso de 100 soldados repartidos entre 50 países, con 99,5 % de probabilidad el error llega a -10 soldados.

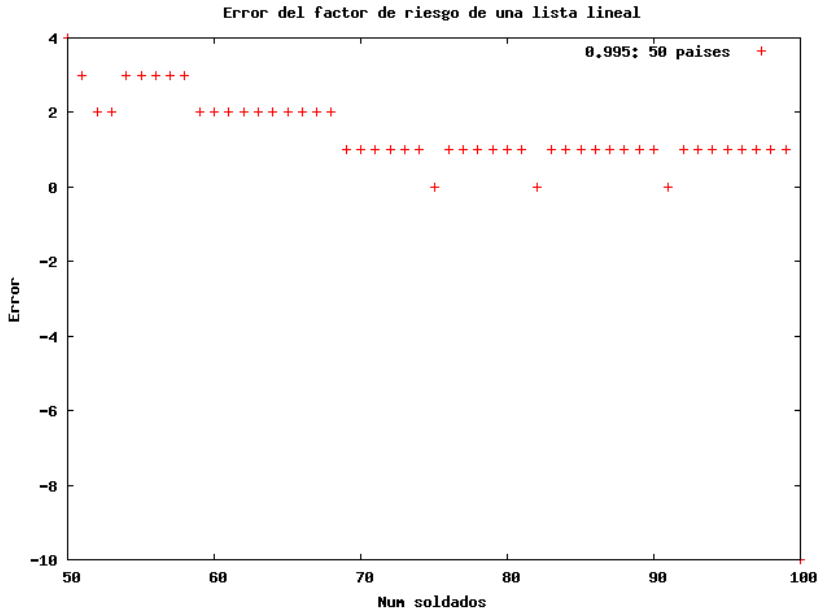


Figura 4.34: Error del factor de riesgo lista lineal. Caso extremo

La conclusión es que el resultado no es perfecto pero en los casos normales en los que se va aplicar la función es bastante buena. Dado que el tiempo del cálculo se reduce notablemente el error en la aproximación es totalmente asumible.

Factor de riesgo de un árbol de ataques

En los árboles de ataque, los nodos con ramificación deben decidir cuántos soldados pasar a cada hijo (el siguiente país) cuando lo conquistan, garantizando la probabilidad de éxito de cada rama. No se busca que el árbol completo tenga la probabilidad de éxito deseada, sino únicamente para cada rama por separado.

Para calcular su factor de riesgo, un árbol se considera como un conjunto de listas de ataque: un tronco con ramas secundarias que parten de él. Además, antes de llegar a conquistar ningún país el nodo puede atacar todos sus hijos hasta que queden sólo dos soldados en cada uno, por lo que todos los hijos forman parte del *tronco* del nodo. La figura 4.35 muestra un esquema del árbol dividido de este modo.

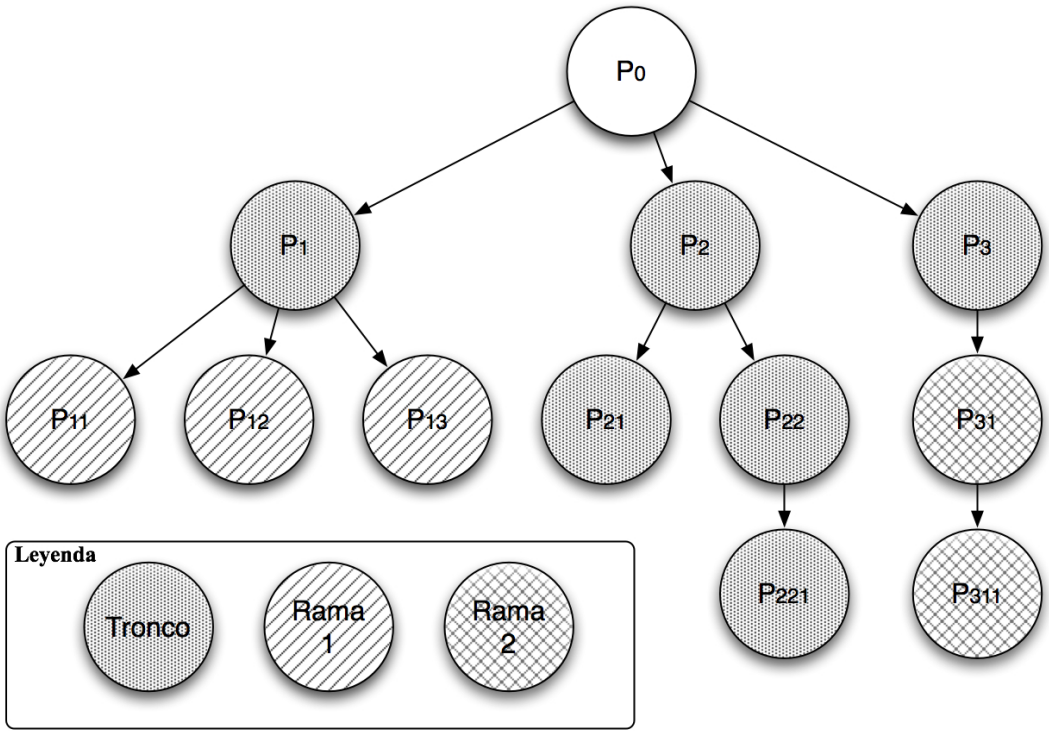


Figura 4.35: División de un árbol de ataques en tronco y ramas secundarias

El factor de riesgo del nodo será igual a la suma del factor de riesgo de todas las listas que lo componen. Para elegir en un nodo se escoge el que minimiza el factor de riesgo total.

4.3.7. Aproximación de la probabilidad de éxito de los ataques

Para realizar esta aproximación se sigue el mismo método que en el factor de riesgo.

Probabilidad de éxito de un ataque

La probabilidad de éxito de un ataque simple depende del número de soldados atacantes S_a y del número de soldados defensores S_d En la figura 4.36 se ve el ajuste de la probabilidad de éxito con la función:

$$Pe = \frac{1}{C + Ae^{-S_a B}}$$

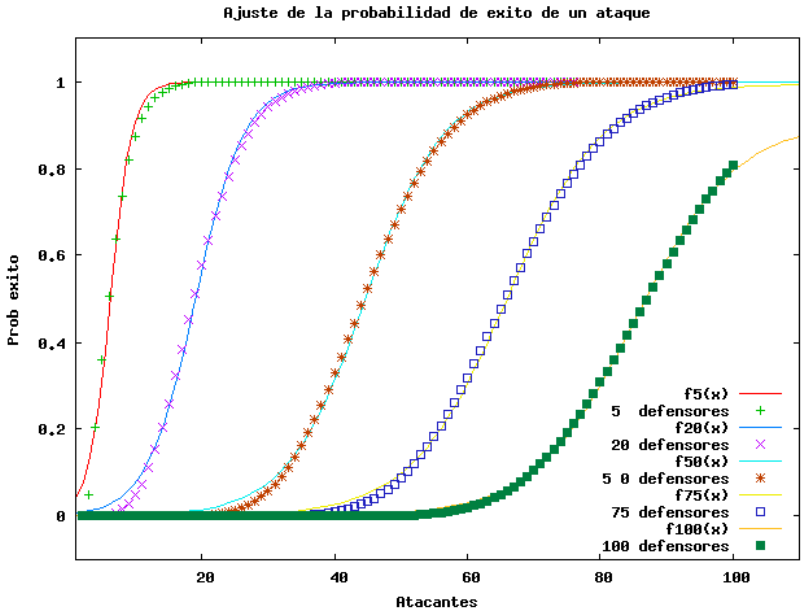


Figura 4.36: Ajuste de la probabilidad de éxito

A, B y C son parámetros que dependen de S_d.

Ajuste del parámetro A

En las figuras 4.37 y 4.38 se ve el ajuste de la probabilidad de éxito con la función:

$$A = \begin{cases} A_{11} S_d^{A_{12}} + A_{13} S_d^{A_{14}} + A_{15} & \text{si } S_d \leq 60, \\ A_{21} S_d^{A_{22}} + A_{23} S_d^{A_{24}} + A_{25} & \text{si } S_d > 60. \end{cases}$$

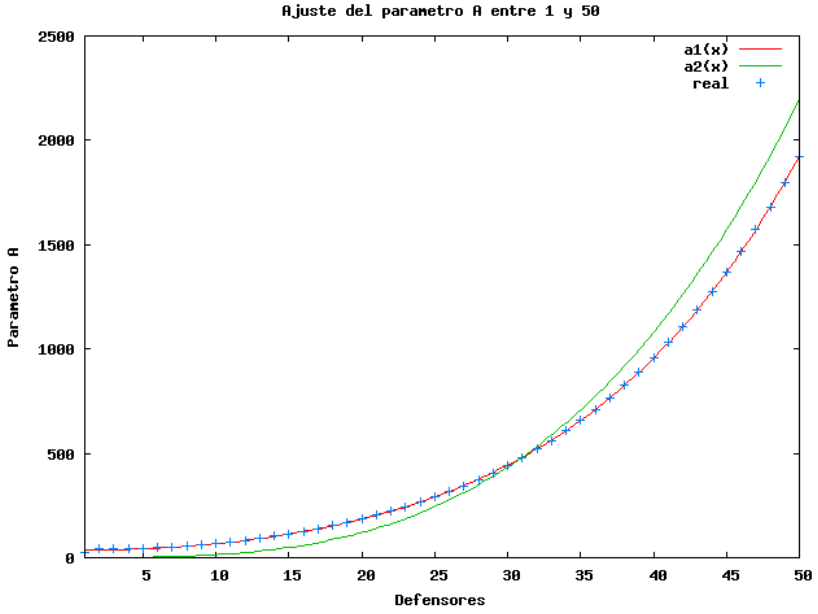


Figura 4.37: Ajuste del parámetro A de la probabilidad de éxito entre 0 y 50

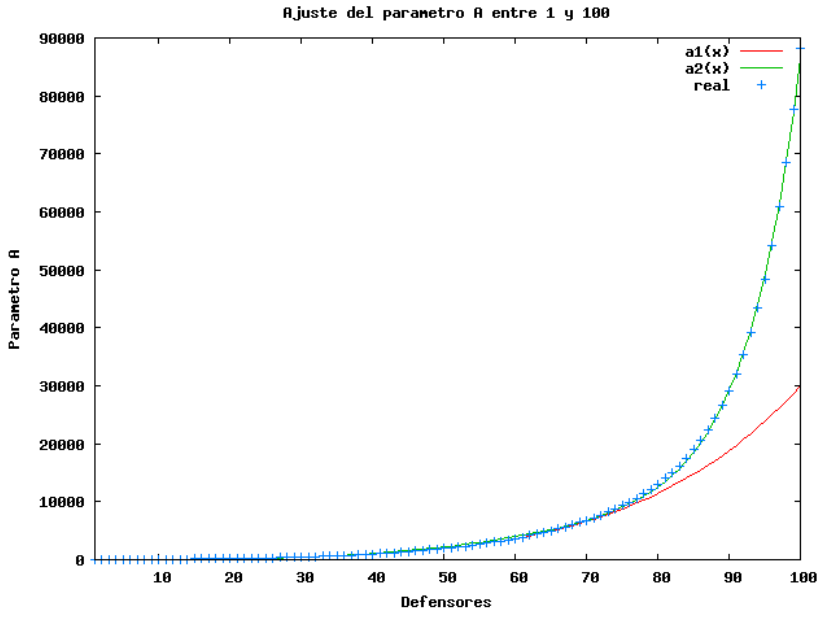


Figura 4.38: Ajuste del parámetro A de la probabilidad de éxito entre 50 y 100

El valor aproximado de los parámetros es:	$A_{11} = 0{,}207435138396346$	$A_{21} = 0{,}00895981364081855$
	$A_{12} = 2{,}17075563701387$	$A_{22} = 3{,}17129800108703$
	$A_{13} = 4{,}99125560335143 \times 10^{-6}$	$A_{23} = 3{,}73216412719542 \times 10^{-24}$
	$A_{14} = 4{,}85273028531451$	$A_{24} = 14{,}1269463122623$
	$A_{15} = 35{,}9895230961728$	$A_{25} = 1{,}01397468694002$

Ajuste del parámetro B

En la figura 4.39 se ve el ajuste de la probabilidad de éxito con la función:

$$B = \frac{B_1}{1 + B_2 S_d^{B_3}}$$

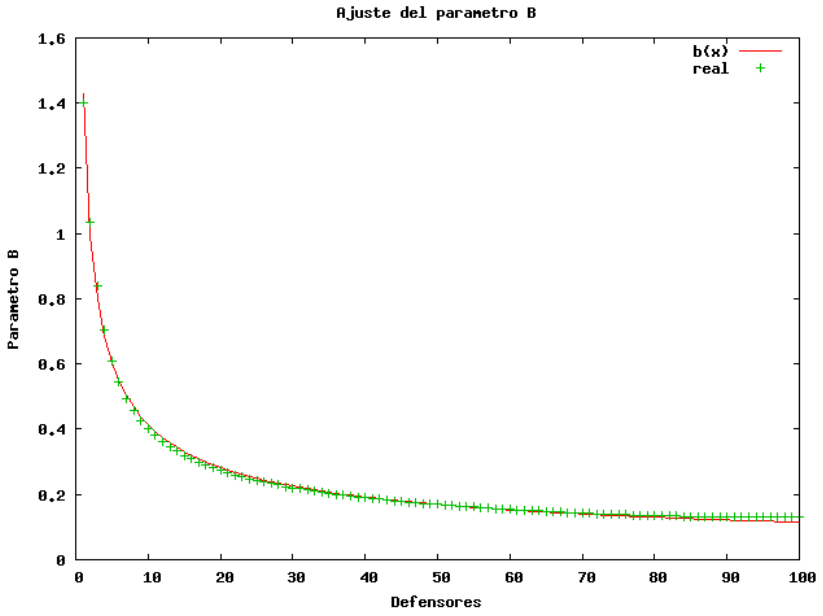


Figura 4.39: Ajuste del parámetro B de la probabilidad de éxito

El valor aproximado para los parámetros es:

$$\begin{aligned} B_1 &= 16{,}6258449454357 \\ B_2 &= 10{,}6463588448413 \\ B_3 &= 0{,}564839545509917 \end{aligned}$$

Ajuste del parámetro C

En la figura 4.40 se ve el ajuste de la probabilidad de éxito con la función:

$$C = C_1 S_d^{C_2} + C_3$$

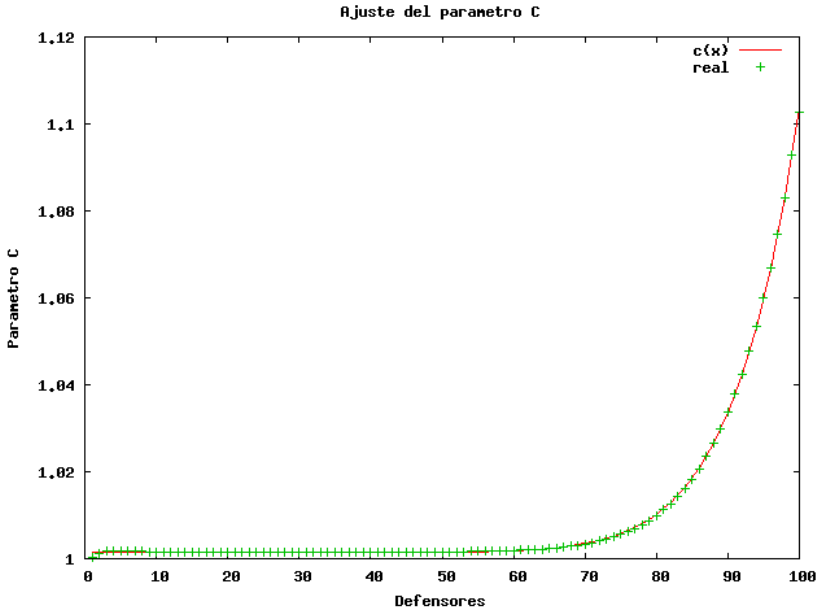


Figura 4.40: Ajuste del parámetro C de la probabilidad de éxito

El valor aproximado para los parámetros es:

$$C_1 = 1{,}03958106418479 \times 10^{-23}$$
$$C_2 = 10{,}9959068545343$$
$$C_3 = 1{,}00146321805143$$

Error de la aproximación de la probabilidad de éxito de un ataque

La gráfica 4.41 muestra el error según el número de atacantes para el caso de 20 defensores.

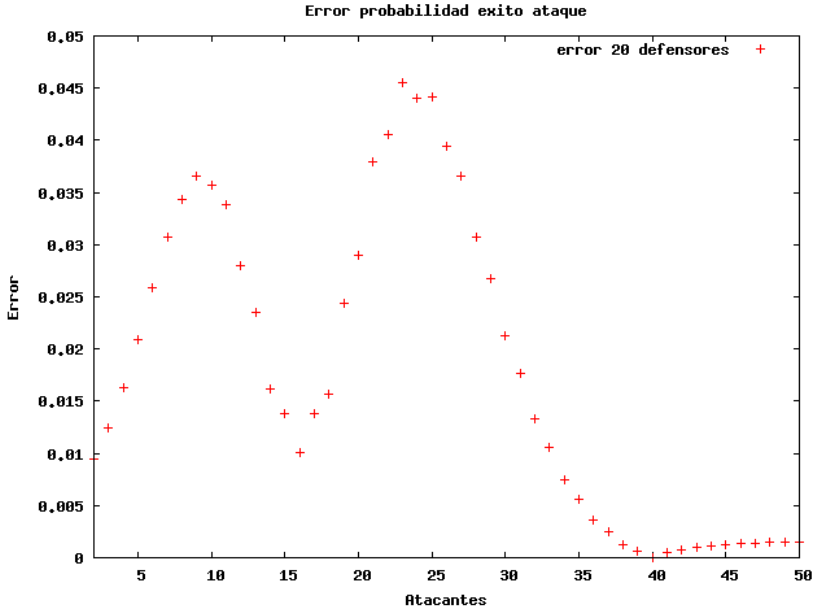


Figura 4.41: Error de la probabilidad de éxito de un ataque

Se puede ver que el ajuste no es perfecto pero la desviación máxima es de aproximadamente un 5 % de probabilidad por lo que la interpolación está bastante bien conseguida.

Probabilidad de éxito de una lista de ataques

Se calcula la probabilidad de éxito de una lista a partir de la probabilidad de éxito de los ataques que la conforman según la ecuación 4.5:

$$Pe(L) = \prod_{A_i \in L} Pe(A_i) \tag{4.5}$$

La ecuación 4.6 calcula el número de atacantes de A_i:

$$a_i = a_{i-1} - Coste(d_{i-1}) \tag{4.6}$$

Probabilidad de éxito de un árbol de ataques

Tal y como ha sido calculada la probabilidad de éxito de una lista puede calcularse del mismo modo la probabilidad de éxito de un árbol.

La única particularidad es que, al contrario que en la lista de ataques, hay que conocer el número de tropas que se trasladan tras cada ataque. Esta decisión vendrá dada por el número de soldados que se desea que acaben en cada nodo del árbol, lo que se decide en la formación del plan.

4.4. Desarrollo del Algoritmo

En esta sección se decide el algoritmo que se implementará en el agente. Se comienza con una propuesta inicial, seguida de la representación del estado de la partida y del plan que se utilizará en el algoritmo para finalizar con una arquitectura general del algoritmo.

4.4.1. Aplicación propuesta en este proyecto

En los estudios teóricos se concluyó que es necesario que haya algún tipo de plan (véase la sección 4.3.1, página 52), aunque aun no se ha detallado formalmente qué es un plan y, por ahora, es un elemento de información que se determina al principio del turno para sincronizar las decisiones que se tomarán en cada una de sus fases.

Por otro lado, se decidió que la opción más prometedora era continuar el trabajo de búsqueda de las acciones a realizar durante el turno (véase la sección 2.5, página 19). Por lo tanto, la propuesta es una búsqueda del mejor plan, guiada por heurísticas lo más objetivas posibles.

Habrá que determinar el algoritmo de búsqueda utilizado, las heurísticas que guíen la búsqueda (escogiendo los planes que son examinados) y las que valoran los planes (para seleccionar el mejor).

La figura 4.42 muestra un esquema de la propuesta realizada.

- **Búsqueda de planes**: Algoritmo de búsqueda que decide el plan escogido para el turno.
- **Valoración del plan**: Valora el plan dando la probabilidad de victoria en caso de aplicar ese plan.
- **Refinamiento del plan**: Mejora el plan decidiendo aspectos secundarios. Se aplica durante la ejecución.
- **Ejecución del plan**: Transforma un plan en acciones concretas sobre el tablero de juego.

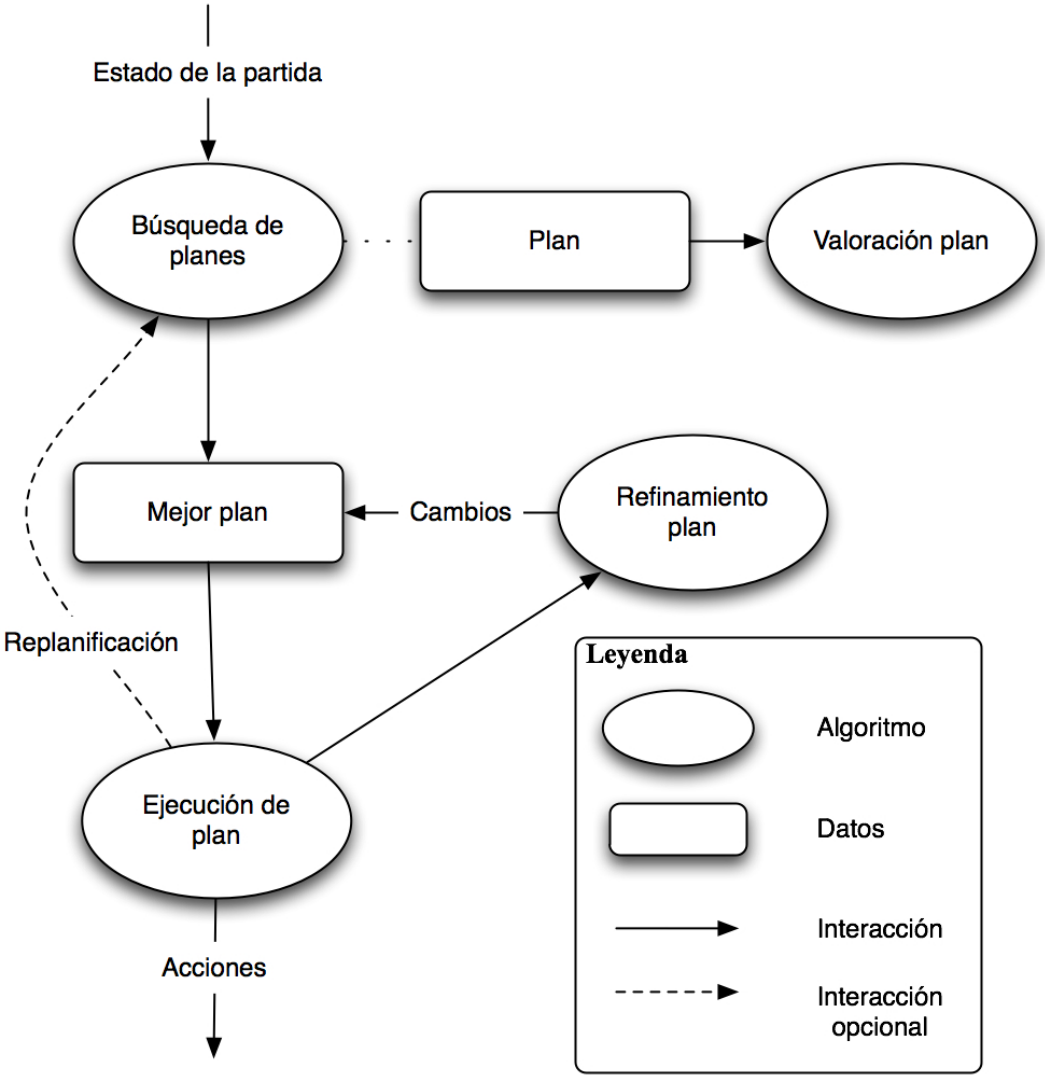


Figura 4.42: Esquema de búsqueda planteado para el proyecto

4.4.2. Representación del estado de la partida

El estado de la partida es la entrada para que el agente realice su turno, por lo que deberá contener toda la información necesaria para escoger las mejores decisiones.

Dentro de esta información pueden considerarse múltiples posibilidades. Por ejemplo, en una partida real entre jugadores humanos podrían incluirse en el estado las acciones que no se realizan propiamente en el juego: gestos, afirmaciones, etc, lo que podría ser información útil para poder predecir sus acciones. En este casos habrá que ceñirse a acciones que ocurren sobre el tablero de juego.

Se definen dos tipos de información del estado: estática y dinámica. La información estática es aquella que no varía durante la partida, mientras que la información dinámica cambiará durante el turno del jugador o de los rivales. El interés en esta clasificación está en que los análisis realizados sobre la información estática podrán realizarse una sola vez y aprovecharse durante toda la partida.

- Información estática
  - Tablero
    - Paises y sus conexiones: grafo conexo no dirigido
    - Continentes: subgrafo de paises y refuerzos aportados
  - Orden de los turnos de los jugadores
  - Reglas concretas del juego
    - Valor de las cartas
- Información dinámica
  - Soldados
    - Número de soldados en cada país $[1 - \infty)$
    - Jugador propietario de cada país
  - Cartas
    - Número de cartas que tiene cada oponente
    - Cartas propias

4.4.3. Representación del plan

La representación escogida para el plan es fundamental para definir toda la arquitectura del agente. Tras descartar el plan maestro (véase la sección B.3, página 208), se describe la opción escogida: el boceto de plan.

Definición del boceto de plan

Al contrario que el plan maestro, el boceto de plan no tiene toda la información de las acciones que se realizarán sino un esquema de lo que quiere conseguir. A la hora de ejecutar las acciones se toman decisiones siguiendo el objetivo marcado en el plan. De este modo se imita el modo de pensar de un jugador humano: primero decide qué países quiere conquistar en el turno y, en base a eso, reparte sus tropas iniciales y va realizando los ataques comprobando que tiene suficientes probabilidades de éxito.

El proceso de búsqueda se simplifica bastante, ya que sólo decide los ataques a realizar. Que algunas acciones tengan un resultado indeterminado ya no es un problema, puesto que no es necesario calcular lo que se hará en cada caso, sino que simplemente calculará la acción a tomar en el momento en el que se sepa el resultado de la anterior acción.

La tabla 4.2 contiene los cálculos o decisiones que deben tomarse para decidir las acciones del jugador clasificados en tres categorías, dependiendo de si se deciden al crear el plan, durante su análisis o en su ejecución.

Búsqueda del plan	Análisis del plan	Ejecución del plan
Ataques	Reparto de soldados	Orden de los ataques
	Guía de cuantos soldados pasarán en los ataques	Soldados que pasan en cada ataque
	Fortificaciones previstas	Fortificaciones realizadas

Tabla 4.2: Clasificación de cálculos para la ejecución de un plan

Por lo tanto, el boceto de plan está formado por los árboles de ataque y cierta información adicional relativa al movimiento de los soldados durante el turno, añadida durante su análisis. La figura 4.43 muestra un ejemplo de un boceto de plan.

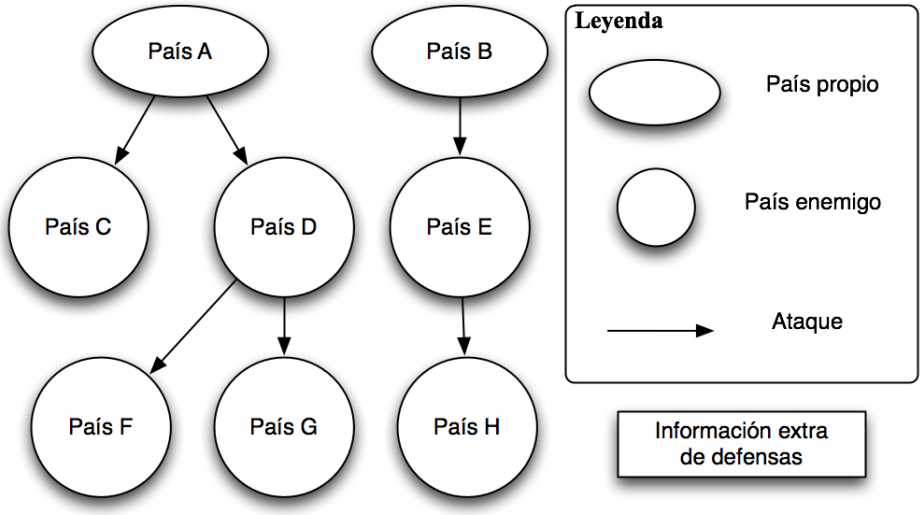


Figura 4.43: Diagrama de un boceto de plan

El mayor problema es que se pierde la posibilidad de realizar un ataque para debilitar al rival, sin el objetivo de conquistarlo. Esto incluye también la posibilidad de conquistar un país rival atacando desde varios países propios, ya que todos los ataques excepto el último serían ataques para debilitarlo. Aun así es una simplificación asumible, puesto que este tipo de ataques es muy infrecuente y puede despreciarse sin esperar grandes pérdidas en el rendimiento.

Estado base del boceto de plan

Para poder valorar un plan es necesario estimar el estado de la partida tras su ejecución, esto es, el estado base del plan. Como los ataques tienen un resultado aleatorio hay que realizar algunas suposiciones:

1. Todos los ataques se realizarán con éxito: todos los países del plan pasan a ser del jugador. Esto produce un gran error en caso de que algún ataque no tenga éxito, por lo que es necesario garantizar que la probabilidad de éxito de los ataques está por encima de un umbral mediante el factor de riesgo 4.3.6 de los árboles de ataque que componen el plan.

2. El número de soldados que sobreviven a los ataques realizados: para cada nodo del árbol de ataques el número de supervivientes es el número de soldados destinado al ataque (su factor de riesgo) menos el coste del ataque y el factor de riesgo de sus hijos.

Por lo tanto, para un nodo P_i con hijos $P_{i1}, \ldots, P_{iH}$ el número de supervivientes es: $S_i = FR_i - (C_i + \sum_{j=0}^{H}(FR_{ij}))$

En este estado no se ha decidido aún la información de defensas, por lo que los soldados supervivientes no pueden situarse y se considera que el jugador tiene un soldado en cada país. Para obtener el estado completo habrá que aplicar la información de defensas sobre el estado base.

Información de defensas del boceto de plan

La información de defensas del plan determina dónde se situarán los soldados supervivientes tras la realización del plan y el camino que seguirán durante su ejecución, determinando las acciones que hay que llevar a cabo en la ejecución del plan. Estas acciones permiten las siguientes decisiones:

- Soldados obtenidos al inicio del turno: pueden pasar a cualquier país.
- Soldados utilizados en los ataques: pueden quedarse en el país atacante o pasar a nodos inferiores en el árbol de ataques.
- Fortificaciones: Al final del turno pasar soldados de un país a sus vecinos.

Se definen *grupos de soldados* que pueden asignarse a un conjunto de países. Además del grupo de soldados posicionables al inicio del turno, hay un grupo de soldados por cada país poseído al finalizar el turno. Cada grupo de soldados se define por:

1. País del grupo: donde se encuentran los soldados al inicio del turno, excepto en el grupo de soldados posicionables.
2. Número de soldados

3. Países asignables.

Número de soldados El número de soldados en los grupos de países que no intervienen en los ataques es igual al número de soldados sobre él menos 1, para garantizar que siempre queda un soldado en ese país. Esto es una simplificación, ya que ese soldado podría ser trasladado a otro país con una fortificación si otro grupo asigna un soldado a su país. Sin embargo, con la representación escogida se evitan posibles informaciones de defensas erróneas en las que se decide no dejar ningún soldado en algún país.

El número de soldados en los grupos de países que intervienen en los ataques es más complejo. En el caso del nodo relativo al padre del ataque A_i su número de soldados es el número de soldados supervivientes menos uno como muestra la ecuación 4.7.

$$SoldG_i = FR(A_i) - \sum_j (Coste(A_{ij}) + FR(A_{ij})) - 1 \tag{4.7}$$

Países asignables Los soldados posicionables pueden asignarse a cualquier país.

Los soldados que no participan en ningún ataque pueden asignarse a su propio país o a sus vecinos.

Los soldados que participan en un árbol de ataques pueden asignarse a su propio país, a los países inferiores dentro del árbol o a los vecinos de todos estos países.

Camino de las tropas El camino de las tropas decide para los soldados de un grupo asignados a un país final, el camino que seguirán durante el turno.

Los árboles de ataque marcan el camino para llevar las tropas a través de los ataques, pero al final del turno pueden realizarse fortificaciones de un país a sus vecinos por lo que durante el posicionamiento y los ataques pueden llevarse al país final o a cualquiera de sus vecinos.

Se define *el país desde el que se atiende la defensa* como aquel al que se llevan los soldados para posteriormente trasladar los soldados al país destino mediante una fortificación.

Evidentemente, si la amenaza se atiende desde el mismo país al que se deben llevar los soldados no es necesario realizar ninguna fortificación.

El grupo de soldados posicionables puede asignarse a un país a través de él o de cualquiera de sus vecinos.

Los grupos de países que participan en ataques pueden asignarse a través de cualquier nodo inferior en el árbol que sea vecino del país final o él mismo. Esto significa que si el país final no forma parte del árbol, sólo podrá atenderse su defensa desde un vecino.

Los grupos de países que no participan en ataques solamente pueden asignarse desde ese mismo país.

Ejemplo de información de defensas

La figura 4.44 muestra un plan en el que el jugador tiene tres países P_1, P_2 y P_3 y quiere conquistar otros cuatro países: P_{11}, P_{111}, P_{12}, P_{21}. Además de las conexiones entre los países atacados y atacantes, son vecinos: $[P_3, P_{111}]$ y $[P_3, P_{21}]$. Las conexiones que haya con el resto de países enemigos son despreciables.

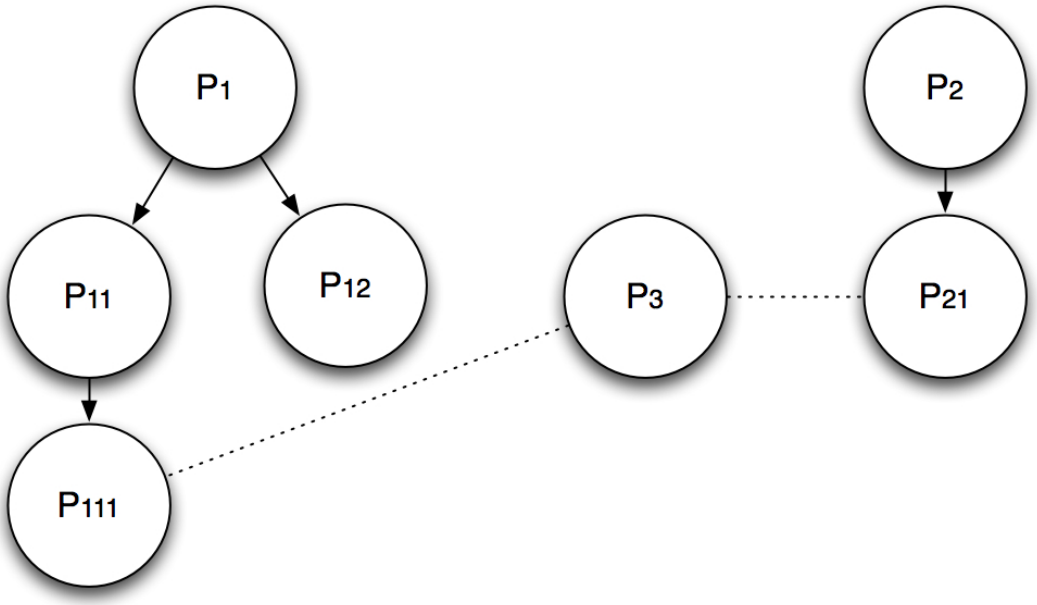


Figura 4.44: Ejemplo de información de defensas

La información de defensas asociada al plan definido esta compuesta por 8 grupos de soldados: uno del posicionamiento inicial de soldados G_p y uno por cada país G_1, G_2, ...

Las asignaciones posibles de cada grupo son:

- G_p: Todos los países
- G_1: P_1, P_{11}, P_{111}, P_{12} y P_3
- G_2: P_2, P_{21}, P_3
- G_3: P_3, P_{111}, P_2
- ...

En cuanto al camino de las tropas, algunos ejemplos de países asignables son:

- $[G_p, P_3]$: P_3, P_{111} y P_{21}
- $[G_1, P_3]$: P_3, P_{111}
- $[G_p, P_{111}]$: P_{111}, P_{11} y P_3
- $[G_1, P_{111}]$: P_{111} y P_{11}
- ...

4.4.4. Algoritmo de búsqueda del mejor plan

Hay muchos algoritmos de búsqueda que pueden utilizarse para buscar el mejor plan. Entre la gran variedad de posibilidades se ha escogido el algoritmo del mejor primero, *best-first search*.

Este algoritmo comienza analizando el plan vacío (no realiza ningún ataque) y tras analizar cada plan se generan los planes que tienen los mismos ataques que él y uno más. Se define el plan padre de un plan como aquél que lo ha generado. Como sólo se

generan los hijos de los planes después de analizarlos, todo plan excepto el plan vacío dispone del análisis de la posición realizado para su plan padre.

Al generar los hijos de un plan se realiza una poda anticipada de todos aquellos que se consideran inviables por no tener suficientes soldados según el factor de riesgo.

La prioridad del plan (véase la sección 4.5.8, página 146) es la heurística que elíge el siguiente plan a analizar.

Para evitar analizar varias veces el mismo plan se utiliza una tabla de transposición de estados en la que se guardan los planes ya valorados.

La búsqueda se termina cuando no quedan planes viables que analizar o si, tras analizar el último plan, se ha superado el tiempo máximo de búsqueda y se han revisado suficientes planes. En este proyecto se limita el tiempo del plan a *60 segundos* siempre que se hayan revisado al menos *30 planes*.

Ventajas Este modelo es muy robusto, ya que va revisando ataques a partir de la raíz. Si la prioridad del plan está bien ajustada no se saltará planes interesantes. Además, al analizar un plan puede reutilizar cálculos realizados para su plan padre.

Inconvenientes Lo peor es que siempre empieza con el plan vacío. En casos en los que el mejor plan incluya muchos ataques, es probable que no se llegue tan lejos en el árbol de búsqueda. Cuanto más refinada este la prioridad del plan irá más directamente a por el mejor, pero siempre tendrá una longitud de ataque limitada.

Otros algoritmos, como la búsqueda en haz, mantendrían una población de planes interesantes e irían explorando esas zonas del espacio de búsqueda. Esto tendría la ventaja de poder inicializar las poblaciones lejos del plan vacío, ahorrando tiempo de análisis pero haría las heurísticas aún más complejas.

4.5. Desarrollo Heurísticas

Una vez decidido el algoritmo, es necesario completarlo con las heurísticas que determinan, para cada plan:

1. Valoración: estimación de la probabilidad de victoria.
2. Refinamiento: completa el plan escogido.
3. Prioridad: heurística que guía el algoritmo de búsqueda.
4. Ejecución: dado el plan completo, decide las acciones concretas sobre el tablero.

A continuación, se describen en distintas secciones todos estos algoritmos y las partes que los componen.

Los parámetros de configuración de los algoritmos seguirán la nomenclatura PC_{Nombre}. Se resumirán, junto con su valor recomendado, en una tabla al final de la explicación de cada algoritmo.

4.5.1. Valoración de un plan

Las heurísticas de valoración de un plan reciben un boceto de plan sin la información de defensas completa y todo el análisis realizado para su plan padre. El valor del plan será la probabilidad de victoria del jugador si realiza el plan.

Factores que influyen en la valoración de un jugador

Los factores considerados en la valoración son:

1. Fortaleza actual
   a) Número de soldados disponibles
   b) Cohesión de los soldados
   c) Movilidad
2. Fortaleza potencial
   a) Número de cartas
   b) Número de refuerzos previstos para el próximo turno
   c) Interés en el ataque por parte de otros jugadores
3. Otras consideraciones
   a) Posición estratégica
   b) Orden de los turnos de los jugadores

Valoración de forma estática y dinámica

Un algoritmo de valoración es dinámico si tiene en cuenta una predicción de los movimientos que se producirán en el tablero, y estático en caso contrario. Por lo tanto, cualquier valoración basada únicamente en calcular factores directamente sobre la posición será estática.

La valoración estática es más sencilla, pero tiene algunas limitaciones. A continuación se exponen algunos ejemplos de posiciones que no pueden comprenderse con una valoración dinámica.

Ejemplo 1 En la figura 4.45 se muestra una posición muy simplificada del Risk, con sólo tres países.

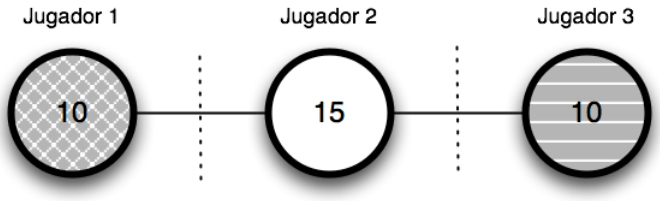


Figura 4.45: Ejemplo 1 de posición de valoración

¿Qué jugador va ganando? Si se atiende al número de soldados, como haría una valoración estática, se diría que el jugador 2 va ganando al tener más soldados. Sin embargo, está claro que sus 15 soldados tendrán que pelear con los 20 de sus enemigos y es, por lo tanto, el jugador con menos posibilidades de ganar la partida.

Aun así, hay algunas valoraciones estáticas que, atendiendo al número de países, número de fronteras, diferencia de soldados en las fronteras, etc. podrían corregir este efecto.

Ejemplo 2 En la figura 4.46 se puede ver un ejemplo en el cuál, a pesar de tener más soldados, más países y ventaja en sus dos fronteras, el jugador 2 está igual de perdido que en el ejemplo 1 debido a que tendrá que enfrentarse él solo a sus oponentes.

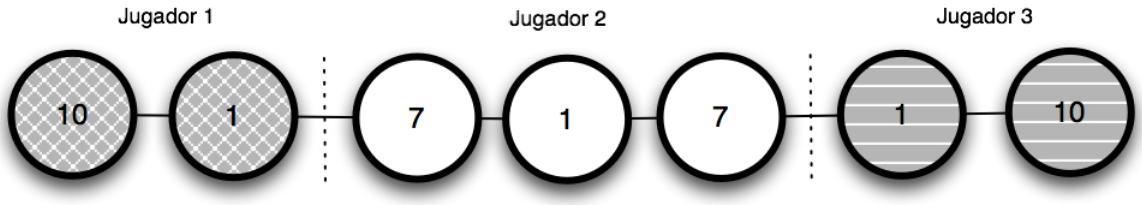


Figura 4.46: Ejemplo 2 de posición de valoración

Con este ejemplo queda demostrado que no puede obtenerse la valoración de la posición sin adelantarse a los movimientos que ocurrirán en la partida.

Ejemplo 3 Este tercer ejemplo muestra que la valoración debe ser dinámica, no sólo para estimar correctamente las probabilidades de ganar la partida, sino para cualquier

observación que se desee hacer sobre la posición.

En la figura 4.47 se muestra una parte del tablero clásico: el continente de Oceanía y su frontera con Asia.

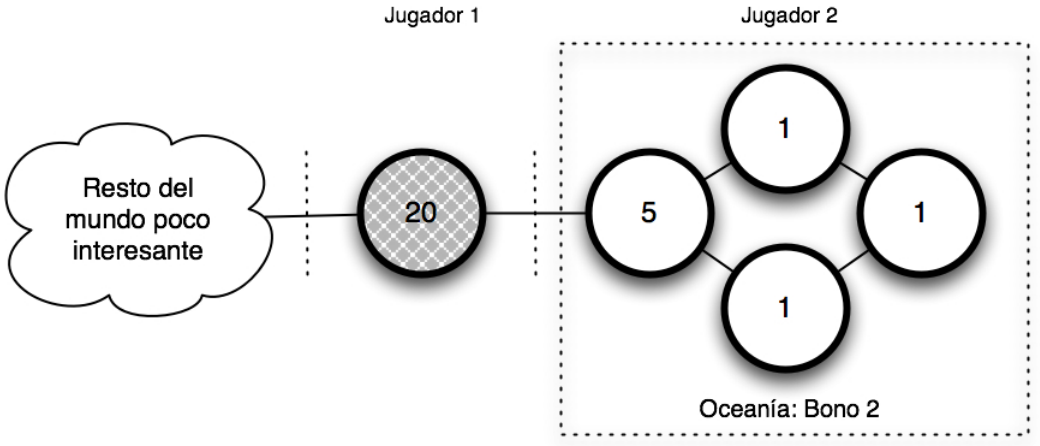


Figura 4.47: Ejemplo 3 de posición de valoración

¿A quién pertenece el continente de Oceanía? En el estado actual está claro que al jugador 2, a menos que sea primero el turno del jugador 1.

Por lo tanto, hay dos conclusiones destacables:

1. Orden de los jugadores: Si el jugador 2 juega antes que el jugador 1 sí que recibirá los refuerzos.

2. Interés: En la descripción del ejemplo, para hacerlo comprensible, se ha tenido que especificar que Asia no es interesante para el jugador 1, porque sino cabría la duda de si esos 20 soldados atacarían en la otra dirección. Por lo tanto, sin una medida del interés que indique la dirección de las tropas de cada jugador es imposible realizar ninguna predicción.

Por lo tanto, para poder valorar correctamente la posición, es obligatorio realizar una predicción acerca de los ataques que se producirán en un futuro. Es decir, *la valoración de la posición debe ser dinámica.*

Las valoraciones recogidas en el estado de la cuestión son estáticas, por lo que el algoritmo de valoración ha sido desarrollado desde cero.

Valoración de los jugadores

La valoración de una posición para un jugador depende de la fuerza de todos los jugadores $F_0, \ldots, F_J$, donde F_i es la fuerza del jugador i.

V_i debe ser directamente proporcional a F_i e inversamente proporcional a F_j $\forall j \neq i$. Lo más sencillo es una media aritmética: $V_i = \frac{F_i}{\sum_{j=0}^{J}(F_j)}$. Para resaltar la diferencia entre las valoraciones de los jugadores se introduce un parámetro de configuración PC_{expF} que sirve como exponente a la fuerza como muestra la ecuación 4.8:

$$V_i = \frac{F_i^{PC_{expF}}}{\sum_{j=0}^{J}(F_j^{PC_{expF}})} \tag{4.8}$$

Esta valoración sirve como estimación de la probabilidad de ganar la partida ya que cumple $\sum_{i=0}^{J}(Vi) = 1$.

Fracaso del plan En la expresión anterior se ha supuesto que el plan se ha ejecutado con éxito. Hay que considerar también la posibilidad de que el plan fracase.

La valoración total de un jugador dado un plan es $P(exito)V(exito)+(1-P(exito))V(fracaso)$ donde el único valor desconocido es $V(fracaso)$

Dado que cuando el plan fracasa la posición es intermedia entre el plan vacío y el valorado, el valor del fracaso deberá estar entre el valor de ambos planes. El parámetro de configuración PC_{opt} mide el optimismo del agente al realizar esta media, del modo indicado en la ecuación 4.9:

$$V(fracaso) = V(plan)PC_{opt}) + (1 - PC_{opt}V(PlanVacio) \tag{4.9}$$

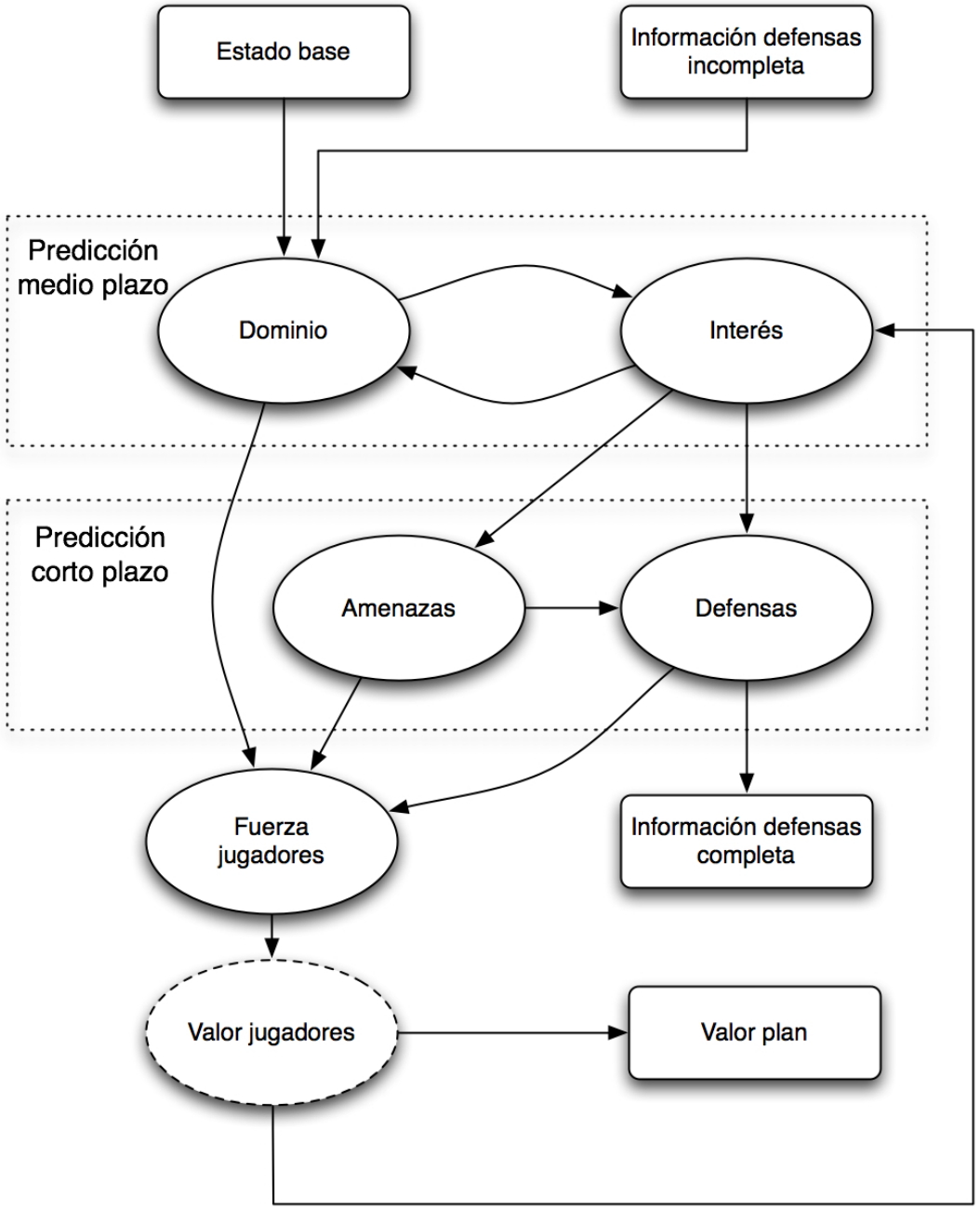


Figura 4.48: Algoritmo de valoración de plan

Algoritmo de valoración

La figura 4.48 muestra los diferentes algoritmos que se utilizan para calcular la valoración de un plan y completar su información de defensas.

Para que la valoración sea dinámica se realiza una predicción de lo que ocurrirá a medio plazo y otra a corto plazo.

La predicción a medio plazo es de tipo estratégico, sin considerar acciones concretas. Intenta estimar el dominio que cada jugador tiene de cada país. Consta del cálculo del dominio y del interés. Como estos algoritmos se realimentan el uno al otro, se define PC_{ItMed} como el número de iteraciones que se realizarán.

La predicción a corto plazo será de tipo táctico, considerando los ataques que pueden realizar los jugadores. También servirá para decidir la defensa de los territorios, completando la información de defensas del plan.

La fuerza del jugador utiliza los análisis de las predicciones para calcular la fuerza teniendo en cuenta la dinámica de la posición.

Tras calcular el valor de los jugadores el algoritmo se realimenta, dando lugar a otro parámetro PC_{It} que indica el número de iteraciones del algoritmo.

Para todos los planes distintos al plan vacío, el dominio, el interés y la valoración de los jugadores se inicializan al valor que tenían en el análisis de su plan padre. En el plan vacío se inicializan a 0, por lo que tendrá un número mayor de iteraciones PC_{ItInit}.

Los algoritmos que componen la valoración serán descritos en mayor profundidad en las siguientes secciones:

Dominio medio plazo Predice la probabilidad que cada jugador tiene de poseer cada país en los siguientes turnos.

Interés Calcula el interés que cada jugador tiene en dominar cada país.

Amenazas Predicción acerca de los posibles ataques que realizarán los oponentes en su siguiente turno.

Defensas Decide cómo defender los territorios y valora la probabilidad de lograr su defensa.

Fuerza jugadores Cuantifica la fuerza de cada jugador.

Parametro	Descripción	Valores	Valor
PC_{espF}	Exponente al que se eleva la fuerza de los jugadores en el cálculo de la valoración	$(0, \infty)$	2
PC_{opt}	Factor de optimismo al calcular el valor del fracaso del plan	$[0, 1]$	0,25
PC_{It}	Iteraciones en la valoración del plan	$[1, \infty)$	1
PC_{ItInit}	Iteraciones en la valoración del plan vacío	$[1, \infty)$	2
PC_{ItMed}	Iteraciones en la predicción a medio plazo	$[1, \infty)$	2

Tabla 4.3: Parámetros del algoritmo de valoración de los jugadores

4.5.2. Dominio a medio plazo

El dominio a medio plazo es la probabilidad de que cada jugador tenga cada país en los próximos turnos. Dado que todos los países del tablero de juego tienen que pertenecer a un jugador, la suma del dominio de cada jugador sobre un país determinado debe ser 1. El cálculo no intenta predecir qué acciones realizarán los jugadores, sino ver la influencia de cada jugador sobre cada país para entender mejor la posición.

Antes de diseñar un algoritmo hay que hacer algunas consideraciones:

1. **Interés de los países** Los jugadores tenderán a conquistar los países en los que estén más interesados.
2. **No basta considerar las fuerzas actuales en la partida** Como la predicción es a medio plazo, también hay que incluir los refuerzos que los jugadores irán recibiendo en los próximos turnos.

3. **Debe ser recursivo** Para predecir cuantos refuerzos recibirá un jugador, se utiliza el dominio, pero el dominio depende de estos refuerzos, lo que supone una recursividad en los cálculos.

4. **No dejar de lado la situación actual** Es necesario mantener en todo momento la situación real para garantizar que el resultado del algoritmo se corresponde con ella.

5. **Independencia entre países** Se ha simplificando el problema considerando que el dominio de los países es independiente.

Algoritmo de grupos de fuerza

La idea de los grupos de fuerza surge de intentar cumplir todas las restricciones planteadas. El centro de este algoritmo son los grupos de fuerza, que representan las fuerzas o grupos de soldados que tienen los jugadores en la partida.

Definición *Grupo de fuerza*: elemento de la partida que pertenece a un jugador y puede ejercer presión o fuerza sobre los países del tablero.

El dominio de un país depende de las fuerzas ejercidas por estos grupos. Los jugadores cuyos grupos de fuerza sean mayores, harán más presión y tendrán un mayor dominio.

Cada grupo de fuerza g se caracteriza por los siguientes atributos:

1. Fuerza o cantidad de presión que ejerce, F_g.

2. Países a los que puede alcanzar de forma directa, P_g.

3. Distancia a los países directos: el coste que tiene alcanzar ese país desde ese grupo, $Dd_{g,p}$.

Se definen tres tipos de grupos de fuerza. El cálculo de sus atributos será diferente para cada uno de ellos:

1. Grupos de fuerza de un país: Representan los soldados que se encuentran sobre el tablero en el estado evaluado.

2. Grupos de fuerza de refuerzos: Representan los soldados que aún no están en la partida pero que llegarán en los próximos turnos en forma de refuerzos.

3. Grupos de fuerza de defensas: Representan la información de defensas incompleta del jugador que esta realizando el plan. Como aún no ha decidido cómo asignar los soldados forman grupos aparte.

El algoritmo recibe como entradas el estado evaluado, la asignación de defensas y el interés de los jugadores. La figura 4.49 muestra una representación gráfica del algoritmo de grupos de fuerza.

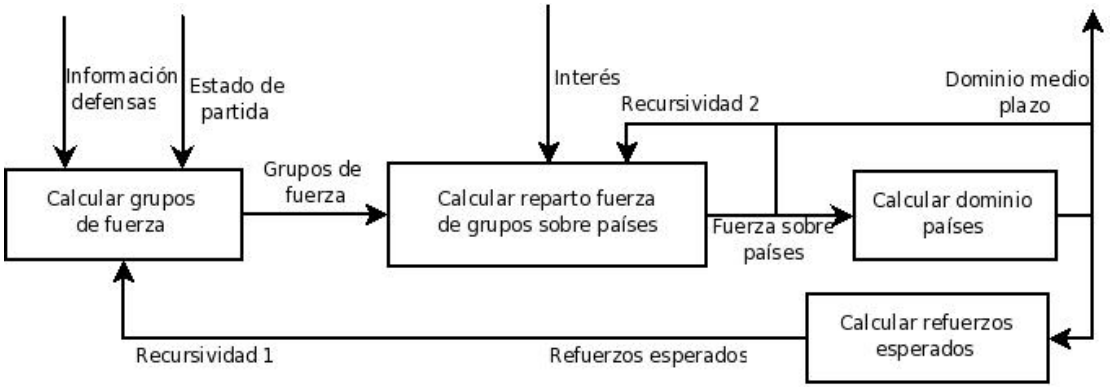


Figura 4.49: Algoritmo de grupos de fuerza

Calcular grupos de fuerza

El primer paso consiste en crear los grupos de fuerza asociados a la posición analizada e inicializar sus atributos. Cada tipo de grupo realiza estos cálculos de forma distinta. Las siguientes tablas muestran, para cada tipo de grupo, cómo obtenerlos (Calc) y calcular sus atributos.

Grupos de fuerza de un país	
Calc	Uno por cada país del tablero. Pertenece al jugador que tiene el país.
F_g	Igual al número de soldados que hay en el país evaluado menos uno que debe quedarse en el país y no puede atacar.
P_g	El país asociado al grupo es el único accesible.
$Dd_{g,p}$	0

Grupos de fuerza de refuerzos	
Calc	Un grupo por cada jugador.
F_g	Refuerzos esperados del jugador en los próximos turnos. Su cálculo se detalla en la página 114.
P_g	Todos los países que el jugador tiene en el estado base.
$Dd_{g,p}$	Es menor cuanto mayor sea el dominio del jugador sobre ese país. Debe valer 0 cuando el dominio es 1 e infinito cuando es 0: $Dd_{g,p} = \frac{1}{Dom_{j,p}} - 1$.

Grupos de fuerza de defensas	
Calc	Un grupo de defensa por cada grupo de soldados en la información de defensas.
F_g	Número de soldados del grupo de soldados asignables asociado.
P_g	Los países asignables por el grupo de soldados asignables asociado.
$Dd_{g,p}$	0

Calcular dominio de países

Como el dominio es la probabilidad estimada de que el país pertenezca al jugador a medio plazo, la suma de los dominios de todos los jugadores debe ser 1: $\sum_{i=0}^{n} Di, p = 1, \forall p$

Lo más sencillo es hacer una normalización de las fuerzas. Sin embargo, al igual que ocurria en el cálculo de la valoración de los jugadores, es necesario favorecer más a los jugadores con más fuerza. La solución es la misma que en dicho caso, realizar la normalización respecto a la fuerza elevada a un parámetro de configuracion $PC_{ExpFDom}$. La ecuación 4.10 muestra el cálculo del dominio de un país.

$$D_{i,p} = \frac{F_{i,p}^{PC_{ExpFDom}}}{\sum_{j=0}^{n}(F_{j,p}^{PC_{ExpFDom}})} \tag{4.10}$$

Dominio de continentes El dominio de un continente es la probabilidad de que el jugador tenga todos los países del continente a medio plazo. Como se asume que el dominio de los países del continente es independiente, la probabilidad de que el jugador tenga todos los países es la multiplicación de las probabilidades individuales como muestra la ecuación 4.11

$$D_{i,c} = \prod_{p \in c} D_{i,p} \tag{4.11}$$

Calcular presión de grupos sobre países

Debe repartirse la fuerza de cada grupo entre todos los países. La primera idea es un reparto proporcional al interés que tiene el jugador en tener ese país dividido entre el coste de dominar el país desde el grupo. El problema es que a los países con menor coste se les asigna mucha fuerza que apenas aporta dominio extra, por lo que debe poner solamente la fuerza necesaria.

Dado que la fuerza necesaria depende de la fuerza que aplican otros jugadores y el resto de grupos del jugador, hay que aplicar un algoritmo iterativo, que reparta la fuerza en función del interés y de las fuerzas aplicadas por los jugadores en la iteración anterior. La figura 4.50 muestra un esquema de este algoritmo.

En este esquema hay dos iteraciones:

1. Iteración de todo el algoritmo: se hará PC_{NumIt} veces.
2. Iteración de sobrefuerza: se hará $PC_{NumItFuerza}$ veces.

Para conseguir mayor velocidad en el algoritmo, solamente repartirán su fuerza entre todos los países aquellos grupos que superen un umbral de fuerza establecido por el

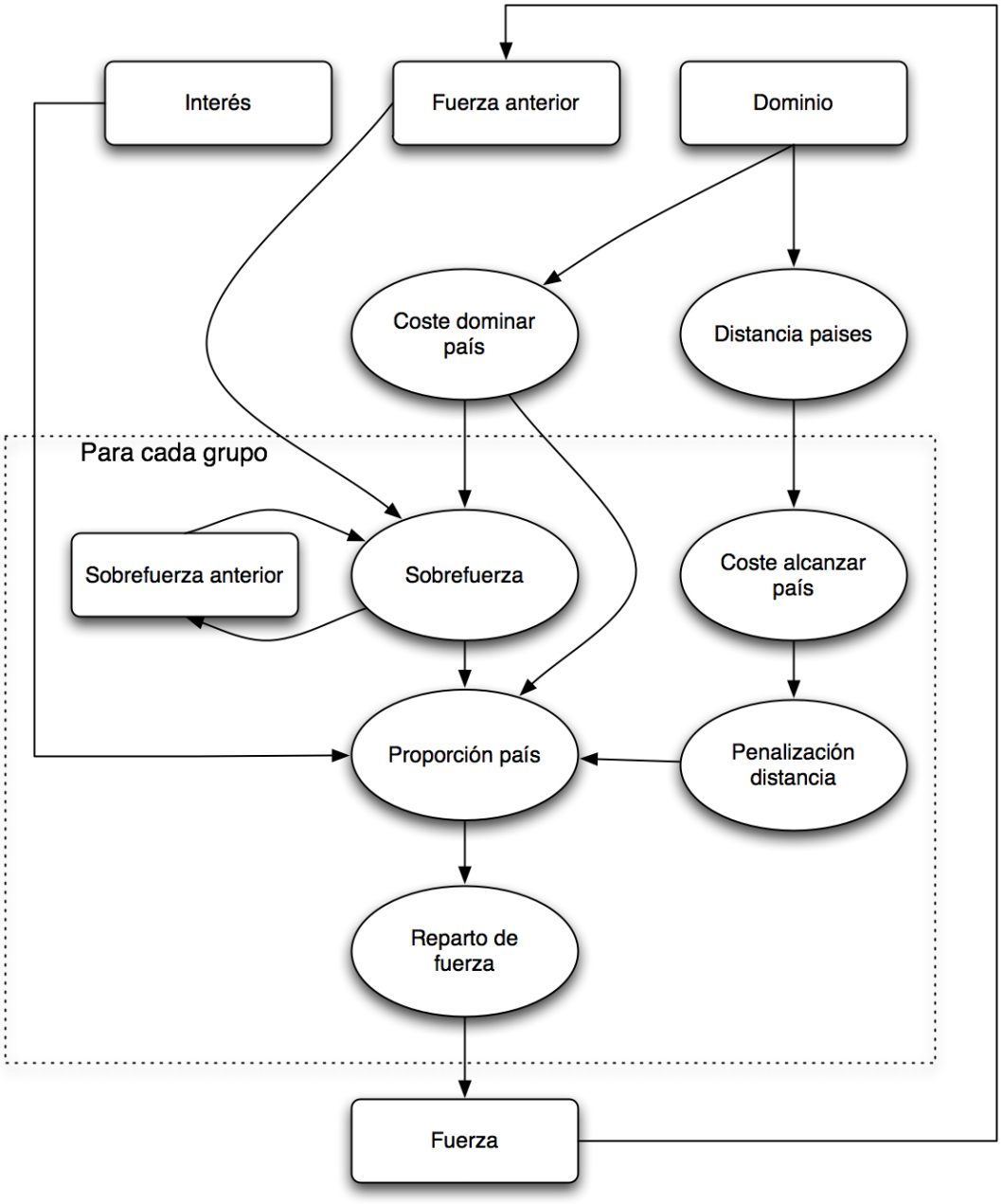


Figura 4.50: Esquema del reparto de la fuerza de los grupos entre países

parámetro $PC_{FMinRep}$. Los grupos con fuerza menor a esta cantidad, la repartirán entre sus países directos.

A continuación se analizan los cálculos que componen el reparto de la fuerza de los grupos.

Coste de dominar un país El coste de dominar un país es la fuerza que es necesario aplicar a ese país para obtener un dominio determinado. Por lo tanto, para poder calcular el coste de dominar un país es necesario definir como parámetro de configuración el dominio que quiere obtenerse del país PC_{DC}.

El coste de dominar el país para el jugador i, será el resultado de despejar F_i de la ecuación 4.10 que calcula el dominio de un país, sustituyendo D_p por PC_{DC}.

$$CD_i = \sqrt[PC_{ExpFDom}]{\frac{PC_{DC}}{1 - PC_{DC}} \left(\sum_{j \neq i} (F_j^{PC_{ExpFDom}}) + PC_{Fbase} \right)} \tag{4.12}$$

En la ecuación 4.12 se puede ver que el coste de dominar depende del multiplicador $\frac{PC_{DC}}{1-PC_{DC}}$. La gráfica 4.51 muestra como según aumenta el valor de PC_{DC}, el multiplicador crece exponencialmente, por lo que no deberán escogerse valores cercanos a 1 si se desea obtener un coste de dominar razonable.

Distancia entre países La distancia entre países del jugador j, $Dist_{p,q}$ es el camino mínimo entre los países. La ecuación 4.13 muestra el coste de ir desde p hasta q, que depende del dominio que cada jugador tiene del país y la fuerza que aplica. Se definen como parámetros de configuración $PC_{CAPMult}$ que multiplica al coste para regularlo y $PC_{CAPpropio}$ que da el coste de alcanzar el país cuando lo domina el jugador.

$$CAP_{j,p,q} = PC_{CAPMult} \left(1 + \sum_{k \neq j} (F_{k,q} D_{k,q}) + PC_{CAPpropio} D_{j,q} \right) \tag{4.13}$$

Se ha utilizado el algoritmo de caminos mínimos de *Floyd* [Gou88] que tiene la ventaja de calcular simultaneamente todas las distancias.

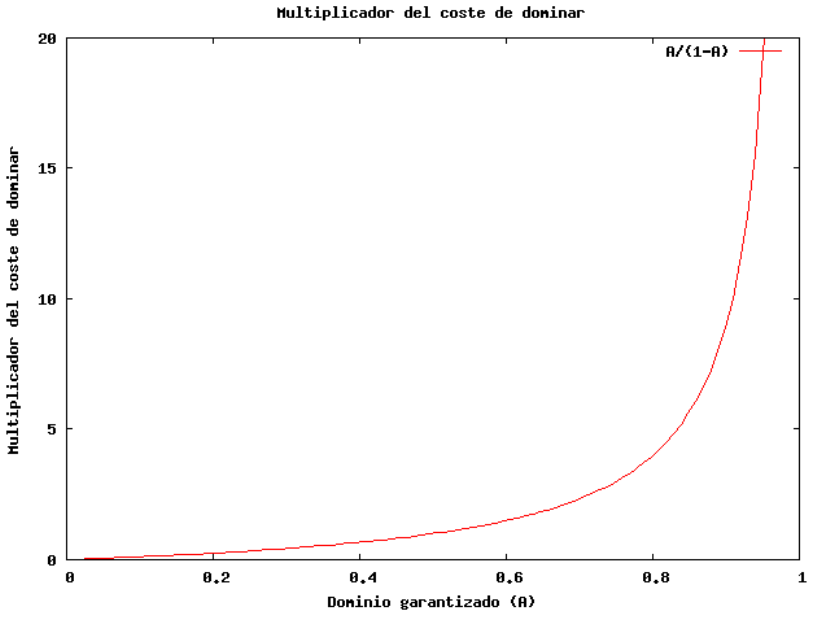


Figura 4.51: Multiplicador del coste de dominar un país

Coste de alcanzar un país La ecuación 4.14 muestra el cálculo del coste de alcanzar un país p desde un grupo g, $CA_{g,p}$ como el mínimo, para todos los países directos del grupo (PDG), de la distancia hasta ese país directo más la distancia entre los países.

$$CA_{g,p} = min_{i \in PDG}(Dd_{g,i} + Dist_{i,p}) \tag{4.14}$$

Sobrefuerza La sobrefuerza mide la cantidad de fuerza de grupo que se está desperdiciando en grupos que ya se dominan. Sólo tiene sentido cuando se aplica más fuerza de la necesaria, por lo que como mínimo debe ser 1. Además, depende de la sobrefuerza calculada en la iteración anterior, para que el efecto se acumule en cada iteración en los casos que sea necesario.

$$SF_{p,i} = max(1, \frac{SF_{p,i-1}F_{j,p}}{CD_{j,p}}) \tag{4.15}$$

Penalización por distancia La penalización por distancia penaliza los países lejanos del grupo. Será mayor cuanto más diferencia haya entre la fuerza del grupo y el coste

de alcanzar el país. La ecuación 4.16 muestra como se calcula.

$$PenDist_p = \frac{F_g - CA_{g,p}}{F_g(CA_{g,p} + 1)} \tag{4.16}$$

Proporcion país Para calcular la proporción de cada país primero se calcula la utilidad de ese país como indica la ecuación 4.18 utilizando los parámetros PC_{ExpInt} y PC_{ExpDom} para ponderar la importancia del interés y el coste, respectivamente.

$$Util_{j,p} = \frac{I_{j,p}^{PC_{ExpInt}}}{CD_{j,p}^{PC_{ExpDom}}} \tag{4.17}$$

A continuación, se normaliza esta cantidad y se le aplican el factor de sobrefuerza y la penalización por distancia tal y como describe la ecuación 4.18.

$$Prop_{g,p} = \frac{Util_{j,p}}{\sum_q Util_{j,q}} \times \frac{PenDist_p}{SF_{p,i}} \tag{4.18}$$

Reparto de fuerza El grupo reparte su fuerza proporcionalmente a la proporción calculada para cada país. Además, los grupos de un país añaden la fuerza del soldado que debe quedarse en su país. La fuerza añadida no es exactamente 1, sino un parámetro de configuración $PC_{FSoldPais}$ que sirve para premiar al jugador que lo posee otorgándole un poco de dominio extra.

La ecuación 4.19 muestra el reparto de la fuerza del grupo.

$$F_{g,p} = \begin{cases} F_g \frac{Prop_{g,p}}{\sum_q Prop_{g,q}} + PC_{FSoldPais} & \text{si } g \text{ es el grupo del país } p, \\ F_g \frac{Prop_{g,p}}{\sum_q Prop_{g,q}} & \text{en caso contrario.} \end{cases} \tag{4.19}$$

Calcular refuerzos esperados

Los refuerzos esperados por un jugador son, basándose en las reglas del Risk y en el dominio a medio plazo, cuántos soldados extra recibirá en los próximos turnos. Se define como parámetro PC_{Tref} el número de turnos considerados.

Como el dominio de los países es independiente, el número de países que controla el jugador se puede calcular mediante la suma del dominio que tiene de cada país. En las reglas se aplica un redondeo hacia abajo pero en este cálculo no se redondea, al ser una estimación. El número mínimo de refuerzos por países es tres, por lo que queda: $max(3, \frac{\sum_{p=0}^{N} D_{i,p}}{3})$.

El número de soldados que obtiene un jugador por dominar continentes es: $\sum_{c=0}^{C}(R_c D_{i,c})$

En cuanto a las cartas, asumiendo que todos los jugadores conseguirán una carta en cada turno: $\frac{(C_i + PC_{Tref})V_{cartas}}{3}$

En resumen, la fuerza del grupo de refuerzos del jugador i queda como muestra la ecuación 4.20.

$$Fref_i = PC_{Tref}\left(max(3, \frac{\sum_{p=0}^{N} D_{i,p}}{3}) + \sum_{c=0}^{C}(R_c \prod_{p \in c} D_{i,p})\right) + \frac{(C_i + PC_{Tref})V_{cartas}}{3} \tag{4.20}$$

4.5.3. Algoritmo de interés de países

Este algoritmo calcula el interés que tiene cada jugador en dominar a medio plazo cada país, a partir del dominio y el interés calculado en la iteración anterior.

Los factores que influyen son:

- Interés base por tener un país: Es la unidad de interés, por lo que su valor es 1.
- Interés del continente ($I_{j,c}$): por tener el continente al que pertenece el país.
- Dominio de la zona ($D_{j,p}$): el dominio que tiene el jugador alrededor del país.

La forma más sencilla de combinar estos factores es con una media ponderada por parámetros de configuración. La ecuación 4.21 interés que tiene cada jugador j e cada país p ($I_{j,p}$), que pertenece al continente c.

$$I_{j,p} = PC_{Pbase} + PC_{Pcont} I_{j,c} + PC_{Pzona} D_{j,i} \tag{4.21}$$

Parametro	Descripción	Valores	Valor
$PC_{ExpFDom}$	Exponente de la fuerza en el cálculo del dominio de un país	$[1:\infty)$	2
PC_{NumIt}	Número de iteraciones que se realizará el algoritmo	$[1,\infty)$	2
$PC_{NumItFuerza}$	Número de iteraciones del cálculo de la fuerza por el cambio de la sobrefuerza.	$[1,\infty)$	5
$PC_{FMinRep}$	La fuerza mínima que debe tener un grupo para que pueda repartirse a países no directos.	$[0:\infty)$	1
PC_{DC}	Dominio que se desea lograr en el cálculo del coste de dominar.	$(0:1)$	0,8
PC_{Fbase}	Para el coste de dominar un país, fuerza que hay por defecto en un país.	$[0:\infty)$	0,1
$PC_{FSoldPais}$	Fuerza que tiene el soldado que debe quedarse en cada país. Cuanto mayor sea el valor, el algoritmo de dominio tenderá a considerar que el jugador que tiene el país actualmente lo domina.	$[1,\infty)$	1,5
$PC_{CAPPropio}$	Coste de alcanzar un país propio.	$[0:\infty)$	0,5
PC_{CAMult}	Multiplicador del coste de alcanzar un país.	$[0:\infty)$	2
PC_{ExpInt}	Exponente del interés en el cálculo de la proporción.	$[0:\infty)$	2
PC_{ExpCD}	Exponente del coste de dominio en el cálculo de la proporción.	$[0:\infty)$	1
PC_{Tref}	Número de turnos para el que se aplican los refuerzos esperados.	$[0,10]$	0,5

Tabla 4.4: Parámetros del algoritmo de dominio a medio plazo

Interés de un continente

Los continentes aportan interés porque, si se domina todo el continente se recibirán soldados extra en cada turno. Por lo tanto, un jugador estará interesado en un continente si cree que puede llegar a dominarlo por completo.

El interés de un jugador en el continente c, (I_c) se compone de los siguientes factores:

- $Dpot_c$: el dominio potencial del jugador sobre el continente.
- R_c: el valor del continente.

$$I_{j,c} = R_c Dpot_{i,c} \tag{4.22}$$

El jugador siempre intentará ir a por algún continente por baja que sea la probabilidad de tenerlo. Por lo tanto, se define como parámetro de configuración el interés mínimo en continentes $PC_{IContMin}$ utilizado en la ecuacion 4.23.

$$\text{Si } \sum_p I_{j,c} < PC_{IContMin} \Rightarrow I_{j,c} = PC_{IContMin} \frac{I_{j,c}}{\sum_c I_{j,c}} \tag{4.23}$$

Definición El *dominio potencial* de un jugador en un país es aquél que podría tener si estuviese muy interesado en él.

Por lo tanto, el dominio potencial de un continente se calcula ejecutando el algoritmo de dominio explicado en la sección 4.5.2, página 106 con las siguientes modificaciones:

- El interés del jugador es el calculado en la iteración anterior sustituyendo el interés del coninente por el máximo ($R_c PC_{Pcont}$).
- Sólo se calcula la fuerza de los grupos del jugador, manteniendo fija la fuerza que los otros jugadores hacen sobre el tablero.

Dominio de la zona

Un país será más interesante cuanto mayor dominio tenga el jugador sobre la zona, ya que podrá ser defendido con mayor facilidad y tendrá más soporte. En la valoración de un país analizada en el estado de la cuestión (véase la sección 2.4.2, página 17), se tenía en cuenta este dominio aportando una bonificación al interés por cada país y soldado propios vecinos al país y restándola por los enemigos.

Aquí se adapta esa expresión, utilizando el dominio a medio plazo y considerando, no solamente a los vecinos directos, sino también a los países a mayor distancia.

El dominio de la zona es una *media proporcional* del dominio del país, sus vecinos, sus países a distancia 2, etc. Se define como parámetro de configuración la distancia máxima para la que se cuentan los países $PC_{DGradZona}$. Evidentemente, los países a mayor distancia tendrán menor peso en esta media, para lo que se define el multiplicador de peso por distancia $PC_{MultGradZona}$. El peso del dominio de los países a distancia i es $PC^i_{MultGradZona}$.

La ecuación 4.24 muestra como se calcula definitivamente el dominio de la zona.

$$Dzona_{j,p} = \sum_{i=0}^{PC_{DGradZona}} \sum_{q \in Vec_i(p)} \frac{D_{j,q} PC^i_{MultGradZona}}{\sum_{k=0}^{PC_{DGradZona}} (NumVec_i(p) PC^k_{DGradZona})} \tag{4.24}$$

Parametro	Descripción	Valores	Valor
PC_{Pbase}	Peso del interés base de un país.	$[-\infty, \infty]$	1
PC_{Pcont}	Peso del interés por continente.	$[-\infty, \infty]$	500
PC_{Pzona}	Peso del interés por dominio de la zona.	$[-\infty, \infty]$	2
$PC_{DGradZona}$	Distancia máxima a la que se realiza la degradación del dominio de la zona.	$[0, \infty]$	2
$PC_{MultGradZona}$	Multiplicador para penalizar la distancia en la degradación del dominio de la zona.	$(0, 1)$	0,3
$PC_{IContMin}$	Interés mínimo que el jugador tiene en los continentes.	$[0, \infty]$	0,1

Tabla 4.5: Parámetros del algoritmo de interés

4.5.4. Algoritmo de amenazas

Este algoritmo intenta predecir los ataques que realizarán los enemigos en su próximo turno y la posición resultante.

Definición Una *amenaza* se define como un posible ataque del jugador en su siguiente turno. Consta de:

- Jugador atacante j: El jugador que realiza el ataque.
- País de origen O: El país desde el que se realiza el ataque.
- Países conquistados $P_1, \ldots, P_n$: La lista de países que se conquistan en el ataque. Puede ser la lista vacía, en cuyo caso implica no realizar ataques desde el país origen.
- Soldados puestos S_a: El número de soldados en el país atacante antes de realizar el ataque.

Hay que recordar que, debido al alto número de posibilidades de cada jugador y la aleatoriedad en los resultados de los ataques, *es imposible realizar una predicción de lo que ocurrirá*, por lo que se asumen las siguientes simplificaciones:

1. Las amenazas realizadas por cada país son independientes, incluyendo su probabilidad.
2. La probabilidad de que un país pertenezca a un jugador es independiente del resto de países.
3. Desde un país solamente se puede conquistar una lista de países y no un árbol.
4. Un jugador debe situar todos sus refuerzos en el mismo país. De este modo, el jugador tiene un número limitado de posibilidades para situar sus refuerzos (una por país) y es viable calcular su probabilidad.

La figura 4.52 muestra que para cada jugador j se obtienen sus amenazas en tres fases:

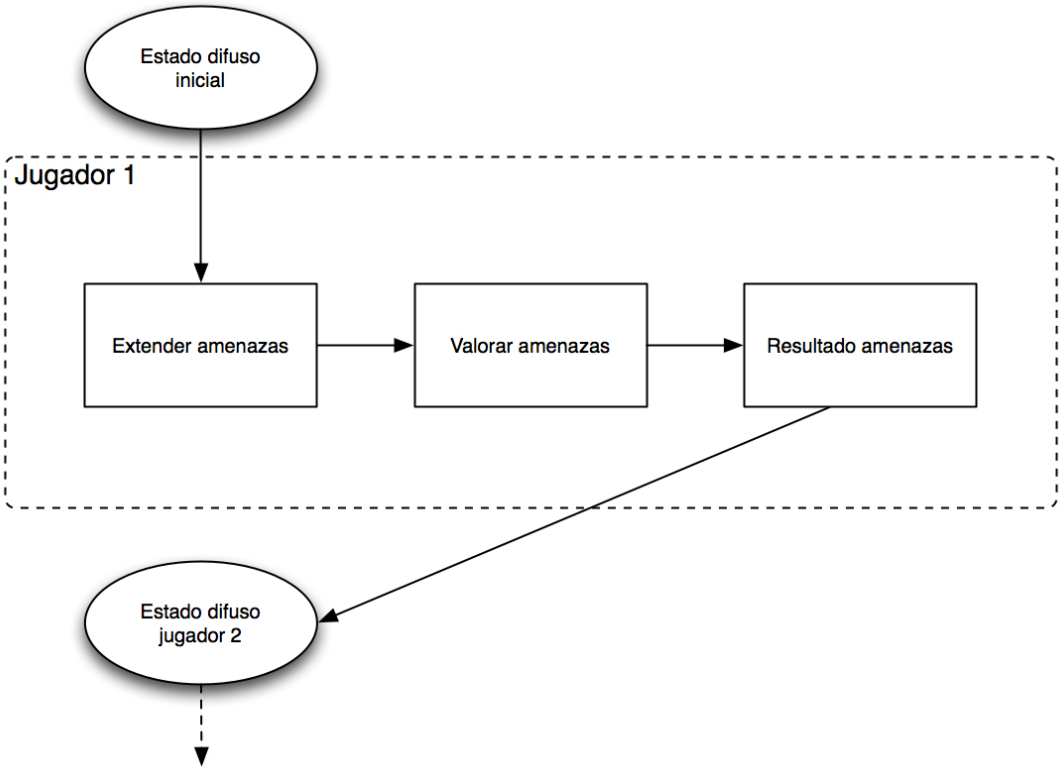


Figura 4.52: Esquema del algoritmo de amenazas

1. Extender amenazas: Se recogen las amenazas que puede realizar el jugador j y se realizan algunos cálculos sobre ellas.

2. Valorar amenazas: Se valoran las mejores amenazas del jugador j y se calcula su probabilidad.

3. Resultado amenazas: Se obtiene el estado difuso resultante de aplicar las amenazas.

Estas fases se realizan para todos los oponentes en el orden que les corresponde jugar, de modo que el estado difuso de entrada para el jugador j es el estado difuso de salida del jugador $j-1$.

Estado difuso

Antes de entrar en cada una de las fases del algoritmo, se define qué es un estado difuso. En este estado no se conoce con exactitud el jugador al que pertenece cada país ni el número de soldados que hay en él, debido a que se desconoce qué amenazas se realizarán y cuál será el resultado de los ataques.

Para tratar con esta incertidumbre, el estado difuso guarda, para cada país, la probabilidad de que pertenezca a cada jugador $P_{j,p}$ y el número medio de soldados que hay en ese caso, $S_{j,p}$.

El número de soldados que hay en el país p es $S_p = \sum_{j=0}^{J}(P_{j,p}S_{j,p})$.

Si un jugador J_i quiere atacar al país p la probabilidad de poder hacerlo será $(1 - P_{J_i,p})$ y el número de soldados al que tendrá que enfrentarse será $\sum_{j \neq J_i}(P_{j,p}S_{j,p})$.

Probabilidad de tener un continente Para conocer el número de soldados recibidos por un jugador en el siguiente turno, es importante saber si conservará sus continentes. Si el jugador no tiene el continente en el estado actual es imposible que lo tenga antes de su siguiente turno. En caso de que tenga todo el continente, la probabilidad de mantenerlo será igual a la probabilidad de mantener todas sus fronteras como indica la ecuación 4.25.

$$P_{j.c} = \prod_{p \in Fronteras_c} P_{j,p} \tag{4.25}$$

Extender las amenazas

En esta primera fase se recogen todas las amenazas que puede realizar el jugador. Para cada país extenderá dos veces las amenazas: una suponiendo que sitúa los refuerzos en ese país, y la otra no.

Para cada amenaza se calcula:

1. Probabilidad de poder realizarla $P_{poder}(a)$: la probabilidad de que el jugador no posea ya los países atacados.
2. Probabilidad de éxito de cada ataque $P_{exito}(P_i)$: la probabilidad de realizar todos los ataques hasta ése correctamente.
3. Probabilidad de éxito total $P_{exito}(a)$: igual a la probabilidad de lograr con éxito el último ataque.
4. Soldados sobrantes para defender: $S_a - \sum_{i=1}^{n}(Coste(P_i) + 1)$
5. Interés logrado por la conquista de países: $\sum_{i=1}^{n}(Ic_{j,p}P(P_i))$
6. Interés máximo esperado por el dominio de países: $\sum_{i=1}^{n} I_{j,p}$

```
L_e = Lista de expandir
L_a = Lista de amenazas
Se inicializan las listas con la amenaza vacía (no realiza ataques)

Mientras num amenazas en L_e > 0 y num amenazas en L_a < maxNumAmenazasPais
 Si Amenaza puede extender hijos
  Para todo v, vecino del último país de la amenaza
   Si v puede ser atacado y es viable
    L_e.add(nuevaAmenaza)
    L_a.add(nuevaAmenaza)
  Ordenar L_e por la probabilidad de las amenazas
```

Figura 4.53: Algoritmo de expandir amenazas

Poda de amenazas expandidas Extender las amenazas es muy costoso, puesto que debe ir recorriendo para cada país sus posibles vecinos y alcanzar toda la profundidad que le permitan los soldados en el país de origen. Es necesario poner un límite a esta profundidad para que el rendimiento del agente no se vea mermado en posiciones con un alto número de soldados y conexiones entre países:

1. Se define el número máximo de amenazas que se calcularán para cada país, $PC_{NumMaxAm}$. Para que las amenazas que se calculan no sean arbitrarias, es necesario garantizar que siempre se explorarán las amenazas con mayor probabilidad de éxito.

2. Se define la probabilidad de éxito mínima requerida para expandir una amenaza como PC_{MinEx}.

Interés de conquista

El interés de conquista es el beneficio que obtiene el jugador por conquistar un país en su turno. Se diferencia del interés de dominio en que en este caso le da igual si puede defender el territorio. El único motivo por el que un jugador esta interesado en conquistar un país sin necesidad de defenderlo es para dañar a sus enemigos. Además, cuanto mayor es la fuerza de un enemigo, en función a la de los demás (la valoración del jugador enemigo), mejor será dañarle.

El interés de conquista se calcula en función de:

- $Dny_{e,p}$: El daño que se hace al enemigo por quitarle p.
- $P_{e,p}$: La probabilidad de que el enemigo tenga el país.
- V_e: El valor del enemigo e, su probabilidad de ganar la partida.

Como indica la ecuación 4.26:

$$Ic_{j,p} = \sum_{e \in enem(j)} (P_{e,p} V_e Dny_{e,p}) \tag{4.26}$$

El daño a un enemigo se calcula como una media ponderada de:

- Daño base: por quitarle un país.
- Daño soldados: por matar los soldados que están en ese país.
- Daño continente: por quitarle un continente que tiene entero impidiendo que reciba su bono R_c.
- Daño por interés: por quitarle un terriotorio interesante para él. Por ejemplo de un continente que esta intentando dominar aunque todavía no lo tiene completo.

Como indica la ecuación 4.27:

$$Dny_{e,p} = PC_{PicBase} + S_{e,p} PC_{PicSold} + P_{e,c} R_c PC_{PicCont} + I_{e,p} PC_{PicInt} \tag{4.27}$$

Valorar las amenazas

Calcula la probabilidad de que se lleven a cabo las amenazas expandidas. La probabilidad de la amenaza se calcula normalizando su utilidad U_a, multiplicando el resultado por la probabilidad de que haya refuerzos (o no) en su país de origen o, como muestra la ecuación 4.28.

$$P_{rea}(a) = \begin{cases} \frac{U_a}{\sum_{i \in A} U_i} P_{Ref}(o) & \text{si } a \text{ tiene refuerzos} \\ \frac{U_a}{\sum_{i \in A} U_i} (1 - P_{Ref}(o)) & \text{si } a \text{ no tiene refuerzos} \end{cases} \tag{4.28}$$

Utilidad de las amenazas La utilidad de las amenazas determina lo bueno que es realizar una amenaza para el jugador. Viene dada por:

- Interés de dominio de los países conquistados $I_{j,p}$.
- Interés de conquista de los países conquistados $Ic_{j,p}$.
- Probabilidad de éxito de la amenaza $P_{exito}(a)$.
- Probabilidad de poder realizarla $P_{poder}(a)$.

Por otra parte, los jugadores normalmente no realizan todos los ataques que pueden porque prefieren guardar soldados para defender sus territorios. Por lo tanto, para valorar una amenaza es necesario calcular la defensa que logra de cada país $P_{def}(P_{di})$.

Calcular el algoritmo de defensas para cada amenaza es excesivamente costoso, por lo que se aplica un algoritmo de defensas simplificado:

- Solamente se mira la defensa en los países que participan en la amenaza, obviando el resto del tablero.
- Solamente se realiza la inicialización de las defensas, sin hacer ninguna búsqueda.
- Se supone que los soldados que sobreviven a los ataques pueden asignarse a cualquier país, por lo que sólo hay un grupo de soldados.

La utilidad de conquista tiene un peso de PC_{PUC} y la utilidad de dominio de PC_{PUD}. La ecuación 4.29 muestra cuál es el valor de realizar una amenaza a.

$$V_a = PC_{PUC} \sum_{i=1}^{n} (Ic_{j,p} Pexito(P_i)) + PC_{PUD} \sum_{i=1}^{n} I_{j,p} P_{def}(P_i) \tag{4.29}$$

La utilidad de la amenaza se calcula a partir de su valor como indica la ecuación 4.30.

$$U_a = \begin{cases} (P_{poder}(a) V_a P_{exito}(a))^{PC_{ExpU}} & \text{si } PC_{PoderElev} \text{ ,} \\ P_{poder}(a)(V_a P_{exito}(a))^{PC_{ExpU}} & \text{en caso contrario} \end{cases} \tag{4.30}$$

Probabilidad de refuerzos en un país El jugador debe decidir en qué país asigna todos sus refuerzos recibidos. Debe calcularse la probabilidad de que decida asignarlos a cada país $P_{Ref}(p)$.

El jugador asignará los soldados en los países que más le interese. Por lo tanto sigue los siguientes pasos:

1. Para cada país p
   - *a*) Valorar las amenazas desde p con refuerzos
   - *b*) Valorar las amenazas desde p sin refuerzos
   - *c*) Valorar el beneficio de poner refuerzos en p, B_p
2. Calcular $P_{Ref}(p)$ en base al beneficio.

Al suponer que deben asignarse todos los soldados a un único país se está introduciendo un error significativo, puesto que si hay un país con ataques muy interesantes que requieren de soldados pero no de todos, acaparará toda la probabilidad de refuerzos. Para resolver esto, se define el pesimismo de refuerzos PC_{PesRef} como el incremento de la probabilidad de refuerzos de cada país.

La ecuación 4.31 muestra como queda el cálculo de la probabilidad de refuerzos de un país.

$$P_{Ref}(p) = \frac{B_p}{\sum_q B_q}(1 - PC_{PesRef}) + PC_{PesRef} \tag{4.31}$$

Poda de amenazas en la valoración Al igual que en el caso de extender las amenazas, este es un proceso muy costoso y hay que podarlo anticipadamente para mejorar el rendimiento del agente. Aunque el número de amenazas ya está limitado en su extensión, valorar cada amenaza consume bastante tiempo al tener que calcular su defensa y no tiene sentido valorar amenazas que seguro tendrán muy baja probabilidad de realización.

Para poder podar las amenazas nos basamos en la utilidad máxima esperada, calculada al extender las amenazas. Es similar a la utilidad de la amenaza pero suponiendo que todos los países podrán ser defendidos, por lo que es una cota superior de la utilidad real de la amenaza.

Para realizar la poda se ordenan las amenazas por su utilidad máxima esperada, y se aplican dos criterios de poda:

1. Mínimo porcentaje de utilidad $PC_{MinPUtilMax}$: Si la utilidad máxima esperada es menor que la utilidad real de la mejor amenaza dividida entre $PC_{MinPUtilMax}$.
2. Mínima probabilidad esperada $PC_{MinProbEsper}$: Si la probabilidad esperada máxima (calculada a partir de la utilidad esperada máxima) es menor que $PC_{MinProbEsper}$.

Resultado de las amenazas

El resultado de las amenazas parte del estado difuso anterior y de las amenazas del jugador, con su probabilidad para calcular el nuevo estado difuso.

El mayor problema para calcular la nueva probabilidad de que un país pertenezca a cada jugador es que puede estar amenazado desde dos países y, aunque las amenazas se han considerado independientes, no puede ser conquistado dos veces por el mismo jugador. Para solucionar esto se aplican *las amenazas desde un mismo país independientemente*, acumulando su probabilidad. Las amenazas desde distintos países se aplican sobre todos los jugadores que poseen el país, incluyendo al jugador que realiza la amenaza. De este

modo, si el jugador ya poseía el país, con su nueva amenaza se quita probabilidad a si mismo y lo único que hace es modificar el número de soldados restantes.

```
Para cada país del jugador: o
 Para cada país: p
  Para cada amenaza desde o que conquista p: a
   Para cada posible estado de p: E_p
    P(victoria) = P(a)P(E_p)Pexito(a,p)
    probConquistado(E_p)-= P(victoria)
    soldRestDesdeO(p) = mediaPond(soldRestDesdeO(p), a.soldDef(p))
    QuitarProbJugador(E_p, P(victoria))

  aplicarProb(P(victoria, soldRestDesdeO(p))
```

Figura 4.54: Algoritmo de resultado de amenazas

4.5.5. Algoritmo de defensas

El algoritmo de defensas se encarga de completar la información de defensas del algoritmo y estimar la probabilidad de que el jugador defienda sus países.

La figura 4.55 muestra un esquema de la solución planteada. En ella, el proceso de completar la información de defensas se divide en dos partes totalmente distinguidas:

1. Defensa: decide la asignación de cada grupo de soldados entre los países a los que pueden ser asignados. Para esto se apoyará en una valoración de las defensas que permitirá cuantificar la defensa lograda y de un algoritmo que transforme las amenazas de la predicción a corto plazo.
2. Caminos: decide desde qué país se atenderá cada asignación de defensa.

Simplificación del problema El problema de las defensas es muy complejo, por lo que conviene realizar algunas simplificaciones que permitan abordarlo:

1. **Separación de defensa y caminos** Al asumir el cálculo de la distribución de soldados sin considerar el camino de las tropas, se están limitando el número de

Parametro	Descripción	Valores	Valor
$PC_{NumMaxAm}$	Número máximo de amenazas expandidas	$[1, \infty)$	30
PC_{MinEx}	Mínima probabilidad de éxito para extender la amenaza.	$[0, 1]$	0,9
$PC_{PicBase}$	Peso del interés de quitar un país al enemigo	$(-\infty, \infty)$	1
$PC_{PicSold}$	Peso del interés de quitar un soldado al enemigo	$(-\infty, \infty)$	1
$PC_{PicCont}$	Peso del interés de quitar un continente al enemigo	$(-\infty, \infty)$	100
PC_{PicInt}	Peso del interés de quitar un territorio interesante al enemigo	$(-\infty, \infty)$	0,05
PC_{ExpU}	Exponente al que se eleva la utilidad de una amenaza.	$[0, \infty)$	2
$PC_{PoderElev}$	Indica si la probabilidad de realizar la amenaza se eleva también al exponente de utilidad.	$[true, false]$	$true$
PC_{PUC}	Peso de la utilidad de conquista.	$[0, \infty)$	2
PC_{PUD}	Peso de la utilidad de dominio.	$[0, \infty)$	0,5
PC_{PesRef}	Factor de pesimismo de refuerzos al calcular los refuerzos que sitúan los enemigos en sus países.	$[0, 1]$	0,3
$PC_{MinPUtilMax}$	Porcentaje mínimo sobre la utilidad máxima que debe tener la utilidad máxima esperada de una amenaza para ser valorada.	$[0, 1]$	0,2
$PC_{MinProbEsper}$	Probabilidad mínima esperada que debe tener una amenaza para ser valorada.	$[0, 1]$	0,01

Tabla 4.6: Parámetros del algoritmo de amenazas

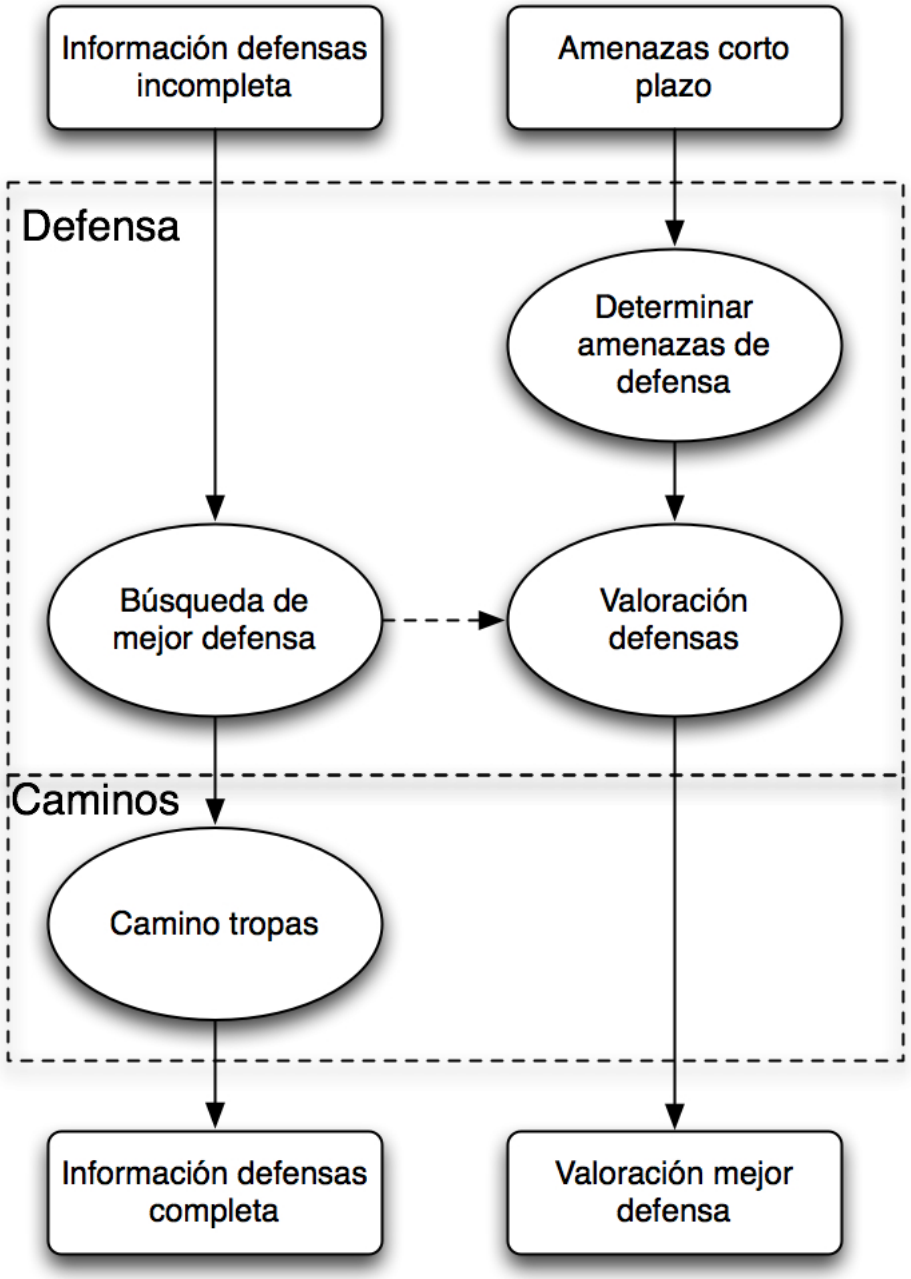


Figura 4.55: Esquema de los pasos del algoritmo de defensas

factores que se tienen en cuenta. Por ejemplo, el algoritmo de defensa no valorará el beneficio por aumentar la probabilidad de los ataques al asignar los soldados a un país.

Por lo tanto, la valoración a corto plazo solamente tiene en cuenta la defensa de los países, que es sin ninguna duda el factor más importante en prácticamente todas las situaciones del juego.

2. **No hay países enemigos** Dentro del algoritmo de defensas sólo se consideran los países del jugador defensor. De este modo, en la propagación de amenazas un ataque no puede salir de un país del jugador y entrar a otro a través de un enemigo.

3. **Intereses de cada país independientes** El interés de cada país se ha considerado independiente del de los demás.

4. **Propagación sin objetivo** Al propagar las amenazas del enemigo por los países del defensor, no se ha tenido en cuenta el interés de los rivales en cada país.

Valoración de las defensas

La valoración de las defensas cuantifica la defensa para poder optimizarla. Se calcula para una matriz de asignaciones concreta, por lo que opera con:

- Número de soldados en cada país
- Interés en defender cada país
- Amenazas recibidas

La ecuación 4.32 valora las defensas como la cantidad de interés de países que logra defenderse más una bonificación por defender los continentes.

$$Vdef = \sum_{p}(I_p Pdef_p) + PC_{VDefCont} \sum_{c}(Pdef_c Rc) \tag{4.32}$$

El cálculo se reduce a la probabilidad de defensa de cada país $Pdef_p$ y continente $Pdef_c$.

Propagación de las amenazas

La propagación de amenazas mostrada en la figura 4.56, extiende las amenazas del enemigo por los países del defensor, calculando el máximo daño que puede llegar a hacer a cada país. Para cada país, solamente se tiene en cuenta la mayor amenaza que llega hasta él.

```
propagarAmenazas(){
  //Todo país esta defendido hasta que no lo amenacen.
  forall p DefPais (p) = 1

  //Para cada amenaza
  forall a in amenazas
    indiceDefensaAmenaza = propagarAmenaza(a)
    forall p DefPais(p) = DefPais(p) indiceDefensaAmenaza[p];
}

propagarAmenaza(amenaza){
  forall p paisesVisitados [p] = -1
  forall p indiceDefensa[p] = 1
  propagarAmenaza(amenaza.pais, amenaza.fuerza)
}

propagarAmenaza(pais, amenaza){
  //Finaliza la recursividad si ya ha pasado esta amenaza o una mayor
  if (amenaza <= paisesVisitados[pais]) return

  //Guarda la amenaza maxima recibida en el pais y cambia su defensa
  paisesVisitados[pais] = amenaza;
  indiceDefensa[pais] = defensaPais(soldadosPais[pais], amenaza)

  forall v in vecinos
    propagarAmenaza(v, amenazaReducida)
}
```

Figura 4.56: Algoritmo de propagación de amenazas

La amenaza reducida se calcula según la ecuación 4.33. Al contar el número de vecinos

solamente cuentan si pertenecen al jugador defensor.

$$AmRed = \begin{cases} \frac{Am - CosteIndiceDef - 1}{NumVecinos^{PC_{ExpDivAmVec}}} & \text{si } NumVecinos > 1, \\ Am - CosteIndiceDef - 1 & \text{en caso contrario} \end{cases} \tag{4.33}$$

Función de defensa de un país

La función de defensa de un país debe decir, en función de la amenaza recibida y de el número de defensores en ese país, cuál es la probabilidad de que sea defendido $DefPais(S_d, A)$. No es igual a la probabilidad de éxito del ataque porque ahora sí se tiene en cuenta que el jugador atacante debe decidir si realizar el ataque o no.

Es muy importante que esta función no crezca demasiado rápido. Si con un número de defensores no muy alto logra una gran defensa, añadir soldados no aportará nada y nunca podrá sobredefender los países más importantes.

Se parte de la función sigmoidal $f(x) = \frac{1}{1+e^{-x}}$ que se utilizaba para el caso de la probabilidad de éxito de los ataques. Basta con modificar x para que dependa de la relación entre A y S_d.

Esta función no contempla que los países con un único soldado tienen una defensa mucho menor, debido a que solamente se lanza un dado. Para compensar esto, se eleva al cuadrado la defensa obtenida, quedando la defensa de un país como muestra la ecuación 4.34.

$$DefPais(S_d, A) = \begin{cases} \frac{1}{1+e^{\frac{A-S_d}{\sqrt{A+S_d}}}} & \text{si } def > 1, \\ \left(\frac{1}{1+e^{\frac{A-S_d}{\sqrt{A+S_d}}}}\right)^2 & \text{si } def = 1 \end{cases} \tag{4.34}$$

En la gráfica 4.57 puede verse cómo el aumento de la defensa para una amenaza de 10 soldados tiene el efecto deseado.

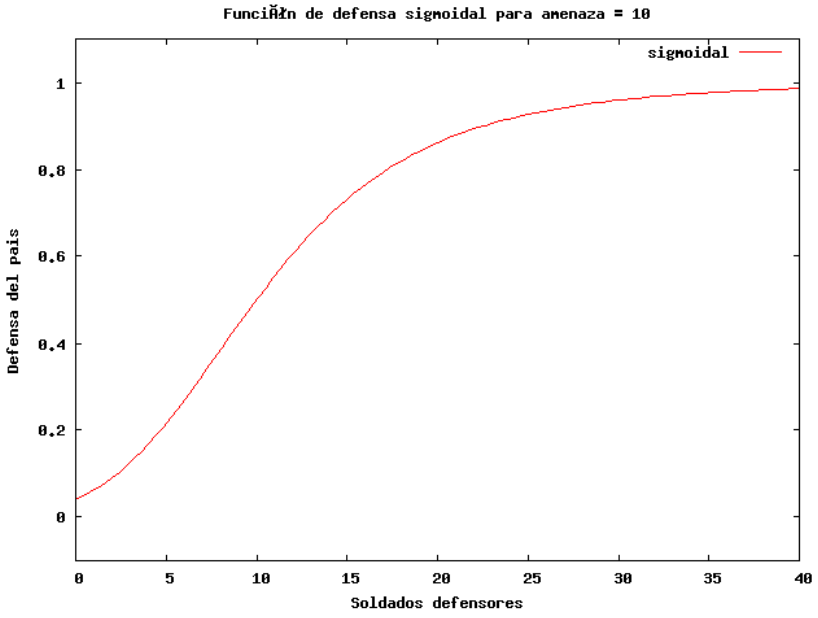


Figura 4.57: Defensa de un país para una amenaza de 10 soldados

Defensa de un continente

Al igual que en el cálculo del dominio del continente en el estado difuso, solamente se tienen en cuenta los países frontera y su defensa se considera independiente, por lo que la defensa del continente se calcula con la ecuación 4.35.

$$Pdef_c = \prod_{p \in Fronteras_c} (Pdef_p) \tag{4.35}$$

Búsqueda de la mejor defensa

La búsqueda de la mejor defensa utiliza la valoración de defensas para encontrar la distribución de los soldados de cada grupo que la maximiza.

Un problema, en el que se deben defender n países y para ello se tienen que asignar soldados agrupados en m grupos de soldados, se define mediante los siguientes elementos:

- Paises a defender: X_i, $1 \leq i \leq n$

- Interés en que cada país sea defendido: I_i, $1 \leq i \leq n$
- Amenaza recibida por cada país frontera desde el exterior: A_i, $1 \leq i \leq n$
- Grupos de soldados que pueden repartirse entre varios países: G_i, $1 \leq i \leq m$
  - Número de soldados a asignar: N_i, $1 \leq i \leq m$
  - Si puede asignarse o no a cada país

En este caso, se define la matriz de asignaciones M_A como:

$$M_A = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{pmatrix}$$

De este modo el problema se reduce a dar un valor a cada x_{ij}. Además, las posibles soluciones a este problema de asignación están sujetas a dos tipos de restricciones:

1. Si el grupo j no puede asignar sus soldados al país i, entonces $x_{ij} = 0$.
2. Un grupo debe asignar exactamente su número de soldados: $\sum_{i=0}^{n} x_{ij} = N_j$

Solución mediante un proceso de búsqueda heurística

Tras descartar otras aproximaciones, como la investigación operativa o los algoritmos evolutivos, se aplica un algoritmo de búsqueda en el espacio de todas las posibles matrices que cumplen las restricciones expuestas.

El algoritmo de búsqueda escogido es el *enforced hill climbing*. Este algoritmo realiza una búsqueda en amplitud desde la matriz actual y cuando encuentra una matriz mejor la convierte en la actual y comienza de nuevo la búsqueda en amplitud. Se define como parámetro de configuración el número de niveles de búsqueda máximos para la búsqueda en amplitud $PC_{NivHill}$. Si en esa búsqueda en amplitud no encuentra ninguna matriz mejor que la actual, esa será la condición de parada.

Este algoritmo es vulnerable a máximos locales, ya que su condición de parada es que el nodo actual no tenga ningún hijo mejor que él, no realiza *backtracking* ni sigue profundizando en el árbol. Por esto, se procurará inicializar la matriz de forma inteligente para que esté lo más cerca posible del óptimo global mejorando el resultado y la velocidad del algoritmo.

Habrá que definir dos partes: la inicialización de la matriz y el proceso de generación de hijos.

Inicialización de la matriz

La inicialización de la matriz intenta alcanzar los siguientes objetivos:

1. **Frenar las amenazas lo antes posible**. Es decir, lo más cerca de la frontera posible.
2. **Cada grupo de soldados se asigna para frenar las amenazas correctas**. En concreto, cada grupo puede asignarse para defender varias amenazas. Es necesario asignar cada amenaza al grupo que menos amenazas pueda atender y cada grupo a las amenazas que puedan ser atendidas por menos grupos.
3. **Amenazas iguales deben intentar atenderse de forma similar**. No tiene sentido dejar de defender una amenaza para atender mejor otra, ya que esto quedará como tarea para el proceso de mejora. Por lo tanto, se intentarán repartir los soldados equitativamente para defender las diferentes amenazas.
4. **Dar preferencia a las amenazas que producen más daños**

El resultado de intentar aplicar estos factores es el algoritmo mostrado en la figura 4.58

Generar matrices hijas

Dada una matriz, se generan matrices parecidas a ella. Para ello se realiza una transferencia de tropas, es decir, se pasan soldados de un país a otro.

```
Para cada amenaza vemos que países tiene que defender en cada nivel.
Para cada amenaza estudiamos el daño que hace por si sola.
Para cada grupo vemos cuantas amenazas pueden atender.
El factor de defensa de un grupo sobre una amenaza = sum_{n=0}^{NumNivelesAmenaza} PaisesNivel_n / n^2

Para cada amenaza (en orden de daños) buscamos frenarla.
 Para cada nivel en el que puede ser frenada
  Para cada país en el que hay que atenderla
   Calculamos la amenaza que debe frenarse
   Para cada grupo (en orden de amenazas que pueden atender)
    Si podemos atender el país lo atendemos
 Actualizamos numero de amenazas que puede frenar cada grupo
```

Figura 4.58: Algoritmo de inicialización de defensas

Para cada pareja de países (p, q) hay $G_t \in [0, m]$ grupos que pueden asignar soldados a ambos países. Todos estos grupos pueden transferir hasta $x_{g,p}$ soldados al país q. Esto significa que hay una transferencia por cada soldado del grupo y cada país al que puede ser transferido, por lo que no pueden utilizarse todas.

Para descartar transferencias se toman dos medidas:

Coger sólo el grupo que más soldados puede transferir Para cada pareja de países (p, q) sólo se consideran las transferencias del grupo G que maximiza $x_{G,p}$ con $q \in G$.

Considerar un subconjunto de transferencias El grupo G puede pasar de 1 a $x_{G,p}$ soldados pero solamente selecciona un subconjunto, para lo que se utilizan dos criterios:

1. Transferencias rango: Se seleccionan $PC_{NumTransfRango}$ transferencias. Siempre se seleccionan la transferencia de un soldado y la de todos los soldados. En caso de que haya más transferencias se seleccionan equidistantes.

2. Transferencia mejor: Se supone que las amenazas que recibe cada país son independientes y se busca la transferencia que maximiza la suma de las defensas de

ambos países.

Determinar amenazas de defensa

Las amenazas con las que trabaja el algoritmo de defensas son distintas de las calculadas por el algoritmo de amenazas, por lo que deben recalcularse. Las diferencias entre los dos tipos de amenaza son:

- Las amenazas de defensas solamente tienen un país y una fuerza.
- El algoritmo de defensas solamente considera las amenazas sobre los países frontera.
- El algoritmo de defensas sólo considera una amenaza por país.

Por lo tanto, hay que juntar todas las amenazas calculadas en el algoritmo de amenazas en una amenaza por cada país frontera y calcular su fuerza. Para esto, todas las amenazas que contengan una frontera se asignan a la primera frontera que atacan. En la figura 4.59 se muestra un pseudocódigo del algoritmo.

```
F es el conjunto de países frontera

Para cada jugador enemigo e
   Para cada país frontera P_f ∈ F
  //fuerzasRealizadasPaisPais [paisOrigen][PaisDestino]
  Para cada amenaza que contenga P_f y no tenga otro p ∈ F antes
   fuerzasAmenazasPais[paisDestino][amenaza.paisOrigen] += fuerzaAmenaza

  fuerzaAmenazasPaisTotal = unionAmenazas(fuerzasAmenazasPais[paisDestino])

  amenazasCalculadas[Pf][e] = max(1, fuerzaAmenazasPaisTotal)

amenaza[Pf] = unionAmenazas(amenazasCalculadas[Pf])
```

Figura 4.59: Algoritmo que determina las amenazas de defensa

Una amenaza i con S_i soldados, P_i probabilidad de realización y Q_i países conquistados de los cuales $Qdef_i$ pertenecen al defensor se transforman en una amenaza de fuerza A_i como indica la ecuación 4.36.

$$A_i = S_i P_i \frac{Qdef_i}{Q_i} \tag{4.36}$$

Par unir las fuerzas de las amenazas que llegan a la frontera, se siguen las siguientes reglas:

1. Las del mismo país de origen se suman.

2. Las amenazas recibidas de países distintos se unen.

Unión de amenazas La unión de amenazas junta varias amenazas en una. Una opción sería sumarlas, pero esto sería erróneo puesto que debe ser mayor una amenaza de fuerza 30 que tres de 10. Por consiguiente, se ordenan las amenazas por su fuerza en orden descendente de modo que A_0 es la mayor y se define el parámetro de configuración PC_{RedAm} que define el peso de la amenaza i como $RedAm^i$. La ecuación 4.37 muestra como calcular la unión de las amenazas.

$$U_{am} = \sum_{i=0}^{N} A_i^{PC_{RedAm}} \tag{4.37}$$

Camino de las tropas

El algoritmo de camino de las tropas completa la información de defensas especificando desde qué país se atiende cada asignación de defensas. Esto es necesario para calcular la probabilidad de éxito del plan y para poder ejecutarlo.

Los factores que se intenta optimizar son:

- Maximizar la probabilidad de éxito de los ataques: Dedicar al ataque mayor número de soldados que los que indica su factor de riesgo.

- Maximizar la capacidad de alcanzar las defensas deseadas, aun si los ataques no salen como los esperados. Para esto, es beneficioso situar los soldados en países que puedan transferir soldados a varios países.

Para obtener el camino de las tropas se calcula, para cada grupo de soldados que deben asignarse a un país y pueden asignarse desde varios países, el beneficio obtenido por asignarlo desde cada país. Los soldados se *reparten entre los países proporcionalmente a dicho beneficio*. Dado que el beneficio se calcula pra cada grupo por separado y depende de la asignación del resto de grupos, el cálculo se repite durante $PC_{NumItCam}$ iteraciones.

```
Mientras no se alcance el número de iteraciones máximas
 Para cada asignación
  Obtener los países desde los que se puede atender
  Calcular el valor para cada país dadas el resto de asignaciones
  Repartir los soldados proporcionalmente al beneficio
```

Figura 4.60: Algoritmo de camino de las tropas

La ecuación 4.38 muestra como calcular el beneficio que aporta el atender una asignación de soldados desde un país, B, como la media ponderada de beneficio de cada uno de los factores mencionados: probabilidad de éxito de los ataques B_a y capacidad de redistribución de tropas B_r.

$$B = PC_{PCamA} B_a + PC_{PCamR} B_r \tag{4.38}$$

Beneficio por probabilidad de los ataques El beneficio por probabilidad de los ataques es la suma de la probabilidad de éxito de cada árbol de ataques del plan, dado el número de soldados que llegan a cada país según esa asignación. Esta puede ser aproximada tal y como se explica en la sección 4.3.7, página 88.

Beneficio por redistribución de los caminos Ante un imprevisto en el resultado de los ataques puede ser bueno cambiar el país asignado para un grupo de soldados. Por esto, en la elección del camino es positivo situar los soldados en países desde los que

pueden llegar a otros países mediante fortificaciones. Esto es especialmente importante en aquellos países con pocos defensores puesto que es más probable que se queden sin defensas si sus soldados se pierden en ataques.

Por lo tanto, para cada país se suma el número de soldados que podrían fortificarse hasta él, C_i dividido entre el número de defensores planeado para ese país D_p.

La capacidad de redistribución se mide según la ecuación 4.39:

$$B_r = \sum_{p=0}^{P} \frac{C_p + \sum_{v \in Vec(p)} C_v}{D_p} \tag{4.39}$$

Parametro	Descripción	Valores	Valor
$PC_{NivHill}$	Número de niveles de búsqueda en amplitud dentro del algoritmo *enforced hill climbing*.	$[0, \infty)$	0
$PC_{NumTransfRango}$	Número de transferencias que se consideran al generar los hijos de una matriz de asignación.	$[0, \infty)$	0
$PC_{VDefCont}$	Valor aportado por defender todos los países de un continente.	$[0, \infty)$	10000
$PC_{ExpDivAmVec}$	Exponente al que se eleva el número de vecinos de un país para reducir la amenaza al pasarla a varios países.	$[0, 1]$	0,5
PC_{RedAm}	Parámetro que permite reducir la fuerza de las amenazas al unirlas.	$[0, 1]$	0,5
PC_{PCamA}	Peso de la probabilidad de los ataques en el cálculo de los caminos de las tropas	$[0 - \infty)$	10
PC_{PCamR}	Peso de la redistribución de soldados en el cálculo de los caminos de las tropas	$[0 - \infty)$	1
$PC_{NumItCam}$	Número de iteraciones en el cálculo del camino de las tropas	$[1 - \infty)$	2

Tabla 4.7: Parámetros del algoritmo de camino de las tropas

4.5.6. Fuerza de un jugador

Según los trabajos anteriores (véase la sección 2.4.2, página 16) los factores que influyen en la fuerza del jugador son:

1. Maximizar el numero de soldados actuales.
2. Maximizar el número de refuerzos esperados en el siguiente turno.
3. Minimizar el número de países frontera.
4. Maximizar el número de soldados en los países frontera.
5. Maximizar el soporte logístico a los países frontera.

En este proyecto los tres últimos factores, relativos a los países frontera del jugador, están considerados implícitamente al realizar las predicciones a corto y largo plazo.

Una clasificación lógica de los factores es agruparlos por los datos del análisis de la posición que se utilizan para calcularlos. De este modo, la fuerza de un jugador j se compone por su fuerza actual Fa_j, fuerza futura inmediata Ffi_j y fuerza futura lejana Ffl_j como se muestra la ecuación 4.40:

$$F_j = Fa_j + Ffi_j + Ffl_j \tag{4.40}$$

Fuerza actual

La fuerza actual se calcula a partir del estado base del plan y su información de defensas y mide las fuerzas que tiene el jugador en esa posición:

1. Número de países ($NumPais_j$)
2. Número de soldados ($NumSold_j$)
3. Número de cartas ($NumCartas_j$): mide el beneficio de conseguir una carta atacando y varias matando a otro jugador. Se multiplica por el valor de canjear cartas V_{Cartas} dividido entre el número de cartas necesario para canjear, 3.

No se tienen en cuenta los continentes porque no aportan ningún beneficio si no se defienden hasta el siguiente turno.

Por lo tanto, la fuerza actual se calcula mediante la ecuación 4.41:

$$Fa_j = NumPais_j PC_{FaPais} + NumSold_j PC_{FaSold} + NumCartas_j PC_{FaCartas} \frac{V_{Cartas}}{3} \tag{4.41}$$

Fuerza futura inmediata

La fuerza futura inmediata se calcula a partir de la posición estimada en la predicción a corto plazo para el inicio del turno del jugador, donde $P(p \in j)$ y $P(c \in j)$ son la probabilidad de que el jugador j tenga el país p y el continente c, respectivamente. Se obtiene a partir de:

1. Países defendidos ($PaisDef_j$): Permite que el agente intente tener todos los países que pueda defender. La suma de la probabilidad de tener cada país en esa posición:

$$PaisDef_j = \sum_{p=0}^{P} P(p \in j)$$

2. Soldados supervivientes ($SoldDef_j$): Permite que el agente intente retirarse de las zonas de conflicto en las que pueden morir muchos soldados. Se asume que sobreviven todos los soldados de los países defendidos:

$$SoldDef_j = \sum_{p=0}^{P} (S_p P(p \in j))$$

3. Refuerzos conseguidos (Ref_j): Permite que el agente intente conquistar continentes y países para recibir más soldados. Se recibe un soldado por cada tres países con un mímimo de tres y el bono de los continentes que tiene el jugador:

$$Ref_j = max(3, \lfloor \frac{PaisDef_j}{3} \rfloor) + \sum_{c=0}^{C} (P(c \in j) R_c)$$

Por lo tanto, la fuerza futura inmediata se calcula mediante la ecuación 4.42:

$$Ffi_j = PaisDef_j PC_{FfiPais} + SoldDef_j PC_{FfiSold} + Ref_j PC_{FfiRef} \tag{4.42}$$

Fuerza futura lejana

La fuerza futura lejana se calcula a partir del dominio a medio plazo del jugador, donde $D_{j,p}$ y $D_{j,c}$ son el dominio a medio plazo del jugador sobre el país p y el continente c, respectivamente. Se obtiene a partir de:

1. Países dominados (*PaisDom*): Permite que el agente se extienda por zonas que va a poder mantener a medio plazo. Se define como:

$$PaisDom = \sum_{p=0}^{P} D_{j,p}$$

2. Continentes dominados (*ContDom*): Permite que el agente ataque continentes que va a poder mantener a medio plazo. Se define como:

$$ContDom = \sum_{c=0}^{C} (D_{j,c} R_c)$$

Por lo tanto, la fuerza futura lejana se calcula mediante la ecuación 4.43:

$$Ffl_j = PaisDom_j PC_{FflPais} + ContDom_j PC_{FflCont} \tag{4.43}$$

4.5.7. Refinamiento del plan

El refinamiento del plan se encarga de completar la información de defensas del plan escogido en dos casos:

1. Al escoger el plan.
2. Al conquistar un país.

Parametro	Descripción	Valores	Valor
PC_{FaPais}	Valor país actual	$(-\infty, \infty)$	1
PC_{FaSold}	Valor soldado actual	$(-\infty, \infty)$	1.5
$PC_{FaCartas}$	Valor cartas	$(-\infty, \infty)$	15
$PC_{FfiPais}$	Valor país corto plazo	$(-\infty, \infty)$	2
$PC_{FfiSold}$	Valor soldado corto plazo	$(-\infty, \infty)$	3
PC_{FfiRef}	Valor refuerzos corto plazo	$(-\infty, \infty)$	1000
$PC_{FflPais}$	Valor país medio plazo	$(-\infty, \infty)$	5
$PC_{FflCont}$	Valor continente medio plazo	$(-\infty, \infty)$	200

Tabla 4.8: Parámetros del algoritmo de fuerza de un jugador

Refinamiento del plan al escogerlo La información de defensas del plan calculada en su valoración estaba muy limitada para reducir el tiempo del análisis. Aquí se realizan de nuevo los algoritmos de amenazas y defensas para obtener las defensas definitivas del plan. Estos algoritmos están configurados con otros parámetros que les permiten ser más exactos.

Refinamiento del plan al conquistar un país Al conquistar un país, hay que decidir cuantos soldados se traspasan. La información de defensas ha supuesto el coste medio del ataque, por lo que hay que actualizarla con el número de supervivientes real.

La predicción de amenazas calculada en el primer refinamiento del plan sigue siendo válida y no es necesario recalcularla pero, al cambiar el número de soldados esperado/real, puede que el reparto calculado ya no sea óptimo.

Por lo tanto, los pasos a seguir son:

1. Ajuste de los soldados del grupo asociado al país origen del ataque proporcionalmente al reparto anterior. Si los soldados del grupo dan negativo es que no quedan soldados suficientes para hacer los ataques planificados y debe replanificarse.

2. Algoritmo de defensa para optimizar las defensas según el nuevo número de soldados.

Parametro	Descripción	Valores	Valor
$PC_{NumMaxAm}$	Número máximo de amenazas expandidas	$[1, \infty)$	1000
PC_{MinEx}	Mínima probabilidad de éxito para extender la amenaza.	$[0, 1]$	0,8
$PC_{MinPUtilMax}$	Porcentaje mínimo sobre la utilidad máxima que debe tener la utilidad máxima esperada de una amenaza para ser valorada.	$[0, 1]$	0,2
$PC_{MinProbEsper}$	Probabilidad mínima esperada que debe tener una amenaza para ser valorada.	$[0, 1]$	0,01
$PC_{NivHill}$	Número de niveles de búsqueda en amplitud dentro del algoritmo *enforced hill climbing*.	$[0, \infty)$	1
$PC_{NumTransfRango}$	Número de transferencias que se consideran al generar los hijos de una matriz de asignación.	$[0, \infty)$	5

Tabla 4.9: Parámetros del algoritmo de refinamiento del plan

4.5.8. Prioridad del plan

La prioridad del plan es la heurística que guía el algoritmo de búsqueda para encontrar el mejor plan.

Es importante explorar buenos planes porque, aunque la función de valoración de un plan sea perfecta, sino se examinan los mejores planes no serán seleccionados.

La prioridad del plan se calcula a partir de tres factores:

1. Valor del padre (V_{padre}): Intenta que se profundice en los planes que van mejor. Sin embargo, un peso demasiado alto puede hacer que la búsqueda se estanque en máximos locales.
2. Prioridad de país conquistado (P_p): Dirige la búsqueda hacia los países más interesantes.
3. Estructura del ataque (Ea): Intenta que se realicen los ataques desde el mejor país origen.

Antes de combinar estos factores, se les aplica la normalización de la ecuación 4.44 para que todos tengan un valor entre 0 y 1 y sus pesos se ajusten correctamente:

$$N(x) = \frac{x - min(x_i)}{max(x_i) - min(x_i)} \tag{4.44}$$

La ecuación 4.45 muestra la prioridad de atacar el país p dado el plan $Padre$:

$$Pplan = N(V_{padre})PC_{Vp} + N(P_p)PC_{Pp} + N(Ea)PC_{Ea} \tag{4.45}$$

El problema de realizar la normalización mencionada es que, al recibir un factor que es un nuevo máximo, cambia la prioridad de toda la lista y debe reordenarse. Se define como parámetro de configuración PC_{Nr}, que marca cuantos nodos deben estar desordenados para reordenar la lista.

El retrasar la ordenación tiene dos efectos positivos:

1. Ahorra tiempo gracias a evitar reordenar la lista cada vez que se examina un plan.

2. Examina más planes antes de saltar al nuevo plan con mejor valor. De este modo, aunque reduce un poco la profundidad de los planes examinados, aumenta las probabilidades de examinar planes diferentes.

Prioridad del país conquistado

La prioridad de conquistar el país p depende de los siguientes factores:

1. Interés del dominio a medio plazo (Id_p): el interés de ese país calculado en la predicción a medio plazo.

2. Interés de conquistar el país (Ic_p): el interés de conquista de ese país calculado en la predicción a corto plazo.

3. Utilidad de matar al enemigo (U_e): La probabilidad de poder conquistar todos los países del enemigo E por el valor de lograrlo:

$$U_e = Pconq(E) NumCartas_e \frac{V_{Cartas}}{3}$$

4. Utilidad de conquistar el continente (U_c): La probabilidad de poder conquistar todos los países del continente C por los refuerzos que aporta:

$$U_c = Pconq(C) R_C$$

5. Reducción de fronteras (U_f): El número de fronteras que reduce, es decir el número de países vecinos cuya única frontera es el país conquistado menos uno si tiene algún vecino enemigo. Solamente se tiene en cuenta si es positivo:

$$U_f = max(0, NumVecinosUnicaFrontera - EsFrontera)$$

La ecuación 4.46 calcula la prioridad del país como la media ponderada de todos estos factores.

$$P_p = Id_p PC_{Id} + Ic_p PC_{Ic} + U_e PC_e + U_c PC_c + U_f PC_f \tag{4.46}$$

La probabilidad de poder conquistar un conjunto de países objetivo O, $Pconq(O)$ se obtiene con el algoritmo de la figura 4.61.

```
P_1 a P_n son los países propios
O_1 a O_m son los países objetivo
S_1 a S_n es el número de soldados en los países propios menos 1

C_q(p) = 1 + CosteAtaque(p): coste de conquistar el país enemigo p
C_q(p) = infinito si p es del jugador

Se calcula el camino mínimo entre países con el algoritmo de Floyd:
 C(P_i,O_j) es el camino mínimo desde P_i hasta O_j
 C(O_i,O_j) es el camino mínimo desde O_i hasta O_j

SoldadosNecesarios = 0
Mientras queden países en la lista de objetivos
 Coger i y j que minimizan V = (C(P_i,O_j) - S_i)
 SoldadosNecesarios += max(0,V)
 S_i = max(0, Si - C(P_i,E_j))
 Para todo k: C(P_i,O_k) = min(C(P_i,O_k), C(O_j,O_k))
 Eliminar O_j de la lista de objetivos

ProbConquistarObjetivos = 1 - DefPaisSigmoidal (SoldadosNecesarios,
                                                PosicionamientosTurno)
```

Figura 4.61: Algoritmo de probabilidad de poder conquistar paises

Estructura del ataque

Generalmente, para mejorar las probabilidades de éxito del ataque y distribuir bien los soldados, es mejor atacar los países en listas que en árboles. Esto es así porque los árboles requieren dividir las tropas por varios caminos.

Por esto, el valor de la estructura del ataque es 0 si el nuevo ataque genera una nueva rama en el árbol de ataques, y 1 en caso contrario.

Parametro	Descripción	Valores	Valor
PC_{Vp}	Peso del valor del plan padre	$(-\infty, \infty)$	1.5
PC_{Pp}	Peso de la prioridad del país	$(-\infty, \infty)$	2
PC_{Ea}	Peso de la estructura de ataques	$(-\infty, \infty)$	1
PC_{Id}	Peso del interés de dominio	$(-\infty, \infty)$	5
PC_{Ic}	Peso del interés de conquista	$(-\infty, \infty)$	15
PC_e	Peso del factor de matar al enemigo	$(-\infty, \infty)$	1000
PC_c	Peso del factor de conquistar continente	$(-\infty, \infty)$	750
PC_f	Peso del factor de reducir fronteras	$(-\infty, \infty)$	500
PC_{Nr}	Número de nodos desordenados hasta reordenar la lista de planes.	$[0, \infty)$	5

Tabla 4.10: Parámetros del algoritmo de prioridad del plan

4.5.9. Ejecución del plan

La ejecución del plan selecciona las acciones concretas que se realizarán durante el turno para cumplir el plan, reaccionando de forma flexible a los diferentes sucesos aleatorios que pueden darse durante la realización de los ataques. Además, durante el proceso de ejecución, se deberá comprobar que el plan sigue siendo factible y, en caso contrario, replanificar diseñando un nuevo plan.

La figura 4.62 muestra un esquema de las fases del turno y las decisiones de ejecución asociadas con cada una de ellas.

Canjear cartas o no

Como la utilidad de reservar los soldados es difícil de calcular y lo más común es que lo mejor sea obtener los soldados lo antes posible, se simplifica esta decisión y siempre se canjean las cartas en cuanto sea posible.

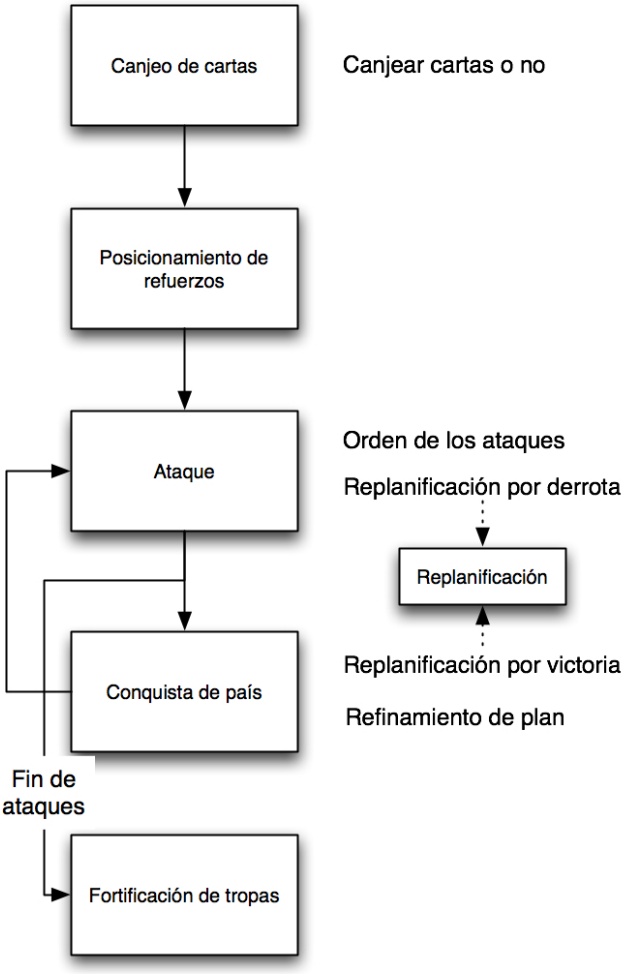


Figura 4.62: Decisiones de ejecución en cada fase de la partida

Orden de los ataques

Cada vez que se ataca, el resultado aporta información útil para decidir si replanificar o no, por lo que el orden de los ataques influye en la capacidad de replanificación del agente. Lo mejor sería realizar primero los ataques más importantes (que deben ejecutarse con mayor probabilidad). Sin embargo, no es sencillo determinar la importancia de un ataque por lo que se utiliza otro criterio.

Si va a ser necesario replanificar, es mejor hacerlo lo antes posible para poder decidir si incluir o no los ataques planificados inicialmente. Por esto, la estrategia seguida es *seleccionar el árbol de ataques que tenga menor probabilidad de éxito.*

Para seleccionar un ataque dentro del árbol de ataques, se escoge el país con más soldados defensores. Ningún jugador humano realiza sus ataques en este orden porque atacar cada vez a un país distinto resulta una tarea pesada. Sin embargo, eso no es un inconveniente para el agente y hace que se pueda considerar que los ataques son simultáneos, lo que aumenta la probabilidad de éxito.

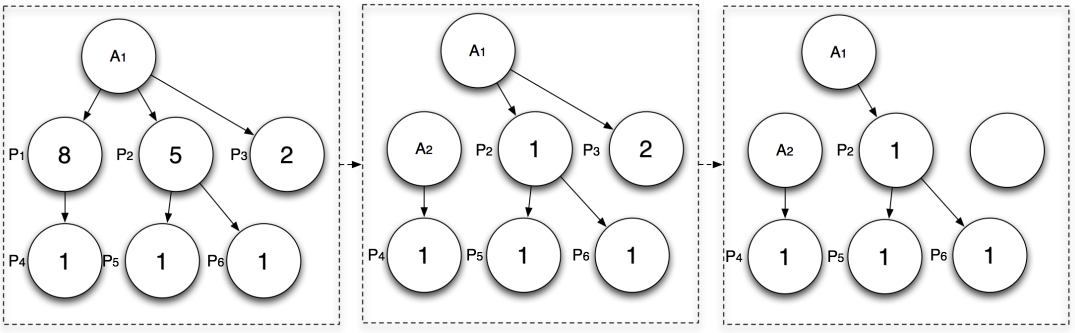


Figura 4.63: Ejemplo del orden de los ataques

Ejemplo del orden de los ataques La figura 4.63 muestra un ejemplo del orden que seguirían los ataques suponiendo que todos los ataques los gana el atacante muriendo dos soldados defensores.

En primer lugar atacaría P_1 hasta quedar 4 soldados, atacaría P_2 y seguiría alternando ataques hasta conquistar P_1. En ese momento los otros dos países no conquistados

tendrían uno o dos soldados.

A continuación escogería A_1 ya que su probabilidad de éxito es menor y atacaría P_3 por tener más soldados.

Replanificación

Mientras se ejecuta el plan es posible que los resultados de los ataques provoquen que el plan ya no sea la mejor opción. En ese caso debe rehacerse el plan.

Lo ideal sería aprovechar los análisis realizados en la búsqueda del mejor plan para acelerar la generación del nuevo plan. Sin embargo, no hay un modo sencillo de hacerlo, por lo que en este proyecto la replanificación consiste en crear un plan entero desde cero, consumiendo el doble de tiempo en el turno.

Replanificación por derrota Al perder soldados tras realizar un ataque, debe comprobarse si realizar una replanificación por tener menos soldados restantes de los esperados. Para ello, basta con que se cumpla alguna de las siguientes condiciones:

- Hay una gran diferencia entre los soldados perdidos S_m y el coste realizado C_{real}, lo que puede provocar que los países no puedan ser bien defendidos. El coste realizado es el coste estimado en el plan inicial menos el coste estimado en el plan actual. Replanifica si:

$$S_m - C_{real} > PC_{LimCost} \text{ y } \frac{S_m}{C_{real}} > 1 + PC_{LimPorcCost}$$

- La misma condición que la anterior pero contando ahora los soldados perdidos y coste realizado en una sola rama del árbol. En este caso se utilizan $PC_{LimCostRama}$ y $PC_{LimPorcCostRama}$. En los nodos que tienen varias continuaciones, el coste realizado en la parte superior del árbol se reparte proporcionalmente al coste de cada rama.

- Mientras realiza el ataque a un país, se detecta que no quedan suficientes soldados para realizar los ataques planificados. Se puede detectar a través de la diferencia

entre soldados atacantes S_a y el factor de riesgo restante. Replanifica si:

$$FR_{restante} - S_a > PC_{DifFR} \text{ y } \frac{FR_{restante}}{S_a} > 1 + PC_{DifPorcFR}$$

- Al conquistar un país, durante el refinamiento del plan, se detecta que no quedan suficientes soldados para realizar los ataques planificados.
- No hay soldados suficientes para seguir atacando. El número mínimo de soldados para atacar es 4 (tirada de 3 dados) si el contrario defiende con dos soldados y 3 (tirada de 2 dados) si el contrario defiende con un solo soldado.

Replanificación por victoria Al conquistar un país se comprueba si se tienen muchos más soldados de los esperados. Replanifica si:

$$S_m - C_{real} < PC_{LimCostVic} \text{ y } \frac{S_m}{C_{real}} < 1 - PC_{LimPorcCostVic}$$

Parametro	Descripción	Valores	Valor
$PC_{LimCost}$	Límite mínimo de la diferencia de coste para replanificar.	$[1 - \infty)$	7
$PC_{LimPorcCost}$	Límite mínimo en porcentaje de la diferencia de coste para replanificar.	$(0 - \infty)$	0,6
$PC_{LimCostRama}$	Límite mínimo de la diferencia de coste en una rama para replanificar.	$[1 - \infty)$	5
$PC_{LimPorcCostRama}$	Límite mínimo en porcentaje de la diferencia de coste en una rama para replanificar.	$(0 - \infty)$	0,5
PC_{DifFR}	Diferencia entre el número de soldados y el factor de riesgo de los ataques que debe haber para replanificar.	$[1 - \infty)$	3
$PC_{DifPorcFR}$	Diferencia en porcentaje entre el número de soldados y el factor de riesgo de los ataques que debe haber para replanificar.	$(0 - \infty)$	0,5
$PC_{LimCostVic}$	Límite mínimo de la diferencia de coste para replanificar por victoria.	$[1 - \infty)$	3
$PC_{LimPorcCostVic}$	Límite mínimo en porcentaje de la diferencia de coste para replanificar por victoria.	$(0 - \infty)$	0,5

Tabla 4.11: Parámetros del algoritmo de replanificación

4.6. Manual de usuario

Para poder utilizar el agente desarrolado en el proyecto basta con seguir los siguientes pasos:

1. Instalar el software Lux Delux [3]
2. Introducir el agente Ender en Lux Delux. Basta con copiar los archivos `.class` de la aplicación y el fichero de configuración `configEnder.txt` en el directorio Agents de Lux:
   - Windows ⇒ Support/Agents/ en la carpeta de instalación de Lux. (p.e. C:/Program Files/Lux/Support/Agents/)
   - Mac ⇒ /Library/Application Support/Lux/Agents/
3. Iniciar el software Lux Delux y seleccionar al agente "Ender" en el menú de selección de jugadores.

Para obtener más información acerca de cómo ejecutar partidas en el software Lux Delux consulte su manual de usuario.

4.7. Manual de referencia

En este manual se dan indicaciones para poder continuar el trabajo del agente, dando los pasos para realizar algunas modificaciones típicas.

4.7.1. Cambiar parámetros de configuración

Para modificar los parámetros de configuración del agente descritos en esta memoria basta con editar el fichero de configuración configEnder.txt situado en la misma carpeta que Ender.

[3] Lux Delux puede ser descargado libremente de su página web: http://sillysoft.net/lux/

Este fichero consta de pares 'atributo = valor'. Para modificar un parámetro debe modificarse su valor. El árbol de parámetros no resulta trivial, por lo que se recomienda consultar las tablas de parámetros incluídas en el desarrollo de las heurísticas para saber qué parámetros se desean modificar.

Por ejemplo, para aumentar el número de amenazas calculadas para cada país en el refinamiento del plan se edita la línea mostrada en la figura 4.64.

```
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.maxNumAmenazasPais=1000
```

Figura 4.64: Línea de fichero de configuración

4.7.2. Nuevos algoritmos

Para añadir un nuevo algoritmo al sistema es necesario implementar `PConfigNombre` y `AlgNombre`, siguiendo las plantillas de las figuras 4.65 y 4.66, respectivamente.

La mayoría de los algoritmos solamente tienen una función. `ParamFunc` puede ser reemplazado por los parámetros que se deseen (uno o varios y de cualquier tipo). `ResFunc` también puede ser de cualquier tipo, `void` incluido.

4.7.3. Nuevas implementaciones de algoritmos

Para agregar una nueva implementación a un algoritmo se crea `ImplAlg` según la plantilla de la figura 4.67.

Además, es necesario editar `PConfigNombre` para incluir la nueva implementación y sus parámetros:

1. Añadir en `tiposAlgoritmo` el nombre del nuevo tipo.
2. Añadir en `getAlgoritmo` el constructor de la nueva implementación.
3. Añadir en `getParametros` el constructor de los parámetros, si los hay.

```
public class PConfigAmenazas extends ParametroConfiguracionAlgoritmo{
  static String [] tiposAlgoritmo = {"NombreImpl1", "NombreImpl2" ,...};

  public PConfigAmenazas(String nombre) {
    super(nombre);
  }

  protected Algoritmo getAlgoritmo(String tipoAlgoritmo) {
    if(tipoAlgoritmo.equals(tiposAlgoritmo[0])){
      return new NombreImpl1();
    }else if(tipoAlgoritmo.equals(tiposAlgoritmo[1])){
      return new NombreImpl2();
    ...
    }else{
      return null;
    }
  }

  protected ParametrosConfiguracion
                  getParametrosConfiguracion(String tipoAlgoritmo) {
    if(tipoAlgoritmo.equals(tiposAlgoritmo[0])){
      return new ParametrosConfiguracionNombreImpl1();
    }else if(tipoAlgoritmo.equals(tiposAlgoritmo[1])){
      return new ParametrosConfiguracionNombreImpl2();
    ...
    }else{
      return null;
    }
  }

  public String[] getTiposAlgoritmo() {
    return tiposAlgoritmo;
  }
}
```

Figura 4.65: Plantilla de `PConfigNombre`

```
public abstract class AlgNombre extends Algoritmo{
    public abstract ResFunc1 funcion1(ParamFunc1 param);
    public abstract ResFunc2 funcion2(ParamFunc2 param);
    ...
}
```

Figura 4.66: Plantilla de `AlgNombre`

```
public class ImplAlg1 extends AlgNombre{
    ParametrosConfiguracion ImplAlg1 parametros;
    public ResFunc1 funcion1(ParamFunc1 param){
        //Implementacion
    }
    public void setParametrosConfiguracion(ParametrosConfiguracion config){
        this.parametros = (ParametrosConfiguracionImplAlg1) config;
    }
}
```

Figura 4.67: Plantilla de `ImplAlg`

4.7.4. Utilizar parámetros u otros algoritmos en un algoritmo

Para utilizar parámetros u otros algoritmos se utiliza `ParametrosConfiguracionImpl`. En la figura 4.68 se muestra cómo incluir un parámetro `double` y un algoritmo en los parámetros de configuración de la implementación.

4.7.5. Recoger datos de algoritmos

Para recoger datos del algoritmo, en su implementación deben incluirse algunas llamadas a métodos heredados desde la clase `Algoritmo`:

1. Tiempo: `this.iniciarMetodo()` y `this.finalizarMetodo()` al inicio y al final de la parte cuyo tiempo desea medirse, respectivamente.

2. Otros datos: `this.addDato(new DatoDouble("nombreDato", dato))`.

```
public class ParametrosConfiguracionImpl
                            implements ParametrosConfiguracion {

  private ParametroConfiguracionDouble parametroReal;
  private PConfigNombreOtro algoritmoOtro;

  public ParametrosConfiguracionExtenderAmenazasPorCoste() {
    parametroReal = new ParametroConfiguracionDouble("parametroReal");
    algoritmoOtro = new PConfigNombreOtro("algoritmoOtro");
  }

  public ArrayList<ParametroConfiguracion> getParametrosConfiguracion() {
    ArrayList<ParametroConfiguracion> parametros =
                                    new ArrayList<ParametroConfiguracion>();
    parametros.add(parametroReal);
    parametros.add(algoritmoOtro);
    return parametros;
  }

  public AlgNombreOtro getAlgoritmoOtro() {
    return (AlgNombreOtro)(algoritmoOtro.getAlgoritmoConfigurado());
  }

  public double getParametroReal() {
    return parametroReal.getValor();
  }
}
```

Figura 4.68: Plantilla de `ParametrosConfiguracionImpl`

4.7.6. Incluir el agente en otros juegos

Incluir el agente en otros juegos depende de la aplicación externa en la que se desea integrar. Dentro del sistema, lo único importante es implementar una clase que hereda de `AplicacionRisk` e implementa sus métodos enviando las ordenes correspondientes a la aplicación.

Juegos en otros lenguajes o plataformas Para acoplar el agente con juegos que no estén realizados en Java puede implementarse una clase `AplicacionRiskSocket` que se comunica mediante sockets con la aplicación. En el juego externo debería implementarse otro `AgenteSocket` que envíe las ordenes del juego y reciba los movimientos que el agente quiere realizar.

Capítulo 5

Resultados

El capítulo de resultados realiza pruebas sobre el agente y evalua su rendimiento.

En primer lugar se prueban los algoritmos de defensas. Después, para el rendimiento del agente se realizan partidas contra otros agentes y se evalúan sus resultados y los tiempos que tarda cada proceso del agente. Por último, se hacen algunas observaciones acerca del comportamiento inteligente que tiene el agente y su comparación con los rivales a los que se ha enfrentado.

5.1. Pruebas del algoritmo de defensas

Antes de pasar a evaluar el agente en su conjunto, se han realizado pruebas para evaluar su algoritmo de defensas.

5.1.1. Definición de los casos de prueba

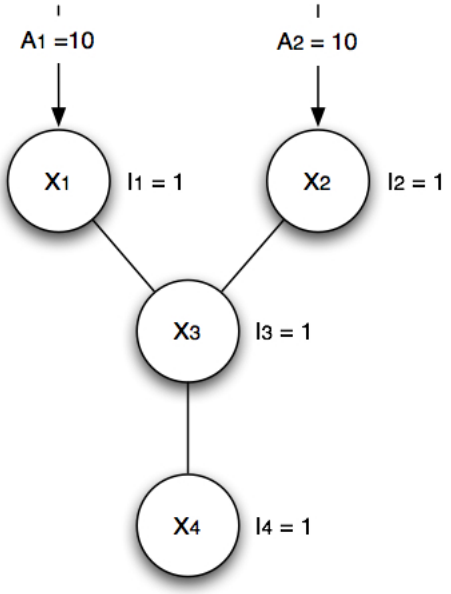
Los algoritmos de defensas son evaluados en un conjunto de casos de prueba, que se definen por:

- Países poseidos y conexiones entre ellos.
- Interés de cada país.

- Amenazas externas recibidas en cada país.
- Grupos de soldados asignables (número de soldados y posibles países asignables).

Para cada caso de prueba definido, se aportan un esquema y una tabla. En el esquema se muestra información acerca de los países poseídos y sus conexiones, las amenazas recibidas en cada país y el interés en defender cada uno de los países. En la tabla se aporta información acerca de los grupos de soldados que se poseen, con su número de soldados y los países a los que puede asignarse.

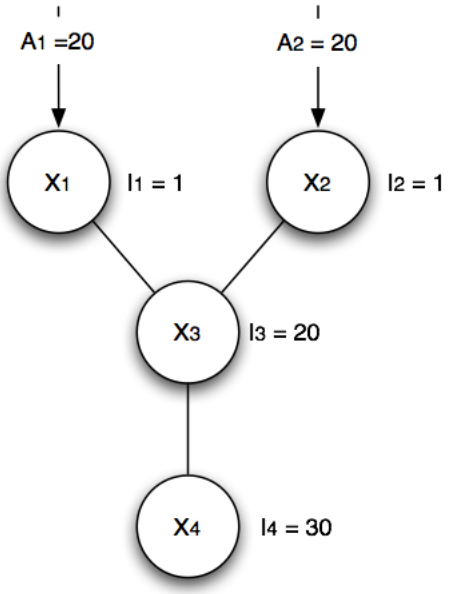
Caso de prueba 1



Grupo	Soldados asignables	Países asignables
G1	15	Todos
G2	13	X_2, X_3

El caso de prueba 1 tiene como objetivo comprobar que se defiende en las fronteras repartiendo los soldados entre los países frontera de forma razonable (a las mismas amenazas se pone el mismo número de soldados)

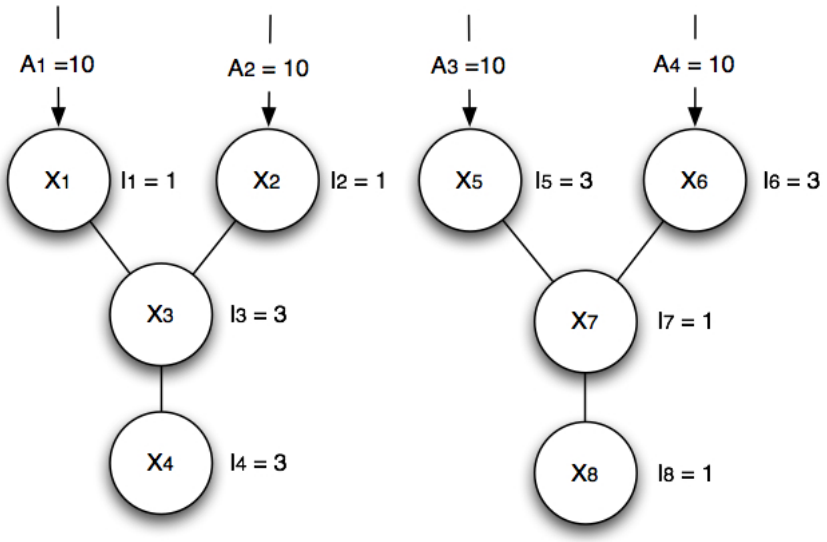
Caso de prueba 2



Grupo	Soldados asignables	Países asignables
G1	10	Todos
G2	15	X_1, X_3

El caso de prueba 2 tiene como objetivo comprobar si es capaz de defender un conjunto de países en un país interior a costa de no defender las fronteras.

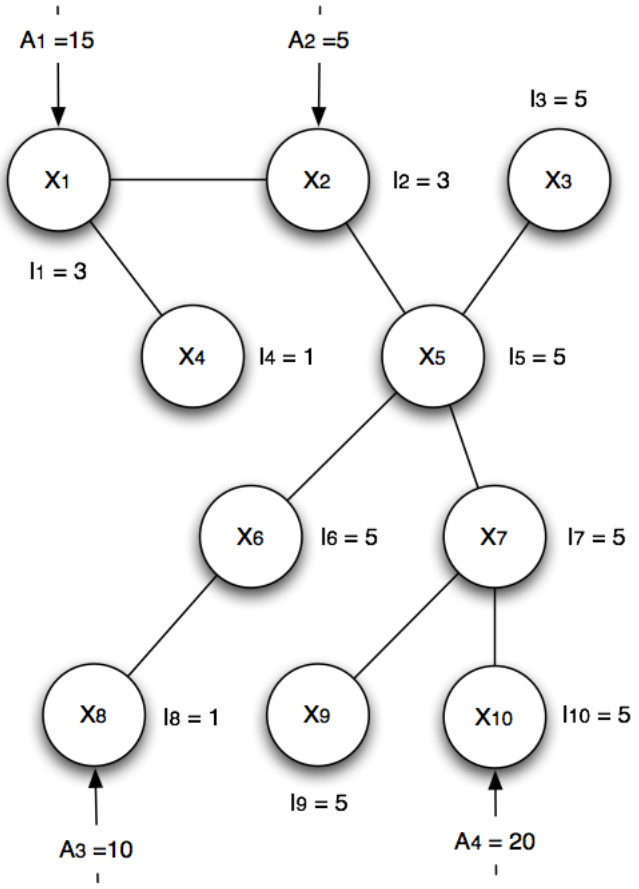
Caso de prueba 3



Grupo	Soldados asignables	Países asignables
G1	7	Todos
G2	5	X_1, X_2, X_3, X_4
G3	5	X_5, X_6, X_7, X_8

El caso de prueba 3 tiene como objetivo comprobar si es capaz de defender un cluster y descartar otro, en función del interés de sus países cuando hay insuficientes soldados para garantizar la defensa de todos.

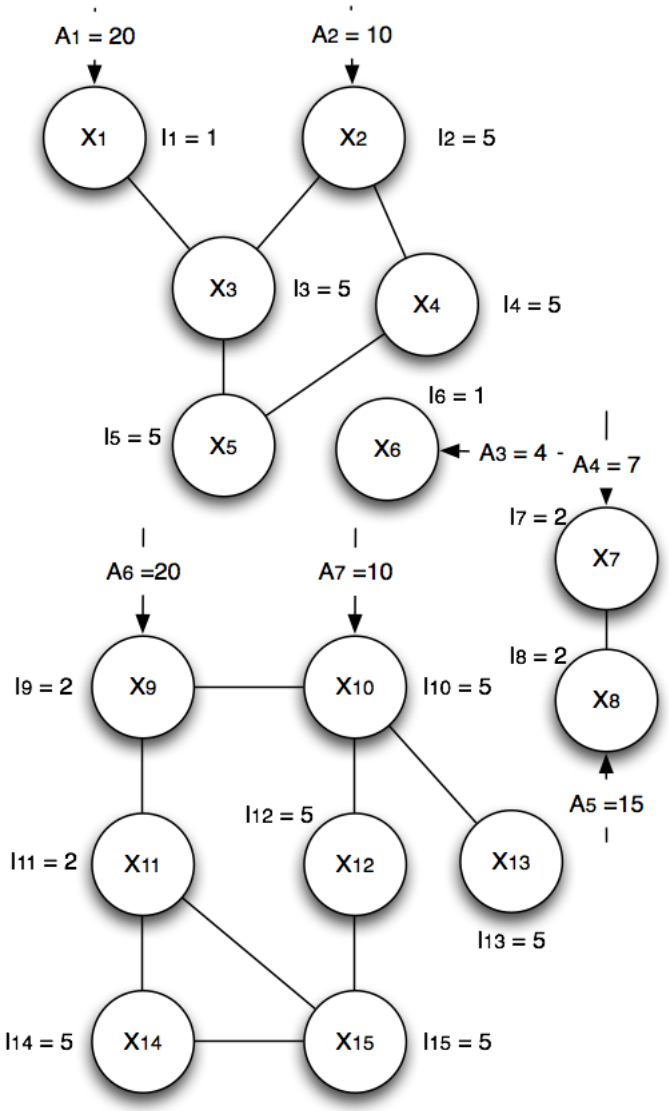
Caso de prueba 4



Grupo	Soldados asignables	Países asignables
G1	10	Todos
G2	15	X_2, X_3, X_5, X_6, X_7
G3	10	X_1, X_2, X_4, X_5
G4	7	X_5, X_6, X_8
G5	3	X_7, X_9

El caso de prueba 4 tiene como objetivo comprobar el impacto en rendimiento, efectividad y eficiencia al aumentar el tamaño del problema incrementando el número de países en un solo cluster.

Caso de prueba 5



Grupo	Soldados asignables	Países asignables
G1	25	Todos
G2	10	X_1, X_3
G3	5	X_2, X_3, X_4
G4	10	X_7, X_8
G5	15	$X_9, X_{10}, X_{11}, X_{12}, X_{13}$

El caso de prueba 5 tiene como objetivo comprobar el impacto en rendimiento en efectividad y eficiencia al aumentar el número de países y de clusters.

5.1.2. Resultados

Para cada caso de prueba se aplican los siguientes algoritmos:

1. Algoritmo heurístico de defensas (véase la sección 4.5.5, página 134)
2. Algoritmo de inicialización: sólo la parte de inicialización del algoritmo heurístico.
3. Algoritmo evolutivo (véase la sección B.4.2, página 213). Inicializando la población con el algoritmo de inicialización.
4. Algoritmo evolutivo puro: inicializando la población aleatoriamente.

Caso de prueba	heuristico	inicializacion	evolutivo	evolutivo puro
C1	**0.9125**	**0.9125**	**0.9125**	**0.9125**
C2	**0.7942**	0.0344	0.0353	**0.7942**
C3	**0.2451**	0.0439	0.0459	0.1583
C4	0.4640	0.1382	0.1349	**0.4683**
C5	0.5084	0.5084	0.4980	**0.5316**

Tabla 5.1: Resultado de los algoritmos de defensa

Caso de prueba	heuristico	inicializacion	evolutivo	evolutivo puro
C1	23	**1**	36	15
C2	**0**	**0**	16	16
C3	1	**0**	36	40
C4	4	**0**	60	120
C5	2	**1**	90	99

Tabla 5.2: Tiempo (ms) de los algoritmos de defensa

Los resultados de las pruebas del algoritmo de defensas son bastante positivos. Aunque el algoritmo de algoritmos evolutivos logra unos resultados ligeramente superiores en los casos más complejos, el algoritmo heurístico es más constante, ya que consigue buenas defensas en todos los casos de prueba. Además es muy rápido, y no parece verse muy afectado por el aumento del tamaño del problema en número de países.

La inicialización solamente funciona en casos muy básicos pero es extremadamente rápida y sirve para su función.

5.2. Evaluación del agente

Para evaluar a **Ender**, se juegan partidas contra otros agentes. La configuración de parámetros utilizada puede ser consultada en el apéndice A, página 191.

Los oponentes de Ender son los agentes incluidos por defecto en Lux Delux. Hay doce agentes, clasificados en tres niveles de dificultad: Fáciles, Intermedios y Difíciles.

Todos estos agentes intentan conquistar continentes y mejorar su posición. Sin embargo, se diferencian en las estrategias preprogramadas que guían su comportamiento. El mayor problema que presentan es que no pueden combinar estrategias fácilmente porque deben concentrar todo su esfuerzo en la estrategia escogida, por ejemplo si deciden eliminar un jugador no quitan un continente al rival.

Los oponentes de nivel fácil e intermedio son bastante inferiores, por lo que las pruebas se centrarán en los cuatro agentes difíciles: Boscoe, EvilPixie, Killbot y Quo. También se han incluido pruebas contra otros dos agentes: Bort y Yakool. Sus estrategias son:

1. **Boscoe** Cuando no es el mejor jugador de la partida se vuelve muy agresivo, dañando en todo lo posible al jugador que considera que va ganando. Muchas veces hace de árbitro, ya que si queda con dos jugadores mejores que él, al atacar a uno de ellos desequilibra la partida a favor del otro.

2. **EvilPixie** Defensivo, siempre intenta asegurar sus territorios.

3. **Killbot** El que mejor mata a sus adversarios. Si ve un enemigo débil y cree que puede matarlo dedica todo su esfuerzo a hacerlo.

4. **Quo** Basa su estrategia en minimizar sus fronteras y defenderlas.

5. **Bort** Agente de nivel difícil que fue retirado al incluir Killbot. Similar a Boscoe pero se expande más lentamente, intentando realizar un único ataque por turno. ‘

6. **Yakool** Agente de nivel mediano. Similar a Boscoe pero realiza más ataques por turno.

5.2.1. Prueba con distinto número de jugadores

Esta prueba evalúa a Ender en el tablero clásico contra distintas combinaciones de oponentes. Para ello, se utilizarán los agentes de nivel difícil del Lux: Boscoe, EvilPixie, Killbot y Quo. Se realiza la prueba para todas las combinaciones de enemigos posibles sin repetición en partidas de tres, cuatro y cinco jugadores.

El número de partidas jugadas ha sido bastante limitado debido al tiempo por partida. Para cada combinación de dos oponentes se han jugado 35 partidas, 210 en total. Para las combinaciones de tres oponentes se han jugado 30 partidas, 120 en total. Para la combinación de todos los oponentes se han jugado 65 partidas. Las tablas 5.3 y 5.4 muestran el porcentaje de victorias de cada jugador según la combinación de oponentes y un resumen, respectivamente.

El primer resultado a destacar es que Ender ha sido el ganador para todas las combinaciones de jugadores. Además, su modelo es mucho más adaptable a los cambios que las estrategias predefinidas. Su número de victorias ronda siempre el 55 % con tres jugadores y el 50 % con cuatro, independientemente de los oponentes. No se puede decir lo mismo de sus oponentes, ya que ninguno ha logrado ser regular:

- Boscoe ha sido el mejor rival de Ender en las combinaciones de tres jugadores pero al aumentar el número de rivales ha descendido notablemente su rendimiento. Esto se debe a que su estrategia agresiva puede funcionar con pocos jugadores pero en una partida con muchos jugadores el daño que puede hacer es mucho menor al repartirse entre sus rivales.

- Evilpixie depende bastante de la agresividad de sus oponentes, funcionando bien cuando no le atacan demasiado (Quo) pero no tan bien cuando intentan quitarle sus continentes (Boscoe).

- Killbot mejora su porcentaje de victorias a medida que aumenta el número de

Combinación	Ender	Boscoe	EvilPixie	Killbot	Quo
BE	**19 (54,3 %)**	9 (25,7 %)	7 (20,0 %)		
BK	**20 (57,1 %)**	11 (31,4 %)		4 (11,4 %)	
BQ	**18 (51,4 %)**	13 (37,1 %)			4 (11,4 %)
EK	**23 (65,7 %)**		8 (22,9 %)	4 (11,4 %)	
EQ	**16 (45,7 %)**		14 (40 %)		5 (14,3 %)
KQ	**20 (57,1 %)**			6 (0,17 %)	9 (25,7 %)
BEK	**14 (46,67 %)**	5 (16,67 %)	4 (13,33 %)	7 (23,33 %)	
BEQ	**17 (56,67 %)**	3 (10 %)	4 (13,33 %)		6 (20 %)
BKQ	**13 (43,33 %)**	8 (26,67 %)		8 (26,67 %)	1 (3,33 %)
EKQ	**15 (50 %)**		6 (20 %)	2 (6,67 %)	7 (23,33 %)
BEKQ	**27 (41,54 %)**	9 (13,85 %)	7 (10,78 %)	16 (24,62 %)	6 (9,23 %)

Tabla 5.3: Resultados para cada combinación de jugadores

Agente	3 jugadores	4 jugadores	5 jugadores	Media
Ender	55,24 %	49,16 %	41,54 %	48,64 %
Boscoe	31,43 %	17,78 %	13,85 %	21,02 %
EvilPixie	27,62 %	15,55 %	10,78 %	17,98 %
Killbot	13,33 %	18,88 %	24,62 %	18,94 %
Quo	17,14 %	15,55 %	9,23 %	13,97 %

Tabla 5.4: Resumen de los resultados. Porcentaje de victorias.

jugadores. Esto parece contradictorio pero se explica por su estrategia de matar jugadores: cuantos menos jugadores hay para matar peores resultados obtiene.

- Quo es el más irregular, ya que su estrategia de minimizar fronteras depende mucho de la situación inicial y de la agresividad de sus rivales.

5.2.2. Prueba en varios tableros

Este análisis trata de probar cómo varía el rendimiento del agente al variar el tablero de juego.

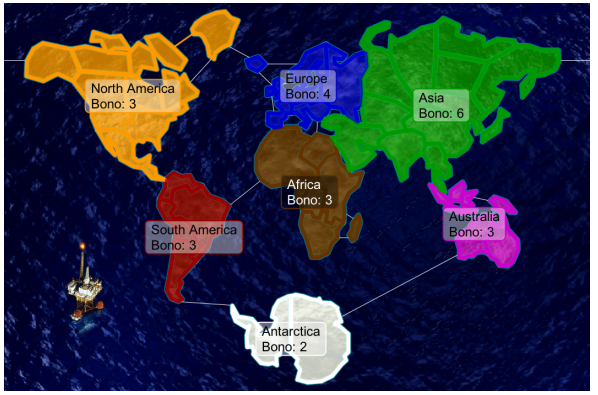
Para esto se jugarán partidas de cuatro jugadores, ya que un número menor simplificaría el juego y un número mayor incrementaría el efecto del azar en tableros pequeños. Se escoge a los oponentes que dieron mejor resultado en las pruebas de varios jugadores: Boscoe, EvilPixie y Killbot. La figura 5.1 muestra los tableros escogidos descritos en la tabla 5.5:

Tablero	Países	Conts	Descripción
Classic	42	6	El tablero clásico.
Risk Plus - Final	44	7	Pequeña variación del tablero clásico modificando el bono de los continentes y añadiendo un continente extra: la Antártida.
Roman Empire II	54	18	Tablero grande con muchos continentes de tamaño diverso, entre uno y seis países
The Dark Ages: West	28	15	Tablero pequeño con muchos continentes de un solo país.

Tabla 5.5: Tableros escogidos para realizar las pruebas

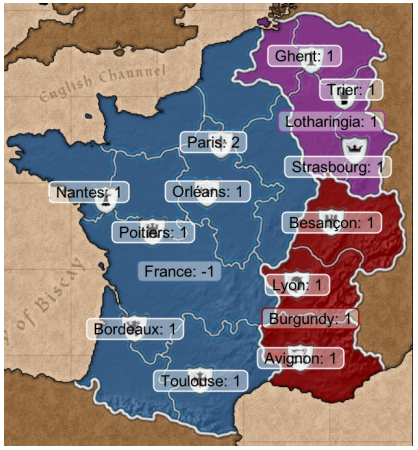
En cada tablero se jugarán 30 partidas, las mismas que se jugaron para esa combinación de oponentes en el tablero clásico.

(a) Classic



(b) Risk Plus - Final

(c) Roman Empire II



(d) The Dark Ages: West

Figura 5.1: Tableros utilizados en las pruebas

Agente	Ender	Boscoe	EvilPixie	Killbot
Classic	14 (46,67 %)	5 (16,67 %)	4 (13,33 %)	7 (23,33 %)
Risk Plus - Final	17 (56,67 %)	1 (3,33 %)	6 (20 %)	6 (20 %)
Roman Empire II	4 (13,33 %)	10 (33,33 %)	12 (40 %)	4 (13,33 %)
The Dark Ages: West	23 (76,67 %)	3 (10 %)	4 (13,33 %)	0 (0 %)

Tabla 5.6: Resultados en varios tableros.

Los principales resultados de Ender mostrados en la tabla 5.6 son:

- En **Risk Plus** se obtiene un ligero incremento de la ventaja de Ender.
- En **Roman Empire II** el porcentaje de victorias disminuye mucho. Ender es capaz de obtener ventaja al inicio pero, en posiciones de final de partida en las que necesita realizar un gran número de ataques no examina planes tan grandes.
- En **The Dark Ages: West** el resultado es excelente, al alcanzar más del 75 % de victorias.

En general, *el cambio de tablero parece que mejora los resultados*, lo que refuerza la idea de la mayor flexibilidad de Ender ante situaciones de juego distintas. Sin embargo, al aumentar el tamaño del tablero los resultados empeoran debido a la reducción del número de planes examinados(véase la sección 5.3.2, página 177). Esto podría solucionarse aumentando el tiempo de la búsqueda según el tamaño del tablero.

En cuanto a los rivales, destaca el caso de Killbot. Es el segundo en el tablero clásico y en el tablero clásico extendido pero al cambiar el tipo de tablero su número de victorias decae rápidamente.

5.2.3. Prueba comparativa con MARS

En esta sección se pretenden comparar los resultados de Ender con los de MARS, el agente basado en una arquitectura multiagente (véase la sección 2.5.3, página 21).

No se dispone del código ni de los ficheros compilados de dicho agente, por lo que es imposible enfrentarlos de forma directa. Sin embargo, dado que MARS fue evaluado con

los agentes de Lux Delux, pueden imitarse sus pruebas y comparar los resultados.

MARS fue evaluado en el tablero clásico con 6 jugadores y un valor de canjeo de cartas de 5. Entre las pruebas realizadas, se ha escogido repetir la que lo enfrenta a los mejores oponentes: Boscoe, EvilPixie, Quo, Bort y Yakool. Destaca la ausencia de Killbot, que aún no había sido desarrollado.

En la evaluación de MARS las partidas que llegan a 100 turnos se declaran empate, al considerar que los jugadores sólo acumulan soldados y no atacan. El empate se considera derrota de todos los jugadores, por lo que se muestra el porcentaje de empates.

MARS jugó 1000 partidas con esta configuración pero en este caso, debido al elevado tiempo por partida, sólo han podido jugarse 50. Por lo tanto, lo que se compara es el porcentaje de victorias de cada agente, mostrado en la tabla 5.7:

Agente	Resultados Ender	Resultados MARS
Ender/MARS	**38 %**	28,6 %
EvilPixie	**22 %**	12,5 %
Bort	14 %	**18,7 %**
Yakool	**12 %**	6,7 %
Quo	8 %	**13,2 %**
Boscoe	6 %	**19,2 %**
Empate	0 %	**1,1 %**

Tabla 5.7: Comparativa de resultados con MARS.

El resultado es muy positivo, ya que Ender ha obtenido un 38 % de victorias respecto al 28,6 % logrado por MARS.

En cuanto al resto de agentes, destaca la mejora de EvilPixie, que pasa a ser segundo. En el proyecto de MARS se comenta que EvilPixie lanzó excepciones en su fase de fortificación en 120 partidas, por lo que probablemente la mejora se deba a que ese error fue solucionado.

5.2.4. Pruebas con los mejores agentes

En Lux Delux pueden descargarse otros agentes además de los incluidos por defecto. Destacan dos: Reaper y BotOfDoom. No se ha encontrado una descripción del algoritmo que utilizan pero su comportamiento durante las pruebas realizadas se resume en lo siguiente:

1. **Reaper**: Acumula soldados hasta que puede dominar un continente completo. En ese caso, conquista el continente y lo defiende para continuar acumulando soldados. En su descripción indica que está optimizado para tableros de gran tamaño. Actualmente está en su versión 6.3.

2. **BotOfDoom**: Basa su estrategia en no ser atacado, para lo que acumula soldados en un conjunto pequeño de países. Puede pasar muchos turnos sin atacar hasta que decide conquistar gran parte del mundo. Actualmente está en su versión 3.21.

La tabla 5.8 muestra los resultados de dos pruebas, una con sólo tres jugadores (Ender, Reaper y BotOfDoom) y otra con seis (Ender, Reaper, BotOfDoom, Boscoe, EvilPixie y Killbot).

Agente	3 jugadores	6 jugadores
Ender	**44 %**	2 %
Reaper	32 %	18 %
BotOfDoom	24 %	**66 %**
Boscoe		6 %
EvilPixie		4 %
Killbot		4 %

Tabla 5.8: Resultados contra los mejores agentes

Mientras la primera prueba es muy positiva para Ender, la segunda ha sido bastante mala. Esto se debe a que los dos nuevos agentes son extremadamente defensivos, sin realizar apenas ataques, especialmente BotOfDoom. Esto favorece a Ender en el caso de tres jugadores ya que puede extenderse más y obtener ventaja. Sin embargo, al añadir

a los otros tres agentes, estos concentran todos sus ataques entre ellos mismos y Ender. El resultado es que los agentes defensivos son capaces de obtener mucha ventaja gracias a que el resto de agentes no se dan cuenta de que deben atacarles para debilitarlos independientemente de si conquistan el país o no.

También hay que señalar que, aunque Reaper no ha obtenido buenos resultados, probablemente en otros tableros más grandes obtendría mejores resultados.

La conclusión es que Ender, Reaper y BotOfDoom están igualados. Dependiendo de las condiciones de la partida (tablero, número de jugadores o valor de canjeo de cartas), ganará uno u otro.

5.3. Análisis de tiempos

Esta sección analiza el tiempo que tarda el agente en realizar un turno y en qué cálculos lo invierte. Además, se compara el tiempo al jugar en distintos tableros para ver qué partes del algoritmo son más sensibles a su tamaño.

5.3.1. Tiempo de un turno

La gráfica 5.2 muestra la distribución de probabilidad del tiempo que el agente tarda en realizar un turno.

La mayor parte de los turnos duran en torno a uno o dos minutos, debido a la condición de parada de la búsqueda de planes a los 60 segundos y a las replanificaciones ocurridas. Gracias a dicha condición, no hay demasiada diferencia en el tiempo de ambos tableros, aunque los picos del tablero clásico son más pronunciados, debido a dos motivos:

1. En Roman Empire II hay ocasiones en las que se la búsqueda se alarga hasta examinar el mínimo de 30 planes, alargando un poco más el tiempo de turno. Por este motivo, en otros tableros de mayor tamaño podría dispararse el tiempo por turno.

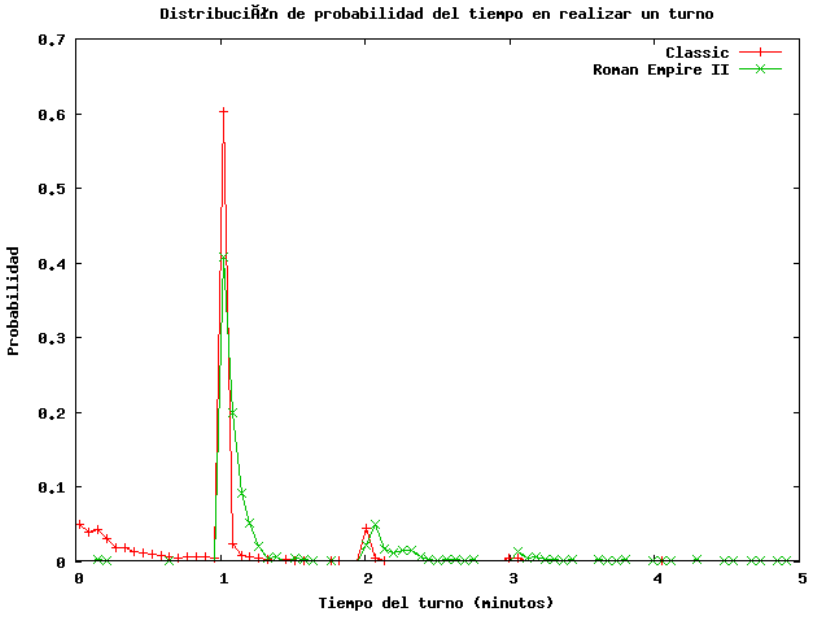


Figura 5.2: Tiempo en realizar un turno

2. Aunque el tiempo de búsqueda del plan está acotado a 60 segundos independientemente del tablero, el refinamiento y ejecución del plan también dependen del tamaño del tablero y no están acotados.

Otra clara diferencia es que en el tablero clásico hay bastantes casos en los que el turno tarda menos de 20 segundos. Al examinar un mayor número de planes, es capaz de analizar todos los planes viables en los casos en los que tiene pocos soldados para planificar sus ataques, lo que lo hace muy exacto en posiciones ajustadas.

5.3.2. Planes examinados

El número de planes examinados es clave en el rendimiento del agente. Los casos en los que hay muy pocos planes que examinar pueden considerarse ruido. La gráfica 5.3 muestra la distribución de probabilidad de examinar cada número de planes, sólo contando con los casos en que la búsqueda terminó por haber agotado su tiempo.

En general, la distribución del número de planes parece asimilarse a distribuciones normales o afines, parametrizadas en función del tablero.

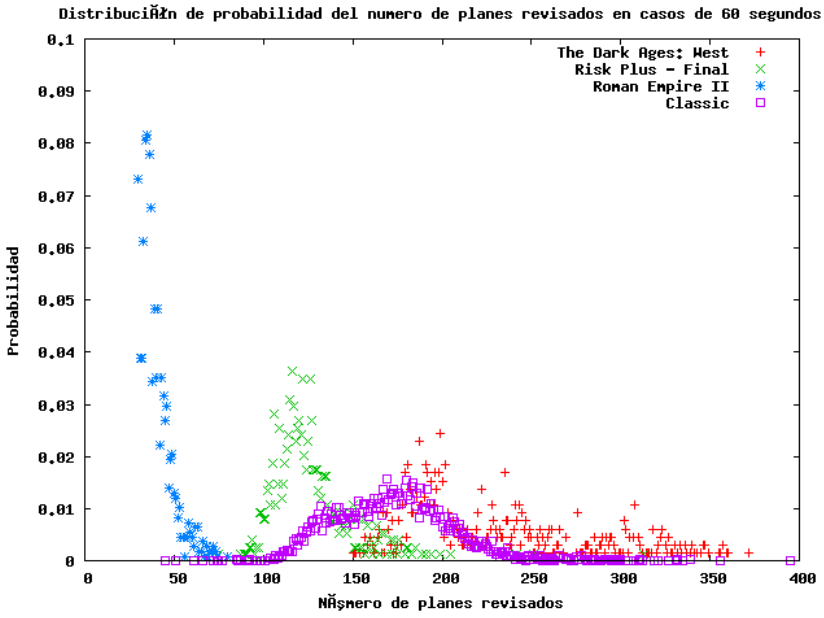


Figura 5.3: Número de planes revisados en búsquedas de 60 segundos o más

En el caso del tablero Roman Empire II dicha normal está cortada debido al criterio de examinar un mínimo de 30 planes. Prácticamente nunca pasa de los 50 planes, lo que explica el bajo rendimiento obtenido en ese tablero.

En The Dark Ages: West se obtuvieron los mejores resultados y, casualmente es el tablero para el que se examinan más planes. Puede verse una clara correlación entre el número de planes examinados y el número de victorias.

5.3.3. Desglose del tiempo de valoración del plan

La tabla 5.9 muestra el tiempo que se invierte de media en analizar un plan en cada tablero.

Puede observarse como el algoritmo de interés es el que dispara el tiempo del análisis en el tablero Roman Empire. El aumento del número de continentes es lo que más afecta al rendimiento del agente, ya que se puede apreciar claramente la diferencia al añadir un sólo continente entre el tablero clásico y el Risk Plus.

Algoritmo	Classic	Risk Plus - Final	Roman Empire II	Dark Ages: West
Dominio	78,278	95,69	157,27	34,53
Interés	144,159	244,64	1029,68	180,07
Amenazas	46,656	52,42	69,01	15,82
Defensa	26,728	22,47	67,13	11,93
Fuerza	0,089	0,09	0,12	0,08
Otros	0,305	0,29	0,25	0,23
Total	296,214	415,59	1323,47	242,65

Tabla 5.9: Desglose del tiempo de valoración del plan

(a) Plan vacío: Classic

(b) Plan vacío: Risk Plus - Final

(c) Plan vacío: Roman Empire II

(d) Plan vacío: Dark Ages: West

(e) Plan no vacío: Classic

(f) Plan no vacío: Risk Plus - Final

(g) Plan no vacío: Roman Empire II

(h) Plan no vacío: Dark Ages: West

Figura 5.4: Gráficas de desglose del tiempo del análisis del plan

Hay que comprobar cómo afecta la diferencia en las iteraciones realizadas por los algoritmos para un plan vacío y uno no vacío. En la figura 5.4 se muestran las gráficas de desglose de tiempos de la valoración del plan para cada uno de los tableros en ambos casos.

No hay grandes diferencias en cuanto a la proporción del tiempo consumido por cada algoritmo. En cambio, el total del tiempo consumido se duplica para el plan vacío debido a que los algoritmos que más tiempo consumen han duplicado sus iteraciones.

Además, se puede comprobar que el porcentaje de consumo de tiempo del interés depende directamente del número de continentes. El tiempo de todos los algoritmos depende de forma similar del número de países, lo que explica que tableros con tiempos muy diferentes como "Roman Empire II" y "Dark Ages: West" tengan proporciones similares.

Por lo tanto, los algoritmos que más urge optimizar son el algoritmo de interés y el de dominio. Dado que el interés utiliza el algoritmo de dominio para obtener el interés de un continente, habría que centrar los esfuerzos en optimizar el dominio a medio plazo.

5.4. Comportamientos inteligentes

Durante las pruebas realizadas, se observó el juego del agente y se analizaron sus principales virtudes y defectos.

Virtudes

- **Multiestrategia** En un mismo turno lleva a cabo ataques para lograr diferentes objetivos: conquistar continentes, quitar continentes, eliminar jugadores, etc.
- **No llevar tropas del enemigo al ataque** Los agentes rivales suelen defender en Indonesia y Ender conquista prácticamente todo el mundo y acumula tropas antes de conquistar Siam para así no dejar que esos soldados entren en juego hasta que es demasiado tarde.
- **Defensa inteligente** Es capaz de *proteger sus soldados*. Es decir, cuando prevé que los enemigos van a conquistarle un país por muchos soldados que ponga, retira las

defensas. Esto, aunque no queda vistoso, le permite lograr que sobrevivan soldados para reconquistar esos países en su siguiente turno.

- **Quitarse de en medio** Aunque no es algo fácil, gracias a sus predicciones, el agente intenta siempre reducir al máximo las amenazas de sus enemigos.
- **Quitar continentes que otros jugadores no pueden** Si debe elegir entre quitarle al mejor jugador de la partida Europa o América del Norte y otro jugador puede quitarle Europa, Ender irá a quitar América del Norte aunque le cueste un poco más.

Defectos

- **No matar a los enemigos** Muchas veces podría eliminar un oponente y no lo hace. Para solucionarlo bastaría con ajustar mejor los parámetros aumentando el valor de las cartas en el cálculo de la fuerza de los jugadores.
- **No es suficientemente reactivo ante los cambios** Debería replanificar más pero es imposible debido al tiempo de replanificación.
- **Errores en la predicción a corto plazo** A veces no predice correctamente las amenazas de los rivales. En estos casos defiende mal o incluso valora mal la utilidad de algunos ataques.

Capítulo 6

Conclusiones

Al finalizar este proyecto, las conclusiones son muy positivas. Aunque Ender no ha resuelto el juego, es una primera versión bastante buena.

Su juego es bastante prometedor, teniendo comportamientos muy inteligentes en ocasiones, aunque aún deben limarse algunos aspectos. Su posición entre los agentes anteriores es:

- **Calidad**: Es comparable a los mejores agentes encontrados. La mayor pega es que pierde en tableros grandes debido a que no ha utilizado suficiente tiempo por turno.
- **Tiempo**: Consume mucho más tiempo en cada turno que las versiones anteriores. Aunque podría mejorarse este aspecto, al ser un sistema basado en búsqueda siempre tardará más que los agentes basados en reglas.
- **Capacidad de mejora**: Todos los agentes anteriores tenían algunas reglas cuya modificación era compleja puesto que pequeños cambios modificarían todo el comportamiento del agente. Ender no ha tenido una gran optimización de parámetros y podría ajustar mucho su comportamiento. Las heurísticas de Ender le pueden permitir añadir nuevos factores y ponderarlos con los ya considerados, mejorando el sistema sin excesivos cambios. Además todo el sistema esta subdividido en algoritmos independientes, que pueden sustituirse fácilmente.

Las *claves del éxito* de Ender son:

1. Búsqueda de planes: Sus acciones están planificadas y coordinadas.
2. Boceto de plan: Centra la búsqueda en los ataques decidiendo con heurísticas la distribución de las tropas.
3. Predicción a medio plazo: Permite comprender correctamente quién controla cada zona del tablero.
4. Predicción a corto plazo: Permite intentar que lo ataquen lo menos posible, quitándose de zonas de conflicto y defendiendo bien sus territorios.

Por otro lado, ha tenido ciertos *problemas* con:

1. Tamaño del tablero: Afecta demasiado al tiempo de valoración del plan.
2. Algoritmo de búsqueda: No siempre examina los mejores planes.
3. Ejecución del plan: No logra ser suficientemente reactiva a los cambios.

Durante el proyecto se encontraron muchas *dificultades*, entre las que destacan:

1. Dependencia entre países: Todos los algoritmos desarrollados tienen que luchar con la dependencia entre países de algún modo y en la mayoría de los casos se tuvo que asumir cierta independencia para que los algoritmos fuesen viables.
2. Aproximación de funciones: En principio parecía fácil aproximar las funciones que calculan el factor de riesgo y la probabilidad de los ataques pero se complicó bastante. Aun así, sin poder realizar esos cálculos en un tiempo mínimo, el agente no hubiera funcionado correctamente, por lo que fue un trabajo bien invertido.
3. Número de posibilidades del jugador exponencial.
4. Tamaño del tablero: Todos los cálculos del agente se realizan para cada país y para cada continente, por lo que el aumento del tamaño del tablero reduce mucho el rendimiento del agente.

Capítulo 7

Lineas futuras

Una vez completado el proyecto, es una buena idea dar algunas directrices de cómo podría continuarse su línea de trabajo en otros proyectos futuros.

7.1. Mejora de los algoritmos

Para continuar el trabajo, principalmente deberían pulirse los algoritmos desarrollados en este proyecto solucionando sus mayores carencias.

Algoritmo de búsqueda (sección 4.4.4, página 97) El algoritmo de búsqueda debería replantearse bastante, ya que no logra encontrar el plan de mayor valoración en los casos en los que el número de soldados posicionables es alto. Probablemente sea la parte del sistema más importante de mejorar.

Boceto de plan (sección 4.4.3, página 92) Para aumentar la capacidad de reacción del agente ante los sucesos del tablero, el boceto de plan debería incluir mucha más información. Para cada ataque del boceto podría calcularse su importancia e incluir también aquellos ataques que se realizarán sólo en caso de que haya suerte en el turno. Utilizando estos datos podría lograrse una replanificación muy rápida en caso de que los cambios respecto a lo previsto no sean demasiado significativos. También permitiría

evaluar mejor la valoración de fracaso del plan, basándose en la probabilidad de éxito de cada ataque y su importancia. También debería poder incluir ataques para debilitar aunque se usen en pocas ocasiones.

Factores de accesibilidad en el algoritmo de interés (sección 4.5.3, página 115) La accesibilidad mide la distancia que hay desde los países dominados hasta un conjunto de países. Podría contarse en el interés de un país, el beneficio por aumentar la accesibilidad que ya tiene el jugador. Es interesante la accesibilidad de un continente si otro jugador puede tenerlo para poder quitarle un país y de enemigos débiles para poder eliminarlos.

Dominio a medio plazo más rápido (sección 4.5.2, página 106) El dominio a medio plazo es el algoritmo que más tiempo consume en la valoración del plan. Además, depende mucho del número de países del tablero por los siguientes motivos:

- El dominio se calcula para todos los países.
- Hay un grupo de fuerza por cada país.
- Se calcula la distancia entre todos los países.

Por lo tanto, el algoritmo a medio plazo podría mejorarse considerando que países lejanos entre sí tienen distancia infinita, por lo que cada grupo de soldados sólo afecta a los países más cercanos.

De este modo, al aumentar el número de países no aumentaría tanto el tiempo de valoración del plan y podría aplicarse el algoritmo en tableros muy grandes.

Interés de continente más rápido (sección 4.5.3, página 117) Aunque el interés que el jugador tiene en un continente se calcula con gran exactitud, debería optimizarse el agente para que no empeore su rendimiento cualitativamente al aumentar el número de continentes del tablero. Hay varias formas de mejorarlo:

- En la configuración de parámetros reducir el número de iteraciones del dominio potencial. Esto empeorará ligeramente el resultado pero aumentará su velocidad.
- No realizar el dominio potencial para continentes imposibles. Incluir alguna regla que determine si un continente es imposible.

  Por ejemplo, para el continente C si $Gasig_C$ es el conjunto de grupos de fuerza del jugador que cumplen $\exists p \in C F_g > Calc_{g,p}$ la ecuación 7.1 determina si es imposible dominar C.

  7.1 .

$$\sum_{g \in Gasig_C} F_g < PC \sum_{e \in E} \sum_{p \in C} F_{e,p} \tag{7.1}$$

donde E es el conjunto de enemigos del jugador.

Algoritmo de amenazas (sección 4.5.4, página 119) Las amenazas que extiende no son del todo correctas. Quizá debería guiar la extensión de amenazas por el interés en vez de la probabilidad de éxito. Además, debería extender más amenazas en el análisis del plan y podarlas en su valoración.

Algoritmo de defensas (sección 4.5.5, página 127) Debería unir la búsqueda de la mejor defensa y el camino de tropas para poder ponderar la importancia del aumento de la probabilidad de éxito de los ataques al decidir la distribución final de los soldados. Además, al propagar una amenaza sería interesante contemplar el interés del jugador atacante en cada dirección.

Orden de los ataques (sección 4.5.9, página 151) El orden de los ataques debería ir guiado por su importancia además de por su probabilidad de éxito.

7.2. Optimización de los parámetros de configuración

Los parámetros de configuración determinan completamente el comportamiento del agente. Sin embargo, optimizarlos no es una tarea nada sencilla y no ha podido realizarse en este proyecto.

Optimización por partes Debido al alto número de parámetros y al alto tiempo que consume una partida, deberían optimizarse los algoritmos por separado con casos de prueba. Con los casos de prueba definidos para el algoritmo de búsqueda de defensas podría optimizarse la parte de búsqueda de la mejor defensa y podrían plantearse casos similares (algo más complejos) para otros algoritmos.

Optimización en función de las reglas El agente podría tener una configuración diferente en función de las reglas o el tablero. Por ejemplo, para tableros más grandes menor número de iteraciones del algoritmo de dominio.

7.3. Incluir semántica en los cálculos

Uno de los mayores problemas encontrados es que hay que asumir independencia entre países para poder realizar los cálculos de interés, dominio, etc. Podría añadirse conocimiento semántico en estos cálculos que considere esas dependencias.

Por ejemplo, en el interés de un país además de incluir el valor del interés podrían incluirse los motivos de ese interés (dominio zona, continente, etc.). Se añade el conocimiento de que el interés de continente aumenta cuantos más países se tengan y, al calcular el interés de una amenaza valora positivamente coger varios países interesantes por el mismo continente.

7.4. Aprendizaje sobre los rivales

Los jugadores humanos son capaces de aprender cómo juegan sus rivales para predecir mejor sus movimientos. Lograr esto en una inteligencia artificial parece casi utópico pero, dado el modelo del valoración de la posición planteado en este proyecto, puede plantearse un esquema que logre este tipo de comportamientos.

Para hacerlo únicamente habría que modificar los algoritmos para que tomaran sus parámetros de forma dependiente al jugador al que se enfrentan. Así, un jugador se caracterizaría por los parámetros que determinan:

- Peso que dá a los factores en el interés de un país.
- Asignación de la fuerza de sus grupos en el dominio.
- Peso que dá a los factores de la utilidad de la amenaza.
- ...

A partir de los movimientos realizados por el jugador, se determinan los parámetros que los hubiesen predicho. Tras varias partidas contra un mismo jugador, si se logran estimar correctamente los parámetros, el agente sería capaz de alejarse de los jugadores más agresivos, por ejemplo.

Apéndice A

Parámetros de configuración

A continuación se expone el fichero de configuración de parámetros utilizado durante la realización de las pruebas de evaluación del agente.

```
Ender.tiempoPlanificacion = 60000

#Busqueda
Ender.algBusqueda.maxEstadosDesordenados=5
Ender.algBusqueda.numMinEstadosRevisados = 30

#Factor de riesgo
Ender.algPlan.algFRArbol.algFRLista.porcentajeRiesgo=0.8
Ender.algPlan.algFRArbol.algFRListaSecundario.porcentajeRiesgo=0.7

#Prioridad
Ender.algPlan.algPrioridadPlan.pesoInteresDominio=5.0
Ender.algPlan.algPrioridadPlan.pesoInteresConquista=15.0
Ender.algPlan.algPrioridadPlan.pesoMatarEnemigo=1000.0
Ender.algPlan.algPrioridadPlan.pesoInteresReduccionFronteras = 750.0
Ender.algPlan.algPrioridadPlan.pesoInteresConquistaContinente = 500.0
Ender.algPlan.algPrioridadPlan.pesoValoracionPadre=1.5
Ender.algPlan.algPrioridadPlan.pesoEstructuraAtaque=1.0
Ender.algPlan.algPrioridadPlan.pesoInteresPais = 2.0
```

```
#Analisis del plan
Ender.algPlan.algAnalisisPlan.algValoracionJugador.exponente = 2
Ender.algPlan.algAnalisisPlan.calcularAmenazas=true
Ender.algPlan.algAnalisisPlan.iteraciones=1
Ender.algPlan.algAnalisisPlan.iteracionesInicializacion=2
Ender.algPlan.algAnalisisPlan.iteracionesInteresDominio=2
Ender.algPlan.algAnalisisPlan.valoracionDerrotaPositiva=0.25

#Fuerza de los jugadores
Ender.algPlan.algAnalisisPlan.algFuerzaJugador.valorFANumPaises = 1
Ender.algPlan.algAnalisisPlan.algFuerzaJugador.valorFANumSoldados = 1.5
Ender.algPlan.algAnalisisPlan.algFuerzaJugador.valorFANumCartas = 15
Ender.algPlan.algAnalisisPlan.algFuerzaJugador.valorFFITenerContinente =
    1000
Ender.algPlan.algAnalisisPlan.algFuerzaJugador.valorFFINumPaises = 2
Ender.algPlan.algAnalisisPlan.algFuerzaJugador.valorFFINumSoldados = 3
Ender.algPlan.algAnalisisPlan.algFuerzaJugador.valorFFLTenerContinente =
    200
Ender.algPlan.algAnalisisPlan.algFuerzaJugador.valorFFLDominarPais = 5

#Interes
Ender.algPlan.algAnalisisPlan.algInteresPais.algMediaDatosInteres.peso2
    =2.0
Ender.algPlan.algAnalisisPlan.algInteresPais.algMediaDatosInteres.peso1
    =500.0
Ender.algPlan.algAnalisisPlan.algInteresPais.algMediaDatosInteres.peso0
    =1.0
Ender.algPlan.algAnalisisPlan.algInteresPais.algDegradacionDominioZona.
    maxDistanciaDominioZona=2
Ender.algPlan.algAnalisisPlan.algInteresPais.algDegradacionDominioZona.
    multiplicadorPesoDistanciaDominioZona=0.3
Ender.algPlan.algAnalisisPlan.algInteresPais.algDegradacionInteres.
    porcentajeGraduacion=0.05
Ender.algPlan.algAnalisisPlan.algInteresPais.algDegradacionInteres.
    numGraduaciones=0
Ender.algPlan.algAnalisisPlan.algInteresPais.algDegradacionInteres.
    reduccionPorcentajeGraduacion=0.01
Ender.algPlan.algAnalisisPlan.algInteresPais.algInteresContinente.
    interesMinimo = 0.1
```

```
#Dominio
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.numIteraciones=2
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.numIteracionesFuerza=5
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.
    numIteracionesDominioPotencial= 2
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.costeDominarPaisM=2.0
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.costeDominarPaisA=0.8
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.costeAlcanzarPaisPropio
    = 0.5
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.
    numeroTurnosRefuerzosEsperados=0.5
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.
    multiplicadorCosteAlcanzarPais=2
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.fuerzaBaseEnPais=0.1
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.exponenteInteres = 2
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.exponenteCosteDominio =
    1
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.exponenteCosteAlcanzar
    = 3
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.fuerzaMinimaParaReparto
    = 1
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.fuerzaSoldadoPais = 1.5

#Defensa Analisis
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.
    numNivelesBusquedaHillClimbing=1
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais.numTransferencias=1
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algInicializacion.algoritmoValoracionDefensas.
    beneficioDefenderContinente = 10000
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algValoracion.beneficioDefenderContinente = 10000
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.algValoracion.
    beneficioDefenderContinente = 10000
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.
    algInicializacion.algoritmoValoracionDefensas.
    beneficioDefenderContinente = 10000
```

```
#Amenazas Analisis
Ender.algPlan.algAnalisisPlan.algAmenazas.maxNumAmenazasPais=30
Ender.algPlan.algAnalisisPlan.algAmenazas.minProbExito=0.9
Ender.algPlan.algAnalisisPlan.algAmenazas.porcentajeUtilidadMaximaMinimo =
    0.2
Ender.algPlan.algAnalisisPlan.algAmenazas.probabilidadEsperadaMinima =
    0.01
Ender.algPlan.algAnalisisPlan.algAmenazas.pesimismoRefuerzos=0.3
Ender.algPlan.algAnalisisPlan.algAmenazas.expUtilidad=2.0
Ender.algPlan.algAnalisisPlan.algAmenazas.
    probabilidadPoderRealizarlaElevada=true
Ender.algPlan.algAnalisisPlan.algAmenazas.importanciaUtilidadConquista = 2
Ender.algPlan.algAnalisisPlan.algAmenazas.importanciaUtilidadDominio = 0.5
Ender.algPlan.algAnalisisPlan.algAmenazas.algAmenazasSitDef.
    algUnionFuerzaAmenazas.decrecimientoPeso=0.5
Ender.algPlan.algAnalisisPlan.algAmenazas.algAmenazasSitDef.
    expDivisionAmenazaPaises = 0.5
Ender.algPlan.algAnalisisPlan.algAmenazasSitDef.algUnionFuerzaAmenazas.
    decrecimientoPeso=0.5
Ender.algPlan.algAnalisisPlan.algAmenazasSitDef.expDivisionAmenazaPaises =
    0.5
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais.numTransferencias=1
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    numNivelesBusquedaHillClimbing=0
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algValoracion.beneficioDefenderContinente = 10000
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algInicializacion.algoritmoValoracionDefensas.
    beneficioDefenderContinente = 10000

#Interes corto plazo
Ender.algPlan.algAnalisisPlan.algAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos.peso0 = 1.0
Ender.algPlan.algAnalisisPlan.algAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos.peso1 = 1.0
Ender.algPlan.algAnalisisPlan.algAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos.peso2 = 100.0
```

```
Ender.algPlan.algAnalisisPlan.algAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos.peso3 = 0.05
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos.peso0 = 1.0
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos.peso1 = 1.0
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos.peso2 = 100.0
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos.peso3 = 0.05

#Defensa Refinamiento
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    numNivelesBusquedaHillClimbing=1
Ender.algPlan.algRefinamientoPlan.algoritmoCaminosTropas.
    proporcionBeneficioAtaque=10.0
Ender.algPlan.algRefinamientoPlan.algoritmoCaminosTropas.
    proporcionBeneficioRedistribucion=1.0
Ender.algPlan.algRefinamientoPlan.algoritmoCaminosTropas.
    numIteracionesMaximas=2
Ender.algPlan.algRefinamientoPlan.algAmenazasSitDef.algUnionFuerzaAmenazas
    .decrecimientoPeso=0.5
Ender.algPlan.algRefinamientoPlan.algAmenazasSitDef.
    expDivisionAmenazaPaises = 0.5

#Amenazas Refinamiento
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algTransferencias.algTransferenciaGrupoPais.
    numTransferencias=3
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.minProbExito=0.8
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.
    probabilidadPoderRealizarlaElevada=true
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.numNivelesBusquedaHillClimbing=0
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algAmenazasSitDef.
    algUnionFuerzaAmenazas.decrecimientoPeso=0.5
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algAmenazasSitDef.
    expDivisionAmenazaPaises = 0.5
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.expUtilidad=2.0
```

```
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.maxNumAmenazasPais
    =1000
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.pesimismoRefuerzos=0.3
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.
    importanciaUtilidadConquista = 2
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.
    importanciaUtilidadDominio = 0.5
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.
    porcentajeUtilidadMaximaMinimo = 0.2
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.
    probabilidadEsperadaMinima = 0.01
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algInicializacion.algoritmoValoracionDefensas.
    beneficioDefenderContinente = 10000
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algValoracion.beneficioDefenderContinente = 10000

#Ejecucion: Replanificacion
Ender.algPlan.algReplanificacion.porcentajeLimiteCosteAtaque=0.6
Ender.algPlan.algReplanificacion.constanteDiferenciaFR=3.0
Ender.algPlan.algReplanificacion.constanteLimiteCosteAtaque=7.0
Ender.algPlan.algReplanificacion.porcentajeDiferenciaFR=0.5
Ender.algPlan.algReplanificacion.constanteLimiteCosteRama=5.0
Ender.algPlan.algReplanificacion.porcentajeLimiteCosteRama=0.5
Ender.algPlan.algReplanificacion.porcentajeLimiteCosteAtaqueVictoria= 0.5
Ender.algPlan.algReplanificacion.constanteLimiteCosteAtaqueVictoria=3



#Seleccion de algoritmos
Ender=AgenteRiskBusqueda
Ender.algBusqueda=MejorPrimero

Ender.algPlan=BocetoPlan
Ender.algPlan.algAnalisisPlan=AnalisisPlan
Ender.algPlan.algCrearBocetoPlan=BocetoPlanVacio
Ender.algPlan.algGenerarHijos=GeneradorHijosUnAtaque
Ender.algPlan.algOrdenAtaques=OrdenAtaquesProbFracaso
Ender.algPlan.algReplanificacion=ReplanificacionCambiosResultados
Ender.algPlan.algRepartoCoste=RepartoProporcionalCosteRestante
```

```
Ender.algPlan.algFRArbol=FactorRiesgoArbol
Ender.algPlan.algFRArbol.algFRLista=FactorRiesgoLista
Ender.algPlan.algFRArbol.algFRListaSecundario=FactorRiesgoLista
Ender.algPlan.algFRArbol.algFRLista.algFRPais=FactorRiesgoPaisInterpolado
Ender.algPlan.algFRArbol.algFRListaSecundario.algFRPais=
    FactorRiesgoPaisInterpolado
Ender.algPlan.algCosteArbol=CosteAtaqueArbol
Ender.algPlan.algCosteArbol.algCosteAtaquePais=CosteFormula
Ender.algPlan.algProbExito=ProbExitoArbol
Ender.algPlan.algProbExito.algProbExitoAtaqueSimple =
    InterpolacionProbExitoSimple

Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo=DominioGruposEnergia
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.algDominioContinentes=
    PorcentajeDominioContinente
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.proporcionDominioPais=
    ProporcionNormalizada
Ender.algPlan.algAnalisisPlan.algDominioMedioPlazo.algRefuerzosEsperados=
    RefuerzosEsperadosMedios


Ender.algPlan.algAnalisisPlan.algInteresPais=InteresPorContinente
Ender.algPlan.algAnalisisPlan.algInteresPais.algInteresContinente=
    InteresDominioPotencial
Ender.algPlan.algAnalisisPlan.algInteresPais.algDegradacionDominioZona=
    DegradacionPorDistancia
Ender.algPlan.algAnalisisPlan.algInteresPais.algMediaDatosInteres=
    MediaPonderada
Ender.algPlan.algAnalisisPlan.algInteresPais.algDegradacionInteres=
    DegradacionNVeces

Ender.algPlan.algAnalisisPlan.algAmenazas = ExtenderAmenazasPorCoste
Ender.algPlan.algAnalisisPlan.algAmenazas.algInteresCortoPlazo=
    InteresDanyoEnemigos
Ender.algPlan.algAnalisisPlan.algAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos=MediaPonderada
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa=
    MejoraDefensaEnforcedHillClimbing
```

```
Ender.algPlan.algAnalisisPlan.algAmenazas.algCoste=CosteFormula
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa=Defensa
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algValoracion=ValoracionDefensasPropagacionAmenazas
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algValoracion.algoritmoDefensaPais=DefensaPaisSigmoidal
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algInicializacion.algoritmoValoracionDefensas=
    ValoracionDefensasPropagacionAmenazas
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algInicializacion.algoritmoValoracionDefensas.algoritmoDefensaPais=
    DefensaPaisSigmoidal
Ender.algPlan.algAnalisisPlan.algAmenazas.proporcionProbRefuerzosPais=
    ProporcionNormalizada
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais=TransferenciasTropasRango
Ender.algPlan.algAnalisisPlan.algAmenazas.algRefuerzosEsperados=
    RefuerzosEsperadosMedios
Ender.algPlan.algAnalisisPlan.algAmenazas.algProbExitoConquista=
    InterpolacionProbExitoLista
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algTransferencias=TransferenciasTropasMaximas
Ender.algPlan.algAnalisisPlan.algAmenazas.algDefensa.algMejoraDefensa.
    algInicializacion=InicializacionDefensasCubirAmenazas
Ender.algPlan.algAnalisisPlan.algAmenazas.algAmenazasSitDef=
    UnionAmenazasRecibidasPais
Ender.algPlan.algAnalisisPlan.algAmenazas.algAmenazasSitDef.
    algUnionFuerzaAmenazas=UnionDatosReducida
Ender.algPlan.algAnalisisPlan.algAmenazas.algAmenazasSitDef.
    algCosteAtaquePropagacionAmenazas=CosteFormula
Ender.algPlan.algAnalisisPlan.algAmenazas.algProbExitoConquista.
    algProbExitoAtaqueSimple = InterpolacionProbExitoSimple
Ender.algPlan.algAnalisisPlan.algAmenazas.algProbExitoConquista.algCoste =
     CosteFormula

Ender.algPlan.algAnalisisPlan.algDefensa=Defensa
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.algValoracion.
    algoritmoDefensaPais=DefensaPaisSigmoidal
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.algValoracion=
```

```
    ValoracionDefensasPropagacionAmenazas
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa=
    MejoraDefensaEnforcedHillClimbing
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.
    algInicializacion.algoritmoValoracionDefensas=
    ValoracionDefensasPropagacionAmenazas
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.
    algInicializacion=InicializacionDefensasCubirAmenazas
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.
    algTransferencias=TransferenciasTropasMaximas
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais=TransferenciasTropasRango
Ender.algPlan.algAnalisisPlan.algDefensa.algMejoraDefensa.
    algInicializacion.algoritmoValoracionDefensas.algoritmoDefensaPais=
    DefensaPaisSigmoidal


Ender.algPlan.algAnalisisPlan.algAmenazasSitDef=UnionAmenazasRecibidasPais
Ender.algPlan.algAnalisisPlan.algAmenazasSitDef.algUnionFuerzaAmenazas=
    UnionDatosReducida
Ender.algPlan.algAnalisisPlan.algAmenazasSitDef.
    algCosteAtaquePropagacionAmenazas=CosteFormula

Ender.algPlan.algAnalisisPlan.algFuerzaJugador=
    FuerzaJugadorValoracionEstatica

Ender.algPlan.algAnalisisPlan.algValoracionJugador=ProporcionAjustada


Ender.algPlan.algRefinamientoPlan=RefinamientoDefensasPlan
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas=
    ExtenderAmenazasPorCoste
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa=Defensa
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algValoracion.algoritmoDefensaPais=
    DefensaPaisSigmoidal
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algInicializacion.algoritmoValoracionDefensas=
    ValoracionDefensasPropagacionAmenazas
```

```
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algInteresCortoPlazo=
    InteresDanyoEnemigos
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algInteresCortoPlazo.
    algMediaDatosInteresDanyoEnemigos=MediaPonderada
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algInicializacion.algoritmoValoracionDefensas.
    algoritmoDefensaPais=DefensaPaisSigmoidal
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa=MejoraDefensaEnforcedHillClimbing
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.
    proporcionProbRefuerzosPais=ProporcionNormalizada
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algInicializacion=InicializacionDefensasCubirAmenazas
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algAmenazasSitDef=
    UnionAmenazasRecibidasPais
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algProbExitoConquista.
    algProbExitoAtaqueSimple = InterpolacionProbExitoSimple
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algProbExitoConquista.
    algCoste = CosteFormula
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algRefuerzosEsperados=
    RefuerzosEsperadosMedios
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algAmenazasSitDef.
    algUnionFuerzaAmenazas=UnionDatosReducida
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algAmenazasSitDef.
    algCosteAtaquePropagacionAmenazas=CosteFormula
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algTransferencias=TransferenciasTropasMaximas
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algValoracion=ValoracionDefensasPropagacionAmenazas
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algCoste=CosteFormula
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algProbExitoConquista=
    InterpolacionProbExitoLista

Ender.algPlan.algRefinamientoPlan.algoritmoDefensa=Defensa
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algInicializacion.algoritmoValoracionDefensas.algoritmoDefensaPais=
    DefensaPaisSigmoidal
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algValoracion.algoritmoDefensaPais=DefensaPaisSigmoidal
```

```
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa=
    MejoraDefensaEnforcedHillClimbing
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algTransferencias=TransferenciasTropasMaximas
Ender.algPlan.algRefinamientoPlan.algoritmoAmenazas.algDefensa.
    algMejoraDefensa.algTransferencias.algTransferenciaGrupoPais=
    TransferenciasTropasRango
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais=
    TransferenciasTropasCombinadas
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais.transferencia1=
    TransferenciasTropasRango
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais.transferencia1.
    numTransferencias=5
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais.transferencia2=
    TransferenciasTropasSelectas
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais.transferencia2.
    posiblesTransferenciasTropas=M
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais.transferencia2.
    algValoracionDefensaPais = DefensaPaisSigmoidal
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algTransferencias.algTransferenciaGrupoPais.transferencia2.
    reduccionCuentaAmenazasSecundarias = 0.75
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algInicializacion=InicializacionDefensasCubirAmenazas
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algValoracion=ValoracionDefensasPropagacionAmenazas
Ender.algPlan.algRefinamientoPlan.algoritmoDefensa.algMejoraDefensa.
    algInicializacion.algoritmoValoracionDefensas=
    ValoracionDefensasPropagacionAmenazas

Ender.algPlan.algRefinamientoPlan.algAmenazasSitDef=
    UnionAmenazasRecibidasPais
Ender.algPlan.algRefinamientoPlan.algAmenazasSitDef.algUnionFuerzaAmenazas
```

```
    =UnionDatosReducida
Ender.algPlan.algRefinamientoPlan.algAmenazasSitDef.
    algCosteAtaquePropagacionAmenazas=CosteFormula

Ender.algPlan.algRefinamientoPlan.algoritmoCaminosTropas=
    CaminosTropasProporcionales
Ender.algPlan.algRefinamientoPlan.algoritmoCaminosTropas.
    algProbExitoAtaques=ProbExitoArbol
Ender.algPlan.algRefinamientoPlan.algoritmoCaminosTropas.
    algProbExitoAtaques.algProbExitoAtaqueSimple =
    InterpolacionProbExitoSimple



Ender.algPlan.algPrioridadPlan.algProbConquistarPaises =
    ProbConquistarPaises
Ender.algPlan.algPrioridadPlan.algProbConquistarPaises.costeAtaque =
    CosteFormula
Ender.algPlan.algPrioridadPlan=PrioridadAtracciones
```

Apéndice B

Ideas descartadas

El objetivo de esta sección es presentar todas las ideas que han sido desechadas por considerarse inviables o no adecuadas. Con esto se pretende dejar constancia de todas las ideas valoradas y evitar que en futuros proyectos se repitan los mismos análisis.

B.1. Equilibrios de Nash

La teoría de juegos es una rama de la matemática que analiza los juegos o modelos matemáticos de conflicto y cooperación entre agentes inteligentes que buscan maximizar su beneficio.

Hay varias representaciones posibles para los juegos [Mye97]:

- Forma estratégica: se definen un conjunto de jugadores, cada uno con varias posibles estrategías a escoger. Para cada combinación de estrategias, se define la valoración que obtiene cada jugador.
- Forma extensiva: La forma extensiva representa un juego como un árbol de decisiones. El juego del Risk puede formalizarse con ella.
- Forma bayesiana.

El equilibrio de Nash se define para juegos en representación estratégica, como una situación en el juego (combinación de estratégias de todos los jugadores) en la que ningún jugador puede aumentar sus beneficios cambiando su estratégia unilateralmente. Esta demostrado que para todo juego en forma estratégica existe al menos un equilibrio de Nash si se consideran estratégias estocásticas —la estrategia de cada jugador se define como la probabilidad que tiene dicho jugador de escoger cada una de sus posibles estrategias "puras".

En todos los juegos se tiende a alcanzar un equilibrio de Nash, por lo que, un modo de guiar al agente sería obtener el equilibrio al que la partida tenderá.

Aunque el juego de Risk no puede representarse en forma estratégica, podría tomarse alguna simplificación de éste y utilizar el equilibrio como heurística.

Por ejemplo, un juego en el que la decisión que debe tomar cada jugador es el conjunto de contienentes que va a intentar conquistar. De este modo, los equilibrios de Nash serían una heurística en un nivel estratégico de continentes que determine el interés de cada jugador por un continente.

Sin embargo, al intentar aplicar esta idea surgen varias complicaciones:

1. Es muy costoso, ya que los equilibrios no son sencillos de calcular y dependen del estado de la partida, por lo que hay que recalcularlos en cada turno o incluso en cada plan analizado.

2. Puede haber más de un equilibrio de Nash, por lo que hay que escoger uno o hacer una media.

3. Al calcular el equilibrio se considera que todos los jugadores toman su decisión de forma simultánea e independiente. Esta puede ser una simplificación asumible al ser una decisión estratégica y no táctica, pero hay que tener en cuenta que los jugadores pueden cambiar su decisión al inicio de su turno, por lo que se pierde exactitud.

B.2. Búsqueda de n-jugadores

La búsqueda entre adversarios ha sido una técnica frecuentemente utilizada en inteligencia artificial, obteniendo resultados notables en juegos como las damas, el ajedrez y juegos con azar como el backgammon o el bridge [RN03].

El algoritmo más común es el min-max (con poda alfa-beta para optimizar recorriendo una menor parte del árbol de búsqueda), apoyado en una función de evaluación específica para cada juego.

B.2.1. Busqueda n-jugadores a nivel táctico

En el nivel táctico, las acciones de los jugadores son los movimientos que realizan en el tablero de juego. La búsqueda daría como resultado una lista de acciones que deben realizarse.

A la hora de aplicar la búsqueda de varios jugadores al dominio del Risk hay que considerar los siguientes aspectos:

1. *N-jugadores*: Como en el Risk pueden jugar de 2 a 6 jugadores, habrá que tener en cuenta las decisiones de cada uno de ellos. Esto hace que, para tener información de un turno de profundidad sea necesario recorrer seis niveles en el árbol de búsqueda. Además, en juegos con n jugadores no puede aplicarse al completo la poda alfa beta [Kor91], por lo que hay que recorrer bastante parte del árbol.
2. *Incertidumbre*: Cuando un jugador realiza un ataque, el resultado es no determinista. Por lo tanto, o se calculan todas las posibilidades aumentando mucho el coste computacional o se busca el caso más probable (o una media), perdiendo exactitud en los cálculos [Bal83].

Sin embargo, el mayor problema a la hora de aplicar la búsqueda n-jugadores al dominio del Risk es el enorme número de posibilidades que tiene cada jugador.

El número de posicionamientos, ataques y fortificaciones que un jugador puede realizar crece exponencialmente respecto al número de países y soldados posicionables. En el

tablero clásico del Risk, con 42 países y posicionamientos que suelen ir entre 3 y 10 soldados el número de opciones que tiene un jugador en su turno es innumerable y todo esto sin contar con que cada ataque tiene un resultado indeterminado.

Por lo tanto, habría que reducir el factor de ramificación radicalmente. La opción propuesta aquí es añadir un generador de planes que genere un número asequible de planes que se consideren buenos para el jugador a partir de un estado.

Además, como los estados del árbol son parecidos, el generador de planes podrá reutilizar información obtenida en anteriores generaciones de planes para otros nodos del árbol (por ejemplo, si se generan planes con computación evolutiva puede inicializarse la población de planes con los previamente generados), como muestra la figura B.1.

Por lo tanto, para seguir esta aproximación, hay que definir

1. Generador de planes: Genera planes a partir de un estado para un jugador. Lo ideal sería que los planes, además de ser viables y dar una buena posición al jugador, fuesen distintos entre si

2. Valoración de la posición: Valora un estado dando una puntuación a cada jugador.

3. Ejecución del plan: Dado un estado y un plan se genera el estado (o posibles estados) resultante.

4. Propagación de la valoración de un jugador por el árbol: Como el resultado de un plan es no determinista, min-max podría no ser la mejor opción.

Se descarta la aplicación de esta arquitectura en el proyecto porque se considera que tras la ejecución de seis planes (para ver la situación en el turno siguiente), la pérdida de información por el efecto estocástico es demasiado alta y se prefiere hacer una valoración heurística de la posición más costosa que tenga en cuenta los movimientos de los adversarios, esto es, la dinámica de la posición.

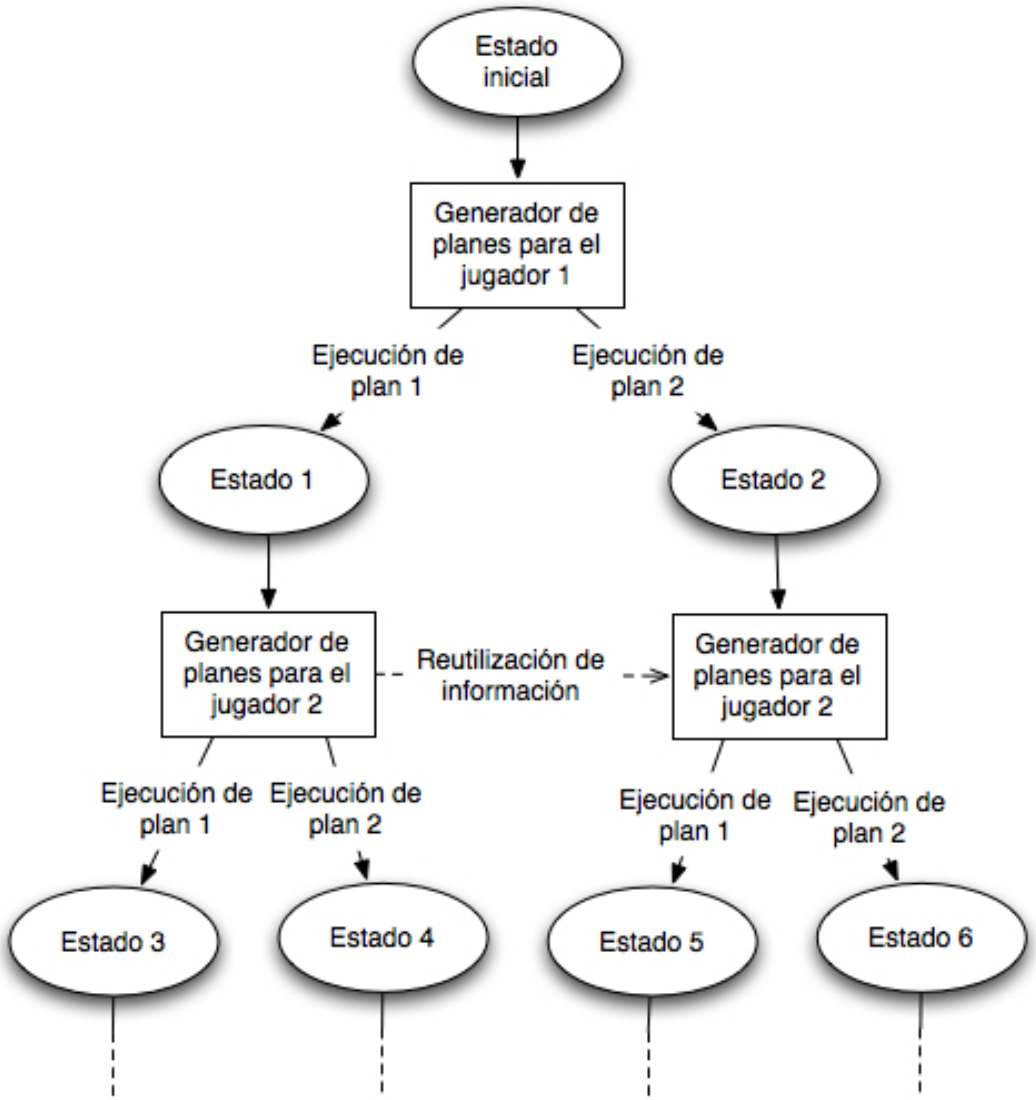


Figura B.1: Búsqueda N-jugadores con generador de planes

B.2.2. Busqueda n-jugadores a nivel estratégico

En el nivel estratégico se toma una situación simplificada del juego para obtener heurísticas que guíen al jugador a escoger las acciones que desea realizar.

Una opción interesante sería considerar la simplificación de que cada jugador selecciona los continentes que desea dominar. El resultado de la búsqueda sería el conjunto de continentes que el jugador debe conquistar, lo que podría utilizarse para configurar el interés de los continentes.

Sin embargo, la valoración de una situación en este juego simplificado no es tan simple. Resulta complejo valorar lo buena que es la situación para cada jugador dados los continentes que desea conquistar.

B.3. Representación del plan maestro

El plan maestro *define todas las acciones que hará el jugador durante su turno.* Como el resultado de los ataques es indeterminado, se deberá planificar qué hacer para cada uno de los posibles resultados.

En la figura B.2 puede verse un esquema de un posible plan maestro. La fase de fortificaciones se ha simplificado considerándola como una única decisión, cuando en realidad es una cadena de traspasos de tropas individuales.

La parte central del esquema es el resultado del ataque. Se puede ver que, por cada ataque, hay $(n + 1 - x)$ posibles resultados del ataque: derrota, sobreviven $(x + 1)$ soldados, ... , sobreviven n soldados. Para cada uno de estos ataques se pasa de nuevo a la fase de ataque (tras una decisión de traslado de soldados) lo que conlleva planificar muchos planes diferentes dependiendo de los resultados de los ataques.

Esta aproximación tiene dos limitaciones muy importantes:

1. Muchas ramas del plan no se ejecutan. De hecho si se tienen m ataques de profundidad en cada ramificación del plan de ataques, cada uno con n resultados posibles,

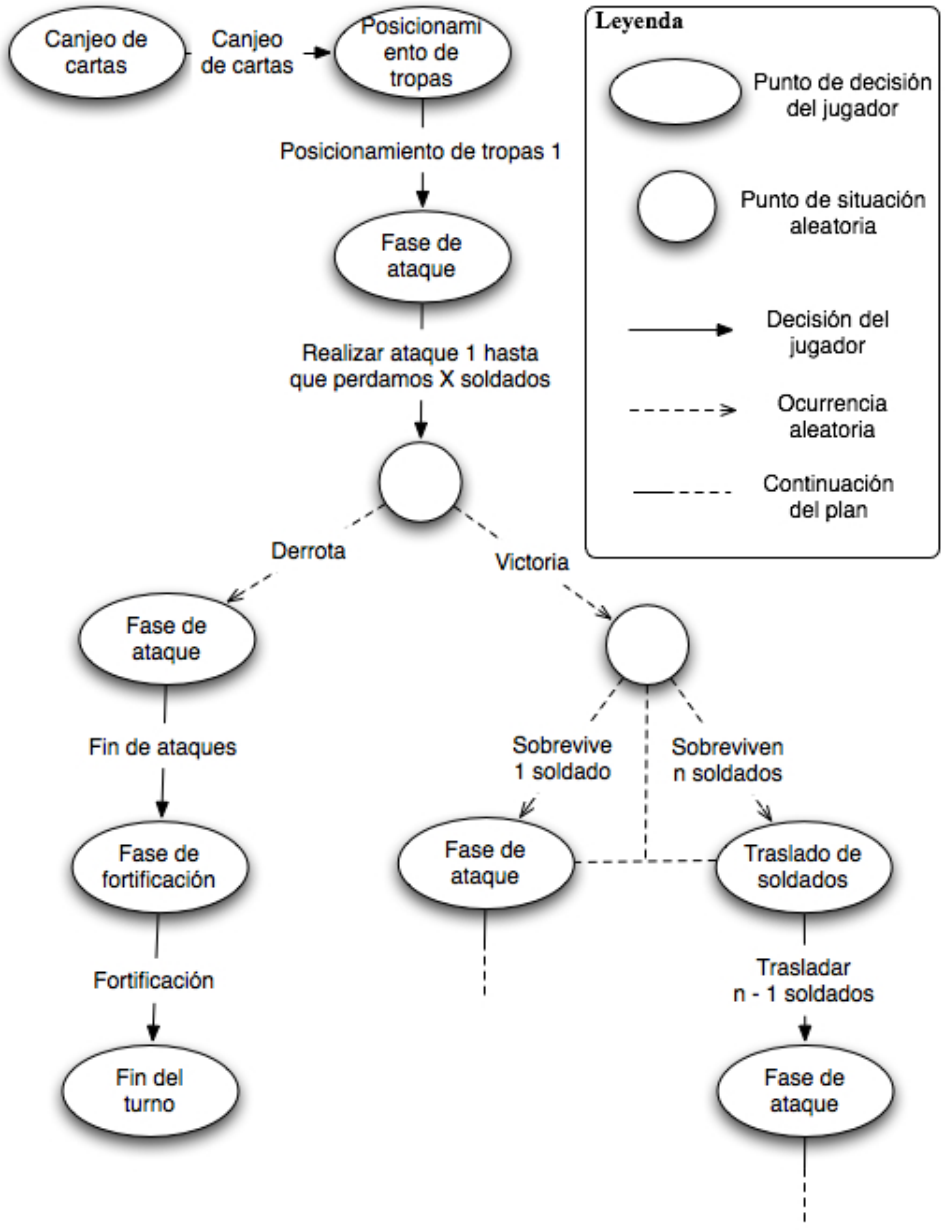


Figura B.2: Diagrama de un plan maestro

habría n^m ejecuciones posibles de ese plan. Dado que el plan se ejecutará una sola vez, $(n^m - 1)$ ejecuciones se habrán calculado sin ninguna utilidad.

2. No se aprovecha la relación entre las diferentes decisiones del jugador. Esto hace la búsqueda más pesada, sobre todo por el posicionamiento inicial de tropas que se realiza independientemente de los ataques.

Pueden mejorarse algunas decisiones de diseño para evitar estas limitaciones o, al menos, reducir su impacto. Por ejemplo, la figura B.3 muestra un plan maestro con dos modificaciones:

1. El número de soldados supervivientes al conquistar un país no modifica el plan y sólo afecta a la decisión de traslado de soldados.

2. Las ramas en las que el ataque no tiene éxito no se planifican al inicio del turno y, en caso de ocurrir, provocan una replanificación.

En consecuencia, el plan maestro puede ser adecuado para un entorno de algoritmos evolutivos, en el que todas las decisiones se toman mediante el proceso de búsqueda y se recorre el espacio mediante pequeñas variaciones en los mejores planes encontrados. Sin embargo en un proceso de búsqueda heurística clásica, la utilización del plan maestro es inviable al hacer el espacio de búsqueda inmanejable. El principal problema es que algunas decisiones (la asignación de soldados por ejemplo) tienen un factor de ramificación muy grande, aunque no sean tan relevantes ya que cambiar un soldado de un país a otro no afecta demasiado al resultado final.

B.4. Soluciones alternativas al problema de defensas

El problema de defensas se resolvió con un proceso de búsqueda heurística, pero también se discutieron otras ideas.

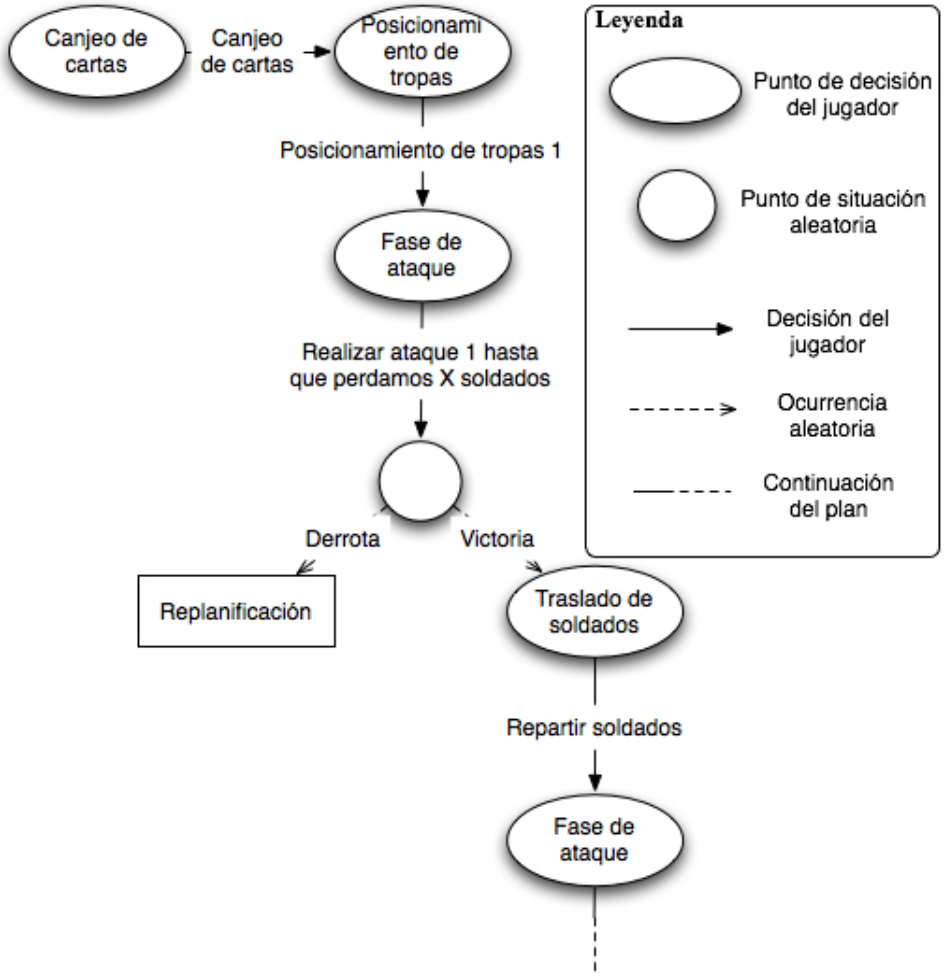


Figura B.3: Diagrama de un plan maestro con replanificación

B.4.1. Solución mediante técnicas de investigación operativa

El problema de defensa se formalizó matemáticamente como un problema de optimización (véase la sección 4.5.5, página 133).

Para realizar la optimización, la investigación operativa [Tah98] [HL67] ofrece varias técnicas como la programación lineal, la programación no lineal y la programación entera, entre otras.

Solución mediante programación lineal

La técnica básica para afrontar problemas de investigación operativa es la programación lineal. Esta técnica requiere que el problema cumpla con las siguientes propiedades:

Restricciones lineales Las restricciones aplicadas sobre las variables de decisión son lineales.

Variables de decisión continuas No son continuas puesto que no puede asignarse medio soldado a un país y medio a otro. Sin embargo, puede solucionarse utilizando programación entera, o simplificando el problema permitiendo asignar medio soldado a un país y redondeando el resultado.

Función objetivo lineal La valoración de las defensas de los territorios es no lineal por la interrelación entre los países. No es posible simplificarla a una función lineal ya que sería equivalente a considerar que la defensa de los países es independiente (poner soldados en un país frontera no defendería los países interiores) y el resultado del algoritmo no tendría ninguna validez.

Por lo tanto la programación lineal no puede aplicarse.

Solución mediante programación no lineal

Las técnicas de programación no lineal permiten abordar problemas en los que la función objetivo es no lineal.

Sin embargo, todas las técnicas de programación no lineal encontradas requieren que la función objetivo sea convexa y derivable, lo cuál, no parece cumplirse en este caso. La valoración de las defensas se realiza de un modo demasiado complejo como para poder representarla de esa manera, por lo que debe descartarse también la posibilidad de la programación no lineal.

Conclusiones acerca de la solución mediante investigación operativa

No parece fácil aplicar técnicas de investigación operativa a la resolución del problema de las defensas. A pesar de que la formalización matemática del problema se ajusta bastante a la de un problema de investigación operativa, la función objetivo es demasiado compleja.

B.4.2. Solución mediante algoritmos evolutivos

Otro posible método para resolver el problema de las defensas es mediante un proceso de algoritmos evolutivos. Este tipo de algoritmos han sido utilizados con buenos resultados en muchos casos de optimización de funciones complejas o desconocidas.

Aunque no interesa esta aproximación, ya que el objetivo del proyecto es estudiar heurísticas que permitan obtener los cálculos de modo más eficiente, resulta interesante intentar obtener buenos resultados con esta aproximación para poder comprobar hasta que punto son buenas las soluciones dadas por el algoritmo de búsqueda de la mejor defensa.

Definición de los individuos La representación de los individuos debe cumplir que un pequeño cambio en la representación implique un pequeño cambio en su evaluación.

Los individuos son una matriz que indica para cada grupo y cada país, un número entre 0 y 1 que pondera el número de soldados del grupo que se asignan a ese país. Por ejemplo, si hay dos grupos de soldados con 10 y 20 soldados respectivamente, a repartir entre 3 paises, el individuo:

$$I = \begin{pmatrix} 0{,}5 & 0 \\ 0{,}25 & 0{,}5 \\ 0{,}5 & 1 \end{pmatrix}$$

se transforma en la matriz de asignaciones:

$$M = \begin{pmatrix} 4 & 0 \\ 2 & 7 \\ 4 & 13 \end{pmatrix}$$

Configuración del algoritmo Se utilizan poblaciones de 20 individuos y la evaluación con torneos de 3 individuos. La condición de parada es 100 generaciones sin encontrar un individuo mejor.

B.5. Otros descartes

En esta sección se enumeran todas aquellas ideas que se plantearon y descartaron sin tener una suficiente relevancia y desarrollo como para profundizar en ellas:

Aproximación no lineal del factor de riesgo de una lista Conseguir aproximar el factor de riesgo de una lista con un conjunto de funciones lineales.

Dominio a medio plazo con propagación de energías Un algoritmo para el dominio a medio plazo que considera las fuerzas del jugador moviendose por el tablero. Fue descartado porque al mover las fuerzas por el tablero se perdía la conexión con la posición real.

Defensa de país lineal La función sigmoidal propuesta en la defensa de un país fue precedida de una función compuesta por tres funciones lineales mostrada en la ecuación B.1:

$$DefPais(S_d, A) = \begin{cases} max(0, 0{,}2 - 0{,}25(0{,}7 - \frac{S_d}{A}) & \text{si } S_d < 0{,}7A, \\ \frac{S_d}{A} - 0{,}5 & \text{si } 0{,}7A <= S_d <= 1{,}3A, \\ min(1, 0{,}8 + 0{,}25(\frac{S_d}{A} - 1{,}3)) & \text{si } 1{,}3A < S_d \end{cases} \quad \text{(B.1)}$$

Sin embargo, fue descartada porque como muestra la gráfica B.4 la función sigmoidal ofrece resultados similares pero con variaciones más suaves que probablemente den unos resultados más lógicos en la defensa de los países.

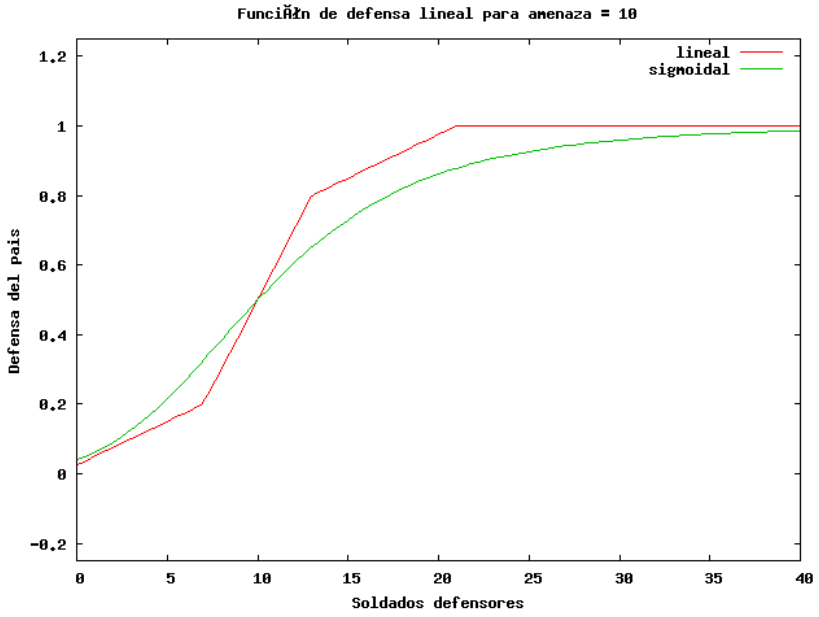


Figura B.4: Comparación de las funciones de defensa de un país

Degradación del interés Del mismo modo que se realiza la degradación del dominio para calcular el interés de dominio de la zona, podría degradarse el interés para obtener un interés que contempla conquistar un país para llegar a otro interesante. Fue descartado porque no tiene sentido para el algoritmo de dominio, pero podría utilizarse en la prioridad del plan.

Bibliografía

[Bal83] Bruce W. Ballard. The *-minimax search procedure for trees containing chance nodes. *Artif. Intell.*, 21(3):327–350, 1983.

[Gou88] Ronald Gould. *Graph Theory.* Benjamin/Cummings, 1988.

[HL67] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research.* Mc Graw Hill, 8 edition, 1967.

[Kor91] Richard E. Korf. Multi-player alpha-beta pruning. 48:99–111, 1991.

[Mye97] Roger B. Myerson. *Game Theory: Analysis of Conflict.* Harvard University Press, September 1997.

[Ols05] Fredrik Olsson. Multi-agent system for playing the board game risk. Master's thesis, Blekinge Institute of Technology, Marzo, Marzo 2005.

[RN03] S. J. Russell and Norvig. *Artificial Intelligence: A Modern Approach (Second Edition).* Prentice Hall, 2003.

[Tah98] Hamdy A. Taha. *Operations Research: An Introduction.* Pearson Prentice Hall, 6 edition, 1998.

[VG04] James M. Vaccaro and Clark C. Guest. Evolutionary bayesian network dynamic planner for game risk. In *EvoWorkshops*, pages 549–560, 2004.

[VG05] James M. Vaccaro and Clark C. Guest. Planning an endgame move set for the game risk: a comparison of search algorithms. *IEEE Trans. Evolutionary Computation*, 9(6):641–652, 2005.