UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

FACULTAD DE INGENIERÍA

DOCTORADO EN CIENCIAS DE LA INGENIERÍA

DESARROLLO DE UNA POLÍTICA DE SELECCIÓN PARA GENERAL GAME PLAYING BASADA EN EL PROBLEMA DEL BANDIDO MULTI-ARMADO

T E S I S

QUE PARA OBTENER EL GRADO DE:

DOCTOR EN CIENCIAS DE LA INGENIERÍA

P R E S E N T A:

M.C.I. IVÁN FRANCISCO VALENCIA

DIRECTORES DE TESIS:

DR. JOSÉ RAYMUNDO MARCIAL ROMERO

DRA. ROSA MARÍA VALDOVINOS ROSAS

Toluca, México, 2021

Índice general

Índice de figuras

Índice de algoritmos

Índice de tablas

Glosario

ABMC Árbol de Búsqueda Monte Carlo. 4–6, 9, 12–16, 18, 29–33, 36–38, 41–48, 50–52, 59, 60, 72–74, 76–92, 94, 96, 100–102

AEPS Aproximación Estocástica de Perturbación Simultánea. 32

BS Balanceo de Simulación. 32

ETC Explore-Then-Commit. 25, 43–61, 72–82, 84, 86–93, 96, 97, 100, 102, 103

EVAR Estimación del Valor de Acción Rápida. 30, 31

EVARG Estimación del Valor de Acción Rápida Generalizado. 31, 33

GGP General Game Playing. 3–7, 9–12, 30, 32, 33, 36–39, 42, 43, 63, 69, 70, 99–103

IA Inteligencia Artificial. 3, 9

MEC Método de Entropía Cruzada. 32

MOSS Minimax Optimal Strategy in the Stochastic Case. 5, 24, 25, 41

PBMA Problema del Bandido Multi-Armado. 4–6, 9, 16–19, 22, 24, 32, 36–38, 99

PBMAC Problema de Bandido Multi-Armado Combinatorio. 32, 33

PR Propagacion Resistente. 32

RS Recocido Simulado. 25, 26, 99

TMMP Técnica de Muestreo de Movimientos-Promedio. 31, 33

TMP Todos los Movimientos como el Primero. 30, 31

UCB Upper Confidence Bound. 5, 6, 18, 20–25, 29–33, 41–51, 54–64, 66, 69, 70, 75–82, 84–89, 99–103, 115, 118

UCBα_1 Upper Confidence Bound α_1. 70–83, 85, 86, 88–93, 95–97, 99–103

UCBα_2 Upper Confidence Bound α_2. 70–74, 76, 78–83, 85–93, 95–97, 99–103

UCB-Minimal Upper Confidence Bounds Minimal. 5, 23, 41, 42

UCB-Tuned Upper Confidence Bounds Tuned. 5, 6, 22, 41–52, 54–63, 73–84, 86–89, 92, 93, 96, 97, 99–103

UCB-V Upper Confidence Bounds with Variance. 5, 41, 42

UCT Upper Confidence Bounds Applied to Trees. 5, 18

UJP Urgente Jugar Primero. 29

Resumen

Durante la historia de la Inteligencia Artificial se han desarrollo diversos agentes inteligentes capaces de jugar juegos de tablero de entre los más destacados se tiene a DeepBlue para el ajedrez y AlphaGo para el juego de Go. Sin embargo, estos agentes solo están enfocados en un solo juego por lo cual el paso natural es desarrollar agentes capaces de jugar más de un juego, idea que persigue el área General Game Playing la cual se enfoca en desarrollar agentes completamente autónomos que puedan jugar cualquier juego de tablero sin intervención humana y sin conocimiento previo. La mayoría de los agentes están basados en el método de Árbol de Búsqueda Monte Carlo que usa simulaciones Monte Carlo para estimar movimientos prometedores, este método consiste en cuatro pasos: Selección, Expansión, Simulación y Propagación Hacia Atrás. Los esfuerzos para incrementar el rendimiento del Árbol de Búsqueda Monte Carlo se enfocan en el paso de simulación, sin embargo, es también el paso de Selección el que repercute en dicho rendimiento. El paso de selección controla la forma en que se recorre el árbol asociado al juego de tablero, dicho recorrido es guiado por una Política de Selección de la cual Upper Confidence Bound es la popularmente usada. En esta tesis de investigación se presentan dos políticas de selección UCBα_1 y UCBα_2, las cuales están basadas en Upper Confidence Bound pero están pensadas completamente para su aplicación en General Game Playing. UCBα_1 y UCBα_2 a diferencia de Upper Confidence Bound aprovechan la estructura del árbol asociados a los juegos de tablero para determinar para un nodo padre, cuánto se debe explorar los nodos hijos que representan los movimientos disponibles antes de decidirse por explotar un nodo hijo con un movimiento prometedor. Upper Confidence Bound controla la exploración de nodos hijos por medio de una constante de exploración, sin embargo, esta constante permanece fija en todo el árbol sin importar el número de nodos hijos. UCBα_1 y UCBα_2 usan una función que depende del

número de nodos hijos para determinar cuánto se debe explorar por lo cual es variable en toda la estructura del árbol. Para determinar el rendimiento de UCBα_1 y UCBα_2 se ha realizado un comparativo de estas contra Upper Confidence Bound, su variante Upper Confidence Bound Tuned y la política que sirve de control Explore Then Commit en 8 juegos de tablero con movimientos por turnos y para dos jugadores los cuales emulan un escenario completamente General Game Playing. Los resultados muestran que las políticas UCBα_1 y UCBα_2 basadas en la idea de variar la etapa de exploración de acuerdo con el número de nodos hijos logran una mayor cantidad de victorias en General Game Playing al contrario de la política comúnmente usada UCB cuya etapa de exploración depende de un valor fijo.

Introducción

Desde sus inicios la Inteligencia Artificial (IA) ha tenido como uno de sus retos el desarrollo de agentes inteligentes capaces de jugar juegos de tablero al mismo nivel (o superior) que el del ser humano [7]. El ánimo de desarrollar agentes jugadores se debe a que éstos presentan características propias de la inteligencia humana como deducción, razonamiento, resolución de problemas, búsqueda inteligente, representación del conocimiento, planificación, aprendizaje, creatividad, percepción y procesamiento de lenguaje natural, entre otros [51].

La IA ha producido agentes capaces de jugar a nivel de campeones humanos en juegos específicos como *Chinook* [43] para Damas (juego para el cual ya existe una estrategia que permite ganar sin importar las jugadas que realice el contrincante [42]), *Deep Blue* [9] para el ajedrez y *Dark Knight* [30] para *Banqi* o ajedrez medio chino. Estos agentes especializados han dado buenos resultados en el dominio para el cual fueron diseñados, sin embargo, por lo regular un agente que sea bastante bueno en un juego como el ajedrez podría ser malo jugando damas o en cualquier otro juego, esto es porque las reglas de un juego son diferentes a otro; lo que permite cuestionar qué tan inteligentes son estos agentes si solo funcionan en un único dominio [51].

Derivado de lo antes expuesto, recientemente, el desarrollo de Agentes Inteligentes se ha enfocado en desarrollar agentes capaces de jugar sin disponer para ello de estrategias aprendidas o a priori, producto de este interés surge el área de General Game Playing (GGP). Comúnmente en agentes diseñados para jugar un sólo juego, los desarrolladores son quienes analizan y determinan las estrategias que permitan tener un buen nivel de juego. En GGP los agentes deben ser autónomos, es decir deben desarrollar sus propias estrategias de juego sin intervención humana, sin haber jugado

el juego con anterioridad y partiendo únicamente de las reglas que le son suministradas momentos antes de jugar [7, 25, 26, 51].

Aunque los agentes en GGP no pueden compararse en rendimiento y eficiencia a los agentes especializados, su utilidad es mayor ya que éstos pueden desempeñarse en diferentes dominios, e incluso las técnicas desarrolladas en GGP pueden ser usadas en otras áreas como gestión de procesos de negocios, comercio electrónico, operaciones militares, entre otros [25, 26].

Un avance importante en el área de GGP ha sido la utilización del método Árbol de Búsqueda Monte Carlo (ABMC), idea propuesta por primera vez por Finsson y Björnsson [20] y usada en el agente *CadiaPlayer*, el cual ganó tres veces la competencia internacional de GGP, definiendo el estado del arte del desarrollo de agentes en GGP [25]. Por lo general el ABMC se aplica a juegos por turnos para dos jugadores, sin embargo, existen versiones para otro tipo de juegos [8].

El ABMC es un método que se guía por simulaciones Monte Carlo [8], usando los resultados de exploraciones previas para estimar con mejor precisión los valores de los movimientos más prometedores [8, 16]. Para ello el ABMC hace uso de un árbol donde cada arista representa un movimiento posible y cada nodo representa un estado del juego. A cada nodo se le asocia una serie de estadísticas como el número de victorias y el número de visitas al nodo. En cada iteración realiza cuatro pasos [16]: selección, expansión, simulación y propagación hacia atrás. En el paso de selección se recorre el árbol desde la raíz hasta un nodo que aún tenga nodos hijos por agregar, el recorrido es guiado por una política de selección que determina qué nodo debe visitarse en cada nivel. En el paso de expansión un nodo se agrega al árbol. En el paso de simulación, se realiza una ejecución del juego partiendo del nodo hijo, en esta simulación el agente realiza los movimientos de los jugadores siguiendo una política de simulación hasta terminar el juego. En el paso de propagación hacia atrás, el resultado de la simulación es propagado en todos los nodos visitados actualizando sus estadísticas.

Cada vez que el algoritmo ABMC realiza el paso de selección se enfrenta con el siguiente dilema: ¿qué nodo se debe visitar?, ¿aquel que parece ser el mejor hasta el momento? o ¿nodos menos prometedores que quizás resulten ser mejores en iteraciones posteriores? Este dilema es conocido como el dilema de exploración-explotación, al cual pertenece el Problema del Bandido Multi-Armado (PBMA) [38, 50, 55]. El PBMA consiste en lo siguiente [5]: dado un conjunto de K máquinas traga

monedas, donde cada una de ellas tiene cierta probabilidad de dar una recompensa, la pregunta es; ¿en qué máquina debe jugarse si se desea maximizar la recompensa?, la que ha dado la mayor recompensa hasta el momento o se debe probar otra máquina con la esperanza de que se obtenga una mejor recompensa. Upper Confidence Bound (UCB), propuesto por Auer et al. en [5], es el algoritmo más popular en PBMA, y se ha usado en ABMC como política de selección con buenos resultados [20], a esta combinación se le conoce como Upper Confidence Bounds Applied to Trees (UCT).

En esta tesis se reporta el desarrollo de dos políticas de selección específicas para General Game Playing que toma como base las políticas desarrolladas para el Problema del Bandido Multi-Armado. La principal característica de estas políticas es que se adaptan al árbol de juego al hacer uso de una función (en lugar de una constante) que controla la etapa de exploración con base al número de nodos del nivel del árbol donde se estén aplicando. Para obtener estas dos políticas de selección se realizaron dos experimentos, en los cuales se hicieron uso de una serie de conjuntos de problemas de bandido con diversas cantidades de máquinas los cuales emulan el número de nodos de un árbol de juego. Con estos conjuntos y junto a un algoritmo basado en intervalos y al algoritmo de recocido simulado (algoritmo utilizado en problemas de optimización) se obtuvieron las funciones que son la principal característica de las políticas propuestas en esta tesis. La finalidad de estas políticas es que, los agentes que las implementen tengan un mejor rendimiento en GGP, en términos de victorias, en comparación con los agentes basados en UCT.

Justificación

UCT debido a sus resultados en juegos específicos, es la política más utilizada en GGP [11, 12], esto se debe principalmente a que es independiente del dominio, es decir no requiere de conocer el juego que se aborde sino cómo es el comportamiento de la distribución de recompensas del mismo. Por otro lado, desde la concepción de UCB se han desarrollado nuevas políticas para PBMA como UCB-Tuned [20], UCB-V [3], UCB-Minimal [36], MOSS [2], entre otros [8]. Estas políticas no se han probado en GGP, esto debido a los buenos resultados obtenidos por UCB. Sin embargo, estas

políticas también son independientes de dominio, por lo cual se pueden aplicar a GGP pudiendo presentar mejores resultados que UCB.

Perick et al. [40] hace un comparativo de varias políticas en ABMC con el juego de *Tron* donde UCB-Tuned es quien tiene el mejor desempeño. Derivado de esto, surge la siguiente pregunta de investigación: Tomando como referencia las políticas existentes en el Problema del Bandido Multi-Armado, ¿Es posible desarrollar una política de selección que dé un rendimiento superior a Upper Confidence Bound en General Game Playing en términos de victorias?

Objetivo de tesis

En esta tesis se propone el desarrollo de una política de selección adecuada a GGP tomando como referencias las políticas UCB, UCB-Tuned y ETC, desarrolladas para PBMA, con la finalidad de generar agentes con mejor rendimiento que aquellos que usan a UCB como política de selección.

Hipótesis

Una política de selección basada en las características de las políticas de activación del Problema del Bandido Multi-Armado y que aproveche las características del árbol de juego, permitirá lograr un mayor número de victorias en contexto de General Game Playing.

Estructura de la tesis

La Tesis cuenta con cinco capítulos, lo cuales tienen el siguiente contenido:

- **Capítulo 1 Marco Teórico**: En este capítulo se tratan los diferentes tópicos relacionados con el proyecto de investigación.

- **Capítulo 2 Estado del Arte**: En este capítulo se tratan los avances y trabajos relacionados con el proyecto de investigación.

- **Capítulo 3 Metodología**: Incluye la metodología para dar solución al problema planteado en esta tesis.

- **Capítulo 4 Políticas Propuestas**: La descripción de las políticas desarrolladas para GGP propuestas en esta Tesis, son descritas en este capítulo además de los resultados experimentales, mediante los cuales se analiza el rendimiento de dichas propuestas.

- **Capítulo 5 Conclusiones**: Con las políticas propuestas en la Tesis, así como las líneas abiertas de estudio, se incluyen en este capítulo las conclusiones que se obtuvieron de este trabajo de tesis.

De igual forma, se incluyen los siguientes Anexos:

- **Anexo A Juegos de Tablero**: Donde se describe los juegos de tablero para dos jugadores que se utilizaron para evaluar la política propuesta en esta tesis.

- **Anexo B Demostración de Teoremas**: En este anexo se muestran las demostraciones de los teoremas mostrados en el Capítulo 1 Marco Teórico.

- **Anexo C: Publicaciones** Las evidencias de las publicaciones, producto de la realización de esta tesis de investigación, se muestran en este anexo.

Capítulo 1

Marco Teórico

En este capítulo se muestran los conceptos, definiciones y teorías referentes al área de General Game Playing, el método de Árbol de Búsqueda Monte Carlo y del Problema del Bandido Multi-Armado, que servirán para sustentar los siguientes capítulos.

1.1. General Game Playing

Para Bjornsson y Finnsson [6] GGP es el área de la IA que tiene como objetivo crear agentes inteligentes que puedan aprender de forma automática cómo jugar hábilmente una amplia variedad de juegos, con solo las descripciones de las reglas de estos.

En GGP se abordan juegos *finitos*, *síncronos* y con un número fijo de jugadores [26, 25]:

- Se dice que un juego es finito si toma lugar en ambientes con un número finito de estados, de los cuales uno es el estado inicial y uno o más estados finales. En cada estado los jugadores disponen de cero o más acciones (movimientos) cuya respuesta es otro estado del juego.
- Un juego es considerado síncrono cuando todos los jugadores realizan movimientos en todos los pasos, incluso cuando algún jugador realice un movimiento de no *acción* o de no hacer nada.
- Un juego se dice que tiene un número fijo de jugadores cuando en todas sus fases el número de jugadores permanece inalterable.

En general los juegos comúnmente utilizados en GGP son los *juegos combinatorios* estos tienen las siguientes características [45]:

- Son únicamente para dos jugadores.
- Son por turnos, es decir los jugadores alternan entre sus jugadas.
- El juego debe continuar hasta que algún jugador no pueda hacer un movimiento válido, es decir son finitos.
- Los jugadores tienen conocimiento de todo lo que sucede en el tablero en cualquier momento, son juegos con información completa.
- No tienen ningún elemento de azar.

Estas características se adecuan a los agentes GGP ya que se garantiza que el juego termina en algún punto, el agente responde a las acciones del otro y al tener conocimiento de lo que sucede y podría suceder en el tablero el agente puede planear sus próximas jugadas sin esperar que exista intervención del azar.

Algunos ejemplos de juegos combinatorios son el ajedrez, havannah, othello, domineering, entre otros. Los juegos de cartas como el póker no se consideran juegos combinatorios (sin embargo, también se han abordado en GGP [44]), ya que no cumplen con varias de las características de los juegos combinatorios como:

- Involucran azar.
- Los jugadores solo tienen conocimiento de una parte del juego.
- No hacen uso de un tablero.

1.2. Árbol de Búsqueda Monte Carlo

Los juegos combinatorios pueden ser representados por un *árbol de juego* (otra forma de representación es por medio de máquinas de estados [27]), donde cada nodo representa un estado del

juego y cada arista representa un movimiento y, en consecuencia, cada nodo hoja representa un estado final del juego. La Figura 1.1 muestra una parte del *árbol de juego* para el juego del *Gato* o *Tic-Tac-Toe*, en esta figura se pueden observar los diferentes movimientos que pueden realizar los jugadores a partir del estado del juego representado por el nodo raíz.

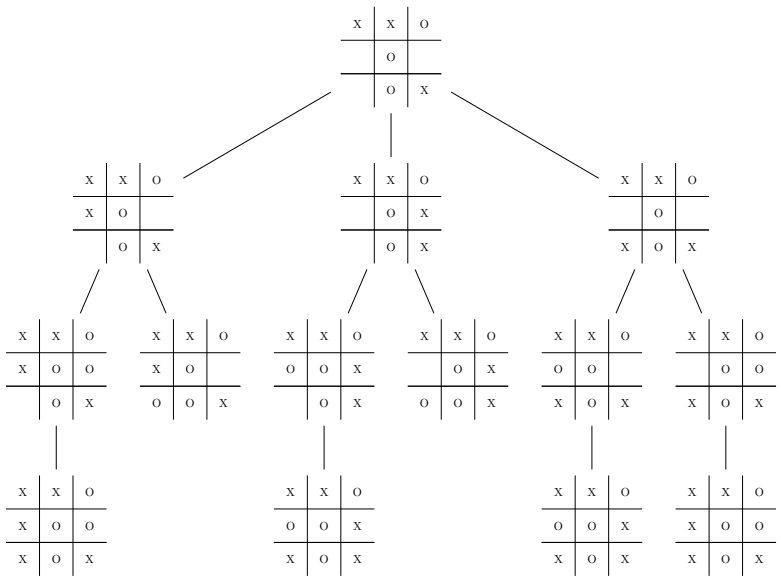


Figura 1.1: Árbol de Juego parcial para Tic-Tac-Toe

De acuerdo con el árbol de juego de la Figura 1.1, la mejor jugada que el jugador X puede realizar es colocar una X en la casilla inferior izquierda con lo cual evitaría perder inmediatamente y quizás ganar dependiendo de la respuesta del contrincante.

Un agente puede aprovechar este árbol de juego para generar sus estrategias y ganar el juego, la idea base radica en que en cada turno el agente consulte el árbol de juego y elija los nodos que conduzcan a hojas donde se produzcan victorias y evite aquellos nodos que conduzcan a hojas donde el agente pierda. Es decir, el agente realiza una búsqueda de nodos en cada turno.

En un inicio los agentes GGP hacían uso de algoritmos de búsqueda en árbol como *Minimax* [27], *Alfa-Beta* [27] y *Búsqueda en Profundidad con Profundidad Limitada* (BPPL) [27]. Sin embargo, debido a que existen juegos cuyos árboles de juego son bastante grandes ya sea en anchura, en

profundidad o ambos, los agentes basados en estos algoritmos resultan bastante ineficientes. No es hasta la llegada del método Árbol de Búsqueda Monte Carlo (ABMC) donde los agentes GGP pudieron enfrentar este tipo de árboles y aumentaron su rendimiento. El éxito de ABMC se debe a lo siguiente:

- Se puede aplicar en cualquier momento ya que el nodo raíz del árbol usado por ABMC puede ser cualquier estado del juego.
- A diferencia de otros métodos ABMC no requiere expandir por completo el árbol de juego. Al ser un método probabilístico, ABMC explora el árbol hasta cierto nivel y hace uso de estadística para determinar qué tan probable es que un nodo conduzca a una victoria.
- No requiere de conocimiento previo como el método Minimax. Lo único que requiere saber es cuántas veces se ha visitado cada nodo y cuántas victorias se han logrado en dicho nodo.

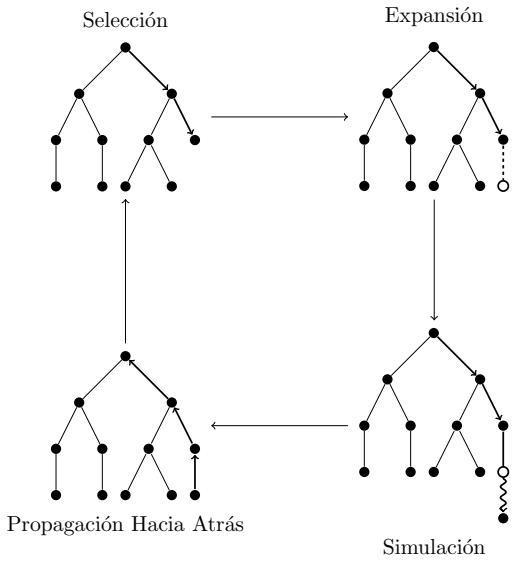


Figura 1.2: Árbol de Búsqueda Monte Carlo

En su ejecución ABMC consiste en cuatros pasos (ver Figura 1.2): Selección, Expansión, Simulación y Propagación Hacia Atrás. Estos pasos son repetidos en ese orden hasta que se cumpla un

criterio de parada, que puede ser por tiempo de ejecución o número de iteraciones. El Algoritmo 1 muestra los pasos de ABMC cuyo criterio de parada es número de iteraciones.

Algoritmo 1: Árbol de Búsqueda Monte Carlo

Entrada: s un nodo que representa un estado del juego

Salida: a un movimiento

```
1 mientras simulaciones < maximo hacer
2     v ← Selección(s);
3     r ← Simulación(v);
4     simulaciones ← simulaciones + 1;
5     Propagación Hacia Atrás(v, r);
6 devolver Movimiento(s)
```

Selección En este paso el método recorre el árbol desde el nodo raíz hasta encontrar un nodo que aún tenga hijos y que no estén en el árbol (Algoritmo 2). Una vez encontrado este nodo, se llama al paso de Expansión.

Algoritmo 2: Árbol de Búsqueda Monte Carlo: Selección

Entrada: v un nodo que representa un estado del juego

```
1 mientras v no es hoja hacer
2     si v no completamente expandido entonces
3         devolver Expandir(v)
4     en otro caso
5         v ← PoliticaSeleccion(v)
6 devolver v
```

El recorrido que realiza el paso de Selección esta guiado por una *política de selección*, la cual indica qué nodo debe ser explorado en cada nivel. En su forma más básica, la política de selección consiste en elegir el nodo que tenga la mayor razón entre victorias y visitas, puede verse este proceso en el Algoritmo 3.

Algoritmo 3: Árbol de Búsqueda Monte Carlo: Política Selección

Entrada: v un estado del juego

```
1 mejor ← nulo;
2 max ← −1;
3 para cada hijo de v hacer
4     t ← hijo.victorias / hijo.visitas;
5     si max < t entonces
6         max ← t;
7         mejor ← hijo;
8 devolver mejor
```

En cada iteración, en el paso de selección, el método ABMC se enfrenta a la siguiente decisión: ¿qué nodo debe visitarse, el nodo que hasta el momento ha dado el mayor número de victorias o algún otro nodo menos prometedor que quizás en futuras interacciones dé mejores resultados? Lo anterior se conoce como el Dilema de Explotación-Exploración donde la idea es encontrar un equilibrio entre explorar las acciones disponibles mientras se elige la mejor acción tan a menudo como sea posible [5].

Expansión En este paso se agrega un nodo hijo (que aún no está en el árbol) del nodo encontrado en el paso de selección, este proceso se muestra en el Algoritmo 4.

Algoritmo 4: Árbol de Búsqueda Monte Carlo: Expansión

Entrada: v un estado del juego

```
1 v' ← nuevo hijo de v;
2 v'.padre ← v;
3 devolver v'
```

Simulación Partiendo del estado representado por el nodo recién agregado, el método realiza los movimientos de los jugadores de manera aleatoria hasta lograr un resultado.

Algoritmo 5: Árbol de Búsqueda Monte Carlo: Simulación

Entrada: v un estado del juego

```
1 mientras v no es hoja hacer
2     v ← Elegir de forma aleatoria un nodo hijo de v;
3 devolver resultado de v
```

Propagación Hacia Atrás En este paso el resultado de la simulación es propagado en todos los nodos visitados actualizando sus estadísticas, es decir se incrementa el número de visitas y en caso de que la simulación resulte en una victoria también se incrementan las victorias (ver Algoritmo 6).

Algoritmo 6: Árbol de Búsqueda Monte Carlo: Propagación Hacia Atrás

Entrada: v un estado del juego

Entrada: r resultado

```
1 mientras v no sea nulo hacer
2     v.visitas ← v.visitas + 1;
3     si resultado es victoria entonces
4         v.victorias ← v.victorias + 1;
5     v ← v.padre;
```

Elección de movimiento Una vez que el método termina, el movimiento que el agente debe realizar se elige entre los hijos del nodo raíz de acuerdo con alguno de los siguientes criterios [8]:

- **Hijo Máximo** Se elige el nodo con el mayor número de victorias.
- **Hijo Robusto** Se elige el nodo con el mayor número de visitas.
- **Hijo Máximo-Robusto** Se elige el nodo con el mayor número de victorias y de visitas. Se debe tener en cuenta que el nodo debe cumplir con las dos condiciones, en caso de que ningún nodo cumpla estas dos condiciones se debe aplicar nuevamente ABMC hasta que un nodo satisfaga dichas condiciones.

- **Promedio de Victorias** Se elige el nodo con el mayor promedio de victorias respecto al número de visitas.

- **Por Política** Se elige el nodo en base a la política de selección que se usó en el paso de Selección.

1.3. Problema del Bandido Multi-Armado

Como se mencionó anteriormente, ABMC debe lidiar con el Dilema de Explotación-Exploración en el paso de Selección. El Problema del Bandido Multi-Armado también lidia con este dilema y debido a su semejanza se han utilizado las técnicas desarrolladas en PBMA en el paso de selección de ABMC. En esta sección se definen algunos conceptos de probabilidad los cuales serán útiles para entender el PBMA y los algoritmos planteados para atacar el dilema de explotación-exploración desde este enfoque.

1.3.1. Conceptos básicos de Probabilidad

En teoría de la probabilidad se define un **experimento** como cualquier acción o proceso cuyo resultado este sujeto a la incertidumbre, por ejemplo, el lanzamiento de una moneda o un dado de seis caras. Se conoce como **espacio muestral** al conjunto de todos los resultados posibles de un experimento. Para el lanzamiento de una moneda, el espacio muestral es conformado por dos elementos $\{\text{cara}, \text{cruz}\}$, para el caso del lanzamiento de un dado el espacio muestral es $\{1, 2, 3, 4, 5, 6\}$.

Se define como **evento** al subconjunto de los resultados del espacio muestral, siendo un **evento simple** aquel evento formado por solo un resultado y **evento compuesto** al evento formado por más de un resultado. Ejemplo de un evento en el lanzamiento de un dado sería obtener un número par, en este caso obtener un número del subconjunto $\{2, 4, 6\}$. La probabilidad de un evento A consiste en asignar a dicho evento un número $\mathbb{P}(A)$ como una medida precisa de la oportunidad de que el evento A ocurra. Para el evento A de obtener un número par en el lanzamiento de un dado $\mathbb{P}(A) = 0.5$.

Se define como **variable aleatoria** a una función que tome como dominio un espacio muestral S y como rango al conjunto de números reales. Se conoce como **variable aleatoria Bernoullli** a aquella variable aleatoria cuyos únicos dos valores posibles son $\{0, 1\}$. Una variable aleatoria se dice que es **discreta** si tiene un número contable de valores posibles es decir constituyen un conjunto finito. En cambio, si la variable aleatoria tiene un número infinito de valores posibles no contables se dice que es **continua**. En este sentido, dos variables aleatorias son **independientes** si el hecho de conocer el valor de una no proporciona información del valor que obtendrá la otra.

La **distribución de probabilidad** es una función que establece la probabilidad de que la variable aleatoria X tome el valor posible x, esto es $p(x) = \mathbb{P}(X = x)$. La distribución de probabilidad debe cumplir las siguientes condiciones:

- $p(x) \geq 0$
- $\sum_x p(x) = 1$

La **media** (también conocido como valor esperado, expectativa o esperanza) de una variable aleatoria discreta X con distribución de probabilidad $p(x)$ se define como $\mu = \mathbb{E}[X] = \sum_x xp(x)$. De este modo, la media para una suma de variables aleatorias discretas como: $\mu = \mathbb{E}[X_1 + X_2 + \cdots + X_n] = \mathbb{E}[X_1] + \mathbb{E}[X_2] + \cdots + \mathbb{E}[X_n]$.

1.3.2. Definición del Problema del Bandido Multi-Armado

El PBMA consiste en lo siguiente: supóngase que se tienen un conjunto de K máquinas tragamonedas, cada una de las cuales tiene una probabilidad p_i de dar una *recompensa* (en el ámbito de las tragamonedas puede ser un objeto de valor). Dicha probabilidad es desconocida e independiente de las otras máquinas, si se tienen n rondas y si solo se puede activar una máquina a la vez, ¿cuál máquina debe activarse en cada ronda con la finalidad de maximizar la cantidad de recompensas obtenidas?

Formalmente Auer et al. [5] define al PBMA por las variables aleatorias $X_{i,n}$ con $1 \leq i \leq K$ y $n \geq 1$, donde cada i es el índice de cada máquina tragamonedas. Activaciones sucesivas de

una máquina i producen recompensas $X_{i,1}, X_{i_2}, \cdots$, las cuales son independientes e idénticamente distribuidas de acuerdo con una ley desconocida con una expectativa μ_i también desconocida.

Una política de activación es un algoritmo que indica qué máquina activar en cada ronda en PBMA, por lo general la política de activación se basa en las activaciones previas y las recompensas obtenidas. Se han propuesto diversas políticas para PBMA siendo Upper Confidence Bound (UCB) la más popularmente usada [35], debido principalmente a su simplicidad y rendimiento.

La elección de un nodo en el paso de selección del método ABMC es equivalente a elegir una máquina tragamonedas en PBMA [8], esto es que cada nodo puede ser representado por una máquina tragamonedas con lo siguiente:

- El número de victorias del nodo corresponde al número de veces en que una máquina ha dado una recompensa.
- El número de simulaciones del nodo corresponde al número de veces en que la máquina se ha activado.

Gracias a esta equivalencia se pueden utilizar las políticas de activación en PBMA como políticas de selección en ABMC, siendo la más popular UCB cuyo uso se conoce como UCT.

1.3.3. Upper Confidence Bound

La principal dificultad del PBMA radica en que se desconoce la media de cada una de las máquinas, sin embargo, puede estimarse usando la Desigualdad Chernoff-Hoeffding.

Desigualdad Chernoff-Hoeffding[5, 29] . Sean $X_1, X_2, \cdots, X_n$ variables aleatorias independientes con $X_j \in [0,1]$, $\bar{x} = \frac{1}{n}\sum_{j=1}^{n} X_j$ es la media empírica y $\mu = \mathbb{E}[\bar{x}]$. La desigualdad de Chernoff-Hoeffding establece que para todo $\epsilon \geq 0$:

$$\mathbb{P}(\bar{x} + \epsilon \leq \mu) \leq e^{-2n\epsilon^2} \qquad \mathbb{P}(\bar{x} - \epsilon \geq \mu) \leq e^{-2n\epsilon^2} \tag{1.1}$$

Es decir que la probabilidad de que la media de un conjunto de variables aleatorias independientes sea superior (o inferior) a la media empírica más (o menos) cierto valor ϵ esta acotada

exponencialmente.

Adaptando la Desigualdad Chernoff-Hoeffding al PBMA se tiene para la máquina i:

$$\mathbb{P}\left(\bar{x}_{i,T_i} + \epsilon_i \leq \mu_i\right) \leq e^{-2T_i\epsilon_i^2} \tag{1.2}$$

donde $\bar{x}_{i,T_i}$ es el promedio de recompensas obtenidas de la máquina i de las T_i veces que se ha activado y μ_i es la media desconocida de la máquina i.

Es decir que la probabilidad de que la media de la máquina i sea superior al promedio de las recompensas obtenidas más cierto valor es a lo mucho $e^{-2T_i\epsilon_i^2}$

Auer et al. [5] propone que esta probabilidad debe decaer conforme el número de rondas t incrementa, esto puede ocurrir al hacer:

$$\mathbb{P}\left(\bar{x}_{i,T_i} + \epsilon_i \leq \mu_i\right) \leq e^{-2T_i\epsilon_i^2} = t^{-2\alpha} \qquad \alpha > 0 \tag{1.3}$$

y de esta manera se puede obtener el valor de ϵ_i:

$$\epsilon_i = \sqrt{\frac{\alpha \log t}{T_i}} \tag{1.4}$$

por lo cual (1.2) queda como:

$$\mathbb{P}\left(\bar{x}_{i,T_i} + \sqrt{\frac{\alpha \log t}{T_i}} \leq \mu_i\right) \leq t^{-2\alpha} \tag{1.5}$$

o lo que es lo mismo

$$\mathbb{P}\left(\bar{x}_{i,T_i} + \sqrt{\frac{\alpha \log t}{T_i}} \leq \mu_i\right) > 1 - t^{-2\alpha} \tag{1.6}$$

es decir se tiene una probabilidad mayor a $1 - t^{-2\alpha}$ de que el **estimado de la media**: $\bar{x}_{i,T_i} + \sqrt{\frac{\alpha \log t}{T_i}}$ sea inferior a la media real μ_i.

Por lo anterior al tener un t lo suficientemente grande, se tiene una alta posibilidad de que se

cumpla

$$\bar{x}_{i,T_i} + \sqrt{\frac{2\alpha \log t}{T_i}} \leq \mu_i$$

por lo cual, siguiendo un enfoque elitista, en cada ronda debe activarse la máquina i cuyo estimado sea el mayor al del resto de máquinas esto se conoce como **Upper Confidence Bound** y se describe en el algoritmo 7.

Algoritmo 7: Upper Confidence Bound

Entrada: K máquinas y T Rondas

1 Activar cada máquina una sola vez;

2 **para** $t \leftarrow K + 1$ **a** T **hacer**

3 Activar la máquina i que maximice;

4

$$\bar{x}_{i,T_i} + \sqrt{\frac{\alpha \log t}{T_i}}$$

1.3.4. Control de la Explotación y la Exploración en UCB

El control de la explotación y exploración en UCB se da en la estimación de la media, la cual puede dividirse en dos términos:

$$\underbrace{\bar{x}_{i,T_i}}_{\text{Término de Explotación}} + \underbrace{\sqrt{\frac{\alpha \log t}{T_i}}}_{\text{Término de Exploración}}$$

Se puede observar que, en específico, el balance entre la explotación y la exploración se da de acuerdo con el número de veces que la máquina ha sido activada, es decir depende del valor de T_i. Después de cierto número de rondas t, si la máquina i ha sido pocas veces activada el término exploración tendrá un valor alto lo que indicará que es necesario más activaciones para determinar qué tan buena es. En cambio, si la máquina ha sido activada frecuentemente el término de exploración tenderá a 0 dejando que la decisión de qué tan buena es la máquina recaiga únicamente en el promedio de recompensas que dicha máquina ha dado.

Es necesario observar que el valor de α también influye en el término de exploración. Originalmente la constante de α en UCB tiene un valor de 2 [5] el cual actualmente se considera un estándar. Sin embargo, se ha propuesto que un valor de $\frac{3}{2}$ produce un mejor resultado [38], no obstante, para ciertos autores el valor de dicha constante debe ser determinado en base al problema a abordar pudiendo incluso a tener un valor de 0 [8].

1.3.5. Arrepentimiento Esperado de UCB

Auer et al. [5] y Munos et al. [38] definen el *arrepentimiento acumulado* como la pérdida obtenida, en términos de recompensas, debido a que la política no siempre activa la máquina óptima (la máquina con mayor probabilidad de dar una recompensa) sino una subóptima. Entonces, una política de activación tiene como objetivo minimizar el arrepentimiento acumulado.

El arrepentimiento acumulado en la ronda T para K máquinas se define como

$$R(T) = T\mu^* - \sum_{j=1}^{T} X_j \qquad \mu^* = \max_{1 \leq i \leq K} \mu_i$$

El *arrepentimiento esperado* o la pérdida que una política de activación obtendrá al no activar siempre la mejor máquina formalmente se define como:

$$\mathbb{E}[R(T)] = \sum_{i=1}^{K} \mathbb{E}[T_i]\, \Delta_i \qquad \Delta_i = \mu^* - \mu_i \qquad T_i = \sum_{j=1}^{t} 1\{I_j = i\}$$

donde T_i es el número de veces que la máquina i ha sido jugada y Δ_i es la diferencia entre la media de la máquina óptima μ^* y la media μ_i de la máquina i.

Para Auer et al. [5] el arrepentimiento esperado para UCB con $\alpha = 2$, está dado por el siguiente teorema:

Teorema 1 (Arrepentimiento Esperado para UCB con $\alpha = 2$)**.** *Si la política UCB es ejecutada en K máquinas ($K > 1$) con distribuciones de recompensas arbitrarias $(p_1, \cdots p_K)$, $p_i \in [0, 1]$, el*

arrepentimiento esperado en la ronda T es a lo mucho

$$R(T) \leq \sum_{i:\Delta_i>0} 8\frac{\ln T}{\Delta_i} + \frac{\pi^2}{3}\Delta_i$$

Jamieson [32] da un arrepentimiento general de UCB para cualquier valor de α:

Teorema 2 (Arrepentimiento Esperado General para UCB)**.** *Si la política UCB es ejecutada en K máquinas ($K > 1$) con distribuciones de recompensas arbitrarias $(p_1, \cdots p_K)$, $p_i \in [0, 1]$, el arrepentimiento esperado después de cada ronda T es a lo mucho*

$$R(T) \leq \sum_{i:\Delta_i>0} 4\alpha\frac{\ln T}{\Delta_i} + \frac{2\alpha}{\alpha - 1}\Delta_i \qquad \alpha > 0$$

La demostración de estos teoremas se puede ver en el apéndice B.

1.3.6. Otras Políticas de Activación

Aunque UCB es la política más usada tanto en PBMA como en ABMC, se han propuesto otras políticas entre las que destacan las siguientes.

Upper Confidence Bound Tuned. Auer [5] propone UCB-Tuned como una mejora a UCB, en la cual la fase de exploración se ve influida por la varianza muestral de las máquinas como puede observarse en el algoritmo 8.

Se debe observar que (1.8) es la varianza muestral donde s es el número de veces que se ha activado la máquina y t el número de veces que todas las máquinas han sido activadas. Debe observarse que la constante α es remplazada por una función de minimización entre el valor $\frac{1}{4}$ y la varianza muestral lo que permite que la fase de exploración se adapte de acuerdo a esos valores.

En el ámbito de los juegos de tablero UCB-Tuned se ha aplicado en juegos como Go, Otelo y Tron [8], pero no en GGP.

Algoritmo 8: Upper Confidence Bound Tuned

Entrada: K máquinas y T Rondas

1 Activar cada máquina una sola vez;

2 **para** $t \leftarrow K+1$ **a** T **hacer**

3 Activar la máquina i que maximice;

4

$$\bar{x}_{i,T_i} + \sqrt{\frac{\log t}{T_i} \text{mín}\left\{\frac{1}{4}, V_i\left(T_i\right)\right)\right\}} \tag{1.7}$$

$$V_j\left(s\right) = \left(\frac{1}{s}\sum_{\tau=1}^{s} x_{j,\tau}^2\right) - \bar{x}_{j,s}^2 + \sqrt{\frac{2\log t}{s}} \tag{1.8}$$

Upper Confidence Bound With Variance (UCB-V) . Propuesto por Audibert et al. [3] es la política UCB la cual usa la varianza de las máquinas para determinar la máquina a activar, ver algoritmo 9.

Algoritmo 9: Upper Confidence Bound With Variance

Entrada: K máquinas y T Rondas

1 Activar cada máquina una sola vez;

2 **para** $t \leftarrow K+1$ **a** T **hacer**

3 Activar la máquina i que maximice;

4

$$\bar{x}_{i,T_i} + \sqrt{\frac{2\sigma_i^2 \zeta \log t}{T_i}} + c\frac{3\zeta \log t}{T_i} \tag{1.9}$$

Para (1.9), σ_j^2 representa la varianza de la máquina j y las constantes $\zeta, c > 0$ son parámetros de afinación que permiten controlar el comportamiento del algoritmo.

Upper Confidence Bound Minimal (UCB-Minimal) . Maes et al. [36] proponen un enfoque sistemático para la obtención automática de políticas que sean simples y tengan un buen rendimiento, del cual resulta UCB-Minimal, política que puede verse como una versión sencilla de UCB,

ya que solo requiere del promedio de las recompensas y el número de activaciones con lo cual ha dado buenos resultados en diferentes instancias del PBMA.

Algoritmo 10: Upper Confidence Bound Minimal

Entrada: K máquinas y T Rondas

1 Activar cada máquina una sola vez;

2 **para** $t \leftarrow K+1$ **a** T **hacer**

3 Activar la máquina i que maximice;

4

$$\bar{x}_{i,T_i} + \frac{c}{T_i} \tag{1.10}$$

Debe observarse que esta política aún depende de una constante c la cual debe determinarse de manera empírica.

Minimax Optimal Strategy in the Stochastic Case (MOSS) . Política inspirada en UCB que de igual forma toma como referencia la desigualdad de Hoeffding [2] (algoritmo 11).

Algoritmo 11: Minimax Optimal Strategy in the Stochastic Case

Entrada: K máquinas y T Rondas

1 Activar cada máquina una sola vez;

2 **para** $t \leftarrow K+1$ **a** T **hacer**

3 Activar la máquina i que maximice;

4

$$\bar{x}_{i,T_i} + \sqrt{\frac{\text{máx}\left(\ln\left(\frac{t}{KT_i}\right), 0\right)}{T_i}} \tag{1.11}$$

La diferencia principal entre MOSS y UCB radica en que cuando una máquina se ha activado más de $\frac{t}{K}$ veces, solo se toma en cuenta el promedio de recompensas de dicha máquina.

De acuerdo con [2], el arrepentimiento de MOSS esta dado por:

$$R(T) \leq 23K \sum_{i:\Delta_i>0} \frac{\text{máx}\left(\ln\left(\frac{T\Delta_i^2}{K}\right), 1\right)}{\Delta_i} \tag{1.12}$$

Explore-Then-Commit (ETC) . Esta política no sigue la idea planteada por UCB sino que tiene la particularidad de que tiene una etapa de exploración pura donde cada máquina es activada una cierta cantidad m de veces antes de iniciar con una etapa de explotación donde la máquina a activarse será de acuerdo a el promedio de sus recompensas, proceso que puede verse en el algoritmo 12.

Algoritmo 12: Explore-Then-Commit

Entrada: K máquinas y T Rondas

Activar cada máquina una sola vez;

para $t \leftarrow K+1$ **a** T **hacer**

  Activar la máquina i tal que;

$$i = \begin{cases} (t \mod K) + 1 & \text{si } t \leq mK \\ \text{máx}_i\, \bar{x}_{i,T_i} & \text{if } t > mK \end{cases} \tag{1.13}$$

1.4. Algoritmo de Recocido Simulado

El algoritmo de Recocido Simulado (RS) es un algoritmo de búsqueda para problemas de optimización cuya particularidad es la implementación de aleatoriedad con la finalidad de evitar caer en mínimos/máximos locales [41] además de poder encontrar el mínimo/máximo global sin importar la configuración inicial [37]. El nombre de RS surge del proceso físico de templado de metales, el cual consiste en calentar los metales a una alta temperatura para posteriormente enfriarlos poco a poco [41].

En cada iteración RS en lugar de escoger una solución que mejore a la solución actual, escoge una solución vecina de manera aleatoria, si esta solución mejora a la solución actual se acepta

como nueva solución, en caso contrario se acepta de acuerdo con una cierta probabilidad [41]. El proceso anterior puede verse más claramente en el algoritmo 13 el cual está enfocado a problemas de minimización.

Algoritmo 13: Recocido Simulado

Entrada: $T_{inicial}$ Temperatura Inicial
Entrada: n Número de iteraciones
Entrada: $S_{inicial}$ Solución Inicial

```
1  T_actual ← T_inicial;
2  S_actual ← S_inicial;
3  para t ← 1 a t ≤ n por 1 hacer
4      S_vecino ← obtenerVecino(S_actual);
5      si valor(S_actual) > valor(S_vecino) entonces
6          S_actual ← S_vecino;
7      en otro caso
8          S_actual ← S_vecino de acuerdo a una probabilidad exp((valor(S_actual) − valor(S_vecino)) / T_actual);
9      T_actual ← esquemaEnfriamiento(T_inicial);
10 devolver S_actual
```

De acuerdo con [37] la eficacia del algoritmo de recocido simulado depende de dos parámetros:

Temperatura Inicial Esta no debe ser muy alta para evitar que RS requiera de un tiempo excesivo de cálculo, pero tampoco debe ser muy baja para evitar caer en mínimos/máximos locales.

Esquema de Enfriamiento La velocidad de enfriamiento repercutirá en la calidad de la solución y el tiempo necesario para encontrarla. Un enfriamiento rápido repercute en una solución de baja calidad por el contrario un enfriamiento lento requería de un mayor tiempo de cálculo. Se han propuesto diversos esquemas de enfriamiento [37]:

Enfriamiento Logarítmico Un esquema de enfriamiento que converge lentamente al mínimo global lo que repercute en el tiempo de cálculo.

$$T_t = \frac{c}{\log(1+t)}$$

donde c es una constante positiva dependiente del problema y t corresponde a la inter-

acción.

Enfriamiento Geométrico El esquema de enfriamiento geométrico es más rápido que el enfriamiento logarítmico:

$$T_t = T_{inicial}\beta^t$$

donde β es una constante entre 0 y 1, si se desea un enfriamiento lento β debe tomar un valor cercano a 1.

Enfriamiento Lineal Al igual que el enfriamiento geométrico, el enfriamiento lineal es más rápido que el enfriamiento logarítmico:

$$T_t = T_{inicial} - n^t$$

donde n es un parámetro de decaimiento. Debe observarse que tanto el enfriamiento geométrico como el lineal converge a un mínimo muy cercano al mínimo global.

Enfriamiento Aritmético-Geométrico Esquema inspirado en la procesión Aritmético-Geométrico y se define como:

$$T_{t+1} = aT_t + b$$

donde a debe ser diferente de 1 y b debe ser diferente de 0.

.

Capítulo 2

Estado del Arte

La política de selección del ABMC indica, en cada nivel, qué nodo se debe recorrer en el árbol del juego. En la literatura se pueden encontrar diferentes formas de aplicar las políticas de selección para influenciar este recorrido y balancear la exploración y la explotación, muchas de estas mejoras se basan en el dominio en que se aplican [8].

Usar UCB como política de selección, tiende a que ABMC se enfoque en la exploración antes de la explotación lo cual es beneficioso cuando se tienen árboles de juego con un factor de ramificación pequeño, sin embargo, cuando el factor de ramificación es grande, UCB tenderá a explorar los nodos de los primeros niveles dejando sin explorar nodos que se encuentren en niveles más profundos. **Urgente Jugar Primero (UJP)** [24], originalmente ideada para el juego de GO, se ha propuesto como una forma de fomentar la exploración temprana de nodos en niveles profundos, la idea base consiste en asignar un valor fijo a nodos no visitados y usar el valor UCB de nodos visitados; de esta manera ABMC visitará los nodos con mayor valor. Un valor UJP alto fomentará la exploración de cada movimiento, sin embargo, un valor bajo provocará la explotación temprana de nodos prometedores de acuerdo con los resultados obtenidos por las simulaciones.

En ocasiones cuando tiene que lidiar con árboles de juego con un factor de ramificación grande se puede observar que muchos movimientos son similares, lo que conduce a que ABMC tenga que realizar una cantidad mayor de simulaciones para poder distinguir entre estos movimientos [8].

Una idea para tratar con factores de ramificaciones grandes es agrupar los movimientos similares y usar una política como UCB para: primero elegir un grupo y posteriormente elegir de ese grupo el movimiento [54]. Esta idea se ha aplicado en el juego de Go y Amazonas probando que con grupos de movimientos se puede reducir el factor de ramificación [54, 18].

Otra forma de influir en el recorrido del árbol de juego, y de las simulaciones, es hacer uso de los **Movimientos Decisivos**, movimientos que conducen a una victoria inmediata, y **Movimientos Anti-Decisivos**, movimientos que evitan que el contrincante haga un movimiento decisivo. Así el recorrido debe hacerse de la siguiente forma, si el jugador tiene un nodo que represente un movimiento decisivo se selecciona este nodo, sino se usa UCB de manera usual. Los Movimientos Decisivos y Anti-Decisivos se han probado en los juegos Hex y Havannah, y se ha observado un incremento en el número de victorias de los agentes [53].

En ABMC los juegos se representan por medio de una estructura tipo árbol sin embargo, dada la naturaleza de los juegos, muchos de los nodos representan un mismo estado del juego [8, 18]. Childs et al. [18] propone el uso de **tablas de transposición** para almacenar información acerca de los estados del juego y compartir en iteraciones sucesivas cuando ABMC encuentre un estado ya almacenado en la tablas. Las tablas de transposición se han usado en GGP en juegos de un solo jugador que en combinación con UCB han probado mejorar el rendimiento de los agentes [39].

Todos los Movimientos como el Primero (TMP) es una mejora originalmente pensada para el juego GO el cual consiste en mantener las estadísticas de todos los movimientos usados durante el paso de simulación [8]. TMP proporciona una forma sencilla de compartir conocimientos entre nodos relacionados en el árbol de juego lo que da como resultado una estimación rápida pero sesgada de los valores de los nodos, la cual a menudo puede determinar el mejor movimiento después de una pequeña cantidad de simulaciones y por ende mejorar el rendimiento del algoritmo de búsqueda [23]. $\alpha-$TMP [28] combina la idea de TMP con UCB al mantener dos conjuntos de estadísticas en cada nodo: uno con las estadísticas estándar de ABMC y el otro con las estadísticas obtenidas por las simulaciones, de esta manera el valor del nodo corresponde a α veces las estadísticas de las simulaciones más $\alpha-1$ veces las estadísticas estándar. **Estimación del Valor de Acción Rápida (EVAR)** [28], originalmente propuesto para Go [22], es una variante de $\alpha-$TMP donde

cada nodo tiene su propio α el cual inicia en 1 y decrementa conforme el número de las simulaciones en el nodo incrementa. **EVARG** [10] es una generalización de EVAR cuyo principio es utilizar los valores EVAR de nodos superiores ya que estos tiene mayor precisión al tener un mayor número de simulaciones, en lugar de un nodo inferior con solo unas cuantas simulaciones. EVARG requiere de una *constante de referencia* que determine cuántas simulaciones son necesarias para considerar que un nodo puede influir en sus descendientes, si esta constante vale 0, EVARG se comporta exactamente igual que EVAR, de aquí que se considere una generalización de este. EVARG resulta tener un mejor desempeño que EVAR en los juegos Atari-Go, Knightthrough, Domineering y Go [10].

Técnica de Muestreo de Movimientos-Promedio (TMMP) o Heurística Histórica es una técnica que se basa en los movimientos previamente usados para guiar el paso de selección en ABMC [8] esta técnica parte de la idea de que, independientemente del dominio, los movimientos que son buenos en un estado pueden serlo en otros [20] de aquí que también sea usada en el paso de simulación. TMMP mantiene un registro de los movimientos realizados y es usado para guiar la búsqueda de ABMC de acuerdo con un muestreo de Gibbs, esta técnica fue implementada en el agente CADIAPlayer campeón de General Game Playing [20].

Una de las mejoras que se ha realizado en ABMC es el ajuste de la constaste de exploración en UCB, generalmente este ajuste se hace en base al conocimiento que se tenga del dominio donde ABMC se aplique. Chaslot et al. [15] establece que la constante de exploración es 0 para el juego de Go cuando se utiliza UCB junto con la técnica TMP/EVAR lo que ha incrementado las victorias de su agente MoGo [15] y la misma idea se ha aplicado al agente Fuego [1]. Arneson et al. [1] extiende esta idea al juego Hex con el agente MoHex e igualmente a los agentes MoGo y Fuego, donde la identificación de nodos prometedores se deja a TMP/EVAR y el trabajo de explotación de nodos prometedores se deja a UCB. Al establecer la constante de exploración en 0 se logra un incremento en las victorias por MoHex [1].

Kozelek [34] propone dos enfoques de ajuste *en línea* a la constante de exploración de UCB en el juego Arimaa. En su primer enfoque **Decaer con el Tiempo** la constante de exploración empieza con una valor alto para fomentar la exploración y conforme el número de simulaciones aumenta el valor decrementa con la finalidad de fomentar la explotación. El segundo enfoque **Basado en**

Varianza, que parte de la idea de que entre más pequeña sea la varianza más estable será el nodo y consecuentemente menor debe ser la constante de exploración. Kozelek establece que estos enfoques lograron un 60 % de victorias sobre UCB con una constante fija (0.2) cuando se tiene un tiempo de ejecución pequeño.

El uso de ABMC y sus mejoras en el juego de Go ha provocado que se tengan que ajustar diversos parámetros, entre los que se encuentra la constante de exploración de UCB, para lograr un rendimiento máximo, generalmente esta tarea si se realiza de manera manual puede ser difícil y requerir de bastante tiempo. Chaslot et al. [17] propone el uso de Método de Entropía Cruzada (MEC) para llevar a cabo este ajuste de manera automática. Chaslot et al. usa MEC para ajustar 11 parámetros de su agente MANGO, para el cual se usó un lote de 500 juegos de Go encontrando que para este juego la constante de exploración de UCB debe ser de 0.43. Los resultados que Chaslot et al. obtuvo muestran que MANGO con los parámetros ajustados gracias a MEC pudo vencer a GNU Go y a MANGO sin parámetros ajustados. Chaslot et al. recalca que MEC puede aplicarse a cualquier agente que se base en ABMC y por ende a cualquier juego, como ejemplo podemos encontrar MEC aplicado al juego SameGame [52]. Kocsis et al. [33] propone otro enfoque para la optimización de múltiples parámetros al usar **Aproximación Estocástica de Perturbación Simultánea (AEPS)** en combinación con **Propagacion Resistente (PR)**, esta combinación se aplicó en los juegos Omaha Hi-Lo Poker (variante del Texas hold'em) y Líneas de Acción. Otra técnica para la optimización de parámetros aplicado a el juego de Go pero para la política de simulación se puede encontrar en el trabajo de Silver y Tesauro [46] donde proponen el algoritmo **Balanceo de Simulación** el cual es posteriormente extendido a tableros de 9×9 en el trabajo de Huang et al. [31].

Debido a la variedad de juegos que se pueden abordar por GGP, una optimización *fuera-de-línea* basada en las características de cada uno de ellos es inviable, sin embargo, se pueden encontrar trabajos de optimización de parámetros *en-línea*. Sironi y Winands [49] proponen tratar la optimización de parámetros en GGP como un **Problema de Bandido Multi-Armado Combinatorio (PBMAC)** el cual es una variante del PBMA donde las recompensas están dadas por una combinación de acciones en lugar de una sola acción. Sironi y Winands en cada iteración de ABMC determinan

qué valores de los parámetros se usarán en dicha iteración, esto de acuerdo con una política de PBMAC, y después de los cuatro pasos de ABMC el resultado obtenido por la simulación se utiliza para actualizar las estadísticas sobre la calidad de la combinación elegida de valores de parámetros. Sironi y Winands compararon las políticas de PBMAC: **Política de Bandido Multi-Armado**, **Expansión Jerárquica**, **Monte-Carlo Ingenuo** e **Información Lateral Lineal** para optimizar la constante de exploración de UCB, la constante de referencia de EVARG, la constante de $\alpha-$TMP y el parámetro ϵ de la estrategia ϵ-greedy a usar en lugar del muestreo Gibbs en TMMP. Los resultados obtenidos por Sironi y Winands muestran que Monte-Carlo Ingenuo e Información Lateral Lineal son las políticas que mejor rendimiento obtienen en la optimización en-línea, aunque en general tienen un rendimiento inferior que la optimización fuera-de-línea, sin embargo, consideran que es una alternativa viable cuando una optimización fuera-de-línea no es posible. Posteriormente Sironi et al. [47] comparan las políticas de PBMAC: **Algoritmo Evolutivo**, **Algoritmo Evolutivo de Bandido de N-Tuplas** y **Estrategia Evolutiva de Adaptación de la Matriz de Covarianza**, Monte-Carlo Ingenuo e Información Lateral Lineal como políticas de optimización de parámetros en-línea en GGP. Los resultados obtenidos por Sironi et al. [47] muestran que cuando se optimizan únicamente dos parámetros las políticas comparadas tiene un rendimiento similar o superior a una optimización fuera-de-línea. Cuando se optimizan cuatro parámetros el rendimiento de las políticas es inferior a la de una optimización fuera-de-línea, siendo más deficientes cuando se optimizan seis parámetros. Sin embargo, se sigue considerando una buena alternativa en dominios como GGP en especial Algoritmo Evolutivo de Bandido de N-Tuplas al tener el mejor rendimiento de las políticas comparadas. Finalmente Sironi y Winands [48] estudian el comportamiento de aleatorizar los parámetros de control de búsqueda para ABMC en GGP determinando que dicha aleatorización de parámetros puede diversificar la búsqueda.

Capítulo 3

Metodologia

En este capítulo se muestran los pasos metodológicos secuenciales (ver imagen 3.1) que se siguieron para obtener la política de selección específica para General Game Playing.

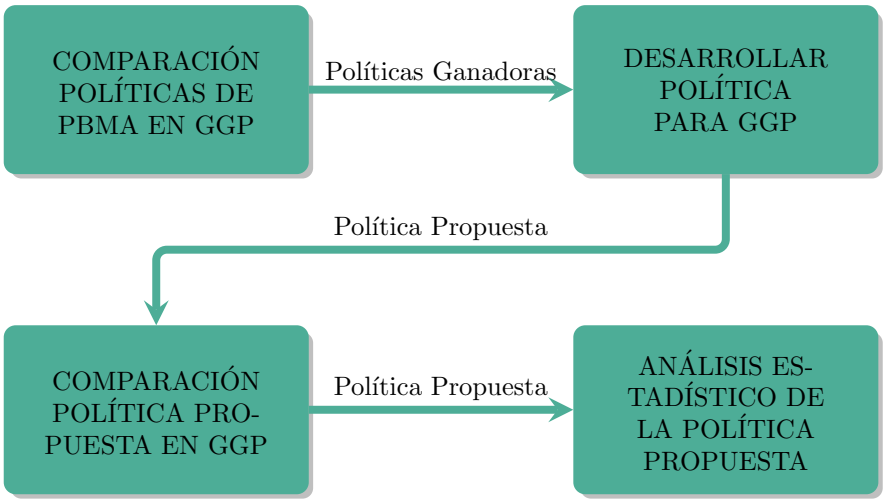


Figura 3.1: Metodología

Cada uno de los pasos mostrados en el diagrama será descritos a detalle en las siguientes secciones.

3.1. Comparativa de políticas del PBMA en GGP

Para desarrollar una política de selección adecuada para GGP que se base en las políticas de PBMA fue necesario comparar e identificar el rendimiento de estas con la finalidad de identificar las características que puedan aprovecharse en GGP.

El objetivo de comparar las políticas de PBMA radica en identificar qué políticas obtienen el mayor número de victorias en el ámbito de GGP. Para ello se realizaron dos *torneos* donde las políticas se enfrentan en distintos juegos de tableros bajo las siguientes consideraciones:

- Para la realización de los torneos se crearon agentes que hacen uso de ABMC e implementan las políticas de PBMA como políticas de selección.
- Se hace uso de distintos juegos de tablero para dos jugadores.
- Por cada juego, los agentes se enfrentan 3 veces a 100 partidas por vez, para un total de 300 partidas.
- En cada nueva partida los agentes cambiaron de rol, con la finalidad de que los torneos sean equilibrados en juegos donde cierto rol tenga ventaja. Esto es: un agente que jugó *blancas* en la partida anterior jugara *negras* en la siguiente.
- En caso de que en alguna partida se dé un empate se considera que ambos agentes han perdido.
- Para medir el rendimiento de las políticas se consideró el total de victorias que los agentes lograron, así como el promedio de los 3 enfrentamientos, la mediana y su desviación estándar.

Los torneos tienen las mismas condiciones antes mencionadas con la siguiente diferencia; en el primer torneo los agentes están limitados a 100 simulaciones del método ABMC por cada turno, en el segundo torneo los agentes están limitados a $1,000$ simulaciones. Estas limitaciones están pensadas para identificar el comportamiento de las políticas cuando se tienen recursos limitados y si el comportamiento se mantiene cuando dichas condiciones cambian.

Para la realización de los torneos se hace uso del framework GGP-Base[1], el cual está escrito

[1] `https://github.com/ggp-org/ggp-base`

en el lenguaje de programación Java. Se elije GGP-Base ya que permite desarrollar agentes GGP, dispone de distintos juegos de tablero y permite la realización de torneos lo que es ideal para el comparativo de las políticas de PBMA.

Otra razón para la elección de GGP-Base, radica en que al estar escrito en Java el cual es un lenguaje de programación orientado a objetos, el cual permite desarrollar un agente *padre* que implemente el método ABMC y desarrollar agentes *hijos* que hereden del agente padre pero que implementen una política de selección propia, en este caso una política del PBMA. Lo anterior garantiza que la única diferencia de rendimiento entre los agentes se deba a la política de selección implementada y no a algún otro factor lo que permite que el comparativo sea lo más justo posible.

3.2. Desarrollo de la política para GGP

Después de realizar el comparativo de las políticas de PBMA, se identificaron aquellas que obtuvieron el mayor número de victorias. Estas políticas se tomaron como referencia para el desarrollo de la política para GGP, la cual se construye considerando la estructura y características de las políticas *ganadoras*.

Para el desarrollo de la política propuesta se utilizó un escenario PBMA que emula un escenario GGP, para ello se tuvieron en cuenta los siguientes lineamientos:

- Se usó una serie de problemas PBMA, en específico los propuestos por Auer et al. [5] los cuales fueron usados para probar el rendimiento de las políticas UCB y UCB-Tuned.
- Se generó una serie de problemas PBMA cuyo número de máquinas está basado en el factor de ramificación de distintos juegos de tablero para dos jugadores. Esta serie de problemas PBMA está pensado para emular un escenario GGP, con lo cual se esperaba determinar el comportamiento que tendría la política propuesta en dicho escenario. Estos problemas PBMA tienen las siguientes características:
    - El número de máquinas de cada PBMA corresponde al factor de ramificación de un juego de tablero.

    - A cada máquina se le asigna una probabilidad aleatoria de dar una recompensa.

- La política propuesta se comparó con las políticas ganadoras en estos escenarios bajo las siguientes condiciones:

    - La política propuesta y las políticas ganadoras se ejecutaron 100 veces en cada PBMA de cada uno de los escenarios.

    - Cada ejecución estuvo limitada a $100,000$ rondas.

    - El rendimiento de las políticas está basado en el porcentaje promedio de veces que se active la máquina óptima en las $100,000$ rondas. Se registró el promedio de veces que se activa la máquina óptima ya que en GGP esto equivaldría a identificar el nodo del árbol de juego que da el mayor número de victorias.

- Este comparativo en estos escenarios se repite hasta que la política propuesta tenga un mayor rendimiento que la política de PBMA que logró el mayor número de victorias en GGP. En caso de que la política propuesta no supere a la política ganadora, se modifica y se repite el comparativo.

3.3. Comparativa de la política propuesta en GGP

Una vez que la política propuesta haya superado a las políticas ganadoras en el escenario PBMA, se identificó su rendimiento en GGP, para ello se realizó un nuevo comparativo.

El comparativo fue idéntico al propuesto en la sección 3.1. En resumen, se realizaron dos torneos en donde la política propuesta se comparó con el resto de las políticas. En cada torneo las políticas se enfrentaron en diversos juegos de tableros 3 veces a 100 partidas cada vez. En un torneo las políticas estaban limitadas a 100 simulaciones de ABMC por turno, en el segundo tornero este límite fue de $1,000$.

3.4. Análisis estadístico de la política propuesta

Una vez finalizado el comparativo anterior se determinó si la política propuesta es adecuada en GGP de acuerdo con los resultados obtenidos. El rendimiento de las políticas se mide de acuerdo con el total, promedio y mediana de victorias que cada una de las políticas logre. De esta manera la política propuesta sera adecuada para GGP si logra tener la mayor cantidad de victorias en la mayoría de los juegos, así como el promedio de victorias más alto.

Además, se realizó un análisis estadístico con las pruebas de Friedman y de Wilcoxon para determinar que la diferencia entre los resultados obtenidos por la política propuesta y las políticas ya establecidas no se deba al azar, así como confirmar que la política propuesta superan las existentes, con lo que se podría afirmar que la política es apta para GGP.

Capítulo 4

Políticas Propuestas

4.1. Comparativa de políticas del PBMA en GGP

Perick et al. [40] llevaron acabó un comparativo empírico entre distintas políticas de seleccion de ABMC para determinar cual tiene el mejor desempeño en el juego para dos jugadores y movimiento simultáneos **Tron**. Las politicas comparadas fueron las siguientes: UCB, UCB-Tuned, UCB-V [3], UCB-Minimal [36], OMC [14], MOSS [2], ϵ-greedy [50], EXP3 [4], Thompson Sampling [13] y PBBM [19]. Debido a las características del juego Tron de ser en tiempo real y de movimientos simultáneos, para el comparativo, Perick et al. tomaron las siguientes condiciones:

Árbol de Juego El método ABMC generalmente es aplicado a juegos con dos jugadores cuyos movimientos los realizan en turnos. Tron es un juego donde los participantes realizan movimientos de manera simultánea, por lo cual Perick et al. modificó a ABMC para tratar los movimientos de ambos jugadores como si de uno se tratasen, para lograr que cada nodo del árbol de juego represente la selección de un par de movimientos en lugar de uno solo como comúnmente se realiza.

Simulaciones Al enfocarse a un único juego, Perick et al. hace uso del conocimiento del domino para mejorar las simulaciones en ABMC y asemejar a agentes más realistas. Comúnmente las

simulaciones se llevan a cabo al realizar movimientos aleatorios, sin embargo, para Tron los movimientos a realizar se deciden en base a probabilidades.

Afinación de Constantes Debido a que varias de las políticas hacen uso de constantes (UCB, UCB-V, UCB-Minimal, ϵ-greedy, Thompson Sampling y EXP3), cuyo valor puede alterar su desempeño, Perick et al. realizó una serie de experimentos para determinar qué valor deben tomar para obtener el máximo rendimiento posible en el juego de Tron. Para cada política se crearon una serie de agentes cada uno con un valor posible para la(s) constante(s), posteriormente estos agentes se enfrentaron a un agente el cual toma el valor de referencia dado en la literatura para la política. Para los enfrentamientos se usó el juego de Tron 20×20, para ABMC el criterio de parada se determinó por tiempo ($100ms$ para un procesador de 2.5Ghz) y la elección de movimiento se determinó por el *nodo hijo robusto* (ver 5). El número de enfrentamientos realizados quedo indeterminado deteniéndose únicamente cuando se lograron $10,000$ enfrentamientos sin empates.

Bajo estos lineamientos Perick et al. comparó las políticas usando las mismas condiciones que se implementaron para afinar las constantes de las políticas. De este comparativo se determinó que la política UCB-Tuned es la política con mejor desempeño ya que es la que mayor porcentaje de victorias presenta y fue la única que logro vencer al resto de políticas. En segundo lugar, se encuentra UCB con un porcentaje de victorias cercano a UCB-Tuned. De estos resultados se concluye que UCB-Tuned es la mejor política para el juego de Tron.

Francisco-Valencia et al. [21] compararon el desempeño de UCB y UCB-Tuned como políticas de selección el ABMC en el ámbito de GGP. Para ello se usaron 3 juegos de tablero para dos jugadores: Breakthrough 6×6, Knightthrough 8×8 y Connect Four 6×8. El comparativo realizado por Francisco-Valencia et al. fue de acuerdo con las siguientes condiciones:

- Se crearon dos agentes GGP basados en ABMC, los cuales implementaron las políticas de selección UCB y UCB-Tuned.
- Se eligió *Por Política* el método para elegir el movimiento a realizar una vez que ABMC termine. Esto es, que se usaron las políticas UCB y UCB-Tuned para elegir el movimiento.

- El comparativo se llevó acabo en dos fases de acuerdo con los siguientes criterios:
  - Por cada juego los agentes se enfrentaron 3 veces a 100 partidas por juego, dando un total de 100 partidas.
  - En la primera fase los agentes fueron limitados a 100 iteraciones del ABMC.
  - En la segunda fase el ABMC se limitó a $1,000$ iteraciones.

Los resultados obtenidos por el comparativo realizado por Francisco-Valencia et al. muestran que un agente basado en ABMC que usa como política de selección UCB-Tuned y con una elección de movimiento basado en política, logra una mayor cantidad de victorias que un agente basado en las mismas características pero en su lugar usa como política de selección a UCB, en especial cuando ABMC se limita a 100 iteraciones, lo que indica que para GGP la política UCB-Tuned tiene un mejor desempeño en este ámbito.

4.1.1. Comparativo

Los trabajos de Perick et al. [40] y Francisco-Valencia et al. [21] sugieren que las políticas UCB-Tuned y UCB son la opción que elegir para desarrollar agentes para General Game Playing. En esta sección se realiza un comparativo similar al de Francisco-Valencia et al. sin embargo, se incrementó el número de juegos, se cambió la elección movimiento a Por Promedio y se agrega la política ETC al comparativo. Para el comparativo se realizaron dos *torneos* donde las políticas se enfrentaron en 8 juegos de tableros para dos jugadores bajo las siguientes consideraciones:

- Se crearon tres agentes implementando ABMC y las políticas UCB, UCB-Tuned y ETC.
- Se hizo uso de los siguientes juegos de tablero para dos jugadores:
  - Atari Go 7×7
  - Breakthrough 6×6
  - Breakthrough with Holes 6×6
  - Breakthrough Suicide 6×6

- Knight-Through 8×8
- Knight Fight 10×10
- Two-Player Free-For-All Zero-Sum
- Sheep and Wolf

Estos juegos se encuentran implementados en el framework GGP-Base [1]

- Por cada juego, los agentes se enfrentarán 3 veces a 100 partidas por vez, para un total de 300 partidas por par de agentes.
- En cada nueva partida los agentes cambiaran de rol, esto es: un agente que jugó *blancas* en la partida anterior jugara *negras* en la siguiente. Esto con la finalidad de que los torneos sean equilibrados en juegos donde cierto rol tenga ventaja.
- En el caso de que en alguna partida se dé un empate se considerará que ambos agentes han perdido (esto aplicara también en las simulaciones del ABMC).
- Para medir el rendimiento de las políticas se tomará el total de victorias que los agentes logren, así como el promedio de los 3 enfrentamientos y su desviación estándar.

Se decidió usar ETC en el comparativo principalmente porque no pertenece a la familia de políticas UCB y UCB-Tuned, y contrario a estas, tiene una etapa fija de exploración después de la cual ETC se vuelve una política de explotación pura al solo escoger el nodo del árbol de juego con el mayor promedio de victorias. Para este comparativo la constante de ETC se estableció en 2, es decir que cada nodo del árbol será visitado dos veces y posteriormente se elegirá aquel con mayor número de victorias con relación al número de visitas.

[1] `http://games.ggp.org/base`

4.1.2. Atari Go 7×7

La tabla 4.1 y figura 4.1 muestran la cantidad de victorias que las políticas lograron en sus enfrentamientos. En dicha tabla se puede observar que la política con mejor rendimiento, en términos de victorias, es ETC ya que en total logra 333 victorias de las 600 partidas posibles, en específico cuando ABMC se limita a 100 simulaciones, esta cantidad de victorias se incrementa a 363 cuando ABMC se limita a $1,000$ simulaciones. En segundo lugar, se encuentra la política UCB la cual obtiene 293 victorias y 295 con ABMC limitado a 100 y $1,000$ simulaciones respectivamente. La política con peor rendimiento es UCB-Tuned ya que de las tres políticas es la que obtiene el menor número de victorias, en específico 274 y 242 con ABMC limitado a 100 y $1,000$ simulaciones respectivamente. Adicionalmente, los rendimientos de las políticas se pueden observar en la figura 4.1 donde se confirma que ETC es la política que logra el mayor número de victorias y se va incrementado cuando se aumentan el número de simulaciones.

	100 Simulaciones			1000 Simulaciones		
Enfrentamientos	UCB	ETC	UCB-Tuned	UCB	ETC	UCB-Tuned
1	106	97	97	91	127	82
2	102	119	79	106	121	73
3	85	117	98	98	115	87
Total	293	**333**	274	295	**363**	242
Promedio	97.67 $\pm$ 11.15	**111 $\pm$ 12.17**	91.33 $\pm$ 10.69	98.33 $\pm$ 7.51	**121 $\pm$ 6.0**	80.67 $\pm$ 7.09

Tabla 4.1: Victorias de UCB, ETC y UCB-Tuned en Atari Go

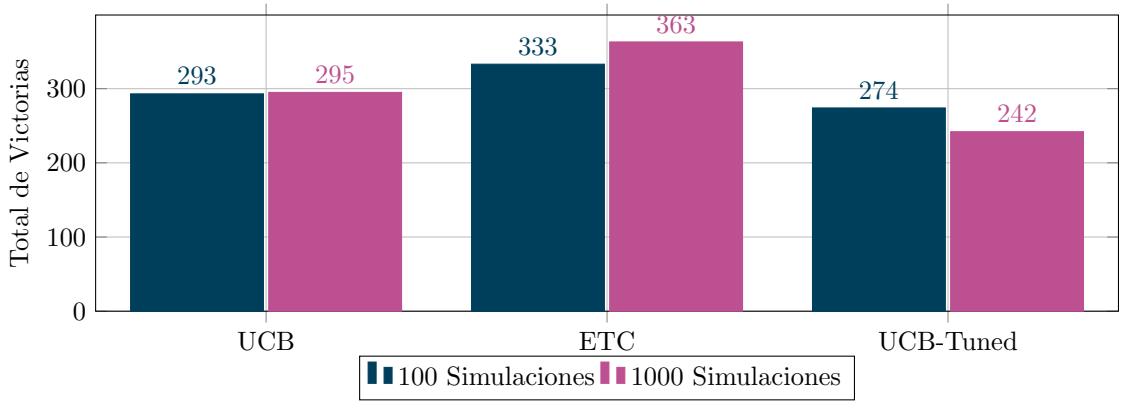


Figura 4.1: Victorias Totales de UCB, ETC y UCB-Tuned en Atari Go

La tabla 4.2 y la figura 4.2 muestran el porcentaje promedio de victorias que cada política obtiene al enfrentarse al resto de políticas a 100 partidas. En esta tabla y figura se observa que ETC es la única política que logra vencer a sus dos contrincantes sin importar el número de simulaciones al que este limitado ABMC. El porcentaje más bajo obtenido por ETC se da cuando se enfrenta a UCB-Tuned con ABMC limitado a 100 simulaciones donde solo logra vencer a este un promedio de 53.67 %, sin embargo, este porcentaje se ve incrementado a un 64.33 % cuando el número de simulaciones aumenta a 1,000. Respecto al encuentro entre ETC y UCB se puede observar que es el primero quien obtiene un mayor porcentaje de victorias con un 57.33 % y 56.67 % cuando ABMC está limitado a 100 y 1,000 simulaciones, respectivamente. Del encuentro entre UCB y UCB-Tuned es UCB quien obtiene el mayor porcentaje de victorias con 55 % en ambos casos de ABMC lo que indica que no hay variaciones si se aumenta el número de simulaciones.

100 Simulaciones			
	UCB	ETC	UCB-Tuned
UCB		42.67 ± 8.96	**55.00 ± 8.19**
ETC	**57.33 ± 8.96**		**53.67 ± 3.51**
UCB-Tuned	45.00 ± 8.19	46.33 ± 3.51	

1000 Simulaciones			
	UCB	ETC	UCB-Tuned
UCB		43.33 ± 4.16	**55.00 ± 3.61**
ETC	**56.67 ± 4.16**		**64.33 ± 6.43**
UCB-Tuned	45.00 ± 3.61	35.67 ± 6.43	

Tabla 4.2: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Atari Go

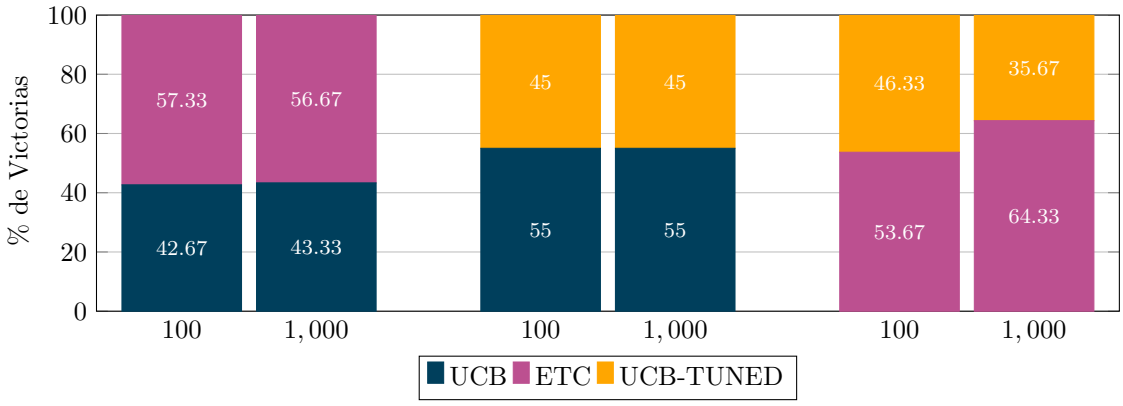


Figura 4.2: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Atari Go

En resumen, los resultados indican que ETC es la política con mejor rendimiento en Atari Go al obtener 363 victorias y vencer a UCB y UCB-Tuned en 56.67 % y 64.33 % partidas respectivamente. Recordando que ETC es una política de explotación con solo una fase inicial de exploración, ya que

en este caso en particular cada nodo solo sera explorado dos veces para posteriormente escoger solo los nodos con mayor número de victorias respecto al número de visitas, lo que va de acuerdo con lo expuesto en [15] que en Go y en su variante Atari Go las políticas enfocadas a explotación son las que tienen mejor rendimiento.

4.1.3. Breakthrough 6×6

Se puede observar en la tabla 4.3 y en la figura 4.3 que cuando se tiene un número bastante limitado de simulaciones es la política ETC quien tiene mejor rendimiento sin embargo, este decae drásticamente conforme el número de simulaciones incrementa pasando de 379 a 205 victorias (ver figura 4.3). Cuando ABMC está limitado a $1,000$ simulaciones es la política UCB-Tuned la cual tiene el mejor rendimiento al lograr el mayor número de victorias.

	100 Simulaciones			1000 Simulaciones		
Enfrentamientos	UCB	ETC	UCB-Tuned	UCB	ETC	UCB-Tuned
1	94	124	82	120	59	121
2	99	124	77	124	57	119
3	84	131	85	100	89	111
Total	277	**379**	244	344	205	**351**
Promedio	92.33 $\pm$ 7.64	**126.33 $\pm$ 4.04**	81.33 $\pm$ 4.04	114.67 $\pm$ 12.86	68.33 $\pm$ 17.93	**117 $\pm$ 5.29**

Tabla 4.3: Victorias de UCB, ETC Y UCB-Tuned en Breakthrough 6×6

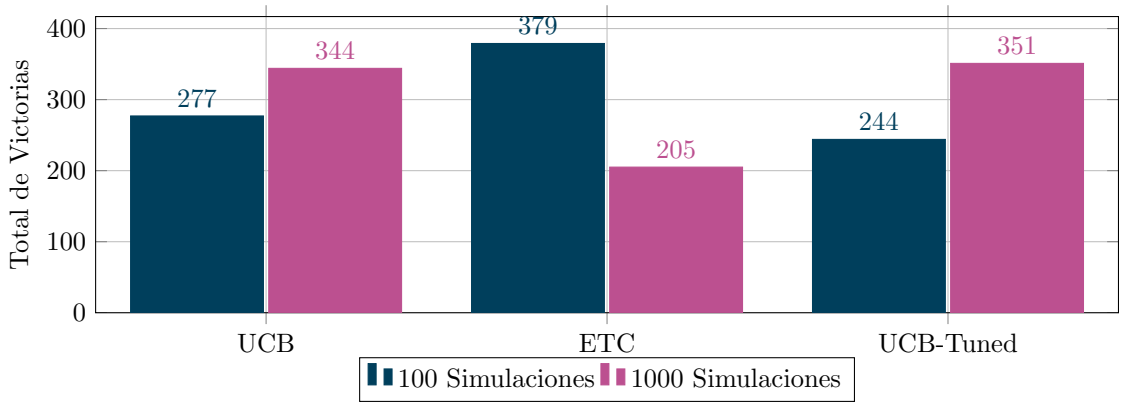


Figura 4.3: Victorias de UCB, ETC Y UCB-Tuned en Breakthrough 6×6

Respecto a los encuentros entre las políticas, de acuerdo a la tabla 4.4, ETC es quien logra vencer tanto a UCB como UCB-Tuned cuando se tiene 100 simulaciones, ya que en 100 partidas ETC

las derrotará él 60.33 % y 66 % respectivamente, sin embargo, cuando el número de simulaciones aumenta a 1,000 ETC solo logra vencer en el 36.33 % y 32 % de las partidas. Con 1,000 simulaciones UCB es la política quien vence tanto a ETC y UCB-Tuned con lo cual tiene el mejor rendimiento, sin embargo, debe observarse que la diferencia entre UCB y UCB-Tuned es de solo el 1 %, también debe observarse que UCB-Tuned es la política con mayor crecimiento en rendimiento ya que como puede observase en la figura 4.4 al enfrentarse a UCB en 100 simulaciones logra vencer un 47.33 % sin embargo esto incrementa a 49 % y para el caso del enfrentamiento con ETC pasa de un 34 % a un 68 %.

En general, el rendimiento de ETC decae conforme el número de simulaciones en ABMC aumenta al contrario de UCB-Tuned cuyo rendimiento aumenta incluso más drásticamente que UCB dando indicios de que probablemente logre vencer a esta política si se aumenta aún más el número de simulaciones, por estas razones y para fines de esta tesis consideramos que UCB-Tuned es quien tiene el mejor rendimiento en Breakthrough 6×6.

100 Simulaciones			
	UCB	ETC	UCB-Tuned
UCB		39.67 ± 3.21	**52.67 ± 5.03**
ETC	**60.33 ± 3.21**		**66.00 ± 1.00**
UCB-Tuned	47.33 ± 5.03	34.00 ± 1.00	

1000 Simulaciones			
	UCB	ETC	UCB-Tuned
UCB		**63.67 ± 9.07**	**51.00 ± 4.58**
ETC	36.33 ± 9.07		32.00 ± 9.85
UCB-Tuned	49.00 ± 4.58	**68.00 ± 9.85**	

Tabla 4.4: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Breakthrough 6×6

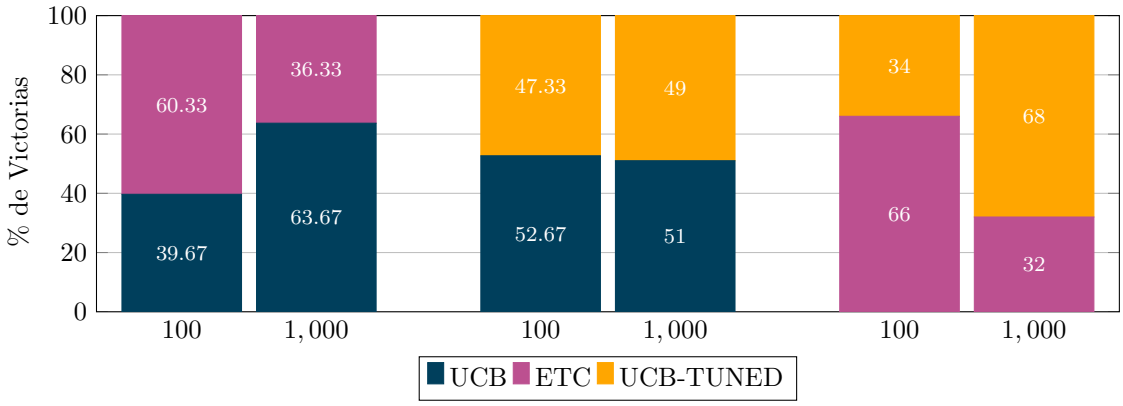


Figura 4.4: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Breakthrough 6×6

4.1.4. Breakthrough with Holes 6×6

De los resultados mostrados en la tabla 4.5 y en la figura 4.5 se puede observar que ETC en 100 simulaciones es quien logra el mayor rendimiento ganando en 321 partidas, sin embargo, en $1,000$ simulaciones solo logra vencer en 194 partidas. UCB-Tuned en 100 simulaciones tiene el peor rendimiento al lograr únicamente en 286 partidas, pero su rendimiento se incrementa cuando el número de simulaciones aumenta a $1,000$ obteniendo el mejor rendimiento al vencer en 358 partidas.

	100 Simulaciones			1000 Simulaciones		
Enfrentamientos	UCB	ETC	UCB-Tuned	UCB	ETC	UCB-Tuned
1	93	109	98	110	67	123
2	102	109	89	122	65	113
3	98	103	99	116	62	122
Total	293	**321**	286	348	194	**358**
Promedio	97.67 $\pm$ 4.51	**107 $\pm$ 3.46**	95.33 $\pm$ 5.51	116 $\pm$ 6.00	64.67 $\pm$ 2.52	**119.33 $\pm$ 5.51**

Tabla 4.5: Victorias de UCB, ETC Y UCB-Tuned en Breakthrough with Holes 6×6

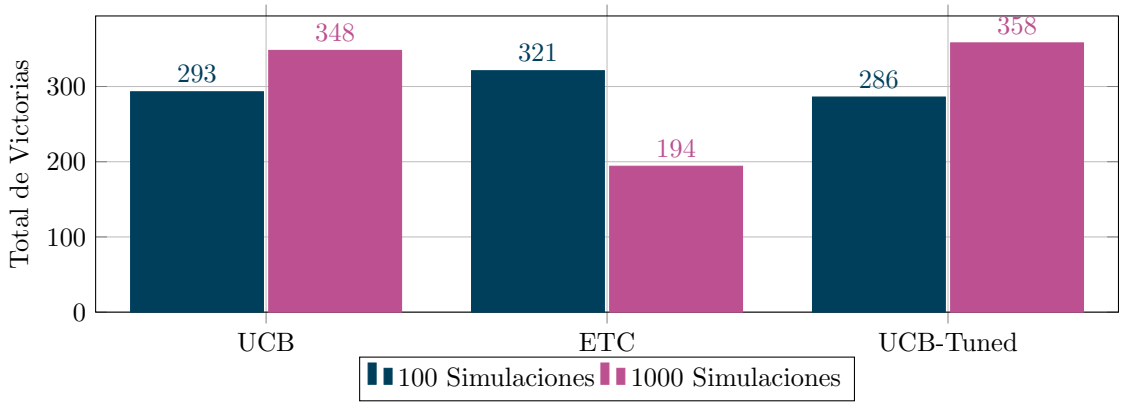


Figura 4.5: Victorias de UCB, ETC Y UCB-Tuned en Breakthrough with Holes 6×6

De la tabla 4.6 y la figura 4.6 se observa que en 100 simulaciones ETC es mejor ya que en promedio logra un mayor porcentaje de victorias que UCB y UCB-Tuned sin embargo, este desempeño decae en $1,000$ simulaciones al no lograr superar a ninguna política. En $1,000$ simulaciones UCB logra vencer tanto a ETC como a UCB-Tuned, sin embargo, a este último solo logra vencerlo por un porcentaje mínimo. El comportamiento presentado por las políticas en Breakthrough with Holes

6×6 es bastante similar al presentado en Breakthrough ya que se observa que en pocas simulaciones ETC es quien tiene el mejor rendimiento, pero conforme el número de simulaciones aumenta UCB y UCB-Tuned presentan mejor rendimiento siendo el segundo con el mayor crecimiento.

100 Simulaciones				1000 Simulaciones			
	UCB	ETC	UCB-Tuned		UCB	ETC	UCB-Tuned
UCB		45.33 ± 4.51	**52.33 ± 0.58**	UCB		**65.33 ± 3.21**	**50.67 ± 3.21**
ETC	**54.67 ± 4.51**		**52.33 ± 5.86**	ETC	34.67 ± 3.21		30.00 ± 4.00
UCB-Tuned	47.67 ± 0.58	47.67 ± 5.86		UCB-Tuned	49.33 ± 3.21	**70.00 ± 4.00**	

Tabla 4.6: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Breakthrough with Holes 6×6

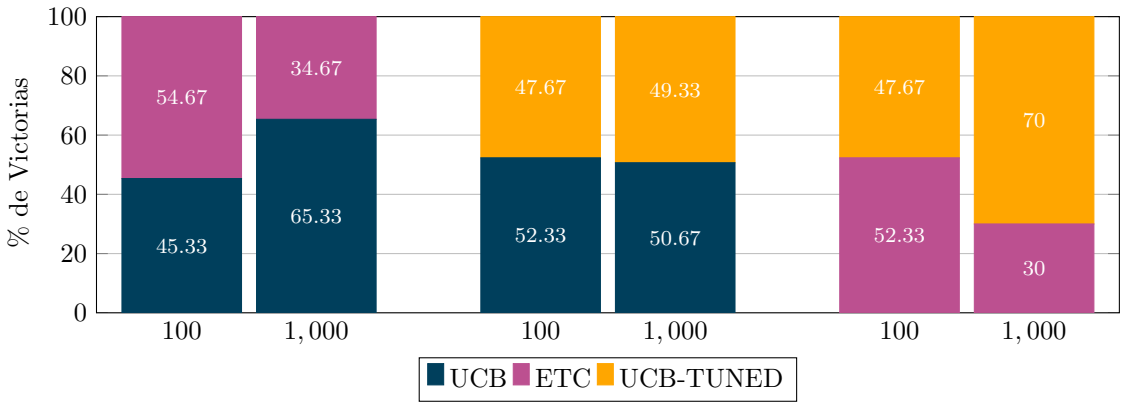


Figura 4.6: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Breakthrough with Holes 6×6

4.1.5. Breakthrough Suicide 6×6

Los resultados (ver tabla 4.7 y la figura 4.7) muestran que la política con mejor rendimiento en Breakthrough Suicide 6×6 es UCB, ya que es la que logra la mayor cantidad de victorias de los tres enfrentamientos por ejemplo cuando ABMC se limita a 100 y $1,000$ simulaciones con 321 y 326 victorias respectivamente. La política con el segundo mejor desempeño es UCB-Tuned cuyo número de victorias supera a ETC pero no a UCB ya que en 100 simulaciones solo obtiene 305 victorias y en $1,000$ simulaciones obtuvo 292. Respecto a ETC como máximo obtiene 282 victorias cuando ABMC se limita a $1,000$ simulaciones con lo cual obtiene el peor rendimiento.

Respecto al porcentaje de victorias que las políticas logran al enfrentarse al resto y el cual puede observarse en la tabla 4.8 y en la figura 4.8, se observa que la política con mejor rendimiento es

	100 Simulaciones			1000 Simulaciones		
Enfrentamientos	UCB	ETC	UCB-Tuned	UCB	ETC	UCB-Tuned
1	109	86	105	114	88	98
2	119	83	98	104	93	103
3	93	105	102	108	101	91
Total	**321**	274	305	**326**	282	292
Promedio	**107** $\pm$ **13.11**	91.33 $\pm$ 11.93	101.67 $\pm$ 3.51	**108.67** $\pm$ **5.03**	94 $\pm$ 6.56	97.33 $\pm$ 6.03

Tabla 4.7: Victorias de UCB, ETC y UCB-Tuned en Breakthrough Suicide 6×6

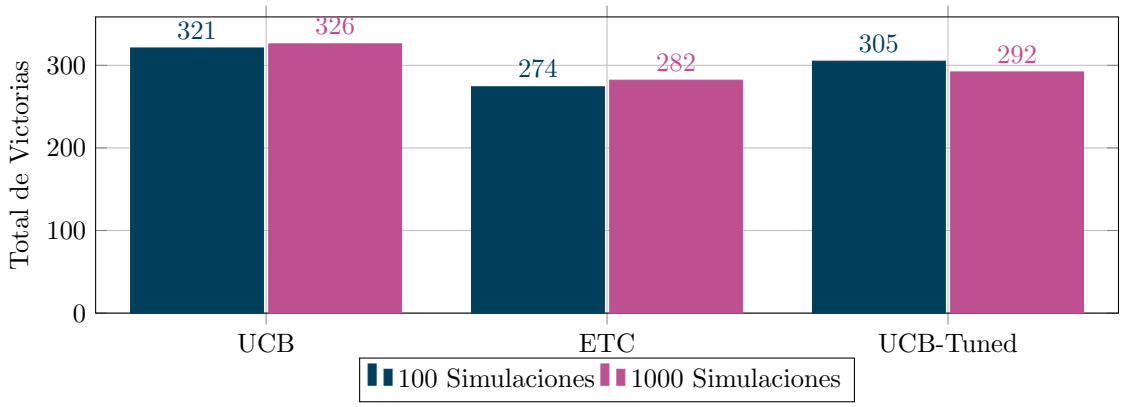


Figura 4.7: Victorias en de UCB, ETC Y UCB-Tuned Breakthrough Suicide 6×6

UCB quien tiene el mejor rendimiento al vencer a sus dos contrincantes teniendo su porcentaje más bajo de partidas ganadas cuando se enfrenta a UCB-Tuned en 100 simulaciones al obtener un 51.33 % y su porcentaje más alto al enfrentarse a ETC en 100 simulaciones con 55.67 %. Respecto a UCB-Tuned solo logra vencer a ETC con lo cual es la segunda política con mayor porcentaje de partidas ganadas con un máximo de 53 %.

	100 Simulaciones				1000 Simulaciones			
	UCB	ETC	UCB-Tuned			UCB	ETC	UCB-Tuned
UCB		**55.67** $\pm$ **8.5**	**51.33** $\pm$ **5.13**		UCB		**55.33** $\pm$ **4.04**	**53.33** $\pm$ **4.73**
ETC	44.33 $\pm$ 8.5		47.00 $\pm$ 3.46		ETC	44.67 $\pm$ 4.04		49.33 $\pm$ 2.52
UCB-Tuned	48.67 $\pm$ 5.13	**53.00** $\pm$ **3.46**			UCB-Tuned	46.67 $\pm$ 4.73	**50.67** $\pm$ **2.52**	

Tabla 4.8: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Breakthrough Suicide 6×6

Por lo cual en el juego Breakthrough Suicide 6×6 la política con mejor rendimiento es UCB al lograr el mayor número de victorias y el porcentaje de partidas ganadas sin importar el número de simulaciones a las que se limite ABMC.

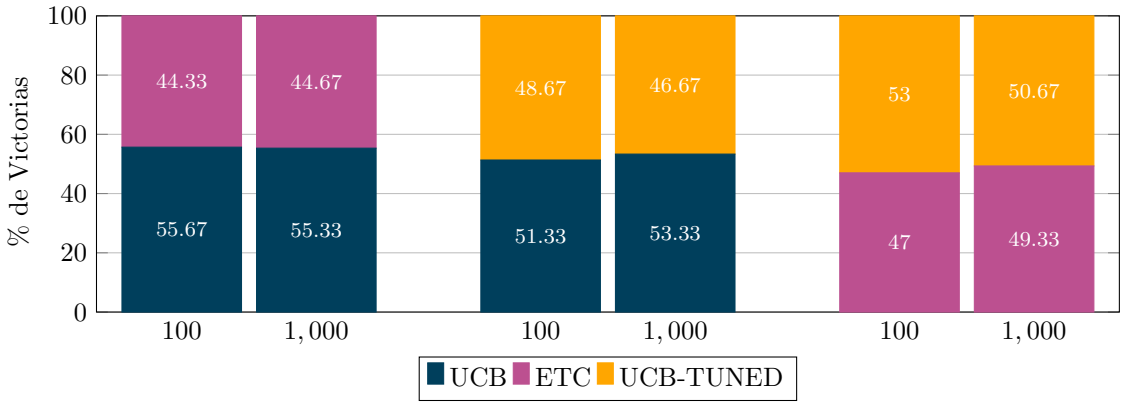


Figura 4.8: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Breakthrough Suicide 6×6

4.1.6. Knight-Through

De acuerdo con la tabla 4.9 junto con la figura 4.9 se observa que ETC tiene el mejor desempeño tanto si ABMC se limita a 100 simulaciones como a $1,000$ simulaciones ya que logra salir victorioso en 316 y 392 partidas respectivamente. ABMC es la segunda política con mejor desempeño cuando se tienen 100 simulaciones al lograr vencer en 296 partidas, pero su desempeño decae a 244 victorias cuando el número de simulaciones se incrementa a $1,000$ donde UCB-Tuned obtiene el segundo lugar al obtener 264 victorias.

	100 Simulaciones			1000 Simulaciones		
Enfrentamientos	UCB	ETC	UCB-Tuned	UCB	ETC	UCB-Tuned
1	106	103	91	76	127	97
2	96	105	99	78	137	85
3	94	108	98	90	128	82
Total	296	**316**	288	244	**392**	264
Promedio	98.67 $\pm$ 6.43	**105.33 $\pm$ 2.52**	96 $\pm$ 4.36	81.33 $\pm$ 7.57	**130.67 $\pm$ 5.51**	88 $\pm$ 7.94

Tabla 4.9: Victorias de UCB, ETC Y UCB-Tuned en Knight-Through

En la tabla 4.10 y en la figura 4.10 se muestra el porcentaje de partidas ganadas por cada política y en las cuales se puede observar cómo ETC vence tanto a ABMC como a UCB-Tuned en especial cuando se tiene $1,000$ simulaciones, al alcanzar el 65.67 % y 65 % respectivamente. En segundo lugar, se encuentra la política UCB-Tuned qué aunque en 100 simulaciones pierde contra ABMC si obtiene un mayor porcentaje cuando se enfrenta a ETC.

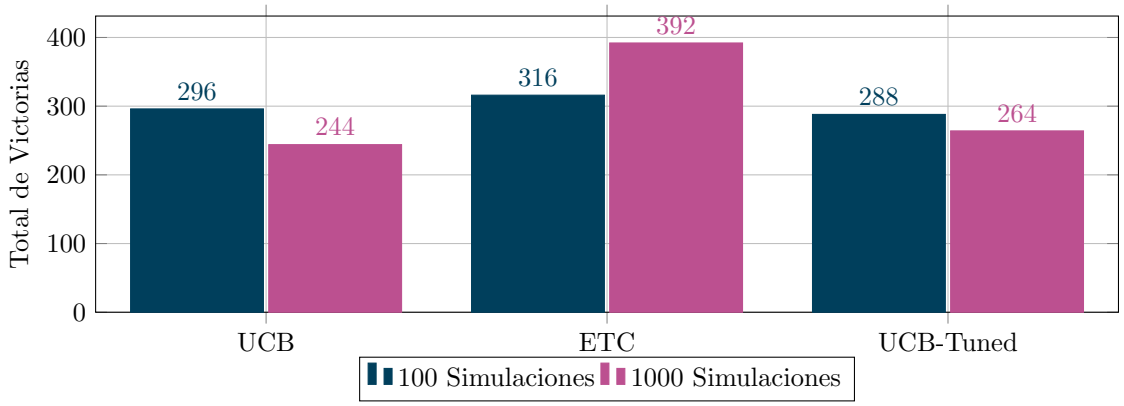


Figura 4.9: Victorias de UCB, ETC Y UCB-Tuned en Knight-Through

De acuerdo con los resultados obtenidos y al igual que en Atari Go, en Knight-Through las políticas basadas en explotación pura son las que mejor desempeño tienen, por lo cual ETC es la mejor política en este juego.

100 Simulaciones			
	UCB	ETC	UCB-Tuned
UCB		45.67 ± 2.52	**53.00 ± 7.00**
ETC	**54.33 ± 2.52**		**51.00 ± 4.36**
UCB-Tuned	47.00 ± 7.00	49.00 ± 4.36	

1000 Simulaciones			
	UCB	ETC	UCB-Tuned
UCB		34.33 ± 3.51	47.00 ± 10.82
ETC	**65.67 ± 3.51**		**65.00 ± 6.00**
UCB-Tuned	**53.00 ± 10.82**	35.00 ± 6.00	

Tabla 4.10: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Knight-Through

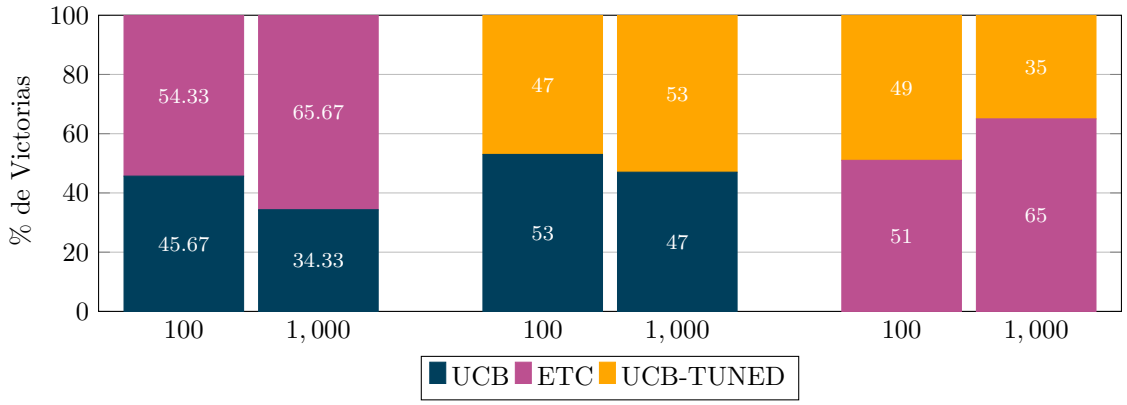


Figura 4.10: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Knight-Through

4.1.7. Sheep and Wolf

En los resultados mostrados en la tabla 4.11 y figura 4.11 se observa que la política con mayor número de victorias es UCB-Tuned sin importar si ABMC este limitado a 100 o 1,000 simulaciones ya que obtiene 317 y 332 victorias respectivamente. En segundo lugar, se encuentra UCB con un desempeño similar a UCB-Tuned pero inferior al obtener 8 victorias menos que UCB-Tuned en 100 simulaciones y 7 victorias menos que UCB-Tuned en 1,000 simulaciones. ETC es la política con peor desempeño logrando un máximo de 274 victorias cuando se tienen 100 simulaciones, 43 menos que UCB-Tuned y su mínimo de 243 victorias en 1,000 simulaciones, 89 victorias menos que UCB-Tuned.

	100 Simulaciones			1000 Simulaciones		
Enfrentamientos	UCB	ETC	UCB-Tuned	UCB	ETC	UCB-Tuned
1	105	91	104	109	81	110
2	101	90	109	109	79	112
3	103	93	104	107	83	110
Total	309	274	**317**	325	243	**332**
Promedio	103.00 $\pm$ 2.00	91.33 $\pm$ 1.53	**105.67 $\pm$ 2.89**	108.33 $\pm$ 1.15	81.00 $\pm$ 2.00	**110.67 $\pm$ 1.15**

Tabla 4.11: Victorias de UCB, ETC Y UCB-Tuned en Sheep and Wolf

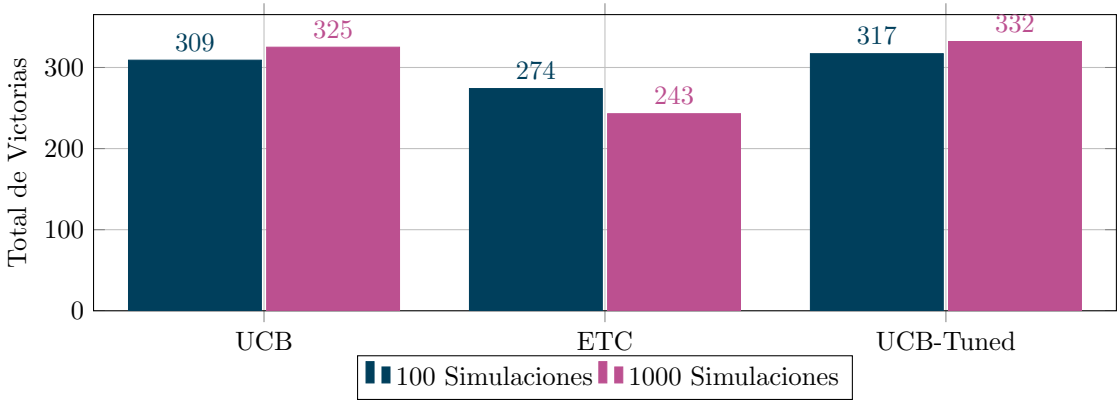


Figura 4.11: Victorias de UCB, ETC Y UCB-Tuned en Sheep and Wolf

La tabla 4.12 y la figura 4.12 muestran los porcentajes de partidas que cada política logra ganar, donde se observa que ETC no logra vencer ni a UCB ni a UCB-Tuned cuando se enfrenta a ellas ya que su máximo porcentaje de partidas ganadas es de 46.33 % cuando se enfrenta a UCB-Tuned

en 100 simulaciones. UCB-Tuned es quien logra vencer a sus dos contrincantes: ETC y a UCB, con 53.67 % y 52 % respectivamente cuando se tiene 100 simulaciones y 58.67 % y 52 % cuando se tienen 1,000, se puede observar que la diferencia entre victorias de UCB y UCB-Tuned se mantiene en 2 % tanto en 100 y 1,000 simulaciones lo que indica que, aunque el rendimiento de las dos políticas es bastante similar, UCB-Tuned es superior.

Por lo tanto, para el juego Sheep and Wolf la política con mejor desempeño es UCB-Tuned al obtener el mayor número de victorias, así como obtener un porcentaje de partidas ganadas mayor que ETC y UCB.

100 Simulaciones				1000 Simulaciones			
	UCB	ETC	UCB-Tuned		UCB	ETC	UCB-Tuned
UCB		**55.00 ± 1.00**	48.00 ± 1.73	UCB		**60.33 ± 0.58**	48.00 ± 1.00
ETC	45.00 ± 1.00		46.33 ± 1.15	ETC	39.67 ± 0.58		41.33 ± 1.53
UCB-Tuned	**52.00 ± 1.73**	**53.67 ± 1.15**		UCB-Tuned	**52.00 ± 1.00**	**58.67 ± 1.53**	

Tabla 4.12: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Sheep and Wolf

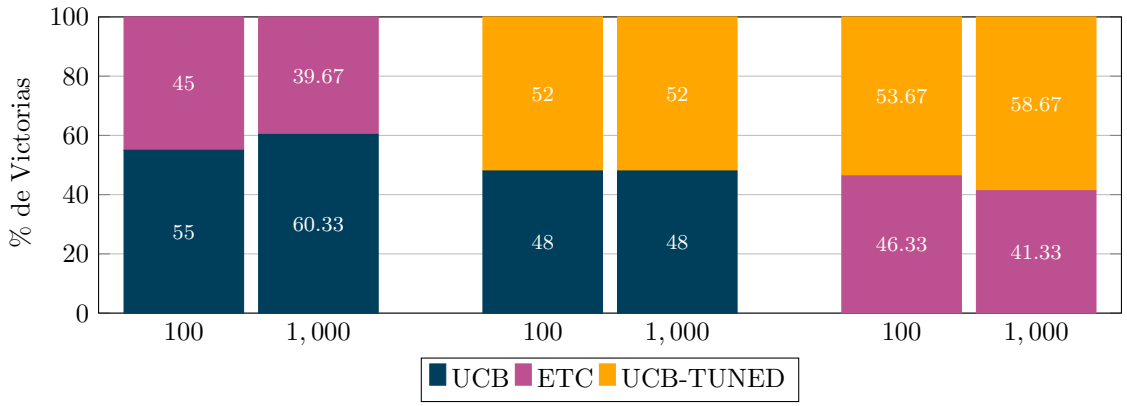


Figura 4.12: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Sheep and Wolf

4.1.8. Tic-Tac-Toe Large

La tabla 4.13 y figura 4.13 muestran que en 100 simulaciones la política con mayor número de victorias es UCB al obtener 308 victorias con respeto al resto de políticas. En segundo lugar, se encuentra UCB-Tuned con 20 victorias menos que UCB. En tercer lugar y con el peor desempeño de las tres, se encuentra la política ETC al lograr solo 273 victorias. En 1,000 simulaciones la diferencia

en el desempeño de las políticas se mantiene similar a 100 simulaciones al tener en primer lugar a UCB con 400 victorias, seguida por UCB-Tuned con 378 victorias y en tercer lugar ETC con solo 104, se puede observar que el rendimiento de ETC disminuye al incrementar el número de victorias.

	100 Simulaciones			1000 Simulaciones		
Enfrentamientos	UCB	ETC	UCB-Tuned	UCB	ETC	UCB-Tuned
1	98	99	93	135	34	120
2	95	93	102	137	32	130
3	115	81	93	128	38	128
Total	**308**	273	288	**400**	104	378
Promedio	**102.67 ± 10.79**	91.00 ± 9.17	96.00 ± 5.20	**133.33 ± 4.73**	34.67 ± 3.06	126.00 ± 5.29

Tabla 4.13: Victorias de UCB, ETC Y UCB-Tuned en Tic-Tac-Toe Large

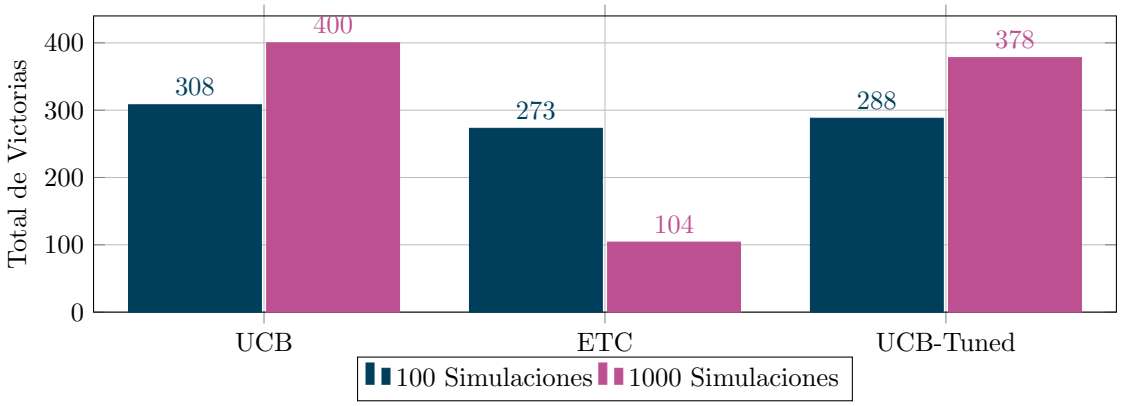


Figura 4.13: Victorias de UCB, ETC Y UCB-Tuned en Tic-Tac-Toe Large

De la tabla 4.14 y la figura 4.14 se observa que tanto en 100 simulaciones y 1,000 simulaciones UCB obtiene el mayor porcentaje de victorias que sus contrincantes. UCB-Tuned solo obtiene un porcentaje mayor de victorias al enfrentarse a ETC, ya que se ve superado en 57.67 % y 2 % al enfrentarse a UCB en 100 y 1,000 simulaciones respectivamente. ETC no logra vencer ni a UCB y UCB-Tuned en especial cuando se tiene 1,000 simulaciones siendo su máximo porcentaje de victorias de 19 % al enfrentarse a UCB-Tuned.

Respecto al porcentaje de empates, el mayor número se da en el enfrentamiento entre UCB y ETC en 100 simulaciones al ser del 5 % sin embargo en 1,000 simulaciones este porcentaje se reduce al 1.33 %. En el enfrentamiento de entre UCB y UCB-Tuned el porcentaje de empates se reduce conforme el número de simulaciones incrementa quedando en 1.34 %. Respecto al enfrentamiento

de ETC y UCB-Tuned el porcentaje de empates incrementan en 0.33 % cuando se incrementa el número de simulaciones, sin embargo, el porcentaje de victorias de UCB-Tuned se incrementa drásticamente.

En general se puede observar que UCB es la política con mejor desempeño en Tic-Tac-Toe Large seguida muy de cerca por la política UCB-Tuned.

Sheep and Wolf							
100 Simulaciones				1000 Simulaciones			
	UCB	ETC	UCB-Tuned		UCB	ETC	UCB-Tuned
UCB		**51.00 ± 7.81**	**51.67 ± 3.06**	UCB		**83.00 ± 5.00**	**50.33 ± 1.53**
ETC	44.00 ± 6.08		47.00 ± 3.61	ETC	15.67 ± 4.73		19.00 ± 1.73
UCB-Tuned	46.00 ± 4.36	**50.00 ± 4.36**		UCB-Tuned	48.33 ± 2.08	**77.67 ± 3.21**	

Tabla 4.14: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Tic-Tac-Toe Large

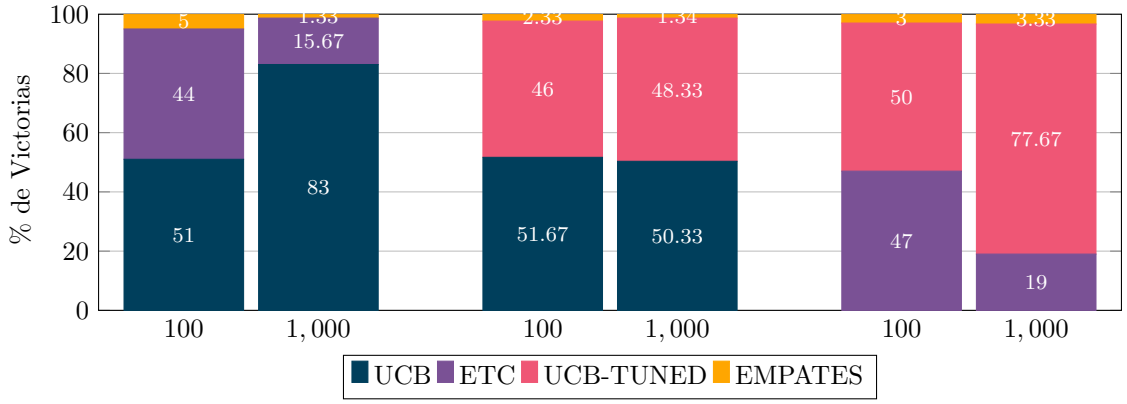


Figura 4.14: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Tic-Tac-Toe Large

4.1.9. Two-Player Free-For-All Zero-Sum

De acuerdo con la tabla 4.15 y la figura 4.15 que muestra los resultados en el juego de Two-Player Free-For-All Zero-Sum, ETC tiene el mejor desempeño cuando se tiene pocas simulaciones ya que obtuvo 27 y 19 victorias más que UCB y UCB-Tuned respectivamente. En segundo lugar, esta UCB-Tuned que logro obtener 8 victorias más que UCB. En 1000 simulaciones el desempeño de ETC decae al solo obtener 208 victorias. En primer lugar, se encuentra UCB-Tuned el cual logro 348 victorias, aunque muy seguido de cerca por UCB quién logro 344 con lo cual podría decirse que

el rendimiento de ambas políticas es bastante similar.

	100 Simulaciones			1000 Simulaciones		
Enfrentamientos	UCB	ETC	UCB-Tuned	UCB	ETC	UCB-Tuned
1	100	104	94	114	64	122
2	96	105	97	115	72	113
3	91	105	104	115	72	113
Total	287	**314**	295	344	208	**348**
Promedio	95.67 $\pm$ 4.51	**104.67 $\pm$ 0.58**	98.33 $\pm$ 5.13	114.67 $\pm$ 0.58	69.33 $\pm$ 4.62	**116 $\pm$ 5.2**

Tabla 4.15: Victorias de UCB, ETC Y UCB-Tuned en Two-Player Free-For-All Zero-Sum

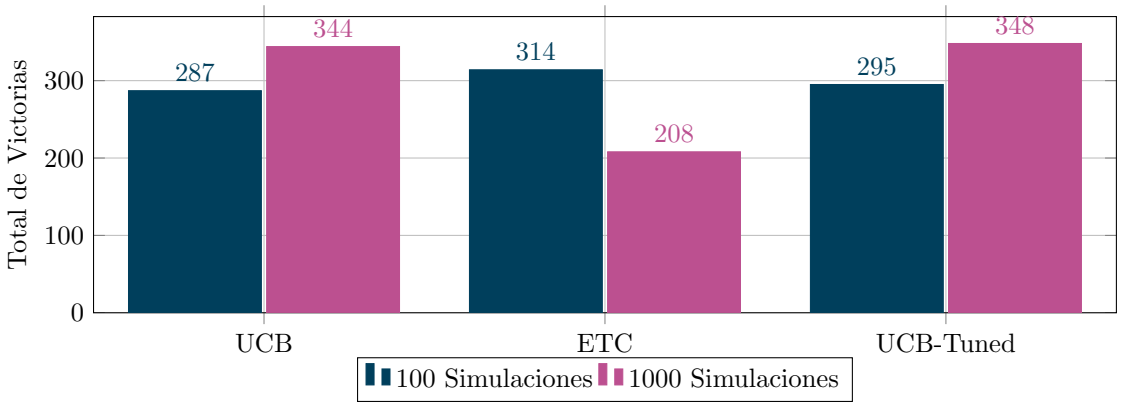


Figura 4.15: Victorias de UCB, ETC Y UCB-Tuned en Two-Player Free-For-All Zero-Sum

Para el caso de porcentaje de victorias en 100 simulaciones ETC tiene el mejor desempeño a vencer a UCB y UCB-Tuned con el 52.67 % y 52 % de las partidas respectivamente (ver figura 4.16), debe observase que de acuerdos a los resultados mostrados en la tabla 4.16 ETC y UCB empatan en el 0.33 % partidas y de igual forma ETC y UCB-Tuned también empatan en 0.33 % partidas. En 1, 000 simulaciones ETC decae en rendimiento al obtener un porcentaje por debajo de los obtenidos por UCB y UCB-Tuned siendo su máximo de 35.67 % al enfrentar a UCB. Aunque UCB vence a UCB-Tuned su rendimiento es bastante similar ya que la diferencia del porcentaje de victorias ganadas por ambas políticas es de solo 0.66 %, lo que parece indicar que ambas políticas presentan un buen rendimiento en Two-Player Free-For-All Zero-Sum pero es UCB-Tuned es quien tiene mayor crecimiento cuando se incrementan el número de simulaciones.

100 Simulaciones				1000 Simulaciones			
	UCB	ETC	UCB-Tuned		UCB	ETC	UCB-Tuned
UCB		47.00 ± 2.65	48.67 ± 2.89	UCB		**64.33 ± 3.06**	**50.33 ± 3.51**
ETC	**52.67 ± 2.89**		**52.00 ± 2.65**	ETC	35.67 ± 3.06		33.67 ± 3.06
UCB-Tuned	**50.67 ± 2.52**	47.67 ± 2.89		UCB-Tuned	49.67 ± 3.51	**66.33 ± 3.06**	

Tabla 4.16: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Two-Player Free-For-All Zero-Sum

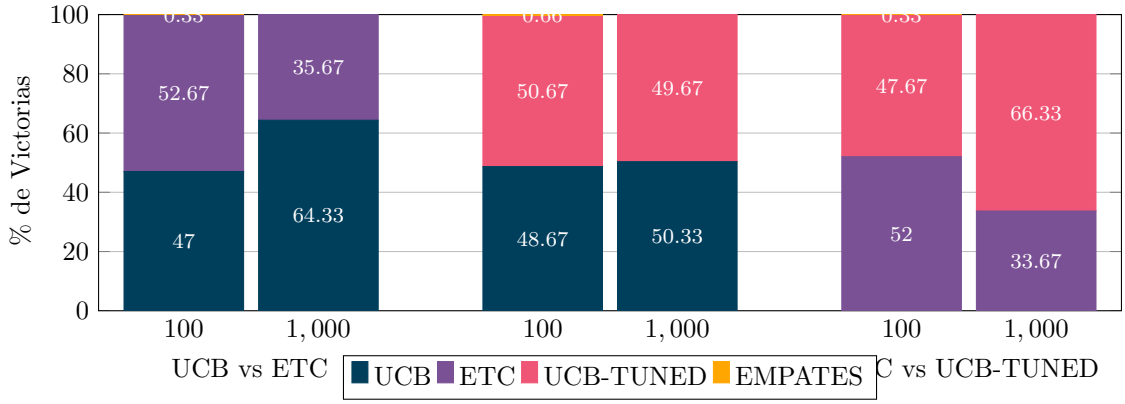


Figura 4.16: Porcentaje de Victorias de UCB, ETC Y UCB-Tuned en Two-Player Free-For-All Zero-Sum

4.1.10. Conclusiones del Comparativo

La tabla 4.17 resume los resultados de comparar las políticas ETC, UCB y UCB-Tuned en los 8 juegos elegidos. En esta tabla se muestran el número de victorias que cada política obtuvo en cada juego y la suma total de victorias en todos los juegos, así como el promedio, desviación estándar y mediana.

Tomando únicamente el total de victorias obtenidas como parámetro para determinar la política con mejor desempeño, se determina que cuando ABMC está limitado a 100 simulaciones, ETC es la mejor política ya que obtiene 2,484 victorias en total, 100 victorias más que UCB y 187 más que UCB-Tuned (ver figura 4.17). Cuando ABMC está limitado a 1,000, UCB es la mejor política ya que obtiene 2,626 victorias, 61 victorias más que UCB-Tuned y 635 victorias sobre ETC (ver figura 4.17).

Tomando el promedio de victorias como parámetro para medir el desempeño, se puede observar que en 100 simulaciones ETC presenta el mayor promedio con 310.5 victorias, en segundo lugar,

	100 Simulaciones			1,000 Simulaciones		
	UCB	ETC	UCB-Tuned	UCB	ETC	UCB-Tuned
Atari-Go	293	**333**	274	295	**363**	242
Breakthrough Small	277	**379**	244	344	205	**351**
Breakthrough Small with Holes	293	**321**	286	348	194	**358**
Breakthrough Small Suicide	**321**	274	305	**326**	282	292
Knight-Through	296	**316**	288	244	**392**	264
Sheep and Wolf	309	274	**317**	325	243	**332**
Tic-Tac-Toe Large	**308**	273	288	**400**	104	378
Two-Player Free-For-All Zero-Sum	287	**314**	295	344	208	**348**
Total	2384	**2484**	2297	**2626**	1991	2565
Promedio	298.00 ± 13.97	**310.50 ± 36.69**	287.13 ± 21.74	**328.25 ± 45.12**	248.88 ± 94.29	320.63 ± 48.84
Mediana	294.5	**315**	288	335	225.5	**340**

Tabla 4.17: Total de Victorias de UCB, ETC Y UCB-Tuned en 8 juegos de tablero

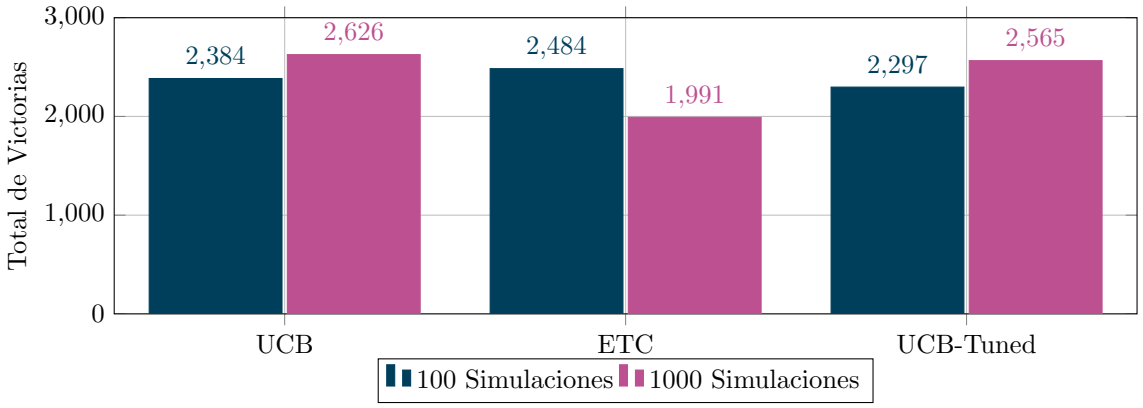


Figura 4.17: Total de Victorias de UCB, ETC Y UCB-Tuned en 8 juegos de tablero

esta UCB con 294.5 victorias y en tercer lugar UCB-Tuned con 288 victorias. Para el caso de 1,000 simulaciones UCB presenta el promedio de victorias más alto con 328.25, seguido muy de cerca por UCB-Tuned con 320.63 victorias y en tercer lugar se encuentra ETC con 248.88 victorias.

La figura 4.18 muestra qué tan dispersas están las victorias que se obtuvieron con las políticas UCB, ETC y UCB-Tuned en 100 simulaciones. En la figura 4.18 se puede observar que ETC tiene el mayor rango de valores, sin embargo, es necesario observar que la mediana de ETC es mayor que casi el 100 % de los valores de UCB-Tuned y más grande que el 75 % de los valores de UCB lo que confirma que ETC tiene el mejor desempeño cuando ABMC está limitado a 100 simulaciones.

Se puede observar un diagrama de cajas para las politicas UCB, ETC y UCB-Tuned cuando ABMC está limitado a 1,000 simulaciones en la figura 4.19. En este diagrama se observa que ETC

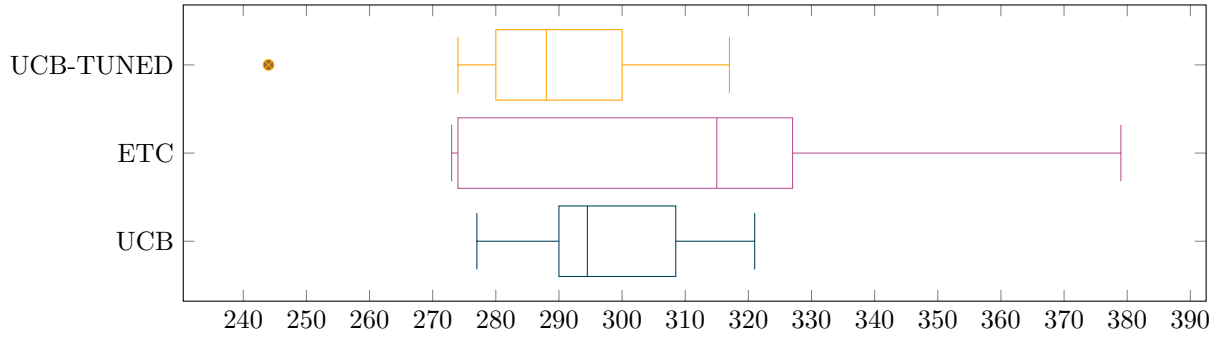


Figura 4.18: Gráfica de cajas de UCB, ETC y UCB-Tuned en 100 simulaciones

presenta el mayor rango de valores y la mediana más baja por lo cual el rendimiento de ETC es el inferior en $1,000$ simulaciones. El rango más bajo es el presentado por UCB-Tuned cuya mediana es ligeramente superior a la de UCB, sin embargo, debe observarse que el valor máximo de UCB-Tuned es inferior al de UCB. De este diagrama se puede inferir que tanto UCB y UCB-Tuned tienen un rendimiento ligeramente similar.

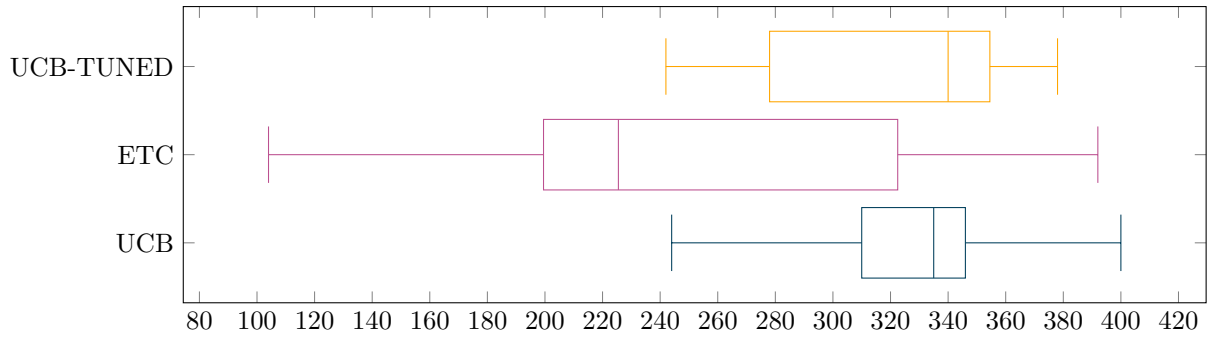


Figura 4.19: Gráfica de cajas de UCB, ETC y UCB-Tuned en $1,000$ simulaciones

En conclusión, como resultado de estos comparativos se determinó que la política con mejor desempeño en 100 simulaciones es ETC ya que obtiene el mayor número de victorias, así como el mayor promedio y mediana. En $1,000$ simulaciones la política con mejor desempeño es UCB al obtener el mayor número de victorias y promedio. Estas dos políticas ETC y UCB serán las que se utilizarán para desarrollar una política de selección propia de GGP.

4.2. Desarrollo de las políticas para GGP

Del comparativo previo se observó que la política ETC es la adecuada cuando se tiene pocas simulaciones, sin embargo, cuando al método ABMC se le permite un mayor número de simulaciones UCB presenta un mejor desempeño seguida muy de cerca por UCB-Tuned.

Debe recordarse que ETC es una política de explotación que tiene una fase inicial y fija de exploración, caso contrario de UCB que realiza un balance entre la explotación y exploración que depende del valor de α y que afecta a su rendimiento (ver sección 1.3.4), por lo cual si el valor de dicha α es 0 o un valor muy pequeño, UCB se comportará como una política de explotación similar a ETC. Por lo anterior surge la idea de variar el valor de α en UCB para que se comporte como una política de exploración pura cuando sea necesario, algo similar a lo que realiza UCB-Tuned donde el valor de α es remplazada por una función de minimización que depende de la varianza muestral de las recompensas y cuyo valor máximo es $1/4$.

Otro punto para considerar es el factor de ramificación del árbol de juego el cual depende de la complejidad del juego a tratar. Por ejemplo, tomemos el juego del Gato, en un inicio el árbol de juego en el primer nivel tendrá 9 nodos, los cuales tendrán 8 nodos hijos y estos a su vez tendrán 7 nodos hijos y estos tendrán 6 hijos y así sucesivamente. Al ejecutar ABMC y UCB en este árbol, para determinar qué movimiento debe realizar el jugador, la política se enfrentará en el primer nivel a un PBMA de $K = 9$ máquinas (nodos), después se enfrentará a un PBMA de $K = 8$ máquinas en el segundo nivel, a otro de $K = 3$ máquinas y así de manera sucesiva conforme se desciende en el árbol.

Como se ha mencionado el rendimiento de UCB depende del valor elegido para α, por lo cual es posible que cierto valor pudiera funcionar para ciertos valores de K pero no para otros valores, dicho esto, en el árbol de juego UCB podría tener un buen rendimiento en ciertos niveles, pero quizás en otros niveles su rendimiento podría ser bajo. De lo anterior se puede plantear una modificación a la política UCB la cual en lugar de tener un valor fijo de α será sustituida por una función que dependa del valor de K tal como se muestra en el algoritmo 14, de esta manera el rendimiento de UCB se adaptará al valor de K.

Algoritmo 14: Upper Confidence Bound

Entrada: K máquinas y T Rondas

Activar cada máquina una sola vez;

para $t \leftarrow K+1$ **a** T **hacer**

  Activar la máquina i que maximice;

$$\bar{x}_{i,T_i} + \sqrt{\frac{\alpha\left(K\right)\log t}{T_i}}$$

4.2.1. Estimación de $\alpha\left(K\right)$

Para estimar el $\alpha\left(K\right)$ adecuado en GGP se realizaron una serie de experimentos con diferentes valores de K y de α para ello se utilizaron los algoritmos 15 y 16.

La idea base del algoritmo 15 es generar un problema de bandido con K máquinas y probar UCB con distintos valores de α para posteriormente determinar cuál es el que presenta mejor rendimiento. Se elijen los valores para K entre 2 y 100 ya que se consideró que estos valores son suficientes para determinar la existencia de una relación entre el número de máquinas y los valores de α que repercute en el rendimiento de UCB.

Algoritmo 15: Estimación de UCB usando Intervalos

Entrada: $N \in (2, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)$

$A \leftarrow$ conjunto de valores de acuerdo al algoritmo 16 con los siguientes valores 0.5, 0.0 y 10 como entrada;

para cada $K \in N$ **hacer**

  para cada $\alpha \in A$ **hacer**

    para 1 **a** $1,000$ **hacer**

      $m \leftarrow$ un problema del bandido multi-armado de K máquinas;

      ejecutar UCB en m, 100 veces usando el valor de α limitado a $100,000$ rondas y se obtiene el arrepentimiento acumulado promedio;

    se registra el arrepentimiento global de los $1,000$ problemas del bandido multi-armado

Para determinar los valores de α a probar con cada problema del bandido, se hace uso del algoritmo 16, el cual básicamente genera n números dentro de un intervalo, para este caso se generan 10 números en el intervalo de $[0, 0.5]$, estos valores son tomados en base a UCB-Tuned

donde el valor máximo de α es 0.25 y el mínimo posible 0. Se debe observar que los valores dados por el algoritmo 16 son generados de manera logarítmica esto es para encontrar el valor α lo más adecuado y ajustado posible.

Algoritmo 16: Generador de Intervalos para α

Entrada: h Límite Superior
Entrada: l Límite Inferior
Entrada: c Cantidad de valores a generar, límite inferior y superior incluidos
Salida: $A \leftarrow$ conjunto de valores

```
1 add h and l to A;
2 para n ← 2 a c hacer
3     t ← (h−l)/2;
4     add t a A;
5     h ← t;
```

De los experimentos realizados se obtuvieron los valores mostrados en la tabla 4.18. En dicha tabla se resalta el arrepentimiento mínimo que se encontró para cada valor de K. La figura 4.20 muestra el arrepentimiento de UCB para los distintos valores de α, en el eje horizontal aparece el número de máquinas K y en el eje vertical el valor de α y los puntos muestran el valor de α que logra el menor arrepentimiento.

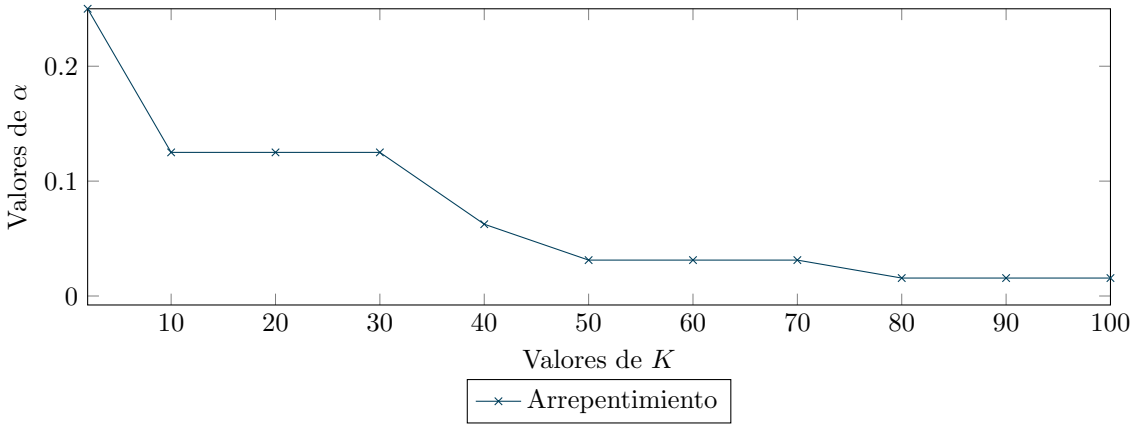


Figura 4.20: Valores de α con menor arrepentimiento de acuerdo al número de máquinas

Es claramente visible, tanto en la tabla como en la figura, que el valor empírico de α que usa en problemas de bandido con máquinas con distribuciones de Bennoulli tiende a disminuir a medida

α	$K = 2$	$K = 10$	$K = 20$	$K = 30$
0.5	32.87599291	209.433417	423.780227	630.8991716
0.25	**18.25833432**	116.3316074	234.4191165	350.0860386
0.125	31.07831557	**100.3666093**	**141.7456925**	**198.039642**
0.0625	161.4920474	355.7119161	247.7846574	211.5564137
0.03125	636.3839619	953.3177813	425.8911306	280.8125353
0.015625	1281.800731	1375.111881	655.3645169	429.7325413
0.0078125	1803.720134	1784.99521	889.4018378	577.6450387
0.00390625	2278.344067	2108.183699	1061.458508	690.2367994
0.001953125	2749.906634	2245.065439	1174.408295	809.2446594
0.0	5369.546394	2620.839196	1507.323117	1044.84556
α	$K = 40$	$K = 50$	$K = 60$	$K = 70$
0.5	845.3930708	1049.563145	1250.438559	1450.08613
0.25	466.4261567	577.2316381	693.2197042	805.4396666
0.125	260.9195798	326.8615694	388.4936043	450.8379373
0.0625	**199.2319155**	210.0140709	233.0756098	260.0195909
0.03125	212.1632531	**194.9900322**	**196.0247022**	**197.1115065**
0.015625	292.9511996	243.9314031	224.7826637	208.8158847
0.0078125	396.9141287	336.145405	290.6328965	247.1634022
0.00390625	523.3516516	413.8622618	368.051293	329.898228
0.001953125	616.1031126	506.9338643	433.0838539	375.4479823
0.0	825.5027413	664.8322039	585.4101464	535.919935
α	$K = 80$	$K = 90$	$K = 100$	
0.5	1649.434009	1838.99005	2035.291679	
0.25	917.4059381	1021.637086	1132.683732	
0.125	513.5874116	580.1546486	643.1948647	
0.0625	291.4136499	320.3191704	349.0558798	
0.03125	**208.3208038**	222.3790973	237.7055589	
0.015625	202.4926674	**198.2595873**	**202.0549332**	
0.0078125	241.2048555	222.9236386	214.0053779	
0.00390625	293.981931	263.759107	250.5537191	
0.001953125	336.6977931	309.7463854	296.9842329	
0.0	471.9239373	437.4116567	418.3437997	

Tabla 4.18: Arrepentimiento para distintos valores de α y número de máquinas

que aumenta el número de máquinas. Un claro ejemplo es con los valores de $K = 2$ y $K = 100$, donde $\alpha = 0.25$ obtiene el mejor arrepentimiento para el primero, pero presenta uno de los peores arrepentimientos para el segundo.

En la figura 4.20 se puede observar que no hay diferencia entre los valores de α para ciertos valores de K, debido a esto y con la finalidad de afinar los valores de α se repitió el mismo procedimiento pero esta vez se alimenta el algoritmo 16 con el valor superior e inferior de los mejores valores de α que se obtuvieron en el procedimiento anterior, por ejemplo para $K = 2$ se utilizaron $\alpha = 0.5$ y $\alpha = 0.125$, y de esta manera para el resto de valores de K.

De manera similar al proceso anterior, los valores de α que producen el menor arrepentimiento para cada valor de K de acuerdo con la tabla 4.19 fueron graficados y se muestran en la figura 4.21. Se puede observar en dicha figura que se confirma el mismo fenómeno de que; α tiende acercarse a 0 conforme K incrementa, lo que indica que a mayor número de máquinas una política debe tender a explotar más que explorar ya que al acercar α a 0 en UCB el término de exploración dejará de influir (ver sección 1.3.4).

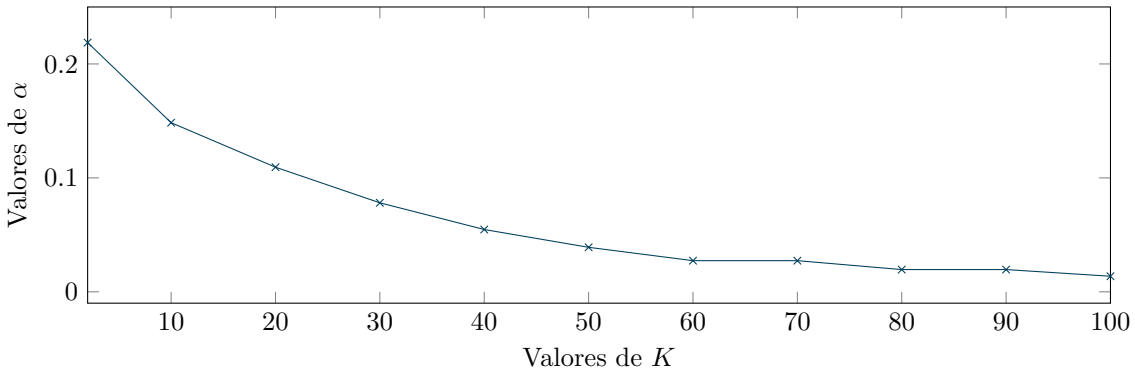


Figura 4.21: Arrepentimiento para distintos valores ajustados de α y número de máquinas

Con la finalidad de comprobar que los valores para α obtenidos por el algoritmo de intervalos 15 no sean mínimos locales se hizo uso del algoritmo de recocido simulado 1.4 bajo las siguientes condiciones:

- La temperatura inicial se establece en 100 la cual en experimentos previos comprobó dar los mejores resultados.

α	$K = 2$	$K = 10$	α	$K = 20$
0.5	32.29595484	208.4764212	0.25	236.129853
0.3125	20.97042004	139.172423	0.15625	156.6299361
0.21875	**17.61600766**	104.2993211	0.109375	**137.6470701**
0.171875	19.06865144	91.28716343	0.0859375	151.206586
0.1484375	22.78609835	**87.86604227**	0.07421875	180.317461
0.13671875	28.8911964	94.73105065	0.068359375	213.1020488
0.130859375	28.56183449	92.85356597	0.065429688	225.2814384
0.127929688	27.2924094	100.2050342	0.063964844	232.9142235
0.126464844	30.59359036	98.29225507	0.063232422	232.5736422
0.125	31.09418382	99.728747	0.0625	247.9711618
α	$K = 30$	$K = 40$	α	$K = 50$
0.125	198.7659014	265.0043682	0.0625	209.379642
0.078125	**169.0664802**	198.696205	0.0390625	**192.6971214**
0.0546875	219.2341173	**197.7570584**	0.02734375	200.6171752
0.04296875	228.1514161	201.0941101	0.021484375	212.145916
0.037109375	251.6520821	199.7605299	0.018554688	230.2994546
0.034179688	249.5441946	210.2559352	0.017089844	236.3387839
0.032714844	268.673412	213.0502394	0.016357422	249.5591561
0.031982422	275.0394893	214.4262525	0.015991211	240.2662121
0.031616211	258.8199039	221.7838157	0.015808105	250.1445775
0.03125	282.1265379	212.7791208	0.015625	238.2748248
α	$K = 60$	$K = 70$	α	$K = 80$
0.0625	235.2669231	260.5479903	0.03125	212.5866831
0.0390625	189.6965079	203.3562005	0.01953125	**190.5264806**
0.02734375	**189.2839767**	**196.4012438**	0.013671875	200.3341841
0.021484375	198.5768633	198.4901067	0.010742188	216.2662349
0.018554688	204.3621865	200.4352023	0.009277344	213.868842
0.017089844	209.8752994	209.3092474	0.008544922	225.5514602
0.016357422	206.2688755	202.3254408	0.008178711	225.5422196
0.015991211	214.5388827	206.7862197	0.007995605	228.2774061
0.015808105	224.3645533	203.9689737	0.007904053	228.550089
0.015625	207.5672402	201.9280646	0.0078125	235.5341334
α	$K = 90$	$K = 100$		
0.03125	223.0388033	243.9334951		
0.01953125	**196.7055075**	207.7824428		
0.013671875	199.2360492	**201.8324488**		
0.010742188	209.5776894	203.2222353		
0.009277344	210.151884	211.8104162		
0.008544922	218.7214327	214.686273		
0.008178711	218.4004633	208.7410358		
0.007995605	229.4354545	218.1692104		
0.007904053	225.6945476	218.9222873		
0.0078125	223.1142452	214.8502284		

Tabla 4.19: Valores de α y funciones aproximadas

K	Intervalos	Recocido Simulado	Diferencia Absoluta
2	**0.21875**	0.231920917666258	0.013170918
10	**0.1484375**	0.150946812286281	0.002509312
20	**0.109375**	0.110409398029701	0.001034398
30	**0.078125**	0.0925477513275956	0.014422751
40	0.0546875	**0.0414137549344286**	0.013273745
50	0.0390625	**0.0353988664320534**	0.003663634
60	**0.02734375**	0.028712263957292	0.001368514
70	0.02734375	**0.0226200347452994**	0.004723715
80	0.01953125	**0.0173697109028678**	0.002161539
90	0.01953125	**0.0182521667064247**	0.001279083
100	0.013671875	**0.0104088991784998**	0.003262976
		Promedio	0.00553369

Tabla 4.20: Valores de α obtenidos por Intervalos y Recocido Simulado

- El esquema de enfriamiento elegido fue enfriamiento geométrico con $\beta = 0.9$.
- El valor inicial de α para el conjunto de máquinas, se estableció de manera aleatoria entre 0 y 1.
- Los α vecinos se generaron aleatoriamente entre $\alpha_{actual}+0.1$ y $\alpha_{actual}-0.1$ con límite superior 1 y límite inferior 0.
- Para determinar la calidad α como solución, se usó, al igual que en el algoritmo de intervalos, el arrepentimiento promedio que se obtiene al ejecutar 100 veces a UCB con α en 1000 problemas de bandido de K máquinas a 100000 rondas.

Bajo las condiciones anteriores se obtuvieron los resultados mostrados en la tabla 4.20 y en la figura 4.22.

De la tabla 4.20 y la figura 4.22 se puede observar que los valores obtenidos son bastante similares a los obtenidos por el algoritmo de intervalos teniendo una diferencia promedio de 0.0055 con lo cual se puede afirmar que ambos algoritmos convergen de manera similar y que los valores obtenidos por el algoritmo de intervalos no son mínimos locales. Es necesario aclarar que en las siguientes secciones se usaran los valores obtenidos por el algoritmo de intervalos.

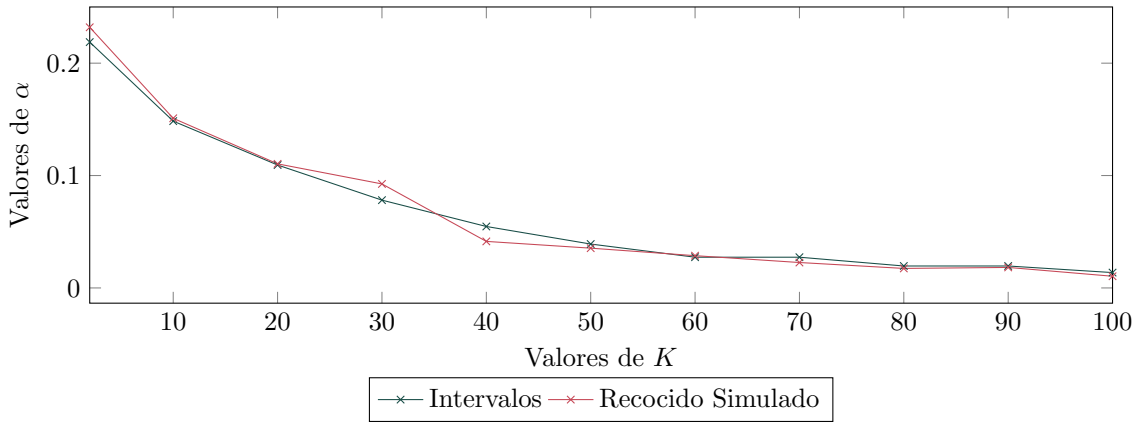


Figura 4.22: Valores de α obtenidos por Intervalos y Recocido Simulado

4.2.2. Análisis de Resultados

Con los valores empíricos encontrados en los experimentos se plantearon dos funciones que aproximan a los valores de α que dieron el mejor rendimiento:

$$\alpha_1(K) = \frac{1}{K} \tag{4.1}$$

$$\alpha_2(K) = \frac{e}{2K} \tag{4.2}$$

La figura 4.23 muestra las funciones $\alpha_1(K)$ y $\alpha_2(K)$ y cómo ambas aproximan la relación entre K y los valores α empíricos con el mejor rendimiento de acuerdo a la tabla 4.19.

Estas dos funciones están pensadas bajo dos condiciones:

- que mantengan lo más posible la simplicidad de UCB.
- mantener el comportamiento de que los valores tiendan a 0 conforme K incrementa.

Con estas dos funciones se generan dos políticas de activación basadas en UCB las cuales se muestran en los algoritmos 17 y 18.

Se espera que estas políticas tengan mejor rendimiento en GGP que la versión de UCB originalmente propuesta por Auer et al. [5]. Cabe resaltar que la principal característica y aportación

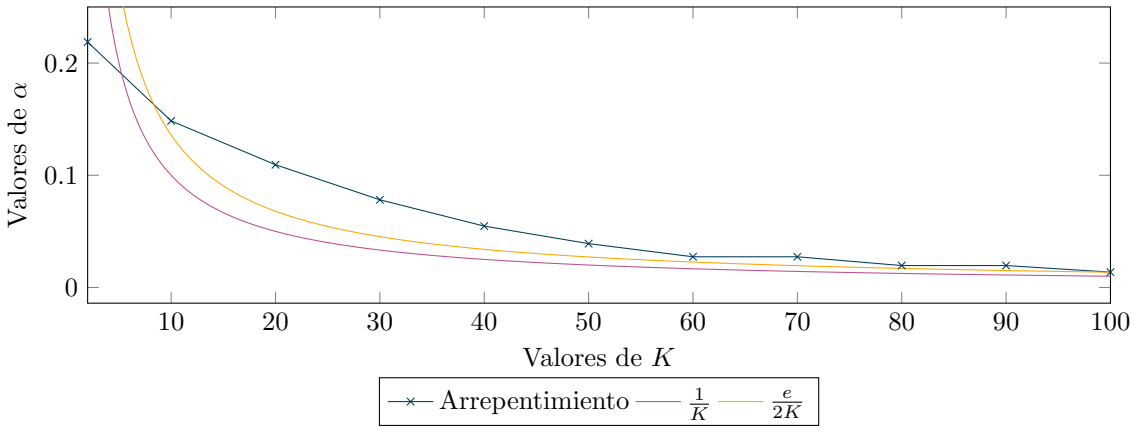


Figura 4.23: Funciones que aproximan a los valores encontrados de α

Algoritmo 17: Upper Confidence Bound α_1

Entrada: K máquinas y T Rondas

Activar cada máquina una sola vez;

para $t \leftarrow K + 1$ **a** T **hacer**

  Activar la máquina i que maximice;

$$\bar{x}_{i,T_i} + \sqrt{\frac{\log t}{KT_i}}$$

es que: estas políticas se adaptarán al número de máquinas a las cuales se enfrenten, lo que en el ámbito de GGP consiste en adaptarse al número de nodos, por ejemplo para el árbol de la figura 4.24, si se usara UCB en su versión original en todos los niveles la constante de α, que controla el balance entre la explotación y exploración, siempre será igual a 2 lo que de acuerdo a los datos empíricos que se obtuvieron no tendría un buen rendimiento (ver tabla 4.19), en cambio las políticas UCBα_1 y UCBα_2 se adaptarán al número de nodos, en el primer nivel para UCBα_1 la constante será $\frac{1}{2}$ y para UCBα_2 será $\frac{e}{4}$ que son valores próximos a los valores óptimos encontrados en los experimentos, el segundo nivel del árbol para el nodo derecho las constantes serán $\frac{1}{3}$ y $\frac{e}{6}$ respectivamente y para el nodo izquierdo serán $\frac{1}{4}$ y $\frac{e}{8}$ y de esta manera para el resto del árbol como se puede observar en la figura 4.24. En la próxima sección se probarán estas políticas en el ámbito de GGP para determinar si las políticas UCBα_1 y UCBα_2 obtienen un mejor rendimiento que UCB.

Algoritmo 18: Upper Confidence Bound α_2

Entrada: K máquinas y T Rondas

1 Activar cada máquina una sola vez;

2 **para** $t \leftarrow K+1$ **a** T **hacer**

3 Activar la máquina i que maximice;

4

$$\bar{x}_{i,T_i} + \sqrt{\frac{e \log t}{2KT_i}}$$

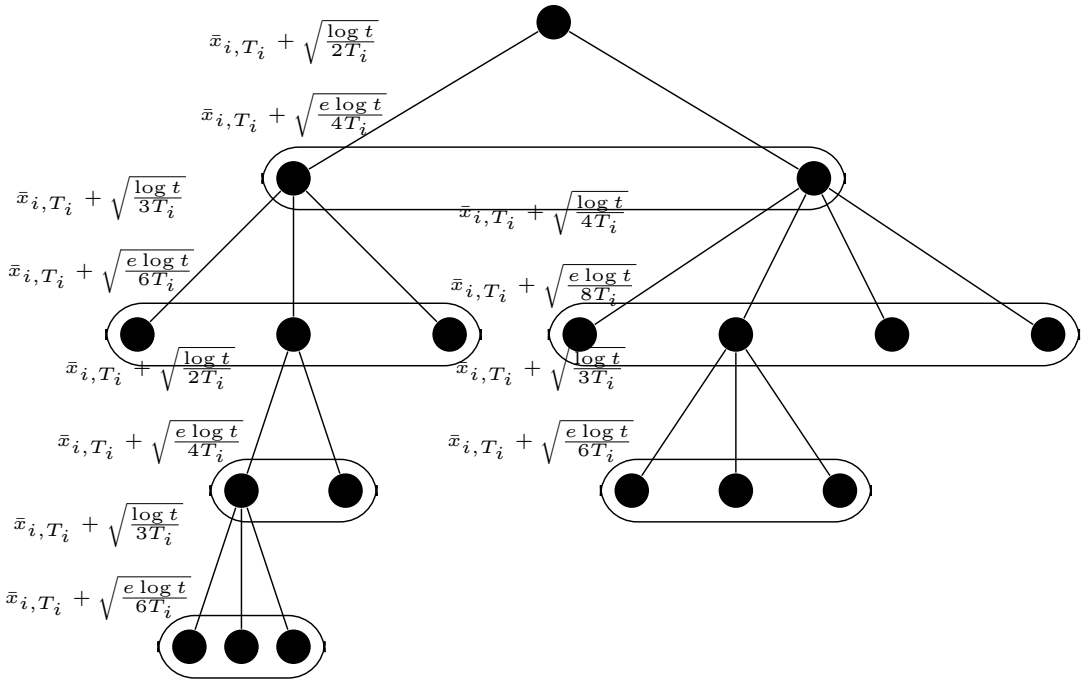


Figura 4.24: Funcionamiento de UCBα_1 y UCBα_2

4.3. Comparativa de las políticas propuestas en GGP

En esta sección se muestra el rendimiento de las políticas UCBα_1 y UCBα_2 con respecto a las políticas UCB, UCB-Tuned y ETC, para ello se realizó un comparativo bajo las mismas condiciones

y juegos que en la sección 4.1. Cabe resaltar que para comparar el rendimiento de las políticas propuestas con respecto a UCB, UCB-Tuned y ETC, se tomaran en cuenta los resultados que estas últimas obtuvieron en el comparativo anterior.

4.3.1. Atari-Go

Respecto a la cantidad de victorias que las políticas logran, se puede observar en la tabla 4.21 y figura 4.25, cuando ABMC está limitado a 100 simulaciones, es UCBα_1 la política con el mayor número al obtener 711 victorias, 99 victorias más que la política ETC la cual en el comparativo anterior obtuvo el mayor número de victorias en este juego.

	100 Simulaciones					1,000 Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
1	177	188	173	241	221	161	215	156	233	235
2	167	210	157	227	239	189	204	139	235	233
3	153	214	170	243	220	168	200	158	236	238
Total	497	612	500	**711**	680	518	619	453	704	**706**
Promedio	165.67	204.00	166.67	**237.00**	226.67	172.67	206.33	151.00	234.67	**235.33**
σ	±12.06	±14.00	±8.5	±8.72	±10.69	±14.57	±7.77	±10.44	±1.53	±2.52

Tabla 4.21: Victorias de UCBα_1 y UCBα_2 en Atari Go

Con respecto a la limitación de 1,000 simulaciones de ABMC se puede considerar que ambas políticas UCBα_1 y UCBα_2 tiene el mejor rendimiento ya que ambas obtienen el mayor número de victorias con 704 y 706 respectivamente, 85 y 87 victorias por encima de las logradas por ETC.

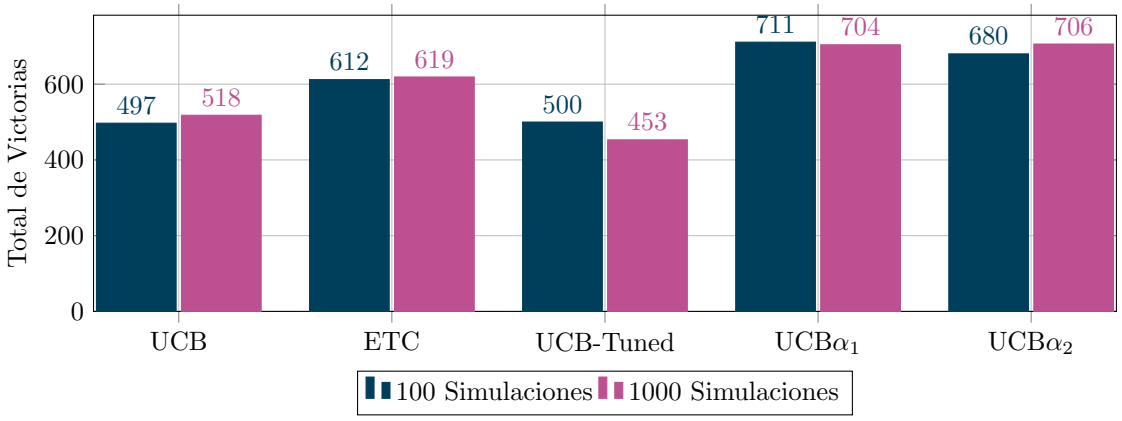


Figura 4.25: Victorias de UCBα_1 y UCBα_2 en Atari Go

La tabla 4.22 muestra el porcentaje que las políticas UCBα_1 y UCBα_2 logran al enfrentarse al resto. En esta tabla se muestra que tanto en 100 y 1,000 simulaciones, la política UCBα_1 tiene el mejor desempeño que las políticas UCB, UCB-Tuned y ETC, al obtener un mayor porcentaje de partidas ganadas que estas, siendo el más bajo (55.67 %) cuando se enfrenta a ETC en 1,000 simulaciones, y el porcentaje más alto el de 66.33 % al enfrentarse a UCB-Tuned igualmente cuando ABMC está limitado a 1,000 simulaciones, como se puede observar en la figura 4.26.

100 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCBα_1	**65.00 ± 5.69**	**57.67 ± 1.15**	**64.00 ± 5.51**		**50.33 ± 2.08**
UCBα_2	**67.00 ± 2.52**	49.33 ± 2.65	**60.67 ± 4.16**	49.67 ± 2.08	
1000 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCBα_1	**63.33 ± 5.69**	**55.67 ± 1.15**	**66.33 ± 5.51**		49.33 ± 2.08
UCBα_2	**62.33 ± 2.52**	**59.00 ± 2.65**	**63.33 ± 4.16**	**50.67 ± 2.08**	

Tabla 4.22: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Atari Go

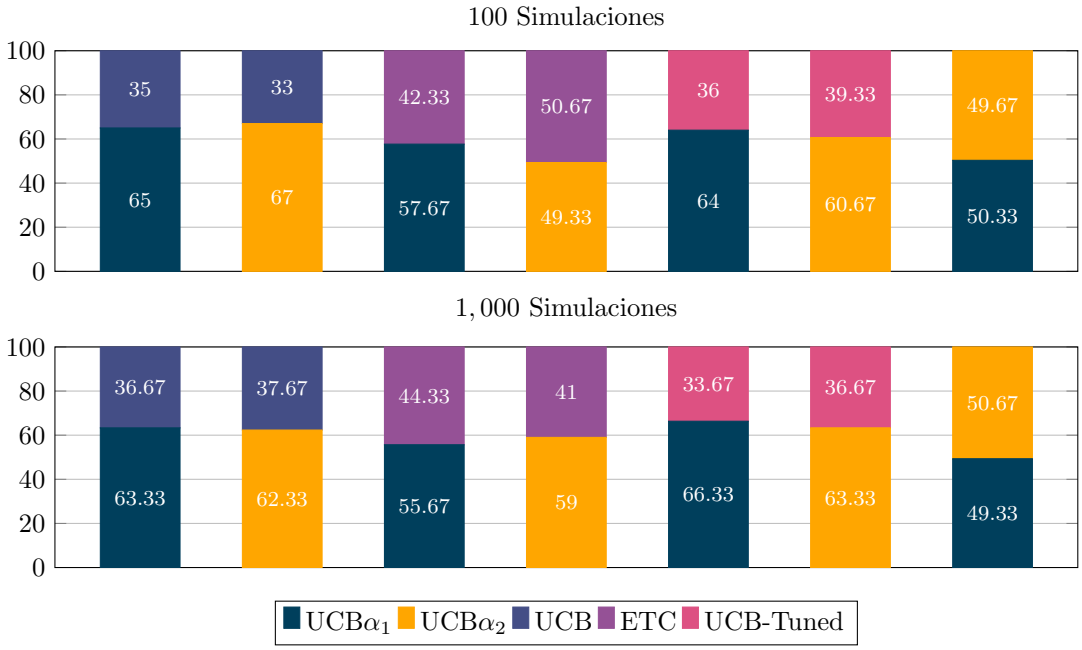


Figura 4.26: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Atari Go

Respecto a UCBα_2, cuando ABMC está limitado a 100, no logra vencer a ETC ya que solo sale victoriosa en solo 49.33 % partidas, sin embargo, logra vencer tanto a UCB como UCB-Tuned al vencer a ambas en más del 60 % de las partidas como se observa en la figura 4.26. En el caso de que ABMC está limitado a $1,000$, UCBα_2 mejora su rendimiento ya que logra vencer el 59 % de las partidas a ETC, a diferencia del caso de 100 simulaciones, y de igual forma logra vencer en más del 60 % a las políticas UCB y UCB-Tuned.

Respecto al enfrentamiento entre UCBα_1 y UCBα_2, el desempeño podría considerarse similar como se puede observar en la figura 4.26, ya que la diferencia entre el porcentaje de partidas que cada una gana es de solo 0.66 % en 100 simulaciones y 1.34 % en $1,000$ simulaciones.

De los resultados obtenidos se puede observar que en el juego de Atari-Go las políticas UCBα_1 y UCBα_2 tiene un mejor desempeño que UCB, UCB-Tuned y ETC.

4.3.2. Breakthrough 6×6

Acorde con los resultados mostrados en la tabla 4.23 y la figura 4.27 se observa que cuando ABMC está limitado a 100 simulaciones, las política UCBα_1 y UCBα_2 tienen el mejor rendimiento que ETC la política con el mayor número de victorias de acuerdo al comparativo anterior, en específico UCBα_1 logro 46 victorias más que ETC y UCBα_2 obtuvo 10 victorias por encima de las logradas por ETC.

	100 Simulaciones					$1,000$ Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
1	166	218	166	236	214	174	104	182	271	269
2	175	217	147	232	229	183	103	185	265	264
3	153	224	160	237	226	165	131	173	256	275
Total	494	659	473	**705**	669	522	338	540	792	**808**
Promedio	164.67	219.67	157.67	**235.00**	223.00	174.00	112.67	180.00	264	**269.33**
σ	±11.06	±3.79	±9.71	±2.65	±7.94	±9	±15.89	±6.24	±7.55	±5.51

Tabla 4.23: Victorias de UCBα_1 y UCBα_2 en Breakthrough 6×6

En el caso de $1,000$ simulaciones UCBα_1 obtiene 252 victorias más que UCB-Tuned, política con el mejor desempeño en este caso de acuerdo con el comparativo anterior, y UCBα_2 obtuvo 260 victorias más que UCB-Tuned, por lo cual las políticas propuestas tienen el mejor desempeño a lograr un mayor número de victorias.

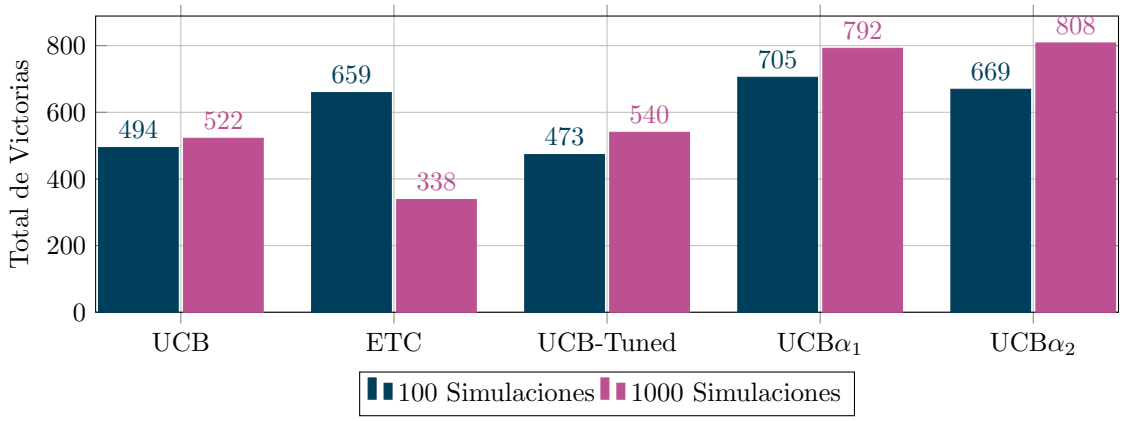


Figura 4.27: Victorias de UCBα_1 y UCBα_2 en Breakthrough 6×6

Respecto al porcentaje de partidas ganadas, UCBα_1 vence a UCB, ETC y UCB-Tuned tanto en 100 y $1,000$ simulaciones siendo el porcentaje mínimo de $57\,\%$ de partidas ganadas al enfrentarse a ETC en 100 simulaciones y el máximo de $77.67\,\%$ al enfrentarse a ETC en $1,000$ simulaciones, estos resultados pueden observarse en la tabla 4.24 y de manera gráfica en la figura 4.28.

100 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCBα_1	**60.67 ± 6.24**	**57.00 ± 1.15**	**63.00 ± 1.15**		**54.33 ± 3.51**
UCBα_2	**67.00 ± 1.53**	49.67 ± 1.00	**60.67 ± 2.52**	45.67 ± 3.51	
1000 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCBα_1	**70.00 ± 6.24**	**77.67 ± 1.15**	**67.67 ± 1.15**		48.67 ± 3.51
UCBα_2	**70.67 ± 1.53**	**78.00 ± 1.00**	**69.33 ± 2.52**	**51.33 ± 3.51**	

Tabla 4.24: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Breakthrough 6×6

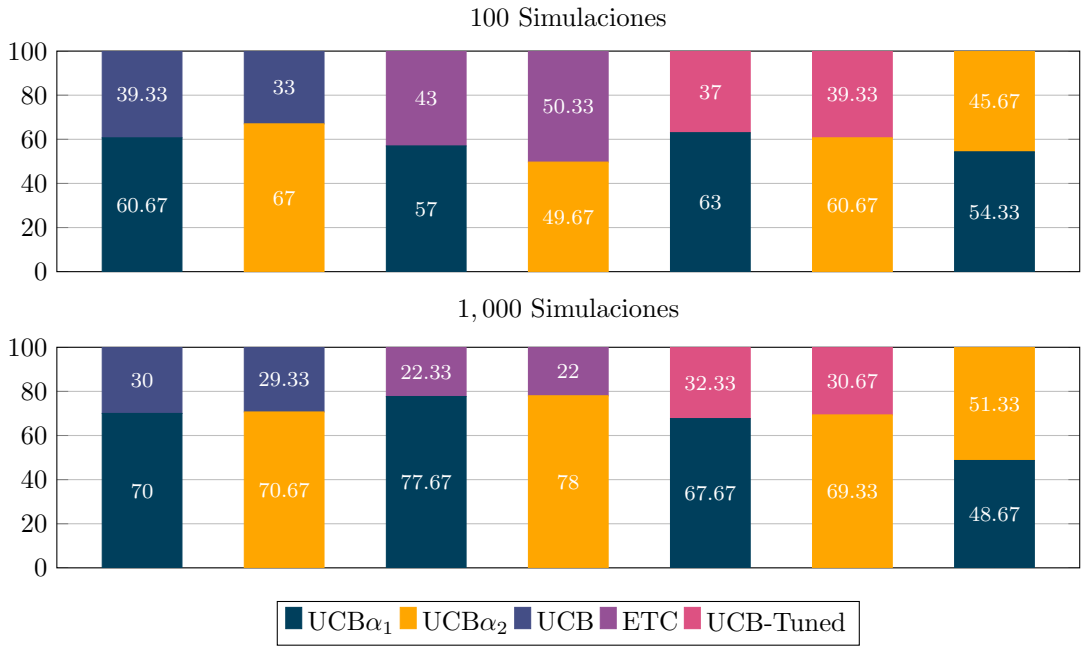


Figura 4.28: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Breakthrough 6×6

UCBα_2 obtiene un mayor porcentaje de partidas ganadas que UCB y UCB-Tuned pero solo logra ganar en 49.67 % de las partidas a ETC, sin embargo, su rendimiento, mejora con $1,000$ simulaciones quedando su mínimo de partidas ganadas en 69.33 % al enfrentarse a UCB-Tuned y su máximo de 78 % al enfrentarse a ETC.

En el enfrentamiento entre UCBα_1 y UCBα_2 de acuerdo a la figura 4.28 se observa que UCBα_1 obtiene el 54.33 % porcentaje de partidas ganadas en 100 simulaciones y UCBα_2 obtiene el 51.33 % en 1,000 simulaciones, por lo cual se considera que la mejor política en Breakthrough 6×6 cuando ABMC está limitado a 100 simulaciones es UCBα_1 y cuando ABMC está limitado a 1,000 es UCBα_2.

4.3.3. Breakthrough with Holes 6×6

En el juego Breakthrough with Holes 6×6, política con mejor desempeño cuando ABMC se limita a 100 simulaciones, es UCBα_2 la que logra 89 victorias más que ETC la política con mejor desempeño en el comparativo anterior, ver tabla 4.25 y figura 4.29.

Para el caso de 1,000 simulaciones la política UCBα_1 obtiene 820 victorias más que la política UCB-Tuned que obtuvo la mayor cantidad de victorias en este juego de acuerdo con el comparativo anterior.

	100 Simulaciones					1,000 Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
1	182	195	184	219	220	172	108	177	285	258
2	201	194	186	208	211	184	113	177	261	265
3	182	182	185	222	229	177	103	180	274	266
Total	565	571	555	649	**660**	533	324	534	**820**	789
Promedio	188.33	190.33	185.00	216.33	**220.00**	177.67	108.00	178.00	**273.33**	263.00
σ	±10.97	±7.23	±1.00	±7.37	±9.00	±6.03	±5.00	±1.73	±12.01	±4.36

Tabla 4.25: Victorias de UCBα_1 y UCBα_2 en Breakthrough with Holes 6 × 6

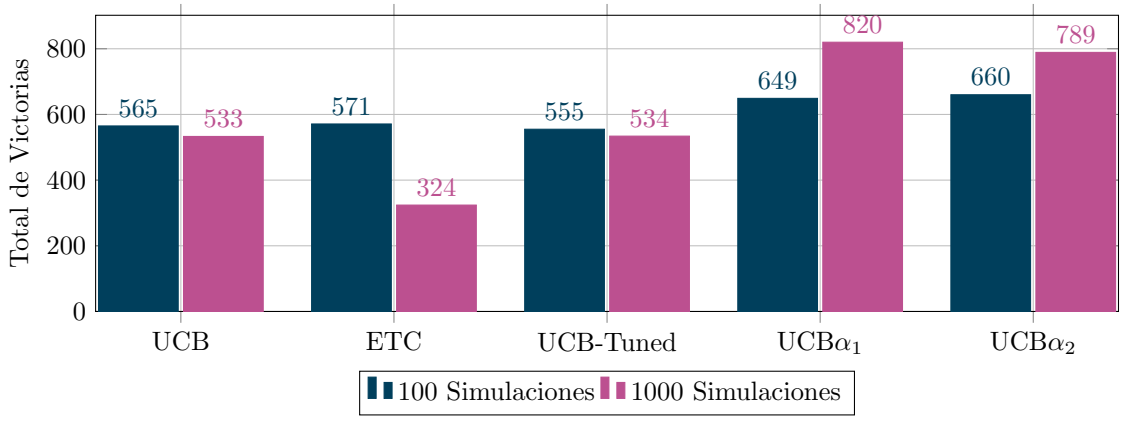


Figura 4.29: Victorias de UCBα_1 y UCBα_2 en Breakthrough with Holes 6 × 6

Respecto al desempeño de la política UCBα_1, se puede observar que logran obtener un mayor porcentaje de victorias que UCB, ETC y UCB-Tuned sin importar la cantidad de simulaciones del ABMC, pero se puede observar que su desempeño es mejor conforme el número de simulaciones aumenta alcanzando un 80 % al enfrentarse a ETC, esto puede ser observado en la tabla 4.26 y de manera gráfica en la figura 4.30.

100 Simulaciones					
	UCB	ETC	UCB-Tuned	UCB-α_1	UCBα_2
UCBα_1	**56.33 ± 3.06**	**57.00 ± 5.29**	**52.33 ± 3.06**		**50.67 ± 6.03**
UCBα_2	**53.00 ± 3.51**	**59.67 ± 1.53**	**58.00 ± 2.00**	49.33 ± 6.03	
1000 Simulaciones					
	UCB	ETC	UCB-Tuned	UCB-α_1	UCBα_2
UCBα_1	**69.67 ± 3.06**	**80.00 ± 5.29**	**71.33 ± 3.06**		**52.33 ± 6.03**
UCBα_2	**68.67 ± 3.51**	**76.67 ± 1.53**	**70.00 ± 2.00**	47.67 ± 6.03	

Tabla 4.26: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Breakthrough with Holes 6×6

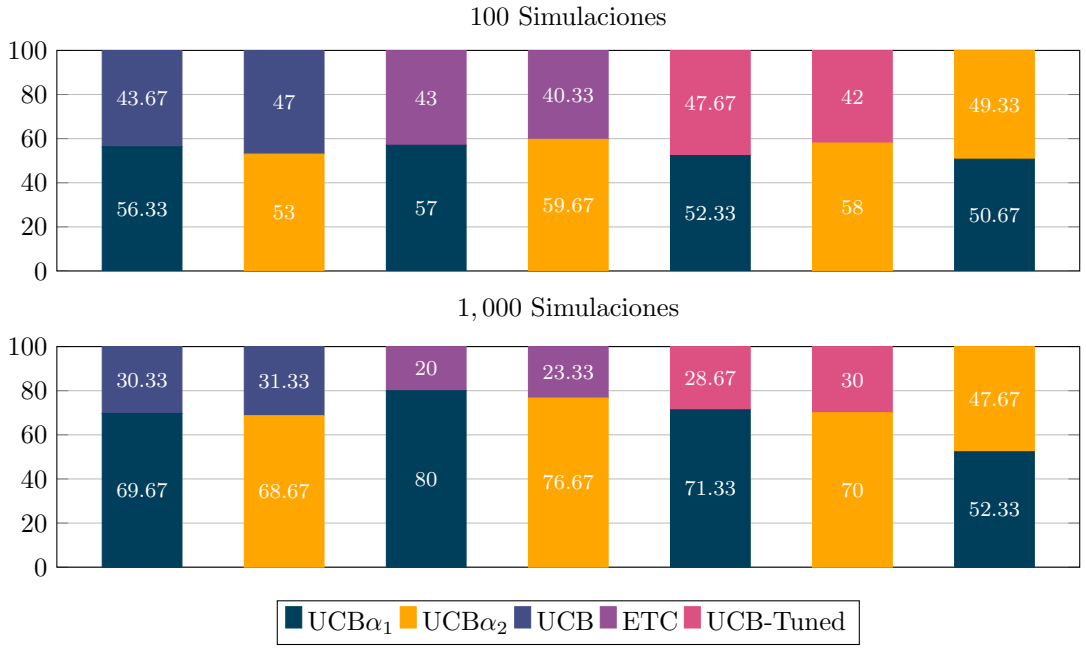


Figura 4.30: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Breakthrough with Holes 6×6

La figura 4.30 muestra el desempeño de UCBα_2 en el cual se puede observar que también logra obtener un porcentaje de victorias más alto que las políticas UCB, ETC y UCB-Tuned, teniendo un máximo de 76.67 % al enfrentarse a ETC cuando ABMC se limita a 1,000 simulaciones.

Del enfrentamiento entre UCBα_1 y UCBα_2 se puede observar que en 100 simulaciones el rendimiento entre las políticas es bastante similar, sin embargo, una vez limitado ABMC a 1,000 simulaciones se observa que UCBα_1 tiene el mejor desempeño quien vence a UCBα_2 en 52.33 % de

las partidas.

Para el juego Breakthrough with Holes 6×6 las políticas UCBα_1 y UCBα_2 tienen el mejor desempeño que las políticas UCB, ETC y UCB-Tuned siendo UCBα_1 la mejor.

4.3.4. Breakthrough Suicide 6×6

De la tabla 4.27 y figura 4.31 se observa que si ABMC se limita a 100 simulaciones, UCBα_1 no supera a la política UCB que en el comparativo anterior obtuvo la mayor cantidad de victorias, caso contrario de la política UCBα_2 que logra superar a UCB únicamente con 2 victorias.

	100 Simulaciones					1,000 Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
1	212	178	200	199	211	194	190	178	229	209
2	220	182	177	199	222	181	190	193	223	213
3	197	210	207	188	198	197	205	168	223	207
Total	629	570	584	586	**631**	572	585	539	**675**	629
Promedio	209.67	190.00	194.67	195.33	**210.33**	190.67	195.00	179.67	**225.00**	209.67
σ	±11.68	±17.44	±15.7	±6.35	±12.01	±8.50	±8.66	±12.58	±3.46	±3.06

Tabla 4.27: Victorias de UCBα_1 y UCBα_2 en Breakthrough Suicide 6×6

Para el caso de 1,000 simulaciones las políticas propuestas incrementan su desempeño ya que UCBα_1 obtiene 103 victorias más que UCB, lo mismo ocurre con UCBα_2 que obtuvo 57 victorias por encima de UCB, sin embargo, debe observarse que la política ETC también incrementa su desempeño al obtener 13 victorias más que UCB pero no logra superar a UCBα_1 y UCBα_2.

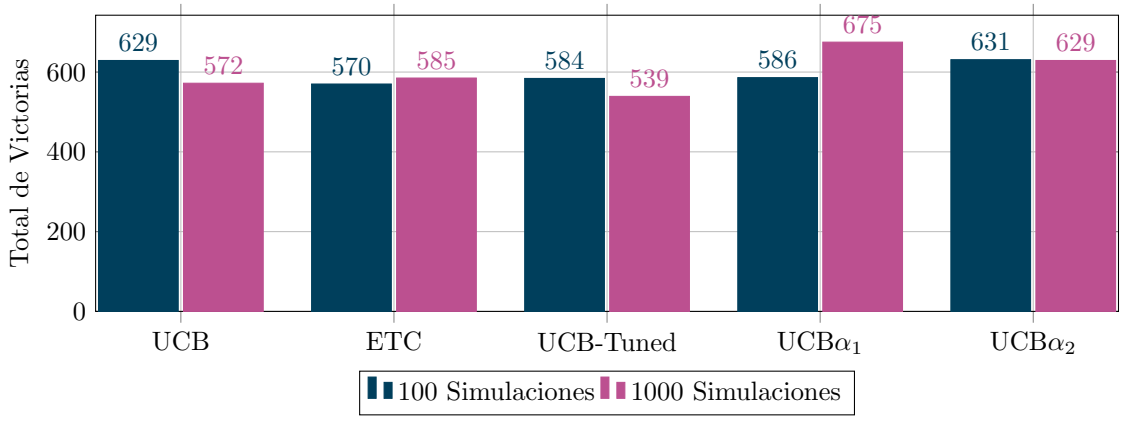


Figura 4.31: Victorias de UCBα_1 y UCBα_2 en Breakthrough Suicide 6×6

Respecto al porcentaje de partidas ganadas de acuerdo a la tabla 4.28 y la figura 4.32, en 100 simulaciones UCBα_1, supera a UCB y ETC, pero no lograr obtener un porcentaje mayor que UCB-Tuned, sin embargo, el desempeño incrementa en $1,000$ simulaciones ya que logra vencer tanto a UCB y ETC como a UCB-Tuned dando el mínimo de porcentaje de partidas ganadas al enfrentarse a ETC con un $52\,\%$.

100 Simulaciones					
	UCB	ETC	UCB-Tuned	UCB-α_1	UCBα_2
UCBα_1	45.33 ± 5.29	49.67 ± 3.61	$\mathbf{51.00 \pm 6.66}$		49.33 ± 0.58
UCBα_2	$\mathbf{52.00 \pm 2.65}$	$\mathbf{51.67 \pm 3.61}$	$\mathbf{56.00 \pm 2.00}$	$\mathbf{50.67 \pm 0.58}$	
1000 Simulaciones					
	UCB	ETC	UCB-Tuned	UCB-α_1	UCBα_2
UCBα_1	$\mathbf{63.00 \pm 5.29}$	$\mathbf{52.00 \pm 3.61}$	$\mathbf{58.67 \pm 6.66}$		$\mathbf{51.33 \pm 0.58}$
UCBα_2	$\mathbf{55.00 \pm 2.65}$	47.00 ± 3.61	$\mathbf{59.00 \pm 2.00}$	48.67 ± 0.58	

Tabla 4.28: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Breakthrough Suicide 6×6

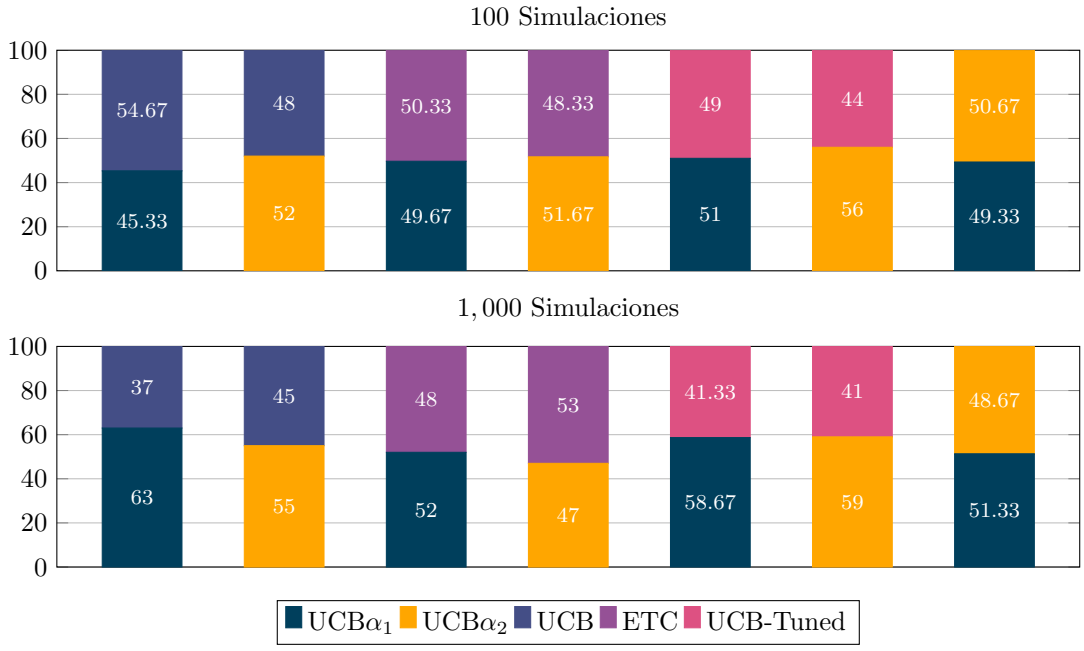


Figura 4.32: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Breakthrough Suicide 6×6

Para el caso de la política UCBα_2, cuando ABMC está limitado a 100 simulaciones, la política logra obtener un mayor porcentaje de victorias que UCB, ETC y UCB-Tuned, sin embargo, cuando

ABMC se limita a $1,000$ simulaciones, UCBα_2 solo logra vencer a UCB y UCB-Tuned, sin embargo, a ETC solo logra vencerla en $47\,\%$ de las partidas.

Respecto al enfrentamiento entre UCBα_1 y UCBα_2 se observa que en 100 simulaciones el rendimiento de ambas políticas podría considerarse similar, ya que UCBα_2 solo supera a UCBα_1 en un $1.34\,\%$ como se observa en la figura 4.32, sin embargo, en $1,000$ simulaciones esta diferencia incrementa a $2.66\,\%$ dando la ventaja a UCBα_1.

Por los resultados obtenidos se puede considerar que UCBα_2 tiene el mejor rendimiento en Breakthrough Suicide 6×6 cuando ABMC tiene como límite 100 simulaciones, seguida muy de cerca por la política UCB, para el caso de $1,000$ simulaciones es la política UCBα_1 la que presenta el mejor desempeño.

4.3.5. Knight-Through

De acuerdo a la tabla 4.29 y figura 4.33 se observa que las políticas propuestas tienen mejor desempeño que la política ETC la cual obtuvo la mayor cantidad de victorias, en específico al limitar a ABMC a 100 simulaciones la política UCBα_1, obtiene 257 victorias más que ETC y UCBα_2 obtuvo 263 más que dicha política.

	100 Simulaciones					$1,000$ Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
1	161	175	163	249	252	148	201	156	244	251
2	151	171	166	243	269	130	219	138	234	279
3	149	168	148	279	256	139	204	145	265	247
Total	461	514	477	771	**777**	417	624	439	743	**777**
Promedio	153.67	171.33	159.00	257.00	**259**	139.00	208.00	146.33	247.67	**259**
Promedio	±6.43	±3.51	±9.64	±19.29	±8.89	±9.00	±9.64	±9.07	±15.82	±17.44

Tabla 4.29: Victorias de UCBα_1 y UCBα_2 en Knight-Through

Para el caso de ABMC limitado a $1,000$ simulaciones, la política, UCBα_1 logra 119 victorias por encima de ETC y UCBα_2 obtiene 153 victorias más que ETC.

La tabla 4.30 y la figura 4.34 muestran el rendimiento de la política UCBα_1, en estas se puede observar que la política propuesta vence a UCB, ETC y UCB-Tuned, en más del $60\,\%$ de las partidas sin importar el número de simulaciones de ABMC, el mayor porcentaje de victorias se da cuando UCBα_1, enfrenta a UCB en 100 simulaciones donde logra vencer a esta última en $74\,\%$ de las

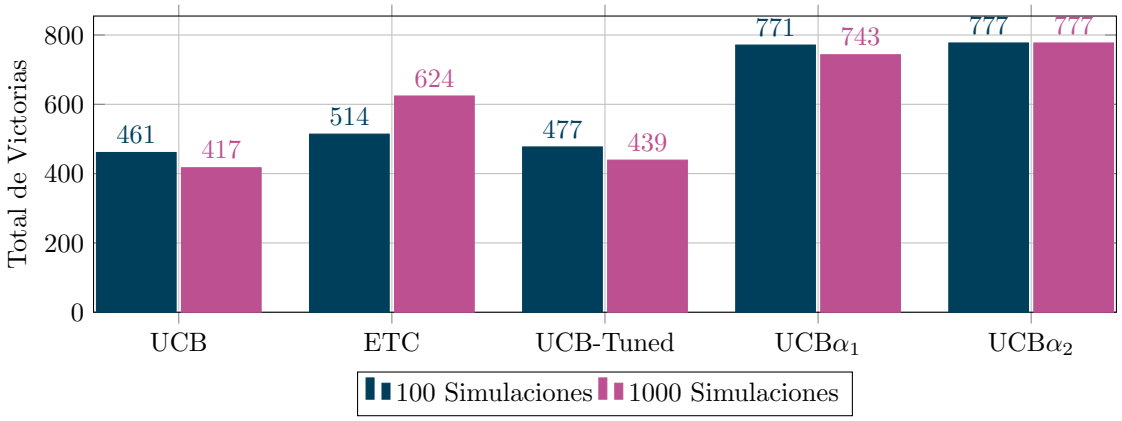


Figura 4.33: Victorias de UCBα_1 y UCBα_2 en Knight-Through

partidas.

100 Simulaciones					
	UCB	ETC	UCB-Tuned	UCB-α_1	UCBα_2
UCBα_1	**74.00 ± 4.51**	**70.33 ± 3.00**	**65.33 ± 2.08**		47.33 ± 9.61
UCBα_2	**71.00 ± 8.72**	**63.67 ± 1.53**	**71.67 ± 7.00**	**52.67 ± 9.61**	
1000 Simulaciones					
	UCB	ETC	UCB-Tuned	UCB-α_1	UCBα_2
UCBα_1	**71.33 ± 4.51**	**61.00 ± 3.00**	**69.67 ± 2.08**		45.67 ± 9.61
UCBα_2	**71.00 ± 8.72**	**61.67 ± 1.53**	**72.00 ± 7.00**	**54.33 ± 9.61**	

Tabla 4.30: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Knight-Through

Al igual que UCBα_1, UCBα_2 logra vencer a las políticas UCB, ETC y Tuned en más del 60 % de las partidas, siendo el porcentaje más alto de 72 % al enfrentar UCB-Tuned en 1,000 simulaciones.

Al enfrentar a UCBα_1 y UCBα_2 se puede observar que es la segunda política es la que tiene el mejor rendimiento ya que UCBα_2 vence en 52.67 % partidas y en 54.33 % partidas cuando ABMC está limitado a 100 y 1,000 simulaciones respectivamente.

Para Knight-Through los resultados muestran que la política UCBα_2 tiene el mejor desempeño ya que mantiene el mismo número de victorias sin importar la cantidad de simulaciones y además de vencer en una mayor cantidad de partidas al resto de políticas.

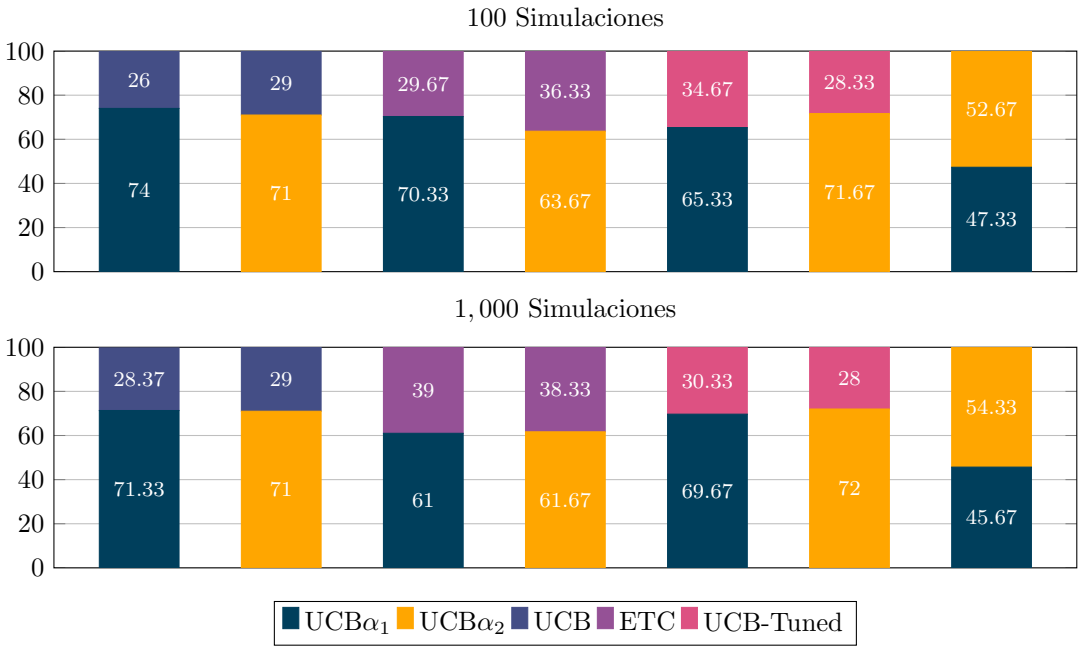


Figura 4.34: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Knight-Through

4.3.6. Sheep and Wolf

La tabla 4.31 y la figura 4.35 muestran los resultados de las políticas en el juego de Sheep and Wolf, de estos se observa que UCBα_1 y UCBα_2 obtienen una mayor cantidad de victorias que la mejor política de acuerdo al comparativo anterior en este caso UCB-Tuned.

	100 Simulaciones					1,000 Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
1	202	188	202	207	201	204	148	208	219	221
2	196	184	205	209	206	206	149	211	222	212
3	196	186	196	211	211	202	144	200	223	231
Total	594	558	603	**627**	618	612	441	619	**664**	**664**
Promedio	198.00	186.00	201.00	**209.00**	206.00	204.00	147.00	206.33	**221.33**	221.33
σ	±3.46	±2.00	±4.58	±2.00	±5.00	±2.00	±2.65	±5.69	±2.08	±9.50

Tabla 4.31: Victorias de UCBα_1 y UCBα_2 en Sheep and Wolf

Para el caso de 100 simulaciones UCBα_1 obtuvo 24 victorias más que UCB-Tuned y UCBα_2 obtuvo 15 victorias por arriba de las logradas por UCB-Tuned. Cuando ABMC está limitado a 1,000 simulaciones, ambas políticas UCBα_1 y UCBα_2 obtuvieron 45 victorias más que UCB-Tuned.

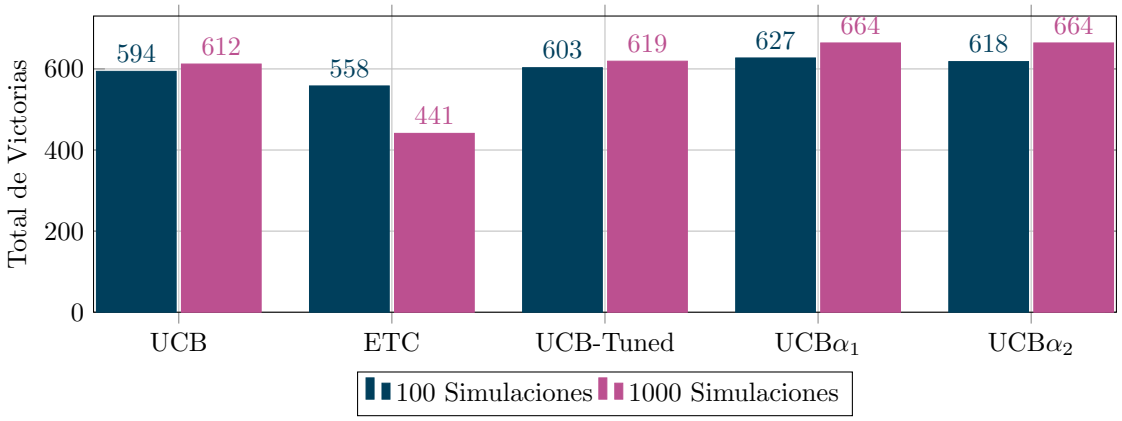


Figura 4.35: Victorias de UCBα_1 y UCBα_2 en Sheep and Wolf

Respecto al porcentaje de partidas ganadas se puede observar que UCBα_1 logra vencer a UCB, ETC y UCB-Tuned, sin importar el número de simulaciones, de acuerdo a las tablas 4.31 y la figura 4.35 el rango de partidas ganadas por UCBα_1 esta entre 52 % y 53 % a excepción del enfrentamiento con ETC con ABMC limitado a 1,000 simulaciones donde UCBα_1 alcanza el 68 % de partidas ganadas.

100 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCBα_1	**53.00 ± 2.65**	**52.67 ± 1.00**	**52.67 ± 2.08**		**50.67 ± 1.53**
UCBα_2	**52.00 ± 3.51**	**52.67 ± 3.61**	**52.00 ± 3.06**	49.33 ± 1.53	
1000 Simulaciones					
	UCB	ETC	UCB-Tuned	UCB-α_1	UCBα_2
UCBα_1	**52.00 ± 2.65**	**68.00 ± 1.00**	**51.67 ± 2.08**		49.67 ± 1.53
UCBα_2	**52.33 ± 3.51**	**66.00 ± 3.61**	**52.67 ± 3.06**	**50.33 ± 1.53**	

Tabla 4.32: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Sheep and Wolf

Al igual que UCBα_1, UCBα_2 vence en más partidas a UCB, ETC y UCB-Tuned, en el rango de 52 % a 52.67 % partidas, a excepción de ETC en que logra 66 %.

De los enfrentamientos entre UCBα_1 y UCBα_2 se puede observar que el rendimiento es bastante similar siendo la diferencia de 1.34 % cuando ABMC se limita a 100 simulaciones dando la ventaja a UCBα_1 y para el caso de 1,000 simulaciones esta diferencia se reduce a 0.66 % dando la ventaja a UCBα_2.

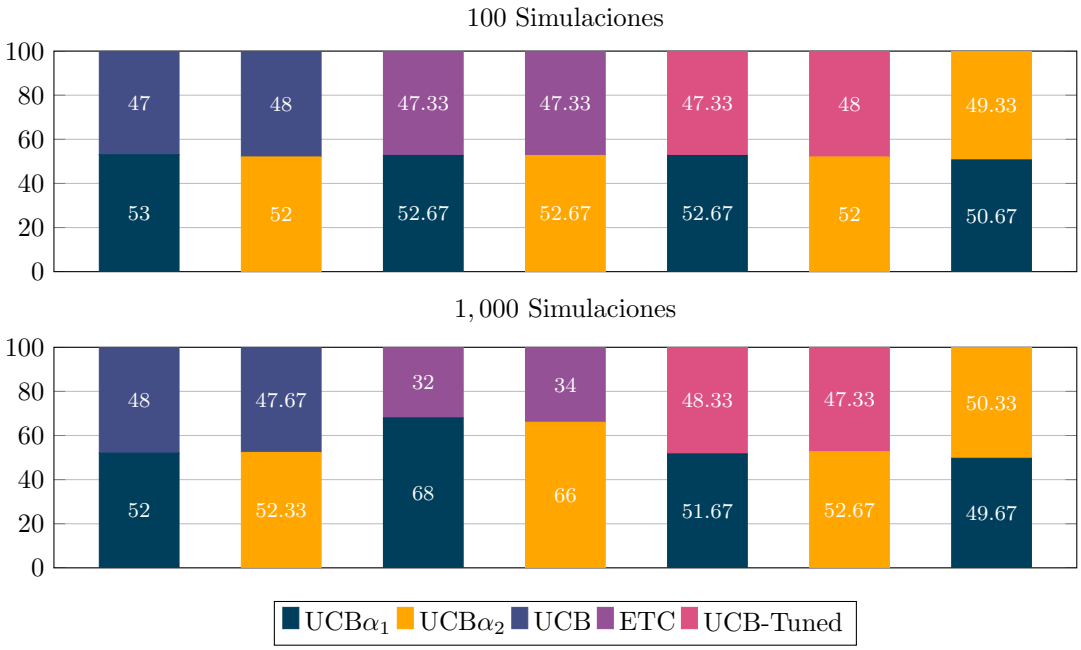


Figura 4.36: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Sheep and Wolf

De los resultados observados se puede asumir que las políticas propuestas tienen el mejor desempeño en el juego de Sheep and Wolf, debe observarse que debido a la naturaleza del juego, los agentes tienden a ganar siempre si juegan como Lobo, y tienden a perder si juegan como ovejas, y de acuerdo a los lineamientos del comparativo los agentes jugaran 50 % de las partidas como lobo y 50 % como ovejas, por lo cual si una política obtiene más de 50 % de partidas ganadas significa que logro vencer jugando como oveja como el caso de las políticas propuestas por eso se considera que tiene un buen desempeño.

4.3.7. Tic-Tac-Toe Large

De acuerdo a la tabla 4.33 y la figura 4.37, cuando ABMC está limitado a 100 simulaciones, UCBα_1 no logra obtener la suficiente cantidad de victorias como para superar a UCB, la mejor política de acuerdo al comparativo anterior, quedando 19 victorias por debajo de las 594 victorias logradas por UCB. Por el contrario, UCBα_2 supera en 28 victorias a UCB, teniendo el mejor

desempeño en 100 simulaciones.

	100 Simulaciones					1,000 Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
1	185	194	188	189	215	215	65	183	230	245
2	196	181	185	199	205	211	78	202	234	227
3	213	172	190	187	202	211	78	201	228	225
Total	594	547	563	575	**622**	637	221	586	692	**697**
Promedio	198.00	182.33	187.67	191.67	**207.33**	212.33	73.67	195.33	230.67	**232.33**
σ	±14.11	±11.06	±2.52	±6.43	±6.81	±2.31	±7.51	±10.69	±3.06	±11.02

Tabla 4.33: Victorias de UCBα_1 y UCBα_2 en Tic-Tac-Toe Large

Para el caso de 1,000 simulaciones las políticas propuestas tienen el mejor desempeño ya que UCBα_1 obtiene 55 victorias más que UCB y UCBα_2 obtuvo 60 victorias más que UCB.

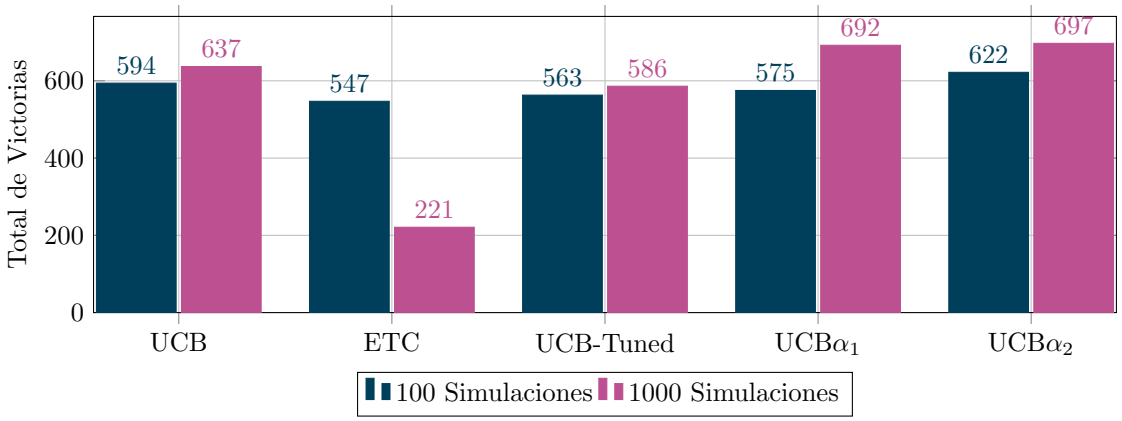


Figura 4.37: Victorias de UCBα_1 y UCBα_2 en Tic-Tac-Toe Large

La tabla 4.34 y la figura 4.38 muestran que cuando ABMC está limitado a 100 simulaciones, UCBα_1 obtiene un mayor porcentaje de victorias que ETC y UCB-Tuned excepto con UCB. En el enfrentamiento contra ETC, UCBα_1 logra 50 % de partidas ganadas contra los 47.67 % del ETC, ambas empatan en 4.33 % partidas. UCBα_1 logra ganar en 49.33 % partidas cuando se enfrenta a UCB-Tuned quien logra ganar 48.33 % partidas, y ambas empatan en 2.34 % partidas. Cuando UCBα_1 enfrenta solo logra ganar 47.67 % de las partidas contra las 48.33 % que logra UCB, aun así, empatan en 4 % partidas.

Para el caso de 1,000 simulaciones, UCBα_1 logra vencer a UCB, ETC y UCB-Tuned, al obtener un porcentaje de partidas ganadas, mayor que 55 % incluso llegando a 74 % cuando se enfrenta a

100 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCBα_1	47.67 $\pm$ 2.08	**50.00 $\pm$ 2.65**	**49.33 $\pm$ 3.79**		44.67 $\pm$ 1.73
UCBα_2	**50.00 $\pm$ 2.31**	**50.00 $\pm$ 5.51**	**53.67 $\pm$ 4.04**	**53.67 $\pm$ 3.06**	
1000 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCBα_1	**55.33 $\pm$ 2.08**	**74.00 $\pm$ 2.65**	**58.33 $\pm$ 3.79**		43.00 $\pm$ 1.73
UCBα_2	**56.33 $\pm$ 2.31**	**73.67 $\pm$ 5.51**	**57.67 $\pm$ 4.04**	44.67 $\pm$ 3.06	

Tabla 4.34: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Tic-Tac-Toe Large

ETC, pero de observarse que la cantidad de empates incrementó llegando a un máximo de 6.67 % en el enfrentamiento con UCB-Tuned.

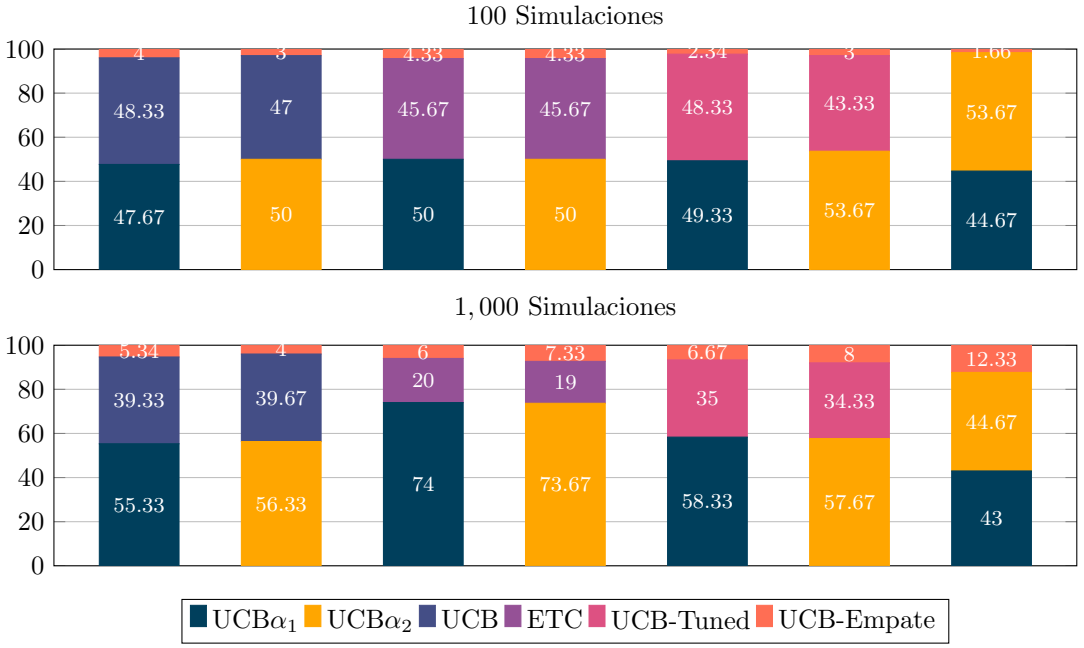


Figura 4.38: Porcentaje de Victorias de UCBα_1 en Tic-Tac-Toe Large

Para el caso de UCBα_2, esta política obtiene un mayor porcentaje de victorias que UCB, ETC y UCB-Tuned sin importar que ABMC este limitado a 100 o 1,000 simulaciones, con un mínino de 50 % en el enfrentamiento contra UCB y ETC en 100 simulaciones y el máximo de 73.67 % a enfrentarse a ETC en 1,000 simulaciones. Respecto a los empates, el máximo número de empates

de UCBα_2 ocurre en el enfrentamiento contra UCB-Tuned donde empatan en 8 % de las partidas.

Al enfrentar a UCBα_1 contra UCBα_2 se puede observar que en 100 simulaciones UCBα_2 vence en 53.67 % partidas a UCBα_1 y empatan en 1.66 %. Cuando se incrementa las simulaciones a 1,000 simulaciones las políticas empatan en 12.33 % sin embargo, UCBα_2 sigue teniendo el mayor porcentaje de partidas ganadas.

Para el juego Tic-Tac-Toe Large la política UCBα_2 tiene el mejor rendimiento, ya que logra el mayor número de victorias y el mayor porcentaje de partidas ganadas al enfrentar el resto de las políticas.

4.3.8. Two-Player Free-For-All Zero-Sum

Para el juego Two-Player Free-For-All Zero-Sum, cuando ABMC se limita a 100 simulaciones, las políticas propuestas obtiene una mayor cantidad de victorias, que ETC la política que en el comparativo anterior obtuvo la mayor cantidad de victorias, en específico UCBα_1 logra 115 victorias y UCBα_2 117 más que ETC.

	100 Simulaciones					1,000 Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
1	187	180	178	239	213	162	117	162	275	281
2	191	186	183	208	228	164	114	163	269	287
3	166	189	191	223	231	158	123	164	263	285
Total	544	555	552	670	**672**	484	354	489	807	**853**
Promedio	181.33	185.00	184.00	223.33	**224.00**	161.33	118.00	163.00	269.00	**284.33**
σ	±13.43	±4.58	±6.56	±15.5	±9.64	±3.06	±4.58	±1.00	±6.00	±3.06

Tabla 4.35: Victorias de UCBα_1 y UCBα_2 en Two-Player Free-For-All Zero-Sum

Para el caso de 1,000 simulaciones, en el comparativo anterior UCB-Tuned obtuvo la mayor cantidad de victorias, sin embargo, su desempeño queda por debajo de las políticas propuestas ya que UCBα_1 obtiene 318 victorias más que UCB-Tuned y UCBα_2 obtuvo 364 victorias más que UCB-Tuned.

La tabla 4.36 y la figura 4.40 muestran que UCBα_1 obtiene un mayor porcentaje de partidas ganadas que UCB, ETC y UCB-Tuned tanto cuando ABMC está limitado a 100 simulaciones como a 1,000 simulaciones. El porcentaje de partidas ganadas por UCBα_1 más bajo se da al enfrentar a UCB con ABMC limitado a 100 simulaciones en específico vence a UCB-Tuned el 55.33 % de las

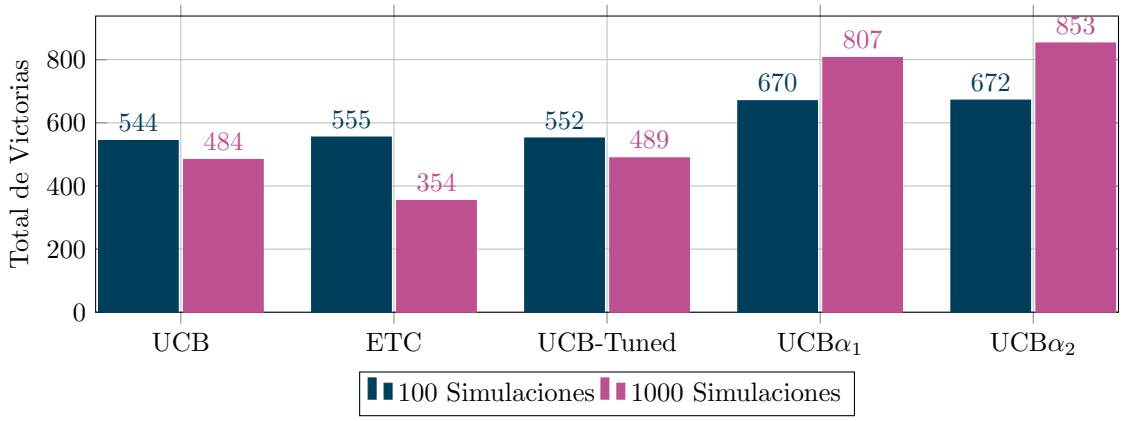


Figura 4.39: Victorias de UCBα_1 y UCBα_2 en Two-Player Free-For-All Zero-Sum

partidas. El porcentaje más alto se da cuando UCBα_1 vence a UCB-Tuned en el 76.33 % de las partidas cuando ABMC está limitado a 1,000 simulaciones.

Two-Player Free-For-All Zero-Sum					
100 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCBα_1	**59.00 ± 1.73**	**59.00 ± 4.62**	**55.33 ± 4.16**		**50.00 ± 2.65**
UCBα_2	**55.00 ± 4.00**	**60.33 ± 1.53**	**58.67 ± 2.89**	**50.00 ± 2.31**	
1000 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCBα_1	**75.00 ± 1.73**	**73.67 ± 4.62**	**76.33 ± 4.16**		44.00 ± 2.65
UCBα_2	**77.00 ± 4.00**	**76.33 ± 1.53**	**75.67 ± 2.89**	**55.33 ± 2.31**	

Tabla 4.36: Porcentaje de Victorias de UCBα_1 y UCBα_2 en Two-Player Free-For-All Zero-Sum

La figura 4.40 muestra de manera gráfica el porcentaje de partidas que UCBα_2 logra ganar, en la misma se puede observar que la política propuesta vence a UCB, ETC y UCB-Tuned sin importar el número de simulaciones a los que se limite ABMC. El porcentaje más bajo de partidas ganadas por UCBα_2 se da en el enfrentamiento con UCB al que logra vencer en 55 % partidas cuando ABMC está limitado en 100 simulaciones. El porcentaje más alto de 77 % se da cuando UCBα_2 vence a UCB con ABMC limitado a 1,000 simulaciones.

La figura 4.40 muestra como las políticas propuestas logran el mismo porcentaje de partidas ganadas al enfrentarse entre sí cuando ABMC está limitado a 100 simulaciones, sin embargo, una

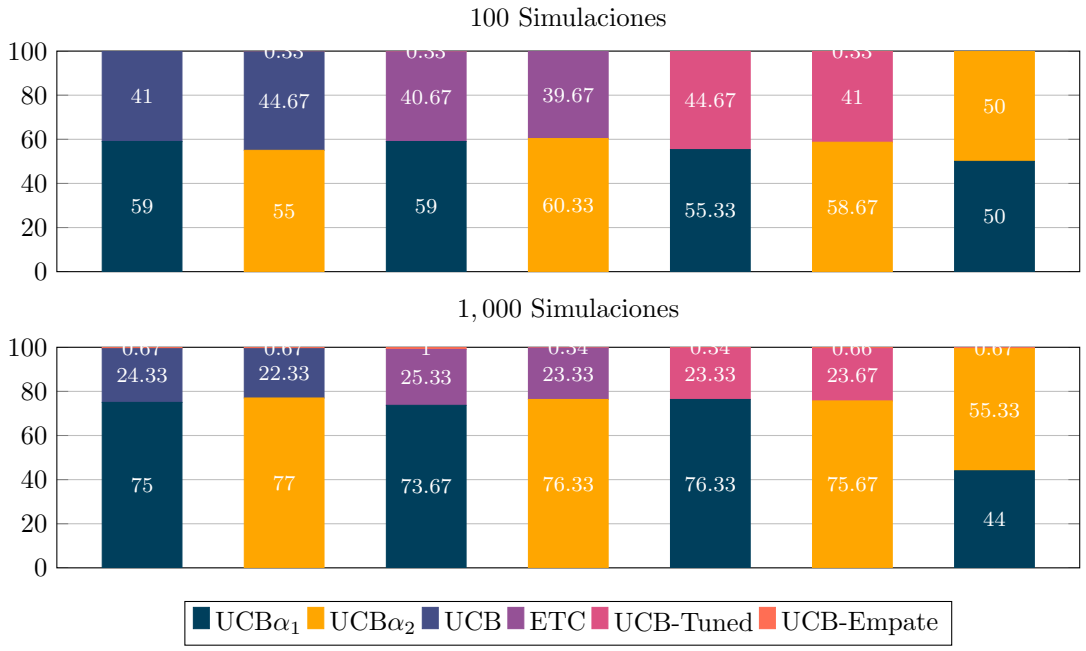


Figura 4.40: Porcentaje de Victorias de UCBα_1 y UCBα_1 en Two-Player Free-For-All Zero-Sum

vez esta limitante se establece en 1,000 simulaciones, se observa que UCBα_2 logra obtener el mayor porcentaje de partidas ganadas sobre UCBα_1.

De acuerdo a los resultados obtenidos la politica con mejor rendimiento en Two-Player Free-For-All Zero-Sum es la política propuesta UCBα_1 ya que logra obtener la mayor cantidad de victorias y el mayor porcentaje de partidas ganadas al enfrentar al resto de políticas.

4.4. Análisis Estadístico De Las Políticas Propuestas

En la tabla 4.37 y figura 4.41 se muestran la cantidad de victorias que cada política logró en cada uno de los juegos con ABMC limitado a 100 y 1,000 simulaciones.

En el comparativo anterior la política ETC resulto tener la mayor cantidad de victorias cuando ABMC estaba limitado a 100 simulaciones, sin embargo, las políticas propuestas obtienen una mayor cantidad de victorias, ya que UCBα_1 obtiene 5,294 victorias, 708 victorias más que ETC, y en el caso de α_2 logra 5,329 victorias que corresponde a 743 victorias por encima de las 4,856 victorias

logradas por ETC.

	100 Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
Atari-Go	497	612	500	**711**	680
Breakthrough Small	494	659	473	**705**	669
Breakthrough Small with Holes	565	571	555	649	**660**
Breakthrough Small Suicide	629	570	584	586	**631**
Knight-Through	461	514	477	771	**777**
Sheep and Wolf	594	558	603	**627**	618
Tic-Tac-Toe Large	594	547	563	575	**622**
Two-Player Free-For-All Zero-Sum	544	555	552	670	**672**
Total	4378	4586	4307	5294	**5329**
Promedio	547.25 $\pm$ 58.77	573.25 $\pm$ 44.14	538.38 $\pm$ 49.05	661.75 $\pm$ 66.48	**666.13 $\pm$ 50.8**
Mediana	554.50	564.00	553.50	659.50	**664.50**
	1,000 Simulaciones				
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
Atari-Go	518	619	453	704	**706**
Breakthrough Small	522	338	540	792	**808**
Breakthrough Small with Holes	533	324	534	**820**	789
Breakthrough Small Suicide	572	585	539	**675**	629
Knight-Through	417	624	439	743	**777**
Sheep and Wolf	612	441	619	**664**	**664**
Tic-Tac-Toe Large	637	221	586	692	**697**
Two-Player Free-For-All Zero-Sum	484	354	489	807	**853**
Total	4295	3506	4199	5897	**5923**
Promedio	536.88 $\pm$ 70.37	438.25 $\pm$ 154.05	524.88 $\pm$ 62.03	737.13 $\pm$ 62.26	**740.38 $\pm$ 77.7**
Mediana	527.50	397.50	536.50	723.50	**741.50**

Tabla 4.37: Total de Victorias de UCB, ETC, UCB-Tuned, UCBα_1 y UCBα_2

Para cuando ABMC está limitado a $1,000$ simulaciones, en el comparativo anterior UCB fue la política que logró la mayor cantidad de victorias, en este nuevo comparativo esta política logro $4,295$ victorias, sin embargo, UCBα_1 logra 5897 victorias y UCBα_2 obtuvo $5,923$, con lo cual superan a UCB y por ende al resto de políticas.

De esta manera si solo se tiene en cuenta el total de victorias, las políticas propuestas en esta tesis tienen el mejor rendimiento en GGP.

Tomando el promedio de victorias que obtuvo cada política, de acuerdo a la tabla 4.37, se observa que las políticas propuestas UCBα_1 y UCBα_2, logran una mayor cantidad de victorias, por ejemplo para el caso de ABMC limitado a 100 las políticas UCBα_1 y UCBα_2 obtienen en promedio 661.75 y 666.13 victorias por juego respectivamente, cantidades superiores a las 573.25 victorias en promedio que logra la política ETC. El mismo fenómeno ocurre cuando ABMC está limitado a $1,000$ simulaciones ya que, la política UCB la cual dio el mejor promedio en el comparativo, en este comparativo obtiene 536.88 victorias en promedio, sin embargo, UCBα_1 obtuvo 737.13 victorias en

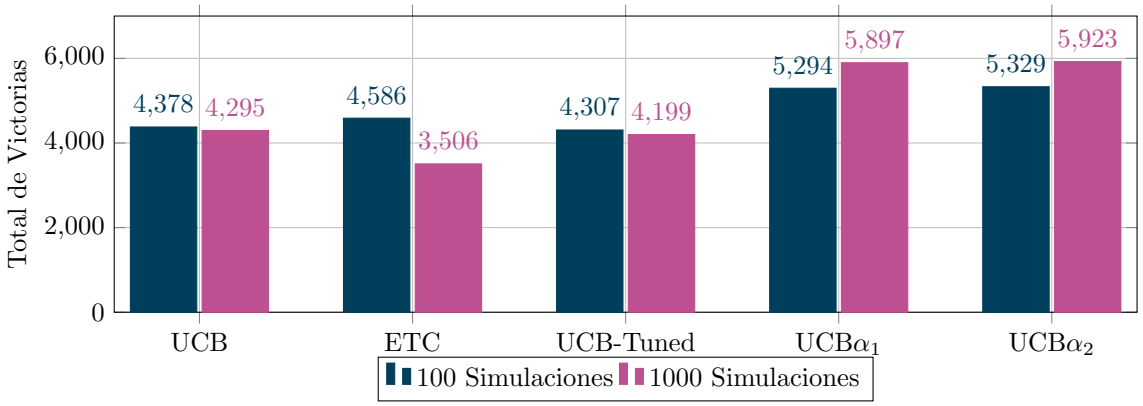


Figura 4.41: Total de Victorias de UCB, ETC, UCB-Tunde, UCBα_1 y UCBα_2

promedio y UCBα_2 obtuvo 741.50 por lo cual las políticas propuestas tienen el mejor promedio de victorias que las políticas propuestas.

La figura 4.42 muestra un diagrama de cajas para las victorias que las políticas obtuvieron cuando ABMC estaba limitado a 100 simulaciones. En esta gráfica se puede observar que las medianas de las dos políticas propuestas son superiores al 100 % de los datos obtenidos por las políticas UCB, ETC y UCB-Tuned, por lo cual se puede considerar que estas políticas propuestas tienen mejor rendimiento que las políticas mencionadas. Respecto a la dispersión de los datos se observa que la política con menor rango es UCBα_2 (si no se toma en cuenta el dato atípico que presenta[2]) y la política ETC sin embargo, las victorias de ETC son inferiores a las de UCBα_2. El rango de datos más grande esta dado por UCBα_1 sin embargo su dato mínino es superior al 75 % de datos de UCB-Tuned y al 50 % de UCB y ETC. En base a la dispersión de datos se puede considerar que UCBα_1 y UCBα_2 tienen el mejor desempeño, el primero por lograr datos superiores a los de las políticas UCB, ETC y UCB-Tuned, y se considera que UCBα_2 tiene mejor desempeño que estas políticas por las mismas razones que UCBα_1 y ser una de las políticas con menor rango lo que indica que el número de victorias que esta política logre será muy similar entre juegos.

La figura 4.43 muestra un diagrama de cajas de las victorias de las políticas del comparativo pero para el caso en el cual ABMC está limitado a $1,000$ simulaciones. En esta gráfica se puede

[2]Se considera dato atípico aquel que es mayor a el tercer cuartil más 1.5 veces el rango intercuartil

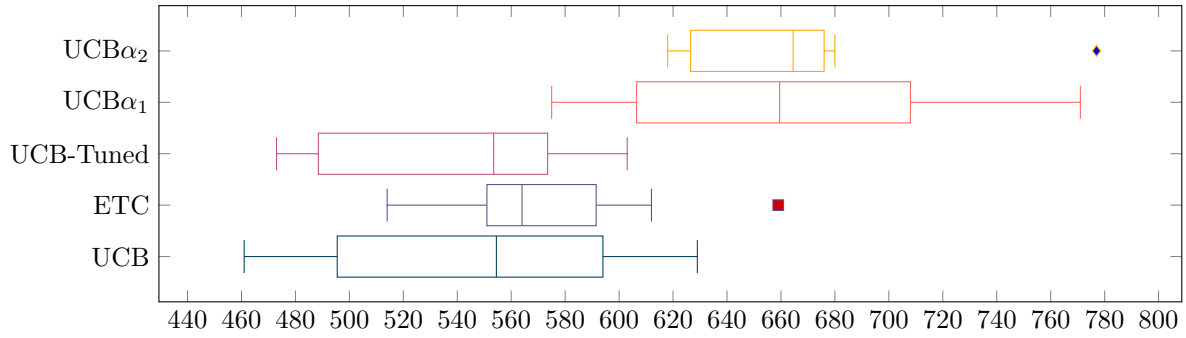


Figura 4.42: Gráfica de cajas para 100 simulaciones

observar que las victorias logradas por las políticas propuestas UCBα_1 y UCBα_2 son superiores a las logradas por UCB, ETC y UCB-Tuned, como se puede observar en los valores mínimos de las políticas propuestas y los valores máximos de las políticas con las que se compararon. Otro punto que observar es que los rangos de las victorias que las políticas obtuvieron se redujeron mostrando unos datos más consistentes, pero como se mencionó las victorias logradas por UCB, ETC y UCB-Tuned son inferiores a las logradas por las políticas propuestas.

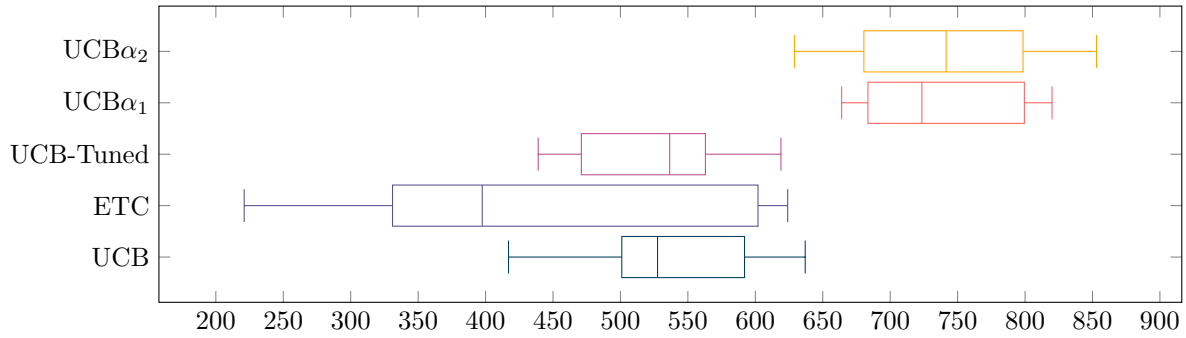


Figura 4.43: Gráfica de cajas para $1,000$ simulaciones

Por lo tanto y de acuerdo con los resultados obtenidos del comparativo, las políticas propuestas UCBα_1 y UCBα_2 tiene un mejor desempeño que las políticas UCB, ETC y UCB-Tuned en General Game Playing por las siguientes razones:

- logran una mayor cantidad de victorias.

- tiene un mayor promedio de victorias por juego.
- el número de victorias obtenidas es consistente entre juegos, esto debido a que las políticas siempre quedaron en los primeros lugares.

Con el fin de reafirmar lo anterior se procedió con la realización de dos pruebas de estadística no paramétrica:

- Prueba de Friedman, procedimiento estadístico usado para determinar la diferencia entre más de dos muestras relacionadas.
- Prueba de los rangos con signo de Wilcoxon, procedimiento estadístico usado para determinar la diferencia entre dos muestras relacionadas.

Estas dos pruebas permitirán determinar que la diferencia entre el número de victorias que las políticas logran en cada uno de los juegos no se debe al alzar sino al desempeño de cada política.

La prueba de Friedman requiere clasificar el desempeño de las políticas en cada uno de los juegos al asignar un valor numérico en este caso dentro de un rango de 1 a 5, asignando 1 a la política que logra el mayor número de victorias y 5 a la política que obtuvo el menor número de victorias, de esta manera se obtuvieron las tablas 4.38 y 4.39 que muestran las clasificaciones de las políticas cuando ABMC está limitado a 100 y $1,000$ simulaciones respectivamente.

	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
Atari-Go	5	3	4	1	2
Breakthrough Small	4	3	5	1	2
Breakthrough Small with Holes	4	3	5	2	1
Breakthrough Small Suicide	2	5	4	3	1
Knight-Through	5	3	4	2	1
Sheep and Wolf	4	5	3	1	2
Tic-Tac-Toe Large	2	5	4	3	1
Two-Player Free-For-All Zero-Sum	5	3	4	2	1
Suma	31	30	33	15	11
Promedio	3.88	3.75	4.13	1.88	1.38

Tabla 4.38: Clasificación de las políticas de acuerdo a la prueba de Friedman en 100 simulaciones

	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
Atari-Go	4	3	5	2	1
Breakthrough Small	4	5	3	2	1
Breakthrough Small with Holes	4	5	3	1	2
Breakthrough Small Suicide	4	3	5	1	2
Knight-Through	5	3	4	2	1
Sheep and Wolf	4	5	3	1.5	1.5
Tic-Tac-Toe Large	3	5	4	2	1
Two-Player Free-For-All Zero-Sum	4	5	3	2	1
Suma	32	34	30	13.5	10.5
Promedio	4.00	4.25	3.75	1.69	1.31

Tabla 4.39: Clasificación de las políticas de acuerdo a la prueba de Friedman en $1,000$ simulaciones

De acuerdo con las tablas 4.38 y 4.39 se observa que las políticas propuestas en esta tesis UCBα_1 y UCBα_2 están mejor clasificadas ya que en promedio dichas políticas obtienen el mejor puntaje en especial la política UCBα_2 incluso se observa que al incrementar el número de simulaciones se incrementa el puntaje de las políticas propuestas.

La prueba de Friedman requiere calcular un estadístico de acuerdo con la siguiente formula:

$$\chi^2_\tau = \frac{12}{nr\,(r+1)} \sum_{j=1}^{r} R_j^2 - 3n\,(r+1)$$

donde n corresponde al número de juegos de tablero, r el número de políticas y R_j es la suma de rangos de la política j, por lo cual para 100 simulaciones se tiene:

$$\chi^2_{100} = \frac{12}{8\,(5)\,(6)} \left(31^2 + 30^2 + 33^2 + 15^2 + 11^2\right) - 3\,(8)\,(6) = \frac{1}{20}\,(3296) - 144 = 20.8$$

y para $1,000$ simulaciones:

$$\chi^2_{1000} = \frac{12}{8\,(5)\,(6)} \left(32^2 + 34^2 + 30^2 + 13.5^2 + 10.5^2\right) - 3\,(8)\,(6) = \frac{1}{20}\,(3372.5) - 144 = 24.625$$

con los estadísticos de Friedman χ^2_{100} y χ^2_{1000} y haciendo uso del valor crítico p de la distribución

chi- cuadrada con $r - 1 = 4$ grados de libertad y un nivel significancia de 0.05 se tiene que:

$$\chi^2_{100} = 20.8 > p = 14.860$$

y

$$\chi^2_{1000} = 24.625 > p = 14.860$$

lo que indica que las diferencias entre los resultados obtenidos por las políticas en los diferentes juegos usados se deben a su rendimiento y no al azar lo que reafirma como superiores a las políticas propuestas.

Las tablas 4.40 y 4.41 muestra los resultados obtenidos de la prueba de Wilcoxon para las políticas UCBα_1 y UCBα_2 al enfrentar al resto de las políticas. En dicha tabla se muestran los rangos positivos (R^+) de las políticas propuestas, los rangos negativos (R^-) de las políticas propuestas, así como el P-valor obtenido. Dado que el P-valor es una probabilidad usada para determinar si existe evidencia suficiente para afirmar si dos muestras son diferentes (en este caso el conjunto victorias de dos políticas) y usando un nivel de significancia de 0.05 podemos afirmar lo siguiente:

- Existen una diferencia estadísticamente significativa entre las victorias obtenidas por las políticas propuestas UCBα_1 y UCBα_2 con respecto a las políticas UCB, ETC y UCB-Tuned ya que el P-valor obtenido es inferior al nivel de significancia 0.05. Debe observarse que existe una excepción al no poder afirmarse que exista una diferencia significativa entre los resultados obtenidos por UCBα_1 y UCB cuando ABMC está limitado a 100 simulaciones, sin embargo, esto cambia conforme se incrementa el número de simulaciones como se muestra en la tabla 4.41 donde el P-valor obtenido es menor a 0.05.

- Los resultados obtenidos por las políticas UCBα_1 y UCBα_2 son estadísticamente similares ya que el P-Valor obtenido es superior al nivel de significancia 0.1 lo que indica que las políticas obtendrán un número de victorias similar si se ejecutaran nuevamente.

La tabla 4.42 muestra un resumen de los resultados obtenidos por la prueba de Wilcoxon para 100 simulaciones y $1,000$ simulaciones. En dicha tabla los resultados mostrados en la diagonal superior

UCBα_1			
VS	R^+	R^-	P-valor
UCB	32.0	4.0	0.054680
ETC	36.0	0.0	0.007812
UCB-Tuned	36.0	0.0	0.007812
UCBα_2	14.0	22.0	≥ 0.2

UCBα_2			
VS	R^+	R^-	P-valor
UCB	36.0	0.0	0.007812
ETC	36.0	0.0	0.007812
UCB-Tuned	36.0	0.0	0.007812
UCBα_1	22.0	14.0	≥ 0.2

Tabla 4.40: Resultados obtenidos por la prueba Wilcoxon para las políticas UCBα_1 y UCBα_2 con ABMC limitado a 100 simulaciones

UCBα_1			
VS	R^+	R^-	P-value
UCB	36.0	0.0	0.007812
ETC	36.0	0.0	0.007812
UCB-Tuned	36.0	0.0	0.007812
UCBα_2	10.5	17.5	≥ 0.2

UCBα_2			
VS	R^+	R^-	P-value
UCB	36.0	0.0	0.007812
ETC	36.0	0.0	0.007812
UCB-Tuned	36.0	0.0	0.007812
UCBα_1	17.5	10.5	≥ 0.2

Tabla 4.41: Resultados obtenidos por la prueba Wilcoxon para las políticas UCBα_1 y UCBα_2 con ABMC limitado a $1,000$ simulaciones

indica un nivel de confianza de 0.9, así mismos los resultados de la diagonal inferior indican un nivel de confianza de 0.95, el símbolo $\bullet$ indica que la política de la fila supera a la política de la columna, el símbolo $\circ$ indica que la política que la columna supera a la política de la fila, una celda vacía indica que no existe evidencia significativa para determinar si una política supera a la otra.

100 Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCB	-			$\circ$	$\circ$
ETC		-		$\circ$	$\circ$
UCB-Tuned			-	$\circ$	$\circ$
UCBα_1		$\bullet$	$\bullet$	-	
UCBα_2	$\bullet$	$\bullet$	$\bullet$		-

$1,000$ Simulaciones					
	UCB	ETC	UCB-Tuned	UCBα_1	UCBα_2
UCB	-			$\circ$	$\circ$
ETC		-		$\circ$	$\circ$
UCB-Tuned			-	$\circ$	$\circ$
UCBα_1	$\bullet$	$\bullet$	$\bullet$	-	
UCBα_2	$\bullet$	$\bullet$	$\bullet$		-

Tabla 4.42: Resumen de la prueba de Wilcoxon test: $\bullet$ indica que la política de la fila supera a la política de la columna. $\circ$ indica que la política de la columna supera a la política de la fila. La diagonal superior indica un nivel de confianza de 0.9 y al diagonal inferior indica un nivel de confianza de 0.95.

De acuerdo a los resultados mostrados por la tabla 4.42 se puede afirmar que las políticas UCBα_1 y UCBα_2 propuestas en esta tesis son significativamente superiores a las políticas UCB, ETC y UCB-Tuned con un nivel de significancia de 0.95 (a excepción del enfrentamiento de UCBα_1

contra UCB en 100 simulaciones donde el nivel de significancia es de 0.9).

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

En esta tesis de investigación se presentaron dos políticas de selección para el método Árbol de Búsqueda Monte Carlo y que están específicamente diseñadas para usarse en General Game Playing: UCBα_1 y UCBα_2. Las políticas UCBα_1 y UCBα_2 están basadas en la política UCB y siguen la idea de la política UCB-Tuned en la cual la constante de exploración es remplazada por una función que controla la etapa de explotación. En lugar de la constante de exploración, UCBα_1 y UCBα_2, hacen uso de dos funciones las cuales fueron determinadas por medio de dos experimentos en el Problema del Bandido Multi-Armado ya que originalmente las políticas UCB y UCB-Tuned se idearon para este ámbito, se decidió realizar los experimentos en este ambiente para determinar el comportamiento de estas políticas en escenarios similares a GGP. En el primer experimento se generaron diversos problemas del bandido de K máquinas (donde $K \in \{2, 10, 20, \cdots, 100\}$) en los cuales, a cada una de las máquinas se le asignó una probabilidad aleatoria, posteriormente se ejecutó UCB con diversos valores para la constante α y se determinó aquellos valores que dieron el mejor rendimiento para cada uno de los valores asignados a K. En el segundo experimento se hizo uso de Recocido Simulado para determinar los mejores valores de α para los diversos problemas del bandido de K máquinas de manera similar al primer experimento. Los resultados obtenidos

en ambos experimentos resultaron ser bastante similares y de estos resultados se generaron dos funciones que aproximaron a valores de α con mejor rendimiento y que se usaron para desarrollar las políticas UCBα_1 y UCBα_2.

Del proceso anterior se determinó lo siguiente:

- Entre mayor número de máquinas menor será la constante α tendiendo a 0, lo que indica que en árbol de juego con un factor de ramificación bastante grande es preferible una política enfocada en la explotación como es el caso del juego Go.
- El valor original de 2 para la constante α de UCB, es un valor bastante alto en comparación a los valores con mejor rendimiento encontrados por el proceso anterior.

Debido a que las políticas UCBα_1 y UCBα_2 están pensadas para GGP, se realizó un comparativo donde se enfrentaron las políticas UCBα_1 y UCBα_2 contra las políticas UCB, UCB-Tuned y ETC una política que se pensó como de política de control al tener una etapa fija de exploración. Para los enfrentamientos se utilizaron 8 diversos juegos de tablero con movimientos por turnos y para dos jugadores, los cuales en conjunto reflejan un escenario GGP. Con el fin de realizar el comparativo lo más justo posible se usaron las mismas características del método Árbol de Búsqueda Monte Carlo en este caso la condición de parada es por número de simulaciones y para la elección de movimiento se determinó usar el nodo con el mayor promedio de victorias. Finalmente, con el fin de determinar si el rendimiento de las políticas se mantiene constante al incrementar el número de simulaciones, se realizó el comparativo con Árbol de Búsqueda Monte Carlo limitado a 100 simulaciones y posteriormente a $1,000$ simulaciones.

Los resultados muestran que las políticas propuestas obtienen el mayor número de victorias en comparación a UCB, UCB-Tuned y ETC además muestran que su rendimiento es consistente ya que al menos una de las políticas propuestas logra estar entre los dos primeros lugares si el número de simulaciones a los que está limitado Árbol de Búsqueda Monte Carlo se incrementa incluso su rendimiento parece aumentar y lo que es más importante el rendimiento de las políticas no se ve alterado si se cambia de juego lo que se espera en un escenario GGP.

Para confirmar los resultados obtenidos se realizó un análisis estadístico que incluyó la prueba

Friedman y la prueba de rangos con signo de Wilcoxon las cuales confirmaron la superioridad de las políticas.

El éxito de las políticas UCBα_1 y UCBα_2 radica en que a diferencia de UCB, estas políticas propuestas se adaptan a la estructura del árbol de juego. UCB trabaja de la siguiente manera: en cada nodo padre del árbol de juego, UCB requiere explorar los nodos hijos con el fin de determinar aquel nodo prometedor el cual debe explorarse, la etapa de exploración es controlada por medio de una constante α la cual dependiendo de su valor será el número de iteraciones del Árbol de Búsqueda Monte Carlo que se dedicarán a explorar los nodos y qué tan sensible será UCB para determinar un nodo prometedor de un nodo que no lo es. El valor comúnmente usado para la constante α y con la cual originalmente fue ideada UCB es de 2, sin embargo, al aplicarse a un árbol de juego la etapa de exploración será controlada por $\alpha = 2$ sin importar si se aplica a un nodo padre con 2 nodos hijos como a un nodo padre con 100 nodos hijos por lo cual pueden darse casos donde UCB requiera de más iteraciones para identificar los nodos prometedores en el árbol de juego lo que repercute en su rendimiento. UCBα_1 y UCBα_2 abordan esta problemática de la siguiente manera; en lugar de tener un valor fijo para la constante que controla la exploración que se use para todo el árbol de juego, se tiene una función que depende del número de nodos hijos con la cual se determina el valor adecuado para la constante de exploración lo que permite al método Árbol de Búsqueda Monte Carlo usar solo la cantidad necesaria de iteraciones para identificar los nodos prometedores y dejar al resto de iteraciones para explotar dichos nodos. Lo anterior significa que las políticas UCBα_1 y UCBα_2 identifican movimientos o jugadas que conducen a victorias en un menor número de iteraciones (lo que es bastante útil cuando se tienen pocos recursos) de las que requeriría UCB, por lo cual al enfrentar UCBα_1 y UCBα_2 contra UCB son las políticas propuestas las que salen victoriosas.

Por lo tanto y de acuerdo con los resultados obtenidos y lo antes expuesto se considera que se ha cumplido con el objetivo de esta tesis al desarrollar dos políticas de selección UCBα_1 y UCBα_2 adecuadas a General Game Playing las cuales están basadas en las políticas UCB y UCB-Tuned y que presentan mejor rendimiento que estas.

5.2. Trabajo Futuro

En esta tesis de investigación se propusieron dos políticas de selección para General Game Playing: UCBα_1 y UCBα_2, para determinar el rendimiento de estas políticas propuestas se compararon con la política comúnmente usada UCB, su variante UCB-Tuned y la política ETC. Con el fin de hacer dicho comparativo lo más justo posible se utilizó una versión *simple* del método Árbol de Búsqueda Monte Carlo la cual no incluye otra técnicas que han aumentado el rendimiento de este método como Urgente Jugar Primero, Movimientos Decisivos y Anti-Decisivos, Estimación del Valor de Acción Rápida y otras de las técnicas descritas en el estado de arte de este documento (ver capitulo 2). Un trabajo inmediato sería la implementación de estas técnicas para determinar aquellas que aumentan el rendimiento de UCBα_1 y UCBα_2, otro punto interesante sería determinar aquella combinación de UCB con estas técnicas cuyo rendimiento sea equivalente a las políticas propuestas en esta tesis.

Muchas de las técnicas mencionadas en el párrafo anterior hacen uso de una constante como en el caso EVARG con lo cual se puede utilizar un enfoque similar al usado en esta tesis, sin embargo, debido a que estas técnicas son ajenas al problema del bandido, se puede hacer uso de árboles sintéticos con diferentes factores de ramificación y niveles de profundidad para determinar el comportamiento de estas constantes y su impacto en el rendimiento de Árbol de Búsqueda Monte Carlo.

Las políticas propuestas en estas tesis hacen uso del número de nodos hijos del nodo padre para determinar la cantidad de exploración que deben usar, lo que da un indicio de una relación existente entre las políticas de selección y el factor de ramificación, la identificación de esta relación se deja como trabajo futuro.

Como se mencionó las políticas propuestas usan el factor de ramificación, sin embargo, otro elemento del árbol de juego es el nivel de profundidad, una posible línea de investigación es determinar si política de selección basada en el nivel de profundidad o su combinación con el factor de ramificación puede presentar un mejor rendimiento que las políticas propuestas o UCB. Incluso se puede realizar un trabajo similar al propuesto en esta tesis usando arboles sintéticos con un mismo

factor de ramificación, pero diferentes niveles de profundidad y generar una variante de UCB que remplace la constante de exploración por una función que dependa del nivel de profundidad.

De los resultados obtenidos del comparativo de UCBα_1 y UCBα_2 contra UCB, UCB-Tuned y ETC, sin embargo, la política ETC en el ámbito del problema del bandido se considera con un rendimiento bajo por lo cual se consideró como una política de control, pero según se pudo observar, en ciertos casos la política ETC presenta el mejor rendimiento en juegos como en el caso de Atari-Go, Breakthrough with Holes y Knight-Through, por lo cual se deja como trabajo futuro investigar más a fondo su comportamiento e identificar si puede mejorarse su rendimiento en GGP.

Apéndices

Apéndices A

Juegos de Tablero

A.1. Atari Go 7×7

Versión del juego de Go jugado en un tablero de 7×7 en la cual el primer jugador en realizar una captura gana.

Elementos

- Un tablero vació de 7×7
- Varias *piedras* blancas y negras

Objetivo

- Ser el primer jugador en capturar un grupo de piedras del contrincante.

Reglas

- En su turno el jugador coloca una piedra de su color en una intersección de las líneas del tablero.

- Un jugador puede capturar un grupo de piedras cuando al colocar una de sus piedras rodea totalmente a dicho grupo siempre y cuando este grupo no tenga ningún hueco libre en su interior.

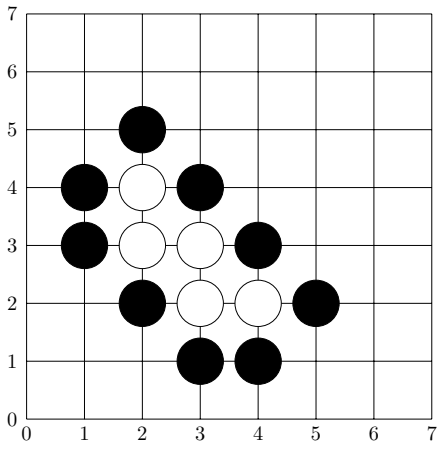


Figura A.1: Atari Go

A.2. Breakthrough 6×6

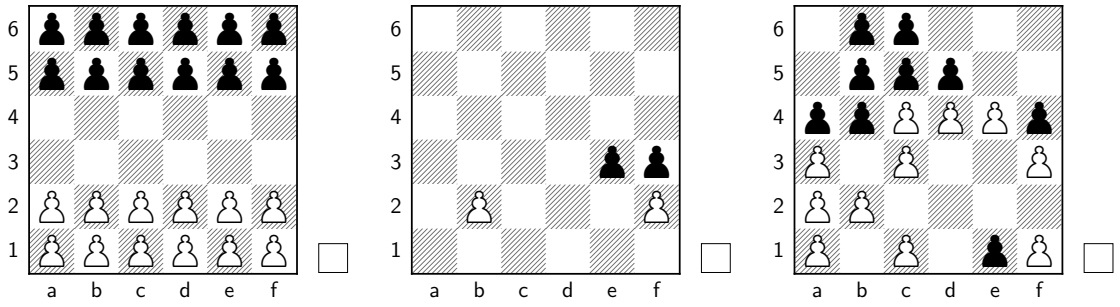
Juego de tablero que hace uso de los peones del ajedrez y cuyo objetivo es penetrar (breakthrough) las defensas del jugador contrario.

Elementos

- Un tablero de 6×6
- 12 peones para cada jugador

Objetivo

- Ser el primer jugador en llevar un peón a la fila inicial del contrincante.



(a) Tablero Inicial (b) Movimientos (c) Fin de la partida

Figura A.2: Breakthrough 6×6

Reglas

- Al inicio del juego se colocan los peones de cada jugador en las dos primeras filas frente a ellos (ver figura A.2a).
- En cada turno el jugador tiene dos posibilidades: Mover o Capturar
  - Mover: El jugador es libre de mover un peón una casilla hacia adelante o una casilla en diagonal siempre y cuando la casilla este vacía. En la figura A.2b el peon en $b2$ puede moverse a $a3$, $b3$ o $c3$.
  - Capturar: El jugador puede capturar un peón enemigo si dicho peón esta en una casilla diagonal a la de su peón, este movimiento es similar al realizado en el juego de ajedrez. En la figura A.2b el peón en $f2$ no puede moverse a $f3$ pero puede capturar al peón enemigo en $e3$
- Los peones no pueden retroceder únicamente avanzar

A.3. Breakthrough with Holes 6×6

Breakthrough with Holes 6×6 es una versión de Breakthrough en donde las casillas $c6, f6, c3, f3$ no pueden ser usadas por ninguno de los jugadores (ver A.3).

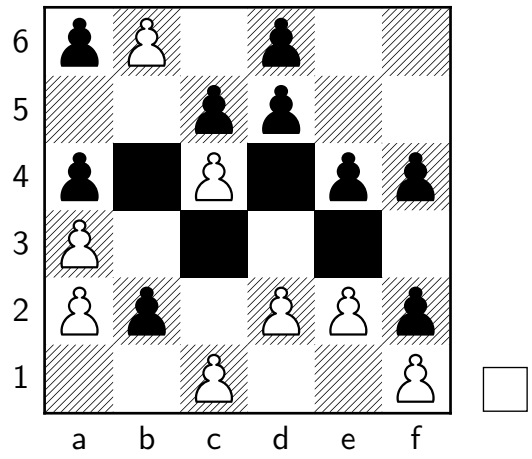


Figura A.3: Breakthrough with Holes 6×6

A.4. Breakthrough Suicide 6×6

Variante de Breakthrough 6×6 donde el primer jugador en llevar un peón a la primera fila contraria pierde.

A.5. Knight-Through

Juego similar a Breakthrough el cual usa caballos en lugar de peones.

Elementos

- Un tablero de 8×8
- 16 caballos para cada jugador

Objetivo

- Ser el primer jugador en llevar un caballo a la fila inicial del contrincante.

Reglas

- Al inicio del juego se colocan los caballos de cada jugador en las dos primeras filas frente a ellos (ver figura A.2a).
- En cada turno el jugador puede mover su caballo o capturar a un caballo enemigo de la misma forma que en el ajedrez
- Los caballos no pueden retroceder únicamente avanzar

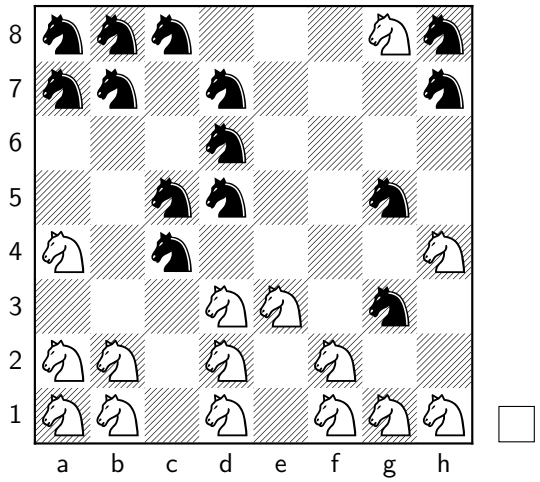


Figura A.4: Knight-Through

A.6. Sheep and Wolf

Juego de tablero para dos jugadores donde los jugadores no tiene movimientos similares.

Elementos

- Un tablero de ajedrez
- 4 peones blancos que representan ovejas

- 1 peón negro que representa al lobo

Objetivo

- El objetivo del lobo es cruzar el tablero de un lado al otro.
- El objetivo de las ovejas es impedir que el lobo logre cruzar el tablero al bloquear sus movimientos

Reglas

- Al inicio las ovejas se colocaran en las casillas $a2$, $a4$, $a6$, $a8$, el lobo sera colocado en la casilla $h5$ ver figura A.5.
- Las ovejas se mueven como en las damas solo una casilla en diagonal únicamente hacia adelante
- El lobo puede moverse una casilla en diagonal ya sea hacia adelante o hacia atrás.
- Ninguna pieza puede capturar o saltar a otra
- Únicamente se puede mover una oveja a la vez
- Empieza a mover el lobo.

A.7. Tic-Tac-Toe Large

Variante de Tic-Tac-Toe (juego del gato o tres en raya) jugado en un tablero de $5x5$.

Elementos

- Un tablero vació de 5×5

Objetivo

- Ser el primer jugador en lograr una línea recta o diagonal de 5 símbolos.

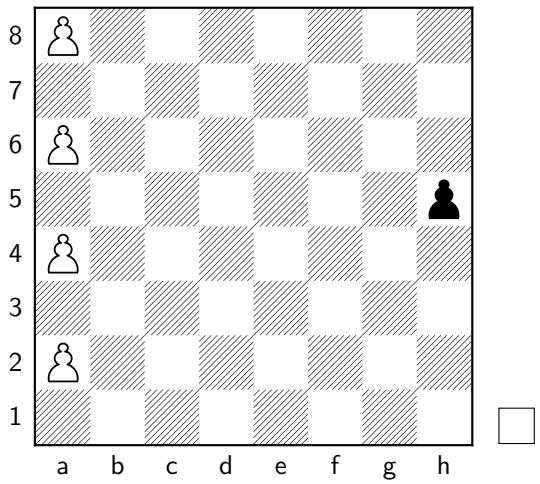


Figura A.5: Sheep and Wolf

Reglas

- En su turno el jugador puede colocar su símbolo en cualquier casilla no ocupada de el tablero.

A.8. Two-Player Free-For-All Zero-Sum

Juego de tablero para dos jugadores donde el jugador que capture mas piezas enemigas en 30 turnos es quien gana.

Elementos

- Un tablero vació de 6×6
- Varias piezas para cada jugador

Objetivo

- Capturar mas piezas que el contrincante en 30 turnos.

Reglas

- El jugador en turno puede agregar una nueva pieza, mover una pieza que ya esta en el tablero o capturar una pieza del contrincante.
- El jugador puede agregar una nueva pieza desde su pila de piezas moviéndose como el caballo en el ajedrez (ver figura A.6).
- El jugador puede mover alguna de sus piezas que ya estén en el tablero como si se tratara de un caballo en el ajedrez.
- El jugador puede capturar otra pieza enemiga con una propia al moverse como el rey en el ajedrez.

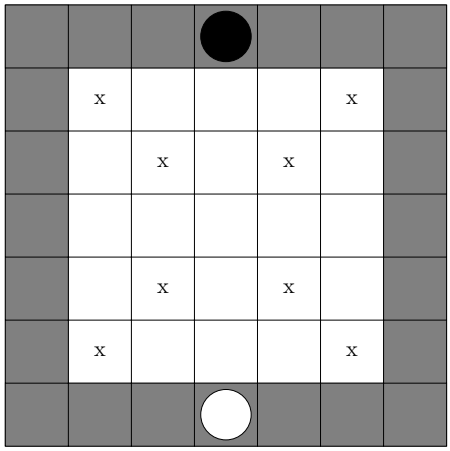


Figura A.6: Two-Player Free-For-All Zero-Sum

Apéndices B

Demostración de Teoremas

Teorema (Arrepentimiento Esperado General para UCB). *Si la política UCB es ejecutada en K máquinas ($K > 1$) con distribuciones de recompensas arbitrarias $(p_1, \cdots p_K)$, $p_i \in [0, 1]$, el arrepentimiento esperado después de cada ronda T es a lo mucho*

$$R(T) \leq \sum_{i:\Delta_i>0} 4\alpha \frac{\ln T}{\Delta_i} + \frac{2\alpha}{\alpha - 1}\Delta_i \qquad \alpha > 0$$

Demostración. Tomado de [32]. Supóngase, sin perder generalidad, que la máquina 1 es la óptima. Entonces, la máquina $i \neq 1$ será activada únicamente en dos casos: que las máquinas 1 y i no han sido los suficientemente muestreadas como para poder distinguir entres sus medias, o el límite de confianza superior dado por la desigualdad de Hoeffding falla ya sea para la máquina 1 o para la máquina i. Se comenzará con limitar la probabilidad de que una máquina subóptima se active debido a muestreo insuficiente.

Supóngase que se tiene dos eventos A_t y B_t

A_t $\hat{\mu}_{i,T_i} \leq \mu_i + \sqrt{\frac{\alpha \ln t}{2T_i}}$

B_t $\hat{\mu}_{1,T_1} \leq \mu_1 + \sqrt{\frac{\alpha \ln t}{2T_1}}$

Se desea limitar las probabilidades de los complementos de los eventos A_t y B_t ocurran. A_t falla

cuando

$$\hat{\mu}_{i,T_i} - \mu_i > \sqrt{\frac{\alpha \ln t}{2T_i}}$$

Aplicando la desigualdad de Hoeffding se tiene

$$\mathbb{P}\left(A_t^c\right) = \mathbb{P}\left(\hat{\mu}_{i,T_i} - \mu_i > \epsilon\right) \leq \exp\left(-2\epsilon^2 t\right)$$

al asignar el valor limite $\epsilon = \sqrt{\frac{\alpha \ln t}{2T_i}}$

$$\mathbb{P}\left(\hat{\mu}_{i,T_i} - \mu_i > \sqrt{\frac{\alpha \ln t}{2T_i}}\right) \leq \exp\left(\frac{-2t\alpha \ln t}{2T_i}\right) \tag{B.1}$$

$$= \exp\left(\frac{-t\alpha \ln t}{T_i}\right) \tag{B.2}$$

$$\leq \exp\left(\frac{-t\alpha \ln t}{t}\right) \tag{B.3}$$

$$= e^{\alpha \ln t} \tag{B.4}$$

$$= t^{-a} \tag{B.5}$$

La declaración y la justificación es idéntica para el complemento del evento B_t.

Regresando a la tarea de acotar el número de activaciones de máquinas subóptimas. Una máquina subóptima i es activada únicamente si su límite de confianza superior limita al de la máquina 1, lo que significa que

$$\hat{\mu}_{i,T_i} + \sqrt{\frac{\alpha \ln t}{2T_i}} \geq \hat{\mu}_{1,T_1} + \sqrt{\frac{\alpha \ln t}{2T_1}} \tag{B.6}$$

Suponiendo que tanto A_t y B_t se cumplen. En este caso, la máquina subóptima i es activada debido a muestreo insuficiente. Dado que A_t se ha asumido ser verdadera, se genera el siguiente limite

$$\mu_i + 2\sqrt{\frac{\alpha \ln t}{T_i}} \geq \hat{\mu}_{i,T_i} + \sqrt{\frac{\alpha \ln t}{2T_i}} \tag{B.7}$$

Asumiendo de igual forma que B_t se cumple, se puede acotar el lado derecho de (B.6)

$$\hat{\mu}_{1,T_1} + \sqrt{\frac{\alpha \ln t}{2T_1}} \geq \mu_1 \tag{B.8}$$

Encadenando (B.7) y (B.8) se tiene que

$$\mu_i + 2\sqrt{\frac{\alpha \ln t}{T_i}} \geq \mu_1$$

$$\sqrt{\frac{\alpha \ln t}{T_i}} \geq \frac{\mu_1 - \mu_i}{2}$$

Definiendo $\Delta_i = \mu_1 - \mu_i$ se tiene

$$\sqrt{\frac{\alpha \ln t}{T_i}} \geq \frac{\Delta_i}{2}$$

$$T_i \leq 4\Delta_i^{-2}\alpha \ln t \leq 4\Delta_i^{-2}\alpha \ln T$$

De esta manera, cuando A_t y B_t se cumplen, se activa la máquina subóptima i a lo mucho $\Delta_i^{-2}\alpha \ln T$ veces.

Recalcando que solo se activa la máquina i si se ha muestreado insuficientemente (menos que $\Delta_i^{-2}\alpha \ln T$) o que los eventos A_t o B_t fallan. Para cualquier máquina i, el número esperado de veces que sea activada hasta la ronda T siguiendo UCB es:

$$\mathbb{E}(T_i) = \sum_{t=1}^{T} \mathbb{E}[1(I_t = 1)] \tag{B.9}$$

$$\leq 4\alpha\Delta_i^{-2} \ln T + \sum_{t=1}^{T} \mathbb{E}[1\{A_t^c \cup B_t^c\}] \tag{B.10}$$

$$\leq 4\alpha\Delta_i^{-2} \ln T + \sum_{t=1}^{T} (\mathbb{E}[1\{A_t^c\}] + \mathbb{E}[1\{B_t^c\}]) \tag{B.11}$$

$$\leq 4\alpha\Delta_i^{-2} \ln T + \sum_{t=1}^{T} \left(t^{-\alpha} + t^{-\alpha}\right) \tag{B.12}$$

$$= 4\alpha\Delta_i^{-2} \ln T + 2\sum_{t=1}^{T} t^{-\alpha} \tag{B.13}$$

dado que

$$\sum_{t=1}^{T} t^{-\alpha} \leq 1 + \int_1^{\infty} x^{-a} dx = 1 + \frac{-1}{1-\alpha} = \frac{-\alpha}{1-\alpha}$$

por lo tanto

$$\mathbb{E}(T_i) \leq 4\alpha\Delta_i^{-2} \ln T + \frac{2\alpha}{\alpha - 1}$$

Al sumar todos las máquinas suboptimas:

$$R(T_i) \leq \sum_{i \neq 1} \Delta_i \mathbb{E}(T_i)$$

$$R(T_i) \leq \sum_{i:\Delta_i > 0} 4\alpha \frac{\ln T}{\Delta_i} + \frac{2\alpha}{\alpha - 1}\Delta_i$$

□

Teorema 3 (Arrepentimiento Esperado para UCB con $\alpha = 2$)**.** *Si la política UCB es ejecutada en K máquinas ($K > 1$) con distribuciones de recompensas arbitrarias $(p_1, \cdots p_K)$, $p_i \in [0, 1]$, el*

arrepentimiento esperado en la ronda T es a lo mucho

$$R(T) \leq \sum_{i:\Delta_i>0} 8\frac{\ln T}{\Delta_i} + \frac{\pi^2}{3}\Delta_i$$

Demostración. Partiendo de (B.13), con $\alpha = 2$

$$\begin{aligned}
\mathbb{E}(T_i) &\leq 8\Delta_i^{-2}\ln T + 2\sum_{t=1}^{T} t^{-2} \\
&\leq 8\Delta_i^{-2}\ln T + 2\sum_{t=1}^{\infty} t^{-2} \\
&= 8\Delta_i^{-2}\ln T + 2\frac{\pi^2}{6} \\
&= 8\Delta_i^{-2}\ln T + \frac{\pi^2}{3}
\end{aligned}$$

Al sumar todos las máquinas suboptimas:

$$R(T) \leq \sum_{i\neq 1} \Delta_i \mathbb{E}(T_i)$$

$$R(T) \leq \sum_{i:\Delta_i>0} 8\alpha\frac{\ln T}{\Delta_i} + \frac{\pi^2}{3}\Delta_i$$

□

Apéndices C

Publicaciones

C.1. A comparison between UCB and UCB-Tuned as selection policies in GGP

Journal of Intelligent & Fuzzy Systems xx (20xx) x–xx
DOI:10.3233/JIFS-179052
IOS Press

A comparison between UCB and UCB-Tuned as selection policies in GGP

Iván Francisco-Valencia*, José Raymundo Marcial-Romero and Rosa María Valdovinos-Rosas
Facultad de Ingeniería, Universidad Autónoma del Estado de México, Cerro de Coatepec S/N Ciudad Universitaria C.P. 50100. Toluca, Estado de México

Abstract. In this paper, we present a comparative analysis of two selection policies in the General Game Playing (GGP) context: Upper Confidence Bound (UCB) and Upper Confidence Bound Tuned (UCB-Tuned). The aim of the analysis is to identify which policy has the best performance in terms of victories in the GGP domain, a measure used in most of literature with other policies. In order to carry out the comparison, two agents were programmed using the GGP-base framework and the Monte Carlo Tree Search (MCTS) method. The games Breakthrough, Knightthrough and Connect Four were used as experimental scenarios, not compared previously to the best of our knowledge. The results show that UCB-Tuned is better when less than 100 simulations are used in MCTS; however, when 1000 simulations are used, both policies have similar performance.

Keywords: General Game Playing, Upper Confidence Bound, Upper Confidence Bound Tuned, policies

Uncorrected Author Proof

1. Introduction

One of the aims of Artificial Intelligence (AI) has been the development of intelligent agents capable of playing board game at the same level -or even better- than humans [4]. The challenge in developing playing agents drift that they must possess characteristics pertaining to human intelligence, such as deduction, reasoning, problem resolution, intelligent search, knowledge representation, planning, learning, creativity, perception and natural language processing among others [21]. AI has proponed agents capable of playing at the level of human champions in specific games, like Chinook [19] (there is an strategy that allows the player to win regardless the plays by its opponent [18]), Deep Blue for Chess [6] and Dark Knight for Banqi or Chinese Chess [13].

*Corresponding author. Iván Francisco-Valencia, Facultad de Ingeniería, Universidad Autónoma del Estado de México, Cerro de Coatepec S/N Ciudad Universitaria C.P. 50100. Toluca, Estado de México. E-mail: if.valencia19@gmail.com.

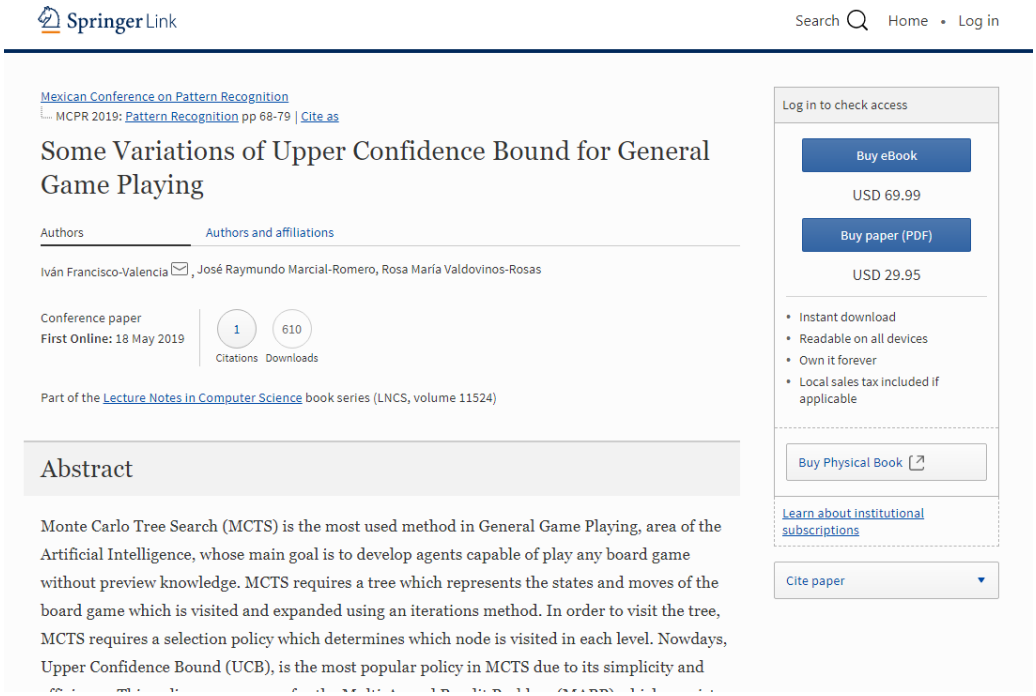
The field of AI that has focused its efforts in developing intelligent agents capable of playing any kind of game is General Game Playing (GGP). In GGP, agents must be able to develop their own playing strategies autonomously, without human intervention, having played the game previously and just considering the rule given at the start of the game (often using the Game Description Language) [4, 11, 12, 21].

Although the agents in GGP cannot be compared to specialized game agents in terms of performance, their use is higher as they can perform in different domains, even the techniques developed in GGP can be used in other areas such as business process, electronic business, military operations, among others [11, 12].

An important result in the GGP area has been the implementation of the Monte Carlo Tree Search method (MCTS) proposed by Finnsson [10] and used at the CadiaPlayer agent which have won the GGP international competition three times. MCTS method defined the state of the art of agents in

1064-1246/18/$35.00 © 2018 – IOS Press and the authors. All rights reserved

C.2. Some Variations of Upper Confidence Bound for General Game Playing



Springer Link

Search Home • Log in

Mexican Conference on Pattern Recognition

MCPR 2019: Pattern Recognition pp 68-79 | Cite as

Some Variations of Upper Confidence Bound for General Game Playing

Authors Authors and affiliations

Iván Francisco-Valencia, José Raymundo Marcial-Romero, Rosa María Valdovinos-Rosas

Conference paper
First Online: 18 May 2019

1 Citations 610 Downloads

Part of the Lecture Notes in Computer Science book series (LNCS, volume 11524)

Log in to check access

Buy eBook

USD 69.99

Buy paper (PDF)

USD 29.95

- Instant download
- Readable on all devices
- Own it forever
- Local sales tax included if applicable

Buy Physical Book

Learn about institutional subscriptions

Cite paper

Abstract

Monte Carlo Tree Search (MCTS) is the most used method in General Game Playing, area of the Artificial Intelligence, whose main goal is to develop agents capable of play any board game without preview knowledge. MCTS requires a tree which represents the states and moves of the board game which is visited and expanded using an iterations method. In order to visit the tree, MCTS requires a selection policy which determines which node is visited in each level. Nowdays, Upper Confidence Bound (UCB), is the most popular policy in MCTS due to its simplicity and

Some Variations of Upper Confidence Bound for General Game Playing

Iván Francisco-Valencia( ), José Raymundo Marcial-Romero, and Rosa María Valdovinos-Rosas

Facultad de Ingeniería, Universidad Autónoma del Estado de México, Cerro de Coatepec S/N Ciudad Universitaria, 50100 Toluca, Estado de México, Mexico
if.valencia@hotmail.com, {jrmarcialr,rvaldovinosr}@uaemex.mx

Abstract. Monte Carlo Tree Search (MCTS) is the most used method in General Game Playing, area of the Artificial Intelligence, whose main goal is to develop agents capable of play any board game without preview knowledge. MCTS requires a tree which represents the states and moves of the board game which is visited and expanded using an iterations method. In order to visit the tree, MCTS requires a selection policy which determines which node is visited in each level. Nowdays, Upper Confidence Bound (UCB), is the most popular policy in MCTS due to its simplicity and efficiency. This policy was propose for the Multi-Armed Bandit Problem (MABP) which consists in set of slot machines each of which has a certain probability of give a reward. The goal is to maximize the accumulative reward that is obtained when a machine is played in a series of rounds. Other policy proposed for MCTS is Upper Confidence Bound$_{\sqrt{\cdot}}$ (UCB$_{\sqrt{\cdot}}$) whose goal is to identify the machine with the highest probability to give a reward. This paper shows a comparative between five modifications of UCB and one of UCB$_{\sqrt{\cdot}}$, this comparative has the goal of finding a policy which be able to identify the optimal machine as quickly as possible, this goal in MCTS is equals to identify the node with the highest probability to leading to a victory. The results show that some policies find the optimal machine before UCB, however, with 10,000 rounds UCB is the policy who plays the optimal machine more often.

Keywords: General game playing · Selection policy · Upper confidence bound

1 Introduction

For Bjornsson and Finnsson [5], General Game Playing (GGP) is the area of Artificial Intelligence of which the objective is to create intelligent agents who can learn, automatically, how to play a wide variety of board games, based only on the descriptions of the rules of the games. The foregoing implies that without prior knowledge about the game and while playing, the agent must be able to

© Springer Nature Switzerland AG 2019
J. A. Carrasco-Ochoa et al. (Eds.): MCPR 2019, LNCS 11524, pp. 68–79, 2019.
https://doi.org/10.1007/978-3-030-21077-9_7

C.3. Tuning Upper Confidence Bound On Large Number of Slot Machines

Noname manuscript No.
(will be inserted by the editor)

Tuning Upper Confidence Bound On Large Number of Slot Machines

Iván Francisco-Valencia[1] · José Raymundo Marcial-Romero[1] · Rosa María Valdovinos-Rosas[1]

Received: date / Accepted: date

Abstract The Multi-Armed Bandit Problem (MABP) consists on a set of slot machines that must be allocated between competing (alternative) choices in a way that maximizes their expected gain and minimizing accumulated regret. Upper Confidence Bound (UCB) is the most used policy in the MABP due primarily to its expected regret that grows logarithmically over number of rounds. For that, UCB uses an index that consists in the sum of two terms: the first one is related with exploitation that only requires the average of rewards and the second one related with exploration that is associated with a confidence interval of average of the rewards. UCB in the second term uses a constant c which controls exploration. Upper Confidence Bound Tuned (UCB-T) is a policy that replaces c by the minimum value between $\frac{1}{4}$ and a sample variance plus a confidence bound. UCB-T has shown better performance than UCB in experimental results. In this paper a new policy is proposed: Upper Confidence Bound Exponential (UCB-exp) as a variant of UCB that replaces c by an exponential function which depends on the number of available slot machines. The results show how UCB-exp outperforms to UCB and UCB-T in terms of accumulated regret when large number of slot machines are considered.

Keywords Multi-Armed Bandit Problem · Upper Confidence Bound · Exponential Regression

Iván Francisco-Valencia
E-mail: if.valencia@hotmail.com

José Raymundo Marcial-Romero
E-mail: jrmarcialr@uaemex.mx

Rosa María Valdovinos-Rosas
E-mail: rvaldovinosr@uaemex.mx

[1]Facultad de Ingeniería, Universidad Autónoma del Estado de México, Toluca, Estado de México, México

Bibliografía

[1] Broderick Arneson, Ryan B Hayward, and Philip Henderson. Monte carlo tree search in hex. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):251–258, 2010.

[2] Jean-Yves Audibert and Sébastien Bubeck. Minimax policies for adversarial and stochastic bandits. In *COLT*, pages 217–226, Montreal, Canada, June 2009.

[3] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Tuning bandit algorithms in stochastic environments. In Marcus Hutter, Rocco A. Servedio, and Eiji Takimoto, editors, *Algorithmic Learning Theory*, pages 150–165, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 322–331, 1995.

[5] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[6] Yngvi Bjornsson and Hilmar Finnsson. Cadiaplayer: A simulation-based general game player. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(1):4–15, 2009.

[7] Yngvi Björnsson and Stephan Schiffel. Comparison of GDL reasoners. In *Proceedings of the IJCAI-13 Workshop on General Game Playing (GIGA'13)*, pages 55–62, 2013.

[8] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton.

A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[9] Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1):57 – 83, 2002.

[10] Tristan Cazenave. Generalized Rapid Action Value Estimation. In *24th International Conference on Artificial Intelligence*, pages 754–760, Buenos Aires, Argentina, July 2015.

[11] Tristan Cazenave. Playout policy adaptation for games. In Aske Plaat, Jaap van den Herik, and Walter Kosters, editors, *Advances in Computer Games*, pages 20–28, Cham, 2015. Springer International Publishing.

[12] Tristan Cazenave. Playout policy adaptation with move features. *Theoretical Computer Science*, 644:43–52, 2016.

[13] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2249–2257. Curran Associates, Inc., 2011.

[14] G.M.J.B. Chaslot, J.T. Saito, Bruno Bouzy, J.W.H.M. Uiterwijk, and H.J. van den Herik. Monte-carlo strategies for computer go. In Pierre-Yves Schobbens, Win Vanhoof, and Gabriel Schwanen, editors, *BNAIC'06: Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence*, pages 83–91. University of Namur, January 2006. BNAIC'06: Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence University of Namur Namen ; BNAIC'06 ; Conference date: 05-10-2006 Through 06-10-2006.

[15] Guillaume Chaslot, Christophe Fiter, Jean-Baptiste Hoock, Arpad Rimmel, and Olivier Teytaud. Adding expert knowledge and exploration in monte-carlo tree search. In *Advances in Computer Games*, pages 1–13. Springer, 2009.

[16] Guillaume Maurice Jean-Bernard Chaslot Chaslot. *Monte-Carlo Tree Search*. PhD thesis, Maastricht University, 2010.

[17] Guillaume MJ-B Chaslot, Mark HM Winands, Istvan Szita, and H Jaap van den Herik. Cross-entropy for monte-carlo tree search. *Icga Journal*, 31(3):145–156, 2008.

[18] B. E. Childs, J. H. Brodeur, and L. Kocsis. Transpositions and move groups in monte carlo tree search. In *2008 IEEE Symposium On Computational Intelligence and Games*, pages 389–395, 2008.

[19] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. (Jeroen) Donkers, editors, *Computers and Games*, pages 72–83, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[20] Hilmar Finnsson and Yngvi Björnsson. Simulation-based approach to general game playing. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 1*, AAAI'08, page 259–264. AAAI Press, 2008.

[21] Iván Francisco-Valencia, José Raymundo Marcial-Romero, and Rosa María Valdovinos-Rosas. A comparison between ucb and ucb-tuned as selection policies in ggp. *Journal of Intelligent & Fuzzy Systems*, 36(5):5073–5079, 2019.

[22] Sylvain Gelly and David Silver. Combining online and offline knowledge in uct. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 273–280, New York, NY, USA, 2007. Association for Computing Machinery.

[23] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856 – 1875, 2011.

[24] Sylvain Gelly and Yizao Wang. Exploration exploitation in Go: UCT for Monte-Carlo Go. In *NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop*, Canada, December 2006.

[25] Michael Genesereth and Yngvi Björnsson. The international general game playing competition. *AI Magazine*, 34(2):107, Jun. 2013.

[26] Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the aaai competition. *AI Magazine*, 26(2):62, Jun. 2005.

[27] Michael Genesereth and Michael Thielscher. General game playing. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(2):1–229, 2014.

[28] David P Helmbold and Aleatha Parker-Wood. All-moves-as-first heuristics in monte-carlo go. In *IC-AI*, pages 605–610, 2009.

[29] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[30] Chu-Hsuan Hsueh, I-Chen Wu, Wen-Jie Tseng, Shi-Jim Yen, and Jr-Chang Chen. Strength improvement and analysis for an mcts-based chinese dark chess program. In Aske Plaat, Jaap van den Herik, and Walter Kosters, editors, *Advances in Computer Games*, pages 29–40, Cham, 2015. Springer International Publishing.

[31] Shih-Chieh Huang, Rémi Coulom, and Shun-Shii Lin. Monte-carlo simulation balancing in practice. In *International Conference on Computers and Games*, pages 81–92. Springer, 2010.

[32] Kevin Jamieson. Lecture 3: Stochastic Multi-Armed Bandits, Regret Minimization. `https://courses.cs.washington.edu/courses/cse599i/18wi/resources/lecture3/lecture3.pdf`, 2018. [Consulted online, may 25, 2020].

[33] Levente Kocsis, Csaba Szepesvári, and Mark H. M. Winands. Rspsa: Enhanced parameter optimization in games. In H. Jaap van den Herik, Shun-Chin Hsu, Tsan-sheng Hsu, and H. H. L. M. (Jeroen) Donkers, editors, *Advances in Computer Games*, pages 39–56, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[34] Tomáš Kozelek. Methods of mcts and the game arimaa. 2009.

[35] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *CoRR*, abs/1003.0146, 2010.

[36] Francis Maes, Louis Wehenkel, and Damien Ernst. Automatic discovery of ranking formulas for playing with multi-armed bandits. In *European Workshop on Reinforcement Learning*, pages 5–17. Springer, 2011.

[37] Walid Mahdi, Seyyid Ahmed Medjahed, and Mohammed Ouali. Performance analysis of simulated annealing cooling schedules in the context of dense image matching. *Computación y Sistemas*, 21(3):493–501, 2017.

[38] Rémi Munos et al. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–129, 2014.

[39] J. Méhat and T. Cazenave. Combining uct and nested monte carlo search for single-player general game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):271–277, 2010.

[40] Pierre Perick, David L St-Pierre, Francis Maes, and Damien Ernst. Comparison of different selection strategies in monte-carlo tree search for the game of tron. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 242–249. IEEE, 2012.

[41] Stuart J Russell and Peter Norvig. *Inteligencia Artificial: un enfoque moderno*. Number 04; Q335, R8y 2004. 2004.

[42] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *science*, 317(5844):1518–1522, 2007.

[43] Jonathan Schaeffer, Robert Lake, Paul Lu, and Martin Bryant. Chinook the world man-machine checkers champion. *AI Magazine*, 17(1):21, 1996.

[44] Michael John Schofield, Timothy Joseph Cerexhe, and Michael Thielscher. Hyperplay: A solution to general game playing with imperfect information. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[45] Aaron N Siegel. *Combinatorial game theory*, volume 146. American Mathematical Soc., 2013.

[46] David Silver and Gerald Tesauro. Monte-carlo simulation balancing. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 945–952, 2009.

[47] C. F. Sironi, J. Liu, and M. H. M. Winands. Self-adaptive monte carlo tree search in general game playing. *IEEE Transactions on Games*, 12(2):132–144, 2020.

[48] C. F. Sironi and M. H. M. Winands. Comparing randomization strategies for search-control parameters in monte-carlo tree search. In *2019 IEEE Conference on Games (CoG)*, pages 1–8, 2019.

[49] Chiara F. Sironi and Mark H. M. Winands. On-line parameter tuning for monte-carlo tree search in general game playing. In Tristan Cazenave, Mark H.M. Winands, and Abdallah Saffidine, editors, *Computer Games*, pages 75–95, Cham, 2018. Springer International Publishing.

[50] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[51] Maciej Świechowski and Jacek Mańdziuk. Specialized vs. multi-game approaches to ai in games. In P. Angelov, K.T. Atanassov, L. Doukovska, M. Hadjiski, V. Jotsov, J. Kacprzyk, N. Kasabov, S. Sotirov, E. Szmidt, and S. Zadrożny, editors, *Intelligent Systems'2014*, pages 243–254, Cham, 2015. Springer International Publishing.

[52] M Tak. The cross-entropy method applied to samegame. *Bachelor thesis, Maastricht University, Maastricht, The Netherlands*, 2010.

[53] F. Teytaud and O. Teytaud. On the huge benefit of decisive moves in monte-carlo tree search algorithms. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 359–364, 2010.

[54] Gabriel Van Eyck and Martin Müller. Revisiting move groups in monte-carlo tree search. In H. Jaap van den Herik and Aske Plaat, editors, *Advances in Computer Games*, pages 13–23, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[55] Li Zhou. A survey on contextual multi-armed bandits. *CoRR*, abs/1508.03326, 2015.