ISSN 1870-4069

Upper Confidence Bound o Upper Confidence Bound Tuned para General Game Playing: Un estudio empírico

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

Universidad Autónoma del Estado de México, Facultad de Ingeniería, Toluca, Estado de México, México
ifranciscov196@alumno.uaemex.mx, {jrmarcialr,rvaldovinos}@uaemex.mx

Resumen. En este artículo se muestra un análisis comparativo de dos políticas de selección: Upper Confidence Bound (UCB) y Upper Confidence Bound Tuned (UCB-Tuned), para el método Monte Carlo Tree Search (MCTS), en el contexto de General Game Playing (GGP). Este análisis tiene como objetivo determinar cual de las políticas tiene el mejor rendimiento, en términos de victorias. Para el comparativo se desarrollaron dos agentes basados en estas políticas de selección usando el framework GGP-Base. Así mismo los agentes se limitaron a 100 simulaciones de juego por cada iteración del método MCTS. El análisis se realizó en tres etapas, en cada una de las cuales, los agentes compitieron en 1000 juegos de tablero Breakthrough y 1000 de Knightthrough. Los resultados muestran que el agente con mejor rendimiento en GGP, es el agente basado en la política UCB-Tuned, ya que en promedio, resulta ganar el 60,8 % de las veces en Breakthrough y el 78,1 % en Knightthrough.

Palabras clave: jugar juegos en general, problema del bandido multi-armado, políticas de selección.

Upper Confidence Bound or Upper Confidence Bound Tuned for General Game Playing: An Empirical Study

Abstract. This paper shows a comparative analysis of two selection policies: Upper Confidence Bound (UCB) and Upper Confidence Bound Tuned (UCB-Tuned), for the Monte Carlo Tree Search method (MCTS), in the context of General Game Playing (GGP). This analysis aims to determine which of the policies has the best performance, in terms of victories. For the comparison two agents were developed based on these selection policies using the GGP-Base framework. The agents were limited to 100 simulations of game for each iteration of MCTS method. The analysis has been done on three stages, in each of which the agents play in 1000 Breakthrough games and 1000 Knightthrough games. The results show that the agent with the best performance in GGP, is the agent based on UCB-Tuned police, since in average, it turns out to win

the 60,8 % of times in Breakthrough and 78,1 % in Knightthrough.

Keywords: general game playing, multi-armed bandit problem, selection policies.

1. 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 ser humano [4]. 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 [20]. En este sentido, la IA ha producido agentes capaces de jugar a nivel de campeones humanos en juegos específicos como Chinook para Damas [18](juego para el cual ya existe una estrategia que permite ganar sin importar las jugadas que realice el contrincante [17]), Deep Blue para el ajedrez [6] y Dark Knight para Banqi o ajedrez medio chino [13].

El área de la IA que se ha enfocado en desarrollar agentes inteligentes capaces de jugar cualquier juego, es el área de General Game Playing (GGP). En GGP los agentes deben ser autónomos capaces de desarrollar sus propias estrategias sin intervención humana, sin haber jugado el juego con anterioridad y partiendo únicamente de las reglas que le son suministrados momentos antes de jugar (generalmente se utiliza Game Description Languaje) [4,11,12,20]. Aunque los agentes en GGP no pueden compararse en rendimiento y eficiencia a 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 [11,12].

Un avance importante en el área de GGP ha sido la implementación del método Monte Carlo Tree Search (MCTS), idea propuesta por primera vez en [10] 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 [11]. MCTS es un método que se guía por simulaciones Monte Carlo [5], usando los resultados de exploraciones previas para estimar con mejor precisión los valores de los movimientos más prometedores [5,9].

Los algoritmos basados en MCTS hacen 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 [9]: 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 al árbol, 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 MCTS 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 denominado Multi-Armed Bandid Problem (MABP) [15,19,21]. El MABP consiste en lo siguiente, 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 ó se debe probar otra máquina con la esperanza de que se obtenga una mejor recompensa?. Upper Confidence Bound (UCB), propuesto en [3], es el algoritmo más popular para MABP, y se ha usado en MCTS como política de selección con buenos resultados [10], a esta combinación se le conoce como Upper Confidence Bounds Applied to Trees (UCT).

UCB debido a sus resultados en juegos específicos, es la política más utilizada en GGP [7,8], 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 juego. Desde la concepción de UCB se han desarrollado nuevas políticas para MABP como UCB-Tuned [10], UCB-V, PAC-UCB [2], UCB-Minimal [14], Minimax Optimal Strategy (MOSS) [1], entre otros [5]. Hasta el momento no se cuenta con información que reporte la aplicación de estas políticas en GGP, incluso la política PAC-UCB no se ha probado en MCTS [5], esto debido quizás a los resultados obtenidos por UCB. Sin embargo, estas políticas también son independientes del dominio por lo cual se pueden aplicar a GGP pudiendo presentar mejores resultados que UCB.

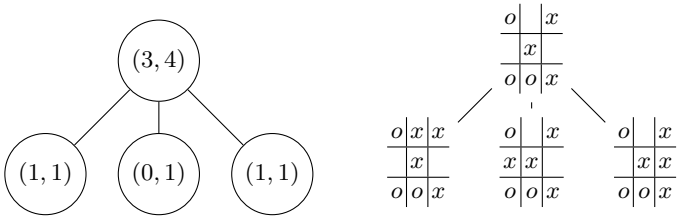
Perick et al. [16] hace un comparativo de estas políticas en MCTS con el juego de Tron donde UCB-Tuned es quien tiene el mejor desempeño. Si estas políticas se aplican al dominio GGP, y debido a la variedad de juegos que éstos abordan, es posible que el resultado obtenido por Perick et al. varie.

En este artículo se muestra un estudio comparativo del rendimiento de las políticas UCB y UCB-Tuned en GGP usando los juegos Breakthrough y Knightthrough. La idea de este comparativo es analizar el comportamiento de cada una de estas políticas y su rendimiento en el contexto de GGP.

2. Monte Carlo Tree Search

El método Monte Carlo Tree Search hace uso de un árbol de juego, en el cual cada nodo representa un estado del juego y cada nodo un movimiento, en la Figura 1, se puede observar una parte del árbol del juego para el juego del gato.

Los nodos adicionalmente de representar los estados del juego, llevan registro del número de victorias y de visitas de sus nodos hijos.



a. Árbol de Juego

b. Un estado del juego del Gato y sus posibles movimientos

Fig. 1. Árbol del juego para el juego del Gato.

El método Monte Carlo Tree Search requiere de un árbol de juego y consta de cuatro pasos [5]:

Selección En este paso el método recorre el árbol de juego hasta un nodo hoja, en cada nivel el método debe decidir que nodo debe de expandirse, esta decisión es guiada por una política de selección que se basada en el número de victorias y de visitas de los nodos. En un inicio se usaba la tasa entre el número de victorias y las visitas, sin embargo popularmente se usa UCB debido a sus buenos resultados.

Expansión Una vez seleccionado el nodo hoja, este se expande por medio de los movimientos posibles del juego en el estado representando por el nodo hoja.

Simulación Partiendo del estado del juego del nodo hoja seleccionado, el método realiza los movimientos de los dos jugadores de manera aleatoria hasta lograr un resultado: victoria o derrota. Una desventaja de usar movimientos aleatorios radica en que el método no refleja a un jugador inteligente, por ejemplo en la Figura 1b el método podría elegir los movimientos de la izquierda o el de en medio, sin embargo, un jugador inteligente elegiría el movimiento de la derecha. Cazenave [7,8] ha mostrado que usando movimientos de simulaciones anteriores, en las que se lograron victorias, en lugar de movimientos aleatorios se puede lograr un mejor rendimiento del método.

Retro-propagación En este paso el resultado de la simulación es propagado en todos los nodos que se recorrieron en el paso de selección, incrementado el número de visitas y de vitorias

Estos pasos se repiten hasta que se cumple una condición de paro, como cierta cantidad de tiempo, de memoria o cierto número de simulaciones. Finalmente se escoge el movimiento del nodo del primer nivel del árbol con la mejor relación entre victorias y visitas.

2.1. Políticas de selección

En cada iteración del MCTS se decide que nodo del árbol debe visitarse en cada nivel, esta selección es guiada por una política de selección que debe ser capaz de balancear la explotación y la exploración, MABP es un problema que se enfrenta al mismo dilema. Dado un conjunto de máquinas traga-monedas que pueda dar o no una recompensa, y cuya distribución es desconocida e independiente del resto, el problema MABP consiste en elegir la máquina para activar en cada momento de tal manera que se obtenga la mayor recompensa posible

La selección de nodos en MCTS es equivalente a la selección de máquinas en MABP, cada nodo puede verse como una máquina, la cual da una recompensa cuando obtiene una victoria en su simulación, y cada visita equivale a una activación. Por lo anterior se pueden utilizar los mismos métodos de MABP en MCTS.

El proceso de selección de la máquina traga-monedas se conoce como política de selección. En un inicio la política de selección consistía en elegir aquella máquina que en promedio da mas veces una recompensa, sin embargo, se han propuesto nuevas políticas de selección en las que destacan UCB y UCB-T

Upper confidence bounds. Propuesta por Auer [3] para el Multi-Armed Bandit Problem, establece que la selección de la máquina que debe jugarse es aquella que maximice la ecuación (1):

$$\bar{x}_j + c\sqrt{\frac{\ln n}{n_j}}, \tag{1}$$

donde $c = \sqrt{2}$, $\bar{x}$ es el promedio de las recompensas de la máquina j, n_j es el número de veces que la máquina j ha sido jugada hasta el momento, y n el número total veces que se han jugado todas las máquinas.

UCB puede ser divido en dos términos:

$$\alpha = \bar{x}_j, \tag{2}$$

$$\beta = c\sqrt{\frac{\ln n}{n_j}}. \tag{3}$$

La ecuación (2) se encarga del proceso de explotación y la ecuación (3) del proceso de exploración. Durante la ejecución se puede ver que conforme una máquina es jugada más veces, el término de exploración deja de influir y el de explotación incrementa su influencia, lo que permite que UCB explore todas las máquinas.

Upper confidence bounds tuned. Auer [3] propone UCB-Tuned como una mejora a UCB, en la cual la máquina a jugar es aquella que tenga el valor máximo:

$$\bar{x}_j + \sqrt{\frac{\ln n}{n_j} \text{mín} \left\{ \frac{1}{4}, V_j\left(n_j\right) \right\}}, \tag{4}$$

$$V_j\left(s\right) = \left(\frac{1}{s} \sum_{\tau=1}^{s} x_{j,\tau}^2 \right) - \bar{x}_{j,s}^2 + \sqrt{\frac{2 \ln t}{s}}, \tag{5}$$

donde s es el número de veces que se ha jugado la máquina, t el número de veces que alguna máquina ha sido jugada. Se debe observar que en la ecuación (5) se hace uso de la varianza muestral y la constante $\frac{1}{4}$ en la ecuación (4) surge del límite superior de la varianza de una variable aleatoria Bernoulli.

3. Escenario de pruebas

Para realizar el comparativo se hizo uso del framework GGP Base[1] y los juegos para dos personas por turnos: Breakthrough y Knightthrough (dos de los juegos usados en [7,8]).

3.1. GGP-base

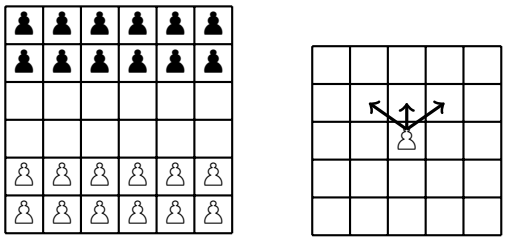
GGP-Base es un framework, escrito en Java, que permite el desarrollo de agentes para GGP, además de incorporar un Servidor que permite realizar encuentros entre agentes. GGP-Base puede conectarse a servidores proveedores de juegos lo que permite probar los agentes desarrollados en distintos escenarios.

3.2. Breakthrough

El juego hace uso de un tablero cuadriculado de nxm. En un inicio las dos primeras filas de cada jugador contienen peones (ver Figura 2). El objetivo radica en llevar un peón a la primera fila del contrincante para ello el peón puede moverse una casilla hacia adelante o en diagonal siempre hacia adelante (ver Figura 2b). El peón puede capturar solamente en diagonal.

Para la investigación aquí mostrada se uso un tablero de 6×6 disponible en GGP-Base.

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

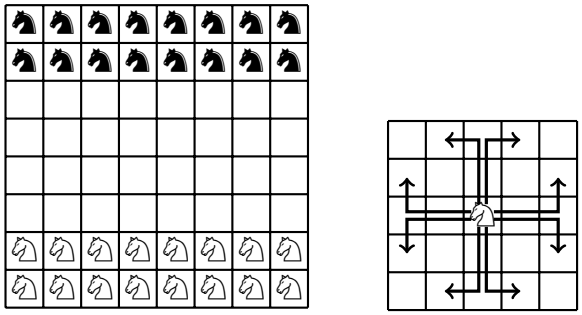


a. Tablero Inicial **b.** Movimientos

Fig. 2. Breakthrough 6×6.

3.3. Knightthrough

Similar a Breakthrough con la diferencia de que en lugar de usar peones se usan caballos que se mueven de igual manera que en el ajedrez (ver Figura 3). El objetivo es llevar un caballo a la primera fila del contrincante. Para la investigación aquí mostrada se usó un tablero de 8×8, de igual manera, disponible en GGP-Base.



a. Tablero Inicial **b.** Movimientos

Fig. 3. Knightthrough 8×8.

4. Ejecución y resultados

Para comparar las políticas se crearon dos agentes, uno basado en UBC y otro basado en UCB-Tuned. Para MCTS en el paso de simulación se estableció que se usarán 100 simulaciones por iteración. Para disminuir el sesgo aleatorio, el análisis se realizó en tres etapas, cada una de las cuales los agentes compitieron en 1000 juegos de Breakthrough y 1000 juegos Knightthrough, registrando el número de vitorias de cada uno de los agentes.

De lo anterior se obtuvieron los resultados mostrados en la Tabla 1. De los resultados se puede observar que el agente que tiene el mejor rendimiento,

en términos de victorias, es el basado en UCB-Tuned, ganando en promedio el 60,86 % de las veces en el juego Breakthrough y 78,1 % en el juego Knightthrough.

Tabla 1. Resultados.

	Etapa	Breakthrough	Knightthrough
UCB	1	392	217
	2	397	220
	3	385	223
	$\bar{x}$	391,333	220,0
	σ	6,0277	3,0
UCB-TUNED	1	608	786
	2	603	780
	3	615	777
	$\bar{x}$	608,666	781
	σ	6,0277	4,5825

5. Conclusiones y trabajo futuro

En este artículo se mostró un análisis comparativo de dos políticas de selección, UCB y UCB-Tuned, en el contexto de GGP, esto con la finalidad de determinar, qué política tiene el mejor rendimiento en términos del número de victorias. Tomando los resultados obtenidos del análisis comparativo se puede concluir que la política de selección que tiene mejor rendimiento en GGP, es UCT-Tuned. El motivo de que UCB-Tuned tenga mejor rendimiento que UCB puede deberse a que UCB-Tuned hace uso de la varianza muestral para determinar qué máquina debe jugarse, en tanto que UCB solo asume una distribución arbitraria.

Se deja como trabajo futuro el uso de otros juegos para determinar si el comportamiento de UCB-Tuned se mantiene estable en éstos. Así mismo se propone realizar un análisis utilizando las políticas UCB-V, PAC-UCB, UCB-Minimal y Minimax Optimal Strategy, con diversas cantidades de simulaciones en MCTS.

Referencias

1. Audibert, J.Y., Bubeck, S.: Minimax policies for adversarial and stochastic bandits. In: Conference on Learning Theory. pp. 217–226 (2009)
2. Audibert, J.Y., Munos, R., Szepesvári, C.: Tuning bandit algorithms in stochastic environments. In: International conference on Algorithmic Learning Theory. vol. 4754, pp. 150–165. Springer (2007)
3. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning 47(2-3), 235–256 (2002)
4. Björnsson, Y., Schiffel, S.: Comparison of GDL reasoners. In: Proceedings of the IJCAI-13 Workshop on General Game Playing (GIGA'13). pp. 55–62 (2013)

5. Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games 4(1), 1–43 (2012)
6. Campbell, M., Hoane, A., hsiung Hsu, F.: Deep blue. Artificial Intelligence 134(1), 57–83 (2002)
7. Cazenave, T.: Playout policy adaptation for games. In: Advances in Computer Games. pp. 20–28. Springer (2015)
8. Cazenave, T.: Playout policy adaptation with move features. Theoretical Computer Science 644, 43–52 (2016)
9. Chaslot, G.: Monte-carlo tree search. Maastricht: Universiteit Maastricht (2010)
10. Finnsson, H., Björnsson, Y.: Simulation-based approach to general game playing. In: AAAI. vol. 8, pp. 259–264 (2008)
11. Genesereth, M., Björnsson, Y.: The international general game playing competition. AI Magazine 34(2), 107 (2013)
12. Genesereth, M., Love, N., Pell, B.: General game playing: Overview of the AAAI competition. AI magazine 26(2), 62 (2005)
13. Hsueh, C.H., Wu, I.C., Tseng, W.J., Yen, S.J., Chen, J.C.: Strength improvement and analysis for an mcts-based chinese dark chess program. In: Advances in Computer Games. pp. 29–40. Springer (2015)
14. Maes, F., Wehenkel, L., Ernst, D.: Automatic discovery of ranking formulas for playing with multi-armed bandits. In: European Workshop on Reinforcement Learning. pp. 5–17 (2011)
15. Munos, R., 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)
16. Perick, P., St-Pierre, D.L., Maes, F., Ernst, D.: 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. pp. 242–249 (2012)
17. Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., Sutphen, S.: Checkers is solved. science 317(5844), 1518–1522 (2007)
18. Schaeffer, J., Lake, R., Lu, P., Bryant, M.: Chinook the world man-machine checkers champion. AI Magazine 17(1), 21 (1996)
19. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction, vol. 1. MIT press Cambridge (1998)
20. Świechowski, M., Mańdziuk, J.: Specialized vs. multi-game approaches to AI in games. In: Intelligent Systems' 2014, pp. 243–254. Springer (2015)
21. Zhou, L.: A survey on contextual multi-armed bandits. CoRR abs/1508.03326 (2015)