Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI5437 - Inteligencia Artificial I
Ene-Mar 2024
Prof. Carlos Infante

Gabriela Panqueva 18-10761
Daniela Ramírez 16-10940

Informe Proyecto 2

1. Introducción

Este informe presenta el desarrollo y los resultados de un proyecto centrado en el estudio y la implementación de algoritmos de solución para árboles de juego, utilizando como caso de estudio una versión reducida del juego de Othello. El objetivo principal del proyecto es profundizar en el entendimiento del modelo de árboles de juego y los algoritmos básicos de solución, incluyendo Negamax sin poda alpha-beta, Negamax con poda alpha-beta, Scout y Negascout.

2. Especificaciones del equipo usado

AMD Ryzen 7 5800H 3.20 GHz, 40,0 GB (39,4 GB usable) de RAM y se ejecutaron en una WSL2 de Ubuntu 22.04.3 LTS.

3. Detalles de implementación

Se completó la variante 6x6 de Othello. Además, se modificó el main agregando la implementación de los algoritmos: negamax, negamax AB, scout, negascout.

A la hora de implementar cada uno de los algoritmos se tomó como guía el pseudocódigo de cada uno dado en clases. Adicionalmente, para implementar un límite de tiempo en la ejecución de los algoritmos, se utilizó una combinación de variables globales y manejo de excepciones. Se declaró una variable global *TIME_LIMIT* en 7200 segundos, equivalente a 2 horas. También, otra variable global *time_limit_reached* que se inicializa en false. Esta variable se utiliza para indicar si se ha alcanzado el límite de tiempo.

Dentro de cada algoritmo, se verifica continuamente si se ha superado el límite de tiempo. Esto se hace obteniendo el tiempo actual y calculando el tiempo transcurrido desde el inicio de la ejecución. Si el tiempo transcurrido supera el límite de tiempo, se establece *time_limit_reached* en true y se lanza una excepción.

4. Tablas de Resultados

- Negamax minmax

PV	Expandidos	Generados	Tiempo (s)	Generados/segundos
34	0	0	5.00004e-06	0
33	1	1	3.00002e-06	250005
32	3	4	3.00002e-06	1.33332e+06
31	1	0	1.00001e-06	0
30	6	7	4.00003e-06	1.74999e+06
29	7	8	3.00002e-06	2.66665e+06
28	49	66	1.8e-05	3.66666e+06
27	98	130	3.3e-05	3.93939e+06
26	575	754	0.000179	4.21229e+06
25	2361	3173	0.000743	4.27052e+06
24	6244	7959	0.001936	4.11105e+06
23	37532	48266	0.010875	4.43825e+06
22	222933	287306	0.058067	4.94784e+06
21	1781833	2263734	0.457284	4.95039e+06
20	6548613	8358060	1.68971	4.94645e+06
19	44595493	57894111	11.4095	5.0742e+06
18	436519885	574629641	113.69	5.05435e+06
17	2805093090	3708262376	726.637	5.10332e+06

- Negamax alfa beta

PV	Expandidos	Generados	Tiempo (s)	Generados/segundos
34	0	0	6.00005e-06	0
33	1	1	2.00002e-06	499996
32	3	4	3.00002e-06	1.33332e+06

31	1	0	1.00001e-06	0
30	6	7	4.00003e-06	1.74999e+06
29	7	8	4.00003e-06	1.99998e+06
28	8	9	4.00003e-06	2.24998e+06
27	43	52	1.7e-05	3.05882e+06
26	132	158	5.2e-05	3.03846e+06
25	450	562	0.000178	3.1573e+06
24	540	666	0.000203	3.28079e+06
23	1669	2014	0.000668	3.01497e+06
22	1860	2249	0.000738	3.04743e+06
21	39931	48312	0.014789	3.26675e+06
20	29161	35242	0.009562	3.68563e+06
19	54035	65707	0.018201	3.61008e+06
18	316866	389093	0.105136	3.70085e+06
17	506940	622704	0.169955	3.66394e+06
16	5013831	6311595	1.63769	3.85397e+06
15	18214619	23017365	5.94248	3.87336e+06
14	105478601	131012668	33.876	3.86742e+06
13	203965475	255460982	65.359	3.90858e+06
12	974252913	1213331121	306.178	3.96283e+06
11	2715455054	3402860791	887.415	3.83458e+06
10	242519228	1407017048	1509.33	932211
9	2750181462	282943040	2370.91	19340

- Scout

PV	**Expandidos**	**Generados**	**Tiempo (s)**	**Generados/segundos**
34	0	0	8.00006e-06	0

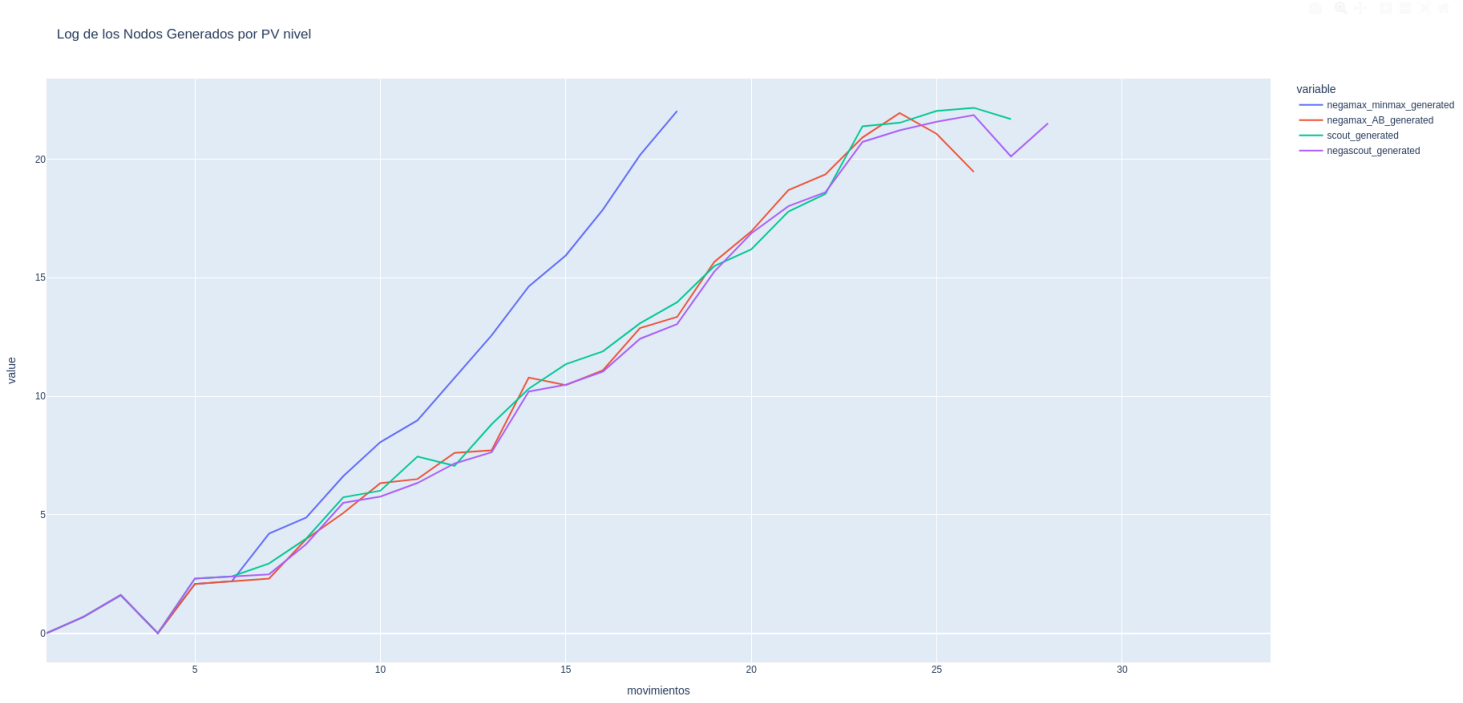
33	1	1	4.00003e-06	249998
32	2	4	6.00005e-06	666662
31	1	0	1.00001e-06	0
30	8	9	9.99984e-06	900014
29	9	10	1.40001e-05	714280
28	13	18	1.69999e-05	1.05883e+06
27	28	53	4.80001e-05	1.10416e+06
26	153	308	0.000215	1.43256e+06
25	171	407	0.000296	1.375e+06
24	753	1728	0.001103	1.56664e+06
23	467	1168	0.000822	1.42092e+06
22	2916	6749	0.004843	1.39356e+06
21	12158	30177	0.020572	1.4669e+06
20	33273	85112	0.059748	1.42452e+06
19	57886	146152	0.098978	1.47661e+06
18	178931	476328	0.322043	1.47908e+06
17	448628	1155534	0.774925	1.49116e+06
16	1932651	5300749	3.4964	1.51606e+06
15	3951237	10776405	7.23809	1.48885e+06
14	18488008	53132303	34.7794	1.5277e+06
13	39161547	112236762	69.367	1.61801e+06
12	693821064	1946806191	816.632	2.38395e+06
11	814164142	2251600125	590.438	3.81344e+06
10	1327843568	3721019245	983.938	3.78176e+06
9	1506460159	4237717590	1127.64	3.75803e+06
8	1029384541	2622504965	4045.63	648232

- Negascout

PV	Expandidos	Generados	Tiempo (s)	Generados/segundos
34	0	0	5.00004e-06	0
33	1	1	2.00002e-06	499996
32	3	4	3.00002e-06	1.33332e+06
31	1	0	1.00001e-06	0
30	9	9	4.00003e-06	2.24998e+06
29	10	10	4.00003e-06	2.49998e+06
28	11	11	4.00003e-06	2.74998e+06
27	38	42	1.5e-05	2.8e+06
26	213	244	6.80001e-05	3.58823e+06
25	264	318	9.3e-05	3.41935e+06
24	477	564	0.000161	3.5031e+06
23	1096	1286	0.000385	3.34026e+06
22	1762	2063	0.00062	3.32742e+06
21	22528	26779	0.007712	3.47238e+06
20	29874	35766	0.010471	3.41572e+06
19	52428	62336	0.018581	3.35483e+06
18	205413	248175	0.071408	3.47545e+06
17	381374	460706	0.133174	3.45943e+06
16	3380066	4190261	1.16434	3.59881e+06
15	16992650	21220286	5.7671	3.67954e+06
14	54098487	66662224	18.3287	3.63704e+06
13	97552015	120694553	32.7425	3.68618e+06
12	816134204	1005264618	269.364	3.732e+06
11	1319457904	1638389048	453.148	3.61557e+06
10	1888020125	2348800529	652.751	3.59831e+06

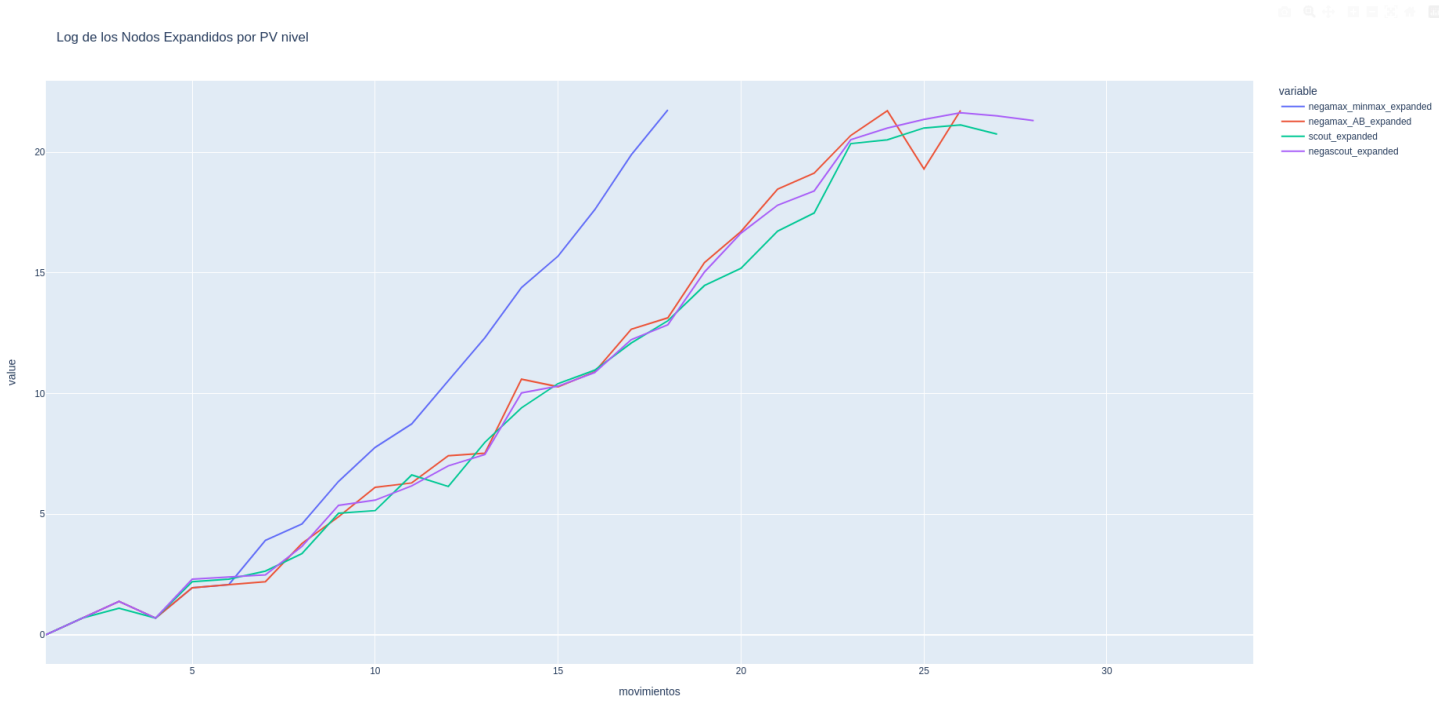
9	2499872789	3109932267	876.682	3.54739e+06
8	2194332207	544781349	3738.68	145715
7	1807917874	2197478059	6679.91	328968

5. Análisis



La gráfica anterior ilustra el logaritmo de los nodos generados en cada nivel. Se puede observar que todos los algoritmos exhiben una tendencia asintótica similar, lo cual es coherente con la naturaleza exponencial de los árboles de juego.

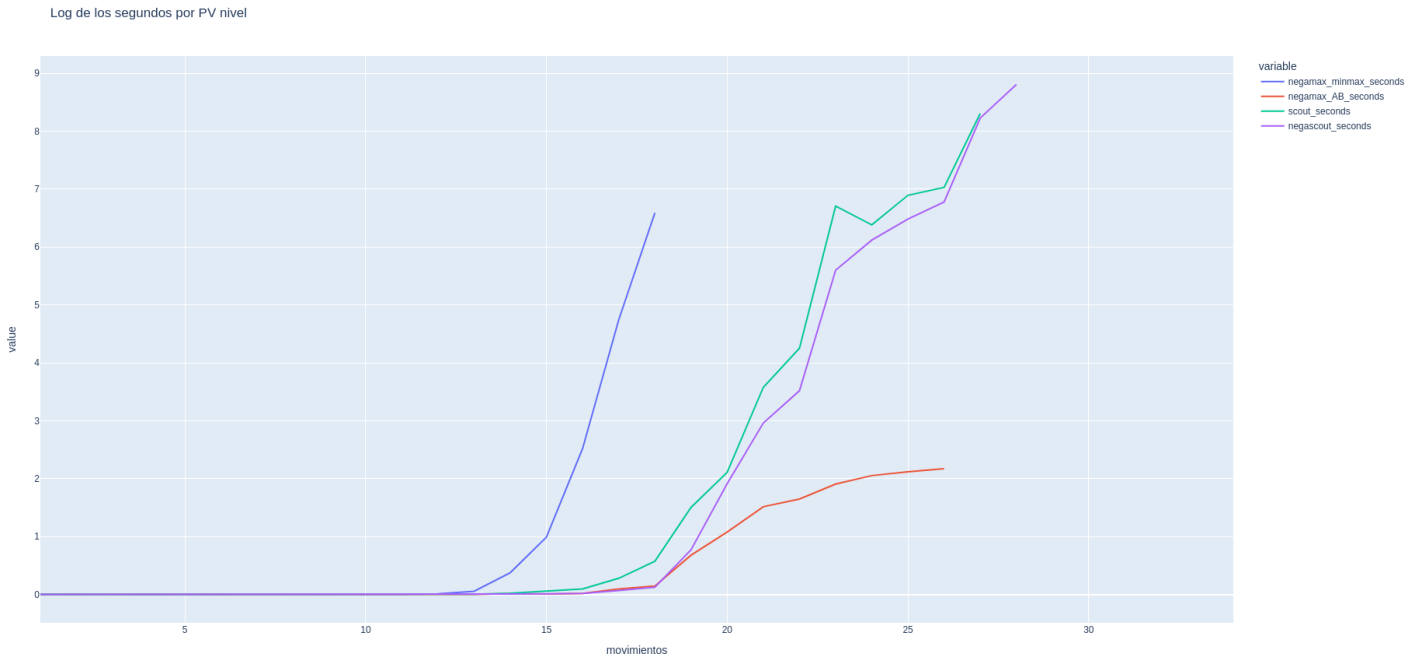
Así mismo, esta tendencia asintótica es una manifestación de la complejidad inherente a los árboles de juego. A medida que nos adentramos en los niveles más profundos del árbol, el número de nodos crece exponencialmente, lo cual es evidente en la gráfica.



En la gráfica previa, observamos que la tendencia asintótica de los nodos expandidos es idéntica a la de los nodos generados, lo que indica que cualquier variación entre las dos gráficas podría deberse a una traslación. Dado que estamos trabajando con una gráfica logarítmica, estas traslaciones representan un cambio constante en la gráfica original.

Esta traslación constante puede ser interpretada como una medida de la eficiencia de cada algoritmo. En otras palabras, aunque todos los algoritmos generan un número similar de nodos (como se refleja en la similitud de las tendencias asintóticas), la cantidad de nodos que cada algoritmo realmente necesita expandir para encontrar la solución puede variar.

De acuerdo a las dos gráficas anteriores, se puede observar que el algoritmo Scout presenta una leve traslación que los demás, esto podría indicar que este algoritmo es capaz de encontrar la solución con menos expansiones de nodos, lo que sugiere que es más eficiente.

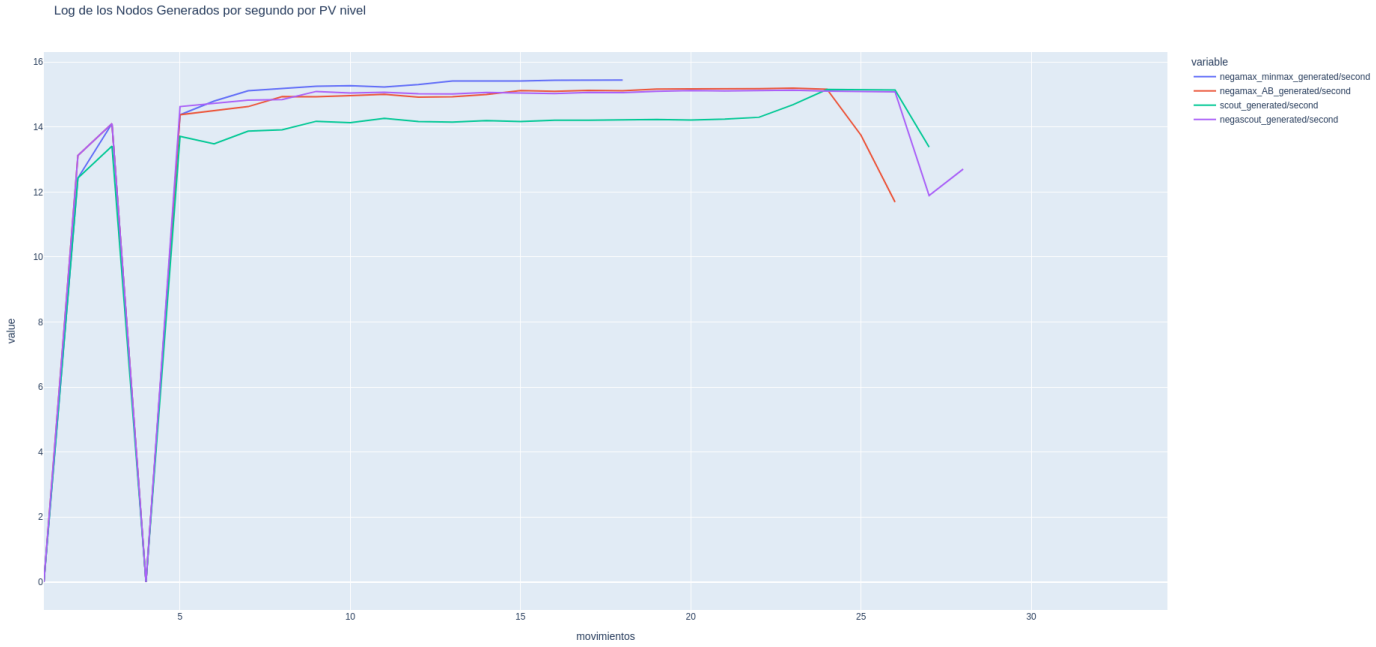


En la gráfica anterior, se puede observar cuándo un algoritmo converge, de esta forma, se puede determinar cuál converge más rápidamente.

De acuerdo a la gráfica, se puede ver que la versión minmax de Negamax no solo consume más tiempo, sino que también diverge después de 15 niveles aproximadamente. Esto puede deberse a que este algoritmo explora todo el árbol de juego.

Así mismo, se podría decir que los algoritmos Scout y Negascout exhiben comportamientos similares. Sin embargo, en la gráfica se visualiza que Scout deja de converger antes que Negascout. Esta diferencia podría atribuirse a un ordenamiento eficiente de los nodos, ya que Negascout asume que el primer nodo a expandir pertenece a la variante principal.

Finalmente, se observa que el algoritmo Negamax con poda alfa-beta supera a los demás. Esto podría deberse a que este algoritmo aprovecha el uso de la poda alfa-beta para realizar la poda minimax, lo que le permite manejar eficientemente la complejidad de los árboles de juego. Esto lo convierte en una herramienta poderosa para resolver problemas en juegos de suma cero y perfecta información, como Othello.



En esta gráfica los algoritmos presentan un comportamiento similar, es decir, producen aproximadamente la misma cantidad de nodos por segundo. Esto sugiere que la optimización de algunos algoritmos sobre otros no se debe tanto a la estructura intrínseca del algoritmo, sino más bien a la implementación de la poda.

Podríamos decir que todos los algoritmos están, como máximo, a un orden de magnitud de distancia entre sí. Esto indica que, aunque hay diferencias en la eficiencia de los algoritmos, estas diferencias no son drásticas. En otras palabras, los 4 algoritmos son capaces de generar nodos a una velocidad similar.

Sin embargo, es importante recordar que la generación de nodos no es el único factor que determina la eficiencia de un algoritmo. La capacidad de un algoritmo para podar eficazmente el árbol de juego y reducir el número de nodos que necesita explorar puede tener un impacto significativo en su rendimiento general. Por lo tanto, aunque todos los algoritmos generan nodos a una velocidad similar, aquellos con poda pueden ser más eficaces y pueden ser capaces de encontrar la solución más rápidamente.

6. Conclusiones

Finalmente, se pueden hacer las siguientes observaciones para concluir:

- Negamax sin poda, muestra claramente el rendimiento más bajo, incapaz de calcular el valor del juego para 17 movimientos en el futuro, incluso después de esperar 2 horas.

- El rendimiento de Negamax mejora con la implementación de la poda Alpha-Beta, logrando calcular el valor del juego para 25 movimientos en el futuro dentro del límite de tiempo establecido de 2 horas.
- Scout y Negascout logran calcular el valor del juego para 26 y 27 movimientos en el futuro, respectivamente. Sin embargo, es importante destacar que, aunque Scout expande menos nodos, genera más.
- El costo en tiempo para calcular el valor del juego para un movimiento aumenta drásticamente. Así mismo, en términos de tiempo, el algoritmo Negamax con poda Alpha-Beta supera a los demás algoritmos por varios órdenes de magnitud, lo que sugiere que es el algoritmo más adecuado para usar en cualquier escenario.

Para terminar, los resultados experimentales demuestran la superioridad del algoritmo Negamax con poda alfa-beta en términos de eficiencia y rendimiento. Aunque todos los algoritmos son capaces de generar nodos a una velocidad similar, los algoritmos con poda son más eficaces para resolver el árbol de juego y reducir el número de nodos que necesita explorar, lo cual puede tener un impacto significativo en su rendimiento general. Esto subraya la importancia de la poda en la eficiencia de los algoritmos de árboles de juego. Por lo tanto, para problemas de juegos de suma cero y perfecta información, como Othello, el algoritmo Negamax con poda alfa-beta parece ser la opción más eficiente y efectiva.