Aplicaciones de Arboles de juego en Othello 6x6.

Pedro Rodriguez - 15-11264 and Daniel Pinto - 15-11139

1 Specs del equipo usado

Todas las pruebas se corrieron en el mismo equipo, el cual cuenta con los siguientes specs:

- **Procesador:** Ryzen 7 3800x, 3.9GHz.
- **RAM:** 16GB, 3200MHz.
- SSD.

2 Generados

La Figura (1) muestra el logaritmo de los nodos generados por nivel. Como es de esperarse, todos los algoritmos presentan un comportamiento asintótico similar. Lo cual tiene mucho sentido puesto que los arboles de juego, al ser arboles tienen un numero exponencial de nodos.

3 Expandidos

En la Figura (2) vemos que el comportamiento asintótico de los nodos expandidos posee la misma forma que los nodos generados, lo cual quiere decir que si hay una variacion la diferencia entre ambos grafos es solo una traslacion. Al estar en un log grafo, las traslaciones representan un cambio constante en el grafo original.

Sin embargo, esto es solo media verdad. Puesto que tambien pueden existir variaciones que representen un cambio constante en el log grafo (y por lo tanto cambios exponenciales en el grafo original). Para probar la veracidad de esto, refiramonos a la Figura (3), en donde se compara $\frac{\#Generados}{\#Expandidos}$.

En ella, vemos el verdadero beneficio con respecto a la podas realizadas. En algoritmos como negamax min max, vemos que la relación entre expandidos y generados se mantiene casi constante en 1 hasta que eventualmente cae a 0 (lo que representa que no hay mas resultados a partir de tal punto). Esto quiere decir que el algoritmo genera mas o menos lo mismo que expande. Una razón para que esto no sea exacto, es debido a que aunque minmax no posee mecanismos de poda, al realizar las corridas no se contaba como expandidos los casos base (profundidad restante 0 o nodo terminal). Por lo tanto, no se cuentan los expandidos del ultimo nivel generando tal perturbación. Esto es un fenomeno que ocurre en todos los algoritmos.

Segun el grafico, el mecanismo de poda de Negascout y nefamax alfa beta (ambos basados en la misma idea) se comportan de manera similar, expandiendo la misma cantidad que se genera. Esto no necesariamente significa que no provean una mejora con respecto a negamax mini max. Puesto que al podar ramas innecesarias, se permiten explorar nodos con verdadero potencial.

Finalmente vemos que scout a diferencia de los demas, posee un comportamiento semejante a una curva, lo que quiere decir que la cantidad de nodos generados crece mas rapido que los expandidos, lo cual es una excelente señal. Scout funciona con una heuristica que le permite descartar

Pedro Rodriguez - 15-11264: 15-11264@usb.ve, http://compositionality-journal.org
Daniel Pinto - 15-11139: 15-11139@usb.ve

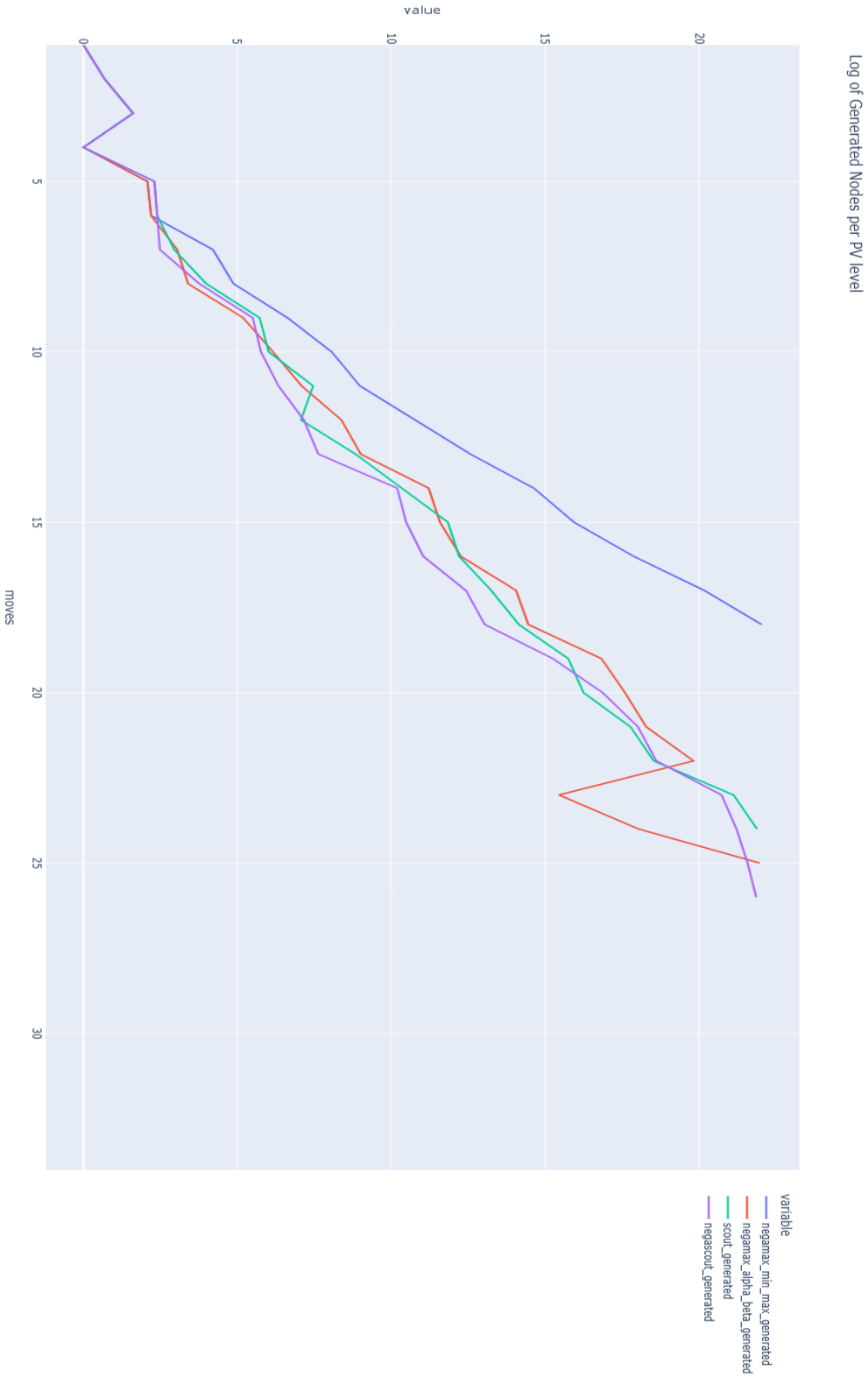

Figure 1: Log de nodos generados por nivel

nodos innecesarios. Esto promete mejorar el tiempo del algoritmo al solo expandir y por lo tanto en ahondar en considerablemente menos ramas.

4 Tiempos

Para la Figura (4) decidimos utilizar un grafo De niveles versus tiempo puesto que para esta etapa, solo nos interesa cual algoritmo tiende a converger primero.

En el gráfico, vemos resultados esperados. Claramente la versión min max de negamax es la que no solo tarda mas tiempo, sino que diverge después de 16 niveles. Esto es esperado porque es nuestro algoritmo base, explora todo el arbol de juego.

A simple vista, podemos suponer que scout y negascout tienen un comportamiento similar, sin embargo, notamos que scout deja de converger 3 niveles antes que negascout, lo cual es una diferencia considerable, sabiendo que los arboles de juego crecen exponencialmente. Esto posiblemente se debe a un buen ordenamiento de los nodos, ya que la ventaja de negascout es precisamente asumir que el primer nodo a expandir es el que pertenece a la variacion principal.

Finalmente, vemos que el algoritmo negamax con alfa beta pruning es el que mejor se desempeña por varios ordenes de magnitud, lo cual es una diferencia significa. Esto puede deberse a que negamax saca el mayor provecho a dos particularidades: la primera es la caracteristica de cero suma de othello para simplificar el minmax usado (una vez se obtiene el minimo, tambien sabemos que el maximo es el negativo sin recursiones adicionales), y la segunda es aprovechando el alfa beta pruning para hacer la poda minimax.

5 Nodos Generados por segundo

Como ultimo addendum, una buena heurística para para determinar cual algoritmo se comporta es mirando la cantidad de nodos generados por segundo. Que es precisamente lo que hace la Figura (5). Sin embargo, esta figura no es demasiado concluyente, puesto que todos los algoritmos estan a lo sumo 1 orden de magnitud alejados de los demas. Por lo tanto, referimos cualquier analisis a las secciones anteriores.

6 Conclusiones

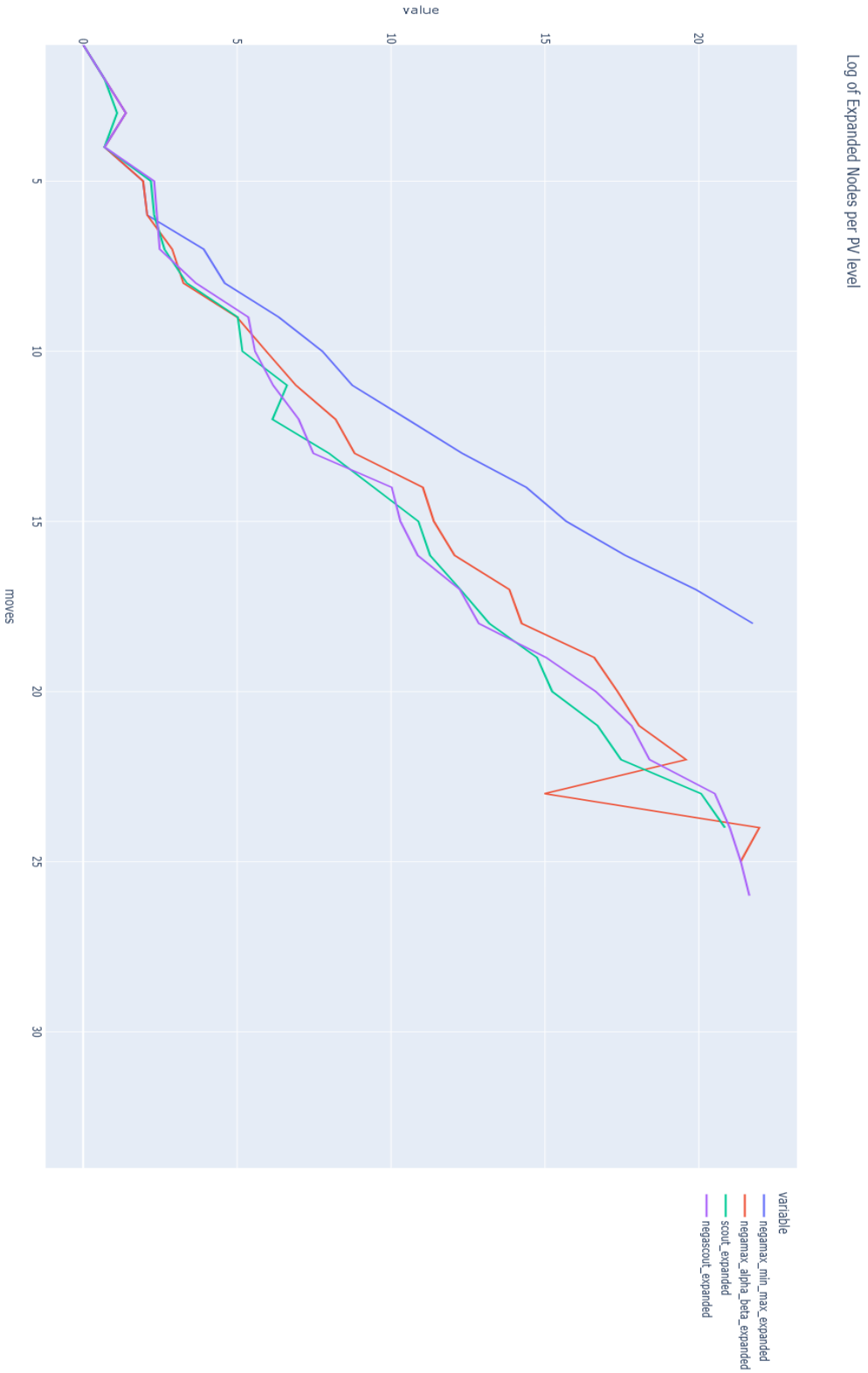

Figure 2: Log de nodos expandidos por nivel

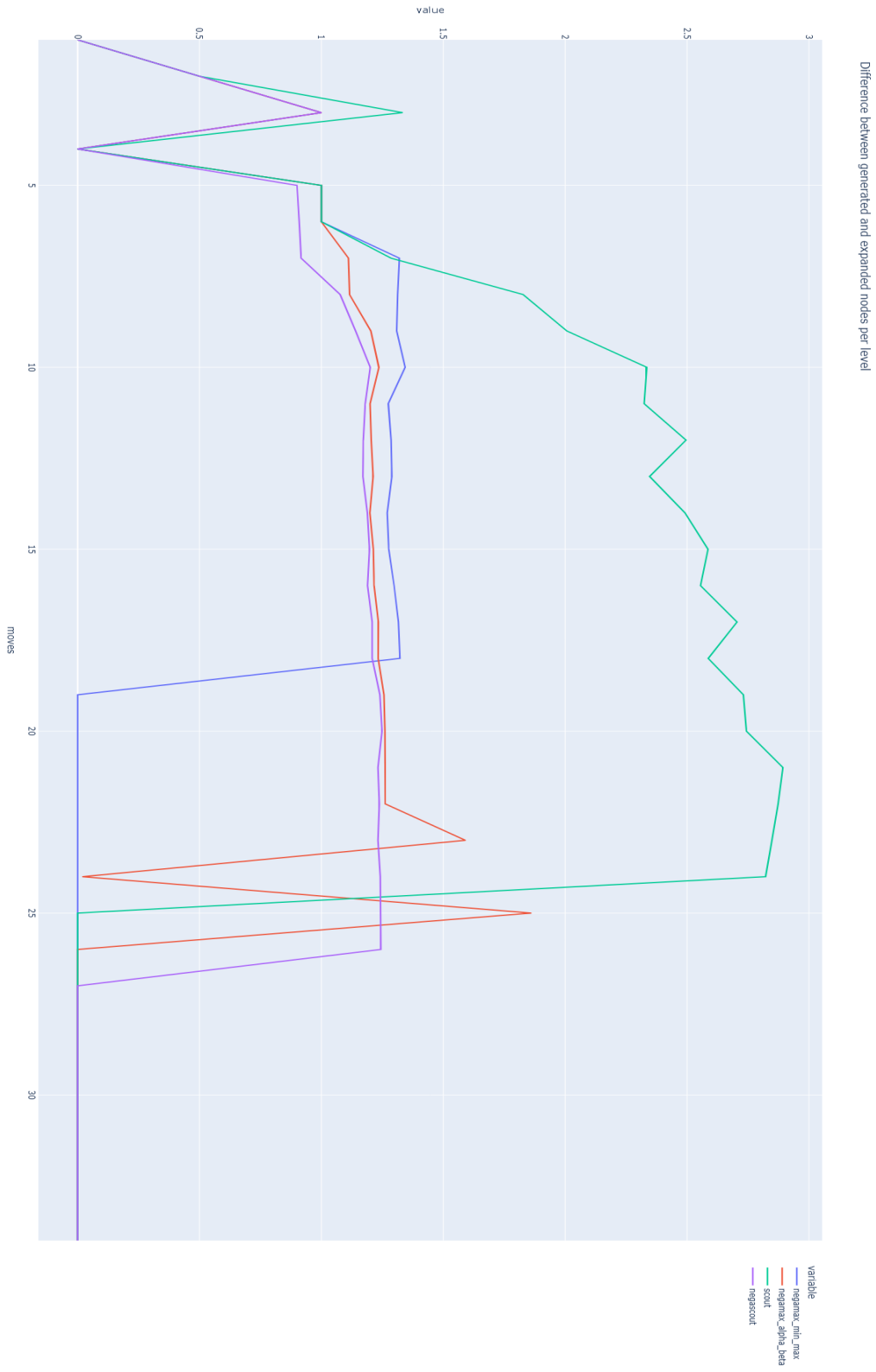

Figure 3: Nodos Generados / Nodos Expandidos

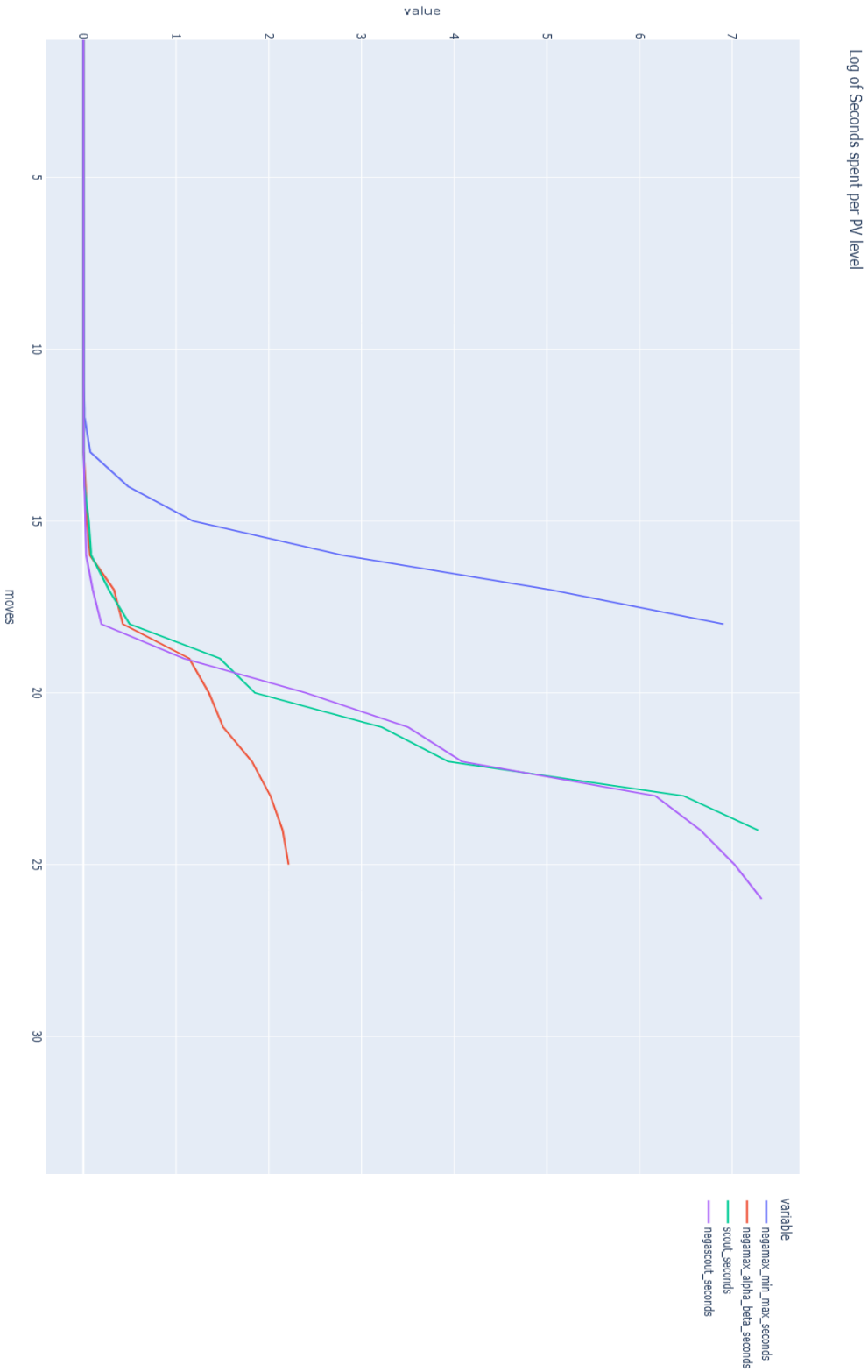

Figure 4: Tiempo expendido (en segundos) por nivel.

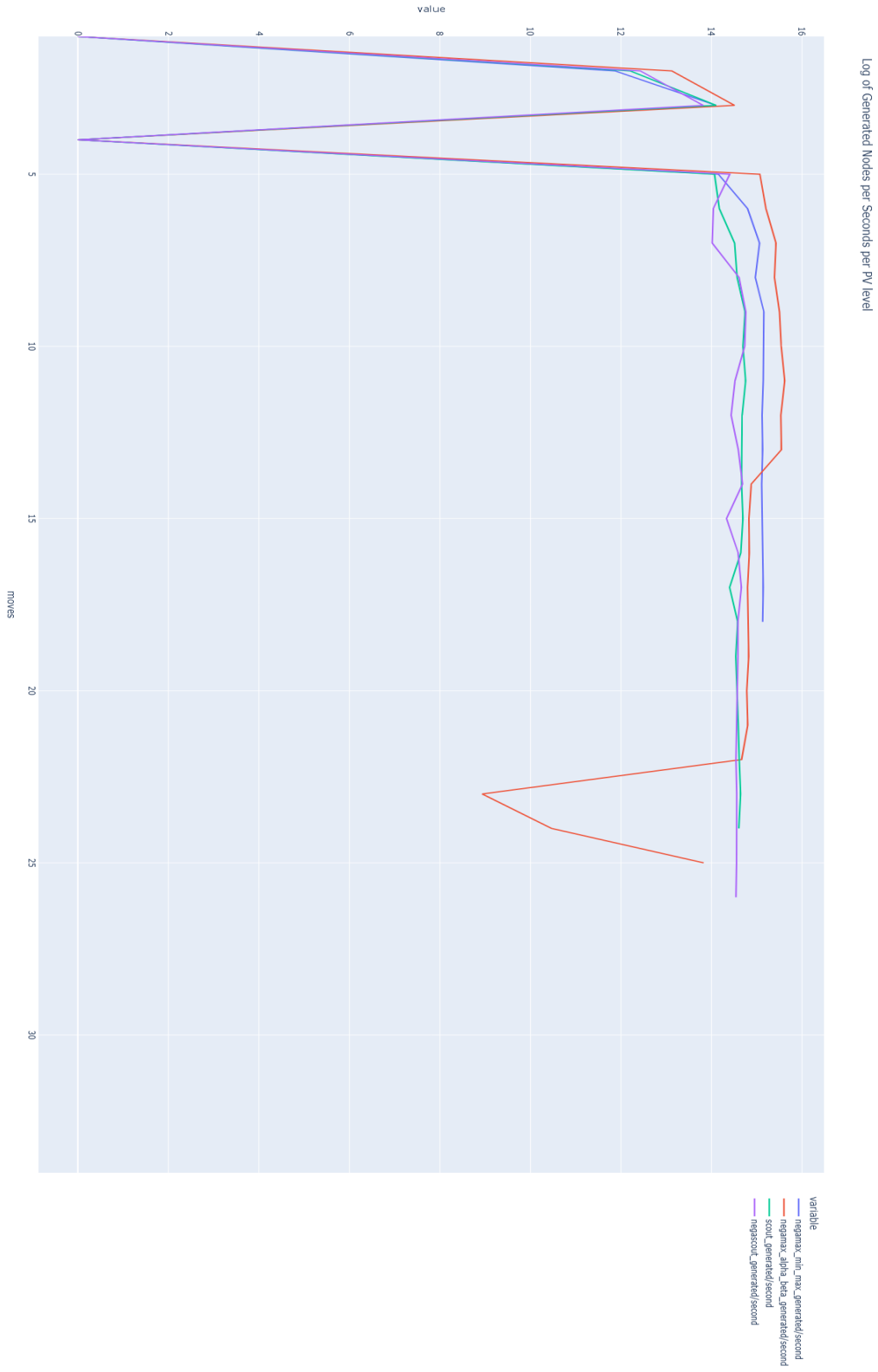

Figure 5: Nodos generados por segundo por nivel por nivel.