Universidad Simón Bolívar

Dpto. de Computación y Tecnología de la Información

Inteligencia Artificial I

Nathalia Silvera 12-10921

Carolina Rivas 13-11209

Informe Proyecto I

ARBOLES DE BUSQUEDA

Se realizaron pruebas de los algoritmos de búsqueda utilizando DFS con un período de ejecución de aproximadamente una hora y media. Cada problema se ejecutó durante 5 minutos, y los resultados obtenidos respaldaron la eficacia de la eliminación parcial de duplicados, para las corridas de 15-puzzle, 24-puzzle, rubik, y topSpin se utilizó una laptop con procesador i5 de 10ma generación, y para la torre de hanoi se utilizó una laptop con un procesador i7 de 4ta generación.

Durante los 5 minutos de ejecución, se pudo observar que la eliminación parcial de duplicados fue notablemente más eficiente que el enfoque sin ella. En ese corto período de tiempo, se logró explorar una mayor cantidad de profundidades utilizando la eliminación parcial de duplicados. Esto demuestra que el algoritmo DFS con eliminación parcial de duplicados pudo alcanzar niveles más profundos de la búsqueda en comparación con el DFS sin eliminación de duplicados.

	Sin eliminación de duplicado			**Con eliminación de duplicado**		
Problema	**Profundidad**	**Nodos Creados**	**Factor De Ramificación**	**Profundidad**	**Nodos Creados**	**Factor De Ramificación**
15-puzzle	18	793090458	3.236068	28	1783887192	2.130383
24-puzzle	18	2006589996	3.499989	25	1969145140	2.302753
Rubik	7	612220032	18.000000	8	1373243544	13.348469
Top- Spin 12-4	9	864813056	2.011279	10	1524902502	7.872983

Top- Spin 14-4	8	1475789056	14.000000	9	412633378	8.425861
Top- Spin 17-4	7	410338673	17.000002	10	1300119565	1.205720
Torre Hanoi 4-12	13	2059693197	5.715302	13	714402993	5.189146
Torre Hanoi 4-14	13	2059693197	5.715302	13	714402993	5.189146
Torre Hanoi 4-18	13	2059693197	5.715302	13	714402993	5.189146

Además, en cada profundidad explorada, se creó una menor cantidad de nodos utilizando la eliminación parcial de duplicados en comparación con el enfoque sin eliminación de duplicados. Esto indica que el factor de ramificación fue menor al utilizar la eliminación parcial de duplicados. Un factor de ramificación más bajo implica que el algoritmo DFS tuvo que considerar menos caminos alternativos desde cada nodo, lo cual es beneficioso en términos de eficiencia computacional.

Es importante tener en cuenta que estos resultados se obtuvieron dentro del límite de tiempo establecido para las pruebas y pueden variar según el problema específico y los recursos disponibles. Sin embargo, los resultados respaldan la eficiencia y el impacto positivo de la eliminación parcial de duplicados en el rendimiento del algoritmo DFS.

En resumen, la eliminación parcial de duplicados demostró ser altamente efectiva en las pruebas de los algoritmos de búsqueda utilizando DFS. Permitió explorar más profundamente en menos tiempo y generó una menor cantidad de nodos, lo que indica una mejora significativa en el factor de ramificación. Estos hallazgos respaldan la elección de utilizar la eliminación parcial de duplicados para mejorar la eficiencia y el rendimiento del algoritmo DFS en los problemas analizados.

HEURÍSTICAS / ALGORITMOS INFORMADOS

15-Puzzle

Se implementó una PDB aditiva para abordar el problema. Para ello, se crearon tres abstracciones disjuntas que fueron mapeadas para asegurar que cada una contará con exactamente 6 fichas (5 Tiles y un blank).

Se llevó a cabo la ejecución del algoritmo IDA* para resolver el problema, mientras que no fue posible utilizar el algoritmo A* debido a las limitaciones de la capacidad de procesamiento de la computadora.

Para el algoritmo de IDA* Se pudo observar que el tiempo de solución varía desde fracciones de segundo hasta varios segundos, dependiendo de la complejidad de la instancia. El número de nodos expandidos también varía ampliamente, lo cual es esperado debido a la naturaleza del algoritmo IDA*. La distancia recorrida es otra medida de la dificultad de cada instancia, donde valores más altos indican problemas más complejos.

24-Puzzle

Lamentablemente, no fue posible ejecutar este caso debido a su alta complejidad. Las computadoras utilizadas para las pruebas no fueron capaces de manejar la carga computacional requerida por el problema y mataban el proceso de ejecución, lo que nos impidió realizar una comparación de resultados en este caso específico.

Cubo de Rubik

Se utilizaron tres abstracciones distintas, cada una de ellas representando tres colores en lugar de todos los colores posibles. Esto se hizo para reducir la carga computacional, ya que sería demasiado complejo trabajar con todos los colores en cada abstracción. En una de las abstracciones se mantuvieron las esquinas, mientras que en las otras dos se mantuvieron los lados.

En general Ambos algoritmos son eficientes para resolver el Cubo de Rubik pero al observar los resultados, tenemos que el algoritmo de IDA* recorre una menor distancia y expande una cantidad de nodos menor al algoritmo de A*.Por lo que podemos decir que el desempeño del IDA* es mejor.

Top spin 12-4

Para TopSpin 12-4 se utilizaron dos abstracciones en las cuales se mapea un conjunto de variables a una sola, en la primera abstracción se seleccionó la mitad del conjunto total y para la segunda abstracción se seleccionó la mitad restante.

En general al observar los resultados obtenemos que ambos algoritmos encontraron la solución en un tiempo razonable y para la mayor parte de las instancias. En cuanto a la rapidez del algoritmo no podemos afirmar que A* fue más rápido que IDA* o viceversa ya que para algunas instancias A* fue más rápido que IDA* y de la misma manera ocurrió en otras instancias donde IDA* fue más rápido que A*

En relación a los nodos expandidos, observamos que IDA* mostros una menor cantidad de nodos explorados en comparación con A*. Por lo tanto, IDA* exploró menos nodos para llegar a la solución, lo cual indica una ventaja en términos de eficiencia y uso de memoria.

En relación a la distancia,ambos algoritmos lograron encontrar soluciones óptimas , pero el algoritmo de IDA* logró encontrar la solución en una menor distancia que A*.

En general, se puede concluir que IDA* demostró ser más eficiente en términos de uso de memoria al expandir menos nodos durante la búsqueda. Sin embargo, A* también mostró

un desempeño sólido en términos de tiempo de solución y distancia de la solución. La elección entre estos algoritmos dependerá de los requisitos específicos del problema y las limitaciones de recursos disponibles.

Top spin 14-4

Se crearon tres abstracciones en las cuales volvemos a establecer un conjunto de variables que se mapearan a una sola para reducir la complejidad del problema y escoger la mayor de la PDB obtenida.

En cuanto a tiempo de ejecución se observa que A* tuvo un tiempo de solución más rápido para la mayoría de las instancias encontradas en comparación con IDA*. Con respecto a los nodos expandidos, se muestra una menor cantidad de nodos expandidos cuando se ejecuta IDA*, lo cual indica que IDA* exploró una menor cantidad de nodos para encontrar la solución.

En relación a la distancia IDA* mostró encontrar las soluciones en menor distancia que A*.

En general, A* mostró ser más rápido en tiempos de ejecución, pero IDA* sigue siendo más eficiente en cuanto a uso de recursos.

Top spin 17-4

Igual que en el caso anterior Top-Spin 14-4 se crearon las tres abstracciones, pero la ejecución de este caso de prueba no fue exitoso para ninguno de los dos algoritmos, ya que en ambos casos las laptops no soportaron el uso de los recursos, por lo que ambos procesos fueron terminados(killed).

Sin embargo, aunque no se pudo completar la ejecución, con los resultados que mostraron se puede decir que A* mostró una distancia mayor en encontrar la solución, mayor tiempo y menor cantidad de nodos. Por lo que IDA* sigue siendo más eficiente que A*.

Torre de Hanói con 4 astas 12 discos

Para la Torre de Hanoi con 4 astas y 12 discos se crearon dos abstracciones disjuntas, en una se proyectaron la primera mitad de los estados y en la otra se proyectaron la segunda mitad de los estados, y para la heurística se escogió la mayor de la PDB.

En cuanto al tiempo de solución ambos algoritmos encontraron la solución en tiempos bastante cortos. Pero en relación a los nodos expandidos se observa que A* expandió más nodos en comparación a IDA*. Y finalmente, la distancia de la solución para las instancias para IDA* fue menor.

En general, se repite el comportamiento de que sigue siendo más eficiente el Algoritmo IDA*.

Torre de Hanói con 4 astas 14 discos

Para este caso se crearon dos abstracciones, una proyecta los estados del 13 al 45 y la otra abstracción proyecta los que faltaron en la anterior.

En cuanto al tiempo de solución, ambos algoritmos dieron tiempos bajos, pero observando los nodos expandidos el algoritmo IDA* expande menos nodos que el A*, y con respecto a la distancia es menor con el algoritmo IDA*.

Ambos algoritmos son capaces de generar una solución para el caso de prueba, sin embargo A* sigue demostrando usar más recursos del computador, por lo que IDA* sigue siendo más eficiente en cuanto uso de recursos y tiempos de ejecución, aunque esto también puede variar de acuerdo a la instancia que se utilice.

Torre de Hanói con 4 astas 18 discos

Finalmente, para este caso se crearon dos abstracciones, una proyecta los estados del 19 al 54 y la otra abstracción proyecta los que faltaron en la anterior.

Observando los resultados podemos concluir que IDA* tuvo tanto mejor tiempo de ejecución, como menor cantidad de nodos expandidos y menor distancia para encontrar la solución, por lo tanto IDA* obtuvo mejores resultados en comparación con el Algoritmo A*.

Conclusiones

En cuanto a los algoritmos informados, el algoritmo IDA* siempre tuvo un mejor comportamiento en cuanto a uso de recursos y eficiencia, pero no es correcto decir que siempre sera asi ya que la eficiencia de A* también depende de la heurística, por lo que su eficiencia puede variar de acuerdo a la naturaleza del problema.