Relatório Projeto Monta-Cargas

Alberto Ângelo Lalanda
2152157
2152157@my.ipleiria.pt

Tiago Alexandre Pinheiro Martins
2151237
2151237@my.ipleiria.pt

1. INTRODUÇÃO

Neste relatório do projeto Monta-Cargas para a disciplina de Inteligência Artificial mostraremos como procedemos na adaptação do problema com a ajuda de métodos de procura e heurísticas. O problema proposto foi um jogo de 10 níveis (estados iniciais) em que o objetivo é fazer um monta-cargas, que apenas se pode mover na horizontal, chegar a uma porta que se encontra na ultima coluna de uma matriz de 6x6. O conflito do jogo está nas peças que se encontram entre o monta-cargas e a porta sendo que as peças de direção horizontal só se podem mexer na horizontal e as peças com orientação vertical só se podem mexer na vertical (operadores). Os métodos de procura não informados (procura cega) que foram usados neste problema foram os seguintes: procura em largura primeiro (Breadth first search), procura uniforme (Uniform cost search), procura em profundidade primeiro (Depth first search), procura em profundidade limitada (Limited depth first search) e procura por aprofundamento progressiva (Iterative deepening search). Em relação aos métodos de procura informados (com heurísticas) usámos os seguintes: Procura sôfrega (Greedy best first search), procura A*, procura em Feixe (Beam search) e Procura IDA* aprofundamento progressivo. Criou-se ainda um conjunto de heurísticas simples e junções que estimam o custo (ou distância) do caminho de um nó n até ao estado objetivo para os métodos de procura informados.

2. IMPLEMENTAÇÃO

2.1 Adaptação do Projeto "Search"

A implementação do problema foi feita com base no projeto "Search" que foi usado nas aulas. Alterou-se então esse projeto para efetuar a resolução do jogo Monta-Cargas, começando por alterar o *EightPuzzleState* para ler estados de uma matriz n x n células em que cada objeto é um numero definido pelo nível, sendo o numero 0 um espaço vazio, o 1 os monta cargas e os restantes algarismos até ao 9, blocos de várias peças com direções predefinidas. Depois dos níveis estarem a ser lidos corretamente, foram feitos os métodos *canMove* e *Move* para cada direção válida para as peças no problema. No *EightPuzzleProblem* é definido que o objetivo do problema é atingido aquando o monta-cargas consegue chegar à ultima coluna da matriz. Também na mesma classe são adicionadas as ações para cada peça, depois de filtradas por ações validas são adicionadas à lista de sucessores para ser um

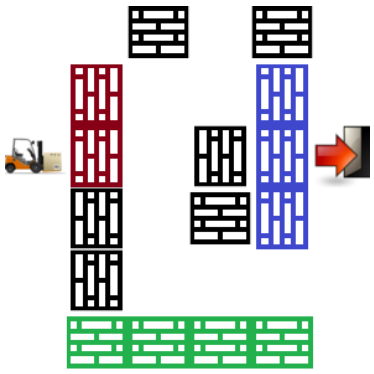

Figura 1. Estado inicial de um nível

deles escolhido pelos métodos de procura e executado, de acordo com as regras mencionadas na introdução.

2.2 Métodos de Procura

Aprofundando mais sobre cada um dos métodos de procura usados neste projeto, iremos começar com o método de procura em largura primeiro (Breadth first search) que é um método que expande todos os nós de *n* antes dos nós de *n* + 1. Trata-se de um método de procura completo (encontra sempre a solução quando ela existe no problema) e ótimo (encontra a melhor solução quando existem várias alternativas). Este método pode ser inútil em problemas de ramificação alto, tendo a solução de explorar tantos nós, já que verifica todas as possibilidades mais curtas, no entanto foi ideal para o monta-cargas pois é um problema que não sofre do obstáculo referido anteriormente, conseguiu-se assim respostas rápidas em todos os níveis.
De seguida o método de procura uniforme (Uniform cost search) é também completo e ótimo, e funciona expandindo sempre o nó da fronteira com menor custo. Este método aplicado neste projeto deu valores muito semelhantes ou completamente iguais aos valores do método de procura em largura primeiro para todos os níveis e por isso pode-se considerar que é um bom método para resolver o jogo do monta-cargas.
O próximo método trata-se da procura em profundidade primeiro (Depth first search) que expande sempre o nó o mais profundo possível, é completo (em espaços finitos) e não é ótimo. Observando os resultados deste método pode-se ver que, por vezes, o custo é muito maior do que os métodos anteriormente referidos, mas explora muitos menos nós, e noutros casos o custo e os nós explorados são ambos incrivelmente altos. Os resultados expõem os problemas do *Depth first search* na aleatoriedade

(grande diferença) dos seus resultados. Em seguida procura em profundidade limitada (Limited depth first search) expande os nós da mesma forma que o método de procura anterior, mas usa um nível imposto pelo utilizador como limite de profundidade, sem nunca o ultrapassar. Trata-se de um método completo se o nível imposto for maior ou igual à profundidade de uma solução e não é ótimo.

O ultimo método não informado é a procura por aprofundamento progressiva (Iterative deepening search), sendo este completo e ótimo, que exibe o melhor da procura em largura e em profundidade. A procura por aprofundamento progressiva é de facto muito lenta porque aumenta o limite progressivamente e faz para cada limite uma procura em largura primeiro até encontrar a melhor solução. Desta forma até para o problema 2 obtemos soluções depois de mais de 30 minutos de espera e neste caso só vai até ao limite 10. Para os restantes problemas este método de procura é lentíssimo.

Para os métodos informados, temos em primeiro lugar a procura sôfrega (Greedy best first search), que não sendo completa nem ótima, seleciona sempre o nó para que o valor *h* da heurística é menor. Nos seus resultados e dada a execução deste método a procura sôfrega foi a que mais variou consoante a heurística utilizada, variando de uma solução perfeita com poucos nós expandidos, a soluções com custo alto com um numero de nós expandidos baixo.

A procura A*, é usada para testar se uma heurística é admissível e expande sempre o nó que pertença ao caminho atual com menos custo, conseguindo assim sempre a melhor solução quando existe se a heurística for de facto admissível. Nesta procura os resultados tiveram efetivamente sempre o custo perfeito, com numero de nós muito menor nos níveis mais fáceis, em comparação, aos resultados de custo igualmente perfeito, dos métodos não informados da procura em largura primeiro e da procura uniforme. No entanto para os níveis mais difíceis, o numero de nós expandidos foi aproximado dos encontrados nos métodos não informados referidos anteriormente.

Em seguida a procura em Feixe (Beam search) funciona de modo semelhante ao A*, mas com um limite estabelecido para o tamanho máximo da fronteira. Depois de inseridos os sucessores de um nó na fronteira, eliminam-se os últimos nos desta lista de modo a que o limite seja respeitado. Permite concentrar a procura apenas sobre os estados mais promissores do espaço, ignorando os menos promissores. Não é ótimo nem completo. Os resultados da procura em feixe são similares aos dos encontrados na A*, mas ao limitar a fronteira máxima o numero de nos gerados vai ser menor assim como o numero de nos expandidos permitindo obter uma solução mais otimizada.

Por fim a procura por aprofundamento progressivo (IDA*), funciona de modo semelhante ao algoritmo de procura por aprofundamento progressivo, mas, em vez de impor limites sucessivos na profundidade, impõe limites sucessivos (contornos) no custo (dado pela heurística) das soluções que se podem calcular. O IDA* é completo e é ótimo e varia o numero de nós expandidos significativamente variando a heurística. Os resultados do IDA* são geralmente altos (comparativamente aos outros métodos de procura) no valor do numero de nós expandidos e numero de nós gerados, devido à sua maneira de operar.

2.3 Heurísticas

Criámos as seguintes heurísticas que vimos ser (geralmente) admissíveis, pois o custo na procura A* é igual ao custo da procura em largura primeiro e da procura uniforme. "*Heuristic tiles distance to final position*" que tem como custo o numero de blocos no caminho a ser feito por o carro até ao estado objetivo. "*Heuristic number of pieces in the way*" que tem como custo o numero de peças na linha do carro, que se encontram à sua direita. "*Heuristic size of objects in the way*" muito semelhante ao anterior, mas tem como custo o tamanho das peças no caminho. "*Heuristic number of occupied tiles in front of the car*" que conta com a ajuda de um ciclo todos os blocos da grelha que não se encontram vazios à direita do carro. Por fim "*Heuristic tiles distance to final position and number of pieces in the way*" é uma junção de duas heurísticas referidas anteriormente. A "*Heuristic number of occupied tiles in front of the car*" não resulta em valores perfeitos nos *puzzles* 10, 9 e 7 e a "*Heuristic size of objects in the way*" no *puzzle* 9.

3. RESULTADOS

No gráfico 1 (nos anexos) compara-se o custo nos diferentes métodos de procura. É possível observar que todos com a exceção do *Depth first search* tem custo ótimo. O gráfico 1 serve para auxiliar a escolha do método de procura mais eficaz (com o custo menor), desta forma é excluído o *Depth first search* pois foi definido que para selecionar o melhor método se iria priorizar o custo mais baixo e de seguida o menor numero de nós gerados.

No gráfico 2 compara-se os métodos de procura pelo numero de nós gerados, tamanho da fronteira máxima e numero de nós expandidos. Neste caso é possível observar que o *Depth first search* é o que tem o numero de nós gerados menor, no entanto, como vimos no gráfico 1 este é o que tem o custo mais elevado, logo é excluído. Dos métodos restantes o que tem o numero de nós gerados menor é o *Uniform cost search*. Assim sendo para o puzzle 1 o método mais eficaz é o *Uniform cost search* e o pior é o *Depth first search*.

No gráfico 3 (nos anexos) também se compara o custo nos diferentes métodos de procura. É possível observar que todos com a exceção do *Depth first search* tem custo ótimo. O gráfico 3 serve para auxiliar a escolha do método de procura mais eficaz, desta forma é excluído o *Depth first search* pois foi definido que para selecionar o melhor método se iria priorizar o custo mais baixo e de seguida o menor numero de nós gerados.

No gráfico 4 da mesma forma se comparam os métodos de procura pelo numero de nós gerados, tamanho da fronteira máxima e numero de nós expandidos. Neste caso é possível observar que o *Depth first search* é o que tem o numero de nós gerados menor, no entanto, como vimos no gráfico 3 este é o que tem o custo mais elevado, logo é excluído. Dos métodos restantes o que tem o numero de nós gerados menor é *o Breadth first search*. Assim sendo para o puzzle 10 o método mais eficaz é o *Breadth first search* e o pior é o *Depth first search*.

No gráfico 5 compara-se as diferentes heurísticas aplicadas ao *Greedy best first search* e foi possível observar-se que a melhor heurística é a "*Heuristic tiles distance to final position + Number of pieces in the way*" (H5). De um modo geral à exceção do IDA* foi possível observar-se que os métodos de procura informada geraram muitos menos nós do que os métodos de procura não informada.

No gráfico 6 compara-se as diferentes heurísticas aplicadas ao *Beam search* e foi possível observar-se que a melhor heurística é a mesma que a mencionada no ponto anterior. O *Beam search* na sua melhor heurística gerou mais nós que o *Greedy best first search* na sua melhor heurística, no entanto este obteve o mesmo custo. Para todas as outras heurísticas o *Beam search* obteve melhor resultado que o *Greedy best first search*.

No gráfico 7 compara-se as diferentes heurísticas aplicadas ao *Greedy best first search* e neste caso, foi possível observar-se que a melhor heurística é a "*Number of ocupied tiles in front of the car*" (H4). Em comparação com o gráfico 5 verifica-se que a heurística ideal é diferente.

No gráfico 8 compara-se as diferentes heurísticas aplicadas ao *Beam search* e foi possível observar-se que a melhor heurística é a "*Heuristic tiles distance to final position + Number of pieces in the way*" (H5). O *Beam search* na sua melhor heurística gerou mais nós que *o Greedy best first search* na sua melhor heurística, no entanto este obteve melhor custo para todas as heurísticas.

4. RESTANTES RESULTADOS

Os restantes resultados podem ser observados no ficheiro *excel* do projeto.

5. CONCLUSÃO

Em conclusão, neste projeto pode-se avaliar e aprender de forma prática como os métodos de procura usados no ramo de ciência de computação de inteligência artificial, funcionam num problema especifico, quais dos métodos obtêm melhores e piores resultados e o porquê.

Folha de anexos (1/4)

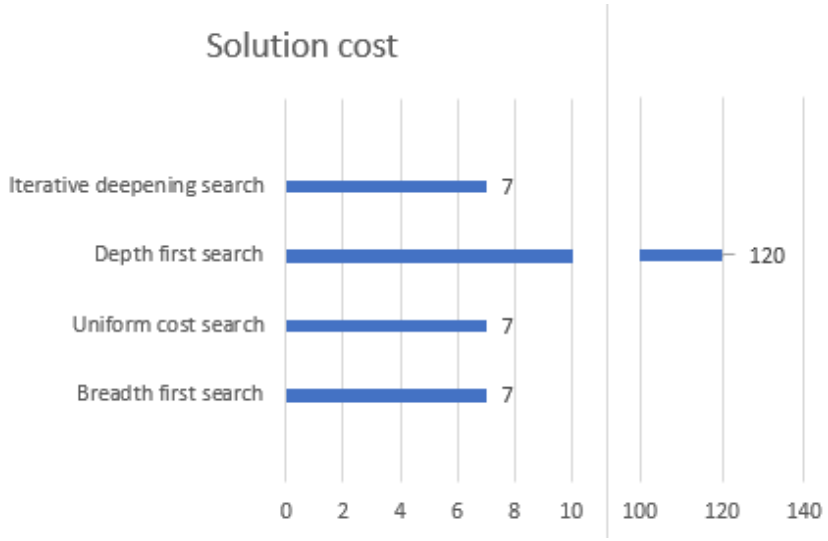

Gráfico 1 – Custo dos métodos não informados do nível 1

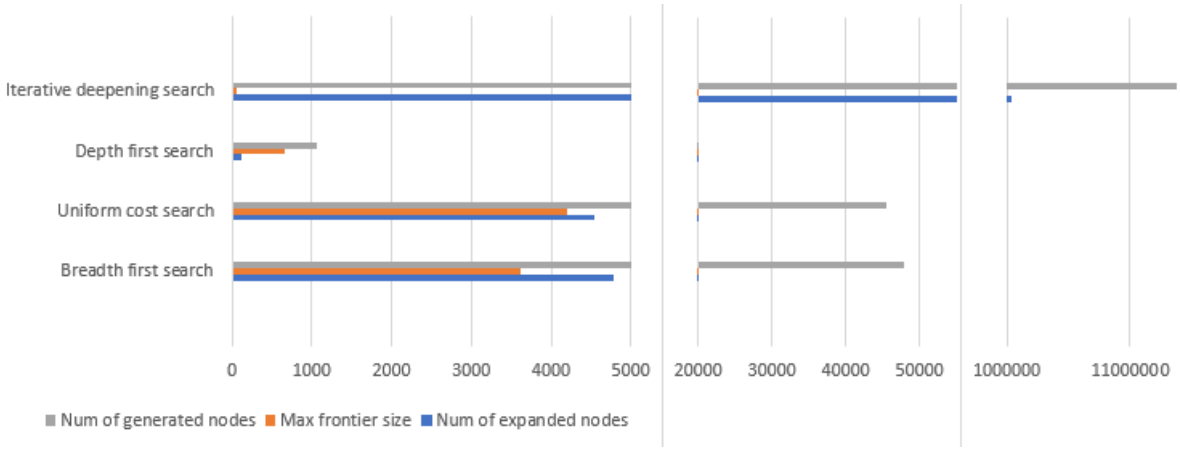

Gráfico 2 – Comparação entre métodos não informados do nível 1

Folha de anexos (2/4)

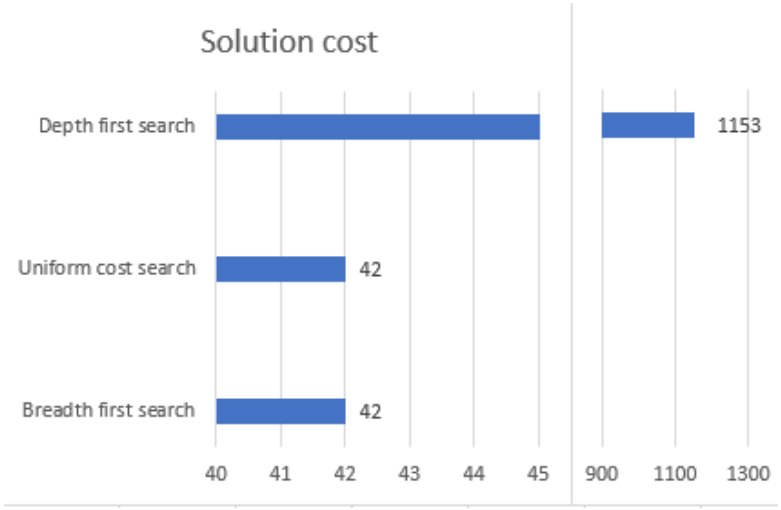

Gráfico 3 – Custo dos métodos não informados do nível 10

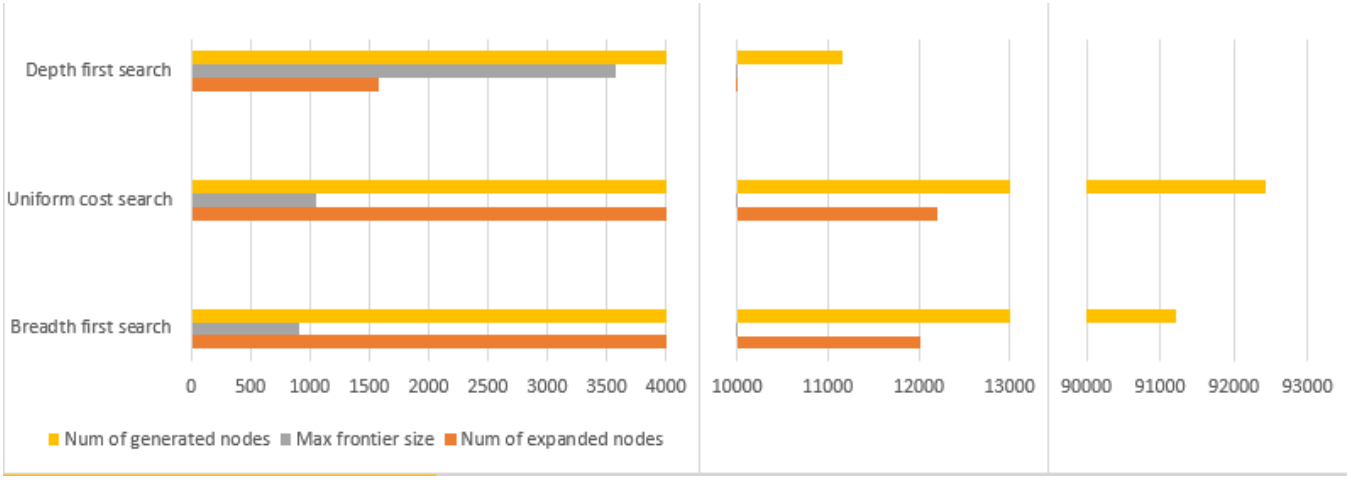

Gráfico 4 - Comparação entre métodos não informados do nível 10

Folha de anexos (3/4)

Heuristicas:

1 - Tiles distance to final position

2 - Number of pieces in the way

3 - Size of objects in the way

4 -Number of ocupied tiles in front of the car

5 - Tiles distance to final position + Number of pieces in the way

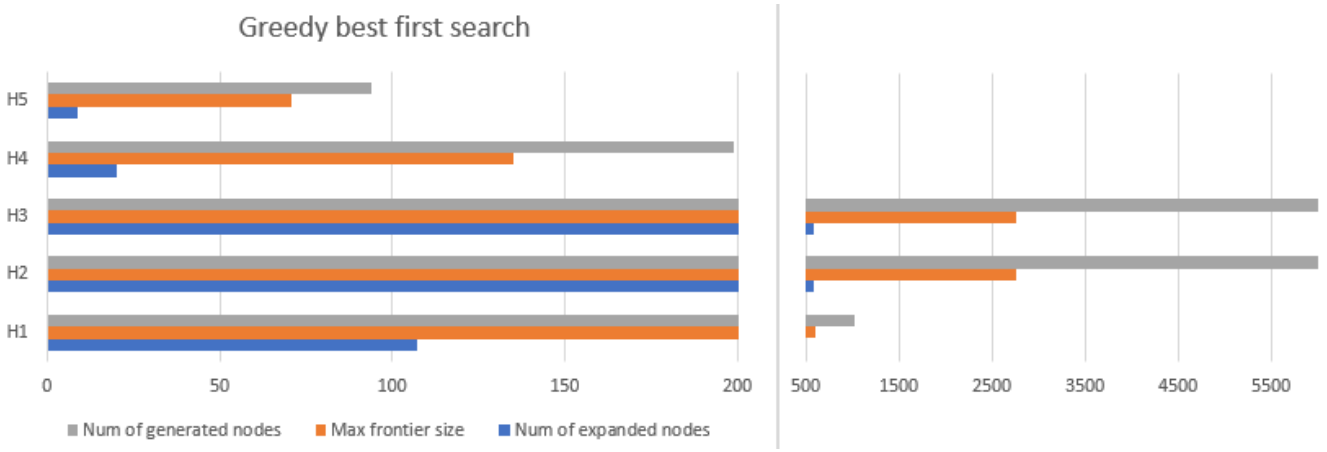

Gráfico 5 – Comparação das heurísticas do Greedy best first search para o nível 1

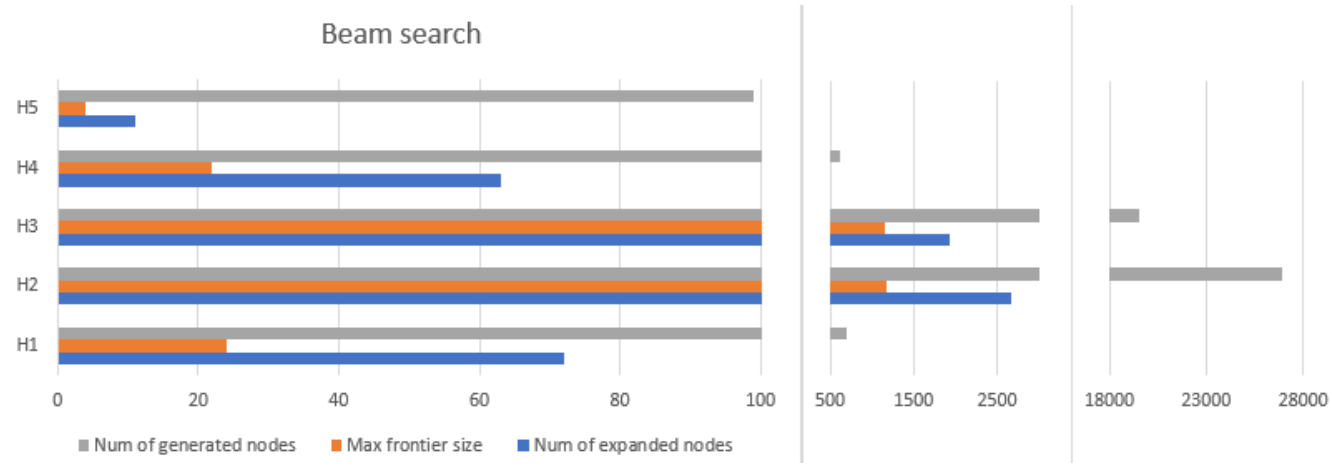

Gráfico 6 – Comparação das heurísticas do Beam search para o nível 1

Folha de anexos (4/4)

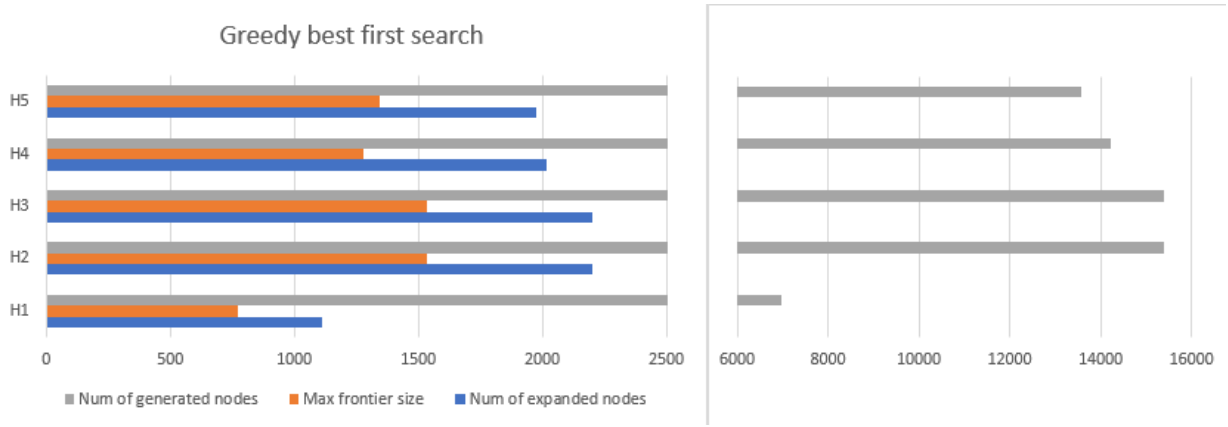

Gráfico 7 - Comparação das heurísticas do Greedy best first search para o nível 10

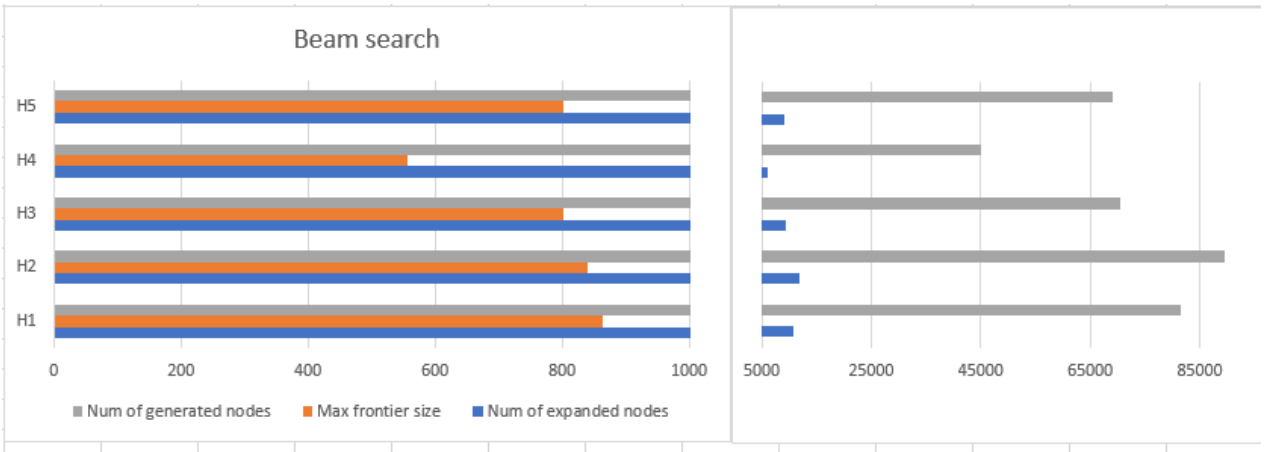

Gráfico 8 - Comparação das heurísticas do Beam search para o nível 10