Estratégias heurísticas

Uma heurística (do grego heuriskein – descobrir) é uma técnica que aponta um caminho a seguir na busca que seja mais promissor, ou seja, que ofereça uma resposta com uma qualidade aceitável, embora não necessariamente a melhor resposta. Para muitos problemas reais, o espaço de busca pode ser muito grande e a adoção de uma heurística pode ser determinante para impedir uma explosão combinatória de caminhos possíveis.

O algoritmo A* é um exemplo de algoritmo de busca em que é proposta uma heurística para a função de custo de cada nó N que possui a seguinte forma:

f(N) = g(N) + h(N) onde:

g(N) = custo do caminho até N

h(N) = estimativa do custo do caminho de N até um nó terminal (heurística).

À função h(N) é uma heurística sobre o menor caminho do nó N até o nó final. A qualidade da nossa solução depende desta estimativa. Se soubermos como calcular h(N), ou seja, se tivermos uma boa estimativa do custo de caminho de N até nosso objetivo, podemos orientar nossa busca apenas por esta estimativa (o que seria equivalente à busca gulosa). Entretanto, o algoritmo A* considera tanto o custo da estimativa (que é um custo estimado sobre o futuro do caminho), quanto o custo contabilizado real do caminho até o nó a partir do início (custo do passado). O algoritmo A* consiste dos seguintes passos:

1. Expandir todas as possibilidades do nó aberto escolhido
2. Para cada nó N gerado,

calcular f(N)

se N já está na árvore (aberto), eliminar aquele com maior f(N)

3. Escolher o nó que tenha o menor f(N) na árvore.
4. Se não foi achado o destino, retornar ao passo 1 .

Este algoritmo é muito parecido com a busca ordenada, exceto pelo fato que o cálculo de f(N) agora exige que se calcule a estimativa h(N). Como calculá-la? Os autores do algoritmo ressaltam que o importante é que a estimativa torne o algoritmo admissível, isto é, permita que o algoritmo encontre a solução ótima, caso ela exista. Em seu trabalho, os autores mostram que o algoritmo será admissível desde que a estimativa h(N) não seja maior que o custo real do caminho de N até o nó terminal, isto é: h(N) < custo real de N até o nó final.

Várias observações podem ser feitas sobre o algoritmo:

Sobre g(N): se fizermos g(N)=0, estaremos sempre nos preocupando somente com o futuro (estimativa). Podemos também aplicar um custo constante de movimentação de um nó para outro, de forma que o algoritmo sirva para encontrar o menor caminho (mais rápido) e não o de menor custo.

Se h(N) for um estimador perfeito, então o algoritmo convergirá para o objetivo pelo caminho ideal. Ao contrário, se h(N) for 0, a busca será orientada somente pelo caminho passado (busca ordenada). Se g(N) também for zero a busca será aleatória. Se o valor de g(N) for sempre 1 a busca será em largura. Para o caso intermediário, isto é, h(N) não for zero nem perfeito, o algoritmo sempre encontrará o caminho ideal, caso ele exista, desde que a estimativa h(N) seja otimista, isto é, seja menor que o custo real do caminho.

Exemplo: na árvore da figura 3.8, os custos entre nós são sempre 1 e a árvore mostra, para cada nó, o valor de h(N) + g(N).

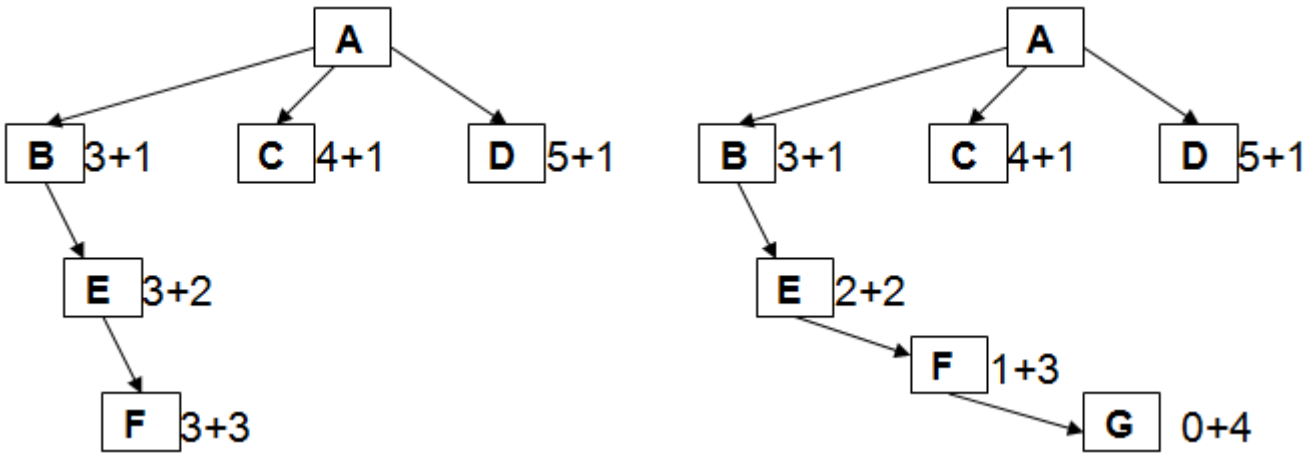


Figura 2.11 – Algoritmo A*

Note que na figura da esquerda, após as expansões realizadas pelo algoritmo, o custo para C (5) é maior que o custo para D (6) e, portanto, devemos abandonar a sub-árvore de B, que foi inicialmente explorada por possuir o menor custo total, e abrir a árvore em C. Isto implica em ter feito uma expansão infrutífera em B, por ter subestimado o custo deste caminho. Entretanto, foi descoberto um outro caminho (C) que pode ser explorado agora como a melhor possibilidade. Ou seja, subestimar a estimativa pode implicar em mais procuras, mas leva a encontrar os caminhos mais curtos. Agora, se, pelo contrário, superestimarmos as estimativas, podem acontecer situações como as expostas na figura da direita, onde a árvore em B foi sendo expandida pois apresentava menor custo que as estimativas para C e D. Entretanto, suponha que existe um caminho entre D e o objetivo de custo 2. Ele não será encontrado porque superestimamos h(D) de tal forma que, antes de explorar o caminho de D, encontraremos outras soluções piores.

Na prática, estimar corretamente pode ser difícil e garantir que a estimativa é otimista, também. Assim, o teorema provado poderia ser de pouca utilidade. Pensando nisso, os autores provaram também um corolário mais útil (degradação suave da admissibilidade), que enuncia:

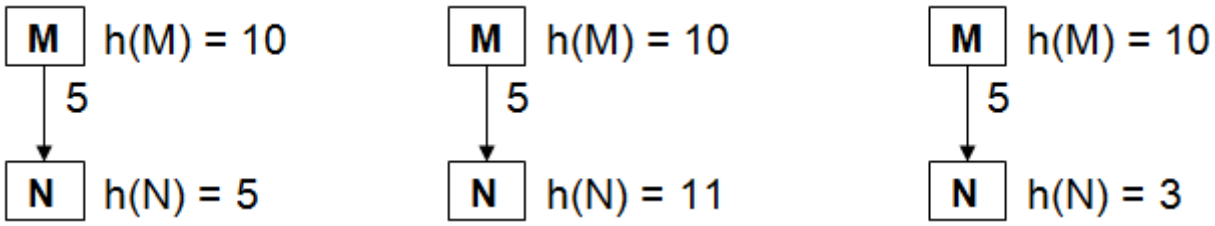
Se a estimativa for otimista na maioria das vezes então o algoritmo A* encontra o melhor caminho também na maioria das vezes.

No problema do labirinto, podemos tomar como estimativa de custo à distância do compartimento até o compartimento de saída sem considerarmos haverem paredes entre eles (que é o melhor possível). Assim a estimativa sempre é melhor ou igual ao custo real. No problema do jogo de nove posições a estimativa otimista poderia ser à distância da casa em que está a posição analisada até a casa em que está a posição desejada. A aplicabilidade da estimativa pode levar em consideração características particulares do problema como facilidades de movimentação em uma determinada direção.

Um resumo dos resultados que os autores provaram sobre o algoritmo:

- Seja h*(N) o custo do caminho ótimo. Se h(N)<h*(N) para todo N, então A* é admissível; (admissibilidade)
- Seja C* o custo da solução ótima. Se A* é admissível, então todo N com f(N)>C* será fechado; (limitação no número de nós fechados)
- Se h(N)=h*(N) para todo N, então apenas os vértices com caminho ótimo serão fechados. (otimalidade)

Ex:



Uma heurística é consistente se para cada par de nós M e N, de um mesmo ramo, h(M) – h(N) < C(M,N), onde C(M,N) é o custo de M a N.

Exercício: Considere o labirinto da figura 3.7, para o qual queremos achar a saída. Considere o custo do caminho já percorrido igual à distância medida quando se caminha, apenas em linhas paralelas às paredes, do centro de uma sala para o centro de outra sala adjacente, como anteriormente. Agora considere a estimativa do caminho a percorrer para cada compartimento, como a distância, medida da mesma forma que anteriormente, entre este compartimento e o compartimento T, sem considerar a existência ou não de passagens. Note que esta estimativa é otimista, na medida em que o custo real é no mínimo este (se existirem portas em todo o caminho) ou maior, caso não haja portas em todos os compartimentos entre o atual e T. Construa a árvore de busca e encontre o caminho com seu respectivo custo, usando esta estimativa e o algoritmo A*.