Backtracking

O que é?

A principal ideia é construir soluções, um componente de cada vez, e avaliar essas soluções parcialmente construídas como segue. Se uma solução parcialmente construída pode ser desenvolvida sem violar as restrições do problema, isso é feito tomando-se a primeira opção legítima restante para o próximo componente. Se não houver opção legítima para o próximo componente, nenhuma alternativa para qualquer componente restante precisam ser considerados. Nesse caso, o algoritmo retrocede para substituir o último componente da solução parcialmente construída com sua próxima opção. É conveniente implementar esse tipo de processamento construindo uma árvore de escolhas sendo feitas, chamada de state space-tree.

Algoritmo

if X[1..i] is a solution write X[1..i] else x ∈ Si+1 consistent with X[1..i] and the constraints do X[i + 1]← x Backtrack(X[1..i + 1])

n-queens

Começamos com o tabuleiro vazio e depois colocamos a rainha 1 no primeiro possível posição de sua linha, que está na coluna 1 da linha 1. Em seguida, colocamos a rainha 2, após tentando sem sucesso as colunas 1 e 2, na primeira posição aceitável para ela, que é o quadrado (2, 3), o quadrado na linha 2 e na coluna 3. Isso prova ser um beco sem saída porque não há posição aceitável para a rainha 3. Portanto, o algoritmo retrocede e coloca a rainha 2 na próxima posição possível em (2, 4). Então a rainha 3 é colocada em (3, 2), o que prova ser outro beco sem saída. O algoritmo então retrocede todos os o caminho para a rainha 1 e move-a para (1, 2). Rainha 2 então vai para (2, 4), rainha 3 para (3, 1) e rainha 4 a (4, 3), que é uma solução para o problema.

Branch and Bound

Comparado ao backtracking, o branch-and-bound requer dois itens adicionais:
1.uma maneira de fornecer, para cada nó de uma árvore de espaço de estados, um limite no melhor valor da função objetivo em qualquer solução que pode ser obtida adicionando componentes adicionais para a solução parcialmente construída representada pelo nó
2.o valor da melhor solução vista até agora

Circuito Hamiltoniano

Sem perda de generalidade, podemos supor que, se existe um circuito hamiltoniano, começa no vértice a. Da mesma forma, tornamos o vértice a a raiz do state-space.tree. O primeiro componente da nossa solução futura, se existir, é um primeiro vértice intermediário de um circuito hamiltoniano a ser construído. Usando o ordem alfabética para quebrar o empate triplo entre os vértices adjacentes a a, nós selecione o vértice b. De b, o algoritmo prossegue para c, depois para d, depois para e e finalmente para f, o que prova ser um beco sem saída. Assim, o algoritmo retrocede de f para e, depois para d e depois para c, que fornece a primeira alternativa para o algoritmo perseguir. Passar de c para e acaba se revelando inútil, e o algoritmo precisa retroceder de e para c e depois para b. A partir daí, segue para os vértices f , e, c, e d, a partir do qual pode retornar legitimamente a a, produzindo o circuito hamiltoniano a, b, f, e, c, d, a. Se quiséssemos encontrar outro circuito hamiltoniano, poderíamos continuar esse processo retrocedendo da folha da solução encontrada.

Exemplos

Knapsack: dados n itens de pesos conhecidos wi e valores vi, i = 1, 2, . . . , n, e uma mochila de capacidade W, encontre o subconjunto mais valioso dos itens que cabem na mochila.
Travelling Saleman:Seremos capazes de aplicar a técnica branch-and-bound a instâncias doproblema do caixeiro viajante se chegarmos a um limite inferior razoável nos comprimentos. Um limite inferior muito simples pode ser obtido encontrando o menor elemento na matriz de distância intermunicipal D e multiplicá-lo pelo número de cidades n.