Algoritmo de Dijkstra

(**SCHWARZ, Gaston Adair.** Grafos – Introdução e Processos de Busca**. Apostila, 1998)**

Considerações

Este algoritmo foi desenvolvido originalmente para grafos finitos com custos positivos, situação em que é admissível.
Se o grafo tem *n* nós, e nenhum arco tem custo negativo, este algoritmo fecha no máximo *n* nós antes de achar a solução ótima, ou concluir pela inexistência de uma solução viável.
Se for admitida a existência de custos negativos, mas não circuitos de custos negativo, uma modificação no algoritmo transforma-o novamente em admissível. Uma modificação na regra de parada torna-o completo (mas não admissível) também para grafos infinitos.
Vários autores mencionam que o algoritmo de Dijkstra é considerado o mais eficiente algoritmo não heurístico para resolver problemas de busca de caminho de mínimo custo em grafos com custos não negativos.
A técnica que este algoritmo emprega para determinar o caminho de mínimo custo manipula dois conjuntos de nós, um contendo os nós rotulados como abertos e outro contendo os fechados, representados por A e F, respectivamente.
Os nós abertos (ou temporários) são aqueles que ainda podem ser explorados, isto é, os custos associados aos caminhos para atingí-los podem ainda ser alterados no decorrer da busca. Já os nós fechados (ou permanentes), não serão mais reabertos ou rotulados como abertos. Isso quer dizer que um nó, uma vez rotulado como aberto, não tem mais como ser alterado, porém, não quer dizer que faz parte do caminho de mínimo custo, apenas é um candidato para compor o referido caminho.
Em cada iteração, o último nó fechado serve como base para a busca de novos caminhos.
Numa iteração, somente um nó pode ser transferido para o conjunto de abertos para o conjunto de fechados e a condição necessária é que o custo de ser caminho seja o menor entre todos da referida iteração.
Como somente um nó é fechado por iteração, pode ocorrer que dois ou mais arcos apresentem custos iguais. Mesmo neste caso é preciso selecionar apenas um nó e esta escolha deve ser efetuada de forma que não interfira no andamento eficiente da busca, nem deixe de favorecer o conjunto de nós terminais. Quando somente um caminho de mínimo custo é desejado, pode-se fazer uma escolha arbitrária. Neste caso, então, a escolha do nó mais recente é pode ser considerada como uma forma de seleção ou desempate desses arcos, sem prejudicar, no entanto, os nós terminais.

Algoritmo de Dijkstra

Inicialização

1. Descrição detalhada de cada nó $x_i \in X$, incluindo nós iniciais $s \in S$ e terminais $t \in T$.
2. Definição de $\Gamma(x_i)$.
3. Criar duas listas: A = lista de nós abertos = ϕ;
   F = lista de nós fechados = ϕ.
4. Criar um conjunto de apontadores (predecessores) $P(x_i)$.
5. Criar um vetor de custos $f(x_i)$.

Algoritmo (com custos positivos)

1. Faça A = lista de S e $\forall s \in S$ faça $g(s) = 0$ e $P(s) = \phi$.
2. Se $A = \phi$, pare com fracasso.
   Se não, tome $v \in A$ tal que $g(v) = \min g(n)\ \forall n \in A$.

Regra de desempate: 1) Escolha um nó terminal;
2) Utilize qualquer critério.

3. Remova v de A e inclua em F. Se $v \in T$, pare com sucesso.
   Se não, gere $\Gamma(v)$. Se $\Gamma(v) = \phi$, volte a 2.
4. Para cada $m \in \Gamma(v)$ faça $f(m) = g(v) + C(v,m)$.
   Se [$m \in A$ e $g(m) \leq f(m)$] ou [$m \in F$], tome o próximo sucessor e repita o teste. Do contrário, faça
   $g(m) = f(m)$;
   $P(m) = v$; e
   $A = \{m\} \cup A$.
5. Volte a 2.

Saída de Resultados

1. Recuperar a solução através dos apontadores $P(x_i)$.
2. Emitir relatório.