Macroentrega 1

Tales Ribeiro Magalhães

O problema proposto possui uma certa quantidade de requisições para transporte. Cada uma delas deve conter, além de outros parâmetros, local de coleta, de entrega, janela de tempo, tempo para entrega, tempo de serviço e demanda e essa demanda deve ser complementar ao local de coleta.

Considerando que exista apenas um depósito onde será a origem e o destino final dos veículos, tendo sua própria janela de tempo, podemos definir o tamanho do planejamento das rotas ou o tempo máximo que uma rota pode ter.

Para este problema, uma versão modificada do algoritmo de Dijkstra pode ser usada, ao incorporar as janelas de tempo e os requisitos de coleta e entrega no algoritmo. Nessa versão, o algoritmo será usado para encontrar o menor caminho entre os nós, levando em consideração as os requisitos incorporados ao algoritmo como citado anteriormente. O algoritmo manterá informações do tempo de cada nó e atualizará conforme é movimentado pelos nós garantindo que as janelas de tempo definidas não sejam violadas. A solução também armazenará as informações dos itens sendo coletados e entregues e consideraria apenas caminhos considerados possíveis dado o estado dos itens.

Para incorporar novos atributos ao algoritmo, necessários para o escopo do problema podemos:

1. Nós e Arestas: Cada nó no grafo representa a localização que quer coleta ou entrega de um item, ou mesmo, o ponto inicial/final do veículo. Cada aresta representa o movimento de um nó a outro e possui um custo associado que representa o tempo ou distância.

2. Janelas de Tempo: Para cada nó, teremos os tempos iniciais e finais permitidos para chegada de um veículo.

3. Custo das Arestas: Ao atualizar o custo das arestas, deve-se considerar o tempo necessário para percorrer um nó a outro, bem como as restrições de janela de tempo. Será calculado o tempo de chegada em cada nó baseando-se no tempo de partida do nó anterior e o custo da aresta. Se o tempo de chegada está fora da janela de tempo, o caminho será rejeitado e o algoritmo move-se para o próximo nó.

4. Parada: Quando todos os nós já tiverem sido visitados, o algoritmo poderá parar e retornar o menor caminho que satisfaz as restrições de janela de tempo e entrega e coleta.

PSEUDO-CÓDIGO

```
custo = [ ] // Armazena o custo para cada nó
visitado = [ ] // Armazena nós visitados
anterior = [ ] // Armazena os nós anteriores a cada nó

// Inicialização
para cada nó no grafo
        custo[nó] = infinito
        visitado[nó] = false
        anterior[nó] = vazio

custo[inicio] = 0

Enquanto todos os nós não forem visitados
        nóAtual = nó com menor custo dentre os visitados
        visitado[nóAtual] = true

// custo() funcao para calcular custo

        Para cada vizinho do nóAtual
                tempoChegada = tempoAtual + custo(nóAtual, nóVizinho)
                Se tempoChegada está dentro da janelaTempo
                        Se custo[nóVizinho] > custo[nóAtual] + custo(nóAtual,vizinho)
                                custo[nóVizinho]=custo[nóAtual] + custo(nóAtual,vizinho)
                                anterior[nóVizinho] = nóAtual
        …
        …
        caminho = [ ]
        nóAtual = final
        Enquanto não chega no inicial
                caminho.insere(nóAtual)
                nóAtual = anterior[nóAtual]
        caminho.insere(inicial)
```

O algoritmo não proverá a solução ótima para o problema mas pode ser um bom começo. Modificações podem e devem ser feitas posteriormente a fim de otimizar esses resultados.