

## 1. Ideias Gerais

O problema retratado no trabalho prático condiz com o **Problema do Caixeiro Viajante (TSP)**, variante métrico, que consiste em encontrar o(s) ciclo(s) hamiltoniano(s) (passagem por cada vértice do grafo exatamente uma vez, retornando ao vértice inicial) de menor(es) custo(s) em um grafo ponderado direcionado ou não. Mais especificamente, estamos trabalhando com o conceito de multiobjective capacitated pickup and delivery problem with time windows (MOCPDPTW).

Todas as instâncias possuem grafos **completos**, **direcionados** e **ponderados** (positivo), este considera-se o tempo, diretamente proporcional ao custo.

Há 6 restrições que devemos respeitar para a solução do problema:

- Restrição 1: Em um pedido  $r = (i, j)$ ,  $j$  não pode ser visitado antes de  $i$ ;
- Restrição 2: Cada veículo deve partir e retornar ao ponto de origem  $\{0\}$ , seguindo o tempo  $[a0, b0]$ ;
- Restrição 3: O tempo de chegada de certo veículo ao ponto  $i$  ou  $j$  não pode exceder o  $ltw$  de tal ponto. Caso chegue antes de  $etw$  de  $i$  ou  $j$ , o veículo deve esperar até o respectivo tempo para realizar a ação;
- Restrição 4: Sendo um pedido  $r = (i, j)$ ,  $i$  e  $j$  devem ser visitados exatamente uma vez;
- Restrição 5: Sendo um pedido  $r = (i, j)$ , o mesmo veículo que passa por  $i$  passar por  $j$ ;
- Restrição 6: O somatório das demandas dos pontos atendidos por certo veículo não pode ultrapassar sua capacidade.

Ao decorrer do curso estudamos alguns algoritmos que proporcionam possíveis soluções para Problemas do Caixeiro Viajante, são eles:

- vizinho mais próximo;
- inserção mais próxima;
- gulosa;
- RSL (TSP métrico);
- Christophides (TSP métrico);
- k-opt.

Toda a proposição que virá a seguir pode ser considerada mera especulação, pois não se sabe com certeza se virá a funcionar completamente para a solução do problema, certas teorias podem se mostrar incorretas.

Pelo livro de referência do curso (Goldbarg & Goldbarg) o problema descrito é **NP-DIFÍCIL**, o livro não apresenta muitas citações para resolver o problema descrito em específico, logo foi pesquisado em artigos acadêmicos ideias que poderiam solucionar tal problema.

É necessário o desenvolvimento de um algoritmo heurístico (que prioriza velocidade do que otimização) onde baseado em várias rotas geradas automaticamente, veremos se tais rotas são eficazes e resolvem o problema.

O primeiro problema está em gerar uma rota aleatória que respeite todas as restrições. Foi gerado nessa etapa somente uma ideia dessa função, porém se implementado pensando estritamente nas restrições, uma hora ou outra, gerará uma solução em que se pode transitar.

Logo, a primeira etapa de implementação da macro entrega 2 se baseia em **reestruturar** a função que gera rotas aleatórias levando em conta as restrições do problema.

A segunda etapa se baseia em **reorganizar** as rotas a fim de deixá-las com o menor custo. Ambas etapas podem ser feitas em paralelo por meio do pseudocódigo abaixo.

## 2. Pseudocódigo (feito com apoio de artigos acadêmicos)

```
1: rotas ← qtd_veiculos listas vazias
2: seja pedidos pares de coleta e entrega
3: inicialize cada rota em rotas com um pedido qualquer de pedidos
4: remova cada pedido inserido de pedidos
5: enquanto todos os pedidos ainda não foram inseridos faça
6:   novaSolucao ← nada
7:   melhorPedido ← nada
8:   para cada index, rota em rotas faça
9:     escolha um pedido com H
10:    insira pedido em novaRota com I
11:    aprimore a novaRota com Q
12:    se novaRota é viável então
13:      se novaSolucao é nada então
14:        novaSolucao ← rotas
15:        novaSolucao[index] ← novaRota
16:        melhorPedido ← pedido
17:      se não
18:        SolucaoTemp ← rotas
19:        SolucaoTemp[index] ← novaRota
20:        se SolucaoTemp é melhor que novaSolucao então
21:          novaSolucao ← SolucaoTemp
22:          melhorPedido ← pedido
24:    se novaSolucao é nada então
25:      melhorPedido ← qualquer pedido de pedidos
26:      junte rotas com uma nova rota contendo melhorPedido
27:    se não
28:      rotas = novaSolucao
29:    remova melhorPedido de pedidos
30: retorne rotas
```

Sendo H uma heurística de inserção, I um operador de inserção e Q um operador de busca local.