1. Objetivo:

Foi considerado um programa em C++ com os seguintes arquivos para a resolução da Macro entrega 1. Dentre os arquivos (main.cpp {o algoritmo principal} / classe.h {todas as funções e classes da resolução}). Foi necessário o uso de bibliotecas como vector, algorithm, fstream para a resolução.

2. Geral

O problema do trabalho prático, é um pouco quanto complexo. Ele tem sua solução baseada no TSP (O Problema do Caixeiro Viajante), que visa em conseguir chegar em seu menor custo possível.

Com ajuda de artigos acadêmicos internacionais, foi elucidado que a solução desse problema é utilizado a **MOCPDPT** (multiobjective capacitated pickup and delivery problem with time windows), numa forma mais visível e compreensível para a solução do problema. Nesse artigo, consegui aplicar uma resolução para a macro entrega 2.

Além do algoritmo de auxílio para a Macro entrega 2, temos também algoritmos que foi de base para a resolução do TSP, tais como:

A. **Vizinho mais Próximo;**
B. **ACO;**
C. **Clarke e Wright (Heurística Construtiva);**
D. **RSL;**
E. **OPT;**
F. **Inserção mais próxima;**

Com as instâncias colocadas no trabalho prático, temos 6 restrições para a resolução do problema da Macro entrega 1 para ser desenvolvida, listadas:

Obs: as variáveis utilizadas, estão propostos no enunciado do trabalho, como i,j,ltw, etw, p. Então estarei utilizando-o para a explicação de cada restrição utilizada.

A. **Precedência de Coleta e Entrega:** para uma entrega R, onde é (i,j), o R em hipótese alguma pode ter visitado i;

B. **Origem e Horário de Serviço:** Veículo F retorna ao ponto de origem, com o retorno do tempo

C. **Janelas de tempo:** O veículo não pode chegar ao ponto i e j no ltw proposto no final do enunciado. Precisa de permissão para prosseguir.

D. **obrigatoriedade e exclusividade de visita:** como na restrição A, i e j só pode ser visitado unicamente e estritamente uma vez.

E. **Atendimento de Pedido:** a visita de i pertencente ao p tem que passar estritamente pelo mesmo veículo.

F. **Capacidade do Veículo:** seu somatório não pode exceder sua capacidade máxima em seu veículo.

3. Pseudocódigo para a resolução do problema (Macro entrega 2)

```
1 -  caminho ← quantidade_de_automoveislistas vazias
2 -  // comece cada caminho em destinos com um pacote para qualquer tipo de entrega
3 -  // enquanto todos os pacotes ainda não foram colocados na função o algoritmo segue fazendo:
4 -      solucao ← nada_a_fazer
5 -      pacote_idealado ← nada_a_fazer
6 -      para cada catalogo, rota em trajetos faça
7 -              escolha um pacote com "X" // inserção
8 -              insira pacote em novoTrajeto com "I" //operação de inserção
9 -              aprimore a novoTrajeto com O //Operação de busca
10 -             if novoTrajeto é viável então
11 -                     if solucao é nada então
12 -                             solucao ← rotas
13 -                             solucao[catalogo] ← novoTrajeto
14 -                             pacote_idealado ← pacote
15 -                     if else
16 -                             solucao_temporario ← trajeto
17 -                             solucao_temporario←[catalogo] ← novoTrajeto
18 -                             if SolucaoTemp é ideal que novaSolucao então
19 -                                     novaSolucao ← SolucaoTemp
20 -                                     idealpacote ← pacote
21 -     if novaSolucao é nada então
22 -             idealpacote ← qualquer pacote de pacotes
23 -             junte rotas com uma nova rota contendo idealpacote
24 -     if não
25 -             trajeto = novaSolucao
26 -             remover idealpacote de pacotes
27 - return trajeto
```

Obs2: Utilizei esse algoritmo incompleto da solução - então na macro entrega 2 pode haver diferenciação da aplicação desse pseudocódigo. Foi utilizado o algoritmo proveniente do artigo acadêmico MDPI da Mathematics.

Fonte:

https://www.mdpi.com/2227-7390/10/22/4308