Grupo:

Fabrício Alves Smargiasse – 202020117,

Otávio Rodrigues de Faria – 202020784,

Weverton Andrade Ferreira – 202020782.

EXPLICAÇÃO DO ALGORITMO

O algoritmo se desenvolveria da seguinte forma:

É colocado um caminhão em rotação;

É executado uma busca em largura, que verifica os pontos de entrega e coleta mais próximos do caminhão;

A partir do momento que são identificados os destinos mais próximos e os custos de cada rota, gerando duas possibilidades:

1 – Coleta;

- Verificações:

A – Se há coletas a serem realizadas;

B – Se o caminhão tem capacidade disponível;

B’ – Caso o caminhão não tenha capacidade há dois senários:

- ➔ Há uma entrega próxima que compense a janela de tempo da chamada do novo caminhão e que libere espaço suficiente para a nova coleta;
- ➔ Colocar um novo caminhão em circulação;

C – A janela de tempo está apta para ser coletada;

2 – Entrega;

- Verificações:

A – Se a mercadoria de entrega está no caminhão;

B - A janela de tempo está apta para ser descarregada;

A rota é calculada baseada no caminho mais curto, podendo variar conforme a capacidade e a janela de tempo que são validadas pelas verificações anteriores de mudo a tender as condições definidas pelo enunciado do trabalho;

O ciclo deve se repedir, até a janela de tempo se esgote e os caminhões voltem para a origem.

Como o problema em questão possui dois pesos nas arestas, vamos utilizar o algoritmo de Dijkstra. Este algoritmo é especificamente projetado para grafos com pesos positivos nas arestas e encontra o menor caminho a partir de um vértice para todos os outros vértices no grafo.

PSEUDOCÓDIGO

```
ALGORITMO(Grafo g, Vértice origem){
        Coloca o caminhão em rota
        Executa BFS buscando a coleta mais próxima
        distancia[origem] = 0
        Cria conjunto de vertices NaoVisitados
        Para cada vértice v do grafo g{
            Se v != origem{
                distancia[v] = INFINITO
            }
            anterior[v] = INDEFINIDO
            Insere v em NaoVisitados
        }
        Enquanto NaoVisitados não for VAZIA{
            Procura vértice com menor distância, denominado u
            Remove u de NaoVisitados
            Para cada i vizinho de u{
                custo = distancia[u] + g[u, i]
                Se i é ponto de coleta && o caminhão tem capacidade && a
                janela de tempo permite{
                    adiciona a carga no caminhão
                }
                Se custo < distancia[i] {
                    distancia[i] = custo
                    anterior[i] = u
                }
            }
        }
        return distancia, anterior
}
```