- Gabriel Fernandes Silva Vieira - 201920340
- Pedro Gabriel Camargo Calisto - 201920337
- Tulio Oliveira Cruz - 202020094

Algoritmos em Grafos

O algoritmo proposto consiste em um algoritmo construtivo pseudo-aleatório, onde deve-se iniciar o algoritmo executado o algoritmo de menor de Floyd Warshall, com o objetivo de computar o tempo de percurso t_{ij} para todo $i,\ j\ \in\ V$.

Em nosso algoritmo iremos considerar que os pedidos estão ordenados pelo tempo inicial de atendimento da coleta. Caso dois ou mais pedidos possuam o mesmo tempo inicial, ordenamos pelo tempo final de atendimento da entrega.

Temos duas informações associadas a cada veículo $f\ \in\ \{1,\ ..,\ numVeiculos\}$: t_f e p_f, onde p_f é a posição atual do veículo e t_f o tempo que finalizou o último atendimento. A cada iteração iremos alocar um pedido $r\ =\ (i,\ j)$ a um veículo f desde que ele atenda às restrições do problema. Desta forma iremos verificar se a demanda atendida não irá ultrapassar a capacidade ($d_i\ \leq q_f$), o veículo conseguirá estar no local antes do final do fim da janela da coleta ($t_f\ +t_{p_f i}\ \leq b_i$), o veículo conseguirá estar no local antes do final do fim da janela da entrega ($t_f\ +t_{p_f i}\ +s_i\ +\ t_{ij}\ \leq b_j$) e se, caso necessário, o veículo conseguirá retornar ao depósito dentro do horizonte de roteirização ($t_f\ +t_{p_f i}\ +\ t_{ij}\ +\ s_i\ +\ s_j\ +t_{j\{0\}}\ \leq H$).

Seja G o conjunto de veículos que podem atender o pedido r. Caso $|G|\ =\ 0$, significa que não existe nenhum veículo que possa atender o pedido r e podemos pausar a construção da solução. O algoritmo irá verificar se estamos na primeira iteração, caso positivo iremos selecionar $g\ \in\ G$ com menor valor $c_{p_g i}\ +\ c_{ij}$. Caso contrário, iremos selecionar o veículo $g\ \in\ G$ de forma aleatória dado pela distribuição uniforme $\pi_g\ =\ \frac{\sum_{g\in G} c_{p_g i} + c_{ij}}{c_{p_g i} + c_{ij}}$, onde quanto menor for o custo do veículo para atender esta entrega maior será a probabilidade de ser selecionado.

Caso a solução gerada na iteração for viável iremos verificar se ela possui ou não melhor valor de função objetivo que a melhor solução já encontrada. Em caso positivo, a melhor solução já encontrada passa a ser a solução desta iteração.

Caso tenhamos feito $ITER_{MAX}$ iterações e ainda não ter encontrado nenhuma solução viável iremos reiniciar o contador de iterações e aumentar em uma unidade o número de veículos a ser utilizado. Caso já tenha encontrado uma solução viável e percorrido $ITER_{MAX}$ iterações, iremos retornar a melhor solução encontrada.

Abaixo o pseudocódigo do algoritmo é descrito:

Algoritmo Proposto

Entrada: Grafo G(V, A), conjunto de pontos de coleta P, conjunto de pontos de entrega D, conjunto de pedidos R.

Saída: Uma relação B que associa cada pedido $r \in R$ a um veículo.

01. t_{ij} ← Floyd_Warshall(G)
02. Ordene os pedidos $R = \{r1, ..., rk\}$ de forma que $(a_{rl_i} < a_{rm_i})$ ou $((a_{rl_i} = a_{rm_i})$ e $(b_{rl_i} \leq b_{rm_j}))$ para todo $l < m$.
03. SolViavel ← F
04. iter ← 0
05. numVeiculos ←1
06. Inicie a relação B com ∞ veículos
07. **Enquanto** $(iter < ITER_{MAX})$ **faça**
08. Inicie a relação B' com numVeiculos veículos
09. **Para todo** $f \in \{1, .., numVeiculos\}$ **faça**
10. $p_f \leftarrow \{0\}$ // Deposito
11. $t_f \leftarrow 0$
12. **Para todo** $r = (i, j) \in R$ **faça**
13. $G \leftarrow \{\}$
14. **Para todo** $f \in \{1, .., numVeiculos\}$ **faça**
15. **Se** $(d_j \leq q_f)$ e $(t_f + t_{p_f i} \leq b_i)$ e $(t_f + t_{p_f i} + s_i + t_{ij} \leq b_j)$ e $(t_f + t_{p_f i} + t_{ij} + s_i + s_j + t_{j\{0\}} \leq H)$ **então**
16. $G \leftarrow G \cup \{f\}$
17. **Se** $|G| == 0$ **então**
18. **break**
19. **Senão**
20. **Se** iter == 0 **então**
21. Selecione $g \in G$ com menor valor $c_{p_g i} + c_{ij}$
22. **Senão**
23. Atribua a cada elemento $g \in G$ o valor $\pi_g = \frac{c_{p_g i} + c_{ij}}{\sum_{g \in G} c_{p_g i} + c_{ij}}$.
24. Selecione aleatoriamente $g \in G$ dado pela distribuição uniforme π_g.
25. $p_g \leftarrow j$
26. $t_g \leftarrow t_g + t_{p_g i} + t_{ij} + s_i + s_j$
27. $B'[r] = g$
28. SolViavel ← SolViavel **ou** Viavel(B')
29. **Se (**Viavel(B') == V) e (ValorSolucao(B') < ValorSolucao(B)) **então**
30. B ← B'
31. $iter \leftarrow iter + 1$
32. **Se** (SolViavel == F) e (iter == $ITER_{MAX}$) **então**
33. $iter \leftarrow 0$
34. $numVeiculos \leftarrow numVeiculos + 1$
35. **Retorne** B