Minimização de rotas com prioridade na entrega

Erick L. Novais[1], **Rebeca B. Neto**[1], **João Carlos R. Couto**[1]

[1]Instituto de Ciências Exatas e Informática - Pontifícia Universidade Católica de Minas Gerais (PUCMG) - Belo Horizonte - MG - Brasil

1. Introdução

O problema escolhido foi otimização de rotas de e-commerce com prioridade na entrega. Nesse contexto, queremos minimizar a distância percorrida, porém respeitando a prioridade. Esse problema se encaixa na lógica do caixeiro viajante o qual tem como objetivo encontrar o ciclo hamiltoniano de um grafo, definido como a rota mais curta em que se pode alcançar todos os vértices de um grafo uma única vez. No nosso caso, um veículo de entregas parte de um depósito, e deve fazer entregas para o depósito de cada cidade, e, por fim, retornar ao depósito original. Além disso, cada depósito possui uma prioridade ≥ 1, sendo 1 a maior prioridade. Escolhemos esse problema porque o e-commerce é uma área crescente de grande potencial para otimizações.

2. Trabalhos relacionados

O artigo [Silva and Barcelos] busca o desenvolvimento de estratégias que proporcionem uma redução do tempo e da distância percorrida podem elevar a lucratividade e proporcionar vantagem competitiva. Isso se torna mais evidente em uma distribuidora, onde o transporte é uma das principais tarefas. Sendo assim, o presente trabalho resulta-se de um estudo de caso do processo de entrega em uma distribuidora. Inicialmente, foram coletados dados sobre o transporte de mercadorias, realizado pela organização, para a construção de um modelo matemático, de acordo com a metodologia do Problema do Caixeiro Viajante. Posteriormente, converteram essa modelagem em uma programação computacional a fim de se obter a rota ótima através do solver AMPL.

O artigo [Tian 2017] tem como objetivo minimizar o custo de transporte enquanto satisfaz os requisitos do cliente. Com base nisso, as condições de tráfego são introduzidas para otimizar a rota de distribuição, e a efetividade do algoritmo é testada e verificada por uma simulação.

Em um outro trabalho [Porfírio et al.], o problema do caixeiro viajante foi aplicado ao grupo crítico do sistema de entregas de um restaurante. Os autores realizaram um estudo de caso em um restaurante que realiza entrega de refeições em uma cidade da zona da mata mineira, coletando os dados durante o período de 1 mês. Os bairros de entrega foram divididos em 5 grupos e através do Microsoft Excel utilizou-se a ferramenta solver para determinar a melhor rota a ser percorrida pelo entregador de maneira a atender todos os clientes.

3. Solução

O modelo matemático escolhido para a modelagem do problema foi o proposto por Dantzig-Fulkerson-Johnson [Dantzig et al. 1954].

Minimizar:

$$z = \sum_{j=1}^{n} \sum_{i=1}^{n} c_{ij} x_{ij}$$

Sujeito a:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in N$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in N$$

$$\sum_{i,j \in S}^{n} x_{ij} \leq |S| - 1 \qquad \forall S \subset N$$

$$x_{ij} \in 0, 1 \qquad \forall i, j \in N$$

Sendo que X_{ii} não existe, pois não se pode escolher um caminho que sai de uma cidade e volta nela mesma. Além disso, tem-se n-1 variáveis e $O(2^n)$ restrições, o que gera o problema de escalabilidade. Nesse modelo, S é um subgrafo de G,em que $|S|$ representa o número de vértices desse subgrafo.

O programa foi desenvolvido em Python, usando a bibliotecas mip. Ela é usada para modelar e solucionas problema de programação linear inteira. Após a declaração do objeto da biblioteca, adicionamos a variável binária de decisão x que irá determinar se a rota entre as cidades será usada. Em seguida, declaramos a nossa função objetivo, que nesse caso é minimizar a rota, ela é dado por c[i][j] * x[i][j], onde c é a matriz que contém as distâncias dos locais. Para as restrições foi usada a função *add_var*, nelas precisamos garantir que para cada cidade só entre/saia uma vez e além disso é necessário eliminar os sub-caminhos.

```
model = Model()

# variáveis binárias indicando se arco (i,j) é usado na rota
x = [[model.add_var(var_type=BINARY) for j in V] for i in V]

# variáveis contínuas para prevenção de sub-rotas: cada cidade terá
# um identificador numérico maior na rota, excetuando-se a primeira
y = [model.add_var() for i in V]

# função objetivo: minimizar custo total
model.objective = minimize(xsum(c[i][j]*x[i][j] for i in V for j in V))

# restrição : sair de cada cidade somente uma vez
for i in V:
    model += xsum(x[i][j] for j in V - {i}) == 1

# restrição : entrar em cada cidade somente uma vez
```

```
for i in V:
    model += xsum(x[j][i] for j in V - {i}) == 1

# eliminação de sub-rotas
for (i, j) in product(V - {0}, V - {0}):
    if i != j:
        model += y[i] - (n+1)*x[i][j] >= y[j]-n
```

Para executar o algoritmo chamamos a função *optimize()*, optamos passar o parâmetro de limite de tempo caso seja um problemas com grande quantidade de cidades. O cálculo do resultado da otimização fica no atributo *objective_value*, a matriz x também é atualizada de acordo com a resposta obtida sendo acessada com x[nc][i].x.

```
# otimização com limite de tempo
model.optimize(max_seconds=limiteTempo)

# verificando se ao menos uma solução válida foi encontrada
if model.num_solutions:
    out.write('route with total distance %g found: %s'
        % (model.objective_value, listaCidades[0]))
    nc = 0
    while True:
        nc = [i for i in V if x[nc][i].x >= 0.99][0]
        out.write(' -> %s' % listaCidades[nc])
        if nc == 0:
            break
```

4. Aplicação

O aplicativo possui uma interface simples apenas com um botão para calcular a menor rota. Todos as configurações e informações são inseridas nas abas Config e Distancia no arquivo *Ecommerce - Layout.xlsx*. Na aba de configuração é possível definir a quantidade de locais e a prioridade de cada um deles, sendo o menor número indica maior prioridade. O limite de tempo padrão no código é 60 segundos caso não tenha sido preenchido no Excel. Os resultados do algoritmo também são colocados nessa aba.

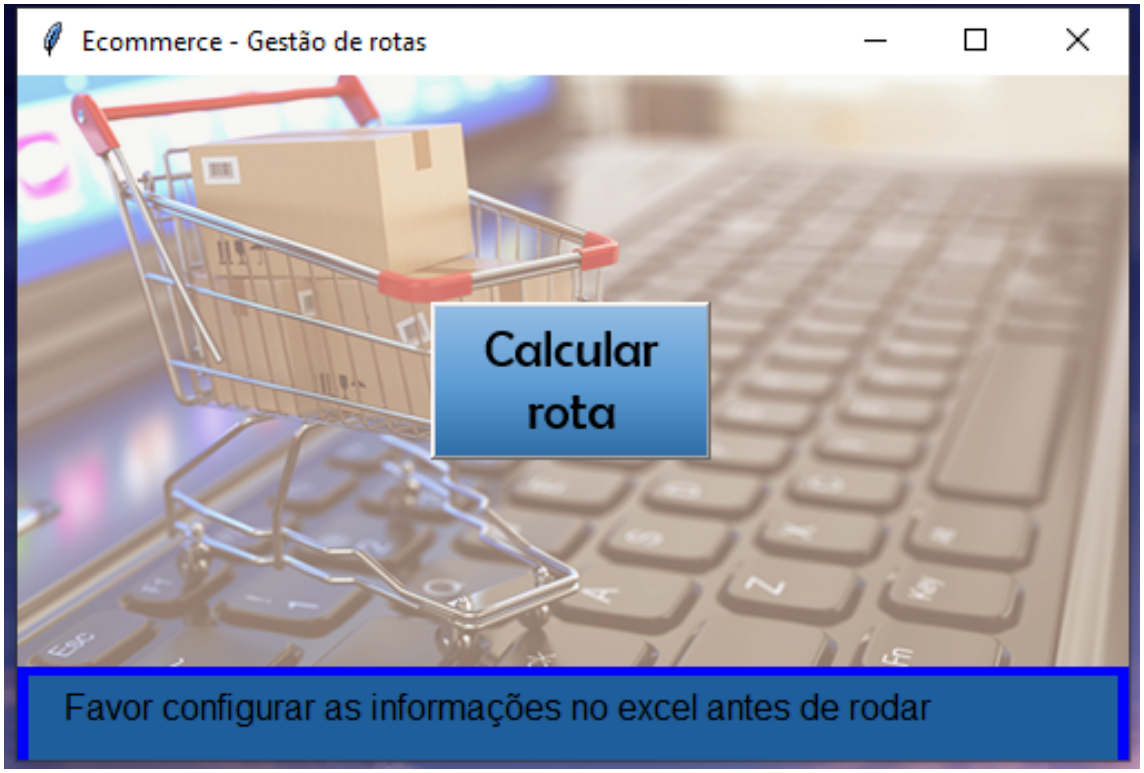


Figura 1. Menu do aplicativo

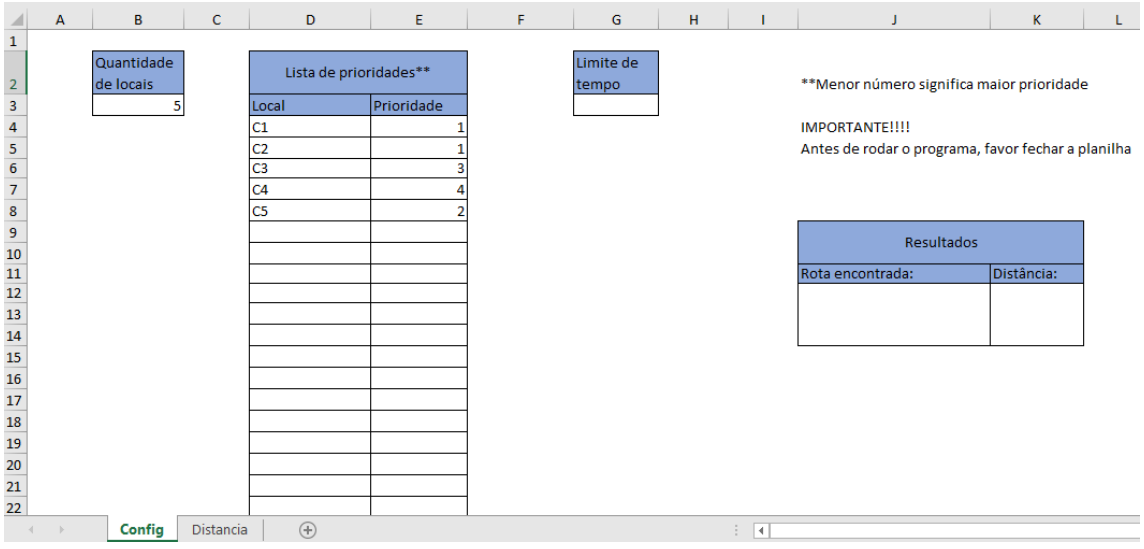
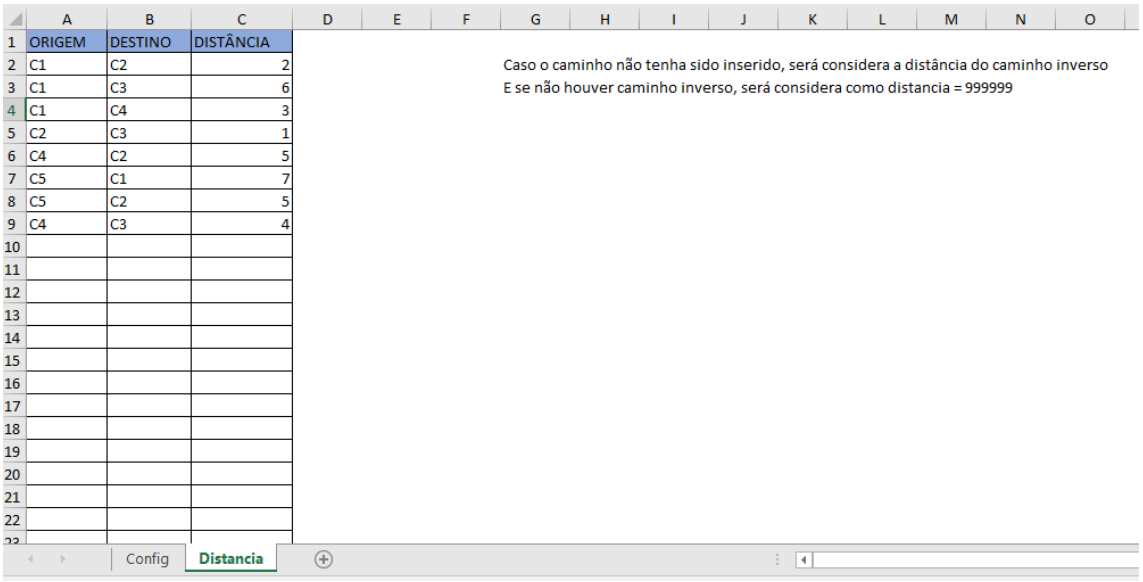


Figura 2. Aba de configurações do Excel

Na segunda aba é necessário inserir as distancias de cada caminho entre os locais. Caso um caminho não tenha sido inserido, será a considerada a distância do caminho inverso e se não houver um, será considerada como 999999 para que o algoritmo evite escolher uma rota que não existe.



	A	B	C
1	ORIGEM	DESTINO	DISTÂNCIA
2	C1	C2	2
3	C1	C3	6
4	C1	C4	3
5	C2	C3	1
6	C4	C2	5
7	C5	C1	7
8	C5	C2	5
9	C4	C3	4

Figura 3. Aba das informações das distâncias do Excel

Resultados	
Rota encontrada:	Distância:
C1-> C5-> C2-> C3-> C4-> C1	20

Figura 4. Exemplo de exibição do resultado no Excel

5. Conclusão

Com o objetivo de implementar uma otimização em rotas de entrega tivemos como base o problema do menor ciclo hamiltoniano de um grafo, que encontra a menor rota que passa por todos os destinos e retorna ao início. Esse tipo de otimização nas entregas é extremamente importante, já que a distância percorrida traduz-se diretamente nos custos operacionais da empresa e também no tempo de entrega do produto aos clientes, dois dos fatores mais relevantes para esse tipo de negócio. Adicionamos também a função de prioridades. Como é muito comum nessa área que algumas entregas precisem ser feitas com mais urgência, achamos necessário a adição dessa funcionalidade no nosso modelo.

De forma geral obtemos resultados positivos. Muitas vezes o caminho encontrado pelo algoritmo não é o mais curto, mas isso se deve pela implementação das prioridades como um multiplicador das distâncias. Um aspecto negativo dessa solução é a falta de escalabilidade que toda implementação do problema do caixeiro viajante possui. Dessa forma, para trabalhos futuros é válido pesquisar bibliotecas que lidam melhor com

a questão de aumentar o número de cidades, além disso seria interessante melhorar a interface de forma que o Excel não fosse necessário e assim todos as informações seriam inseridas diretamente na aplicação.

Referências

[Dantzig et al. 1954] Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410.

[Porfírio et al.] Porfírio, V., Mariquito, J., Silva, L., and Nazaré, T. O problema do caixeiro viajante aplicado ao grupo crítico do sistema de entregas de um restaurante.

[Silva and Barcelos] Silva, V. and Barcelos, B. Aplicacao do problema do caixeiro viajante para otimizar rota de entrega em uma distribuidora.

[Tian 2017] Tian, Z. (2017). Analysis of route optimization based on genetic algorithm. pages 112–116.