Vítor Gabriel Reis Caitité

1 - Variable Neighborhood Descent (VND)

O método de descida em vizinhança variável explora o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança. O algoritmo implementado pode ser visto abaixo:

Algorithm 1 VND

Seja S_0 uma solução inicial e N o número de estruturas de vizinhança;
S $\leftarrow S_0$; // Solução atual
k $\leftarrow$ 1; // Tipo de estrutura de vizinhança
while $k \leq N$ **do**
    S' $\leftarrow$ MelhorVizinho(S, k) // Realiza uma busca local utilizando o algoritmo k
    if custo_atual $<$ melhor_custo **then**
        k $\leftarrow$ 1
        melhor_custo $\leftarrow$ custo_atual
        S $\leftarrow$ S'
    else
        k $\leftarrow$ k + 1
    end if
end while
return S

Neste trabalho foram testados 2 metodologias VND. A primeira utilizando os algoritmos de busca local Swap e 2-Opt. Já no segundo caso foram utilizados os algoritmos de busca local Swap, 2-Opt e 3-Opt. No algoritmo Swap as posições de 2 vértices no caminho são trocadas entre si. A troca que resultar no caminho de menor custo é realizada. Esse processo é repetido até não se conseguir mais diminuir o custo trocando-se 2 vértices. No algoritmo 2-opt, elimina-se 2 arestas não adjacentes, reconecta-as usando duas outras arestas (formando um ciclo) e verifica-se se houve melhora. Este processo é repetido para todos os pares de arestas. A melhor troca (o novo ciclo com menor custo) é então realizada. Já o 3-Opt começa com um caminho inicial e exclui 3 arestas. Estas arestas podem ser reconectadas com 8 combinações diferentes, incluindo caminho original. Este processo é repetido para todos os trios de arestas possíveis e a melhor troca é realizada. Assim, como nos algoritmos anteriores, o 3-Opt permanece executando até não se conseguir mais diminuir o custo.

2 - Resultados:

Os algoritmos foram implementado em Python (3.9). E os resultados estão expressos na forma "*média +/- desvio padrão*". O número de testes foi fixado em 50 ou um valor igual ao número de cidades (para arquivos com menos de 50 cidades). Os testes foram executados em um notebook com processador i7 7th Gen., 8 Gb de RAM e sistema Linux.

Table 1: Custo obtido por cada algoritmo em cada arquivo de teste.

Arquivo	VND (Swap + 2-Opt)	VND (Swap + 2-Opt + 3-Opt)
kroA150.tsp	29339.52 +/- 317.99	28551.49 +/- 407.18
kroB100.tsp	23716.96 +/- 245.83	23179.74 +/- 282.46
pr107.tsp	45895.00 +/- 640.47	44817.41 +/- 283.24
kroC100.tsp	21826.36 +/- 405.00	21370.17 +/- 432.72
rat99.tsp	1287.30 +/- 15.14	1274.29 +/- 16.23
st70.tsp	716.23 +/- 8.66	706.03 +/- 3.83
kroB150.tsp	27700.77 +/- 234.91	27200.85 +/- 158.69
kroB200.tsp	31737.76 +/- 453.04	31024.10 +/- 401.93
pr136.tsp	99544.41 +/- 841.10	99136.02 +/- 738.06
pr144.tsp	60462.56 +/- 422.21	60316.24 +/- 91.38
pr124.tsp	60748.45 +/- 531.13	59758.03 +/- 275.89
pr76.tsp	112670.69 +/- 2017.12	111779.27 +/- 657.15
kroD100.tsp	22206.66 +/- 251.37	21845.86 +/- 164.09
kroA200.tsp	31859.15 +/- 319.43	30803.94 +/- 331.19
kroE100.tsp	23650.31 +/- 200.30	23024.41 +/- 323.72
lin105.tsp	15325.79 +/- 249.18	14820.35 +/- 306.62
rat195.tsp	2671.05 +/- 34.28	2532.01 +/- 24.67
berlin52.tsp	8023.91 +/- 256.56	7944.81 +/- 226.26
kroA100.tsp	23534.02 +/- 647.66	22165.88 +/- 532.26
att48.tsp	11088.85 +/- 297.74	11008.40 +/- 214.59
pr152.tsp	76374.64 +/- 408.79	76003.82 +/- 354.97

Table 2: Tempo de execução (*ms*).

Arquivo	VND (Swap + 2-Opt)	VND (Swap + 2-Opt + 3-Opt)
kroA150.tsp	176.6 +/- 29.9	6999.8 +/- 1039.0
kroB100.tsp	86.0 +/- 19.1	2246.2 +/- 235.2
pr107.tsp	72.2 +/- 26.3	2802.1 +/- 372.1
kroC100.tsp	58.0 +/- 15.7	2096.7 +/- 228.9
rat99.tsp	42.2 +/- 13.0	1613.1 +/- 235.2
st70.tsp	30.4 +/- 11.3	629.6 +/- 142.7
kroB150.tsp	174.0 +/- 30.2	6514.1 +/- 671.9
kroB200.tsp	353.6 +/- 64.1	17053.7 +/- 2706.1
pr136.tsp	196.6 +/- 31.2	4209.5 +/- 1494.7
pr144.tsp	139.2 +/- 25.6	2112.9 +/- 1492.9
pr124.tsp	116.6 +/- 25.0	3530.1 +/- 687.6
pr76.tsp	40.7 +/- 8.9	669.8 +/- 121.2
kroD100.tsp	76.7 +/- 15.2	1857.0 +/- 574.1
kroA200.tsp	297.7 +/- 60.4	15987.9 +/- 1412.2
kroE100.tsp	67.0 +/- 14.3	1737.9 +/- 390.9
lin105.tsp	54.9 +/- 20.7	2213.0 +/- 571.1
rat195.tsp	290.4 +/- 59.6	16995.5 +/- 2815.4
berlin52.tsp	9.1 +/- 5.2	161.0 +/- 97.9
kroA100.tsp	45.9 +/- 13.4	1810.4 +/- 370.1
att48.tsp	13.4 +/- 5.1	86.9 +/- 56.3
pr152.tsp	170.3 +/- 28.4	4027.9 +/- 1715.5