Trabalho de Implementação 02

Thiago Martin Poppe

09/12/2021

1 Variable Neighborhood Descent

Para o segundo trabalho de implementação foi solicitado que construíssemos uma heurística para o problema do caixeiro viajante baseada em Variable Neighborhood Descent (ou VND). Optei por utilizar apenas 2 vizinhanças para compor o VND, sendo elas as vizinhanças geradas pelo 2-OPT e por um algoritmo que denominei *Simple Swap*, que nada mais é do que trocar dois vértices da nossa solução, gerando uma outra com potencialmente um custo menor.

Já que o código foi desenvolvido em Python, a exploração das vizinhanças de forma separada tomou um tempo considerável. Por isso, optei por implementar o framework apresentando durante a aula de VND, que consiste em basicamente fazermos uma busca local utilizando a vizinhança mais simples, no caso o 2-OPT e, uma vez que chegamos no seu ótimo local, utilizamos a outra vizinhança para amostrar uma outra solução potencialmente melhor. Ao encontrarmos uma nova solução melhor, aplicamos uma busca local sobre a mesma, repetindo essa ideia até chegar no ótimo local de ambas vizinhanças.

2 Resultados

A seguir podemos observar duas tabelas que sumarizam os resultados obtidos com o VND para as instâncias fornecidas. Na primeira, podemos perceber que o tempo de execução foi consideravelmente maior do que o que gastamos no trabalho de implementação 1 (heurística construtiva). A segunda tabela apenas traz uma comparação entre a heurística construtiva desenvolvida no trabalho anterior (NN), a solução aplicando apenas uma busca local 2-OPT e utilizando o VND implementado. Podemos perceber que em alguns casos houve melhorias ao utilizarmos o VND, já em outros não.

	instância	número de cidades	tempo de execução	custo da solução (VND)	distância utilizada
0	att48	48	0.03	10782	pseudo-euclidiana
1	berlin52	52	0.12	8043	euclidiana 2D
2	pr76	76	0.22	121128	euclidiana 2D
3	st70	70	0.23	709	euclidiana 2D
4	kroD100	100	0.29	24721	euclidiana 2D
5	kroB100	100	0.34	25393	euclidiana 2D
6	kroC100	100	0.47	22968	euclidiana 2D
7	rat99	99	0.47	1399	euclidiana 2D
8	kroE100	100	0.48	24268	euclidiana 2D
9	kroA100	100	0.52	22489	euclidiana 2D
10	lin105	105	0.55	18230	euclidiana 2D
11	pr144	144	0.56	61400	euclidiana 2D
12	pr136	136	0.58	110103	euclidiana 2D
13	pr124	124	0.80	61823	euclidiana 2D
14	pr152	152	0.85	79996	euclidiana 2D
15	pr107	107	0.91	44921	euclidiana 2D
16	kroA150	150	1.20	28612	euclidiana 2D
17	kroB200	200	2.58	33375	euclidiana 2D
18	kroB150	150	3.18	27442	euclidiana 2D
19	rat195	195	3.62	2493	euclidiana 2D
20	kroA200	200	4.06	31518	euclidiana 2D

Figure 1: Resultado do VND para cada instância

	instância	número de cidades	custo da solução (NN)	custo da solução (2-OPT)	custo da solução (VND)
0	att48	48	12861	10782	10782
1	berlin52	52	8980	8385	8043
2	pr76	76	153462	121315	121128
3	st70	70	830	770	709
4	kroD100	100	26947	24721	24721
5	kroB100	100	29158	25393	25393
6	kroC100	100	26227	23107	22968
7	rat99	99	1554	1418	1399
8	kroE100	100	27460	24459	24268
9	kroA100	100	27807	23737	22489
10	lin105	105	20356	18281	18230
11	pr144	144	61652	61400	61400
12	pr136	136	120769	110103	110103
13	pr124	124	69297	61910	61823
14	pr152	152	85699	79996	79996
15	pr107	107	46680	45340	44921
16	kroA150	150	33633	28612	28612
17	kroB200	200	36980	33375	33375
18	kroB150	150	34499	28435	27442
19	rat195	195	2752	2497	2493
20	kroA200	200	35859	31749	31518

Figure 2: Comparação da solução dos algoritmos para cada instância