DCC215 Trabalho de Implementação

Vinicius Silva Gomes – 2021421869

Data: 21/10/2023

Heurística baseada em *Variable Neighborhood Descent* para o TSP

O trabalho consiste na implementação de uma heurística baseada em *Variable Neighborhood Descent* (VND) para o Problema do Caixeiro Viajante (*Traveling Salesman Problem* ou TSP) métrico. Esse trabalho é uma continuação direta do trabalho anterior, no qual era proposto a implementação de uma heurística construtiva para o mesmo problema. Dessa forma, as instâncias utilizadas no processo de experimentação permanecem as mesmas e, portanto, as discussões e propriedades extraídas sobre elas na primeira parte do trabalho continuam válidas.

A ideia de heurísticas baseadas na estratégia de *Variable Neighborhood Descent* é, partindo de uma solução inicial para o problema, que pode ser gerada de maneira aleatória e procedural ou a partir de alguma heurística construtiva, amostrar diversas funções de vizinhança e executar uma busca local na união dessas vizinhanças. Para tanto, a heurística percorre cada uma das vizinhanças das funções escolhidas, da mais fácil para a mais difícil em termos de complexidade temporal, e procura por vizinhos com custo melhor que a solução atual. Quando um vizinho aprimorante é encontrado, a solução atual é substituída por ele e o processo é recomeçado. Assim, a solução atual apenas será retornada quando o custo dela for um ótimo local para todas as vizinhanças das funções escolhidas.

As funções de vizinhança escolhidas para essa implementação foram SWAP, 2-OPT, 3-OPT e 4-OPT. A primeira realiza uma troca (*swap*) simples de dois nós adjacentes na solução e a segunda, terceira e quarta realizam a troca de duas, três e quatro arestas na solução, respectivamente. Existem outras funções bastante conhecidas e empregadas, como o k-OPT (que é uma generalização das funções 2, 3 e 4-OPT), OR-OPT, Lin-Kernighan, etc. No entanto, essas foram escolhidas pela simplicidade e pelos bons resultados obtidos.

Em relação aos aspectos de implementação, foi usada a estratégia do vizinho melhor aprimorante para cada uma das vizinhanças e, para acelerar a computação, o algoritmo não é recomeçado logo que uma solução aprimorante é encontrada. Sendo assim, todas as vizinhanças serão exploradas até que o algoritmo recomece com a melhor solução obtida até aquele ponto e busque por aquela solução que será um ótimo local para todas as vizinhanças percorridas.

A linguagem de programação utilizada para implementar a heurística, assim como na primeira parte, foi a linguagem Rust. Os motivos para a escolha permanecem os mesmos citados anteriormente, acrescidos do fato que a implementação da heurística construtiva feita no trabalho anterior foi em Rust. Assim, essa implementação pode ser aproveitada para gerar o circuito inicial do qual o VND começa, o que melhora a qualidade da solução final e reduz o tempo empregado executando a heurística de busca local.

Executando o código

Para executar a heurística, é necessário ter a linguagem Rust e seu ecossistema instalados na máquina. Para tanto, é recomendado que a instalação seja feita através da documentação oficial da linguagem[1].

Uma vez que o ambiente esteja configurado, o comando

```
cargo run -- -p "./instances/instance-name.tsp"
```

pode ser utilizado para executar o algoritmo para uma instância disponível na pasta de instâncias, e, então, verificar qual foi o circuito obtido, o custo desse circuito e o tempo empregado para computá-lo.

Adicionalmente, um script foi desenvolvido para automatizar a execução do algoritmo para todas as instâncias disponíveis. Ao executar

```
./run.sh
```

esse script será chamado e os resultados obtidos para cada instância serão apresentados no terminal.

[1] Disponível em: https://www.rust-lang.org/tools/install

Resultados obtidos

As instâncias disponibilizadas foram testadas contra a heurística implementada, sendo possível observar na tabela 1 o custo do circuito obtido e o tempo empregado para computar esse circuito/custo. Além disso, as instâncias no formato TSP-LIB também armazenam o custo ótimo para aquela instância, o que torna possível a análise do desempenho da heurística em relação ao valor ótimo.

Instância	**Distância**	**Custo**	**Ótimo**	**Razão**	**Tempo (s)**
att48	pseudo-euclidiana	11610	10628	1.09	0.0200
berlin52	euclidiana	7984.0504	7542	1.06	0.0284
kroA100	euclidiana	23278.2780	21282	1.09	0.4417
kroA150	euclidiana	30208.2795	26524	1.14	6.1043
kroA200	euclidiana	31536.2312	29368	1.07	31.7630
kroB100	euclidiana	23235.2377	22141	1.05	1.2401
kroB150	euclidiana	30242.5010	26130	1.16	16.9743
kroB200	euclidiana	31777.8924	29437	1.08	37.2663
kroC100	euclidiana	22276.2279	20749	1.07	0.4400
kroD100	euclidiana	23637.3665	21294	1.11	0.4365
kroE100	euclidiana	22664.7063	22068	1.03	0.6713
lin105	euclidiana	14932.3107	14379	1.04	0.7861
pr107	euclidiana	48996.8899	44303	1.11	0.9287
pr124	euclidiana	67772.6358	59030	1.15	4.3709
pr136	euclidiana	103737.8911	96772	1.07	4.5575
pr144	euclidiana	63187.3024	58537	1.08	8.1027
pr152	euclidiana	76515.6718	73682	1.04	9.5094
pr76	euclidiana	115989.3762	108159	1.07	0.1310
rat195	euclidiana	2563.9394	2323	1.10	25.5318
rat99	euclidiana	1310.6696	1211	1.08	1.0113
st70	euclidiana	707.0869	675	1.05	0.2773

Tabela 1: Tempo e custo obtidos para cada instância.

Apesar do tempo gasto computando cada solução ser consideravelmente maior do que aquele obtido com a heurística construtiva, por exemplo, a qualidade das soluções obtidas para a heurística baseada em VND também é bem maior. De fato, a maioria das razões de custos obtidas $\frac{C_h}{C_o}$ são menores que 1.10. Ou seja, a solução obtida com a heurística é, na maior parte dos casos, no máximo 10% pior que a solução ótima.

Além disso, graças a versatilidade dessa heurística, é possível que existam escolhas e ordens diferentes de funções de vizinhança que resultem em custos semelhantes, senão melhores, empregando um tempo menor para computar cada solução. Essa característica faz com que abordagens baseadas em VND sejam fortes candidatas a serem usadas em aplicações no mundo real para resolver esse ou outros problemas semelhantes.