Análise de Estratégias para o Problema do Caixeiro Viajante com Pedágios

Guilherme Augusto Rocha de Figueiredo - 108197

17 de Maio, 2025

1 Introdução

Este relatório apresenta a análise de três estratégias aplicadas a uma variação do Problema do Caixeiro Viajante (TSP, *Traveling Salesman Problem*), adaptado para considerar a presença de pedágios em certas estradas. Nessa versão, além das cidades e das distâncias euclidianas entre elas (arredondadas para inteiros), os seguintes parâmetros adicionais são considerados:

1. **T**: valor inteiro que representa o número máximo de pedágios gratuitos.
2. **P**: valor real que indica o custo adicional de cada pedágio excedente.
3. **L**: valor real que define o limite de distância para que uma estrada tenha pedágio (todas as estradas com distância menor ou igual a L têm pedágio).

Assim, ao construir uma rota, se mais de T pares de cidades tiverem distância menor ou igual a L, será cobrado um valor adicional de P por cada pedágio excedente. Como consequência, a solução ótima pode não ser mais a de menor distância total.

2 Metodologia

As soluções foram obtidas a partir de diferentes estratégias: uma solução inicial trivial (ordem crescente ou ímpar-par), aplicação de busca local 2-opt sobre essa solução, e uma abordagem gulosa combinada com busca local.

Foi implementado a versão Hill Climbing do algorimto busca local, que sempre escolhe a troca com melhor ganho entre todas as opções da vizinhança a cada passo. Além disso, foram testadas duas variações do algoritmo guloso: uma que adiciona cidades apenas no final da solução parcial e outra que considera as duas pontas para escolher a melhor inserção. A primeira foi aplicada sobre as instâncias com solução trivial crescente e a segunda, sobre aquelas com solução ímpar-par.

3 Resultados

As instâncias avaliadas foram geradas com base em dados públicos do TSP disponíveis no GitHub [1]. A Tabela 1 resume os parâmetros de cada instância. A Tabela 2 apresenta os resultados obtidos por cada estratégia (os melhores resultados encontrados para cada instância estão destacados em negrito).

[1] `https://github.com/mastqe/tsplib/`

id	arquivo_origem	solucao_inicial	T	P	L
it_1	burma14	crescente	14	0	0
it_2	burma14	impar-par	14	0	0
it_3	berlin52	crescente	52	0	0
it_4	berlin52	impar-par	52	0	0
it_5	st70	crescente	70	0	0
it_6	st70	impar-par	70	0	0
it_7	gil262	crescente	262	0	0
it_8	gil262	impar-par	262	0	0
it_9	gr666	crescente	666	0	0
it_10	gr666	impar-par	666	0	0
it_11	dsj1000	crescente	1000	0	0
it_12	dsj1000	impar-par	1000	0	0
it_13	burma14	crescente	2	1	1
it_14	burma14	crescente	4	1	2
it_15	burma14	crescente	4	2	2
it_16	berlin52	crescente	10	200	400
it_17	berlin52	crescente	15	100	300
it_18	berlin52	crescente	15	200	300
it_19	st70	crescente	15	2	8
it_20	st70	crescente	15	6	8
it_21	st70	crescente	10	6	10
it_22	st70	crescente	10	10	15
it_23	gil262	crescente	30	7	12
it_24	gil262	crescente	30	5	12

Table 1: Instâncias geradas para avaliação do problema.

id	trivial	busca_local	guloso_busca_local	guloso
it_1	42	31	**30**	40
it_2	60	**30**	**30**	37
it_3	22205	8492	**7842**	8980
it_4	28043	8383	**8056**	8790
it_5	3410	**712**	722	816
it_6	3454	718	**686**	784
it_7	26298	2675	**2444**	2955
it_8	27213	2521	**2485**	2756
it_9	5554	3354	**3262**	3994
it_10	8788	3339	**3270**	4080
it_11	557633555	**20205428**	20327483	24630960
it_12	557770011	20281802	**19940298**	23570849
it_13	45	**38**	**38**	46
it_14	43	**36**	**36**	44
it_15	44	39	**37**	48
it_16	25605	15768	**15578**	16580
it_17	23005	10835	**10710**	11780
it_18	23805	13062	**13010**	14580
it_19	3410	**745**	750	854
it_20	3410	**791**	793	930
it_21	3410	**871**	877	1038
it_22	3410	1128	**1111**	1306
it_23	26298	3539	**3500**	4131
it_24	26298	3293	**3239**	3795

Table 2: Resultados das diferentes estratégias.

4 Análise

A análise a seguir aborda diferentes questões relacionadas ao desempenho das abordagens.

4.1 O algoritmo guloso encontra rotas melhores que a solução trivial?

id	tipo_guloso	melhora (%)
it_1	simples	4.76
it_2	duas-pontas	38.33
it_3	simples	59.56
it_4	duas-pontas	68.66
it_5	simples	76.07
it_6	duas-pontas	77.3
it_7	simples	88.76
it_8	duas-pontas	89.87
it_9	simples	28.09
it_10	duas-pontas	53.57
it_11	simples	95.58
it_12	duas-pontas	95.77
it_13	simples	-2.22
it_14	simples	-2.33
it_15	simples	-9.09
it_16	simples	35.25
it_17	simples	48.79
it_18	simples	38.75
it_19	simples	74.96
it_20	simples	72.73
it_21	simples	69.56
it_22	simples	61.7
it_23	simples	84.29
it_24	simples	85.57

Table 3: Porcentagem de melhora na solução que a abordagem gulosa gera em relação a solução trivial.

A Tabela 3 apresenta a melhoria percentual que a abordagem gulosa oferece em relação à solução trivial, mais especificamente a relação entre as colunas "trivial" e "guloso" da Tabela 2. Como se pode observar, o método guloso não garante uma melhoria em todos os casos. Isso ocorre porque sua implementação considera apenas a distância euclidiana entre as cidades. Assim, ao incluir os pedágios no custo total, é possível que a rota gerada pelo método guloso seja inferior à ordem trivial ou aleatória — como ilustrado nas instâncias 13, 14 e 15.

Apesar dessas exceções, o método guloso, de forma geral, apresenta resultados significativamente superiores à solução trivial. Houve uma melhora média de 55,60% nas soluções geradas pelo método guloso, com 87,50% das instâncias apresentando alguma melhoria. Isso evidencia que, embora a melhora não seja garantida em todos os casos, o método guloso representa uma alternativa mais segura do que uma inicialização aleatória ou trivial.

4.2 Existe uma versão do guloso superior à outra?

A análise anterior considerou ambas as implementações do método guloso. No entanto, ao diferenciá-las, a superioridade de uma das versões torna-se evidente. A versão que adiciona elementos apenas ao final da solução parcial gerou uma melhora média de 50,60%, enquanto a versão que permite inserções em ambas as extremidades da rota apresentou uma melhora média de 70,58%.

Embora o número de instâncias analisadas em cada versão e as respectivas soluções iniciais utilizadas sejam diferentes, essa comparação já oferece uma estimativa confiável de que a abordagem de duas pontas tende a gerar soluções iniciais superiores. Essa conclusão também é respaldada pelo funcionamento das versões: a abordagem de duas pontas é um aprimoramento direto da versão simples e, inclusive, é redutível a ela nos casos em que a menor distância está sempre associada à cidade mais à direita.

4.3 A busca local melhora significativamente a solução trivial?

id	melhora_trivial (%)	melhora_guloso (%)	diff_trivial_guloso (%)
it_1	26.19	25.0	3.23
it_2	50.0	18.92	0.0
it_3	61.76	12.67	7.65
it_4	70.11	8.35	3.9
it_5	79.12	11.52	-1.4
it_6	79.21	12.5	4.46
it_7	89.83	17.29	8.64
it_8	90.74	9.83	1.43
it_9	39.61	18.33	2.74
it_10	62.01	19.85	2.07
it_11	96.38	17.47	-0.6
it_12	96.36	15.4	1.68
it_13	15.56	17.39	0.0
it_14	16.28	18.18	0.0
it_15	11.36	22.92	5.13
it_16	38.42	6.04	1.2
it_17	52.9	9.08	1.15
it_18	45.13	10.77	0.4
it_19	78.15	12.18	-0.67
it_20	76.8	14.73	-0.25
it_21	74.46	15.51	-0.69
it_22	66.92	14.93	1.51
it_23	86.54	15.27	1.1
it_24	87.48	14.65	1.64

Table 4: Porcentagem de melhora que a aplicação do heuristica busca_local tem sobre as abordagens trivial e gulosa para geração da solução inicial.

A Tabela 4 apresenta o percentual de melhoria obtido pela aplicação da heurística `busca_local` sobre as soluções iniciais geradas pelas abordagens trivial e gulosa.

Considerando primeiramente a solução trivial, a melhora proporcionada pela busca local é bastante expressiva: em média, 62,14% (valores derivados das colunas `trivial` e `busca_local` da Tabela 2). Diferente do que ocorre na aplicação do algoritmo guloso, a busca local garante que a solução não será piorada, já que ela apenas aceita movimentos que resultam em uma redução do custo total. Ainda que esteja sujeita a ficar presa em ótimos locais — e não alcançar o ótimo global —, a busca local é uma estratégia bastante válida, pois é computacionalmente eficiente e, na maioria das vezes, consegue melhorar consideravelmente uma solução inicialmente aleatória.

4.4 A busca local melhora significativamente a solução gulosa?

Já ao aplicar a busca local sobre a solução gerada pela abordagem gulosa, a melhoria média observada é mais modesta: cerca de 14,95% (colunas `guloso` e `guloso_busca_local` da Tabela 2). Isso ocorre porque o algoritmo guloso já busca construir uma solução de qualidade razoável, o que significa que a busca local parte de um ponto mais próximo de um ótimo local. Ainda assim, vale a pena aplicar a busca local sobre o guloso, pois, mesmo que as melhorias sejam menores, a qualidade final da solução tende a ser superior — e a heurística nunca irá piorar o resultado inicial.

4.5 É mais vantajoso aplicar a busca local à solução gulosa do que à trivial?

Embora a busca local produza melhorias percentuais mais altas quando aplicada à solução trivial, ao comparar os resultados finais após a aplicação da busca local em ambas as abordagens, a diferença é pequena: em média, a solução final obtida a partir do guloso é apenas 1,85% melhor que a obtida a partir da trivial (valores extraídos das colunas `busca_local` e `guloso_busca_local` da Tabela 2).

Esse comportamento se justifica pelo fato de que a qualidade do ótimo local alcançado depende fortemente da solução inicial, mas ainda assim trata-se apenas de um ótimo local. O algoritmo guloso pode ser interpretado, nesse contexto, como uma forma de "acelerar" a convergência da busca local para uma boa solução — embora isso possa, por vezes, limitar a diversidade dos resultados.

Portanto, a escolha da estratégia depende do objetivo. Se a ideia é executar uma única iteração do algoritmo, partir do guloso pode ser a melhor opção por oferecer uma solução inicial mais promissora. No entanto, se houver recursos para executar múltiplas rodadas da busca local, pode ser mais interessante aplicá-la sobre várias soluções iniciais aleatórias. Isso introduz diversidade à busca e pode aumentar as chances de escapar de ótimos locais inferiores. Ressalta-se ainda que a implementação do algoritmo guloso utilizada neste trabalho é determinística — o que reduz significativamente a variabilidade das soluções iniciais, embora ainda haja pequenas variações devido a empates e detalhes da implementação.

5 Conclusão

Este trabalho analisou o impacto das estratégias gulosa e de busca local aplicadas sobre soluções iniciais triviais no contexto do Problema do Caixeiro Viajante com pedágios. Os resultados demonstram que ambas as abordagens são capazes de gerar melhorias substanciais no custo das soluções, destacando-se por sua eficiência e simplicidade de implementação.

Apesar disso, cada estratégia apresenta limitações. O algoritmo guloso pode, em alguns casos, piorar a solução inicial, enquanto a busca local tende a ficar presa em ótimos locais, sem garantia de alcançar o ótimo global. Essas limitações indicam a necessidade de empregar heurísticas mais sofisticadas quando se busca maior qualidade nas soluções — ainda que, por sua natureza, nenhuma heurística consiga garantir a obtenção do ótimo global para todas as instâncias (ou mesmo para qualquer instância, em alguns casos).

Mesmo dentro das estratégias avaliadas, ainda há espaço para melhorias. No caso do algoritmo guloso, uma possível evolução seria incorporar os pedágios no cálculo das distâncias durante a construção da rota. Já a busca local poderia ser aplicada a múltiplas soluções iniciais aleatórias, aumentando a diversidade e a chance de escapar de ótimos locais inferiores.

Essas observações reforçam que o estudo e a aplicação de metaheurísticas constituem um processo contínuo de refinamento, sempre em busca de melhores resultados para problemas complexos de otimização.