Implementação de Metaheurísticas de Busca Local para o Problema do Caixeiro Viajante com Pedágios

Guilherme Augusto Rocha de Figueiredo - 108197

05 de Junho, 2025

1 Introdução

Este relatório apresenta a análise de duas metaheurísticas baseadas em busca local, 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), são considerados os seguintes parâmetros adicionais:

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 para cada pedágio excedente. Como consequência, a solução ótima pode não corresponder mais à de menor distância total.

Este trabalho dá continuidade a um projeto anterior, no qual foram implementados um método guloso e uma busca local para resolver essa variação do problema. As funções desenvolvidas anteriormente, assim como a classe base TSP, foram reaproveitadas neste trabalho. Além disso, os melhores resultados obtidos anteriormente foram utilizados como baseline. O objetivo deste estudo é aplicar e avaliar metaheurísticas, buscando melhorar os resultados já obtidos.

2 Metodologia

A solução é representada por uma permutação das cidades que define a ordem do percurso (iniciando pela cidade 1). A vizinhança utilizada é a *2-swap*, as instâncias avaliadas são as mesmas do Trabalho 1, e os resultados para cada algoritmo correspondem à média aritmética de 5 execuções. Duas metaheurísticas foram implementadas: ILS e GRASP. Cada uma delas apresenta variações em relação à sua forma tradicional. No entanto, antes de detalhá-las, é importante descrever os algoritmos guloso e de busca local utilizados como base.

No caso do algoritmo guloso, foi reutilizada a implementação do Trabalho 1. A partir do vértice 1, ele adiciona iterativamente o próximo vértice que apresenta o menor custo para ser conectado ao início ou ao final da rota. Inicialmente, esse custo era calculado apenas com base na distância euclidiana entre os vértices, originando a versão chamada *GreedyDistInitialization*. Para este novo trabalho, foi criada uma versão que também leva em conta os custos com pedágios ao adicionar um novo vértice. Após testes, observou-se que essa nova abordagem não apresentou melhores resultados e, em alguns casos, até os piorou, sendo assim descartada.

Quanto à busca local, também foi reutilizada a implementação chamada *HillClimbing*, que avalia toda a vizinhança a cada iteração e escolhe a troca que proporciona a maior melhoria (*best improvement*). Uma nova versão foi desenvolvida, denominada *LocalSearch*, que adota a estratégia de *first improvement*: à medida que percorre as trocas possíveis, aplica imediatamente aquelas que resultam em melhoria. Isso permite que o

algoritmo realize múltiplas trocas em uma única iteração, acelerando o processo de convergência. O critério de parada, em ambas as versões, é a ausência de melhorias em uma iteração completa.

Para gerar a solução inicial, ambas as metaheurísticas iniciam com a execução de *GreedyDistInitialization + HillClimbing*. Toda a implementação foi realizada em C++, com modelagem orientada a objetos. Cada classe de metaheurística possui uma variável interna da classe *Solution*, que armazena a melhor solução encontrada durante a execução, incluindo seu valor, iteração correspondente e ordem das cidades.

2.1 Iterated Local Search

A primeira metaheurística implementada foi a Iterated Local Search (ILS), um algoritmo de fácil implementação que alterna entre perturbações na solução atual e buscas locais. O Algoritmo 1 apresenta o pseudocódigo da versão desenvolvida.

Algorithm 1 ILS para o Problema do Caixeiro Viajante com Pedágios

```
1:  procedure ILS-TSP
2:      t ← 0
3:      temp ← 1
4:      bestSolution ← initialSolution
5:      lastSolution ← bestSolution
6:      while true do
7:          t ← t + 1
8:          PERTURB( )
9:          newValue ← LOCALSEARCH( )
10:         ACCEPT(lastSolution, newValue, t, temp)
11:         if t − bestSolution.iteration = 25 then
12:             break
13:         end if
14:         temp ← temp × 0.95
15:     end while
16:     return bestSolution.value
17: end procedure
```

A implementação é bastante próxima da versão tradicional. Utiliza busca local para rápida convergência, e o método de perturbação adotado é o *double-bridge move*, que remove quatro arestas da solução atual e as reconecta de forma cruzada (como duas pontes). O processo de aceitação da nova solução difere ligeiramente, sendo baseado no critério probabilístico do algoritmo Simulated Annealing (SA). Se a nova solução for melhor que a atual, ela é aceita automaticamente; caso contrário, é aceita com probabilidade:

$$p = \exp\left(-\frac{\Delta}{T}\right), \quad \text{onde} \quad \Delta = \frac{\text{newValue} - \text{oldValue}}{\text{oldValue}} \tag{1}$$

$$\text{Aceitar nova solução se} \quad u < p, \quad \text{com} \quad u \sim \mathcal{U}(0,1) \tag{2}$$

Nessas equações, T representa a variável *temp*, que inicia em 1 e é multiplicada por $\alpha = 0{,}95$ a cada iteração, conforme mostrado no pseudocódigo. O algoritmo é interrompido após 25 iterações consecutivas sem encontrar uma nova solução melhor.

2.2 Greedy Randomized Adaptive Search Procedure

A segunda metaheurística implementada foi a Greedy Randomized Adaptive Search Procedure (GRASP), um algoritmo que combina uma construção gulosa randomizada com busca local. O Algoritmo 2 mostra o pseudocódigo da versão utilizada.

A implementação segue de forma geral o padrão da metaheurística. Executa 25 vezes a construção gulosa seguida da busca local *HillClimbing*, já implementada no Trabalho 1. O novo algoritmo guloso é uma modificação do *GreedyDistInitialization*, onde, em vez de sempre selecionar o vértice de menor custo, todos os

Algorithm 2 GRASP para o Problema do Caixeiro Viajante com Pedágios

```
1: procedure GRASP-TSP
2:     bestSolution ← initialSolution
3:     maxIterations ← 25
4:     for i ← 1 to maxIterations do
5:         GreedyRandomizedConstruction( )
6:         HillClimbing( )
7:         UpdateBestSolution( )
8:     end for
9:     return bestSolution.value
10: end procedure
```

vértices possíveis são ordenados pelo custo, e um deles é escolhido aleatoriamente entre os p melhores, sendo p igual a 20% (arredondado para cima) do total de opções.

O diferencial desta abordagem está na função *UpdateBestSolution*, que executa o algoritmo de *Path Relinking* entre a melhor solução atual e a nova solução recém-gerada. A direção escolhida é da solução de melhor valor em direção à outra, com o intuito de explorar melhor a vizinhança da solução promissora. A cada passo, é realizada uma troca entre duas cidades, de forma que pelo menos uma delas atinja sua posição final na solução-alvo, sempre selecionando a troca que resulta no menor custo. Ao final deste processo, a variável *bestSolution* armazena a melhor solução encontrada durante todo o percurso (soluções iniciais incluso, para os casos onde não ocorre melhora).

3 Resultados

A Tabela 1 apresenta as instâncias utilizadas para a avaliação das metaheurísticas, enquanto a Tabela 2 detalha os valores de custo final obtidos para cada uma delas. Adicionalmente, a Tabela 3 apresenta a melhoria percentual de cada metaheurística em relação ao baseline, calculada como $\frac{\text{baseline}-\text{metaheurística}}{\text{baseline}} \times 100$. Valores positivos indicam melhoria, enquanto valores negativos indicam piora.

Table 1: Instância usadas para avaliação das metaheurísticas.

ID	**Arquivo**	**Tamanho**	**T**	**P**	**L**
inst_1	burma14	14	14	0	0
inst_2	burma14	14	2	1	1
inst_3	burma14	14	4	1	2
inst_4	burma14	14	4	2	2
inst_5	berlin52	52	52	0	0
inst_6	berlin52	52	10	200	400
inst_7	berlin52	52	15	100	300
inst_8	berlin52	52	15	200	300
inst_9	st70	70	70	0	0
inst_10	st70	70	15	2	8
inst_11	st70	70	15	6	8
inst_12	st70	70	10	6	10
inst_13	st70	70	10	10	15
inst_14	gil262	262	262	0	0
inst_15	gil262	262	30	7	12
inst_16	gil262	262	30	5	12
inst_17	gr666	666	666	0	0
inst_18	dsj1000	1000	1000	0	0

Table 2: Resultados dos Custos Finais para cada Instância

ID	Baseline	GRASP	ILS
inst_1	**30**	**30**	**30**
inst_2	**38**	**38**	**38**
inst_3	**36**	**36**	**36**
inst_4	**37**	**36**	**36**
inst_5	7842	**7611**	7801
inst_6	15578	15483	**15471**
inst_7	10710	10483	**10460**
inst_8	13010	12899	**12874**
inst_9	**686**	**686**	689
inst_10	750	**719**	720
inst_11	793	**760**	765
inst_12	877	**854**	863
inst_13	1111	1098	**1097**
inst_14	**2444**	2478	2477
inst_15	3500	**3485**	3498
inst_16	**3239**	3257	3256
inst_17	3262	**3247**	**3247**
inst_18	19940298	**19939421**	19940298

Table 3: Melhoria Percentual em Relação à Linha de Base

ID	GRASP (%)	ILS (%)
inst_1	0.00	0.00
inst_2	0.00	0.00
inst_3	0.00	0.00
inst_4	2.70	2.70
inst_5	2.95	0.52
inst_6	0.61	0.69
inst_7	2.12	2.33
inst_8	0.85	1.05
inst_9	0.00	-0.44
inst_10	4.13	4.00
inst_11	4.16	3.53
inst_12	2.62	1.60
inst_13	1.17	1.26
inst_14	-1.39	-1.35
inst_15	0.43	0.06
inst_16	-0.56	-0.52
inst_17	0.46	0.46
inst_18	0.00	0.00

Analisando a Tabela 3, observa-se que tanto o GRASP quanto o ILS foram capazes de encontrar soluções melhores ou iguais ao baseline na maioria das instâncias. Ambas as metaheurísticas obtiveram melhorias em 11 das 18 instâncias (61,11%). O GRASP apresentou piora em apenas duas instâncias (11,11%), enquanto o ILS em três (16,67%).

Um ponto a ser destacado é que, em instâncias muito pequenas, não houve espaço para melhoria, já que o baseline aparentemente já encontrava soluções ótimas. Por outro lado, em instâncias muito grandes (como a *inst_ 18*), mesmo quando o GRASP encontrou um resultado ligeiramente melhor, o custo total era tão elevado que a melhoria percentual se tornou desprezível, arredondando para 0 na análise com duas casas decimais.

Table 4: Métricas estatísticas e teste de hipótese para GRASP e ILS.

Metaheurística	Média Melhoria (%)	Desvio Padrão	IC 95%	Valor-p
GRASP	1.12	1.61	[0.32, 1.92]	0.0087
ILS	0.88	1.45	[0.16, 1.60]	0.0192

Para uma visualização mais clara da distribuição das melhorias percentuais, a Figura 1 apresenta um boxplot que ilustra a dispersão dos dados para cada metaheurística.

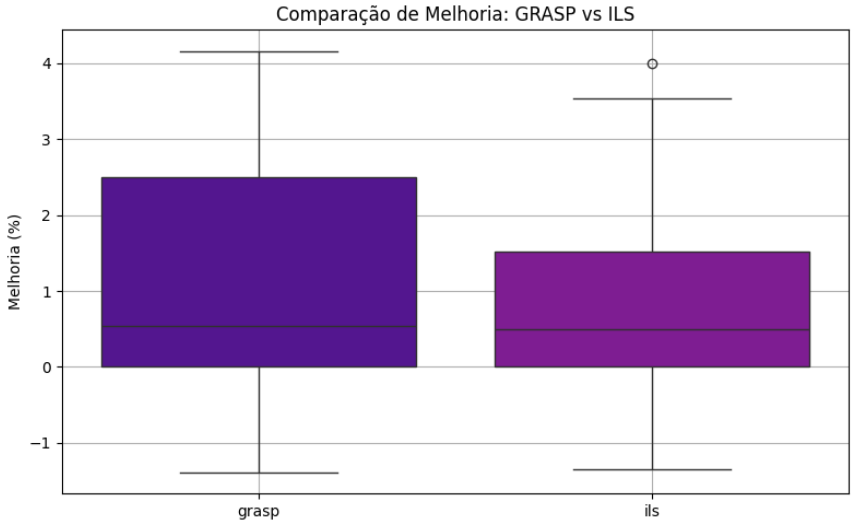

Figure 1: Boxplot da porcentagem de melhoria obtida pelas metaheurísticas GRASP e ILS em relação ao baseline.

O boxplot reforça a observação de que ambas as metaheurísticas geralmente conseguem proporcionar melhorias. A mediana do GRASP é próxima à do ILS, mas, nos casos em que há ganho, o GRASP tende a obter valores ligeiramente superiores. Dado que os valores obtidos pelas duas abordagens são, em geral, próximos, foram realizados testes estatísticos para avaliar se essa diferença é significativa.

3.1 Análise Estatística e Comparação de Desempenho

Ao analisar a Tabela 2, não é evidente qual abordagem apresenta desempenho superior de forma consistente. Por isso, realizou-se uma análise estatística para apoiar essa comparação. A Tabela 4 apresenta as métricas descritivas e os resultados dos testes de hipótese.

A primeira análise refere-se à comparação das metaheurísticas com o baseline. Os intervalos de confiança de 95% para ambas estão acima de zero, o que indica que, em média, GRASP e ILS proporcionam melhorias estatisticamente significativas em relação ao baseline. Os valores-p corroboram essa conclusão, com níveis de significância inferiores a 0,05, sugerindo forte evidência (acima de 98%) de que ambas superam o que foi implementado no Trabalho 1.

A comparação direta entre GRASP e ILS é mais sutil. Embora a média de melhoria do GRASP seja ligeiramente superior à do ILS, essa diferença não é imediatamente conclusiva. Para investigar, foi aplicado o teste não paramétrico de Wilcoxon para amostras pareadas.

Hipóteses do teste de comparação entre GRASP e ILS:

- $\boldsymbol{H_0}$: Não há diferença significativa na melhoria média entre GRASP e ILS ($\mu_{\text{GRASP}} - \mu_{\text{ILS}} = 0$);
- $\boldsymbol{H_1}$: O GRASP é significativamente melhor que o ILS em termos de melhoria média ($\mu_{\text{GRASP}} - \mu_{\text{ILS}} > 0$).

O teste foi realizado em Python, utilizando a biblioteca `scipy.stats`. O resultado obtido foi um valor-p de *0,1047*. Portanto, não se pode rejeitar a hipótese nula com um nível de significância de 5%, o que indica que não há evidência estatística suficiente para afirmar que o GRASP é significativamente melhor que o ILS. Assim, o resultado do teste é considerado inconclusivo.

4 Conclusão

Neste trabalho, foram desenvolvidas e avaliadas duas metaheurísticas baseadas em busca local (GRASP e ILS) aplicadas ao Problema do Caixeiro Viajante com Pedágios. Os resultados demonstraram que ambas as abordagens superaram consistentemente a solução do Trabalho 1, evidenciando a eficácia das técnicas de intensificação e diversificação para escapar de ótimos locais e explorar melhor o espaço de busca.

Embora o GRASP tenha apresentado desempenho ligeiramente superior ao ILS em termos de melhoria percentual média, a análise estatística indicou que essa diferença não é significativa, tornando a comparação entre os métodos inconclusiva do ponto de vista estatístico. Observou-se também que, em instâncias pequenas, as metaheurísticas não trouxeram ganhos relevantes, o que sugere que seu potencial é mais bem aproveitado em problemas de maior complexidade.

Por fim, os resultados indicam que ainda há espaço para avanços. Abordagens híbridas, que combinem os pontos fortes do GRASP e do ILS, ou a experimentação com metaheurísticas populacionais, como Algoritmos Genéticos, podem ser caminhos promissores para alcançar soluções ainda mais eficazes.