Trabalho de Implementação 3 - Heurísticas e Metaheurísticas

Francisco Teixeira Rocha Aragão - 2021031726

Data: Dezembro de 2024

1 Introdução

O presente trabalho busca resolver de maneira aproximada o problema do caixeiro viajante (TSP), fazendo uso da metaheurística GRASP como implementação. Como o problema pertence a classe NP, faz-se necessário o uso de tais estratégias, sendo utilizado no trabalho uma heurística construtiva randomizada adaptativa para definição do caminho inicial, além de diferentes funções de vizinhança para implementar a busca local durante as iterações. Abaixo encontra-se mais informações sobre a implementação além dos resultados obtidos.

2 Heurísticas utilizadas

Primeiramente sobre a heurística utilizada, a estratégia implementada para definição inicial de uma solução refere-se a uma heurística construtiva, ou seja, uma heurística em que a solução é construída do zero, desde o início até a resolução do problema. Iniciando-se assim de uma solução vazia, obtendo então uma solução parcial a cada iteração em que ao final é transformada em uma solução completa válida.

Desse modo, a estratégia utilizada foi baseada em uma abordagem gulosa, em que a cada ponto (ou cidade), o próximo trajeto escolhido é aquele com a menor distância. O início é feito a partir de uma cidade (será explicado mais frente) e a cada iteração novas cidades são adicionadas no caminho até todas as cidades serem visitadas, voltando assim ao vértice inicial resolvendo o problema. Com isso, garante-se a validade da solução retornada, em que a cada iteração acrescenta-se uma nova cidade não visitada anteriormente, terminando o algoritmo até visitar a última cidade, retornando ao ponto inicial. Sobre o ponto inicial escolhido, pode tanto iniciar da primeira cidade quanto da cidade central, sendo escolhida a segunda por apresentar melhores resultados na prática. Vale destacar que a estratégia implementada possui um alto custo, embora seja simples, sendo necessário encontrar todas as distâncias a partir do nó atual para os vizinhos, ordenar essas distâncias para então o próximo candidato ser escolhido da lista restrita. Como o grafo é completo, todos os vizinhos estão ligados em todos, com o custo sendo O(n) pois todos os nós são percorridos, O(n) para olhar todos os vizinhos de cada nó e O(nlogn) para ordenar os vizinhos, totalizando $O(n^2)$ para a construção do caminho inicial.

A implementação dessa estratégia construtiva foi um pouco alterada no GRASP, tendo em vista o caráter randomizado da metaheurística. Assim, a cada iteração é feita uma escolha aleatória entre as cidades disponíveis, de modo a evitar soluções presas em ótimos locais. Além disso, a escolha aleatória é feita a partir de uma lista de candidatos, que possui tamanho adaptativo de acordo com o nó atual, podendo ter um tamanho pequeno caso os possíveis próximos nós forem muito diferentes, ou um tamanho maior caso sejam parecidos. Isso é feito utilizando a fórmula L = c_min - alpha * (c_max - c_min) em que c_min e c_max são as menores e maiores distâncias para o próximo vizinho, alpha é um fator de ajuste e L é o tamanho da lista de candidatos. Com isso, a cada iteração é feita uma escolha aleatória entre os candidatos, de modo a diversificar as soluções encontradas.

Após isso, foi implementada a busca local utilizando diferentes vizinhanças, semelhante a ideia de VND, cujo obtivo é, após encontrar uma primeira solução, utilizar diferentes funções de vizinhanças para melhorar a solução obtida, com as diferentes vizinhanças ajudando a diversificar as soluções. Para isso foram utilizado 2 funções de vizinhança diferentes, sendo elas a troca de 2 cidades e a troca de 3 cidades. A troca de 2 cidades (2-opt, com complexidade $O(n^2)$) funciona trocando apenas dois caminhos da solução inicial, enquanto a troca de 3 cidades (3-opt, com complexidade $O(n^3)$) troca 3 caminhos. Para a estratégia da busca local,

é utilizada a abordagem melhor aprimorante, em que ao encontrar um vizinho melhor, automaticamente a busca nessa vizinhança é retomada, buscando encotrar o ótimo local em cada vizinhança antes de passar para a próxima.

Com isso, todo o processo ocorre buscando a primeira solução através da estratégia gulosa randomizada, e após isso é feita a busca local utilizando as diferentes vizinhanças. Ao final, uma solução é encontrada e armazeanda, com uma nova iteração sendo feita, repetindo o processo até o número de limites de iterações, garantindo assim a diversificação das soluções encontradas com o GRASP.

3 Execução e Resultados

O código foi desenvolvido em C++ e os testes foram realizados em uma máquina com debian 12, 16GB de ram e processador I5-11 geração. Sua execução pode ser realizada com os seguintes comandos:

```
// compilação
make

// limpar arquivos gerados
make clean

// rodar programa
make run ARGS="<pasta com instâncias de entrada> <tipo da cidade inicial> <alpha> <iteracoes>"
// <tipo de cidade inicial> = 0 para usar a primeira cidade e 1 para usar cidade central
// idealmente <alpha> deve ser entre 0 e 1
```

Vale destacar que os arquivos de entrada foram encontrados no site TSPLIB95, presente nas referências no trabalho, com o projeto executando apenas as instâncias que terminam com a extensão '.tsp'. Além disso, a execução foi realizada diferentes vezes para cada instância, com as médias dos resultados disponível nas tabelas abaixo. Essa abordagem é importante visto o caráter randomizado do GRASP, em que a cada iteração é feita uma escolha aleatória, podendo assim variar o resultado final.

Desse modo, observando as imagens 1 e 2 presentes abaixo, é possível visualizar os resultados do GRASP para as instâncias de teste utilzadas, utilizando alpha como 0 e 5 iterações. Percebe-se como o método é eficaz para encontrar soluções aproximadas para o problema, com um erro percentual de aproximadamente 10% para as instâncias menores, e 20% para as instâncias maiores, embora o erro aumente em alguns casos específicos, podendo estar relacionado a escolha aleatória de cidades feita pelo GRASP além da disposição da entrada. Vale destacar no entanto que o tempo médio de execução é baixo, embora aumente bem rápido para maiores instâncias. Vale destacar que como alpha é 0, os resultados dependem apenas da abordagem gulosa, não possuindo aleatoriedades na escolha das próximas cidades.Percebe-se também obseravando a imagem 3 como a diferença entre a solução inciial (heurística construtiva) e a solução final (GRASP) é pequena, o que é justificado pelo alpha ser igual a 0, ou seja, a escolha das cidades é feita de forma gulosa, sem aleatoriedade, levando a resultados melhores inicialmente, tendo menor margem de melhoria com a busca local.

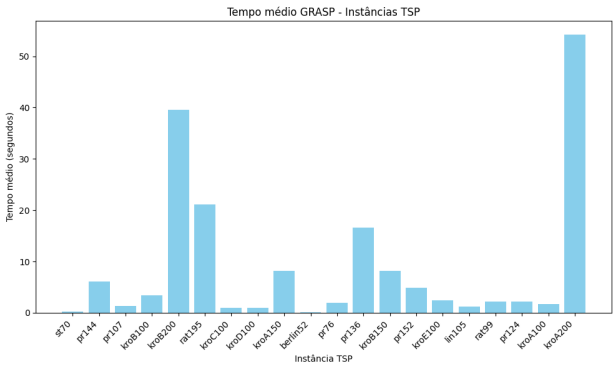

Figure 1: Tempo médio alpha 0 iterações 5

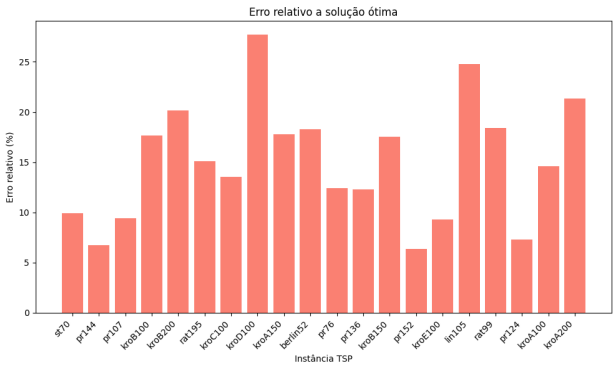

Figure 2: Erro percentual alpha 0 iterações 5

Adicionar uma figura somente

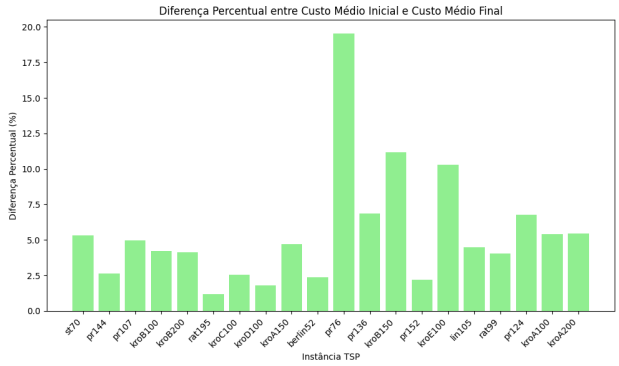

Figure 3: Diferença percentual solução inicial e final alpha 0 iterações 5

Agora variando o parâmetro alpha, temos os resultados presentes nas imagens 4 e 5 mostrando a variação obtida com alpha = 0.2 e 5 iterações. Percebe-se a diferença no tempo de execução visto na imagem 4, obtendo um tempo maior em comparação ao alpha = 0, visto que como existe um fator aleatório na escolha da próxima cidade, o método gasta mais tempo até chegar um ótimo local. Além disso, a influência do fator aleatório é vista também nas imagens 5 e 6, com os resultados sendo mais distantes em relação ao ótimo, além da maior difença entre a solução inicial e final, visto que a escolha aleatória pode levar a soluções piores inicialmente, com a busca local levando a mais melhorias.

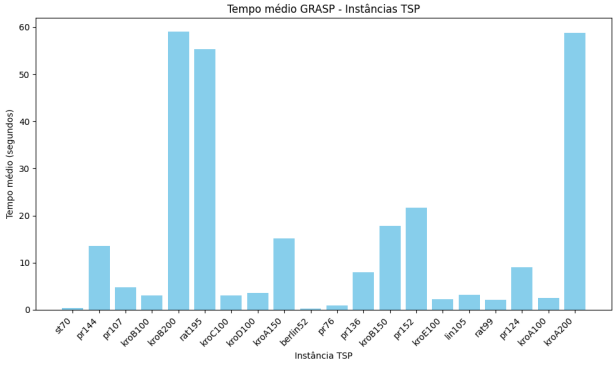

Figure 4: Tempo médio alpha 0.2 iterações 5

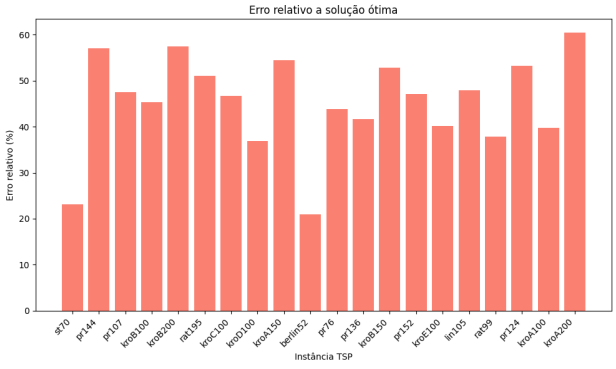

Figure 5: Erro percentual alpha 0.2 iterações 5

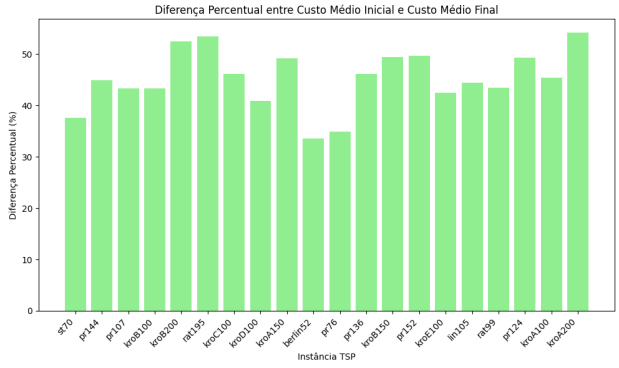

Figure 6: Diferença percentual solução inicial e final alpha 0.2 iterações 5

O mesmo fato é perceptível nas imagens 5,6 e 7 presentes abaixo, com os resultados para alpha = 0.7 e novamente com 5 iterações. Desse modo, a tendência continua, em que ao se aumentar o fator aleatório do método, piores resultados são encontrados inicialmente, com a maior distância entre o a solução inicial e a final, além de piores resultados no erro relativo a solução ótima, visto que é mais complicado encontrar o ótimo local com a escolha aleatória das cidades.

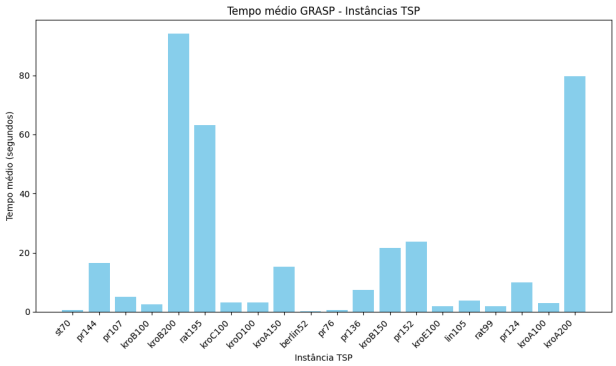

Figure 7: Tempo médio alpha 0.7 iterações 5

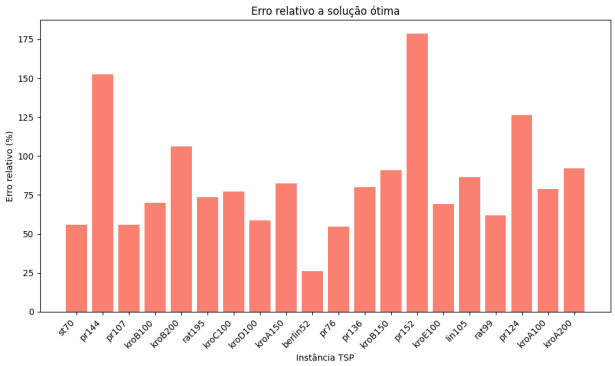

Figure 8: Erro percentual alpha 0.7 iterações 5

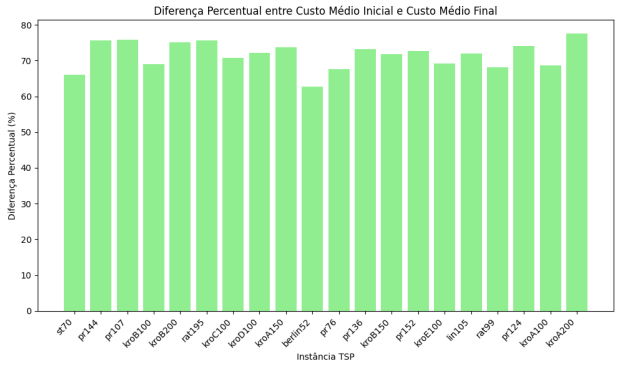

Figure 9: Diferença percentual solução inicial e final alpha 0.7 iterações 5

Visto que mais aleatoriedade foi sendo introduzida a cada experimento, manter o número de iterações como 5 é um fator que piora os resultados para maiores valores de alpha. Desse modo, o ideal é aumentar o número de iterações do método a medida que alpha aumenta, para assim conseguir amostrar mais soluções. Isso é válido já que, como os resultados estão mais aleatorios, a tendência é que piores resultados sejam encontrados. Assim, são necessárias mais amostras coletadas para então boas soluções serem escolhidas ao fazer a média das execuções. Tal fato é observado nos gráficos abaixo 8, 9 e 10 em que o algoritmo foi executado com os parâmetros alpha = 0.7 e 10 iterações. Percebe-se observando os gráficos como os erros foram menores em relação a solução ótima já que melhores resultados foram amostrados. Já para o erro em relação aos resultados iniciais, os valores agora se mantêm semelhantes, visto que embora a solução final seja melhor na média, a escolha inicial também é melhor na média.

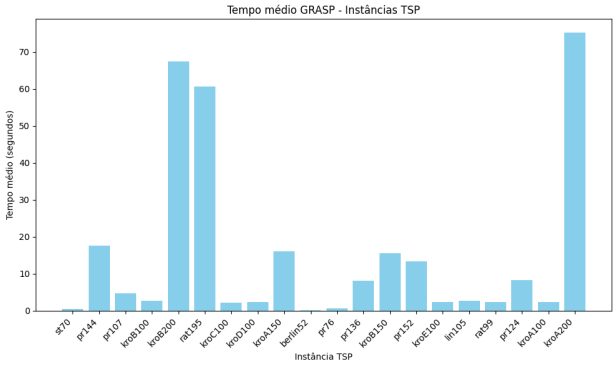

Figure 10: Tempo médio alpha 0 iterações 10

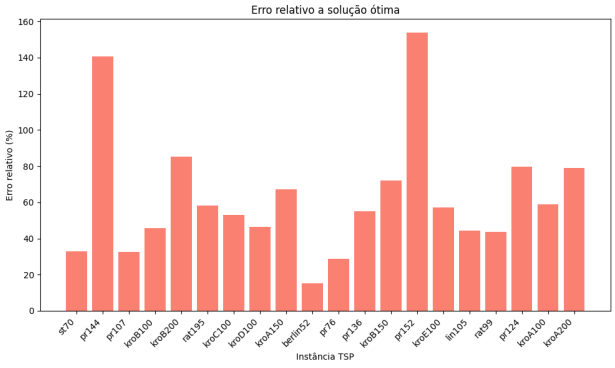

Figure 11: Erro percentual alpha 0.7 iterações 10

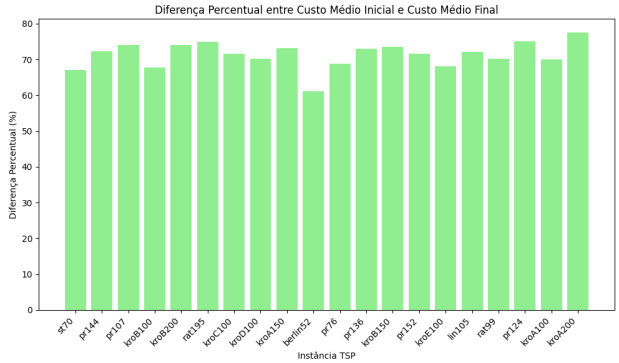

Figure 12: Diferença percentual solução inicial e final alpha 0.7 iterações 10

4 Conclusão

Observando os resultados presentes na tabela acima, percebe-se como o método GRASP é útil e importante em problemas com um grande espaço de busca, conseguindo amostrar diferentes soluções, chegando em resultados diversos e também, na média, com pouco erro. Vale destacar a necessidade de aumentar o número de iterações, além de métodos de busca local mais eficientes a medida que o parâmetro alpha aumenta, visto que a escolha aleatória das cidades pode levar a soluções piores inicialmente, com a busca local sendo essencial para melhorar a solução, sendo mais custoso e demorado realizar tais ações a medida a busca inicial fica menos gulosa. Com isso, para futuros trabalhos tais fatores podem ser explorados para melhorar os resultados obtidos.

5 Referências

GRASP – A speedy introduction
Site com instâncias e ótimos de TSP