Soluções para o Problema de Visitação Máxima e Instalação de Guichês no Metrô de Paris

Lucas Abreu Lopes[1]

[1]Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
ICEI – Ciência da Computação – Projeto e Análise de Algoritmos

lalopes@sga.pucminas.br

Abstract. *Este trabalho aborda dois problemas combinatórios relacionados à proposta de um passe turístico no metrô de Paris. O primeiro problema objetiva maximizar o número de estações distintas visitadas, retornando à estação inicial sem repetições. O segundo busca minimizar o número de guichês instalados, garantindo que todas as estações estejam a, no máximo, uma estação de um guichê. Foram propostas e implementadas soluções exatas e heurísticas para ambos, com comparação empírica entre elas. Além disso, foram exploradas outras meta-heurísticas (GRASP, Simulated Annealing, Tabu Search e Ant Colony Optimization) como demonstração de possíveis abordagens, embora algumas destas não tenham sido testadas exaustivamente nem estejam totalmente afinadas.*

Resumo. *Este trabalho endereça dois problemas no metrô de Paris: (1) maximizar o número de estações distintas visitadas em um ciclo simples; (2) encontrar o menor conjunto de estações (guichês) que domina o grafo em distância 1. Foram implementados métodos exatos e heurísticos, avaliando-os empiricamente. Foram apresentados também esboços de outras meta-heurísticas (SA, Tabu, ACO) como perspectivas futuras.*

1. Introdução

Com o objetivo de promover o turismo, a companhia do metrô de Paris pretende criar um passe especial que permita ao turista visitar pontos turísticos próximos das estações. A principal regra desse passe é que, ao retornar a uma estação previamente visitada, o passe perde a validade. Este trabalho resolve dois problemas:

1. Determinar o maior número de estações possível de serem visitadas com um único passe.
2. Posicionar guichês de venda do passe de forma otimizada, para que todas as estações estejam a, no máximo, uma estação de distância de um guichê.

2. Solução Proposta

2.1. Problema do Ciclo Simples Mais Longo

As abordagens utilizadas incluem:

- **Força Bruta:** Varre todos os caminhos simples de profundidade ilimitada, impraticável para o tamanho real do grafo.

- **Branch-and-Bound:** Usa poda com estimativas de limite superior por busca em largura para cortar ramos inviáveis.
- **Algoritmo Genético:** Evolui populações de caminhos por operadores de crossover e mutação, permitindo boa qualidade em tempo moderado.
- **Greedy DFS:** Exploração gulosa seguindo sempre o vizinho de maior grau.
- **GRASP (Greedy Randomized Adaptive Search Procedures):** Reconstrói caminhos por seleção restrita de candidatos e reinicia aleatoriamente, coletando o melhor ciclo em múltiplas iterações.
- **Simulated Annealing (SA):** Otimização por resfriamento gradual; como está implementado, supõe grafo completo e serve apenas como protótipo.
- **Tabu Search:** Meta-heurística com lista tabu para evitar ciclos de busca; implementação experimental.
- **Ant Colony Optimization (ACO):** Meta-heurística inspirada no depósito de feromônios e probabilidades associadas; implementada de forma preliminar e não otimizada.

2.2. Problema do Conjunto Mínimo Dominante-1

Para cobertura de todos os vértices a uma distância de no máximo 1 (neste caso foram implementados poucos algoritmos, pois o Guloso apresentou desempenho satisfatório):

- **Força Bruta:** Testa todas as combinações de estações de forma crescente e interrompe a execução quando encontra a primeira solução; inviável para grafos grandes.
- **Branch-and-Bound:** Poda usando limites inferiores baseados no grau máximo.
- **Guloso:** Seleciona iterativamente a estação que domina o maior número de vértices não dominados.

3. Implementação

Todos os algoritmos foram codificados em Python 3 usando a biblioteca NetworkX para modelagem de grafos. Desenvolvemos uma interface Tkinter para visualização do grafo. Algumas observações:

- Uso de `set` e `deque` para eficiência nas operações de vizinhança e poda.
- As meta-heurísticas adicionais (SA, Tabu, ACO) compartilham funções auxiliares de caminhada aleatória e operadores de vizinhança, mas ainda carecem de ajustes finos.
- A implementação de SA supõe grafo completo e não foi adaptada para grafos esparsos; Tabu e ACO possuem parâmetros não calibrados e foram incluídos como prova de conceito.

3.1. Problemas com a Implementação

Dentre as abordagens, algumas apresentaram problemas na implementação:

- **Greedy DFS:** Não foi encontrada heurística melhor do que escolher o vizinho de maior grau, abordagem que se mostrou ineficiente e não garante ciclos (e nem os incentiva).
- **Simulated Annealing (SA), Tabu Search e ACO:** Não foi possível realizar testes e ajustes aprofundados dentro do prazo do projeto.

4. Relatório de Testes

Os testes foram realizados com dois subconjuntos, um principal, selecionado pelo professor, com 10 linhas e 51 estações, e outro secundário, com 3 linhas e 23 estações do metrô de Paris. Os principais resultados são:

4.1. Subconjunto Principal

4.1.1. Maior Ciclo Simples

Força Bruta:

- Tamanho do Caminho: Não foi possível completar a execução.
- Tempo: Não foi possível completar a execução.

Branch-and-Bound:

- Tamanho do Caminho: 36
- Tempo: 1481.603 s

Genético:

- Tamanho do Caminho: 26
- Tempo: 3.12 s

Guloso:

- Tamanho do Caminho: 14
- Tempo: 0.01 s

GRASP (10000 iterações):

- Tamanho do Caminho: 32
- Tempo: 9.14 s

4.1.2. Conjunto Mínimo Dominante-1

Força Bruta:

- Tamanho: Não foi possível completar a execução.
- Tempo: Não foi possível completar a execução.

Branch-and-Bound:

- Tamanho: 16
- Tempo: 379.34 s

Guloso:

- Tamanho: 17
- Tempo: 0.00 s

4.2. Subconjunto Reduzido

4.2.1. Maior Ciclo Simples

Força Bruta:

- Tamanho do Caminho: 12
- Tempo: 0.00 s

Branch-and-Bound:

- Tamanho do Caminho: 12
- Tempo: 0.00 s

Genético:

- Tamanho do Caminho: 12
- Tempo: 1.17 s

Guloso:

- Tamanho do Caminho: Não encontrou ciclo
- Tempo: 0.00 s

GRASP (10000 iterações):

- Tamanho do Caminho: 12
- Tempo: 2.19 s

4.2.2. Conjunto Mínimo Dominante-1

Força Bruta:

- Tamanho: 8
- Tempo: 1.27 s

Branch-and-Bound:

- Tamanho: 8
- Tempo: 0.00 s

Guloso:

- Tamanho: 8
- Tempo: 0.00 s

5. Conclusão

Este trabalho abordou dois problemas combinatórios no contexto do metrô de Paris: maximizar o número de estações visitadas em um ciclo simples e minimizar o número de guichês necessários para garantir cobertura por distância-1. Os resultados obtidos demonstram que, para o primeiro problema, abordagens exatas como Branch-and-Bound fornecem soluções de alta qualidade, porém com custo computacional elevado, sendo inviáveis em grafos maiores. Entre as heurísticas, o método GRASP destacou-se ao alcançar soluções próximas das ótimas (ciclo de 32 estações) com tempo de execução significativamente reduzido, configurando-se como a abordagem mais eficiente dentre as

testadas. O algoritmo genético também apresentou bom desempenho, embora inferior ao GRASP, enquanto a busca gulosa não foi satisfatória.

Para o segundo problema, o algoritmo guloso produziu soluções muito próximas das obtidas por métodos exatos, mas com tempo de execução praticamente nulo, revelando-se a melhor alternativa para aplicações práticas. A solução ótima foi obtida com 16 guichês, enquanto o guloso alcançou 17, evidenciando a eficiência da heurística diante da complexidade do problema.

As meta-heurísticas Simulated Annealing, Tabu Search e Ant Colony Optimization foram implementadas em caráter experimental e ainda exigem ajustes e testes mais amplos.

Conclui-se que abordagens heurísticas bem projetadas são capazes de oferecer soluções de qualidade com desempenho computacional compatível com as exigências de aplicações reais. O estudo também abre espaço para trabalhos futuros, especialmente no aprimoramento das meta-heurísticas.

6. Bibliografia

Referências

[1] Sedgewick, R.; Wayne, K. *Algorithms*. 4th ed. Addison-Wesley, 2011.

[2] NetworkX. *Documentation*. Disponível em: `https://networkx.org`

[3] Feo, T. A.; Resende, M. G. "Greedy Randomized Adaptive Search Procedures." *Journal of Global Optimization*, 1995.

[4] Glover, F.; Laguna, M. *Tabu Search*. Kluwer Academic Publishers, 1997.

[5] Dorigo, M.; Stützle, T. *Ant Colony Optimization*. MIT Press, 2004.