TSP - Modelo Linear Iterativo

José Tobias

20 de Julho de 2019

1 Introdução

Neste trabalho devemos buscar entender o funcionamento de um *solver*, no caso o IBM CPLEX, para resolver um problema NP-difícil como o TSP-*Travelling Salesman Problem* (Problema do Caixeiro Viajante). O problema nos diz que, tendo-se n localidades a serem visitadas por um caixeiro viajante, o mesmo deve fazer uma rota partindo de ponto inicial qualquer p e completar a visita em todos as $n-1$ outas localidades e voltando ao ponto p no *menor caminho possível*.

Este problema, o TSP, já foi estudado utilizando-se heurísticas apenas, como algoritmos de força bruta, genéticos e caminho mínimo em grafos. Para resolvê-lo utilizando-se modelos de programação linear em um *solver*, no caso o CPLEX, precisaríamos de um número exponencial de restrições. Assim sendo, devemos utilizar um modelo linear que evolui com as soluções, tendo a certeza de que quando apenas um ciclo Hamiltoniano encontrado, este é de fato o menor caminho possível.

2 Indução

As restrições introduzidas no modelo fazem com que ele se torne pesado, tanto no sentido de computações quando no sentido de ocupação de memória que o mesmo precisa. Então, começamos com duas restrições básicas que vem da observação do problema: todas as arestas q_{ij}, ou seja a aresta q que liga localidades p_i e p_j, devem ser um ponto de chegada e saída do viajante. Essas restrições são formalizadas como:

Toda localidade é ponto inicial de uma aresta

$$\forall i \in [1, 2, ..., n]; \sum_{j=1} q_{ij} = 1; \tag{1}$$

Toda localidade é ponto final de uma aresta

$$\forall j \in [1, 2, ..., n]; \sum_{i=1} q_{ij} = 1; \tag{2}$$

Dado que cada aresta possui uma distância euclidiana d_{ij}, ou seja a distância de p_i a p_j, sendo o tamanho de uma aresta q_{ij}, podemos dizer que a função com objetivo é dada por:

Minimize

$$z(i,j) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} q_{ij} \tag{3}$$

Podemos identificar nesse modelo a falta de restrições para garantir um ciclo Hamiltoniano único. Seria como se nosso caixeiro tivesse superpoderes, podendo se teletransportar de um local ao outro enquanto faz as visitar. Ele poderia então visitar a localidade p_i e depois a p_j e voltar a localidade p_i, o que respeita todas as restrições, e então reaparecer numa localidade p_k novamente e não terminando sua rota no ponto de partida. Falando em termos de grafos, todos os vértices teriam uma aresta incidente e uma aresta acidente, mas o caminho não forma um círculo Hamiltoniano **único e mínimo**.

3 Desenvolvimento

Para resolver o problema dos diversos ciclos Hamiltoniano no caminho, suponho um modelo iterativo para resolução, adicionando restrições a cada iteração, de maneira a se alcançar o resultado ótimo com um modelo mais leve quando constatado apenas um ciclo Hamiltoniano para todo o caminho.

O relatório infelizmente é limitado para expressar todas as ideias que eu tive, juntando heurística e expressões lineares, para que se pudesse alcançar o caminho de fato ótimo. Assim sendo, vou expressar a ideia principal do algoritmo implementado: **a retirada de ciclos Hamiltonianos mínimos múltiplos com aceitação**. Fazendo-se diversas iterações do *solver* adicionando-se restrições em cada uma delas as quais devem retirar os menores ciclos, podemos chegar a resultados ótimos para instâncias de diversos tamanhos.

Por exemplo, utilizando-se do modelo linear descrito acima, na primeira iteração temos o resultado na Figura 3 a esquerda, e na iteração de número 29 temos o resultado na Figura 3 a direita, para uma instância de 50 pontos em 0.18 segundos para o melhor no meu processador, enfim.

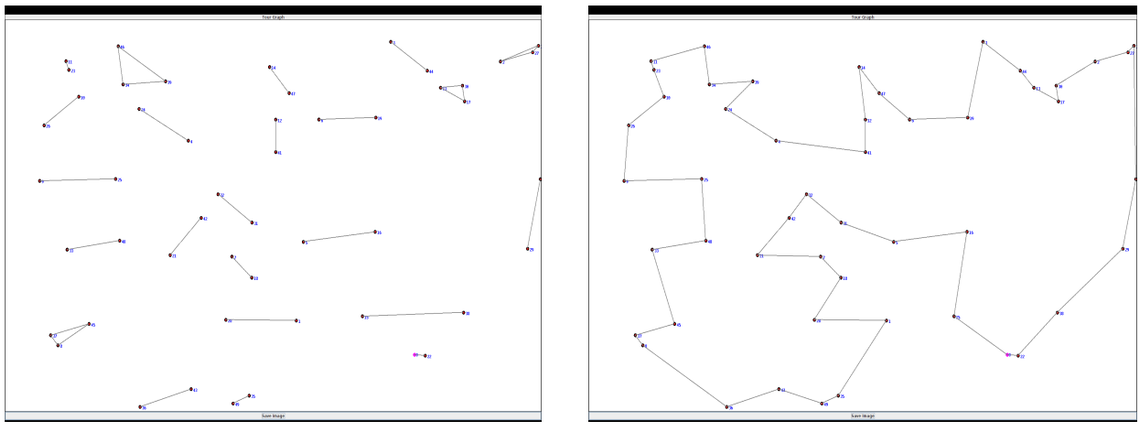

Figura 1: Na figura a esquerda vemos o resultado da primeira iteração, enquanto na figura a direita o resultado da última iteração. Nota-se na primeira figura a presença de múltiplos ciclos Hamiltonianos, e no último a presença de apenas um ciclo comprovadamente mínimo.

Estes resultados porém não são ainda consistentes para todas as instâncias, uma vez que arestas que se cruzam podem acontecer bem como inconsistências pontuais dado o ponto de parada do algoritmo, como mostrado na Figura 2. Este problema pode ser arrumado inserindo-se outras restrições que impeçam o cruzamento, e assim como no caso acima, retirando-se restrições que impedem soluções de acontecer.

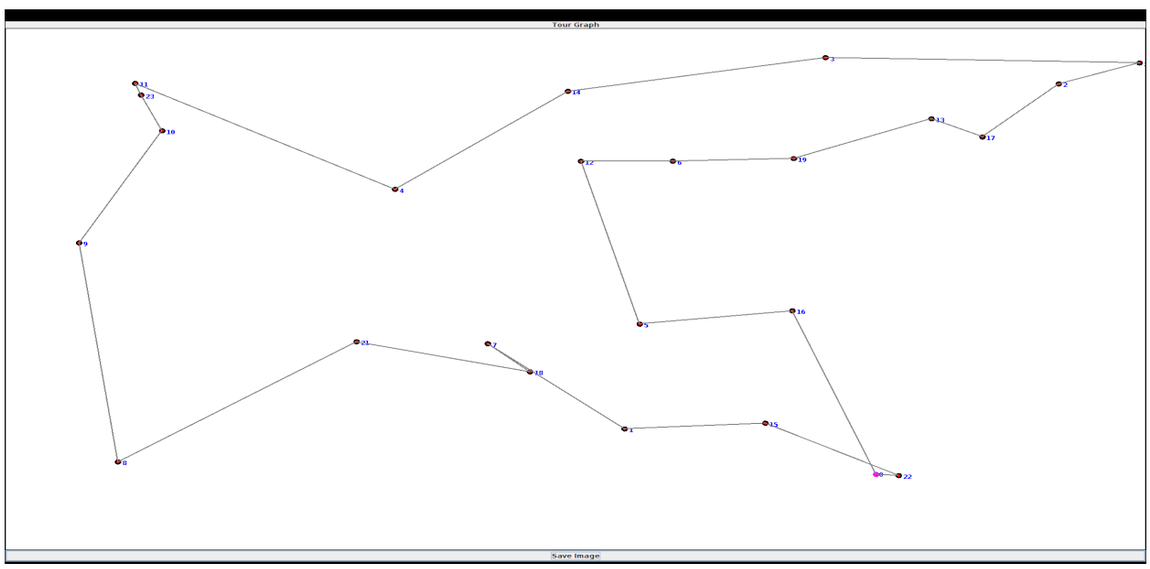

Figura 2: Pode-se perceber, no centro em baixo da imagem uma inconsistência, bem como no canto direito em baixo arestas cruzantes.

3.1 Recomendações

Caso o modelo puder ser salvo em seu estado atual e mais restrições adicionadas para uma próxima iteração, acredito que salvando-se a posição de locais sem cruzamento e deixando um algoritmo como **2-opt** resolver estas arestas possa vir a ser uma solução viável. Ou, por exemplo, descartar restrições que envolvam os quatro vértices que fazem o cruzamento das duas arestas e acrescentar restrições para que os mesmos não se cruzem novamente caso um deles já esteja conectado. Isso pode ser feito utilizando uma restrição linear simples, por exemplo: seja p_i e p_k vértices formadores de aresta q_1 em cruzamento com outra aresta q_2 formada por p_j e p_l, então podemos dizer que:

$$\begin{aligned} \sum q_1 + q_2 \leq 1 \text{ ; ou seja} \\ \sum q_{ik} + q_{jl} \leq 1 \end{aligned} \tag{4}$$

Para inconsistências locais, como a apresentada, dado um número de vértices pequenos que permita uma busca exaustiva, a mesma é sugerida. Caso não seja possível pelo número grande de vértices, outras técnicas podem ser usadas dentro de modelos lineares como já explicado anteriormente.

Também para inconsistências e cruzamentos, utilizando-se a otimização local após destacados os vértices que fazem parte da imperfeição, e criando-se um ciclo Hamiltoniano mínimo destacado com uma aresta artificial de fechamento, seria uma forma de tratamento.

3.2 Outros resultados ótimos

Outros resultados notáveis observados foram os abaixo na Figura 3.2, os mesmos trazem o número de pontos e o tamanho do caminho final como legenda.

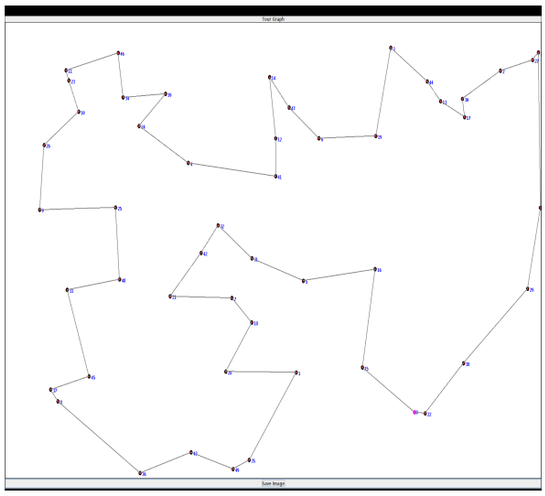

Figura 3: pontos: 50, distância: 563.090

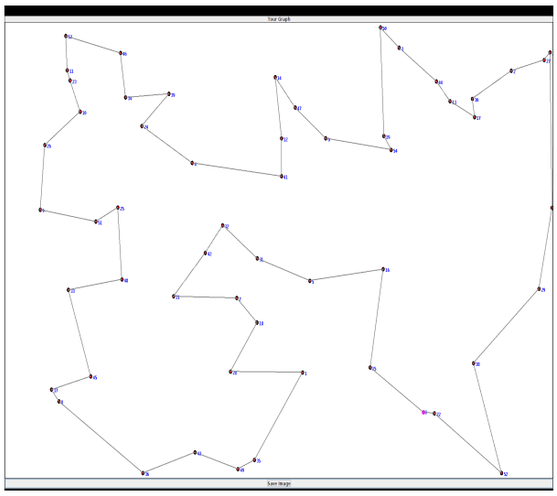

Figura 4: pontos: 55, distância: 616.389

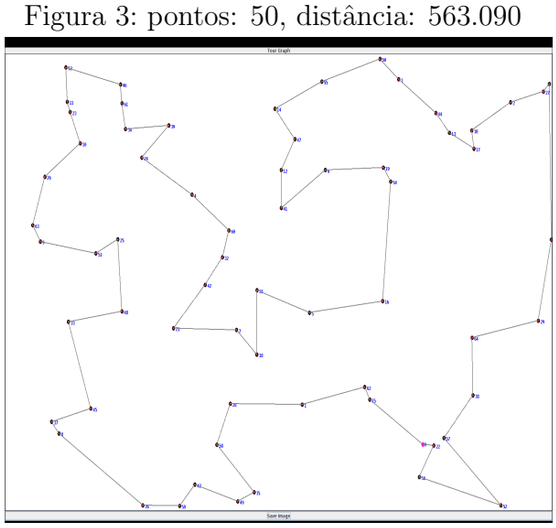

Figura 5: pontos: 65, distância: 639.053

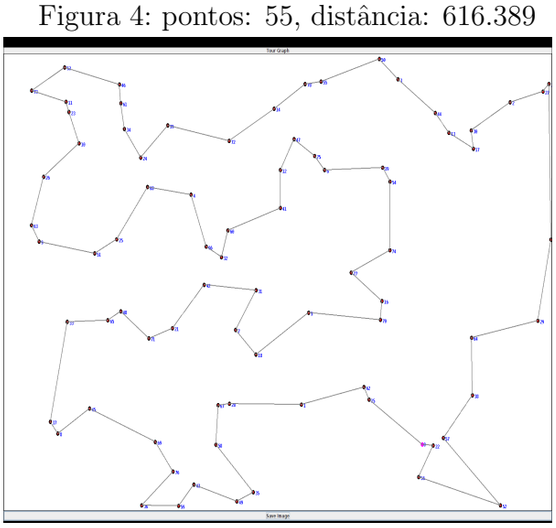

Figura 6: pontos: 80, distância: 691.037

Tempo de execução e utilização de memória podem ser estudos para outro relatório.

4 Conclusão

Acredito que não estou fazendo o algoritmo utilizando de maneiras otimizadas para guardar restrições e continuar o modelo de um ponto de parada para adicionar-se outras restrições que retirem os ciclos múltiplos. Além do mais, as arestas cruzantes e pontos de inconsistência.

Dado o pouco tempo para estudo, acredito que eu tenha chegado num local de entendimento da fronteira entre heurísticas e técnicas exatas de programação linear. Pouca prática de heurística foi aplicada nesse meu trabalho em específico, mas ficam evidentes os pontos onde a mesma levaria a resultados melhores que os apresentados.