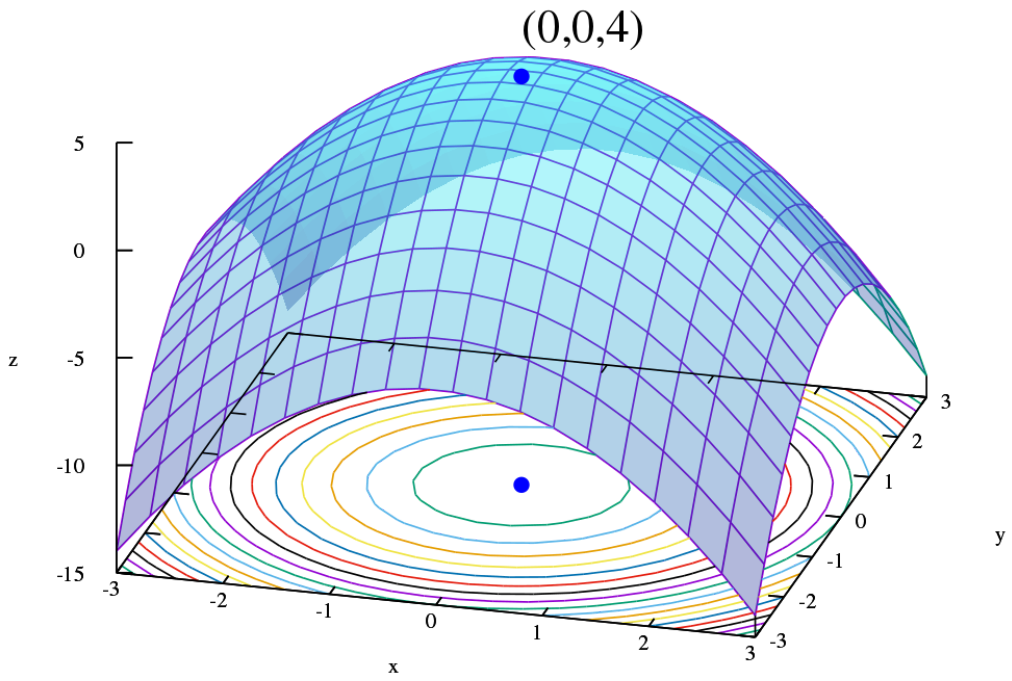
UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

SME0110 - Programação Matemática

Problema do Caixeiro Viajante



Alunos:

Bruno Del Monde nº 10262818

Flavio Vinicius V. Santana nº 9866552

Vinicius Henrique Borges nº 9771546

1. Introdução

1.1. Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante pode ser descrito da seguinte forma:

"Dada uma lista de cidades e as distâncias entre cada par de cidades, encontre o menor caminho que passe por todas as cidades apenas uma vez e retorne à cidade original."

Este problema é classificado na Teoria da Computação como NP-Difícil. Uma das consequências de ser NP-Difícil é que não se conhece um algoritmo que pode ser executado em tempo polinomial em uma máquina de Turing determinística para resolver este problema.

1.2. MTZ

Podemos formular diversos modelos para tentar resolver esse problema utilizando programação matemática. O modelo utilizado neste trabalho foi o modelo de Miller, Tucker e Zemlin - MTZ. Este modelo tem as seguintes características:

Função objetivo:

$$Min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij}$$

Onde:

- c_{ij} indica o custo de ir da cidade i até a cidade j.
- x_{ij} é uma variável de decisão que vale 1 se o caminho contém a aresta ij e 0 caso contrário.

Restrições:

$\sum_{j=1}^{N} x_{ij} = 1, \ \forall j \in \{1, ..., N\}$, ou seja, o grau de saída de cada nó deve valer 1.

$\sum_{i=1}^{N} x_{ij} = 1, \ \forall i \in \{1, ..., N\}$, ou seja, o grau de entrada de cada nó deve valer 1.

$x_{ii} = 0, \ \forall i \in \{1, ..., N\}$, não há aresta do nó para ele mesmo.

É colocada uma restrição para evitar ciclos na solução:

$u_j \geq u_i + x_{ij} - n(1 - x_{ij}), \ i \in \{1, ..., N\}, \ j \in \{2, ..., N\}, \ i \neq j, \ u_1 = 1$

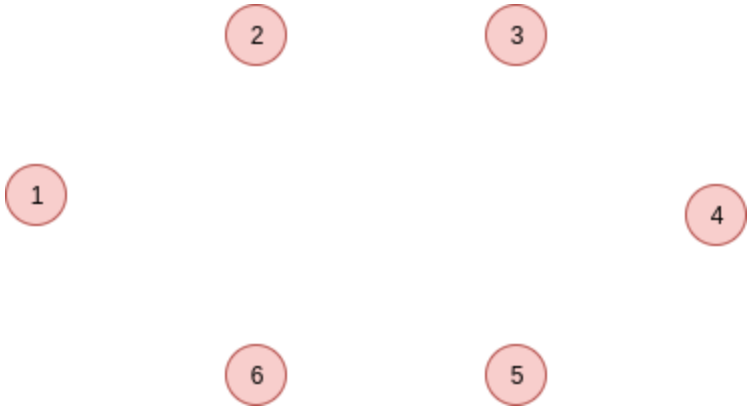
A variável u_i indica a ordem em que serão visitados os nós. Assim em $u_1 = 1$, o índice do u_1 indica o nó visitado, que nesse caso é o nó 1. Já do lado direito da equação, o 1 indica a ordem em que esse nó foi visitado, nesse caso foi o primeiro nó a ser visitado.

Por fim temos as restrições de não negatividade e integralidade:

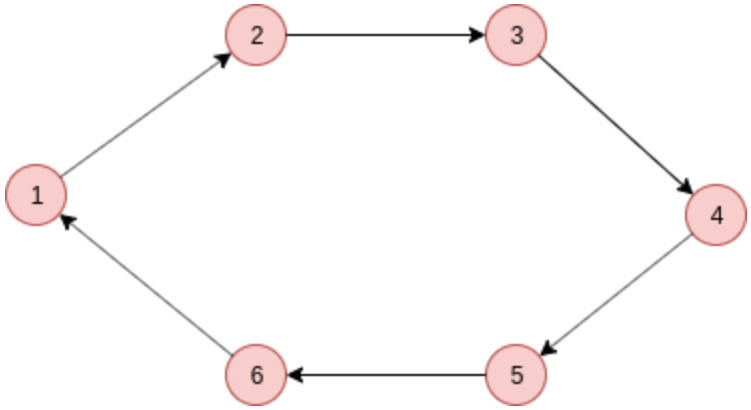
$u_i \geq 0$, $u_i \in Z$, $\forall i \in \{1, ..., N\}$ e $x_{ij} \in \{0, 1\}$, $\forall i \in \{1, ..., N\}, \ \forall j \in \{1, ..., N\}$

1.3. Exemplo Ilustrativo

Para deixar a modelagem um pouco mais clara, suponha o exemplo de cidades distribuídas geograficamente da seguinte forma:



Com esse conjunto de cidades, o menor caminho que passa por todas as cidades seria:



Os valores das variáveis de decisão do nosso problema seriam:

```
U = [1,2,3,4,5,6]
```

Onde o valor u_i está colocado na posição i do vetor U.

```
X = [
      [0,1,0,0,0,0]
      [0,0,1,0,0,0]
      [0,0,0,1,0,0]
      [0,0,0,0,1,0]
      [0,0,0,0,0,1]
      [1,0,0,0,0,0]
]
```

Onde o valor x_{ij} está colocado na posição de linha i e coluna j da matriz X.

2. Implementação

2.1. Resolvendo relaxações lineares

Escolhemos quatro problemas, entre eles o burma14 e os resolvemos utilizando relaxação linear. Os problemas escolhidos foram:

- Burma14
- Ulysses16
- Berlin52
- A280

O número no nome dos arquivos indica o número de cidades.

Durante os testes observamos que o arquivo burma14.tsp, aparentemente, teve suas coordenadas divididas por 100. Assim, o resultado obtido aqui foi multiplicado por 100 para comparar com a solução ótima que consta no site.

Com a solução da relaxação linear obtivemos os seguintes resultados:

Problema	Tempo de execução (s)	Valor da relaxação	Solução ótima	Gap (%)
burma14	0,017	2690,98	3323	-19
ulysses16	0,021	5934,74	6859	-13,5
berlin52	0,170	6317,59	7542	-16,2
a280	8,734	2431,19	2579	-5,7

2.2. Resolvendo problemas inteiros

Dos sete problemas escolhidos acima, apenas os problemas burma14 e ulyssess16 puderam ser resolvidos em tempo hábil utilizando variáveis de decisão binárias. Não foi possível resolver os outros problemas no tempo especificado.

Novamente tivemos problema com o arquivo burma14.tsp, agora conseguimos encontrar uma solução melhor do que a solução “ótima” indicada no site. Acreditamos que o valor da solução ótima que consta no site está errado ou o arquivo burma14.tsp está errado

De qualquer forma, abaixo estão os resultados obtidos ao tentar resolver os problemas inteiros:

Problema	Tempo de execução (s)	Solução encontrada	Solução ótima	Gap (%)
burma14	3,863	3087,85	3323	-7.1
ulysses16	66,40406441688538	7398,76	6859	7,9

3. Heurísticas

Para a resolução com heurísticas utilizamos uma heurística construtiva e uma de melhoria.

A heurística construtiva constitui de sempre pegar o vizinho mais próximo ao último nó visitado. Precisamos ter cuidado para não pegarmos nós que já foram visitados ao construir a solução.

A heurística de melhoria adotada foi variar a solução fazendo trocas 2 a 2 entre nós vizinhos na solução original. Caso a solução não melhore, fazemos trocas 3 a 3 e assim por diante. Na primeira melhoria escolhemos a nova solução.

O código em python implementado está anexado junto com este relatório.

3.1. Comparando tempo

Durante a execução medimos o tempo para cada heurística:

Problema	**Tempo da relaxação (s)**	**Tempo da heurística gulosa (s)**	**Tempo da heurística de melhoria (s)**
burma14	0,017	~0	~0
ulysses16	0,021	~0	~0
berlin52	0,170	~0	~0
a280	8,734	0,011	0,011

Como podemos ver, as heurísticas foram muito mais rápidas até que a própria relaxação linear. Além disso, a heurística de melhoria quase não consumiu tempo, a alteração aconteceu depois da terceira casa e por isso não está visível na tabela.

3.2. Comparando soluções

Para podermos comparar os valores de soluções obtidos foi construída a seguinte tabela:

Problema	**Solução ótima**	**Valor da heurística gulosa**	**Valor da heurística de melhoria**	**Gap (%)**
burma14	3323	3868,81	3156,48	-5
ulysses16	6859	10473,49	8045,43	17,3
berlin52	7542	8980,91	8379,82	11,1
a280	2579	3148,10	3101,78	20,3

Podemos ver que apesar de as heurísticas terem gasto pouco tempo para executar, conseguiram um resultado relativamente bom. Considerando que o maior gap foi de 20,3 % e o tempo de execução da heurística foi quase imediato, poderíamos utilizar essa solução heurística como ponto de partida para o método exato e pouparíamos bastante tempo para encontrar a solução ótima.

Novamente o arquivo burma14 apresentou problema, pois acabamos achando uma solução heurística melhor que a “ótima” apresentada. Como já comentado, pode haver um erro no site ou no arquivo, não sabemos.

4. Referências

Foi utilizada a modelagem do MTZ apresentada no material auxiliar: “Tópicos em Otimização II Otimização aplicada ao Roteamento de Veículos”.
Descrição do TSP:
https://pt.wikipedia.org/wiki/Problema_do_caixeiro-viajante
Imagem da capa:
https://upload.wikimedia.org/wikipedia/commons/thumb/7/72/Max_paraboloid.svg/1200px-Max_paraboloid.svg.png
Dados para os TSPs simétricos:
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/
Soluções ótimas para os TSPs simétricos:
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html