EVENTO ON-LINE
ConBRepro
XII CONGRESSO BRASILEIRO DE ENGENHARIA DE PRODUÇÃO
ESG nas Engenharias
30 a 02 de dezembro 2022

# Problema do Caixeiro Viajante: Um comparativo da utilização da linguagem C++ com auxílio do *Gurobi* e o *Solver* do Excel.

**Michel do Vale Pereda**
**Departamentos de Construção Civil e de Matemática, Setores de Tecnologia e de Ciências Exatas - UFPR**

**Resumo:** O presente artigo mostra uma visão simplificada do Problema do Caixeiro Viajante (PVC) quanto ao seu histórico de resolução e aplicabilidade logística. Visto a complexidade de sua resolução, foram realizados dois testes computacionais comparativos, com 10 e 15 nós, empregando o suplemento Solver do Microsoft Office Excel e a linguagem de programação C++ com o otimizador Gurobi. Observou-se que, até certa quantidade de nós (cidades) aplicadas no problema, o Solver do Excel atingiu solução ótima com tempo computacional pouco acima do C++ com Gurobi. Contudo, com o aumento de nós, o problema atinge um grau de complexidade não suportado pelo Solver, mas resolvido com destreza pelo Gurobi na linguagem C++.

**Palavras-chave:** Problema do Caixeiro Viajante. Linguagem de Programação C++. Gurobi. Solver.

# TRAVELING SALESMAN PROBLEM: A COMPARATIVE USE OF THE C++ LANGUAGE WITH THE HELP OF GUROBI AND EXCEL SOLVER

**Abstract:** This paper shows a simplified view of the Traveling Salesman Problem (PVC) regarding its solving history and logistical applicability. Considering the complexity of its resolution, two comparative computational tests were performed, with 10 and 15 nodes, using the Microsoft Office Excel add-in Solver and the C++ programming language with the Gurobi optimizer. It was observed that, up to a certain amount of nodes (cities) applied in the problem, the Excel Solver reached an optimal solution with computational time slightly above C++ with Gurobi. However, with the increase of nodes, the problem reaches a degree of complexity not supported by the Solver, but solved with skill by the Gurobi in C++ language.

**Keywords:** Traveling Salesman Problem. C++ Programming Language. Gurobi. Solver.

## 1. Introdução

Com o crescimento do mercado, observa-se cada vez mais a importância de um processo logístico bem estruturado, visando a redução dos custos e serviços diferenciados aos clientes com o intuito de se manter a competitividade disponibilizando o produto certo, no menor tempo e com custo mínimo para a entrega (MACHADO; ROCHA; PACHECO, 2015).

O Problema do Caixeiro Viajante (PCV ou *Traveling Salesman Problem* – TSP) consiste em calcular a melhor rota, com menor custo possível, deslocando-se de um ponto inicial, passando por todos os nós do grafo e retornando ao ponto inicial.

Uma das várias aplicabilidades cabíveis ao PCV é a distribuição de produtos através de entregas onde, dependendo da empresa, a definição de rota diária é realizada de maneira empírica, apenas embasada nos conhecimentos prévios que os motoristas têm em relação às cidades em questão. Por conta disso, o suplemento Solver poderá se tornar um mecanismo no auxílio dessas rotas.

O presente artigo tem por proposta realizar dois comparativos entre o tempo utilizado para gerar uma solução ótima do PCV entre o Solver, suplemento do Microsoft Excel, e a linguagem de programação C++ com a utilização do otimizador Gurobi.

## 2. Metodologia

O presente artigo tem por proposta observar e avaliar o desempenho/ tempo computacional ao tratar dois testes do PCV envolvendo o Solver do Excel 2019 e o Gurobi aplicado na linguagem C++. Para tal, os experimentos foram realizados em um computador com processador Intel(R) Core (TM) i3-10110U CPU @ 2.10GHz/2.59 GHz, 12 GB de memória RAM e sistema operacional Windows 10.

Os dados empregados para alimentar os modelos de otimização nos testes foram gerados aleatoriamente. Contudo, os mesmos dados foram utilizados tanto no Excel quanto em C++, nos dois comparativos.

O Comparativo 1 consiste em um modelo com 10 nós (cidades). Já o segundo, aplicou-se um Problema do Caixeiro Viajante com 15 nós.

## 3. Revisão de Literatura

### 3.1 O Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante corresponde a um problema de programação binária que pode ser modelado como uma problematização de programação em redes (BELFIORI e FÁVERO, 2013).

Objeto de estudo desde o século XIX, possui um amplo ramo de pesquisa atualmente, tendo em conta o forte impacto no comércio mundial. Possuindo variações, o PCV pode dispor de várias heurísticas e meta-heurísticas para obtenção da solução e refinamento da solução (REBOUÇAS, 2016).

Consiste em encontrar um caminho através de um conjunto de cidades afim de minimizar o tempo, distância ou custo total entre as cidades, retornando à cidade de partida. Trata-se de um ciclo hamiltoniano, uma vez que todas as cidades serão visitadas, apenas uma vez, com exceção da cidade de partida, a qual o caixeiro retorna ao final do ciclo.

BELFIORI e FÁVERO (2013, p. 394) ressaltam que

> O problema do caixeiro-viajante – PCV (*Traveling Salesman Problem* – TSP) é um problema de otimização associado à determinação dos caminhos denominados hamiltonianos. Sua origem advém de Willian Rowan Hamilton, que propôs um jogo cujo desafio consistia em encontrar uma rota por meio dos nós de um dodecaedro (sólido regular com 20 nós, 30 arcos e 12 faces) de tal modo que a rota iniciasse e terminasse no mesmo nó, sem nunca repetir uma visita. (BELFIORI e FÁVERO, 2013, p. 394).

Apesar de, aparentemente, ser um problema de simples resolução, o PCV é um problema de otimização combinatória, ou seja, dependendo da quantidade de cidades (*n*) a serem visitadas, o problema torna-se cada vez mais complexo, pois gerará uma quantidade de rotas fatorialmente, sendo simétrico ou não.

Pressupondo que *N* é o conjunto dos *n* nós (cidades) e *A* o conjunto dos arcos (estradas) entre as cidades e um grafo não orientado *G = (N, A)*, o modelo matemático de programação binária para o Problema do Caixeiro Viajante proposto pelos autores DANTZIG, FULKERSON e JOHNSON (1954, tradução autoral), utilizando os seguintes parâmetros:

$c_{ij}$ = custo (ou distância) da cidade *i* (com *i* = 1, ..., *n*) até a cidade *j* (com *j* = 1, ..., *n*).

$x_{ij}$ = $\begin{cases} 1 \text{ se o caixeiro irá direto da cidade } i \text{ para a cidade } j, \text{ com } i = j. \\ 0 \text{ caso o caixeiro não vá diretamente.} \end{cases}$

Tendo como função objetivo:

$$\min Z = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} c_{ij} \qquad (1)$$

Sujeito às seguintes restrições:

- $\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall\, j \in N \qquad (2)$
- $\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\, i \in N \qquad (3)$
- $\sum_{i,j \in S} x_{ij} \leq |S| - 1 \qquad \forall\, S \subset N \qquad (4)$
- $x_{ij} \in \{0,1\} \qquad \forall\, i,j \in N \qquad (5)$

As equações (2) e (3) nas restrições garantem que cada nó (cidade) seja explorado apenas uma vez. Já a restrição (3) tem por objetivo impossibilitar a criação de sub – rotas considerando que *S* é um sub-grafo de G. E, por fim, a restrição (5) garantem que as variáveis são binárias (programação binária, como dito anteriormente).

## 3.2 Linguagens de Programação C++

Relatos históricos apontam que a linguagem de programação já havia surgido muito antes de 1940 quando, efetivamente, os primeiro computadores e linguagens começaram a surgir no formato de códigos matemáticos que, tempos depois, deram origem a abstração de complexos cálculos matemáticos para utilização da resolução de problemas específicos (BERTOLINI *et al*, 2019).

As linguagens de programação têm por objetivo descrever uma sequência lógica de passos para obter a solução de um problema. A essa sequência de passos nomeamos de algoritmos.

Pode-se diferenciar as linguagens existentes em duas categorias principais conhecidas como linguagens de baixo e alto nível. Segundo WILLRICH (2000, cap. 4, p. 2) “As linguagens de programação podem ser classificadas em níveis de linguagens, sendo que os níveis mais baixos são mais próximos da linguagem interpretada pelo processador e mais distante das linguagens naturais”.

Já a linguagem de programação C++ iniciou-se como uma versão expandida da linguagem C. Essas extensões foram desenvolvidas por Bjarne Stroustrup no final da década de 1970 em Nova Jersey e foi chamada, a princípio, de *C with Classes* (C com Classes) passando a se chamar C++ (C *plus plus*) (SCHILDT, 1998).

Das categorias apresentadas anteriormente o C++ é classificado, por diversos autores, como sendo uma linguagem de alto nível. Entretanto, SHILDT (1998, tradução autoral) a considera como sendo uma linguagem de nível médio, uma vez que une as funcionalidades das linguagens de alto nível (orientação de objetos e classes por exemplo) com recursos das linguagens de baixo nível, como a manipulação de bytes.

## 4. Resultados

### 4.1. Testes Computacionais

#### 4.1.1 Comparativo 1

Como descrito anteriormente, o Comparativo 1 consiste em PCV com 10 nós para comparativo entre o suplemento Solver do Microsoft Office Excel e a programação na linguagem C++ com o otimizador Gurobi.

##### 4.1.1.1 Resolução com Solver do Microsoft Office Excel

Como pode-se observar na FIGURA 1, a modelagem matemática do problema no suplemento Solver do Excel possui uma matriz distância onde seus valores foram gerados aleatoriamente.

FIGURA 1 – MATRIZ DISTÂNCIA

| Distância | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 39 | 97 | 51 | 38 | 59 | 19 | 66 | 20 | 29 |
| 2 | 39 | 0 | 85 | 17 | 84 | 86 | 92 | 30 | 18 | 58 |
| 3 | 97 | 85 | 0 | 99 | 52 | 11 | 5 | 89 | 3 | 1 |
| 4 | 51 | 17 | 99 | 0 | 100 | 84 | 29 | 16 | 30 | 7 |
| 5 | 38 | 84 | 52 | 100 | 0 | 53 | 24 | 4 | 89 | 42 |
| 6 | 59 | 86 | 11 | 84 | 53 | 0 | 78 | 5 | 55 | 55 |
| 7 | 19 | 92 | 5 | 29 | 24 | 78 | 0 | 63 | 39 | 24 |
| 8 | 66 | 30 | 89 | 16 | 4 | 5 | 63 | 0 | 4 | 75 |
| 9 | 20 | 18 | 3 | 30 | 89 | 55 | 39 | 4 | 0 | 34 |
| 10 | 29 | 58 | 1 | 7 | 42 | 55 | 24 | 75 | 34 | 0 |

FONTE: o autor (2021)

As variáveis de decisão foram formuladas na matriz representada na FIGURA 2, bem como as restrições para o problema.

FIGURA 2 – MATRIZ VARIÁVEIS E RESTRIÇÕES

Matriz Variáves de Decisão (Xij)

| Xij | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Restrição 1

| | | |
|---|---|---|
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |

Restrição 2

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | = | = | = | = | = | = | = | = | = | = |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | | |
|---|---|---|
| 0 | = | 0 |

Restrição 3

| | |
|---|---|
| Valor ótimo | 0 |

FONTE: Autor (2021)

A Restrição 1 garante que, para um valor de *i* (para todo *i* = 1, ...,10), terá apenas um valor de *j* correspondente. Bem como a Restrição 2 assegura que, para cada *j* (para todo *j* = 1, ..., 10), possuirá apenas um valor de *i*. Já a Restrição 3 afirma a inexistência dos caminhos entre *i* e *j*, quando *i* = *j*.

Otimizando com suplemento Solver do Microsoft Office Excel, encontra-se os resultados e parâmetros da FIGURA 3.

FIGURA 3 – RESULTADOS

**Resultado: O Solver encontrou uma solução. Todas as Restrições e condições de adequação foram satisfeitas.**

**Mecanismo do Solver**

Mecanismo: LP Simplex

Tempo da Solução: 0,062 Segundos.

Iterações: 96 Subproblemas: 0

**Opções do Solver**

Tempo Máx. Ilimitado, Iterações Ilimitado, Precision 0,000001

Subproblemas Máx. Ilimitado, Soluç. Máx. Núm. Inteiro Ilimitado, Tolerância de Número Inteiro 1%, Assumir Não Negativo

Célula do Objetivo (Mín.)

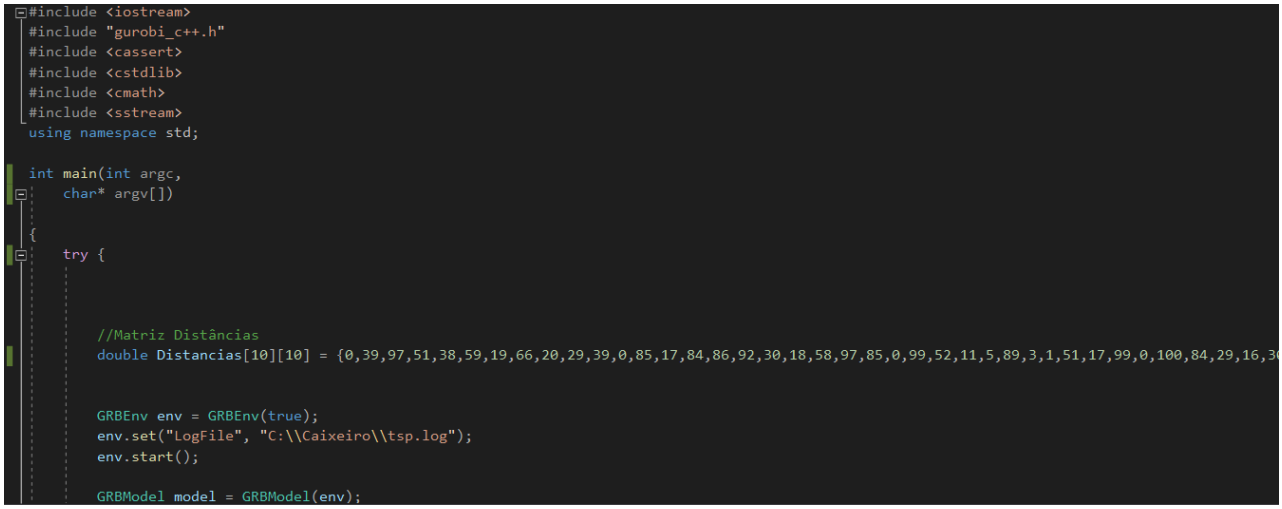
| Célula | Nome | Valor Original | Valor Final |
|---|---|---|---|
| $K$31 | Valor ótimo = | 0 | 118 |

FONTE: Autor (2021)

4.1.1.2 Resolução com Linguagem C++ e Otimizador *Gurobi*

Tomando como base de cálculo a mesma matriz distância utilizada no problema resolvido, foi gerado o modelo matemático com a aplicação do otimizador Gurobi, como observa-se parte do mesmo na FIGURA 4.

FIGURA 4 – PARTE DO CÓDIGO EM C++



```
#include <iostream>
#include "gurobi_c++.h"
#include <cassert>
#include <cstdlib>
#include <cmath>
#include <sstream>
using namespace std;

int main(int argc,
    char* argv[])

{
    try {



      //Matriz Distâncias
      double Distancias[10][10] = {0,39,97,51,38,59,19,66,20,29,39,0,85,17,84,86,92,30,18,58,97,85,0,99,52,11,5,89,3,1,51,17,99,0,100,84,29,16,3


      GRBEnv env = GRBEnv(true);
      env.set("LogFile", "C:\\Caixeiro\\tsp.log");
      env.start();

      GRBModel model = GRBModel(env);
```

FONTE: Autor (2021)

O resultado ótimo alcançado foi o mesmo, entretanto, com tempo de resolução e iterações diferentes (FIGURA 5).

FIGURA 5 – RESULTADOS EM C++

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0     118.0000000  118.00000  0.00%     -    0s

Explored 0 nodes (19 simplex iterations) in 0.02 seconds
Thread count was 4 (of 4 available processors)
```

FONTE: Autor (2021)

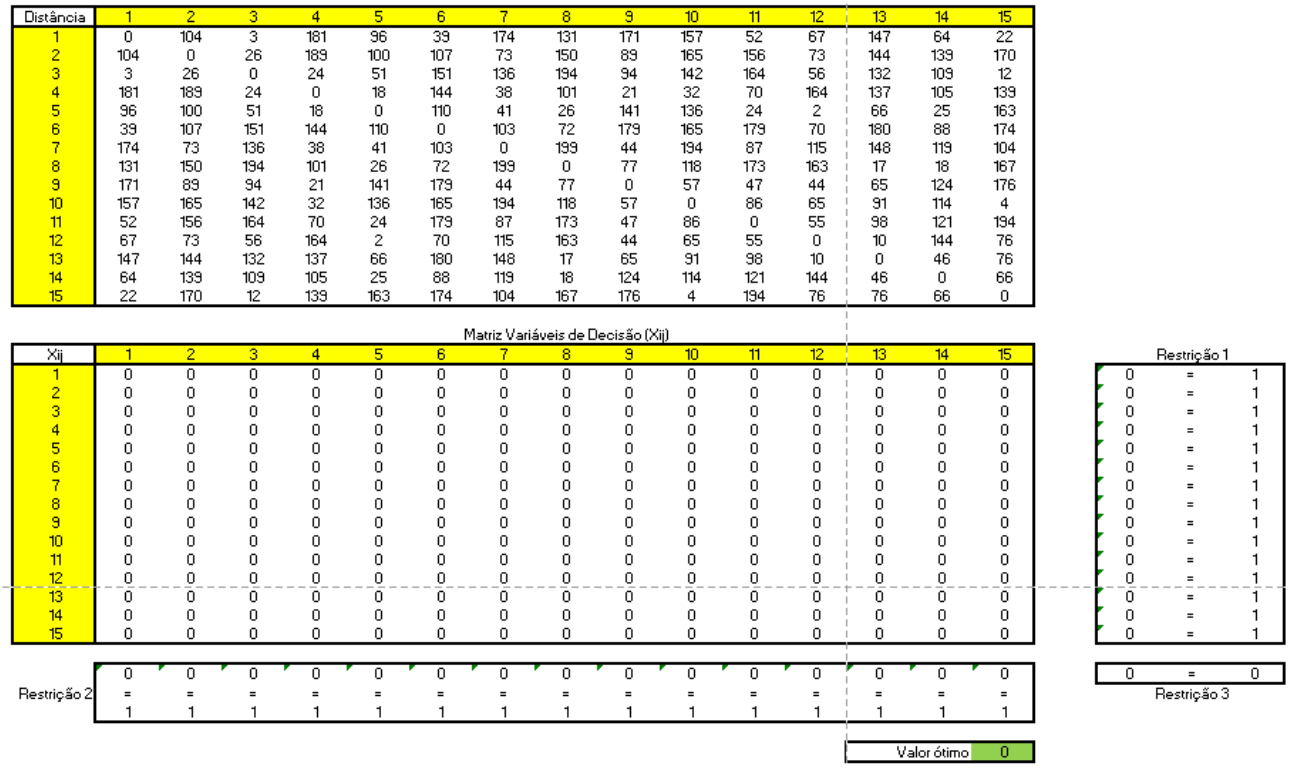
### 4.1.2 Comparativo 2

Os testes computacionais apresentados no Comparativo 2 também referem-se ao Problema do Caixeiro Viajante, contudo, foi utilizado um grafo de 15 nós, com valores gerados aleatoriamente para a matriz distância.

#### 4.1.2.1 Resolução com Solver do Microsoft Office Excel

A modelagem matemática consiste na mesma empregada no teste computacional do Comparativo 1 (FIGURA 6).

FIGURA 6 – MODELAGEM MATEMÁTICA



| Distância | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 104 | 3 | 181 | 96 | 39 | 174 | 131 | 171 | 157 | 52 | 67 | 147 | 64 | 22 |
| 2 | 104 | 0 | 26 | 189 | 100 | 107 | 73 | 150 | 89 | 165 | 156 | 73 | 144 | 139 | 170 |
| 3 | 3 | 26 | 0 | 24 | 51 | 151 | 136 | 194 | 94 | 142 | 164 | 56 | 132 | 109 | 12 |
| 4 | 181 | 189 | 24 | 0 | 18 | 144 | 38 | 101 | 21 | 32 | 70 | 164 | 137 | 105 | 139 |
| 5 | 96 | 100 | 51 | 18 | 0 | 110 | 41 | 26 | 141 | 136 | 24 | 2 | 66 | 25 | 163 |
| 6 | 39 | 107 | 151 | 144 | 110 | 0 | 103 | 72 | 179 | 165 | 179 | 70 | 180 | 88 | 174 |
| 7 | 174 | 73 | 136 | 38 | 41 | 103 | 0 | 199 | 44 | 194 | 87 | 115 | 148 | 119 | 104 |
| 8 | 131 | 150 | 194 | 101 | 26 | 72 | 199 | 0 | 77 | 118 | 173 | 163 | 17 | 18 | 167 |
| 9 | 171 | 89 | 94 | 21 | 141 | 179 | 44 | 77 | 0 | 57 | 47 | 44 | 65 | 124 | 176 |
| 10 | 157 | 165 | 142 | 32 | 136 | 165 | 194 | 118 | 57 | 0 | 86 | 65 | 91 | 114 | 4 |
| 11 | 52 | 156 | 164 | 70 | 24 | 179 | 87 | 173 | 47 | 86 | 0 | 55 | 98 | 121 | 194 |
| 12 | 67 | 73 | 56 | 164 | 2 | 70 | 115 | 163 | 44 | 65 | 55 | 0 | 10 | 144 | 76 |
| 13 | 147 | 144 | 132 | 137 | 66 | 180 | 148 | 17 | 65 | 91 | 98 | 10 | 0 | 46 | 76 |
| 14 | 64 | 139 | 109 | 105 | 25 | 88 | 119 | 18 | 124 | 114 | 121 | 144 | 46 | 0 | 66 |
| 15 | 22 | 170 | 12 | 139 | 163 | 174 | 104 | 167 | 176 | 4 | 194 | 76 | 76 | 66 | 0 |

Matriz Variáveis de Decisão (Xij)

| Xij | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Restrição 1 | | |
|---|---|---|
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |
| 0 | = | 1 |

| Restrição 2 | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | = | = | = | = | = | = | = | = | = | = | = | = | = | = | = | |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

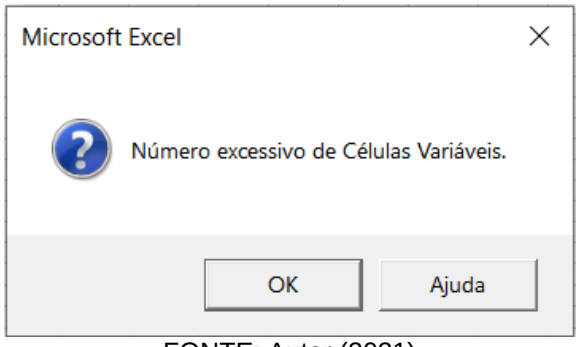
| Restrição 3 | | |
|---|---|---|
| 0 | = | 0 |

Valor ótimo: 0

FONTE: Autor (2021)

No entanto, o suplemento Solver do Microsoft Excel não suportou a resolução, apresentando uma mensagem em tela (FIGURA 7).

FIGURA 7 – MENSAGEM




FONTE: Autor (2021)

4.1.2.2 Resolução com Linguagem C++ e Otimizador Gurobi

Empregando o mesmo código utilizado anteriormente para a resolução do problema envolvendo 10 nós (apenas realizando as alterações necessárias no tamanho da matriz; o código completo encontra-se no apêndice) e, como matriz distância, os mesmos valores aplicados no Excel do experimento com 15 nós , obtém-se a solução ótima (FIGURA 8) para o Problema do Caixeiro Viajante.

FIGURA 8: SOLUÇÃO ÓTIMA TESTE 2

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0     345.0000000  345.00000  0.00%     -    0s

Explored 0 nodes (24 simplex iterations) in 0.03 seconds
Thread count was 4 (of 4 available processors)
```

FONTE: Autor (2021)

## 5. Conclusões

O Problema do Caixeiro Viajante tem grande aplicabilidade na área logística e pode ser adaptado a outras situações com suas variações. Para a resolução do PVC clássico observa-se, através dos testes computacionais do presente artigo, uma limitação no que tange o uso do suplemento Solver do Microsoft Office Excel.

Observa-se através da TABELA 1 que, para um problema clássico simétrico de 10 nós, os dois métodos para resolução atingiram o valor ótimo, sem sub-rotas, com uma diferença de tempo computacional de 0,042 segundo. Entretanto, o Solver do Excel realizou 77 iterações a mais que o modelo gerado na linguagem C++.

TABELA 1 – RESULTADOS OBTIDOS NOS DOIS TESTES COMPUTACIONAIS

| | **Teste Computacional 1** | | **Teste Computacional 2** | |
|---|---|---|---|---|
| | **Solver Excel** | **C++ e *Gurobi*** | **Solver Excel** | **C++ e *Gurobi*** |
| **Nós** | 10 | 10 | 15 | 15 |
| **Solução Ótima** | 118 | 118 | - | 345 |
| **Nº de Iterações** | 96 | 19 | - | 24 |
| **Tempo Computacional (em segundo)** | 0,062 | 0,02 | - | 0,03 |

FONTE: Autor (2021)

Já na observação do segundo teste computacional, encontra-se uma diferença evidente entre as duas formas de resolução do mesmo problema. O suplemento Solver do Microsoft Office Excel não foi capaz de fornecer a solução ótima, uma vez que gera a mensagem de "Número excessivo de Células Variáveis" (FIGURA 7). Contudo, a linguagem C++ com o auxílio do otimizador Gurobi mostrou exímio desempenho obtendo a solução ótima em 0,03 segundos e um total de 24 iterações.

**Referências**


BELFIORE, P.; FÁVERO, L.P. **Pesquisa Operacional Para Cursos de Engenharia.** Rio de Janeiro: Elsevier, 2013.

BERTOLINI, C. *et al*. **Linguagem de Programação I**. Santa Maria: UFSM, NTE, 2019. E-book. ISBN 978-85-8341-246-5. Disponível em: https://www.ufsm.br/app/uploads/sites/358/2020/02/linguagem-1.pdf (Acesso em: 20 junho 2021).

DANTZIG, G.B.; FULKERSON, D.R.; JOHNSON, S.M. *Solution of a Large-Scale Traveling-Salesman Problem*. ***Journal of the Operations Research Society of America*****,** v. 02, n. 04, p. 393 – 410, 1954. Disponível em: https:// www.jstor.org/stable/166695 (Acesso em: 27 junho 2021).

MACHADO, A. N. U.; ROCHA, B. S.; PACHECO, D. A. J. Diagnóstico e Proposta de Melhorias para a Cadeia de Suprimentos de Microempresas Distribuidoras de Alimentos. **Produção em Foco**. Joinville, v. 05, n.01, p. 26-41, maio 2014. Disponível em: https://www.researchgate.net/profile/Diego-Pacheco-7/publication/276433991_Diagnostico_e_proposta_de_melhorias_para_a_cadeia_de_suprimentos_de_microempresas_distribuidoras_de_alimentos/links/564622c308ae9f9c13e73b2f/Diagnostico-e-proposta-de-melhorias-para-a-cadeia-de-suprimentos-de-microempresas-distribuidoras-de-alimentos.pdf (Acesso em: 25 junho 2021).

PRESTES, A. N. **Uma Análise Experimental de Abordagens Heurísticas Aplicadas ao Problema do Caixeiro Viajante.** Dissertação (Mestrado em Sistemas e Computação) – Universidade Federal do Rio Grande do Norte. Natal. 2006.

REBOUÇAS, R. S. **Problema do Caixeiro Viajante com Coleta de Prêmios e Janelas de Tempo.** Dissertação (Mestrado em Matemática Aplicada) - Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas. Campinas. 2016.

SHILDT, H. ***C++: The Complite Reference*****.** Terceira edição. EUA: McGRAW-HILL, 1998. Disponível em: http://www.uml.org.cn/c%2B%2B/pdf/C%2B%2BComplete%20Reference%20(3rd%20Ed.).pdf (Acesso em: 21 de junho de 2021)

WILLRICH, R. Linguagens de Programação. *In:* **Introdução à Informática.** Florianópolis. UFSC: 2000. P. 2-15. Disponível em: http://algol.dcc.ufla.br/~monserrat/icc/Introducao_linguagens.pdf (Acesso em: 22 de junho 2021).

## ANEXO

```
// Programa de Pós-Graduação em Métodos Numéricos em Engenharia
// Disciplina: Tópicos Especiais em Programação Matemática
// Professor Dr. Gustavo Valentim Loch.
// Problema do Caixeiro Viajante
// Aluno: Michel do Vale Pereda


#include <iostream>
#include "gurobi_c++.h"
#include <cassert>
#include <cstdlib>
#include <cmath>
#include <sstream>
using namespace std;

int main(int argc,
    char* argv[])

{
    try {



        //Matriz Distâncias
        double                    Distancias[15][15]                    =                    {
0,104,3,181,96,39,174,131,171,157,52,67,147,64,22,104,0,26,189,

100,107,73,150,89,165,156,73,144,139,170,3,26,0,24,51,151,136,194,

94,142,164,56,132,109,12,181,189,24,0,18,144,38,101,21,32,70,164,

137,105,139,96,100,51,18,0,110,41,26,141,136,24,2,66,25,163,39,107,

151,144,110,0,103,72,179,165,179,70,180,88,174,174,73,136,38,41,103,

0,199,44,194,87,115,148,119,104,131,150,194,101,26,72,199,0,77,118,

173,163,17,18,167,171,89,94,21,141,179,44,77,0,57,47,44,65,124,176,

157,165,142,32,136,165,194,118,57,0,86,65,91,114,4,52,156,164,70,24,

179,87,173,47,86,0,55,98,121,194,67,73,56,164,2,70,115,163,44,65,55,

0,10,144,76,147,144,132,137,66,180,148,17,65,91,98,10,0,46,76,64,139,

109,105,25,88,119,18,124,114,121,144,46,0,66,22,170,12,139,163,174,104,
                                    167,176,4,194,76,76,66,0 };


        GRBEnv env = GRBEnv(true);
        env.set("LogFile", "C:\\Caixeiro\\tsp.log");
        env.start();

        GRBModel model = GRBModel(env);

        //Variáveis de Decisão
        GRBVar X[15][15];
        for (int i = 0; i < 15; i++)
        {
            for (int j = 0; j < 15; j++)
            {
                if (j != i)
                {
```

```
                    X[i][j] = model.addVar(0, 1, Distancias[i][j], GRB_BINARY, "X_" +
to_string(i) + "_" + to_string(j));
                }

            }

        }

        GRBLinExpr expr = 0;

        //Restrições
            //Restrição 1
        for (int i = 0; i < 15; i++)

        {
            expr = 0;
            for (int j = 0; j < 15; j++)
            {
                if (i != j)
                    expr += X[i][j];

            }
            model.addConstr(expr == 1, " R_" + to_string(i));
        }
        //Restrição 2
        for (int j = 0; j < 15; j++)

        {
            expr = 0;
            for (int i = 0; i < 15; i++)
            {
                if (j != i)
                    expr += X[i][j];

            }
            model.addConstr(expr == 1, " R2_" + to_string(j));

        }
        // Restricoes (evitar sub-rotas)
        model.addConstr(X[0][8] + X[8][0] <= 1, "R_0_8_0");
        // Modelo Otimizado
        model.optimize();

        model.write("C:\\Caixeiro10\\modelPCV.lp");
        model.write("C:\\Caixeiro10\\modelPCV.sol");

        cout << "Obj: " << model.get(GRB_DoubleAttr_ObjVal) << endl;

    }
    catch (GRBException e) {
        cout << "Error Code = " << e.getErrorCode() << endl;
    }



}
```