UNIVERSIDADE DO ESTADO DE MATO GROSSO CAMPUS DE SINOP

FACULDADE DE CIÊNCIAS EXATAS E TECNOLÓGICAS

MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL – PROFMAT

FELIPE NASCIMENTO DE SOUZA LEÃO

SOLUÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE UTILIZANDO A META-HEURÍSTICA BUSCA TABU NO MATLAB®

SINOP

2021

FELIPE NASCIMENTO DE SOUZA LEÃO

SOLUÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE UTILIZANDO A META-HEURÍSTICA BUSCA TABU, NO MATLAB®

Dissertação apresentada ao Programa de Mestrado Profissional em Matemática em Rede Nacional – PROFMAT, do departamento de Matemática da Universidade do Estado do Mato Grosso – UNEMAT, como requisito parcial para obtenção do grau de Mestre em Matemática.

Prof. Dr. Emivan Ferreira da Silva
Orientador

Prof. Dra. Adriana Souza Resende
Coorientadora

Sinop

2021

Luiz Kenji Umeno Alencar CRB 1/2037

LEÃO, Felipe Nascimento de Souza.
L433s Solução do Problema do Caixeiro Viajante Utilizando a Meta-Heurística Busca Tabu no Matlab® / Felipe Nascimento de Souza Leão – Sinop, 2021.
58 f.; 30 cm. (ilustrações) Il. color. (sim)

Trabalho de Conclusão de Curso (Dissertação/Mestrado) – Curso de Pós-graduação Stricto Sensu (Mestrado Profissional) Profmat, Faculdade de Ciências Exatas e Tecnológicas, Câmpus de Sinop, Universidade do Estado de Mato Grosso, 2021.
Orientador: Emivan Ferreira da Silva
Coorientador: Adriana Souza Resende

1. Meta-Heurística. 2. Busca Tabu. 3. Programação. 4. Matlab®. I. Felipe Nascimento de Souza Leão. II. Solução do Problema do Caixeiro Viajante Utilizando a Meta-Heurística Busca Tabu no Matlab®: .

CDU 004.434:5

FELIPE NASCIMENTO DE SOUZA LEÃO

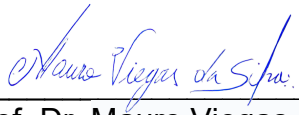
SOLUÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE UTILIZANDO A META-HEURÍSTICA BUSCA TABU, NO MATLAB

Dissertação apresentada ao Programa de Mestrado Profissional em Matemática em Rede Nacional – Profmat da Universidade do Estado de Mato Grosso/UNEMAT – Campus Universitário de Sinop, como requisito parcial para obtenção do título de Mestre em Matemática.

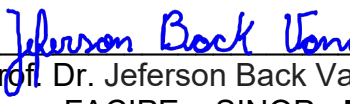
Orientador: Prof. Dr. Emivan Ferreira da Silva
Coorientadora: Adriana Souza Resende
Aprovado em 29/10/2021

BANCA EXAMINADORA

Prof. Dr. Emivan Ferreira da Silva
UNEMAT – SINOP - MT



Prof. Dr. Mauro Viegas da Silva
UNEMAT – CÁCERES - MT



Prof. Dr. Jeferson Back Vanderlinde
FACIPE – SINOP - MT

Sinop/MT
2021

AGRADECIMENTOS

Não dá para negar que foi um período de grandes batalhas. Muita dedicação, muita entrega, muitas escolhas.

Primeiramente agradecer a nosso pai, Jesus Cristo, Senhor dos senhores. Responsável pela nossa vida, por nossas vitórias, por tudo que nos rodeia.

Agradecer então a minha família. Minha esposa Giselli Leão, grande pilar, que sempre me incentivou, me apoiou (diria até que me suportou) e que nunca saiu do meu lado. Aos meus filhos Lis e Téo, ainda criança, mesmo que inconscientemente, suportaram minhas ausências nos momentos de estudo (que não foram poucos). Ao meu pai Souza e minha mãe Geralda, e irmãos Victor e Bárbara, primeiros incentivadores que sempre me mostraram que a educação, estudo e esforço são sempre o melhor caminho para a realização de sonhos.

Um agradecimento muito especial a meu orientador, Prof. Dr. Emivan, e minha coorientadora Prof. Dra. Adriana, que carregam consigo a nobre arte, o dom e a responsabilidade da docência não desistindo de mim, entendendo e apoiando. Fundamental para o desenvolvimento e conclusão deste trabalho. Sem sua orientação não teria conseguido.

Quero agradecer ao Prof. Antônio, ex-aluno do PROFMAT e grande amigo, que me apresentou o programa e me incentivou (obrigou) a entrar nele. Se não fosse por ele não teria iniciado no programa.

Aos meus amigos da classe, grandes amigos, grandes professores, grandes parceiros, mais que fundamentais durante todo o programa, me ajudando e com muita cordialidade e compromisso, dividindo seu conhecimento.

RESUMO

Nesse trabalho, é apresentado o Problema do Caixeiro Viajante (PCV) e uma metodologia para resolvê-lo que é a meta-heurística Busca Tabu (BT), como estratégia matemática e o *software* de programação *MATLAB*® como ferramenta para implementação do modelo. Assim, tem-se por objetivo geral, resolver o PCV e como objetivo específico, refletir a importância do tema programação computacional para alunos do ensino médio. Como metodologia da pesquisa, fez-se a leitura de alguns livros e sites de internet, para melhor conhecer problemas de otimização, em especial o PCV. Além disso, as meta-heurísticas e em particular a BT, por fim a construção do algoritmo e a implementação no *MATLAB*®. O uso do *software MATLAB*® e do *GNU-OCTAVE* para fins de comparação dos resultados. Como resultados, a solução do PCV nos dois *softwares* propostos. Além disso, foi resolvido um problema de Programação Linear (PL) utilizando o método geométrico e um código fonte baseado em BT para o *software Python*.

Palavras-chave: MATLAB®, meta-heurística, busca tabu, programação

ABSTRACT

This work presents the Traveling Salesman Problem (PCV) and a methodology to solve it, which is the Tabu Search (BT) meta-heuristic, as a mathematical strategy and the MATLAB® programming software as a tool to implement the model. Thus, the general objective is to solve PCV and, as a specific objective, to reflect the importance of the subject of computer programming for high school students. As a research methodology, some books and internet sites were read to better understand optimization problems, especially PCV. Furthermore, the meta-heuristics and in particular the BT, finally the construction of the algorithm and its implementation in MATLAB®. The use of MATLAB® and GNU-OCTAVE software for comparison purposes. As a result, the PCV solution in the two proposed software. Furthermore, a Linear Programming (PL) problem was solved using the geometric method and a BT-based source code for the Python software.

Keywords: *MATLAB®*, metaheuristics, *Tabu Search*, programming

LISTA DE ILUSTRAÇÕES

Figura 1: Modelo de representação do Problema do caixeiro viajante em forma de grafo..... 16
Figura 2: Modelo de representação do Problema do caixeiro viajante em forma de grafo com as distâncias entre as cidades 16
Figura 3: Grafo representando PCV de 4 vértices. 18
Figura 4: Grafo de um PCV com 18 cidades. 20
Figura 5: Gráfico do aumento de arestas com o aumento do número de vértices. 20
Figura 6: Modelo de representação do Problema do caixeiro viajante em forma de grafo com as distâncias entre as cidades 23
Figura 7: Interface do sistema de criação de grafos..... 25
Figura 8: Interface do sistema de criação de grafos na escolha das distâncias (custo) entre as cidades (vértices). 25
Figura 9: Interface do sistema de criação de grafos..... 26
Figura 10: Representação do espaço de soluções 30
Figura 11: Algoritmo da Meta-heurística Busca Tabu..... 31
Figura 12: Janela principal do MATLAB®..... 34
Figura 13: Início do programa PCV, indicando o número de cidades a ser percorrida. 35
Figura 14: Indicando ao programa número de iterações proibidas. 35
Figura 15: Indicando o número de transições. 36
Figura 16: Resultado do PCV 37
Figura 17: Ambiente do *software* OCTAVE 38
Figura 18: Inserindo os mesmos parâmetros no programa executado no OCTAVE..... 38
Figura 19: Resultado do PCV no Octave..... 39
Figura 20: Região de acordo com o máximo de alqueires..... 42
Figura 21: Região da função de acordo com a quantidade de água..... 43
Figura 22: Região da função de acordo com a quantidade de fertilizantes..... 43
Figura 23: Região de viabilidade ou espaço de busca 44
Figura 24: Solução inicial para $x = 4$ e $y = 0$..... 45
Figura 25: Solução para $x = 4$ e $y = 1$..... 46
Figura 26: Melhor solução..... 47

LISTA DE QUADROS

Quadro 1: Possíveis caminhos para um problema simétrico com $n = 4$ 19
Quadro 2: Quantidade de caminhos possíveis de acordo com o número de cidades 21
Quadro 3: Resultado do PCV através de escolha de caminho aleatória 23
Quadro 4: Resulta do PCV com Método do Caminho Mais Curto 24
Quadro 5: Busca Tabu 32
Quadro 6: Resultado das 20 simulações 40

LISTA DE ABREVIATURAS

BNCC – BASE NACIONAL CURRICULAR COMUM

PCV – PROBLEMA DO CAIXEIRO VIAJANTE

SO – SISTEMA OPERACIONAL

AG – ALGORITMO GENÉRICO

TSP – TABU SEARCH PROBLEM

MD – MATRIZ DE DISTÂNCIA

SUMÁRIO

Sumário

1. Introdução 10
2. Fundamentação Teórica 12
2.1 Otimização Combinatória e sua importância para a resolução de situações- problemas. 14
2.2 O Problema do Caixeiro Viajante 15
2.3 A Meta-heurística Busca Tabu (BT) 26
2.4 O MATLAB® 32
2.5 Exemplos de solução do PCV usando o algoritmo através da BT 37
2.6 Programação Matemática 40
3. Considerações finais 49
4. Referências Bibliográficas 51
APÊNDICE 52

1. Introdução

O mundo caminha a passos largos em direção a informatização. Processos, tecnologias, atividades, ações cada dia mais autônomas. O adulto do futuro deve ser preparado para estar mais confortável com os desafios dessa informatização.

Nesse momento, onde o contato com diferentes tecnologias se torna cada vez mais importante, o computador possui papel fundamental. Porém, um computador sozinho nada mais é que uma caixa sem utilidade. Eles precisam ser projetados, manipulados e manobrados para auxiliar e resolver problemas. E se ensina um computador a resolver esses problemas, especificando-o, delimitando-o, ensinando-o a analisar alternativas. Essa estruturação de um problema, a fim de ensinar o computador a resolvê-lo é o que se chama de algoritmo. Assim, apresenta-se a seguinte questão: como utilizar a meta-heurística Busca Tabu (BT) e o *software MATLAB®* para resolver o Problema do Caixeiro Viajante (PCV)?

A tarefa da programação é pautada no raciocínio lógico, na criatividade e no conhecimento de ferramentas adequados aos propósitos que se deseja cumprir (CHAIA, 2013). Isto é, exige do programador um profundo conhecimento do problema que se pretende resolver.

O objetivo neste trabalho é resolver o Problema do Caixeiro Viajante (PVC) utilizando a meta-heurística Busca Tabu, no ambiente de programação do *MATLAB®*. Além disso, trazer ou iniciar reflexões sobre a possibilidade de introduzir nas aulas do Ensino Médio, situações-problemas, na perspectiva da otimização e da programação computacional através das aulas de matemática. O PCV, é um exemplo de problema que pode ajudar nesse caminho, assim como outros problemas de otimização (GOLDBARG, LUNA, 2005), talvez de resolução mais simples pois, a partir deles se tem uma ideia da importância da Programação Computacional no auxílio à resolução de modelos matemáticos ou situações problemas importantes, usando modelagem matemática em todas as áreas do conhecimento humano.

É importante ressaltar que com o uso de novas tecnologias como *smartfones*, *tablets* e outros equipamentos portáteis, a inclusão de problemas que envolvam *softwares* como o GeoGebra, o *MATLAB®* e outros no currículo do ensino médio nas aulas de matemática poderá despertar no aluno maior interesse pela disciplina e oferecer ao professor uma possibilidade a mais de diversificar os métodos usados para ensinar matemática.

Contato prévio com o *MATLAB®*, fez-nos escolhê-lo como ferramenta para implementação do algoritmo de resolução do PCV. Em outras experiências, inclusive no

Trabalho de Conclusão de Curso da minha graduação, o utilizei como base do projeto. O *MATLAB®* é uma ferramenta poderosa e muito utilizada por cientistas e engenheiros para a solução e modelagem de problemas das mais diversas áreas. A meta-heurística Busca Tabu é um método que permite a análise de possíveis respostas e suas vizinhanças com o intuito de se encontrar uma "boa" resposta com um custo (tempo, custo computacional, etc.) aceitável. O PCV é um problema de Otimização que pode ser associado a diversas ações do cotidiano dos alunos e, por sua vez, ser de simples entendimento, porém, sua solução pode ser complexa. A meta-heurística Busca Tabu é, neste trabalho, a metodologia utilizada para resolução do PCV.

O trabalho está disposto em capítulos, de forma que no Capítulo 2 a teoria será fundamentada explicando o que é o PCV. Na seção 2.3 será discutido a meta-heurística BT. Além disso, o *MATLAB®*, programa escolhido para ser usado como ferramenta nas simulações e soluções do problema proposto será apresentado na seção 2.4. Será apresentado um pouco do histórico do programa e comentários a respeito de aplicações já desenvolvidas através dele, vantagens e desvantagens na sua utilização. Além disso, será resolvido um exemplo de otimização auxiliado por programação matemática de menor complexidade utilizando BT. No terceiro capítulo, são apresentadas as considerações finais e a discussão dos resultados.

2. Fundamentação Teórica

Os processos educacionais passam pela discussão sobre o ensino de algoritmos. De acordo com a BNCC - Base Nacional Curricular Comum (BRASIL, 2018)[1].

> II – matemática e suas tecnologias: aprofundamento de conhecimentos estruturantes para aplicação de diferentes conceitos matemáticos em contextos sociais e de trabalho, estruturando arranjos curriculares que permitam estudos em resolução de problemas e análises complexas, funcionais e não-lineares, análise de dados estatísticos e probabilidade, geometria e topologia, robótica, automação, inteligência artificial, programação, jogos digitais, sistemas dinâmicos, dentre outros, considerando o contexto local e as possibilidades de oferta pelos sistemas de ensino;.

Assim, podemos constatar que em matemática e suas tecnologias estão inseridos termos como robótica, automação, inteligência artificial, programação e jogos digitais, que requer do estudante mais do que conhecimento isolado dos conceitos, propriedades e fórmulas matemáticas, mas, exige o desenvolvimento do raciocínio e conhecimento lógico dedutivo que permitirá ao aluno e futuro profissional, seja em qualquer área do conhecimento, maiores chances de ser um ator bem sucedido no desenvolvimento de novas tecnologias, *softwares* e outros. Notemos que essas tecnologias estão permeadas em todos os setores produtivos e de serviços, assim é fundamental a capacitação, e porque não começar dentro da sala de aula.

A BNCC afirma também que (BRASIL, 2018)[2]:

> AS TECNOLOGIAS DIGITAIS E A COMPUTAÇÃO
>
> A contemporaneidade é fortemente marcada pelo desenvolvimento tecnológico. Tanto a computação quanto as tecnologias digitais de informação e comunicação (TDIC) estão cada vez mais presentes na vida de todos, não somente nos escritórios ou nas escolas, mas nos nossos bolsos, nas cozinhas, nos automóveis, nas roupas e etc. Além disso, grande parte das informações produzidas pela humanidade está armazenada digitalmente. Isso denota o quanto o mundo produtivo e o cotidiano estão sendo movidos por tecnologias digitais, situação que tende a se acentuar fortemente no futuro.
>
> Essa constante transformação ocasionada pelas tecnologias, bem como sua repercussão na forma como as pessoas se comunicam, impacta diretamente

[1] Disponível em: <http://basenacionalcomum.mec.gov.br/abase/#medio>
[2] Disponível em: <http://basenacionalcomum.mec.gov.br/abase/#medio>

> no funcionamento da sociedade e, portanto, no mundo do trabalho. A dinamicidade e a fluidez das relações sociais – seja em nível interpessoal, seja em nível planetário – têm impactos na formação das novas gerações. É preciso garantir aos jovens, aprendizagens para atuar em uma sociedade em constante mudança, prepará-los para profissões que ainda não existem, para usar tecnologias que ainda não foram inventadas e para resolver problemas que ainda não conhecemos. Certamente, grande parte das futuras profissões envolverá, direta ou indiretamente, computação e tecnologias digitais.
>
> Diferentes dimensões que caracterizam a computação e as tecnologias digitais são tematizadas, tanto no que diz respeito a conhecimentos e habilidades quanto a atitudes e valores:

Talvez seja o momento de se perguntar, o que pode ser feito para, além de ensinar matemática, torna-la útil e interessante, dar a ela o protagonismo de desenvolver no aluno as primeiras noções de programação e funcionamentos de *softwares*? Talvez seja hora de refletir sobre o tipo de matemática que se pretende ter nas aulas de matemática do ensino médio.

> pensamento computacional: envolve as capacidades de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções, de forma metódica e sistemática, por meio do desenvolvimento de algoritmos;

Assim o ensino e aprendizagem de matemática no Ensino Médio é um desafio para os professores e alunos. Para os professores, porque necessitam de constante aprendizado e atualização em relação às tecnologias e a preparação dos alunos a elas. Aos alunos, por terem de se conscientizar da necessidade de conhecimento específico no ambiente da informatização e familiarização com a criação de algoritmos, e como próximo passo, *softwares* para solução de problemas.

Neste capítulo será apresentado o PCV, aspectos históricos e estratégias de solução. Serão apresentadas algumas meta-heurísticas e, em especial, a Busca Tabu que será a estratégia utilizada para solução do PCV. O PCV será resolvido no ambiente de programação *MATLAB®*, que será abordado também neste capítulo. O *GNU-OCTAVE* é um *software* gratuito similar ao *MATLAB®*, este será testado com o mesmo programa solução e comparado ao primeiro como alternativa de baixo custo à esta simulação.

2.1 Otimização Combinatória e sua importância para a resolução de situações-problemas.

De acordo com Taveira (2005), a otimização é responsável pelo processo de tomada de decisão. Cabendo a quem tem essa função decidir, identificar e definir o problema, formular objetivos, analisar limitações, avaliar as alternativas e por fim, escolher a melhor decisão a ser tomada.

Neste tipo de problemas não é possível ter uma solução qualquer e sim uma que atenda as restrições e otimize os objetivos propostos, otimizar tempo, diminuir percursos, melhorar custos, escolher rotas entre outros.

A Base Nacional Comum Curricular - BNCC, em suas competências, explicita o seguinte:

> "o Ensino Médio deve atender às necessidades de formação geral indispensáveis ao exercício da cidadania e construir aprendizagens sintonizadas com as necessidades, as possibilidades e os interesses dos estudantes e, também, com os desafios da sociedade contemporânea" (BRASIL, 2018).

Como definido na Introdução da BNCC (p. 14; ênfases adicionadas) deixa claro que a nova vertente da educação deve deixar a escola mais interessante ao aluno. Os conteúdos a serem tratados na escola devem ser abordados, elaborados e associados, sempre que possível, a um contexto real. A evolução digital é uma realidade desta nova geração e este fato deve ser considerado para justificar a abordagem da programação computacional no Ensino Médio e mostrar a importância da matemática nesse processo.

A otimização matemática, em se tratando de uma ferramenta para a tomada de decisão, pode ser aplicada na modelagem de grande parte das situações cotidianas, por exemplo: qual o melhor caminho que o aluno pode fazer de sua casa até a escola. Em especial, o interesse desse trabalho, é que esse tipo de situação possa ser explorado em sala de aula, buscando soluções através de artifícios tecnológicos, como planilhas ou até na confecção de *softwares,* aplicativos para *smartphones*, jogos e etc.

Problemas de otimização combinatória podem ser de minimização ou de maximização de uma determinada Função Objetivo, sujeita a algumas restrições. Em ambas as situações, tem-se uma função aplicada a um domínio finito (GOLDBARG; LUNA, 2005). Apesar de ser finito, o domínio da função pode ser enorme e alguns algoritmos menos complexos não

conseguem realizar a tarefa. Desta maneira, surge a necessidade de usar técnicas mais elaboradas para encontrar soluções de valor ótimo, que pode ser de valor mínimo ou máximo, caso o problema seja de minimização ou maximização.

2.2 O Problema do Caixeiro Viajante

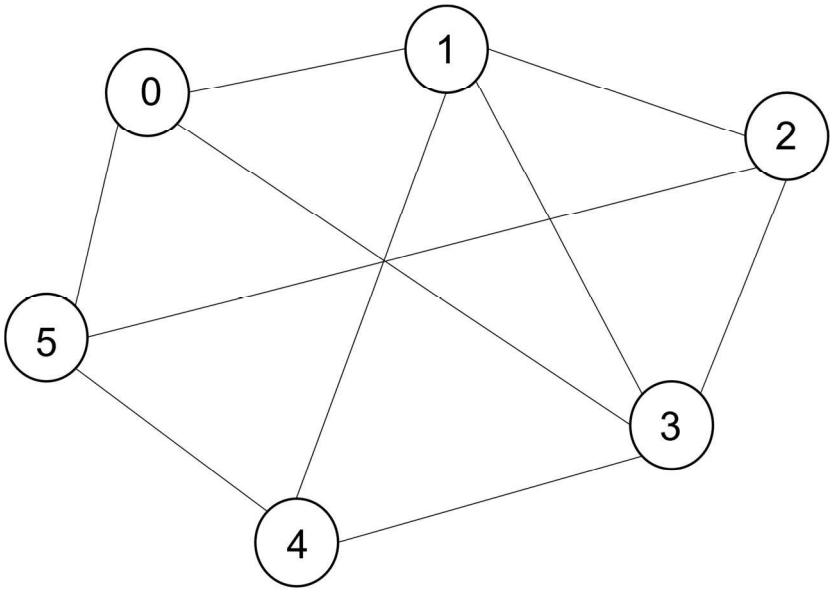
O PCV é um tradicional e conhecidos problema de Otimização. O PCV se dá da seguinte forma: Um caixeiro viajante deve visitar *n* cidades e retornar à cidade de onde partiu, sendo que, com exceção da cidade inicial, cada cidade deve ser visitada uma única vez. O objetivo é minimizar o custo dessa viagem percorrendo a menor rota. Ou seja, o problema consiste na procura de um percurso que possua a menor distância, começando numa cidade qualquer, e visitando cada cidade uma única vez e retornando à cidade inicial. Deve se levar em consideração que não importa a ordem.

A escolha desse problema se baseia na facilidade de associá-lo a situações cotidianas, assim fica mais confortável para o aluno ambientar-se e familiarizar-se através de situações problemas que tem os mesmos aspectos de PCV. Por exemplo na escolha dele na melhor rota para ir para escola. Outro problema que pode ser inspirado no PCV é o Problema de Roteamento de Veículos (PRV). Além disso, a resolução do problema exigirá do aluno conhecimento de alguns assuntos, vistos no Ensino fundamental e médio, como por exemplo: Conjuntos numéricos, funções, geometria, desenho geométrico, matrizes, análise combinatória e interpretação de textos, assuntos esses encontrados nos textos da BNCC.

De forma que para entender e desenvolver a programação no *MATLAB®*, um longo caminho deverá ser percorrido primeiro pela necessidade de entendimento dos temas citados acima, depois do estudo mais aprofundado sobre algoritmos, o estudo sobre a linguagem em si e como transformar o algoritmo planejado em um programa no *MATLAB®* e a avaliação das respostas.

Uma forma simples de representação desse problema está em desenhá-lo na forma de grafo, onde as cidades a serem visitadas são os vértices e os caminhos são as arestas entre os vértices.

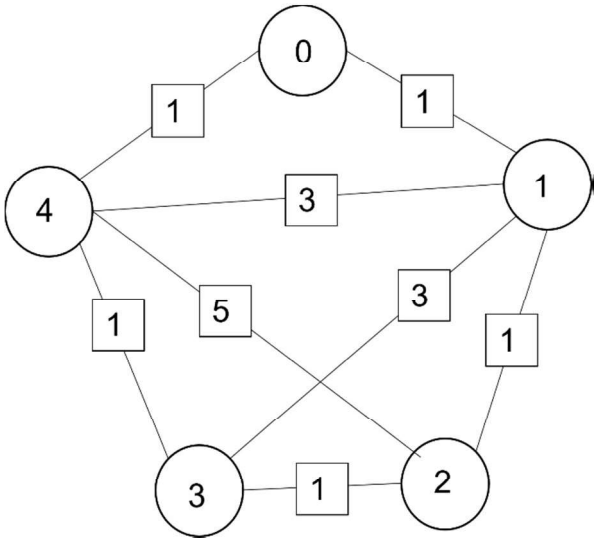
Figura 1: Modelo de representação do PCV em forma de grafo



Fonte: Criado pelo autor

A Figura 1 mostra o PCV em forma de grafo onde cada aresta possui um valor, que representa a distância (ou qualquer outro custo estabelecido no problema) entre as duas cidades situadas nos extremos da aresta.

Figura 2: Modelo de representação do PCV em forma de grafo com as distâncias entre as cidades



Fonte: Criado pelo autor

Assim para padronização do entendimento dos termos utilizados neste trabalho, é importante definir os termos aresta, caminho, vértice. Os vértices serão as cidades às quais o caixeiro deve visitar em sua jornada. Aresta será a distância (ou qualquer outro tipo de custo apresentado pelo problema) entre os vértices (cidades) e o caminho será a solução do problema, será o resultado final do custo do percurso ou jornada do caixeiro viajante.

O problema então se apresenta nos valores das arestas entre as cidades que representam uma distância, tempo, ou qualquer outro tipo de "custo". Respeitando essas distâncias, a intenção então será a busca pelo melhor, ou menor, caminho a ser percorrido passando por todas as cidades uma única vez.

Sendo i a cidade de origem e j a cidade de destino, tem-se que a_{ij} é a distância (ou custo) entre estas cidades. Ainda, a distância entre as cidades 1 e 2 será a mesma distância entre as cidades 2 e 1, assim $a_{ij} = a_{ji}$. Sabendo também que a distância entre uma cidade e ela mesma é 0 (zero), ou seja, $a_{ii} = a_{jj} = 0$. É possível criar a matriz de distância (MD) de um PCV a partir de um grafo como o da Figura 2. Considere o problema com $n \in \mathbb{N}$ cidades.

$$\text{MD} = \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & 0 & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & 0 & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \vdots & 0 \end{bmatrix}$$

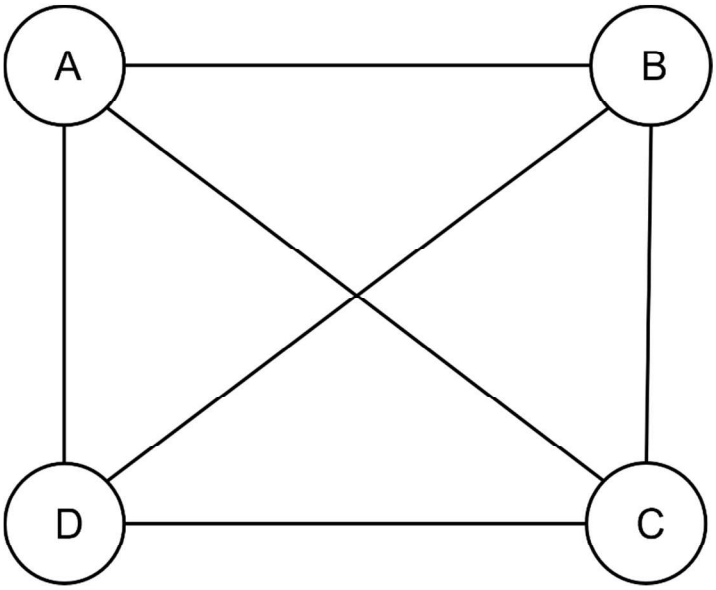
A MD é uma matriz quadrada $n \times n$, dado que a quantidade de elementos acima da diagonal principal e abaixo dela são iguais e representam distâncias entre duas cidades i e j, variando de 1 a n. Os elementos da diagonal principal serão todos iguais a zero pelo motivo já explicado anteriormente, ou seja, a MD será uma matriz simétrica. Essa matriz pode ser utilizada com artifício para solução do PCV, além de ser mais um conceito interessante a se trabalhar em sala de aula a partir deste problema.

A importância e grande interesse pelo PCV pode ser justificada pela grande quantidade de aplicações práticas e pela complexidade em encontrar uma solução de boa qualidade ou ótima. Em (GOLDBARG; LUNA, 2005) são destacadas algumas das aplicações práticas do PCV. Dentre elas é possível citar: solução de problemas de roteamento de veículos, otimização de perfurações em placas de circuitos impressos, roteamento de entrega postal e a solução de problemas de sequenciamento de tarefas. Existem outras formas de ilustrar esse tipo de problema e associá-lo a situações, por exemplo, por rotas de ônibus escolar. Este ônibus deve percorrer um caminho (o melhor possível) para coletar todos os alunos distribuídos em

diferentes pontos da cidade. Outro bom exemplo, e cada vez mais real nos últimos tempos, é a entrega de refeições e serviços de *delivery*, da mesma forma o entregador deve escolher o melhor trecho para fazer suas entregas visitando cada cliente uma única vez.

O PCV aumenta sua complexidade com o aumento de vértices ou de pontos a serem visitados (cidades). Tomando como exemplo um problema desse com $n = 4$ cidades, *A*, *B*, *C* e *D*. Considerando que o caixeiro deve sair da cidade *A* e retornar a ela após passar por todas as outras cidades apenas uma vez.

Figura 3: Grafo representando PCV de 4 vértices.



Fonte: Criado pelo autor

Assim, uma solução do problema é um conjunto com as quatro cidades, com uma ordem de visitas estabelecido, a melhor solução ou solução ótima é aquela que determina a menor distância a ser percorrida. No Quadro 1 abaixo estão os possíveis caminhos a serem percorridos pelo Caixeiro.

Quadro 1: Possíveis caminhos para um problema simétrico com $n = 4$.

N°	CAMINHO
1	ABCDA
2	ADCBA
3	ACBDA
4	ADBCA
5	ABDCA
6	ACDBA

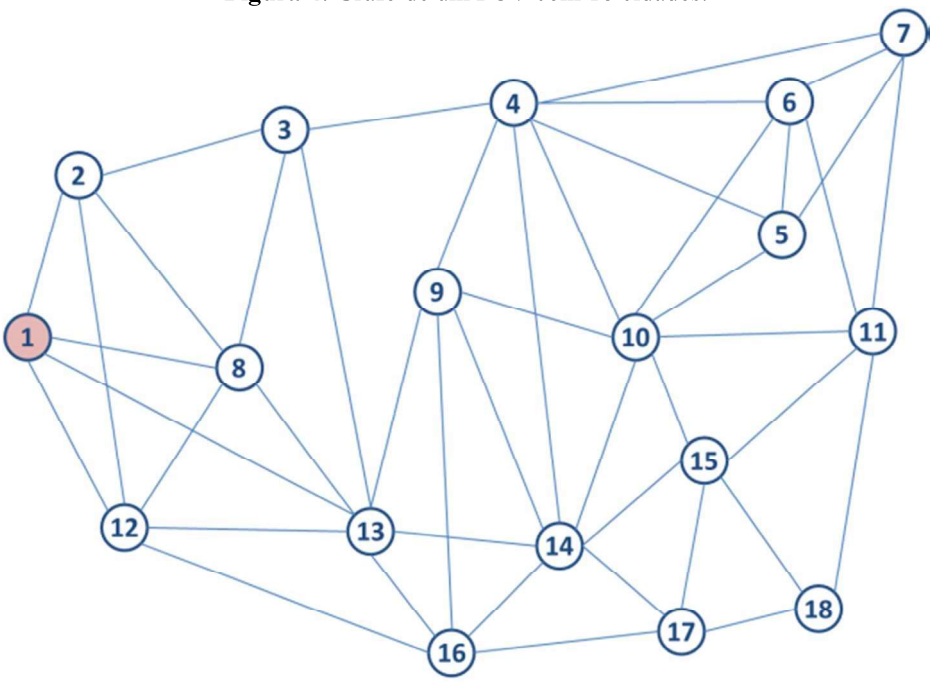
Fonte: Criado pelo autor

Como é possível perceber, os caminhos são similares 2 a 2. O caminho 1 e o caminho 2, o caminho 3 e o caminho 4, o caminho 5 e o caminho 6, são iguais percorridos no sentido contrário um do outro. Podemos conjecturar então que há num total de 3 (três) caminhos distintos em relação aos custos ou distâncias.

A partir da Figura 4 pode-se constatar que cada vértice não tem conexão com todos os outros diretamente. É possível deduzir também que um grafo de n vértices pode ter até $\frac{n(n-1)}{2}$ arestas. No caso de ter exatamente $\frac{n(n-1)}{2}$ arestas, todos os vértices têm ligações entre si diretamente.

À medida que o valor de "*n*" cresce, ou seja, aumenta o número de cidades (vértices) o problema passa a ficar mais complexo pois, o número de arestas cresce de acordo com a fórmula acima, enquanto o número de caminhos possíveis cresce em proporções fatoriais em relação ao número de vértices.

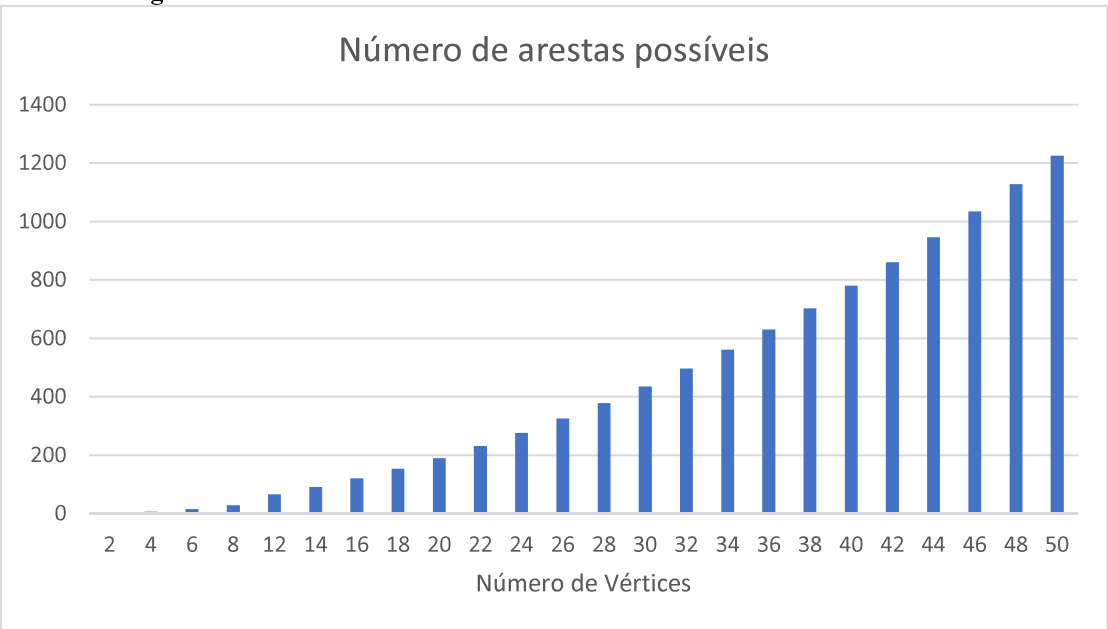
Figura 4: Grafo de um PCV com 18 cidades.



Fonte: Extraído do site https://otimizacaonapratica.com/2015/11/09/o-problema-do-caixeiro-viajante/

A Figura 5 ilustra o aumento da complexidade da solução do PCV com o aumento da quantidade de vértices. Se torna inviável fazer uma lista, como mostrado anteriormente, e assim resolver esse problema de forma a analisar todas as possíveis respostas na função objetivo. O número de possibilidades de caminhos aumenta de forma considerável.

Figura 5: Gráfico do aumento de arestas com o aumento do número de vértices.



Fonte: Criado pelo autor

Assim, o Quadro 2 mostra o número de caminhos de acordo com o número de cidades. É possível perceber que é impossível resolver o PCV com um número grande de cidades por exaustão, ou seja, testado todos os possíveis caminhos na função objetivo. Nesse quadro, o número de caminhos para 10, 15 e 20 cidades são aproximados. Na verdade, para n cidades, o número de caminhos possíveis é dado por $(n-1)!$.

Quadro 2: Quantidade de caminhos possíveis de acordo com o número de cidades

Número de cidades	Caminhos possíveis
4	6
5	24
10	$3{,}63x10^5$
15	$8{,}72x10^{10}$
20	$1{,}22x10^{17}$

Fonte: Criado pelo autor

Outra maneira de analisar a impossibilidade de verificação de todos os caminhos para um problema com muitas cidades, está no seguinte raciocínio: supõe-se que para 10 cidades o computador leve 8 segundos para verificar todos os caminhos, como são 9! de caminhos, para 11 cidades são $10! = 10 \times 9!$ De caminhos, ou 80 segundos. Continuando com esse raciocínio teremos os seguintes valores aproximados, para 12 cidades 15 minutos, para 13 cidades 3 horas, para 14 cidades 1,5 dias, para 15 cidades 3 semanas, para 16 cidades 11 meses, para 17 cidades 15 anos e para 18 cidades 2,5 séculos.

Esse é um típico problema de Otimização Combinatória, isto é, problema em que é preciso combinar as várias possibilidades de formação das soluções apresentadas em seu conjunto ou espaço de busca. O termo "Otimização" em matemática, se refere ao estudo de problemas no qual a intenção destes é minimizar ou maximizar uma função através da escolha de valores dentro de um determinado conjunto e esta função a ser otimizada é chamada de Função Objetivo (FO). Assim, em um problema de otimização temos uma função objetivo e nele pode haver ou não um conjunto de restrições. Os valores possíveis às variáveis de decisão são delimitados pelas restrições impostas (quando existirem) sobre essas variáveis, formando um conjunto de soluções factíveis a um problema. O problema pode ser de minimização, como o caso do Problema do Caixeiro Viajante (PCV) ou de maximização da função objetivo. A

resposta para o problema de otimização, será o menor (ou maior) valor possível para a função objetivo.

O PCV pode ser formulado da seguinte maneira. Seja a matriz $D = \left(d_{ij}\right)_{n\times n}$ e o conjunto P de permutações dos inteiros de 1 a n. Deseja-se encontrar uma permutação $p = \left(\pi(1), \pi(2), \cdots, \pi(n)\right) \in P$ que minimize

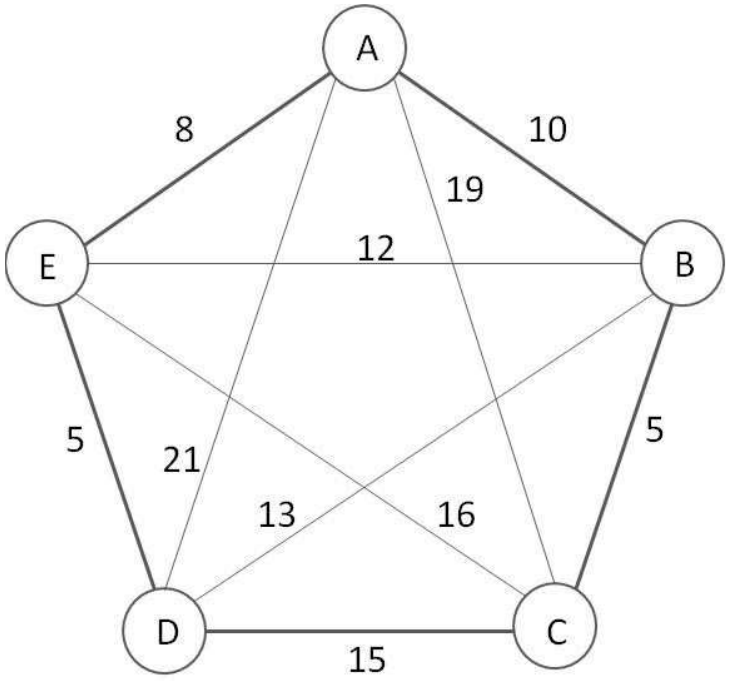
$$Z(p) = \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)}$$

Onde n é o número de cidades; Z é a Função Objetivo; D é a matriz de distâncias entre as cidades; $j = \pi(i)$ é a visita à cidade j na etapa i. As permutações são os caminhos ou possíveis soluções e os pares $\left(\pi(1), \pi(2)\right), \cdots, \left(\pi(i), \pi(i+1)\right), \cdots, \left(\pi(n), \pi(1)\right)$ são as arestas ou ligação entre duas cidades. Já o número positivo, $d_{\pi(i),\pi(i+1)}$ é a distância da cidade j onde o caixeiro chegou na etapa i, para a próxima cidade onde se completará a etapa $(i+1)$ do caminho a ser percorrido. Então, resolver o PCV é encontrar o caminho mais curto em que cada cidade é visitada exatamente uma vez. Ou seja, encontrar o menor valor para Z.

A fim de comparação com outros métodos e como exemplo de solução será mostrado um exemplo de um PCV simples com resposta de forma aleatória.

Na Figura 6 está o exemplo a ser discutido. Os números em cada aresta representam as distâncias entre as cidades. Serão utilizados 2 (dois) métodos de resolução deste problema, o primeiro será escolher aleatoriamente o caminho e o outro será o método de escolher em cada passo a menor aresta, ou a menor distância entre duas cidades.

Figura 6: Modelo de representação do Problema do caixeiro viajante em forma de grafo com as distâncias entre as cidades



Fonte: Criado pelo autor

No primeiro método o Quadro 3 mostra como resultado que o custo da viagem foi de 63 (a unidade deve ser determinada na descrição do exemplo, pode ser 'km', 'R$', entre outras). Neste método a próxima cidade a ser visitada é escolhida aleatoriamente e então se computa o valor já atribuído a cada aresta.

Quadro 3: Resultado do PCV através de escolha de caminho aleatório

CAMINHO	--	AC	CE	ED	DB	BA	TOTAL
CIDADE	A	C	E	D	B	A	
CUSTO	0	19	16	5	13	10	63

Fonte: Criado pelo autor

Em um teste simples para comparação, foi utilizado o Método do Caminho mais curto que se baseia em escolher a próxima cidade aquele que possui menor aresta em suas adjacências, e a cada novo vértice (cidade) realizando o mesmo processo sem retornar ao vértice anterior. O Quadro 4 mostra a resposta e uma significativa melhora no custo da viagem.

Quadro 4: Resulta do PCV com Método do Caminho Mais Curto

CAMINHO	--	AE	ED	DB	BA	CA	TOTAL
CIDADE	A	E	D	B	C	A	
CUSTO	0	8	5	13	5	19	50

Fonte: Criado pelo autor

Percebe-se uma melhoria no resultado com a escolha de um método simples e sem grande refinamento. O custo da viagem reduziu de 63 para 50 apenas considerando uma estratégia básica para solução do problema.

Existem alguns ambientes que permitem a criação dos Grafos apresentados acima na Figura 6. Bem como, a solução do problema através de algum algoritmo disponível para seleção. Por exemplo, no site https://graphonline.ru/pt/ é possível inserir diversas cidades ou vértices, também é possível determinar o peso ou a distância entre as cidades e o método que deseja utilizar para solução do problema. Algo que pode ser usada como uma ferramenta pedagógica para estimular os alunos nas aulas de matemática.

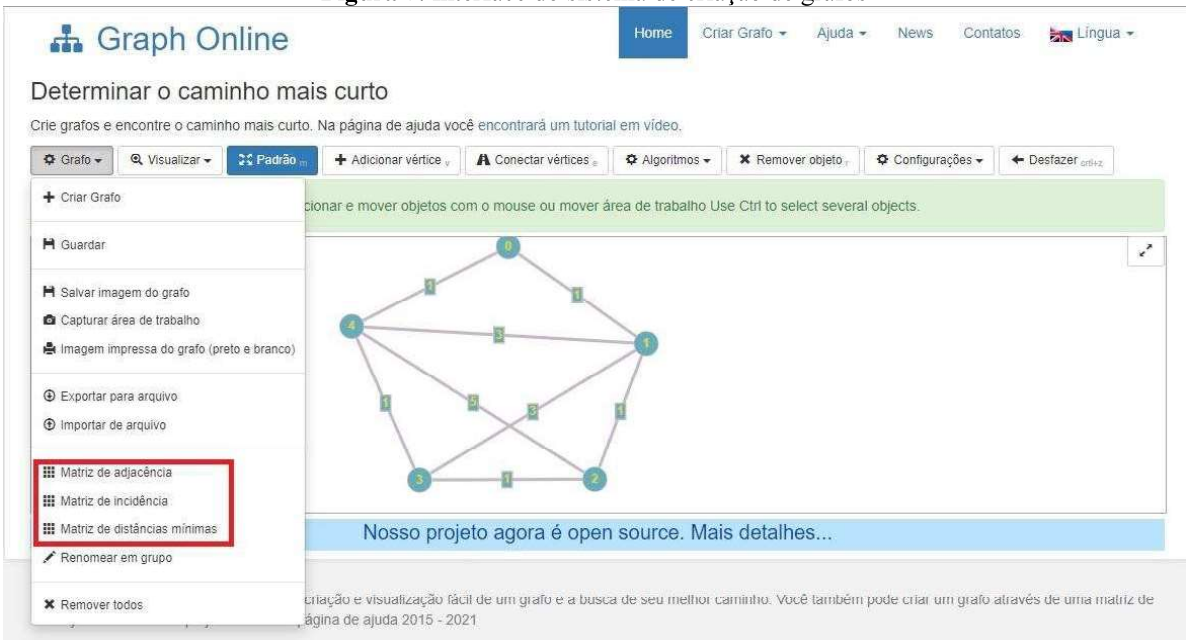
Daí se valer dos conceitos e problemas de otimização combinatória, que podem utilizar este tipo de sistema para atividades em classe. Por ser um site pode facilmente ser acessado em sala através de *smartphones* dando possibilidades à resolução de atividades utilizando dessa ferramenta. Esse tipo de atividade pode ser algo que chame a atenção do aluno, que cause interesse nele. A utilização desse tipo de tecnologia como recurso pela escola e professor para desenvolver trabalho pode tonar alguns conteúdos mais amigáveis e interessantes para os alunos.

A Figura 7 ilustra a possibilidade do tratamento das informações dos grafos através da geração de matrizes de Adjacências[3], Matriz de Incidência[4] e Matriz de distância mínima. Essas matrizes auxiliarão na solução e entendimento do problema, a depender do método escolhido para solução.

[3] Matriz de Adjacências: É uma forma de representar um grafo associando seus vértices e os vértices adjacentes a eles. Se há alguma aresta interligando dois vértices i, j então na matriz recebe valor 1, caso contrário, 0.

[4] Matriz de Incidência: Forma de representar um grafo relacionando seus vértices e as arestas adjacentes. Em uma matriz onde as linhas são os vértices e as colunas são as arestas, todas as arestas que partem de um determinado vértice no grafo recebe o valor 1 na matriz, as arestas não que são adjacentes aos vértices recebem o valor 0.

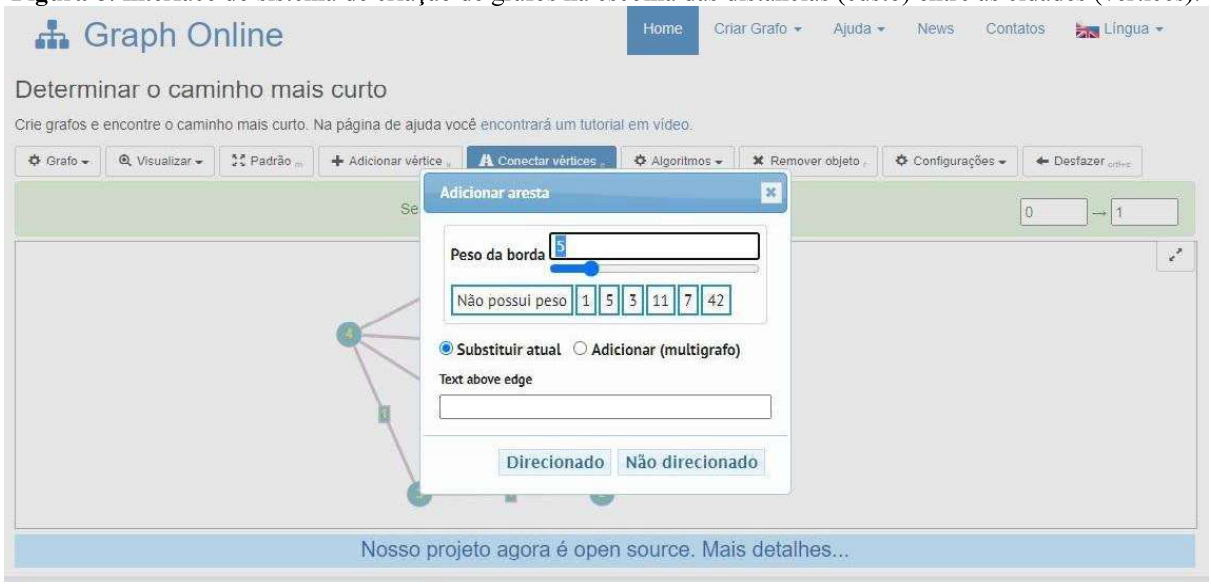
Figura 7: Interface do sistema de criação de grafos



Fonte: Extraído do site https://graphonline.ru/pt/

A Figura 8 mostra o local em que o site permite que se determine o custo (peso, distância etc.) entre os vértices. Esta informação é fundamental para o desenvolvimento do problema que se deseja resolver.

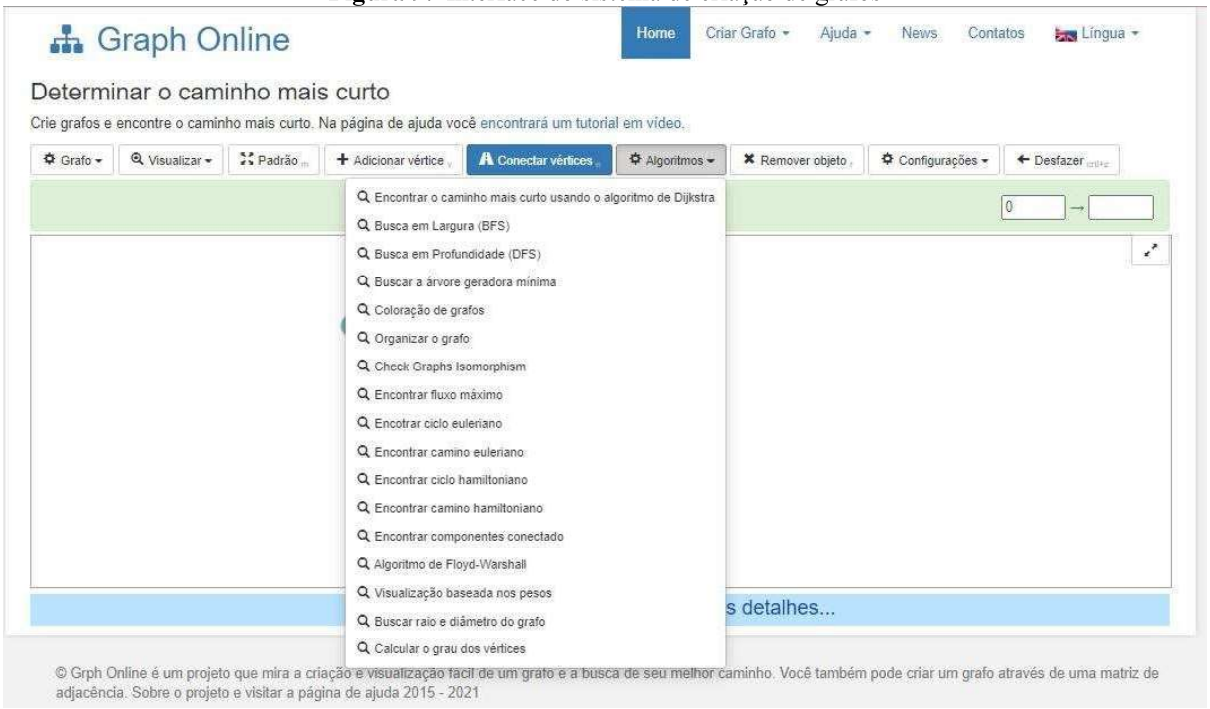
Figura 8: Interface do sistema de criação de grafos na escolha das distâncias (custo) entre as cidades (vértices).



Fonte: Extraído do site https://graphonline.ru/pt/

Na Figura 9 vê-se uma lista de algoritmos e possibilidades de soluções para o problema, com literatura disponível para pesquisa e que podem ser escolhidos para determinar o possível melhor caminho para o PCV, entre outros resultados.

Figura 9: Interface do sistema de criação de grafos



Fonte: Extraído do site https://graphonline.ru/pt/

2.3 A Meta-heurística Busca Tabu (BT)

Como foi mostrado acima, o PCV é de fácil entendimento, mas difícil de resolver. Existem várias formas de se abordar e resolver um problema, dentre elas algumas podem ser citadas:

- tentativa e erro;
- pensar na solução de um problema semelhante;
- desenhar uma tabela ou diagrama;
- procurar um modelo;
- escrever uma equação ou operação;
- trabalhar o problema de trás para frente;

Esses "caminhos" são conhecidos como HEURÍSTICAS. Heurísticas são regras, sugestões, guias ou técnicas que podem ser úteis em fazer progresso na direção da solução do problema (GONÇALVES, 2006). Heurística era o nome de um ramo do estudo da Lógica, Filosofia ou Psicologia que estuda os métodos e regras do descobrimento e da invenção.

POLYA (2000) denomina heurística moderna como o estudo que procura compreender o processo solucionador de problemas.

A palavra “Heurística” no dicionário tem as seguintes definições:

1. [História] Ramo da História que se dedica à pesquisa de documentos que tem por objeto a descoberta de fatos.
2. Método educacional que busca ensinar o aluno autonomamente, para que ele descubra e aprenda tendo em conta a sua experiência, com os próprios erros e acertos.
3. [Informática] Método investigativo e de pesquisa que se pauta na aproximação, através da quantificação, de um determinado objeto.

De acordo com Dante (2008),

> “Aprender a resolver problemas matemáticos deve ser o maior objetivo da instrução matemática. Certamente outros objetivos da Matemática devem ser procurados, mesmo para atingir o objetivo da competência em resolução de problemas. Desenvolver conceitos matemáticos, princípios e algoritmos através de um conhecimento significativo e habilidoso é importante. Mas o significado principal de aprender tais conteúdos matemáticos é ser capaz de usá-los na construção das soluções-problemas”. (apud DANTE, 2000, p.8).

Dito isso, o que deve ser considerado mais importante na resolução de um problema são os procedimentos e as estratégias escolhidas, mais que a solução em si.

Na maioria dos problemas propostos, nem sempre é possível encontrar a melhor solução em tempo razoável por meio de algoritmos exatos, o que como visto anteriormente é o caso do PCV. Nesses casos, uma solução relativamente boa, pode já ser suficiente para a aplicação que se tem em mãos. Os Métodos Heurísticos são algoritmos que não garantem encontrar a solução ótima de um problema, mas são capazes de retornar uma solução de boa qualidade em um tempo adequado para as necessidades da aplicação (GOLDBARG; LUNA, 2005).

Várias propostas de meta-heurísticas surgiram nos últimos anos impulsionadas por problemas classificados como difíceis. Dentre as meta-heurísticas mais conhecidas podemos destacar:

- **Algoritmos Genéticos**: constituem uma técnica de busca e otimização inspirada no princípio de seleção natural e reprodução genética. AG's possuem aplicações em diversas áreas, por exemplo: Síntese de circuitos analógicos; gerenciamento de redes; problemas de otimização complexos; além de modelar processos biológicos para estudo do comportamento de estruturas genéticas.
- **Colônia de Formigas**: se baseia no comportamento das formigas, que se movimentam de forma organizada em busca de um objetivo que pode ser encontrar comida ou encontrar o caminho de volta ao formigueiro. A ideia básica dessa meta-heurística é utilizar formigas artificiais, representadas por processos, que traçam caminhos em um grafo nos quais os vértices representam componentes da solução;
- **Recozimento Simulado *(Simulated Annealing)***: realiza suas iterações da seguinte maneira: sorteia-se uma solução da vizinhança da solução corrente atualiza-se a solução corrente com probabilidade, assim os movimentos que permitem aumento de qualidade são permitidos;
- **Partida Múltipla:** realizam diversas buscas, com diferentes soluções iniciais aleatórias, explorando regiões variadas do espaço de busca, aumentando a qualidade da melhor solução encontrada, bem como a probabilidade de que esta seja ótima;
- **Busca Tabu**: é uma meta-heurística de busca na qual a principal característica é a capacidade de exploração do histórico do processo de busca, organizado de forma a possuir uma memória adaptativa;

A Meta-heurística Busca Tabu é a estratégia escolhida para o desenvolvimento deste trabalho e resolver o PCV.

Tendo sua criação datada do fim da década de 60 e início da década de 70, a Busca Tabu, ou do inglês *Tabu Search*, foi proposta por Fred Glover em 1986 (GLOVER, 1986) , sendo utilizada nas mais diversas áreas de conhecimento. Porém, muitas explicações que de fato comprovassem a eficiência da Busca Tabu foram investigadas somente alguns anos depois, após a definição da sua forma de hoje (GOMES, 2009), permanecendo sem explicação por algum tempo o porquê da sua assertividade na busca de ótimas soluções.

A palavra "tabu" tem sua origem ligada ao *Tongan*, um idioma da Polinésia, que era utilizado pelos nativos da ilha *Tonga* para tratar objetos que não podiam ser tocados, por serem tidos como sagrados. Ainda, de acordo com o dicionário Aurélio da Língua Portuguesa, a palavra "tabu" significa "uma proibição imposta por costumes sociais como uma medida de proteção" ou de algo "banido por constituir risco", ou ainda "mantido distante pelo temor à

punição". Estes sentidos mais objetivos da palavra traduzem melhor a própria Busca Tabu: O risco a ser evitado neste caso é o de seguir um caminho não produtivo. Uma importante relação com o uso mais tradicional da palavra tabu vem do fato de que os ditos tabus são normalmente passados através da memória de ser a ser, e está sujeita a alterações com o passar do tempo. Isto cria uma conexão para o significado de "tabu" em Busca Tabu: os elementos tidos como momentaneamente proibidos da Busca Tabu recebem essa característica pela dependência em uma memória que evolui, e permite que esse momento de tabu altere de acordo com a circunstância do problema (GOMES, 2009).

A meta-heurística Busca Tabu é um procedimento adaptativo que guia um algoritmo de busca na exploração dentro de um espaço de busca. O cerne desse método está em que o algoritmo evita retornar a um ótimo local já visitado, de forma a atingir um resultado ótimo ou próximo ao ótimo (GOMES, 2009). Como dito anteriormente, é uma meta-heurística de busca na qual sua principal característica é a capacidade de explorar o histórico do processo de busca, organizado em estruturas como memória (memória adaptativa) (SUCUPIRA, 2004). Por exemplo, podem ser mantidas tabelas com os componentes que aparecem com maior frequência, e os componentes removidos ou adicionados à solução no passado recente. Este último tipo de tabela a, a ser chamada de lista tabu, é utilizada para classificar elementos como proibidos por um certo número de iterações, não permitindo que esses sejam removidos ou consideradas como solução (SUCUPIRA, 2004).

O método possui uma grande quantidade de aplicações a depender da complexidade que se deseja inserir nele, levando a Busca Tabu a ter uso importante em áreas como: métodos evolucionários e genéticos, melhoras de processos de redes neurais e aplicações práticas em campos variados como economia, física, transporte, dentre outros.

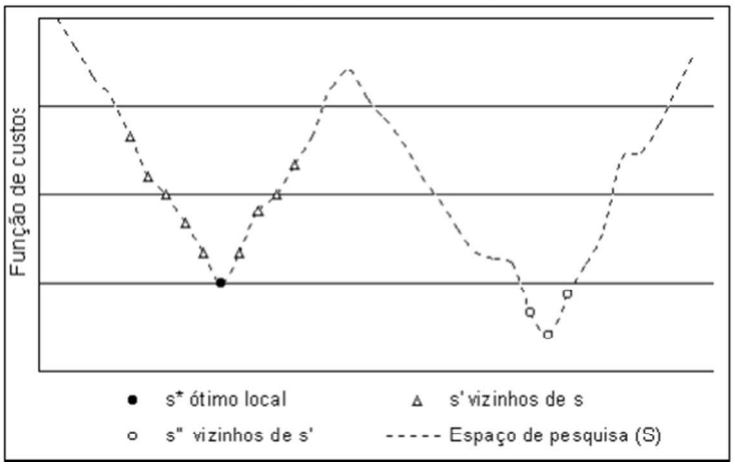
A Busca Tabu mantém uma lista de soluções candidatas (Lista Tabu) que já foram visitadas e se recusa a revisitá-las, até que um determinado período se passe. Ou seja, a partir de uma solução já obtida o algoritmo testa soluções vizinhas. Entende-se por vizinho de uma solução s a solução s' que sofreu uma modificação m. Esta equação pode ser representada da seguinte forma: $s' \leftarrow s + m$, sendo S o espaço de pesquisa de um problema de otimização. A função N, que depende da forma que o problema é apresentado, associa a cada candidata a solução $s \in S$, sua vizinhança $N(s) \subseteq S$, ou seja, todos as soluções s' geradas a partir de um movimento m. Cada solução $s \in N(s)$ recebe o nome de vizinho de s. De acordo com Schopf

et al (2004), a Busca Tabu utiliza memória para armazenar conhecimento sobre os espaços percorridos.

Algumas definições importantes para se trabalhar com meta-heurística

- **Espaço de busca**: é o espaço de todas as possíveis soluções que podem ser consideradas durante a busca;
- **Vizinhança**: Como já explicado anteriormente. As modificações que podem ser aplicadas na solução atual formam um conjunto de soluções vizinhas no espaço de busca. Cada espaço de busca considerado gera diferentes vizinhanças.
- **Tabus**: São candidatas a solução já visitadas que estão proibidas por um certo número de iterações.

Figura 10: Representação do espaço de soluções



Fonte: Brazil, 2006

Pode se observar na Figura 10 que s^* é um ótimo local. É possível perceber analisando seus vizinhos s', que todos eles têm valor maior. As soluções s'' mostram ter valores menores na função, mas não fazem parte da vizinhança de s^*.

Os principais critérios de parada do algoritmo de Busca Tabu são:

- **Número de iterações sem melhora**: o processo para após certo número de iterações sem melhoria;
- **Tempo de processamento**: o processo para após um tempo de processamento pré-determinado;
- **Limite a atingir**: o processo para quando um limite de iterações pré-definido é alcançado;

De acordo com Brazil, Leite e Mendes (2006) o código do algoritmo Busca Tabu pode ser descrito conforme Figura 11.

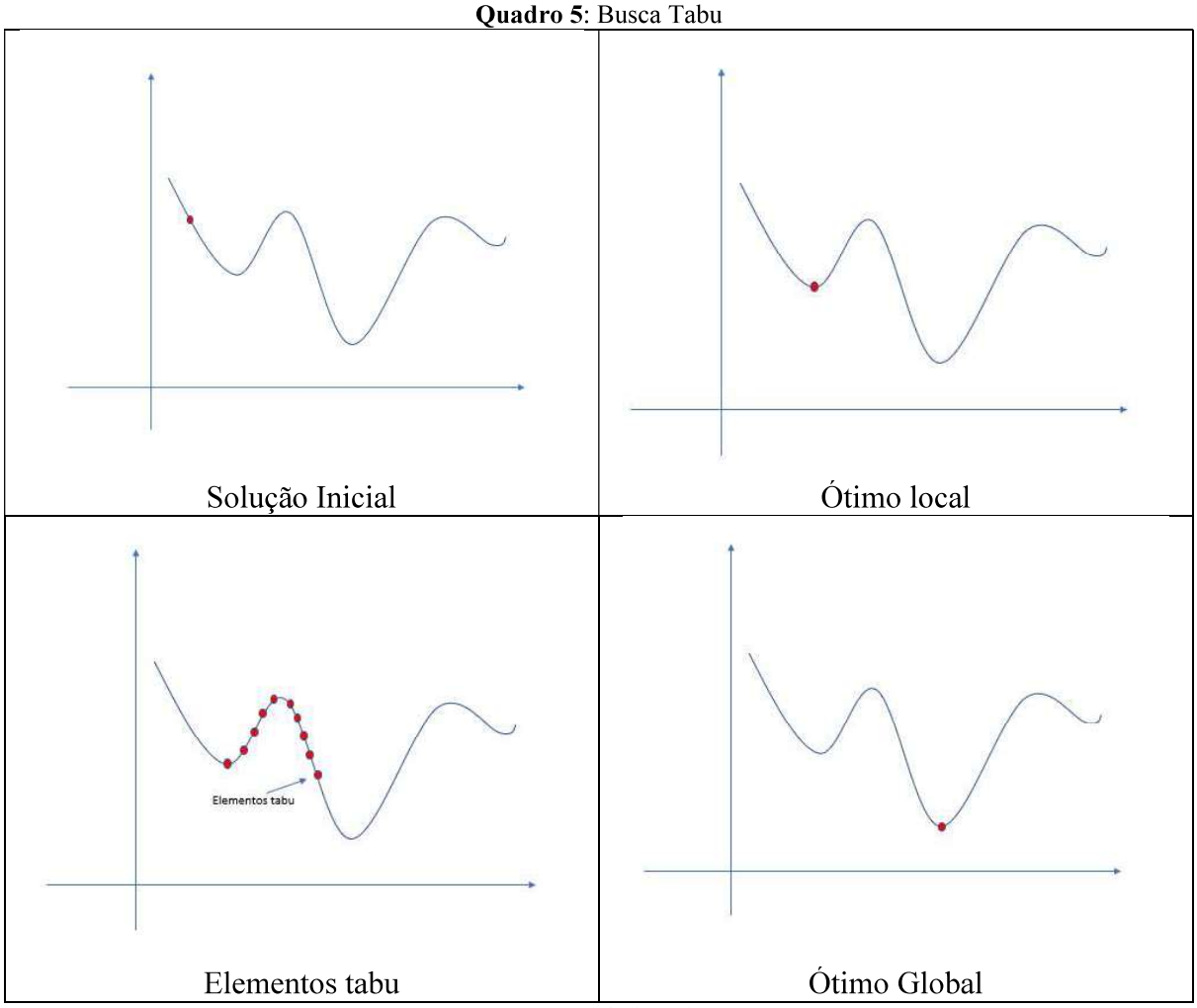
Figura 11: Algoritmo da Meta-heurística Busca Tabu

```
So ← Solução Inicial
S ← So
S* ← So
listaTabu ← { }
Enquanto critério de parada não satisfeito faça
   encontrar melhor solução S' ∈ viz(S) e ∉ listaTabu
   se custo (S') < custo (S*) então
      S* ← S'
   senão
      se custo (S') > custo (S) então
         listaTabu ← listaTabu U {S}
      fim se
   fim se
   S ← S'
fim enquanto
fim BuscaTabu
```

Fonte: Brazil, 2006

Para ajudar o entendimento, deve se ter uma solução inicial (Quadro 5), que pode ser obtida por algum dentre vários métodos e critérios possíveis, a busca continua, a cada iteração, para a melhor solução na vizinhança, não são permitidos movimentos que levem a locais já visitados por serem armazenadas em uma lista tabu. Esta lista permanece guardando soluções já visitadas (tabu) na memória durante um espaço de tempo ou por algumas iterações. Como resultado, espera-se que seja encontrado um ótimo global ou a solução mais próxima dele.

Quadro 5: Busca Tabu



Fonte: Criado pelo autor

2.4 O MATLAB®

O *MATLAB®* é um *software* de uso não livre, com linguagem simples voltado para cálculos matemáticos, e por isso possui uma ampla biblioteca de funções pré-definidas. Essas funções permitem que problemas de programação sejam implementados e resolvidos de maneira muito mais simples do que com outras linguagens computacionais. Por isso é tão usado por engenheiros e cientistas na solução de cálculos matemáticos.

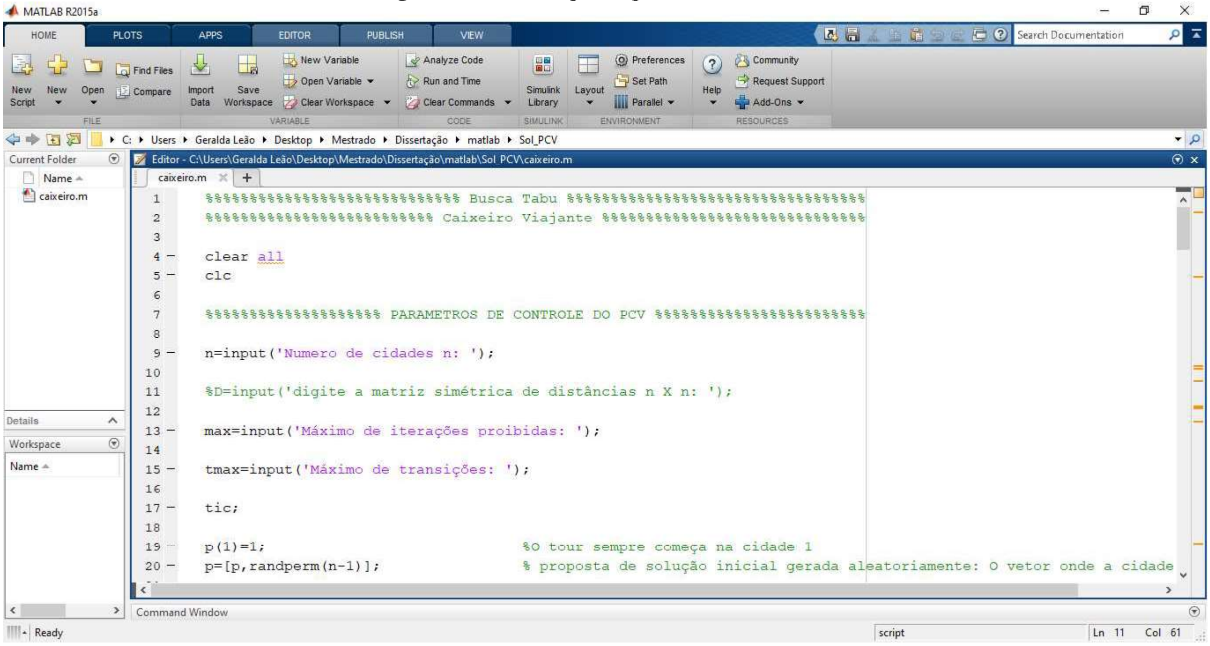
MATLAB®, que é a abreviação de *Matrix LABoratory* – Laboratório de Matrizes, é um programa de computador de uso muito específico, melhorado para executar cálculos matemáticos, científicos e de engenharia (CHAIA; DAIBERT, 2013). Foi desenvolvido na década de 80 por Clever Moler, no Departamento de Ciências da Computação da Universidade do Novo México, EUA (CHAIA; DAIBERT, 2013). Teve origem como um programa para operações matemáticas sobre matrizes e com o passar do tempo e utilização, passou a ser um sistema computacional com a capacidade de resolver muitos problemas técnicos.

O *software* possui diversas vantagens frente a outras linguagens, como por exemplo, a facilidade na sua utilização. Muitas ferramentas são oferecidas para facilitar o uso, como um editor integrado, a documentação disponível na rede, manuais etc. O programa independe de plataforma, assim pode ser utilizado em muitos dos sistemas operacionais disponíveis e programas escritos em qualquer nestes diferentes SO irão funcionar nos outros. Uma extensa biblioteca de funções predefinidas que fornecem soluções para várias atividades básicas técnicas. "Por exemplo, suponha que você está escrevendo um programa que precisa calcular as estatísticas associadas com uma série de dados de entrada. Na maioria das linguagens, você precisaria escrever suas próprias sub-rotinas e funções, para cálculos como média, desvio padrão, mediana, entre outros" (DE SANTIS, 2016), no *MATLAB*® isso está integrado. É possível adquirir *toolboxes* (caixas de trabalho), extremamente úteis para resolver problemas complexos em áreas específicas. "Alguns exemplos de *toolboxes* disponíveis são: processamento de sinal, controle de sistemas, comunicações, processamento de imagens, redes neurais, entre outras" (DE SANTIS, 2016). Possui uma interface gráfica, assim, permite que o programador escreva programas mais bem elaborados em análise de dados, que possam ser operados por usuários pouco experientes.

Das poucas desvantagens que possui, vale a pena citar duas delas. É uma linguagem interpretada e, devido a isso, pode ser mais lenta que outras linguagens, pois as funções diretas e prontas no *MATLAB*® possuem uma complexa matemática por trás. Outra desvantagem relevante é o valor a ser aplicado para aquisição do *MATLAB*®, porém, do ponto de vista técnico e de sua utilização em matemática, ciência e engenharia, o custo-benefício se torna deveras atraente. Mas, em relação a este ponto existem outras opções livres, por exemplo, o *GNU-OCTAVE* e *SCILAB*.

O *MATLAB*® tem sido utilizado nas mais diversas áreas de conhecimento. O processamento digital de imagens, por exemplo, é parte fundamental para o desenvolvimento de visualização e análise de imagens por meio de *softwares* para melhorar a interpretação e destacar aspectos de acordo com as necessidades requeridas (FARIA, 2010). Os processos industriais têm utilizado cada vez mais esse tipo de técnica em situações de avaliação e classificação automática de produtos. No *MATLAB*®, existem funções já definidas para processamento de imagens digitais pela manipulação do seu histograma, existem funções pré-definidas para filtragem das imagens, existem funções para detecção de contornos baseadas na procura de transições de intensidade rápida, desta forma é uma importante ferramenta para efetuar o processamento de imagens digitais e grandemente utilizada em investigação, medicina, atividades subaquáticas. Veja na Figura 12 a interface do *MATLAB*®.

Figura 12: Janela principal do MATLAB®



Fonte: Imagem feita pelo autor

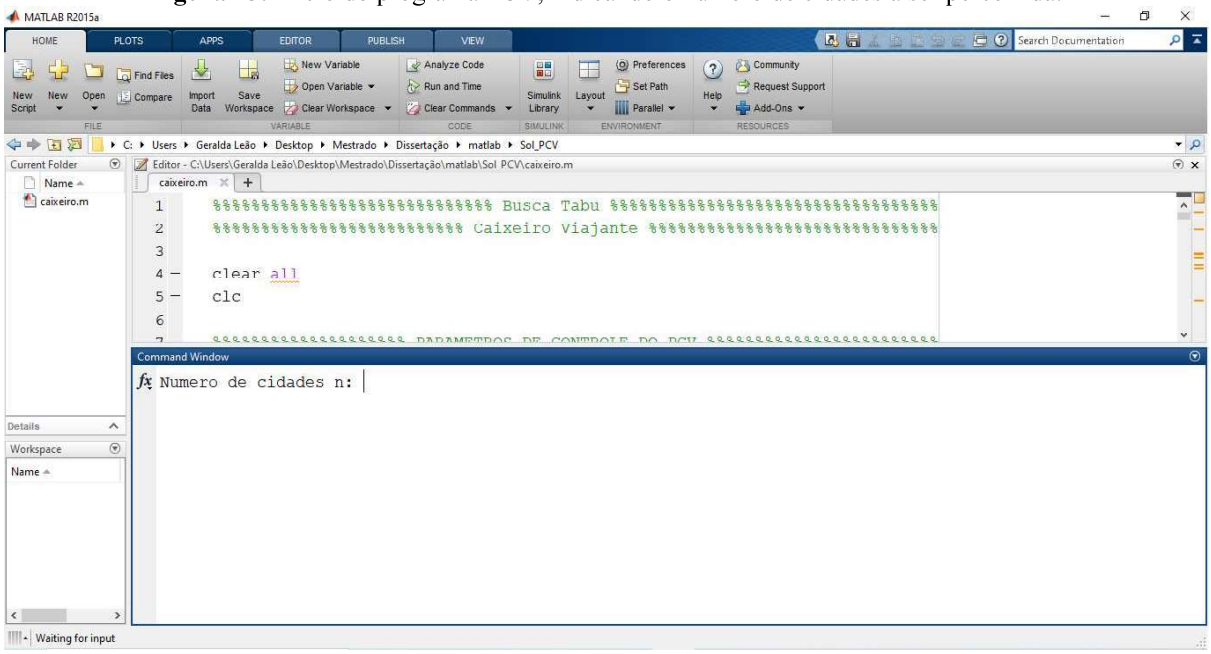
Como citado anteriormente utilizei o *MATLAB®* para criar um sistema de reconhecimento de voz comparando sinais através de diferentes ferramentas matemáticas a fim de apontar qual delas seria a mais adequada para a execução da atividade. Foram testadas e comparadas 2 (duas) ferramentas: Transformada de Fourier e Transformada Wavelet. O estudo apontou que a Transformada *Wavelet* se saiu melhor nos testes feitos.

Como solução do Problema do Caixeiro Viajante, foi escrito um algoritmo baseado na meta-heurística Busca Tabu e implementado no *MATLAB®*. Nesse algoritmo o programa faz algumas perguntas ao usuário ao ser iniciado da seguinte forma: é perguntado o número de cidades que o caixeiro viajante deve percorrer, o número de interações proibidas, isto é, se uma solução foi visitada, então quantas iterações ela deve ficar proibida de ser visitada novamente ficando na Lista Tabu, esse procedimento evita loops em torno de ótimos locais. Pede também o número máximo de transições que é um critério de parada. Isto é, encontrado um ótimo loca, depois desse número predefinido pelo usuário de transições, o algoritmo para e retorna a melhor solução encontrada.

Nas Figuras, 13, 14 e 15 a seguir, uma demonstração da interação do algoritmo com o usuário. É importante ressaltar que as distâncias entre as cidades são dados importantes no PCV. No algoritmo proposto neste trabalho, a matriz de distância é criada aleatoriamente pelo *MATLAB®* ou introduzido no algoritmo pelo programador, porque são situações criadas apenas para teste. Mas, uma melhoria do algoritmo para torna-lo mais realista necessita que essa matriz

de distância seja inserida pelo usuário, afinal nesse caso, seria possível criar uma calculadora par resolver o PCV mais genérico.

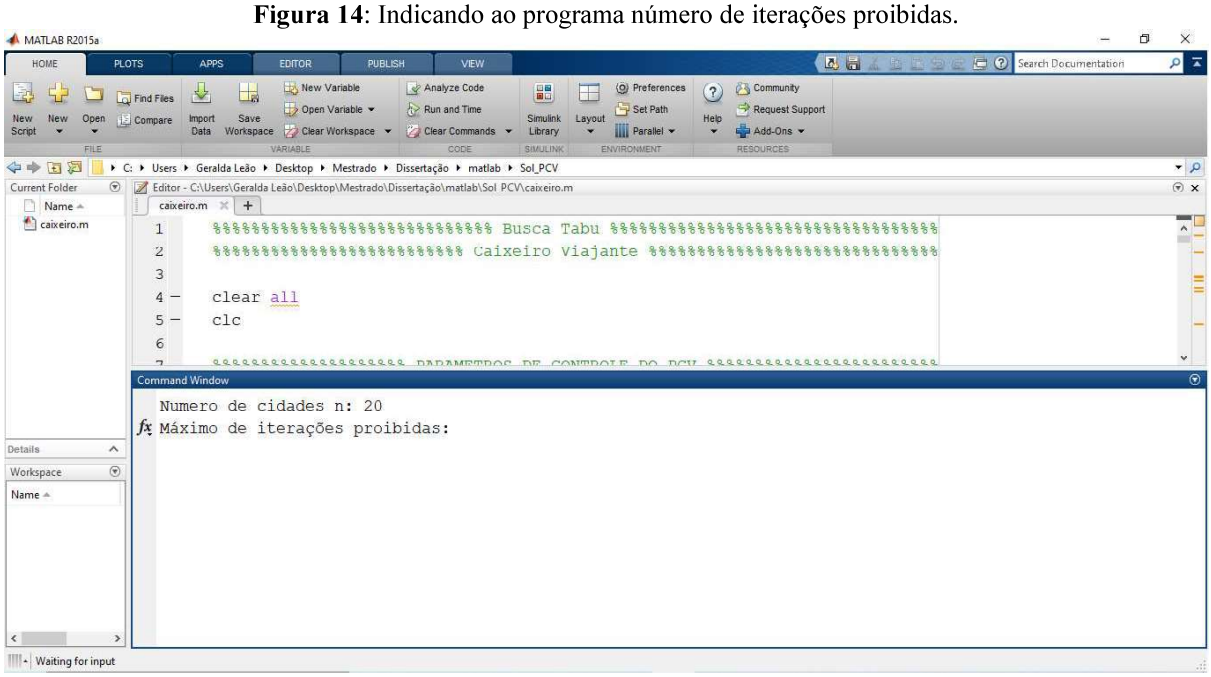
Figura 13: Início do programa PCV, indicando o número de cidades a ser percorrida.



Fonte: Imagem criada pelo autor

Na Figura 13 acima o usuário ou programador digita o número de cidades do PCV. E na Figura 14 abaixo é escolhido o número de iterações proibidas.

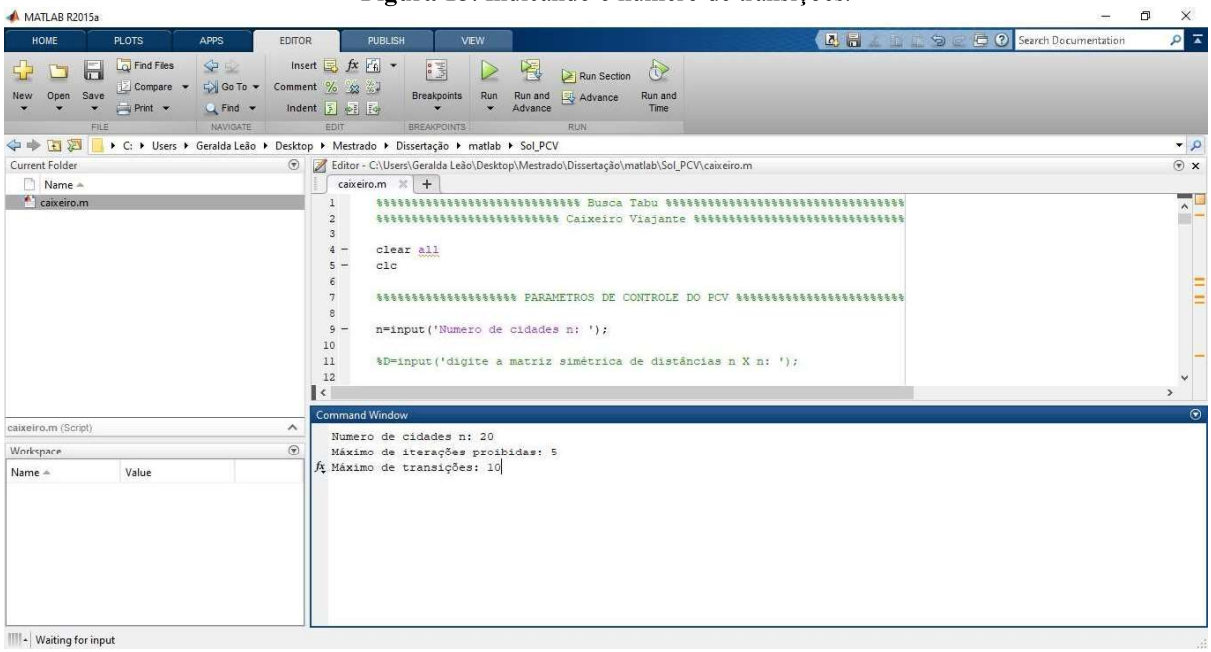
Figura 14: Indicando ao programa número de iterações proibidas.



Fonte: Imagem criada pelo autor

O Número Máximo de Transições é a quantidade de iterações a ser feita após se identificar um ótimo local. Este valor determina a parada do programa. Caso o programa não identifique uma solução melhor, dentro do número de iterações determinada pelo usuário ou programador, o programa para e retorna o melhor valor encontrado até aquele momento.

Figura 15: Indicando o número de transições.



Fonte: Imagem criada pelo autor

Dando sequência neste programa a solução inicial é gerada aleatoriamente através da função *randperm*. A matriz simétrica com as distâncias entre as cidades, com elementos da diagonal principal iguais a zero (criada a partir da função *toeplitz)*, também é gerada aleatoriamente, através da mesma função (*randperm*). Isso significa que a matriz de distância é criada aleatoriamente apenas para testar o algoritmo.

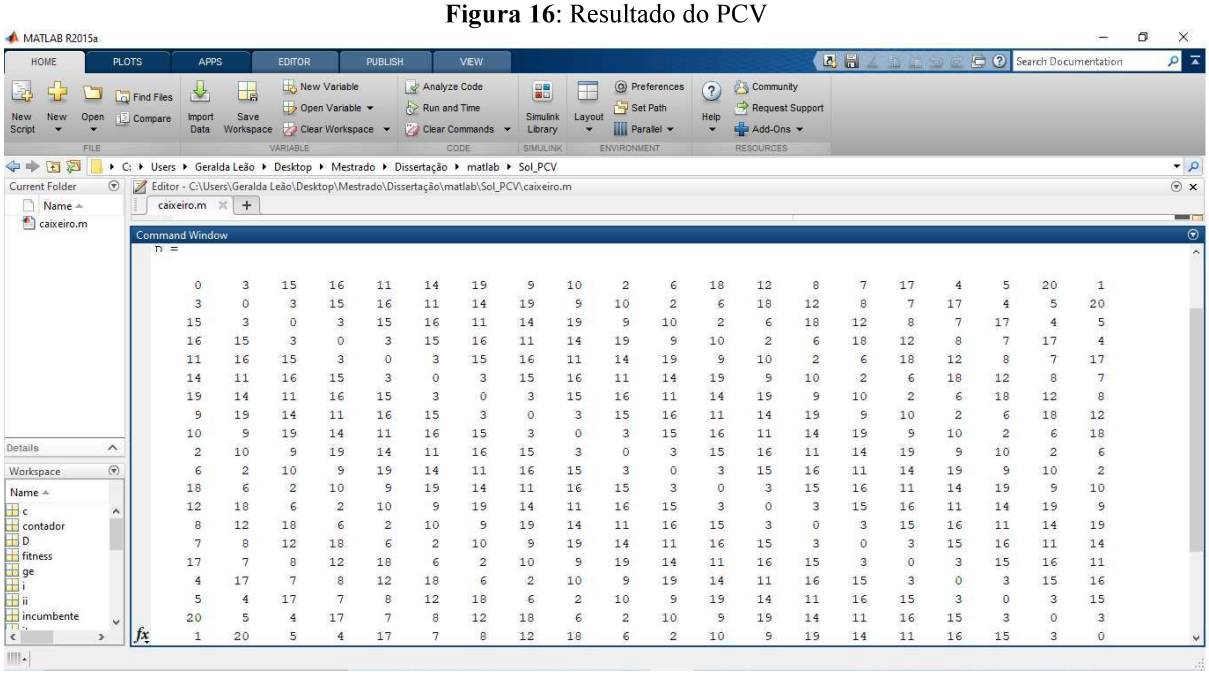
A matriz das distâncias pode ser fixa, alimentada direto no programa, mas para a simulação optou-se por criá-la aleatoriamente. Esta matriz faz parte dos dados do problema que será resolvido, porém, esse programa não faz esse pedido ao usuário por que é um programa de teste, mas é possível no futuro passar essa informação a ser algo informado pelo usuário.

A partir da resposta inicial ou solução aleatória criada pelo programa, o mesmo faz a soma dos valores e cria um "vetor incumbente" composto pelo número da iteração corrente (iniciando pelo 0 (zero)), o valor da soma das distâncias na função objetivo naquela iteração, e o vetor solução que tem menor soma. O vetor incumbente é atualizado à medida que o programa é executado.

2.5 Exemplos de solução do PCV usando o algoritmo através da BT

Em uma simulação feita com 20 cidades, 5 iterações proibidas (considerando ¼ do número de cidades) e número máximo de transições igual a 10, o resultado foi obtido em menos de meio segundo. Conforme resposta abaixo, retirada do programa. A Figura 16 mostra a matriz de distâncias *D*, criada aleatoriamente pelo *MATLAB®*.

Figura 16: Resultado do PCV



Fonte: Imagem criada pelo autor

Para melhorar a visualização do resultado obtido pelo programa, copiei o mesmo abaixo para verificação.

.....RESULTADO ENCONTRADO.....

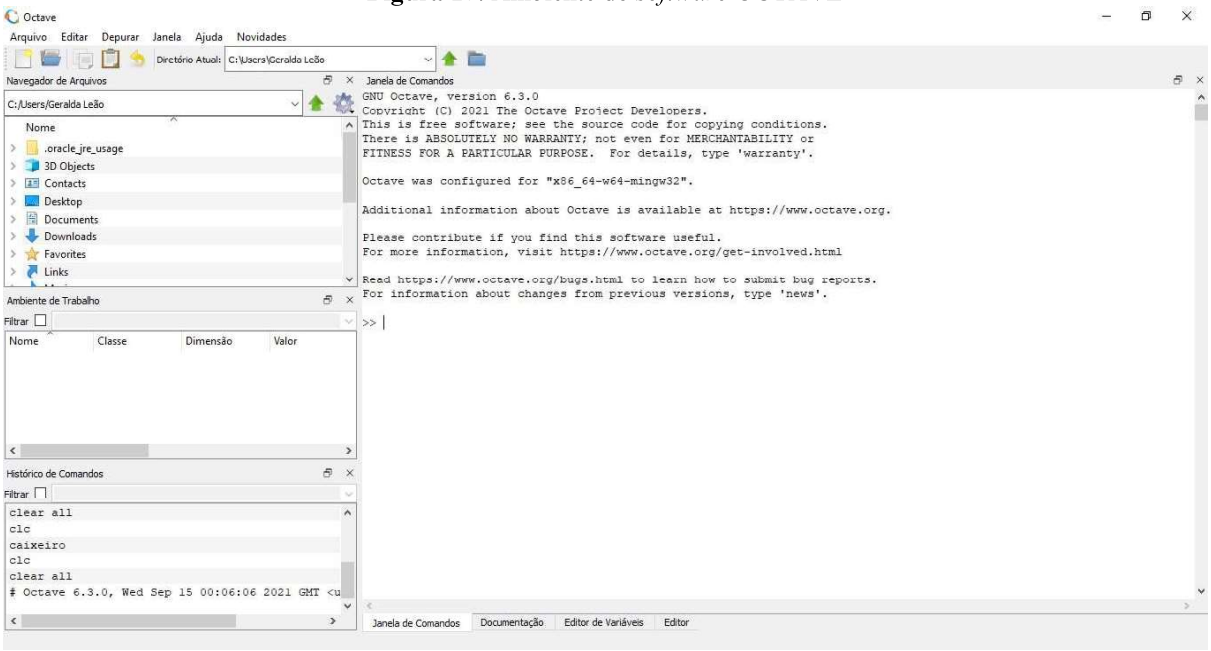
Número de iterações:..... 10.00000000

Tempo decorrido para gerar pop. inicial:..... 0.0665

Valor da incumbente:..... 81.00000000

A título de comparação e como alternativa ao uso do *MATLAB®*, tendo em vista o custo para se obter tal programa, o mesmo programa foi executado no *GNU-OCTAVE*, *software* livre e que admite a mesma linguagem utilizada no *MATLAB®*. Segue abaixo sequência de respostas.

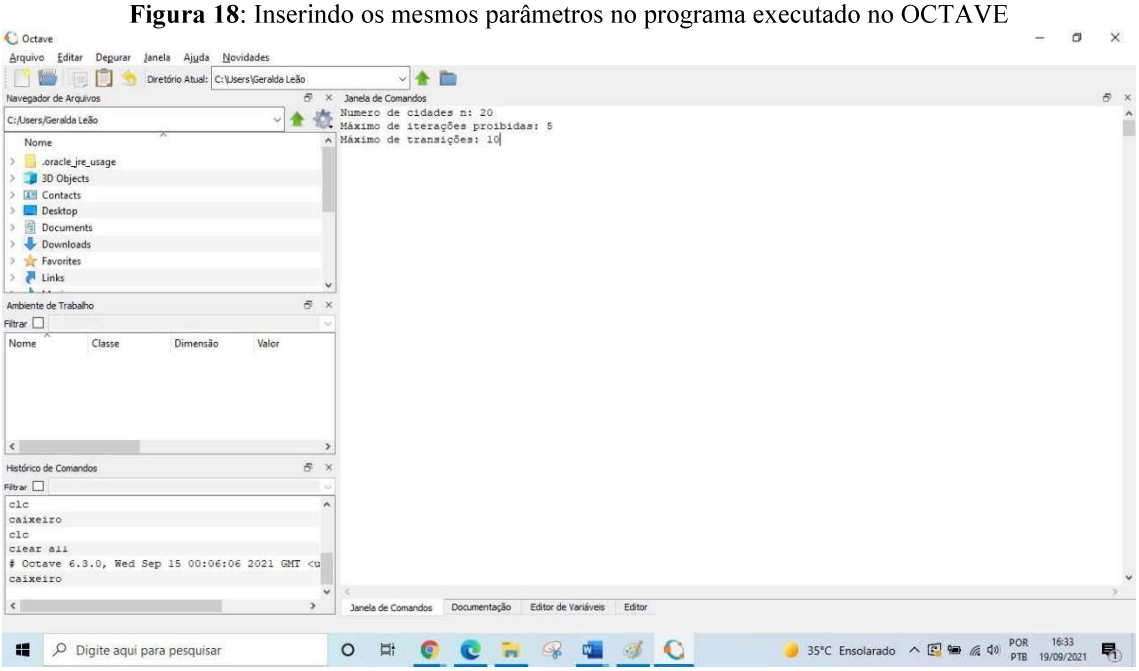
Figura 17: Ambiente do *software* OCTAVE



Fonte: Imagem criada pelo autor

A Figura 17 mostra como é o ambiente do GNU-OCTAVE ao inicializar o *software.* Na Figura 18 está o programa escrito para resolver o PCV interagindo com o usuário.

Figura 18: Inserindo os mesmos parâmetros no programa executado no OCTAVE



Fonte: Imagem criada pelo autor

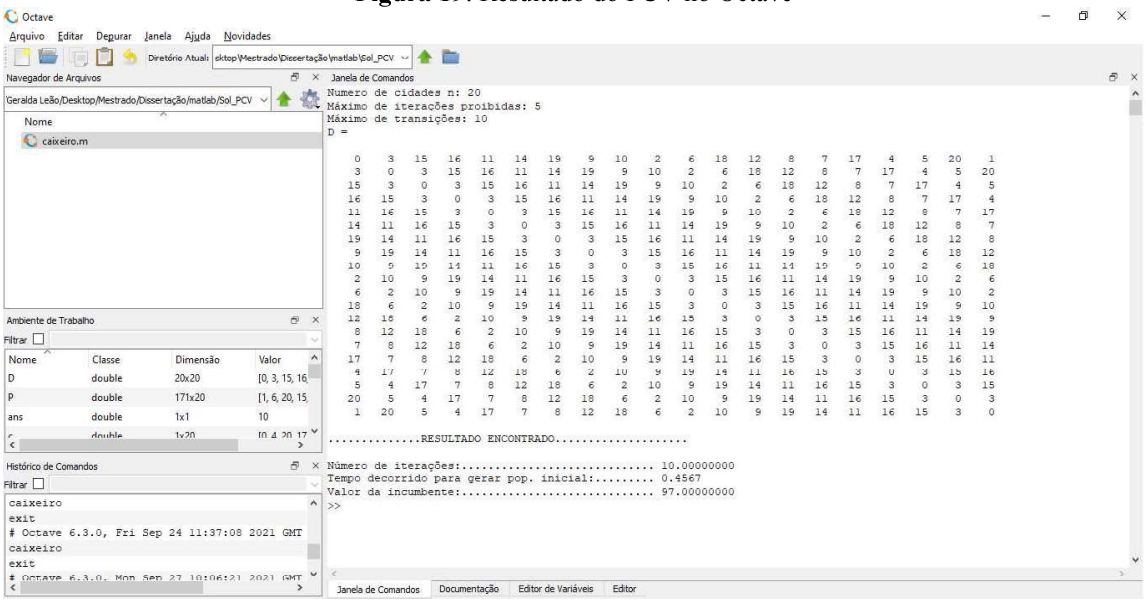
Acima, na Figura 18, estão listadas as entradas do programa, como é possível ver, as entradas são as mesmas que as utilizadas no *MATLAB®*, assim como foi utilizada a mesma Matriz de Distâncias, para que seja possível comparar o desempenho de cada programa.

Número de cidades n: 20

Máximo de iterações proibidas: 5

Máximo de transições: 10

Figura 19: Resultado do PCV no Octave



Fonte: Imagem criada pelo autor

Na Figura 19 acima, além da matriz de distâncias é apresentada a solução do PCV. Também com a intenção de melhorar a visualização das respostas obtidas no *GNU-OCATVE*, abaixo estão os resultados obtidos.

.....RESULTADO ENCONTRADO.....

Número de iterações:..... 10.00000000

Tempo decorrido para gerar pop. inicial:..... 0.4567

Valor da incumbente:..... 97.00000000

Comparando as respostas e o desempenho dos dois *softwares* para uma primeira simulação em ambos, percebe-se que o *MATLAB®* apresentou um melhor desempenho dado o tempo percorrido de 0,0665 segundos contra 0,4567 segundos obtidos pelo *GNU-OCTAVE* para retornar a resposta, ou seja, uma resposta que foi mais de 5 (cinco) vezes menor. Além disso a solução do problema (caminho) para esta simulação também determinou um menor valor da Função Objetivo no *MATLAB®*, sendo 81 obtido por este *software* contra 97 pelo *GNU-OCTAVE*. Além disso, em 20 simulações feitas em ambos os programas o *MATLAB®* se saiu melhor em todas as simulações quando o quesito é tempo, já em relação a obtenção do menor caminho, ou seja, melhor solução, o resultado foi um empate. Cada um teve melhor

resultado em 10 simulações. Os resultados das simulações estão apresentados no Quadro 6 e os melhores resultados estão destacados.

Quadro 6: Resultado das 20 simulações

	MATLAB®		GNU-OCTAVE	
Simulação	Tempo de simulação (s)	Caminho total	Tempo de simulação (s)	Caminho total
1	0.3847	81	0.7377	94
2	0.0701	80	0.5325	71
3	0.0744	81	0.5357	83
4	0.0712	89	0.5339	85
5	0.0695	83	0.5322	95
6	0.0709	65	0.5363	86
7	0.0715	72	0.5403	81
8	0.0750	88	0.5424	81
9	0.0729	83	0.5364	82
10	0.0711	80	0.5292	66
11	0.0712	74	0.5582	87
12	0.0684	65	0.5538	77
13	0.0694	87	0.5317	83
14	0.0706	83	0.6939	75
15	0.0714	63	0.5377	78
16	0.0702	80	0.6268	70
17	0.0771	81	0.5334	97
18	0.0712	89	0.5364	64
19	0.0699	83	0.5335	71
20	0.0693	67	0.5362	77

Fonte: Criado pelo autor

2.6 Programação Matemática

Para ilustrar uma aplicação e a importância de estudantes terem contato com técnicas e ferramentas voltadas para a programação matemática mais precocemente, ainda no Ensino Médio, e com o intuito de amadurecimento do pensamento matemático e aproximação de tecnologias que poderão fazer parte da vida adulta deste estudante, apresento abaixo um problema de otimização, que pode ser uma possibilidade pedagógica, a ser solucionado por método matemático, com conteúdo do ensino médio e método computacional ou uso de *softwares* livre.

Segue o problema:

Um fazendeiro dispõe de 5 alqueires para plantar feijão e arroz. Ele deve decidir quanto plantar de cada cultura de modo que o rendimento (lucro) seja máximo. Cada alqueire de

feijão produz um rendimento líquido de R$ 800,00; cada alqueire de arroz, R$ 1000,00. Para o feijão são necessários 10.000 litros de água por alqueire; para o arroz, 20.000 litros. Cada alqueire de feijão necessita de 150 kg de fertilizantes, o que não é necessário no cultivo de arroz. Dispõe-se de um total de 80.000 litros de água e de 600 kg de fertilizantes para a empreitada. Quantos alqueires de feijão e quantos de arroz devem ser plantados?

A questão gira em torno da intenção de se responder quanto de arroz e quanto de feijão deve ser plantado para ter o aproveitamento máximo, segundo as condições do problema.

A matemática para solução do problema é simples. Os dados estão dispostos na tabela abaixo.

Tabela 1: Dados do problema

	Alqueires	**Água/Alqueire**	**Fertilizante/Alqueire**	**Rendimento/Alqueire**
Feijão	*x*	10.000	150	800
Arroz	*y*	20.000	0	1000
Total	5	80.000	600	R

Fonte: Criada pelo autor

Modelando o problema em termos matemáticos tem-se,

$$R = 800x + 1.000y \rightarrow \text{Rendimento} \qquad (1)$$

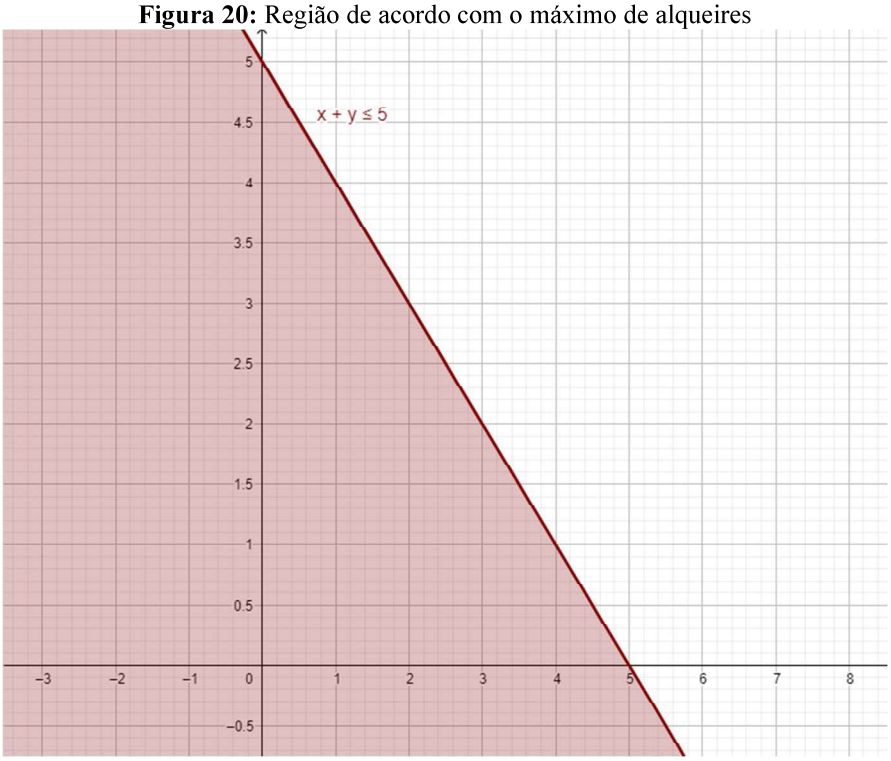
$$I - \begin{cases} x + y \leq 5 & (2) \\ 10.000x + 20.000y \leq 80.000 & (3) \\ 150x + 0y \leq 600 & (4) \end{cases}$$

$$II - \begin{cases} x + y \leq 5 & (5) \\ x + 2y \leq 8 & (6) \\ x \leq 4 & (7) \\ x \geq 0 & (8) \\ y \geq 0 & (9) \end{cases}$$

Assim, quanto de arroz e quanto de feijão para que o rendimento (1) seja melhor possível?

Conforme equação (2) o valor de x e y (alqueires de feijão e arroz) somados deve ser no máximo 5 (cinco), de acordo com o enunciado do problema. Assim essa soma deve ser menor ou igual, conforme inequação (5). Observe abaixo na Figura 20, um gráfico representando geometricamente a inequação (5). Evidentemente, qualquer ponto *P*(*x*,*y*) na região colorida satisfaz $x + y \leq 5$.

Figura 20: Região de acordo com o máximo de alqueires

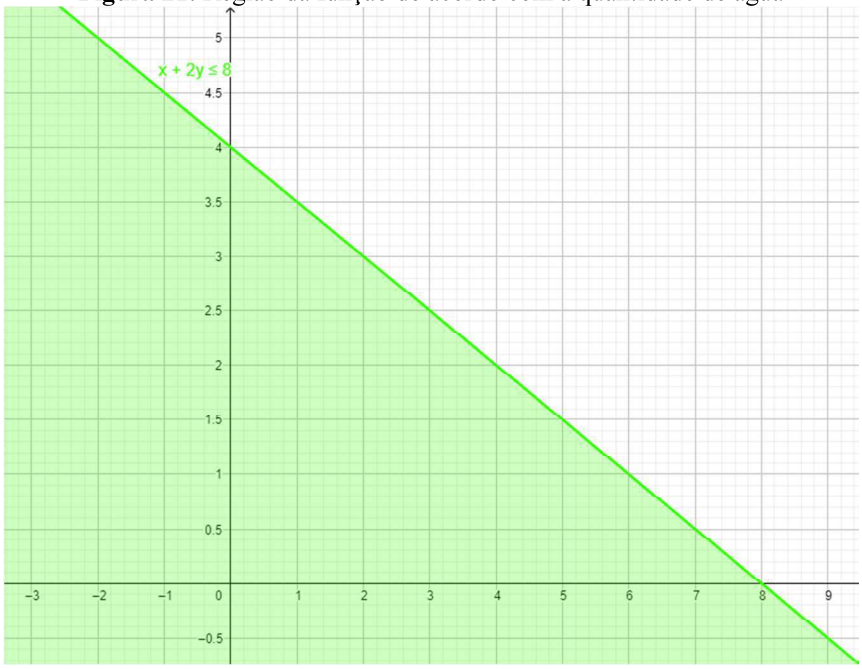


Fonte: Criado pelo autor no Geogebra

Observe que neste tipo de solução, vários elementos podem ser trabalhados com alunos do ensino médio: inequações e sua representação geométrica, noção de pontos de máximo e de mínimo para uma dada função linear e claro a apresentação do *software* GeoGebra. Dessa forma, este problema da produção agrícola retratado aqui poderá servir de estimulo para o ensino de determinados conteúdos e uso de computador ou *smartfones* nas aulas de matemática.

A inequação (3) diz respeito à quantidade de água gasta com a plantação. Assim a quantidade de água não pode ultrapassar 80.000 litros. Fazendo simplificações podemos chegar na inequação equivalente (6). A Figura 21, representa geometricamente através da região colorida, todos os pontos $P(x, y)$ do plano $\mathbb{R}^2$, que satisfazem $x + 2y \leq 8$.

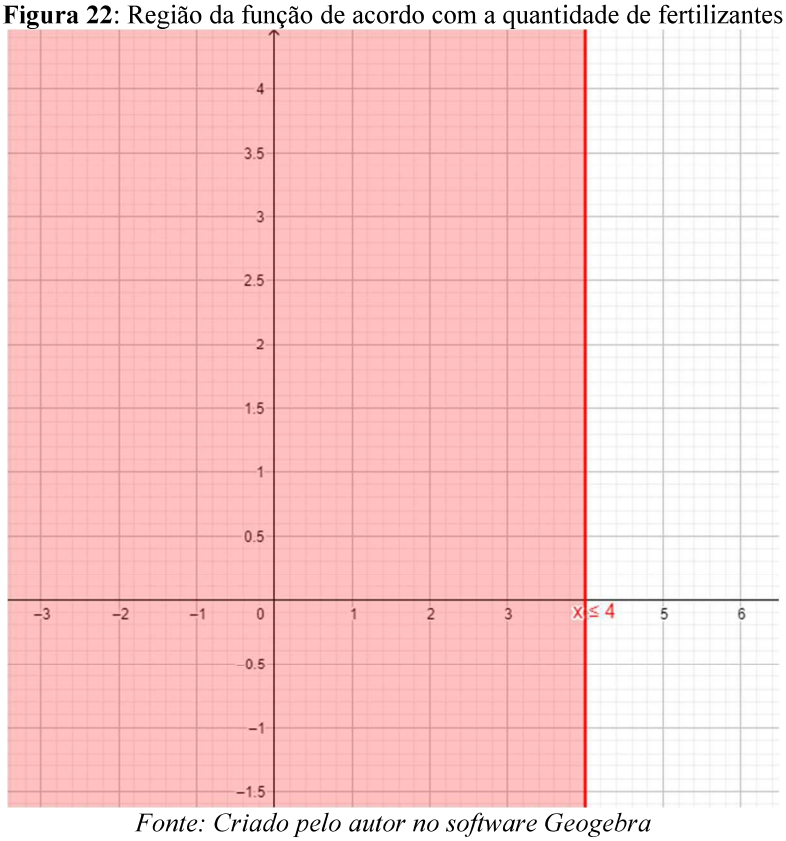
Figura 21: Região da função de acordo com a quantidade de água



Fonte: Criado pelo autor no software Geogebra

A inequação (4) trata da quantidade de fertilizantes a ser utilizado nesta plantação. Veja que ela é equivalente à inequação (7) representada geometricamente pela Figura 22 a seguir.

Figura 22: Região da função de acordo com a quantidade de fertilizantes

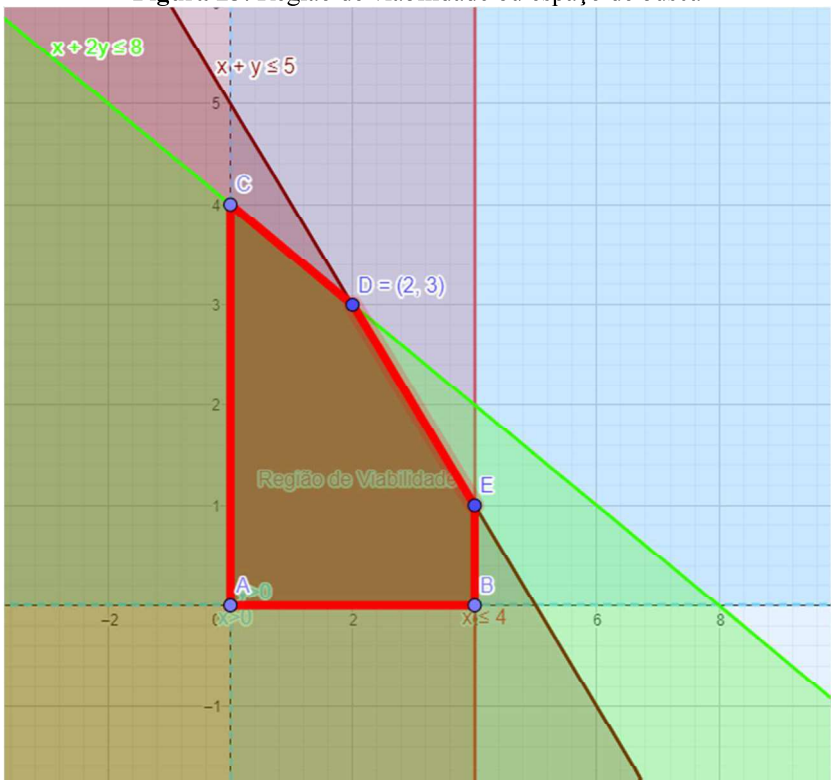


Fonte: Criado pelo autor no software Geogebra

Simplificando tem-se o conjunto de inequações conforme chave II.

Colocando essas situações no mesmo plano cartesiano, plotando esses gráficos, percebe-se que a região de interesse, espaço de busca (região de viabilidade) da curva (conforme Figura 23) é a região positiva do plano cartesiano, isto é, com $x \geq 0$ e $y \geq 0$, que seja a região abaixo da curva (5), região abaixo da curva (6) e região a esquerda da reta (7). A intersecção dessas regiões é a região de viabilidade desse problema, representada na Figura 23.

Figura 23: Região de viabilidade ou espaço de busca



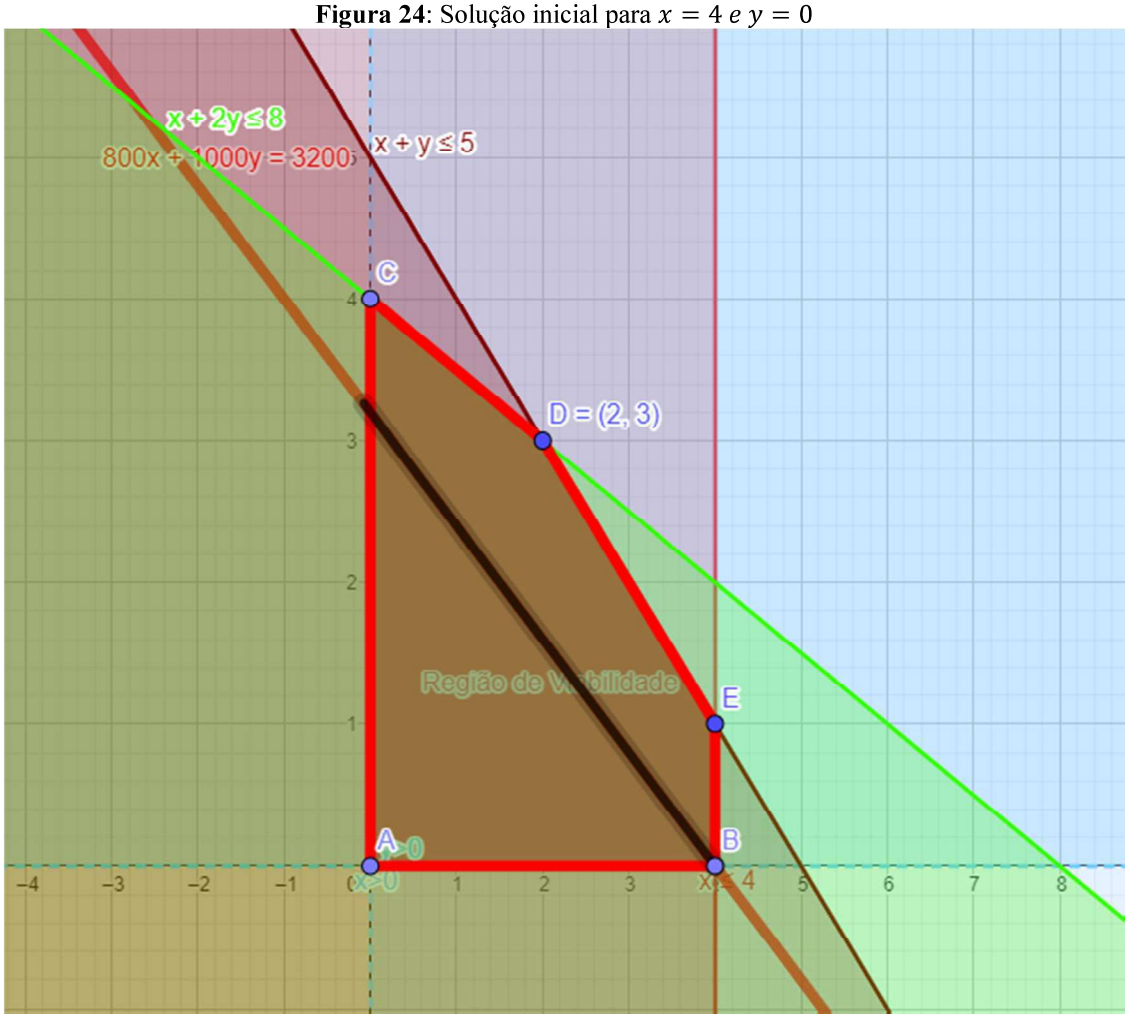
Fonte: Criado pelo autor no software Geogebra

Pegando qualquer par ordenado aleatoriamente, por exemplo, o ponto de coordenadas $x = 4, y = 0$. Substituindo esses valores na equação (1), função objetivo, obtém-se o Rendimento de 3200, ou seja,

$$R = 800x + 1.000y = 3200 \quad (8)$$

Assim, o rendimento é expresso por uma reta, ou seja, a reta expressa pela equação em (8), de modo que o rendimento é o mesmo para qualquer ponto daquela reta dentro do espaço de viabilidade, veja Figura 24.

Figura 24: Solução inicial para $x = 4\ e\ y = 0$



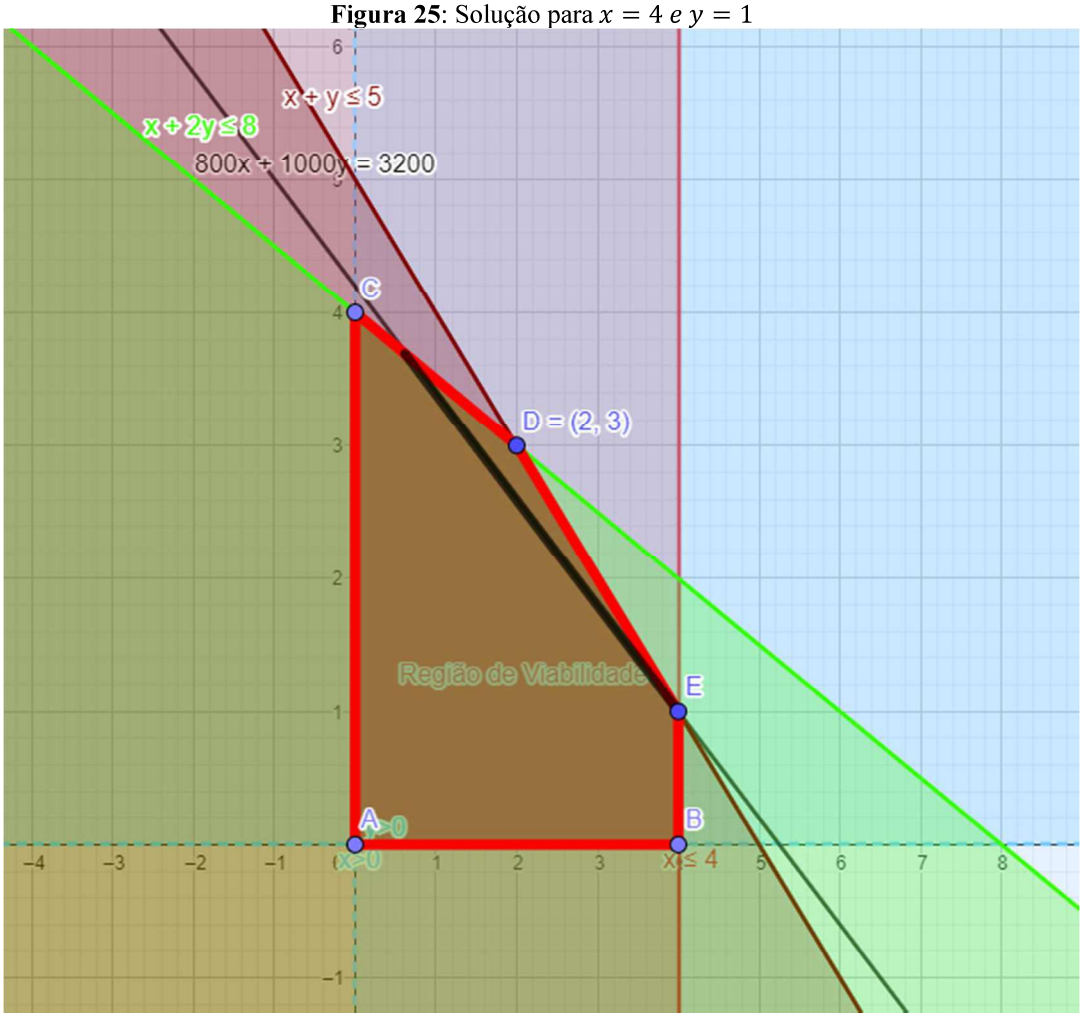
Fonte: Criado pelo autor no software Geogebra

Mas, o objetivo do problema é saber qual o máximo rendimento possível (nas condições fornecidas), então, peguemos arbitrariamente outro ponto que também seja um vértice do polígono dado pela Região de Viabilidade. O ponto escolhido é o de coordenadas $x = 4, y = 1$, tem-se,

$$R = 800x + 1.000y = 4200 \quad (9)$$

Note que este ponto $P(4,1)$ é um dos vértices do Espaço de Busca, assim como o ponto $Q(4,0)$. Além disso, as retas (8) e (9) são paralelas. Um ótimo momento para conversar com os alunos sobre retas paralelas e as diferenças entre as duas equações e seus significados.

Figura 25: Solução para $x = 4$ *e* $y = 1$



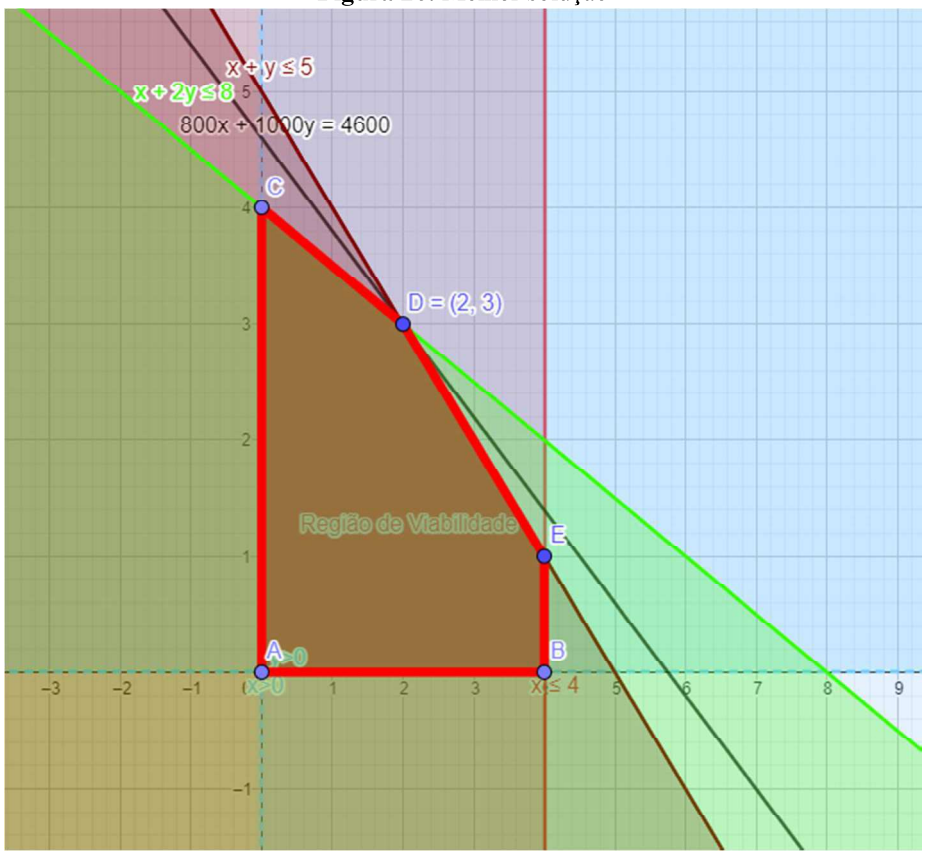
Fonte: Criado pelo autor no software Geogebra

Mais uma vez (9), representa outra reta, paralela à reta dada por (8). Percebe-se que quanto maior o valor de R, mais alta essa reta corta o eixo y. Então podemos concluir como resposta, por observação do gráfico na Figura 25, que o maior rendimento é dado pela reta que passa pelo ponto D que é intersecção das retas limites das regiões definidas pelas inequações (5) e (6), que corta o eixo y no ponto mais alto e D está na região de viabilidade. Analisando a Figura 26 tem-se que o ponto ótimo é dado pelo ponto D de coordenadas $x = 2$ alqueires de feijão e $y = 3$ alqueires de arroz, então o rendimento será de:

$$R = 800x + 1.000y = 800 \times 2 + 1.000 \times 3 = 4.600{,}00 \quad (10)$$

Desta forma resolve-se o problema e apresenta uma situação que poderá ser usada em sala de aula, no ensino médio como aplicação de conteúdos inerentes ao currículo.

Figura 26: Melhor solução



Fonte: Criado pelo autor no software Geogebra

Este é um problema de complexidade baixa, envolve apenas funções em primeiro grau, e por isso mesmo é interessante do ponto de vista de implementação em programação matemática e mesmo a partir de exploração geométrica à mão ou através do *software* GeoGebra.

A seguir está uma sugestão de roteiro para resolução de problemas de otimização tal qual o exposto acima:

O professor deve escolher ou formular uma situação problema cuja modelagem recai em uma função objetivo de duas variáveis e um conjunto de restrições com duas ou mais inequações e duas variáveis, e a partir daí seguir a sequência:

1- Desenhar num sistema de coordenadas cartesianas a primeira equação;
2- Marcar a região de interesse de acordo com o problema;
3- Desenhar num sistema de coordenadas cartesianas a segunda equação;
4- Marcar a região de interesse de acordo com o problema;
5- Fazer o mesmo com todas as inequações dadas no problema;
6- Considerar que as variáveis são maiores ou iguais 0 (zero);

7- Com todas as regiões marcadas a intersecção entre elas é o espaço de busca;
8- A partir daí, considerando que a solução ótima é um dos pontos de vértice dessa região do espaço de busca, testar na função objetivo cada um deles para verificar qual deles oferece o melhor (maior ou menor) valor para a função objetivo.

Partindo do resultado obtido pela análise geométrica do problema, devemos entender que o aluno deve estar preparado para achar uma resposta, qualquer que seja ela. Não precisa ser a resposta ótima, mas precisa ser uma resposta. A partir dela se inicia o trabalho de busca pela resposta ótima, ou o mais próximo dela que seja possível chegar com custo aceitável (tempo, custo computacional) utilizando qualquer técnica, como a geométrica apresentada acima, ou algébrica ou qualquer outra que se tenha interesse e domínio. Interpretar corretamente o problema e ser capaz de apresentar uma resposta é o mais importante.

Sabendo disso, o problema acima foi resolvido também utilizando a meta-heurística Busca Tabu no *Google Colab* (*Python*), como mais uma alternativa ao *MATLAB®*. O *Google Colab* é um ambiente virtual e gratuito que permite que se escrevam códigos e textos, é um ambiente colaborativo, que permite o compartilhamento das atividades desenvolvidas com colegas, permitindo que outros rodem seu código e até modifiquem criando suas próprias versões é, portanto, um ambiente interativo. Do ponto de vista geométrico o programa vai "passeando" pelos pontos dentro da região de viabilidade ou espaço de busca, no domínio dos inteiros e salvando boas respostas. Após encontrar uma boa resposta ele repete o programa por 4 (quatro) vezes (quatro iterações) e então retorna o valor do ponto e a resposta do problema. Veja o código fonte no Apêndice. Entende-se que os ambientes interativos como este pode ser de grande ajuda para estudantes com interesse em aprender programar e resolver situações problemas de otimização ou de modelagem matemática.

O ambiente iterativo *Google Colab* (*Python*) (https://colab.research.google.com/) é um espaço que poderá ser visitado por alunos e professores e dependendo do propósito e objetivo é possível ser usado em sala de aula para explorar alguns conhecimentos matemáticos com seus alunos. A resolução desse problema teve como objetivo apresentar uma possibilidade mais adequada aos estudantes do ensino médio por meio da exploração geométrica, mas também se apresenta a possibilidade da programação em computador através de um ambiente iterativo e mais voltado aos jovens com interesse na área.

3. Considerações finais

O Problema do Caixeiro Viajante possui uma matemática mais complexa. Apesar de ser de fácil entendimento seu enunciado e apesar de toda a sua associabilidade à situações do dia a dia do aluno, é ainda um problema complexo e que a medida que o número de cidades aumenta sua complexidade aumenta proporcionalmente. O *MATLAB®* e o *GNU-OCTAVE* são ferramentas de programação similares, porém seus resultados foram bastante diferentes em relação ao tempo para obtenção da resposta o *MATLAB®* teve melhor resultado em todas as simulações, porém, resultados próximos com relação ao tamanho do caminho.

O estudante estar ambientado com ferramentas tecnológicas que o auxiliem na resolução deste tipo de problema é importantíssimo. Vai dar condições a esse aluno a enfrentar e resolver de forma similar e outros problemas que possam surgir.

O segundo problema proposto possui matemática mais elementar, porém, muito interessante do ponto de vista de construções de estratégias de solução e comparação com métodos computacionais tendo em vista que este pode ser resolvido algébrica ou geometricamente. Além disso, é um problema condizente com a nossa região, norte de Mato Grosso, por causa da agricultura que é forte componente econômico. E, utiliza-se conteúdos ministrados no ensino médio, o que possibilita introduzir conceitos de otimização para trabalhar com funções, sistemas de equações e inequações lineares, matrizes e operações elementares com matrizes.

Problemas com complexidades bem diferentes e abordagens das mais diversas possíveis em se tratando de programação matemática. Apesar de se ter sugerido e utilizado a meta-heurística Busca Tabu como estratégia de solução, existem outras com vasta literatura disponíveis para provocar o aluno na busca de uma solução mais ou tão interessante quanto, que envolva menos custo computacional, gaste menos tempo, que tenha aplicabilidades em outros tipos de problemas e etc. Da mesma forma deve ser entendida a utilização do *MATLAB®* como ferramenta para implementação dessa solução. Por ser um *software* pago e entendendo que a realidade financeira da maioria das escolas públicas do país impede a aquisição desse programa, o foco não deve ser nos problemas em específico ou programa a ser utilizado, mas sim na sedimentação de uma base técnica que prepare esse aluno para desafios que possam ter no seu futuro acadêmico, e quiçá, laboral. De se iniciar uma discussão sobre a importância do contato desses alunos com programação.

O artifício de ferramentas tecnológicas serem cada vez mais inseridas em sala é algo que pode aumentar o interesse do aluno por determinado assunto, assim como, é importante para a escola e sala de aula criarem essa sintonia com o mundo no qual vivemos. Não é só o ensino da matemática, mas o conteúdo tem que ser tratado de outra forma e o currículo escolar vai se privilegiar desse tipo de artifício. Por exemplo, o professor dedica algumas horas no ensino do cálculo do determinante de uma matriz e existem ferramentas *online* disponíveis capazes de fazer esse cálculo com muita rapidez e facilidade. O ensino da matemática deve ressignificar isso. Ensinar o aluno a calcular e mostrar que existe tecnologia que vai auxiliá-lo a resolver determinado problema, desde que ele domine tanto o conteúdo quanto a ferramenta que se deseja utilizar.

O mundo caminha a passos largos em direção à informatização. Tecnologias surgem aos montes em forma de linguagem de programação, tecnologia embarcada, aplicativos de celular, drones, sistema de georreferenciamento, entre outros. A única constante em todos eles é a matemática. Ela é o idioma que a natureza escolheu para explicar tudo. Galileu outrora afirmou que "o livro da natureza está escrito em linguagem matemática". O homem tem entendido isso e moldado todo esse conhecimento em forma de tecnologia que auxilia no entendimento de fenômenos e resolução de problemas. O intuito deste trabalho foi fomentar a importância de se começar a introduzir teorias de algoritmo e programação na resolução de problemas matemáticos, tentando contribuir como que esse tema de otimização poderia incentivar os alunos de ensino médio a estudarem matemática observando que aqueles conteúdos estudados têm importância do ponto de vista de tecnologia utilizada no dia a dia por eles, por exemplo celular, na resolução de problemas

A BNCC tem visto isso e direcionado suas competências nessa direção. E assim possibilitar um tipo de conhecimento que pode ser de muita utilidade na vida adulta e profissional desse aluno.

4. Referências Bibliográficas

BRASIL. Base Nacional Comum Curricular (BNCC). Consulta Pública. Brasília, MEC/CONSED/UNDIME, 2018. Disponível em: <http://basenacionalcomum.mec.gov.br/abase/#medio >. Acesso em: 10.junho.2021.

BRAZIL, J.C.; LEITE, M.S.; MENDES, R.B.S.; Uma Metaheurística Grasp Busca Tabu para o Problema do Caixeiro Viajante. II Workshop de Computação Científica da UENF - IIWCC, 2006

CHAIA, A. V.; DAIBERT, M. R. Mini Curso Introdução ao MATLAB®. p. 27, 2013.

DE SANTIS, R. B. MATLAB® - Seminário. p. 29, 2016.

FARIA, D. Trabalhos Práticos Análise e Processamento de Imagem. p. 44, 2010.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, v. 13, n. 5, p. 533–549, jan. 1986.

GOLDBARG, M. C.; LUNA, H. P. L. Otimização combinatória e programação linear: modelos e algoritmos. Rio de Janeiro: Elsevier, 2005.

GOMES, A. Uma Introdução à Busca Tabu. p. 27, 2009.

MATLAB. *Software* Matemático. Versão 8.5.0 (R2015a). Disponível em: https://www.mathworks.com/products/matlab.html. Acesso em: set. 2021.

MOSCATO, P. An Introduction to Population Approaches for Optimization and Hierarchical Objective Function: A Discussion on the role of Tabu Search. Annals of Operations Research, Vol. 41, Number 1-4, pp. 85-121, 1993.

POLYA, G. - "A arte de resolver Problemas". Editora Interciência - (1977). "How to solve it", 1943

SUCUPIRA, I. R. MÉTODOS HEURÍSTICOS GENÉRICOS: META- HEURÍSTICAS E HIPER-HEURÍSTICAS. p. 41, 2004.

TAVEIRA, R. C. L. Programa de Mestrado Profissional em Matemática em Rede Nacional. p. 65, 2005.

VIANA, Anderson. Algoritmos em Grafos e o Problema do Caixeiro Viajante: uma abordagem no Ensino Médio utilizando planilhas eletrônicas. Dissertação para o Profmat. 2014

APÊNDICE

PROBLEMA DO CAIXEIRO VIAJANTE RESOLVIDO NO MATLAB® UTILIZANDO A META-HEURÍSTICA BUSCA TABU

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Busca Tabu %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% Caixeiro Viajante %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
clc

%%%%%%%%%%%%%%%%%%%%%% PARAMETROS DE CONTROLE DO TSB %%%%%%%%%%%%%%%%%%%%%%%%

n=input('Numero de cidades n: ');

%D=input('digite a matriz simétrica de distâncias n X n: ');

max=input('Máximo de iterações proibidas: ');

tmax=input('Máximo de transições: ');

tic;

p(1)=1;                                 %O tour sempre começa na cidade 1
p=[p,randperm(n-1)];                    %proposta de solução inicial gerada
aleatoriamente: O vetor onde a cidade 1 é fixa

%p=[1 3 1 8 1];                         %Uma possibilidade é deixar a solução
inicial fixa (para teste).

for j=2:n
    if p(j)==1
        p(j)=n;                         %permutando (n-1) cidades, colocar a
cidade que ficou fora no lugar do número 1 que aparecerá duas vezes
    end
end


                                        %Cria uma matriz simétrica aleatória de
distâncias, onde os elementos da diagonal principal são iguais a zero

c=randperm(n);
c(1)=0;
D=toeplitz(c,c);

% save('matriz distancia','D');         %Salva a matriz de distâncias criada
anteriormente;

%load('matriz distancia','D');          %Lê a matriz de distâncias e a usa de
forma fixa;

%D=[0 5 1 3 4;5 0 5 1 3;1 5 0 5 1;3 1 5 0 5;4 3 1 5 0]; %Uma possibilidade
é deixar a matriz D fixa.
```

```
soma=0;                          %A distância da primeira cidade à ela mesma
for j=1:(n-1)
    soma=soma+D(p(j),p(j+1));    %Função objetivo, soma as distâncias das
cidades na ordem dada pelo vetor p
end

soma=soma+D(p(1),p(n));          %soma das distâncias das n cidades +
distância da última cidade para a primeira.

% as próximas três linhas constituem um vetor, que chamaremos vetor
incumbente

incumbente(1,1)=0;           %(isso significa que na linha 1 coluna 1 da
matriz incumbente está recebendo o número zero que indica o num da iteração
atual)
incumbente(1,2)=soma;        %(isso significa que na linha 1 coluna 2 da
matriz incumbente está recebendo o número soma que indica o valor da função
                             % objetivo para a iteração atual)
incumbente(1,3:(n+2))=p(:); %(Isso significa que da coluna 3 até a (n+2)
será colocado o vetor solução encontrada da linha 5 até a 12 que tem menor
soma)

tabu(1:n,1:n)=0;             % matriz tabu: o programa constrói uma matriz
nula nxn

ge=0;                        %Geracao estagnada
proibicao=max;
                             %for iteracao=1:10 %criterio de parada 10
iterações.
for iteracao=1:tmax;         %criterio de parada tmax de iterações dado pelo
usuário.

P=[];                        %Matriz das soluções (vizinhos de p) vazia por
enquanto;
fitness=[];                  %Queremos construir aqui uma matriz com 4
colunas e tantas linhas quantas necessárias, sendo que terceira e quarta
coluna ficará os atributos trocados;

contador=0;
       for i=2:(n-1)         % para i variando de 2 até (n-1)
            p1=p(i);         %O elemento p(i) é fixado, por exemplo se i=2,
fixa-se p(2);
            for j=(i+1):n

                contador=contador+1;
                fitness(contador,3)=i; %Coloca na linha contador e coluna
3 o i que foi trocado com o j na solução anterior e j será guardado na
linha contador
                fitness(contador,4)=j; % e coluna 4;

                pt=p; %Toma-se, pt=p vetor solução do momento, pt é um
vetor temporário (solução criada no momento através da permutação de dois
elementos de p;
                p2=p(j); %Para j=i+1 faça p2=p(j), isto é p(j) é o
elemento do vetor solução que está na posição j;
                pt(i)=p2; %p(i)=pt(i)=p2, i fixo é trocado primeiro com
                pt(j)=p1; %p1=pt(j)=p(i+1), depois com p1=pt(j)=p(i+2),
p1=pt(j)=p(i+3) e assim por diante até ser cambiado por fim com
p1=pt(j)=p(n);
```

```
                P=[P;pt]; % Uma vez criados todos vetores de permutações
com i fixo, atualiza-se a matriz P acrescentando nela esses vetores como
novas
             end           %novas linhas e volta ao for maior para fixar
agora (i+1) e permutar p(i+1) com p(j)=p(i+2);
       end

nv=size(P,1); %é o número de linhas da matriz P;
soma=0; %soma inicial igual a zero;

       for i=1:nv %para i fixo entre 1 e nv;
            for j=1:(n-1) %para jo variando de 1 a (n-1);
                soma=soma+D(P(i,j),P(i,j+1)); %faz-se a soma inicial mais a
distância das cidades uma por uma até a última;
            end
            soma=soma+D(P(i,1),P(i,n));% soma anterior mais a distância da
última cidade até a primeira;
            fitness(i,1)=i; %Vai construir uma coluna de i até nv. Por
exemplo neste problema se n=10, nv=36;
            fitness(i,2)=soma; % constroi a segunda coluna da matriz
fitness com os resultados das somas dos elementos das linhas da matriz P;
            soma=0;
       end

z=sortrows(fitness,2); %coloca os elementos da matriz fitness em ordem
crescente com base na coluna 2;

       if z(1,2)<incumbente(1,2) %Se o valor que estiver na primeira linha
e na segunda coluna for menor que o valor objetivo da solução anterior;
            incumbente(1,1)=iteracao; %A incumbente passa a ter função
objetivo igual a z(1,2) e a solução corrente passa a ser p=P(z(1,1),:);
            incumbente(1,2)=z(1,2);
            incumbente(1,3:(n+2))=P(z(1,1),:);
            ge=0; %Geracao estagnada igual a zero;
            proibicao=max; %Máximo de iterações proibidas;
            else
            ge=ge+1; %Geração estagnadas +1 ( Aqui abre contagem no sentido
de aumentar a proibição a partir de um número de estagnação para incentivar
diversificação);

            end

if ge>50 % Se geração da incumbente está estagnada a 50 iterações então
aumenta-se a proibição para diversificar;
    proibicao=2*max; %Número de proibição após a estagnação;
end

for i=1:nv
    if tabu(z(i,3),z(i,4))==0 % aqui se o elemento que está na linha z(i,3)
e coluna z(i,4) da matriz tabu forem iguais a zero, p será dado pela
primeira
                              %linha da matriz P
        p=P(z(i,1),:); % é a melhor solução, que está na linha 1 da matriz
P
        %fprintf('nao estava proibido em i=')
        %i
        for ii=1:n
            for jj=1:n
                if tabu(ii,jj)>0
```

```
                    tabu(ii,jj)=tabu(ii,jj)-1; %Este for é para atualizar a
proibição, a cada nova iteração a proibição diminui 1 até zerar e deixar
                end                                 %o atributo livre para
transição;
            end
        end

        tabu(z(i,3),z(i,4))=proibicao;

        break %Se p for dado neste se, pare;
    else %Caso contrário,

        if z(i,2)<incumbente(1,2) %Se a solução proibida é melhor que a
incumbente, esquece-se a proibição e passa para ela (cumpre-se o critério
de aspiração);
            p=P(z(i,1),:);% é a melhor solução, que está na linha 1 da
matriz P
        %fprintf('cumpriu aspiração em i=')
        %i
            for ii=1:n
                for jj=1:n
                    if tabu(ii,jj)>0
                        tabu(ii,jj)=tabu(ii,jj)-1; %Este for é para
atualizar a proibição, a cada nova iteração a proibição diminui 1 até zerar
e deixar
                    end                                 %o atributo livre para
transição;
                end
            end

            tabu(z(i,3),z(i,4))=proibicao;
            break
        end
    end


end

end

tempo=toc;

fprintf(1, '..............RESULTADO ENCONTRADO.....................\n\n');
fprintf(1, 'Número de iterações:.............................
%6.8f\n',incumbente(1,1));
fprintf(1, 'Tempo decorrido para gerar pop. inicial:.........
%6.4f\n',tempo);
fprintf(1, 'Valor da incumbente:.............................
%6.8f\n',incumbente(1,2));
fprintf(1, 'Incumbente:.............................
%6.8f\n',incumbente(1,3:(n+2)));
```

PROBLEMA EXEMPLO RESOLVIDO COM A META HEURÍSTICA BUSCA TABU

```
# Parâmetros TABU
s_0 = [0,0]
T=[ ]
BT_max = 4
T_ = 3
it_max = 100 # máximo de iterações

# Função objetivo
def funcao_objetivo (s):
  return 800*s[0] + 1000*s[1]


# Função que verifica a validade do solução
def valido( s ):
  return s[0] >= 0 and s[1] >= 0 and s[0] + s[1] <= 5 and s[0] + 2*s[1]
 <= 8 and s[0]<=4


# Função que retorna o conjunto da vizinhança
def vizinhos( s ):
  return [
          [ s[0]   , s[1] -1 ],
          [ s[0]   , s[1] +1 ],
          [ s[0] -1, s[1]    ],
          [ s[0] +1, s[1]    ]
  ]
# Função que verifica se a solução é tabu
def e_tabu(s):
  return s in T

def add_tabu(s):
  if len(T) > T_:
    T.pop()
  T.insert(0, s)

def aspiracao( s, s_star ):
  return e_tabu(s) and funcao_objetivo(s) <  funcao_objetivo (s_star )

#primeiramente tenta buscar o melhor vizinho valido não tabu
#se ele não achar nenhum a função busca de novo com o criterio de
def maior_vizinho( s, s_star):
  for s_ in vizinhos( s ):
    if valido(s_) and (not e_tabu(s_) ):
      try:
        if  funcao_objetivo(s_) > funcao_objetivo(max) :
          max = s_
```

```
      except:
        max = s_
  #aspiração:
  try:
    return max
  except:
    #print("Aspiração")
    for s_ in vizinhos( s ):
      if valido(s_) and aspiracao(s_, s_star):

        try:
          if  funcao_objetivo(s_) > funcao_objetivo(max) :
            max = s_
        except:
          max = s_
    return max


def busca_tabu ():
  it = melhor_it = 0
  s_star = s = s_0
  while  (it - melhor_it ) < BT_max   :
    it += 1
    add_tabu(s)
    s = maior_vizinho( s, s_star )
    if funcao_objetivo(s) > funcao_objetivo(s_star) :
      s_star = s
      melhor_it = it
    print([it, s ])

  return [melhor_it, it, s_star]

# Apresentado o resultado

bt = busca_tabu()
print('Solução--------')
print('Iterações: {0}'.format( bt[1])  )
print('Melhor it: {0}'.format(bt[0]) )
print('Melhor Solução: {0}'.format(bt[2]))
print('função objetivo: {0}'.format( funcao_objetivo(bt[2]) ) )


Resultado:

Solução--------
Melhor Solução: [2, 3]
função objetivo: 4600
```