2016: DISSERTAÇÃO DE MESTRADO
Mestrado Profissional em Matemática - PROFMAT, Universidade Federal de São João Del-Rei - UFSJ
Sociedade Brasileira de Matemática - SBM

Algoritmos em Grafos e o Problema do Caixeiro Viajante:
uma abordagem no Ensino Médio utilizando planilhas eletrônicas.

Anderson Oliveira Viana [1]
Alexandre Celestino Leite Almeida[2]

Resumo: Este trabalho apresenta uma visão geral e básica sobre Grafos e uma abordagem especial de dois problemas relacionados a esse tema da Matemática: o Problema do Caminho Mais Curto e o Problema do Caixeiro Viajante. O Problema do Caminho Mais Curto, pelo grande número de aplicações práticas, mereceu um destaque especial neste texto. Dois algoritmos resolutivos do problema, o Algoritmo de Dijkstra e o Algoritmo de Floyd-Warshall, são apresentados e modelos são implementados em planilhas eletrônicas. O Problema do Caixeiro Viajante (PCV), um dos problemas mais conhecidos e estudados na Otimização Combinatória, é caracterizado e variantes são apresentadas. Modelos do problema na versão clássica e da variante PCV com Coleta de Prêmios são implementados em planilhas eletrônicas. Na seção final do trabalho, apresentamos atividades que podem ser ministradas por professores de Matemática em aulas sobre o tema Grafos em turmas do ensino médio.

Palavras-chave: Grafos, Problema do Caminho Mínimo, Problema do Caixeiro Viajante, Algoritmos em Grafos em planilha eletrônica, PCV em planilha eletrônica.

1 Introdução

Otimizar é empregar técnicas para selecionar as melhores alternativas para se atingir os objetivos propostos. Significa estabelecer prioridades para uma maior eficiência e eficácia em busca de obter os melhores rendimentos. Na área da Informática, otimizar um sistema é torná-lo mais rápido e eficiente, reduzindo o tempo de execução de tarefas. Em Matemática, o termo otimização, refere-se ao estudo de problemas em que se busca minimizar ou maximizar uma função através da escolha de valores de variáveis reais ou inteiras dentro de um conjunto viável.

Por muitas vezes, em sala de aula, os professores de Matemática são questionados pelos alunos quanto à aplicabilidade e necessidade de aprender determinados conceitos matemáticos. Quando falamos em *otimização*, a Administração, a Engenharia, a Logística, a Biologia, por exemplo, fazem uso de modelos matemáticos para representar problemas

[1]Aluno de Mestrado do PROFMAT, Turma 2014, Universidade Federal de São João Del-Rei - UFSJ, andersonoviana@hotmail.com
[2]Professor orientador, Departamento de Física e Matemática - DEFIM, UFSJ - CAP, celestino@ufsj.edu.br

e aplicando técnicas matemáticas chegam a resultados e soluções que contribuem para a melhoria de algum sistema relacionado a alguma dessas áreas. O índice de desempenho, o índice de performance e o índice de eficiência são calculados objetivando encontrar uma "solução ótima", isto é, aquela que resulta no melhor desempenho possível de um sistema, segundo critérios previamente definidos.

Em um problema de otimização temos uma função objetivo e um conjunto de restrições, ambos relacionados às variáveis de decisão. Os valores possíveis às variáveis de decisão são delimitados pelas restrições impostas sobre essas variáveis, formando um conjunto discreto de soluções factíveis a um problema. O problema pode ser de minimização ou de maximização da função objetivo. A resposta para o problema de otimização, será o menor (ou maior) valor possível para a função objetivo para o qual o valor atribuído às variáveis não viole nenhuma restrição.

Entre muitos problemas de Otimização Combinatória, este trabalho dedica-se a problemas que envolvem Grafos, por ser um tema com grande número de aplicações práticas e pouco presente em planejamentos curriculares e livros didáticos do ensino básico.

Na secão 2 apresentamos os conceitos básicos sobre grafos, tipos, caracterização e formas de representação. A seção 3 aborda o Problema do Caminho Mais Curto ou Problema do Caminho Mínimo em grafos. Ainda na seção 3, dois algoritmos são apresentados e modelos são implementados numa planilha eletrônica. Na seção 4 apresentamos o Problema do Caixeiro Viajante (PCV), algumas de suas variantes, formulação matemática, métodos de resolução e também implementamos dois modelos do PCV em planilhas eletrônicas. Na seção 5, sugerimos algumas atividades que podem ser ministradas em aulas que abordem os temas aqui apresentados. Neste trabalho utilizamos uma linguagem de fácil compreensão, o que torna esse texto um singelo guia para professores que desejam trabalhar esses temas no ensino básico.

2 Grafos

Definimos grafo como o par de conjuntos (V, A) onde $V = \{v_1, v_2, ..., v_n\}$ é um conjunto de vértices ou nós e $A = \{(v_i, v_j) \mid v_i, v_j \in V\}$ é um conjunto de arestas ou arcos. Uma representação geométrica de um grafo pode ser associar os vértices a pontos e as arestas a linhas que ligam os pares de vértices que as formam. Por exemplo, podemos construir um grafo que represente pessoas apertando as mãos numa reunião. Os vértices representam as pessoas e, se duas pessoas se cumprimentam, ligamos os vértices que as representam, formando assim uma aresta (ou vértice). (figura 1).

Além da notação (v_i, v_j), outra notação admitida para representar uma aresta que liga o vértice i ao vértice j é a_{ij} ou simplesmente a_j, quando se tratar de grafos não direcionados (constituído de arestas não orientadas).

A figura 2 mostra dois exemplos de grafos: o grafo G_1 consiste dos conjuntos $V = \{v_1, v_2, v_3, v_4, v_5\}$ e $A = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_3, v_4), (v_3, v_5), (v_4, v_5)\}$; G_2 possui o vértice v_4 que não é conectado com nenhum outro vértice do grafo. Nesse caso, o vértice é chamado de *isolado*.

Dois vértices i, j são vizinhos se eles estão conectados por uma aresta , e representamos

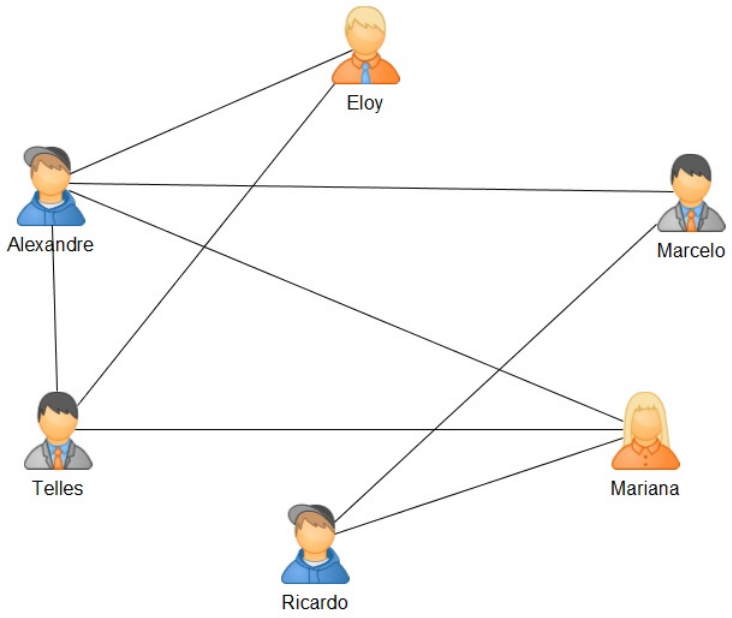


Figura 1: Grafo representando o cumprimento de pessoas através do aperto de mãos.

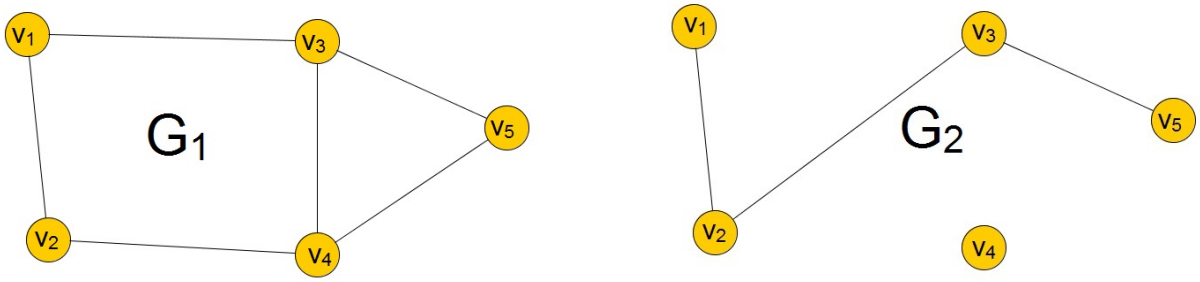


Figura 2: Exemplos de grafos.

por $i \sim j$. Nesse caso, dizemos que os vértices são *adjacentes* e que a aresta é incidente aos vértices. O número de vezes que as arestas incidem sobre o vértice v é chamado *grau do vértice* v, simbolizado por $d(v)$. Por exemplo, nos grafos da figura 2, em G_1 temos que $d(v_3) = 3$, $d(v_5) = 2$ e em G_2, $d(v_4) = 0$. Um grafo é dito *regular* de grau r, quando todos os seus vértices possuem o mesmo grau.

Uma aresta pode ligar um vértice a ele mesmo. É o que chamamos de *laço* ou *loop*. Neste caso a aresta é do tipo (v_i, v_i), formada por um par de vértices idênticos (figura 3). Dois vértices podem estar ligados por mais de uma aresta. Neste caso, chamamos o grafo de *multigrafo* (figura 4). Grafos sem laços ou arestas múltiplas são chamados de *grafos simples*.

Um *percurso* ou *passeio* em um grafo é um conjunto de vértices (ou de ligações) sequencialmente adjacentes. Ou seja, um primeiro vértice é adjacente ao segundo, que é adjacente ao terceiro, e assim sucessivamente até chegar ao último. Um percurso é dito *fechado* se o primeiro vértice da sequência for também o último. Caso contrário, será

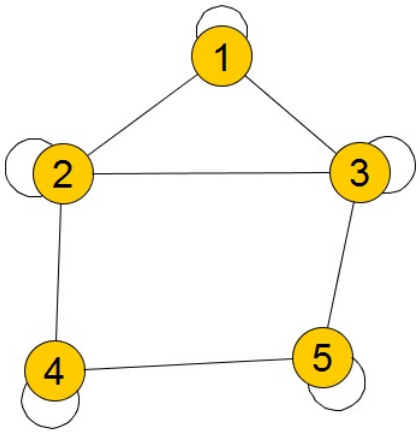


Figura 3: Grafo com laço ou loop.

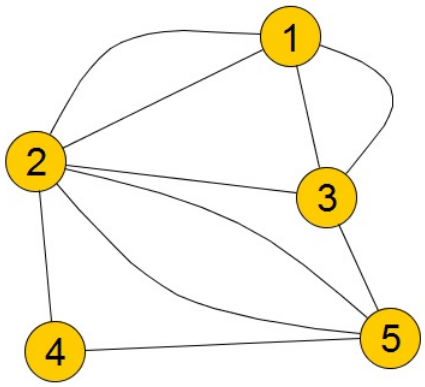


Figura 4: Multigrafo.

denominado *aberto*.

O percurso sem repetição de arestas é chamado de *cadeia*. *Caminho* é uma cadeia sem repetição de vértices. O caminho de v_1 a v_k é uma sequência de vértices $v_1, ..., v_k$ tal que $(v_j, v_{j+1}) \in A, 1 \leq j <\mid k-1 \mid$ ([26]). Um caminho de k vértices é formado por $k-1$ arestas $(v_1, v_2), (v_2, v_3), ..., (v_{k-1}, v_k)$. Um *ciclo* é um caminho $v_1, ..., v_k, v_{k+1}$, sendo $v_1 = v_{k+1}$ e $k \geq 3$.

Um caminho que contenha cada vértice do grafo exatamente uma vez é chamado de *hamiltoniano*. Um ciclo $v_1, ..., v_k, v_{k+1}$ é hamiltoniano quando o caminho $v_1, ..., v_k$ o for. O termo *hamiltoniano* é devido a Willian Rowan Hamilton que, em 1857, propôs um jogo que denominou *A Voyage Around the World* (Uma viagem ao redor do mundo). O objetivo do jogo era encontrar um roteiro de viagem passando por cada cidade exatamente uma vez, iniciando e terminando numa mesma cidade e para tornar o problema mais interessante, as cinco primeiras cidades a serem visitadas eram pré-estabelecidas. O roteiro era composto pelas vinte cidades mais importantes da época, representadas pelos vértices de um dodecaedro. O grafo que representa o jogo é mostrado na figura 5. Uma solução de *A Voyage Around the World* passou a ser denominado de ciclo hamiltoniano, em homenagem a Hamilton. Uma das soluções do jogo é mostrada na figura 6. Mais adiante, apresentamos uma seção dedicada a um dos problemas mais conhecidos que envolvem grafos, o Problema do Caixeiro Viajante, cujo desafio na sua versão clássica é encontrar um ciclo hamiltoniano de custo mínimo em um grafo ponderado.

Por outro lado, algum caminho ou ciclo que contenha cada aresta do grafo, também exatamente uma vez cada, é denominado *euleriano*. O termo *euleriano* é devido a Leonhard Euler. Em 1736, Euler visitou a cidade de Königsberg (atualmente Kaliningrado na Rússia) que, na época, possuía uma ilha rodeada pelo rio Prega com 7 pontes (figura 7). Euler tomou conhecimento de um problema que intrigava os habitantes da cidade e que ficou conhecido como o Problema das Pontes de Königsberg: existia um modo de atravessar as 7 pontes, sem passar duas vezes na mesma ponte, retornando ao ponto de partida? O problema dá origem a um multigrafo e Euler mostrou que não é possível realizar tal percurso.

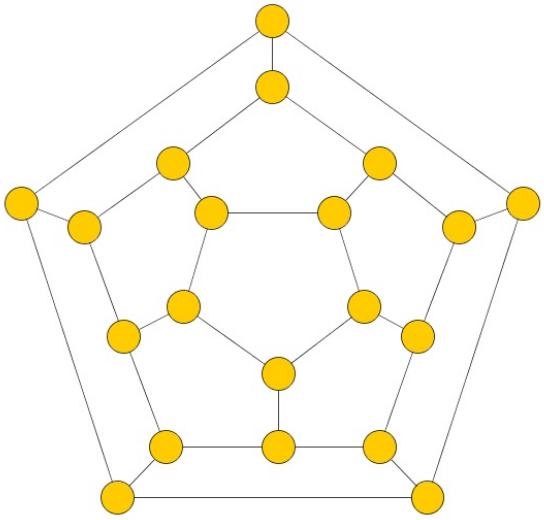


Figura 5: Jogo de Hamilton

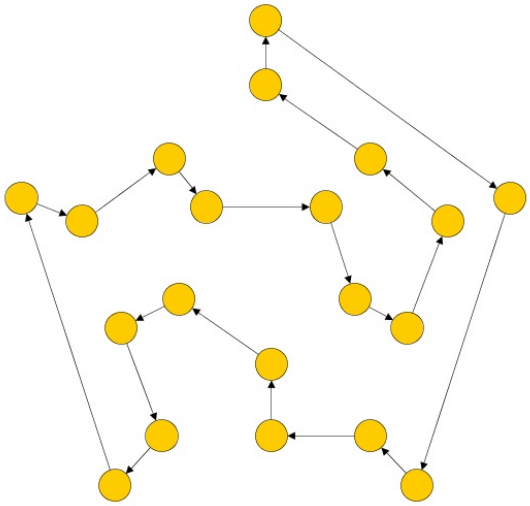


Figura 6: Uma solução do Jogo de Hamilton

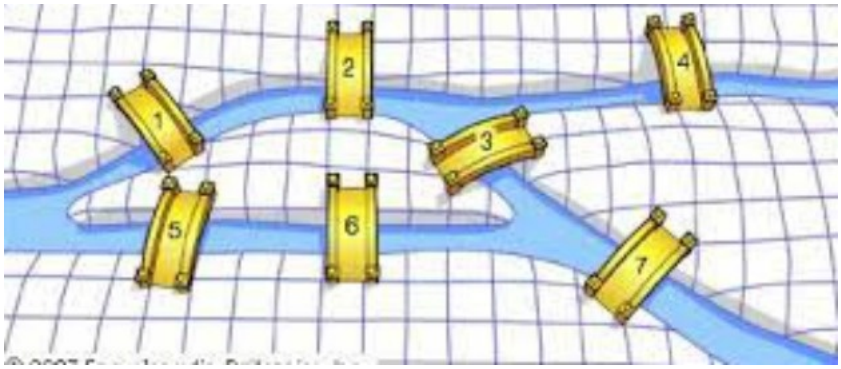


Figura 7: Problema das Pontes de Königsberg.

Denominamos simplesmente grafo hamiltoniano ou grafo euleriano quando um grafo G possui ciclo hamiltoniano ou euleriano, respectivamente.

Um grafo $G(V, A)$ é denominado *conexo* quando existe caminho entre cada par de vértices de G. Caso contrário G é *desconexo*. No grafo desconexo da figura 8, por exemplo, não podemos estabelecer um caminho de 1 a 7. Um grafo G que não possui arestas é considerado *totalmente desconexo*.

Um grafo é dito *direcionado* ou *orientado* ou *dirigido* ou *dígrafo* quando é necessário ser estabelecido um sentido (orientação) para as arestas. O sentido da aresta é indicado através de uma seta, como ilustrado na figura 9. Se a seta "*sai*" de v_i e "*chega*" a v_j, dizemos que a aresta é divergente de v_i e convergente a v_j.

Um grafo G é denominado *completo* se existe uma aresta ligando cada par de vértices de G, como mostra o exemplo da figura 10.

Se associamos um peso ou conjunto de pesos a cada aresta, temos um grafo *ponderado* ou *valorado*. O peso ou custo de uma aresta é representado por w_{ij}, ou seja, w_{35} é o peso

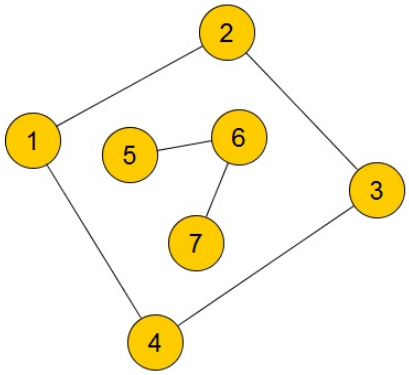


Figura 8: Um grafo desconexo.

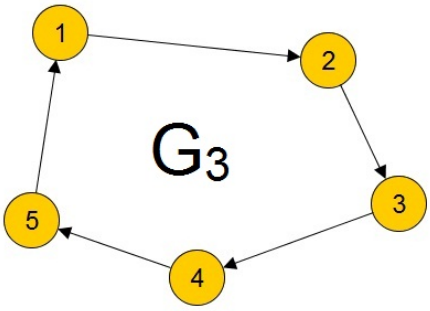


Figura 9: G_3 Grafo direcionado.

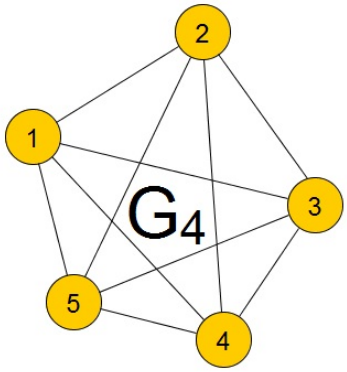


Figura 10: G_4 Grafo completo.

associado a aresta que une os vértices 3 e 5. Podemos também incluir o peso das arestas em um conjunto P. Dessa forma o grafo G é definido por $G = (V, A, P)$.

Um *subgrafo* de um grafo $G = (V, A)$ é o grafo $H = (V', A')$ tal que $V' \subseteq V$ e $A' \subseteq A$. Podemos representar na forma $H \subseteq G$ e dizer que G contém H. Na figura 11, os grafos G_6 e G_7 são subgrafos do grafo G_5.

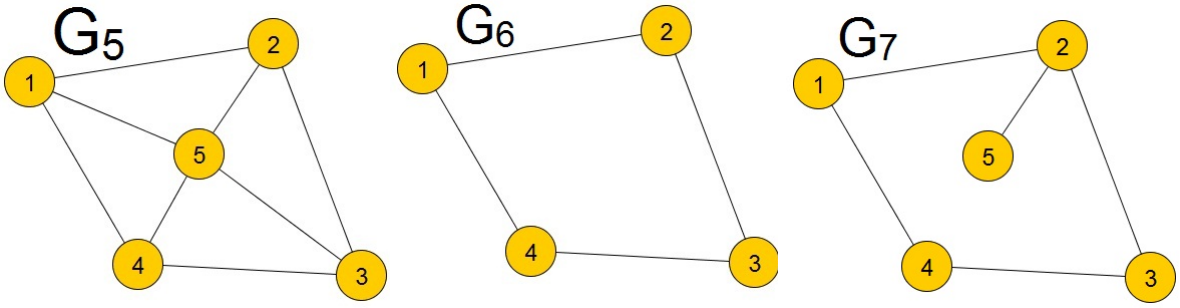


Figura 11: Um grafo G_5 com dois subgrafos, G_6 e G_7.

2.1 Representação de Grafos

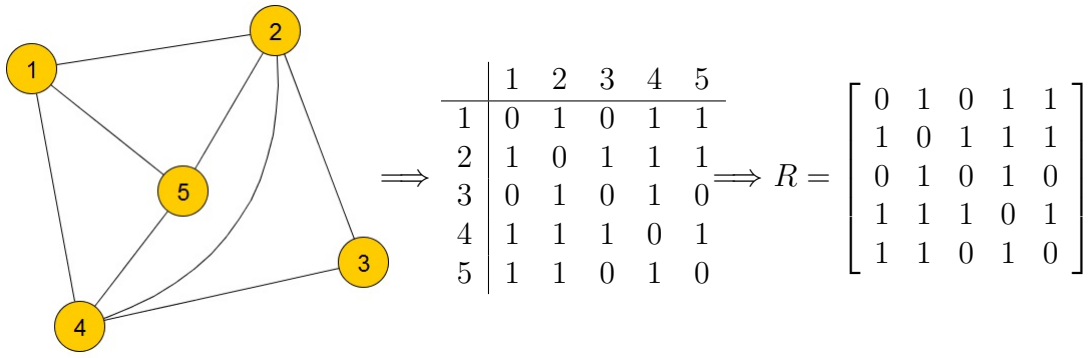
Alguns problemas que são resolvidos utilizando grafos são de complexidade elevada, tornando o computador uma ferramenta indispensável para resolvê-los. Em um computador faz-se necessário representar essas estruturas de modo que possam ser armazenadas e manipuladas sem dificuldade. A representação geométrica não satisfaz essa condição. Nesta seção, apresentaremos alguns tipos de representações de grafos adequadas ao computador.

2.1.1 Matriz de adjacência

Uma das formas mais utilizadas para representar grafos é via matriz de adjacência. Seja $R = [r_{ij}]$ uma matriz $n \times n$, onde n é o número de nós de um grafo $G = (V, A)$ simples e não direcionado, a matriz de adjacência R é construída da seguinte forma:

$$R(i,j) = \begin{cases} 1 & \text{se } i \sim j \\ 0 & \text{caso contrário.} \end{cases}$$

A figura 12 ilustra o conceito de matriz de adjacência para um grafo não direcionado.



	1	2	3	4	5
1	0	1	0	1	1
2	1	0	1	1	1
3	0	1	0	1	0
4	1	1	1	0	1
5	1	1	0	1	0

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Figura 12: Um grafo não direcionado e sua matriz de adjacência.

Para grafos direcionados, a matriz de adjacência R é construída da seguinte forma:

$$R(i,j) = \begin{cases} 1 & \text{se } (v_i, v_j) \text{ for aresta divergente de } v_i \text{ e convergente a } v_j \\ 0 & \text{caso contrário.} \end{cases}$$

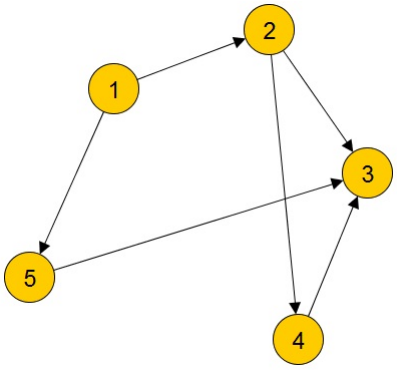
A figura 13 mostra o conceito de matriz de adjacência para um grafo direcionado.

Quando o grafo é ponderado, matriz de adjacência é construída com os valores dos pesos associados às arestas. Seja w_{ij} o peso associado à aresta (i, j), a matriz de adjacência R de um grafo ponderado é construída da seguinte forma:

$$R(i,j) = \begin{cases} w_{ij} & \text{se } i \sim j \\ 0 & \text{caso contrário.} \end{cases}$$

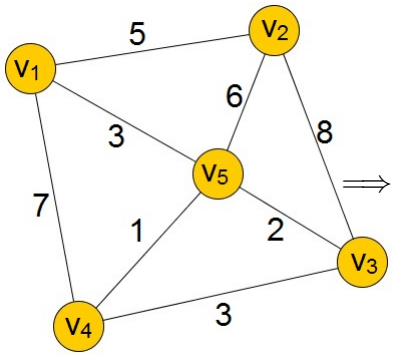
A figura 14 mostra o conceito de matriz de adjacência para um grafo ponderado.

Quando se tratar de grafo direcionado ponderado, basta atribuir sinais aos pesos das arestas, observando a convergência ou divergência da aresta no vértice.



$$\Longrightarrow \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 \end{array} \Longrightarrow R = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figura 13: Um grafo direcionado e sua matriz de adjacência.



$$\Longrightarrow \begin{array}{c|ccccc} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \hline v_1 & 0 & 5 & 0 & 7 & 3 \\ v_2 & 5 & 0 & 8 & 0 & 6 \\ v_3 & 0 & 8 & 0 & 3 & 2 \\ v_4 & 7 & 0 & 3 & 0 & 1 \\ v_5 & 3 & 6 & 2 & 1 & 0 \end{array} \Longrightarrow R = \begin{bmatrix} 0 & 5 & 0 & 7 & 3 \\ 5 & 0 & 8 & 0 & 6 \\ 0 & 8 & 0 & 3 & 2 \\ 7 & 0 & 3 & 0 & 1 \\ 3 & 6 & 2 & 1 & 0 \end{bmatrix}$$

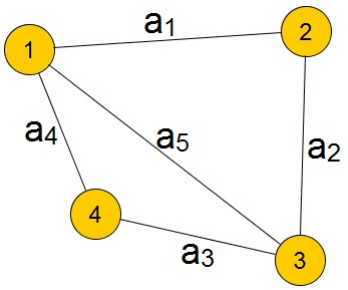
Figura 14: Um grafo ponderado e sua matriz de adjacência.

2.1.2 Matriz de incidência

A matriz de incidência $B = [b_{ij}]$ de um grafo $G = (V, A)$ não direcionado, com $V = (v_1, v_2, ..., v_n)$ e $A = (a_1, a_2, ..., a_m)$, de dimensão $m \times n$, onde n é o número de vértices e m é o número arestas do grafo, é definida da seguinte forma:

$$B(i,j) = \begin{cases} 1 & \text{se } v_j \in a_i \\ 0 & \text{caso contrário} \end{cases}$$

A figura 15 mostra um grafo não direcionado e sua matriz de incidência.



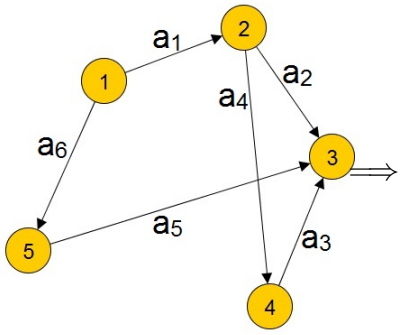
$$\Longrightarrow \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline a_1 & 1 & 1 & 0 & 0 \\ a_2 & 0 & 1 & 1 & 0 \\ a_3 & 0 & 0 & 1 & 1 \\ a_4 & 1 & 0 & 0 & 1 \\ a_5 & 1 & 0 & 1 & 0 \end{array} \Longrightarrow B = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Figura 15: Um grafo não direcionado e sua matriz de incidência.

Para grafos direcionados (dígrafos), é preciso observar o sentido do caminho entre os nós e adotar um padrão para o sinal dos 1's. Por exemplo, podemos construir a matriz de incidência B de um dígrafo da seguinte forma:

$$B(i,j) = \begin{cases} +1 & \text{se a aresta } a_i \text{ for divergente de } v_j \\ -1 & \text{se a aresta } a_i \text{ for convergente a } v_j \\ 0 & \text{se a aresta } a_i \text{ não incide no vértice } v_j. \end{cases}$$

A figura 16 mostra o conceito de matriz de incidência para o grafo direcionado da figura 13.



	1	2	3	4	5
a_1	1	-1	0	0	0
a_2	0	1	-1	0	0
a_3	0	0	-1	1	0
a_4	0	1	0	-1	0
a_5	0	0	-1	0	1
a_6	1	0	0	0	-1

$$\Longrightarrow B = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Figura 16: Matriz de incidência do grafo direcionado da figura 13.

Para grafos ponderados, devemos adotar o mesmo procedimento no que diz respeito à escolha de sinais para representar os arcos e seus pesos.

2.1.3 Lista de Adjacência

Há muitas representações de grafos através de listas. A mais comum é a *lista de adjacência*. De acordo com [26], a lista é uma representação que favorece a visualização de informações dos grafos, basicamente por não possuir informações de não adjacência (os zeros na matriz de adjacência). Seja $G = (V, A)$ um grafo, a lista de adjacências L de G é um conjunto de n listas $L(v)$, uma para cada $v \in V$, e que contém os vértices w adjacentes a v em G. A figura 17 traz um exemplo de uma lista de adjacências do grafo não direcionado da figura 12.

Para grafos direcionados, a lista de adjacências é um conjunto de n listas $L(v)$, uma para cada $v \in V$, e que contém os vértices w divergentes de v em G. Se o grafo for ponderado, a informação do peso da aresta que liga v a w deve ser armazenada em uma outra lista.

Na próxima seção apresentamos o Problema do Caminho Mais Curto ou Caminho Mínimo, um dos problemas mais estudados na Teoria dos Grafos.

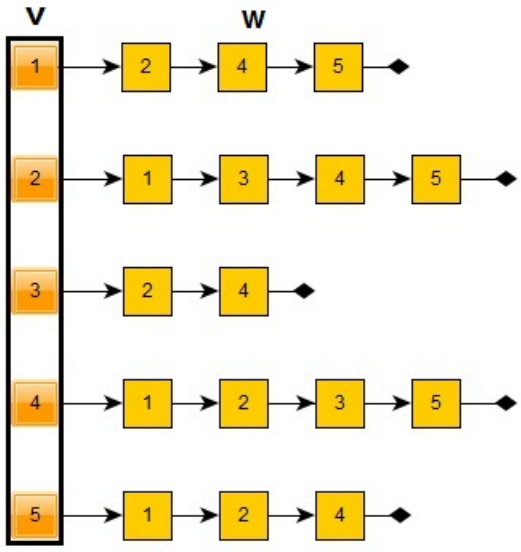


Figura 17: Lista de adjacência do grafo não direcionado da figura 12.

3 O Problema do Caminho Mais Curto ou Caminho Mínimo

O Problema do Caminho Mínimo ou Caminho mais Curto é um dos mais importantes problemas que envolvem grafos, por ter muitas aplicações práticas. Uma das aplicações é a determinação, por parte de aparelhos popularmente chamados de GPS, da melhor rota ou rota mais rápida entre uma localização atual e um destino especificado. O GPS trabalha com um grafo direcionado ponderado, no qual o peso das arestas representa a distância entre dois pontos, o tempo de viagem ou o custo para cumprir um determinado percurso. A figura 18 (obtida em [1]) mostra a melhor rota, calculada pelo GPS, entre as cidades mineiras de Ouro Branco e Pará de Minas.

Em um grafo não ponderado, o caminho mínimo entre os vértices v e w é o caminho que reúne o menor número de arestas utilizadas no percurso entre os referidos vértices. Em um grafo ponderado, o caminho mínimo entre os vértices v e w é o caminho cuja soma dos pesos das arestas tem o menor valor possível dentre todos os caminhos existentes entre v e w. O termo *custo* também pode ser utilizado quando nos referimos ao peso de uma aresta. Neste trabalho, focaremos no estudo do caminho mínimo em grafos ponderados.

A formulação matemática do problema do caminho mais curto em um grafo $G = (V, A)$ é descrita em [19] assim:

$$\text{Minimizar } z = \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\text{sujeito a}$$

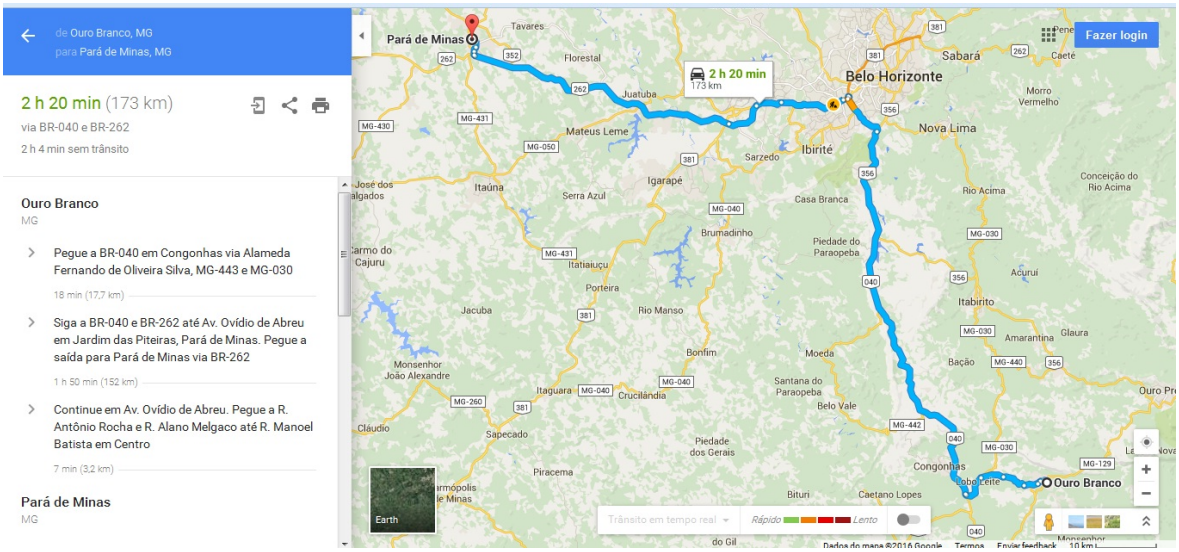


Figura 18: Uma das aplicações do Problema do Caminho Mínimo, a determinação da melhor rota em um GPS.

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(k,i)\in A} x_{ki} = \left\{ \begin{array}{ll} +1 & \text{se } i = v \\ 0 & \text{se } i \neq v \text{ e } i \neq w \\ -1 & \text{se } i = w \end{array} \right\}$$
$$x_{ij} \in \{0,1\}, \quad (i,j) \in A,$$

onde os vértices v e w representam os vértices de início e término do caminho; c_{ij} representa o custo (ou distância) associado à aresta (i,j) e x_{ij} assume o valor 1 se a aresta (i,j) é incluída no caminho mais curto e 0, caso contrário.

3.1 Algoritmos para resolução do Problema do Caminho Mínimo

Algoritmos são rotinas ou passos a serem seguidos numa certa ordem com a finalidade de resolver um problema. A partir do conhecimento acumulado ao longo dos anos, devido ao trabalho de grandes matemáticos, hoje fazemos uso de algoritmos que tornam simples e fácil a resolução de alguns problemas. Estes, nos últimos tempos, assumiram um papel importante na Ciência da Computação, por serem responsáveis pela execução de tarefas em um computador.

Abordaremos, a seguir, dois algoritmos que resolvem o Problema do Caminho Mais Curto: o algoritmo de Dijkstra e o algoritmo de Floyd-Warshall.

3.1.1 Algoritmo de Dijkstra

Em 1959, o holandês Edsger Wybe Dijkstra (1930-2002) apresentou um algoritmo que soluciona o Problema do Caminho Mínimo de uma única origem num grafo ponderado direcionado ou não direcionado com arestas de peso não negativo. A execução do algoritmo consiste basicamente na ideia de que, para chegar do vértice v ao vértice w construímos

um conjunto que contém inicialmente apenas v e, durante a execução, adicionamos a ele os vértices cujas menores distâncias de v vão sendo determinadas. Para todo vértice fora deste conjunto guardamos uma distância de v e avaliamos se esta distância foi reduzida durante a execução. No caso de redução desta distância, anotamos o vértice antecessor no caminho e incluímos o vértice analisado no conjunto. Ao final da execução, obtemos o custo ou distância total do caminho mais curto de v a w e os vértices incluídos na rota. O algoritmo faz uma visita a todos os vértices do grafo, iniciando no vértice fixo dado e encontrando sucessivamente o vértice mais próximo, o segundo mais próximo, o terceiro mais próximo e assim por diante, um por vez, até que todos os vértices do grafo tenham sido visitados.

Escolhido um vértice como raiz da busca, este algoritmo calcula o custo mínimo deste vértice para todos os demais vértices do grafo. Este algoritmo parte de uma estimativa para o custo mínimo e vai sucessivamente ajustando esta estimativa, mantendo o comprimento do menor caminho conhecido até o momento para cada vértice. Ele considera que um vértice estará fechado quando já tiver sido obtido um caminho de custo mínimo do vértice tomado com raiz da busca até ele. Caso contrário, ele é dito estar aberto ou na fila. Dessa forma, o algoritmo mantém dois conjuntos de vértices: S que é o conjunto dos vértices para os quais já determinamos o caminho mais curto (vértices fechados) e $Q = V - S$ o conjunto dos vértices que ainda estão na fila ou abertos.

Os passos do algoritmo serão apresentados a seguir. Seja $G(V, A)$ um grafo e v, k e i vértices de G:

1. Atribua zero à estimativa de custo mínimo do vértice inicial v e infinito às demais estimativas;

2. Atribua um valor qualquer aos antecessores (no caso de grafo direcionado) ou vizinhos (grafo não direcionado);

3. Enquanto houver vértice aberto:

   (a) Seja k um vértice ainda aberto cuja estimativa seja a menor dentre todos os vértices abertos;

   (b) Feche k;

   (c) Para todo vértice i ainda aberto que seja sucessor ou vizinho de k faça:

      i. Some a estimativa do vértice k com o custo da aresta (k, i);

      ii. Caso esta soma seja melhor que a estimativa anterior para o vértice i, substitua-a e anote k como precedente de i.

Cada vértice i do grafo é rotulado com um par de informações: uma é o vértice anterior ao vértice i no caminho entre a origem e o vértice i, segundo a distância calculada; a outra informação é a distância calculada pelo algoritmo entre o vértice de origem e o vértice i. No ínício, o algoritmo rotula o vértice inicial com as informações: vértice inicial e o valor zero, que é a distância do vértice inicial a ele mesmo. Os demais vértices são rotulados com o valor zero para representar o vértice anterior no caminho e infinito, representando que a distância entre eles e o vértice inicial não foi calculada ou não existe (figura 20).

Em cada iteração, o algoritmo escolhe e remove um vértice k do conjunto $Q = V - S$. O vértice k escolhido é aquele que tiver o menor valor da distância em relação ao vértice inicial, de acordo com o rótulo. O vértice é incluído no conjunto S e todos os seus vizinhos (grafos não direcionados) ou sucessores (grafos direcionados) que não estão no conjunto Q são examinados. Para cada vértice i de Q, verifica-se se a distância da origem até ele é maior que a distância da origem até k (que chamaremos de d_k) mais o valor do peso da aresta (k, i) (que chamaremos de w_{ki}). Em caso positivo, o caminho da origem até o vértice i passando pelo vértice k é menor que o caminho anteriormente encontrado entre a origem e i. Neste caso, os valores do rótulo são atualizados com:

- k, indicando que o vértice anterior a i no caminho entre a origem e i é o vértice k.
- $d_k + w_{ki}$ que representa a distância de i ao vértice inicial.

Vamos usar o algoritmo de Dijkstra para encontrar o caminho mais curto com origem no vértice v e destino final o vértice w do grafo G da figura 19.

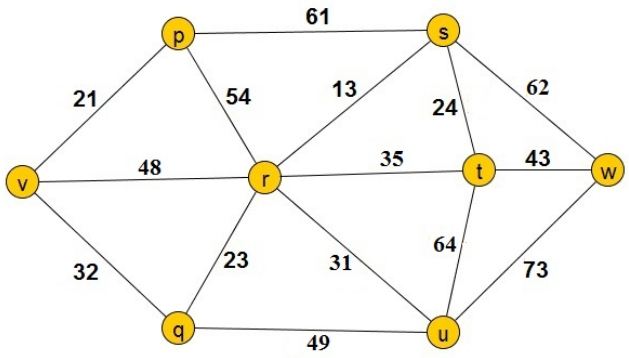


Figura 19: Grafo G

A sequência de figuras 20 a 26 mostram a evolução do algoritmo de Dijkstra no grafo da figura 19, com os custos representados nas arestas, percorrendo todos os vértices na busca do caminho mais curto entre os vértices e um ponto fixo (vértice inicial v). O algoritmo inicia a busca a partir do vértice inicial v (figura 20).

A próxima etapa é a busca do caminho mais curto desde v para se chegar aos vértices p, q, e r (figura 21).

Os vértices p, q e r são incluídos no conjunto S e a busca continua, agora com os vértices s, t e u (figura 22).

Neste caso, o caminho mais curto para se chegar a s e a u é vindo de r (figura 23).

Os vértices s, t e u são incluídos no conjunto S, pois já encontramos o seu caminho mínimo desde o vértice inicial v. A próxima busca ocorre com o vértice w, o último vértice a ser analisado, que é vértice final do caminho (figuras 24, 25 e 26). O algoritmo define que o caminho mais curto para se chegar a w é vindo de s (figura 26). Portanto, o caminho mais curto, partindo de v e chegando a w é (v, r, s, w) com peso, distância ou custo total igual a 123. Assim, ao final da execução do algoritmo, todos os vértices

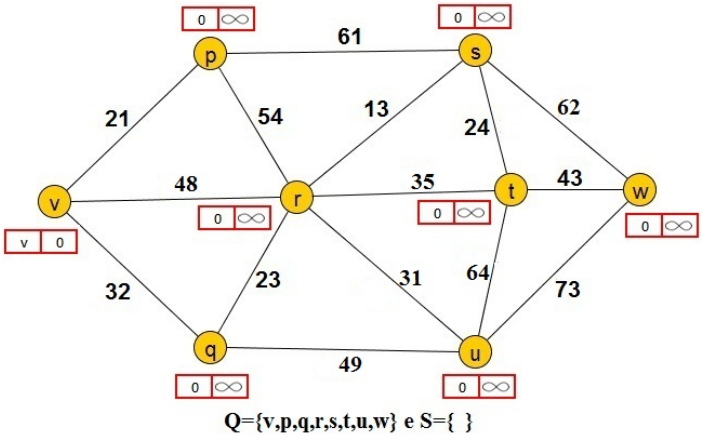


Figura 20: Rotulação inicial do Algoritmo de Dijkstra.

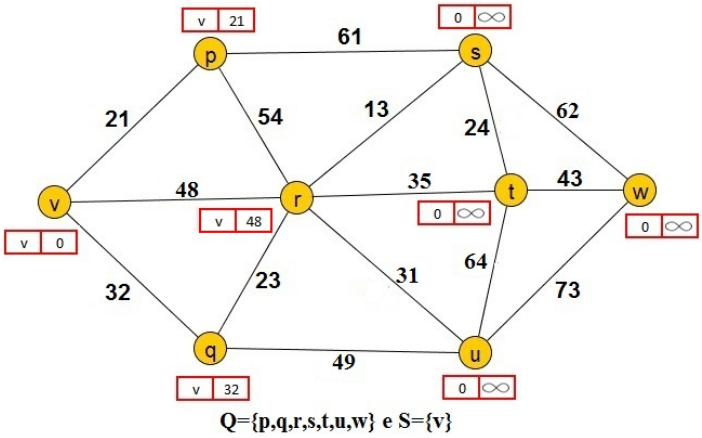


Figura 21: Busca pelo caminho mais curto para chegar a p, q e r.

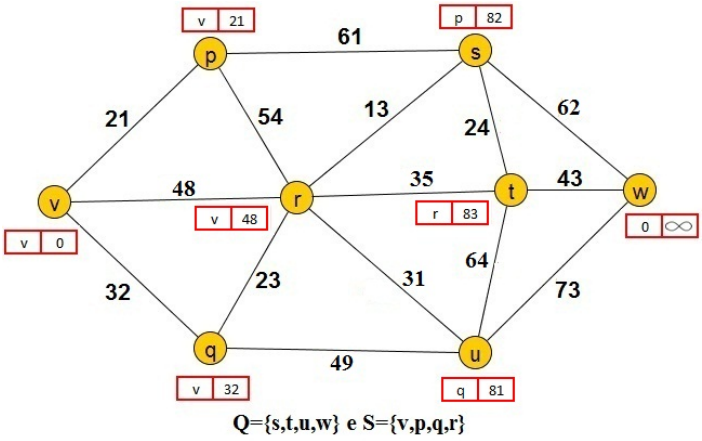


Figura 22: Busca pelo caminho mais curto para chegar a s, t e u.

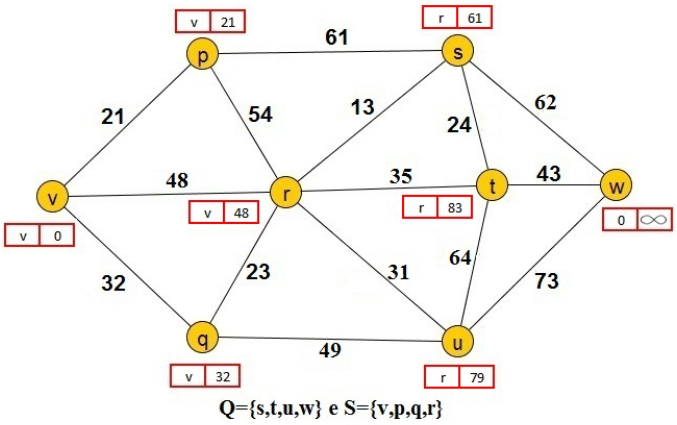


Figura 23: O caminho mais curto para chegar a s e a u é vindo de r.

estarão agrupados no conjunto S dos vértices fechados (que já tem seu caminho mínimo encontrado), e o conjunto Q dos vértices abertos estará vazio, pois não haverá mais buscas a serem realizadas.

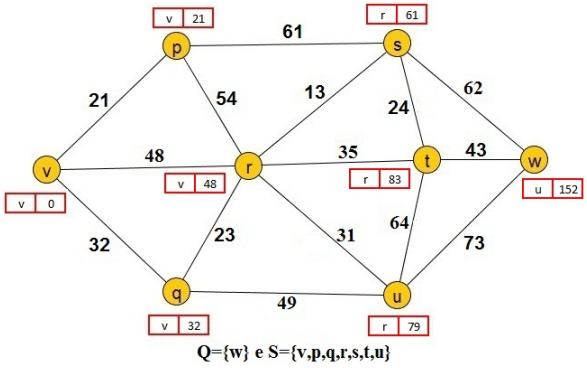


Figura 24: Busca pelo caminho mais curto para chegar a w.

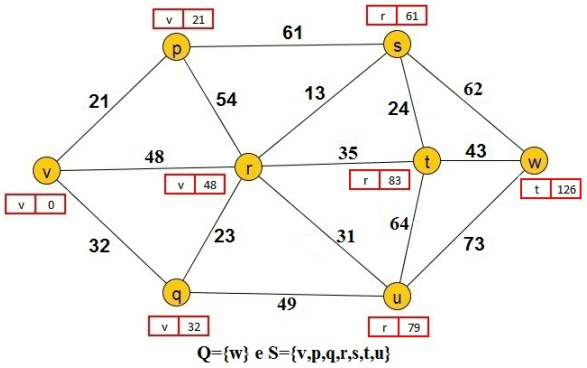


Figura 25: Busca pelo caminho mais curto para chegar a w.

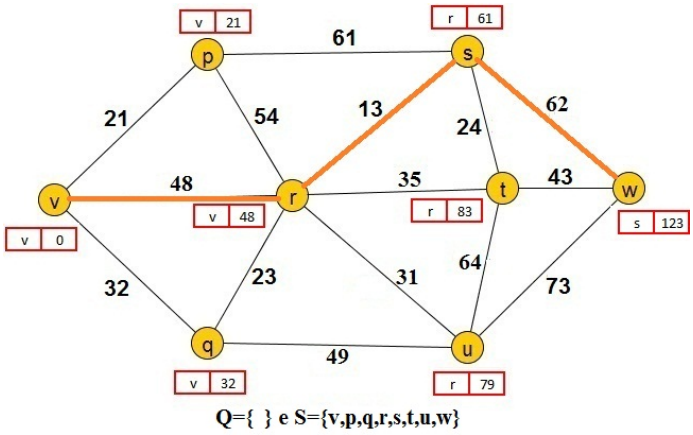


Figura 26: O caminho mais curto para chegar a w é vindo de s.

3.1.2 Implementação do Algoritmo de Dijkistra numa planilha eletrônica

Em computadores, o algoritmo de Dijkstra pode ser implementado utilizando o recurso *solver* em planilhas eletrônicas. O modelo que será exibido a seguir utilizou o suplemento *Solver* do software *Microsoft Office Excel versão 2007* para resolver o problema do caminho mínimo do grafo da figura 19. Inicialmente é necessário construir duas tabelas. A primeira com informações sobre o grafo, como arestas e seus respectivos pesos. No caso de grafos não direcionados com $c_{ij} = c_{ji}$, a tabela das distâncias deverá conter essas duas informações. A segunda tabela contém restrições do problema, conforme a figura 27.

A coluna *Caminho* da tabela da esquerda deve inicialmente ser preenchida com valor zero para todas as células amarelas, uma vez que não conhecemos as arestas que compõem o caminho mínimo. Na tabela da direta, a 3ª coluna deve ser preenchida com os valores 1 para o vértice inicial e -1 para o vértice final do caminho.

Faz-se necessário inserirmos algumas fórmulas na planilha antes de utilizarmos o solver para resolver o problema. Colocamos o cursor na célula E35 e inserimos a fórmula

$$=\text{SOMARPRODUTO(D5:D34;E5:E34)}$$

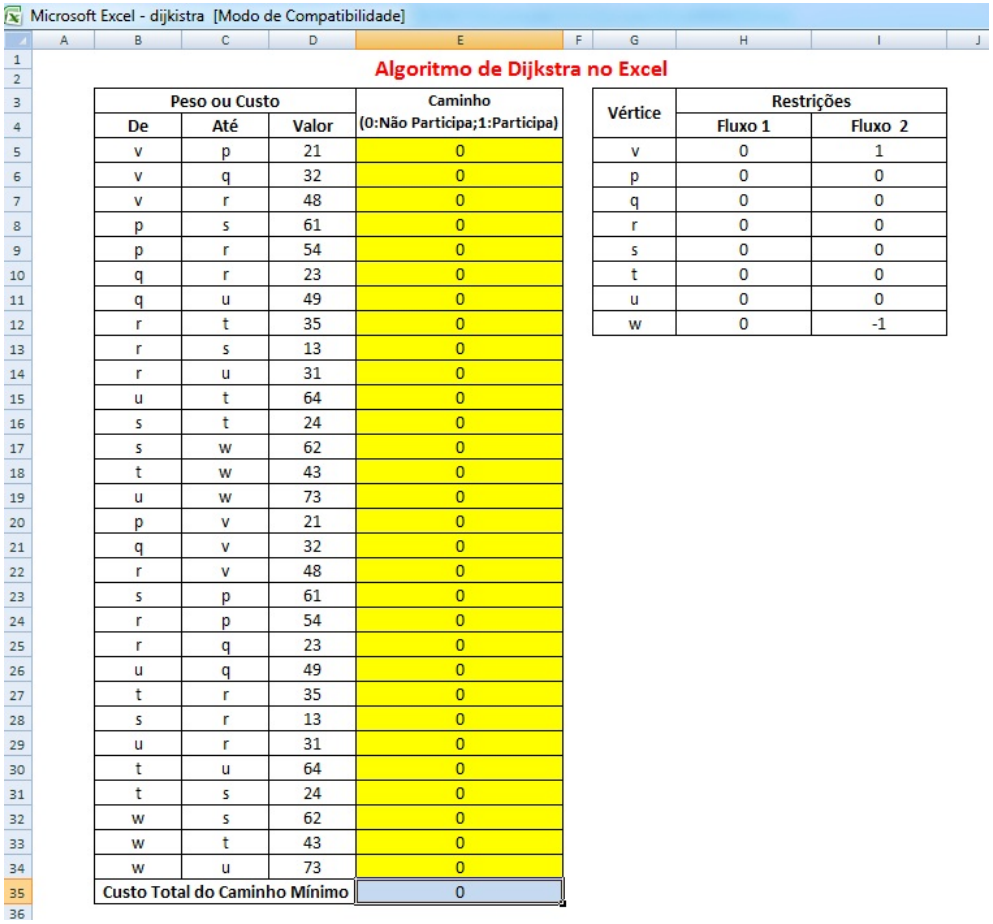
que soma os produtos dos valores do peso de cada aresta por 1, se a aresta faz parte do caminho, ou por 0 caso contrário.

Na tabela das restrições, o cursor deve ser colocado na célula H5 e deve-se inserir a fórmula

$$=\text{SOMASE(B5:B34;G5;E5:E35)}-\text{SOMASE(C5:C34;G5;E5:E34).}$$

Nas células H6 até H12, inserimos a mesma fórmula substituindo o número da linha da coluna G pelo número da linha que estamos inserindo a fórmula. Por exemplo, a fórmula da célula H12 será

$$=\text{SOMASE(B5:B34;G12;E5:E35)}-\text{SOMASE(C5:C34;G12;E5:E34).}$$



Microsoft Excel - dijkstra [Modo de Compatibilidade]

Algoritmo de Dijkstra no Excel

Peso ou Custo			Caminho
De	Até	Valor	(0:Não Participa;1:Participa)
v	p	21	0
v	q	32	0
v	r	48	0
p	s	61	0
p	r	54	0
q	r	23	0
q	u	49	0
r	t	35	0
r	s	13	0
r	u	31	0
u	t	64	0
s	t	24	0
s	w	62	0
t	w	43	0
u	w	73	0
p	v	21	0
q	v	32	0
r	v	48	0
s	p	61	0
r	p	54	0
r	q	23	0
u	q	49	0
t	r	35	0
s	r	13	0
u	r	31	0
t	u	64	0
t	s	24	0
w	s	62	0
w	t	43	0
w	u	73	0
Custo Total do Caminho Mínimo			0

Vértice	Restrições	
	Fluxo 1	Fluxo 2
v	0	1
p	0	0
q	0	0
r	0	0
s	0	0
t	0	0
u	0	0
w	0	-1

Figura 27: Tabelas do Excel com informações do grafo da figura 19.

A planilha já está preparada para a utilização do solver. Porém é necessário ativá-lo nas opções do software. Após ativá-lo podemos utilizá-lo clicando na aba *Dados* e depois procurando por *Solver*. A sequência de figuras 28 a 34 mostram a utilização do solver na resolução do Problema do Caminho Mínimo da figura 19.

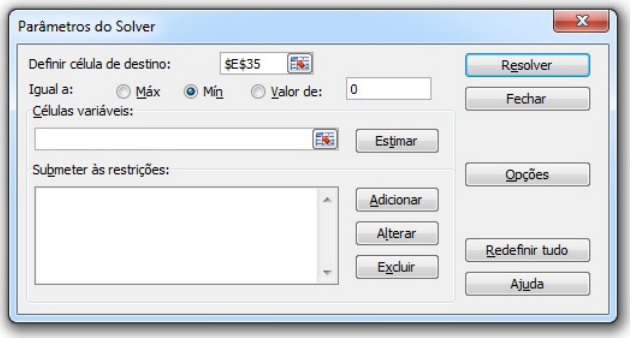


Figura 28: Parâmetros do solver: célula de destino é a célula que contém a função objetivo, o dado que queremos minimizar.

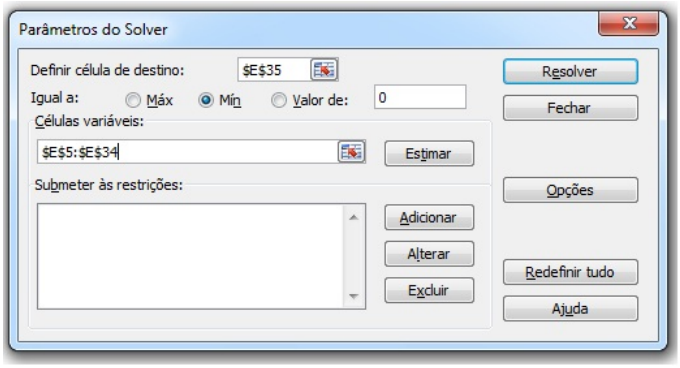


Figura 29: Parâmetros do solver: células variáveis são aquelas cujos valores serão alterados pelo solver, até que a solução do problema seja encontrada.

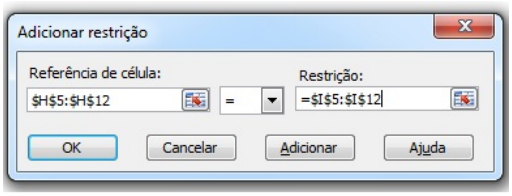


Figura 30: Parâmetros do solver: inclusão de restrições que garantem o fluxo do caminho no grafo.

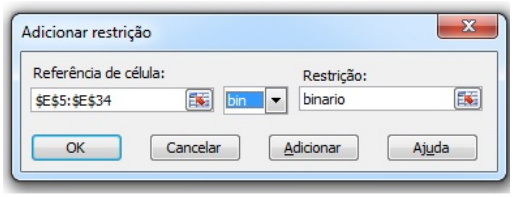


Figura 31: Parâmetros do solver: inclusão de restrição das condições binárias das variáveis.

Figura 32: Opções do solver: modelo de programação linear e condições de não-negatividade.

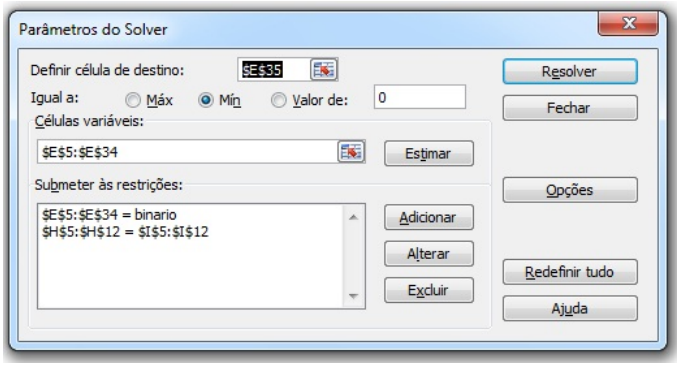


Figura 33: Parâmetros do solver. Clicando em *Resolver*, o solver mostrará a solucão do problema.

Microsoft Excel - dijkistra [Modo de Compatibilidade]

Algoritmo de Dijkstra no Excel

Peso ou Custo			Caminho
De	Até	Valor	(0:Não Participa;1:Participa)
v	p	21	0
v	q	32	0
v	r	48	1
p	s	61	0
p	r	54	0
q	r	23	0
q	u	49	0
r	t	35	0
r	s	13	1
r	u	31	0
u	t	64	0
s	t	24	0
s	w	62	1
t	w	43	0
u	w	73	0
p	v	21	0
q	v	32	0
r	v	48	0
s	p	61	0
r	p	54	0
r	q	23	0
u	q	49	0
t	r	35	0
s	r	13	0
u	r	31	0
t	u	64	0
t	s	24	0
w	s	62	0
w	t	43	0
w	u	73	0
Custo Total do Caminho Mínimo			123

Vértice	Restrições	
	Fluxo 1	Fluxo 2
v	1	1
p	0	0
q	0	0
r	0	0
s	0	0
t	0	0
u	0	0
w	-1	-1

Figura 34: Solução no Excel do Problema do Caminho Mínimo da figura 19.

3.1.3 Algoritmo de Floyd-Warshall

Baseado num algoritmo apresentado por Stephen Warshall para o cálculo de fechos transitivos em um grafo, o cientista norte-americano Robert W Floyd apresentou (vide

[16]), em 1962, um algoritmo que resolve o Problema do Caminho mais Curto entre pares de vértices de um grafo ponderado $G = (V, A)$. Importante salientar que este algoritmo encontra apenas os valores de tais caminhos, e não a sequência de arestas a ser percorrida. Em relação ao algoritmo de Dijkstra, apresenta as vantagens de admitir arestas de peso negativo e a entrada e saída do algoritmo é uma matriz.

A entrada do algoritmo é uma matriz de adjacência $R = [r_{ij}]$ construída assim:

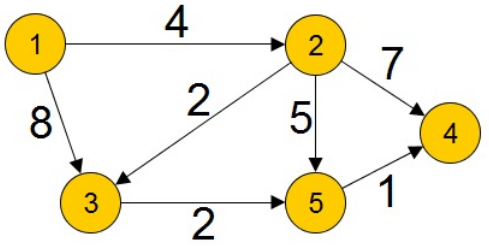
$$R(i,j) = \begin{cases} 0 & \text{se } i = j \\ w_{ij} & \text{se } i \neq j \text{ e } (i,j) \in A \\ \infty & \text{se } i \neq j \text{ e } (i,j) \notin A \end{cases} \quad (1)$$

onde w_{ij} é o peso, custo ou distância associada à aresta (i, j). O algoritmo de Floyd-Warshall compara os caminhos entre os vértices i e j passando por k vértices intermediários, $k = 1, ..., n$. No algoritmo são feitas n iterações que corresponde ao número de vértices do grafo. A cada iteração corresponde uma matriz $n \times n$ cujos valores são modificados utilizando a fórmula de recorrência:

$$w_{ij}^k = \min\{w_{ij}^{k-1}, (w_{ik}^{k-1} + w_{kj}^{k-1})\},$$

onde w_{ij}^k é o peso do caminho mais curto entre os vértices i e j na $k-$ésima matriz de iteração.

O grafo da figura 35 será utilizado como exemplo para a execução do algoritmo de Floyd-Warshall. A matriz R_0 inicial é mostrada ao lado do grafo.



$$\Longrightarrow R_0 = \begin{bmatrix} 0 & 4 & 8 & \infty & \infty \\ \infty & 0 & 2 & 7 & 5 \\ \infty & \infty & 0 & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Figura 35: Grafo ponderado para execução do algoritmo de Floyd-Warshall.

1ª iteração: o vértice 1 é considerado intermediário no caminho entre os vértices i e j.
$k = 1, \quad i = 1, 2, 3, 4, 5$: sem alterações, pois $w_{i1}^0 = \infty$. A matriz $R_1 = R_0$.

2ª iteração: o vértice 2 é considerado intermediário no caminho entre os vértices i e j.
$k = 2, \quad i = 1, \quad j = 1, 2$: sem alterações, pois $w_{2j}^1 = \infty$.
$k = 2, \quad i = 1, \quad j = 3$: $\quad w_{13}^2 = \min\{w_{13}^1, (w_{12}^1 + w_{23}^1)\} = 6$ (alterar).
$k = 2, \quad i = 1, \quad j = 4$: $\quad w_{14}^2 = \min\{w_{14}^1, (w_{12}^1 + w_{24}^1)\} = 11$ (alterar).
$k = 2, \quad i = 1, \quad j = 5$: $\quad w_{15}^2 = \min\{w_{15}^1, (w_{12}^1 + w_{25}^1)\} = 9$ (alterar).
$k = 2, \quad i = 2, 3, 4, 5$: sem alterações, pois $w_{i2}^1 = \infty$.

Assim

$$R_2 = \begin{bmatrix} 0 & 4 & 6 & 11 & 9 \\ \infty & 0 & 2 & 7 & 5 \\ \infty & \infty & 0 & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

3ª iteração: o vértice 3 é considerado intermediário no caminho entre os vértices i e j.
$k = 3, \quad i = 1, \quad j = 1, 2, 3, 4$: sem alterações, pois $w_{3j}^2 = \infty$.
$k = 3, \quad i = 1, \quad j = 5$: $\quad w_{15}^3 = \min\{w_{15}^2, (w_{13}^2 + w_{35}^2)\} = 8$ (alterar).
$k = 3, \quad i = 2, \quad j = 1, 2, 3, 4$: sem alterações, pois $w_{3j}^2 = \infty$.
$k = 3, \quad i = 2, \quad j = 5$: $\quad w_{25}^3 = \min\{w_{25}^2, (w_{23}^2 + w_{35}^2)\} = 4$ (alterar).
$k = 3, \quad i = 3, 4, 5$: sem alterações, pois $w_{i3}^2 = \infty$.

Assim:

$$R_3 = \begin{bmatrix} 0 & 4 & 6 & 11 & 8 \\ \infty & 0 & 2 & 7 & 4 \\ \infty & \infty & 0 & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

4ª iteração: o vértice 4 é considerado intermediário no caminho entre os vértices i e j.
$k = 4$: sem alterações, pois $w_{4j}^3 = \infty$. A matriz $R_4 = R_3$.

5ª iteração: o vértice 5 é considerado intermediário no caminho entre os vértices i e j.
$k = 5, \quad i = 1, \quad j = 1, 2, 3, 5$: sem alterações, pois $w_{5j}^4 = \infty$.
$k = 5, \quad i = 1, \quad j = 4$: $\quad w_{14}^5 = \min\{w_{14}^4, (w_{15}^4 + w_{54}^4)\} = 9$ (alterar).
$k = 5, \quad i = 2, \quad j = 1, 2, 3, 5$: sem alterações, pois $w_{5j}^4 = \infty$.
$k = 5, \quad i = 2, \quad j = 4$: $\quad w_{24}^5 = \min\{w_{24}^4, (w_{25}^4 + w_{54}^4)\} = 5$ (alterar).
$k = 5, \quad i = 3, \quad j = 1, 2, 3, 5$: sem alterações, pois $w_{5j}^4 = \infty$.
$k = 5, \quad i = 3, \quad j = 4$: $\quad w_{34}^5 = \min\{w_{34}^4, (w_{35}^4 + w_{54}^4)\} = 3$ (alterar).
$k = 5, \quad i = 4, 5$: sem alterações, pois $w_{i5}^4 = \infty$.

Assim, a matriz R_5 abaixo mostra o resultado final do algoritmo.

$$R_5 = \begin{bmatrix} 0 & 4 & 6 & 9 & 8 \\ \infty & 0 & 2 & 5 & 4 \\ \infty & \infty & 0 & 3 & 2 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

3.1.4 Implementação do Algoritmo Floyd-Warshall numa planilha eletrônica

O Algoritmo Floyd-Warshall também pode ser implementado numa planilha eletrônica. Usaremos o exemplo apresentado na subseção anterior para mostrar a implementação. Mais uma vez, utilizaremos a planilha eletrônica do software *Microsoft Office Excel versão*

2007. Inicialmente construiremos 6 tabelas, sendo que a primeira delas deve representar a matriz de adjacência, conforme mostra a figura 36. Os valores ∞ serão representados, na planilha, por um grande inteiro positivo. Nessa planilha foi usado o valor 99^{99}.

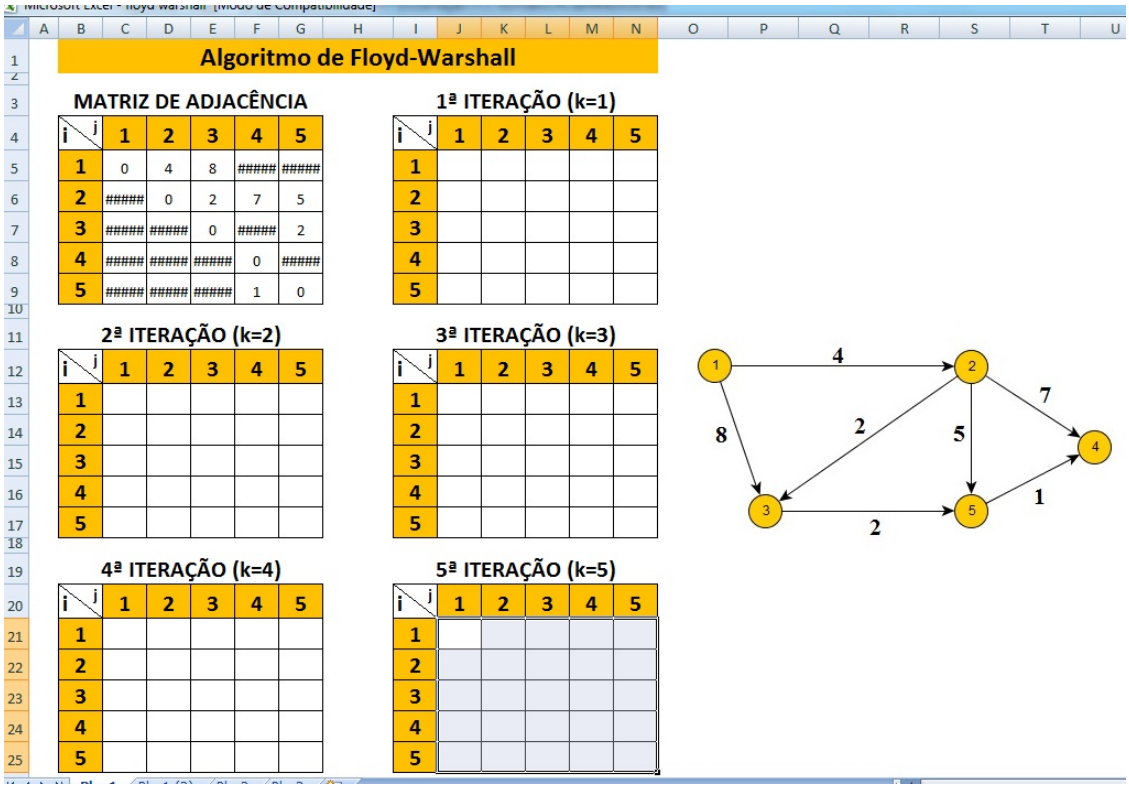


Figura 36: Tabelas para implementação do algoritmo de Floyd-Warshall numa planilha eletrônica.

As outras tabelas representarão as matrizes de iteração da execução do algoritmo. Em cada uma das células dessas tabelas vamos inserir a seguinte fórmula padrão:

=MÍNIMO(célula que contém w_{ij}^{k-1};célula que contém w_{ik}^{k-1}+ célula que contém w_{kj}^{k-1}).

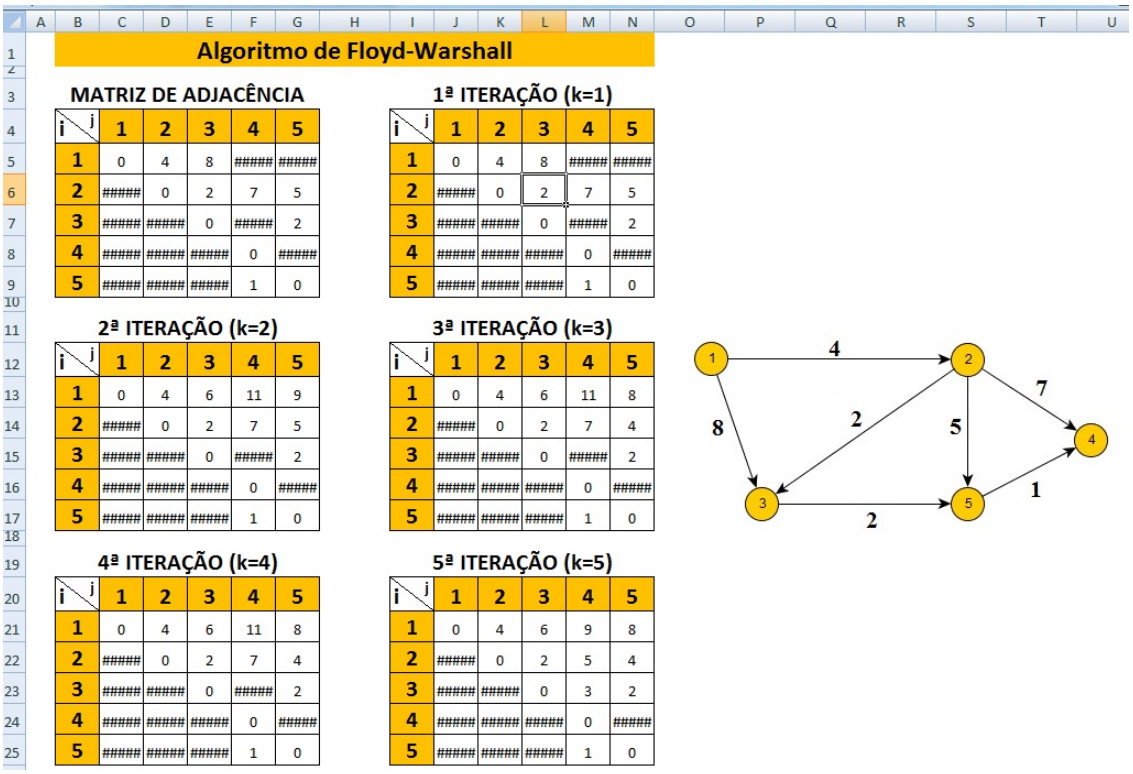
Na célula L6 da tabela da figura 36, por exemplo, foi inserida a seguinte fórmula:

=MÍNIMO(E6;C6+E5).

Ao final, a planilha apresentou a tabela-solução da figura 37.

O Problema do Caminho Mais Curto desde o vértice v até o vértice w no grafo da figura 19 também foi solucionado executando o algoritmo de Floyd-Warshall numa planilha eletrônica. A figura 38 mostra a 8ª iteração da execução, cuja matriz representa o resultado final do algoritmo. Como era esperado, os dois algoritmos apresentaram o mesmo resultado.

Na próxima seção, apresentamos um problema relacionado ao Problema do Caminho mais Curto, o Problema do Caixeiro Viajante, que consiste em determinar o caminho mais curto, passando exatamente uma vez por cada vértice e retornando ao vértice de partida.



Algoritmo de Floyd-Warshall

MATRIZ DE ADJACÊNCIA

i \ j	1	2	3	4	5
1	0	4	8	#####	#####
2	#####	0	2	7	5
3	#####	#####	0	#####	2
4	#####	#####	#####	0	#####
5	#####	#####	#####	1	0

1ª ITERAÇÃO (k=1)

i \ j	1	2	3	4	5
1	0	4	8	#####	#####
2	#####	0	2	7	5
3	#####	#####	0	#####	2
4	#####	#####	#####	0	#####
5	#####	#####	#####	1	0

2ª ITERAÇÃO (k=2)

i \ j	1	2	3	4	5
1	0	4	6	11	9
2	#####	0	2	7	5
3	#####	#####	0	#####	2
4	#####	#####	#####	0	#####
5	#####	#####	#####	1	0

3ª ITERAÇÃO (k=3)

i \ j	1	2	3	4	5
1	0	4	6	11	8
2	#####	0	2	7	4
3	#####	#####	0	#####	2
4	#####	#####	#####	0	#####
5	#####	#####	#####	1	0

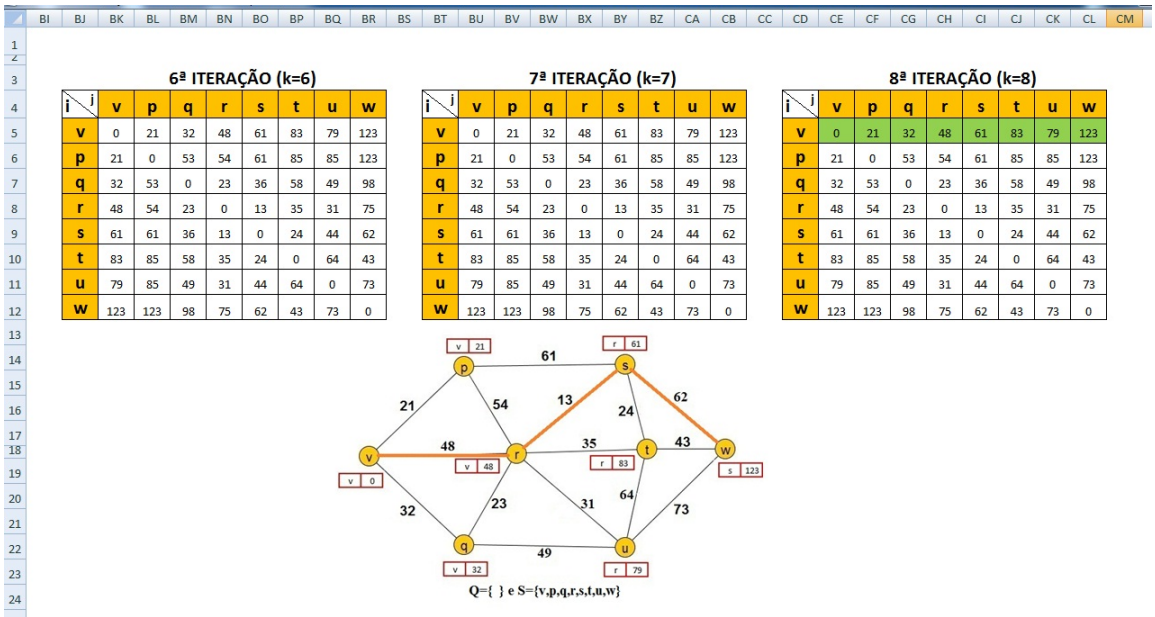
4ª ITERAÇÃO (k=4)

i \ j	1	2	3	4	5
1	0	4	6	11	8
2	#####	0	2	7	4
3	#####	#####	0	#####	2
4	#####	#####	#####	0	#####
5	#####	#####	#####	1	0

5ª ITERAÇÃO (k=5)

i \ j	1	2	3	4	5
1	0	4	6	9	8
2	#####	0	2	5	4
3	#####	#####	0	3	2
4	#####	#####	#####	0	#####
5	#####	#####	#####	1	0

Figura 37: Tabela com a solução do Caminho Mais Curto no grafo da figura 35, utilizando o algoritmo de Floyd-Warshall.



6ª ITERAÇÃO (k=6)

i \ j	v	p	q	r	s	t	u	w
v	0	21	32	48	61	83	79	123
p	21	0	53	54	61	85	85	123
q	32	53	0	23	36	58	49	98
r	48	54	23	0	13	35	31	75
s	61	61	36	13	0	24	44	62
t	83	85	58	35	24	0	64	43
u	79	85	49	31	44	64	0	73
w	123	123	98	75	62	43	73	0

7ª ITERAÇÃO (k=7)

i \ j	v	p	q	r	s	t	u	w
v	0	21	32	48	61	83	79	123
p	21	0	53	54	61	85	85	123
q	32	53	0	23	36	58	49	98
r	48	54	23	0	13	35	31	75
s	61	61	36	13	0	24	44	62
t	83	85	58	35	24	0	64	43
u	79	85	49	31	44	64	0	73
w	123	123	98	75	62	43	73	0

8ª ITERAÇÃO (k=8)

i \ j	v	p	q	r	s	t	u	w
v	0	21	32	48	61	83	79	123
p	21	0	53	54	61	85	85	123
q	32	53	0	23	36	58	49	98
r	48	54	23	0	13	35	31	75
s	61	61	36	13	0	24	44	62
t	83	85	58	35	24	0	64	43
u	79	85	49	31	44	64	0	73
w	123	123	98	75	62	43	73	0

Figura 38: Solução do Problema do Caminho mais Curto de v a w no grafo da figura 19, utilizando o algoritmo de Floyd-Warshall.

4 O Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante ou simplesmente PCV é um dos mais tradicionais e conhecidos problemas de Otimização Combinatória e consiste em determinar um ciclo hamiltoniano de menor custo num grafo ponderado $G = (V, A)$, onde V representa o conjunto de vértices e A o conjunto de arestas. É comum tratarmos os vértices como cidades, depósitos, pontos de visita, etc.. Como já foi mencionado neste trabalho, o ciclo hamiltoniano visita cada vértice do grafo uma única vez, iniciando e terminando o percurso num mesmo vértice, chamado de vértice inicial. Algumas variantes do PCV relaxam algumas dessas características.

A ideia de ciclo hamiltoniano de menor custo surge pela primeira vez na década de 1920, através do matemático austríaco Karl Menger. Durante a realização de um seminário na Princeton University em New Jersey (EUA), Menger conheceu o matemático norte-americano Hassler Whitney, que em seminários posteriores realizados entre 1931 e 1932, fez referência ao problema denominando-o de Caixeiro Viajante.

A importância e grande interesse pelo Problema do Caixeiro Viajante pode ser justificada pelo considerável número de aplicações práticas, tanto da versão clássica quanto das variantes, e pela grande dificuldade de encontrar uma solução exata. Em [19] são destacadas algumas das aplicações práticas do PCV. Dentre elas, podemos citar: solução de problemas de roteamento de veículos, otimização de perfurações de furos em placas de circuitos impressos, na solução de problemas de sequenciamento de tarefas, roteamento de entrega postal e a solução de problemas de sequenciamento de tarefas.

Matematicamente, o PCV é descrito como um grafo $G = (V, A)$, onde $V = \{1, ..., n\}$ é o conjunto dos vértices ou nós do grafo e $A = \{(i, j) \mid i, j \in V\}$ o conjunto de arestas ou arcos ligando esses vértices. Associado a cada arco existe um custo c_{ij}, tal que $c_{ii} = \infty$ para evitar laços ou *loop*. O problema consiste na determinação de um caminho hamiltoniano de custo mínimo sobre G. É comum associarmos vértices a cidades e custo à distância entre as cidades ou tempo de deslocamento de uma cidade a outra.

Na Otimização Combinatória, o PCV pertence à classe dos problemas *NP-difíceis* ([17]), o que significa dizer que, apesar do uso de computadores superpotentes, não se pode determinar a solução exata para problemas de grande porte (com número grande de vértices) em um tempo computacional viável. Atualmente a literatura apresenta algoritmos que encontram soluções para o problema próximas da otimalidade em um tempo computacional razoável.

4.1 Variantes do PCV

Dado a sua aplicação, semelhança com diversos problemas do cotidiano e ligação com outros problemas de otimização combinatória, diversas variantes do PCV foram apresentadas na literatura. As variações decorrem, normalmente, pela presença de um segundo objetivo associado à sequência de visitas e/ou acrescentando alguma restrição específica. O PCV apresenta um número significativo de variantes, entre as quais podemos citar:

- *PCV Simétrico*: o PCV é denominado simétrico quando a matriz de custos é simétrica, ou seja, o custo de deslocamento de qualquer vértice i para qualquer

vértice j tem o mesmo valor de deslocamento de j para i, ou seja $c_{ij} = c_{ji}$ para todo $i, j \in V$.

- *PCV Euclidiano*: seja c o custo de deslocamento entre dois vértices quaisquer do grafo $G = (V, A)$, se $c_{ik} \leq c_{ij} + c_{jk}$, $\forall i, j, k \in V$ o PCV satisfaz à chamada desigualdade triangular. Um PCV é denominado euclidiano (PCVE) quando sua matriz é simétrica e satisfaz à desigualdade triangular.

- *PCV Simétrico com Agrupamento*: é um caso especial de PCV Simétrico, em que existem agrupamentos que possuem restrições que os obrigam a estar em uma determinada sequência de atendimento. Esses conjuntos de nós são denominados de *cluster* ou *grupamentos*, conforme [19].

- *PCV com Sequências de Clientes*: o PCV com Sequências de Clientes pode ser considerado um caso especial de PCV Generalizado onde os nós são divididos em dois grupamentos ou conjuntos disjuntos: o conjuntos dos nós denominados L (nós linehauls) e o conjuntos dos nós B (nós backhauls). Uma condição imposta pelo problema é a sequência de visitação aos conjuntos L e B. Uma versão do problema determina que os nós de L sejam visitados inicialmente e, posteriormente, os nós de B. Uma aplicação prática dessa estratégia no ramo de transporte de cargas é a opção por visitar primeiramente as cidades onde ocorrerá o descarregamento dos veículos e, posteriormente, realizar o carregamento em outros pontos, em direção ao depósito inicial.

- *PCV com Bônus*: o PCV com bônus associa um bônus ou prêmio a cada nó do grafo e procura uma rota hamiltoniana de menor comprimento atendendo a restrição de recolher um valor total de bônus igual ou superior a um valor pré-estabelecido. Este trabalho apresentará, nas próximas páginas, um estudo minucioso sobre essa variante do PCV.

- *PCV com Janelas de Tempo*: essa variante do PCV consiste em definir rotas de custo mínimo atendendo restrições de tempo quanto à chegada, permanência e saída dos nós do ciclo hamiltoniano. Uma das restrições dessa variante pode ser a exigência de que o local de coleta e/ou entrega deverá ser visitado em uma janela de atendimento estipulada, forçando o caixeiro a estar em um determinado vértice em um instante de tempo dentro dessa janela. Além disso, um tempo de serviço está associado a cada coleta e entrega. O tempo de serviço indica quanto tempo levará para a coleta ou a entrega ser realizada.

- *PCV Generalizado*: é semelhante ao PCV Simétrico com Agrupamento. No PCV Generalizado, o ciclo hamiltoniano visita, exatamente uma vez, cada grupamento de nós, passando por todos eles. Ou seja, deseja-se encontrar a rota parcial de menor custo passando por, pelo menos, um nó em cada agrupamento. Cada nó deve pertencer a um, e somente um, grupamento. A figura 39 apresenta um modelo de uma das muitas alternativas de percorrer um circuito com quatro agrupamentos. Uma subclasse dessa variante é denominada *equality* e exige que apenas um vértice

em cada *cluster* seja visitado (figura 40). Quando todos os nós dos grupamentos são visitados o problema recai no PCV clássico (figura 41).

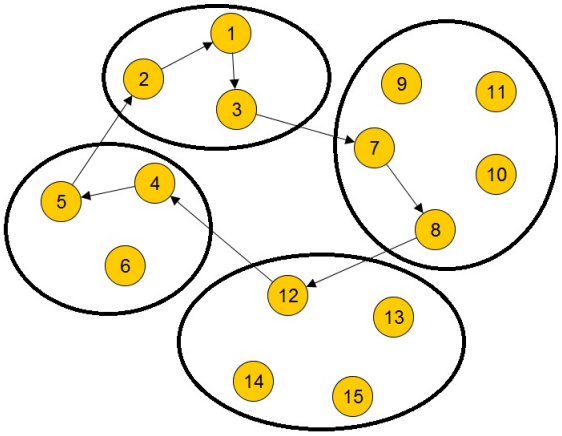


Figura 39: O PCV Generalizado.

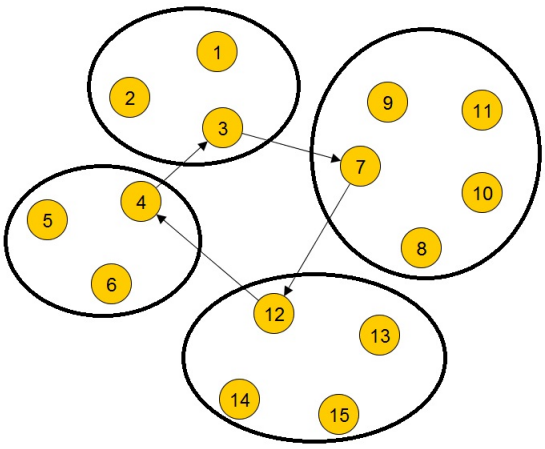


Figura 40: O PCV Generalizado na versão *equality*.

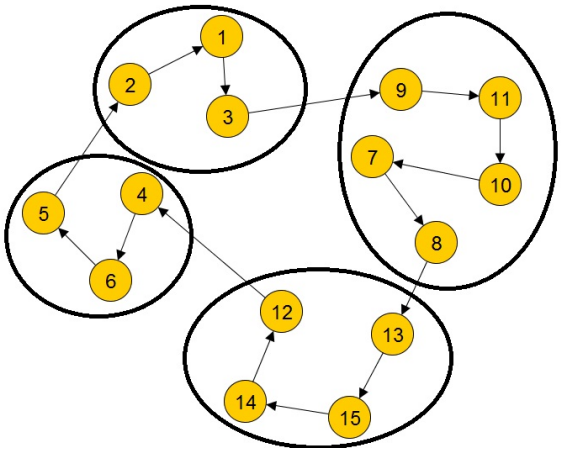


Figura 41: O PCV clássico com os nós divididos em grupamentos.

- *PCV Múltiplo*: o PCV Múltiplo é uma variante do PCV onde é necessário usar mais de um caixeiro viajante. O grafo do problema é dividido em conjuntos disjuntos de nós, cada conjunto percorrido por um caixeiro e o objetivo é minimizar os custos total da rota. Os caixeiros podem ser diferenciados, com a determinação de quais nós percorrerão e qual a distância (ou custo) total a ser percorrida por cada um deles.

- *PCV Estocástico*: no PCV estocástico, a existência ou valores relativos a demandas, custos das arestas, bônus associados a vértices, janelas de tempo são elementos aos quais podemos associar distribuições de probabilidades. Algumas subclasses

dessa variante tratam os clientes como estocásticos e as rotas definidas excluem tais clientes baseados em cálculos probabilísticos.

- *PCV com Coleta e Entrega*: Considerando que o veículo do caixeiro viajante possua uma determinada capacidade e que o caixeiro deva entregar e recolher produtos de seus clientes e considerando ainda que os produtos coletados possam ser entregues a outros clientes na rota, o PCV com Coleta e Entrega consiste em programar um trajeto que, atendendo a todos os clientes de demanda e oferta dentro da capacidade do veículo minimize a distância percorrida.([19])

- *PCV Seletivo Euclidiano*: o PCV Euclidiano consiste em minimizar o custo do ciclo hamiltoniano ligando um conjunto de pontos no plano euclidiano.

- *PCV Alugador*: Nessa variante do PCV, o caixeiro aluga veículos para realizar seu percurso e os custos de percorrer cada aresta dependem do veículo utilizado. O problema é de minimização do custo da viagem, a partir da definição de quais veículos serão alugados, das cidades da rota em que cada um desses veículos será recebido e devolvido às locadoras e da rota percorrida por cada veículo. O percurso do caixeiro é o ciclo hamiltoniano, ou seja, inicia e termina na mesma cidade, e todas as cidades do grafo são visitadas exatamente uma vez.

Em [18] e [19] são descritas e caracterizadas outras variantes do PCV.

4.2 Métodos de Resolução do PCV

Os métodos desenvolvidos para resolver o PCV podem ser divididos em duas categorias: os métodos exatos e os métodos heurísticos.

- os métodos exatos são aqueles que têm como característica a capacidade de determinar sempre uma solução ótima para o problema. Para o PCV, um algoritmo exato deve encontrar o ciclo hamiltoniano de menor custo possível para um dado problema.

- os métodos aproximados ou heurísticos: as heurísticas ou algoritmos heurísticos foram desenvolvidos com a finalidade de se resolver problemas de elevado nível de complexidade em tempo computacional razoável. Por ser um problema de natureza combinatória, uma opção para solucionar o PCV seria analisar todas as combinações de rotas possíveis para conhecer a melhor. Se o problema possui um número reduzido de cidades, esta é a maneira correta de se buscar a melhor solução, mas os problemas com muitas cidades possuem um número elevado de combinações, o que torna inviável a análise de todas as combinações, uma vez que o tempo computacional exigido fica impraticável. As heurísticas encontram soluções próximas da otimalidade em um tempo computacional razoável, sem, no entanto, conseguir definir se esta é a solução ótima, nem o nível de proximidade que ela está da solução ótima.

O PCV é um problema NP-difícil, não sendo conhecidos algoritmos de resolução em tempo polinomial (vide [12]). Para o PCV, no caso assimétrico, o número total de rotas possíveis é $(n-1)!$, onde n é o número de vértices. Sendo assim, à medida que n cresce o tempo de resolução cresce exponencialmente, tornando sua resolução exata inviável, seja avaliando recursos computacionais ou tempo de resolução. O mesmo ocorre no caso simétrico, que tem $(n-1)!/2$ rotas possíveis, pois neste caso o custo para percorrer a aresta a_{ij} é igual ao custo para percorrer a aresta a_{ji}. Esta complexidade justifica o estudo e desenvolvimento de métodos heurísticos para solucionar o problema de forma viável, que apesar de não garantir a resolução do problema de forma exata, alcançam soluções finais de boa qualidade de forma rápida.

Para efeitos de comparação, imaginemos um computador arbitrário superpotente que gere e avalie 1 bilhão de rotas por segundo (poucos computadores, atualmente, têm essa capacidade). Neste caso, é possível construir a tabela 1, que apresenta o número de vértices ou cidades (n), o número de rotas possíveis e o tempo aproximado gasto no cálculo pelo computador supracitado, considerando o problema PCV simétrico.

Tabela 1: Tempo aproximado de cálculo do número de rotas entre n cidades (PCV simétrico) por um computador superpotente.

n	Número de rotas $(n-1)!/2$	Tempo Aproximado de Cálculo
5	12	insignificante
10	181 440	$0,0002$ segundos
15	43 589 145 600	$43,5$ segundos
20	60 822 550 204 416 000	704 dias
25	310 224 200 866 619 719 680 000	9 837 145 anos

Pela tabela, percebe-se que a utilização de um algoritmo que resolva o problema na exatidão, gerando todas as possibilidades de rotas possíveis, só é possível em problemas com número reduzido de vértices. No entanto, há métodos ou recursos que reduzem o tempo computacional gasto. Um desses métodos é o *branch and bound* e baseia-se na ideia de uma enumeração inteligente das soluções candidatas a solução ótima de um problema, efetuando sucessivas partições do espaço das soluções e cortando a árvore de pesquisa através da consideração de limites calculados ao longo da enumeração.

No caso do PCV, o método *branch and bound* realiza o corte ou poda de partes do grafo que não interessam. Um exemplo de poda pode ser visualizado na figura 42, que representa um grafo de geração de rotas arbitrário para um problema com quatro cidades. Os vértices representam as cidades, e as arestas as possibilidades de percurso, iniciando e terminando no vértice ou cidade 1. O grafo em questão é completo, ou seja, é possível partir de uma cidade i para uma cidade j para todos i e $j \in V$ e o problema é assimétrico.

Vamos listar todas as rotas possíveis sempre iniciando e terminando no vértice 1 e o custo de cada uma delas :

- $(1, 2, 3, 4, 1)$ com custo de 21.
- $(1, 2, 4, 3, 1)$ com custo de 32.

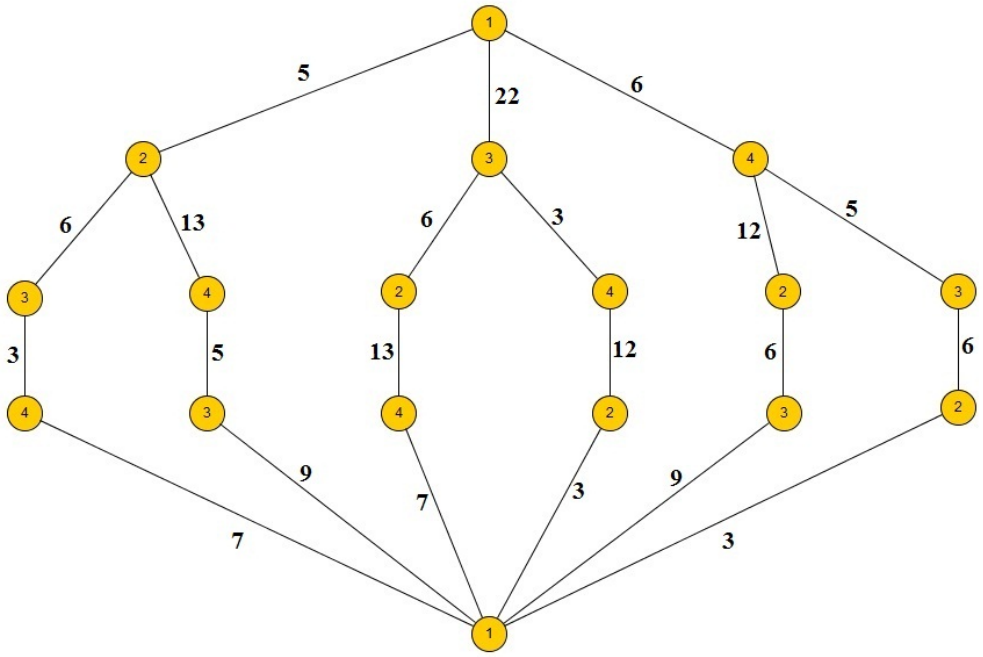


Figura 42: Grafo de busca completo para um PCV de quatro cidades.

- $(1, 3, 2, 4, 1)$ com custo de 48.
- $(1, 3, 4, 2, 1)$ com custo de 40.
- $(1, 4, 2, 3, 1)$ com custo de 33.
- $(1, 4, 3, 2, 1)$ com custo de 20.

Neste exemplo, a rota ótima é $(1, 4, 3, 2, 1)$ com o custo total de 20 e foi obtida através da poda de partes do grafo. A idéia do método *branch and bound* é simples: ao encontrar uma sub-rota cujo custo atual seja superior ao da melhor rota completa já descoberta, as sub-rotas subsequentes são desconsideradas. É fácil perceber que, se uma rota incompleta, ou seja, faltando cidades para serem analisadas, já contempla um custo superior a melhor rota atual, não é necessário prosseguir no percurso desta parte do grafo, pois não é possível que o custo da rota reduza seu valor. A aplicação desta poda no grafo da figura 42 geraria um grafo menor, apresentado na figura 43, e de menor custo computacional em termos de tempo.

O algoritmo inicia sua análise gerando a primeira rota $(1, 2, 3, 4, 1)$ com custo total 21. Esta rota é considerada a melhor rota até o momento e o custo dela é armazenado como custo ótimo parcial. Uma rota ótima parcial representa a de menor custo encontrada até o momento. Se o valor desse custo mínimo parcial não for melhorado até o final, essa rota torna-se a rota ótima do sistema.

A segunda rota parte de 1, vai até 2, depois até 4 terminando em 3. No entanto, ao chegar em 3, a rota analisada já possui o custo de 23 $(5 + 13 + 5)$. Como este custo é

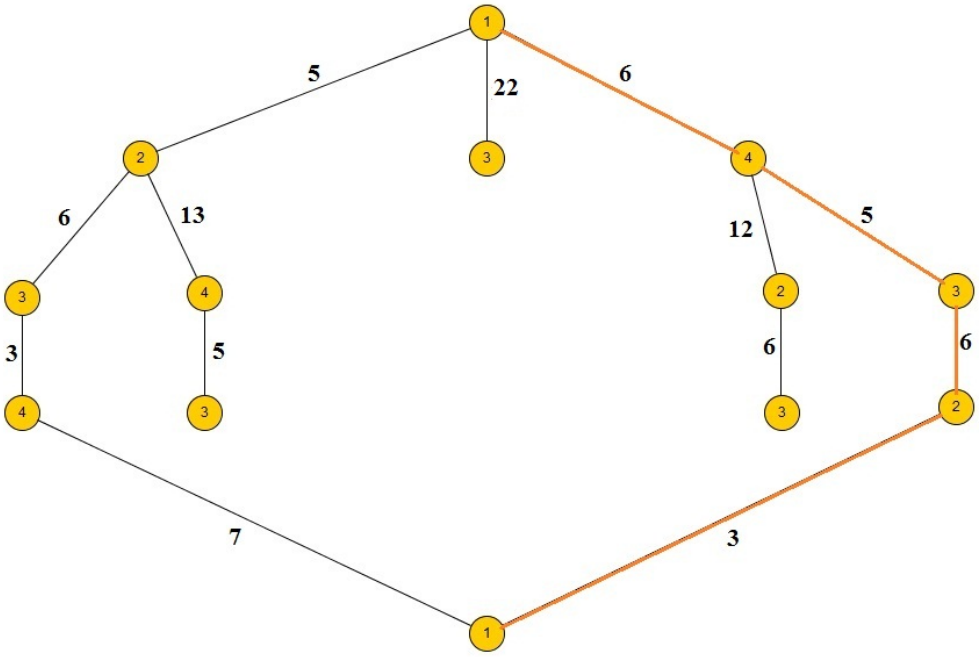


Figura 43: Exemplo de poda em um grafo de um PCV de quatro cidades.

maior que o custo ótimo parcial, o valor do custo do último passo não é calculado e o algoritmo segue para o próximo passo. A próxima rota inicia em 1 e vai até 3. Novamente, o custo deste pequeno trecho já é maior que o custo ótimo parcial. Desta forma, todo o subgrafo resultante das possibilidades de rotas a partir da sub-rota $1-3$ são ignorados e o algoritmo continua. A penúltima opção é gerada, formando uma rota partindo de 1 para 4, depois para 2 e finalmente chegando a 3, com um custo de 24. Esta sub-rota também é finalizada e a última rota possível é analisada, formando a rota completa $(1, 4, 3, 2, 1)$, cujo custo tem valor inferior ao ótimo parcial. A nova rota é armazenada e o custo ótimo parcial é atualizado. Como esta era a última rota do grafo a ser gerada, o algoritmo finaliza, apresentando a solução ótima final.

Na literatura há muitos modelos e algoritmos exatos desenvolvidos para a resolução do PCV, onde é possível citar os trabalhos de [22], [4] e [9].

No entanto, apesar do avanço tecnológico colocar à disposição das ciências, computadores cada vez mais poderosos, mesmo os melhores algoritmos exatos tem dificuldade em encontrar as soluções ótimas para problemas de alta complexidade em um tempo computacional razoável. Os métodos heurísticos fornecem boas soluções em tempos menores.

4.3 Formulação Matemática do PCV

Existem muitas formulações matemáticas para o Problema do Caixeiro Viajante difundidas na literatura. Neste trabalho, apresentaremos duas delas: a Formulação de Dantzig-Fulkerson-Johnson (DFJ) e a Formulação de Formulação de Miller-Tucker-Zemlin (MTZ).

4.3.1 Formulação de Dantzig-Fulkerson-Johnson (DFJ)

Em [12], Dantzig, Fulkerson e Johnson apresentam uma formulação do PCV como um problema de programação binária sobre um grafo $G = (V, A)$, como se segue:

$$\text{Minimizar } z = \sum_{j=1}^{n} \sum_{i=1}^{n} c_{ij} x_{ij}$$

sujeito a

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j \in V \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in V \tag{3}$$

$$\sum_{i,j \in S} x_{ij} \leq \mid S \mid -1, \forall S \subset V \tag{4}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in V, \tag{5}$$

onde a c_{ij} representa o custo para percorrer a arco (i, j); a variável binária x_{ij} assume valor igual a 1, se o arco $(i, j) \in A$ for escolhido para integrar a solução, e 0 em caso contrário; S é um subgrafo de G, em que $\mid S \mid$ representa o número de vértices desse subgrafo.

A restrição 2 força a chegada de exatamente um arco no vértice j e a restrição 3 força a saída de exatamente um arco do vértice i. Mas essas duas últimas restrições não impedem a formação de subcircuitos (figura 44). O conjunto de restrições 4 determina a eliminação de subcircuitos. As equações em $\mid S \mid$ tornam os subcircuitos ilegais e determinam que $x_{ii} = 0$ quando $\mid S \mid = 1$. Para cada subcircuito possível é necessária uma restrição do tipo 4. Por fim, a restrição 5 garante as condições binárias e de não negatividade das variáveis.

Nessa formulação teremos $n.(n-1)$ variáveis inteiras binárias (é implícito considerar $x_{ii} = 0$), e uma quantidade de restrições na ordem de 2^n.

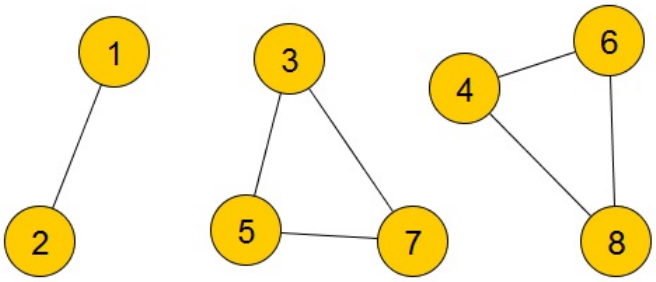


Figura 44: Representação de subcircuitos.

Essa formulação evidencia a natureza combinatória do PCV. Solucionar o PCV é determinar uma certa permutação legal de custo mínimo.

Para essa formulação, apresentamos um exemplo genérico do problema do caixeiro viajante com cinco vértices ou cidades. O grafo que representa o problema pode ser visualizado na figura 45 .

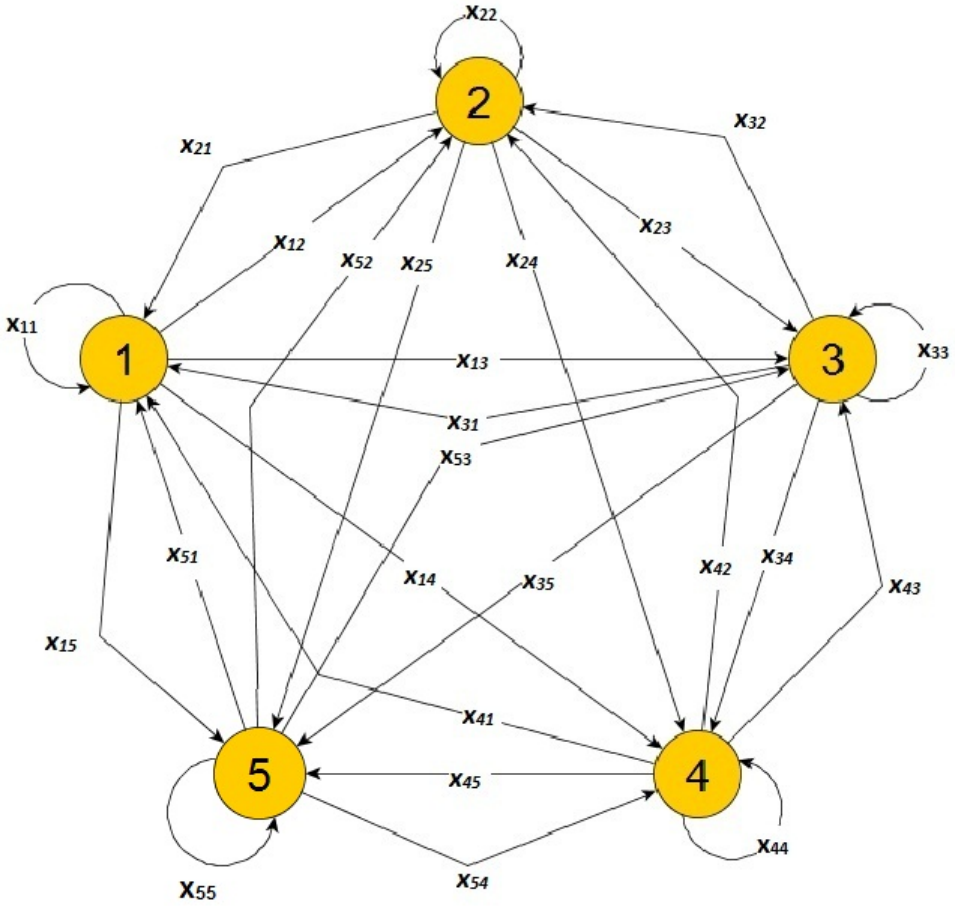


Figura 45: PCV com 5 nós.

A função objetivo será minimizar o custo das distâncias, ou seja:

$$\text{Minimizar } z = \sum_{j=1}^{5}\sum_{i=1}^{5} c_{ij}x_{ij}$$

ou melhor,

$$\begin{aligned}\text{Minimizar } z = {} & c_{11}x_{11} + c_{21}x_{21} + c_{31}x_{31} + c_{41}x_{41} + c_{51}x_{51} + c_{12}x_{12} + c_{22}x_{22} + c_{32}x_{32} + \\ & c_{42}x_{42} + c_{52}x_{52} + c_{13}x_{13} + c_{23}x_{23} + c_{33}x_{33} + c_{43}x_{43} + c_{53}x_{53} + c_{14}x_{14} + \\ & c_{24}x_{24} + c_{34}x_{34} + c_{44}x_{44} + c_{54}x_{54} + c_{15}x_{15} + c_{25}x_{25} + c_{35}x_{35} + c_{45}x_{45} + c_{55}x_{55},\end{aligned}$$

onde: c_{ij} é o custo ou comprimento do arco (i, j) e x_{ij} é o fluxo no arco (i, j).

As restrições do PCV poderão ser modeladas matematicamente da seguinte forma:
Restrição 2: O fluxo de entrada em cada vértice deve ser exatamente 1, para que não tenha mais que um caminho convergindo para um único vértice. Então:

$$\sum_{i=1}^{5} x_{ij} = 1, \quad \forall j \in \{1, 2, 3, 4, 5\},$$

ou seja

$$\begin{aligned}
x_{11} + x_{21} + x_{31} + x_{41} + x_{51} &= 1 \\
x_{12} + x_{22} + x_{32} + x_{42} + x_{52} &= 1 \\
x_{13} + x_{23} + x_{33} + x_{43} + x_{53} &= 1 \\
x_{14} + x_{24} + x_{34} + x_{44} + x_{54} &= 1 \\
x_{15} + x_{25} + x_{35} + x_{45} + x_{55} &= 1
\end{aligned}$$

Restrição 3: O fluxo de saída de cada vértice deve ser exatamente 1, garantindo que, de cada vértice, partirá um único caminho. Então:

$$\sum_{j=1}^{5} x_{ij} = 1, \quad \forall i \in \{1, 2, 3, 4, 5\},$$

ou seja

$$\begin{aligned}
x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &= 1 \\
x_{21} + x_{22} + x_{23} + x_{24} + x_{25} &= 1 \\
x_{31} + x_{32} + x_{33} + x_{34} + x_{35} &= 1 \\
x_{41} + x_{42} + x_{43} + x_{44} + x_{45} &= 1 \\
x_{51} + x_{52} + x_{53} + x_{54} + x_{55} &= 1
\end{aligned}$$

Conjunto de restrições 4:
a) O fluxo do vértice para ele mesmo deve ser igual a 0 para evitar laços. Então:

$$\sum_{i,j \in S} x_{ij} \leq \mid S \mid -1 \Rightarrow 1 - 1 = 0$$

ou seja

$$\begin{aligned}
x_{11} &= 0 \qquad S = \{1\} \\
x_{22} &= 0 \qquad S = \{2\} \\
x_{33} &= 0 \qquad S = \{3\} \\
x_{44} &= 0 \qquad S = \{4\} \\
x_{55} &= 0 \qquad S = \{5\}
\end{aligned}$$

b) Não podem ocorrer subciclos, ou seja, o ciclo deve percorrer todos os vértices do grafo, satisfazendo uma das características do problema. Então:

$$\sum_{i,j \in S} x_{ij} \leq \mid S \mid -1 \Rightarrow 2 - 1 \leq 1$$

ou seja

$$\begin{aligned}
x_{12} + x_{21} &\leq 1 \qquad S = \{1,2\} \\
x_{13} + x_{31} &\leq 1 \qquad S = \{1,3\} \\
x_{14} + x_{41} &\leq 1 \qquad S = \{1,4\} \\
x_{15} + x_{51} &\leq 1 \qquad S = \{1,5\} \\
x_{23} + x_{32} &\leq 1 \qquad S = \{2,3\} \\
x_{24} + x_{42} &\leq 1 \qquad S = \{2,4\} \\
x_{25} + x_{52} &\leq 1 \qquad S = \{2,5\} \\
x_{34} + x_{43} &\leq 1 \qquad S = \{3,4\} \\
x_{35} + x_{53} &\leq 1 \qquad S = \{3,5\} \\
x_{45} + x_{54} &\leq 1 \qquad S = \{4,5\}
\end{aligned}$$

e

$$\sum_{i,j \in S} x_{ij} \leq \mid S \mid -1 \Rightarrow 3 - 1 \leq 2$$

ou seja

$$\begin{aligned}
x_{12} + x_{21} + x_{13} + x_{31} + x_{23} + x_{32} &\leq 2 \qquad S = \{1,2,3\} \\
x_{12} + x_{21} + x_{14} + x_{41} + x_{24} + x_{42} &\leq 2 \qquad S = \{1,2,4\} \\
x_{12} + x_{21} + x_{15} + x_{51} + x_{25} + x_{52} &\leq 2 \qquad S = \{1,2,5\} \\
x_{13} + x_{31} + x_{14} + x_{41} + x_{34} + x_{43} &\leq 2 \qquad S = \{1,3,4\} \\
x_{13} + x_{31} + x_{15} + x_{51} + x_{35} + x_{53} &\leq 2 \qquad S = \{1,3,5\} \\
x_{14} + x_{41} + x_{15} + x_{51} + x_{45} + x_{54} &\leq 2 \qquad S = \{1,4,5\} \\
x_{23} + x_{32} + x_{24} + x_{42} + x_{34} + x_{43} &\leq 2 \qquad S = \{2,3,4\} \\
x_{23} + x_{32} + x_{25} + x_{52} + x_{35} + x_{53} &\leq 2 \qquad S = \{2,3,5\} \\
x_{24} + x_{42} + x_{25} + x_{52} + x_{45} + x_{54} &\leq 2 \qquad S = \{2,4,5\} \\
x_{34} + x_{43} + x_{35} + x_{53} + x_{45} + x_{54} &\leq 2 \qquad S = \{3,4,5\}
\end{aligned}$$

e

$$\sum_{i,j \in S} x_{ij} \leq \mid S \mid -1 \Rightarrow 4 - 1 \leq 3$$

ou seja

$$x_{12} + x_{21} + x_{13} + x_{31} + x_{14} + x_{41} + x_{23} + x_{32} + x_{24} + x_{42} + x_{34} + x_{43} \leq 3, S = \{1, 2, 3, 4\}$$
$$x_{12} + x_{21} + x_{13} + x_{31} + x_{15} + x_{51} + x_{23} + x_{32} + x_{25} + x_{52} + x_{35} + x_{53} \leq 3, S = \{1, 2, 3, 5\}$$
$$x_{12} + x_{21} + x_{14} + x_{41} + x_{15} + x_{51} + x_{24} + x_{42} + x_{25} + x_{52} + x_{45} + x_{54} \leq 3, S = \{1, 2, 4, 5\}$$
$$x_{13} + x_{31} + x_{14} + x_{41} + x_{15} + x_{51} + x_{34} + x_{43} + x_{35} + x_{53} + x_{45} + x_{54} \leq 3, S = \{1, 3, 4, 5\}$$
$$x_{23} + x_{32} + x_{24} + x_{42} + x_{25} + x_{52} + x_{34} + x_{43} + x_{35} + x_{53} + x_{45} + x_{54} \leq 3, S = \{2, 3, 4, 5\}$$

Restrição 5: As variáveis devem ser binárias, admitindo valor 0 ou 1. Então:

$$x_{ij} \in \{0, 1\}$$

ou seja $x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{31}, x_{32}, x_{33}, x_{34}, x_{35}, x_{41}, x_{42}, x_{43}, x_{44}, x_{45}, x_{51}, x_{52}, x_{53}, x_{54}, x_{55} \in \{0, 1\}$.

No exemplo apresentado acima, para ciclo com 5 vértices, foram necessárias 40 restrições relativas ao fluxo, como já era esperado (número de restrições na ordem de $2^5 = 32$).

4.3.2 Implementação do PCV em uma planilha eletrônica na Formulação DFJ

O modelo matemático do PCV formulado por Dantzig, Fulkerson e Johnson, apresentado na seção anterior, pode ser implementado em planilhas eletrônicas. Abaixo, será descrito os passos para a construção e cálculo da planilha.

A planilha eletrônica utilizada nesse modelo foi a do software *Microsoft Office Excel versão 2007*. O modelo proposto aqui é de um PCV Simétrico com 7 cidades: A, B, C, D, E, F e G. Sem perda de generalidade, a cidade A é considerada cidade inicial. Inicialmente construímos quatro tabelas como mostra a figura 46.

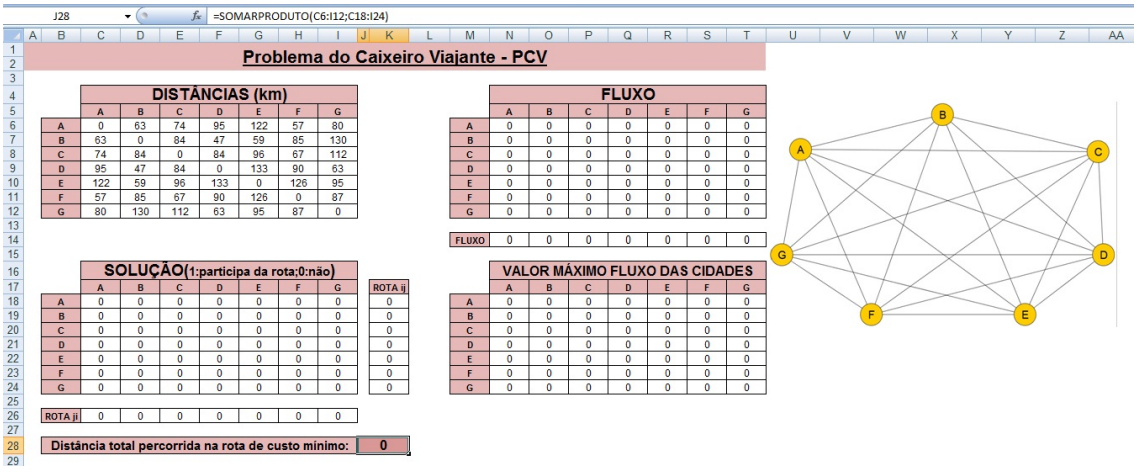


Figura 46: Tabelas de uma planilha eletrônica para cálculo da distância total percorrida no ciclo hamiltoniano de custo mínimo do PCV com 7 cidades.

Tabela 2: Fórmulas relativas à eliminação de subciclos no PCV.

Célula	Fórmula
N14	=SOMA(N6:N12)-SOMA(N6:T6)
O14	=SOMA(O6:O12)-SOMA(N7:T7)
P14	=SOMA(P6:P12)-SOMA(N8:T8)
Q14	=SOMA(Q6:Q12)-SOMA(N9:T9)
R14	=SOMA(R6:R12)-SOMA(N10:T10)
S14	=SOMA(S6:S12)-SOMA(N11:T11)
T14	=SOMA(T6:T12)-SOMA(N12:T12)

Na tabela FLUXO, serão inseridas as fórmulas relativas à eliminação de subciclos, conforme mostra a tabela 2.

Em todas as células da tabela VALOR MÁXIMO FLUXO CIDADES serão inseridas uma fórmula que multiplica os respectivos valores das células da tabela SOLUÇÃO por 6, que representa o valor máximo da quantidade de fluxo enviada da cidade i para a cidade j. Por exemplo, a fórmula inserida na célula N18 será =C18*6 na célula T24 será =I24*6.

Na tabela SOLUÇÃO, nas células da linha denominada ROTA ji e da coluna ROTA ij serão inseridas fórmulas de soma dos valores da coluna e da linha referente à cada cidade, respectivamente. Por exemplo, a fórmula inserida na célula C26 será =SOMA(C18:C24) e na célula K18 será =SOMA(C18:I18). Essas fórmulas combinadas com as restrições do solver, forçarão o caixeiro visitar todas as cidades do ciclo, passando exatamente uma vez por cada uma delas.

Por fim, devemos inserir a fórmula da função objetivo que minimiza o custo total da rota do caixeiro no ciclo hamiltoniano. Colocamos o cursor na célula J28 e inserimos a fórmula:

=SOMARPRODUTO(C6:I12;C18:I24).

A planilha está preparada para utilização do solver. As figuras 47 e 48 mostram as configurações do solver e a figura 49 mostra a planilha com a solução do problema.

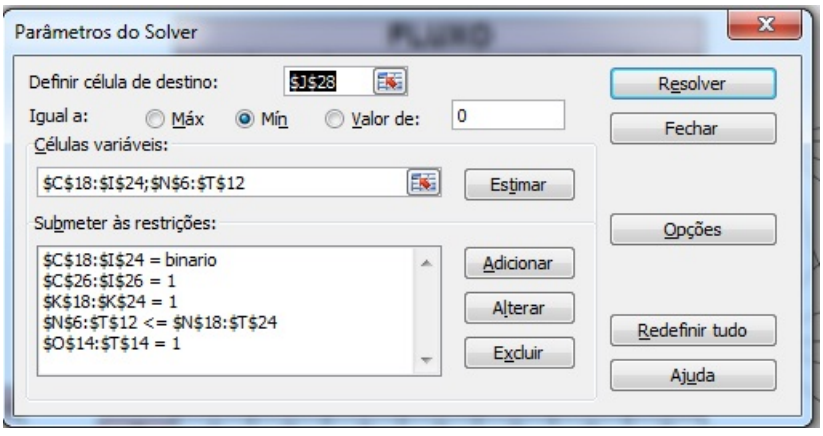


Figura 47: Função objetivo e restrições adicionadas no solver.

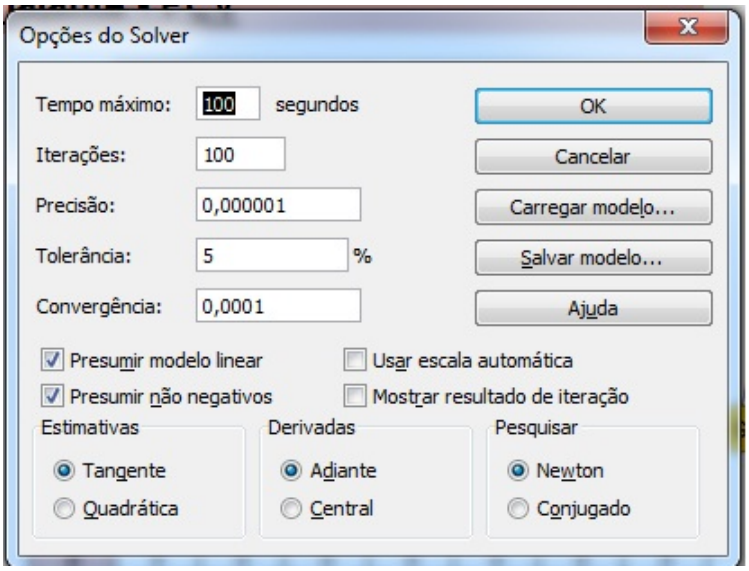


Figura 48: Opções do solver.

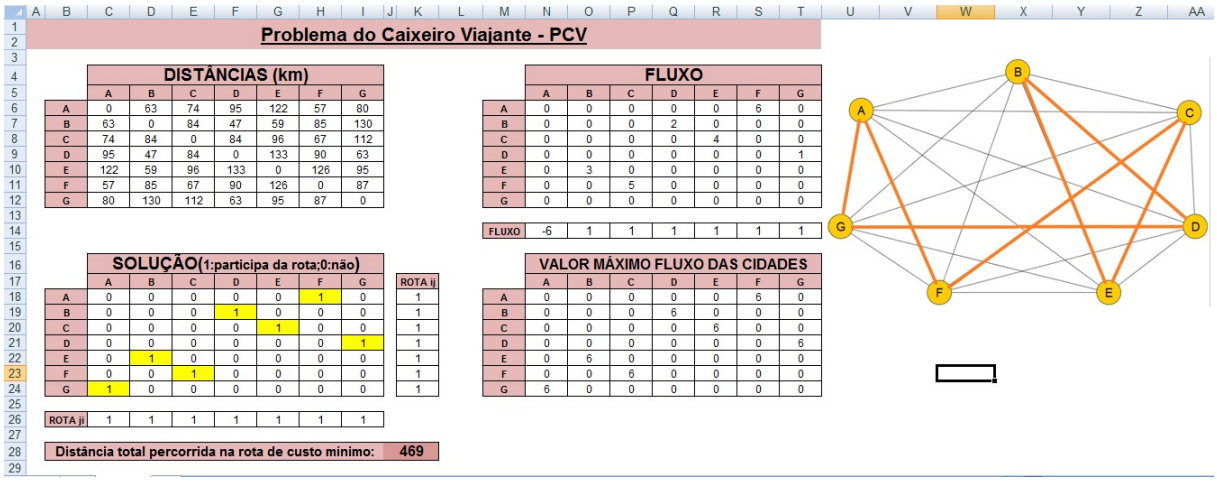


Figura 49: Solução do PCV com 7 cidades numa planilha eletrônica.

4.3.3 Formulação de Miller-Tucker-Zemlin (MTZ)

Em [24], Miller,Tucker e Zemlin propõem a uma formulação matemática para o PCV denominada, segundo [19], Folha de Cravo, que é descrita a seguir:

> [...]denotamos a cidade 1 como a cidade início-fim ou cidade origem. O caixeiro viajante deve visitar as outras $n-1$ cidades exatamente uma vez. Durante seu trajeto deve retornar a cidade origem exatamente t vezes, incluindo o retorno final, e não deve visitar mais de p cidades diferentes em um tour ou ciclo. A formulação requer que
>
> $$\left\lceil \frac{n-1}{p} \right\rceil \leq t \leq n-1,$$

onde $\lceil a \rceil$ a denota o menor inteiro maior ou igual ao valor de a, ou teto da divisão, para garantir a existência de tours viáveis.

Isto posto, segundo Miller,Tucker e Zemlin, o PCV pode ser formulado assim:

$$\text{Minimizar } z = \sum_{j=1}^{n} \sum_{i=1}^{n} c_{ij} x_{ij}$$

sujeito a

$$\sum_{i=2}^{n} x_{i1} = t \tag{6}$$

$$\sum_{i=1}^{n} x_{ji} = 1, \quad j = 2, ..., n \tag{7}$$

$$\sum_{j=1}^{n} x_{ji} = 1, \quad i = 2, ..., n \tag{8}$$

$$u_i - u_j + p x_{ij} \leq p - 1, \quad 2 \leq i \neq j \leq n \tag{9}$$

$$u_i \geq 0, \quad 2 \leq i \leq n \tag{10}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V, \tag{11}$$

A restrição 6 garante que a cidade origem 1 é visitada exatamente t vezes. O arco (i, j) em toda a solução viável binária dos conjuntos de restrições 6, 7, 8 obrigam o conjunto de restrições em p a transformar-se em:

$$u_i - u_j \leq -1.$$

Essa formulação modela o PCV clássico, quando $t = 1$ e $p \geq n - 1$.

4.4 PCV com Bônus

O Problema do Caixeiro Viajante com Bônus é uma variante do PCV em que não há a exigência de que todos os vértices do grafo sejam visitados pelo ciclo hamiltoniano do caixeiro. Um bônus ou prêmio é associado aos vértices e um custo (ou distância entre vértices) é associado às arestas do problema. O objetivo geral é a otimização simultânea do bônus recolhido no ciclo e dos custos de viagem.

De acordo com [18], podemos definir três tipos de problemas do PCV com Bônus, classificados de acordo como os bônus e os custos são considerados:

1º tipo *PCV com Lucros ou Profitable tour problem*: foi proposto por [14]. O objetivo é encontrar uma rota que minimiza os custos de viagem subtraídos os bônus coletados.

2º tipo *PCV Seletivo ou Orienteering Problem*: denominado assim por [7]. O objetivo é encontrar uma rota que maximize a coleta bônus, desde que os custos da rota não ultrapassem um valor pré-estabelecido, chamado de c_{max}. Os custos da rota são alocados como restrição do problema. Em alguns casos, c_{max} é o tempo máximo para efetuar o trajeto (t_{max}).

3º tipo *PCV com Coleta de Prêmios ou Prize-Collecting Traveling Salesman Problem*: foi proposto por [3]. O objetivo é encontrar a rota de menor custo desde que a coleta do bônus ou prêmio não seja menor do que um valor pré-estabelecido. A coleta de bônus é considerada restrição do problema. Uma penalidade é atribuída aos vértices não visitados. No problema conhecido por *Quota TSP*, as penalidades tem valor nulo.

4.4.1 Problema do Caixeiro Viajante com Lucros

Uma formulação matemática para o *PCV com Lucros* também conhecido na literatura como *Profitable tour problem*, é apresentada em [15] e a descrevemos a seguir:

$$\begin{aligned}
\text{Minimizar } z = & \sum_{V_i \in V} \sum_{V_j \in V \setminus \{v_i\}} c_{ij} x_{ij} - \sum_{V_i \in V} p_i y_i \\
& \text{sujeito a} \\
& \sum_{v_j \in V \; v_i} x_{ij} = y_i, \quad v_i \in V \\
& \sum_{v_i \in V \; v_j} x_{ij} = y_j, \quad v_j \in V \\
& y_1 = 1 \\
& \sum_{v_i \in S} \sum_{v_j \in S \; v_i} x_{ij} \leq \mid S \mid -1, \quad S \subset V \\
& x_{ij} \in \{0,1\}; \quad \forall i,j = 1,...N \\
& y_i \in \{0,1\}; \quad \forall i = 1,...N.
\end{aligned}$$

Para definir a formulação do problema *Quota TSP*, a função objetivo é substituída por:

$$\text{Minimizar } z = \sum_{V_i \in V} \sum_{V_j \in V \setminus \{v_i\}} c_{ij} x_{ij}$$

e a inserção da restrição

$$\sum_{V_i \in V} p_i y_i \geq p_{min},$$

onde p_{min} é o bônus mínimo a ser coletado.

4.4.2 Problema do Caixeiro Viajante Seletivo

Em [27], o *PCV Seletivo* ou *Orienteering Problem* é definido da seguinte forma: seja $G = (V, A)$ um grafo completo e não orientado onde $V = \{v_1, ..., v_n\}$ é o conjunto de vértices e $A = \{(v_i, v_j) \mid v_i, v_j \in V, i < j\}$ é o conjunto de arestas. Um prêmio não-negativo S_i é associado a cada vértice $v_i \in V$ e o tempo de viagem t_{ij} é associado a cada

aresta (v_i, v_j). O problema consiste em determinar um circuito hamiltoniano $G' \subset G$ sobre o conjunto V, incluindo a definição de um vértice de saída (v_i) e um vértice de chegada (v_n), de forma a maximizar o prêmio total coletado, não excedendo o tempo máximo disponível t_{max}.

Este problema pode ser de definido como um caminho, no qual o vértice inicial (v_i) difere do vértice final (v_n), bem como também pode ser definido como tendo o vértice final coincidente com o vértice de saída, originando assim um caminho fechado.

Considerando a notação definida acima, e ainda que a variável de decisão $x_{ij} = 1$ indica que o vértice i foi visitado após o vértice j, e $x_{ij} = 0$ caso contrário e que a variável u_i indica a posição do vértice i no caminho, então a formulação matemática para o *Orienteering Problem* é definida de acordo com [27], assim:

$$\begin{aligned}
\text{Maximizar } z = & \sum_{i=2}^{n-1} \sum_{j=2}^{n} S_i x_{ij} \\
& \text{sujeito a} \\
& \sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1 \\
& \sum_{i=1}^{n-1} x_{ik} = \sum_{j=2}^{n} x_{kj} \leq 1 \quad \forall k = 2, ..., n-1 \\
& \sum_{i=1}^{n-1} \sum_{j=2}^{n} t_{ij} x_{ij} \leq t_{max} \\
& 2 \leq u_i \leq n; \quad \forall i = 2, ..., n \\
& u_i - u_j + 1 \leq (n-1)(1 - x_{ij}); \quad \forall j = 2, ..., n \\
& x_{ij} \in \{0, 1\}; \quad \forall i, j = 1, ...n.
\end{aligned}$$

4.4.3 Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP)

Para definir a formulação do *Problema do Caixeiro Viajante com Coleta de Prêmios* também conhecido como *Prize-Collecting Traveling Salesman Problem*, proposta em [3], considere o seguinte: seja $G' = (V, A)$ um grafo completo direcionado, para cada arco (i, j) de A é dado um custo c_{ij}, e para cada vértice i de V, é associada uma penalidade p_i, a ser paga se o vértice i não compor a rota. Além disso, para cada vértice i, existe um prêmio w_i associado. Os vértices são numerados de 1 até $n =\mid V \mid (\mid V \mid$ é o número de vértices do grafo $G')$, sendo o vértice 1, sem perda de generalidade, assumido como cidade inicial da rota ou depósito. Para este vértice inicial, temos $w_1 = 0$ e $p_1 = \infty$.

Assumindo que y_i seja 1 se o vértice i for incluído na rota e 0 caso contrário, que x é o vetor de incidência associado à rota (ou seja, assume valor 1 caso a aresta (i, j) esteja na rota, e 0 caso contrário), e que f garante que a diferença entre o fluxo que chega e que sai do vértice seja igual a 1, se o vértice for visitado, e 0 caso contrário, e também não permite que a quantidade de fluxo entre os vértices i e j ultrapasse o número de vértices

possíveis de serem visitados, o Problema do Caixeiro Viajante com Coleta de Prêmios é formulado por [3] assim:

$$\text{Minimizar } z = \sum_{i \in V} \sum_{j \in V-\{1\}} c_{ij} x_{ij} + \sum_{i \in V} p_i (1 - y_i) \tag{12}$$

sujeito a

$$\sum_{j \in V-\{i\}} x_{ij} = y_i \quad \forall i = 1, ..., n \tag{13}$$

$$\sum_{i \in V-\{j\}} x_{ij} = y_j \quad \forall j = 1, ..., n \tag{14}$$

$$\sum_{i \in V} (w_i . y_i) \geq w_0 \tag{15}$$

$$\sum_{i \in V} \sum_{j \in V} f_{ij} - \sum_{i \in V} \sum_{j \in V} f_{ji} = y_i \quad \forall i = 1, ..., n \quad i \neq 1 \tag{16}$$

$$f_{ij} \leq (n-1) x_{ij} \quad \forall (i,j) \in A \tag{17}$$

$$x_{ij} \in \{0, 1\}; \quad \forall (i,j) \in A \tag{18}$$

$$y_j \in \{0, 1\}; \quad \forall i = 1, ...n. \tag{19}$$

A função objetivo 12 é composta pelo custo total do deslocamento somado aos custos de penalidades pagos por não visitação de vértices. As restrições 13 e 14 são para garantir que em todo vértice da rota tenha apenas um arco chegando e um arco saindo. A restrição 15 assegura que o prêmio coletado na rota será maior ou igual ao premio mínimo pré-estabelecido. As restrições 16 e 17 garantem que a diferença de fluxo que chega e que sai do vértice seja igual a 1 caso o vértice seja visitado, 0 caso contrário e que o fluxo máximo que pode passar por uma aresta é $(n-1)$ (número máximo de arestas possíveis), respectivamente. As restrições 18 e 19 são relativas aos limites de valores que podem ser assumidos pelas váriaveis de decisão, váriaveis que são influenciadas pela presença ou não do vértice na rota.

4.4.4 Implementação do PCV com Coleta de Prêmios numa planilha eletrônica

O modelo matemático do PCVCP, formulado em [3], é classificado como um modelo de Programação Linear Inteira, uma vez que, tanto a função objetivo como as restrições são equações/inequações lineares e as variáveis do modelo somente admitem valores inteiros. A Programação Linear é uma técnica de otimização utilizada para encontrar o ótimo global, seja ele máximo ou mínimo, em situações nas quais temos diversas alternativas de escolha sujeitas a algum tipo de restrição ou regulamentação.

Para a resolução deste modelo de Programação Linear Inteira, fez-se uso do suplemento *Solver* do software *Microsoft Office Excel versão 2007* e da versão demonstração de *LINGO 15.0* (obtida em [2]), e as etapas para a implementação de um modelo de PCVCP em uma planilha eletrônica serão mostradas nas figuras a seguir.

A figura 50 mostra a planilha e as tabelas que foram construídas: uma tabela de distâncias entre cinco cidades do estado de Minas Gerais, uma tabela de bônus ou prêmios associados à cada cidade, uma tabela de penalidades aplicadas no caso da cidade não participar da rota, tabelas referentes às restrições de fluxo, o bônus mínimo pré-estabalecido e a tabela solução. Todas as rotas foram iniciadas na cidade de Belo Horizonte.

PROBLEMA DO CAIXEIRO VIAJANTE COM COLETA DE PRÊMIOS

DISTÂNCIAS (km)

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	0	84	118	84	99
P.Minas	84	0	197	30	172
João Monl	118	197	0	194	198
Itaúna	84	30	194	0	168
Cons.Laf	99	172	198	168	0

VALOR MÁXIMO y

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	0	0	0	0	0
P.Minas	0	0	0	0	0
João Monl	0	0	0	0	0
Itaúna	0	0	0	0	0
Cons.Laf	0	0	0	0	0

BÔNUS ou PRÊMIO

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	12	15	20	13	17
P.Minas	12	15	20	13	17
João Monl	12	15	20	13	17
Itaúna	12	15	20	13	17
Cons.Laf	12	15	20	13	17

Bônus Recolhido 0

Fluxo y

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	0	0	0	0	0
P.Minas	0	0	0	0	0
João Monl	0	0	0	0	0
Itaúna	0	0	0	0	0
Cons.Laf	0	0	0	0	0
Fluxo y	0	0	0	0	0

PENALIDADE

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
Penalidade	30	20	15	5	30

SOLUÇÃO (1:é cidade da rota;0:não) — Rota ij

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf	
BH	0	0	0	0	0	0
P.Minas	0	0	0	0	0	0
João Monl	0	0	0	0	0	0
Itaúna	0	0	0	0	0	0
Cons.Laf	0	0	0	0	0	0
Rota ji	0	0	0	0	0	

Bônus Mínimo 55

Função Objetivo 100

Figura 50: Tabelas para implementação do PCVCP no Excel.

A fórmula da função objetivo do modelo proposto digitada na célula C38 é :

$$=\text{SOMARPRODUTO}(C5{:}G9;C29{:}G33)+\text{SOMARPRODUTO}(C24{:}G24;1-C34{:}G34)$$

Sendo o PCVCP uma variante do PCV, as tabelas referentes às restrições de fluxo e a tabela solução foram construídas seguindo o modelo de PCV clássico, implementado por este trabalho na subseção 4.3.2.

A figura 51 mostra os parâmetros do solver, tais como a função objetivo e as restrições do problema.

A figura 52 mostra a melhor rota, partindo de Belo Horizonte, para a coleta de um bônus mínimo pré-estabelecido no valor de 55. A melhor rota tem o custo total 396, que foi apurado somando as distâncias percorridas no trajeto Belo Horizonte-Pará de Minas-Itaúna-Conselheiro Lafaiete-Belo Horizonte à penalidade por não visitar a cidade de João Monlevade.

A validação do modelo se deu através de vários testes, inclusive para um problema com 6 cidades, incluindo a cidade de Lavras e paralelamente os cálculos foram checados com a utilização da versão demonstração do software *LINGO 15.0*. *LINGO* utiliza uma linguagem de programação própria e trabalha integrado com o *Excel*, utilizando-o como

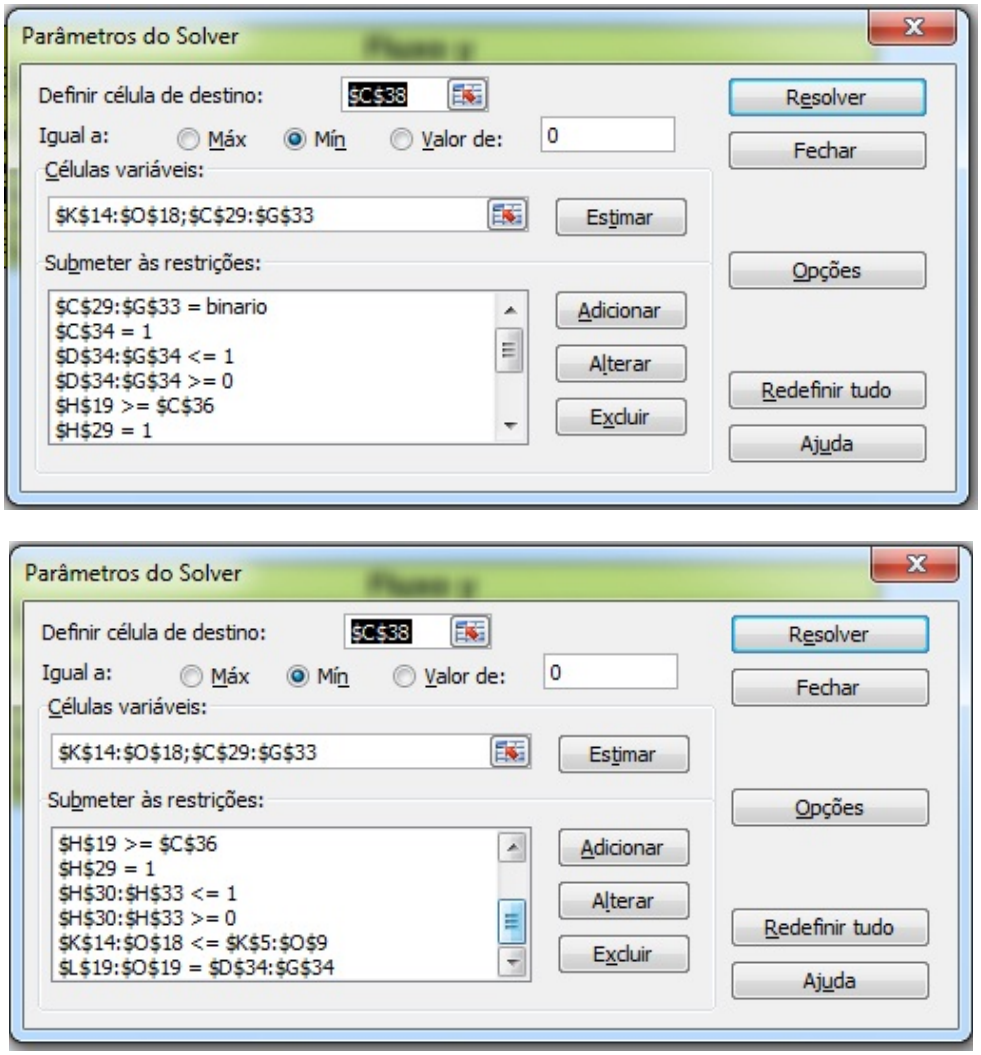


Figura 51: Parâmetros do Solver do Excel para o PCVCP.

base de dados. O dados e parâmetros de configuração do *LINGO* para este modelo de PCVCP foram obtidos em [8]. Após a execução dos cálculos, os resultados são transportados para as tabelas do *Excel*. As figuras 53 e 54 mostram os parâmetros de cálculo do *LINGO* e a tabela do *Excel* com os dados de um problema PCVCP.

Pela natureza combinatorial do problema abordado, observa-se que tal modelo só consegue resolver problemas de pequenas dimensões. Os testes e os dados da tabela 1 apontam para a inviabilidade da modelagem exata do PCVCP à medida que o número de cidades aumenta. Na resolução de problemas com alto nível de complexidade, que envolve um elevado número de rotas a serem analisadas, utilizamos os algoritmos heurísticos. De acordo com [8], os modelos heurísticos, apesar de não garantirem a otimalidade da solução final, têm capacidade de encontrar boas soluções a um custo computacional razoável.

Há muitos algoritmos heurísticos desenvolvidos para a solução do PCV. Em [18] são descritos alguns:

PROBLEMA DO CAIXEIRO VIAJANTE COM COLETA DE PRÊMIOS

DISTÂNCIAS (km)

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	0	84	118	84	99
P.Minas	84	0	197	30	172
João Monl	118	197	0	194	198
Itaúna	84	30	194	0	168
Cons.Laf	99	172	198	168	0

VALOR MÁXIMO y

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	0	4	0	0	0
P.Minas	0	0	0	4	0
João Monl	0	0	0	0	0
Itaúna	0	0	0	0	4
Cons.Laf	4	0	0	0	0

BÔNUS ou PRÊMIO

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	12	15	20	13	17
P.Minas	12	15	20	13	17
João Monl	12	15	20	13	17
Itaúna	12	15	20	13	17
Cons.Laf	12	15	20	13	17
				Bônus Recolhido	57

Fluxo y

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	0	0	0	0	0
P.Minas	0	0	0	1	0
João Monl	0	0	0	0	0
Itaúna	0	0	0	0	2
Cons.Laf	3	0	0	0	0
Fluxo y	-3	1	0	1	1

PENALIDADE

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
Penalidade	30	20	15	5	30

SOLUÇÃO (1:é cidade da rota;0:não)

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf	Rota ij
BH	0	1	0	0	0	1
P.Minas	0	0	0	1	0	1
João Monl	0	0	0	0	0	0
Itaúna	0	0	0	0	1	1
Cons.Laf	1	0	0	0	0	1
Rota ji	1	1	0	1	1	

Bônus Mínimo 55

Função Objetivo 396

Figura 52: Utilização do Solver do Excel no cálculo da melhor rota entre 5 cidades com bônus pré-estabelecido.

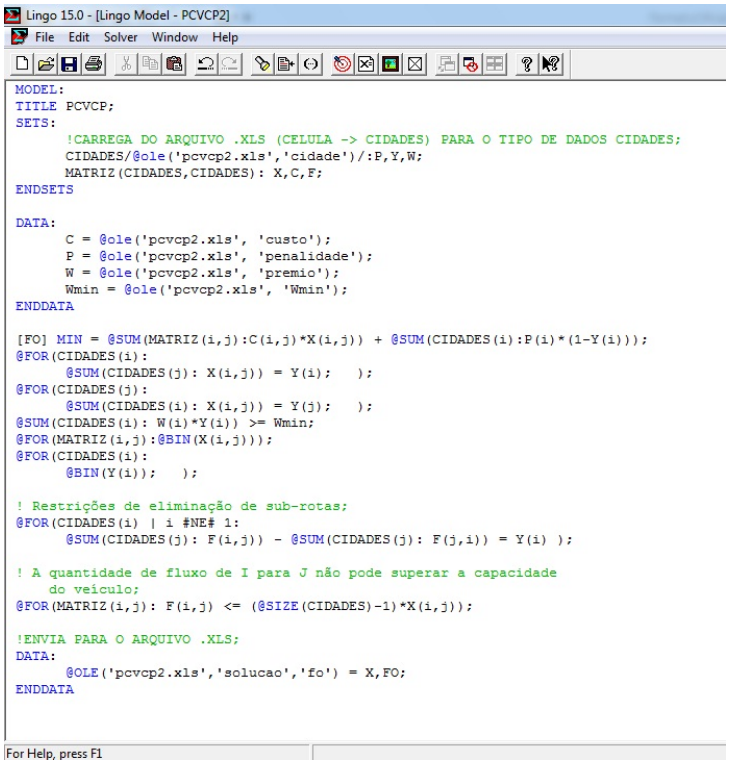


Figura 53: Parâmetros do LINGO para cálculo da melhor rota do PCVCP.

Microsoft Excel - pcvcp2 [Modo de Compatibilidade]

DISTÂNCIAS

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	0	84	118	84	99
P.Minas	84	0	197	30	172
João Monl	118	197	0	194	198
Itaúna	84	30	194	0	168
Cons.Laf	99	172	198	168	0

Cidades	BH	P.Minas	João Monlev	Itaúna	Cons.Laf	Total
Bônus	12	15	20	13	17	77

Cidades	BH	P.Minas	João Monlev	Itaúna	Cons.Laf
Penalidade	30	20	15	5	30

Solução

Cidades	BH	P.Minas	João Monl	Itaúna	Cons.Laf
BH	0	0	0	0	1
P.Minas	1	0	0	0	0
João Monl	0	0	0	0	0
Itaúna	0	1	0	0	0
Cons.Laf	0	0	0	1	0

Bônus Mínimo	55
Função Objetivo	396

Figura 54: Tabela do Excel com dados para cálculo da melhor rota do PCVCP pelo LINGO 15.0.

- Algoritmo de Bellmore & Nemhauser ou heurística do vizinho mais próximo.
- Heurísticas de Inserção, Deslocamentos de Vértices e Inserção de Arestas.
- Heurísticas Exchange ou troca simples.
- Heurísticas de k-substituições.
- Heurística Twice-Around.
- Heurística de Christofides.
- A Técnica de Lin & Kernignan.

A abordagem de técnicas heurísticas escapa dos objetivos deste trabalho que foi desenvolvido focado em modelos exatos de resolução do PCV, aplicáveis no ensino básico da disciplina Matemática.

Na próxima seção, propomos algumas atividades que podem ser ministradas em turmas de ensino médio, com o objetivo de abordar e mostrar as aplicações práticas dos temas apresentados neste trabalho.

5 Propostas de Atividades em Sala de Aula

Com o objetivo de auxiliar os professores de Matemática na abordagem de conceitos e modelos aqui apresentados, sugerimos algumas atividades que poderão ser utilizadas em sala de aula.

Inicialmente expomos nosso entendimento de que os modelos aqui apresentados que foram implementados em planilhas eletrônicas podem ser construídos por alunos do ensino médio, seja para mostrar a aplicabilidade, despertar o interesse pelos temas, como também para validar respostas de exercícios feitos manualmente. Outra ferramenta muito boa a ser explorada são os softwares para construção de grafos. Encontramos alguns disponíveis na versão livre ou demonstração como *yEd Graph Editor*, *Graphviz*, *Grafos*, além do *Geogebra* e *Paint*, esses dois últimos com algumas limitações em relação a grafos.

A seguir, apresentamos um exercício que pode ser proposto para alunos do 2º ou 3º ano do Ensino Médio, dependendo da sequência em que o tema Grafos será inserido no planejamento didático.

5.1 *A família de Pedro mudou-se recentemente para o bairro da Fontinha e ele está ansioso por conhecer os novos colegas da escola do bairro. Mas antes ele terá que resolver um problema: para ir da sua casa à escola, Pedro tem várias opções de trajeto, passando por alguns pontos de referência do bairro e ele quer saber qual desses caminhos é o mais curto. No grafo da figura 55, os vértices representam os pontos de referência do bairro e os arcos, as vias de ligação entre estes pontos. Nos arcos há a indicação das distâncias, dadas em metros.*

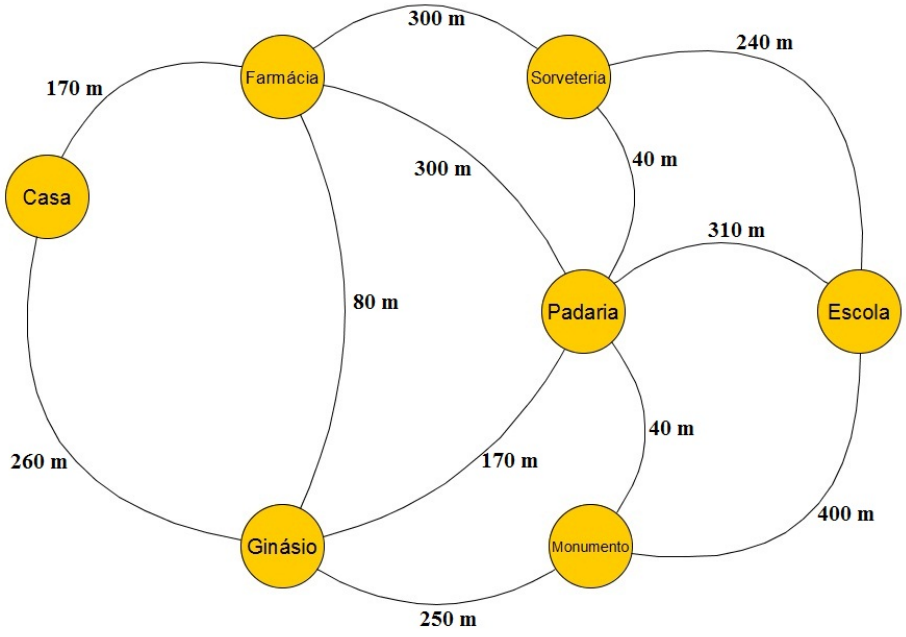


Figura 55: Grafo representando as ligações entre os pontos de referência do bairro da Fontinha.

a) *Escreva a matriz de adjacência de pesos correspondente ao grafo.*

Solução: Na maioria dos planejamentos curriculares, o tópico *Matrizes* é abordado no 2º ano do ensino médio. As entradas de uma matriz de adjacência de pesos de um grafo G guardam informações sobre a conectividade dos vértices e os custos (ou distâncias) associados às arestas. . Representando por w_{ij} a distância, em metros, entre os pontos de referência i e j, a matriz de adjacência R do grafo deste exercício será construída da seguinte forma:

$$R(i,j) = \begin{cases} w_{ij} & \text{se } i \text{ está conectado a } j \\ 0 & \text{caso contrário.} \end{cases}$$

Antes de escrever a matriz, é conveniente o aluno construir a tabela de distâncias abaixo, que auxilia muito na definição das entradas. Para simplificar a escrita , os pontos de referência do grafo serão representados assim: C (Casa), F (Farmácia), G (Ginásio), S (Sorveteria), P (Padaria), M (Monumento) e E (Escola).

	C	F	G	S	P	M	E
C	0	170	260	0	0	0	0
F	170	0	80	300	300	0	0
G	260	80	0	0	170	250	0
S	0	300	0	0	40	0	240
P	0	300	170	40	0	40	310
M	0	0	250	0	40	0	400
E	0	0	0	240	310	400	0

Desta forma, a matriz R deste exercício é apresentada a seguir:

$$R = \begin{bmatrix} 0 & 170 & 260 & 0 & 0 & 0 & 0 \\ 170 & 0 & 80 & 300 & 300 & 0 & 0 \\ 260 & 80 & 0 & 0 & 170 & 250 & 0 \\ 0 & 300 & 0 & 0 & 40 & 0 & 240 \\ 0 & 300 & 170 & 40 & 0 & 40 & 310 \\ 0 & 0 & 250 & 0 & 40 & 0 & 400 \\ 0 & 0 & 0 & 240 & 310 & 400 & 0 \end{bmatrix}$$

b) *Calcule o comprimento do caminho mais curto desde a casa de Pedro até a escola, utilizando o algoritmo de Dijkstra.*

Solução: Inicialmente o aluno deve rotular os vértices do grafo. O vértice inicial C é rotulado com as informações: vértice inicial e o valor zero, que é a distância do vértice inicial a ele mesmo. Os demais vértices são rotulados com o valor zero para representar o vértice anterior no caminho e infinito, representando que a distância entre eles e o vértice inicial ainda não foi calculada. A rotulação inicial do algoritmo é mostrada na figura 56.

A próxima etapa é a busca do caminho mais curto desde C para chegar aos vértices F e G. (figura 57). O caminho mais curto para chegar a G é vindo de F (figura 58). A próxima busca ocorre com os vértices S, P e M (figura 59). Nesse caso, o caminho

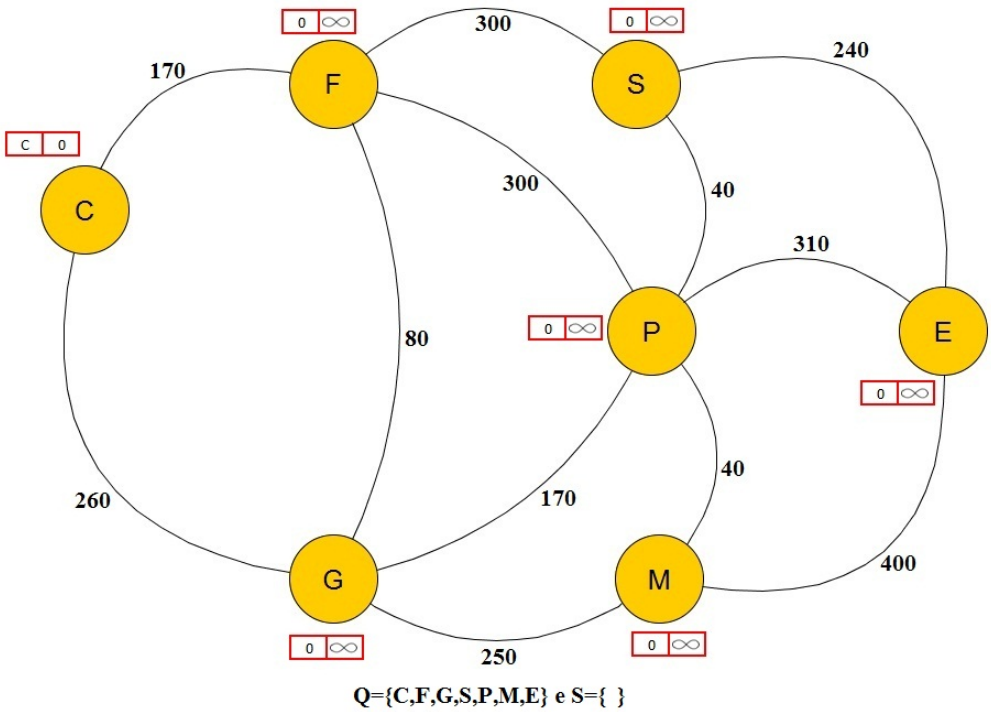


Figura 56: Rotulação inicial do grafo do exercício 5.1 pelo algoritmo de Dijkstra.

mais curto para chegar a S, P e M é vindo de P, G e P, respectivamente (figura 60). A última busca ocorre com o vértice final E (figura 61). A figura 62 mostra o resultado final do algoritmo e o caminho mais curto que Pedro irá percorrer será Casa-Farmácia-Ginásio-Padaria-Sorveteria-Escola com distância total percorrida no valor de 700 metros.

c) *Calcule o comprimento do caminho mais curto desde a casa de Pedro até a escola, construindo uma planilha eletrônica e utilizando o algoritmo de Floyd-Warshall.*

Solução: Seguindo as orientações das notas deste trabalho, o aluno deve construir 8 tabelas numa planilha eletrônica, sendo a primeira delas, a matriz de adjacência dos pesos obtida a partir da fórmula 1 da subseção 3.1.3. As outras 7 tabelas representam as matrizes de iteração do algoritmo e nelas devem ser inseridas as fórmulas de recorrência, cujo padrão é

$$w_{ij}^{k} = \min\{w_{ij}^{k-1}, (w_{ik}^{k-1} + w_{kj}^{k-1})\},$$

onde w_{ij}^{k} é o comprimento do caminho mais curto entre os vértices i e j na $k-$ésima matriz de iteração. Por exemplo, a fórmula a ser inserida na célula X6 da planilha da figura 63 é

$$=\text{MÍNIMO(O6;M6+O6)}$$

A figura 63 mostra a planilha eletrônica com as tabelas, sendo que a matriz da 7ª iteração apresenta o resultado final do algoritmo.

Após a conclusão do exercício, é importante o professor discutir com a turma os resultados obtidos nos itens 5.1.b e 5.1.c, fazendo comparações e mostrando as vantagens

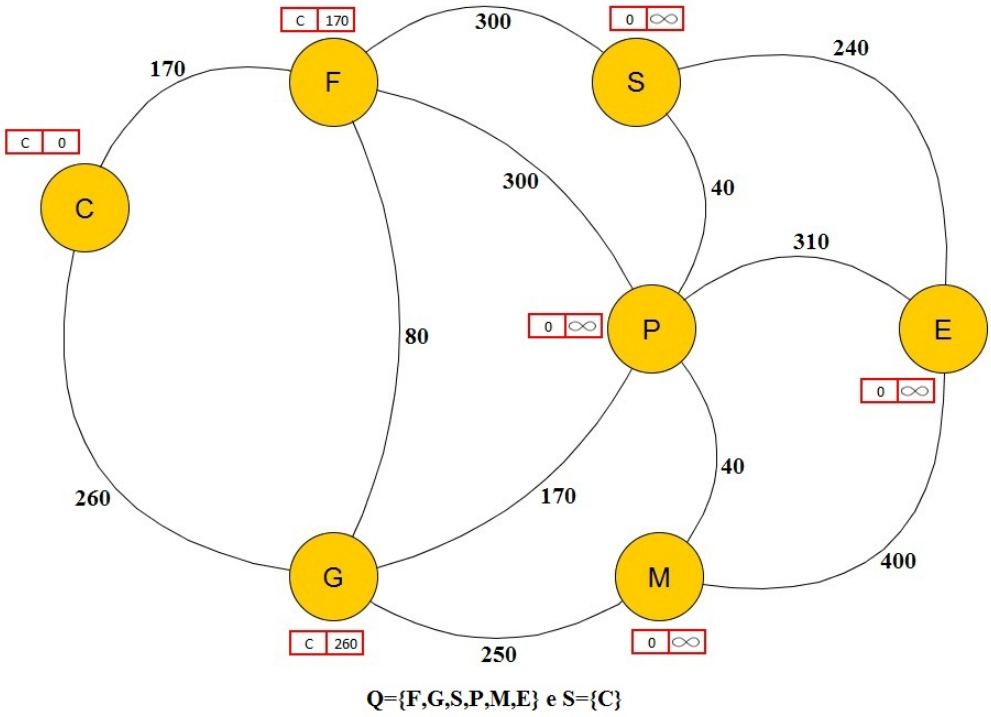


Figura 57: 1ª iteração do algoritmo Dijkstra (exercício 5.1.b).

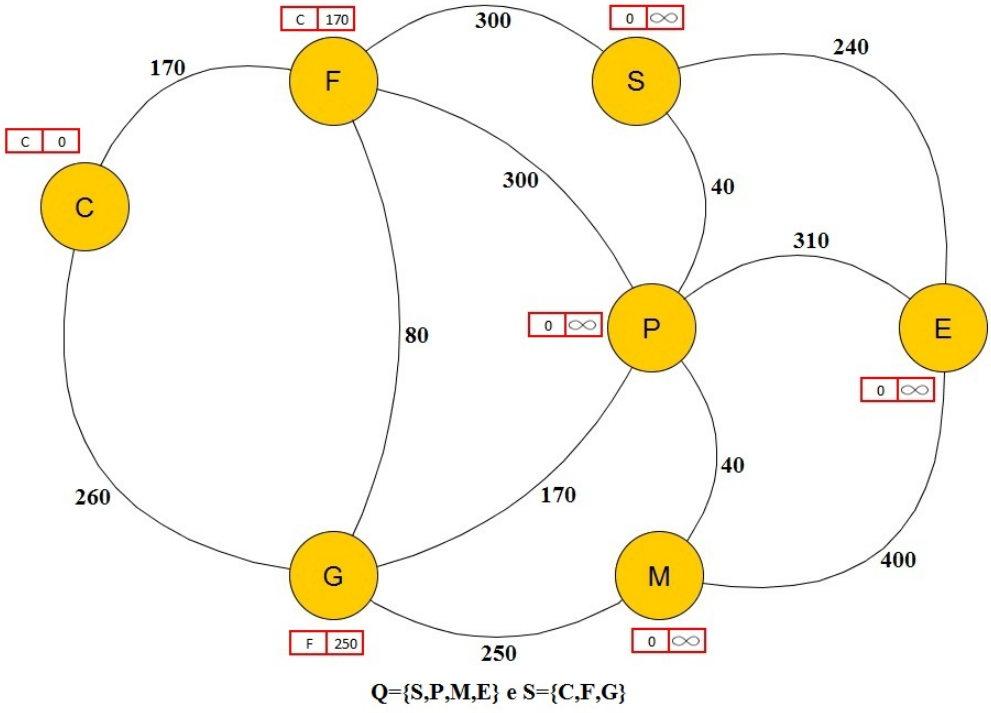


Figura 58: 2ª iteração do algoritmo Dijkstra (exercício 5.1.b).

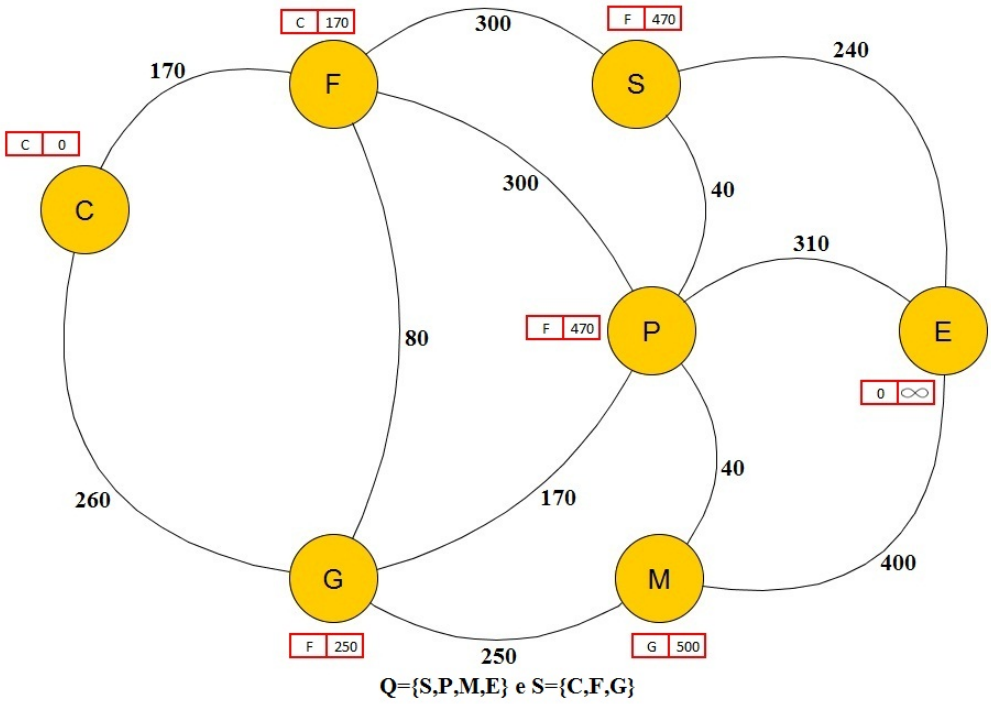


Figura 59: 3ª iteração do algoritmo Dijkstra (exercício 5.1.b).

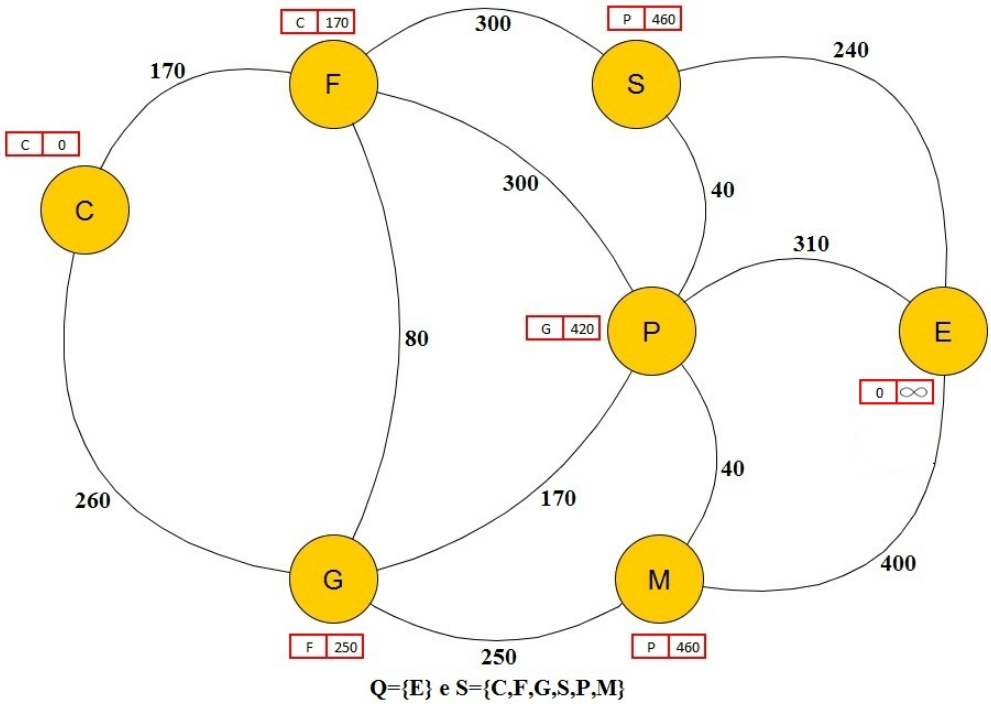


Figura 60: 4ª iteração do algoritmo Dijkstra (exercício 5.1.b).

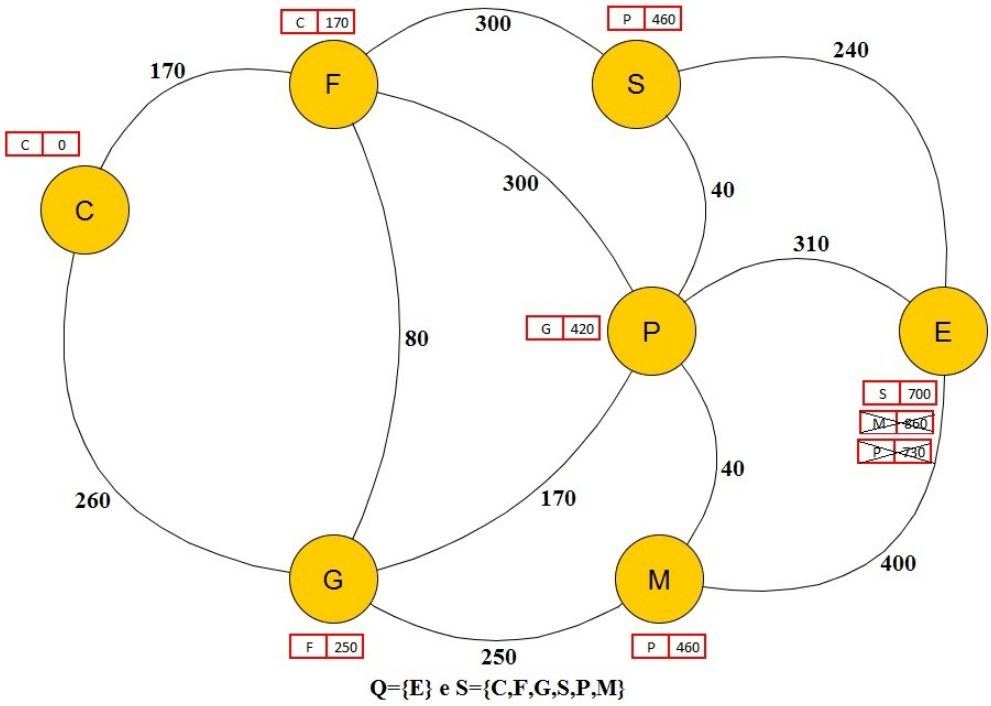


Figura 61: 5ª iteração do algoritmo Dijkstra (exercício 5.1.b).

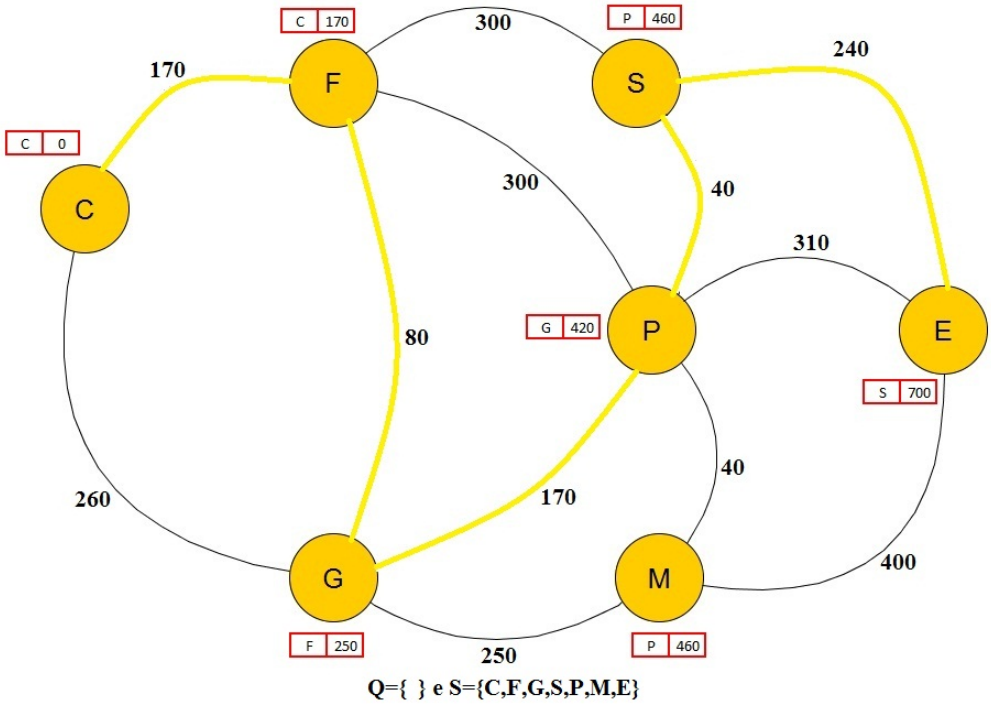


Figura 62: Resultado final do algoritmo Dijkstra (exercício 5.1.b).

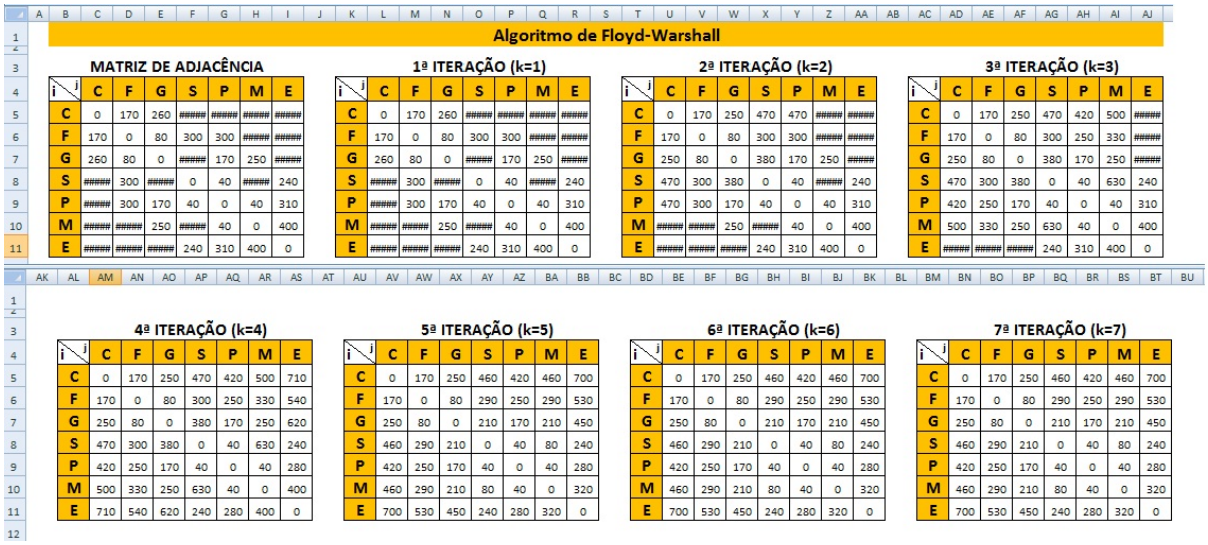


Figura 63: Tabelas de uma planilha eletrônica representando matrizes de iteração e o resultado final do exercício 5.1.c.

e desvantagens da utilização de cada algoritmo e da implementação dos modelos em planilhas eletrônicas.

Apresentamos a seguir, outras sugestões de atividades, descritas de forma sucinta e separadas por seção:

Atividades relativas à seção 2: construção de grafos utilizando softwares, como forma de fixar os conceitos básicos sobre Grafos; construção de matrizes de adjacência ou incidência a partir de um grafo dado ou vice-versa.

Atividades relativas à seção 3: atividades relacionadas a encontrar soluções de caminhos mais curtos em problemas que envolvem o cotidiano dos alunos, utilizando os algoritmos apresentados, que posteriormente devem ser checadas e validadas através das planilhas eletrônicas.

Atividades relativas à seção 4: atividades de combinatória como, por exemplo, a exploração das análises de rotas possíveis do PCV e o tempo que um computador despende para efetuar os cálculos. Atividades para encontrar a rota de custo mínimo do caixeiro viajante envolvendo um número reduzido de cidades e como na seção anterior, as soluções poderão ser checadas e validadas através de planilhas eletrônicas.

Outras atividades para a abordagem dos temas aqui propostos podem ser encontradas em [21], [6] e [18].

Comentários finais

Numa breve pesquisa num encontro de professores de Matemática da cidade de Pará de Minas-MG, questionei se os mesmos abordavam o tema Grafos em suas aulas. Nos relatos constatei que o tema é pouco explorado durante as aulas e que alguns não conheciam conceitos básicos relativos a Grafos. O famoso Problema das Pontes de Königsberg, por exemplo, não era conhecido pela maioria deles. Nessa mesma abordagem procurei encontrar uma justificativa para esse quadro. A ausência do tema em planejamentos curriculares e a escassa exploração por parte dos livros didáticos adotados, apareceram com frequência entre as justificativas apresentadas.

Esse quadro despertou em mim a vontade de dedicar esta dissertação a explorar um tema que trouxesse impacto na prática didática em sala de aula e que fosse pertinente ao currículo de Matemática do ensino básico. Sendo assim, a minha pretensão é que este material sirva de referência na abordagem do tema, principalmente em aulas de Matemática para alunos do ensino médio. O trabalho apresenta conceitos básicos, modelos de implementação em planilhas eletrônicas e sugestões de atividades em sala de aula. Na elaboração utilizou-se uma linguagem simples e de fácil compreensão. A opção de implementação de modelos em planilhas eletrônicas justifica-se pelo fato dessas serem amplamente difundidas, de fácil manipulação e presentes em laboratórios de informática de escolas públicas.

A expectativa é que a proposta apresentada aqui cumpra os objetivos de servir de referência para professores de Matemática na abordagem do tema Grafos no ensino básico e mostrar a relevância e todo o potencial de utilização desse conhecimento.

Por fim, sugerimos outros trabalhos que podem ser desenvolvidos a partir deste, objetivando ampliar as fontes de referência para professores que desejam abordar o tema Grafos em aulas do ensino médio ou cursos de graduação:

- aplicações em grafos eulerianos e coloração de grafos,
- estudo mais detalhado de algoritmos heurísticos para resolução do Problema do Caixeiro Viajante,
- implementação de outros modelos de variantes do PCV em planilhas eletrônicas.

Em trabalhos futuros, pretende-se desenvolver estudo sobre algoritmos exatos e heurísticos utilizados na resolução de problemas de busca e caminhos em grafos.

Agradecimentos

Gostaria de agradecer inicialmente à minha esposa Hérika, meus filhos Bárbara e Arthur e minha irmã Andreia, pelo apoio e paciência. Sei que não foi fácil superar as dificuldades e o sacrifício impostos pela minha ausência em virtude da dedicação aos estudos. Vocês foram o motivo da minha inspiração para buscar mais essa vitória.

Agradeço também aos meus professores e coordenadores da UFSJ Campus Ouro Branco, por proporcionarem aulas de qualidade aliadas a uma relação muito respeitosa e prazerosa. Ao professor orientador Alexandre Celestino Leite Almeida por toda atenção dispensada, e por sempre estar à disposição para contribuir de todas as formas para que alcançasse o meu objetivo.

Aos meus colegas de curso, pelo apoio, companhia em momentos de estudo e incentivo nas horas difíceis.

Não poderia esquecer também de agradecer aos meus grandes amigos Adelson Viegas e Gesiany Faria, companheiros com os quais dividi carona nesses anos de curso e que me ajudaram enfrentar 180 perigosos quilômetros de estrada que separam Pará de Minas de Ouro Branco.

Finalmente, agradeço à CAPES pelo apoio financeiro e oferta do PROFMAT, um programa que contribui muito para o aprimoramento da formação profissional do professor de Matemática do ensino público.

Referências

[1] "Rota de Ouro Branco-MG a Pará de Minas-MG". (22 jul. 2016). Google Maps. Google Inc. Consultado em < https://maps.google.com/ >.

[2] Lingo 15.0 versão demonstração de Lindo Systems Inc. . Disponível em <http://www.lindo.com/index.php?option=com_content&view=article&id=35&Itemid=20>. Acesso em 15 jan. 2016.

[3] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[4] E. Balas and M. Guignard. Report of the session on: Branch and bound/implicit enumeration. *Annals of Discrete Mathematics*, 5:185–191, 1979.

[5] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Mathematical programming*, 59(1-3):413–420, 1993.

[6] P. O. Boaventura and S. Jurkiewicz. *Grafos: introdução e prática*. Editora Blucher, 2009.

[7] I.-M. Chao, B. L. Golden, and E. A. Wasil. The team orienteering problem. *European journal of operational research*, 88(3):464–474, 1996.

[8] A. A. Chaves, F. BIAJOLI, O. MINE, and M. SOUZA. Modelagens exata e heurística para resolução do problema do caixeiro viajante com coleta de prêmios. *Relatório Técnico–DECOM, Universidade Federal de Ouro Preto, Ouro Preto. Disponível em <http://www. decom. ufop. br/prof/marcone/Orientacoes/OrientacoesConcluidas. htm>. Acesso em 08 jan. 2016*, 2003.

[9] N. Christofides. The shortest hamiltonian chain of a graph. *SIAM Journal on Applied Mathematics*, 19(4):689–696, 1970.

[10] N. Christofides. Combinatorial optimization. In *A Wiley-Interscience Publication, Based on a series of lectures, given at the Summer School in Combinatorial Optimization, held in Sogesta, Italy, May 30th-June 11th, 1977, Chichester: Wiley, 1979, edited by Christofides, Nicos*, 1979.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Algoritmos: teoria e prática*. Editora Campus, 2002.

[12] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[13] E. L. de Andrade. *Introdução à pesquisa operacional: métodos e modelos para a análise de decisão*. Grupo Gen-LTC, 2009.

[14] M. Dell'Amico, F. Maffioli, and P. Värbrand. On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research*, 2(3):297–308, 1995.

[15] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation science*, 39(2):188–205, 2005.

[16] R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[17] M. R. Garey and D. S. Johnson. A guide to the theory of np-completeness. *WH Freemann, New York*, 1979.

[18] M. Goldbarg. *Grafos: Conceitos, algoritmos e aplicações*. Elsevier Brasil, 2012.

[19] M. C. Goldbarg and H. P. L. Luna. *Otimização combinatória e programação linear: modelos e algoritmos*. Elsevier, Rio de Janeiro, 2005.

[20] A. d. C. G. Júnior and M. J. F. Souza. Solver (excel): manual de referência. *Universidade Federal de Ouro Preto, MG*, 2004.

[21] S. Jurkiewicz. Grafos–uma introdução. *São Paulo: OBMEP*, 2009.

[22] J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. *Operations research*, 11(6):972–989, 1963.

[23] M. d. S. Menezes. O problema do caixeiro alugador com coleta de bônus: um estudo algorítmico. *Universidade Federal do Rio Grande do Norte*, 2014.

[24] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

[25] M. J. F. Souza. Otimização combinatória - notas de aula. 2010.

[26] J. L. Szwarcfiter. *Grafos e algoritimos computacionais*. Campus, 1986.

[27] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.