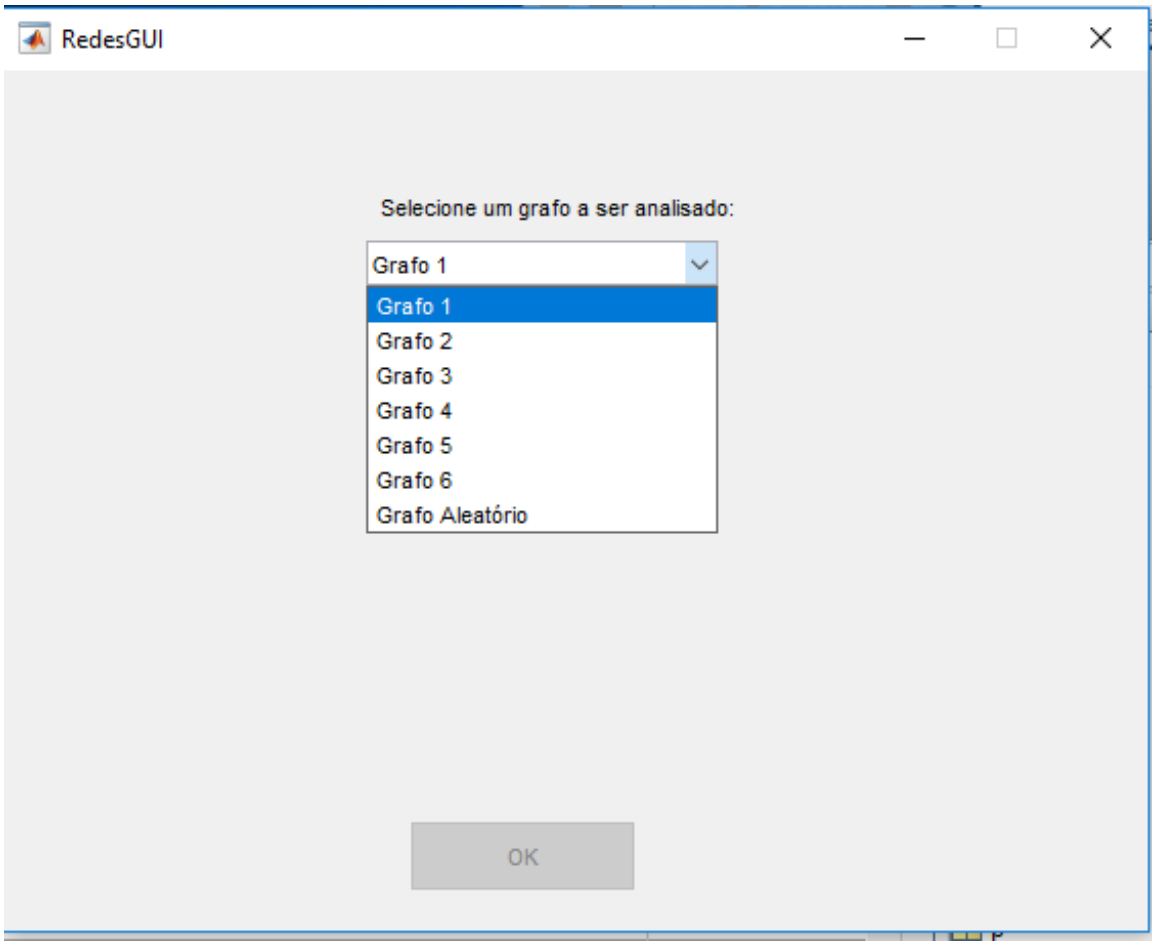
Nome: Guilherme Antonio Pavelski

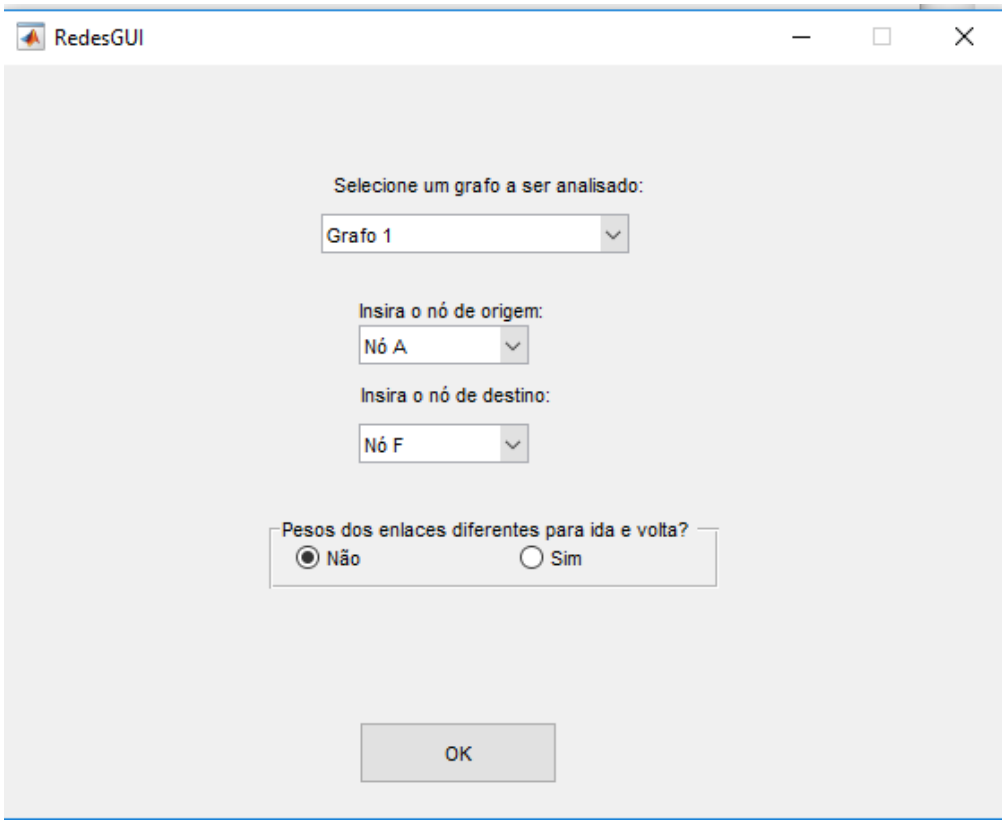
Tarefa – Implementação do algoritmo de Dijkstra

Implementou-se no Matlab uma interface gráfica mostrada na figura a seguir:



Primeiramente o usuário tem a opção de escolher entre 6 modelos de grafos pré-definidos criados a partir de modelos encontrados na internet ou gerar um grafo aleatório, definindo o número de nós e enlaces apenas.

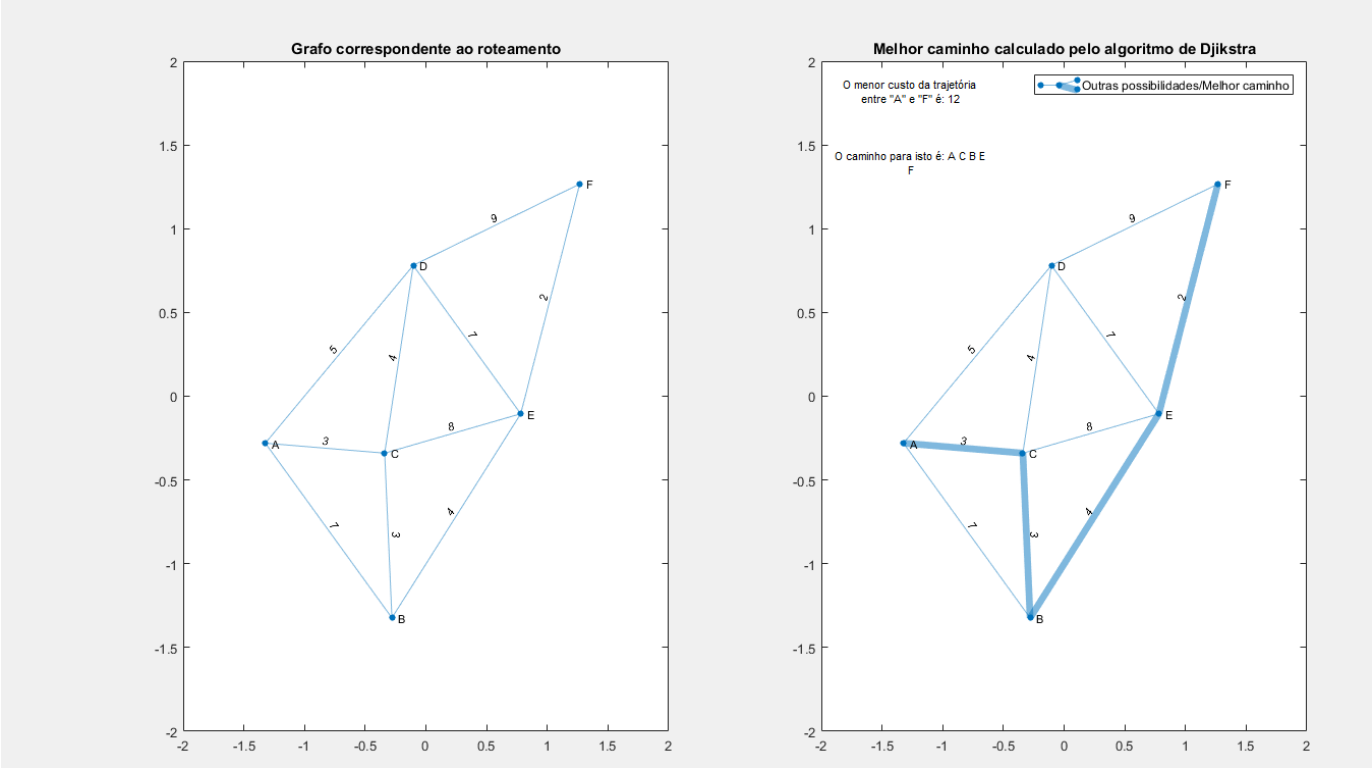
Uma vez selecionado o grafo, novas opções aparecem na tela:



Primeiramente o usuário informa qual o nó de origem, sendo que as possibilidades para escolha são definidas pelo número de nós do grafo escolhido anteriormente. Caso o usuário deseje, pode a qualquer momento mudar o grafo escolhido e as opções de nó de origem e destino se atualizarão automaticamente.

O programa atribui uma letra do alfabeto para cada nó diferente do arquivo de entrada. Uma vez escolhidos os nós de origem e destino, é possível definir se esse grafo deve ser analisado considerando que os pesos dos enlaces são os mesmos para ir e para voltar ou são sempre iguais tanto para ir quanto para voltar.

Ao pressionar OK o programa calcula o melhor caminho, assim como traça o grafo original e o grafo com o melhor caminho destacado.

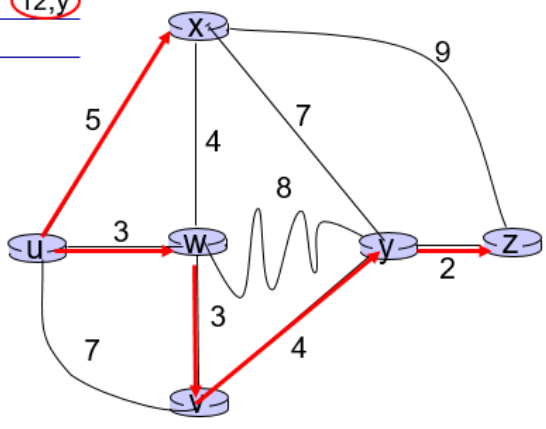


No exemplo acima, o melhor caminho entre “A” e “F” foi calculado pelo algoritmo como sendo A-C-B-E-F e o menor custo encontrado é 12. Esse exercício foi feito baseado em um exemplo do livro do Kurose:

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

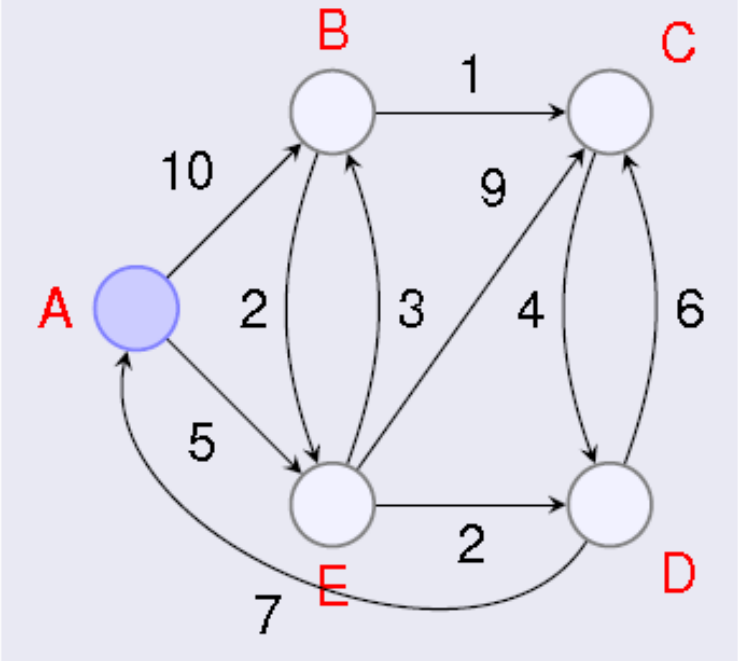
notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

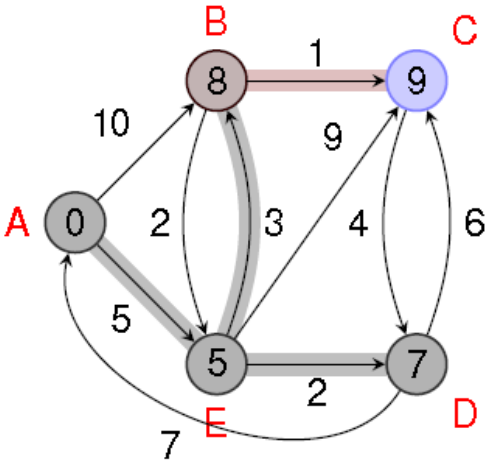


É possível notar que o caminho calculado pelo algoritmo é equivalente ao caminho mostrado na figura do exemplo.

Uma vez plotado, o usuário pode voltar à interface gráfica e alterar qualquer um dos parâmetros e plotar novamente o gráfico. O caso abaixo mostra a resolução do grafo 3, retirado do seguinte exemplo:

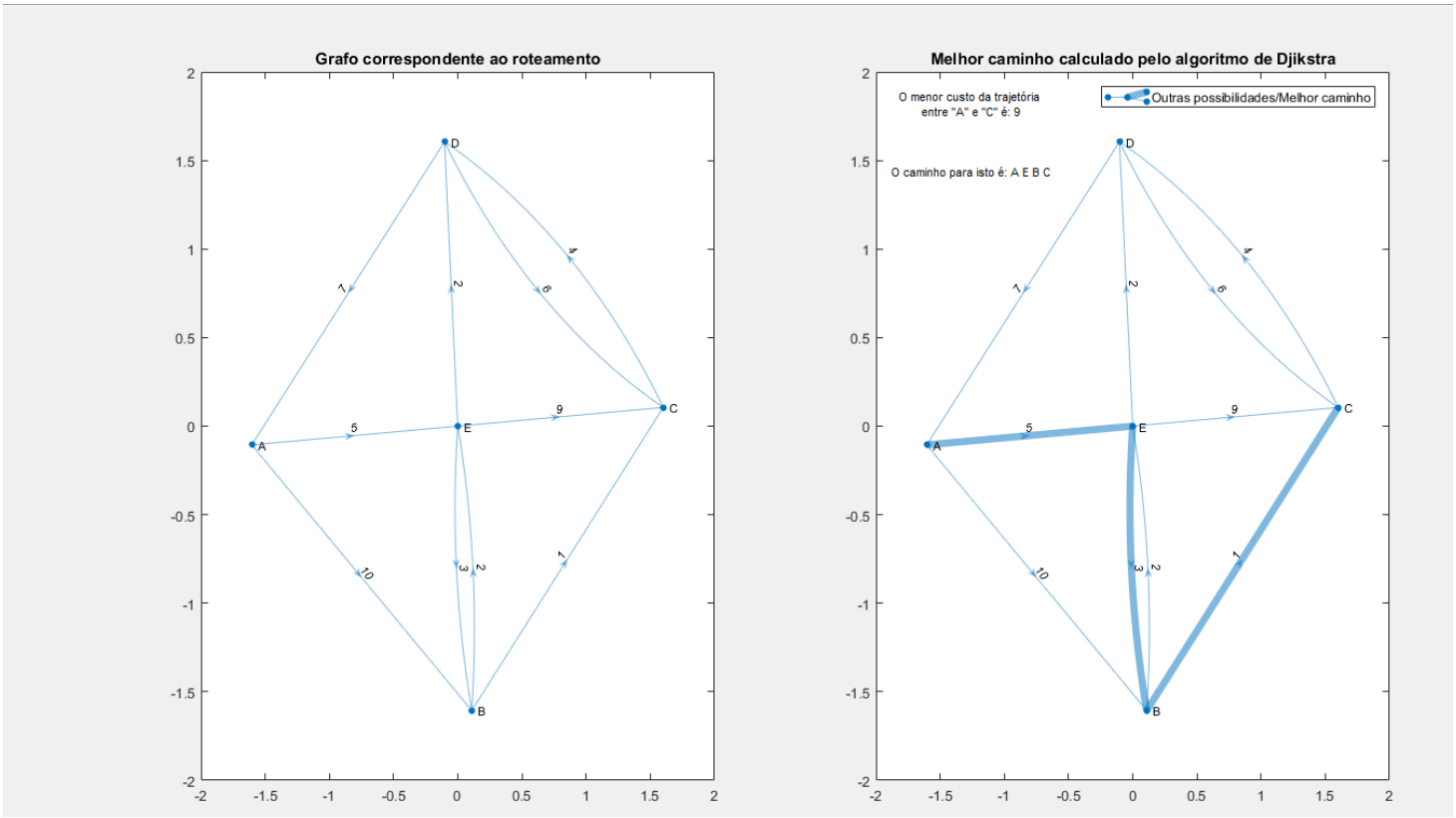


É possível notar que desta vez os pesos para ir e voltar são distintos e, na maior parte dos casos, não é possível voltar pelo caminho de ida. A resposta para esse exemplo é dada a seguir:



A	B	C	D	E
0	∞	∞	∞	∞
•	10_A	∞	∞	5_A
•	8_E	14_E	7_E	•
•	8_E	13_D	•	•
•	•	$\boxed{9_B}$	•	•
•	•	•	•	•

Ou seja, neste exemplo o melhor caminho entre os nós “A” e “C” é A-E-B-C e o custo é 9. Simulando esse grafo no programa criado:



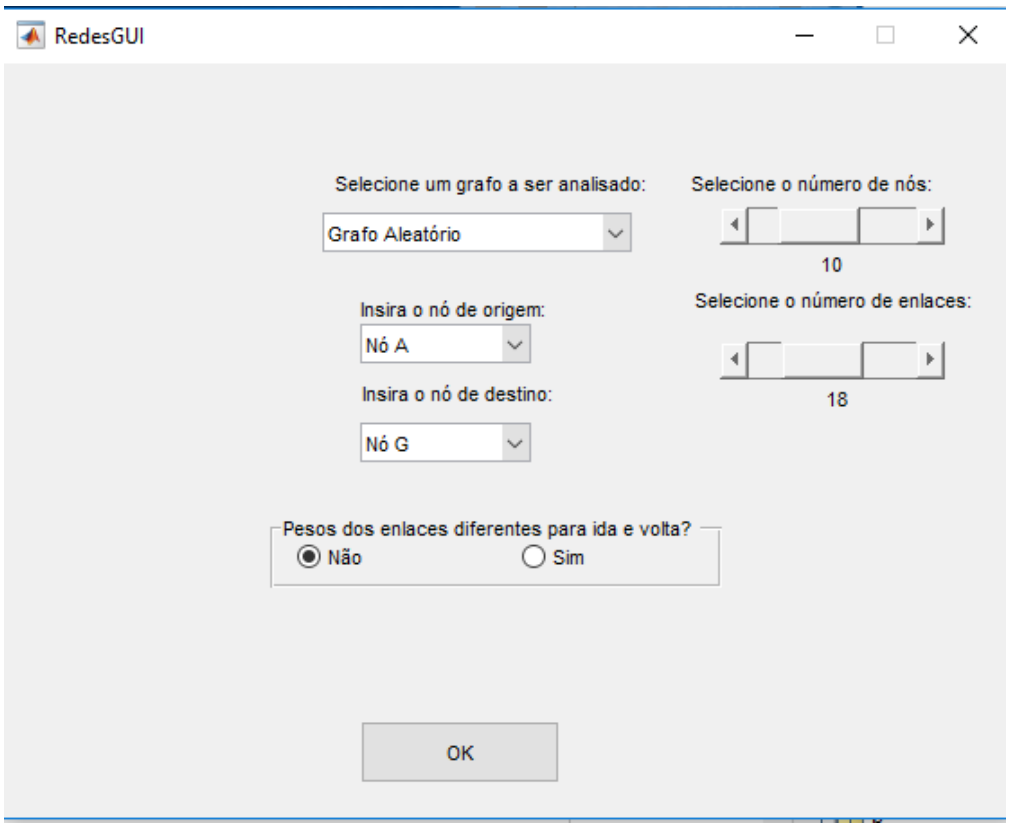
Nota-se que o caminho encontrado foi exatamente A-E-B-C com custo 9, assim, pode-se verificar que o código implementado funciona bem tanto para o caso em que os pesos de ida e volta são iguais quanto para quando são distintos como no exemplo acima.

É importante citar que para o desenvolvimento do trabalho utilizou-se as funções *graph*, *digraph*, *shortestpath* e *highlights*, próprias do Matlab para esse tipo de cálculo.

A outra funcionalidade implementada foi a possibilidade de se gerar aleatoriamente um grafo, para isso escreveu-se um código capaz de gerar um arquivo de texto com o mesmo formato de entrada dos outros arquivos. Por exemplo:

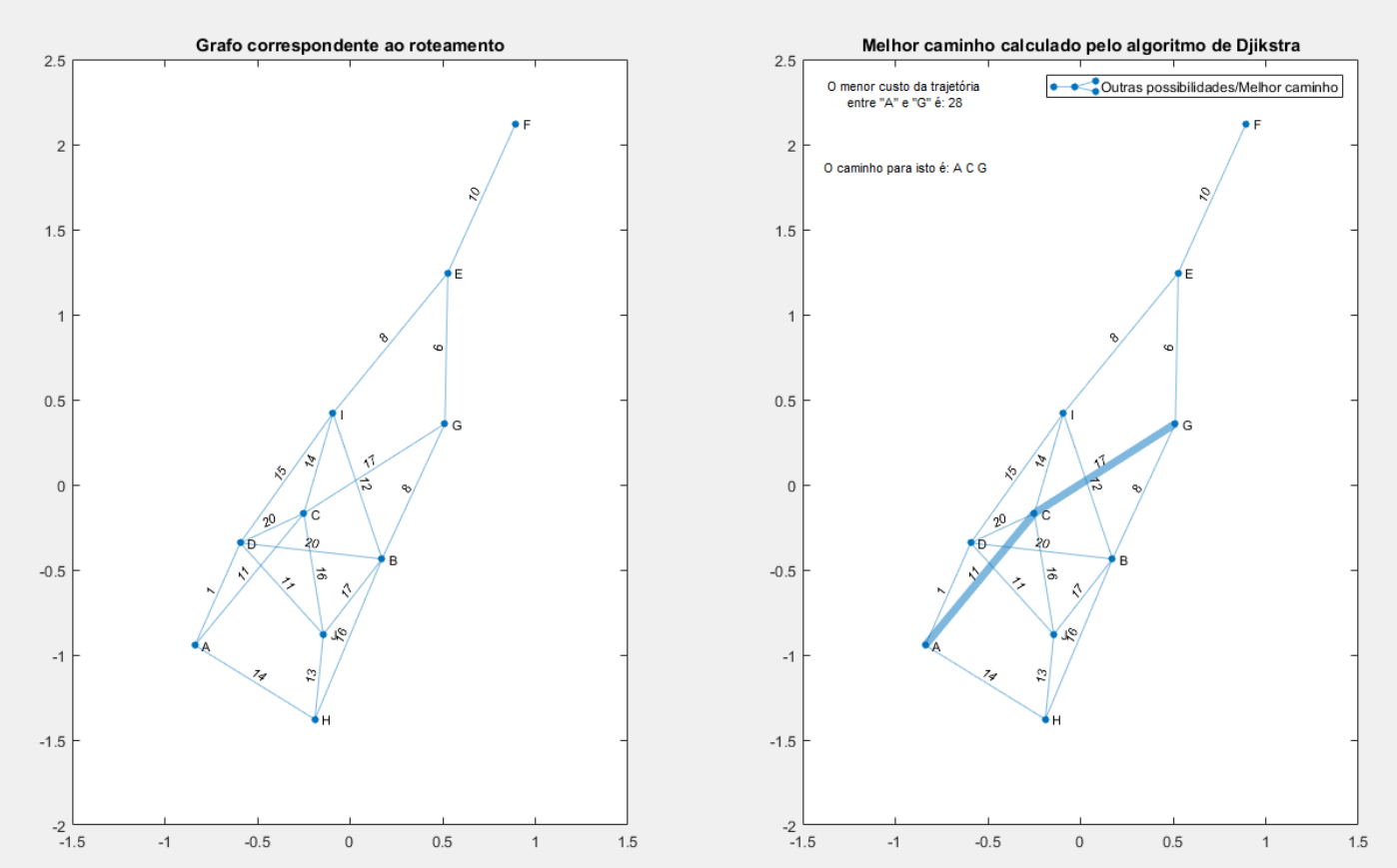
O D P
1 2 10
1 5 5
2 3 1
2 5 2
3 4 4
4 1 7
4 3 6
5 2 3
5 3 9
5 4 2

Onde a primeira coluna indica o nó de origem, a segunda coluna indica o nó de destino e a terceira coluna indica o peso do enlace. As opções para o Grafo aleatório são:



Neste caso o usuário pode selecionar o número de nós através do *Slider*, os valores possíveis foram pré-definidos entre 2 e 26. Também é possível definir o número de enlaces, o valor máximo é atualizado em função do número de nós escolhidos. O Peso dos enlaces, neste caso é sorteado como um número aleatório entre 1 e 20, mas o objetivo é implementar a função em que o usuário define o peso máximo possível de ser sorteado.

Da mesma forma como para os outros grafos, o usuário então escolhe nós de origem, destino e se pode haver mais de um caminho entre dois nós. As possíveis escolhas para nós de origem e destino são atualizadas a cada vez que o usuário modifica o número de nós. Quando clica em “OK”, o programa traça o grafo e define o melhor caminho:



Vale lembrar que, como neste caso o grafo é gerado aleatoriamente, mesmo que não se modifique nenhum parâmetro da interface gráfica, a cada vez que se clica em “OK” um novo grafo aparece na tela.

Quando o nó de destino e origem são os mesmos então o peso calculado é zero e o melhor caminho é o próprio nó.

Quando não há nenhuma possibilidade de se sair de um nó e chegar a outro então o menor custo é dado como infinito e não há nenhum melhor caminho.

Códigos utilizados:

(a) Interface Gráfica

```
functionvarargout = RedesGUI(varargin)
% REDESGUI MATLAB code for RedesGUI.fig
%      REDESGUI, by itself, creates a new REDESGUI or raises the existing
%      singleton*.
%
%      H = REDESGUI returns the handle to a new REDESGUI or the handle to
%      the existing singleton*.
%
%      REDESGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in REDESGUI.M with the given input
arguments.
%
%      REDESGUI('Property','Value',...) creates a new REDESGUI or raises
the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before RedesGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to RedesGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help RedesGUI

% Last Modified by GUIDE v2.5 27-Jun-2017 13:24:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @RedesGUI_OpeningFcn, ...
'gui_OutputFcn',  @RedesGUI_OutputFcn, ...
```

```
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
ifnargin&&ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

ifnargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
end

% --- Executes just before RedesGUI is made visible.
functionRedesGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
%hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%varargin   command line arguments to RedesGUI (see VARARGIN)

% Choose default command line output for RedesGUI
handles.flag = 0;
handles.flagSlider = 0;
setappdata(0, 'Flag', handles.flag);
setappdata(0, 'Flag2', 0);
setappdata(0, 'FlagSlider', handles.flagSlider);
setappdata(0, 'MaxEnlaces',45);
handles.G = [1 1 1 1];
    c = handles.G;
setappdata(0, 'block', c);
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
set(handles.pushbutton1,'enable','off');
Pb = handles.pushbutton1;
setappdata(0, 'Pbutton', Pb);

set(handles.uibuttongroup1,'visible','off');
Bg = handles.uibuttongroup1;
setappdata(0, 'Bgroup', Bg);
% UIWAIT makes RedesGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

end

% --- Outputs from this function are returned to the command line.
functionvarargout = RedesGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
%hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
guidata(hObject, handles);

handles.G = getappdata(0,'block');
    handles.uibuttongroup1 = getappdata(0, 'Bgroup');
handles.numNos = getappdata(0,'NumNos');
handles.numEnlaces = getappdata(0, 'numEnlaces');
flagSlider = getappdata(0, 'FlagSlider');

Bg = handles.uibuttongroup1;
setappdata(0, 'Bgroup', Bg);
    v=get(handles.uibuttongroup1,'SelectedObject');
    w = get(v, 'String');
switch w
case'Não'
handles.G(2) = 1;
case'Sim'
handles.G(2) = 0;
end
ifhandles.G(1) == 3
handles.G(2) = 0;
end
ifflagSlider == 1
nNos = get(handles.numNos, 'Value');
nEnlaces = get(handles.numEnlaces, 'Value');
Redes_Rand_Input(floor(nNos),floor(nEnlaces),handles.G(2),20);
end

    [msg_1 msg_2] = Redes_sem_Fio(handles.G(1),handles.G(2),handles.G(3),
handles.G(4));
uicontrol('Style', 'text', ...
'String', msg_1, ...
'Units', 'pixels', ...
'Position', [825, 660, 152, 71], ...
'BackgroundColor', 'white');

uicontrol('Style', 'text', ...
'String', msg_2, ...
'Units', 'pixels', ...
'Position', [825, 630, 152, 31],...
'BackgroundColor', 'white');
end
% --- Executes on selection change in popupmenu1.
function [c] = popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1

% Determine the selected data set.
str = get(hObject, 'String');
val = get(hObject,'Value');

handles.flag = getappdata(0,'Flag');
flag = handles.flag;
flag2 = getappdata(0,'Flag2');
    flagSlider = getappdata(0, 'FlagSlider');
lmax = getappdata(0, 'MaxEnlaces');

if flag == 1
flag = 0;
        setappdata(0, 'Flag', flag);
        handles.popupdestino = getappdata(0, 'PopUpDest');
set(handles.popupdestino, 'Visible', 'off');

handles.popDestTxt = getappdata(0, 'PopDestTxt');
set(handles.popDestTxt, 'Visible', 'off');

end
if flag2 == 1
        flag2 = 0;
setappdata(0, 'Flag2', flag2);
set(handles.pushbutton1,'enable','off');
        handles.uibuttongroup1 = getappdata(0, 'Bgroup');
set(handles.uibuttongroup1,'visible', 'off');
end

% Set current data to the selected data set.
switchstr{val};
case'Grafo 1'
handles.G(1) = 1;
case'Grafo 2'
handles.G(1) = 2;
case'Grafo 3'
handles.G(1) = 3;
case'Grafo 4'
handles.G(1) = 4;
case'Grafo 5'
handles.G(1) = 5;
case'Grafo 6'
handles.G(1) = 6;
case'GrafoAleatório'
handles.G(1) = 255;
end

ifhandles.G(1) == 255

handles.numNosTxt = uicontrol('Style', 'text', ...
'String', 'Selecione o número de nós: ', ...
```

```
'Units', 'pixels', ...
'Position', [380, 340, 145, 21]);

handles.numNos = uicontrol('Style', 'slider', ...
'Min', 2, 'Max', 26, 'Value', 10, ...
'SliderStep', [1/24 , 1],...
'Units', 'pixels', ...
'Position', [400, 320, 125, 21], ...
'Callback', @numNos);

sliderValue = get(handles.numNos,'Value');


lmax = getappdata(0,'MaxEnlaces');

    handles.numNosTxt2 = uicontrol('Style', 'text', ...
'String', sliderValue, ...
'Units', 'pixels', ...
'Position', [400, 295, 125, 21]);

handles.numEnlacesTxt = uicontrol('Style', 'text', ...
'String', 'Selecione o número de enlaces: ', ...
'Units', 'pixels', ...
'Position', [380, 275, 170, 21]);

handles.numEnlaces = uicontrol('Style', 'slider', ...
'Min', 2, 'Max', lmax, 'Value', 2,...
'SliderStep', [1/(lmax-2), 1],...
'Units', 'pixels', ...
'Position', [400, 245, 125, 21], ...
'Callback', @numEnlaces);

sliderEnlacesValue = get(handles.numEnlaces, 'Value');

          handles.numEnlTxt2 = uicontrol('Style', 'text', ...
'String', sliderEnlacesValue, ...
'Units', 'pixels', ...
'Position', [380, 220, 170, 21]);

setappdata(0, 'numEnlacesTxt',handles.numEnlacesTxt);
setappdata(0, 'numEnlaces', handles.numEnlaces);
setappdata(0, 'numEnlacesTxt2',handles.numEnlTxt2);
setappdata(0, 'NumNos', handles.numNos);
setappdata(0, 'NumNosTxt', handles.numNosTxt);
setappdata(0, 'NumNosTxt2', handles.numNosTxt2);

flagSlider = 1;
setappdata(0, 'FlagSlider', flagSlider);


elseifflagSlider == 1

set(handles.numNos,'visible','off');
set(handles.numNosTxt, 'visible', 'off');
```

```
set(handles.numNosTxt2, 'visible', 'off');
set(handles.numEnlacesTxt, 'visible', 'off');
set(handles.numEnlTxt2, 'visible', 'off');
set(handles.numEnlaces,'visible', 'off');
flagSlider = 0;
setappdata(0, 'FlagSlider', flagSlider);
end

handles.G(3) = 1;
Arquivo = 'Input_Grafo';


    Entrada = [Arquivo, num2str(handles.G(1)), '.txt'];
    A = importdata(Entrada);

    n=1;
    n = max(unique(A.data(:,1:2)));
ifhandles.G(1) == 255
        n = 10;
end
    Alphabet = char('A' + (1:n)-1)';
nomes = cellstr(Alphabet)';

str = sprintf('Nó %c|', nomes{1:n});
str(end) = [];

CString = regexp(str, '#', 'split');
uicontrol('Style', 'text', ...
'String', 'Insira o nó de origem: ', ...
'Units', 'pixels', ...
'Position', [200, 270, 105, 21], ...
'Callback', @popuporigem);

handles.poporigem = uicontrol('Style', 'popupmenu', ...
'String', str, ...
'Units', 'pixels', ...
'Position', [200, 255, 95, 21], ...
'Callback', @popuporigem);

    setappdata(0, 'PopUpOrig', handles.poporigem);
% Save the handles structure.
guidata(hObject,handles)
drawnow

setappdata(0, 'block', handles.G);
setappdata(0, 'text', str);
end

function [c] = popuporigem(hObject, eventdata, handles)

flag = getappdata(0,'Flag');
    flag2 = getappdata(0,'Flag2');
    flagSlider = getappdata(0, 'FlagSlider');
    handles.numNos = getappdata(0, 'NumNos');
    handles.numNosTxt = getappdata(0, 'NumNosTxt');
```

```
    handles.numNosTxt2 = getappdata(0, 'NumNosTxt2');
    handles.numEnlacesTxt = getappdata(0, 'numEnlacesTxt');
    handles.numEnlTxt2 = getappdata(0, 'numEnlacesTxt2');
    handles.numEnlaces = getappdata(0, 'numEnlaces');
    handles.poporigem = getappdata(0, 'PopUpOrig');


if flag == 1
       flag = 0;
       handles.popupdestino = getappdata(0, 'PopUpDest');
       handles.popDestTxt = getappdata(0, 'PopDestTxt');
setappdata(0, 'Flag', flag);
set(handles.popDestTxt, 'visible','off');
set(handles.popupdestino, 'visible', 'off');
end

if flag2 == 1
        flag2 = 0;
        handles.pushbutton1 = getappdata(0, 'Pbutton');
setappdata(0, 'Flag2', flag2);
set(handles.pushbutton1,'enable','off');
        handles.uibuttongroup1 = getappdata(0, 'Bgroup');
set(handles.uibuttongroup1,'visible', 'off');
end

drawnow
handles.G = getappdata(0,'block');
    Opt = getappdata(0, 'text');

    str2 = get(hObject, 'String');
val = get(hObject,'Value');
str = cellstr(str2);
    h = str{val};
    [token remain]= strtok(h, ' ');
    b = strtrim(remain);
handles.G(3) = double(b) -64;
handles.G(4) = 1;

handles.flag = 1;
setappdata(0,'Flag',handles.flag);

handles.popDestTxt = uicontrol('Style', 'text', ...
'String', 'Insira o nó de destino: ', ...
'Units', 'pixels', ...
'Position', [200, 223, 110, 21]);

handles.popupdestino = uicontrol('Style', 'popupmenu', ...
'String', Opt, ...
'Units', 'pixels', ...
'Position', [200, 200, 95, 21], ...
'Callback', @popupdestino);

% Save the handles structure.
guidata(hObject,handles);
setappdata(0, 'block', handles.G);
```

```
setappdata(0, 'PopUpDest', handles.popupdestino);
setappdata(0, 'PopDestTxt', handles.popDestTxt);
setappdata(0, 'NumNos', handles.numNos);
setappdata(0, 'NumNosTxt', handles.numNosTxt);
setappdata(0, 'NumNosTxt2', handles.numNosTxt2);
setappdata(0, 'numEnlacesTxt',handles.numEnlacesTxt);
setappdata(0, 'numEnlacesTxt2',handles.numEnlTxt2);
setappdata(0, 'numEnlaces', handles.numEnlaces);
setappdata(0, 'PopUpOrig', handles.poporigem);
    setappdata(0, 'text', str2);
    flag2 = getappdata(0, 'Flag2');
    setappdata(0, 'Flag2', flag2);
setappdata(0, 'FlagSlider', flagSlider);
%guidata(hObject,handles)
end

function [c] = popupdestino(hObject, eventdata, handles)
handles.G = getappdata(0,'block');
    handles.pushbutton1 = getappdata(0, 'Pbutton');
    handles.uibuttongroup1 = getappdata(0, 'Bgroup');
handles.numNos = getappdata(0, 'NumNos');
handles.numNosTxt = getappdata(0, 'NumNosTxt');
    handles.numNosTxt2 = getappdata(0, 'NumNosTxt2');
handles.numEnlacesTxt = getappdata(0, 'numEnlacesTxt');
    handles.numEnlTxt2 = getappdata(0, 'numEnlacesTxt2');
handles.numEnlaces = getappdata(0, 'numEnlaces');
    handles.poporigem = getappdata(0, 'PopUpOrig');
    handles.popupdestino = getappdata(0, 'PopUpDest');
    str2 = getappdata(0, 'text');
val = get(handles.popupdestino, 'Value');
flagSlider = getappdata(0, 'FlagSlider');
str = cellstr(str2);
    String = str{val};
    [token remain]= strtok(String, ' ');
    b = strtrim(remain);
handles.G(4) = double(b) -64;

ifhandles.G(1) ~= 3
Bg = handles.uibuttongroup1;
setappdata(0, 'Bgroup', Bg);
set(handles.uibuttongroup1,'visible','on')
else
handles.G(2) = 0;
set(handles.uibuttongroup1,'visible','off')
end
guidata(hObject,handles);
    c = handles.G;
setappdata(0, 'block', c);
set(handles.pushbutton1,'enable','on')
Pb = handles.pushbutton1;
setappdata(0,'Pbutton', Pb);

flag2 = 1;
    setappdata(0,'Flag2',flag2);
    flag = getappdata(0, 'Flag');
    setappdata(0, 'Flag', flag);
```

```
    setappdata(0, 'FlagSlider', flagSlider);
    setappdata(0, 'NumNos', handles.numNos);
    setappdata(0, 'NumNosTxt', handles.numNosTxt);
    setappdata(0, 'NumNosTxt2', handles.numNosTxt2);
    setappdata(0, 'numEnlacesTxt',handles.numEnlacesTxt);
    setappdata(0, 'numEnlacesTxt2',handles.numEnlTxt2);
    setappdata(0, 'numEnlaces', handles.numEnlaces);
    setappdata(0, 'PopUpOrig', handles.poporigem);
    setappdata(0,'PopUpDest',handles.popupdestino);
end

functionnumNos(hObject, eventdata, handles)
handles.G = getappdata(0,'block');
flagSlider = getappdata(0, 'FlagSlider');
flag = getappdata(0, 'Flag');
handles.numNos = getappdata(0, 'NumNos');
sliderValue = get(handles.numNos, 'Value');
    handles.numNosTxt2 = getappdata(0, 'NumNosTxt2');
handles.poporigem = getappdata(0, 'PopUpOrig');
handles.popupdestino = getappdata(0,'PopUpDest');
set(handles.numNosTxt2, 'String', floor(sliderValue));
    n = floor(sliderValue);
    Alphabet = char('A' + (1:n)-1)';
nomes = cellstr(Alphabet)';

str = sprintf('Nó %c|', nomes{1:n});
str(end) = [];
set(handles.poporigem, 'String', str);
set(handles.poporigem, 'Value', 1);
handles.G(3) = 1;
if flag == 1
set(handles.popupdestino, 'String', str);
set(handles.popupdestino,'Value', 1);
setappdata(0,'PopUpDest',handles.popupdestino);
handles.G(4) = 1;
end
handles.numEnlacesTxt= getappdata(0, 'numEnlacesTxt');
    handles.numEnlTxt2 = getappdata(0, 'numEnlacesTxt2');
handles.numEnlaces = getappdata(0, 'numEnlaces');
lmax = nchoosek(floor(sliderValue),2); %Número máximo de enlaces.
set(handles.numEnlaces,'Max',lmax);
set(handles.numEnlaces,'Value',lmax);
setappdata(0, 'block', handles.G);

iflmax ~= 1
set(handles.numEnlaces, 'SliderStep', [1/(lmax-2),1]);
else
set(handles.numEnlaces, 'SliderStep', [1,1]);
end
set(handles.numEnlTxt2,'String',lmax);
setappdata(0,'MaxEnlaces',lmax);
setappdata(0,'NumNos',handles.numNos);
setappdata(0,'NumNosTxt2',handles.numNosTxt2);
setappdata(0,'numEnlacesTxt', handles.numEnlacesTxt);
setappdata(0,'numEnlaces',handles.numEnlaces);
setappdata(0,'numEnlacesTxt2',handles.numEnlTxt2);
```

```
setappdata(0, 'FlagSlider', flagSlider);
    setappdata(0, 'PopUpOrig', handles.poporigem);
setappdata(0, 'block', handles.G);


end

functionnumEnlaces(hObject, eventdata, handles)
handles.G = getappdata(0,'block');
flagSlider = getappdata(0, 'FlagSlider');
handles.numNos = getappdata(0, 'NumNos');
    handles.numNosTxt2 = getappdata(0, 'NumNosTxt2');
handles.numEnlacesTxt= getappdata(0, 'numEnlacesTxt');
handles.numEnlaces = getappdata(0, 'numEnlaces');
    handles.numEnlTxt2 = getappdata(0, 'numEnlacesTxt2');
    handles.poporigem = getappdata(0, 'PopUpOrig');
    handles.popupdestino = getappdata(0,'PopUpDest');
sliderEnlaces = get(handles.numEnlaces, 'Value');
setappdata(0,'SliderEnlaces', sliderEnlaces);
set(handles.numEnlTxt2, 'String', floor(sliderEnlaces));
setappdata(0,'numEnlacesTxt', handles.numEnlacesTxt);
setappdata(0,'numEnlaces',handles.numEnlaces);
setappdata(0,'numEnlacesTxt2',handles.numEnlTxt2);
setappdata(0, 'FlagSlider', flagSlider);
setappdata(0, 'NumNos', handles.numNos);
setappdata(0, 'PopUpOrig', handles.poporigem);
setappdata(0,'PopUpDest',handles.popupdestino);
setappdata(0,'block', handles.G);
end

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
ifispc&&isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

end
```

(b) Gerando grafo aleatório:

```
function Redes_Rand_Input( n, l, i, Pmax )

%% Descrição:
```

```
%O objetivo deste programa é criar uma função que crie automaticamente um
%arquivo de entrada que represente um grafo que será posteriormente
%utilizado na função Redes_sem_Fio. Esse arquivo deve ser gerado com
pesos
%e enlaces aleatórios a partir de dados de entrada fornecidos pelo
usuário.

%% Inicialização das variáveis

Int = 0;

lmin = n - 1;
v = 1:n;
C = nchoosek(v,2);
If i == 1
lmax = nchoosek(n,2); %Número máximo de enlaces
elseif i == 0
    lmax = 2*nchoosek(n,2); %Número máximo de enlaces
C = union(C,fliplr(C),'rows');
end


Int = randperm(lmax,l);

%% Escreve o arquivo de saída

fileID = fopen('Input_Grafo255.txt','w');
fprintf(fileID,'%0s %1s %1s\r\n','O','D', 'P');
K = 1;
while K <= length(Int)
fprintf(fileID,'%0d %1d %1d\r\n',C(Int(K),1), C(Int(K),2),  randi([1
Pmax]));
    K = K + 1;
end
fclose(fileID);
```

(c) Calculando o melhor caminho e traçando as figuras:

```
function [ msg_1 msg_2 ] = Redes_sem_Fio( GraphNumber, x, P_0, P_f )

%% Inicialização das variáveis
Arquivo = 'Input_Grafo';
Entrada = [Arquivo, num2str(GraphNumber), '.txt'];
A = importdata(Entrada);  %Importa os dados a partir de um arquivo de
texto
%x = 1; %x = 0 ->> sentido único, x = 1 ->> duplo-sentido (ida e volta);
%P_0 = 1;  %Nó de partida para o algoritmo (a = 1, b = 2, ...)
%P_f = 5;  %Nó de destino para o algoritmo (a = 1, b = 2, ...)

%% Obtenção da matriz de dados
```

```
K = zeros(max(unique(A.data(:,1:2))), max(unique(A.data(:,1:2))));
%Inicializa a matriz X

orig = A.data(:,1)'; %Define os enlaces origem a partir da 1a coluna do
arquivo de texto
dest = A.data(:,2)'; %Define os enlaces destino a partir da 2a coluna do
arquivo de texto
peso = A.data(:,3)'; %Define os enlaces peso a partir da 3a coluna do
arquivo de texto

Alphabet=char('A'+(1:max(unique(A.data(:,1:2))))-1)'; %Cria uma letra que
possa ser atribuída a cada nó
%Obs.:Caso o número de nós seja maior do que 26, repensar esse comando
nomes = cellstr(Alphabet)'; %Dá nome aos bois

for k = 1:size((A.data),1)

K(orig(k),dest(k)) = peso(k); %Cria uma matriz correspondente aos pesos
de cada enlace
if x == 1
K(dest(k),orig(k)) = peso(k);
end
end

%% Cálculo do melhor caminho pelo algoritmo de Djikstra

if x == 0
    G = digraph(K);
elseif x == 1
    G = graph(K);
end

 [L e] = shortestpath(G,P_0,P_f);
%[e L] = dijkstra(K,P_0,P_f);  %Utiliza a função para resolver o
algoritmo de Djikstra para encontrar o melhor caminho
% e = menor custo
% L = caminho de menor custo

if e == Inf
     L = P_0;
end

Ld = L; %O algoritmo dá o melhor caminho de trás para frente, essa função
apenas inverte a ordem

  %% Apresentação dos resultados

 msg_1 = ['O menor custo da trajetória entre "', strjoin(nomes(P_0)),'" e
"', strjoin(nomes(P_f)),'" é: ', num2str(e)];
%disp(msg_1); %Mostra o resultado através de mensagem no prompt de
comando
if e == Inf
    msg_2 = ['Não há melhor caminho, sentido impossível'];
else
```

```
     msg_2 = ['O caminho para isto é: ', strjoin(nomes(Ld))];
end

%disp(msg_2); %Mostra o melhor caminho através de mensagem no prompt de
comando

  B = ones(length(peso),3); %Cria uma matriz auxiliar para facilitar a
plotagem do melhor caminho
  B(:,1) = orig;    %Atribui à primeira coluna da matriz B os nós de
origem dados na entrada
  B(:,2) = dest;    %Atribui à segunda coluna da matriz B os nós de
destino dados na entrada

  m = 1; %Inicializa a variável m, que servirá como contador no loop a
seguir
  k = 1; %Inicializa a variável k, que conta as linhas no loop a seguir
while k <= length(peso)
if m == length(Ld)
break;  %Quando o número de enlaces de maior peso for igual ao tamanho do
vetor Ld, para o loop

elseif x == 0
if B(k,1) == Ld(m) && B(k,2) == Ld(m+1)
B(k,3) = 15;    %Caso o enlace faça parte do melhor caminho calculado
anteriormente, atribui peso maior
                m = m + 1;      %Caso um enlace já tenha sido
incrementado de um peso maior, soma 1
k= 0;
end
elseif x == 1
if B(k,1) == Ld(m) && B(k,2) == Ld(m+1) || B(k,1) == Ld(m+1) && B(k,2) ==
Ld(m)
B(k,3) = 15;    %Caso o enlace faça parte do melhor caminho calculado
anteriormente, atribui peso maior
                m = m + 1;      %Caso um enlace já tenha sido
incrementado de um peso maior, soma 1
                k= 0;
end
end

    k = k + 1; %Incrementa o loop
end
figure(1);
set(gcf, 'units','normalized','outerposition',[0 0 1 1]);
subplot(1,2,1); %Divide a figura em duas partes: Grafo antes do algoritmo
e depois
G = digraph(orig,dest,peso,nomes); %Gera o grafo original
if x == 1
    clear G;
    G = graph(orig,dest,peso,nomes); %Gera o grafo original
end
plot(G,'EdgeLabel', G.Edges.Weight) %Plota o grafo original
title('Grafo correspondente ao roteamento'); %Título da figura
p = G.Edges.Weight; %Salvo os pesos originais em uma variável p

subplot(1,2,2); %Muda para a segunda parte da figura
```

```
G = digraph(orig,dest,B(:,3),nomes); %Gera o grafo destacando melhor
caminho
if x == 1
    clear G;
    G = graph(orig,dest,B(:,3),nomes); %Gera o grafo destacando melhor
caminho
end
if e ~= 0 && e ~= Inf
    LWidths = 5*G.Edges.Weight/max(G.Edges.Weight); %Atribui espessura
maior para o melhor caminho
    plot(G, 'EdgeLabel', p, 'LineWidth',LWidths) %Plota o grafo com
melhor rota utilizando os pesos originais
elseif e== 0
    h = plot(G, 'EdgeLabel', p); %Plota o grafo com melhor rota
utilizando os pesos originais
highlight(h,L);
else
    h = plot(G, 'EdgeLabel', p);
highlight(h,P_0, 'NodeColor', 'red');
end
title('Melhor caminho calculado pelo algoritmo de Djikstra'); %Título do
gráfico
legend('Outras possibilidades/Melhor caminho') %Legenda do gráfico

end
```