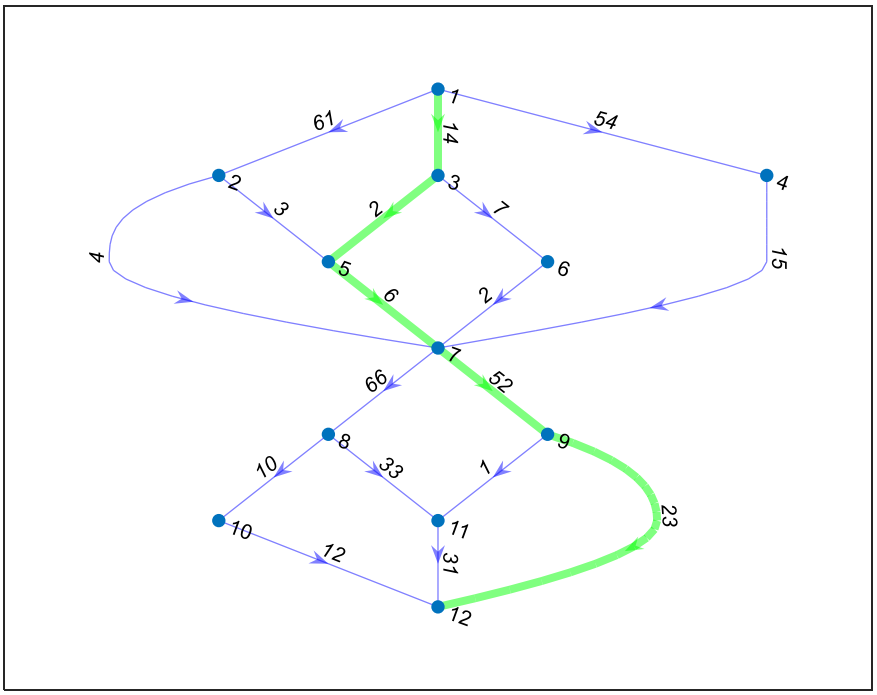
Symulacja sieci antycypacyjnych

Instancja I(Własna)

Zadana jest jedna macierz kosztów, sprzężenia antycypacyjne występują od pierwszego wierzchołka, do wierzchołków 5,6. Instancje wejściowe znajdują się w dodatku. Otrzymano jedno rozwiązanie przedstawione na rysunku poniżej.

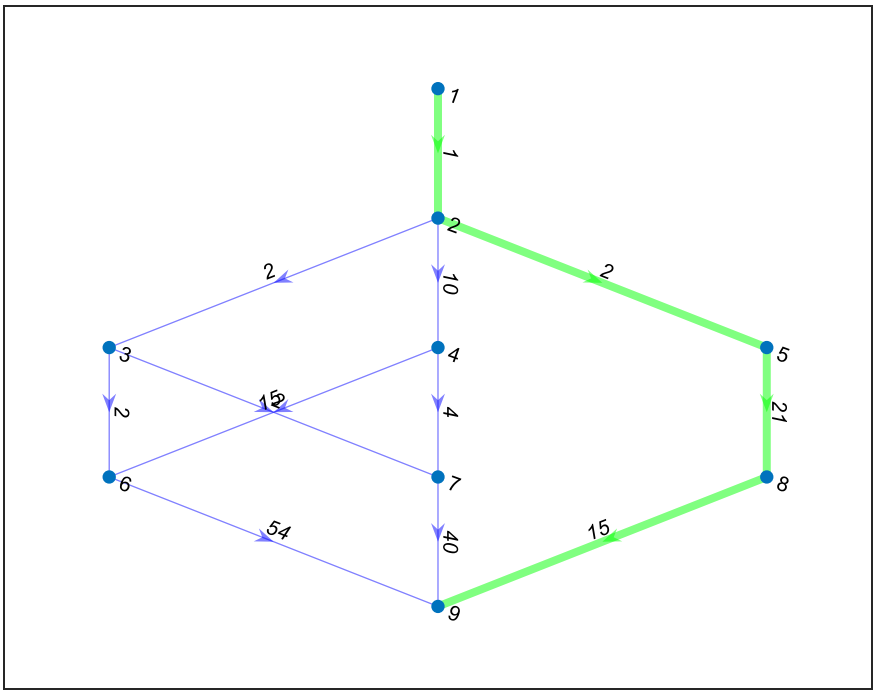


Rysunek 1 Rozwiązanie optymalne, z uwzględnieniem sprzężeń.

Koszt znalezionego rozwiązania K1 = 14+2+6+7+23=52

Instancja II(Własna)

Zadana jest jedna macierz kosztów, sprzężenia antycypacyjne występują od drugiego wierzchołka, do wierzchołków 3,4,5,6,7,8 oraz od czwartego wierzchołka do 6 i 7. Instancje wejściowe znajdują się w dodatku. Otrzymano jedno rozwiązanie przedstawione na rysunku poniżej.

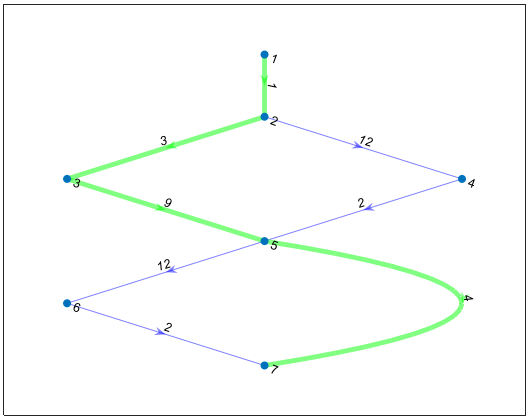


Rysunek 2 Rozwiązanie optymalne, stosując kryterium lokalne oraz sprzężenia antycypacyjne.

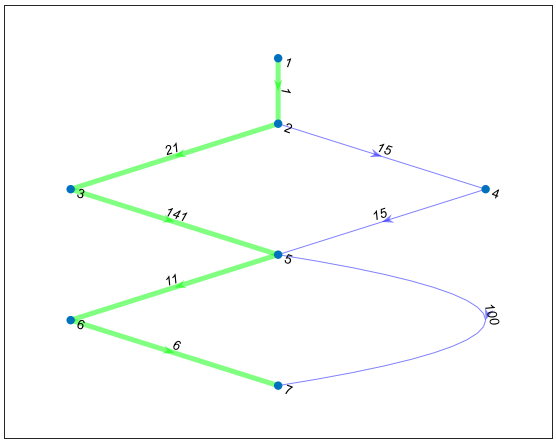
Koszt rozwiązania K = 1+2+21+15=49

Instancja III (Własna)

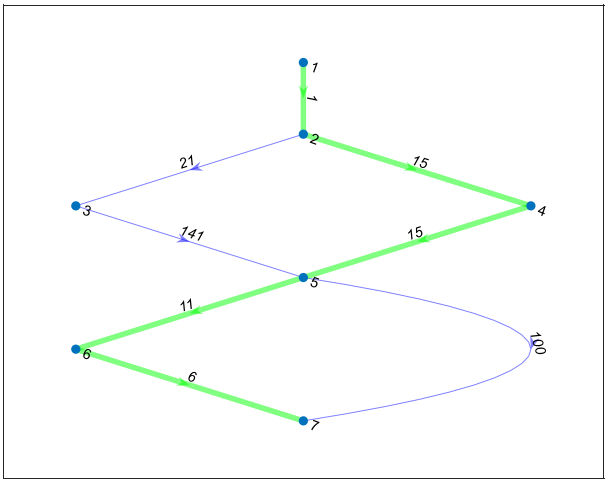
Dane wejściowe znajdują się w dodatku



Rysunek 3 Kryterium pierwsze, rozwiązanie 1.



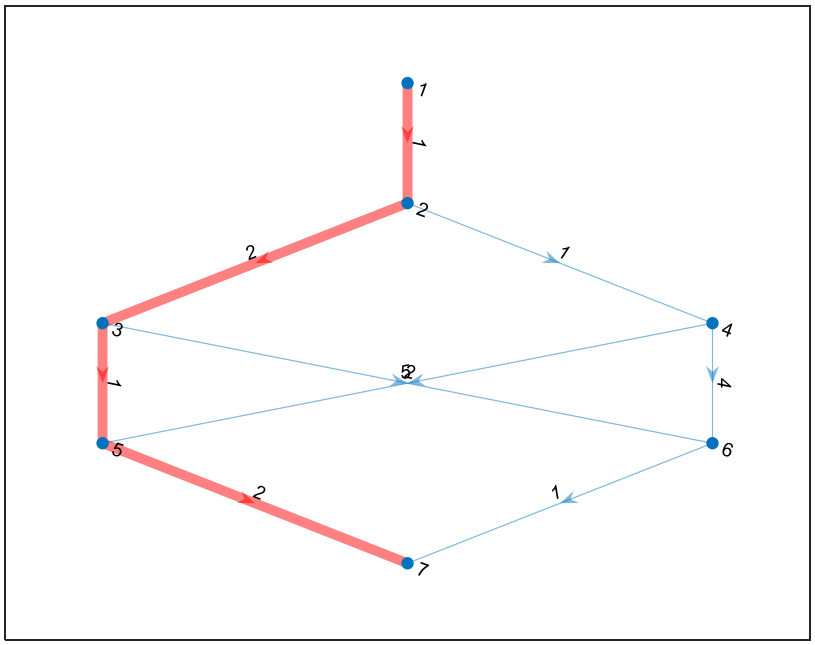
Rysunek 4 Kryterium 2, rozwiązanie 1



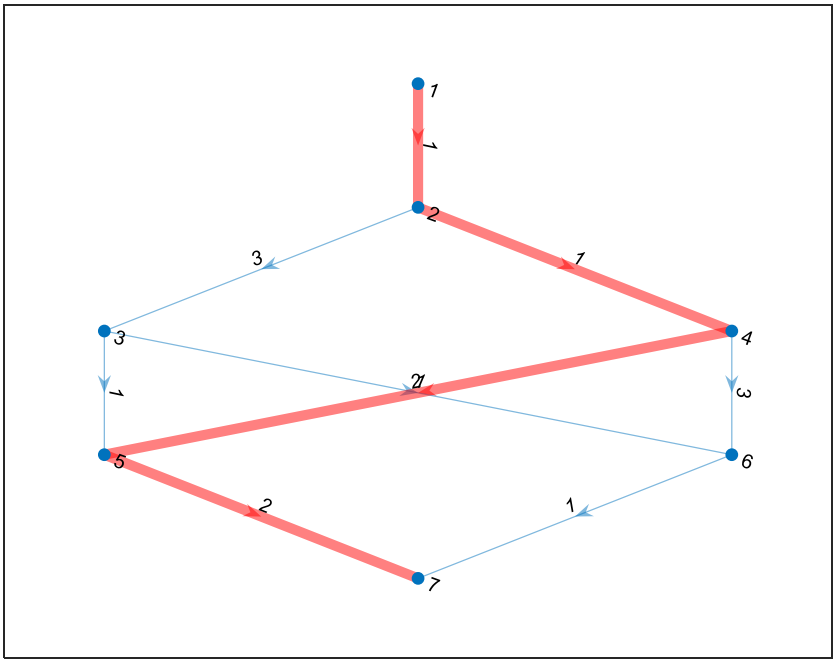
Rysunek 5 Kryterium 2, rozwiązanie 2

Instancja IV (Zadana na zajęciach)

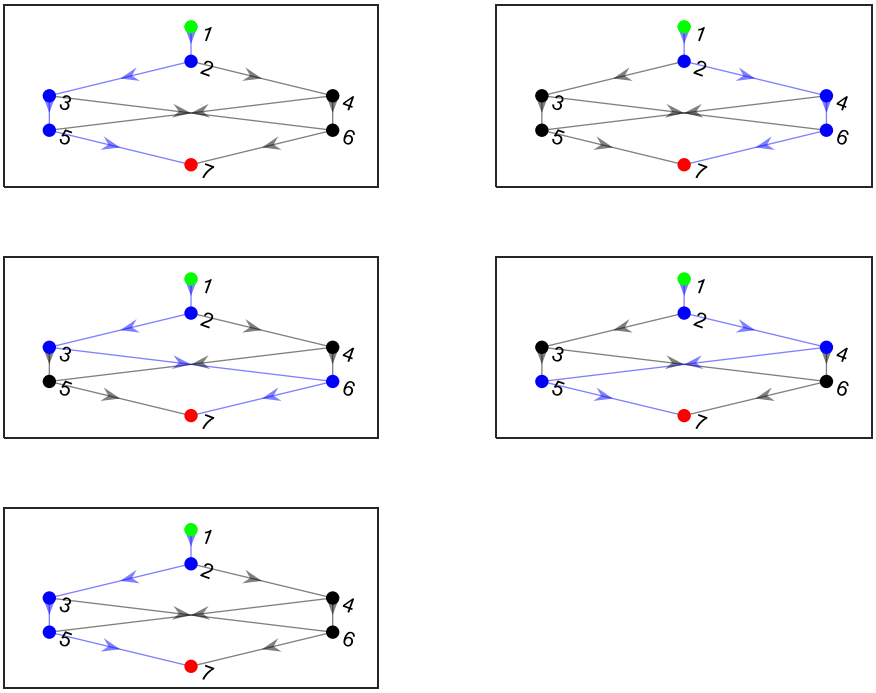
Najkrotsza sciezka dla funkcji kosztu 1 K=6



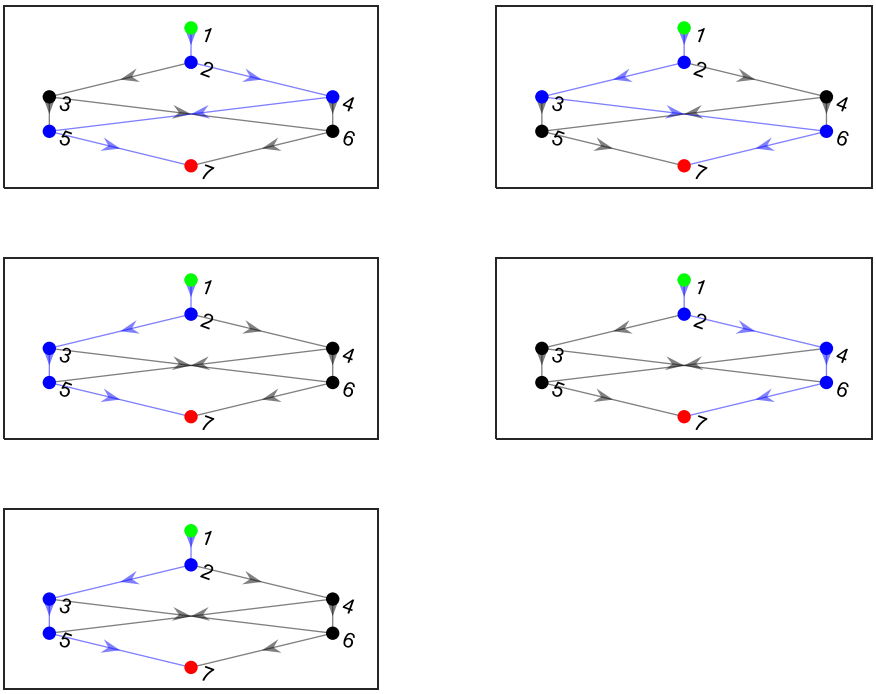
Najkrotsza sciezka dla funkcji kosztu 2 K=6



Poniżej przedstawione zostaną wszystkie możliwe ścieżki dojścia, z wierzchołka 1 do 7. Są one takie same dla obu grafów, jednak determinują one różne koszty przejścia. Dla porządku G1 to graf z krawędziami o Koszcie funkcji kosztu numer 1, natomiast G2 to graf z krawędziami o Koszcie funkcji kosztu numer 2.



Rysunek 6 Możliwe Przejścia z wierzchołka 1 do 7 dla grafu G1



Rysunek 7 Możliwe ścieżki przejścia dla gafu G2

Porównanie otrzymanych ścieżek oraz kosztów przez nie generowanych znajduje się w tabeli.

ścieżka	F(p) dla K1	F(p) dla K2
[1,2,4,5,7]	9	6
[1,2,3,6,7]	6	6
[1,2,3,5,7]	6	7
[1,2,4,6,7]	7	6

Wnioski

- Idea sieci antycypacyjnych pomaga rozwiązywać zadania znajdowania najkrótszych ścieżek w grafach, jednak nie wpływa na wartość funkcji celu tego rozwiązania. Wykorzystuje ona do tego prognozę edycji w powiązanych problemach.
- Napisanie programu, który umożliwia znalezienie wszystkich rozwiązań z uwzględnieniem sprzężeń antycypacyjnych okazało się zadaniem trudnym do rozwiązania, tak by było ono skalowalne, uniwersalne i zawsze poprawne.
- Zaimplementowana funkcja do znajdowania wszystkich połączeń między w grafie, między dwoma wierzchołkami czasami zwraca jedną ścieżkę więcej, nie udało się jednak wyjaśnić tego zjawiska.

```
close all;
clear all;

t1 =[1,2,2,3,3,4,4,5,6];
s1 =[2,3,4,5,6,5,6,7,7];
w1 =[1,2,1,1,2,5,4,2,1];
w2 =[1,3,1,1,1,2,3,2,1];

S = zeros(7);
S(3,2)=1;S(2,3)=1;S(5,2)=1;S(2,5)=1;
S(4,2)=1;S(2,4)=1;S(6,2)=1;S(2,6)=1;
G1= digraph(t1,s1,w1);
G2= digraph(t1,s1,w2);

figure(1)
plot(G1,'EdgeLabel',G1.Edges.Weight)
figure(2)
plot(G2,'EdgeLabel',G2.Edges.Weight)

%%% dodanie wiadomości wynikających z sprzezen antycypacyjnych
for i=1:size(S)
    for j=1:size(S)
        if S(i,j)==1
            t1 = [t1,i];
            s1 = [t1,j];
            w1 =[w1 shortestpath(G1,i,j)]
            w2 =[w2 shortestpath(G2,i,j)]
        end
    end
end
[P1,d1] = shortestpath(G1,1,7)
[P2,d2] = shortestpath(G2,1,7)

figure(1)
GP1=plot(G1,'EdgeLabel',G1.Edges.Weight)
highlight(GP1,P1,'EdgeColor','r','LineWidth',4)
title(strcat('Najkrotsza sciezka dla funkcji kosztu 1 K=',num2str(d1)))
figure(2)
GP2=plot(G2,'EdgeLabel',G2.Edges.Weight)
highlight(GP2,P2,'EdgeColor','r','LineWidth',4)
title(strcat('Najkrotsza sciezka dla funkcji kosztu 2 K=',num2str(d2)))


%%%%%%%%
figure(3);
pth = pathof(G1,1,7);
for ii = 1:length(pth)
    subplot(3,2,ii);
    h(ii) = plot(G1, 'edgecolor', 'k', 'nodecolor', 'k');
```

```
    highlight(h(ii), pth{ii}, 'edgecolor', 'b', 'nodecolor', 'b');
    highlight(h(ii), 1, 'nodecolor', 'g');
    highlight(h(ii), 7, 'nodecolor', 'r');
end

figure(4);
pth = path_to_end_node(G2,1,7);
for ii = 1:length(pth)

    subplot(3,2,ii);
    h(ii) = plot(G2, 'edgecolor', 'k', 'nodecolor', 'k');
    highlight(h(ii), pth{ii}, 'edgecolor', 'b', 'nodecolor', 'b');
    highlight(h(ii), 1, 'nodecolor', 'g');
    highlight(h(ii), 7, 'nodecolor', 'r');
end
```

```
last=12;
g(:,:,1)=[inf 61 14 54 inf inf inf inf inf inf inf inf;
    inf inf inf inf 3 inf 4 inf inf inf inf inf;
    inf inf inf inf 2 7 inf inf inf inf inf inf;
    inf inf inf inf inf inf 15 inf inf inf inf inf;
    inf inf inf inf inf inf 6 inf inf inf inf inf;
    inf inf inf inf inf inf 2 inf inf inf inf inf;
    inf inf inf inf inf inf inf 66 52 inf inf inf;
    inf inf inf inf inf inf inf inf inf 10 33 inf;
    inf inf inf inf inf inf inf inf inf inf 1 23;
    inf inf inf inf inf inf inf inf inf inf inf 12;
    inf inf inf inf inf inf inf inf inf inf inf 31;
    inf inf inf inf inf inf inf inf inf inf inf inf];

g(:,:,2)=[inf 35 22 11 inf inf inf inf inf inf inf inf;
    inf inf inf inf 2 inf 3 inf inf inf inf inf;
    inf inf inf inf 2 37 inf inf inf inf inf inf;
    inf inf inf inf inf inf 82 inf inf inf inf inf;
    inf inf inf inf inf inf 75 inf inf inf inf inf;
    inf inf inf inf inf inf 42 inf inf inf inf inf;
    inf inf inf inf inf inf inf 4 23 inf inf inf;
    inf inf inf inf inf inf inf inf inf 2 1 inf;
    inf inf inf inf inf inf inf inf inf inf 1 4;
    inf inf inf inf inf inf inf inf inf inf inf 2;
    inf inf inf inf inf inf inf inf inf inf inf 3;
    inf inf inf inf inf inf inf inf inf inf inf inf];

s=[0 0 0 0 1 1 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0];
```

```
last=9;
g(:,:,1)=  [inf 1 inf inf inf inf inf inf inf;
                  inf inf 2 10 2 inf inf inf inf;
                  inf inf inf inf inf 2 2 inf inf;
                  inf inf inf inf inf 15 4 inf inf;
                  inf inf inf inf inf inf inf 21 inf;
                  inf inf inf inf inf inf inf inf 54;
                  inf inf inf inf inf inf inf inf 40;
                  inf inf inf inf inf inf inf inf 15;
                  inf inf inf inf inf inf inf inf inf];


s=[0 0 0 0 0 0 0 0 0;
           0 0 1 1 1 1 1 1 0;
           0 0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 1 1 0;
           0 0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0 0;
           0 0 0 0 0 0 0 0 0];
```

```
last=7;
g(:,:,1)=  [inf 1 inf inf inf inf inf;
                  inf inf 3 12 inf inf inf;
                  inf inf inf inf 9 inf inf;
                  inf inf inf inf 2 inf inf;
                  inf inf inf inf inf 12 4;
                  inf inf inf inf inf inf 2;
                  inf inf inf inf inf inf inf];

g(:,:,1)=  [inf 1 inf inf inf inf inf;
                  inf inf 21 15 inf inf inf;
                  inf inf inf inf 141 inf inf;
                  inf inf inf inf 15 inf inf;
                  inf inf inf inf inf 11 100;
                  inf inf inf inf inf inf 6;
                  inf inf inf inf inf inf inf];

s=[0 0 0 0 0 0 0;
           0 0 1 1 1 1 0;
           0 0 0 0 0 0 0;
           0 0 0 0 0 0 0;
           0 0 0 0 0 0 0;
           0 0 0 0 0 0 0;
           0 0 0 0 0 0 0];
```

```
function path=path_to_end_node(graph,startn,endn)
stop_loop=0;
n=0;
while stop_loop~=1
    n=n+1;
    tmp=shortestpath(graph,startn,endn);
    ID=findedge(graph,tmp(1:end-1),tmp(2:end));
    if n~=1
        if length(tmp)==length(path{n-1,1})
            if tmp==path{n-1,1}
                stop_loop=1;
            else
                path{n,1}=tmp;
                graph.Edges.Weight(ID)=100;
            end
        else
            path{n,1}=tmp;
            graph.Edges.Weight(ID)=100;
        end
    else
        path{n,1}=tmp;
        graph.Edges.Weight(ID)=100;
    end
    clear tmp ID;
end
```