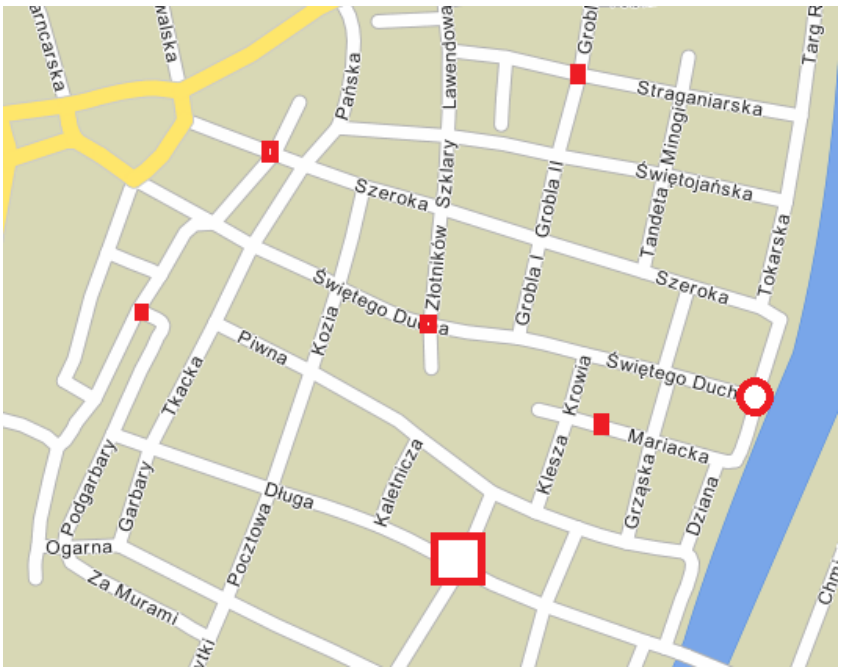
Optymalizacja. Problem marszrutyzacji (Vehicle Routing Problem)

https://en.wikipedia.org/wiki/Vehicle_routing_problem

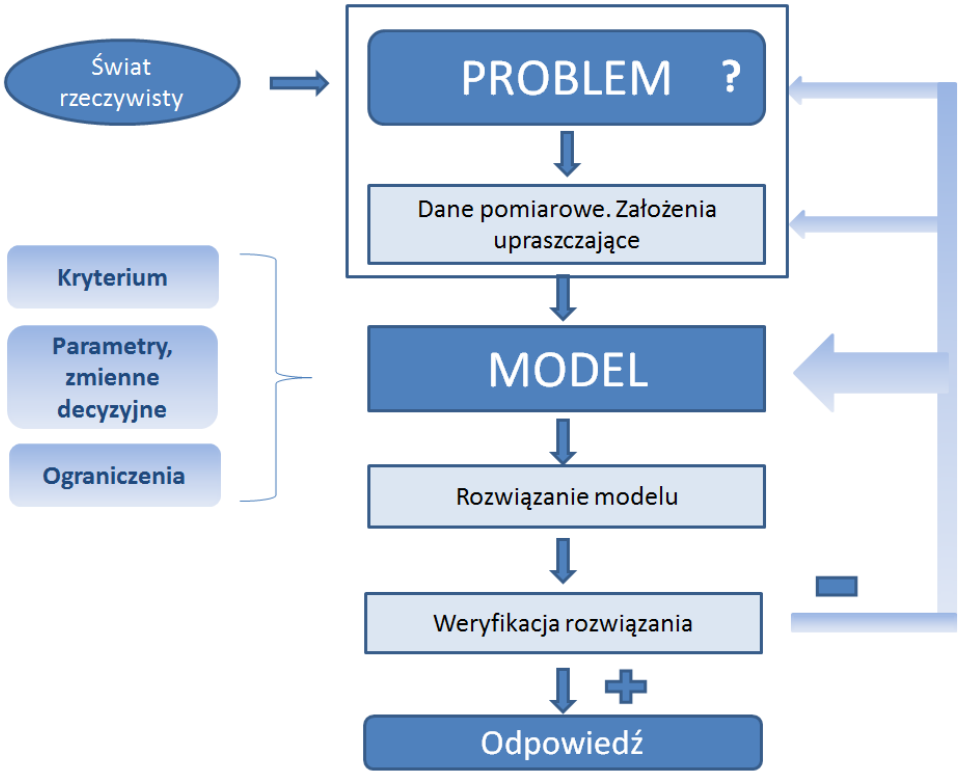
Zadanie 1. Zaplanuj **optymalną** trasę kuriera (**możliwie najkrótszą**), który wyjeżdża z siedziby firmy (duży kwadrat), odbiera n=5 przesyłek (oznaczonych małymi kwadratami), dostarcza przesyłki do odbiorcy (okrąg) a następnie wraca do siedziby firmy (duży kwadrat).



Zadanie 2. Problem marszrutyzacji z ograniczeniami kolejnościowymi.

Kurier wyjeżdża z bazy  i do niej wraca po zrealizowaniu zleceń. Realizacja zlecenia polega na odbiorze przesyłki od nadawcy (pełne figury ) i dowiezieniu do odbiorców (puste figury ).

Powtórzenie



Laboratorium 1 Zagadnienia programowania liniowego

Środki produkcji	Jednostkowe nakłady		Limit środków
	W_1	W_2	
I	16	24	96 000
II	16	10	80 000
Limit produkcji	3 000 szt	4 000 szt	
Zysk jednostkowy	30 zł	40 zł	

Dodatkowe ograniczenie: „Komórka analizy rynku ustaliła optymalne proporcje produkcji W_1:W_2, które kształtują się odpowiednio jak 3:2”

Zmienne decyzyjne x_1, x_2 oznaczające liczbę sztuk W_1, W_2

Cel / Kryterium $FC(x_1, x_2) = 30x_1 + 40x_2 \rightarrow$ max

Ograniczenia:

$16x_1 + 24x_2 \leq 96\,000$

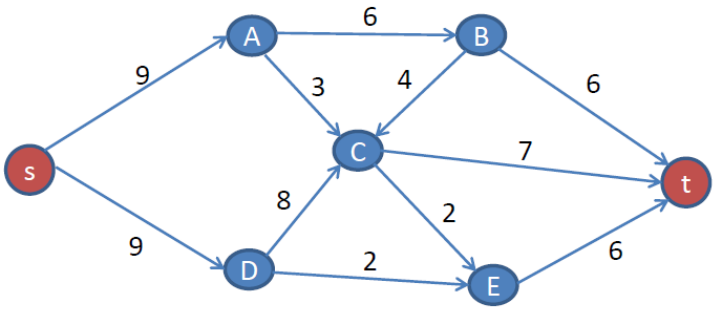
$16x_1 + 10x_2 \leq 80\,000$

$0 \leq x_1 \leq 3\,000 \quad 0 \leq x_2 \leq 4\,000$

$x_1 = \frac{2}{3}x_2$

Laboratorium 2-3 Problem maksymalnego przepływu

Problem maksymalnego przepływu w sieci



FC: $x_{SA} + x_{SD}$ = Przepływ $\rightarrow max$

$$x_{SA} = x_{AB} + x_{AC}$$
$$x_{AB} = x_{BC} + x_{Bt}$$
$$x_{AC} + x_{DC} + x_{BC} = x_{CE} + x_{Ct}$$
$$x_{SD} = x_{DC} + x_{DE}$$
$$x_{CE} + x_{DE} = x_{Et}$$
$$\boldsymbol{x_{SA} + x_{SD} = x_{Bt} + x_{Ct} + x_{Et}}$$
$$x_{SA} \leq 9, x_{AB} \leq 6, x_{AC} \leq 3$$
$$x_{SD} \leq 9, x_{DC} \leq 8, x_{DE} \leq 2$$
$$x_{BC} \leq 4, x_{Bt} \leq 6, x_{CE} \leq 2$$
$$x_{Ct} \leq 7, x_{Et} \leq 6, x_{ij} \geq 0$$

Sformułowanie zagadnienia programowania liniowego

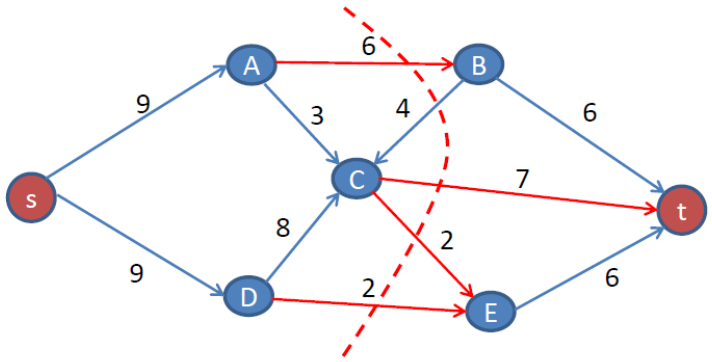
$$x = \begin{bmatrix} x_{SA} \\ x_{AB} \\ x_{AC} \\ x_{SD} \\ x_{DC} \\ x_{DE} \\ x_{BC} \\ x_{Bt} \\ x_{CE} \\ x_{Ct} \\ x_{Et} \end{bmatrix} \quad xU = \begin{bmatrix} 9 \\ 6 \\ 3 \\ 9 \\ 8 \\ 2 \\ 4 \\ 6 \\ 2 \\ 7 \\ 6 \end{bmatrix} \quad x_{opt} = \begin{bmatrix} 9 \\ 6 \\ 3 \\ 8 \\ 6 \\ 2 \\ 0 \\ 6 \\ 2 \\ 7 \\ 4 \end{bmatrix}$$

Rozwiązanie optymalne

$FC = 17$

Algorytmy grafowe

Problem minimalnego przekroju sieci



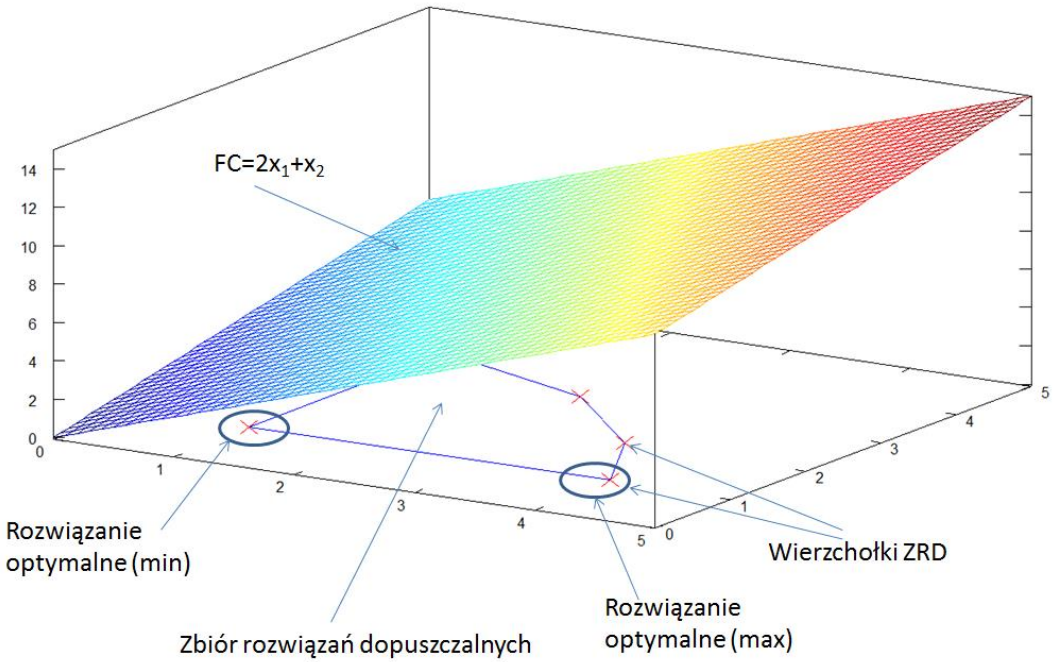
Sformułowanie dualnego zagadnienia programowania liniowego

$$y = \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ t \end{bmatrix} \quad y_{opt} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

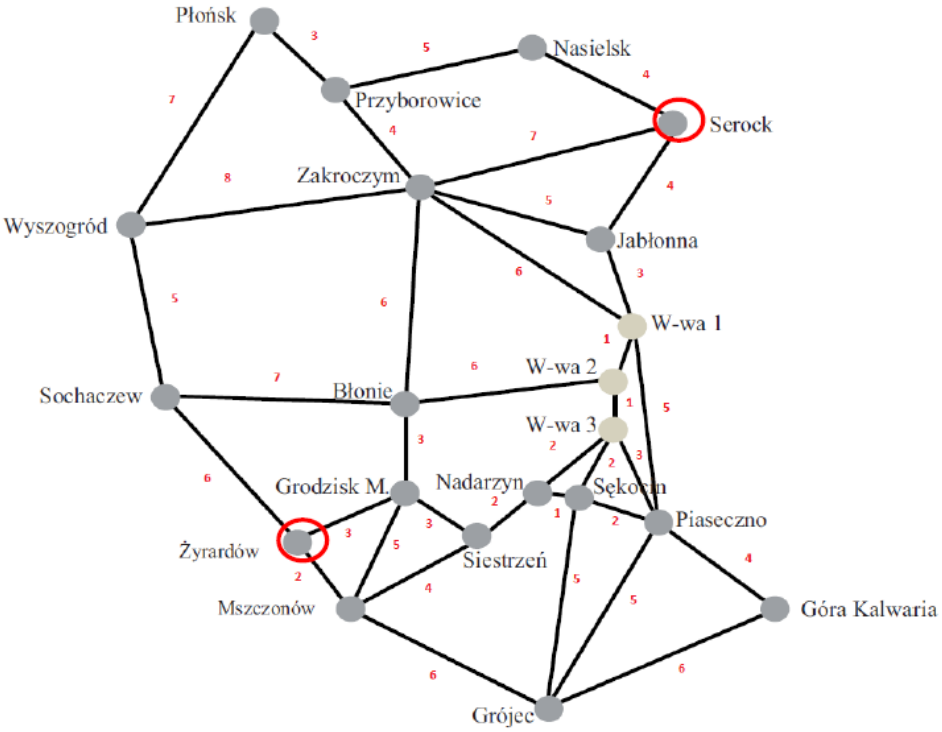
Podział wierzchołków na 2 podzbiory

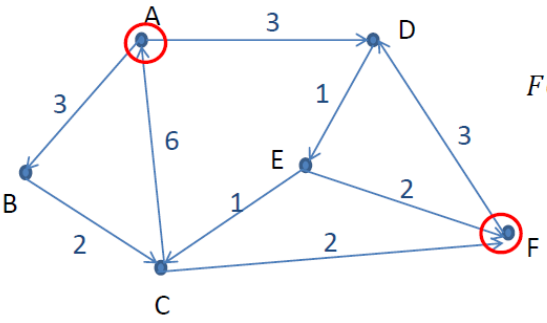
Rozwiązanie optymalne $FC = 17$

Zagadnienie programowania liniowego


FC=2x1+x2
14
12
10
8
6
4
2
0
0
1
2
3
4
5
0
1
2
3
4
5
Rozwiązanie optymalne (min)
Zbiór rozwiązań dopuszczalnych
Rozwiązanie optymalne (max)
Wierzchołki ZRD

Problem najkrótszej ścieżki





Model pierwotny

$$FC = 3x_{AD} + 3x_{AB} + 6x_{CA} + 2x_{BC} + x_{EC} + x_{DE} + 2x_{EF} + 3x_{FD} + 2x_{CF} \rightarrow min$$

$$\mathbf{Ax} = \mathbf{b}$$

$$x_i \geq 0$$

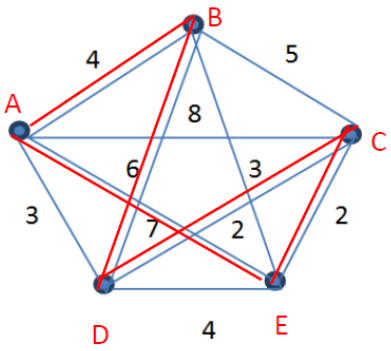
$$A = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \end{bmatrix} \quad \mathrm{x} = \begin{bmatrix} x_{AD} \\ x_{AB} \\ x_{CA} \\ x_{BC} \\ x_{EC} \\ x_{DE} \\ x_{EF} \\ x_{FD} \\ x_{CF} \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

Krawędzie

$$x_{opt} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Krawędzie tworzące najkrótszą ścieżkę

Problem komiwojażera



	A	B	C	D	E
A	-	4	8	3	7
B		-	5	6	3
C			-	2	2
D				-	4
E					-

$$FC = 4x_{AB} + 8x_{AC} + 3x_{AD} + 7x_{AE} + 4x_{BA} + 5x_{BC} + 6x_{BD} + 3x_{BE} + 8x_{CA} + 5x_{CB} + 2x_{CD} + 2x_{CE} + 3x_{DA} + 6x_{DB} + 2x_{DC} + 4x_{DE} + 7x_{EA} + 3x_{EB} + 2x_{ED} + 4x_{ED} \rightarrow min$$

Dokładnie jedna jedynka w wierszu

$$x_{AB} + x_{AC} + x_{AD} + x_{AE} = 1$$
$$x_{BA} + x_{BC} + x_{BD} + x_{BE} = 1$$
$$x_{CA} + x_{CB} + x_{CD} + x_{CE} = 1$$
$$x_{DA} + x_{DB} + x_{DC} + x_{DE} = 1$$
$$x_{EA} + x_{EB} + x_{EC} + x_{ED} = 1$$

Przykładowe rozwiązanie

	A	B	C	D	E
A	-	1			
B		-		1	
C			-		1
D			1	-	
E	1				-

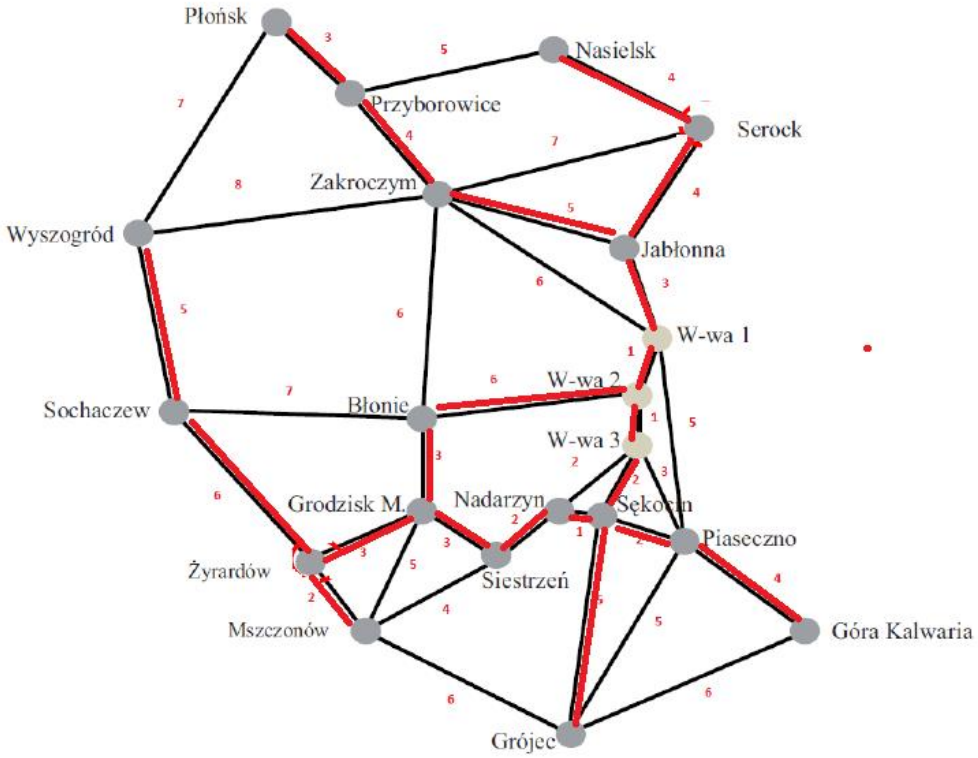
Krawędź co najwyżej raz

$$x_{AB} + x_{BA} \leq 1$$
$$x_{AC} + x_{CA} \leq 1$$
$$x_{AD} + x_{DA} \leq 1$$
$$x_{AE} + x_{EA} \leq 1$$
$$x_{BC} + x_{CB} \leq 1$$
$$x_{BD} + x_{DB} \leq 1$$
$$x_{BE} + x_{EB} \leq 1$$
$$x_{CD} + x_{DC} \leq 1$$
$$x_{CE} + x_{EC} \leq 1$$
$$x_{DE} + x_{ED} \leq 1$$

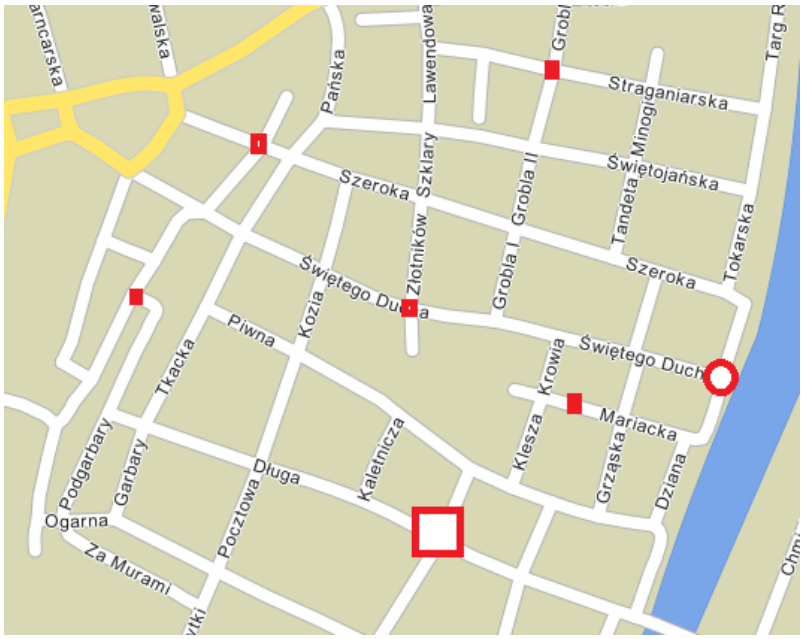
Dokładnie jedna jedynka w kolumnie

$$x_{BA} + x_{CA} + x_{DA} + x_{EA} = 1$$
$$x_{AB} + x_{CB} + x_{DB} + x_{EB} = 1$$
$$x_{AC} + x_{BC} + x_{DC} + x_{EC} = 1$$
$$x_{AD} + x_{BD} + x_{CD} + x_{ED} = 1$$
$$x_{AE} + x_{BE} + x_{CE} + x_{DE} = 1$$

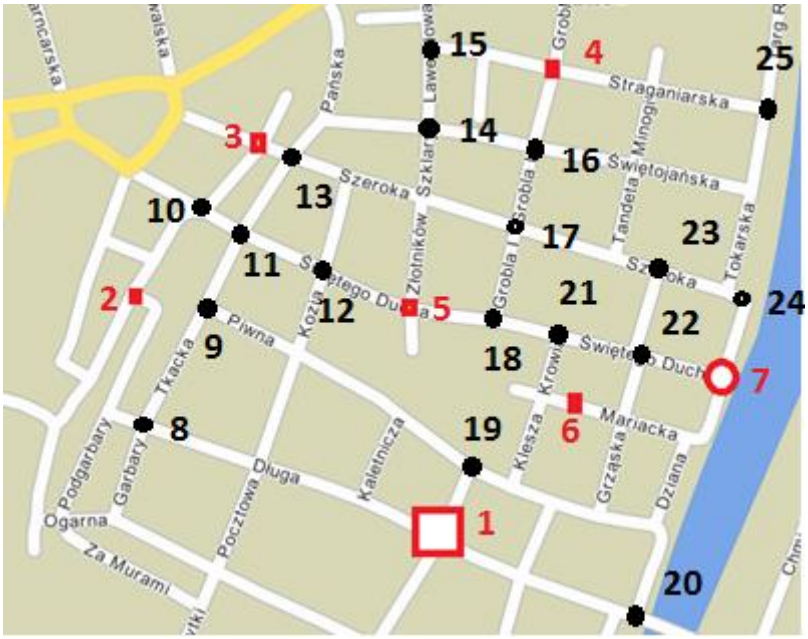
Problem minimalnego drzewa rozpinającego



Zadanie: Zaplanuj **optymalną** trasę kuriera (**możliwie najkrótszą**), który wyjeżdża z siedziby firmy (duży kwadrat), odbiera n=5 przesyłek (oznaczonych małymi kwadratami), dostarcza przesyłki do odbiorcy (okrąg) a następnie wraca do siedziby firmy (duży kwadrat).



1. Tworzymy graf. Propozycja pomocniczego grafu o 25 wierzchołkach.



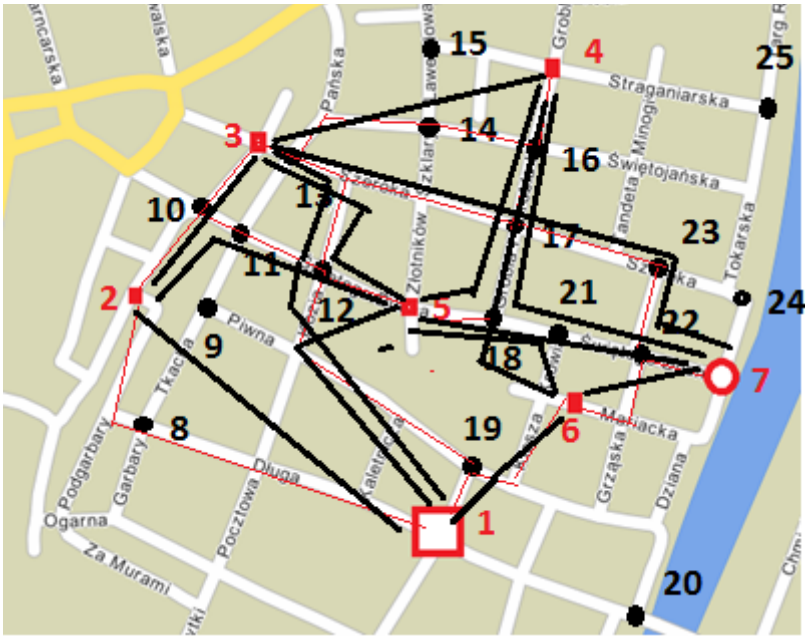
ta=[1 1 1 2 2 3 3 4 4 4 5 5 5 5 5 6 6 6 6 6 7 7 7 8 8 9 9 10 11 11 12 12 12 12 13 13 14 14 14 16 16 16 17 17 18 19 21 22 23 23 24];
he=[8 19 20 8 10 10 13 15 16 25 12 13 14 17 18 19 21 22 7 20 20 22 24 9 12 11 19 11 12 13 13 14 17 19 14 17 15 16 17 17 24 25 18 23 21 20 22 23 24 25 25];
dl=[40 10 27 22 14 11 4 16 10 28 11 33 24 27 11 20 11 13 25 40 35 8 8 18 42 10 40 5 11 10 21 30 32 36 18 30 10 13 20 10 42 38 12 20 8 38 10 10 11 33 27];

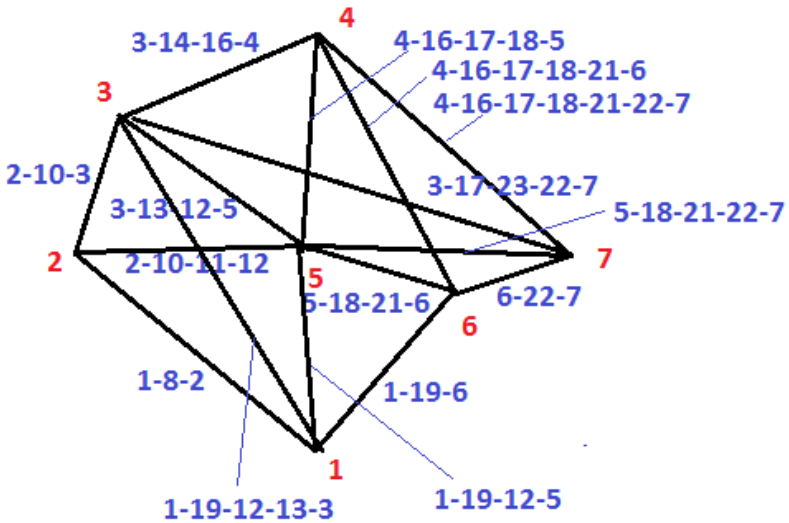
2. Ustalamy przebieg/długość najkrótszych ścieżek pomiędzy wierzchołkami 1-7. Tworzymy docelowy graf o 7 wierzchołkach, którego krawędzie będą odpowiadały znalezionym najkrótszym ścieżkom. Nie tworzymy krawędzi

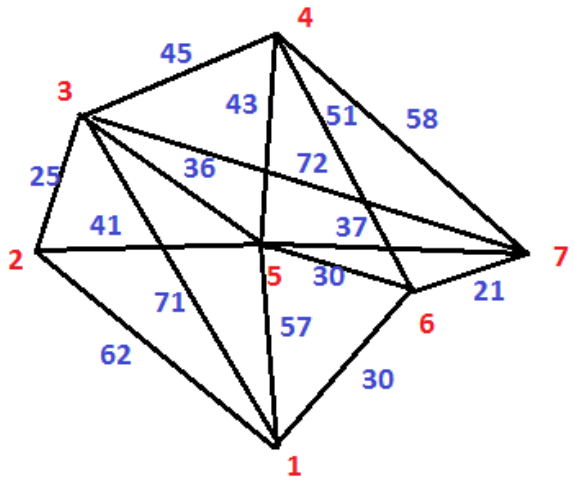
pomiędzy wierzchołkami, jeżeli najkrótsza ścieżka wiedzie przez inny niż początkowy lub końcowy wierzchołek 1-7. Przykład: znaleziono najkrótszą ścieżkę 1-7: 1-19-6-22-7, która nie zostanie uwzględniona jako krawędź 1-7 gdyż zawiera wierzchołek 6. Zostaną uwzględnione odrębne krawędzie 1-6 i 6-7.

$$\begin{bmatrix} - & 62 & 71 & - & 57 & 30 & - \\ 62 & - & 25 & - & 41 & - & - \\ 71 & 25 & - & 45 & 36 & - & 72 \\ - & - & 45 & - & 43 & 51 & 58 \\ 57 & 41 & 36 & 43 & - & 30 & 37 \\ 30 & - & - & 51 & 30 & - & 21 \\ - & - & 72 & 58 & 37 & 21 & - \end{bmatrix}$$

3. Powstaje docelowy graf o 7 wierzchołkach opisany powyższą macierzą sąsiedztwa. Jego krawędzie odpowiadają najkrótszym ścieżkom.

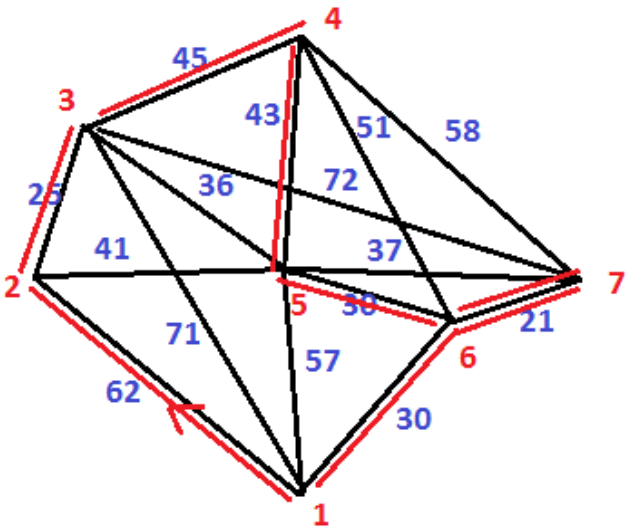






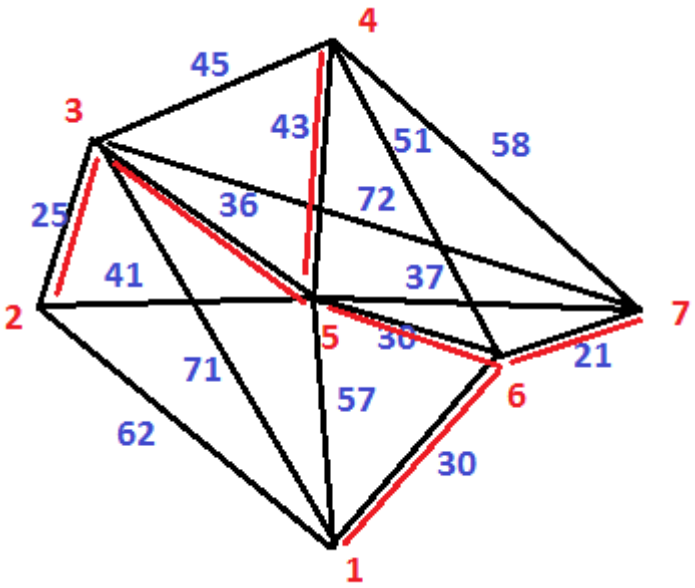
Długości najkrótszych ścieżek.

4. Wyznaczamy cykl Hamiltona, który może pomóc w wyznaczeniu optymalnej marszruty

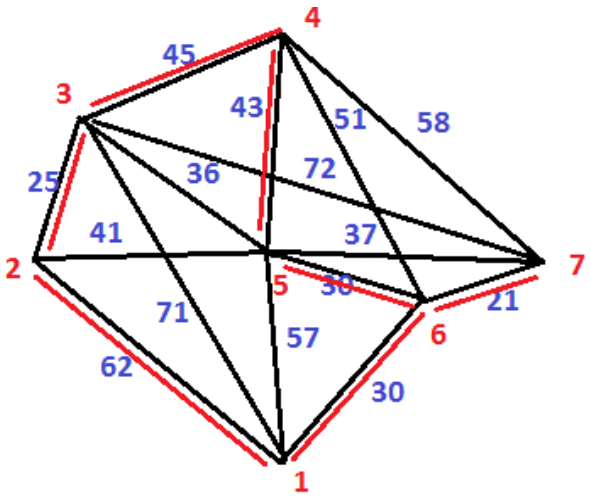



```
ta2=[ 1  1  1  1  2  2  3  3  3  4  4  4  5  5  6];
he2=[ 2  3  5  6  3  5  4  5  7  5  6  7  6  7  7];
dl2=[62 71 57 30 25 41 45 36 72 43 51 58 30 37 21];
g2=make_graph('graf2',1,7,[ta2 he2],[he2 ta2]);
g2=add_edge_data(g2,'length',[dl2 dl2]);
circ=salesman(g2);
wierz=path_2_nodes(circ,g2);

wierz  =    1.    6.    7.    5.    4.    3.    2.    1.

//  wyznaczony cykl został przekształcony 1-2-3-4-5 i dalej 6-
7-1
```

5. Można wyznaczyć minimalne drzewo rozpinające i spróbować dokonać przekształcenia, by utworzyć cykl.

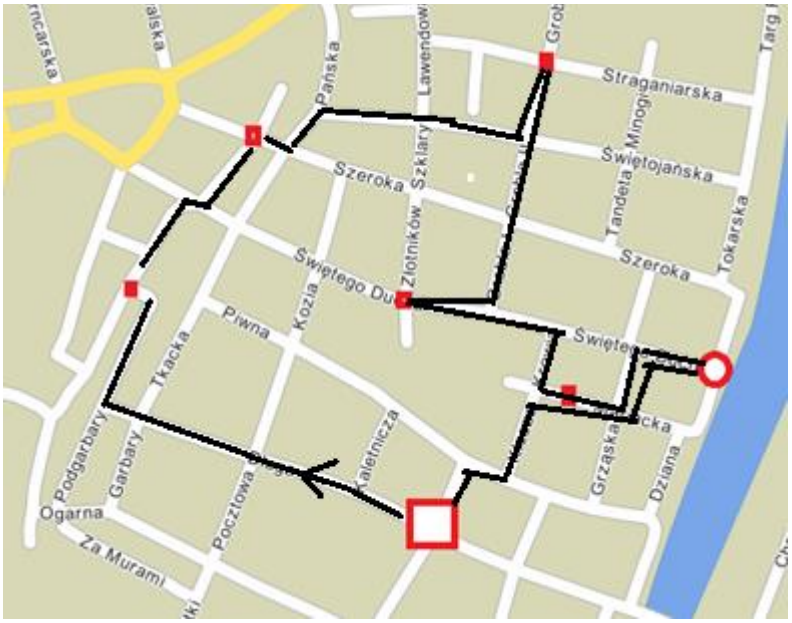



```
g3=make_graph('graf3',0,7,ta2, he2);
g3=add_edge_data(g3,'weight',dl2);
tr=min_weight_tree(g3);
tr  =    5.    8.    10.    13.    4.    15.
// indeksy krawędzi
```

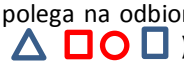


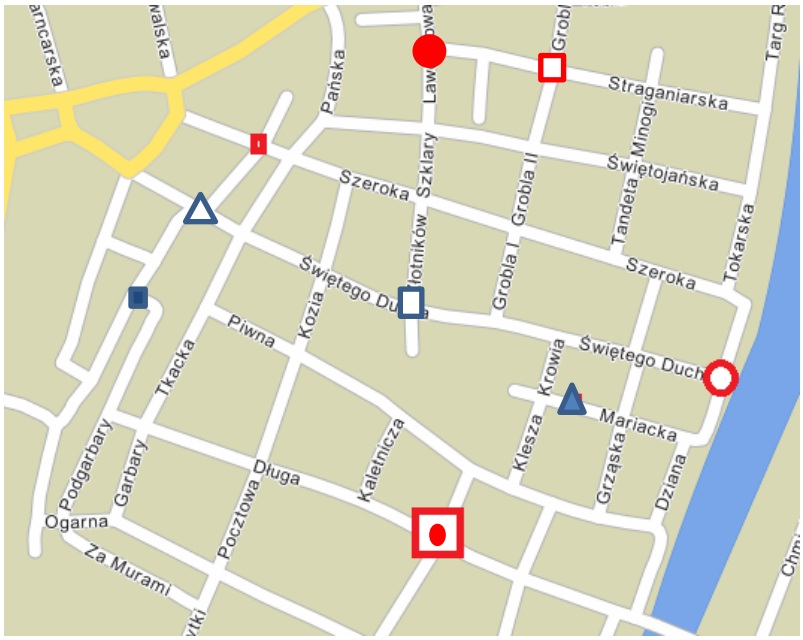
Dodano krawędź 1-2, krawędź 3-5 zastąpiono krawędzią 3-4.

6. Na podstawie utworzonego cyklu i wcześniej wyznaczonych najkrótszych ścieżek ustalamy marszrutę



Problem marszrutyzacji

Kurier wyjeżdża z bazy  i do niej wraca po zrealizowaniu zleceń. Realizacja zlecenia polega na odbiorze przesyłki od nadawcy (pełne figury ) i dowiezieniu do odbiorców (puste figury ).

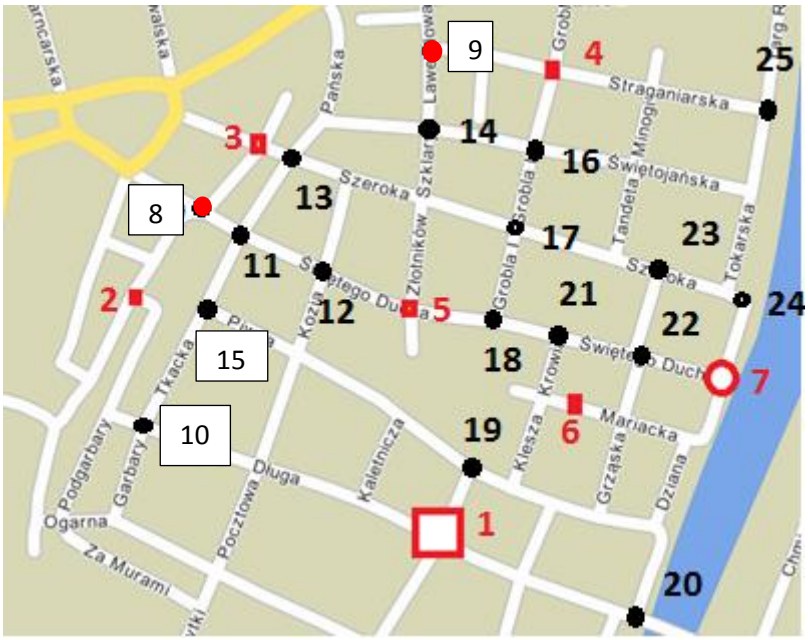


Problem marszrutyzacji z ograniczeniami kolejnościowymi.

Inne możliwe ograniczenia:
- okna czasowe
- ograniczenia pojemnościowe

Krok 1

Tworzymy graf. Propozycja pomocniczego grafu o 25 wierzchołkach.




```
ta=[ 1  1  1  2  2  3  3  4  4  4  5  5  5  5  5  6  6  6  6  6  7  7  7  10 10  15
15 8 11 11 12 12 12 12 13 13 14 14 14 16 16 16 17 17 18 19 21 22 23 23 24];
he=[ 10 19 20  10 8 8 13 9 16 25 12 13 14 17 18 19 21 22  7 20 20 22 24  15 12 11
19 11 12 13 13 14 17 19 14 17 9 16 17 17 24 25 18 23 21 20 22 23 24 25 25];
dl=[40 10 27 22 14 11  4 16 10 28 11 33 24 27 11 20 11 13 25 40 35  8  8 18 42 10
40  5 11 10 21 30 32 36 18 30 10 13 20 10 42 38 12 20  8 38 10 10 11 33 27];
```

```
g1=make_graph('graf1',1,25,[ta he],[he ta]);

g1=add_edge_data(g1,'length',[dl dl]);
```

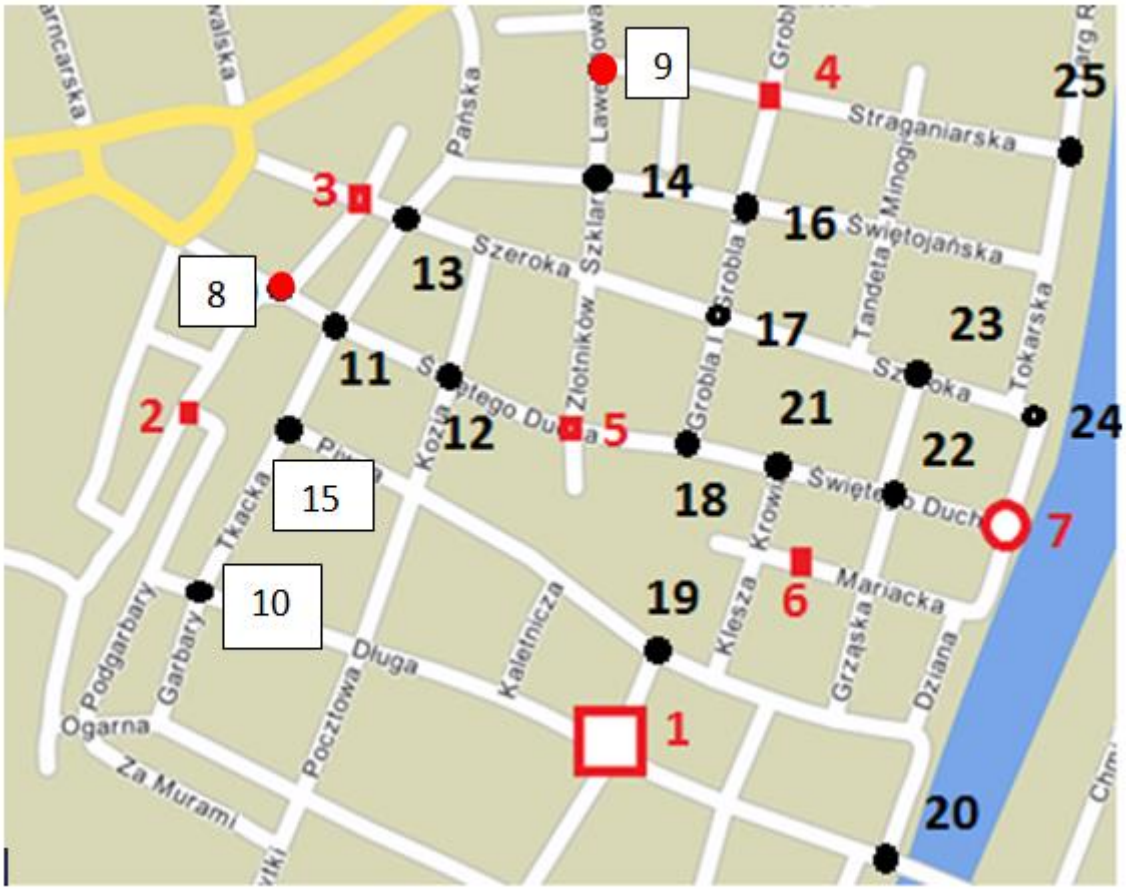
Krok 2

Szukamy marszruty realizującej zlecenia

```
Zlecenia=[2 5; 6 8; 3 4; 9 7];
```

Podejście 1

Realizacja zleceń wg kolejności



Marszruta:

1–10–**2**–8–11–12–**5**–18–21–**6**–21–18–5–12–11–**8**–**3**–13–14–16–**4**–**9**–14–17–23–22–**7**–22–6–19–**1**

Długość marszruty **381**

```
zlecenia=[2 5; 6 8; 3 4; 9 7];

/// podejscie 1
baza=1;
kurier=baza;
dlug1=0;
trasa1=[baza];
for k=1:4
  [p,dl]=shortest_path(kurier,zlecenia(k,1),g1,'length'); // odbior
  dlug1=dlug1 + dl
  wierz=path_2_nodes(p,g1);
  trasa1=[trasa1 wierz(1,2:$)];

  kurier=zlecenia(k,1);
  [p,dl]=shortest_path(kurier,zlecenia(k,2),g1,'length'); // dostarczenie
  dlug1=dlug1 + dl
  wierz=path_2_nodes(p,g1);
  trasa1=[trasa1 wierz(1,2:$)];
  kurier=zlecenia(k,2);
end
[p,dl]=shortest_path(kurier,baza,g1,'length'); // powrót
dlug1=dlug1 + dl
wierz=path_2_nodes(p,g1);
trasa1=[trasa1 wierz(1,2:$)]
```

Podejście 2

Realizacja zleceń wg kolejności, ale z obsługą pozostałych zleceń ‘przy okazji’

Marszruta:

1–10–**2**–8–11–12–**5**–18–21–**6**–21–18–5–12–11–**8**–**3**–13–14–**9**–**4**–16–17–23–22–**7**–22–6–19–**1**

Długość marszruty < **381**

Podejście 3

Podejście zachłanne. Idź do aktualnie najbliższego punktu odbioru lub dostarczenia.

Ustalamy przebieg/długość najkrótszych ścieżek pomiędzy wierzchołkami 1-9. Tworzymy docelowy graf o 9 wierzchołkach, którego krawędzie będą odpowiadały znalezionym najkrótszym ścieżkom.

```
m2=zeros(9,9);
for i=1:9
  for j=1:9
      if i<>j then
         [p,dl]=shortest_path(i,j,g1,'length');
          m2(i,j)=dl;
      end
  end
end
```

```
m2  =

    0.    62.    71.    81.    57.    30.    51.    62.    86.
    62.   0.    25.    70.    41.    71.    78.    14.    57.
    71.   25.   0.    45.    36.    65.    72.    11.    32.
    81.   70.   45.   0.    43.    51.    58.    56.    16.
    57.   41.   36.   43.   0.    30.    37.    27.    34.
    30.   71.   65.   51.   30.   0.    21.    57.    61.
    51.   78.   72.   58.   37.   21.   0.    64.    68.
    62.   14.   11.   56.   27.   57.   64.   0.    43.
    86.   57.   32.   16.   34.   61.   68.   43.   0.
```

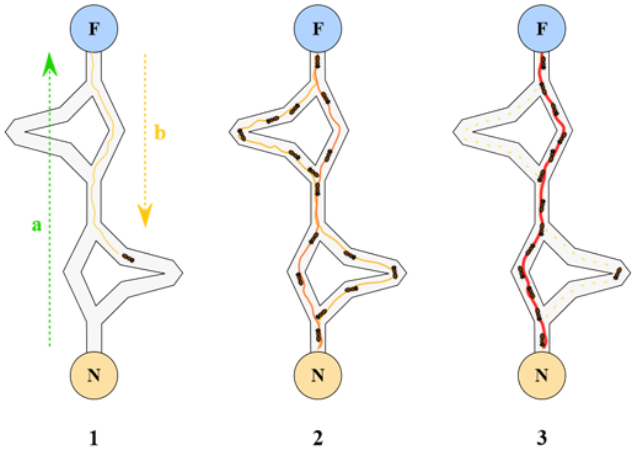
1-6-8-3-2-5-9-4-7-1 = **323**

```
ta2=[1 1  1 1  1 1  3 3  3 3  3 4 4  4 5  5 5  5 6  6 7  8];
he2=[2 3  5 6  8 9  4 5  6 7  8 5 6  7 6  7 8  9 7  9 9  9];
dl2=[62 71 57 30 62 86 45 36 65 72 32 43 51 58 30 37 27 34 21
61 68 43];
g2=make graph('graf2',1,9,[ta2 he2],[he2 ta2]);
g2=add edge data(g2,'length',[dl2 dl2]);
```

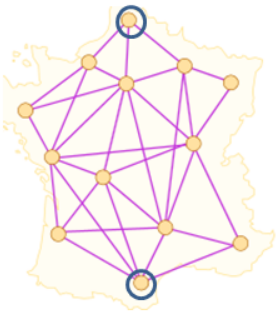
Podejście 4

Algorytm mrówkowy Ant Colony Optimization

1991 Marco Dorigo, Alberto Colorni, Vittorio Maniezzo



- Znakowanie ścieżki feromonem
- Losowy wybór ścieżki przez pojedynczą mrówkę z prawdopodobieństwem proporcjonalnym do stężenia feromonu
- Parowanie feromonu



$$\eta_{ij} = \begin{bmatrix} \square & \square & \dots & \square \\ \square & \square & & \square \\ \dots & & & \\ \square & \square & & \square \end{bmatrix}$$

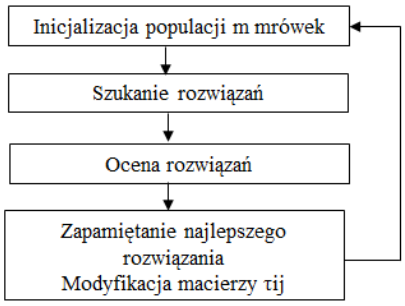
Heurystycznie wyznaczona „atrakcyjność" krawędzi

$$\tau_{ij} = \begin{bmatrix} \square & \square & \dots & \square \\ \square & \square & & \square \\ \dots & & & \\ \square & \square & & \square \end{bmatrix}$$

Ilość feromonu na krawędzi

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{j \in A_k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} & jeżeli\ j \in A_k^i \\ 0 & w\ pozostałych\ przypadkach \end{cases}$$

α, β – parametry algorytmu określające względny wpływ ilości feromonu i wartości heurystycznej na wybór węzła sąsiedniego dokonywany przez mrówkę



$$\tau_{ij} = \begin{bmatrix} \square & \square & \dots & \square \\ \square & \square & & \square \\ \dots & & & \\ \square & \square & & \square \end{bmatrix}$$

$$\tau_{ij}(l+1) = \rho\tau_{ij}(l) + \Delta\tau_{ij}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad \Delta\tau_{ij}^{k} = \frac{Q}{L_k}$$

ρ – współczynnik parowania feromonu $0 < \rho \leq 1$;
L_k–wartość funkcji oceny k-tej mrówki;
Q – parametr systemowy.

Zastosowania ACO:

- Problem komiwojażera (*Traveling Salesman Problem*)
- Problem przydziału (*Assignment Problem*)
- Problem marszrutyzacji (*Vehicle Routing Problem*)
- Szeregowanie zadań (*Job Scheduling*)
- Inne ...

W problemie najkrótszej ścieżki w grafie szukanie rozwiązania polega na trawersowaniu przez każdą mrówkę grafu, którego krawędzie opisane są, oprócz długości (kosztu, czasu), dodatkową parą liczb $\{\tau_{ij},\eta_{ij}\}$, gdzie liczba τ_{ij} oznacza ilość feromonu na drodze $i \to j$, podczas gdy η_{ij} odpowiada wartości heurystycznej określającej atrakcyjność wyboru danej krawędzi.

Każda mrówka k z m–licznej populacji wykonuje ruch pomiędzy sąsiednimi wierzchołkami $i \to j$ z prawdopodobieństwem wyrażonym wzorem:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{j \in A_k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} & jeżeli\ j \in A_k^i \\ 0 & w\ pozostałych\ przypadkach \end{cases}$$

gdzie:

τ_{ij} – ilość feromonu na krawędzi $i \to j$;

η_{ij} – wartość heurystyczna określająca atrakcyjność wyboru krawędzi $i \to j$;

α, β – parametry algorytmu określające względny wpływ ilości feromonu i wartości heurystycznej na wybór węzła sąsiedniego dokonywany przez mrówkę;

A_k^i–lista mrówki k węzłów sąsiadujących z węzłem i.

Wartość prawdopodobieństwa przejścia $i \rightarrow j$ rośnie wraz ze wzrostem ilości feromonu τ_{ij} na krawędzi lub łuku $i \rightarrow j$ oraz wartości heurystycznej η_{ij}. Dobór parametrów α i β pozwala na balansowanie pomiędzy zachłannym korzystaniem z heurystyki (α=0) a wyborem w oparciu o stężenie feromonów (β=0).

Każda mrówka przechodzi graf (szuka rozwiązania), po czym następuje ocena każdej mrówki wyrażona wartością L_k. Rozwiązanie o najwyższej wartości L_k jest zapamiętywane. W dalszej kolejności odbywa się modyfikacja ilości feromonu τ_{ij} poprzez symulację zjawiska parowania feromonu oraz uwzględnienie wyników osiągniętych przez mrówki, które przechodziły krawędzią $i \rightarrow j$ zgodnie z formułą:

$$\tau_{ij}(l+1) = \rho\tau_{ij}(l) + \Delta\tau_{ij}$$

gdzie:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\,\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$$

ρ – współczynnik parowania feromonu $0 < \rho \leq 1$;

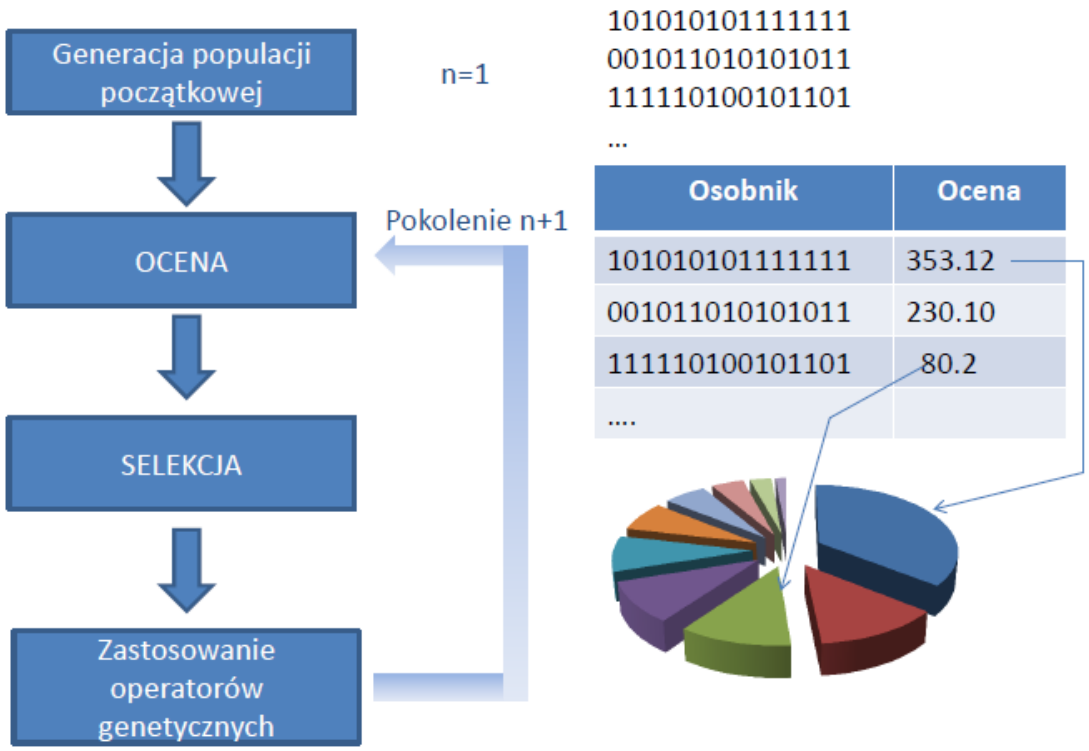
L_k–wartość funkcji oceny k-tej mrówki;

Q – parametr systemowy.

Podejście 5

Algorytm genetyczny

Algorytmy genetyczne



Operatory genetyczne (1)

- Krzyżowanie

```
001010 010100001    Rodzic 1
010111 000011100    Rodzic 2
        ↓
001010 000011100    Potomek
```

- Mutacja

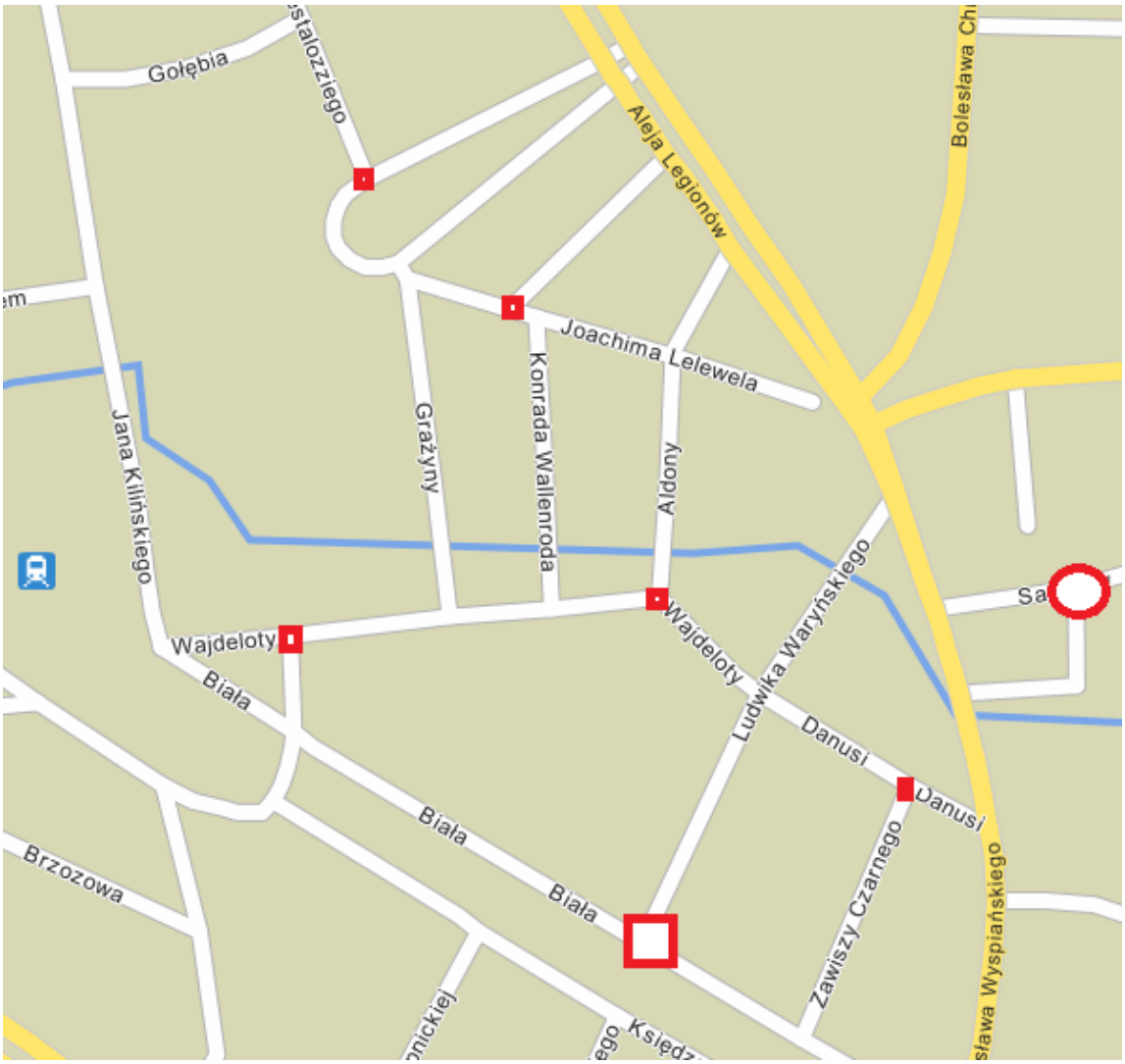
```
001010 010100001
        ↓
000010 011100001
```

```
PopSize = 100;
Proba_cross = 0.7;
Proba_mut = 0.5;
NbGen = 10;
Log = %T;
nb_disp = 10;
[pop_opt, fobj_pop_opt, pop_init, fobj_pop_init] = optim_ga(f_ocen, PopSize, NbGen, Proba_mut,
Proba_cross, Log, ga_params);

optim_ga: iteration 1 / 10
   min / max value found = 333.000000 / 436.000000
optim_ga: iteration 2 / 10
   min / max value found = 324.000000 / 411.000000
optim_ga: iteration 3 / 10
   min / max value found = 324.000000 / 402.000000
optim_ga: iteration 4 / 10
   min / max value found = 324.000000 / 388.000000
optim_ga: iteration 5 / 10
   min / max value found = 320.000000 / 378.000000
optim_ga: iteration 6 / 10
   min / max value found = 320.000000 / 358.000000
optim_ga: iteration 7 / 10
   min / max value found = 305.000000 / 344.000000
optim_ga: iteration 8 / 10
   min / max value found = 305.000000 / 333.000000
optim_ga: iteration 9 / 10
   min / max value found = 305.000000 / 333.000000
optim_ga: iteration 10 / 10
   min / max value found = 305.000000 / 333.000000
-->Wierz = get_wierzcholki(get_permut(ceil(abs(pop_opt(1))*2*k)));
-->[Dl,Tr]=oblicz_dlugosc_trasa(Wierz,g);
-->Wierz =    1.   6.   2.   8.   3.   9.   4.   5.   7.   1.
-->Dl   =   305.
```

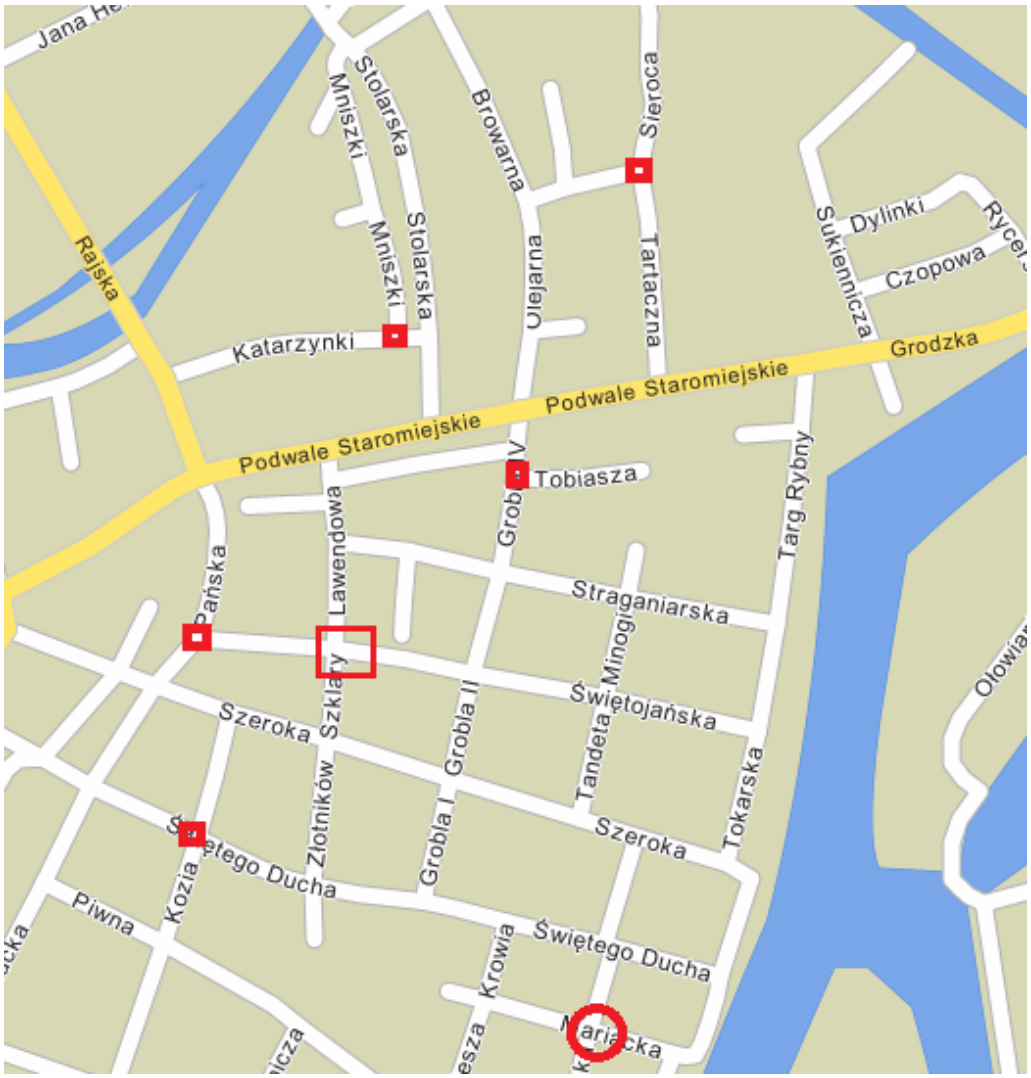
Problemy do rozwiązania – praca domowa dla osób zainteresowanych

Problem 1

Problem 2


Gołębia
Aleja Legionów
Bolesława Ch
Joachima Lelewela
Konrada Wallenroda
Grażyny
Aldony
Jana Kilińskiego
Wajdeloty
Wajdeloty
Ludwika Waryńskiego
Biała
Danusi
Danusi
Biała
Biała
Brzozowa
Zawiszy Czarnego
sława Wyspiańskiego

Problem 3


Rajska
Mniszki
Stolarska
Browarna
Sierocа
Tartaczna
Sukiennicza
Dylinki
Czopowa
Katarzynki
Podwale Staromiejskie
Grodzka
Ulejarna
Tobiasza
Lawendowa
Targ Rybny
Straganiarska
Pańska
Szklary
Minogi
Świętojańska
Szeroka
Grobla II
Grobla I
Tandeta
Tokarska
Złotników
Kozia
Piwna
Krowia
Mariacka
Ołowiana