Metody optymalizacji

Rafał Adamczak
Katedra Informatyki Stosowanej

www.fizyka.umk.pl/~raad/optymalizacja.pdf

Literatura

Maciej M. Sysło, Narsingh Deo, Janusz S. Kowalik, Algorytmy optymalizacji dyskretnej, Wydawnictwo naukowe PWN

Andrzej Stachurski, Wprowadzenie do optymalizacji, Oficyna wydawnicza Politechniki Warszawskiej

Reinhart Diestel, Graph Theory

David E. Goldberg, Algorytmy genetyczne i ich zastosowania.

Plan wykładu

Skojarzenia w grafach i ich zastosowania

a. Definicje

b. Twierdzenie Halla i kojarzenie małżeństw

c. Typy skojarzeń

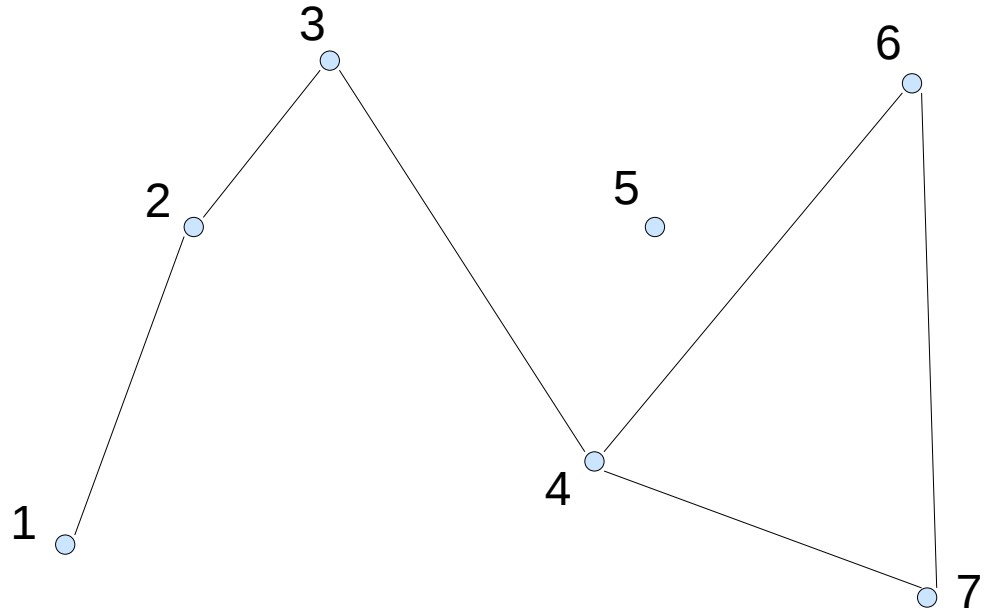
1. Algorytm węgierski
2. Algorytm Christofidesa
3. Algorytmy sieciowe (np. CPM, PERT), szeregowanie zadań, ścieżki krytyczne
4. Algorytmy optymalizacji liniowej z ograniczeniami
5. Zadanie transportowe z kosztami
6. Algorytmy optymalizacji kwadratowej z ograniczeniami
7. Kolorowanie grafów
8. Gradientowe algorytmy dla problemów optymalizacji bez ograniczeń
9. Algorytmy ewolucyjne w problemach optymalizacji

Przykłady praktycznych zadań optymalizacyjnych

- Optymalne planowanie produkcji
- Projektowanie sieci dróg, tras komunikacyjnych, sieci telefonicznych, gazowych, elektrycznych, komputerowych.
-

Przypomnienie podstawowych definicji

Grafem (prostym, nieskierowanym) nazywamy uporządkowaną parę $G=(V,E)$ gdzie *V* jest skończonym zbiorem wierzchołków (punktów, węzłów) $V=\{v_1, v_2, .., v_n\}$ natomiast *E* jest skończonym zbiorem krawędzi (linii) $E=\{e_1, e_2, .., e_m\}$



$V=\{1,...,7\}\ E=\{(1,2),(2,3),(3,4),(4,6),(6,7),(7,4)\}$

V(G) oznacza zbiór wierzchołków grafu G, a E(G) zbiór krawędzi grafu G

Wierzchołek *v* grafu jest incydentny z krawędzią *e* wtedy, gdy $v \in e$

Dwa wierzchołki *x*, *y* grafu *G* nazywane są sąsiadującymi, gdy są incydentne z tą samą krawędzią e=(x,y) w grafie *G*.

Zbiór sąsiadujących wierzchołków w grafie G do danego wierzchołka v oznaczany jest N(v).

Stopień wierzchołka oznacza liczbę krawędzi E(v) związanych z tym wierzchołkiem i równy jest liczbie wierzchołków sąsiadujących.

Minimalny stopień grafu G

$$\delta(G) = min\{N(v) : v \in V\}$$

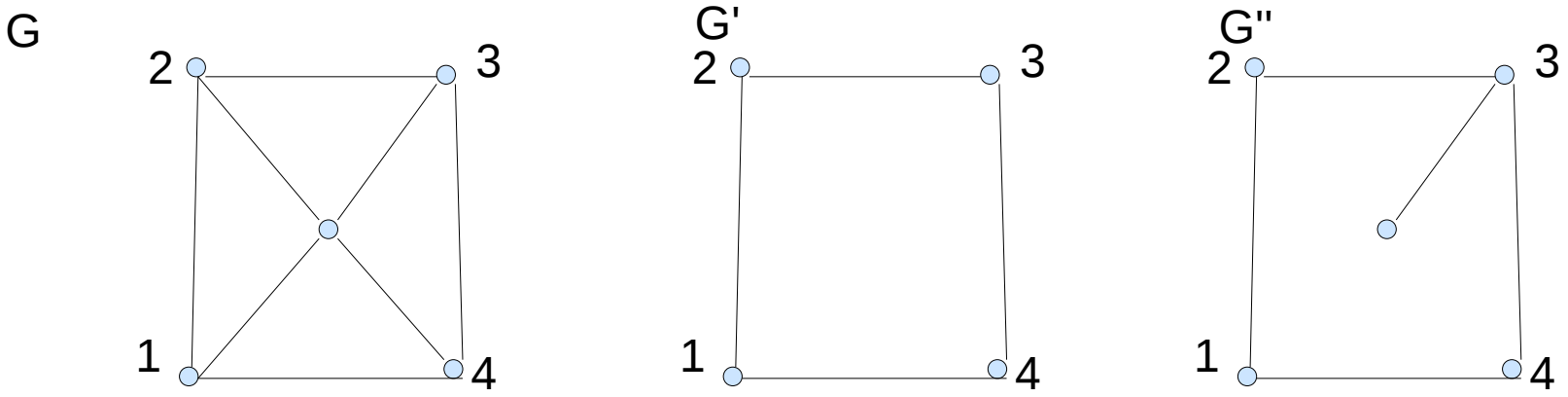
Maksymalny stopień grafu G

$$\Delta(G) = max\{N(v) : v \in V\}$$

Graf nazywamy skierowanym (digrafem) jeśli para wierzchołków (x,y) incydentnych z krawędzią e (nazywaną łukiem) jest parą uporządkowaną. Mówimy wtedy, że łuk jest skierowany z wierzchołka x do wierzchołka y, kierunek ten jest zaznaczony strzałką

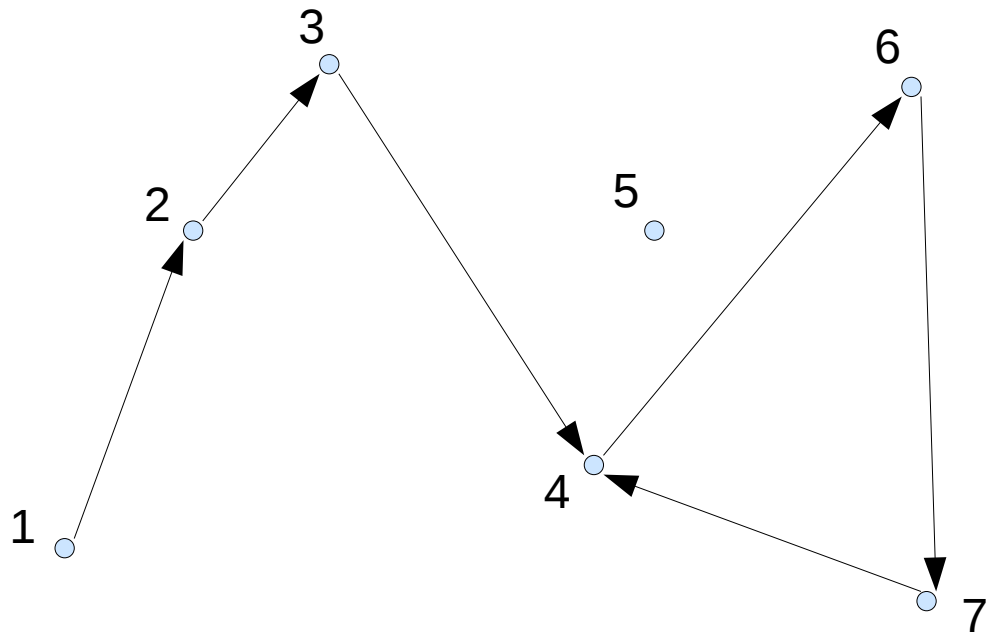
Załóżmy, że dane są grafy G=(V,E) i G'=(V',E'). Jeżeli $V' \subseteq V \; i \; E' \subseteq E$ wówczas G' jest podgrafem grafu G (G jest supergrafem dla grafu G'), lub mniej formalnie G zawiera G'.

Jeżeli $G' \subseteq G$ i wszystkie krawędzie grafu G' xy ϵ E, których krańce x,y ϵ V', wówczas G' jest indukowanym wierzchołkowo podgrafem G, natomiast V' indukuje lub rozpina graf G' na G. Albo inaczej G' nazywamy graf powstały przez usunięcie z grafu G pewnej liczby wierzchołków oraz wszystkich wychodzących z nich i wchodzących krawędzi



G' jest pod grafem indukowanym a G" nie!

$G' \subseteq G$ Jest rozpinającym podgrafem G jeżeli V' rozpina się na całe G czyli V'=V.



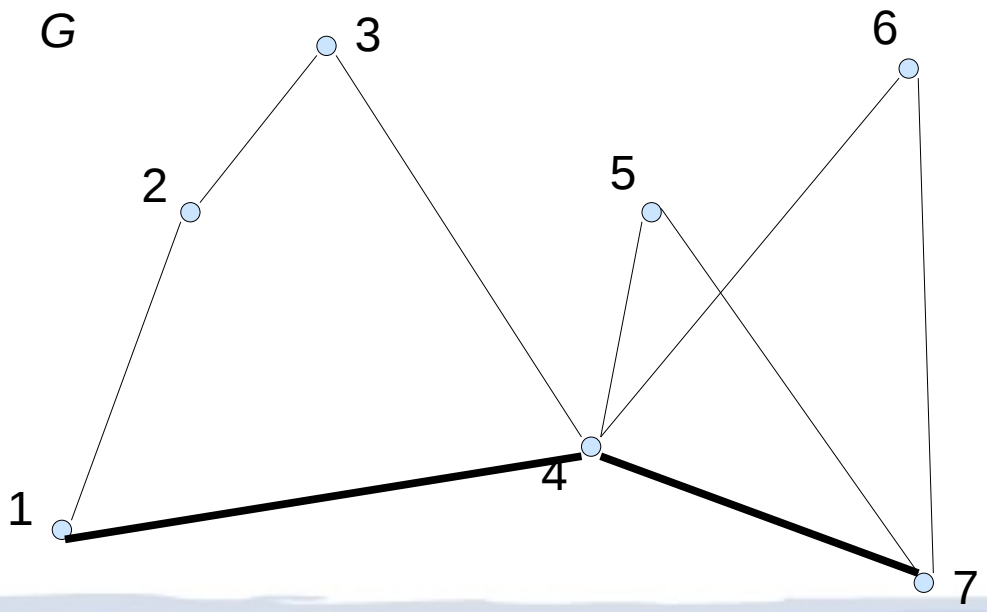
Siecią nazywamy graf, w którym każdemu łukowi (krawędzi) jest przyporządkowana liczba, zwana wagą.

Ścieżki i cykle

Ścieżką nazywamy nie pusty graf P=(V,E)

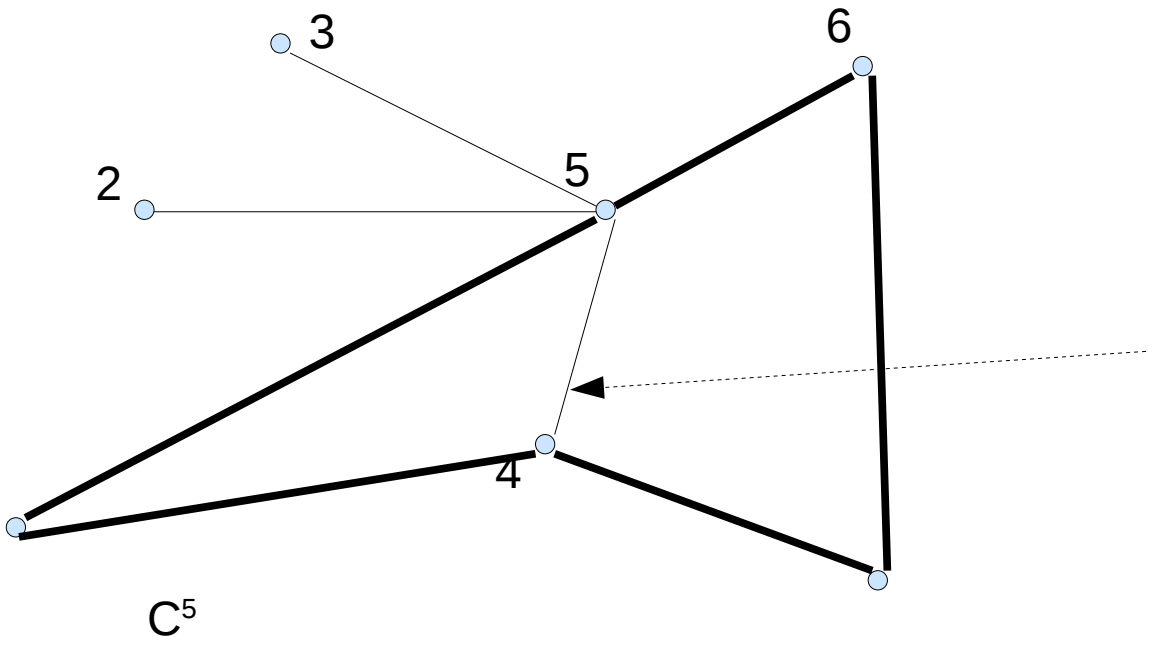
$$V=\{x_{0,}..,x_k\}\ E=\{x_0x_1,x_1x_2,...,x_{k-1}x_k\}$$

gdzie wszystkie węzły x_i są różne
Węzły x_0 i x_k są nazywane końcami grafu P, natomiast pozostałe węzły węzłami wewnętrznymi. Liczba krawędzi grafu P nazywana jest długością ścieżki, a ścieżka o długości k oznaczana P^k



Graf nazywamy spójnym jeżeli dla każdej pary wierzchołków istnieje ścieżka, która je łączy.

Załóżmy, że $P=x_0...x_{k-1}$ i $k>=3$, wówczas graf C nazywany jest cyklem gdy $C=P+x_0x_{k-1}$.
Cykl o długości k oznaczany jest C^k



C^5

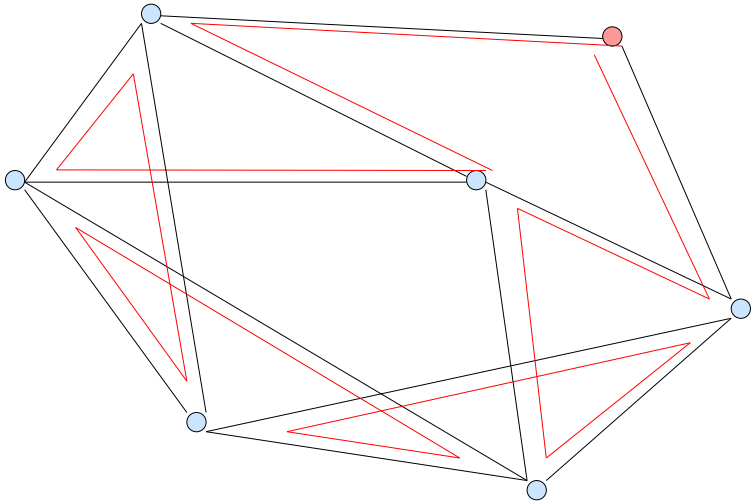
Cięciwa cyklu C^5

Cykl C grafu G nazywamy indukowanym gdy nie ma żadnej cięciwy
Graf nie zawierający żadnego cyklu nazywamy acyklicznym.

Cykl Eulera i cykl Hamiltona

Cykl Hamiltona to ścieżka, która przechodzi przez wszystkie jego wierzchołki dokładnie raz i wraca do wierzchołka startowego.

Cykl Eulera (rozpoczyna się i kończy w tym samym węźle) jest cyklem przechodzącym przez każdą krawędź grafu tylko raz.

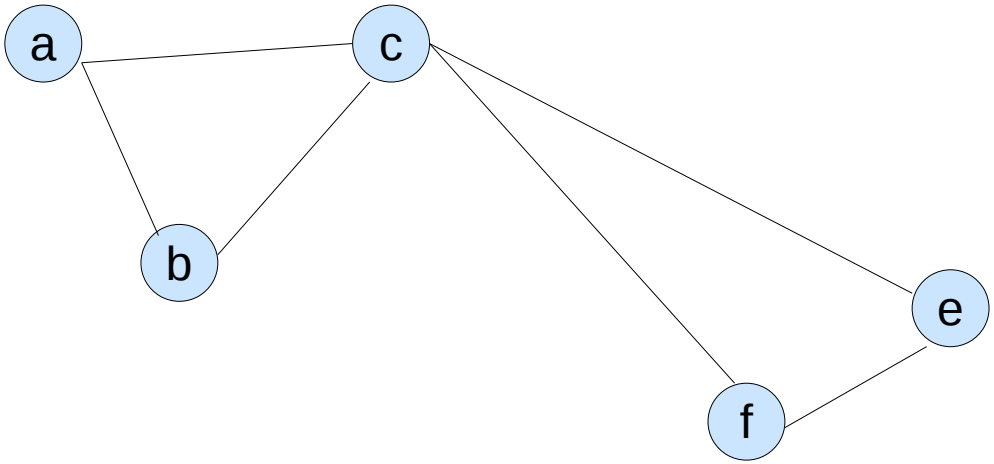


Dla grafu spójnego i skierowanego: Cykl Eulera istnieje wtedy i tylko wtedy, gdy dla każdego Wierzchołka liczba krawędzi wchodzących i wychodzących jest równa

Dla grafu spójnego nieskierowanego: Cykl Eulera istnieje wtedy i tylko wtedy gdy stopień Każdego wierzchołka jest parzysty.

Wyznaczenia cyklu Eulera

Zaczynamy od dowolnego wierzchołka, dodajemy ten wierzchołek na stos i idziemy do następnego osiągalnego wierzchołka, a łącząca z nim drogę usuwamy, dodajemy ten wierzchołek na stos i idziemy do następnego osiągalnego wierzchołka łączącą drogę Usuwamy itd.
Jeżeli nie możemy nigdzie pójść pobieramy element ze stosu (będzie kolejnym w cyklu), z ostatniego wierzchołka na stosie idziemy dalej. Wszystko powtarzamy tak długo jak jak długo istnieją jakieś wierzchołki na stosie.



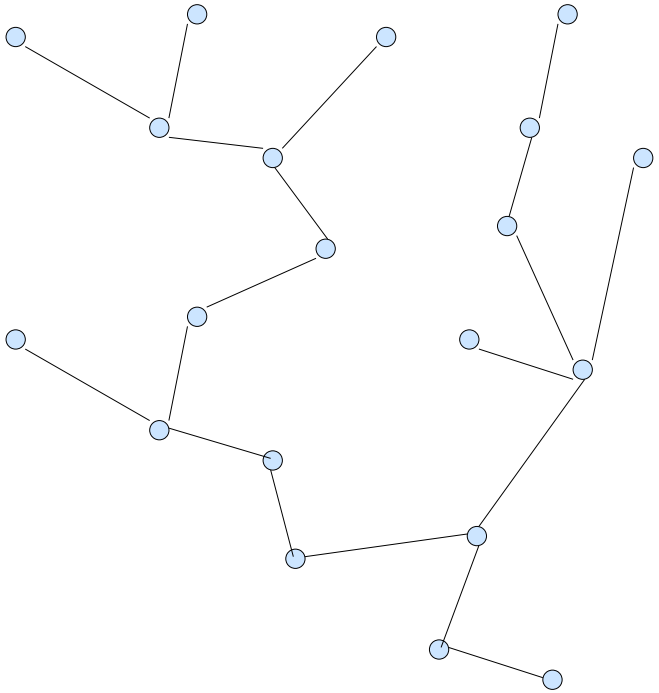
Znajdowanie cyklu Eulera

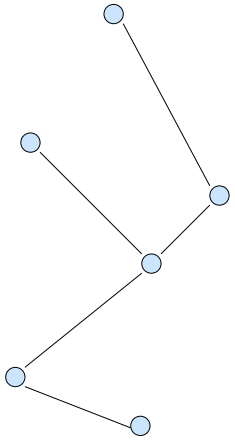
```
procedure CyklEulera(G, u)
        push(STOS,u);
        while STOS <> ø do
                v:=top(STOS);
                if S[v]=ø then
                begin
                        pop(STOS);
                        LISTA ← v; {wstaw v na początek}
                end
                else
                begin
                        w:=pop(S[v]);
                        push(STOS,w);
                        Usuń krawędź
                end
        return LISTA;
end
```

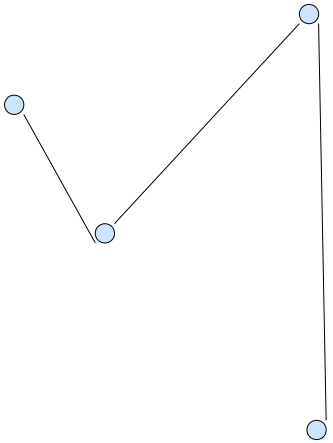
Drzewa i lasy

Drzewo - graf acyklicznym i spójny.

Las – graf, którego każdy spójny podgraf jest drzewem.



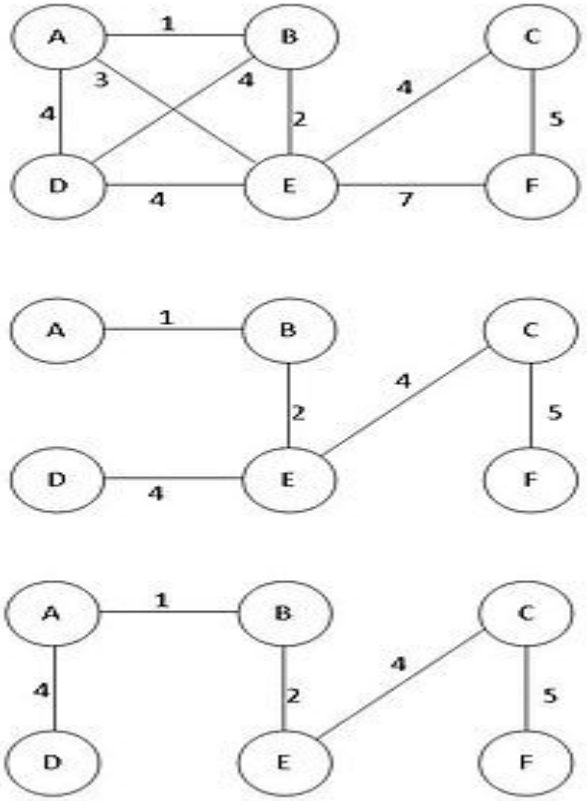




Drzewo rozpinające

Drzewo rozpinające grafu G=(V,E) to drzewo, które zawiera wszystkie wierzchołki grafu G, zaś zbiór krawędzi drzewa jest podzbiorem krawędzi grafu

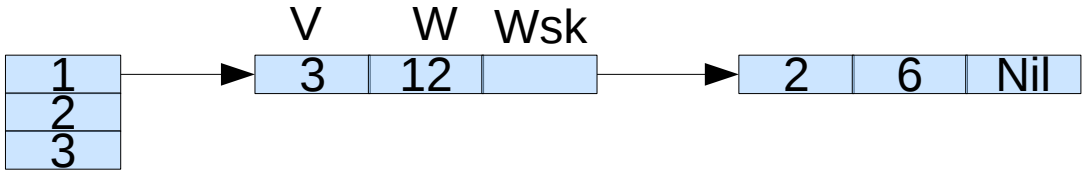
Minimalne drzewo rozpinające grafu G(V,W,w) jest to drzewo, dla którego suma wag jest najmniejsza z możliwych.



http://en.wikipedia.org/wiki/Minimum_spanning_tree

Komputerowa reprezentacja sieci

- Najprostsza i najbardziej popularna reprezentacja sieci ma postać macierzy wag W=[w_{ij}]stopnia n, w której element wij jest wagą łuku (i,j)
- Reprezentacja w postaci wag zajmuje n^2 elementów. Jeśli sieć jest rzadka to lepiej zastosować reprezentację w postaci listy krawędzi, w której każdy łuk reprezentowany jest przez 3 liczby: dwa wierzchołki końcowe łuku i jego waga. Taka reprezentacja może być łatwo zaimplementowana w postaci 3 tablic.
- Połączone listy sąsiadów. Reprezentacja przy pomocy n list



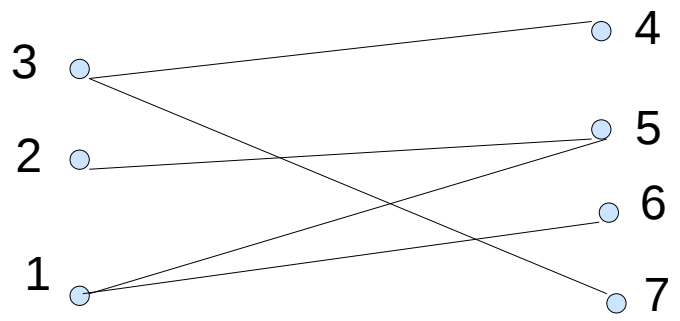
- Pęk wyjściowy. Jeśli nie jest konieczne dołączanie i usuwanie łuków to wystarczy pamiętać tylko sąsiadów

Grafy r-dzielne

Niech r>=2 będzie liczba naturalną. Graf G=(V,E) nazywamy r-dzielnym wtedy gdy zbiór V daje się podzielić na r klas (podzbiorów) $V_1 ... V_r$, takich, że każda krawędź grafu ma koniec w innym podzbiorze zbioru V.

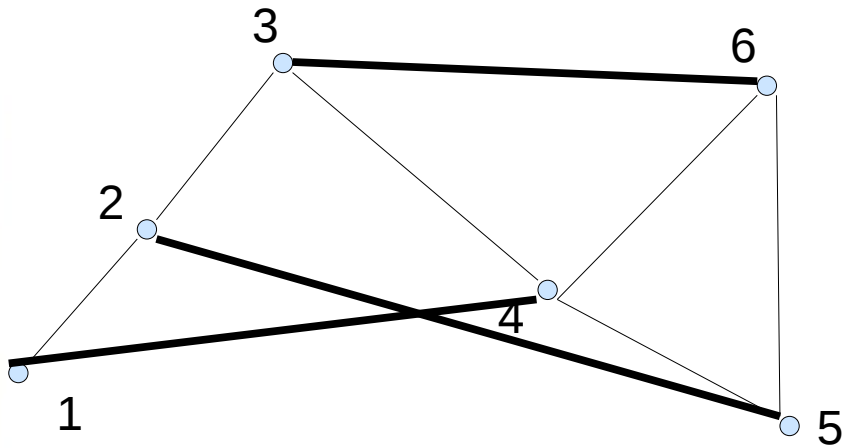
$$V = V_1 \cup V_2 \cup V_3 \cup ... V_r \qquad \forall_{i,j;i \neq j} V_i \cap V_j = \emptyset$$

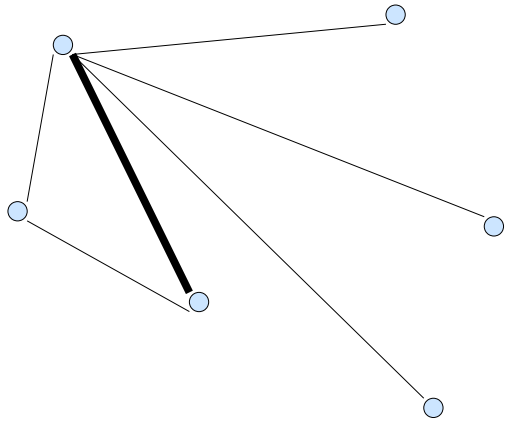
W przypadku gdy, r=2 mówi się o grafie dwudzielnym.



Skojarzenia i pojęcia z nimi związane

Podzbiór krawędzi M w nieskierowanym grafie G=(V,E) nazywa się skojarzeniem, jeżeli żadna krawędź w M nie ma wspólnego wierzchołka.





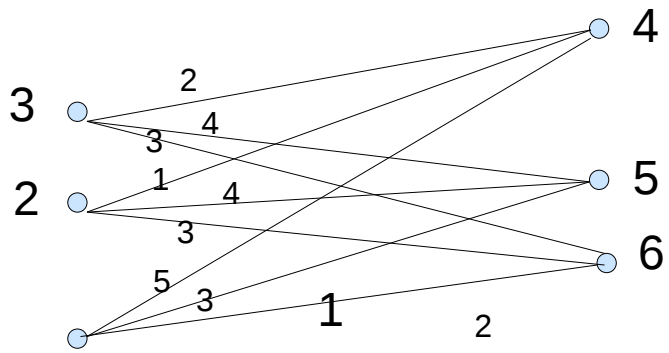
(1,4) (2,5) (3,6) (1,2)(4,3)(5,6)

Krawędź nazywa się skojarzoną jeżeli należy do M i nieskojarzoną jeżeli do M nie należy.
Podobnie wierzchołek jest skojarzony (nasycony) jeżeli jest końcem krawędzi ze skojarzenia w przeciwnym razie wierzchołek jest eksponowany lub wolny.
Skojarzenie nazywamy pełnym gdy suma wierzchołków należących do krawędzi skojarzenia M równa jest zbiorowi wierzchołków w grafie.

$$\coprod_{e \in M} e = V$$

W przypadku grafów dwudzielnych z wagami G=(V_1,V_2;E;W), a więc grafów, w których każdej krawędzi (i,j) przydzielona jest waga w_{ij} skojarzenie M jest sumą wag odpowiednich krawędzi należących do M

$$w(m)=\sum_{e\in M} w(e)$$

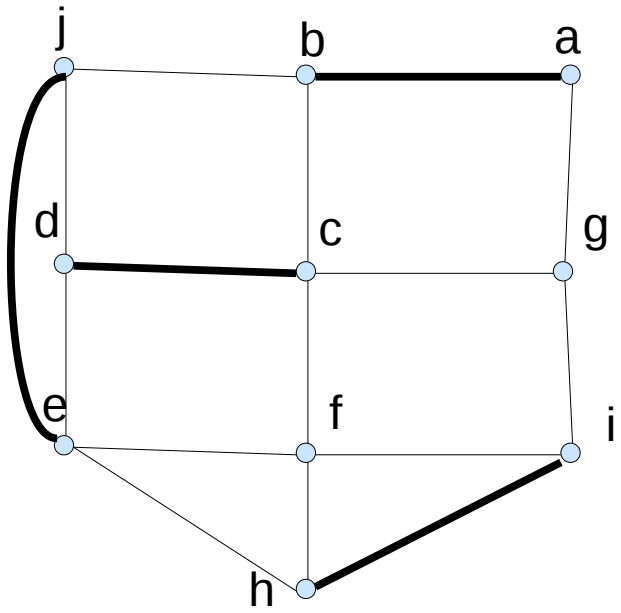


W grafie dwudzielnym G=(V_1,V_2;E) skojarzenie jest pełne gdy

$$\forall x\in V_1 \exists e\in E\ x\in e$$

Droga naprzemienna i rozszerzające względem skojarzenia M

Drogę $P=(v_1,...,v_k)$ nazywa się drogą naprzemienną względem skojarzenia M, jeśli jej krawędzie na przemian należą i nie należą do M.



M={(a,b),(c,d),(e,j),(h,i)}

Drogi naprzemienne
P=(a,b,c,d)
P=(e,h,i,g)

Drogę naprzemienną zaczynającą się w wierzchołku wolnym i kończącą w innym wierzchołku wolnym nazywa się drogą powiększającą (P=(g,a,b,c,d,j,e,f))

Algorytm znajdowania drogi rozszerzającej

Dla grafu dwudzielnego $G=(V_1 \cup V_2, E)$ oraz skojarzenia M konstruujemy graf skierowany $G_M=(V_1 U V_2, E_M)$

$$E_M=(v_1, v_2): v_1 v_2 \in E, v_1 \in V_1, v_2 \in V_2 \cup (v_2, v_1): v_1 v_2 \in M, v_1 \in V_1, v_2 \in V_2$$

```
ZNAJDŹ-ŚCIEŻKĘ-rozszerzającą G=(V1UV2,E),M
 1   V1' zbiór wierzchołków wolnych w V1
 2   V2' zbiór wierzchołków wolnych w V2
 3  skonstruuj graf skierowany GM=(V1UV2,EM)
 5  znajdź ścieżkę p z V1' do V2' w GM
 6  if p nie istnieje then
 7    return NIL (nie ma ścieżki powiększającej)
 8  usuń cykle z p tak aby p była ścieżką prostą
 9  return p ( to ścieżka powiększająca w )
```

Wyszukanie ścieżki rozszerzającej ma złożoność O(|E|)

Twierdzenie Berge'a

Twierdzenie Berge'a. Skojarzenie M w grafie G jest najliczniejsze wtedy i tylko wtedy, gdy G nie zawiera drogi powiększającej względem M.

Załóżmy, że istnieje skojarzenie M' liczniejsze niż M. Rozważmy graf G'=(V,M⊕M') (M∪M'-M∩M'). Każdy wierzchołek w grafie G; ma stopień co najwyżej 2, w związku z tym składa się z rozłącznych ścieżek i cykli. W każdym cyklu liczba krawędzi ze zbioru M i M' jest taka sama. Natomiast w ścieżkach może występować co najwyżej o jedną krawędź więcej z któregoś skojarzenia. W grafie G' jest więcej krawędzi z M' niż z M, a więc musi też istnieć ścieżka, na której jest więcej krawędzi z M'. Oczywiście jest to ścieżka powiększająca.

Twierdzenie Halla o kojarzeniu małżeństw

Mamy zbiór panien na wydaniu V_1 i zbiór kawalerów V_2, każda panna x ma zbiór upatrzonych kandydatów na męża N(x). Panny ze zbioru V_1 można wydać za mąż za kawalerów ze zbioru V_2 tak, aby każda dostała upatrzonego męża wtedy i tylko wtedy gdy dla każdego zbioru panien S liczba kawalerów upatrzonych przez panny z S jest nie mniejsza niż |S|.

Załóżmy,że mamy dwudzielny graf G=(V,E), w którym wierzchołki pochodzą z klas V_1 i V_2. W grafie tym istnieje skojarzenie, którego krawędzie są incydentne ze wszystkimi wierzchołkami z V_1 wtedy i tylko wtedy, gdy dla każdego podzbioru wierzchołków $S \subseteq V_1$ zachodzi $|N(S)| \geq |S|$, gdzie N(S) to zbiór wierzchołków z V_2 połączonych krawędzią z jednym z wierzchołków z S

Dowód Twierdzenia Halla

Niech M będzie dokładnym skojarzeniem oraz niech $S \subseteq V_1$.

Rozważmy wszystkie wierzchołki zbioru V_2, które są skojarzone z danym zbiorem S Oznaczmy go przez M(S). Oczywiście z definicji skojarzenia $|M(S)|=|S|$. Natomiast $M(S) \subseteq N(S)$, ponieważ wszystkie elementy M(S) są sąsiadami elementów zbioru S. Stąd $|N(S)|>=|M(S)|$ czyli $|N(S)|>=|S|$

A teraz udowodnimy, że jeżeli $|N(S)|>=|S|$ dla wszystkich $S \subseteq V_1$ wówczas $G(V_1,V_2)$ ma skojarzenie dla każdego wierzchołka V_1.

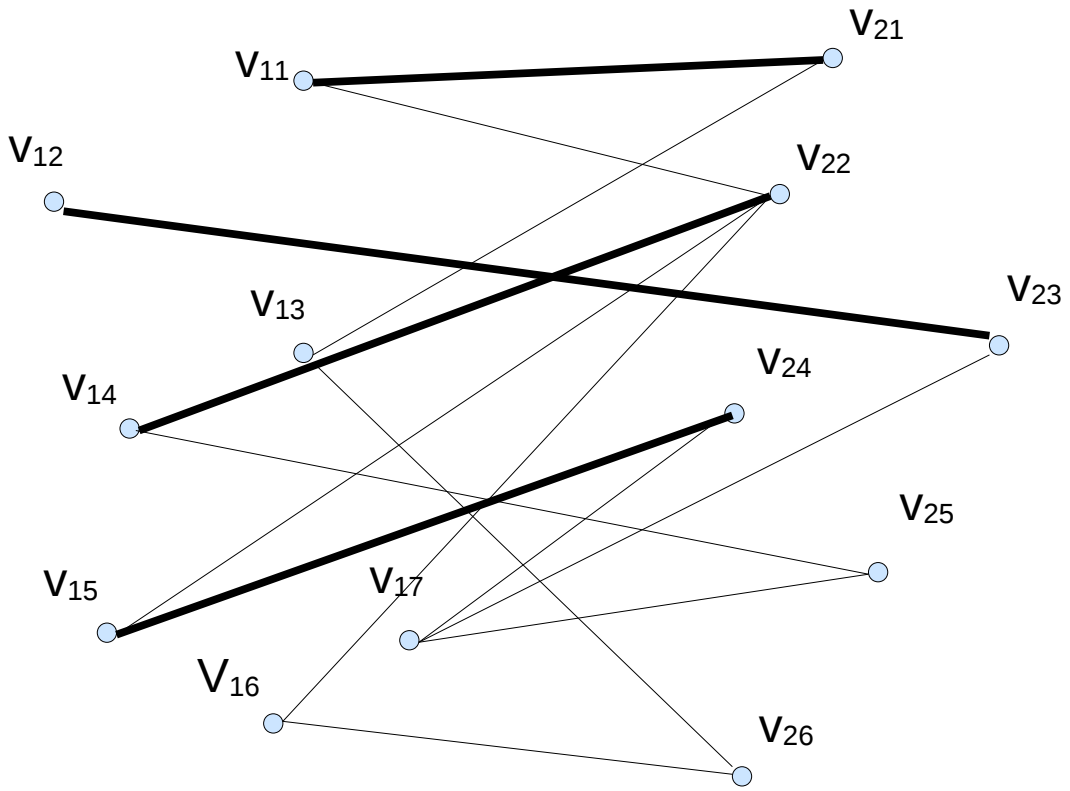
Załóżmy, że nie ma takiego skojarzenia, dla którego graf G ma skojarzenie dla każdego wierzchołka V_1. Niech M będzie maksymalnym skojarzeniem w grafie G. Ponieważ M nie jest skojarzeniem pełnym, w zbiorze V_1 musi istnieć wierzchołek s nie skojarzony. Niech Z będzie zbiorem wierzchołków w G, które są osiągalne z s przez ścieżkę naprzemienną względem skojarzenia M. Ponieważ M jest maksymalnym skojarzeniem, nie może istnieć ścieżka rozszerzająca. Niech $S=Z \cap V_1$ i $T=Z \cap V_2$. Wówczas każdy wierzchołek w T jest skojarzony z jakimś wierzchołkiem z S-{s} i każdy wierzchołek z S-{s} jest skojarzony z Jakimś wierzchołkiem z T. A więc $|T|=|S|-1$, oraz T=N(S). Czyli S jest podzbiorem V_1 takim, że $|N(S)|=|S|-1$, co jest sprzeczne z założeniem.

Skojarzenia w grafach dwudzielnych

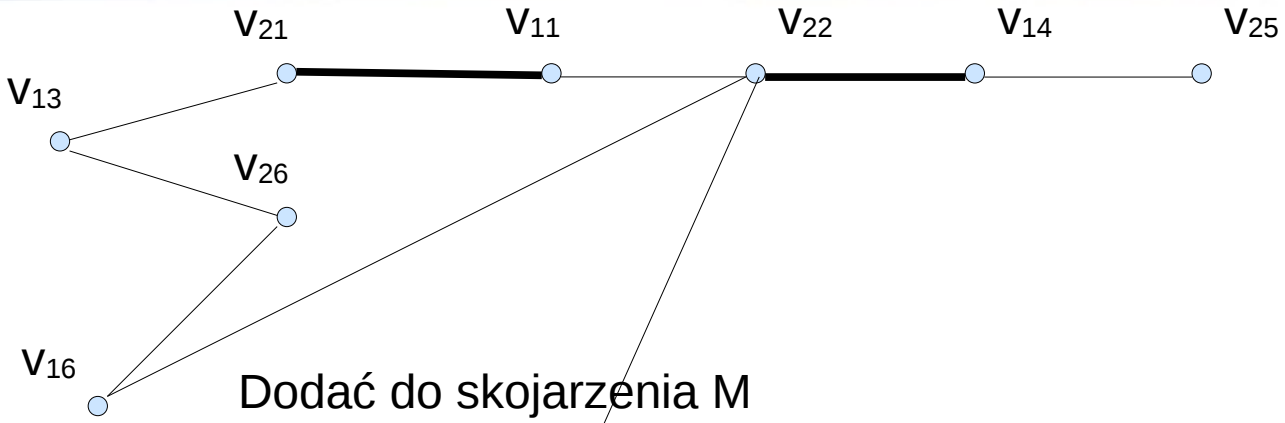
Przypuśćmy, że zgłosiło się 4 kandydatów $k_1,..,k_4$ na 5 wakatów $w_1,...,w_5$. Kandydat k_1 ma kwalifikacje do obsadzenia stanowisk w_1 i w_5. Kandydat k_3 jest wykwalifikowany by obsadzić stanowiska w_1, w_2, w_3, w_4, w_5, natomiast kandydat k_4 do stanowisk w_1, w_4.

Zadania powyższe odpowiadają znalezieniu odpowiednich skojarzeń w grafie dwudzielnym. Rozwiązanie problemu skojarzenia w grafie dwudzielnym jest znacznie łatwiejsze niż w przypadku dowolnego grafu

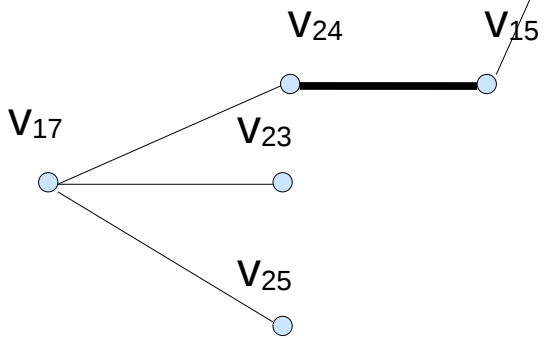
Pełne skojarzenie V_1 z V_2 w grafie dwudzielnym $G=(V_1 \cup V_2,E)$ to skojarzenie, w którym każdy wierzchołek z V_1 jest skojarzony. Jak znaleźć skojarzenie w G o jak największej liczbie krawędzi?



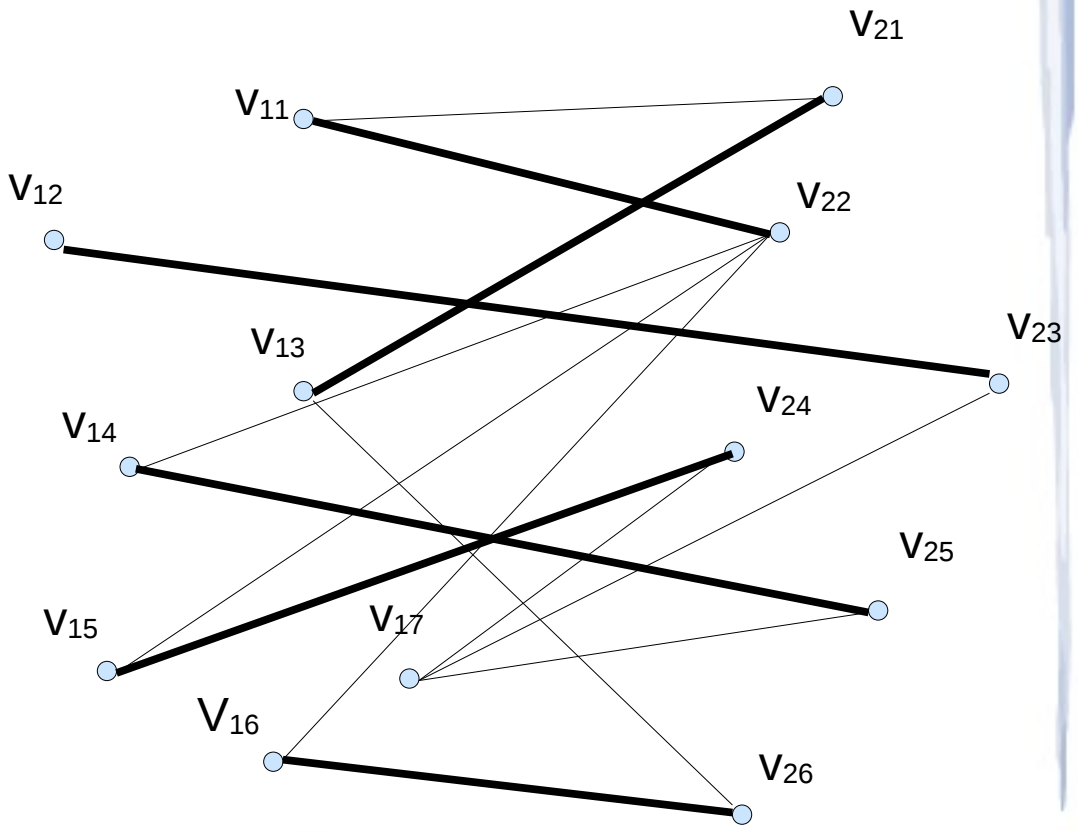
	V_{21}	V_{22}	V_{23}	V_{24}	V_{25}	V_{26}
V_{11}	1	1				
V_{12}			1			
V_{13}	1					1
V_{14}		1				1
V_{15}		1		1		
V_{16}		1				1
V_{17}			1	1	1	



Dodać do skojarzenia M



	v21	v22	v23	v24	v25	v26
v11	1	1				
v12			1			
v13	1					1
v14		1			1	
v15		1		1		
v16		1				1
v17			1	1	1	



Twierdzenie Berge'a jest podstawą prawie wszystkich algorytmów rozwiązywania problemu skojarzenia.

Procedura znajdowania największego skojarzenia jest następująca: rozpocznij z dowolnym skojarzeniem M, znajdź drogę powiększającą P względem M i następnie utwórz skojarzenie $M'=M \oplus P$ ($M \cup P - M \cap P$) składające się z tych krawędzi w M lub w P, które nie należą jednocześnie do M i P. zbiór M' ma oczywiście o jeden element więcej niż M. Powtarzaj tą procedurę aż do momentu otrzymania skojarzenia, względem którego graf nie zawiera drogi powiększającej.

MAKSYMALNE-SKOJARZENIE($G = (V_1 \cup V_2, E)$)
1)$M=\emptyset$
2)repeat
3) p =ZNAJDŹ-ŚCIEŻKĘ-POWIĘKSZAJĄCĄ(G,M)
4) if $p \neq$ NIL then
5) $M=M \oplus p$
6) until p = NIL
7) return M

Złożoność algorytmu O(|V||E|)

Przedstawiona metoda może być czasochłonna, gdyż liczba dróg w grafie na ogół rośnie wykładniczo wraz ze wzrostem rozmiaru grafu.

Skojarzenie początkowe

Odpowiedni duże skojarzenie początkowe można otrzymać kojarząc wierzchołek wolny v z pierwszym przyległym do niego innym wierzchołkiem wolnym.

Niech *expo* oznacza liczbę wierzchołków wolnych w grafie, a tablica (rozmiaru n) *mate* zawiera skojarzenie w grafie. To znaczy , jeśli (x,y) jest krawędzią skojarzenia, to *mate*(x)=y i *mate*(y)=x. Dla wierzchołka wolnego i mamy *mate*(*i*)=0.

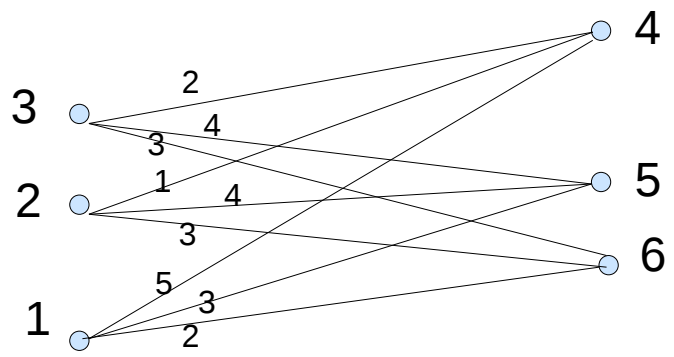
```
1) for v ∈ V do mate(v)←0;
2) expo←n;
3)for u ∈ V do begin
4)    if(mate(u)=0) then
5)          if(u jest przyległy do wierzchołka wolnego v the begin
6)                mate(u)←v;mate(v)←u;
7)                expo←expo-2
8)          end
9)end
```

Algorytm węgierski

Załóżmy, że mamy N pracowników i N różnych prac, które powinny zostać przydzielone pracownikom. Znane są płace, które trzeba by zapłacić za wykonanie poszczególnych prac przez każdego z pracowników. Zadaniem jest zminimalizowanie całkowitych kosztów.

Niech c będzie macierzą kosztów NxN, gdzie c_{ij} oznacza koszt wykonania j pracy przez i tego pracownika.

$$\begin{pmatrix} 2 & 4 & 3 \\ 1 & 4 & 3 \\ 5 & 3 & 2 \end{pmatrix}$$



Z punktu widzenia teorii grafów problem można przedstawić jak znalezienie minimalnego (maksymalnego) skojarzenia ważonego w grafie G=(V,E,C)

Etykietowanie

Etykietowanie wierzchołka odbywa się poprzez użycie funkcji $l : V \rightarrow \mathbb{R}$

takiej, że $l(x) + l(y) \geq c(x,y), x \in V_1, y \in V_2$

Graf równy $G_l(V,E_l)$ (dla danego l) to taki graf dla którego

$$E_l = (x,y) : l(x) + l(y) = c(x,y)$$

Twierdzenie KM. Niech *l* będzie funkcją etykiety spełniająca warunek dla grafu dwudzielnego z wagami – G. Jeżeli podgraf G_l zawiera skojarzenie doskonale M* , to skojarzenie M* jest skojarzeniem optymalnym grafu G (maksymalnym).

Dowód
Niech M' będzie dowolnym pełnym skojarzeniem grafu G (niekoniecznie w E_l). Ponieważ Każdy wierzchołek występuje tylko raz w skojarzeniu M' to mamy

$$w(M') = \sum_{e \in M'} C(e) \leq \sum_{e \in M'} (l(e_x) + l(e_y)) = \sum_{v \in V} l(v)$$

Jest to więc górne ograniczenie kosztu dla dowolnego skojarzenia w grafie G

Niech M będzie pełnym skojarzeniem w E_l. Wówczas

$$w(M)=\sum_{e\in M} C(e)=\sum_{v\in V} l(v)$$

A więc

$$w(M')\leqslant w(M)$$

Twierdzenie KM transformuje nam problem z problemu optymalizacyjnego do problemu kombinatorycznego znajdowania pełnego skojarzenia.

Znajdowanie początkowych wartości etykiet

$$\forall y \in V_{2,}\, l(y) = 0, \qquad \forall x \in V_{1,}\, l(x) = \max_{y} x \in V_2\, w(x, y)$$

Przy takim etykietowaniu relacja poniższa jest oczywiście spełniona

$$\forall\, x \in V_{1,}\, \forall\, y \in V_2, c(x, y) \leqslant l(x) + l(y)$$

Algorytm Kuhna Munkersa

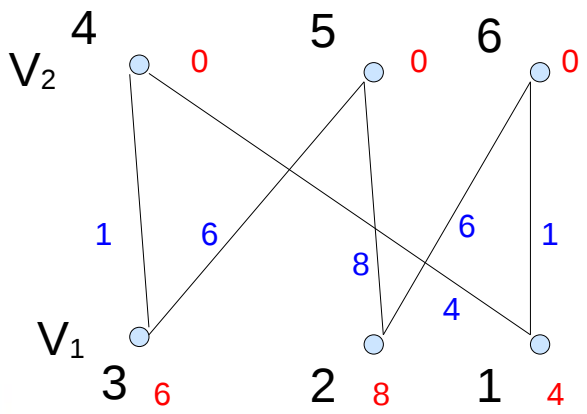
1)Ustaw początkową wartość etykiet dla wszystkich węzłów i wyznacz początkowe M w E_l
2)Jeżeli skojarzenie jest pełne to stop. W przeciwnym razie wybierz wolny wierzchołek u w V1 i utwórz zbiór S={u}, T=Ø
3) Jeżeli $N_l(S)=T$ to dokonaj zmiany etykietowania (wymuszamy $N_l(S)\neq T$)

$$\alpha_l = \min_{x\in S,\, y\notin T} l(x)+l(y)-w(x,y)$$

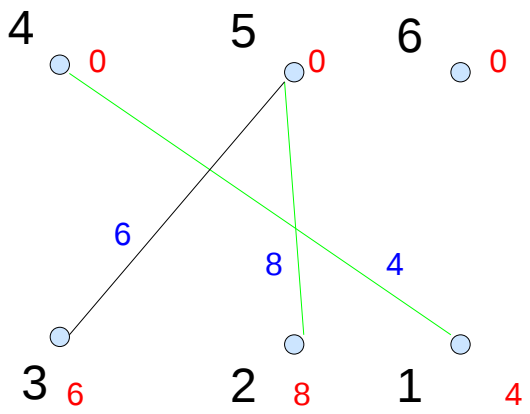
$$l'(v)=\begin{cases} l(v)-\alpha_l \text{ jeżeli } v\in S \\ l(v)+\alpha_l \text{ jeżeli } v\in T \\ l(v) \text{ w przeciwnym razie} \end{cases}$$

4)Jeżeli $N_l(S)\neq T$ wybierz $y\in N_l(S)-T$, jeżeli y jest wolny to istnieje ścieżka rozszerzająca z u do y Znajdź ją i idź do kroku 2. Jeżeli y jest skojarzony z z to dodaj krawędź (y,z) do drzewa naprzemiennego i S=SU{z}, T=TU{y} i idź do kroku 3.

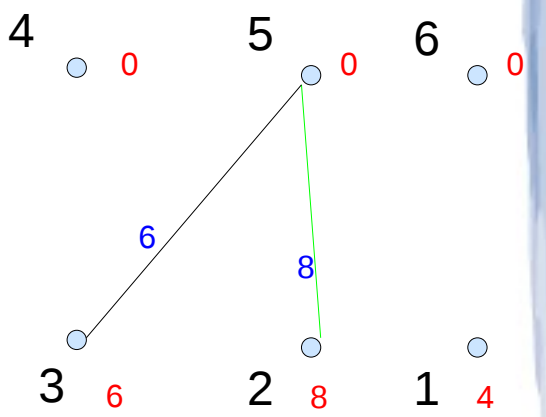
Przykład działania



Początkowy graf G



Graf G_l+skojarzenie

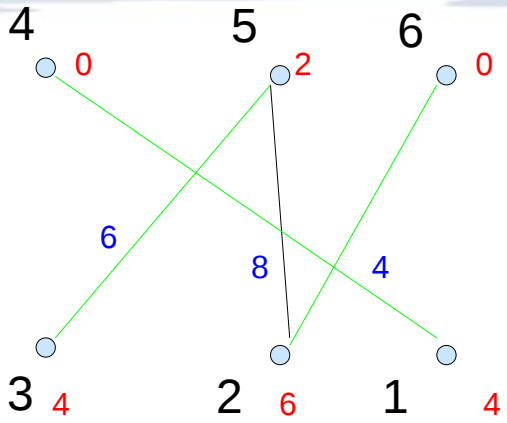


Drzewo naprzemienne

S={3},T=0, ponieważ $N_l(S) \neq T$ idziemy do kroku 4, wybieramy 5, które należy do $N_l(S)$-T
5 jest skojarzony z 2 więc dodajemy do drzewa krawędź (5,2), S={3,2}, T={5}, teraz $N_l(S)=T$ więc idziemy do kroku 3
Wyliczanie alfy

$$\alpha_l = \min_{x \in S,\, y \notin T} \begin{cases} 6+0-1,(3,4) \\ 6+0-0,(3,6) \\ 8+0-0,(2,4) \\ 8+0-6(2,6) \end{cases} = 2$$

Zredukuj etykiety S o 2, zwiększ etykiety T o 2



S={3,2}, N_l(S)={5,6}, T={5}

Wybierz $6 \in N_l(S) - T$ *i dodaj do* T

6 nie jest w skojarzeniu M w związku z czym znaleziona została ścieżka rozszerzająca 3,5,2,6 (występują dwa wolne końce). Można więc powiększyć skojarzenie M. To skojarzenie jest pełne, więc musi być optymalne!

Złożoność algorytmu $O(|V|^3)$

Inna wersja algorytmu węgierskiego

1)Dla każdego węzła z lewej strony znajdź krawędź minimalną i odejmij tą wartość od wszystkich wartości krawędzi wychodzących z tego węzła

2) Wykonaj tą sama procedurę dla wszystkich węzłów po prawej stronie (ten krok nie jest absolutnie konieczny ale zmniejsza liczbę iteracji)

3) Pokryj minimalną liczbę wierszy i kolumn zawierających zera, jeżeli liczba wierszy/kolumn pokryta jest równa liczbie wierszy/kolumn w macierzy to koniec, w przeciwnym razie znajdź przejdź do następnego kroku

4) Znajdź wartość minimalną w obszarze nie pokrytym. Znaleziona wartość jest odjęta od wszystkich nie pokrytych wartości i dodana do wartości pokrytych zarówno w kolumnie jak i wierszu (znajdujących się na przecięciu linii zakreślonych). Idź do kroku 3.

Przykład

14	5	8	7
2	12	6	5
7	8	3	9
2	4	6	10

14	5	8	7	**5**
2	12	6	5	**2**
7	8	3	9	**3**
2	4	6	10	**2**

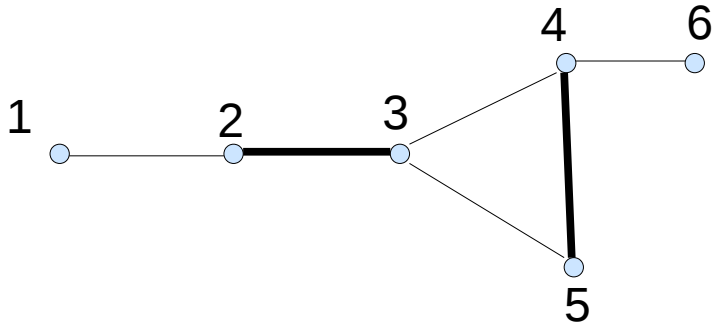
9	0	3	2
0	10	4	3
4	5	0	6
0	2	4	8
0	**0**	**0**	**2**

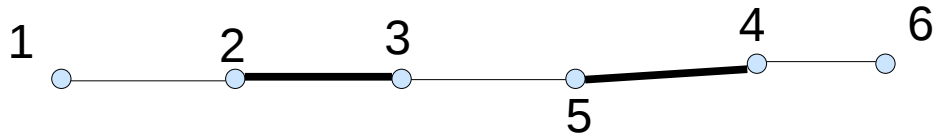
→	9	0	3	0
	0	10	4	1
→	4	5	0	4
	0	2	4	6
	↑			

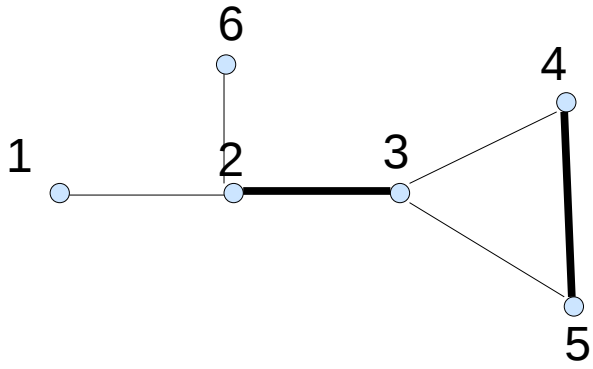
→	10	0	3	0
	0	9	3	0
→	5	5	0	4
	0	1	3	5
	↑			↑

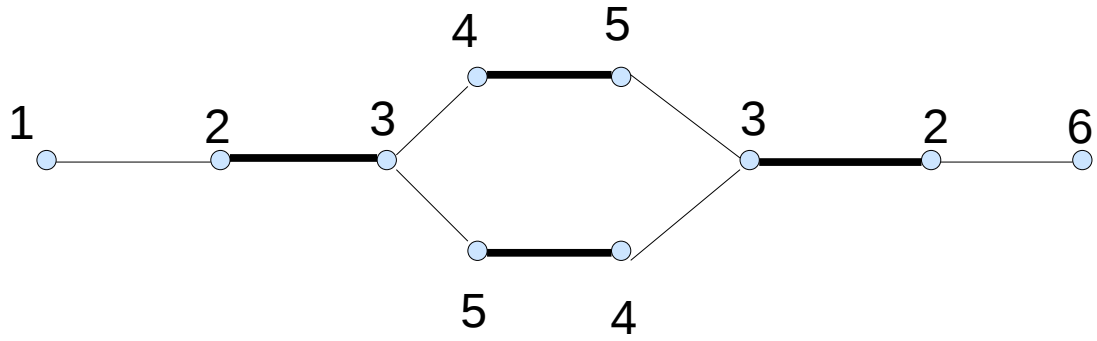
Skojarzenia w grafach nie dwudzielnych

Poprzednio opisana metoda opiera się na założeniu, że długość cyklu w grafie dwudzielnym nigdy nie jest nieparzysta. Gdy w grafie występuje cykl nieparzysty wówczas w drzewie naprzemiennym powstają tzw kielichy









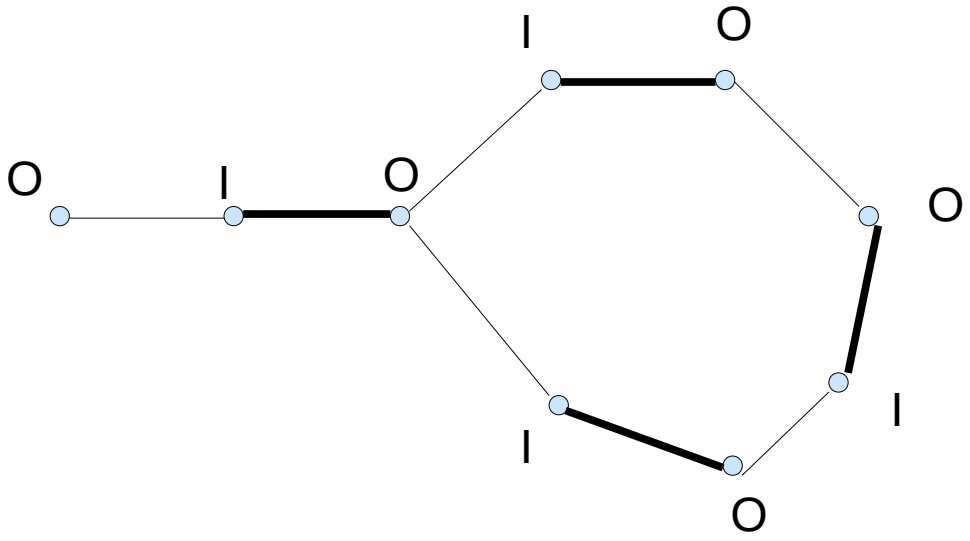
Powstała ścieżka rozszerzające pomimo że widać z grafu, że zadane skojarzenie jest maksymalne

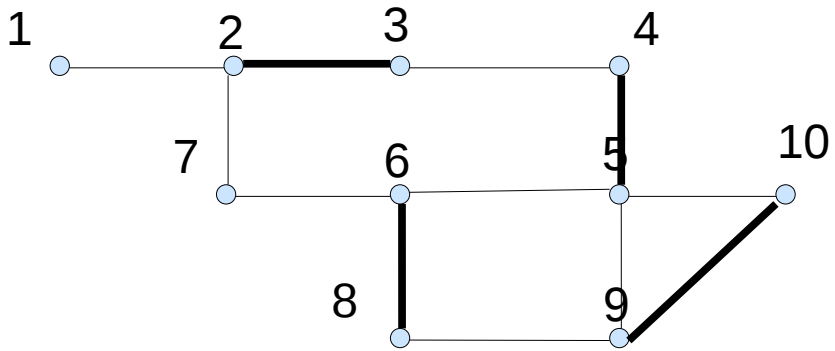
W artykule „Paths, Trees and Flowers” Jack Edmonds podał rozwiązanie tego problemu.

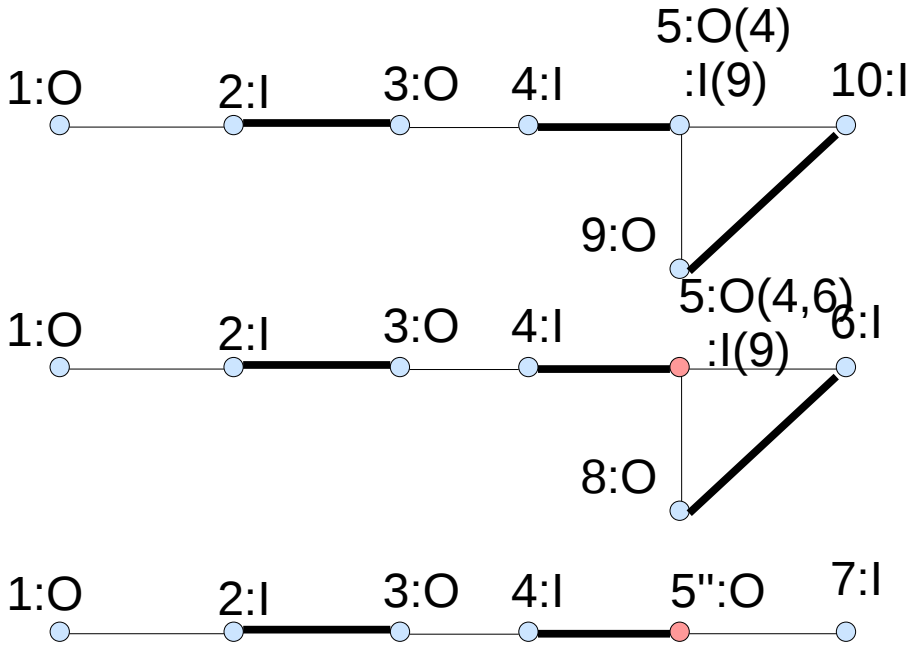
W celu znalezienie ścieżki rozszerzającej z zadanego wolnego węzła algorytm Edmondsa buduje drzewo ścieżek naprzemiennych. Korzeń i wszystkie wierzchołki będące w odległości parzystej od korzenia nazywane są wierzchołkami zewnętrznymi, pozostałe to wierzchołki wewnętrzne. Jeżeli w trakcie poszukiwania natrafimy na wewnętrzny wolny wierzchołek, to do tego wierzchołka można utworzyć ścieżkę powiększającą.

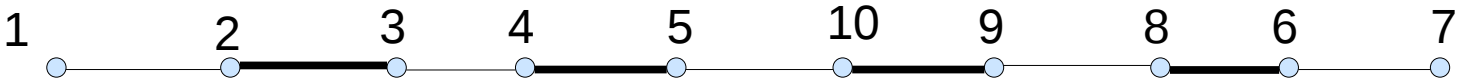
Budowa drzewa polega na wyszukiwaniu wierzchołków zewnętrznych. Każda „pogrubiona” krawędź (v,w), gdzie w jest wierzchołkiem, który jeszcze w drzewie nie istnieje, jest dodawana do drzewa. Wierzchołek w oznaczony jest jako wewnętrzny i kolejna krawędź (w,x), która jest incydentna z w jest dodawana do drzewa a x oznaczany jest jako zewnętrzny.

Jeżeli jednak znaleziona zostanie krawędź (v,w), taka że w jest oznaczony jako zewnętrzny to powstaje cykl.


I
O
O
I
O
O
I
I
O

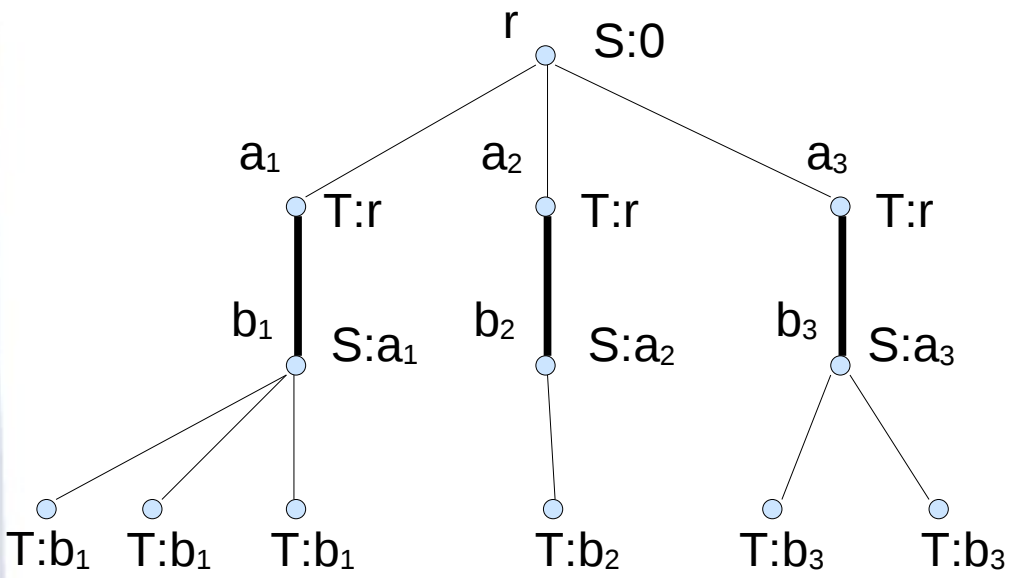

1
2
3
4
7
6
5
10
8
9

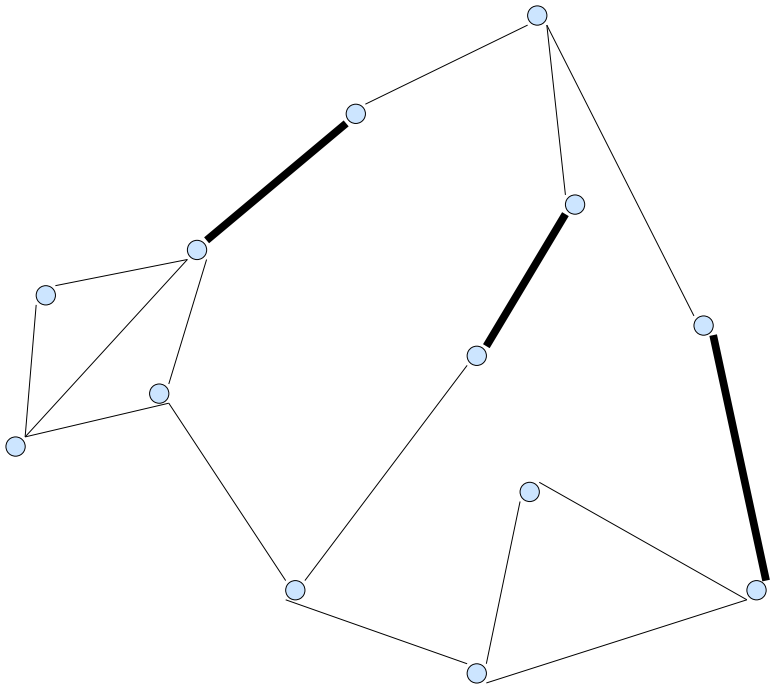

1:O
2:I
3:O
4:I
5:O(4)
:I(9)
10:I
9:O
1:O
2:I
3:O
4:I
5:O(4,6)
:I(9)
6:I
8:O
1:O
2:I
3:O
4:I
5":O
7:I


1
2
3
4
5
10
9
8
6
7

Wyznaczanie drogi powiększającej względem skojarzenia M

Rozpoczynamy od dowolnego wierzchołka wolnego r. Jeżeli istnieje wierzchołek wolny s sąsiadujący z r to krawędź (r,s) dołączam do bieżącego skojarzenia M. W przeciwnym razie rozpoczynamy budowę drzewa naprzemiennego.





Każdy z wierzchołków otrzymuje jedną z cech S lub T wraz z numerem wierzchołka z którego został ocechowany. Wierzchołek, który otrzymuje cechę staje się ocechowany, a jego dopiero co nadana cechę uznajemy za niezbadaną.

Procedura cechowania

1) Wybrać niezbadaną cechę, w zależności od typu cechy wykonać 2 lub 3.
2) Jeśli to cecha S to rozważyć wszystkie krawędzie wychodzące z tego węzła nie należące do skojarzenia M. Jeśli wierzchołek i jego potomek należą do tego samego kielicha pomiń rozważanie tej krawędzi. Jeśli potomek ma cechę S to sprawdzić czy obydwa węzły należą do tego samego drzewa. Jeżeli nie znaleziona została droga rozszerzająca. Jeśli należą do tego samego drzewa to przejdź do obróbki kielicha.
Jeśli wierzchołek nie zawiera żadnej cechy to oznaczyć go cechą T:rodzic.
3) Jeśli to cecha T to rozważyć krawędź należącą do skojarzenia M. Jeżeli potomek i dany węzeł należą do tego samego kielicha to pomiń rozważanie krawędzi. W przeciwnym razie Jeśli potomek jest T wierzchołkiem to sprawdzić czy potomek i rodzic należą do tego samego drzewa. Jeżeli nie to mamy drogę powiększająca, jeżeli nie to przejdź do obróbki kielicha.

Obróbka kielicha

Przypuśćmy, że drogi znalezione dla i-tego i j-tego węzła mają następującą postać

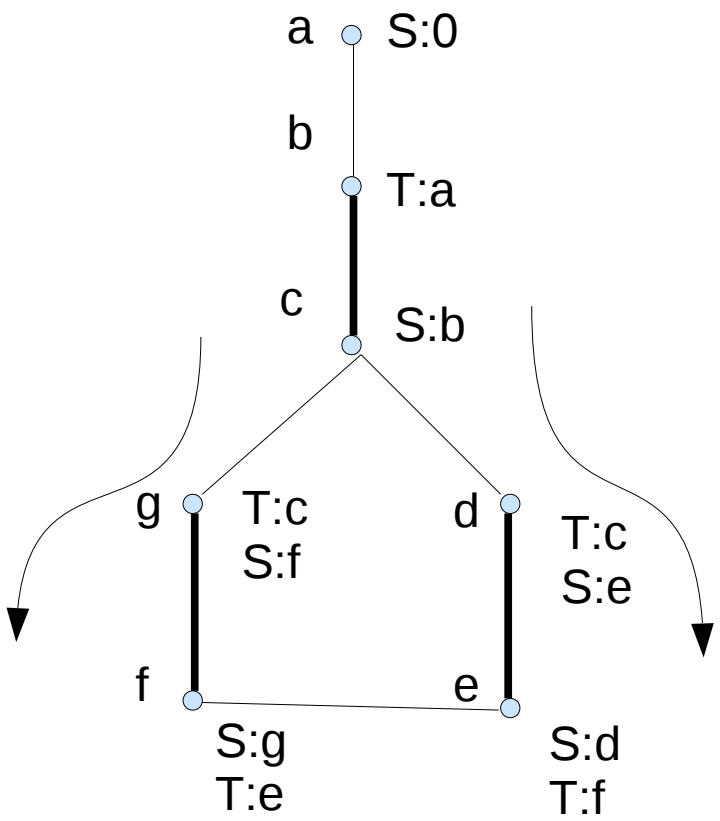
Pi=r, i1, i2, i3, i5, i6, i

Pj=r, i1, i2, i3, j4, j5, j

Ponieważ mamy kielich to i i j należą do tego samego drzewa w związku z tym ścieżki w w pewnym punkcie muszą się przeciąć i w tym przypadku jest to węzeł i3. Wszystkim węzłom kielicha {i5, i6, i, j, j5, j4} przypisana jest ta sama wartość b (pochodząca od węzła przecinającego czyli i3).

Każdy wierzchołek drzewa naprzemiennego, należący do kielicha (oprócz wierzchołka kielicha) jest końcem parzystej i nieparzystej drogi z korzenia, dlatego musi mieć obie cechy S i T.

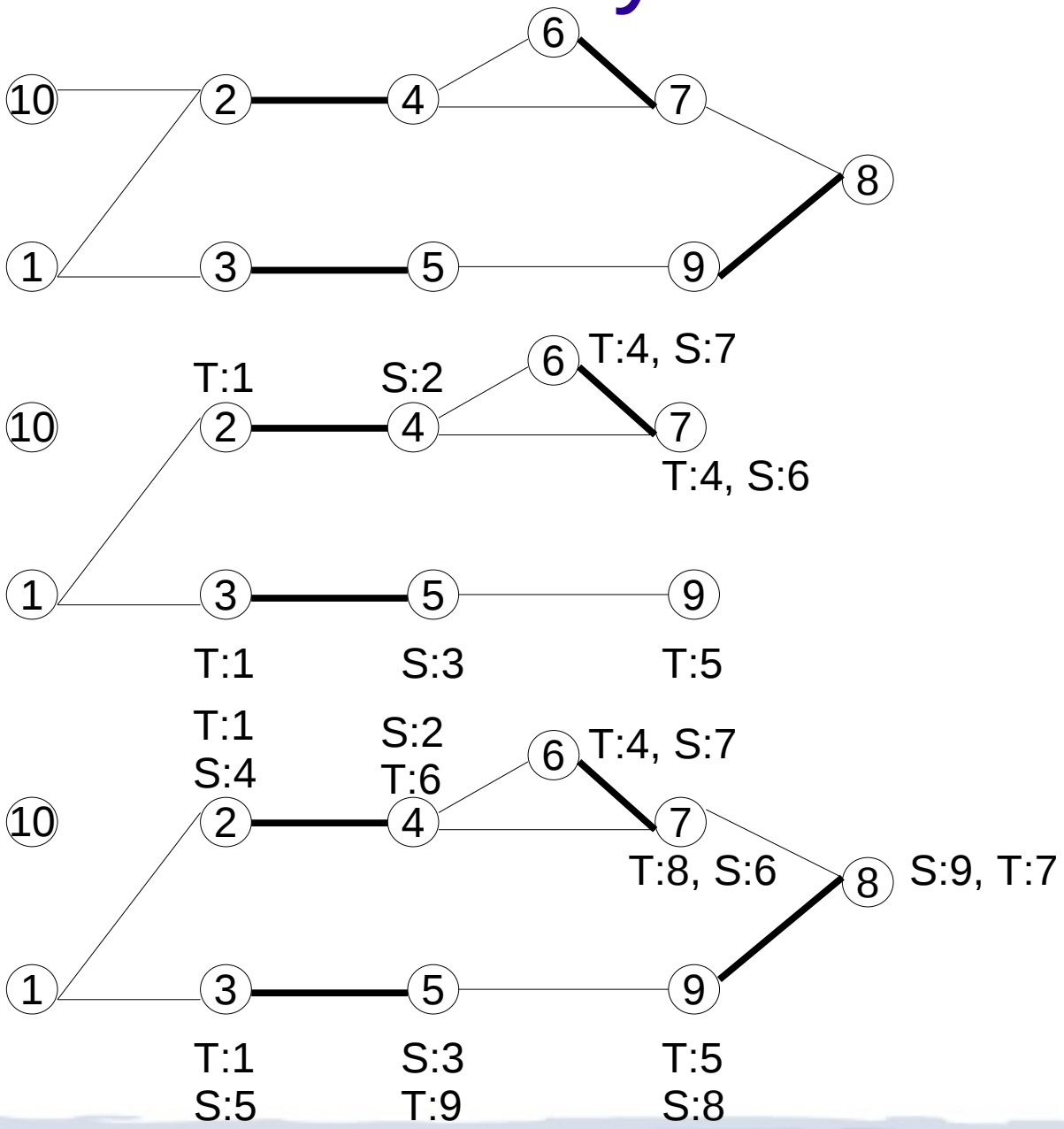
Przykład cechowania kielicha

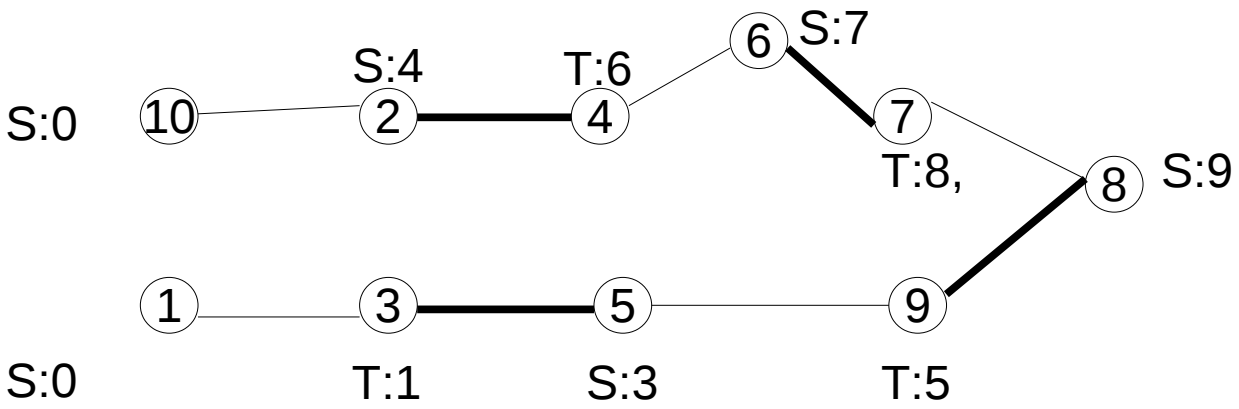


Algorytm Edmondsa (realizacja Lawlera)

```
1) Utwórz początkowe skojarzenie
2) while(expo<=1 lub brak niezbadanej cechy)
3)      Usuń wszystkie etykiety
4)      while(istnieje niezbadana cecha i nie znaleziono drogi powiększającej)
5)      {
6)                      weź cechę niezbadaną k i uznaj za zbadaną;
7)                      if(k jest typu S)
8)                              for(j przyległe do k i j i j nie są skojarzone)
9)                                      if(b(j)<>b(k))
10)                                             if(j ma cechę S)
11)                                                     Sprawdź korzenie dla j i k
12)                                                     if(ten sam korzeń)
13)                                                             Obróbka kielicha
14)                                                     else
15)                                                             Znajdź ścieżkę i zakończ
16)                                                     end;
17)                                             end;
18)                                     end;
19)                             else
20)                                     if(b(i)<>(b(j))
21)                                             if(j jest typu T)
22)                                                     Sprawdź korzenie dla j i k
23)                                             if(ten sam korzeń)
24)                                                     Obróbka kielicha
25)                                             else
26)                                                     Znajdź ścieżkę i zakończ
27)
```

Przykład


10
2
4
6
7
8
1
3
5
9
T:1
S:2
T:4, S:7
T:4, S:6
T:1
S:3
T:5
T:1
S:4
S:2
T:6
T:4, S:7
T:8, S:6
S:9, T:7
T:1
S:5
S:3
T:9
T:5
S:8


S:0
10
S:4
2
T:6
4
6
S:7
7
T:8,
8
S:9
1
3
5
9
S:0
T:1
S:3
T:5

Złożoność algorytmu Edmondsona

Znalezienie skojarzenie początkowego wymaga przejrzenia każdego wierzchołka i jego Wierzchołków sąsiadujących a więc złożoności jest O(|E|).
W grafie z n wierzchołkami możliwych jest nie więcej niż n/2 powiększeń skojarzenia Początkowego. Każde powiększenie skojarzenia jest poprzedzone cechowaniem wierzchołków, w którym każda z dwóch cech (S,T) w każdym wierzchołku jest badana dokładnie raz a każde takie badanie trwa O(n). Stąd całkowita złożoność cechowania wynosi $O(n^3)$

Metryczny problem komiwojażera (travelling salesman problem)

Komiwojażer ma odwiedzić dokładnie raz każdą z n wybranych miejscowości i powrócić do miejscowości z której rozpoczął swoją podróż. Czyli problem polega na znalezieniu najkrótszego cyklu długości n (cyklu Hamiltona) w N-wierzchołkowej sieci pełnej.

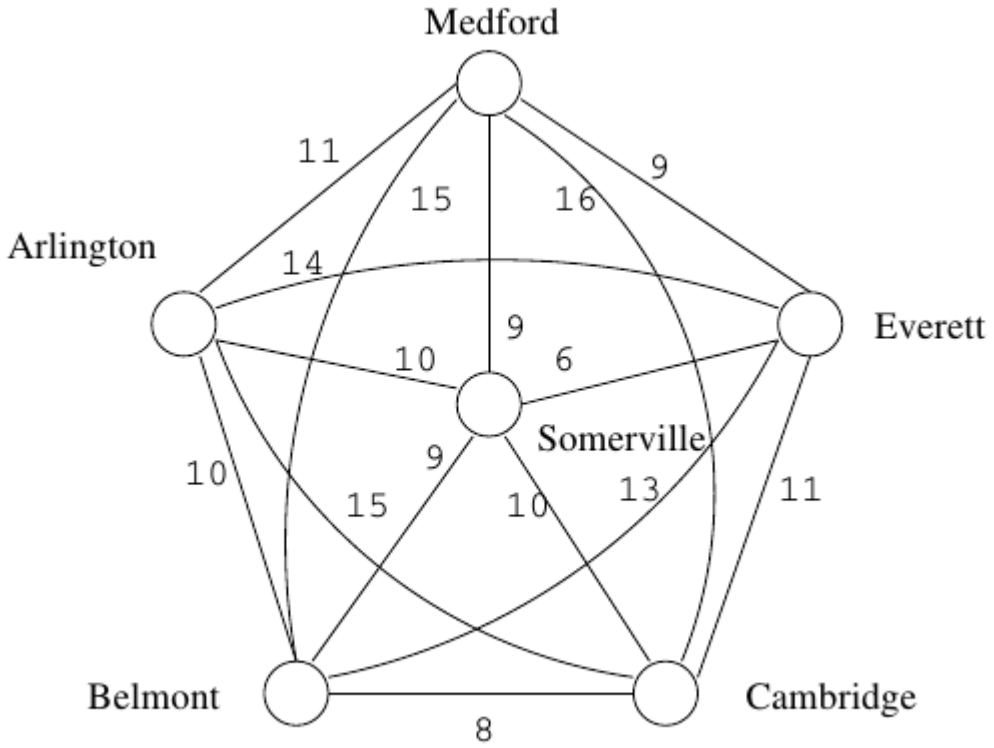


Figure 1: Example input to the TSP

Nierówność trójkąta

$$w_{ij} \leqslant w_{ik} + w_{jk}$$

Pomimo spełnionej nierówności trójkąta problem dalej jest NP trudny, ale można w czasie wielomianowym znaleźć się blisko optymalnego rozwiązania.

Rysunek z pracy www.cs.tufts.edu/comp/260/tspscribe-final.pdf

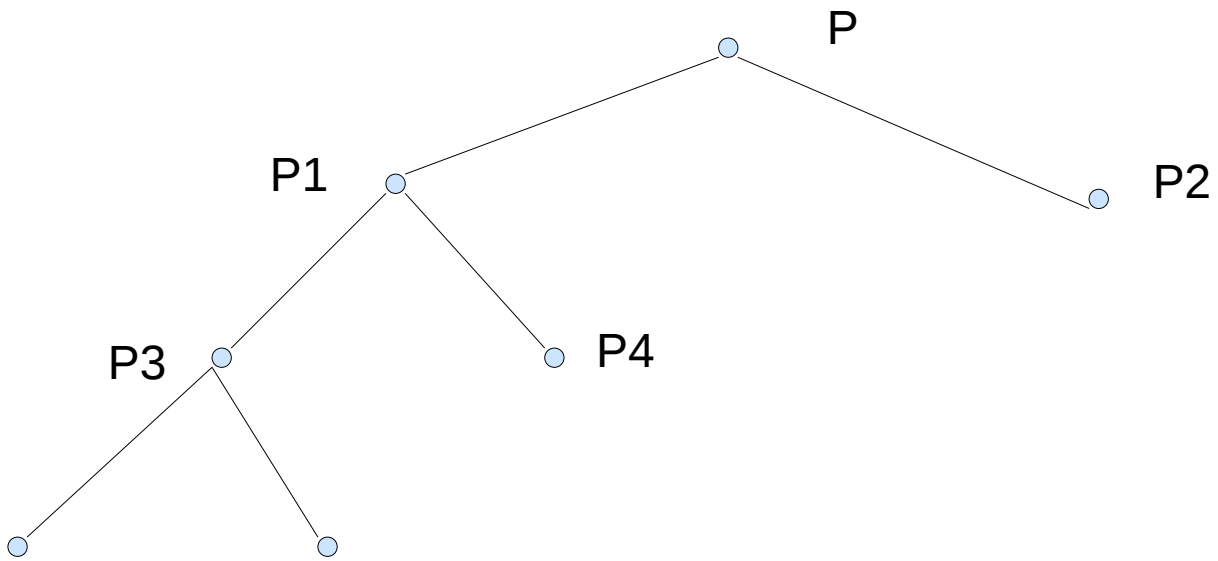
Jedną z najprostszych metod rozwiązania tego problemu jest przegląd całej przestrzeni rozwiązań. Dla n wierzchołków mamy n! permutacji i (n-1)! Z nich odpowiada różnym cyklom hamiltona, gdyż możemy ustalić którykolwiek z wierzchołków jako pierwszy a pozostałych n-1 wierzchołków można ustalić na (n-1)! Sposobów. Sieć symetryczna zawiera natomiast (n-1)!/2 różnych cykli.

Ponieważ takie podejście jest przy dużym n nie pozwala w sensownym czasie znaleźć rozwiązania. Istnieją dwa sposoby podejścia do tego problemu:

1) Zastosowanie metody podziału i ograniczeń, która znacząco zredukuje nieefektywność pełnego przeglądu

2) Zastosowanie metod przybliżonych (działających w czasie wielomianowym), które nie zawsze dają rozwiązanie optymalne, ale generują rozwiązania bliskie optymalnym.

Metoda podziału i ograniczeń (branch and bound)



Idea tego typu metod polega na podziale bieżącego rozwiązania na dwa lub więcej podzbiorów. W tym przypadku zbiór rozwiązań jest rozbijany na 2 podzbiory: jeden złożony z rozwiązań zawierających wyróżniony łuk (i,j), drugi pozostałe rozwiązania. Podział wykonywany jest zgodnie z zadaną (jakąś) heurystyką. Po wykonaniu podziału liczone są dolne ograniczenia wag rozwiązań w obu podzbiorach (LB). Następnie badany jest ten podzbiór, który ma mniejsze dolne ograniczenie.
Postępowanie to kontynuowane jest aż do otrzymania cyklu hamiltona.

Redukcja

Rozważmy następujący przykład

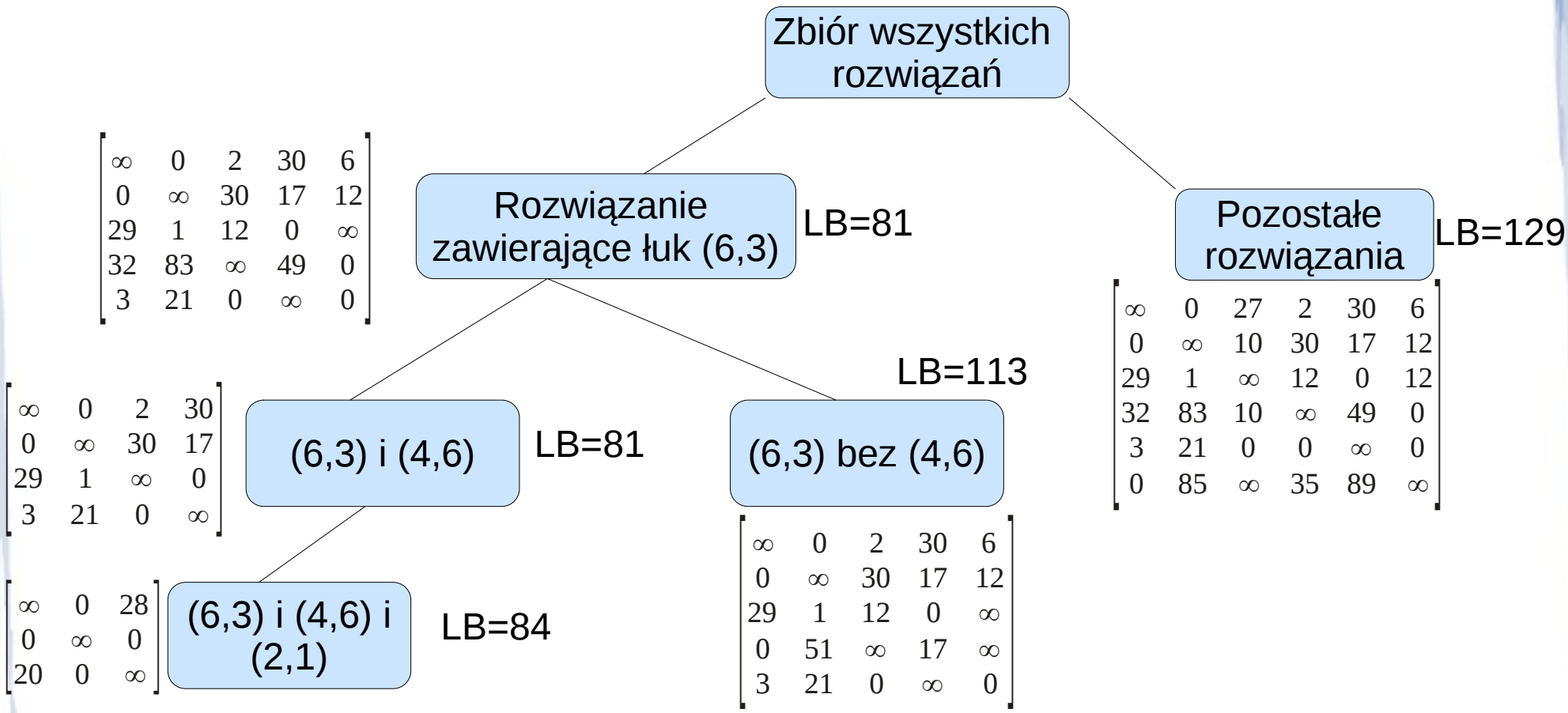
$$\begin{bmatrix} \infty & 3 & 93 & 13 & 33 & 9 \\ 4 & \infty & 77 & 42 & 21 & 16 \\ 45 & 17 & \infty & 36 & 16 & 28 \\ 39 & 90 & 80 & \infty & 56 & 7 \\ 28 & 46 & 88 & 33 & \infty & 25 \\ 3 & 88 & 18 & 46 & 92 & \infty \end{bmatrix} \rightarrow \begin{bmatrix} \infty & 0 & 75 & 2 & 30 & 6 \\ 0 & \infty & 58 & 30 & 17 & 12 \\ 29 & 1 & \infty & 12 & 0 & 12 \\ 32 & 83 & 58 & \infty & 49 & 0 \\ 3 & 21 & 48 & 0 & \infty & 0 \\ 0 & 85 & 0 & 35 & 89 & \infty \end{bmatrix}$$

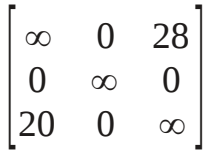
Jeśli odejmiemy stałą od wszystkich elementów wiersza lub kolumny, to waga każdego cyklu Hamiltona zmniejszy się dokładnie o tę stałą. Względny koszt wszystkich cykli pozostanie więc ten sam. Jeśli, po odjęciu stałych od wierszy lub kolumn każdy wiersz i każda kolumna zawierają dokładnie jedno zero, a pozostałe elementy są nieujemne to całkowita suma jest dolnym ograniczeniem długości jakiegokolwiek rozwiązania!

Dokonujemy redukcji poprzez odjęcie 3,4, 16, 7, 25 i 3 od poszczególnych wierszy oraz 15 i 8 od elementów kolumn 3 i 4. Suma liczb odjętych wynosi 81 i to jest dolne ograniczenie kosztu.

Podział zbioru rozwiązań

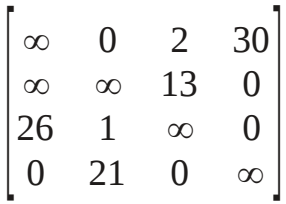
Każdy podział określony jest za pomocą ustalonego łuku, jeden podzbiór składa się z tych rozwiązań, które ten łuk zawiera drugi to pozostałe rozwiązania.





(6,3) i (4,6) i (2,1)

LB=84



(6,3) i (4,6) bez (2,1)

LB=101

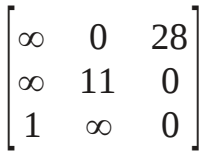
(6,3) i (4,6) i (2,1) i (1,4)

LB=84

Rozwiązanie (1,4,6,3,5,2,1)

LB=104

(6,3) i (4,6) i (5,1) bez(2,1)



Rozwiązanie (1,4,6,3,2,5,1)

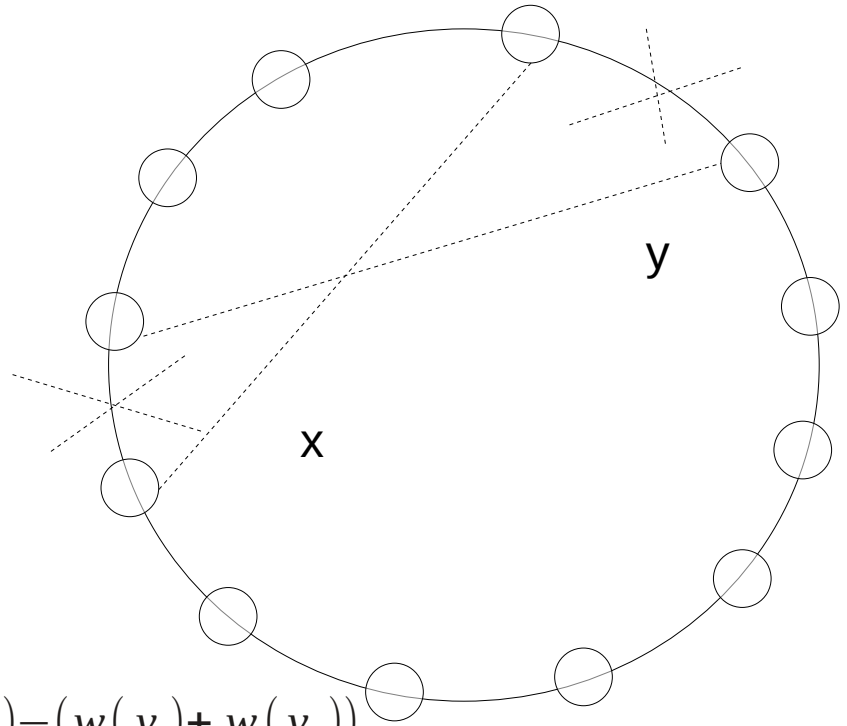
LB=104

Analiza złożoności

Algorytm podziału i ograniczeń jest właściwie metodą pełnego przeglądu przestrzeni rozwiązań. W najgorszym przypadku może się okazać, że przeszukiwana jest cała przestrzeń rozwiązań

Algorytm r-optymalny

Rozpoczynamy od dowolnego cyklu Hamiltona H i usuwamy z niego r łuków. Otrzymane r drogi łączymy r łukami w cykl Hamiltona H'. Cykl H i H' różnią się r łukami, pozostałe łuki są identyczne. Jeśli waga w(H) jest większa od wagi w(H') to zastępujemy cykl H cyklem H' i ponawiamy wymianę, w przeciwnym razie próbujemy wymienić inne r łuki. Wymianę kontynuujemy do momentu aż nie jest możliwe dalsze polepszanie wyniku



$$w(H)-w(H')=w(x_i)+w(x_j)-(w(y_i)+w(y_j))$$

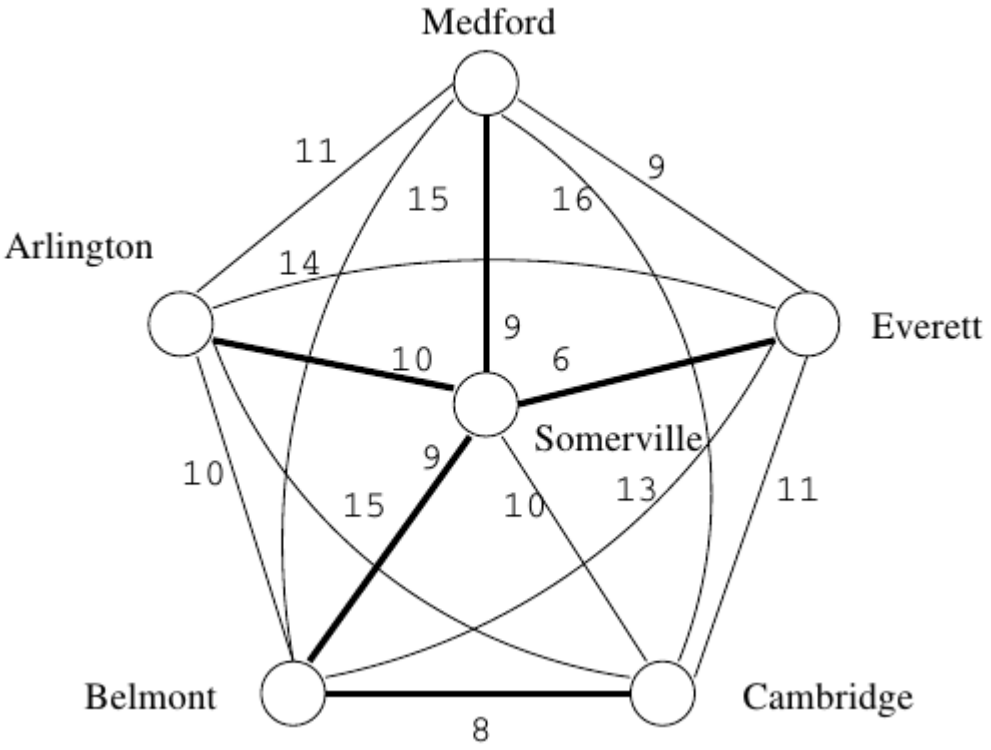
Im większa wartość r tym lepsze wyniki, ale czas obliczeń rośnie wraz ze wzrostem r ($O(n^r)$).

Algorytm oparty na MST

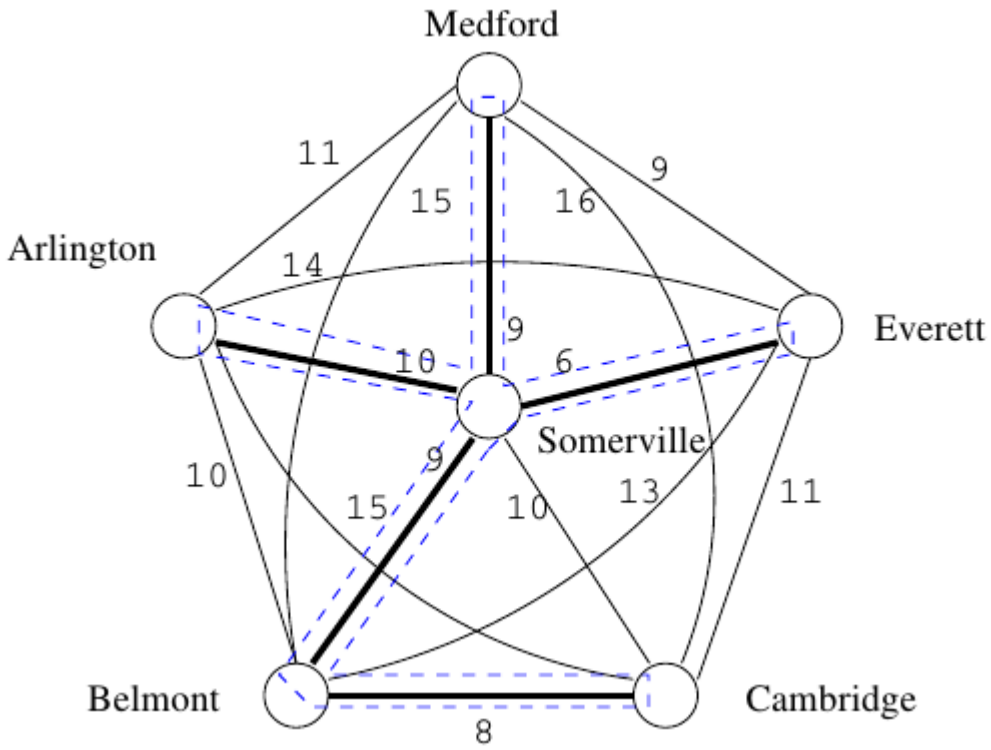
Suma wag otrzymanych z minimalnego drzewa rozpinającego (DM) jest mniejsza od sumy wag w optymalnym rozwiązaniu TSP (T)!
Usuńmy jedną krawędź z optymalnego rozwiązania T (usuwamy tą krawędź po usunięciu której otrzymamy drzewo rozpinające)

Ponieważ M to minimalne drzewo rozpinające to oczywiście

$$w(DM) \leqslant w(T-e) = w(T) - w(e) = OPT - w(e)$$



Mając MST pozostaje tylko z tego rozwiązania utworzyć cykl. W tym celu należy powielić każdą krawędź, a następnie przejść przez każdy węzeł dwukrotnie. Czyli Zaczynając od miast S i budując ścieżkę w głąb otrzymujemy następującą drogę: S,M,S,E,S,B,C,B,S,A.



W ten sposób nasze rozwiązanie jest co najwyżej 2 razy większe od rozwiązania MST

$$w(W) = 2 * w(DM) < 2 * OPT$$

Oczywiście trudno to nazwać rozwiązaniem ponieważ odwiedzamy dany węzeł 2 krotnie!

Żeby znaleźć dobre rozwiązanie musimy się pozbyć powtarzających się węzłów w taki sposób żeby nie zwiększyć wag.
Wykorzystując nierówność trójkąta możemy po prostu zostawić tylko pierwsze wystąpienie węzła a drugie opuścić, a więc rozwiązaniem jest S,M,E,B,C,A

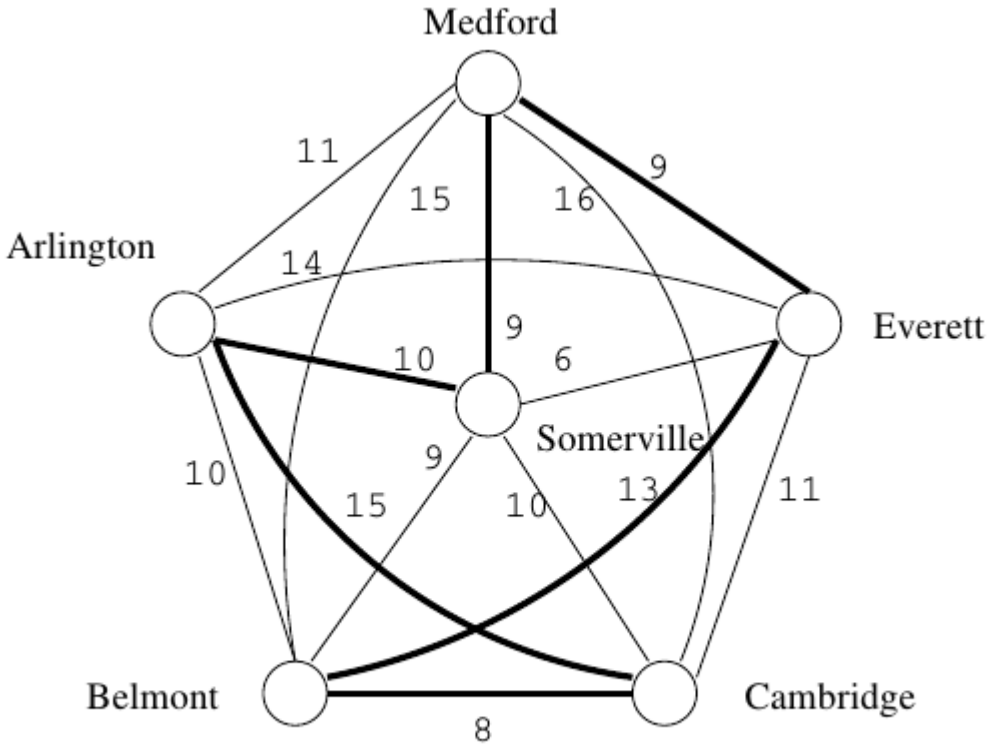


Figure 4: Solution with weight less than $2 * OPT$

Algorytm Christofidesa

- 1 Znaleźć minimalne drzewo rozpinające T
- 2 Wyznaczyć zbiór wierzchołków stopnia nieparzystego V^{odd} (G*)
- 3 Dla zbioru V^{odd} znaleźć minimalne skojarzenie M

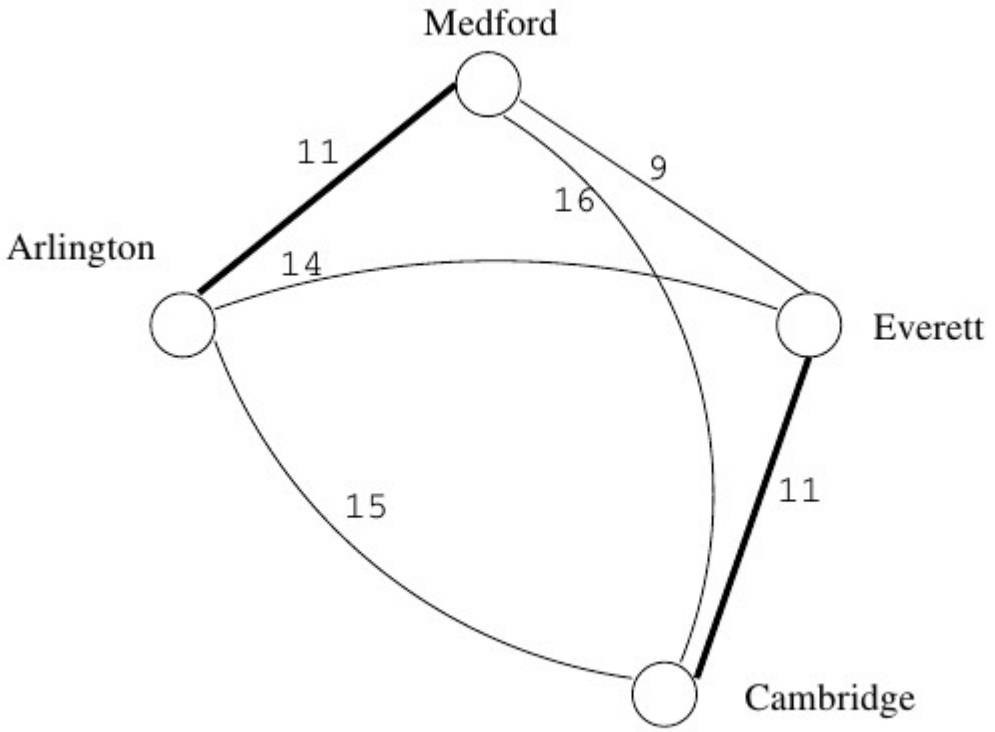


Figure 5: Minimal matching on vertices with odd degree in the MST

$$w(M) \leqslant 0.5\,OPT$$

Uzasadnienie

Optymalne rozwiązanie S* problemu TSP na grafie G* ma sumę wag równą co najwyżej OPT. Usuwanie wierzchołków nie może spowodować zwiększenia sumy wag. Ponieważ liczba węzłów w grafie G* jest parzysta to rozwiązanie S* może zostać podzielone na 2 skojarzenia przez ścieżkę naprzemienną. „Lżejsze” skojarzenie M1 ma wagi co najwyżej 0.5*w(S*)<=0.5*OPT

- Ponieważ M jest minimalnym skojarzeniem w(M)<=w(M1)<=0.5*OPT
- 4) Utworzyć podgraf rozpinający D=T∪M

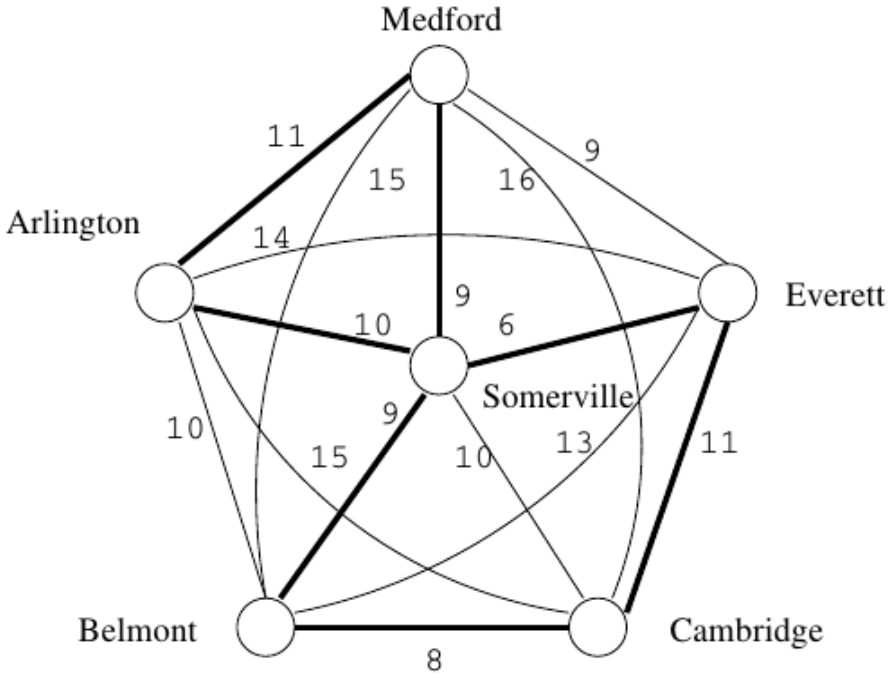


Figure 6: Union of MST and minimal matching

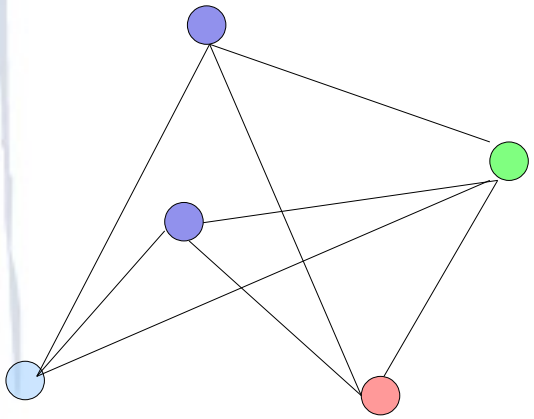
Podgraf ma tylko węzły o parzystym stopniu

$$w(T) + w(M) \leqslant 1.5 * OPT$$

- 5) Znaleźć cykl Eulera w grafie D
  6) Przekształcić cykl Eulera w cykl Hamiltona, poprzez eliminację powtarzających się węzłów

Kolorowanie grafów
Podstawowe pojęcia

Przypisanie kolorów wierzchołkom grafu G, po jednym kolorze dla każdego wierzchołka, w taki sposób aby sąsiednie wierzchołki otrzymały różne barwy nazywamy kolorowaniem Grafu (kolorowanie całkowite lub pełne). Pokolorowanie częściowe grafu polega na Przypisaniu kolorów ale niekoniecznie wszystkim wierzchołkom grafu.
Pokolorowanie k kolorami nazywamy k-pokolorowaniem. Graf jest k-barwny jeśli istnieje L-pokolorowanie, gdzie l<=k.
Dla grafu prostego najmniejszą wartość k dla której graf jest k-barwny nazywamy liczbą Chromatyczną grafu G i oznaczamy χ(G). Graf G jest k-chromatyczny, jeżeli χ(G)=k.

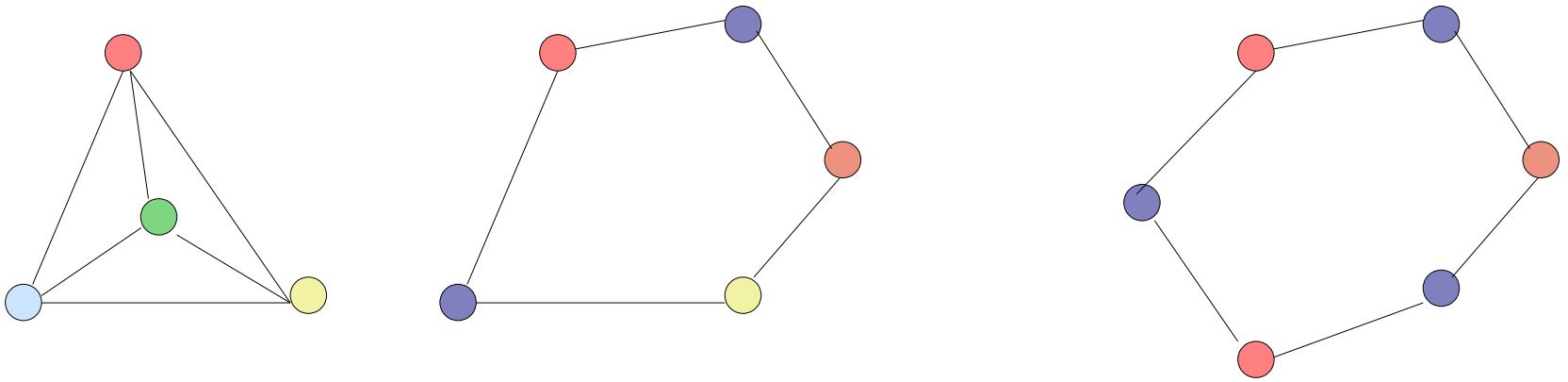


$\chi(G)=4$

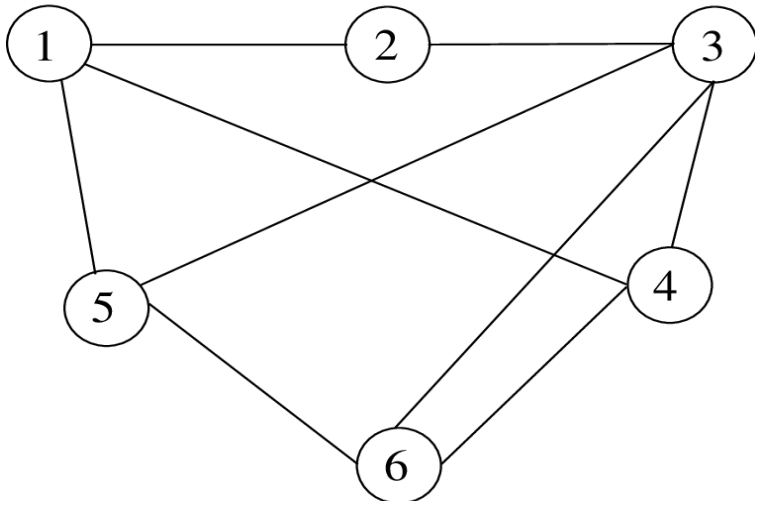
Liczba Chromatyczna

Dla grafu pełnego K_n $\chi(K_n)=n$.
Jeśli graf jest grafem cyklicznym C_n to

$$\chi(C_n)=\begin{cases} 2 \, dla \, n \, parzystego \\ 3 \, dla \, n \, nieparzystego \end{cases}$$



Zbiór wierzchołków tego samego koloru nazywamy klasą barwności.
Zbiory wierzchołków, w których żadne 2 wierzchołki nie sąsiadują ze sobą nazywamy niezależnymi.



Zbiory niezależne: {1,6}, {2,5,4}, {3,1}, {4,2,5}, {6,2}
Liczbę

$$\alpha(G) = \max_i |S_i|$$

gdzie S_i są wszystkimi zbiorami niezależnymi grafu G, nazywamy liczbą niezależności grafu. Zbiór S*, który zawiera największą liczbę elementów nazywamy maksymalnym zbiorem niezależnym (MIS).

Twierdzenie Brooksa

Jeżeli graf G jest grafem prostym, w którym stopień grafu jest oznaczony przez Δ(G) to

$$\chi(G) \leqslant \Delta(G) + 1$$

Znak równości zachodzi tylko w dwóch przypadkach:

1) gdy graf jest grafem pełnym
2) gdy graf jest grafem cyklicznym o nieparzystej liczbie wierzchołków

Metoda zbiorów niezależnych

Wyznaczenie k-pokolorowania grafu G jest równoważne z podzieleniem zbioru Wierzchołków grafu G na k zbiorów niezależnych $V_1,...V_k$ ($V_i \cap V_j = \emptyset$ $i \neq j$). Podział taki Nazywa się k-barwnym podziałem zbioru V.

Metoda zbiorów niezależnych znajdowania k-pokolorowania grafu G polega na pomalowaniu najpierw wierzchołków kolorem 1 tzn. znalezienie zbioru V_1, następnie V_2 itd.

Twierdzenie. Dla każdego k-barwnego podziału $\{V_1,...,V_k\}$ grafu G istnieje l-barwny podział $\{V_{1'},..,V_{l'}\}$ taki, że l<=k i przynajmniej 1 zbiór w tym drugim podziale jest maksymalnym zbiorem niezależnym (czyli nie istnieje zbiór zawierający go w sposób właściwy, który jest niezależny).

Dowód. Niech $\{V_1,...,V_k\}$ będzie k-barwnym podziałem zbioru V. Jeżeli V_1 nie jest niezależnym zbiorem maksymalnym grafu G, to może on być powiększony do takiego zbioru $V_{1'}$ poprzez dołączenie wierzchołków z innych podzbiorów. Każdy zbiór V_i-V1', dla i=2,3,..k, Jest niezależny, ale niektóre z nich mogą być puste. V1' i Vi'=Vi-V1': Vi'≠Ø, i=2,...k. Tworzą l barwny podział grafu G, l<=k, a V1' jest maksymalnym zbiorem niezależnym.

Niech

$$G_W = (W, E')\, W \subseteq V$$

Na podstawie twierdzenia, w każdym optymalnym pokolorowaniu grafu G albo jedna klasa barwności jest maksymalnym zbiorem niezależnym, albo można ją powiększyć do takiego zbioru.

$$\chi(G) = \chi(G_{V-W}) + 1$$

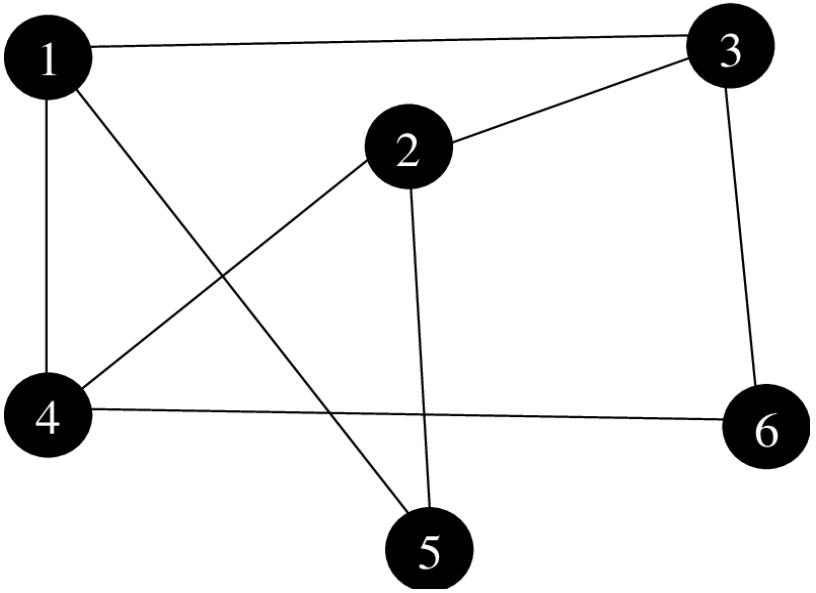
Algorytm Christofidesa

Jeżeli G jest k-chromatyczny to to można go pomalować k kolorami, najpierw malując kolorem 1 maksymalny zbiór niezależny W1 grafu G następnie, kolorem 2 maksymalny zbiór niezależny podgrafu $G|_{V-W1}$ itd.
G oznacza rodzinę wszystkich maksymalnych l-chromatycznych podgrafów grafu G

```
Begin
    K←0;
    G0←{∅}
    While Gk nie zawiera grafu H takiego że V(H)=V(G) do begin
        F←∅
        For każdego H∈Gk do
            For każdego MIS W w G|V(G)-V(H) do
                F←F∪{G|V(H)∪W}
        k←k+1
        Gk←maksymalne grafy w zbiorze F
    End;
    /*k jest liczba chromatyczną grafu G */
End
```

Złożoność algorytmu $O(mn2.445^n)$ gdzie n liczba wierzchołków a m liczba Krawędzi w grafie.

Przykład



G1={{1,2,6},{3,4,5},{5,6}}
G2={{1,2,6}∪{3,4,5}…}

Algorytmy przybliżone

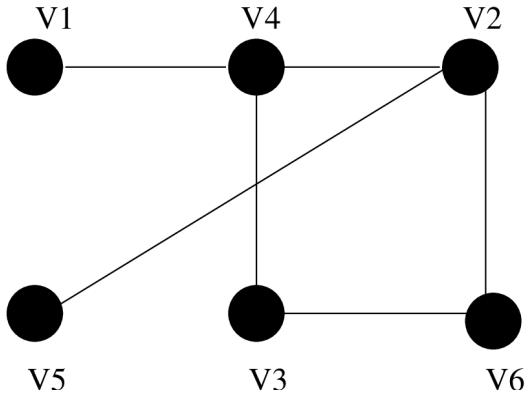
Algorytm największych zbiorów niezależnych
Begin
    U←V
    K←0
    While U≠∅ do begin
        k←k+1
        Znaleźć najliczniejszy zbiór niezależny W w podgrafie $G|_U$
        For każdego u∈W do f(u)←k;
        U←U-W
    End
    /* k jest liczbą kolorów użytych do pokolorowania grafu*/
end

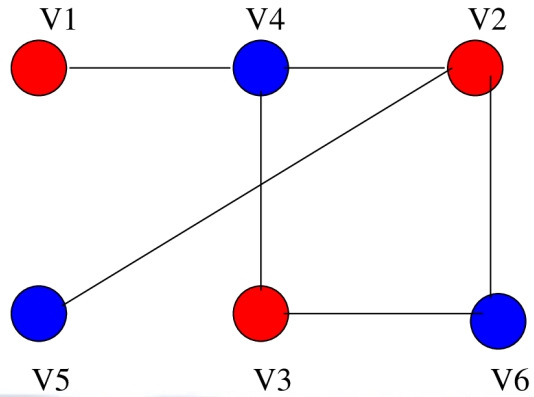
Algorytm przybliżonego najliczniejszego podzbioru niezależnego

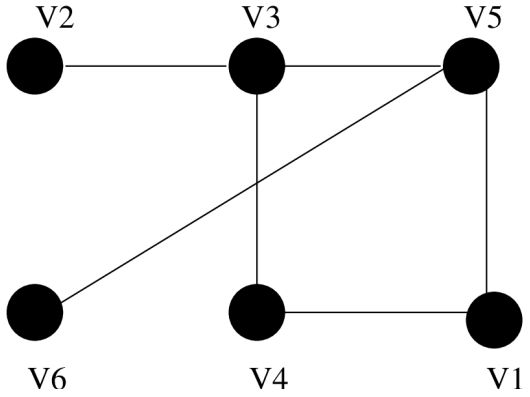
```
AMmISet(U,W)
Begin
    U1←U
    W←∅
    While U1≠∅ do begin
        Znaleźć wierzchołek u o minimalnym stopniu w G|U1
        W←W∪{u}
        U1←U1-{u}-{v∈U1:(v,u)∈E(G)}
end
```

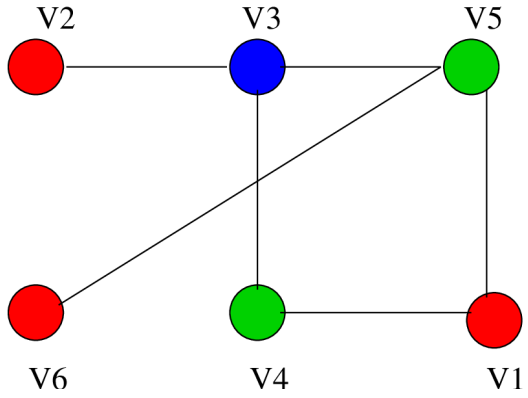
Algorytm zachłanny (sekwencyjny)

Kolorowanie(G)
1 przypisz kolor 1 do wierzchołka v_1
2 for k ← 2 to n
3 do
4 dla wierzchołka v_k użyj pierwszego dostępnego koloru, który nie był
5 wykorzystany do pokolorowania wierzchołka sąsiedniego









Można wyznaczyć górne oszacowanie liczby kolorów użytych przez algorytm sekwencyjny, do pomalowania wierzchołków grafu G w kolejności $v_1,..,v_n$. Każdy wierzchołek v_i może być pomalowany kolorem i, więc $f(v_i)<=i$, z drugiej strony przynajmniej jeden z pierwszych $|S(v_i)|+1$ kolorów może być przydzielony wierzchołkowi v_i.
Więc

$$f(v_i) \leqslant min\{i, |S(v_i)| + 1\}$$

Stąd

$$\chi(G) \leqslant \max_{1 \leqslant i \leqslant n} min\{i, |S(v_i)| + 1\}$$

Jedną z pierwszych wersji algorytmu sekwencyjnego zaproponowali Welsh i Powell, którzy jako pierwsi porządkowali wierzchołki w kolejności nierosnących stopni. Takie Uporządkowanie nazywa się pierwszym-największym.

Algorytm sekwencyjny z nawrotami

Niech $v_1,...,v_n$ będzie dowolnym uporządkowaniem wierzchołków grafu. Początkowo, wierzchołkowi v_1 przydzielamy kolor 1, po czym po kolei malowane są pozostałe wierzchołki z zastosowaniem następującej reguły:

Przyjmując, że wierzchołki $v_1,...,v_i$ są już pomalowane, znaleźć zbiór U_i, kolorów dopuszczalnych dla v_i i przydzielić mu kolor min{U_i}.

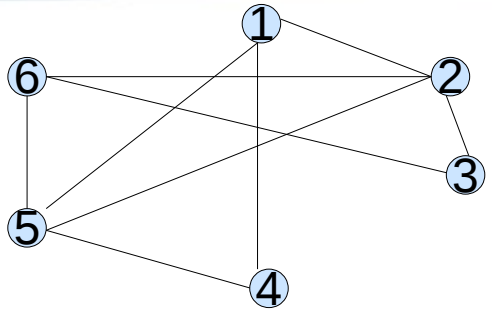
Twierdzenie.

Niech f:($v_1,...,v_{i-1}$}->{1,2,...,*l*}. będzie pokolorowaniem *i*-1 wierzchołków grafu *G*, wykorzystującym dokładnie l kolorów. Jeśli mają być generowane wyłącznie pokolorowania nieredundantne, to kolor wierzchołka v_i powinien spełniać nierówność $1 \leq f(v_i) \leq l+1$

Niech l_{i-1} oznacza maksymalny numer koloru przydzielonego wierzchołkom $v_1,..,v_{i-1}$.
Każdy kolor j, należący do U_i, musi spełniać następujące warunki:

1) $j \leq l_{i-1}+1$
2) $j \leq \min\{i,|S_i|+1\}$
3) j nie jest kolorem żadnego wierzchołka v_h ($1 \leq h \leq i-1$)będącego sąsiadem Wierzchołka v_i

```
Begin
    l←1;k←1;f(v1)←1;q←n+1;increase←true;
    Repeat
        If increase then begin
            l_k←l;k←k+1;
            Znaleźć U_k;
        End;
        If Uk=∅ the begin
            k←k-1;l←l_k;
            increase←false
        End
        Else begin
            j←najmniejsza liczba w U_k;
            U_k←U_k-{j};
            f(v_k)←j;
            If j>l then l←l+1;
            If k<n then increase←true
            Else begin
                Zapamiętać aktualne rozwiązanie;
                Znaleźć najmniejsze i takie, że f(vi)=l;
                Usunąć kolory l,l+1,...,q-1 z U_1, U_2, ...,U_i-1;
                q←l;l←q-1;k←i-1;
                increase←false;
            End
        End
    Until (k=1) or (q=lowerbound)
end
```



1

2

3

4

5

6

1

U2={2} 2

U3={1,3} 1 3 U3={3}

U4={2,3} 2 3 U4={3} 2 U4={2,3}

U5={3} 3 U5=0 3 U5={3}

U6={4} 4 U6={1}

q<−4

q<−3

l=1;k=1;f(1)=1;q=6+1

l_1=1;k=2;U_2={2}
j=2;U_2={}
f(2)=2;l=2
l_2=2; k=3;
U_3={1,3}
j=1;U_3={3}
f(3)=1
l_3=2;k=4;
U_4={2,3};
j=2;U_4={3};
f(4)=2
l_4=2;k=5
U_5={3}
j=3;U_5={};f(5)=3;l=3;
l_5=3; k=6;
U_6={4};
j=4;U_6={};f(6)=4;l=4;

f(6)=4; l=4;
q=4;l=3;k=5;
U_5={};k=4;l=2;
U_4={3}
j=3;U_4={};f(4)=3;
l=3;
l_4=2;k=5;
k=4;l=3;
k=3;l=2;
j=3;U_3={};f(3)=3;
l=3;
l_3=3;k=4;
U_4={2,3}
j=2;U_4={3}
f(4)=2;
l_4=2;k=5
U_5={3};
j=3;U_5={}
f(5)=3;
l_5=3;k=6
U_6={1}
j=1;U_6={};f(6)=1;

Zastosowanie kolorowania grafów

Załóżmy, że chcemy ustalić terminy egzaminów (A B C D) dla 5 studentów (P Q R S T). W danym dniu egzaminy przeprowadzane są w tym samym czasie. W związku z tym zależy nam na tym żeby uniknąć sytuacji w której student ma wyznaczone dwa egzaminy w jednym dniu. Pytanie jest następujące jaka jest minimalna liczba dni potrzebna do przeprowadzenia wszystkich egzaminów.

Założenia:

- Student P jest zarejestrowany na egzamin A i B
- Student Q jest zarejestrowany na egzamin C i B
- Student R jest zarejestrowany na egzamin C i D
- Student S jest zarejestrowany na egzamin A i D
- Student T jest zarejestrowany na egzamin C i B

	A	B	C	D
P	*	*		
Q		*	*	
R			*	*
S	*			*
T		*	*	

Macierz konfliktów

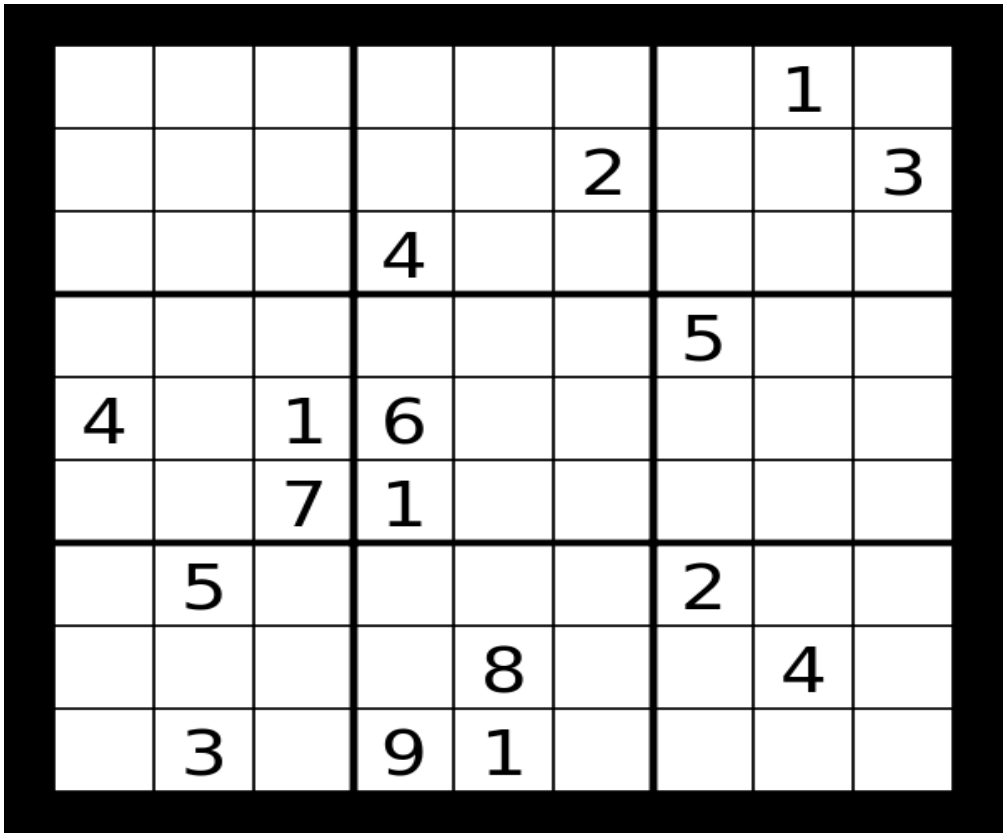
	A	B	C	D
A		*		*
B	*		*	
C		*		*
D	*		*	

Gwiazdka na pozycji *ij* oznacza, że co najmniej jeden student jest zarejestrowany na *i* i *j* egzamin.

Teraz wystarczy z macierzy konfliktów utworzyć graf!

Liczba kolorów użyta do pokolorowania grafu określi wymaganą liczbę dni.

Sudoku



							1	
					2			3
			4					
						5		
4		1	6					
		7	1					
	5					2		
				8			4	
	3		9	1				

Liczba możliwości ponumerowania „czystego” sudoku (3x3)
6670903752021072936960 = c. 6.7×10^{21}

Reprezentacja w postaci grafu

Graf musi mieć 81 wierzchołków, wierzchołki powinny być poetykietowane uporządkowaną parą (x,y) gdzie x i y są liczbami z zakresu 1-9. W związku z tym dwa wierzchołki (x,y) i (x',y') są połączone krawędzią tylko wtedy gdy:

x=x' (węzły są w tej samej kolumnie)
y=y' (węzły są w tym samym wierszu)
[x/3]=[x'/3] i [y/3]=[y'/3] (węzły znajdują się w tym samym kwadracie o rozmiarze 3x3)

Tak otrzymany graf należy pokolorować!

Inne przykłady

Szeregowanie lotów samolotów
Załóżmy, że mamy k samolotów i muszą być one wykorzystane w n lotach, i-ty lot trwa (ai,bi). Jeżeli dwa loty się nakładają to nie można wykorzystać do nich tego samego samolotu.
Problem ten przedstawiany jest w postaci grafu, którego węzły odpowiadają lotom. Dwa węzły łączą się ze sobą jeżeli czasy lotów się nakładają natomiast kolory reprezentują samoloty.

Przypisywanie częstotliwości (telefony komórkowe)
Załóżmy, że mamy pewna liczbę stacji radiowych, które identyfikowane są przez współrzędne x i y. Musimy przydzielić stacjom częstotliwości ale z powodu interferencji stacjom, które są blisko siebie musimy przydzielić inne częstotliwości.

Szeregowanie zadań

Jest to jedna z najważniejszych dziedzin informatyki i zajmuje się układaniem Harmonogramów zadań na maszynach (procesorach obrabiarkach itp.)
Zastosowania praktyczne:

- Planowanie projektów
- Organizacja pracy
- Plany zajęć szkolnych, spotkań itp.
- Przetwarzanie procesów w wielozadaniowych systemach operacyjnych
- Organizacja obliczeń rozproszonych

Sześć zadań o czasach wykonania p1,...,p6=4,5,1,6,1,7 należy uszeregować na 3 maszynach tak aby zakończyły się jak najwcześniej

Diagram Gantta

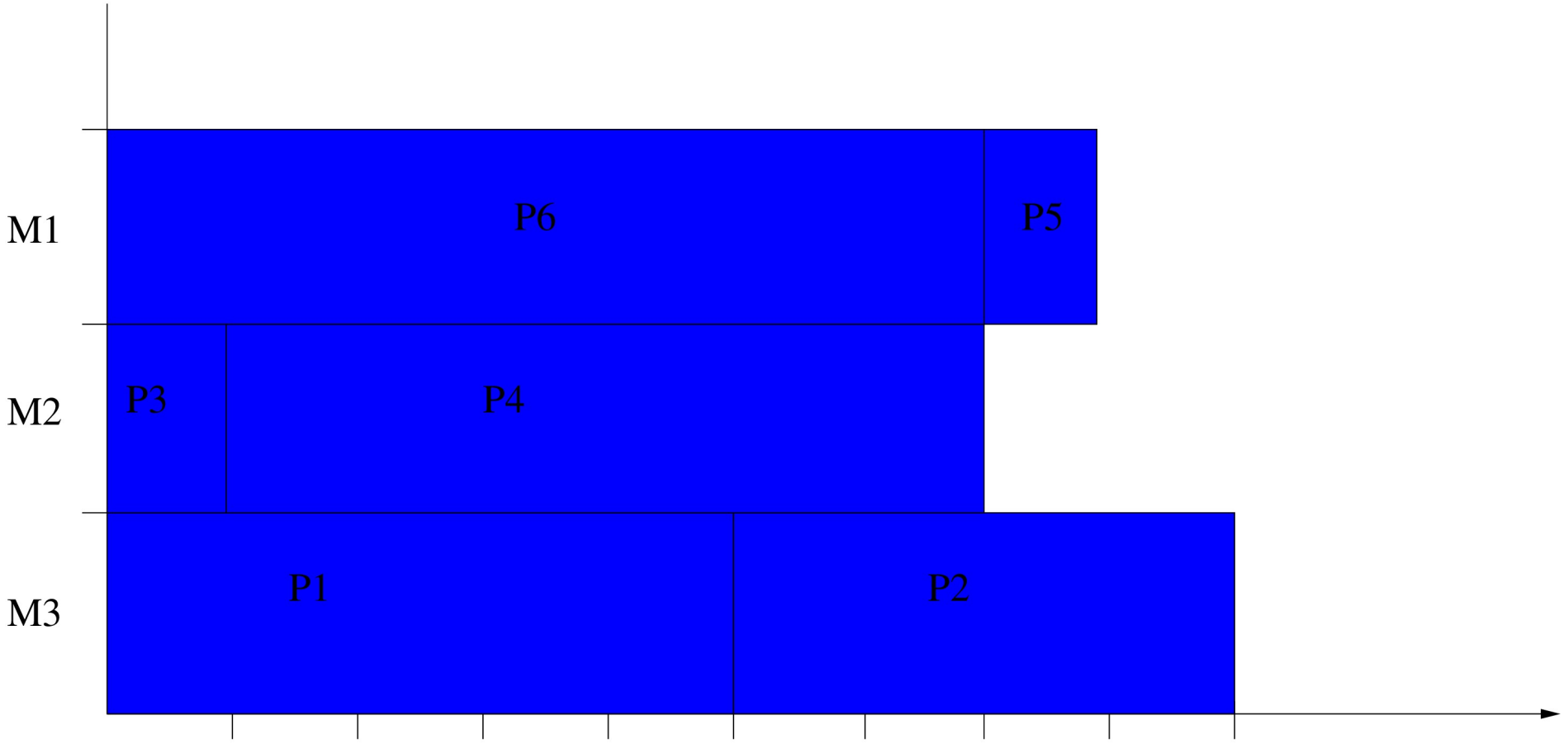
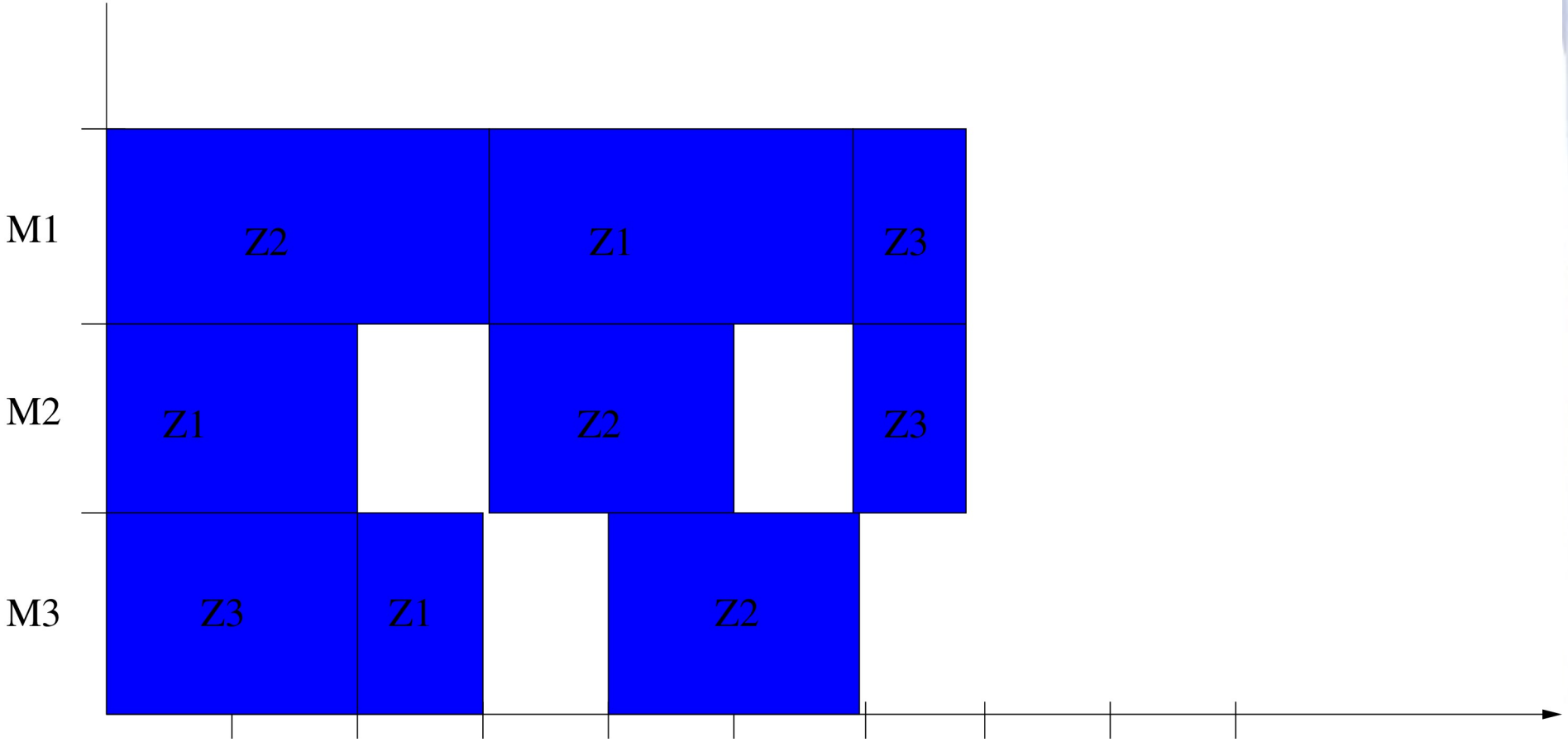


Diagram Gantta

Plan zajęć szkolnych

	M1	M2	M3
Z1	3	2	1
Z2	3	2	2
Z3	1	1	2



Optymalizacja projektu

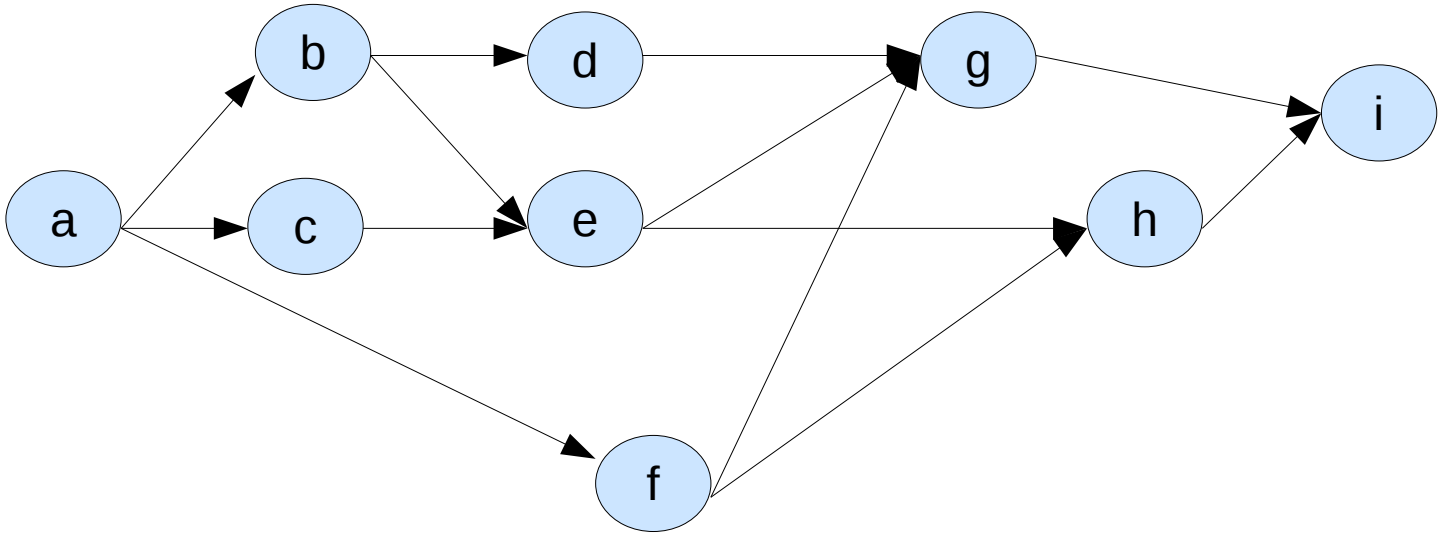
Optymalizacja przedsięwzięcia polega na:

- wyodrębnieniu i zestawieniu wchodzących w jego skład czynności
- ocenie parametrów poszczególnych czynności i zdarzeń
- konstrukcji sieci zależności technologicznych
- wyznaczeniu podstawowych charakterystyk sieci dotyczących poszczególnych czynności, zdarzeń i całego projektu
- wyznaczeniu ścieżki krytycznej

Szeregowanie sieciowe (metoda ścieżki krytycznej CPM)

Projekt składa się z 2 elementów: czynności i ograniczeń kolejnościowych nałożonych na te czynności. Czynności scharakteryzowane są czasami trwania, a relacje między czynnościami tworzą ograniczenia kolejnościowe.
Występują 2 podstawowe metody reprezentacji projektu przez sieć

- Reprezentacja czynności w węźle (AN – activity on node) zwana siecią czynności

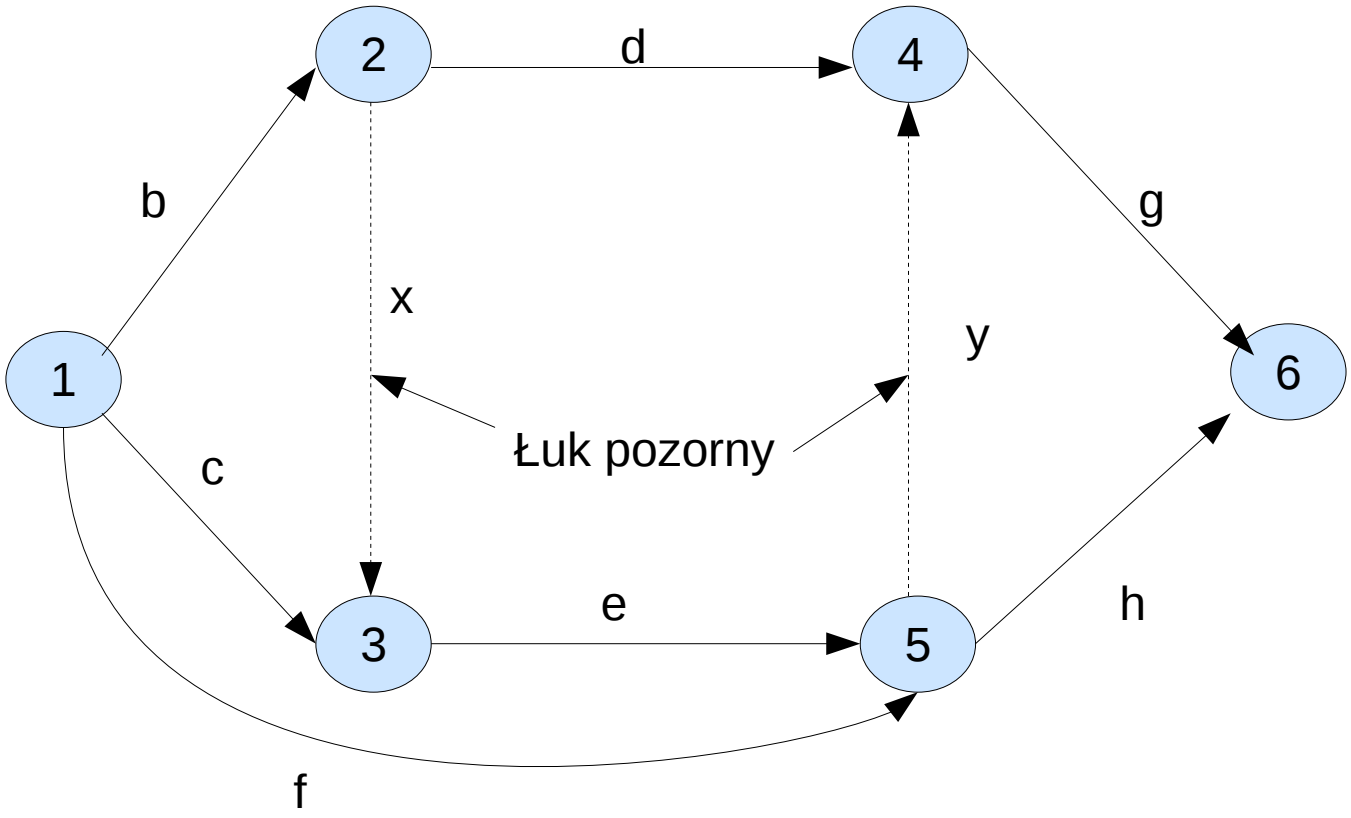


- Reprezentacja czynności na łuku (AA – activity on arc). Węzły noszą nazwę zdarzeń.

Sieć czynności

To graf mający następujące własności

- Jest unigrafem (czyli między dowolną parą wierzchołków występuje co najwyżej jeden łuk)
- spójny
- skierowany
- Acykliczny
- Ma jedno zdarzenie początkowe i jedno zdarzenie końcowe



Łuki pozorne nie reprezentują czynności oznaczają jedynie zależności kolejnościowe. Sieć zdarzeń ma mniej węzłów i łuków niż sieć czynności (nawet jeśli włączy się łuki pozorne). Niestety nie istnieje efektywny algorytm budowania sieci zdarzeń o najmniejszej liczbie czynności pozornych (problem NP-zupełny).

Analiza czasowa

Niech G=(W,B,p) będzie siecią z nieujemną funkcją p zdefiniowaną na zbiorze łuków B. p_{ij} dla (i,j)∈B jest czasem trwania czynności (i,j), jeżeli czynność (i,j) jest pozorna to $p_{ij}=0$

Pytania:

- Ile czasu będzie wymagać ukończenie projektu
- Kiedy poszczególne czynności mogą być rozpoczęte
- Jak długo jakaś czynność może być opóźniana nie powodując opóźnienia w realizacji projektu

Najkrótszy czas trwania projektu jest równy długości najdłuższej drogi ze źródła do odpływu

Ponieważ sieć jest acykliczna, to można ponumerować węzły w porządku topologicznym, To oznacza, że do przeanalizowania zdarzenia i wystarczy rozważyć zdarzenia j takie,że j<i

Niech π_i^e oznacza najwcześniejszy możliwy moment czasu, w którym zdarzenie i może nastąpić, jest to oczywiście długość najdłuższej drogi z 1 do i

Ścieżka krytyczna

Czas wystąpienia ostatniego zdarzenia π_n^e odpowiada najwcześniejszej realizacji całego projektu. Najdłuższa droga w G nosi nazwę **ścieżki krytycznej**. Czynności jej nazywa się **krytycznymi** ponieważ opóźnienie którejkolwiek z nich spowoduje opóźnienie zakończenia całego projektu.

Czynności, które nie leżą na ścieżce krytycznej mają pewną swobodę w realizacji. Dla takich czynności można znaleźć π_i^l tj. najpóźniejszy możliwy termin, w którym zdarzenie i mogłoby wystąpić bez opóźnienia zakończenia projektu.

Korzystając z czasów trwania p_{ij} oraz π_n^e i π_i^l dla każdej czynności (i,j) można wyliczyć:

- Najwcześniejszy czas rozpoczęcia $ES_{ij}=\pi_i^e$
- Najwcześniejszy czas zakończenia $Ef_{ij}=ES_{ij}+p_{ij}=\pi_i^e+p_{ij}$
- Najpóźniejszy czas zakończenia $LF_{ij}=\pi_j^l$
- Najpóźniejszy czas rozpoczęcia $LS_{ij}=LF_{ij}-p_{ij}=\pi_j^l-p_{ij}$
- Rezerwę czasową czynności $sij=LFij-EFij=LSij-ESij=\pi_j^l-\pi_i^e-p_{ij}$

Numerowanie wierzchołków

1) Przydziel wierzchołkowi swobodnemu (do którego nie dochodzą żadne łuki) numer i=1

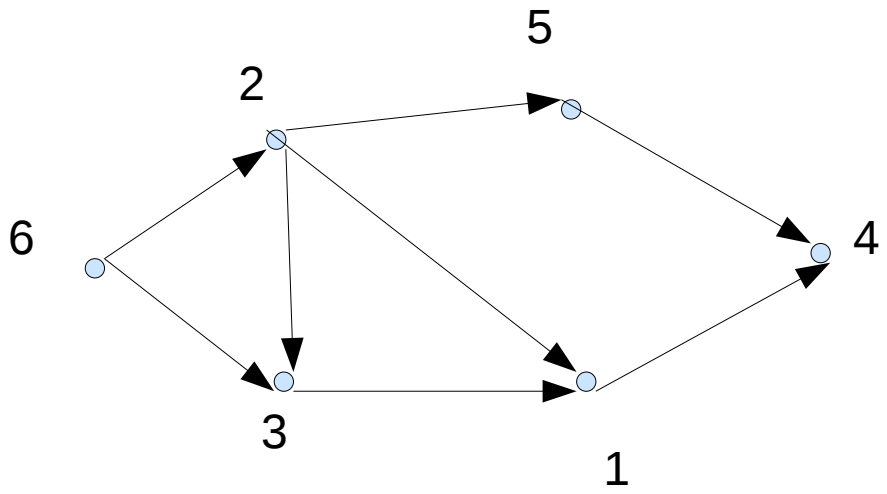
2) Usuń łuki łączące wierzchołki ponumerowane z pustymi.

3) Wierzchołkom swobodnym przydziel kolejne numery i+1, i+2, …

4) Jeśli nie ponumerowano wszystkich wierzchołków idź do kroku 2.

$$\mathbf{B} = \begin{matrix} \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \end{matrix} \quad \mathbf{B} = \begin{matrix} \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \end{matrix} \quad \mathbf{B} = \begin{matrix} \\ 0 \\ 1 \\ 0 \\ 0 \end{matrix} \begin{matrix} \begin{matrix} 1 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{matrix} \quad \mathbf{B} = \begin{matrix} \\ 0 \\ 0 \end{matrix} \begin{matrix} \begin{matrix} 1 & 4 \end{matrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{matrix}$$

1. Wykreślamy 6 kolumnę i 6 wiersz (warstwa w_0)
2. Wykreślamy 2 kolumnę i 2 wiersz (warstwa w_1)
3. Wykreślamy 3 i 5 kolumnę, 3 i 5 wiersz (warstwa w_2)
4. Wykreślamy 1 kolumnę i 1 wiersz (warstwa w_3)
5. Pozostaje 4 kolumna (warstwa w_4)


5
2
6
4
3
1

Obliczenie terminów zdarzeń

1) Obliczenie terminów najwcześniejszych (procedura w przód)

$\pi_1^e = 0$ *zdarzenie początku projektu*

$\pi_i^e = max\left\{\pi_j^e + p_{ij} : (i,j) \in B\right\}$ *dla* $i = 2,3,4,\ldots,n$

2) Obliczenie terminów najpóźniejszych (procedura w tył)

$\pi_n^l = T -$ *czas zakończenia projektu uzyskany z poprzedniej procedury*

$\pi_i^l = min\left\{\pi_j^l - p_{ij} : (i,j) \in B\right\}$ *dla* $i = n-1, n-2, n-3,\ldots,1$

3) Obliczenie rezerw czasu i luzów zdarzeń

$s_{ij}^1 = \pi_j^l - \pi_i^e - p_{ij}$ *Jeżeli* $s_{ij}^1 = 0$ *łuk leży na drodze krytycznej*

$s_i = \pi_j^l - \pi_j^e$ *luz*

http://zasoby1.open.agh.edu.pl/dydaktyka/matematyka/c_badania_operacyjne/krok/krok4_01.html

PERT

PERT (Program Evaluation and Review Technique) – Stochastyczna analiza czasowa przedsięwzięcia

W klasycznym podejściu PERT przyjmuje się, że czasy trwania czynności mają rozkład Beta

Wielkości w sieci PERT

t_{ij}^{o} *czas optymistyczny*

t_{ij}^{m} *czas najpardziej prawdopodobny* $(dominanta)$

t_{ij}^{p} *czas pesymistyczny*

Prawdopodobieństwo zrealizowania czynności ij

$$P\left\{t_{ij} < t_{ij}^{o}\right\} = 0 \qquad P\left\{t_{ij} > t_{ij}^{p}\right\} = 0$$

Analiza sieci PERT

Faza I

Oszacowanie każdej czynności wartości oczekiwanej i wariancji czasu trwania czynności

$$m_{ij} = \frac{t_{ij}^{o} + 4t_{ij}^{m} + t_{ij}^{p}}{6} \qquad S_{ij}^{2} = \left(\frac{t_{ij}^{p} - t_{ij}^{o}}{6} \right)^{2}$$

Faza II

Analiza czasowa jest identyczna z analizą czasową przeprowadzoną sieci CPM, z tym że zamiast ustalonych czasów jak to jest w CPM używane są wartości oczekiwane.

Faza III

Wyznaczenie terminu realizacji projektu.

Termin realizacji w metodzie PERT ma rozkład normalny z wartością oczekiwaną m(tn) równą wartości oczekiwanej terminu najwcześniejszego (t_{n0}) i z wariancją $S^2(t_n)$ równą sumą wariancji czasów trwania czynności należących do zbioru czynności krytycznych C.

Szansa dotrzymania dowolnego terminu dyrektywnego (terminu planowanego) wyliczana jest z wykorzystaniem dystrybuanty rozkładu normalnego

$$m(t_n)=t_n^0, S(t_n)=\sqrt{\sum_{ij:(i,j)\in C} S_{ij}^2}$$

$$t_n : N\,¿$$

$$P\{t_n < TD\} = F(TD) = \Phi\left(\frac{TD - m(t_n)}{S(t_n)}\right)$$

gdzie

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}\, dx$$

Wartość prawdopodobieństwa dotrzymania terminu planowanego powinna znajdować się w granicach [0.3 , 0.6] a to oznacza, że $z \in (-0.5, 0.25)$

Programowanie liniowe (LP)

Postać standardowa

$$\min_{x \in R^n} f(x) = \boldsymbol{c}^T \boldsymbol{x}$$

Przy warunkach

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b} \qquad x \geqslant 0, m \leqslant n$$

Macierz A jest rozmiaru *nxm,* liczba wierszy powinna być mniejsza od liczby zmiennych. C jest n elementowym wektorem kosztów, b jest wektorem o m składowych, a x jest Wektorem niewiadomych o n składowych

Przykład zadania programowania liniowego

$$\min_{x \in R^4} x_1 + 4x_2 + 2x_3 + x_4$$

$$x_1 + x_2 + x_3 + x_4 = 2$$

$$x_1 + x_2 - 3x_3 = -2$$

$$x_1 \geqslant 0, x_2 \geqslant 0, x_3 \geqslant 0, x_4 \geqslant 0$$

Ogólne sformułowania

Bardziej ogólne sformułowania LP

Pełne ograniczenia nierównościowe typu ≤

$$\boldsymbol{a}^T \boldsymbol{x} \leqslant b$$
$$\boldsymbol{a}^T \boldsymbol{x} + z = b, z \geqslant 0$$

Pełne ograniczenia nierównościowe typu ≥

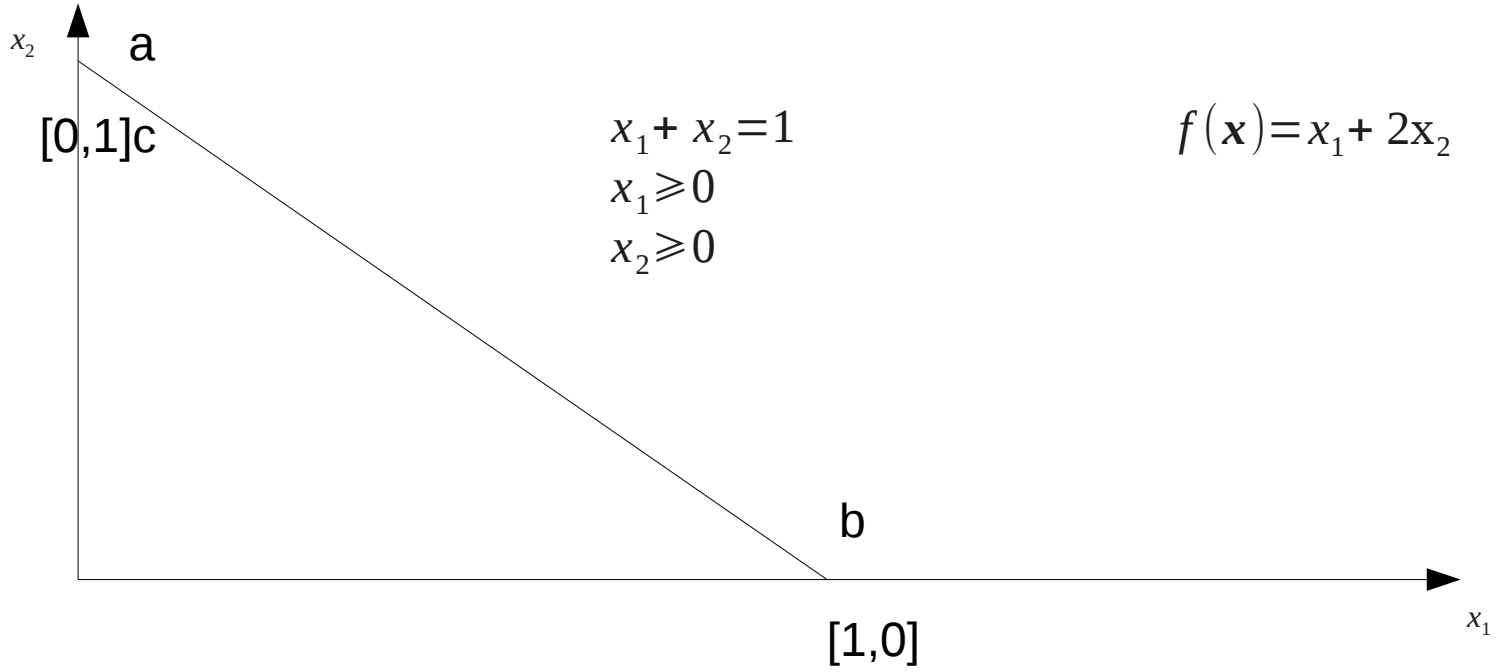
$$\boldsymbol{a}^T \boldsymbol{x} \geqslant b$$
$$\boldsymbol{a}^T \boldsymbol{x} - z = b, z \geqslant 0$$

Przykład użycia programowania liniowego

Wytworzenie jednostki produktu I wymaga zużycia 3 jednostek surowca A i 4 jednostek Surowca B, zaś jednostki produktu II – 2 jednostek surowca A i 5 jednostek surowca B.
Dostawy surowców w każdym dniu wynoszą 12 jednostek surowca A i 23 jednostki Surowca B.
Produkt I sprzedaje się po 14 zł za jednostkę, produkt II po 15. Koszty produkcji wynoszą 3 zł na jednostkę produktu niezależnie od jego rodzaju.
Cel: zmaksymalizować zysk przedsiębiorstwa.

$$f(x_{1,}x2)=14x_1+15x_2-3(x_1+x_2)=11x_1+12x_2$$
$$3x_1+2x_2\leqslant 12$$
$$4x_1+5_{x2}\leqslant 23$$
$$x_{1,}x_2\geqslant 0$$

Metoda sympleks



Cały odcinek [a,b] jest zbiorem punktów optymalnych

W metodzie sympleks optimum jest poszukiwane przez przemieszczanie się po punktach wierzchołkowych zbioru punktów dopuszczalnych. Zbiór punktów dopuszczalnych przeszukiwany jest w uporządkowany sposób generując punkty x_1, x_2, x_k, o wartości funkcji celu nie gorszej niż w poprzednim punkcie.

$$\min_{x \in R^2} -2x_1 - x_2$$

$$x_2 \leqslant 6$$

$$x_1 + x_2 \leqslant 10$$

$$x1 - x2 \leqslant 6$$

$$x1 \geqslant 0, x2 \geqslant 0$$

Postać standardowa

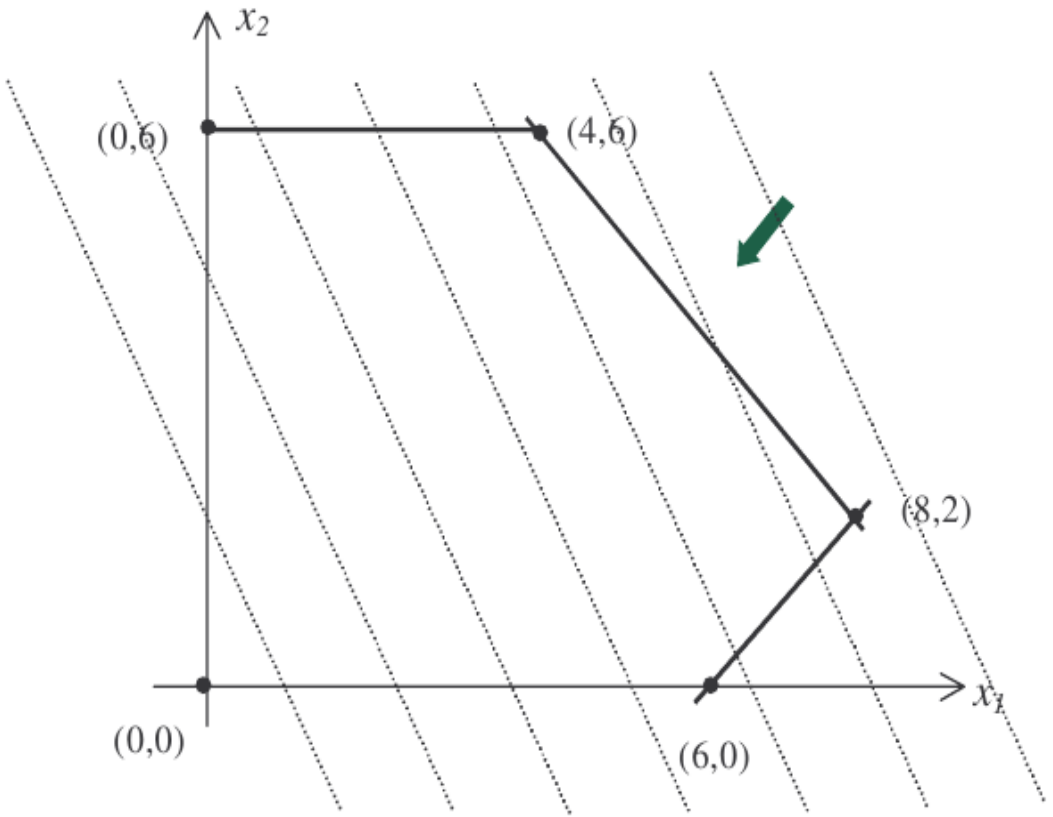
$$\min_{x \in R^4} -2x_1 - x_2$$

$$x_2 + x_3 = 6$$

$$x_1 + x_2 + x_4 = 10$$

$$x_1 - x_2 + x_5 = 6$$

$$x_1, x_2, x_3, x_4, x_5 \geqslant 0$$



Rozwiązanie optymalne można znaleźć przemieszczając się po punktach ekstremalnych zbioru dopuszczalnego

Rozwiązanie bazowe

Każdy punkt ekstremalny ma *n-m* współrzędnych o wartościach równych 0, a wartości m pozostałych wynikają z żądania spełnienia równań ***Ax*=*b***. W związku z tym zbiór indeksów zmiennych można podzielić na dwie grupy:

N - *n-m* zmiennych o wartościach zerowych (określane jako niebazowe)

B – zbiór *m* indeksów zmiennych bazowych

$$[A_B, A_N]\begin{bmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_N \end{bmatrix} = b$$

Rozwiązaniem bazowym zadania programowania liniowego w postaci standardowej jest taki punkt **x**, w którym spełnione są warunki :

- $x_i = 0$ dla wszystkich $i \in N$,
- Macierz bazowa $\boldsymbol{A}_B$ jest nieosobliwa
- $\boldsymbol{x}_B = \boldsymbol{A}_B^{-1}\boldsymbol{b}$

Uwaga, nie każde rozwiązanie bazowe jest punktem dopuszczalnym!
Każde bazowe rozwiązanie dopuszczalne jest wierzchołkiem zbioru rozwiązań dopuszczalnych.

Rozwiązywany układ równań powstaje poprzez wybranie *m* - kolumn z macierzy ograniczeń zadania.

Twierdzenie

Liczba bazowych rozwiązań jest skończona i nie przekracza

$$\frac{n!}{m!(n-m)!}$$

Pierwsze dopuszczalne rozwiązanie bazowe można uzyskać dla $\boldsymbol{A}_B = \boldsymbol{I}$

Bazowe rozwiązanie dopuszczalne

Bazowym rozwiązaniem dopuszczalnym jest taki punkt **x**, w którym spełnione są warunki:

- Wartości zmiennych bazowych $x_B = \tilde{b} = A_B^{-1} b$ są dopuszczalne tzn. że wszystkie współrzędne wektora $\tilde{b} \geqslant 0$ muszą być nieujemne.

Z każdym rozwiązaniem bazowym zadania związana jest pewna specyficzna postać Zadania programowania liniowego. Powstaje w wyniku wyznaczenia zależności zmiennych bazowych od zmiennych niebazowych i eliminacji zmiennych bazowych z funkcji celu.

$$A_B x_B + A_N x_N = b$$

Ponieważ $\mathbf{A}_B$ jest zawsze nieosobliwa to

$$x_B + \tilde{A}_N x_N = \tilde{b} \qquad \tilde{A}_N = A_B^{-1} A_N, \quad \tilde{b} = A_B^{-1} b$$

$$x_B = \tilde{b} - \tilde{A}_N x_N$$

tzw. cena zredukowana

$$f(x) = c_B^T x_B + c_N^T x_N = c_B^T \tilde{b} + \tilde{c}_N x_N, \quad \tilde{c}_N = c_N - A_B^{-1} A_N c_B$$

Definicja

Jeżeli macierz bazowa $\mathbf{A_B}$ jest równa macierzy jednostkowej $\boldsymbol{I}$ a w funkcji celu ceny (współczynniki) odpowiadające zmiennym bazowym mają wartości 0, to mówimy, że zadanie jest w postaci kanonicznej.

Przykład

$$\min_{x\in R^2} x_1 - x_2$$
$$2x_1 + x_2 \leqslant 4$$
$$3x_1 + 2x_2 \leqslant 3$$
$$x_1, x_2 \geqslant 0$$

Postać standardowa

$$\min_{x\in R^4} x_1 - x_2$$
$$2x_1 + x_2 + x_3 = 4$$
$$3x_1 + 2x_2 + x_4 = 3$$
$$x_1, x_2, x_3, x_4 \geqslant 0$$

Rozwiązania bazowe

1) B={1,2}

$$\min_{x\in R^4} 11 - 5x_3 + 3x_4$$
$$x_1 + 2x_3 - x_4 = 5$$
$$x_2 - 3x_3 + 2x_4 = -6$$
$$x_1, x_2, x_3, x_4 \geqslant 0$$

$$\boldsymbol{x} = \begin{bmatrix} 5 \\ -6 \\ 0 \\ 0 \end{bmatrix}$$

$$\tilde{\boldsymbol{c}}_N = \begin{bmatrix} -5 \\ 3 \end{bmatrix}$$

2) B={1,3}

$$\min_{x\in R^4} 1 - \frac{5}{3}x_2 - \frac{1}{3}x_4$$

¿

$$x_1 + \frac{2}{3}x_2 + \frac{1}{3}x_4 = 1$$
$$-\frac{1}{3}x_2 + x_3 - \frac{2}{3}x_4 = 2$$

¿

$$x_1, x_2, x_3, x_4 \geqslant 0$$

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \end{bmatrix}$$

$$\tilde{\boldsymbol{c}}_N = \begin{bmatrix} -\frac{5}{3} \\ -\frac{1}{3} \end{bmatrix}$$

Rozwiązania bazowe

3) B={1,4}

$$\min_{x\in R^4} \; 2-\frac{3}{2}x_2-\frac{1}{2}x_3$$

$$\begin{aligned} x_1 \;+\frac{1}{2}x_2 \;+\frac{1}{2}x_3 &= 2 \\ \frac{1}{2}x_2 \;-\frac{3}{2}x_3 \;+x_4 &= -3 \end{aligned}$$

$$x_1, x_2, x_3, x_4 \geqslant 0$$

$$\boldsymbol{x}=\begin{bmatrix} 2 \\ 0 \\ 0 \\ -3 \end{bmatrix}$$

$$\tilde{\boldsymbol{c}}_N=\begin{bmatrix} -\frac{3}{2} \\ -\frac{1}{2} \end{bmatrix}$$

4) B={2,3}

$$\min_{x\in R^4} \; -\frac{3}{2}+\frac{5}{2}x_1+\frac{1}{2}x_4$$

$$\begin{aligned} \frac{3}{2}x_1 \;+x_2 \;+\frac{1}{2}x_4 &= \frac{3}{2} \\ \frac{1}{2}x_1 \;+x_3 \;-\frac{1}{2}x_4 &= \frac{5}{2} \end{aligned}$$

$$x_1, x_2, x_3, x_4 \geqslant 0$$

$$\boldsymbol{x}=\begin{bmatrix} 0 \\ \frac{3}{2} \\ \frac{5}{2} \\ 0 \end{bmatrix}$$

$$\tilde{\boldsymbol{c}}_N=\begin{bmatrix} \frac{5}{2} \\ \frac{1}{2} \end{bmatrix}$$

5) B={4,2}

$$\min_{x\in R^4} \; -4+3x_1+x_3$$

$$\begin{aligned} -x_1 \;-2x_3 \;+x_4 &= -5 \\ 2x_1 \;+x_2 \;+x_3 &= 4 \end{aligned}$$

$$x_1, x_2, x_3, x_4 \geqslant 0$$

$$\boldsymbol{x}=\begin{bmatrix} 0 \\ 4 \\ 0 \\ -5 \end{bmatrix}$$

$$\tilde{\boldsymbol{c}}_N=\begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

6) B={3,4}

$$\min_{x\in R^4} \; x_1-x_2$$

$$\begin{aligned} 2x_1 \;+x_2 \;+x_3 &= 4 \\ 3x_1 \;+2x_2 \;+x_4 &= 3 \end{aligned}$$

$$x_1, x_2, x_3, x_4 \geqslant 0$$

$$\boldsymbol{x}=\begin{bmatrix} 0 \\ 0 \\ 4 \\ 3 \end{bmatrix}$$

$$\tilde{\boldsymbol{c}}_N=\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Wybór zmiennej wprowadzanej do bazy

Przejście do sąsiedniego rozwiązania bazowego dopuszczalnego polega na wymianie którejś ze zmiennych bazowych ze zmienną niebazową. Oczywiście wybór musi być taki, że gwarantuje zmniejszenie funkcji celu. Ponieważ żądamy dla rozwiązania bazowego

$$x \geqslant 0$$

To wartość dotychczasowej zmiennej niebazowej musi wzrosnąć (dotychczas była 0). Aby dokonać zmniejszenia funkcji celu musimy wybrać zmienną niebazową o ujemnych cenach zredukowanych. Jeśli wszystkie ceny zredukowane są nieujemne tzn. że w żadnym z sąsiednich punktów bazowych wartość funkcji celu nie może zostać zmniejszona i takie rozwiązanie jest punktem optymalnych zadania

Kryterium stopu

$$\tilde{c}_i \geqslant 0, \forall i \in N$$

Kryterium wyboru zmiennej

$$\tilde{c}_q = \min_{i \in N} \tilde{c}_i$$

Wybór zmiennej wyprowadzanej z bazy

Ograniczenia zadania programowania liniowego w postaci kanonicznej:

$$\boldsymbol{x}_B + \tilde{\boldsymbol{A}}_N \boldsymbol{x}_N = \tilde{\boldsymbol{b}}$$

Niech q oznacza indeks zmiennej niebazowej wprowadzanej do bazy. Między zmienną niebazową a zmiennymi bazowymi występują następujące związki

$$\begin{aligned} x_{l(1)} + \tilde{a}_{1q} x_q &= \tilde{b}_1 \\ &\vdots \\ x_{l(i)} + \tilde{a}_{iq} x_q &= \tilde{b}_i \\ &\vdots \\ x_{l(m)} + \tilde{a}_{mq} x_q &= \tilde{b}_m \end{aligned}$$

Zmienna x_q zmniejsza wartość zmiennej bazowej tylko wtedy, gdy współczynnik przy tej zmiennej jest dodatni. Zmienna bazowa osiąga wartość zero gdy

$$x_q = \frac{\tilde{b}_i}{\tilde{a}_{iq}}$$

Ponieważ zmienne bazowe muszą być dodatnie to z bazy można wyprowadzić tą zmienną, która maleje wraz ze wzrostem wartości zmiennej niebazowej wprowadzanej do bazy. Wybierana jest ta z nich, która jako pierwsza osiąga wartość 0. Czyli wybierana jest zmienna dla której zachodzi warunek

$$\frac{\tilde{b}_p}{\tilde{a}_{pq}} = \min_{\substack{j=1,\ldots,m \\ \tilde{a}_{jq}>0}} \frac{\tilde{b}_j}{\tilde{a}_{jq}}$$

W każdej iteracji wszystkie zmienne bazowe muszą mieć wartości dodatnie w związku z tym w każdym kroku następuje zmniejszenie wartości funkcji celu co prowadzi do tego, że nie jest możliwy powrót do rozwiązania bazowego, które już raz pojawiło się w procesie obliczeń. To oznacza, że liczba iteracji metody sympleks jest skończona bo liczba rozwiązań bazowych jest skończona.

Może pojawić się problem w przypadku gdy bi=0 dla pewnych indeksów i (degeneracja), gdyż może dojść do zapętlenia. W takich przypadkach przejście z jednej bazy do drugiej nie Powoduje zmniejszenia wartość funkcji celu (pozostaje ta sama wartość).

Przykłady działania metody sympleks

Startujemy ze zbioru indeksów B={3,4}

$$\min_{x \in R^4} \; x_1 - x_2$$

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 4 \\ 3x_1 + 2x_2 + x_4 &= 3 \\ x_1, x_2, x_3, x_4 &\geqslant 0 \end{aligned}$$

$$\boldsymbol{A}_B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \tilde{\boldsymbol{A}}_N = \boldsymbol{A}_N = \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}$$

$$\boldsymbol{x}_B = \tilde{\boldsymbol{b}} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}, \quad \tilde{\boldsymbol{c}}_N = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Tylko jedna wartość ceny zredukowanej jest ujemna, w związku z tym wprowadzana jest Zmienna niebazowa x2.
Ponieważ

$$\frac{3}{2} = min\left\{\frac{4}{1}, \frac{3}{2}\right\} \rightarrow p = 2, l(p) = 4$$

To z bazy wyprowadzona zostanie zmienna x_4
Nowy zbiór indeksów B={3,2}

$$\boldsymbol{A}_B = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}, \quad \boldsymbol{A}_B^{-1} = \frac{1}{2}\begin{bmatrix} 0 & 1 \\ 2 & -1 \end{bmatrix} \quad \tilde{\boldsymbol{A}}_N = \begin{bmatrix} \frac{3}{2} & 1 \\ \frac{1}{2} & 0 \end{bmatrix}$$

$$\boldsymbol{x}_B = \tilde{\boldsymbol{b}} = \begin{bmatrix} \frac{3}{2} \\ \frac{5}{2} \end{bmatrix}, \quad \tilde{\boldsymbol{c}_N} = \begin{bmatrix} \frac{5}{2} \\ \frac{1}{2} \end{bmatrix}$$

Ponieważ wszystkie koszty zredukowane są dodatnie to jest to optymalne rozwiązanie

Postać tablicowa metody sympleks

$$\begin{matrix}(f)\\(\boldsymbol{B})\end{matrix}\begin{bmatrix}\boldsymbol{c}^T & 0\\ \boldsymbol{A} & \boldsymbol{b}\end{bmatrix}$$

Wykonując na tablicy operacje wierszowe (nie dotyczy wiersza f):

- dodawanie, odejmowanie wierszy
- przeskalowanie wiersza

Można powyższą tabele doprowadzić do postaci

$$\begin{matrix}(f)\\(\boldsymbol{B})\end{matrix}\begin{bmatrix}\boldsymbol{0}^T & \tilde{\boldsymbol{c}}_N^T & -\tilde{f}\\ \boldsymbol{I} & \tilde{\boldsymbol{A}}_N & \tilde{\boldsymbol{b}}\end{bmatrix}$$

Tablica ta reprezentuje postać kanoniczną tablicy sympleksowej.

$$\min_{x \in R^4} \quad x_1 - x_2$$

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 4 \\ 3x_1 + 2x_2 + x_4 &= 3 \\ x_1, x_2, x_3, x_4 &\geqslant 0 \end{aligned}$$

	1	-1	0	0	0
x_3	2	1	1	0	4
x_4	3	2	0	1	3

	1	-1	0	0	0
x_3	2	1	1	0	4
x_4	3	2	0	1	3

	5/2	0	0	1/2	0
x_3	1/2	1	0	-1/2	5/2
x_2	3/2	0	1	1/2	3/2

Algorytm metody sympleks

Na początku dane jest rozwiązanie bazowe dopuszczalne o następujących własnościach

$$\boldsymbol{x}_N^0 = \mathbf{0}$$

$$\boldsymbol{A}_B \, nieosobliwa$$

$$\boldsymbol{x}_B^0 = \tilde{b} = \boldsymbol{A}_B^{-1}\boldsymbol{b} \geqslant 0$$

1) Wyznaczanie indeksu q ∈N o najmniejsze cenie zredukowanej
2) Sprawdzenie testu STOP-u, jeśli w aktualnym rozwiązaniu wszystkie wartości cen zredukowanych są dodatnie to koniec.
3) Wyznaczanie indeksu zmiennej wprowadzanej do bazy, zmienna dla której cena zredukowana jest ujemna.
4) Wyznaczenie macierzy

$$\tilde{\boldsymbol{A}}_N = \boldsymbol{A}_B^{-1}\boldsymbol{A}_N$$

5) Wyznaczanie indeksu p ∈ {1,2,...m} zmiennej wyprowadzanej z bazy
   Jeśli taki indeks nie istnieje to zadanie jest nieograniczone i przerywamy algorytm

6) Wymiana indeksów zmiennej niebazowej x_q i zmiennej bazowej $x_l(p)$

$$B \rightarrow B \cup \{q\} \setminus \{l(p)\}$$

$$N \rightarrow N \cup \{l(p)\} \setminus \{q\}$$

7) Wyznaczenie elementów nowej postaci kanonicznej zadania: $\boldsymbol{A}_B^{-1}$, $\tilde{\boldsymbol{b}} = \boldsymbol{A}_B^{-1}\boldsymbol{b}$, $\tilde{\boldsymbol{c}}_N$
8) Powrót do punktu 1.

Znajdowanie początkowego punktu dopuszczalnego

Wprowadzamy dodatkowe zmienne tzw. zmienne sztuczne, dobierane w taki sposób, by można było utworzyć rozwiązanie bazowe dopuszczalne z macierzą bazową będącą macierzą identyczności.

1) Trzeba doprowadzić do prawdziwości tego warunku $\boldsymbol{b} \geq 0$. Dlatego każde równanie, dla którego $b_i \leq 0$ trzeba pomnożyć przez -1.
2) Wprowadzamy sztuczne zmienne

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$$

3) Definiujemy zadanie pomocnicze

$$\min_{x,r} w = \sum_i r_i$$
$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{r} = \boldsymbol{b}$$
$$\boldsymbol{b} \geqslant 0, \quad \boldsymbol{r} \geqslant 0$$

4) Rozwiązać powyższe zadanie

Znajdowanie początkowego punktu dopuszczalnego

Wprowadzamy dodatkowe zmienne tzw. zmienne sztuczne, dobierane w taki sposób, by można było utworzyć rozwiązanie bazowe dopuszczalne z macierzą bazową będącą macierzą identyczności.

1) Trzeba doprowadzić do prawdziwości tego warunku $\boldsymbol{b} \geq 0$. Dlatego każde równanie, dla którego $b_i \leq 0$ trzeba pomnożyć przez -1.
2) Wprowadzamy sztuczne zmienne

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$$

3) Definiujemy zadanie pomocnicze

$$\min_{x,r} w = \sum_i r_i$$
$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{r} = \boldsymbol{b}$$
$$\boldsymbol{b} \geqslant 0, \quad \boldsymbol{r} \geqslant 0$$

4) Rozwiązać powyższe zadanie

Niech $\hat{w}$ oznacza optymalną wartość funkcji celu. Rozpatrzmy następujące przypadki:

$\hat{w} \geqslant 0$ w tym przypadku zbiór punktów dopuszczalnych jest pusty

$\hat{w} = 0$ i żadna ze zmiennych sztucznych nie występuje w bazie. Wówczas można wyeliminować zmienne sztuczne.

$\hat{w} = 0$ i wśród zmiennych bazowych występują zmienne sztuczne. Możliwe są dwa przypadki:
1) r_i występuje w bazie ale współczynniki $\tilde{a}_{ij} = 0$. Wówczas i-te równanie jest nadmiarowe i można je usunąć
2) r_i występuje w bazie, współczynniki $\tilde{a}_{ij} \neq 0$ dla pewnego j≤n. Zastępuje się wówczas w bazie zmienną r_i zastępuje się zmienną x_j.

Zagadnienie transportowe

Zadanie transportowe polega na rozwiązaniu problemu opracowania planu przewozu produktu z różnych źródeł zaopatrzenia do punktów, które zgłaszają zapotrzebowanie na ten produkt.

Znaleźć minimum

$$z=\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\,x_{ij}$$

Przy warunkach

$$\sum_{j=1}^{n} x_{ij}\leqslant a_i, i=1,2,3,\ldots,m$$

Odbiorcy nie przyjmują więcej towarów niż potrzebują

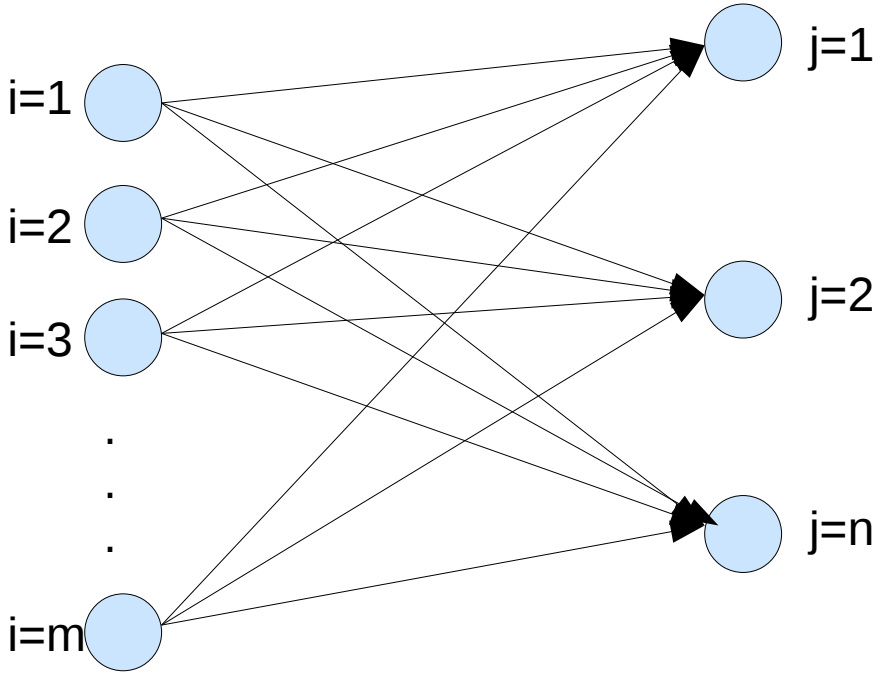
$$\sum_{i=1}^{m} x_{ij}\leqslant b_j, j=1,2,3,\ldots,n$$

Dostawcy nie dostarczą więcej towaru, niż wynoszą ich zdolności podażowe

$$x_{ij}\geqslant 0, i=1,2,3,\ldots,m;\ j=1,2,3,\ldots,n$$

Parametry a, b, c są nieujemne i całkowite.

Zagadnienie transportowe ma prosta interpretację sieciową.



Wszystkie wierzchołki można podzielić na dwie grupy, na węzły dostawy ponumerowane I=1,2,...,m i węzły odbioru ponumerowane j=1,2,...,n. Dla każdego łuku występującego pomiędzy węzłami określony jest jednostkowy koszt c_{ij} przewozy transportowanego towaru.

Zadanie polega na wyznaczeniu takich wielkości przewozu x_{ij}, które minimalizują całkowity koszt transportu z. Pierwszych m nierówności odnosi się do wierzchołków dostawy, następne N nierówności odnosi się do wierzchołków odbioru.

Programowanie całkowitoliczbowe (PCL)

W wielu praktycznych zastosowaniach wymagane jest by rozwiązanie optymalne było całkowitoliczbowe.

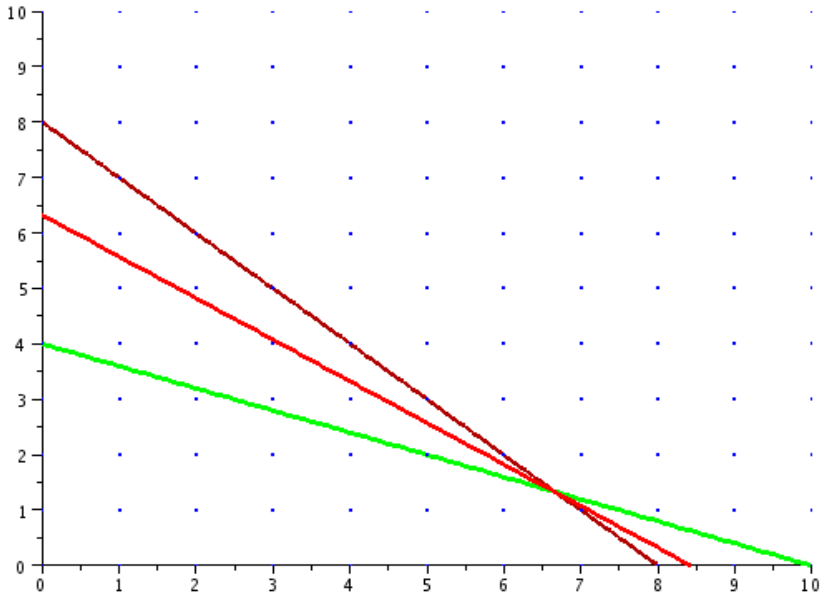
$$\max_{x \in R^2} z = 4x_2 + 4x_1$$

$$x_1 + x_2 \leqslant 12$$

$$2x_1 + 5x_2 \leqslant 41$$

$$x_1 \geqslant 0, x_2 \geqslant 0, x1, x_2 \text{ musi być całkowite}$$

Rozwiązanie optymalne dla problemu ciągłego jest następujące 41.6 x_1=5.666 x_2=6.333
Gdyby próbować zaokrąglić to rozwiązanie czyli x1=6, x2=6 wówczas rozwiązania to nie jest dopuszczalne



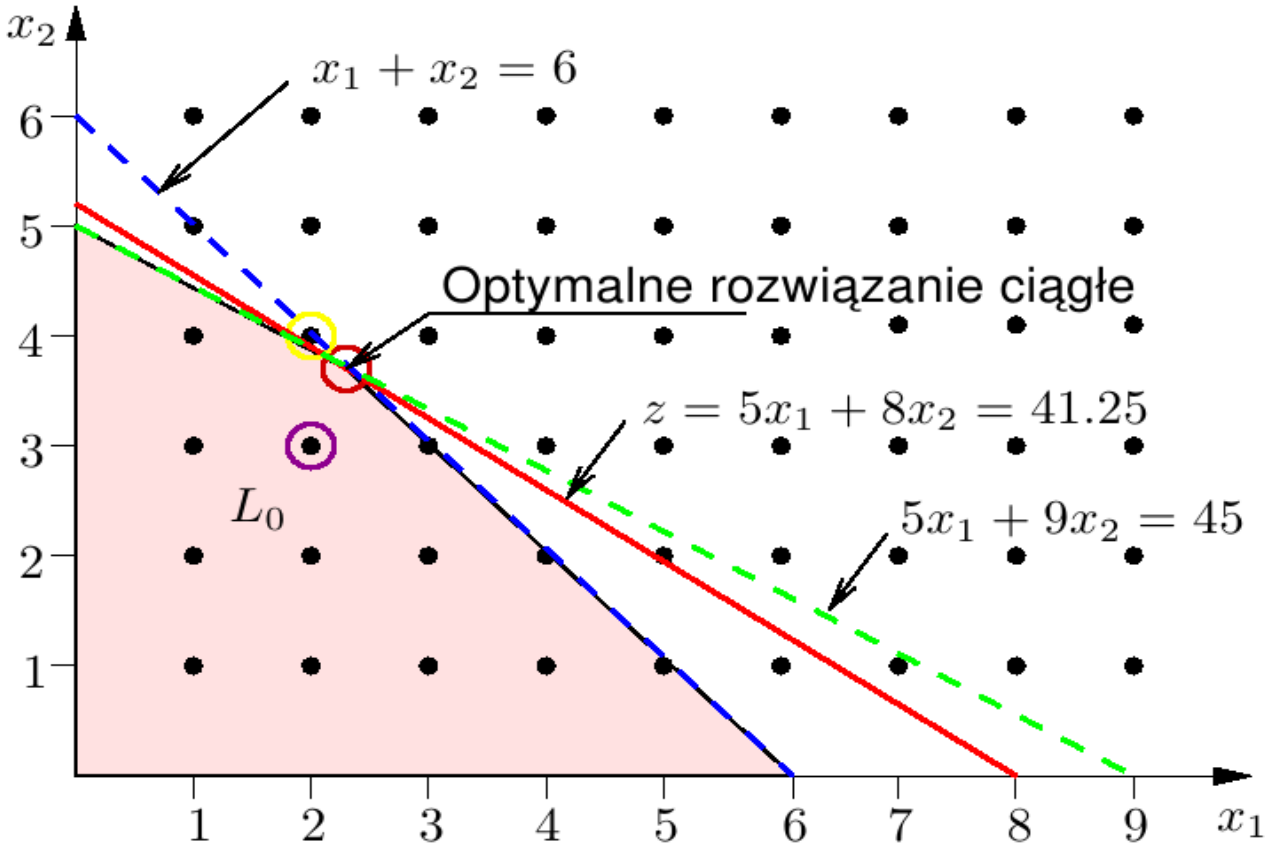
$$\max_{x \in R^2} z = 5x_1 + 8x_2$$

$$x_1 + x_2 \leqslant 6$$

$$5x_1 + 9x_2 \leqslant 45$$

$$x_1 \geqslant 0, x_2 \geqslant 0, x1, x_2 \text{ musi być całkowite}$$

Rozwiązanie optymalne
x1=2.25, x2=3.75
z=41.25



www.ioz.pwr.wroc.pl/pracownicy/.../ztaw9m.pdf

Metoda podziału i ograniczeń dla PCL

Rozwiązanie ciągłe jest górnym ograniczeniem rozwiązania całkowitoliczbowego i jest to punkt startowy dla algorytmu podziału i ograniczeń.
Wybieramy zmienną, której wartość jest ułamkowa (dowolną zmienną) np. x2 i narzucamy dodatkowe warunki $x2 \leq 3$ i $x2 \geq 4$.
Jeżeli rozwiązanie, które otrzymujemy jest całkowitoliczbowe to nie rozwijamy takiego węzła!

x1=2.25, x2=3.75
z=41.25

x2≤ 3

x2≥4

$\max_{x \in R^2} z = 5x_1 + 8x_2$

$x_1 + x_2 \leqslant 6$

$5x_1 + 9x_2 \leqslant 45$

$x_2 \leqslant 3$

$x_1 \geqslant 0, x_2 \geqslant 0$

x1=1.8, x2=4
z=41

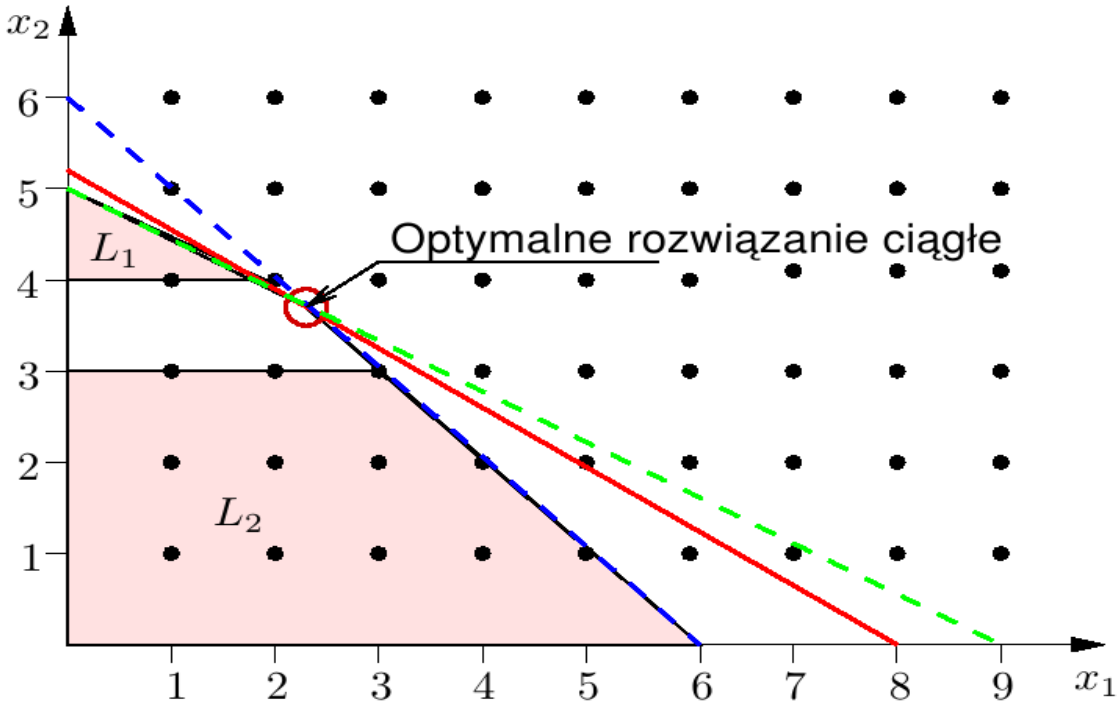
$\max_{x \in R^2} z = 5x_1 + 8x_2$

$x_1 + x_2 \leqslant 6$

$5x_1 + 9x_2 \leqslant 45$

$x_2 \geqslant 4$

$x_1 \geqslant 0, x_2 \geqslant 0$



x1=2.25, x2=3.75
z=41.25

x2≤ 3

x2≥4

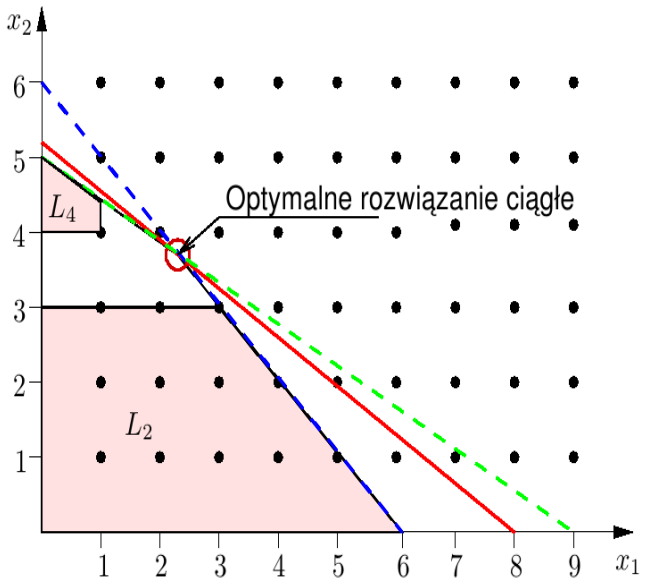
x1=1.8, x2=4
z=41

x1≤ 1

x1≥2

x1=1, x2=4.(4)
Z=40.(5)

sprzeczne



$$\max_{x \in R^2} z = 5x_1 + 8x_2$$

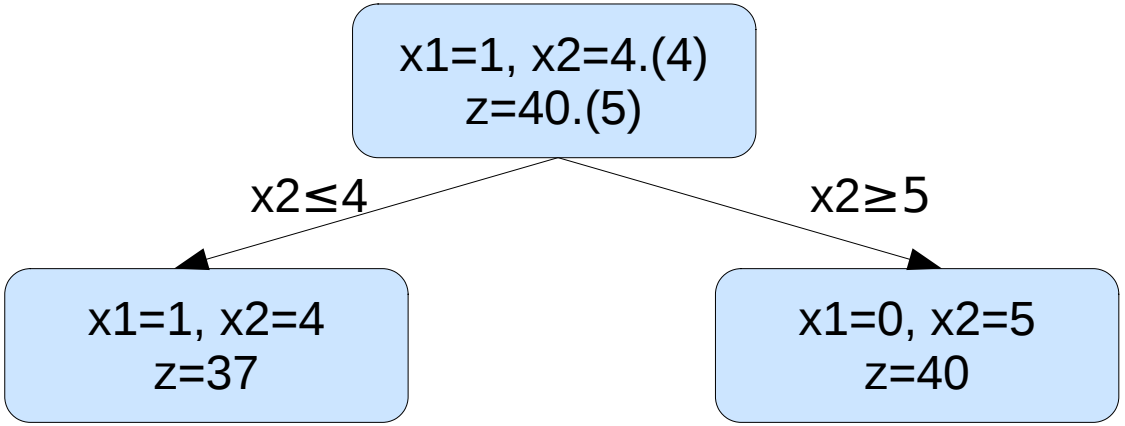
$$x_1 + x_2 \leqslant 6$$

$$5x_1 + 9x_2 \leqslant 45$$

$$x_2 \geqslant 4$$

$$x_1 \leqslant 1$$

$$x_1 \geqslant 0, x_2 \geqslant 0$$



Ponieważ obydwa rozwiązania są całkowitoliczbowe to dalej nie rozwijamy

Zagadnienie plecakowe

Istnieje wiele różnych wersji zagadnienia plecakowego. Będziemy rozważać 0-1 Zagadnienie plecakowe, które ma następującą postać:

znaleźć maksimum

$$\sum_{i=1}^{n} p_i x_i$$

przy warunkach

$$\sum_{i=1}^{n} w_i x_i \leqslant V \, , x_i = 0 \, lub \, 1, i = 1,2,\ldots,n$$

gdzie p_i i w_i (i=1,2, …,n) oraz V są całkowite

Naszym zadaniem jest tak wypełnić plecak różnymi przedmiotami o wartościach p_i i wagach w_i by nie przekroczyć łącznego udźwigu plecaka V. Trzeba tak wybrać przedmioty aby wartość przedmiotów załadowanych do plecaka była jak największa.
Dodatkowo można założyć (bez utraty ogólności), że

$$p_i \, i \, w_i \, są \, dodatnie \, i \, całkowite$$

$$w_i \leqslant V \, , i = 1,\ldots,n$$

$$\sum_{i=1}^{n} w_i > V$$

Zero-jedynkowe zagadnienie plecakowe można uogólnić na zagadnienie wieloplecakowe, w którym przedmioty można włożyć do m plecaków o pojemnościach V_j (j=1, 2, …, m)

$$\sum_{i=1}^{n}\sum_{j=1}^{m} p_i x_{ij}$$

$$\sum_{i=1}^{n} w_{ij} x_i \leqslant V_j, j=1,2,\ldots,m$$

$$\sum_{j=1}^{m} x_{ij} \leqslant, i=1,2,\ldots,n$$

$$x_{ij}=0 lub 1, i=1,\ldots,n; j=1,\ldots,m$$

Zagadnienie plecakowe nazywa się często zagadnieniem załadunku, które polega na przydzieleniu przedmiotów o znanych rozmiarach do pudełek o ograniczonej pojemności W taki sposób aby zużyć najmniejszą liczbę pudełek.

Znajdź minimum $\sum_{j=1}^{m} y_j$

$$\sum_{j=1}^{m} x_{ij}=1, i=1,2,\ldots,n$$

$$\sum_{i=1}^{n} W_i x_{ij} \leqslant k_j y_j, j=1,2,\ldots,m$$

$$y_j, x_{ij}=0 lub 1, i=1,\ldots,n; j=1,\ldots,m$$

W dopuszczalnym załadunku x_{ij}=1 jeśli przedmiot i ma być umieszczony w pudełku j, y_j=1 jeśli używa się pudełka j.

Przedstawione odmiany zagadnienia plecakowego są oczywiście specjalnymi przypadkami Programowania całkowitoliczbowego. Co więcej istnieje również relacja odwrotna między ILP i 0-1 KP tzn. większość problemów ILP można sprowadzić do równoważnych z nimi 0-1 KP. Stąd jeżeli udało by się znaleźć rozwiązanie wielomianowe problemu KP to również istnieje wielomianowe rozwiązanie problemów ILP.
Z obliczeniowego punktu widzenia problemy KP są trudne (NP-trudne) i jak dotychczas nie udało się znaleźć żadnego algorytmu o wielomianowym czasie działania, co więcej uważa się, że jest mało prawdopodobne aby taki algorytm istniał.

Rozpatrzmy zachłanny algorytm znajdowania dolnego oszacowania wartości zagadnienia plecakowego. Algorytm ten można stosować również do częściowo wypełnionego plecaka (zbiór przedmiotów w plecaku ozn. J). Możemy też założyć że istnieje zbiór przedmiotów, który do plecaka wkładać nie będziemy ozn. K. Otrzymane rozwiązanie to zbiór J elementów znajdujących się w plecaku

$$
\begin{aligned}
&LOWERBOUND(J,K)\\
&begin\\
&x \leftarrow V - \sum_{i \in J} w_i ; y \leftarrow \sum_{i \in J} p_i\\
&for\ i \leftarrow 1\ n\ do\\
&\quad if\ (i \neq J \cup K) \wedge (w_i \leqslant x)\ then\\
&\quad begin\\
&\qquad J \leftarrow J \cup i ; x \leftarrow x - w_i ; y \leftarrow y + p_i\\
&\quad end\\
&lowerbound \leftarrow y\\
&end
\end{aligned}
$$

Oczywiście nawet dla ustalonych zbiorów J i K wartość lowerbound zależy od porządku rozpatrywania przedmiotów. Nie istnieje żadna ogólna reguła, która zapewniałaby, że otrzymamy optymalne rozwiązanie. Często jednak używa się heurystycznej reguły porządkowania przedmiotów zgodnie z ich nierosnącymi wartościami p_i/w_i.

Przybliżony algorytm rozwiązywania zagadnienia plecakowego

$KNAP(k)$
begin
$Q \leftarrow 0$
for wszystkie takie podzbiory $I \subset 1,2,3,\ldots,n$, *że* $|I| \leqslant k$ *i* $\sum_{i \in I} w_i \leqslant V$ *do*
$Q \leftarrow max\{Q, LOWERBOUND(I, \emptyset)\}$
$res \leftarrow Q$
end

Przykład

$$max\ z = 200x_1 + 155x_2 + 115x_3 + 90x_4$$
$$50x1 + 40x_2 + 30x_3 + 25x_4 \leqslant 95$$
$$x_i = 0\ lub\ 1, i = 1,2,3,4$$

Porządek zmiennych dla procedury zachłannej jest zgodny z przedstawionym przykładem gdyż 200/50>=155/40>=115/30>=90/25.
Procedura zachłanna dla tego przykładu daje następujące rozwiązanie (1,1,0,0) i z =355
Zaczynamy od *k*=1, podzbiory przedmiotów *I*={1}, *I*={2} dają to samo rozwiązanie w algorytmie zachłannym, dla *I*={3} otrzymujemy (1,0,1,0) i *z*=315, *I*={4} (1,0,0,1) *z*=290.
Podobną analizę przeprowadzamy dla
k=2
I={1,2} → (1,1,0,0) z=355
I={1,3} → (1,0,1,0) z=315
I={1,4} → (1,0,0,1) z=290
I={2,3} → (0,1,1,1) z=360
I={2,4} → (0,1,1,1) z=360
I={3,4} → (0,1,1,1) z=360
k=3
Tu występuje tylko jeden dopuszczalny zbiór
I={2,3,4} z=360

Tw. Rozwiązanie, które otrzymuje się za pomocą powyższego algorytmu dla *k*>0 spełnia nierówność

$$\frac{P_{opt}(J)-KNAP(k)}{P_{opt}(J)}<\frac{1}{k+1}$$

dla każdego zagadnienia J. Czas obliczeń jest rzędu O(kn^{k+1}) i wymagana jest pamięć rzędu O(n), gdzie n jest liczbą przedmiotów w J

Algorytm przeglądu pośredniego (Metoda podziału i ograniczeń)

Algorytm ten wykorzystuje reprezentację przestrzeni rozwiązań w postaci drzewa. Istnieje wiele drzew odpowiadających rodzinie 2^n wektorów zero-jedynkowych. Wykorzystamy drzewo, w którym zmienne odpowiadają kolejnym poziomom drzewa i w którym każdy węzeł drzewa ma co najwyżej dwóch synów. Zakładamy, że każdy lewy syn odpowiada wartości 1 zmiennej.
Budując drzewa trzeba pamiętać o tym, że nie wszystkie węzły odpowiadają dopuszczalnym Rozwiązaniom i tylko niektóre dopuszczalne odpowiadają rozwiązaniem optymalnym. Ponieważ rozmiar drzewa jest wykładniczy, metoda powinna unikać (o ile to możliwe) tworzenia węzłów, które są albo niedopuszczalne albo nie prowadzą do rozwiązań lepszych od dotychczas znalezionego.
Aby uniemożliwić generowanie węzłów, które nie są w stanie polepszyć bieżącego rozwiązania korzysta się z funkcji ograniczającej. Niech PROF oznacza wartość najlepszego znanego rozwiązania i niech x_1,x_2, …, x_n ozn. to rozwiązanie.
Wartość funkcji ograniczającej można wyznaczyć stosując twierdzenie Danziga. Jeśli oszacowanie to oszacowanie od góry nie jest większe od PROF to dalsze poszukiwanie w dół od tego węzła może być zaniechane i przeszukiwanie może się przenieść do prawego syna bieżącego węzła.
Jeśli jakiś węzeł ma dopuszczalnego lewego syna, to oba te węzły mają tą samą wartość funkcji ograniczającej, dlatego powinna ona być stosowana tylko dla lewych synów.

Tw. (Dantzig)
Optymalnym rozwiązaniem zagadnienia plecakowego z warunkiem

$$x_i = 0 \, lub \, 1, i = 1, 2, \ldots, n$$

Osłabionym do warunku

$$0 \leqslant x_i \leqslant 1, i = 1, 2, \ldots, n$$

jest

$$x_i = 1, i = 1, 2, \ldots, r$$
$$x_i = 0, i = r + 2, \ldots, n$$
$$x_{r+1} = \frac{1}{w_{r+1}} \left(V - \sum_{i=1}^{r} w_i \right)$$

$$gdzie \, r \, jest \, największym \, wskaźnikiem, dla \, którego \sum_{i=1}^{r} w_i \leqslant V$$

Stąd górne oszacowanie wartości zagadnienia plecakowego

$$U_1 = \sum_{i=1}^{r} p_i + \left[\frac{p_{r+1}}{w_{r+1}} \left(V - \sum_{i=1}^{r} w_i \right) \right]$$

$BOUND(P, W, k, Q, U, l)$

begin

$Q \leftarrow P; U \leftarrow W; l \leftarrow k+1; B \leftarrow true$

while $(l \leqslant n)$ *i B do*

begin

$B \leftarrow U + w_l \leqslant V$

if (B) *then*

begin

$Q \leftarrow Q + p_l; U \leftarrow U + w_l; y_l \leftarrow 1; l \leftarrow l+1$

end

end

if B then $BOUND \leftarrow Q$ *else* $BOUND \leftarrow Q + \dfrac{(V-U)\,p_l}{w_l}$

end

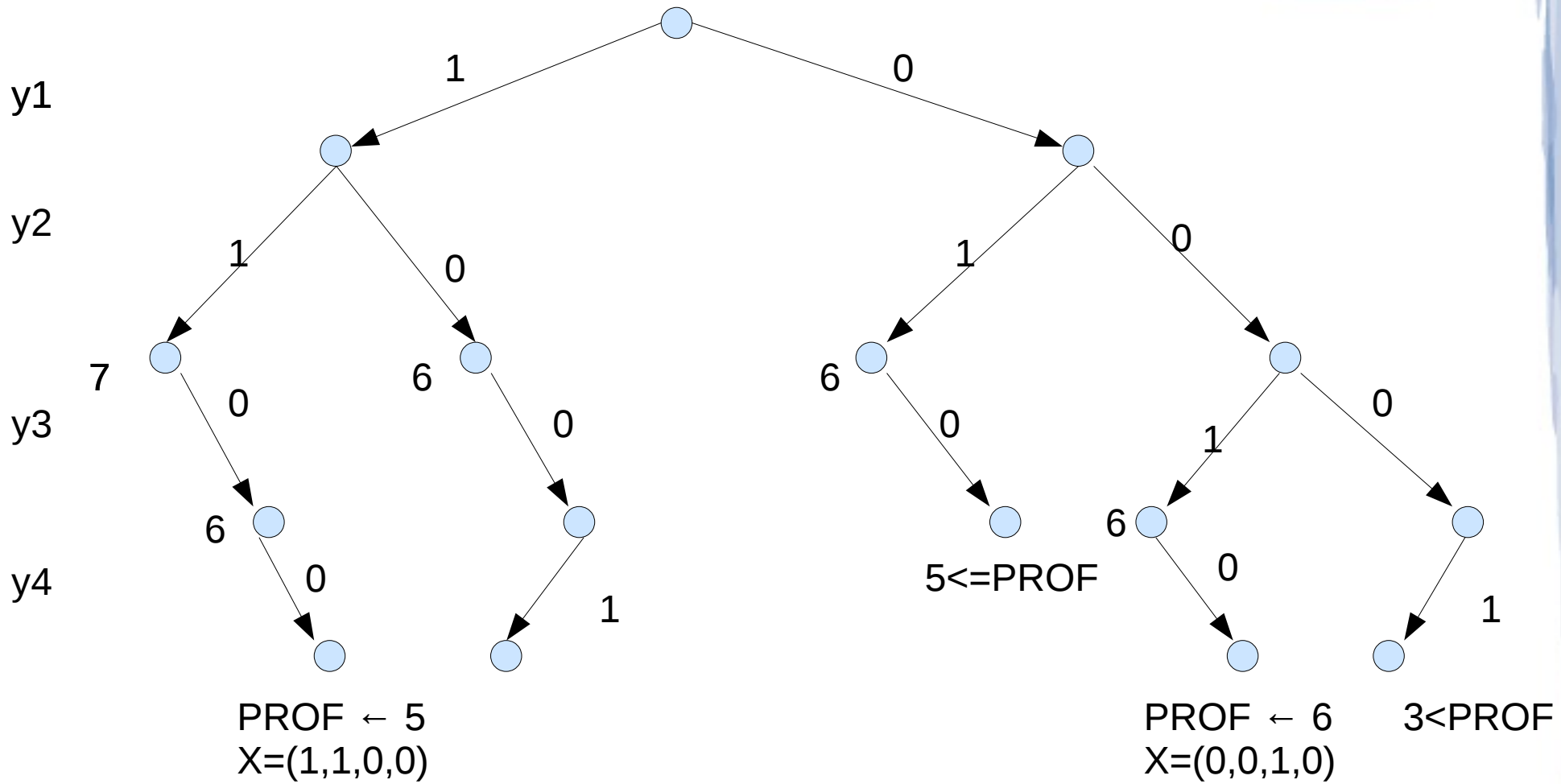
Algorytm przeglądu Horowitza i Shaniego

```
begin
 k ← 0 ; PROF ← −1 ; P ← 0 ; W ← 0 ;
 repeat {do ukończenia przeglądu tzn. k = 0}
  repeat {do momentu gdy można uczynić znaczący krok do przodu}
  B ← BOUND( P , W , k , Q , U , l ) ⩽ PROF ;
  if B then
  begin {krok do tyłu do ostatniego przedmiotu w plecaku}
   znaleźć największe j , takie że y_j = 1 i j ⩽ k
   k ← j
   if k > 0 then
   begin {usunięcie k − tego przedmiotu z plecaka}
    y_k ← 0
    P ← P − p_k
    W ← W − w_k
   end
  end
  until (not B) ∨ (k = 0)
  if (k > 0) lub (PROF = −1) then
  begin {Krok do przodu}
   P ← Q ; W ← U ; k ← l
   if k > n then
    begin {Znaleziono lepsze rozwiązanie}
    PROF ← P ; k ← n
    x_i ← y_i (i = 1, 2, ... , n)
    end else y_k ← 0
   end
   until k = 0
  end
```

Przykład

$$min\, 2x_1 + 3x_2 + 6x_3 + 3x_4$$
$$x_1 + 2x_2 + 5x_3 + 4x_4 \leqslant 5$$
$$x_1, x_2, x_3, x_4 = 1 \; lub \; 0$$

Początkowo P=W=0; k=0 i PROF=-1, po pierwszym wywołaniu BOUND mamy y1=y2=1, Q=5, U=3, l=3, BOUND=5+2*6/5=7. Ponieważ PROF=-1 to robimy krok do przodu P ← Q, W ← U, k ← l, y3 ← 0 i ponownie wywoływane jest BOUND. Wartość y4 jest niedopuszczalna w związku z tym y4=0. Ponownie wywołujemy BOUND l=5 >n=4, ponieważ wartość funkcji nie jest większa niż -1 więc bieżące rozwiązanie staje się najlepszym, dotąd znalezionym czyli za x podstawiamy y=(1,1,0,0) PROF=P=5, k=4.
Następny krok przesuwa przeszukiwanie do ostatniego przedmiotu w plecaku i przedmiot ten zostaje usunięty y2 ← 0, P ← 5-3 =2 W ← 3-2=1. Wywołujemy funkcję BOUND próbując Umieścić przedmiot 3 w placeku co się nie udaje bo U+w3=6>5. Kolejne wywołanie BOUND Umieszcza przedmiot 4 w plecaku, to jednak nie polepsza istenijącego rozwiązania.
Teraz przeszukiwanie przechodzi do y1 i y1 ← 0 i wywoływana jest funkcja BOUND, po ustaleniu y2 ← 1 BOUND daje 6 itd.....


y1
y2
y3
y4
1
0
1
0
1
0
7
6
6
0
0
0
0
6
1
0
5<=PROF
6
0
1
PROF ← 5
X=(1,1,0,0)
PROF ← 6
X=(0,0,1,0)
3<PROF

Problem plecakowy w kryptografii

- Szybko rosnące ciągi

$$\sum_{i}^{j-1} a_i < a_j \; dla \; j = 2,3,\ldots,n$$

$$2, 3, 7, 14, 27$$

- System kryptograficzny Merklego Hellmana, do roku 1982 uważany był za najlepszy system o tzw. kluczu publicznym
- Załóżmy, że $a_1,\ldots,a_n$ jest szybko rosnącym ciągiem liczb naturalnych (klucz prywatny). Niech *m* będzie liczbą naturalną większą od $2a_n$. Weźmy dowolną liczbę całkowitą nieujemną *w* względnie pierwszą z *m* i utwórzmy ciąg $b_1, b_2, \ldots, b_n$ biorąc $wa_1, wa_2, \ldots, wa_n$ mod *m* (tzw. klucz publiczny).

Generacja kluczy

- Alicja generuje klucz publiczny i klucz prywatny, które zostaną użyte do procesu szyfrowania.
- 1 Alicja wybiera liczbę całkowitą n, która jest ustalona jako parametr systemowy.
- 2 Alicja wybiera wektor superrosnący (b1, b2, ..., bn) i moduł M taki, że M > b1 + b2 + ... + bn.
- 3 Alicja wybiera losową liczbę całkowitą W , 1<= W <= M − 1, taka, że NWD(W , M) = 1.
- 4 Alicja wybiera losową permutację π na liczbach całkowitych 1, 2, ..., n.
- 5 Alicja oblicza ai = $Wb_{\pi(i)}$ (mod M) dla i = 1, 2, ..., n.
- 6 Alicja publikuje PK = (a1, a2, ..., an) tzw. klucz publiczny. Alicja utrzymuje w sekrecie SK = (π, M, W ,(b1, b2, ..., bn)) tzw. klucz prywatny

Algorytm szyfrowania

- Bob chce zaszyfrować wiadomość i wysłać do Alicji.
- 1 Bob otrzymuje klucz publiczny PK = (a_1, a_2, ..., a_n).
- 2 Bob przyjmuje za wiadomość ciąg binarny m o długości n, m = $m_1m_2...m_n$.
- 3 Bob oblicza liczbę całkowitą c = $m_1a_1m_2a_2...m_na_n$
- 4 Bob wysyła do Alicji kryptotekst c.

Algorytm deszyfrowania

- Alicja po otrzymaniu kryptotekstu od Boba chce odszyfrować wiadomość m z kryptotekstu c według poniższej procedury.
- 1 Alicja otrzymuje c, zna klucz prywatny

  $SK = (\pi, M, W, (b_1, b_2, ..., b_n))$ i na ich podstawie oblicza $d = W^{-1} c \pmod{M}$.
- 2 Alicja, by poznać m używa algorytmu rozwiązującego problem plecakowy i znajduje liczby całkowite $r_1, r_2, ..., r_n$, $r_i \in 0, 1$ takie, że $d = r_1b_1 + r_2b_2 + ... + r_nb_n$.
- 3 Bity wiadomości $mi = r_{\pi(i)}$, $i = 1, 2, ..., n$.

- Przedstawmy przykład przebiegu kryptosystemu plecakowego Merkle-Hellman'a. W przykładzie użyjemy sztucznie małych parametrów w celu zapewnienia przejrzystości rozwiązania i ułatwienia obliczeń.
- Generacja kluczy:
- Wybieramy liczbę całkowitą n = 6. oraz wektor superrosnący (b1, b2, ..., bn) = (12, 17, 33, 74, 157, 316) i moduł M = 737 taki, że

  M > b1 + b2 + ... + bn. Wybieramy losową liczbę całkowitą W = 635, 1 <= W <= 736, taką, że NWD(W , M) = 1. Wybieramy losową permutację π na liczbach całkowitych 1, 2, 3, 4, 5, 6,zdefiniowaną

  π(1) = 3, π(2) = 6, π(3) = 1, π(4) = 2, π(5) = 5, π(6) = 4

- Publikujemy PK = (319, 196, 250, 477, 200, 559) - klucz publiczny, a w sekrecie utrzymujemy klucz prywatny SK = (π, M, W ,(12, 17, 33, 74, 157, 316).
- Szyfrowanie:
- Chcemy zaszyfrować wiadomość m = 101101. Obliczamy zatem

  c = 319 + 250 + 477 + 559 = 1605

  Przekazujemy do odszyfrowania kryptotekst c = 1605.

Deszyfrowanie:

- By odszyfrować wiadomość m obliczamy $d = W^{-1}c \pmod M = 136$ i rozwiązujemy problem plecakowy wektora super rosnącego i otrzymujemy. W^{-1} jest tzw. modularną odwrotnością muliplikatywną, którą można znaleźć przy użyciu rozszerzonego algorytmu euklidesa.
- $136 = 12r1 + 17r2 + 33r3 + 74r4 + 157r5 + 316r6$.
- Uzyskujemy

  $136 = 12 + 17 + 33 + 74$
- Mamy zatem

  $r1 = 1, r2 = 1, r3 = 1, r4 = 1, r5 = 0, r6 = 0$.
- Stosujemy teraz permutację π i otrzymujemy kolejne bity wiadomości:
- $m1 = r3 = 1, m2 = r6 = 0, m3 = r1 = 1$
- $m4 = r2 = 1, m5 = r5 = 0, m6 = r4 = 1$

Zagadnienie pokrycia

Niech M={1,2,...,m} będzie zbiorem bazowym i niech $\mathbb{P}$={P_1,P_2,...,P_n} będzie rodziną podzbiorów zbioru M ($P_j \subset M$).
Rodzina $Q \leq \mathbb{P}$ stanowi upakowanie zbiorów, jeśli wszystkie elementy Q są wzajemnie rozłączne. Jeśli upakowanie zbiorów pokrywa wszystkie elementy zbioru bazowego M, to nazywa się go rozbiciem. Pokryciem zbioru jest podrodzina, która pokrywa wszystkie elementy M (składniki nie muszą być rozłączne).

Niech c będzie nieujemną funkcją wagową (wektorem), określoną na $N=\{1,2,...,n\}$
tzn. każdy podzbiór P_j może kosztować (dawać zysk).

Zagadnienie upakowania (SP) zbioru polega na znalezieniu upakowania o maksymalnej wadze,
zagadnienie rozbicia (SPP) zbioru polega na znalezieniu rozbicia o minimalnej wadze,
Zagadnienie pokrycia zbioru (SC) polega na znalezieniu pokrycia o minimalnej wadze.
Zagadnienie pokrycia można również sformułować jako specjalne zagadnienie programowani całkowitoliczbowego.

Niech $A=[a_{ij}]$ będzie macierzą incydencji (wymiaru mxn), reprezentującą elementy podzbioru zbioru $\mathbb{P}$. Każdy wiersz odpowiada elementom zbioru M, kolumna odpowiada elementom rodziny $\mathbb{P}$. Elementy a_{ij} mogą przyjmować wartość 0, 1. Niech R_i oznacza i-ty wiersz macierzy A, a C_j j-tą kolumnę macierzy A.

Wówczas zagadnienia pokrycia można sformułować następująco:

Znaleźć maksimum
przy warunkach
(SP)

$$\boldsymbol{c}^T \boldsymbol{x}$$
$$A\boldsymbol{x} \leqslant \boldsymbol{e}$$
$$x_j = 0 \; lub \; 1, \; j = 1, 2, \ldots, n$$

Znaleźć minimum
przy warunkach
(SPP)

$$\boldsymbol{c}^T \boldsymbol{x}$$
$$A\boldsymbol{x} = \boldsymbol{e}$$
$$x_j = 0 \; lub \; 1, \; j = 1, 2, \ldots, n$$

Znaleźć minimum
przy warunkach
(SC)

$$\boldsymbol{c}^T \boldsymbol{x}$$
$$A\boldsymbol{x} \geqslant \boldsymbol{e}$$
$$x_j = 0 \; lub \; 1, \; j = 1, 2, \ldots, n$$

Gdzie ***e*** jest *m*-wymiarowym wektorem 1

Zastosowanie przykłady

Zagadnienie planowania lotów personelu lotniczego. Linia lotnicza opracowuje plan lotów a następnie przydzielany jest personel latający do każdego lotu. Przydziały te muszą być zgodne z licznymi ograniczeniami, a linie lotnicze dążą do zminimalizowania kosztów. Plan lotów może być wykorzystany do wygenerowania wszystkich możliwych rotacji. Każdej rotacji można przydzielić koszt obsługi, w związku z tym zagadnienie polega na podziale wszystkich lotów na rotacje, których sumaryczny koszt jest minimalny. W tym przypadku zbiór bazowy składa się z lotów, a rodzina podzbiorów $\mathbb{P}$ ze wszystkich możliwych rotacji.

Projektowanie obwodów połączeniowych. Dany jest zbiór k symboli 0,1 i jeden binarny symbol na wyjściu, którego wartość zależy od 2^k możliwych wektorów danych. Podana Jest funkcja przełączająca lub równoważna tablica logiczna, która podaje wartość wyjścia dla 2^k możliwych wektorów. Zagadnienie polega na zbudowaniu najtańszego Obwodu, który realizuje funkcję i składa się tylko z bramek ”and” lub „or”.

Zagadnienie testowania. Polega na znalezieniu najtańszego zbioru testów, który byłby zdolny do wykrycia błędów w sieci. Test odpowiada ścieżce w sieci i może wykryć tylko błędy w wierzchołkach znajdujących się na tej ścieżce. W związku z tym zagadnienie polega na znalezieniu optymalnej rodziny ścieżek pokrywających wszystkie wierzchołki sieci.

Metoda przeglądu pośredniego dla zagadnienia rozbicia zbioru

Niech *Y* oznacza rozwiązanie częściowe tzn. żaden element *M* nie jest pokryty więcej niż jedną kolumną *Y*. Algorytm składa się z dwu głównych ruchów: do przodu i do tyłu. Ruch do przodu powiększa bieżące rozwiązanie *Y* o kolumnę, która była już przetestowana ze względu na dopuszczalność i dominację a potem przenosi poszukiwanie na następnego kandydata do włączenia. Ruch do tyłu następuje, gdy rozwiązania częściowego nie daje się już powiększyć do rozwiązania dopuszczalnego. Usuwana jest z *Y* ostatnio dodana kolumna i szukana jest następna kolumna dopuszczalna.

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

$$m=9, n=11, \boldsymbol{h}^T=(3,3,1,2,2,2,3,3,2,1,1)$$

Niech B_i będzie podzbiorem kolumn, które mają pierwszy niezerowy element w wierszu *i*

Jeśli Y={1} to element 4 nie może być pokryty żadną z kolumną z bloków B_k (k>=2). Jeśli Y={2,4} to element 6 nie może być pokryty żadną z kolumną z bloków B_k (k>=5). Podczas rozważania kolumn C_j, test dopuszczalności sprawdza czy nie C_j nie ma jedynek w miejscach, które występują w istniejącym rozwiązaniu częściowym Y.
W celu szybkiej eliminacji pewnych rozwiązań, gdy znalezione zostanie rozwiązanie dopuszczalne, wówczas jego koszt może być użyty do eliminacji pewnych rozwiązań częściowych, które nie są w stanie polepszyć najlepszego rozwiązania bieżącego. Niech $c(Y)$ oznacza koszt rozwiązania częściowego Y, niech c^{opt} będzie kosztem najlepszego rozwiązania bieżącego. W takim przypadku można nie rozważać kandydatów C_j którzy spełniają następującą nierówność.

$$c(Y)+c_j \geqslant c^{opt}$$

Przypuśćmy, że kolumną, która ma być dodana do rozwiązania częściowego jest jedna z kolumn C_r, C_{r+1}, …, C_n. Niech $m_r(Y)$ oznacza liczbę elementów nie pokrytych rozwiązaniem Y tj. $m_r(Y)=m-|M(Y)|$. Każde dopuszczalne powiększenie rozwiązania Y odpowiada rozwiązaniu dopuszczalnemu zagadnienia:

$$z_r=\sum_{j=r}^{n} c_j y_j$$

$$\sum_{j=r}^{n} h_j y_j = m_r(Y), y_j = 0 \, lub \, 1, j=r, r+1, \ldots, n$$

Tak naprawdę chcemy wiedzieć czy rozwiązanie powyższego problemu spełnia nierówność

$$c(Y) + z_r^{opt} \geqslant c^{opt}$$

Jeśli tak to dalsze ruchy do przodu nie mają sensu i mogą zostać zaniechane, należy więc przystąpić do cofania się. Ponieważ jednak znalezienie *z* optymalnego nie jest łatwe zamiast tego trzeba dokonać oszacowania dolnego optymalnego z. Można skorzystać z różnych oszacowań w tym przypadku skorzystamy z oszacowania wyrażonego najmniejszym stosunkiem $c_l/h_l=min\{c_j/h_j: r<=j<=n\}$.
Niech

$$ch_r = min\left\{c_j/h_j : r \leqslant j \leqslant n\right\} dla\, r = 1,2,\ldots,n$$
$$mc_i = min\left\{c_j : j \in B_i\right\} dla\, i = 1,2,\ldots,m$$

Jeśli i jest pierwszym (najmniejszym) elementem nie pokrytym rozwiązaniem częściowym Y, i spełniony jest jeden z poniższych warunków to można natychmiast przejść do cofania

$$c(Y) + mc_i \geqslant c^{opt}$$
$$c(Y) + m(Y)ch_k \geqslant c^{opt}$$

Gdzie *k* jest wskaźnikiem pierwszej kolumny w B_i

Jeśli możliwe jest przeszukiwanie w bloku B_i to opuszczamy takie kolumny C_j dla których spełnione są poniższe warunki

$$c(Y) + c_j \geqslant c^{opt}$$
$$c(Y) + c_j + (m(Y) - h_j) ch_{j+1} \geqslant c^{opt}$$

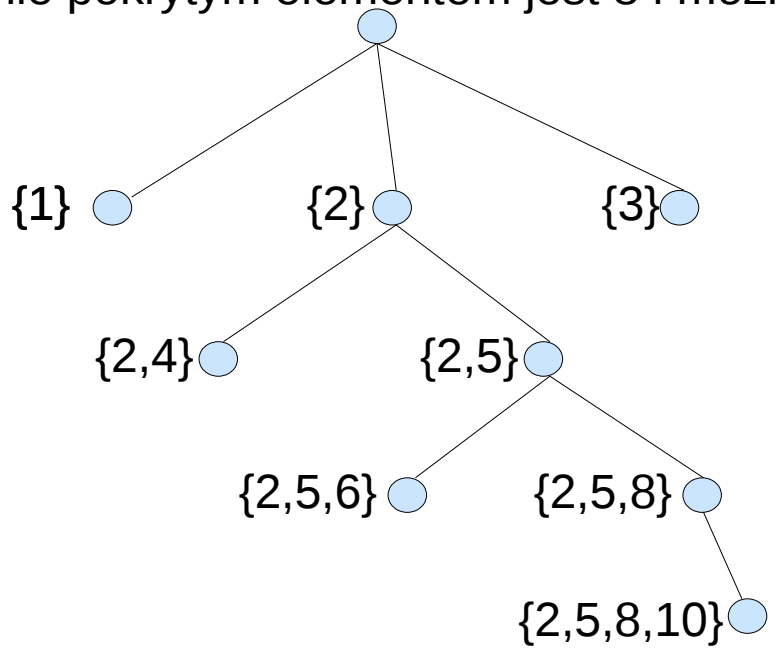
```
begin
c^opt ← inf ; nonfeasible ← true ;
if nie istnieje wiersz zerowy then
begin¿   Y ← Ø ; c(Y) ← 0 ; m(Y) ← m ;
   FORWARDMOVE(Y, 1, i, j, move)
   while move do
    begin
     repeat
      repeat
       znaleźć kolumnę C_j'(j' ⩾ j) w bloku B_i, która nie pokrywa ponownie rozwiązania Y i dla której c(Y) + c_j' < c^opt
       if kolumna taka nie istnieje then BACKWARDMOVE(Y, i, j, move)
      until(not move) ∨ (C_j' istnieje)
      if C_j' istnieje i spełnia warunek 4 then j ← j' + 1
     until(not move) ∨ ((C_j' istnieje) ∧ (C_j' nie spełnie warunku 4))
     if (C_j' istnieje) ∧ (C_j' nie spełnia warunku 4)
      FORWARDMOVE(Y, j', i, j, move)
    end
   end pierwotne zagadnienie nie ma wiersza zerowego
end
```

```
FORWARDMOMVE(Y, j1, i, j2, move)
begin
  Y ← Y ∪ {j1}
  c(Y) ← c(Y) + cj1 ; m(Y) ← m(Y) − hj1
  if m(Y) = 0 then
   begin
    nonfeasible ← false
    uaktualnić naljepsze rozwiązanie
    BACKWARDMOVE(Y, i, j2, move)
   end
   else
   begin
    znaleźć najmniejszy nie pokryty element i
    if bieżące nie pokryte elementy nie mogą być pokryte kolumnami bloków Bl (l ≥ i) then BACKWARDMOVE(T, i, j2, move)
     else if (warunek 1 lub 2) then BACKWARDMOVE(Y, i, j2, move)
    else move ← true
   end
  end
```

Procedura BACKWARDMOVE(Y,i,j,move) przenosi poszukiwanie wstecz do pierwszego Bloku B_i, który zawarty jest w rozwiązaniu częściowym Y i ma niebadaną kolumnę C_j Zmienna logiczna move przyjmuje wartość true, gdy kolumna taka istnieje i wartość false w przeciwnym razie

Przykład

Po inicjalizacji zostaje wywołana procedura FORWARDMOVE. Przydziela Y ← {1} i stwierdza że nie można powiększyć takiego rozwiązania częściowego (element 4), dlatego wywołana zostaje procedura BACKWARDMOVE. Następnie Y ← {2}, kolejny ruch Y ← {2,4}, pierwszy test dopuszczalności że element 6 nie może zostać pokryty, więc jest ruch wstecz po czym w przód Y ← {2,5}. Pierwszym nie pokrytym elementem jest 3, kolumna C6 jest dopuszczalnym kandydatem, Y ← {2,5,6}. Pierwszym nie pokrytym Elementem jest 5, ale jedyna kolumna B5 jest niedopuszczalna następuje więc cofnięcie Y ← {2,5,8}. Jedynym nie pokrytym elementem jest 8 i można go pokryć kolumna C_{10}.



Po uaktualnieniu BACKWARDMOVE wraca do pierwszego bloku jednak Y ← {3} jest niedopuszczalny

Optymalizacja nieliniowa bez ograniczeń

Rozważmy model liniowy z jednym wejściem i jednym wyjściem:

$$y = ax + b$$

gdzie x – sygnał wejściowy, y – sygnał wyjściowy z układy, natomiast a i b są nieznanymi parametrami modelu.
Niech będzie dany ciąg par liczbowych $\{x_i, y_i\}$ (i=1,..., N)
Poszukiwane parametry a, b znajdowane są poprzez minimalizację funkcji

$$\min_{(a,b)} f(a,b) = \sum_{i=1}^{N} (y_i - ax_i - b)^2$$

Funkcja f(a,b) reprezentuje miarę odchyleń między wyjściem modelu i wielkościami obserwowanymi. Jak widać funkcja celu jest kwadratowa. Zaprezentowana metoda, w której minimalizowane są odchylenie zadanych punktów doświadczalnych od zależności funkcyjnej nazywana jest metodą najmniejszych kwadratów.

W ogólności w optymalizacji nieliniowej celem jest znalezienie minimum funkcji wielu zmiennych

$$\min_{x \in R^n} f(\boldsymbol{x}), gdzie\, f : R^n \rightarrow R^1$$

Można wyróżnić 4 rodzaje minimów: minimum lokalne, globalne, ścisłe minimum lokalne, ścisłe minimum globalne.

Def.

Punkt $\hat{x}$ jest minimum lokalnym funkcji f w przestrzeni R^n, jeżeli istnieje takie otoczenie $U \subset R^n$ punktu $\hat{x}$, że

$$f(\boldsymbol{x}) \geqslant f(\hat{\boldsymbol{x}}), \forall\, x \in U$$

Def.

Punkt $\hat{x}$ jest ścisłym minimum lokalnym funkcji f w przestrzeni R^n, jeżeli istnieje takie otoczenie $U \subset R^n$ punktu $\hat{x}$, że

$$f(\boldsymbol{x}) > f(\hat{\boldsymbol{x}}), \forall\, \boldsymbol{x} \in U, \boldsymbol{x} \neq \hat{\boldsymbol{x}}$$

Def.

Punkt $\hat{x}$ jest minimum globalnym funkcji f w przestrzeni R^n, jeżeli

$$f(\boldsymbol{x}) \geqslant f(\hat{\boldsymbol{x}}), \forall \boldsymbol{x} \in R^n$$

Def.

Punkt jest ścisłym minimum globalnym funkcji f w przestrzeni R^n, jeżeli

$$f(\boldsymbol{x}) > f(\hat{\boldsymbol{x}}), \forall\, \boldsymbol{x} \in R^n, \boldsymbol{x} \neq \hat{\boldsymbol{x}}$$

Warunki optymalności

Warunek konieczny dla funkcji różniczkowalnych

Tw.

Jeśli $\hat{\boldsymbol{x}}$ jest minimum lokalnym funkcji f, która jest ciągle różniczkowalna w pewnym otwartym otoczeniu punktu $\hat{\boldsymbol{x}}$, to pierwsza pochodna f w punkcie $\hat{\boldsymbol{x}}$ ma wartość **0**

$$\nabla f(\hat{\boldsymbol{x}}) = \boldsymbol{0}$$ warunek I rzędu

Jeżeli ponadto funkcja f jest dwukrotnie różniczkowalna, to macierz drugich pochodnych jest dodatnio określona

$$\boldsymbol{d}^T \nabla^2 f(\hat{\boldsymbol{x}}) \boldsymbol{d} \geqslant 0, \forall d \in R^n$$ warunek II rzędu

Warunek dostateczny

Tw.

Niech funkcja f będzie dwukrotnie ciągle różniczkowalna w pewnym otoczeniu punktu $\hat{\boldsymbol{x}}$ i spełnione są następujące warunki

$$\nabla f(\hat{\boldsymbol{x}}) = \boldsymbol{0}$$

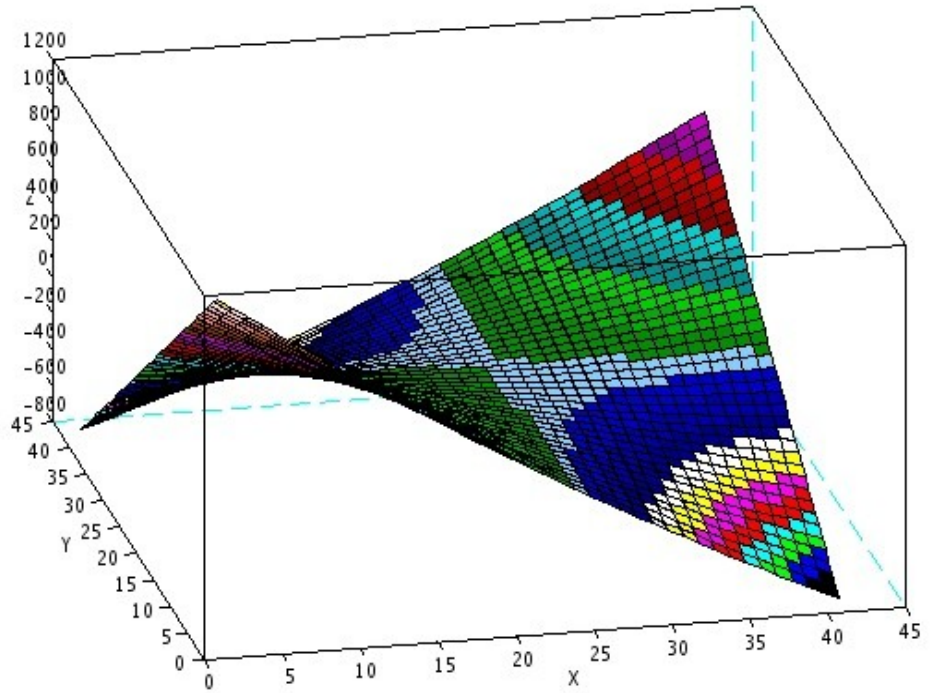
$$\boldsymbol{d}^T \nabla^2 f(\hat{\boldsymbol{x}}) \boldsymbol{d} > 0, \forall d \in R^n \left(\textit{hesjan jest ściśle dodatnio określony} \right)$$

Wówczas punkt $\hat{x}$ jest punktem ścisłego lokalnego minimum funkcji f oraz w pewnym, dostatecznie małym otoczeniu punktu $\hat{x}$ prawdziwe jest następujące oszacowanie

$$f(\boldsymbol{x}) \geqslant f(\hat{\boldsymbol{x}}) + c_1 \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$$

Punkty stacjonarne

Każdy punkt lokalnego minimum funkcji ciągle różniczkowalnej jest punktem stacjonarnym minimalizowanej funkcji celu tzn. punktem w którym zeruje się pierwsza pochodna.
Nie każdy punkt stacjonarny jest punktem lokalnego minimum!



W przypadku funkcji 2 krotnie różniczkowalnych punkt minimum odpowiada dodatnio określonej macierzy hesjanu, punkt maksimum ujemnie określonej macierzy hesjanu i punkt siodłowy macierzy nieokreślonej.

Metody gradientowe

- Punktem wyjściowym jest aktualne przybliżenie rozwiązania $\boldsymbol{x}_k$.
- Określenie kierunku poszukiwań $\boldsymbol{d}_k$.
- Znalezienie α_k minimalizującego $f(\alpha)=f(\boldsymbol{x}_k+\alpha\boldsymbol{d}_k)$ ze względu na α
- Nowym przybliżeniem punktu optymalnego jest $\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\alpha_k$

Poszczególne metody różnią się przede wszystkim: sposobem wyznaczania kierunku poszukiwań $\boldsymbol{d}^k$, wyborem sposobu minimalizacji kierunkowej i stosowanym kryterium stopu.

Kierunki poprawy

Jest to kierunek wzdłuż którego, po wykonaniu przesunięcia z dostatecznie małym dodatnim współczynnikiem, uzyskana zostanie lepsza czyli mniejsza wartość minimalizowanej funkcji
Zmniejszenie wartości funkcji celu przy małych dodatnich przesunięciach może wystąpić gdy

$$f'(\alpha)=\nabla f(\boldsymbol{x}_k+\alpha\boldsymbol{d}_k)^T\boldsymbol{d}_k$$

$$f'(0)=\nabla f(\boldsymbol{x}_k)^T\boldsymbol{d}_k\leqslant 0$$

Wynika z rozwinięcia w szereg Taylora funkcji $f(\alpha)$ w punkcie $\boldsymbol{x}_k$

Warunek ten definiuje kierunki poprawy. Można go zinterpretować geometrycznie w Następujący sposób:

Kierunek $\boldsymbol{d}_k$ tworzy z kierunkiem $-\nabla f(\boldsymbol{x}_k)$ kąt mniejszy lub równy 90.

Jednostajne kierunki poprawy

Generowane kierunki są jednostajnymi kierunkami poprawy gdy, we wszystkich iteracjach metody kąt Θ^k między wektorami $-\nabla f(\boldsymbol{x}_k)$ i $\boldsymbol{d}_k$ jest kątem ostrym, jednostajnie mniejszym od kąta 90 tj. dla pewnego $\mu>0$

$$0 \leqslant \Theta_k \leqslant \frac{\pi}{2} - \mu, \forall k$$

Co można zapisać tak

$$\frac{(\boldsymbol{g}_k)^T \boldsymbol{d}_k}{\|\boldsymbol{g}_k\|\|\boldsymbol{d}_k\|} \leqslant -\kappa, \forall k$$

Dla pewnej stałej $\kappa>0$, $\boldsymbol{g}_k=\nabla f(\boldsymbol{x}_k)$. Równoważność warunków wynika z własności funkcji Cosinus oraz związku definiującego iloczyn skalarny wektorów

$$-(\boldsymbol{g}_k)^T \boldsymbol{d}_k = \cos(\Theta_k)\|\boldsymbol{g}_k\|\|\boldsymbol{d}_k\|$$

Jednostajność kierunków poprawy zapewnia zbieżność algorytmu!

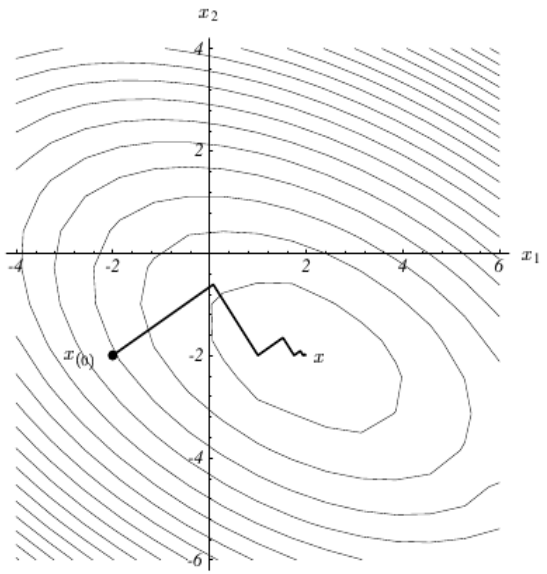
W celu zapewnienia zbieżności metody wykorzystującej pochodne konieczne jest zachowanie własności jednostajności ale również spełnienie odpowiednich warunków w minimalizacji kierunkowej.
Jeśli funkcja celu jest różniczkowalna, to dokładna minimalizacja wzdłuż kierunku oznacza punktu w którym zeruje się pochodna kierunkowa

$$\nabla f(\boldsymbol{x}_{k+1})^T \boldsymbol{d}_k = 0$$

Metoda gradientu prostego

Najprostszą z metod gradientowych jest metoda najszybszego spadku, w której kierunek

$$\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k) \qquad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k)$$



Powstający zygzak jest skutkiem tego że każdy gradient
Jest prostopadły do poprzednika

$$f(\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k))$$

$$0 = \frac{\partial f(\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k))}{\partial \alpha} = \nabla f(\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k))(-\nabla f(\boldsymbol{x}_k))$$

Metoda najszybszego spadku

- Zacznij od dowolnego punktu $\boldsymbol{x}_k$ k=0
- Wyznacz kierunek poszukiwania

$$\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k)$$

- Wyznacz długość kroku

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

Sprawdź czy punkt $\boldsymbol{x}^{k+1}$ jest optymalny. Jeżeli tak to koniec, jeżeli nie zwiększ numer iteracji k=k+1 i idź do kroku 2

Przykład

Punkt startowy (0,0)

$$f(\boldsymbol{x}) = x_1 - x_2 + 2 * x_1 * x_2 + x_2^2$$

$$f'(\boldsymbol{x}) = \begin{bmatrix} 1 + 2 * x_2 \\ -1 + 2 * x_1 + 2 * x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$f(\boldsymbol{x}_0 - \alpha f'(0)) = -\alpha - \alpha - 2 * \alpha^2 + \alpha^2 = -\alpha^2 - 2 * \alpha$$

$$\alpha^{opt} = -1$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \alpha^{opt} * f'(\boldsymbol{x}_0) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Kryterium stopu

1 Zmiana wartości funkcji celu w kolejnej iteracji jest wystarczająco mała

$$\left|\frac{f(x_{k+1})-f(x_k)}{f(x_k)}\right|<\varepsilon_1$$

2 Pochodne cząstkowe są wystarczająco małe

$$\frac{\partial f}{\partial x_i}<\varepsilon_2$$

3 Zmiana położenia punktu jest wystarczająco mała

$$\left|\frac{\boldsymbol{x}_{k+1}}{\boldsymbol{x}_k}\right|<\varepsilon_3$$

Wady metody: wolno zbieżna, szczególnie w pobliżu punktu optymalnego (wtedy gradient jest mały).

Zbieżność

Jeżeli zastosować metodą najszybszego spadku do minimalizacji funkcji kwadratowej o ściśle dodatnio określonej macierzy drugich pochodnych, to można tak dobrać punkt początkowy $\boldsymbol{x}_0$, że następujące oszacowanie może być spełnione równościowo

$$f(\boldsymbol{x}_{i+1}) - f(\hat{\boldsymbol{x}}) \leqslant \left(\frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}} \right)^2 (f(\boldsymbol{x}_i) - f(\hat{\boldsymbol{x}}))$$

Gdzie $\boldsymbol{x}$ jest punktem optymalnym, λ_{max} jest największą wartością własną, a λ_{min} najmniejszą wartością własną. Oznacza to, że szybkość zbieżności metody jest liniowa. Współczynnik zbieżności

$$\left(\frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}} \right)^2 = \left(\frac{1 - \frac{1}{K}}{1 + \frac{1}{K}} \right)^{2,} \quad gdzie\ K = \frac{\lambda_{max}}{\lambda_{min}}$$

Zbliża się do wartości 1, gdy zwiększa się wskaźnik uwarunkowania zadania K. Metoda jest tym wolniejsza im gorszy jest wskaźnik uwarunkowania.

Metoda Newtona

$$f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^T \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$$

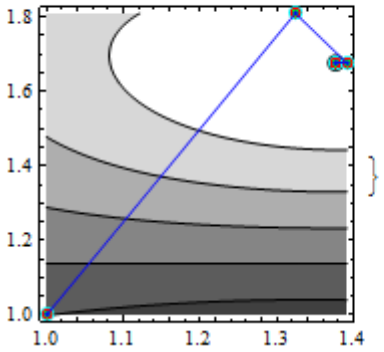
W punkcie optymalnym żądamy aby pochodna była równa zero

$$\nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = 0$$

$$x_{k+1} = x_k - \frac{\nabla f(\boldsymbol{x}_k)}{\nabla^2 f(\boldsymbol{x}_k)} \quad czyli \quad \boldsymbol{d}_k = \frac{\nabla f(\boldsymbol{x}_k)}{\nabla^2 f(\boldsymbol{x}_k)}$$

Jeśli macierz $\nabla^2 f(\boldsymbol{x}_k)$ jest dodatnio określona w otoczeniu minimum lokalnego. Metoda Newtona jest więc dobra wtedy kiedy punkt startowy jest w pobliżu minimum lokalnego. Poza tym metoda jest bardzo kosztowna obliczeniowo $O(n^3)$.

```
{{-2., {x → 1.37638, y → 1.67867}},
 {Steps → 4, Function → 89, Gradient → 26, Hessian → 5},
```



}

Zbieżność

Jeśli minimalizowana funkcja jest dwukrotnie różniczkowalna (w sposób ciągły) , elementy Hesjanu spełniają warunek Lipschitza tzn. istnieje stała liczbowa K taka że

$$|G(\boldsymbol{x})-G(\boldsymbol{y})|\leqslant K\|\boldsymbol{x}-\boldsymbol{y}\|$$

Dla wszystkich ***x*** i ***y***, w pewnym otoczeniu punktu optymalnego $\hat{\boldsymbol{x}}$, macierz Jest ściśle dodatnio określona i punkt początkowy $\mathbf{x}^0$ jest dostatecznie blisko punktu optymalnego, to metoda Newtona jest dobrze określona dla wszystkich *k* i zbiega do punktu optymalnego z szybkością drugiego rzędu (kwadratową). Jeśli odległość od punktu optymalnego jest rzędu 10^{-2}, to w następnej iteracji można oczekiwać, że odległość ta będzie rzędu 10^{-4}, czyli można uzyskać w przybliżeniu podwajanie się dokładności w kolejnych iteracjach.

Metody quasi-newtonowskie

W metodach quasi-newtonowskich kierunek poszukiwań wyznaczany jest podobnie jak w metodzie Newtona

$$\boldsymbol{d}_k = -\boldsymbol{H}_k \nabla f(\boldsymbol{x}_k) \; albo \; \boldsymbol{d}_k = -\boldsymbol{B}_k^{-1} \nabla f(\boldsymbol{x}_k)$$

Gdzie $\boldsymbol{H}_k$ jest aproksymacją odwrotności hesjanu minimalizowanej funkcji w punkcie $\boldsymbol{x}_k$, Natomiast $\boldsymbol{B}_k$ aproksymacją samej macierzy pochodnych w punkcie $\boldsymbol{x}^k$.
Po wyznaczeniu punktu $\boldsymbol{x}_{k+1}$ modyfikuje się odpowiednio macierz $\boldsymbol{H}_k$ ($\boldsymbol{B}_k$), wykorzystując przyrosty zmiennych $\boldsymbol{s}_k=\boldsymbol{x}_{k+1}-\boldsymbol{x}_k$ oraz pochodnych $\boldsymbol{r}_k=\nabla f(\boldsymbol{x}_{k+1})-\nabla f(\boldsymbol{x}_k)$. Modyfikacja polega na dodaniu do poprzedniej aproksymacji odpowiedniej poprawki

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \boldsymbol{U}_k$$

Poprawka $\boldsymbol{U}_k$ jest konstruowana w taki sposób, by kolejna macierz $\boldsymbol{H}_{k+1}$ ($\boldsymbol{B}_{k+1}$) spełniała tzw. warunek quasi-newtonowski.

$$\boldsymbol{H}_{k+1}\boldsymbol{r}_k = \boldsymbol{s}_k, \quad \boldsymbol{B}_{k+1}\boldsymbol{s}_k = \boldsymbol{r}_k$$

Najpopularniejsze metody obliczania przybliżeń

Metoda Wolfe'a, Broydena, Davidona

$$H_{k+1} = H_k + \frac{(s_k - H_k r_k)(s_k - H_k r_k)^T}{(s_k - H_k r_k)^T r_k}$$

Metoda Broydena

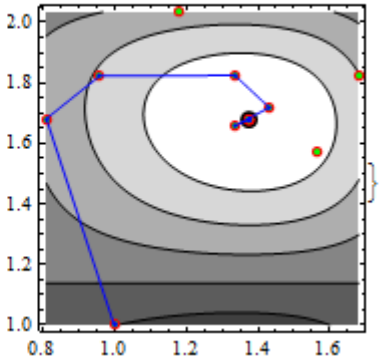
$$H_{k+1} = H_k + \frac{(s_k - H_k r_k)}{r_k^T r_k} r_k^T$$

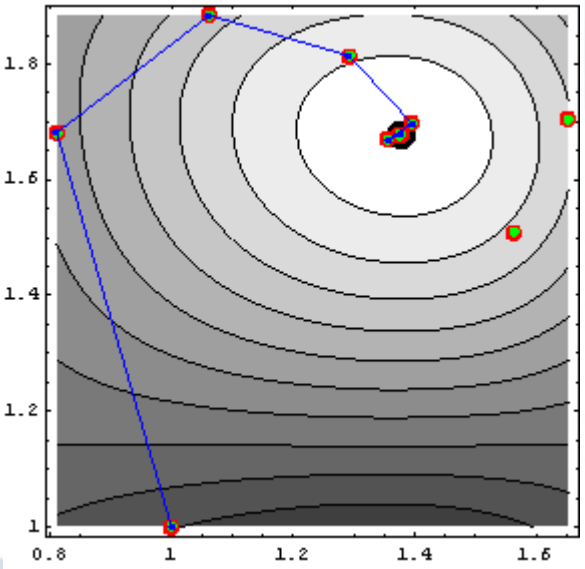
Metoda BFGS

$$H_{k+1} = H_k + \frac{(s_k s_k^T)}{s_k^T r} - \frac{(H_k r_k)(H_k r_k)^T}{r_k^T H_k r_k}$$

```
{{-2., {x → 1.37638, y → 1.67868}},

 {Steps → 9, Function → 13, Gradient → 13},
```





Metody sprzężonych kierunków

Kierunki

$$\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3, \ldots, \boldsymbol{d}_n; \boldsymbol{d}_i \neq 0$$

Są G sprzężone (ortogonalne) względem symetrycznej, ściśle dodatnio określonej macierzy G, jeśli

$$\boldsymbol{d}_i^T \boldsymbol{G}\, \boldsymbol{d}_j = 0, \forall i \neq j$$

Załóżmy że chcemy znaleźć minimum formy kwadratowej

$$F(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{x}^T \boldsymbol{G}\, \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x}$$

Ponieważ macierz G jest nxn, dodatnio kreślona to wektory sprzężone są liniowo niezależne
Jeżeli by tak nie było tzn. że

$$\alpha_1 \boldsymbol{d}_1 + \alpha_2 \boldsymbol{d}_2 + \ldots + \alpha_n \boldsymbol{d}_n = 0; \alpha_i = 0$$

$$\alpha_i \boldsymbol{d}_i^T \boldsymbol{G}\, \boldsymbol{d}_i = 0$$

Tymczasem z definicji dodatniej określoności $\boldsymbol{d}_i^T \boldsymbol{G}\, \boldsymbol{d}_i > 0 \; czyli \; \alpha_i = 0$

Po co ***G*** ortogonalność?
Załóżmy, że mamy zbiór wektorów ***G*** ortogonalny, wówczas rozwiązanie minimalizacyjne naszej formy kwadratowej możemy zapisać w przestrzeni rozpiętej przez te wektory

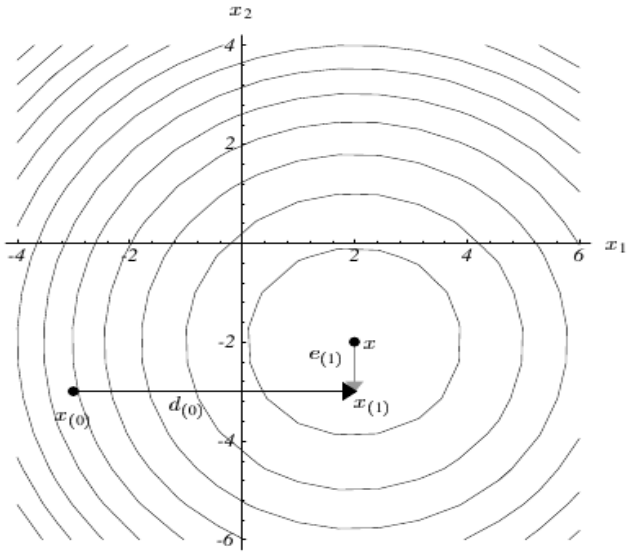
$$\boldsymbol{x}^{opt} = \alpha_1 \boldsymbol{d}_1 + \alpha_2 \boldsymbol{d}_2 + \ldots + \alpha_n \boldsymbol{d}_n$$

Stąd

$$\alpha_i = \frac{\boldsymbol{d}_i^T \boldsymbol{b}}{\boldsymbol{d}_i^T \boldsymbol{G} \boldsymbol{d}_i} \qquad \boldsymbol{b} = \boldsymbol{x}^{opt} \boldsymbol{G} = \sum_{i=1}^{n} \alpha_i \boldsymbol{G} \boldsymbol{d}_i$$

A więc optymalne rozwiązanie można przedstawić tak

$$\boldsymbol{x}^{opt} = \sum_{i=1}^{n} \frac{\boldsymbol{d}_i^T \boldsymbol{b}}{\boldsymbol{d}_i^T \boldsymbol{G} \boldsymbol{d}_i} \boldsymbol{d}_i$$



Całkowita optymalizacja musi zajęć tylko n kroków, w każdym wymiarze tylko jeden krok.
A więc mając kierunki sprzężone w łatwy sposób możemy wyznaczyć optymalne rozwiązanie pozostaje tylko znaleźć te kierunki :-).

Wszystko czego potrzebujemy to metoda pozwalająca generować kierunki ortogonalne, metoda ta nazywana jest ortogonalizacją Grama-Schmidta. Pewną wadą jest to, że muszą być pamiętane wszystkie poprzednie wektory! Złożoność wygenerowania wszystkich wektorów jest $O(n^3)$.

Ortogonalizacja Grama-Schmidta

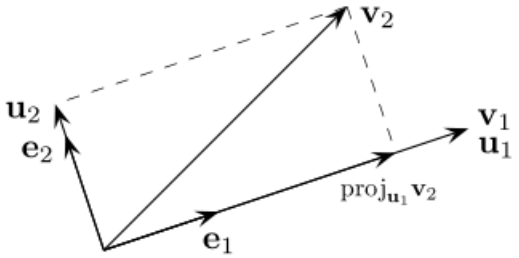
$$proj_{\vec{u}}\,\vec{v} = \frac{\langle \vec{u} | \vec{v} \rangle}{\langle \vec{u} | \vec{u} \rangle}\,\vec{u}$$

$$\vec{u}_1 = \vec{v}_1$$

$$\vec{u}_2 = \vec{v}_2 - proj_{\vec{u}_1}\,\vec{v}_2$$

$$\vec{u}_3 = \vec{v}_3 - proj_{\vec{u}_2}\,\vec{v}_3 - proj_{\vec{u}_1}\,\vec{v}_3$$

$$\vec{u}_k = \vec{v}_k - \sum_{i=1}^{k-1} proj_{\vec{u}_i}\,\vec{v}_k$$



Metoda sprzężonych gradientów

Dla formy kwadratowej $F(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{G}\,\boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x}$

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \alpha_i \boldsymbol{d}_i \qquad \alpha_i = \frac{\boldsymbol{g}_i^T \boldsymbol{b}}{\boldsymbol{d}_i^T \boldsymbol{G}\,\boldsymbol{d}_i}$$

$$\boldsymbol{g}_i = \boldsymbol{G}\,\boldsymbol{x}_i - \boldsymbol{b}$$

$$\boldsymbol{d}_{i+1} = -\boldsymbol{g}_i + \sum_{k=0}^{i} \beta_{ki} \boldsymbol{d}_k$$

Żeby wyznaczyć parametr beta wystarczy pomnożyć równanie

$$\boldsymbol{d}_{i+1}^T \boldsymbol{G}\,\boldsymbol{d}_j = -\boldsymbol{g}_i^T \boldsymbol{G}\,\boldsymbol{d}_j + \sum_{k=0}^{i} \beta_{ki} \boldsymbol{d}_k^T \boldsymbol{G}\,\boldsymbol{d}_j$$

$$0 = -\boldsymbol{g}_i^T \boldsymbol{G}\,\boldsymbol{d}_j + \beta_{ij} \boldsymbol{d}_j^T \boldsymbol{G}\,\boldsymbol{d}_j$$

$$\beta_{ii+1} = \beta_i = \frac{\boldsymbol{g}_i^T \boldsymbol{G}\,\boldsymbol{d}_i}{\boldsymbol{d}_i^T \boldsymbol{G}\,\boldsymbol{d}_i}$$

W przypadku gdy mamy do czynienia z funkcją, która nie jest kwadratowa, może być ona lokalnie aproksymowana przez funkcje kwadratową

Istnieją inne metody wyznaczania parametru beta

Metoda sprzężonych gradientów
W wersji Fletchera-Reevesa

$$\beta_i = \frac{\boldsymbol{g}_{i+1}^T \boldsymbol{g}_{i+1}}{\boldsymbol{g}_i^T \boldsymbol{g}_i}$$

Metoda sprzężonych gradientów
W wersji Polaka-Ribiere'a

$$\beta_i = \frac{(\boldsymbol{g}_{i+1} - \boldsymbol{g}_i)^T \boldsymbol{g}_{i+1}}{\boldsymbol{g}_i^T \boldsymbol{g}_i}$$

Trudno wskazać jakieś argumenty przemawiające za wyborem, któregoś z podanych wzorów. W praktyce podane wzory są równo często stosowane.

Testy zatrzymania algorytmu

- Przybliżona stacjonarność rozwiązania.

$$\left\| \boldsymbol{g}_k \right\| \leq \epsilon$$

W teście nie wiadomo jak wybrać próg, może też nie sprawdzić się w zadaniach źle uwarunkowanych

- Tzw. test praktyczny

$$|x_{ji+1} - x_{ji}| < \epsilon_j \; lub$$
$$f_k - f_{k+1} < \epsilon_1$$

- Ocena przewidywanej zmiany wartości funkcji f np.. wyrażona wzorem

$$\frac{1}{2} \boldsymbol{g}_i^T \boldsymbol{H}_i \boldsymbol{g}_i < \epsilon$$

Gdzie $\mathbf{H}^k$ jest aproksymacyjną odwrotnością hesjanu.

Złożoność algorytmu $O(m\sqrt{k})$, gdzie m liczba niezerowych elementów macierzy G
Teoretyczne wyniki dotyczące zbieżności metody i jej szybkości nie gwarantują dobrego działania w praktyce ponieważ:

- Same rezultaty teoretyczne nie dają pełnej gwarancji akceptowalnego przebiegu realizowanego algorytmu
- W rezultatach teoretycznych pomijany jest zazwyczaj wpływ błędów zaokrągleń
- Typ zbieżności związany jest z asymptotycznymi własnościami metody, a o praktycznej skuteczności decyduje zwykle jego zachowanie początkowe

Metody bezgradientowe

Typowym założeniem dla metod bezgradientowych jest założeniu o unimodalności funkcji $f(\alpha)=f(\boldsymbol{x}_k+\alpha\boldsymbol{d}_k)$

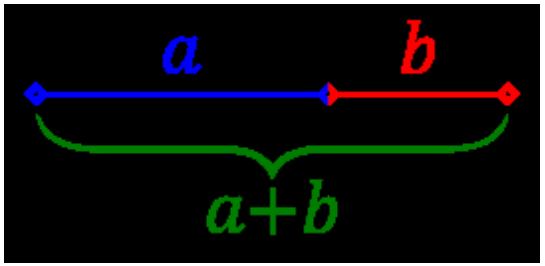
Funkcja jest unimodalna w przedziale [a,b] tzn. istnieje punkt $\hat{\alpha}\in[a,b]$ taki że

$$jeśli\ \alpha_1<\alpha_2,\ \alpha_1,\alpha_2\in[a,b]\ wówczas$$
$$f(\alpha_1)>f(\alpha_2)\ dla\ \alpha_2<\hat{\alpha}$$
$$f(\alpha_1)<f(\alpha_2)\ dla\ \alpha_1>\hat{\alpha}$$

Czyli w przedziale [a,b] musi istnieć jedynie ścisłe minimum, a ponadto na lewo od tego punktu funkcja musi być malejąca a na prawo rosnąca.

Bezgradientowa metoda złotego podziału

Złoty podział (łac. sectio aurea), podział harmoniczny, złota proporcja, boska proporcja (łac. divina proportio) – podział odcinka na dwie części tak, by stosunek długości dłuższej z nich do krótszej był taki sam, jak całego odcinka do części dłuższej.



$$\frac{a+b}{a}=\frac{a}{b}=\frac{1}{\tau}$$

$$\frac{a+b}{a}=1+\frac{b}{a}=1+\tau=\frac{1}{\tau}$$

$$\tau^2+\tau-1=0$$

Bezgradientowa metoda złotego podziału

Metoda ta opiera się na zasadzie, że w każdej iteracji podprzedział zawierający poszukiwane minimum będzie się zmniejszał o stały współczynnik oraz jeden z dwu pomocniczych punktów wewnętrznych, które są wykorzystywane, zostaje zachowany w kolejnej iteracji.

Inicjalizacja Wybierz dopuszczalną dokładność końcową ε>0. Niech

$$\alpha_1^k = b^1 - \tau(b^1 - a^1) = a^1 + (1-\tau)(b^1 - a^1)$$
$$\alpha_2^k = a^1 + \tau(b^1 - a^1)$$
$$gdzie\ \tau = 0.618$$
$$Oblicz\ f(\alpha_1^k)\ i\ f(\alpha_2^k)$$

Krok główny

$$1\ Jeśli\ b^k - a^k < \epsilon \rightarrow STOP$$
$$w\ przeciwnym\ przypadku,\ jeśli:$$
$$f(\alpha_1^k) > f(\alpha_2^k)\ idź\ do\ kroku\ 2$$
$$f(\alpha_1^k) \leqslant f(\alpha_2^k)\ idź\ do\ kroku\ 3$$

$2)\ a^{k+1}=\alpha_1^k,\ b^{k+1}=b^k$

$\alpha_1^{k+1}=\alpha_2^k$

$\alpha_2^{k+1}=a^{k+1}+\tau\left(b^{k+1}-a^{k+1}\right)$

$Oblicz\ F\left(\alpha_2^{k+1}\right)\ idź\ do\ kroku\ 4$

$3)\ a^{k+1}=a^k,\ b^{k+1}=\alpha_2^{k+1}$

$\alpha_2^{k+1}=\alpha_1^k$

$\alpha_1^{k+1}=a^{k+1}+(1-\tau)\left(b^{k+1}-a^{k+1}\right)$

$Oblicz\ f\left(\alpha_1^{k+1}\right)\ idź\ do\ 4$

$4)\ k=k+1,\ wróć\ do\ 1$

Zbieżność metody jest liniowa. Ilość iteracji N potrzebna do zawężenia przedziału początkowego [a,b] do zadanej dokładności wynosi:

$$N=\log_\tau \frac{\epsilon}{b-a}$$

Bezgradientowa metoda aproksymacji kwadratowej

W każdym kroku wykorzystuje się informacje o wartościach funkcji $f(\mathbf{x}_k+\alpha d_k)$ w trzech punktach f_a, f_b, f_c w punktach α_a, α_b, α_c takich że

$$\alpha_a < \alpha_b < \alpha_c$$

Do budowy aproksymacji parabolicznej można zastosować różne metody, jednak bardzo Wygodnie jest zastosować interpolację Lagrange'a postaci

$$\phi(\alpha)=f_a\frac{(\alpha-\alpha_b)(\alpha-\alpha_c)}{(\alpha_a-\alpha_b)(\alpha_a-\alpha_c)}+f_b\frac{(\alpha-\alpha_a)(\alpha-\alpha_c)}{(\alpha_b-\alpha_a)(\alpha_b-\alpha_c)}+f_c\frac{(\alpha-\alpha_a)(\alpha-\alpha_b)}{(\alpha_c-\alpha_a)(\alpha_c-\alpha_b)}$$

Kolejny punkt wyznacza się z warunku ekstremum dla funkcji $\varphi(\alpha)$ $\frac{d\phi}{d\alpha}=0$

$$\alpha_m=\frac{1}{2}\frac{(\alpha_b^2-\alpha_c^2)f_a+(\alpha_c^2-\alpha_a^2)f_b+(\alpha_a^2-\alpha_b^2)f_c}{(\alpha_b-\alpha_c)f_a+(\alpha_c-\alpha_a)f_b+(\alpha_a-\alpha_b)f_c}$$

Metoda sympleksu Neldera-Meada

Simpleks w N wymiarowej przestrzeni ma N+1 wierzchołków.

- Wybierz $\gamma>1$, $\beta\epsilon(0,1)$ oraz parametr zakończenia ε.
- Znajdź najgorszy punkt x_w, najlepszy punkt xb oraz drugi w kolejności x_m. Oblicz punkt względem którego będziemy odbijać x_w

$$x_c = \frac{1}{N} \sum_{i=1,\, j \neq h}^{N+1} x_i$$

Oblicz punkt odbicia $x_r=2x_c-x_w$. Ustal $x_{new}=x_r$

Jeśli $f(x_r)<f(x_b)$ $x_{new}=(1+\gamma)x_c-\gamma x_w$ (ekspansja)
Jeśli $f(x_r)>=f(x_w)$ $x_{new}=(1-\beta)x_c+\beta x_w$(kontrakcja)
Jeśli $f(x_m)<f(x_r)<f(x_w)$ $x_{new}=(1-\beta)x_c-\beta x_w$
Wymień x_w na x_{new}

- Zakończ jeśli

$$\left(\frac{\sum_{i=1}^{N+1} \left(f\left(x_i\right)-f\left(x_c\right)\right)^2}{N+1} \right)^{\frac{1}{2}} \leqslant \epsilon$$

Algorytmy genetyczne

- Twórcą algorytmu jest Holland (1962).
- Są oparte na mechanizmach doboru naturalnego i dziedziczenia.
- Nie wymagają liczenia gradientu, a więc nadają się do funkcji nieciągłych
- Należą do metod stochastycznych
- Bardzo łatwe do implementacji
- Łatwe do zrównoleglania

Metody ewolucyjne powstały w celu znajdowania przybliżonego rozwiązania problemów optymalizacyjnych w taki sposób, by znajdować wynik w miarę szybko oraz uniknąć lokalnych minimów.

Podstawowe pojęcia

• Populacja – zbiór osobników (jednostek) o określonej liczebności, których przystosowanie do środowiska jest określone.
• Osobniki populacji w algorytmach genetycznych to zakodowane w postaci chromosomów zbiory parametrów zadania, czyli rozwiązania, określane też jako punkty przestrzeni poszukiwań.
• Chromosomy – inaczej łańcuchy lub ciągi kodowe – to uporządkowane ciągi genów.
• Gen – nazywany też cecha, znakiem, detektorem – jest to pojedynczy element genotypu, w szczególności chromosomu.
• Genotyp, czyli struktura - to zespół chromosomów danego osobnika. Osobnikami populacji mogą być genotypy albo pojedyncze chromosomy.
• Fenotyp - zestaw wartości odpowiadający danemu genotypowi, czyli zdekodowana struktura. Jest to zbiór parametrów zadania (rozwiązanie, punkt przestrzeni poszukiwań).
• Allel to wartość danego genu (określana też jako wartość cechy lub wariant cechy).
• Funkcja przystosowania (ang. fitness function) nazywana też funkcją dopasowania lub funkcją oceny. Stanowi ona miarę przystosowania (dopasowania) danego osobnika w populacji.
• Generacja – to kolejna iteracja w algorytmie genetycznym.
• Pokolenie (nowe pokolenie lub pokolenie potomków) – to nowo utworzona populacja osobników.

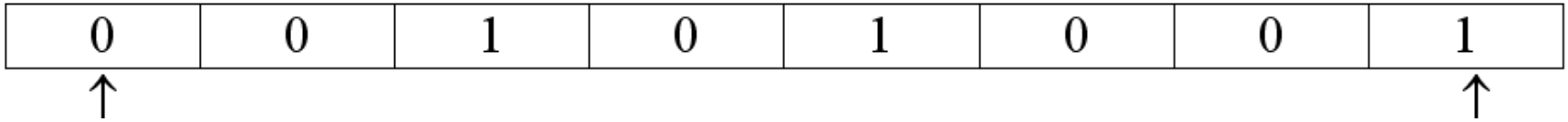
Algorytmy genetyczne

Załóżmy, że mamy czarną skrzynkę z pięcioma przełącznikami na wejściu. Na wyjściu natomiast odbierany jest sygnał f, który jest jakąś funkcją stanu przełączników wejściowych. Zadanie polega na takim ustawieniu przełączników, aby uzyskać największy możliwy poziom sygnału f.
Zakodujmy stany przełączników np. w postaci ciągu binarnego o długości 5, w którym 1 oznacza przełącznik włączony, 0 wyłączony. Czyli 11000, oznacza ustawienie 2 pierwszych przełączników na włączony a pozostałe zera na włączniki wyłączone.
Algorytm genetyczny startuje z początkowej populacji ciągów kodowych a następnie generuje kolejne populacje.
Początkowa populacja generowana jest losowo i niech wygląda tak:
10101
00001
10000
11000
W algorytmie genetycznym kolejne populacje generowane są poprzez zastosowanie następujących operacji:

1) reprodukcja
2) krzyżowanie
3) mutacja

Metody kodowania

Kodowanie binarne



Cecha kodowana na tej pozycji nie występuje w rozwiązaniu

Cecha kodowana na tej pozycji występuje w rozwiązaniu

Allele wszystkich genów w chromosomie są równie 0 lub 1.

Kod Graya

Jest alternatywą dla kodowania binarnego, charakteryzuje się tym że chromosomy odpowiadające dwóm kolejnym liczbom całkowitym różnią się jednym bitem.

Zamiana binarnego na Graya

$g_1=b_1$;
for(k=2;k<m;k++)
    $g_k=b_{k-1}$ xor b_k

Zamiana Graya na binarny

$b_1=g_1$
for(k=2;k<m;k++)
    $b_k=g_k$ xor b_{k-1}

Dec	Gray	Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

Kodowanie logarytmiczne

Ten sposób kodowania jest stosowany w celu zmniejszenia długości chromosomu. W tym kodowaniu pierwszy bit chromosomu oznacza znak funkcji wykładniczej, drugi znak wykładnika funkcji, pozostałe natomiast są reprezentacją wykładnika funkcji.

Goldebrg o kodowaniu:

W pewnym sensie wybór kodu dla zadania rozwiązywanego przy użyciu algorytmu genetycznego nie stanowi żadnego problemu, gdyż programista jest ograniczony głównie swoją własną wyobraźnią.(...) Patrząc z innego punktu widzenia, ta swoboda wyboru stanowi wątpliwe błogosławieństwo dla niedoświadczonego użytkownika; widok całych szeregów możliwych sposobów kodowania może zarówno dodawać otuchy, jak i oszałamiać”.

Reprodukcja

Proces w którym elementy populacji zostają powielone w stosunku zależnym od wartości, jakie przybiera funkcja celu *f* (funkcja przystosowania) inaczej mówiąc przeżywają najlepiej przystosowani.

Niech f_i oznacza dopasowanie *i*-tego osobnika, wówczas do reprodukcji *i*-ty osobnik wybrany zostanie z prawdopodobieństwem p_i

$$p_i = \frac{f_i}{\sum_{i}^{N} f_i}$$

Rodzaje selekcji: turniejowa, rankingowa, elitarna itp.

Selekcja turniejowa

Polega na losowym wyborze z całej populacji kilku osobników, a następnie z puli wybranych osobników wybierany jest ten, który jest najlepiej przystosowany. Cała procedura powtarzana jest tak długo aż osiągnie się zadaną liczbę osobników.

Selekcja rankingowa

Dla każdego osobnika wyliczane jest jego przystosowanie, na tej podstawie osobnicy są sortowani i wybierany jest k najlepszych.

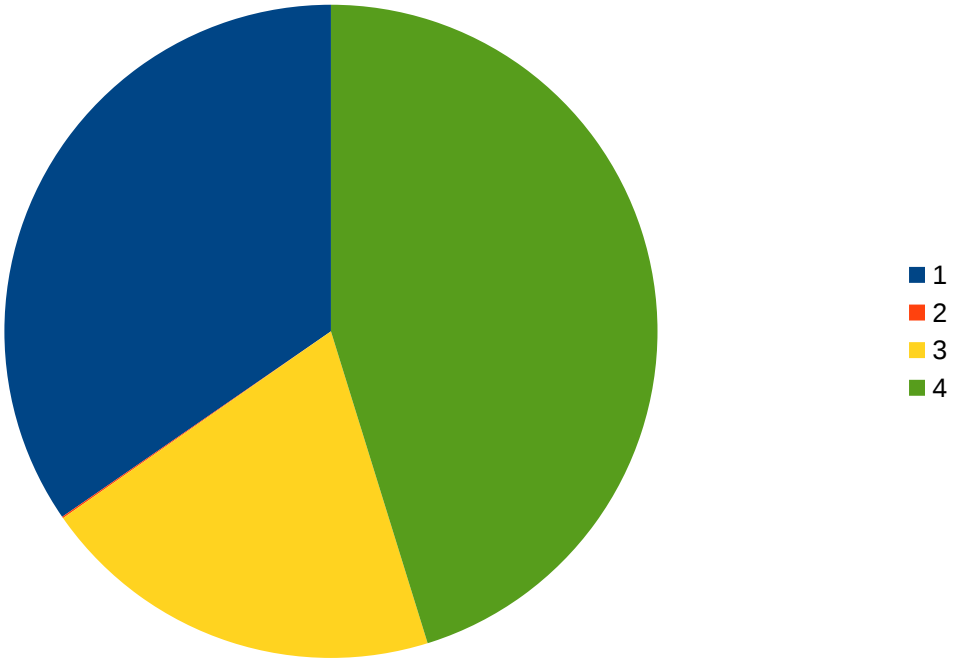
Selekcja elitarna

W selekcji elitarnej do następnej populacji zawsze kopiowany jest najlepszy osobnik, lub k najlepszych osobników. Selekcja tego typu nie jest samodzielna, musi towarzyszyć innemu typowi selekcji.

Selekcja typu ruletka

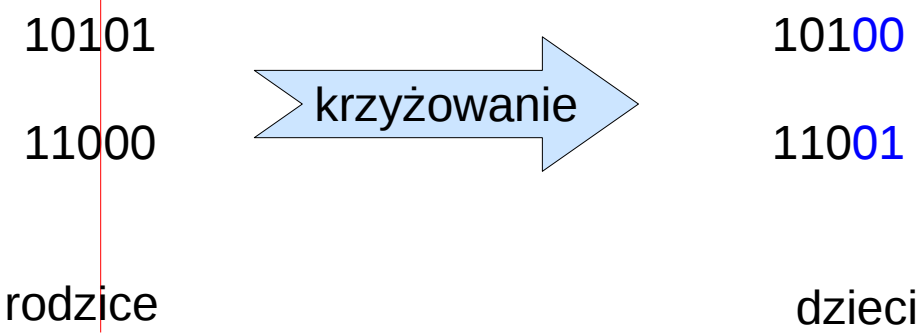
Przypuśćmy, poszczególnym ciągom kodowym z populacji odpowiada sektor koła ruletki o wymiarze proporcjonalnym do przystosowania. Dla każdego ciągu kodowego możemy wyznaczyć wartość funkcji celu

Nr	Ciąg kodowy	Przystosowanie	%całości
1	10101	441	34.6
2	00001	1	0
3	10000	256	20.1
4	11000	576	45.2
Łącznie		1274	100



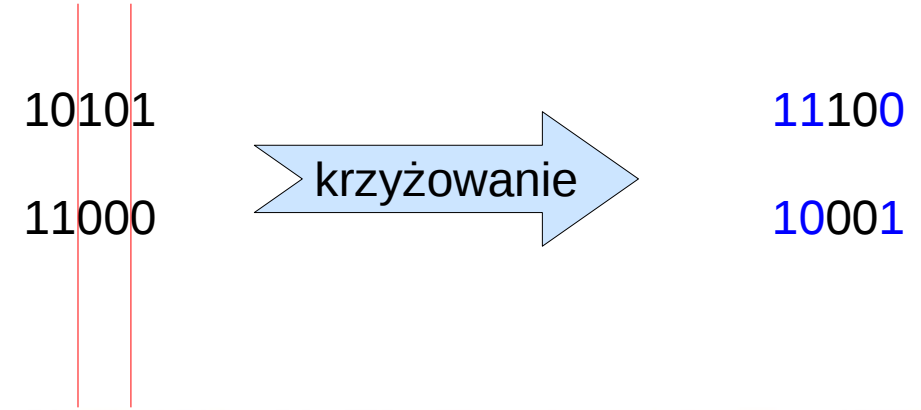
Krzyżowanie proste (jednopunktowe)

W pierwszej fazie kojarzymy w sposób losowy ciągi z puli rodzicielskiej w pary. Każda para
przechodzi proces krzyżowania: wybieramy w sposób losowy pozycję k na ciągu kodowym (spośród l-1 pozycji, gdzie długość ciągu), następnie zamieniamy miejscami wszystkie znaki od pozycji k+1 do l włącznie w obu elementach pary.



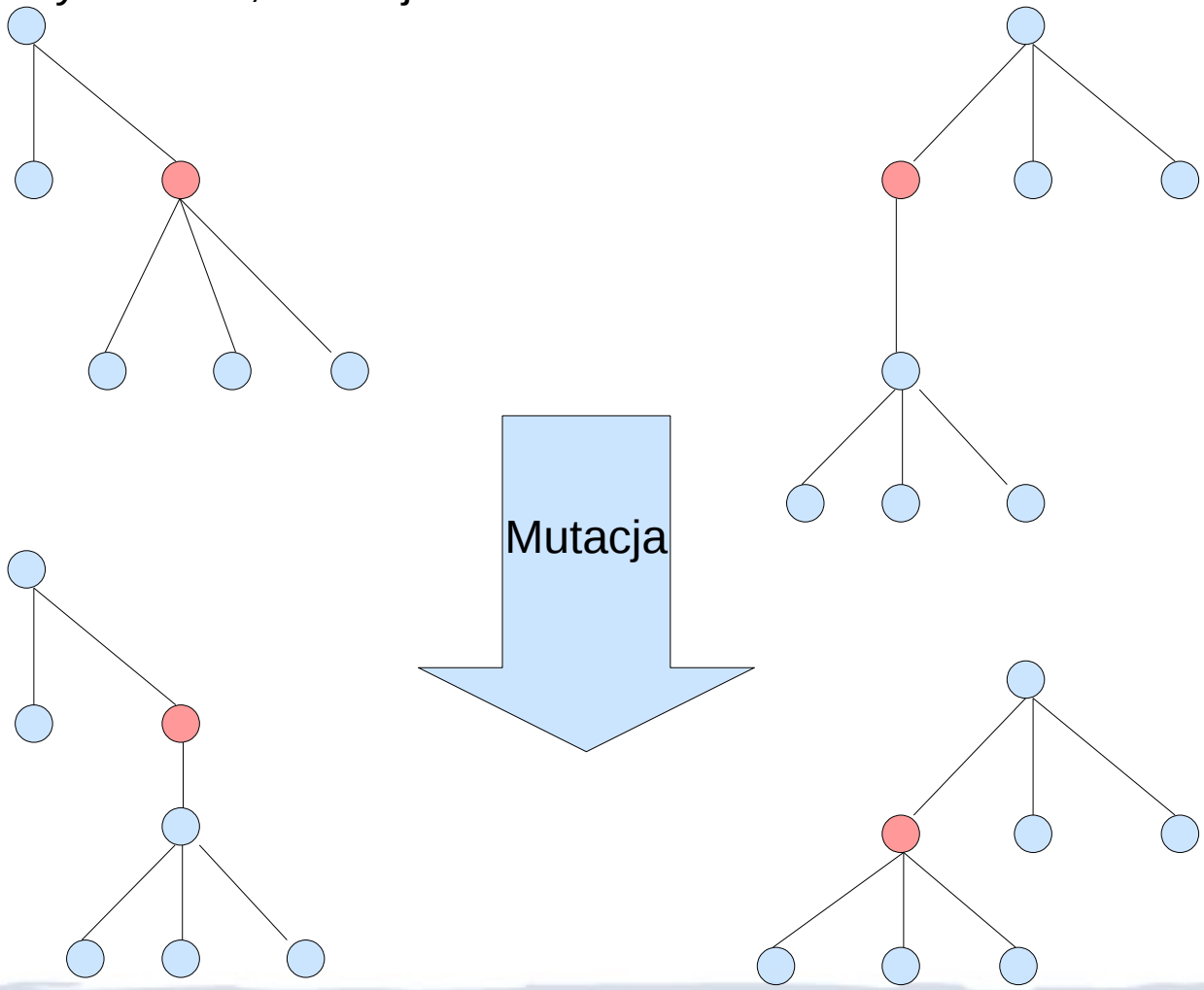
W wyniku krzyżowania mogą powstać nowe ciągi wchodzące w skład nowego pokolenia!

Występuje również inne typy krzyżowań: krzyżowanie dwupunktowe, krzyżowanie wielopunktowe.

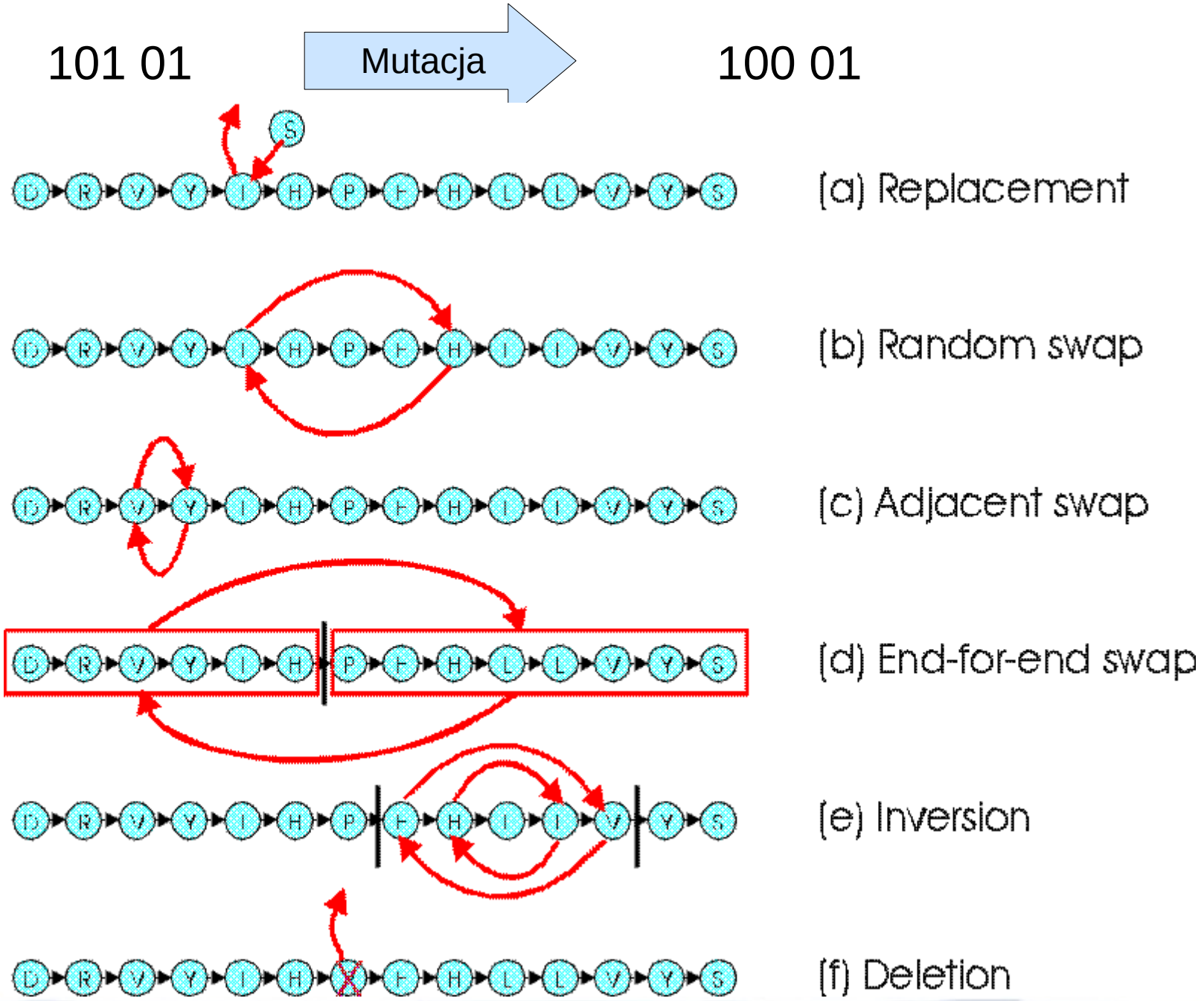


Reprezentacja problemu

Można użyć dowolnej reprezentacji genomów. Holland w swych pracach stosował reprezentacje łańcuchową, ale można użyć dowolnej innej, macierzową, listy, drzewa. To co jest absolutnie konieczne to to, że każda reprezentacja musi mieć zdefiniowane operatory krzyżowania, mutacji.



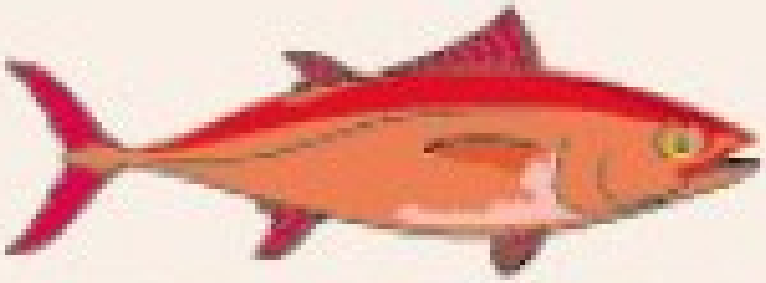
Mutacja


101 01
Mutacja
100 01
(a) Replacement
(b) Random swap
(c) Adjacent swap
(d) End-for-end swap
(e) Inversion
(f) Deletion

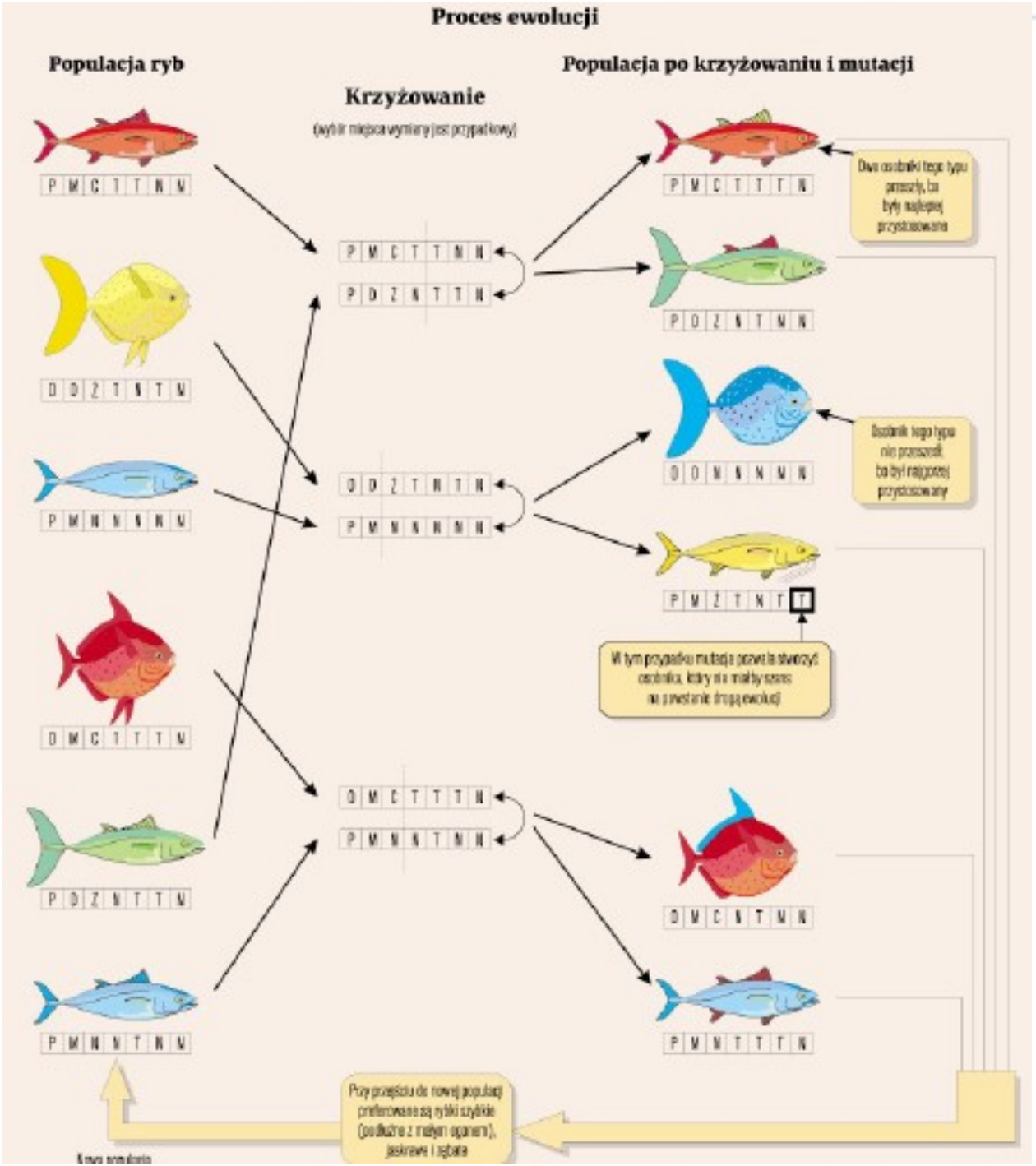
Przykład skutków działania operacji krzyżowania

kodowanie chromosomu

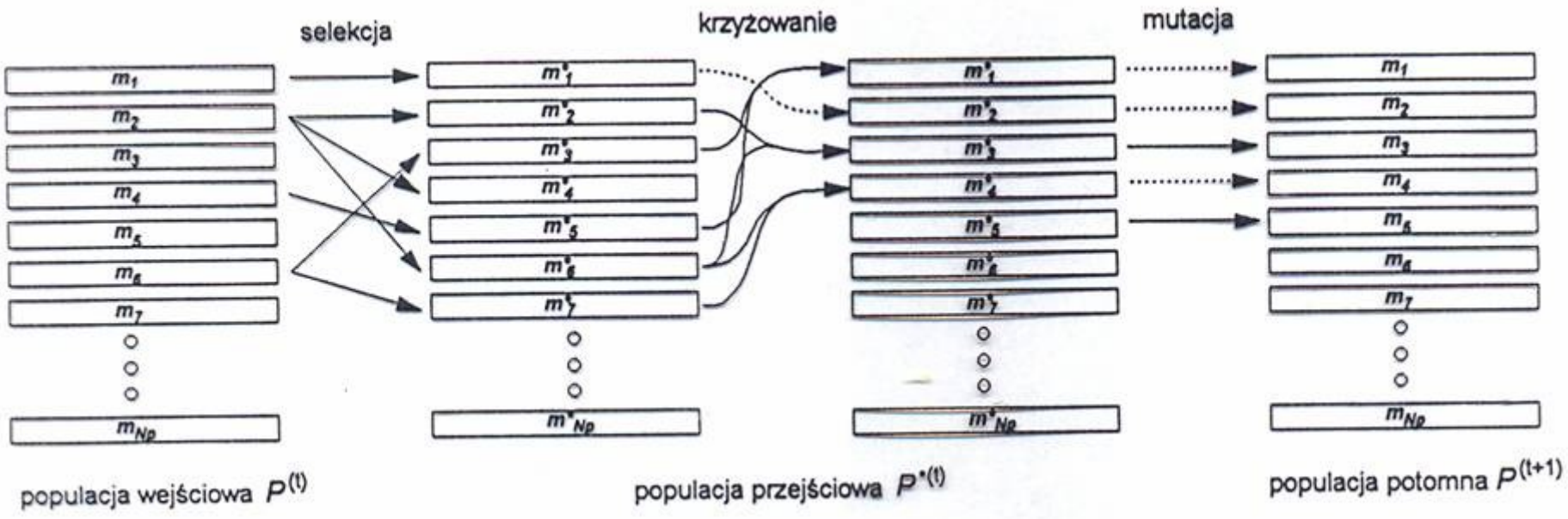
kształt	ogon	kolor	płetwy dolne	płetwy górne	zęby	wąsy
P - podłużny O - okrągły	M - mały D - duży	C - czerwony N - niebieski Z - zielony Ż - żółty	T - są N - brak	T - są N - brak	T - są N - brak	T - są N - brak



P	M	C	T	T	N	N


Proces ewolucji
Populacja ryb
Krzyżowanie
(wybór miejsca wymiany jest przypadkowy)
Populacja po krzyżowaniu i mutacji
P M C T T N N
D D Z T N T N
P M N N N N N
D M C T T T N
P D Z N T T N
P M N N T N N
P M C T T T N
P D Z N T M N
D D N N N M N
P M Z T N T T
D M C N T M N
P M N T T T N
Dwa osobniki tego typu przeszły, bo były najlepiej przystosowane
Osobnik tego typu nie przeszedł, bo był najgorzej przystosowany
W tym przypadku mutacja pozwoliła stworzyć osobnika, który ma małe szanse na powodzenie drogą ewolucji
Przy przejściu do nowej populacji preferowane są ryby szybkie (podłużne z małym ogonem), jaskrawe i agresywne

Schemat tworzenia kolejnych populacji



Algorytm genetyczny

1) Utwórz losową populację zawierającą n chromosomów
2) Oceń przystosowanie każdego chromosomu.
3) Utwórz nową populację przez zastosowanie następujących kroków:
    - Selekcja - wybierz dwa chromosomy z populacji w sposób losowy, ale z uwzględnieniem dopasowania
    - Z wybranych chromosomów utwórz 2 nowe poprzez zastosowanie operacji krzyżowania
    - Geny w chromosomie ulegają mutacji z pewnym prawdopodobieństwem *p*
    - Umieść utworzone chromosomy w nowej populacji
4) Sprawdź warunek stopu, jeśli nie spełniony idź do punktu 2.

Punkt stopu:

1) Znalezione zostało rozwiązanie z satysfakcjonującą wartością minimalizowanej (maksymalizowanej) funkcji.
2) Osiągnięta została ustalona liczba generacji.
3) Osiągnięty został maksymalny czas.
4) Osiągnięto plateau tzn. nie następuje poprawa dopasowania.

Wady algorytmów genetycznych

1) Częste wyliczanie dopasowania.
2) Lepsze rozwiązanie można oszacować tylko na podstawie osiągniętych dotychczas rozwiązań
3) GA nie bardzo nadają się do problemów, w których funkcja dopasowania ma tylko dwie wartości; dobrze, źle.
4) Trudno jest znaleźć jakieś charakterystyczne problemy o których wiadomo, że tylko przy pomocy GA można je rozwiązać. Są inne komplementarne metody optymalizacyjne jak np.: symulowane wyżarzanie, algorytmy mrówkowe itp..

Flow Chart for Genetic Programming

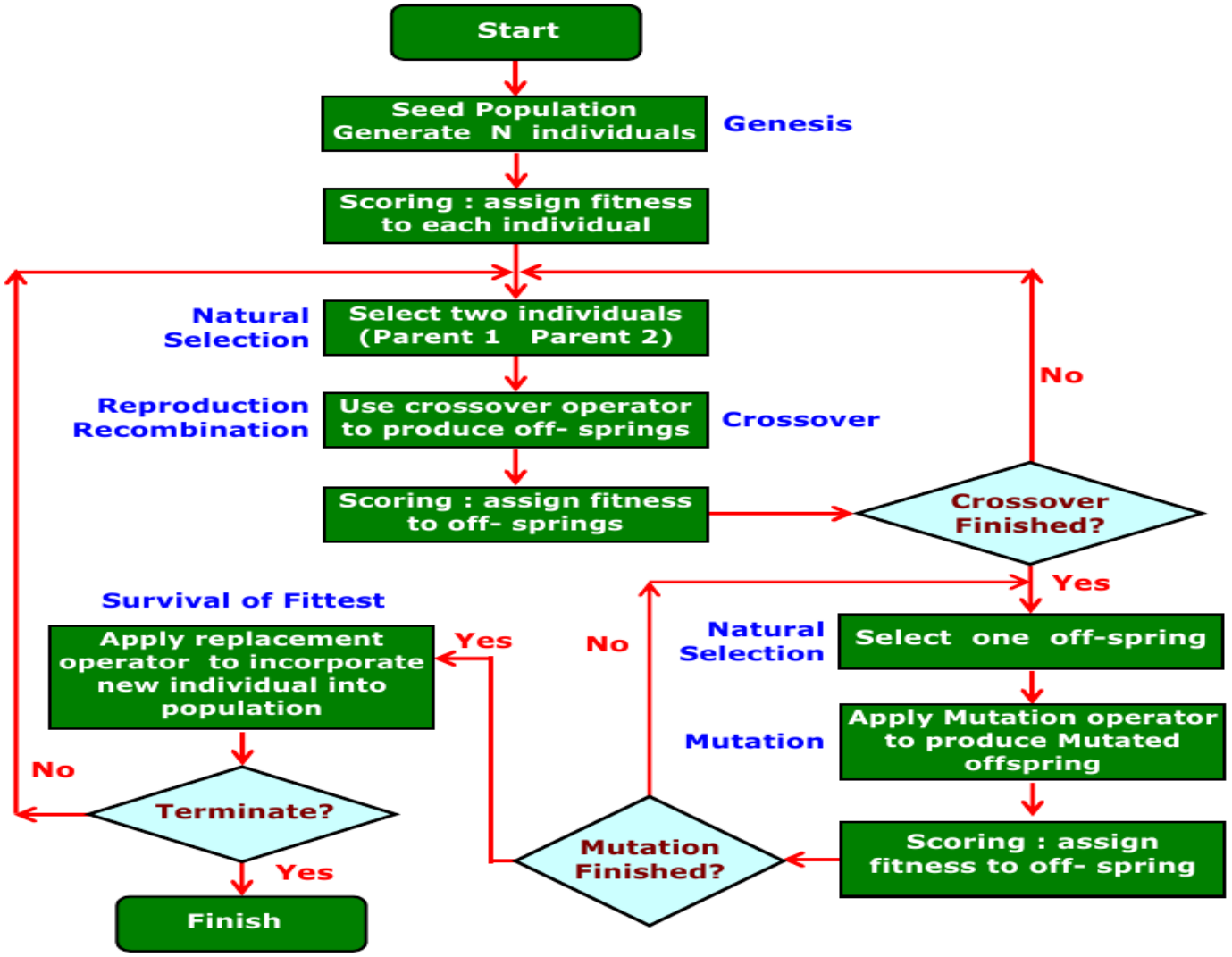
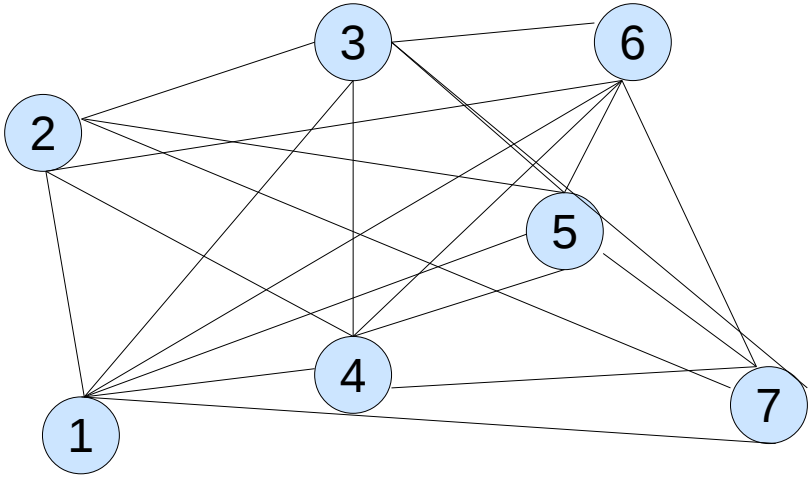


Fig. Genetic algorithm – program flow chart

Przykład zastosowania AG (TSP)



Budowa chromosomu

1	2	3	4	5	6	7

Tworzenie początkowej populacji

Wartość funkcji przystosowawczej

1	1 5 2 3 4 7 6	158	795
2	5 7 2 4 1 6 3	80	873
3	1 2 3 4 5 6 7	185	768
4	4 6 7 1 3 2 5	130	823
5	3 7 1 4 5 2 6	400	553
		953	3812

Tworzymy nową generację na podstawie ruletki

0.2085519412

0.2290136411

0.2014690451

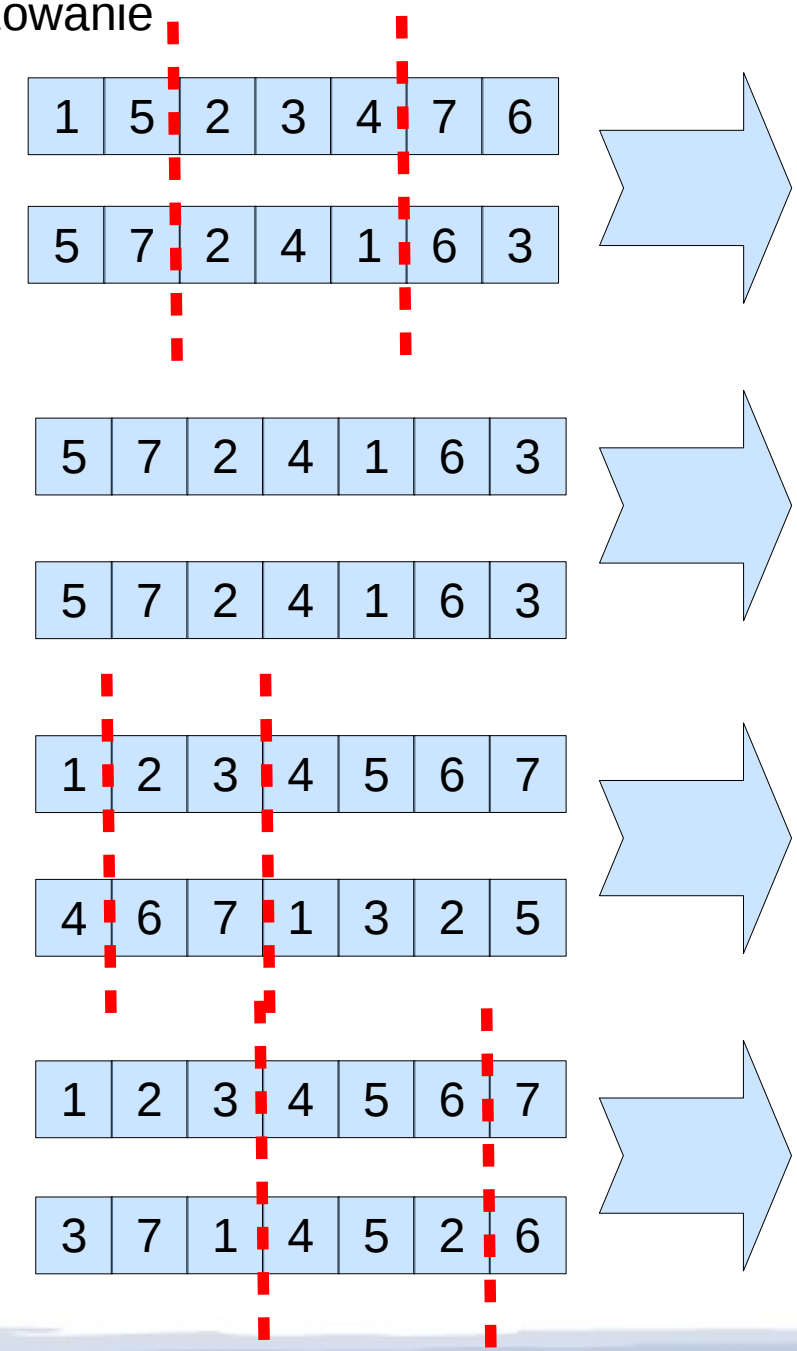
0.2158971668

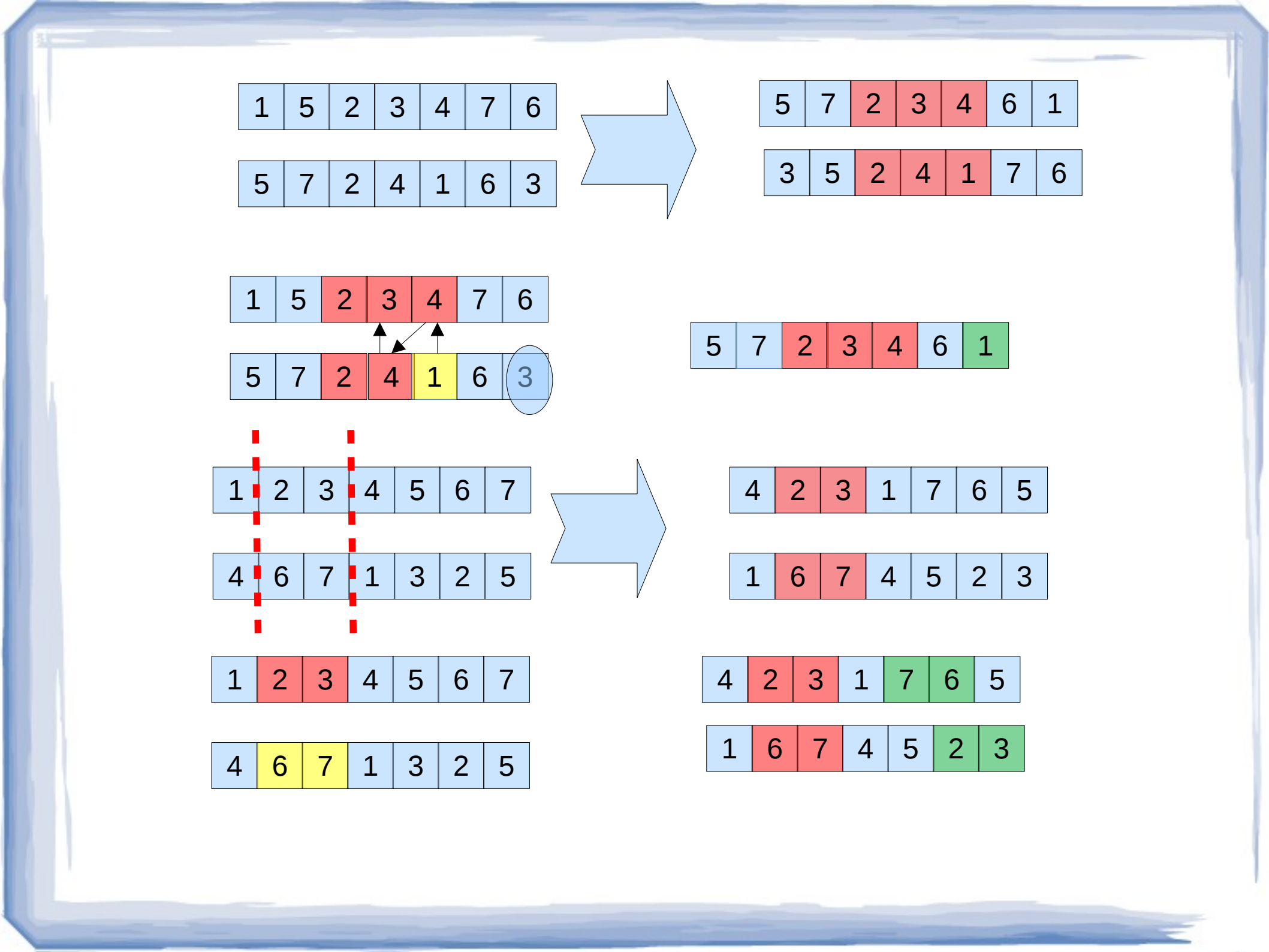
0.1450682057

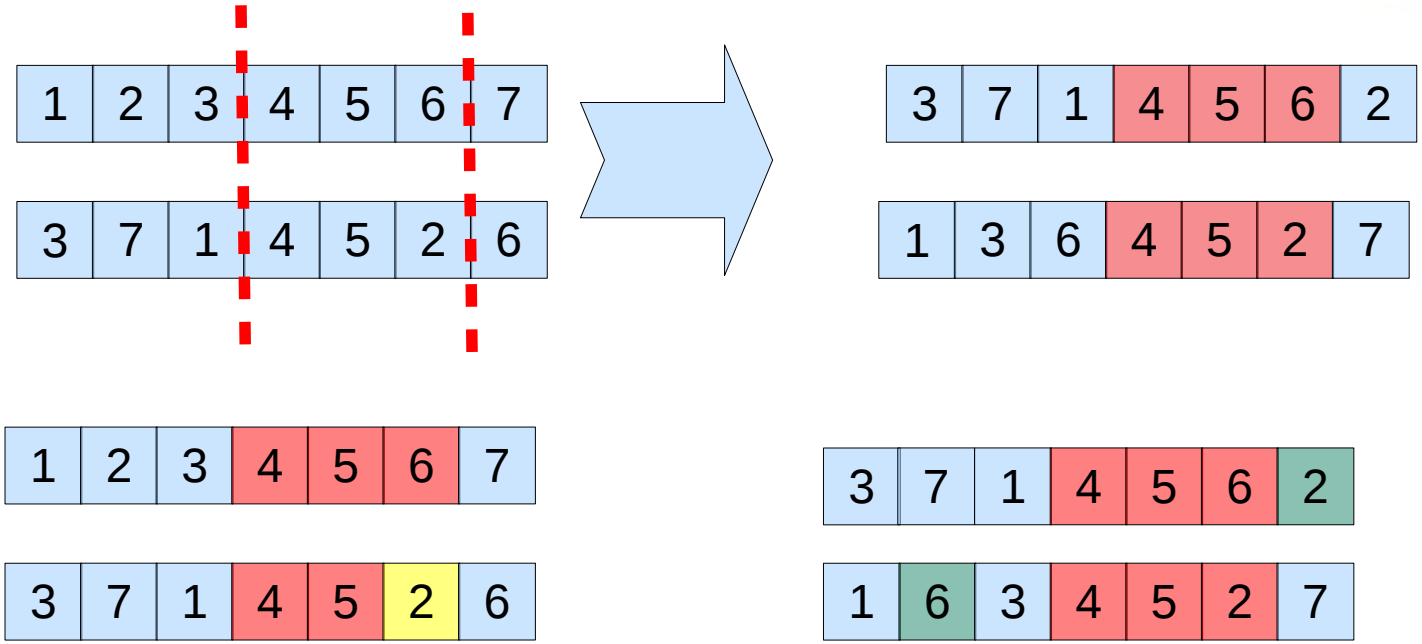
Wybrane zostały następujące pary:

(1,2) (2,2) (3,4) (3,5) (4,1)

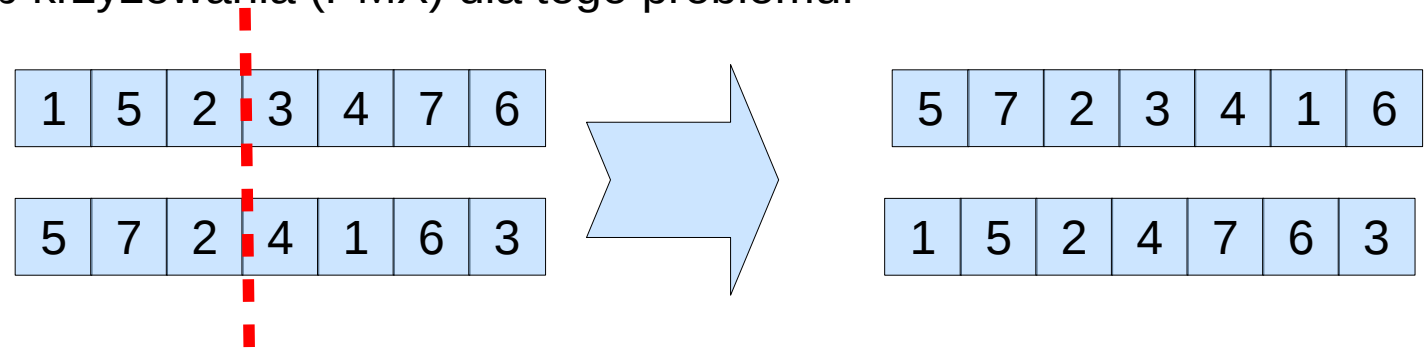
Teraz krzyżowanie



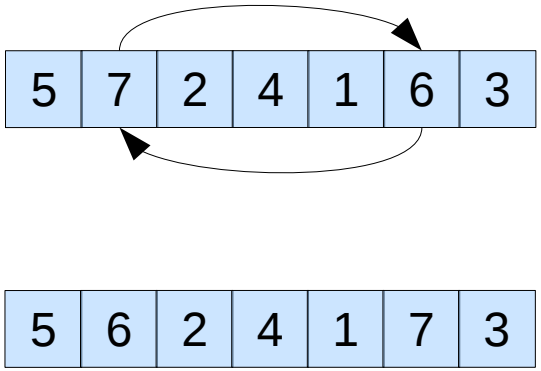

1 5 2 3 4 7 6
5 7 2 4 1 6 3
5 7 2 3 4 6 1
3 5 2 4 1 7 6
1 5 2 3 4 7 6
5 7 2 4 1 6 3
5 7 2 3 4 6 1
1 2 3 4 5 6 7
4 6 7 1 3 2 5
4 2 3 1 7 6 5
1 6 7 4 5 2 3
1 2 3 4 5 6 7
4 6 7 1 3 2 5
4 2 3 1 7 6 5
1 6 7 4 5 2 3


1 2 3 4 5 6 7
3 7 1 4 5 2 6
3 7 1 4 5 6 2
1 3 6 4 5 2 7
1 2 3 4 5 6 7
3 7 1 4 5 2 6
3 7 1 4 5 6 2
1 6 3 4 5 2 7

Inny sposób krzyżowania (PMX) dla tego problemu:



Teraz na otrzymanych potomkach dokonujemy mutacji

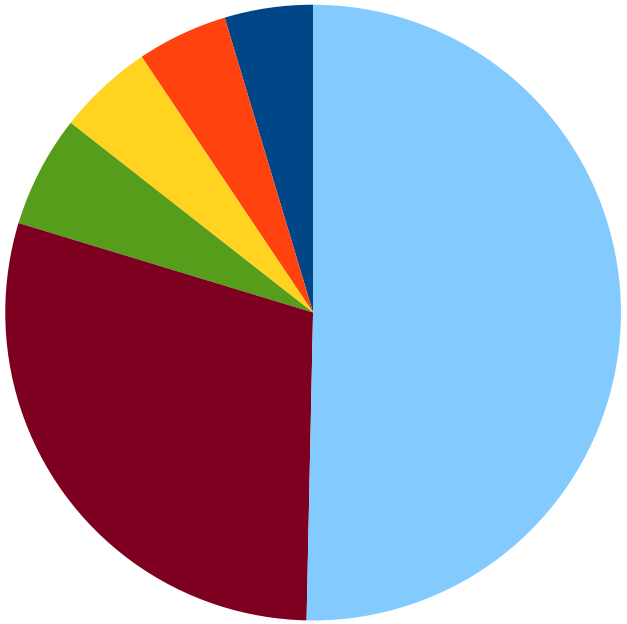


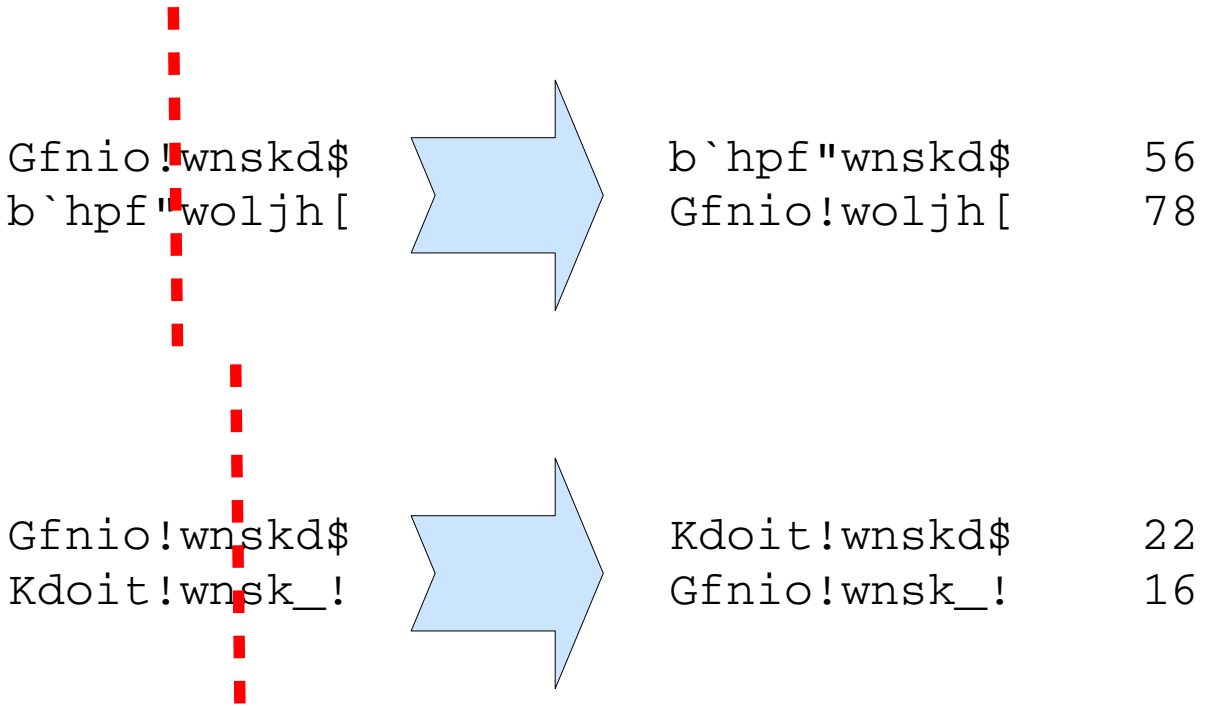
Dla otrzymanych potomków wyliczamy funkcje przystosowania i zaczynamy wszystko od początku

Drugi przykład dla GA
Hello world

Generujemy populacje	Wartość funkcji przystosowania
`IQQte=Yggem#`	152
`Crmt`!qrya+6`	148
`8ufxp+Rigfm*`	140
`b`hpf"woljh[`	120
`Kdoit!wnsk_!`	24
`Gfnio!wnskd$`	14

0.0065789474	0.0463609351
0.0067567568	0.0476139333
0.0071428571	0.0503347295
0.0083333333	0.0587238511
0.0416666667	0.2936192556
0.0714285714	0.5033472953
0.1419071327	




Gfnio!wnskd$
b`hpf"woljh[
b`hpf"wnskd$ 56
Gfnio!woljh[78
Gfnio!wnskd$
Kdoit!wnsk_!
Kdoit!wnskd$ 22
Gfnio!wnsk_! 16

Algorytm mrówkowy

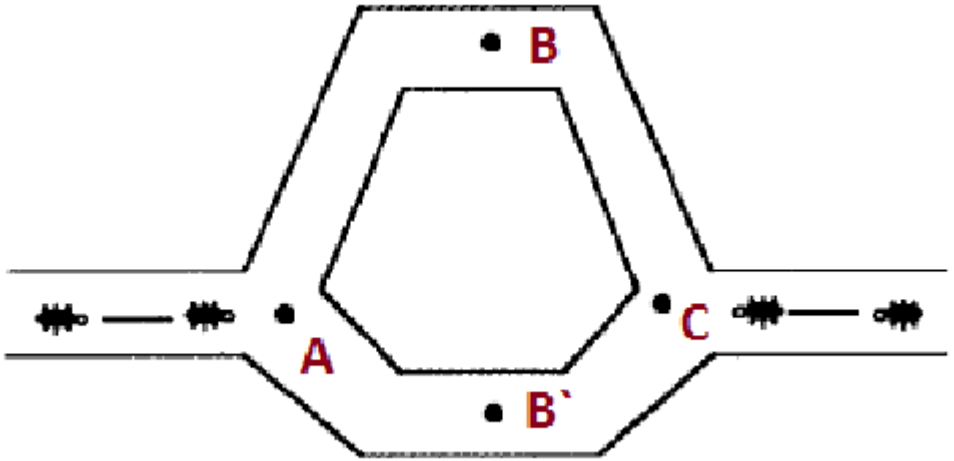
Systemy mrówkowe,podobnie jak algorytmy genetyczne, są systemami opartymi na populacji

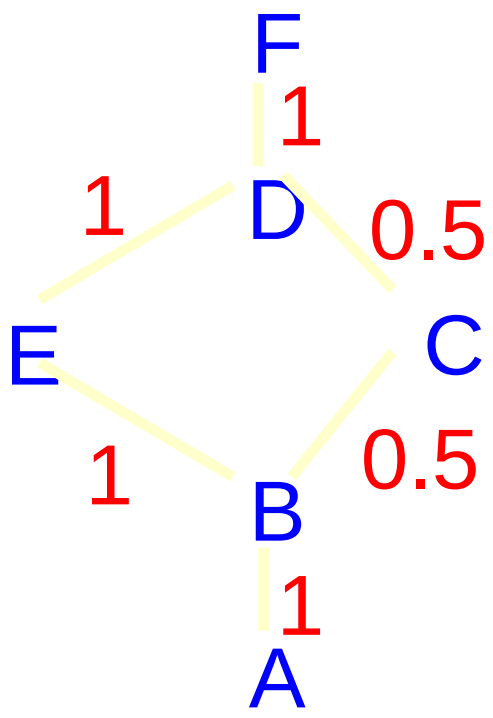
Występuje populacja mrówek, w której każda z mrówek, która znajdzie rozwiązanie, informuje o tym pozostałe mrówki

- Mrówki są właściwie ślepe, a mimo to są w stanie znaleźć drogę do pożywienia i z powrotem do mrowiska. Jak to możliwe?
- Obserwacje zachowania mrówek stały się inspiracją do powstania nowego typu algorytmów zwanych mrówkowymi (albo systemami mrówkowymi)
- Algorytm mrówkowy powstał w 1996, został wymyślony przez Dorigo, w chwili obecnej jest wciąż intensywnie rozwijany

- Czas t w algorytmie mrówkowym jest wielkością dyskretną
- W każdej chwili czasu mrówka przemieszcza się o odległość d=1
- Po każdym ruchu mrówka pozostawia na drodze jedną jednostkę feromonu
- W chwili t=0 nie ma na żadnej ścieżce żadnej jednostki feromonu.

- Wyjście mrówki z mrowiska
- Mrówka zostawia za sobą ślad feromonowy
- Druga mrówka wychodząca z mrowiska ma do wyboru pójść własną ścieżką albo po śladach feromonowych poprzedniej mrówki, im silniejszy feromon tym większe prawdopodobieństwo że nowa mrówka wybierze tą ścieżkę
  - Silny ślad feromonowy jest wówczas, gdy ścieżka jest często odwiedzana, lub gdy odległość do jedzenia jest bliska (feromon paruje)





Z punktu A do F wędruje 16 mrówek, z F do A także 16

W chwili t=1 w punkcie B i D znajduje się 16 mrówek. W chwili t=2 w punkcie E znajduje się 16 mrówek, natomiast w punkcie D i B 8 mrówek.

Na poszczególnych krawędziach występuje następująca „intensywność”

FD=16, AB=16, BE=8, ED=8, BC=16,CD=16

- Interesuje nas eksploracja przestrzeni, a nie tylko wyznaczanie trasy
- W związku z tym mrówka wybierać będzie drogę z pewnym prawdopodobieństwem, które będzie wprost proporcjonalne do intensywności feromonów
- Prawdopodobieństwo jest nie tyko funkcją intensywności feromonów, ale również tego co mrówka widzi
- Ścieżka feromonowa nie może trwać wiecznie dlatego konieczne jest żeby feromony parowały

Algorytm

```
procedure ACO
    while(nie koniec)
       Wygeneruj_rozwiązanie()
       Odśwież_znaki_feromonowe()
    end while
  end procedure
```

Algorytm mrówkowy dla przypadku komiwojażera

- Pojedyncze mrówki umieszczone są w każdym z miast. (t=0), w czasie t+1 mrówki przemieszczają się do kolejnego miasta
- Każdy z węzłów grafu zawiera informację o intensywności feromonów prowadzących do tego miasta.
- Niech $T_{ij}(t)$ oznacza intensywność krawędzi (i,j) w czasie t
- Gdy mrówka decyduje do którego miasta powędrować robi to z prawdopodobieństwem, które zależne jest od odległości do tego miasta i intensywności feromonów na krawędzi do niego prowadzącej
- Odległość do następnego miasta określana jest jako to co widzi mrówka, n_{ij}, i zdefiniowana jest jako $1/d_{ij}$,

- W każdej chwili czasu następuje parowanie foromonów. Ilość wyparowanego feromonu p, jest wartością pomiędzy 0 i 1
- Ażeby uniknąć odwiedzania przez mrówkę tego samego miasta stosowana jest tablica Tabu
- $Tabu_k$ jest listą miast dla k-tej mrówki, które zostały przez nią odwiedzone
- Po każdej turze intensywność feromonów na odpowiednich ścieżkach jest uaktualniana zgodnie ze wzorem:

$$T_{ij}(t+n) = (1-\rho)\, T_{ij}(t) + \Delta T_{ij}$$

gdzie ρ współczynnik parowania feromonu

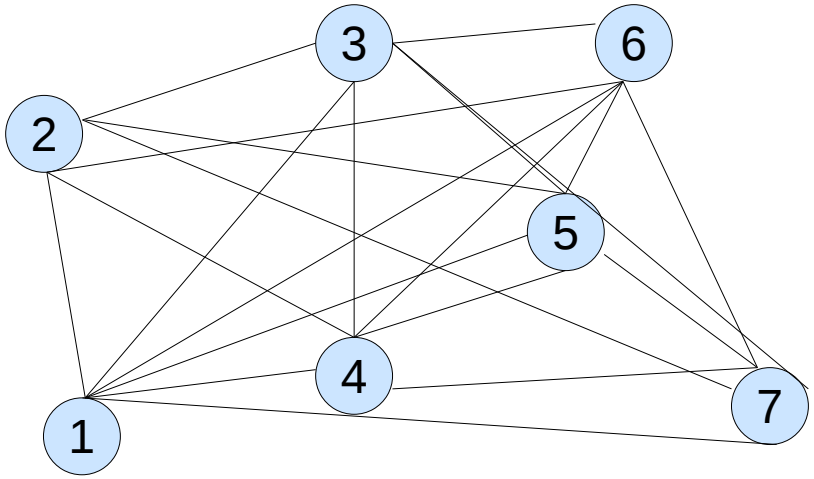
$$\Delta T_{ij} = \begin{cases} \frac{Q}{L_k} & jeśli\ krawędź\ (i,j)\ należy\ do\ trasy\ mrówki\ k \\ 0 & w\ przeciwnym\ razie \end{cases}$$

Gdzie Q to pewna dobrana stała, zależna od rozpatrywanego problemu, a L_k to koszt rozwiązania związanego z mrówką k

Prawdopodobieństwo przejścia z i do j

$$p_{ij}^{k}=\begin{cases}\dfrac{T_{ij}^{\alpha}\sigma_{ij}^{\beta}}{\sum_{h\notin Tabu_k} T_{ih}^{\alpha}\sigma_{ih}^{\beta}} & jeżeli\ j\notin Tabu_k \\ 0 & \end{cases}$$

Gdzie sigma to parametr wskazujący na atrakcyjność przejścia, w tym przypadku związana Jest z odległością do rozpatrywanego miasta.
Parametry alfa i beta to parametry regulujące ważność intensywności feromonu względem atrakcyjności




```
- 6 8 4 3 4 7
0 - 4 5 9 3 5
0 0 - 5 6 3 2
0 0 0 - 8 2 1
0 0 0 0 - 6 3
0 0 0 0 0 - 2
```

Zaczynamy od tego, że w każdym mieście umieszczamy 1 mrówkę i wyliczamy prawdo podobieństwa przejścia dla wszystkich mrówek. Mrówka *k* znajdująca się w mieście *r* wybiera miasto s zgodnie ze wzorem:

$$s = \begin{pmatrix} \underset{u \notin Tabu_k}{argmax}\left[\tau(r,u)\right]\left[\eta(r,u)\right]^{\beta} \; if \; q < q_0 \\ S \; otherwise \end{pmatrix}$$

Gdzie $\tau(r,u)$ oznacza ilość feromonu na ścieżce (r,u), natomiast $\eta(r,u)=1/d_{ru}$, q jest wartością losową z przedziału [0,1] z rozkładu jednostajnego, q_0 jest parametrem [0,1], S jest zmienną losową oznaczającą miasto, które zostanie wylosowane zgodnie ze wzorem na prawdopodobieństwo(na poprzednim slajdzie).

Ścieżka feromonowa tworzona jest lokalnie i globalnie tzn. po przejściu wszystkich miast Wyliczana jest długość ścieżki u na jej podstawie nagradzane są ścieżki krótsze.

$$T_{ij}=(1-\rho)T_{ij}+\rho\Delta T_{ij};\Delta T_{ij}=(najkrótsza\ droga)^{-1}$$

Wyliczmy T1j (czylii lokalna ścieżka feromonowa)(prawopodobienstwa dla 1 mrowki

T12=10/6; p12=10/36/(10/36+10/64+10/16+10/9+10/16+10/49)=0.09
T13=10/8; p13=0.05
T14=10/4; p14=0.21
T15=10/3; p15=0.37
T16=10/4; p16=0.21
T17=10/7; p17=0.07

Losujemy miasto dla 1 mrówki i powtarzamy wszystko dla pozostałych mrówek

Po przejściu wszystkich miast uzupełniamy ścieżkę feromonową zgodnie z wzorem dla globalnego nagradzania i zaczynamy wszystko od początku mając już aktualne ścieżki feromonowe!

Zastosowania algorytmu mrówkowego

Problem komiwojażera (TSP)

Problemy klasyfikacyjne

Problemy szeregowania

Znajdowanie trasy dla pojazdów

Pokrycia zbiorów,

Wykrywanie krawędzi na obrazach rastrowych.

itp

Warunki Kuhna-Tuckera

Warunki konieczne istnienia ekstremum.

$$\begin{aligned} &\min_{x \in R^n} F(\boldsymbol{x}) \\ &c_i(\boldsymbol{x}) \leqslant 0, i = 1 \ldots m \\ &F(\boldsymbol{x}), c_i(\boldsymbol{x}) \in C^1 \end{aligned} \qquad (1)$$

Każdy punkt spełniający ograniczenia nazywany jest punktem dopuszczalnym. Celem Optymalizacji z ograniczeniami jest znalezienie punktu dopuszczalnego, w którym minimalizowana funkcja osiąga przynajmniej lokalnie najmniejszą możliwą wartość.

Tw. Kuhna-Tuckera

Jeśli w punkcie x^o funkcja $F(x^o)$ osiąga minimum lokalne, to w punkcie tym istnieją mnożniki lambda spełniające warunki

$$\nabla_x F(\boldsymbol{x^o}) + \sum_i \lambda_i^o \nabla c_i(\boldsymbol{x}^o) = \boldsymbol{0}$$

$$c_i(\boldsymbol{x}^o) \leqslant 0$$

$$\lambda_i^o c_i(x^o) = 0$$

$$\lambda_i^o \geqslant 0$$

Przy wykorzystaniu funkcji Lagrange'a można te warunki zapisać tak

$$\nabla_x F(x^0) + \sum_{i \in E} \lambda_i^0 \nabla c_i(x^0) = 0$$
$$c_i(x^0) \leqslant 0$$
$$\lambda_i^0 c_i(x^0) = 0$$
$$\lambda_i^0 \geqslant 0$$

Uzasadnienie (http://aq.ia.agh.edu.pl/Aquarium/Dydaktyk/Wyklady/MO/2005-06/Wyklad07.PDF)
Def.

Warunek ograniczający $c_j(x^o)$ jest aktywny w punkcie dopuszczalnym x^o, jeśli $c_j(x^o)=0$, w przeciwnym razie warunek nazywany jest nieaktywnym.

Rozważmy 3 przypadki, zakładając że jesteśmy w punkcie optymalnym, dla przykładu sformułowanego w R^2 funkcja Lagrange'a ma postać

$$L = F(x) + \sum_{j=1}^{3} \lambda_j c_j(x)$$

Przypadek 1, brak ograniczeń aktywnych

Warunki KT będą spełnione, gdy

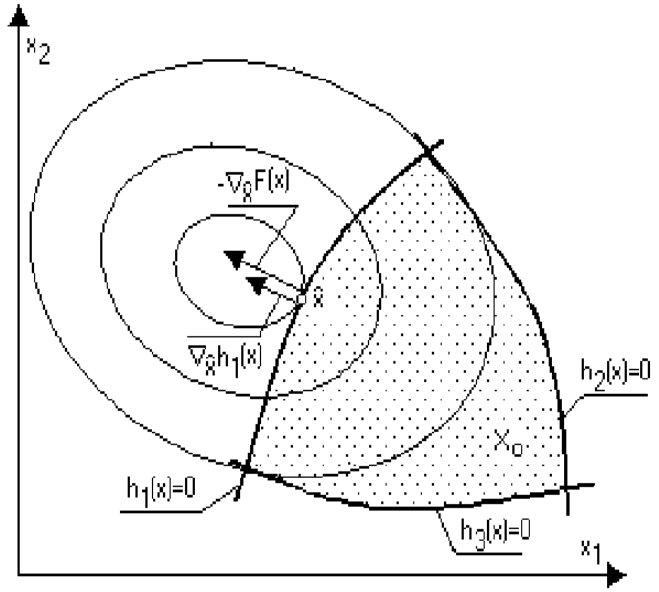
$$\lambda_1^o=\lambda_2^o=\lambda_3^o=0, \nabla F_x(\boldsymbol{x}^o)=0$$

Przypadek 2, jedno ograniczenie aktywne

$$c_1(\boldsymbol{x}^o)=0, c_2(\boldsymbol{x}^o)<0, c_3(\boldsymbol{x}^o)<0$$

Zadanie jest równoznaczne z poszukiwaniem min F(x) przy ograniczeniu $c_1(x)=0$
Stąd warunek konieczny optymalności ma postać

$$\lambda_2^o=\lambda_3^o=0, -\nabla_x F_x(\boldsymbol{x}^o)=\lambda_1^o \nabla_x c_1(\boldsymbol{x}^o)$$

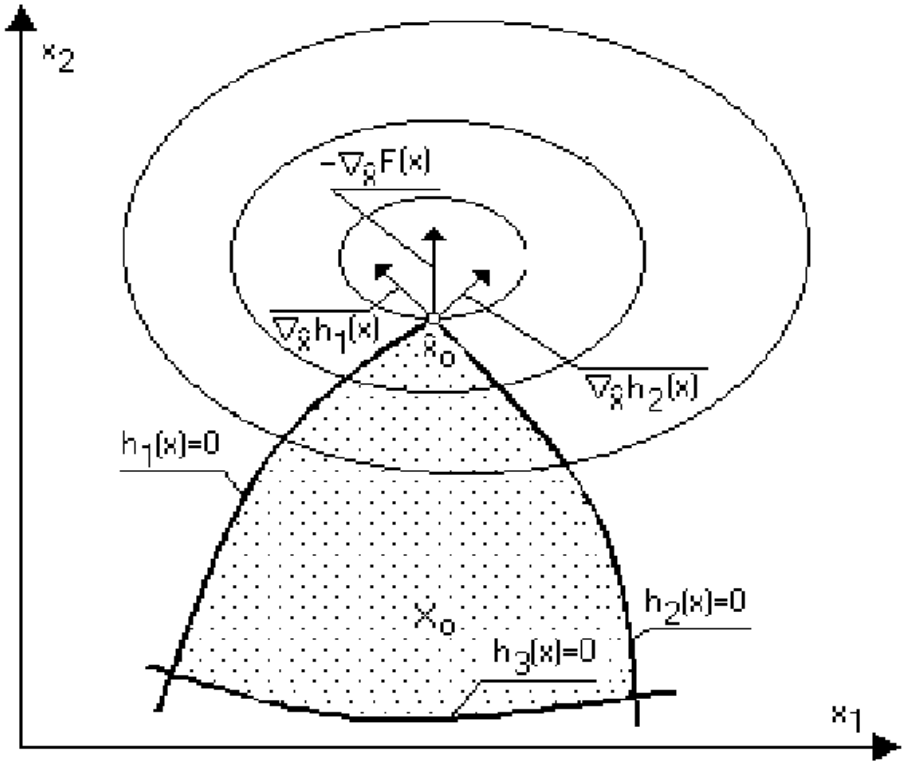


Ponieważ gradient zawsze wskazuje na kierunek wzrostu, to kierunki obu gradientów są takie same w związku z czym lambda>0.

Przypadek 3, dwa ograniczenia aktywne

$$c_1(\boldsymbol{x}^o)=0, c_2(\boldsymbol{x}^o)=0, c_3(\boldsymbol{x}^o)<0$$

$$\lambda_3^o=0, -\nabla_x F_x(\boldsymbol{x}^o)=\lambda_1^o \nabla_x c_1(\boldsymbol{x}^o)+ \lambda_2^o \nabla_x c_2(\boldsymbol{x}^o)$$



Podsumowując:

$$\lambda_j \geqslant 0\,,\ j=1,2,3$$
$$Jeżeli\ c_j(x^0)<0 \rightarrow zawsze\ \lambda_j^0=0$$
$$\lambda_j^0>0 \rightarrow c_j(x^o)=0$$

Żeby uwzględnić te warunki do warunku Lagrange'a dokładamy tzw. warunek komplementarności

$$\lambda_j^0 c_j(\boldsymbol{x}^o)=\mathbf{0}$$

Warunki regularności

Konieczne jest sformułowanie kryteriów gwarantujących istnienie mnożników Lagrange'a w punkcie rozwiązania x^o.
W punkcie rozwiązania

$$-\nabla_x F(\boldsymbol{x}) = + \sum_j \boldsymbol{\lambda}_j \nabla_x c_j(\boldsymbol{x})$$

Mamy więc n równań o J niewiadomych, gdzie J – liczba ograniczeń aktywnych w punkcie x^o; J<=n. Oczywiście macierz nabla c musi mieć odpowiedni rząd, co jest gwarantowane przez liniową niezależność tych wektorów. Gdy rząd macierzy jest równy J to rozwiązanie dla współczynników Lagrange'a jest jednoznaczne.
War. 1

Punkt $\boldsymbol{x}^o$ jest regularny, jeśli istniejące w nim gradienty ograniczeń aktywnych są liniowo niezależne.

Geometrycznie oznacza to, że gradient funkcji celu w punkcie rozwiązania może być reprezentowany przez liniową kombinację gradientów ograniczeń aktywnych.

Przykład

Znaleźć minimum funkcji

$$-x^2-x^3$$

Przy ograniczeniu $x^2 \leqslant 1$

Funkcja Lagrange'a ma postać

$$L(x,\lambda)=-x^2-x^3+\lambda(x^2-1)$$

Warunki KT

$$-2x-3x^2+2\lambda x=0$$
$$x^2 \leqslant 1$$
$$\lambda \geqslant 0$$
$$\lambda(x^2-1)=0$$

Przypadek 1: Ograniczenie nieaktywne $\lambda=0, x^2-1\neq 0$

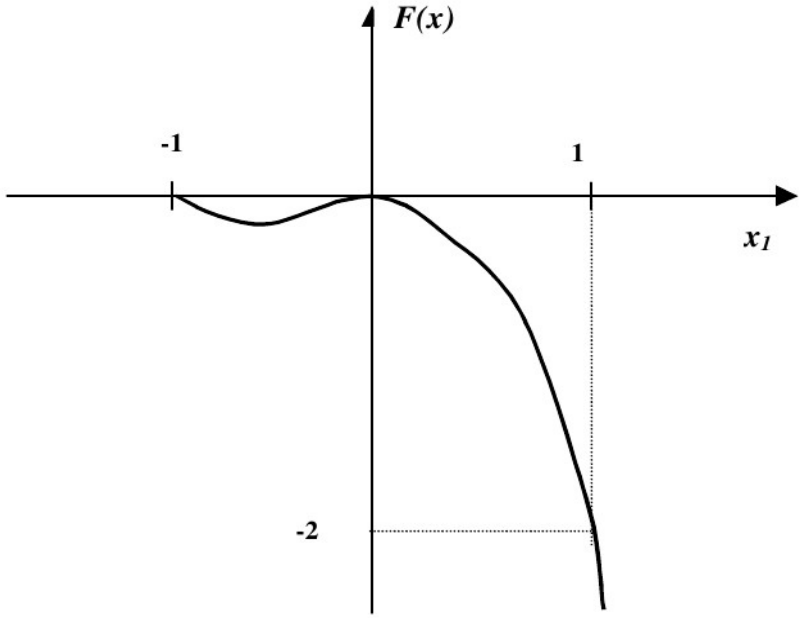
$$-2x-3x^2=0 \rightarrow x=-\frac{2}{3} \; lub \; x=0$$

Przypadek 2: Ograniczenie aktywne $\lambda \neq 0, x^2 - 1 = 0$

$$x^o = 1, \lambda = \frac{5}{2}$$

$$x^o = -1, \lambda = -\frac{1}{2} \left(warunek\ \lambda \geqslant 0\ nie\ spełniony \right)$$

Czyli rozwiązaniem są pary (0,0), (1,5/2), (-2/3,0). Problem nieregularności nie występuje



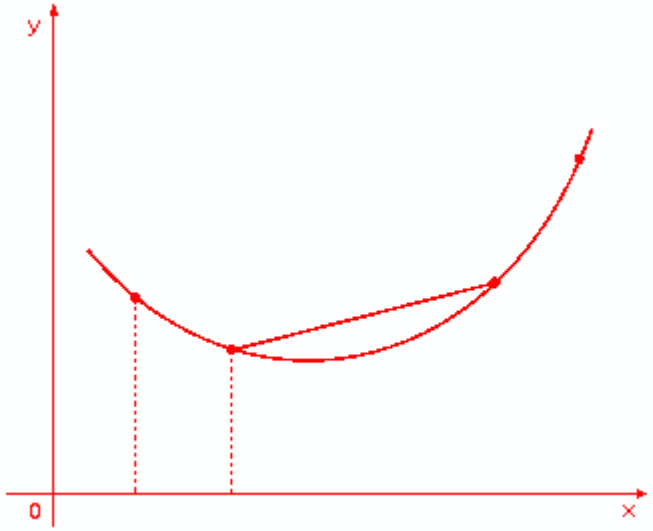
Warunek konieczny i dostateczny KT

Każdy punkt regularny będący rozwiązaniem zadania (1) spełnia warunki KT, ale nie każdy punkt x^o spełniający warunki musi być szukanym minimum.

Tw.
Jeśli dla zadania (1) są spełnione warunki KT, a funkcje ograniczające oraz funkcja F(x) są wypukłe to $\boldsymbol{x}^o$ jest rozwiązaniem.

Funkcję rzeczywistą f określoną na zbiorze wypukłym C nazywamy wypukłą jeżeli

$$\forall x_1, x_2 \in C \; \forall \alpha, \beta \in [0,1], \alpha + \beta = 1 \; f(\alpha x_1 + \beta x_2) \leqslant \alpha f(x_1) + \beta f(x_2)$$



Przykład, w którym nie są spełnione warunki regularności

$$\min_{x \in R^2} x_2$$

$$(x_1 - 0.75)^2 + (x_2 - 0.75)^2 = 1.125$$

$$(x_1 + 0.75)^2 + (x_2 + 0.75)^2 = 1.125$$

$$\nabla c_1(\boldsymbol{x}) = \begin{bmatrix} 2(x_1 - 0.75) \\ 2(x_2 - 0.75) \end{bmatrix}, \nabla c_2(\boldsymbol{x}) = \begin{bmatrix} 2(x_1 + 0.75) \\ 2(x_2 + 0.75) \end{bmatrix}$$

Z rysunku widać że punkt optymalny to xopt=0 stąd wartości pochodnych są następujące

$$\nabla c_1(\mathbf{0}) = \begin{bmatrix} -1.5 \\ -1.5 \end{bmatrix}, \nabla c_2(\boldsymbol{x}) = \begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix}$$

Widać od razu, że nie ma możliwości spełnić pierwszy warunek KKT bo pochodna funkcji nie da się przedstawić w postaci liniowej kombinacji ∇c

$$\min_{x \in R^2} f(\boldsymbol{x}) = x_1 + x_2$$
$$x_1^2 + x_2^2 = 1$$

Warunki konieczne optymalności przyjmują postać

$$1 + 2\lambda x_1^{opt} = 0;$$
$$1 + 2\lambda x_2^{opt} = 0;$$
$$\left(x_1^{opt}\right)^2 + \left(x_2^{opt}\right)^2 = 1$$

$$x_1^{opt} = -\frac{1}{2\lambda}$$
$$x_2^{opt} = -\frac{1}{2\lambda}$$
$$\lambda^2 = \frac{1}{2}$$

Istnieją więc dwa rozwiązania jedno dodatnie drugie ujemne. W jednym rozwiązaniu funkcja przyjmuje wartość $f = -\frac{1}{\sqrt{2}}$ a w drugim $f = \frac{1}{\sqrt{2}}$

Tak więc pierwszy punkt jest punktem minimum a drugi maksimum.

Przypadek ograniczeń równościowych i nierównościowych

$$\min_{x \in R^n} F(\boldsymbol{x})$$
$$c_i(\boldsymbol{x}) \leqslant 0, i = 1 \ldots m$$
$$g_j(\boldsymbol{x}) = 0, j = 1 \ldots k$$
$$F(\boldsymbol{x}), c_i(\boldsymbol{x}), g_j(\boldsymbol{x}) \in C^1$$

Warunki KT przyjmują postać

$$\nabla_x F(\boldsymbol{x^o}) + \sum_{i=1}^{m} \lambda_i^o \nabla_{x_i} c_i(\boldsymbol{x^o}) + \sum_{i=1}^{k} \nu^o \nabla_{x_i} g_i(\boldsymbol{x^o}) = \boldsymbol{0}$$
$$c_i(\boldsymbol{x}^o) \leqslant 0$$
$$g_i(\boldsymbol{x}^o) = 0$$
$$\lambda_i^o c_i(x^o) = 0$$
$$\lambda_i^o > 0$$

Ograniczenia równościowe dopuszczają ujemną wartość mnożnika Lagrange'a. Brak Ograniczeń nierównościowych powoduje, że warunek komplementarności jest zbędny

W przypadku ograniczeń równościowych również konieczne jest spełnienie warunku Regularności
Warunki dostateczne otrzymuje się przy podobny założeniach jak poprzednio (wypukłości).
Bardziej ogólne warunki dostateczne (gdy spełnione są warunki konieczne KT) mają postać

$$\boldsymbol{d}^T \nabla_x L(\boldsymbol{x}, \boldsymbol{\lambda}) \boldsymbol{d} > 0$$
$$\nabla_x c_j(\boldsymbol{x}^o)^T \boldsymbol{d} = 0, \textit{dla wszystkch aktywnych warunków}$$
$$\nabla_x g_i(\boldsymbol{x}^o)^T \boldsymbol{d} = 0, i = 1 \ldots k$$

Programowanie kwadratowe

Funkcja celu, którą należy zminimalizować (zmaksymalizować)

$$z = \frac{1}{2} \boldsymbol{x}^T \boldsymbol{A}\, \boldsymbol{x} + \boldsymbol{g}^T \boldsymbol{x}$$

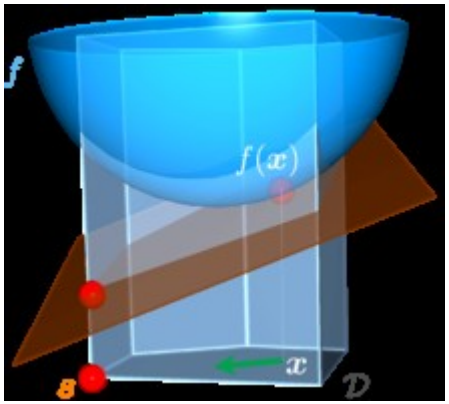
x wektor *n* wymiarowy, ***A*** macierz kwadratowa *n*x*n* dodatnio określona (dzięki czemu mamy formę kwadratową wypukłą), ***g*** wektor *n* wymiarowy

Ograniczenia

$$\boldsymbol{C}\, \boldsymbol{x} \leqslant \boldsymbol{b}$$

Macierz ograniczeń ***C*** wymiaru *n*x*m*

$$\boldsymbol{x} \geqslant \boldsymbol{0}$$



Rozwiązanie

1) Zbudowanie funkcji Lagrange'a
2) Utworzenie warunków KKT
3) Rozwiązanie zadania zastępczego metodą Wolfe'a

Funkcja Lagrange'a

$$L(\boldsymbol{x},\boldsymbol{\lambda})=\frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}+\boldsymbol{g}^T\boldsymbol{x}+\boldsymbol{\lambda}^T(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b})$$

Warunki KKT

$$\nabla_x L(\boldsymbol{x},\boldsymbol{\lambda})=\boldsymbol{0}$$
$$\boldsymbol{\lambda}^T\nabla_{\lambda} L(\boldsymbol{x},\boldsymbol{\lambda})=\boldsymbol{0}$$
$$\boldsymbol{\lambda}\geqslant\boldsymbol{0}$$

$$\boldsymbol{A}\boldsymbol{x}+\boldsymbol{g}^T+\boldsymbol{\lambda}\boldsymbol{C}=\boldsymbol{0}$$
$$\boldsymbol{\lambda}^T(\boldsymbol{C}\boldsymbol{x}-\boldsymbol{b})=\boldsymbol{0}$$
$$\boldsymbol{C}\boldsymbol{x}\leqslant\boldsymbol{b}$$
$$\boldsymbol{\lambda}\geqslant\boldsymbol{0}$$

Przykład

$$10x_1 + 25x_2 - 10x_1^2 - 4x_1 x_2 - x_2^2 \rightarrow MAX$$
$$x_1 + 2x_2 \leqslant 10$$
$$x_1 + x_2 \leqslant 9$$
$$x_1, x_2 \geqslant 0$$

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = 10x_1 + 25x_2 - 10x_1^2 - 4x_1 x_2 - x_2^2 + \lambda_1(10 - x_1 - 2x_2) + \lambda_2(9 - x_1 - x_2) + \lambda_3 x_1 + \lambda_4 x_2$$

Warunki KT

$$\frac{\partial L}{\partial x_1} = 10 - 20x_1 - 4x_2 - \lambda_1 - \lambda_2 + \lambda_3 = 0$$
$$\frac{\partial L}{\partial x_2} = 25 - 4x_1 - 2x_2 - 2\lambda_1 - \lambda_2 + \lambda_4 = 0$$

$$\lambda_1(10 - x_1 - 2x_2) + \lambda_2(9 - x_1 - x_2) + \lambda_3 x_1 + \lambda_4 x_2 = 0$$

$$10 - x_1 - 2x_2 \geqslant 0$$
$$9 - x_1 - x_2 \geqslant 0$$
$$x_1, x_2 \geqslant 0$$
$$\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geqslant 0$$

Sformułowanie zadania zastępczego

Zamiana oznaczeń

Współczynniki Lagreange'a dla ograniczeń z λ przechodzą do y, natomiast z warunków brzegowych do y'

$$\lambda_1 \rightarrow y_{1,} \lambda_2 \rightarrow y_{2,} \lambda_3 \rightarrow y_1', \lambda_4 \rightarrow y_2'$$

Sprowadzanie ograniczeń do postaci bazowej

$$x_1 + 2x_2 + x_1' = 10$$
$$x_1 + x_2 + x_2' = 9$$

Gdzie x1' i x2' to tzw. zmienne bilansujące

Zapisanie funkcji Lagrange'a z uwzględnieniem zmiennych bilansujących

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = 10x_1 + 25x_2 - 10x_1^2 - 4x_1 x_2 - x_2^2 + y_1(10 - x_1 - 2x_2 - x_1') + y_2(9 - x_1 - x_2 - x_2') + y_1' x_1 + y_2' x_2$$

Pierwszy warunek KT ma postać

$$\frac{\partial L}{\partial x_1} = 10 - 20x_1 - 4x_2 - y_1 - y_2 + y_1' = 0$$
$$\frac{\partial L}{\partial x_2} = 25 - 4x_1 - 2x_2 - 2y_1 - y_2 + y_2' = 0$$

Wprowadzanie zmiennych sztucznych
Zmienne sztuczne w wprowadzane są do każdego ograniczenia zadania zastępczego powstałego z pierwszego warunku KT

$$20x_1 + 4x_2 + y_1 + y_2 - y_1' + w_1 = 10$$
$$4x_1 + 2x_2 + 2y_1 + y_2 - y_2' + w_2 = 25$$

Zmienne sztuczne wprowadzane są do funkcji celu ze współczynnikiem 1, stąd zadanie zastępcze ma postać

$$w_1 + w_2 \leftarrow MIN$$
$$x_1 + 2x_2 + x_1' = 10$$
$$x_1 + x_2 + x_2' = 9$$
$$20x_1 + 4x_2 + y_1 + y_2 - y_1' + w_1 = 10$$
$$4x_1 + 2x_2 + 2y_1 + y_2 - y_2' + w_2 = 25$$
$$x_1, x_2, x_1', x_2', y_1, y_2, y_1', y_2', w_1, w_2 \geqslant 0$$

Pary zmiennych komplementarnych x_i, y_i' x_i', y_i

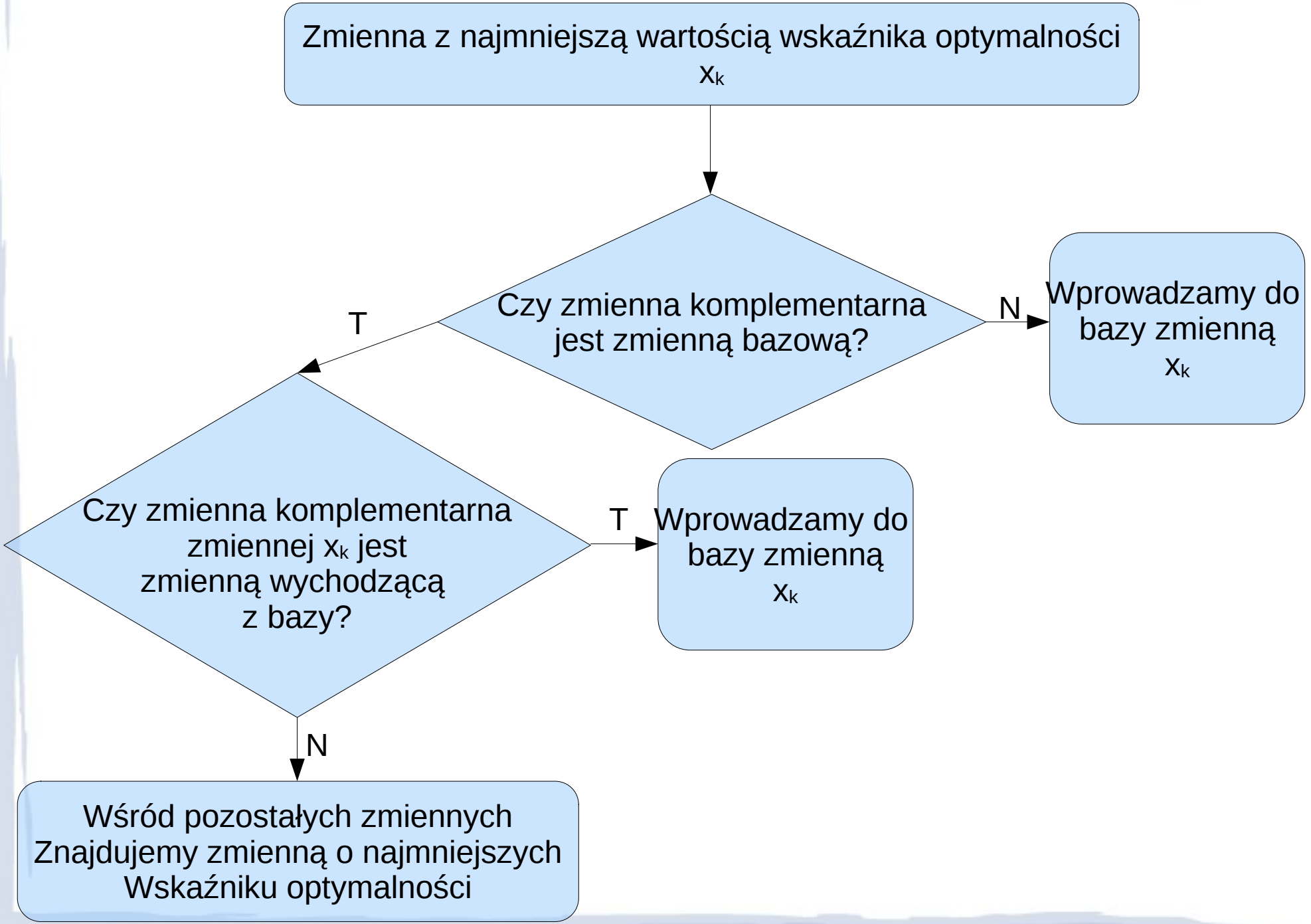
2 warunek KT

$$\lambda_1(10 - x_1 - 2x_2) + \lambda_2(9 - x_1 - x_2) + \lambda_3 x_1 + \lambda_4 x_2 = 0$$

Zapisany w nowych zmiennych ma postać

$$y_1 x_1' + y_2 x_2' + y_1' x_1 + y_2' x_2 = 0$$

Modyfikacja metody SYMPLEX w metodzie Wolfe'a



Rozwiązanie przykładu

Naturalną bazą dla naszego problemu jest [x1',x2',w1,w2]. W tej bazie funkcja, którą minimalizujemy ma postać

$$35-24x_1-6x_2-3y_1-2y_2+ y_1'+ y_2'$$

Tak więc wektor cen zredukowanych

$$c_N=\begin{bmatrix}-24\\-6\\-3\\-2\\1\\1\end{bmatrix}$$

Ponieważ występują wartości ujemne to to rozwiązanie nie jest optymalne, najmniejszą cenę zredukowaną ma zmienna x1, a ponieważ zmienna komplementarna nie jest w bazie to do bazy wchodzi x1

Współczynniki przy x1=[1,1,20,4], wyrazy wolne b=[10,9,10,25], stąd wektor ilorazów Q=[10,9,0.5,6.25], najmniejsza wartość jest dla w1, więc w1 wychodzi z bazy.

Po przejściu do nowej bazy minimalizowana funkcja ma postać

$$23 - 1.2x_2 - 1.8y_1 - 0.8y_2 - 0.2y_1' + y_2' + 1.2w_1$$

Rozwiązanie nie jest optymalne bo współczynniki cen zredukowanych maja wartość ujemną. Najmniejsza wartość jest przy y1 ale y1 nie może wejść do bazy bo jego zmienna Komplementarna w bazie już jest, w związku z tym bierzemy następny najmniejszy czyli x2, jej zmienna komplementarna nie jest w bazie w związku z czym x2 może wejść do bazy.

Zmienna bazowa	Wektor wyrazów wolnych	Kolumna współczynników dla zmiennej wchodzącej do bazy	Wektor ilorazów
X1'	9.5	1.8	5.28
X2'	8.5	0.8	10.6
x1	0.5	0.2	2.5
w2	23	1.2	19.2

Z bazy wychodzi x1

Po przejściu do nowej bazy minimalizowana funkcja ma postać

$$C + 6x_1 - 1.5y_1 - 0.5y_2 - 0.5y_1' + y_2' + 1.5w_1$$

Najmniejszą wartość współczynnika cen zredukowanych ma y1. Zmienna zredukowana Jest już w bazie więc rozpatrujemy kolejny czyli y2, znowu zmienna komplementarna Jest w bazie więc rozpatrujemy y1'. Do bazy wchodzi y1'

Zmienna bazowa	Wektor wyrazów wolnych	Kolumna współczynników dla zmiennej wchodzącej do bazy	Wektor ilorazów
x1'	5	0.5	10
x2'	6.5	0.25	26
x2	2.5	-0.25	-
w2	20	0.5	40

Z bazy wychodzi x1'
Itd..
Procedura powtarzana jest aż do znalezienia rozwiązania. W rozwiązaniu suma zmiennych sztucznych musi być równa 0. W tym przypadku rozwiązaniem jest x1=0, x2=5.

Modele Markowa

Jest to probabilistyczny model sekwencji symboli, w którym w którym prawdopodobieństwo każdego zdarzenia zależy jedynie od wyniku poprzedniego.

Załóżmy, że mamy zbiór stanów $\{s_1, s_2, \ldots, s_n\}$
Załóżmy, że mamy jakiś proces, który polega na generowaniu stanów (przechodzenie z jednego stanu do drugiego): $s_1, s_2, \ldots, s_k$
Korzystając z definicji łańcucha Markowa (pierwszego rzędu)

$$P(s_k|s_{1,}s_{2,}\ldots,s_{k-1}) \approx P(s_k|s_{k-1})$$

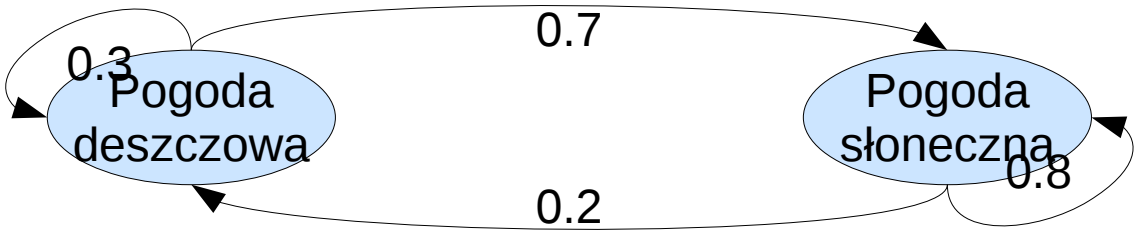
Stąd widać, że do zdefiniowania łańcucha Markowa potrzebne są:
prawdopodobieństwa przejścia z jednego stanu do drugiego $a_{ij} = P(s_i|s_j)$
i prawdopodobieństw początkowych $\pi_i = P(s_i)$

$$P(s_i|s_j) = \frac{P(s_i, s_j)}{P(s_j)} \qquad P(s_i, s_j|s_k) = P(s_i|s_j)P(s_j|s_k)$$

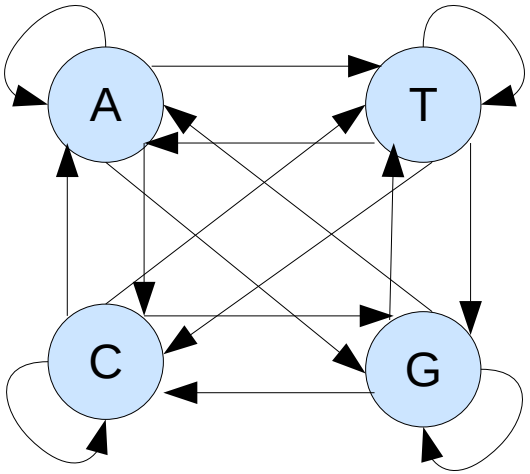
Łańcuch Markowa to model Markowa z czasem dyskretnym

$$P(s_{1,}s_{2,}s_{3,}..,s_n) = \prod_{i=1}^{n} p(s_i|s_{i-1})$$

Przykład modelu Markowa



Występują dwa stany: pogoda deszczowa (sd), pogoda słoneczna (ss)
Prawdopodobieństwa przejścia: P(sd|sd)=0.3, P(ss|sd)=0.2, P(ss|ss)=0.8, P(sd|ss)=0.7
Prawdopodobieństwa początkowe: P(ss)=0.6, P(sd)=0.4



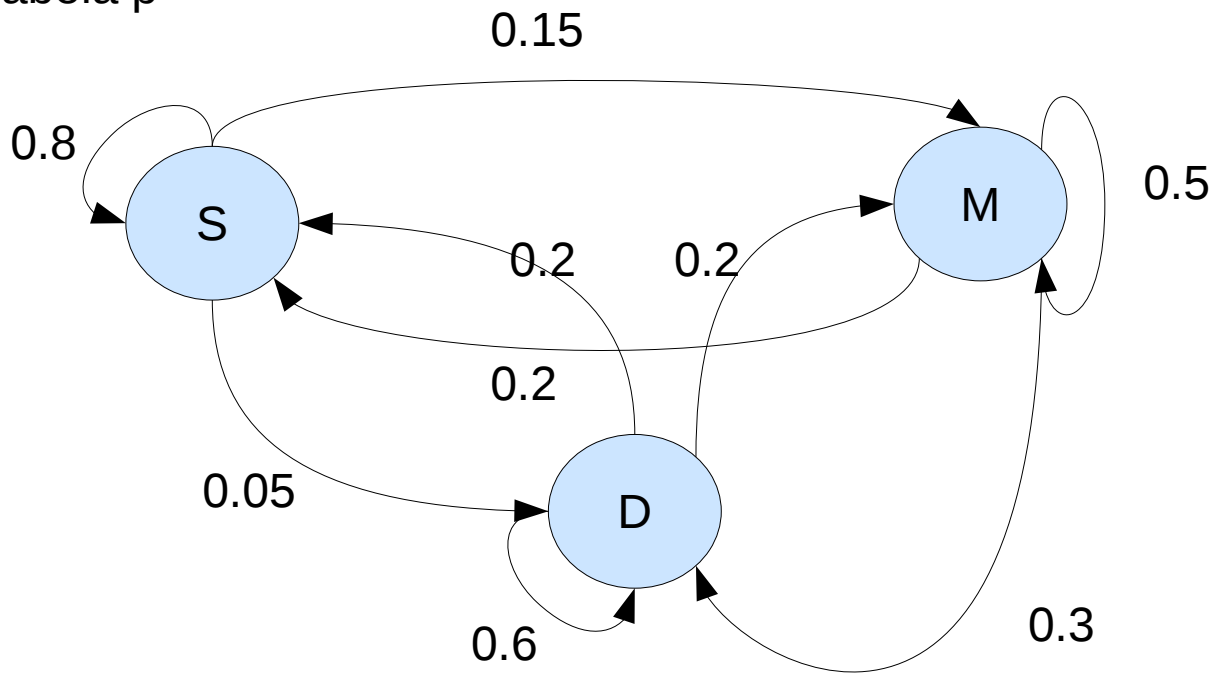
Obliczanie prawdopodobieństwa wystąpienia sekwencji

$$P(s_1, s_2, s_3, .., s_k) = P(s_k | s_1, s_2, s_3, .., s_{k-1}) P(s_1, s_2, s_3, .., s_{k-1})$$
$$= P(s_k | s_{k-1}) P(s_1, s_2, s_3, .., s_{k-1}) = ...$$
$$= P(s_k | s_{k-1}) P(s_{k-1} | s_{k-2}) \cdots P(s_2 | s_1) P(s_1)$$

Załóżmy, że chcemy policzyć prawdopodobieństwo sekwencji stanów { ss,ss,sd,sd}

$$P(ss, ss, sd, sd) = P(ss|ss) P(ss|sd) P(sd|sd) P(sd) = 0.8 * 0.2 * 0.3 * 0.4 = 0.0192$$

Rozpatrzmy sytuację w której pogoda ma 3 stany : słonecznie, deszczowo, mgliście.
Tabela p



		Pogoda jutro		
		S	D	M
Pogoda dzisiaj	S	0.8	0.05	0.15
	D	0.2	0.6	0.2
	M	0.2	0.3	0.5

1)Zakładając, że dzisiaj jest słonecznie jakie jest prawdopodobieństwo tego, że jutro będzie słonecznie a pojutrze deszczowo

$$P(s_2=S, s_3=D|s_1=S)=P(s_3=D|s_1=S, s_2=S)P(s_2=S|s_1=S)=$$
$$P(s_3=D|s_2=S)P(s_2=S|s_1=S)=0.05*0.8$$

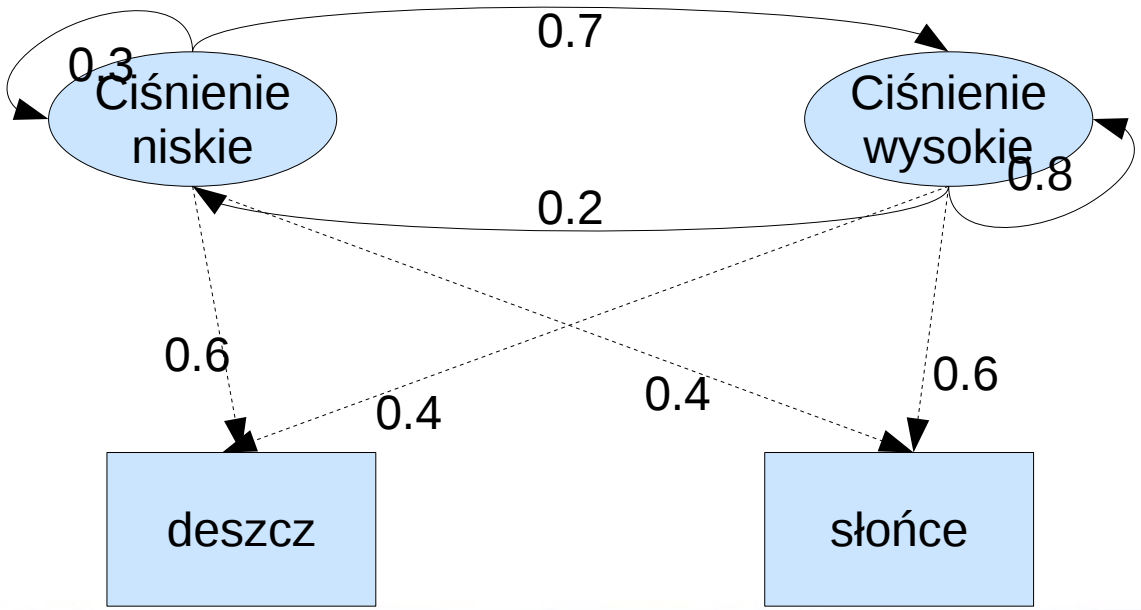
Zakładając, że dzisiaj jest mgliście jakie jest prawdopodobieństwo tego, że pojutrze będzie deszcz

$$P(s_3=D|s_1=M)=P(s_2=M, s_3=D|s_1=M)+P(s_2=D, s_3=D|s_1=M)+P(s_2=S, s_3=D|s_1=M)=$$
$$P(s_3=D|s_2=M)P(s_2=M|s_1=M)+P(s_3=D|s_2=R)P(s_2=M|s_1=M)+$$
$$+P(s_3=D|s_2=S)P(s_2=M|s_1=M)=0.3*0.5+0.6*0.3+0.05*0.2=0.34$$

Ukryte łańcuchy Markowa

Stany nie są widoczne (obserwowalne), ale każdy ze stanów może generować jedną z M obserwacji {v1,v2, … , vn}
Do zdefiniowania ukrytego modelu Markowa oprócz wielkości potrzebnych dla modelu Markowa potrzebna jest jeszcze macierz prawdopodobieństw stanów obserwowanych B=bi(vm), bi(vm)=P(vm|si). Stąd cały model zapisany jest jako

$$M = (A, B, \pi)$$



Mam więc 2 stany : {c niskie, c wysokie}
Dwa stany obserwowane :{deszcz, słońce}
Prawdopodobieństwa początkowe: P(c niskie)=0.4, P(c wysokie)=0.6

Obliczenie prawdopodobieństwa obserwowanej sekwencji {słońce, deszcz}

Do wyliczenia prawdopodobieństwa potrzebne jest rozważenie wszystkich ukrytych stanów

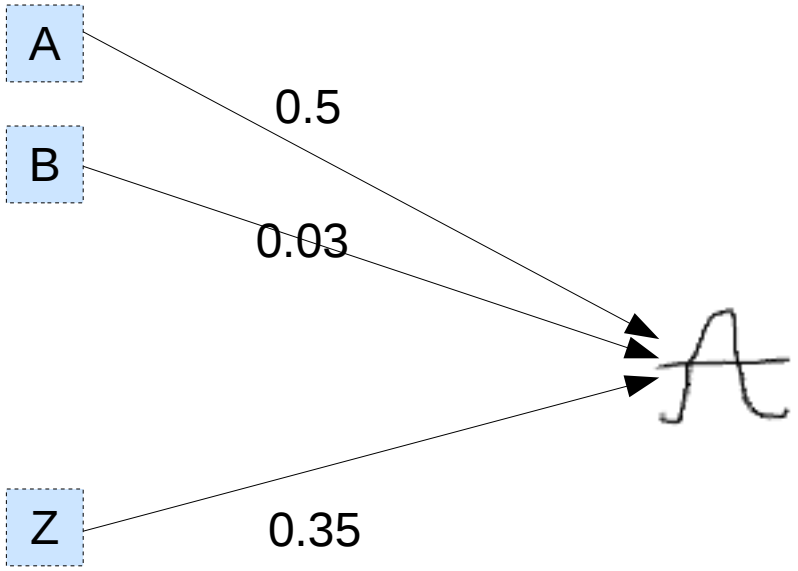
$$P(słońce, deszcz) = P(słońce, deszcz, c\ niskie, c\ niskie)\,P(słońce, deszcz, c\ niskie, c\ wysokie)$$
$$P(słońce, deszcz, c\ wysokie, c\ niskie)\,P(słońce, deszcz, c\ wysokie, c\ wysokie)$$

Główne problemy związany z używaniem HMM

1) **Oceny**
   Przy zadanym modelu HMM M=(A,B,π) i obserwowanej sekwencji O=o1, o2, o3, …, ok wylicz prawdopodobieństwo wygenerowania sekwencji O przy użyciu modelu M.
2) **Dekodowanie**
   Mając dany model HMM M=(A,B,π) i obserwowaną sekwencję O, znajdź najbardziej prawdopodobną sekwencję stanów ukrytych, która produkuje sekwencję obserwowaną
3) **Uczenie**
   Mając daną treningową sekwencję obserwowaną i ogólną strukturę modelu HMM (np. liczbę obserwowanych i ukrytych stanów) wyznacz parametry modelu HMM tak aby jak najlepiej opisywały dane treningowe.

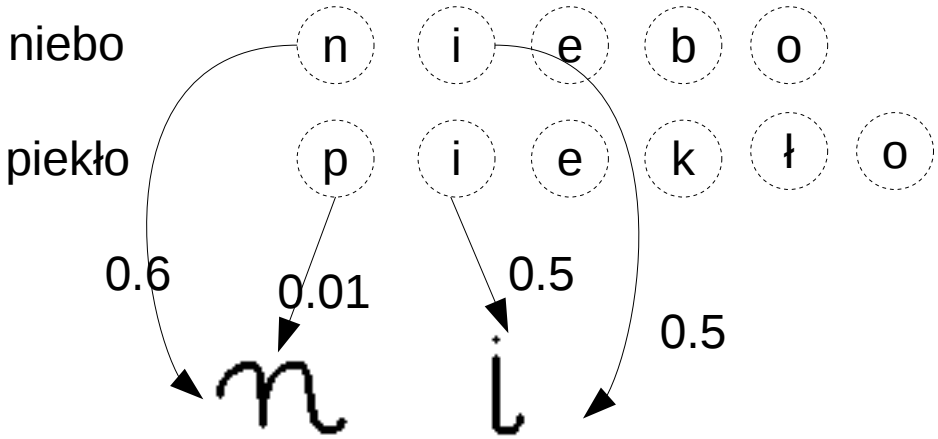
Przykład zastosowania HMM

Najczęstszym przykładem zastosowania HMM, który podaje się w literaturze jest rozpoznawanie słów. Dla prostoty zakładamy, że każda literka w słowie jest pisana oddzielnie



W tym przypadku stanami ukrytymi są litery alfabetu występujące w danym słowie, natomiast stanami obserwowanymi obrazki z literkami. Liczba stanów ukrytych jest skończona, natomiast liczba stanów obserwowanych jest nieskończona.

Jeżeli mam zadany zbiór wyrazów to dla każdego wyrazu w zbiorze możemy skonstruować osobny model HMM



Tak więc rozpoznanie wyrazu wymaga oceny kilku modeli HMM. Jest to przykład problemu oceny.

Można jednak skonstruować jeden HMM dla wszystkich wyrazów. W takim przypadku stany ukryte to po prostu wszystkie litery w danym języku. Prawdopodobieństwa przejścia i prawdopodobieństwa początkowe wyliczane są z modelu językowego. Natomiast Stany obserwowane i ich prawdopodobieństwa wyznaczane są tak jak poprzednio z obrazków.
W tym wypadku musimy wyznaczyć najlepszą sekwencję stanów ukrytych, która jest najbardziej prawdopodobna (najbliżej odpowiada obrazkowi słowa).
Jest to tzw. problem dekodowania.