Liczba skoków posetu przedziałowego. Algorytmy dokładne

Przemysław Krysztowiak, Wydział Matematyki i Informatyki UMK w Toruniu

FIT 2013, 11-14 kwietnia 2013 r., Toruń

Liczba skoków posetu

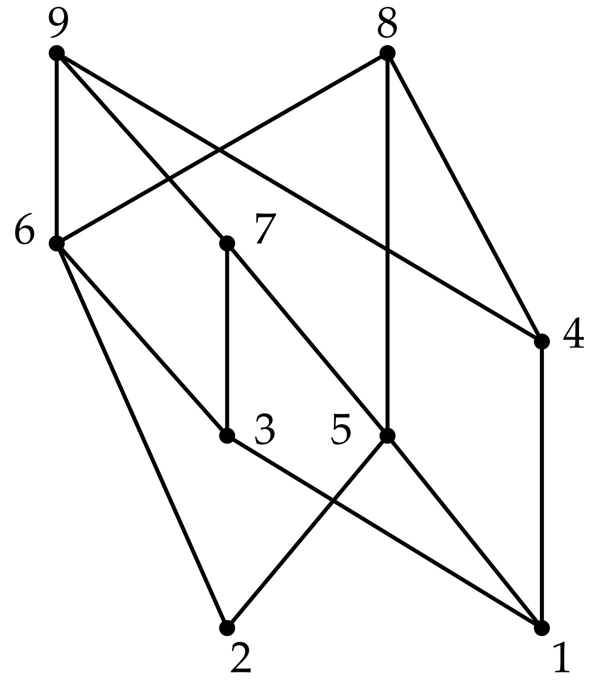
Rozszerzenie liniowe posetu $(P, <_P)$:

$$L = p_1, p_2, \ldots, p_n$$

całkowite uporządkowanie, zachowujące porównywalności, tj.

$$p_i <_P p_j \implies i < j.$$

Problem skoków



Pomiędzy łańcuchami są **skoki**.
Wewnątrz łańcuchów są **progi**.

- $(2) + (1, 5) + (3, 7) + (6) + (4, 9) + (8)$ – 5 skoków, 3 progi.
- $(1, 4) + (2, 5) + (3, 6, 8) + (7, 9)$ – 3 skoki, 5 progów.

Cel: minimalizacja liczby skoków.

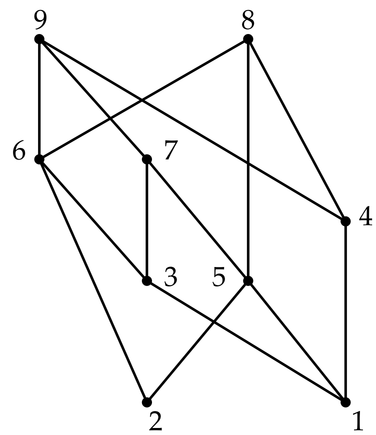
Stan wiedzy nt. problemu

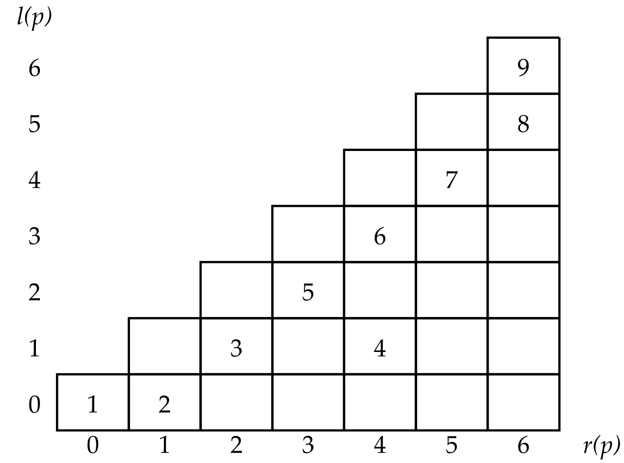
Problem wyznaczenia liczby skoków jest:

- NP-zupełny na posetach przedziałowych.
- NP-zupełny na posetach wysokości 1.
- Wielomianowy na posetach N-wolnych.
- Wielomianowy na posetach przedziałowych, gdy przedziały są długości 1.

- Algorytm dokładny: Sysło, 1988; czas $\mathcal{O}^\star(k!)$, k – liczba łuków pozornych.
- Algorytmy aproksymacyjne tylko dla posetów przedziałowych (Felsner 1990, Mitas 1991, Sysło 1992).

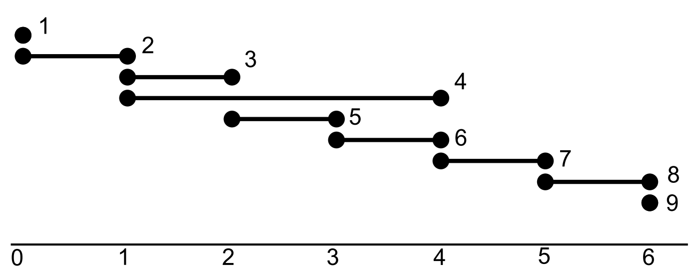
Posety przedziałowe



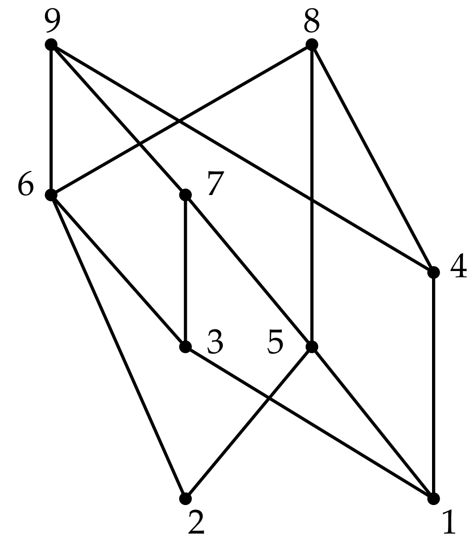


$(P, <_P)$ zdefiniowany przedziałami: $p <_P q$ wtw, gdy p jest na lewo od q.

Kanoniczna reprezentacja za pomocą zwartej rodziny przedziałów, wpisanych do tabeli rozmiaru $e \times e$.

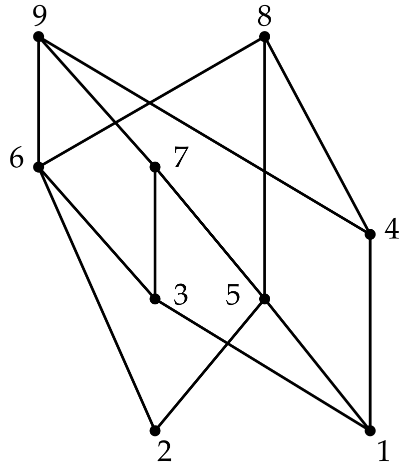


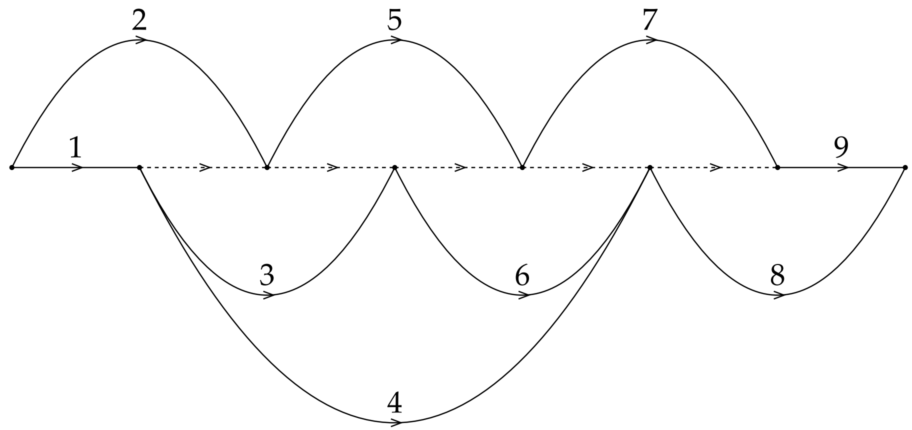
Zachłanne rozszerzenia liniowe

(1, 3, 2, 5, 7, 4, 6, 8, 9): 4	(2, 1, 3, 6, 4, 5, 7, 9, 8): 4	
(1, 3, 2, 5, 7, 4, 6, 9, 8): 4	(2, 1, 3, 6, 4, 5, 8, 7, 9): 4	
(1, 3, 2, 5, 7, 6, 4, 8, 9): 4	(2, 1, 3, 6, 5, 7, 4, 8, 9): 4	
(1, 3, 2, 5, 7, 6, 4, 9, 8): 4	(2, 1, 3, 6, 5, 7, 4, 9, 8): 4	
(1, 3, 2, 6, 4, 5, 7, 9, 8): 4	(2, 1, 4, 3, 6, 5, 7, 9, 8): 4	
(1, 3, 2, 6, 4, 5, 8, 7, 9): 4	(2, 1, 4, 3, 6, 5, 8, 7, 9): 4	
(1, 3, 2, 6, 5, 7, 4, 8, 9): 4	(2, 1, 4, 5, 3, 6, 8, 7, 9): 4	
(1, 3, 2, 6, 5, 7, 4, 9, 8): 4	(2, 1, 4, 5, 3, 7, 6, 8, 9): 5	
(1, 3, 4, 2, 5, 7, 6, 8, 9): 4	(2, 1, 4, 5, 3, 7, 6, 9, 8): 5	
(1, 3, 4, 2, 5, 7, 6, 9, 8): 4	(2, 1, 5, 3, 6, 4, 8, 7, 9): 4	
(1, 3, 4, 2, 6, 5, 7, 9, 8): 4	(2, 1, 5, 3, 6, 7, 4, 8, 9): 5	
(1, 3, 4, 2, 6, 5, 8, 7, 9): 4	(2, 1, 5, 3, 6, 7, 4, 9, 8): 5	
(1, 4, 2, 5, 3, 6, 8, 7, 9): 3	(2, 1, 5, 3, 7, 4, 6, 8, 9): 5	
(1, 4, 2, 5, 3, 7, 6, 8, 9): 4	(2, 1, 5, 3, 7, 4, 6, 9, 8): 5	
(1, 4, 2, 5, 3, 7, 6, 9, 8): 4	(2, 1, 5, 3, 7, 6, 4, 8, 9): 5	
(1, 4, 3, 2, 5, 7, 6, 8, 9): 4	(2, 1, 5, 3, 7, 6, 4, 9, 8): 5	
(1, 4, 3, 2, 5, 7, 6, 9, 8): 4	(2, 1, 5, 4, 3, 6, 8, 7, 9): 4	
(1, 4, 3, 2, 6, 5, 7, 9, 8): 4	(2, 1, 5, 4, 3, 7, 6, 8, 9): 5	
(1, 4, 3, 2, 6, 5, 8, 7, 9): 4	(2, 1, 5, 4, 3, 7, 6, 9, 8): 5	

Algorytm Sysły

Diagram łukowy





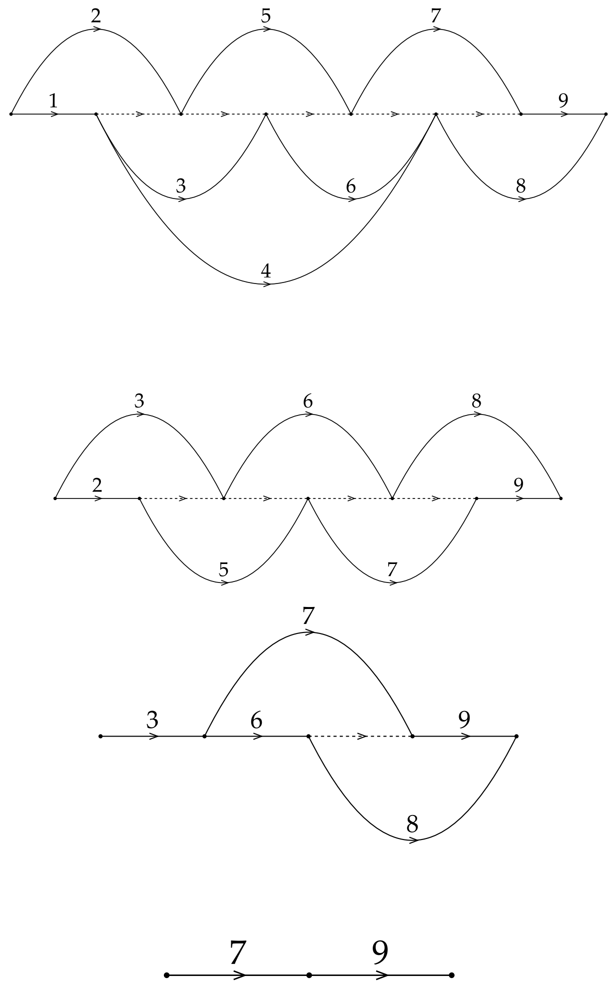
- Elementy posetu – łuki.
- Łańcuch w posecie – ścieżka w diagramie.
- Łuki pozorne niezbędne do zachowania relacji wzdłuż ścieżek.

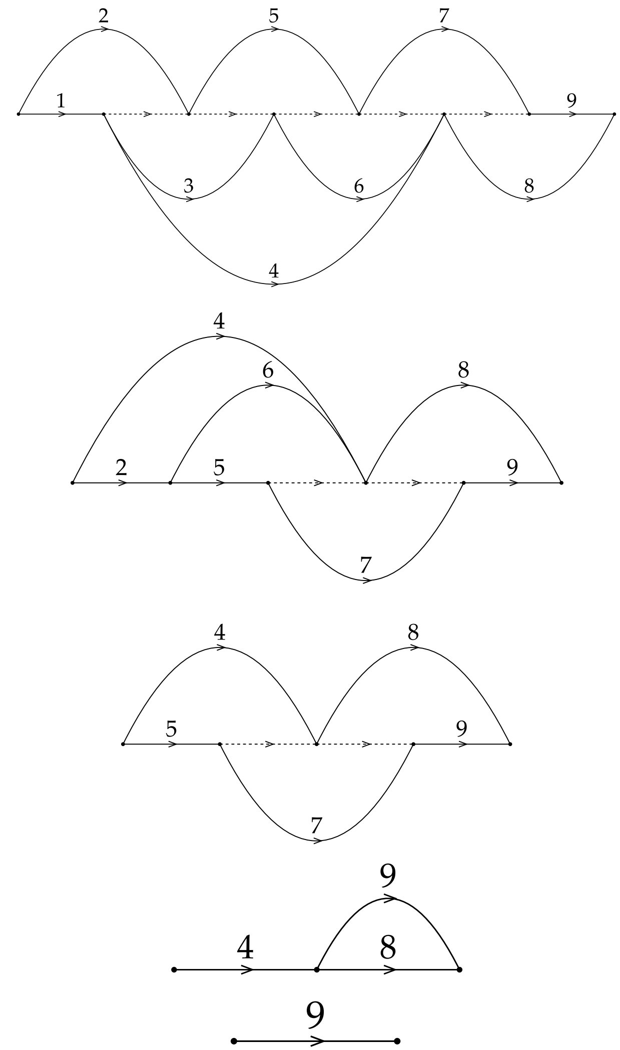
Idea algorytmu

- Strategia zachłanna: w każdym kroku następuje wybór łańcucha zachłannego (ścieżki).
- Ścieżki **silnie zachłanne** „prowadzą do optimum” (można wziąć dowolną).
- Ścieżki **półsilnie zachłanne** mogą prowadzić do optimum (nie wiadomo, którą z nich wybrać).

- **Algorytm dokładny**: przegląd wszystkich rozszerzeń liniowych złożonych ze ścieżek silnie i półsilnie zachłannych.

Przykład





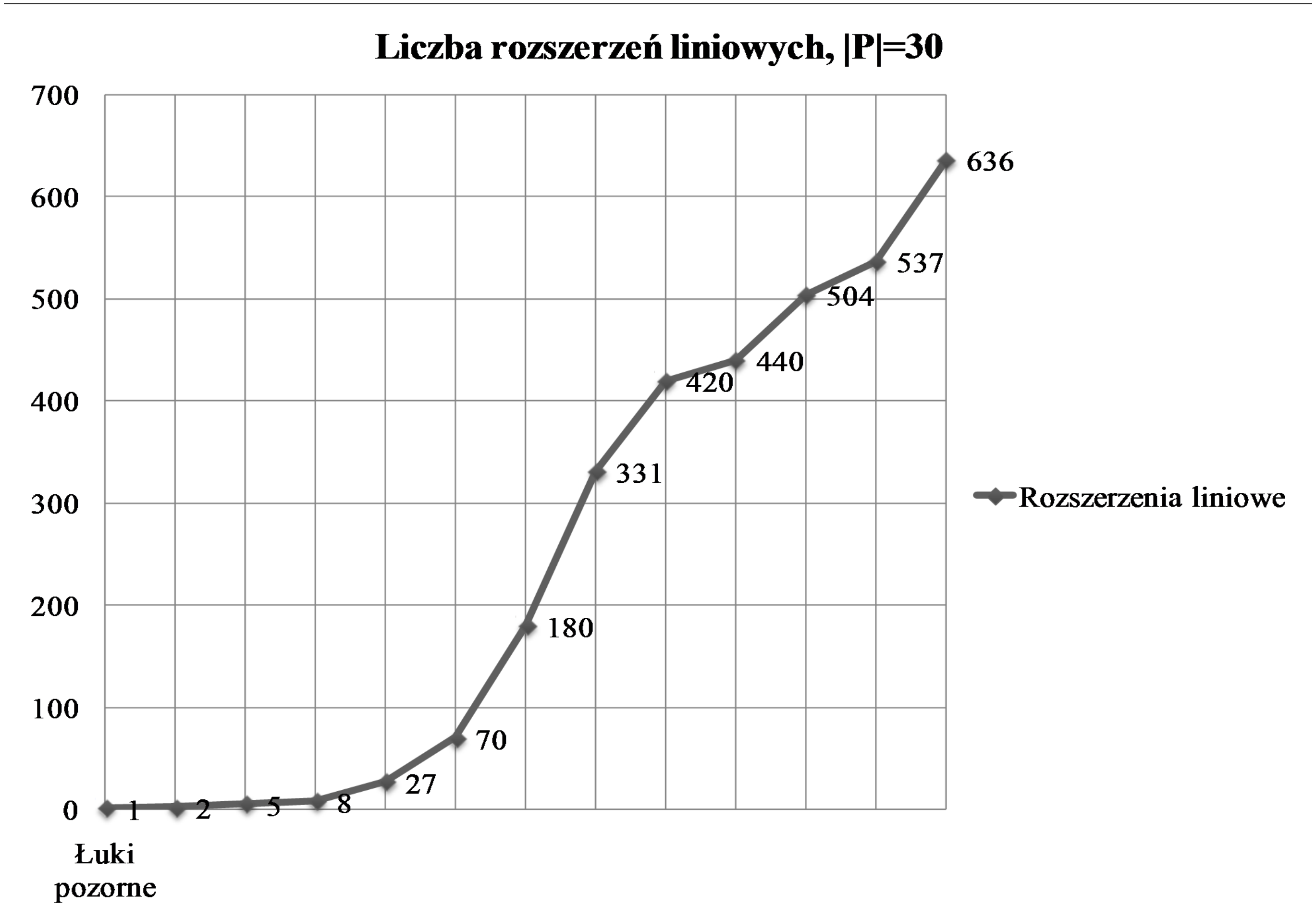
$(1,4)+(2,5)+(3,6,8)+(7,9)$ $\qquad$ $(1,3)+(2,6)+(5,7)+(4,8)+(9)$

Złożoność czasowa

- Diagram ma k łuków pozornych $\longrightarrow$ co najwyżej k ścieżek półsilnie zachłannych.
- Usunięcie ścieżki półsilnie zachłannej $\longrightarrow$ nowy diagram ma co najwyżej $k-1$ łuków pozornych.
- $\longrightarrow$ co najwyżej $k!$ przeglądanych rozszerzeń liniowych.

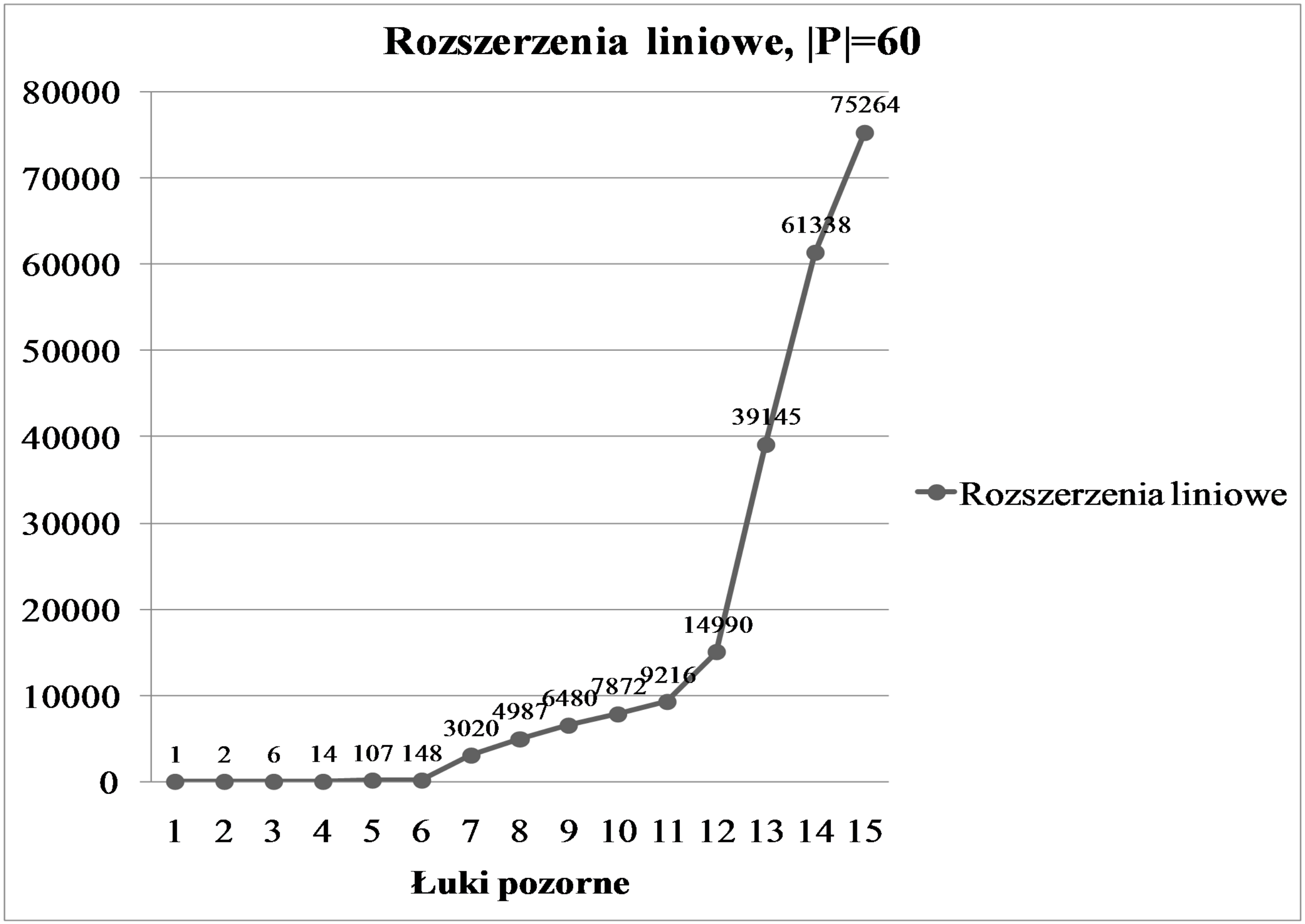
- Często redukuje się więcej łuków za jednym razem.

Czas działania (i)



- Czas: od 3ms (1 łuk pozorny) do 82ms (13 łuków pozornych).

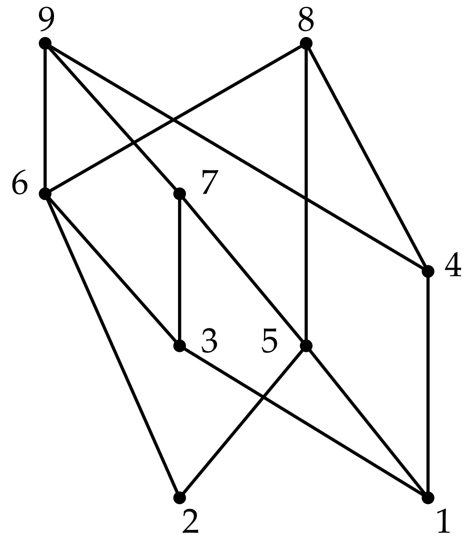
Czas działania (ii)



- Czas: od 8ms (1 łuk pozorny) do 1127ms (15 łuków pozornych).

Algorytm dla posetów przedziałowych

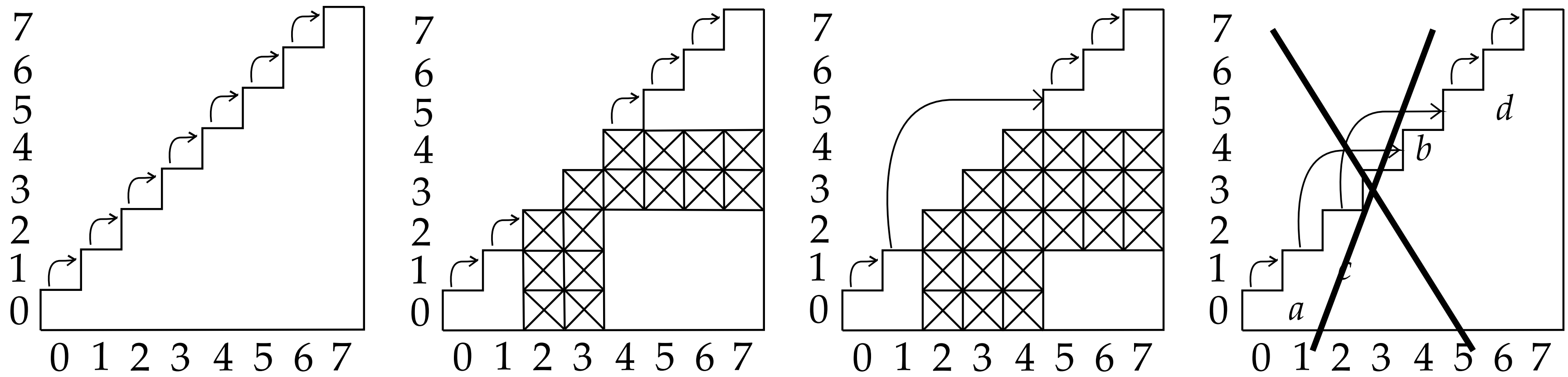
Relacja w tabeli



$l(p)$							
6							9
5							8
4						7	
3					6		
2				5			
1			3		4		
0	1	2					
	0	1	2	3	4	5	6 $r(p)$

Następniki elementów z kolumny i leżą w wierszach $i+1$ i wyższych.

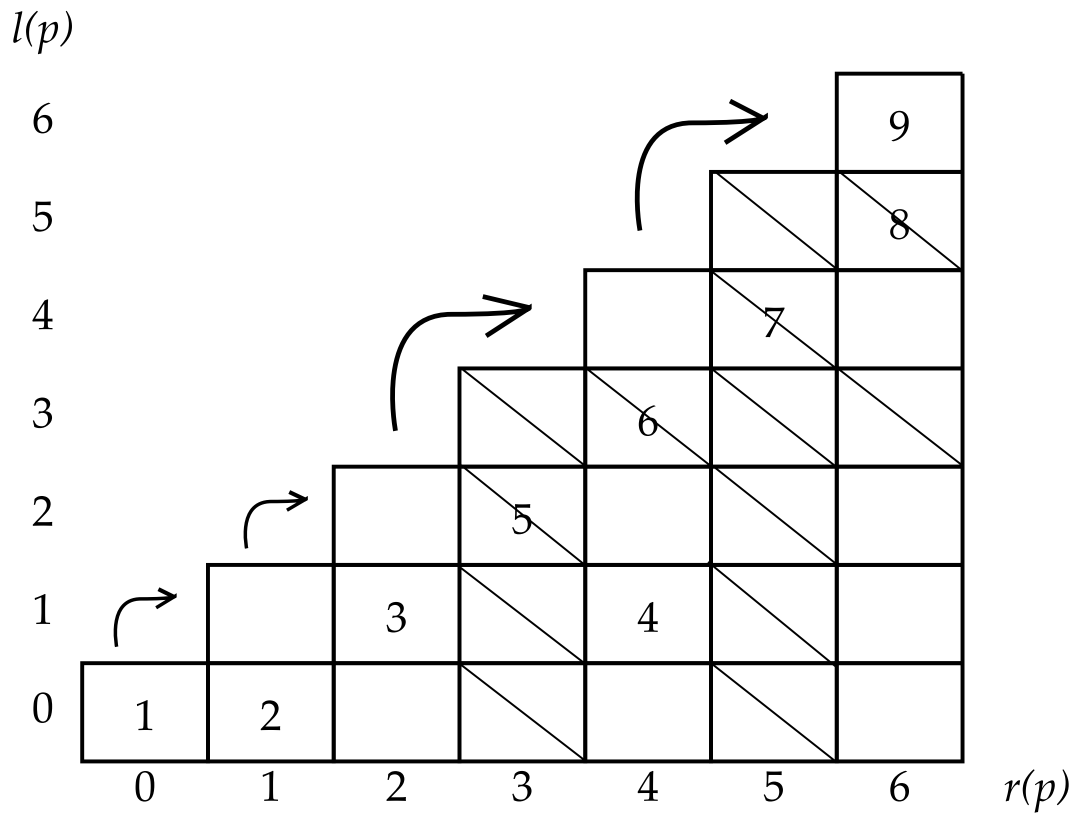
Ciąg progów



- **Ciąg progów** to szablon dla rozszerzenia liniowego.
- Jest w pełni wyznaczony przez **omijane wiersze i kolumny**.
- Im dłuższy ciąg progów, tym więcej progów w rozszerzeniu liniowym, pod warunkiem, że jest on **realizowalny**.

- Omijane wiersze i kolumny zawsze składają się z **rozłącznych** par (col_i, row_i) lub (col_i, row_{i+1}).

Przykład 1

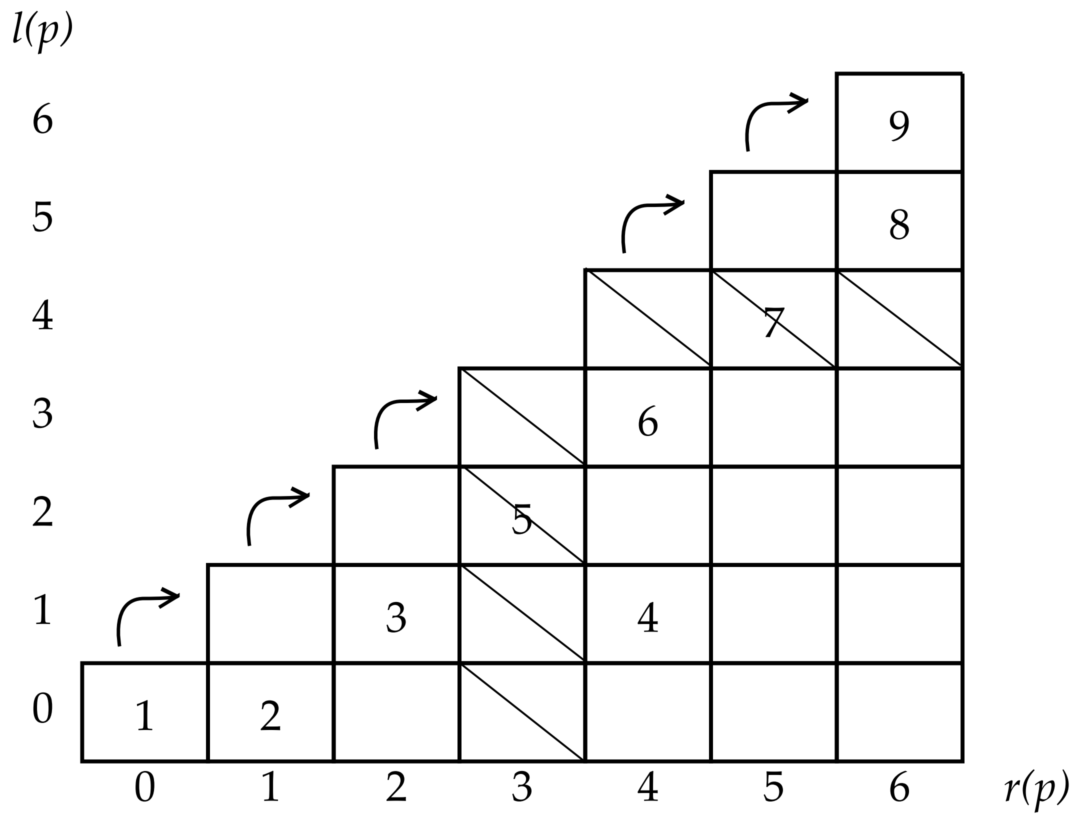


Ciąg progów $(0,1),(1,2),(2,4),(4,6)$ jest realizowalny jako:

$$L_1 = (1,4) + (2,5) + (3,7) + (6,9) + (8),$$

omijane pary to: (col_3, row_3) i (col_5, row_5).

Przykład 2

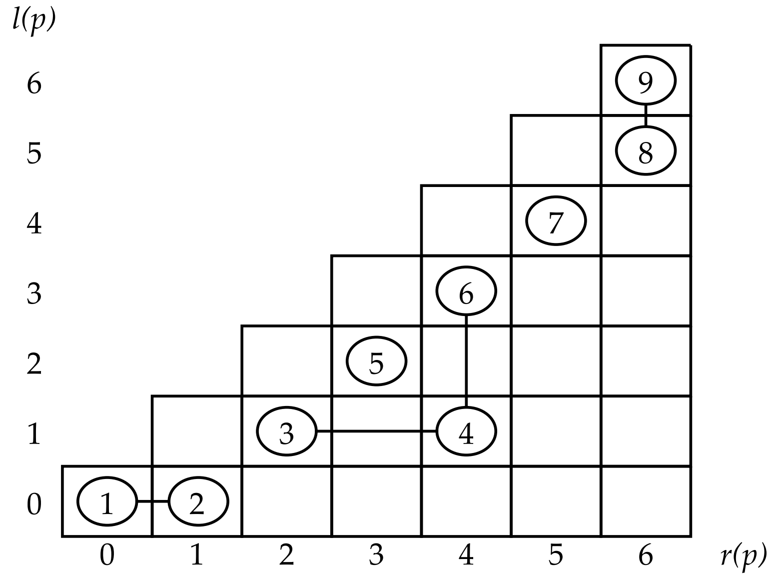


Ciąg progów $(0,1),(1,2),(2,3),(4,5),(5,6)$ jest realizowalny jako:

$$L_2 = (1,4) + (2,5) + (3,6,8) + (7,9),$$

omijaną parą jest: (col_3, row_4).

Graf przedziałów



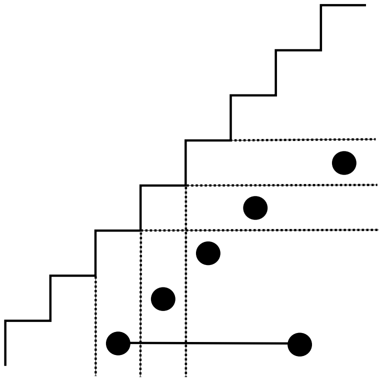
- **Wierzchołki** $\equiv$ niepuste komórki tabeli.
- **Krawędź** łączy sąsiadujące wierzchołkiw w jednym wierszu lub kolumnie.
- **Składowa** jest **nienasycona**, jeśli nie ma wierzchołka na obrzeżu tabeli, ani elementu wielokrotnego, ani cyklu.
- Jest u nienasyconych składowych.

Twierdzenie (Mitas). Ciąg progów T jest realizowalny, jeśli każda nienasycona składowa C spełnia:

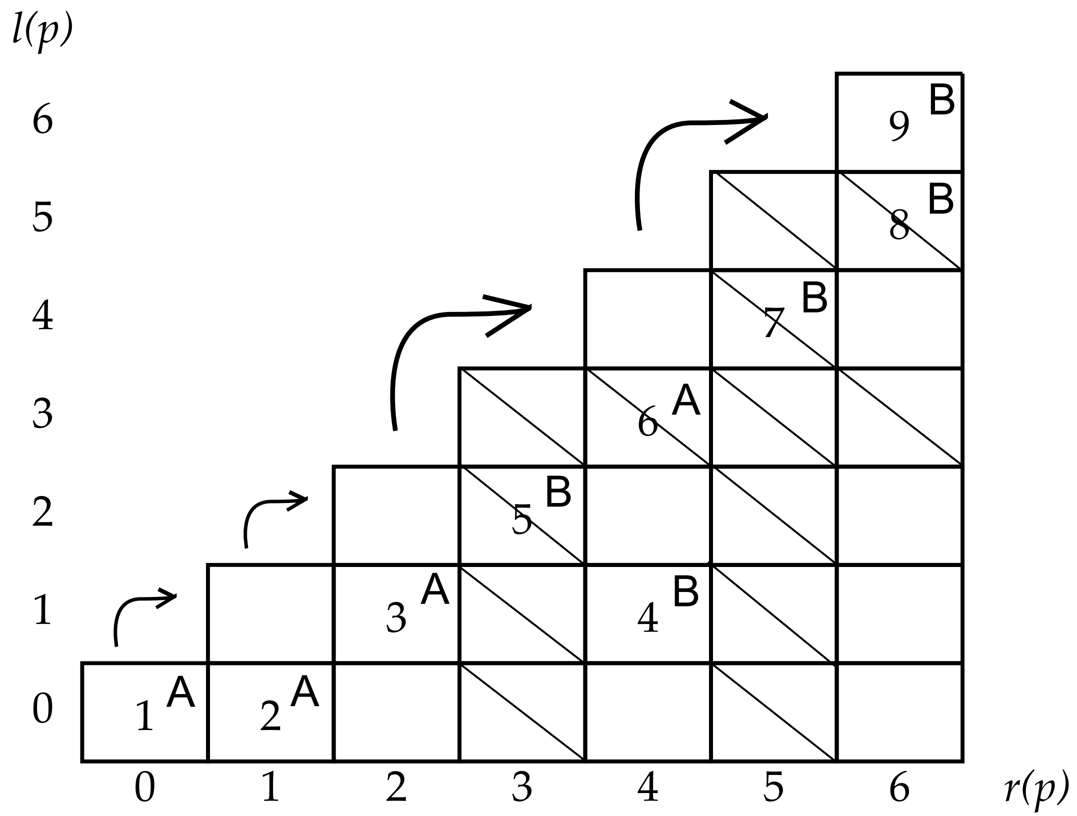
$(P1)$ C ma wierzchołek w kolumnie lub wierszu omijanym przez T,

lub

$(P2)$ C ma element $[j, j+q]$ tż. kolumny $j, \ldots, j+q-1$ i wiersze $j+1, \ldots, j+q$ są omijane przez T.



Przykład 1'

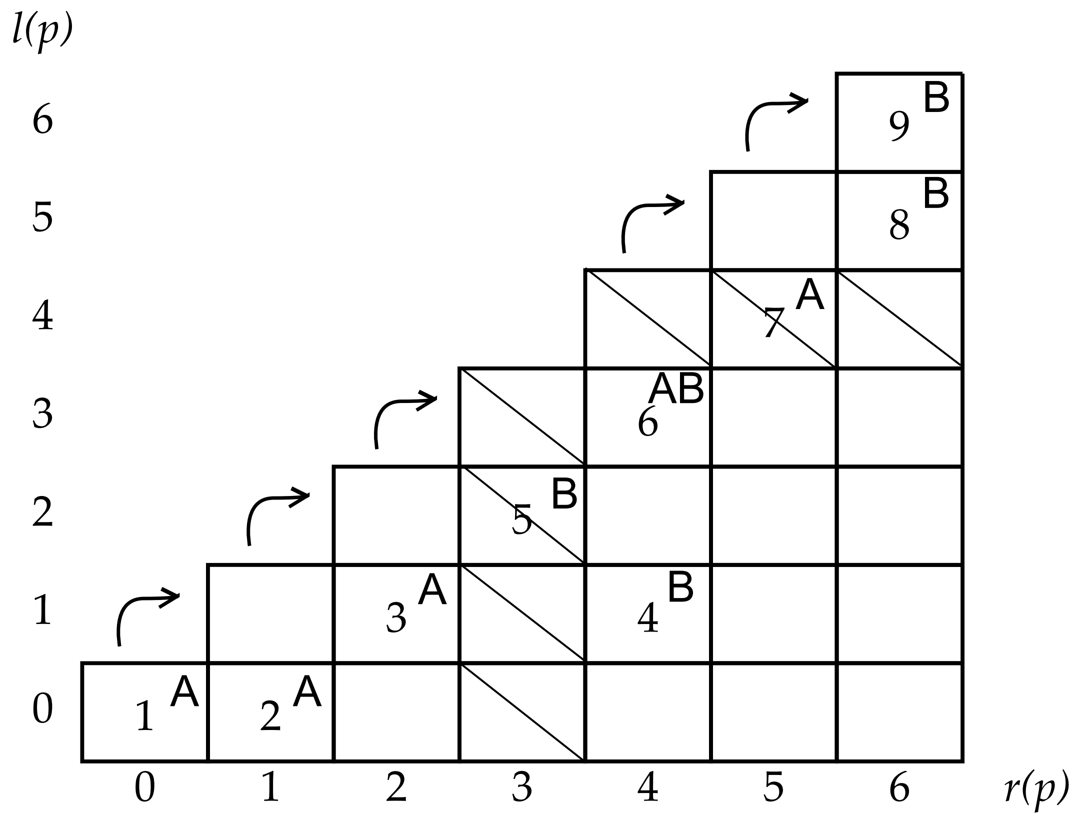


Ciąg progów $(0,1), (1,2), (2,4), (4,6)$ jest realizowalny jako:

$$L_1 = (1,4) + (2,5) + (3,7) + (6,9) + (8),$$

omijane pary to: (col_3, row_3) i (col_5, row_5).

Przykład 2'



Ciąg progów $(0,1),(1,2),(2,3),(4,5),(5,6)$ jest realizowalny jako:

$$L_2 = (1,4) + (2,5) + (3,6,8) + (7,9),$$

omijaną parą jest: (col_3, row_4).

Przeliczanie ciągów progów

- **Algorytm**: przeliczyć wszystkie ciągi progów dla danego e. Sprawdzić realizowalność i wybrać najdłuższy.
  $\longrightarrow$ Przeliczyć pary (col_i, row_i) i (col_i, row_{i+1}) omijane przez ciąg progów.
- Oczywiste oszacowanie: e par typu (col_i, row_i) oraz e par typu (col_i, row_{i+1})
  $\longrightarrow$ $\mathcal{O}(4^e)$ potencjalnych rozwiązań.

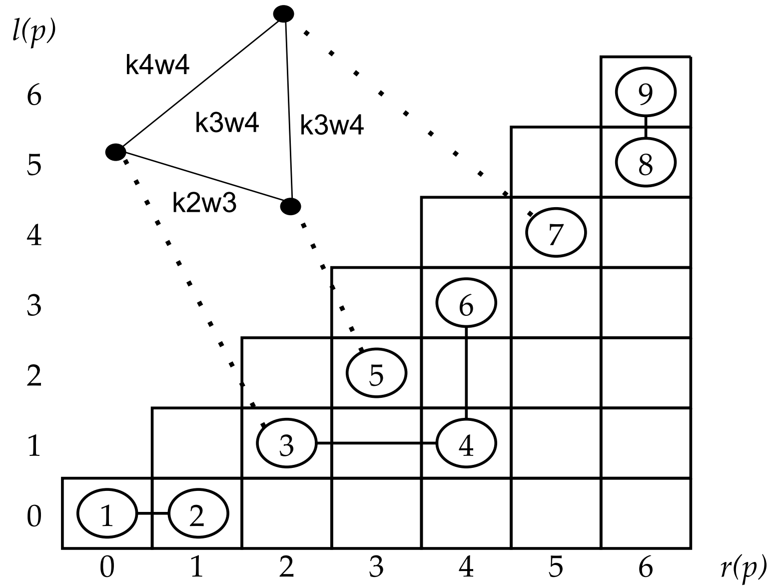
- **Obserwacja**: graf zależności pomiędzy tymi parami jest ścieżką długości $2e$.
- Niezależność (rozłączność) tych par $\longrightarrow$ generowanie słowa binarnego bez sąsiadujących jedynek $\longrightarrow$ $\mathcal{O}(2.62^e)$ ciągów progów.

($e \leqslant |P|$ jest rozmiarem tabeli).

Pakowanie podgrafów

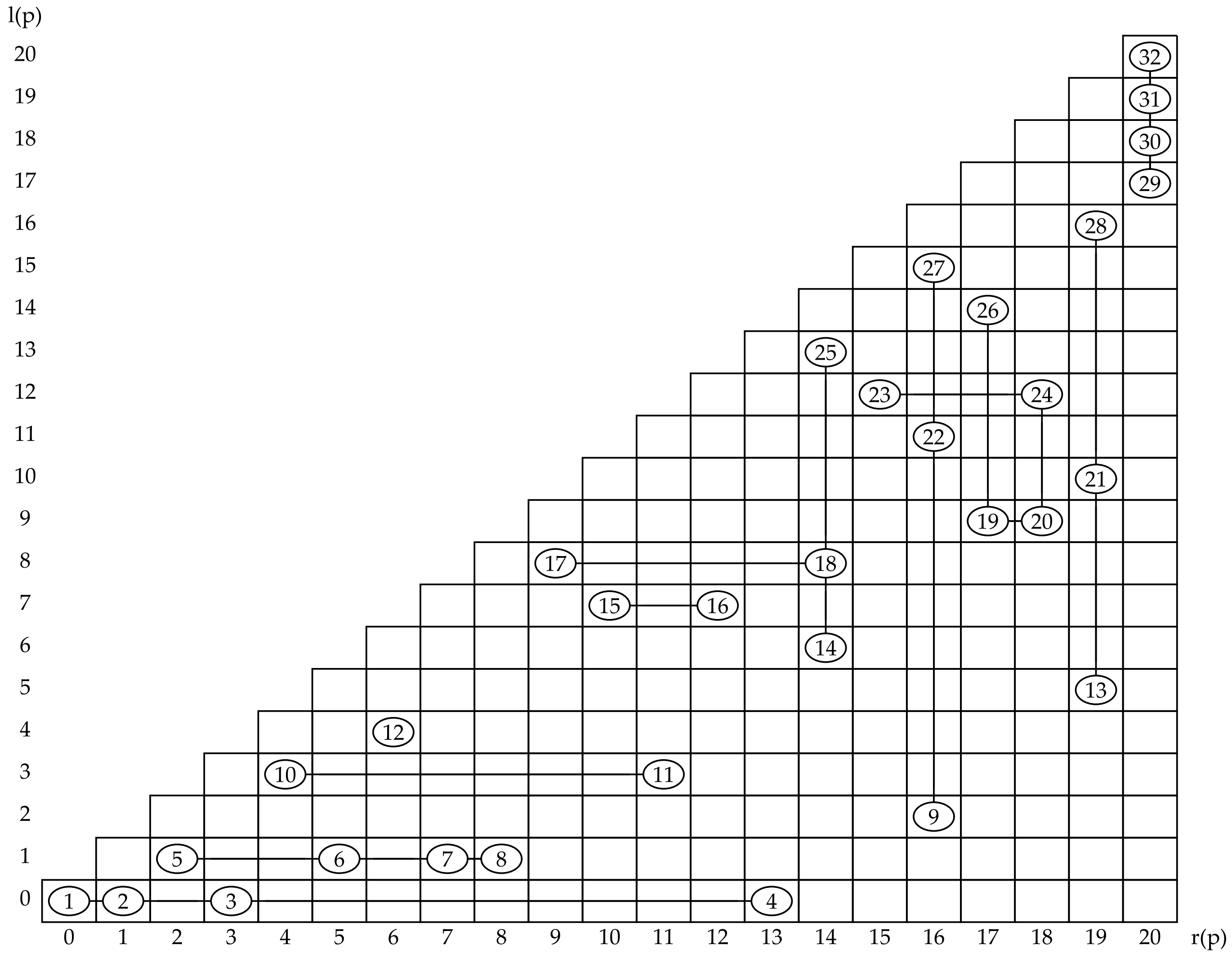
- Definiujemy **graf nienasyconych składowych**.
- **Wierzchołki** $\equiv$ nienasycone składowe.
- **Krawędź** kładziemy, gdy dwie składowe spełniają$(P1)$ kosztem jednej pary(col_i, row_i) lub (col_i, row_{i+1}).
- $(P2)$ pojawia się jako **cykl nieparzysty**!
- **Cel**: optymalny zbiór rozłącznych cykli i krawędzi.
- **Obserwacja**: cykli jest co najwyżej e.

Przeliczanie upakowań



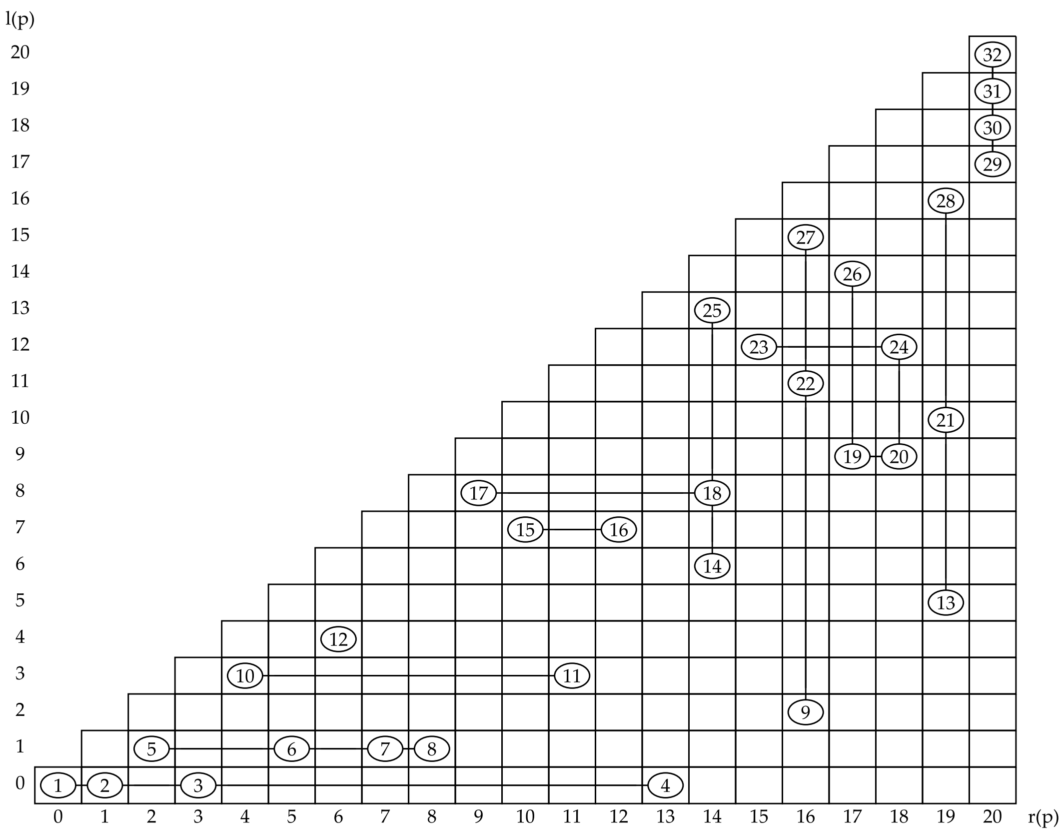
- Wystarczy znaleźć **optymalne upakowanie** cykli i krawędzi.
- Gdy ustalimy podzbiór wybranych cykli, na pozostałym podgrafie szukamy najliczniejszego skojarzenia. $\longrightarrow \mathcal{O}(2^e)$ przeliczanych rozwiązań.
- Dla każdego upakowania obliczamy długość odpowiadającego ciągu progów i wybieramy najdłuższy.
- Często cykli jest wyraźnie mniej niż e.

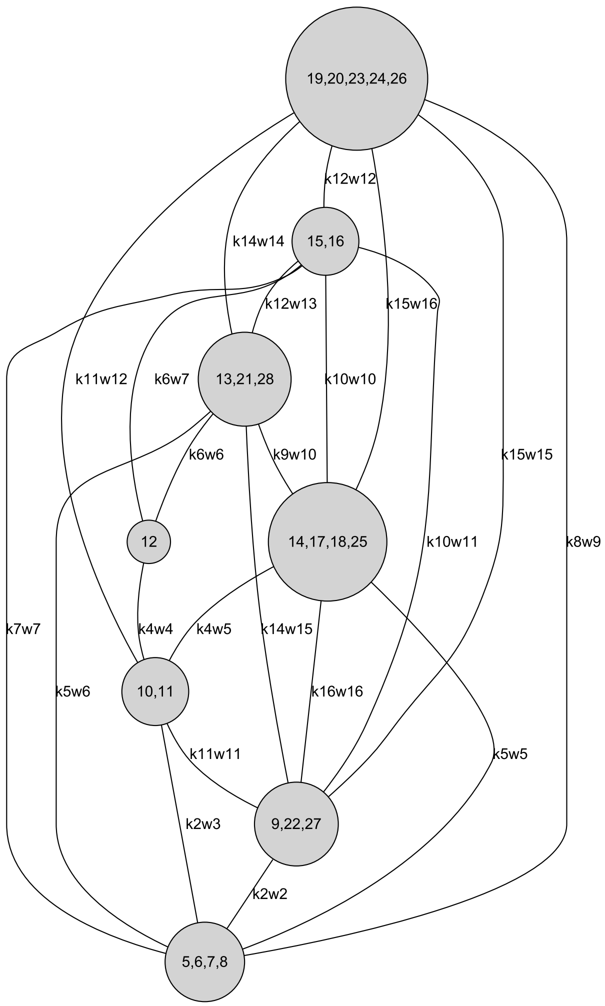
Przykład 3 (i)



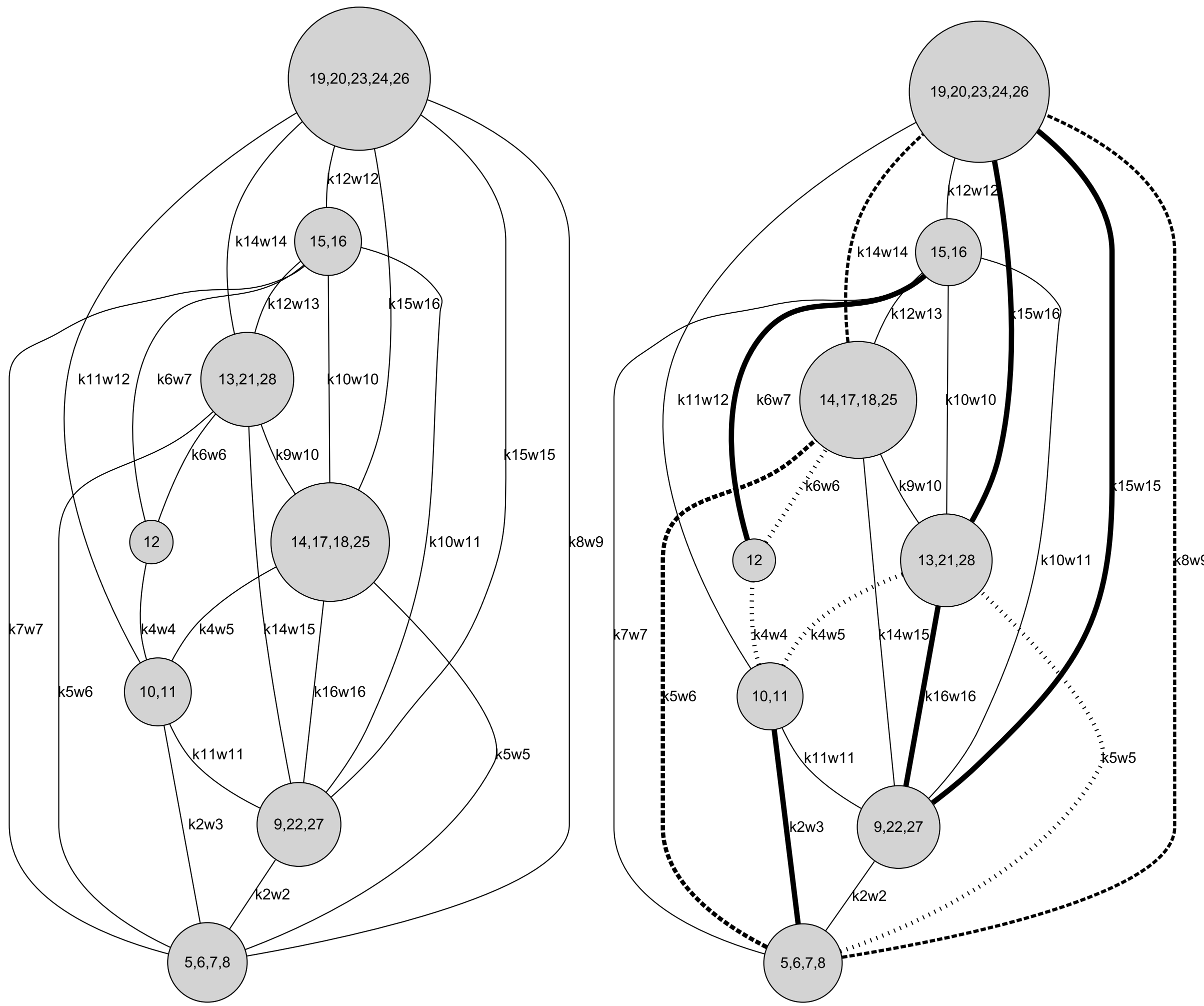
l(p)
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
r(p)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

Przykład 3 (ii)

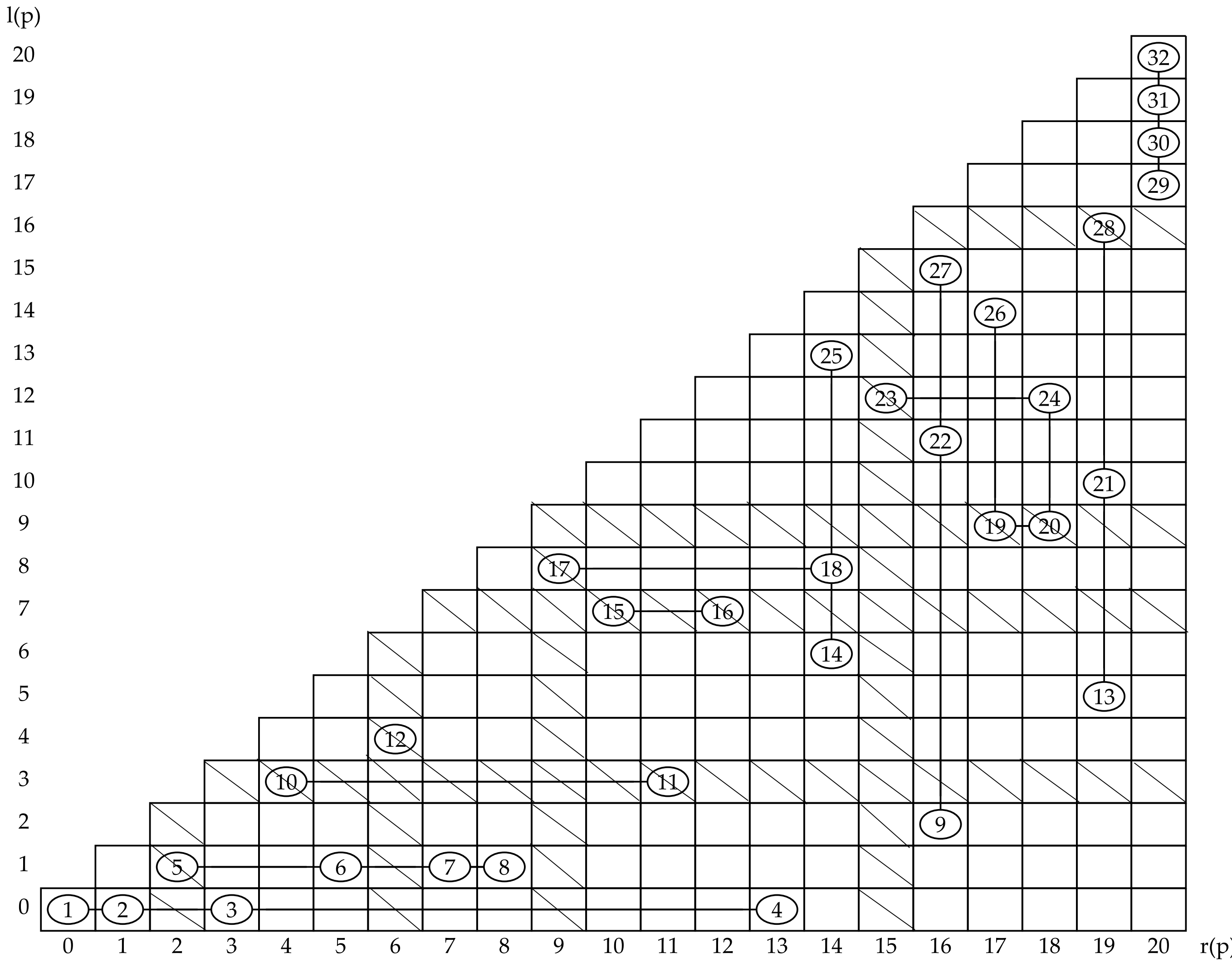




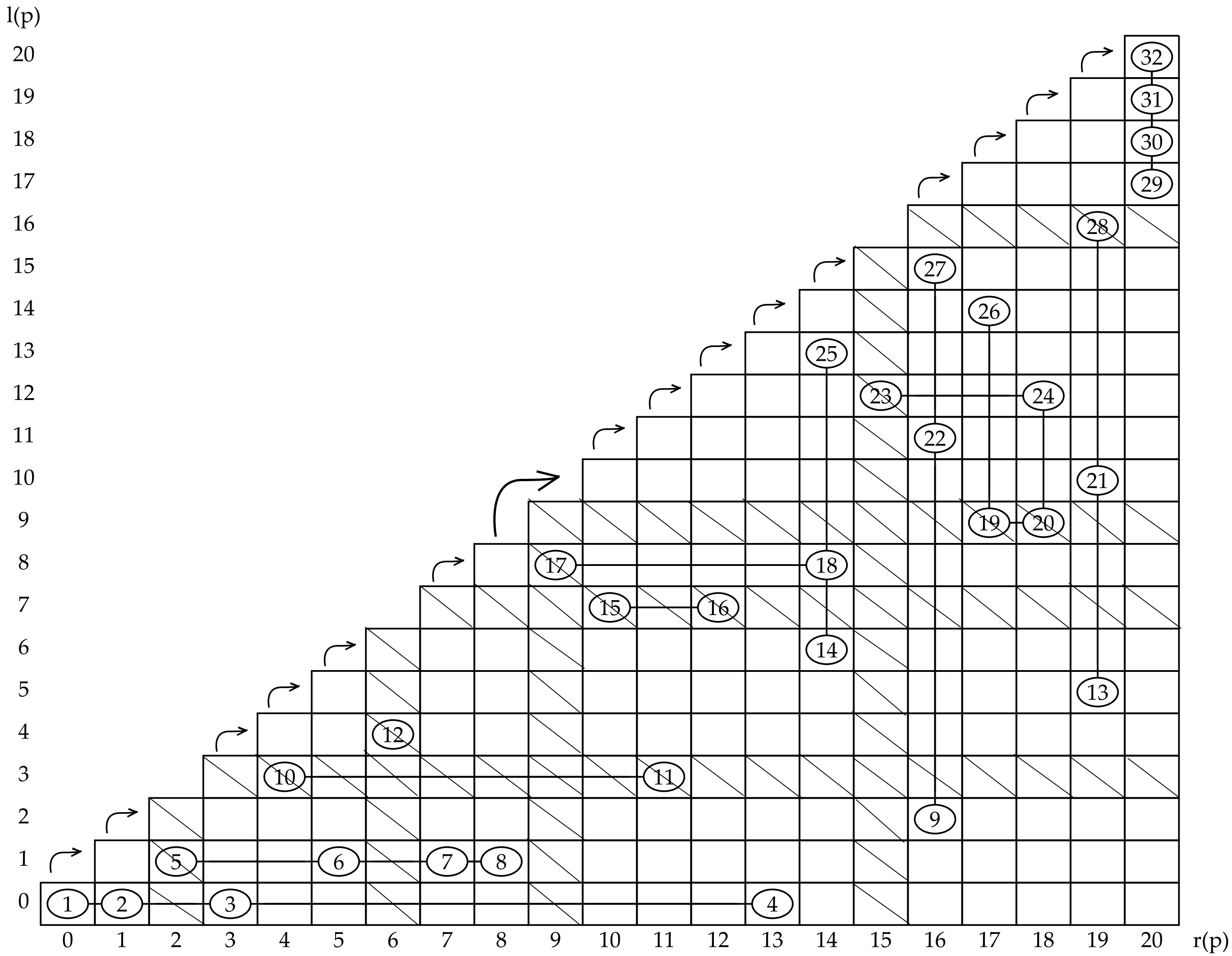
Przykład 3 (iii)



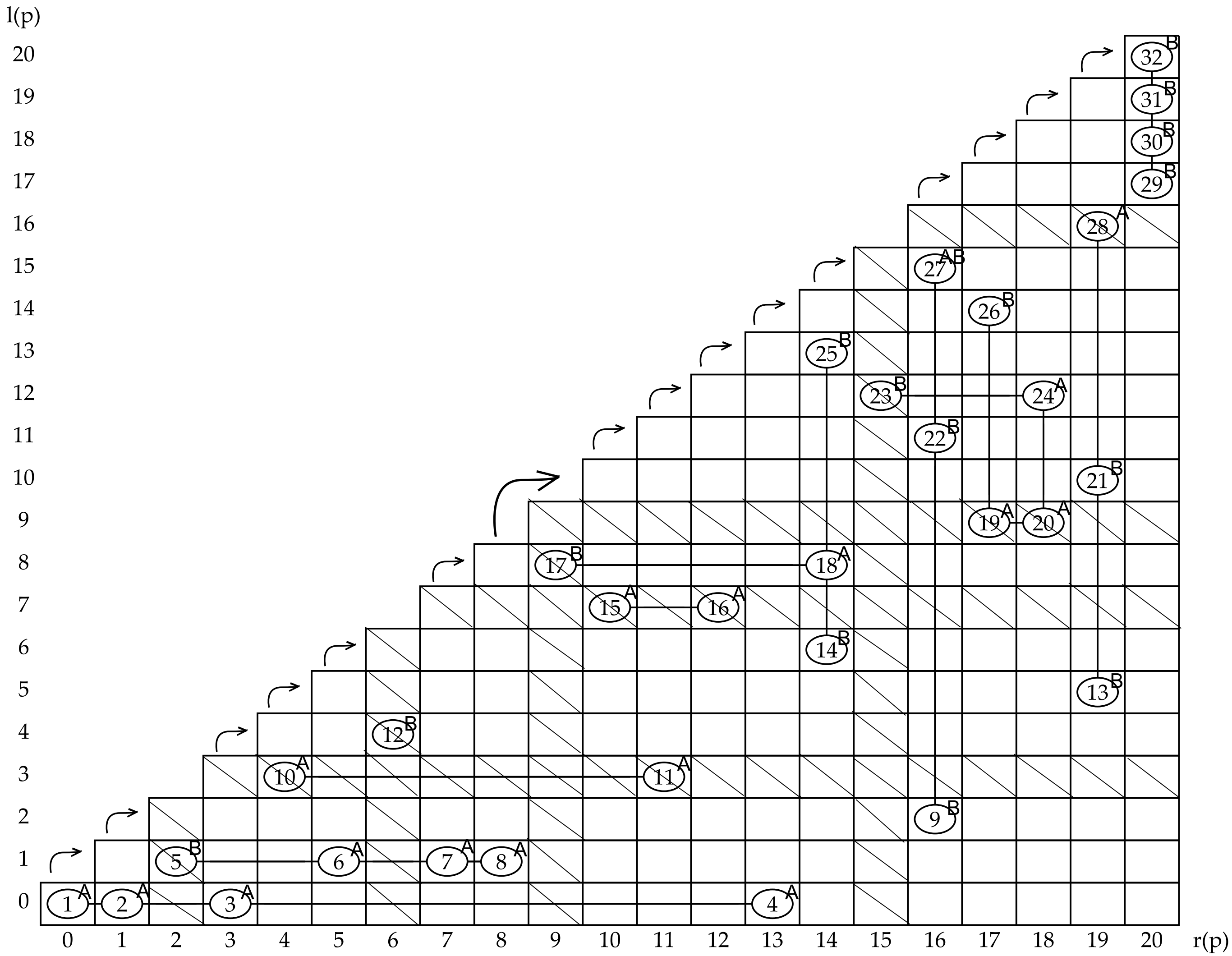
Przykład 3 (iv)



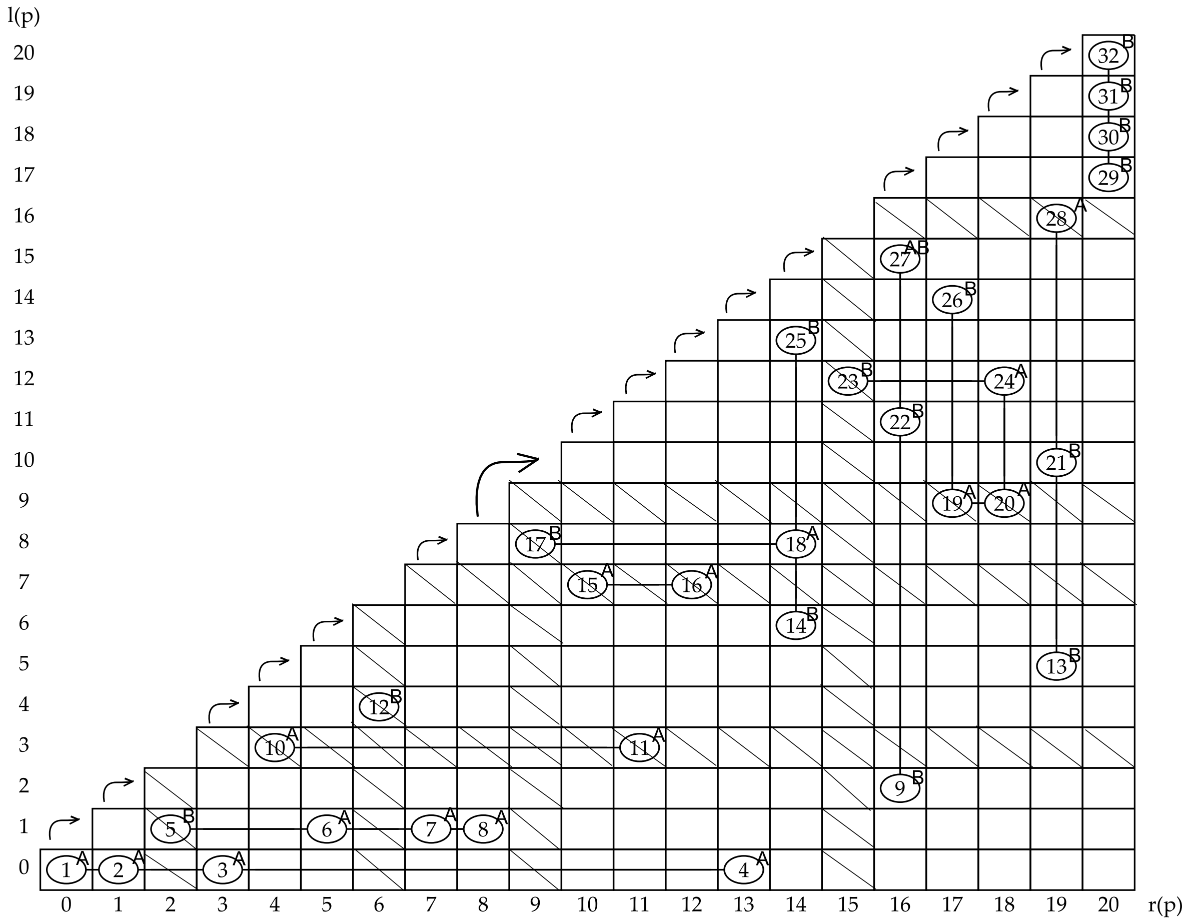
Przykład 3 (v)


l(p)
r(p)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

Przykład 3 (vi)



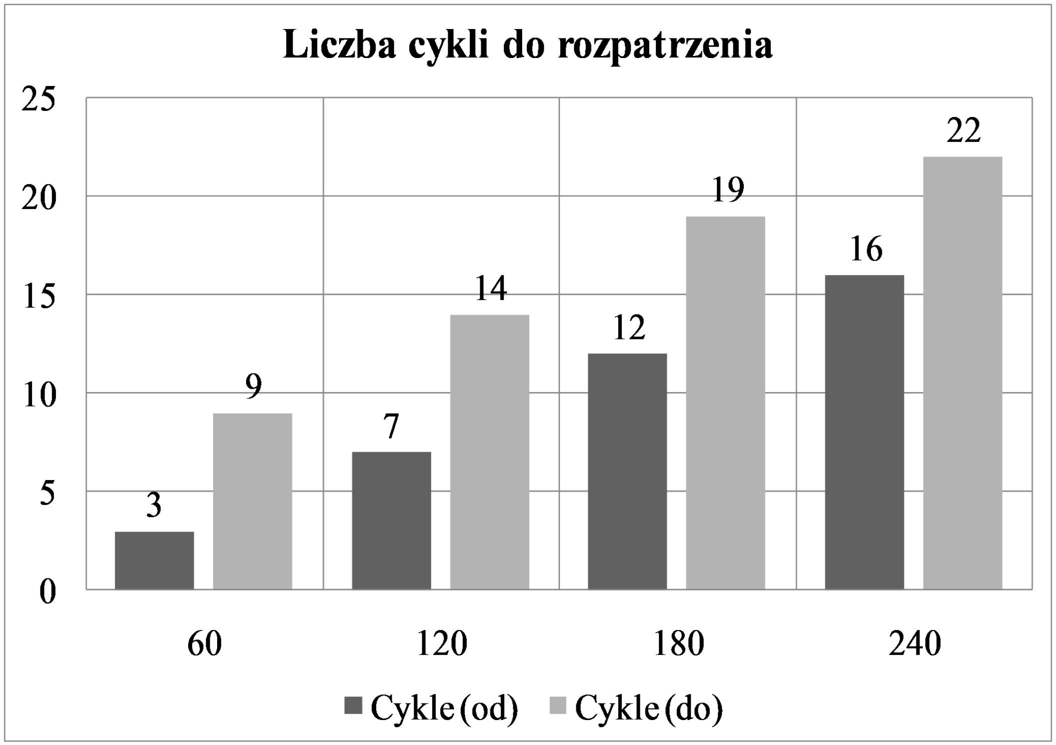
Przykład 3 (vii)



$$L = (1,5) + (2,9) + (3,12) + (10,13) + (6,14) + (7,17) + (8,21) + (15,22) + (11,23) +$$

$$(16,25) + (4,26) + (18,27,29) + (19,30) + (24) + (20,31) + (28,32)$$

Czas działania



- Czas: od 44ms ($|P| = 60$) do 620ms ($|P| = 240$).