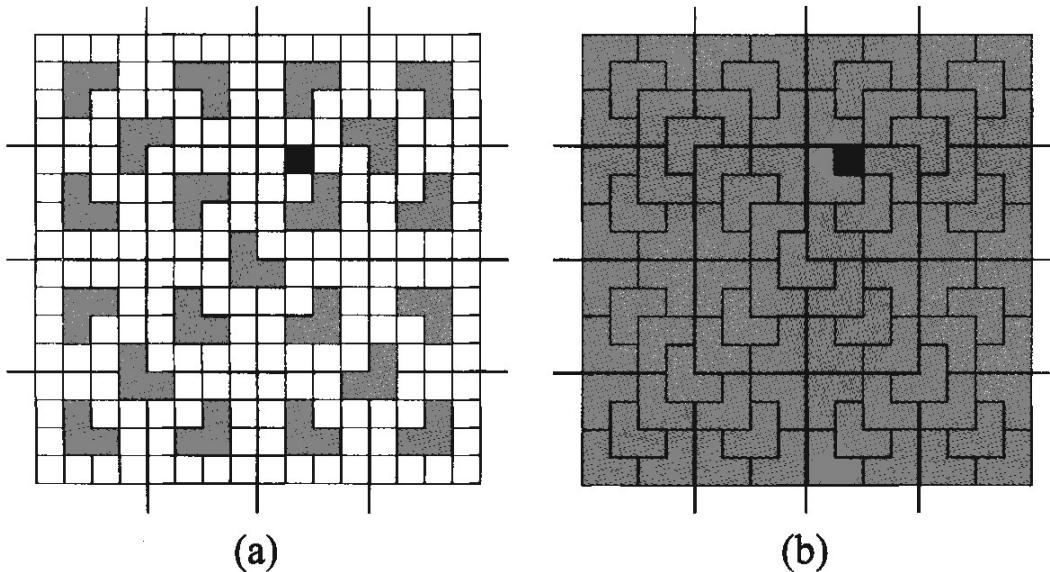


Zwróćmy uwagę, że każdy kwadrat (mają one teraz rozmiar  $4 \times 4$ ) ma dokładnie jedno brakujące pole, tj. jedną dziurę. Kontynuując tę procedurę, możemy każdy z kwadratów podzielić na cztery kwadraty  $2 \times 2$ , a po odpowiednim umieszczeniu kostek każdy z tych nowych małych kwadratów będzie miał dokładnie jedną dziurę (rys. 4.5a). W tym momencie mamy przed sobą najłatwiejsze na świecie zadanie: umieścić kostkę w kwadracie  $2 \times 2$  mającym jedno brakujące pole! W ten sposób łatwo otrzymujemy ostateczne rozwiązanie problemu (rys. 4.5b).



Rysunek 4.5. Rozmieszczenie kolejnych 16 kostek (a) oraz ostateczne rozwiązanie (b)

### 4.3. Programowanie dynamiczne

Programowanie dynamiczne działa w ten sposób, że znajduje pełne rozwiązanie na podstawie pośredniego punktu, który znajduje się między naszym obecnym miejscem pobytu a celem, do którego dążymy<sup>1</sup>. Procedura ta ma charakter rekurencyjny w tym sensie, że każdy następny punkt pośredni jest obliczany jako funkcja już obliczonych punktów. Problem nadający się do rozwiązywania metodą programowania dynamicznego ma następujące własności:

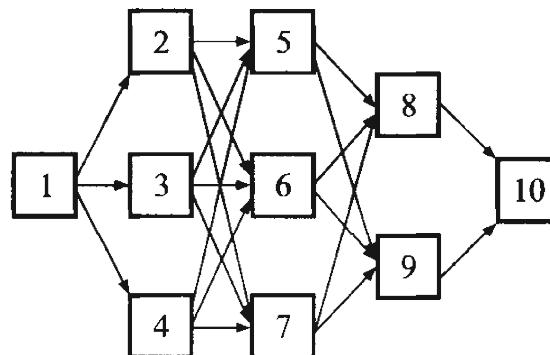
- można go rozłożyć na ciąg decyzji podejmowanych na różnych etapach;
- każdy etap ma pewną liczbę możliwych stanów;
- podjęcie decyzji oznacza przejście z jednego stanu na pewnym etapie do innego stanu na następnym etapie;

<sup>1</sup>Określenie *dynamiczne* zostało użyte, żeby wskazać, że podejście to jest przydatne w problemach, „w których ważną rolę odgrywa czas i w których kolejność wykonywania operacji może być istotna” [40].

- najlepszy ciąg decyzji (zwany też *strategią*) na dowolnym etapie nie zależy od decyzji podjętych na wcześniejszych etapach;
- istnieje dobrze określony koszt przejścia z jednego stanu do drugiego; co więcej, istnieje rekurencyjna zależność umożliwiająca określenie najlepszej możliwej decyzji.

Metodę tę możemy zastosować w ten sposób, że zaczynamy od celu i cofamy się krok po kroku do stanu bieżącego. Innymi słowy, możemy najpierw określić, jak wygląda najlepsza decyzja na ostatnim etapie. Na tej postawie określamy najlepszą decyzję na przedostatnim etapie, mając na uwadze, jak wygląda najlepsza decyzja na ostatnim itd.

Najprostszym przykładem jest *problem dyliżansu* opracowany przez Harveya Wagnera [217]. W pierwotnym opisie tego problemu powiedziano bardzo dużo na temat zmieniania zasłonek w dyliżansie przez komiwojażera przy przeprowadzaniu dyliżansu przez terytoria wrogo nastawionych Indian amerykańskich; darujemy sobie te opisy i przejdziemy od razu do sedna.



**Rysunek 4.6.** Diagram przepływu wskazujący wybory komiwojażera przeprowadzającego dyliżans przez wrogie terytorium

Komiwojażer musi zacząć od swojej bieżącej pozycji i dostać się do ustalonego celu. W zasadzie ma on trzy etapy, na których może dokonać wyboru drogi (rys. 4.6). Na pierwszym z nich ma do dyspozycji trzy ścieżki. Podobnie na drugim etapie ma znowu trzy ścieżki. Wreszcie na trzecim ma dwie. Jest jeszcze czwarty etap, ale tam nie ma żadnej możliwości wyboru. Koszty przejścia z każdego ze stanów na każdym etapie do wszystkich możliwych następnych stanów pokazano na rys. 4.7. Zadanie polega na znalezieniu ścieżki o najmniejszym koszcie, biegającej z pierwszego stanu (1) do stanu ostatniego (10).

Przed przystąpieniem do omawiania sposobu dojścia do odpowiedzi musimy jeszcze pewne sprawy wyjaśnić.

- Rozważamy  $n$  etapów; decyzja, do którego stanu należy dalej przejść, jest oznaczona  $x_n$ . W naszym przykładzie  $n = 4$ .
- Wyrażenie  $f_n(s, x_n)$  oznacza koszt najlepszego ciągu decyzji (strategii) na wszystkich pozostałych etapach, przy założeniu że komiwojażer jest w stanie  $s$ , na etapie  $n$  i wybrał  $x_n$  jako następny stan.

	2	3	4		5	6	7
1	2	4	3	2	7	4	6
5	1	4	6	3	3	2	4
6	6	3	3	4	4	1	5
7	3	3		8	10		
				9	3		
					4		

**Rysunek 4.7.** Koszt wykonania każdego z możliwych wyborów na każdym etapie decyzyjnym. Wiersz oznacza stan bieżący, kolumna zaś stan następny. Cztery tabelki odpowiadają czterem etapom. Zwróćmy uwagę, że tylko trzy etapy dają możliwość wyboru stanu

- Wyrażenie  $x_n^*(s)$  oznacza wartość  $x_n$ , która minimalizuje  $f_n(s, x_n)$ , a  $f_n^*(s)$  oznacza odpowiedni minimalny koszt.
- Celem jest znalezienie  $f_1^*(1)$ , ponieważ komiwojażer zaczyna w stanie 1. Uzyskujemy to, znajdując  $f_4^*(s)$ ,  $f_3^*(s)$ ,  $f_2^*(s)$  i wreszcie  $f_1^*(1)$ .

Czym zatem jest  $f_4^*(s)$ ? Dla jakiego  $x_4$  wartość  $f_4(s, x_4)$  jest minimalna? Jest to, prawdę mówiąc, podchwytliwe pytanie, gdyż na czwartym etapie jest tylko jeden możliwy stan do wybrania:  $x_4 = 10$ . Łatwo jest zatem obliczyć poniżej przedstawione wartości, czym kończymy pierwszą fazę procedury programowania dynamicznego.

$s$	$f_4^*(s)$	$x_4^*(s)$
8	3	10
9	4	10

Powyższa tabelka ukazuje koszt przejścia od stanu 8 lub 9 do stanu w końcowym etapie (w tym wypadku jest tylko jeden taki stan).

W następnej fazie pytamy się o wartość  $f_3^*(s)$ . Przypomnij sobie, że decyzje są podejmowane niezależnie, a zatem  $f_3(s, x_3) = c_{sx_3} + f_4^*(x_3)$ , przy czym  $c_{sx_3}$  oznacza koszt podróży z  $s$  do  $x_3$ . W poniższej tabelce przedstawiono odpowiednie wartości  $f_3(s, x_3)$  dla każdego możliwego wyboru celu, przy założeniu że  $s$  jest równe 5, 6 lub 7.

$s$	$f_3(s, 8)$	$f_3(s, 9)$	$f_3^*(s)$	$x_3^*(s)$
5	4	8	4	8
6	9	7	7	9
7	6	7	6	8

Widzimy, że jeśli jesteśmy w stanie 5, to stan 8 minimalizuje pozostały koszt. Stan 9 minimalizuje koszt, gdy  $s = 6$ , a stan 8 minimalizuje koszt, gdy  $s = 7$ .

Robiąc krok do tyłu, musimy znaleźć wartość  $f_2^*(s)$ . Poniższa tabelka w analogiczny sposób jak poprzednia pokazuje odpowiednie wartości  $f_2(s, x_2)$ . Zauważ, że  $f_2(s, x_2) = c_{sx_2} + f_3^*(x_2)$ .

$s$	$f_2(s, 5)$	$f_2(s, 6)$	$f_2(s, 7)$	$f_2^*(s)$	$x_2^*(s)$
2	11	11	12	11	5 lub 6
3	7	9	10	7	5
4	8	8	11	8	5 lub 6

Wreszcie, jeśli chodzi o  $f_1^*(s)$ , ostatnia tabelka daje koszty dotarcia do stanów 2, 3 i 4:

$s$	$f_1(s, 2)$	$f_1(s, 3)$	$f_1(s, 4)$	$f_1^*(s)$	$x_1^*(s)$
1	13	11	11	11	3 lub 4

To już jest ostateczna tabelka w naszym problemie dyliżansu. Tak oto użyliśmy metody programowania dynamicznego do cofnięcia się wzdłuż całej drogi do pierwszego etapu. Wartości w powyższej tabelce określają koszt podróży ze stanu 1 do stanów 2, 3 i 4.

W tym momencie możemy określić najlepsze rozwiązanie całego problemu. W rzeczywistości istnieją tu trzy najlepsze odpowiedzi (strategie), każda z nich ma ten sam koszt równy 11 jednostek:

- $1 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 10$
- $1 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 10$
- $1 \rightarrow 4 \rightarrow 6 \rightarrow 9 \rightarrow 10$

Zauważmy, że odwrotna procedura, polegająca na wybieraniu ścieżki o najmniejszym koszcie przy poruszaniu się do przodu przez kolejne etapy, nie daje optymalnego rozwiązania. Takie rozwiązanie otrzymane metodą zachłanną ma koszt 13 jednostek (możesz to sprawdzić). Czyli mamy tutaj do czynienia z sytuacją, gdy podejście zachłanne zawodzi, a programowanie dynamiczne daje poprawną odpowiedź. Wadą tego podejścia jest jednak objawiająca się czasami jego złożoność obliczeniowa. Jeśli  $N$  oznacza zarówno liczbę etapów, jak i stanów w każdym z etapów, to wymagana liczba operacji sięga  $N^3$ . Metodę tę możemy jednak tak zaadaptować, żeby dawała wyniki dla wielu problemów optymalizacyjnych, w tym dla problemu znajdowania kolejności mnożenia macierzy oraz znajdowania takiej organizacji drzewa, która minimalizuje koszt przeszukiwania [425].

Algorytmy programowania dynamicznego są niekiedy trudne do zrozumienia. Wynika to z tego, że konstrukcja programu dynamicznego zależy od samego problemu. Opracowywanie jej jest „działaniem intelektualnym o charakterze artystycznym częściowo zależącym od konkretnej struktury rozwiązywanego sekwencyjnego problemu decyzyjnego” [432]. Dlatego zilustrujemy tę metodę, przedstawiając jeszcze dwa przykłady: mnożenie macierzy i TSP.

Założymy, że wymiary macierzy  $A_1, A_2, A_3$  i  $A_4$  wynoszą odpowiednio  $20 \times 2, 2 \times 15, 15 \times 40$  i  $40 \times 4$  i że chcemy znaleźć optymalny sposób obliczenia  $A_1 \times A_2 \times A_3 \times A_4$ , tzn. chcielibyśmy obliczyć ten iloczyn, stosując minimalną

liczbę mnożeń. Zakładamy tutaj, że aby pomnożyć dwie macierze,  $P$  o wymiarze  $n \times k$  i  $Q$  o wymiarze  $k \times m$ , należy wykonać  $nkm$  mnożeń. Macierz wynikowa  $R$  ma wymiar  $n \times m$  i

$$r_{ij} = \sum_{v=1}^k p_{iv}q_{vj}$$

dla wszystkich  $1 \leq i \leq n$  oraz  $1 \leq j \leq m$ .

Zauważmy, że mnożenie macierzy w różnej kolejności daje różne koszty. Na przykład:

- $A(B(CD))$  wymaga  $15 \cdot 40 \cdot 4 + 2 \cdot 15 \cdot 4 + 20 \cdot 2 \cdot 4 = 2680$  mnożeń
- $(AB)(CD)$  wymaga  $20 \cdot 2 \cdot 15 + 15 \cdot 40 \cdot 4 + 20 \cdot 15 \cdot 4 = 4200$  mnożeń
- $((AB)C)D$  wymaga  $20 \cdot 2 \cdot 15 + 20 \cdot 15 \cdot 40 + 20 \cdot 40 \cdot 4 = 15\,800$  mnożeń!

Stosując programowanie dynamiczne, tworzymy tutaj strukturę  $M(i, j)$ , w której zapisujemy minimalną liczbę mnożeń wymaganych do pomnożenia macierzy od  $A_i$  do  $A_j$  ( $i \leq j$ ). Oczywiście  $M(1, 1) = M(2, 2) = M(3, 3) = M(4, 4) = 0$ , gdyż tutaj nie wykonujemy żadnych mnożeń. Zauważmy jednak, że problem polega na znalezieniu  $M(1, 4)$ .

Związek między rozwiązaniami dla mniejszych problemów a rozwiązaniem dla większego problemu jest następujący:

$$M(i, j) = \min_{j \leq k < j} \{M(i, k) + M(k + 1, j) + cost_{ij}^k\}$$

przy czym  $cost_{ij}^k$  oznacza liczbę mnożeń wymaganych do pomnożenia iloczynu  $A_i \dots A_k$  przez  $A_{k+1} \dots A_j$ . Chodzi o to, że aby optymalnie pomnożyć ciąg od  $A_i$  do  $A_j$ , musimy znaleźć najlepszy punkt podziału  $k$ , dla którego całkowita liczba mnożeń wymaganych do obliczenia iloczynu  $A_i \dots A_k$ , wynosząca  $M(i, k)$ , iloczynu  $A_{k+1} \dots A_j$ , wynosząca  $M(k + 1, j)$  oraz obu tych iloczynów razem (równa  $cost_{ij}^k$ ) była minimalna.

Pamiętając o tym, możemy wykonać następujące kroki. Uzyskanie:

$$M(1, 2) = 600$$

$$M(2, 3) = 1200$$

$$M(3, 4) = 2400$$

jest łatwym ćwiczeniem, gdyż przy mnożeniu dwóch macierzy nie ma punktu podziału. Dalej

$$M(1, 3) = 2800$$

$$M(2, 4) = 1520$$

Zauważmy, że  $M(1, 3)$  jest tutaj tą mniejszą z dwóch wartości:

$$M(1, 1) + M(2, 3) + cost_{13}^1 = 0 + 1200 + 1600 = 2800$$

$$M(1, 2) + M(3, 3) + cost_{13}^2 = 600 + 0 + 12000 = 12600$$

Podobnie  $M(2, 4)$  jest mniejszą z dwóch wartości:

$$M(2, 2) + M(3, 4) + \text{cost}_{24}^2 = 0 + 2400 + 120 = 2520$$

$$M(2, 3) + M(4, 4) + \text{cost}_{24}^3 = 1200 + 0 + 320 = 1520$$

Wreszcie znajdujemy

$$M(1, 4) = 1680$$

jako najmniejszą z trzech wartości:

$$M(1, 1) + M(2, 4) + \text{cost}_{14}^1 = 0 + 1520 + 160 = 1680$$

$$M(1, 2) + M(3, 4) + \text{cost}_{14}^2 = 600 + 2400 + 1200 = 4200$$

$$M(1, 3) + M(4, 4) + \text{cost}_{14}^3 = 2800 + 0 + 3200 = 6000$$

Tak więc minimalny koszt wyrażony liczbą mnożeń wynosi 1680, ale jeszcze musimy znaleźć odpowiadającą mu kolejność mnożeń macierzy. Znalezienie takiej najlepszej kolejności wymaga użycia dodatkowej struktury danych  $O(i, j)$ , przeznaczonej do przechowywania indeksu najlepszego punktu podziału. Innymi słowy,  $O(i, j) = k$  wtedy i tylko wtedy, gdy  $M(i, j)$  osiąga minimalną wartość dla  $M(i, k) + M(k + 1, j) + \text{cost}_{ij}^k$ . Indeksy umieszczone w tablicy  $O$  ukazują poszukiwaną przez nas kolejność:  $A_1((A_2 A_3) A_4)$ .

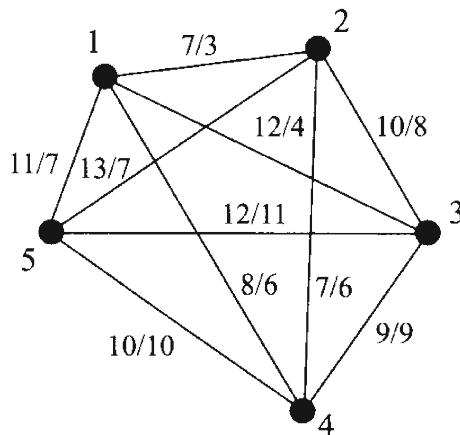
Zakończymy ten podrozdział przykładem zastosowania programowania dynamicznego do TSP. Ostrzegamy jednak – przykład jest długi. Aby jego opis był zrozumiały, będziemy się tutaj zajmować zadaniem TSP dla pewnych pięciu miast. Oto macierz  $L$  zawierająca odległości między miastami:

$$L = \begin{bmatrix} 0 & 7 & 12 & 8 & 11 \\ 3 & 0 & 10 & 7 & 13 \\ 4 & 8 & 0 & 9 & 12 \\ 6 & 6 & 9 & 0 & 10 \\ 7 & 7 & 11 & 10 & 0 \end{bmatrix}$$

Koszt podróży z miasta  $i$  do miasta  $j$  jest podany w  $i$ -tym wierszu i  $j$ -tej kolumnie tej macierzy. Na przykład odległość między miastami 1 i 3 wynosi  $L(1, 3) = 12$ . Odległość ta jest różna od odległości między miastami 3 i 1, która wynosi  $L(3, 1) = 4$ . Mamy zatem tutaj do czynienia z asymetrycznym TSP. Zauważmy, że  $L(i, i) = 0$  dla wszystkich  $1 \leq i \leq n$ . Całą tę sytuację przedstawiono na rys. 4.8.

Trasa w TSP jest cyklem, który odwiedza *każde* miasto raz i tylko raz, a zatem jest nieistotne, od którego miasta zaczynamy. Przypuśćmy, że zaczynamy od miasta 1. W tym momencie jesteśmy gotowi do podzielenia problemu na mniejsze problemy. Niech  $g(i, S)$  oznacza długość najkrótszej ścieżki od miasta  $i$  do miasta 1, przechodzącej dokładnie raz przez *każde* miasto ze zbioru  $S$ . W szczególności oznacza to, że

$$g(4, \{5, 2, 3\})$$



**Rysunek 4.8.** TSP z pięcioma miastami. Dla każdej pary miast są podane dwie liczby oddzielone ukośnikiem. Pierwsza z nich określa odległość od miasta o mniejszym numerze do miasta o większym numerze, druga zaś oznacza odległość w przeciwnym kierunku

przedstawia najkrótszą ścieżkę, która prowadzi z miasta 4 przez miasta 5, 2 i 3 (w jakiejś bliżej nieokreślonej kolejności) do miasta 1. Obliczenie długości najkrótszej całej trasy TSP sprowadza się do znalezienia

$$g(1, V - \{1\})$$

przy czym  $V$  oznacza zbiór *wszystkich* miast w TSP. Innymi słowy, próbujemy znaleźć najkrótszą ścieżkę, która zaczyna się w mieście 1, przechodzi przez każde miasto ze zbioru  $V - \{1\}$  dokładnie raz i wraca do miasta 1.

Sformułowanie problemu jako programowania dynamicznego oznacza pojęcie związku między rozwiązaniami mniejszych problemów a rozwiązaniem większego problemu. Twierdzimy, że mamy tutaj

$$g(i, S) = \min_{j \in S} \{L(i, j) + g(j, S - \{j\})\}$$

Innymi słowy, wybranie najkrótszej ścieżki zaczynającej się od miasta  $i$  oraz prowadzącej przez każde miasto w  $S$  przed powrotem do miasta 1 wymaga znalezienia w zbiorze  $S$  miasta  $j$ , dla którego suma odległości  $L(i, j)$  oraz długości pozostałej drogi  $g(j, S - \{j\})$  jest minimalna. Jeśli znamy rozwiązania mniejszych problemów (tj. problemów, w których rozmiar  $S$  wynosi  $k$ ), to możemy uzyskać rozwiązanie większego problemu (tj. takiego, gdzie rozmiar  $S$  wynosi  $k + 1$ ).

Wróćmy do naszego przykładu. Zadanie polega na znalezieniu

$$g(1, \{2, 3, 4, 5\})$$

Zwróćmy także uwagę, że:

$$g(2, \emptyset) = L(2, 1) = 3$$

$$g(3, \emptyset) = L(3, 1) = 4$$

$$g(4, \emptyset) = L(4, 1) = 6$$

$$g(5, \emptyset) = L(5, 1) = 7$$

gdyż zbiór  $S$  jest tutaj pusty i przechodzimy bezpośrednio od miasta  $i$  ( $i = 2, 3, 4, 5$ ) do miasta 1. Następny krok jest równie prosty – musimy znaleźć rozwiązania dla wszystkich problemów, w których rozmiar  $S$  wynosi jeden. Wy-maga to rozwiązania 12 podproblemów, gdyż będziemy zaczynać od miast 2, 3, 4 oraz 5 i dla każdego z nich będziemy rozważali trzy możliwe zbiory jednoelementowe. Na przykład dla miasta 2 mamy:

$$g(2, \{3\}) = L(2, 3) + g(3, \emptyset) = 10 + 4 = 14$$

$$g(2, \{4\}) = L(2, 4) + g(4, \emptyset) = 7 + 6 = 13$$

$$g(2, \{5\}) = L(2, 5) + g(5, \emptyset) = 13 + 7 = 20$$

Zwróćmy uwagę, że wynik  $g(2, \{3\}) = 14$  oznacza, iż długość najkrótszej ścieżki z miasta 2 do 1, która przed dojściem do 1 przechodzi przez miasto 3, wynosi 14. Wynik ten jest oczywisty, ponieważ  $S$  składa się tylko z jednego elementu i nie są tu możliwe żadne wybory. Podobnie, dla miasta 3 mamy:

$$g(3, \{2\}) = L(3, 2) + g(2, \emptyset) = 8 + 3 = 11$$

$$g(3, \{4\}) = L(3, 4) + g(4, \emptyset) = 9 + 6 = 15$$

$$g(3, \{5\}) = L(3, 5) + g(5, \emptyset) = 12 + 7 = 19$$

dla miasta 4 mamy:

$$g(4, \{2\}) = L(4, 2) + g(2, \emptyset) = 6 + 3 = 9$$

$$g(4, \{3\}) = L(4, 3) + g(3, \emptyset) = 9 + 4 = 13$$

$$g(4, \{5\}) = L(4, 5) + g(5, \emptyset) = 10 + 7 = 17$$

oraz dla miasta 5 mamy:

$$g(5, \{2\}) = L(5, 2) + g(2, \emptyset) = 7 + 3 = 10$$

$$g(5, \{3\}) = L(5, 3) + g(3, \emptyset) = 11 + 4 = 15$$

$$g(5, \{4\}) = L(5, 4) + g(4, \emptyset) = 10 + 6 = 16$$

Jesteśmy teraz gotowi do wykonania następnego kroku naszego dynamicznego programu, gdy  $S$  wynosi 2. Musimy tutaj rozwiązać również 12 podproblemów, gdyż każde z czterech miast może być rozważane wraz z dwoma innymi miastami (dwoma spośród trzech). I tak dla miasta 2 mamy:

$$\begin{aligned} g(2, \{3, 4\}) &= \min\{L(2, 3) + g(3, \{4\}), L(2, 4) + g(4, \{3\})\} = \\ &= \min\{10 + 15, 7 + 13\} = \min\{25, 20\} = 20 \end{aligned}$$

$$\begin{aligned} g(2, \{3, 5\}) &= \min\{L(2, 3) + g(3, \{5\}), L(2, 5) + g(5, \{3\})\} = \\ &= \min\{10 + 19, 13 + 15\} = \min\{29, 28\} = 28 \end{aligned}$$

$$\begin{aligned} g(2, \{4, 5\}) &= \min\{L(2, 4) + g(4, \{5\}), L(2, 5) + g(5, \{4\})\} = \\ &= \min\{7 + 17, 13 + 16\} = \min\{24, 29\} = 24 \end{aligned}$$

Wynik  $g(2, \{3, 4\}) = 20$  oznacza, że długość najkrótszej ścieżki z miasta 2 do 1, która przed dojściem do 1 przechodzi do miasta 3 i 4 (w dowolnej kolejności), wynosi 20. Tym razem musimy rozważyć dwie możliwości (możemy pojechać

najpierw do miasta 3 lub najpierw do miasta 4) i wybrać lepszą z nich. Podobnie, dla miasta 3 mamy:

$$\begin{aligned} g(3, \{2, 5\}) &= \min\{L(3, 2) + g(2, \{5\}), L(3, 5) + g(5, \{2\})\} = \\ &= \min\{8 + 20, 12 + 10\} = \min\{28, 22\} = 22 \\ g(3, \{2, 4\}) &= \min\{L(3, 2) + g(2, \{4\}), L(3, 4) + g(4, \{2\})\} = \\ &= \min\{8 + 13, 9 + 9\} = \min\{21, 18\} = 18 \\ g(3, \{4, 5\}) &= \min\{L(3, 4) + g(4, \{5\}), L(3, 5) + g(5, \{4\})\} = \\ &= \min\{9 + 17, 12 + 16\} = \min\{26, 28\} = 26 \end{aligned}$$

dla miasta 4 mamy:

$$\begin{aligned} g(4, \{2, 3\}) &= \min\{L(4, 2) + g(2, \{3\}), L(4, 3) + g(3, \{4\})\} = \\ &= \min\{6 + 14, 9 + 15\} = \min\{20, 24\} = 20 \\ g(4, \{2, 5\}) &= \min\{L(4, 2) + g(2, \{5\}), L(4, 5) + g(5, \{2\})\} = \\ &= \min\{6 + 20, 10 + 10\} = \min\{26, 20\} = 20 \\ g(4, \{3, 5\}) &= \min\{L(4, 3) + g(3, \{5\}), L(4, 5) + g(5, \{3\})\} = \\ &= \min\{9 + 19, 10 + 15\} = \min\{28, 25\} = 25 \end{aligned}$$

i dla miasta 5 mamy:

$$\begin{aligned} g(5, \{2, 3\}) &= \min\{L(5, 2) + g(2, \{3\}), L(5, 3) + g(3, \{2\})\} = \\ &= \min\{7 + 14, 11 + 11\} = \min\{21, 22\} = 21 \\ g(5, \{2, 4\}) &= \min\{L(5, 2) + g(2, \{4\}), L(5, 4) + g(4, \{2\})\} = \\ &= \min\{7 + 13, 10 + 19\} = \min\{20, 29\} = 20 \\ g(5, \{3, 4\}) &= \min\{L(5, 3) + g(3, \{4\}), L(5, 4) + g(4, \{3\})\} = \\ &= \min\{11 + 15, 10 + 13\} = \min\{26, 23\} = 23 \end{aligned}$$

Jesteśmy już gotowi do wykonania następnego kroku, w którym rozmiar zbioru  $S$  wynosi trzy. Musimy się tutaj zmierzyć tylko z czterema podproblemami – po jednym dla każdego miasta. W ten sposób dla miasta 2 musimy wybrać najkrótszą spośród trzech ścieżek:

$$\begin{aligned} g(2, \{3, 4, 5\}) &= \\ &= \min\{L(2, 3) + g(3, \{4, 5\}), L(2, 4) + g(4, \{3, 5\}), L(2, 5) + g(5, \{3, 4\})\} = \\ &= \min\{10 + 26, 7 + 25, 13 + 23\} = \min\{36, 32, 34\} = 32 \end{aligned}$$

Wynik  $g(2, \{3, 4, 5\}) = 32$  oznacza, że długość najkrótszej ścieżki z miasta 2 do 1, która przed dojściem do 1 przechodzi przez miasta 3, 4 i 5 (w dowolnej kolejności), wynosi 32. Musimy teraz rozważyć trzy możliwości: możemy pojechać najpierw do miasta 3, do miasta 4 lub do miasta 5, i wybrać najlepszą z nich. Podobnie, dla miast 3, 4 i 5 mamy:

$$\begin{aligned} g(3, \{2, 4, 5\}) &= \\ &= \min\{L(3, 2) + g(2, \{4, 5\}), L(3, 4) + g(4, \{2, 5\}), L(3, 5) + g(5, \{2, 4\})\} = \\ &= \min\{8 + 24, 9 + 20, 12 + 20\} = \min\{32, 29, 32\} = 29 \end{aligned}$$

$$\begin{aligned}
 g(4, \{2, 3, 5\}) &= \\
 &= \min\{L(4, 2) + g(2, \{3, 5\}), L(4, 3) + g(3, \{2, 5\}), L(4, 5) + g(5, \{2, 3\})\} = \\
 &= \min\{6 + 28, 9 + 22, 10 + 21\} = \min\{34, 31, 31\} = 31 \\
 g(5, \{2, 3, 4\}) &= \\
 &= \min\{L(5, 2) + g(2, \{3, 4\}), L(5, 3) + g(3, \{2, 4\}), L(5, 4) + g(4, \{2, 3\})\} = \\
 &= \min\{7 + 20, 11 + 18, 10 + 20\} = \min\{27, 29, 30\} = 27
 \end{aligned}$$

Wreszcie nadszedł czas na końcowy krok, w którym zaczynamy od miasta 1 i zamykamy cykl. Wracamy w tym momencie do pierwotnego problemu:

$$g(1, \{2, 3, 4, 5\}) = ?$$

Zaczynając od miasta 1, mamy cztery możliwości: pójść najpierw do miasta 2, miasta 3, miasta 4 lub miasta 5. Odpowiedzi dla pojawiających się tutaj podproblemów (tzn. znajdowanie najkrótszych długości dla wszystkich możliwych ciągów dalszych wymienionych początków) są już znane, możemy więc podjąć ostateczną decyzję:

$$\begin{aligned}
 g(1, \{2, 3, 4, 5\}) &= \min\{L(1, 2) + g(2, \{3, 4, 5\}), L(1, 3) + g(3, \{2, 4, 5\}), \\
 &\quad L(1, 4) + g(4, \{2, 3, 5\}), L(1, 5) + g(5, \{2, 3, 4\})\} = \\
 &= \min\{7 + 32, 12 + 29, 8 + 31, 11 + 27\} = \min\{39, 41, 39, 38\} = 38
 \end{aligned}$$

Najkrótsza trasa ma zatem długość 38.

Jak jednak ta trasa przebiega? Podobnie jak w wypadku problemu mnożenia macierzy, znaleźliśmy optymalną wartość funkcji oceny, ale musimy jeszcze ustalić, która trasa ją dała. I znowu zadanie to jest łatwe, jeśli śledzimy nasze obliczenia. Potrzebujemy w tym celu dodatkowej struktury danych  $W$ , która przechowuje informacje, jakie w następnej kolejności wybrać miasto, żeby ścieżka miała minimalną długość. Zatem na przykład

$$W(5, \{2, 3, 4\}) = 2$$

gdyż najkrótsza ścieżka z miasta 5 do miasta 1, która musi przejść przez miasta 2, 3 i 4 przed dojściem do 1, musi najpierw przechodzić przez miasto 2. Podobnie

$$W(1, \{2, 3, 4, 5\}) = 5$$

Stąd globalne rozwiązanie dla analizowanego TSP daje trasę

$$1 - 5 - 2 - 4 - 3 - 1$$

której długość wynosi

$$11 + 7 + 7 + 9 + 4 = 38$$

Czy myśl o użyciu programowania dynamicznego do rozwiązywania problemu TSP z 50 miastami budzi grozę?