Schemat programowania dynamicznego *(ang. dynamic programming)*

Jest jedną z metod rozwiązywania problemów optymalizacyjnych.

Jej twórcą (1957) był amerykański matematyk *Richard Ernest Bellman*.

Schemat ten określa ogólne podejście polegające na przekształceniu zadania optymalizacyjnego w wieloetapowy proces decyzyjny, w którym stan na każdym etapie zależy od decyzji wybieranej ze zbioru decyzji dopuszczalnych.

Metoda ma zastosowanie dla problemów ciągłych i dyskretnych.

Zadanie: zminimalizować koszt przejścia pomiędzy stanami

Dane: graf (I) reprezentujący problem

$S \to A : 2 \to C : 3 \to B : 4$

$A \to I : 4 \to F : 6 \to D : 7$

$B \to I : 2 \to D : 5 \to F : 7$

$C \to I : 1 \to D : 4 \to F : 7$

$D \to E : 1 \to H : 9$

$E \to G : 3$

$F \to E : 1 \to H : 2$

$G \to$

$H \to G : 4$

$I \to E : 3 \to H : 6$

Zadanie: zminimalizować koszt przejścia pomiędzy stanami

Dane: graf (I)

analiza sytuacji...

Zadanie: zminimalizować koszt przejścia pomiędzy stanami

Dane: graf (I)

Metoda: zachłanna (A)

wynik: ...

koszt: ...

Zadanie: zminimalizować koszt przejścia pomiędzy stanami

Dane: graf (I)

Metoda: zachłanna (A)

wynik: S-A-I-E-G

koszt: 13

Czy jest rozwiązanie optymalne?

Jak je znaleźć?

Jaki będzie koszt znalezienia rozwiązania optymalnego?

Zadanie: zminimalizować koszt przejścia pomiędzy stanami

Dane: graf (I)

Metoda: zachłanna (A)

wynik: S-A-I-E-G

koszt: 12

Czy jest rozwiązanie optymalne? NIE

Jak je znaleźć? np. metodą *brute force*

Jaki będzie koszt znalezienia rozwiązania optymalnego? ...

Zadanie: zminimalizować koszt przejścia pomiędzy stanami

Dane: graf (I)

Metoda: zachłanna (A)

wynik: S-A-I-E-G

koszt: 12

Czy jest rozwiązanie optymalne? NIE

Jak je znaleźć? np. metodą *brute force*

Jaki będzie koszt znalezienia rozwiązania optymalnego? 3 x 3 x 2 sum po 4 dane, czyli 72 działania.

Zadanie: zminimalizować koszt przejścia pomiędzy stanami

Dane: graf (I)

Metoda: *Dijkstry* (B)

wynik: ...

koszt: ...

Zadanie: zminimalizować koszt przejścia pomiędzy stanami

Dane: graf (I)

Metoda: *Dijkstry* (B)

wynik: S-C-I-E-G

koszt: 10

Zadanie: zminimalizować koszt przejścia pomiędzy stanami

Dane: graf (I)

Metoda: programowanie dynamiczne (C)

wynik: ...

koszt: ...

PD - metoda rozwiązywania problemów optymalizacyjnych dekomponowalnych rekurencyjnie na podproblemy.

1. scharakteryzowanie struktury rozwiązania optymalnego
2. zdefiniowanie kosztu rozwiązania optymalnego jako funkcji optymalnych rozwiązań podproblemów (układ równań rekurencyjnych)
3. obliczenie optymalnego kosztu metodą *bottom-up* (rozwiązywanie problemów od najmniejszego do największego)
4. skonstruowanie rozwiązania optymalnego (na podstawie wykonanych obliczeń) - można pominąć jeżeli interesuje nas tylko koszt rozwiązania optymalnego a nie jego przebieg

ad. 1 problem wykazuje **optymalną strukturę**, jeśli jego rozwiązanie jest funkcją optymalnych rozwiązań podproblemów.

czy (nasz) problem posiada optymalną podstrukturę?

ad. 1 problem wykazuje **optymalną strukturę**, jeśli jego rozwiązanie jest funkcją optymalnych rozwiązań podproblemów.

czy (nasz) problem posiada optymalną podstrukturę? TAK

ad. 2 zdefiniowanie kosztu rozwiązania optymalnego jako funkcji optymalnych rozwiązań podproblemów (układ równań rekurencyjnych)

jak zdefiniowany jest (rekurencyjnie) koszt rozwiązania?

ad. 1 problem wykazuje **optymalną strukturę**, jeśli jego rozwiązanie jest funkcją optymalnych rozwiązań podproblemów.

czy (nasz) problem posiada optymalną podstrukturę? TAK

ad. 2 zdefiniowanie kosztu rozwiązania optymalnego jako funkcji optymalnych rozwiązań podproblemów (układ równań rekurencyjnych)

jak zdefiniowany jest (rekurencyjnie) koszt rozwiązania? (patrz: tablica)

ad. 1 problem wykazuje **optymalną strukturę**, jeśli jego rozwiązanie jest funkcją optymalnych rozwiązań podproblemów.

czy (nasz) problem posiada optymalną podstrukturę? TAK

ad. 2 zdefiniowanie kosztu rozwiązania optymalnego jako funkcji optymalnych rozwiązań podproblemów (układ równań rekurencyjnych)

jak zdefiniowany jest (rekurencyjnie) koszt rozwiązania? (patrz: tablica)

ad. 3 obliczenie optymalnego kosztu

od czego zaczynamy, czyli gdzie mamy najmniejszy problem i ile wynosi koszt optymalny? (patrz: tablica)

ad. 1 problem wykazuje **optymalną strukturę**, jeśli jego rozwiązanie jest funkcją optymalnych rozwiązań podproblemów.

czy (nasz) problem posiada optymalną podstrukturę? TAK

ad. 2 zdefiniowanie kosztu rozwiązania optymalnego jako funkcji optymalnych rozwiązań podproblemów (układ równań rekurencyjnych)

jak zdefiniowany jest (rekurencyjnie) koszt rozwiązania? (patrz: tablica)

ad. 3 obliczenie optymalnego kosztu

od czego zaczynamy? od końca, czyli G, a koszt wynosi 10

ad. 4 skonstruowanie rozwiązania optymalnego

jak odczytać rozwiązanie, pamiętając o spełnionym warunku optymalnej podstruktury?

ad. 1 problem wykazuje **optymalną strukturę**, jeśli jego rozwiązanie jest funkcją optymalnych rozwiązań podproblemów.

czy (nasz) problem posiada optymalną podstrukturę? TAK

ad. 2 zdefiniowanie kosztu rozwiązania optymalnego jako funkcji optymalnych rozwiązań podproblemów (układ równań rekurencyjnych)

jak zdefiniowany jest (rekurencyjnie) koszt rozwiązania? (patrz: tablica)

ad. 3 obliczenie optymalnego kosztu

od czego zaczynamy, czyli gdzie mamy najmniejszy problem? od końca, czyi G, a koszt wynosi 10

ad. 4 skonstruowanie rozwiązania optymalnego

jak odczytać rozwiązanie? S-C-I-E-G

Przyjmijmy następujące oznaczenia:

- d_i - decyzja podejmowana na i-tym etapie procesu,
- S_i - zbiór stanów procesu na kolejnych etapach,
  - s_i - stan przed podjęciem decyzji d_i,
  - s_{i+1} - stan po podjęciu decyzji d_i,
- D_i - zbiór decyzji dopuszczalnych na i-tym etapie procesu,

Przebieg wieloetapowego procesu podejmowania decyzji można zapisać w postaci transformacji

$$s_k = T(s_{k-1}, d_{k-1})$$

stąd

$$s_{i+1} = T(s_i, d_i)$$

Z procesem podejmowania decyzji związana jest skalarna funkcja celu służąca do oceny ciągu decyzji $d_1, d_2, \ldots, d_N$

$$F = (s_1, s_2, \ldots, s_N; d_1, d_2, \ldots, d_N)$$

Rozwiązaniem rozważanego problemu jest znalezienie strategii optymalnej, czyli takiego ciągu decyzji $d_1, d_2, \ldots, d_N$, dla którego funkcja F osiąga ekstremum (minimum bądź maksimum).

Jeżeli S_1 jest znany, to przebieg wieloetapowego procesu decyzyjnego jest wyznaczony przez ciąg decyzji dopuszczalnych $d_1, d_2, \ldots, d_N$ nazywanych **strategią**.

Wyznaczanie strategii optymalnej w ogólnym przypadku wymaga rozwiązania zadania optymalizacji nieliniowej

$$\min_{d_1,d_2,\ldots,d_N} F(s_1, s_2, \ldots, s_N; d_1, d_2, \ldots, d_N)$$

$$s_k = T(s_{k-1}, d_{k-1}), k = 2, \ldots, N,$$

gdzie S_1 jest dane.

Metoda DP ma zastosowanie do rozwiązywania tzw. problemów bez pamięci, spełniających *własność Markowa*, która mówi, że ...

prawdopodobieństwo przejścia ze stanu s do stanu s' zależy tylko od stanu s, a nie od historii poprzednich stanów. Przyszłe stany procesu są warunkowo niezależne od stanów przeszłych.

Zatem, wieloetapowy proces decyzyjny ma własność Markowa, jeżeli po dowolnych k decyzjach, wartość funkcji celu f zależy tylko od stanu procesu na końcu k-tego etapu i od decyzji następnych.

W oparciu o własność Markowa sformułowana została tzw *zasada optymalności Bellmana* mówiąca, że ...

strategia optymalna ma tę własność, że niezależnie od wyboru stanu początkowego i decyzji początkowej, pozostałe decyzje muszą tworzyć strategię optymalną z punktu widzenia stanu wynikłego z pierwszej decyzji.

Rozważmy przykład maksymalizacji funkcji addytywnej.

$$F(s_1, s_2, \ldots, s_N; d_1, d_1, \ldots, s_N) = \sum_{i=1}^{N} f_i(s_i, d_i)$$

Zgodnie z zasadą optymalności Bellmana mamy

$$\max_{d_1, d_2, \ldots, d_N} (f_1(s_1, d_1) + \ldots + f_N(s_N, d_N)) =$$

$$\max_{d_1}(f_1(s_1, d_1) + \max_{d_k, \ldots, d_N} (f_2(s_2, d_2) + \ldots + f_N(s_N, d_N)))$$

Oznaczając

$$g_k(s_k) = \max_{d_k, \ldots, d_N} (f_k(s_k, d_k) + \ldots + f_N(s_N, d_N))$$

otrzymamy *równania Bellmana*

$$g_N(s_N) = \max_{d_N} f_N(s_N, d_N)$$

$$g_k(s_k) = \max_{d_k} f_k(s_k, d_k) + g_{k+1}(s_{k+1}))$$

Wynik można zapisać w postaci *równania rekurencyjnego Bellmana*

$$g_k(s_k) = \max_{d_k} f_k(s_k, d_k) + g_{k+1}(T(s_k, d_k)))$$

Wykorzystanie równań Bellmana można przedstawić w postaci następującego algorytmu:

krok 1. etap n wyznacz

$$g_N(s_N) := \max_{d_N \in D_N} \left\{ f_N(s_N, d_N) \right\}$$

kroki 2. . . . , $n-1$. etapy $n-k$ wyznacz

$$g_k(s_k) := \max_{d_k \in D_k} \left\{ f_k(s_k, d_k) + g_{k+1}(s_{k+1}) \right\}$$

krok n. etap 1 wyznacz

$$g_1(s_1) := \max_{d_1 \in D_1} \left\{ f_1(s_1, d_1) + g_2(s_2) \right\}$$

Niemniej jednak, do wyboru decyzji optymalnej wykorzystuje się odpowiednio skonstruowaną **funkcję dominacji**. Wtedy dycyzja optymalna to taka, dla której wartość funkcji dominacji przyjmuje minimum bądź maksimum.

Pytanie 1. Czy zatem, dla każdego problemu można skonstruować algorytm programowania dynamicznego?

Niemniej jednak, do wyboru decyzji optymalnej wykorzystuje się odpowiednio skonstruowanej **funkcję dominacji**. Wtedy dycyzja optymalna to taka, dla której wartość funkcji dominacji przyjmuje minimum bądź maksimum.

Pytanie 1. Czy zatem, dla każdego problemu można skonstruować algorytm programowania dynamicznego?

Niestety nie.

Niemniej jednak, do wyboru decyzji optymalnej wykorzystuje się odpowiednio skonstruowanej **funkcję dominacji**. Wtedy dycyzja optymalna to taka, dla której wartość funkcji dominacji przyjmuje minimum bądź maksimum.

Pytanie 1. Czy zatem, dla każdego problemu można skonstruować algorytm programowania dynamicznego?

Niestety nie.

Pytanie 2. Czym powinien się charakteryzować problem, aby można było zastosować programowanie dynamiczne?

Niemniej jednak, do wyboru decyzji optymalnej wykorzystuje się odpowiednio skonstruowanej **funkcję dominacji**. Wtedy dycyzja optymalna to taka, dla której wartość funkcji dominacji przyjmuje minimum bądź maksimum.

Pytanie 1. Czy zatem, dla każdego problemu można skonstruować algorytm programowania dynamicznego?

Niestety nie.

Pytanie 2. Czym powinien się charakteryzować problem, aby można było zastosować programowanie dynamiczne?

Powinien posiadać tzw. **optymalność podstruktury**, żeby na każdym etapie procesu podejmując pewną decyzję można uzyskać kolejny podproblem.
Inaczej mówiąc. Problem musi dać się zdekomponować na podproblemy tak, żeby można było zastosować do podjęcia decyzji tę samą funkcję dominacji.

Pytanie 3. Kiedy warto (można) stosować metodę programowania dynamicznego?

Pytanie 3. Kiedy warto (można) stosować metodę programowania dynamicznego?

Pytanie 3. Kiedy warto (można) stosować metodę programowania dynamicznego?

Jeżeli problem wykazuje optymalną podstrukturę, czyli jego optymalne rozwiązanie jest funkcją optymalnych rozwiązań podproblemów (dowód przez założenie istnienia lepszego rozwiązania podproblemu i pokazanie, że takie założenia przeczy optymalności rozwiązania całego problemu) oraz

Przestrzeń istotnie różnych podproblemów jest rozsądnie mała, czyli jeśli problem ma własność wspólnych podproblemów. Oznacza to, że algorytm rekurencyjny wielokrotnie oblicza rozwiązanie tego samego podproblemu, czego nie robią algorytmy oparte na DP, które rozwiązują każdy podproblem tylko raz, zapamiętują te rozwiązania oraz wykorzystują je w do rozwiązywania innych podproblemów.

Zalety DP

1. Problemy posiadające rekurencyjną strukturę podproblemów są przetwarzane w czasie wielomianowym w związku z eliminacją rozwiązywania tych samych podproblemów wielokrotnie.
2. Rozwiązywane są wszystkie podproblemy, natomiast wyniki gromadzone są w strukturze (np. tablicy) o rozmiarze wielomianowym (np. $O(n^2)$).
3. Konstruowanie optymalnego rozwiązania wykonywane jest zwykle w czasie liniowym (np. $O(n)$).
4. Możliwe jest wyznaczenie rozwiązania optymalnego dla problemu, który nie posiada algorytmu wielomianowego (np. problem plecakowy).

Pytanie 4. Czy DP posiada jakieś wady?

Zalety i wady(?) DP

1. Problemy posiadające rekurencyjną strukturę podproblemów są przetwarzane w czasie wielomianowym w związku z eliminacją rozwiązywania tych samych podproblemów wielokrotnie.
2. Rozwiązywane są wszystkie podproblemy, natomiast wyniki gromadzone są w strukturze (np. tablicy) o rozmiarze wielomianowym (np. $O(n^2)$).
3. Konstruowanie optymalnego rozwiązania wykonywane jest zwykle w czasie liniowym (np. $O(n)$).
4. Możliwe jest wyznaczenie rozwiązania optymalnego dla problemu, który nie posiada algorytmu wielomianowego (np. problem plecakowy).

1. Konieczność rekurencyjnego formułowania problemu.
2. Konieczność dowodzenia istnienia optymalnej podstruktury.
3. Konieczność zapewnienia pamięci o rozmiarze zależnym o instancji problemu.