Riassunto programmazione dinamica

Vincenzo Fraello

July 15, 2021

1 Introduzione

La programmazione dinamica è un approccio che consente di risolvere problemi di massimo e di minimo in modo esatto.

Questa tecnica di risoluzione è applicabile ai problemi che godono delle seguenti proprietà:

i. Il problema può essere suddiviso in n blocchi;

ii. In ogni blocco k, $k = 1, \ldots, n$ ci si trova in uno degli stati s_k appartenenti all'insieme di stati $Stati_k$. L'insieme $Stati_1$ del blocco 1 è costituito da un singolo stato s_1;

iii. In ogni blocco k si deve prendere una decisione d_k appartenente ad un insieme di possibili decisioni D_k. L'insieme di possibili decisioni può dipendere dallo stato s_k, ovvero $D_k = D_k(s_k)$;

iv. Se al blocco k si prende la decisione d_k e ci si trova nello stato s_k, il blocco k fornisce un *contributo* alla funzione obiettivo f del problema pari a $u(d_k, s_k)$. La funzione obiettivo f sarà pari alla somma dei contributi degli n blocchi (esistono anche varianti in cui i contributi non si sommano ma si moltiplicano tra loro ma qui ne omettiamo la trattazione);

v. Se al blocco k ci si trova nello stato s_k e si prende la decisione d_k, al blocco $k + 1$ ci troveremo nello stato $s_{k+1} = t(d_k, s_k)$. La funzione t viene detta *funzione di transizione*.

2 Principio di ottimalità

Per poter applicare un approccio di programmazione dinamica, una proprietà essenziale è quella del principio di ottimalità.

Se al blocco k mi trovo nello stato s_k, la sequenza di decisioni ottime da prendere nei blocchi $k, k+1, \ldots, n$ è totalmente indipendente da come sono giunto allo stato s_k, ovvero dalle decisioni ai blocchi $1, \ldots, k-1$ che mi hanno fatto arrivare allo stato s_k.

Attenzione a non confondere il concetto di stato con decisione. Il fatto che nel blocco k ci si trova nello stato s_k dipende dalle decisioni passate, in quanto hanno causato la transizione in quello stato. Tuttavia, all'interno di ogni stato s_k nel blocco k si possono prendere delle decisioni. La decisione presa è quella che massimizza il contributo. Ovvero, indipendentemente da come sono arrivato in un determinato stato, in questo prenderò la decisione migliore possibile.

3 Funzione f_k^*

La funzione $f_k^*(s_k)$ restituisce il valore ottimo delle somme dei contributi dei blocchi $k, k+1, \ldots, n$ quando si parte dallo stato s_k al blocco k.

3.1 Calcolo di f_k^*

Grazie al principio di ottimalità, il calcolo di f_k^* avviene a ritroso. Infatti, la decisione ottima che massimizza il contributo può essere presa indipendentemente dalle altre decisioni negli altri stati degli altri blocchi (passati).

Tipicamente è semplice calcolare $f_n^*(s_n)$ per ogni $s_n \in Stati_n$, cioè il valore ottimo del solo contributo del blocco n quando ci si trova nello stato s_n.

$$f_n^*(s_n) = \max_{d_n \in D_n(s_n)} u(d_n, s_n) \tag{1}$$

Ovvero, nel blocco n ci saranno degli stati. Per ogni stato s_n si possono prendere un insieme di decisioni. Tra tutte le possibili decisioni d_n, si deve prendere quella che fornisce il contributo massimo. Tale decisione ottima si indica con $d_n^*(s_n)$.[1]

A questo punto si procede a ritroso per calcolare $f_{n-1}^*(s_{n-1})$ per ogni stato $s_{n-1} \in Stati_{n-1}$ (cioè per tutti gli stati del blocco $n-1$).

Anche in questo caso, per ogni stato del blocco $n-1$, si possono prendere un insieme di decisioni. Tra tutte le possibili decisioni associate allo stato (per ogni stato) si deve scegliere quella che massimizza il contributo dato dalla funzione $u(d_{n-1}, s_{n-1})$. La decisione ottima si indica con d_{n-1}^*.

Prendendo tale decisione, grazie alla funzione di transizione, si giunge nello stato $s_n = t(d_{n-1}^*, s_{n-1})$.

Quando si giunge nello stato s_n, grazie al principio di ottimalità, la decisione ottima da prendere in questo stato è stata già calcolata nell'iterazione precedente.

Quindi, la funzione f_k^* fornisce la somma dei contributi ottimi a partire dal blocco $n-1$ fino ad n per tutti gli stati dei blocchi.

$$\begin{aligned} f_{n-1}^*(s_{n-1}) &= u(d_{n-1}^*(s_{n-1}), s_{n-1}) + f_n^*(t(d_{n-1}^*(s_{n-1}), s_{n-1})) = \\ &= \max_{d_n \in D_n(s_n)} \left[u\left(d_{n-1}, s_{n-1}\right) + f_n^*(t(d_{n-1}, s_{n-1})) \right] \end{aligned} \tag{2}$$

La parte di formula dopo l'uguale (quella con 'max'), dice che $f_{n-1}^*(s_{n-1})$ per ogni stato s_{n-1} nel blocco $n-1$ fornisce la somma massima tra il contributo dato dallo stato s_{n-1} e quello dato dallo stato s_n a cui si salta grazie alla funzione di transizione.

---

[1] In questo momento la funzione f_k^* restituisce il valore ottimo delle somme dei contributi dei blocchi $k, k+1, \ldots, n$ quando si parte dallo stato s_k al blocco k. In questo caso $k = n$. Il valore ritornato è il massimo, più di quello non si può ottenere, allora il valore ottenuto dalla funzione è quello ottimo.

Sostanzialmente, si possono intraprendere diverse decisioni d_{n-1} ognuna delle quali fornisce un contributo $u\left(d_{n-1}, s_{n-1}\right)$; tuttavia, si prende la decisione che massimizza questo contributo tra tutte le possibili decisioni.

Dopodiché, si procede ancora a ritroso fino a giungere al blocco 1. Arrivati al blocco 1 si ha che $Stati_1$ è formato da un unico stato s_1 e il valore $f_1^*(s_1)$ coincide con il *valore ottimo* del problema.

Per ricostruire la soluzione ottima si può partire dal blocco 1:

- al blocco 1 la decisione ottima è $d_1^*(s_1)$ e con tale decisione ci si sposta allo stato $s_2^* = t(d_1^*(s_1), s_1)$ del blocco 2;
- al blocco 2 la decisione ottima è $d_2^*(s_2^*)$. Con tale decisione ci si sposta allo stato $s_3^* = t(d_2^*(s_2^*), s_2^*)$ del blocco 3;
- al blocco 3 la decisione ottima è $d_3^*(s_3^*)$.
- si procede in questo modo fino ad arrivare al blocco n.

4 Problema dello zaino

Il problema KNAPSACK o problema dello zaino è un problema caratterizzato da:

- Uno zaino con capacità b (b è un intero positivo);
- n oggetti caratterizzati da:
  - Un valore v_i;
  - Un peso p_i.

Quello che si vuole fare, è determinare quali oggetti inserire all'interno dello zaino in maniera tale da massimizzare il valore totale ($max \sum_i v_i$) rispettando il limite di capacità b ($\sum_i p_i \leq b$).

4.1 KNAPSACK e programmazione dinamica

Il problema KNAPSACK ammette un approccio di risoluzione di programmazione dinamica:

- Gli oggetti rappresentano i blocchi.
- al blocco k lo stato s_k rappresenta la capacità residua dello zaino una volta prese le decisioni relative agli oggetti $1, 2, \ldots, k-1$. Quindi gli insiemi di stati possibili in ogni blocco sono: $Stati_k = \{0, 1, \ldots, b\}$ per $k = 2, \ldots, n$; $Stati_1 = \{b\}$.
- Le decisioni sono:

$$D_k(s_k) = \begin{cases} \{\text{NO}\} & \text{se } s_k < p_k \\ \{\text{NO}, \text{SI}\} & \text{se } s_k \geq p_k \end{cases} \tag{3}$$

- Contributo del blocco k

$$u(d_k, s_k) = \begin{cases} 0 & \text{se } d_k = \text{NO} \\ v_k & \text{se } d_k = \text{Sl} \end{cases} \tag{4}$$

- funzione di transizione:

$$t(d_k, s_k) = \begin{cases} s_k & \text{se } d_k = \text{NO} \\ s_k - p_k & \text{se } d_k = \text{Sl} \end{cases} \tag{5}$$

- Il principio di ottimaità è soddisfatto perché: se ho capacità residua dello zaino s_k, le decisioni ottime per i blocchi $k, k+1, \ldots, n$ si ottengono risolvendo un problema dello zaino con i soli oggetti da k fino a n e con capacità dello zaino s_k e non dipendono da come sono arrivato ad avere capacità residua s_k nello zaino.

  È come se si dovessero risolvere problemi dello zaino indipendenti con insiemi di oggetti ogni volta più piccoli.

References

[1] Slides del prof. Marco Locatelli. Modelli e algoritmi per il supporto alle decisioni. https://personale.unipr.it/it/ugovdocenti/person/17378, 2021.