Dynamic Programming

Arnaud Malapert, Gilles Menez, Marie Pelleau

Master Informatique, Université Côte d'Azur

Dynamic programming

Definition

Dynamic programming is a technique used to avoid computing multiple time the same subproblem in a recursive algorithm.

Alternative definition

Dynamic programming is a very powerful algorithmic paradigm in which a problem is solved by identifying a collection of subproblems and tackling them one by one, smallest first, using the answers to small problems to help figure out larger ones, until the whole lot of them is solved.

Relation to graph theory

In dynamic programming we are not given a dag; the dag is implicit. Its nodes are the subproblems we define, and its edges are the dependencies between the subproblems: if to solve subproblem B we need the answer to subproblem A, then there is a (conceptual) edge from A to B.
In this case, A is thought of as a smaller subproblem than B—and it will always be smaller, in an obvious sense

Knapsack problem

- Un des 21 problèmes NP-complets de Richard Karp.
- La formulation du problème est fort simple, mais sa résolution est plus complexe.
- Présent en tant que sous-problème d'autres problèmes plus généraux.

Référence

Knapsack Problem, Martello and Toth.

Énoncé du problème de sac-à-dos

Description

On doit remplir un sac-à-dos, ne pouvant supporter plus d'un certain poids, avec tout ou partie d'un ensemble donné d'objets ayant chacun un poids et une valeur.
Les objets mis dans le sac-à-dos doivent maximiser la valeur totale, sans dépasser le poids maximum.

Données

- Ensemble d'objets numérotés par l'indice $i \in [1, n]$.
  - w_i est le poids de l'objet i.
  - v_i est sa valeur.
- Sac-à-dos de capacité W.

Solution : de nombreuses représentations

ensemblistes, binaires, ou graphiques.

Programme linéaire

La variable x_i indique le nombre d'occurences de l'objet i.

$$\max \sum_{i=1}^{n} v_i x_i$$

$$\sum_{i=1}^{n} w_i x_i \leq W$$

Avec répétition

$x_i \in \mathbb{N}$

Sans répétition

$x_i \in \{0, 1\}$

Subset Sum

$v_i = w_i$

Question

Comment maximiser ou minimiser le nombre d'objets dans le sac-à-dos ?

Complexité

- The decision problem form of the knapsack problem (Can a value of at least V be achieved without exceeding the weight W?) is NP-complete, thus there is no known algorithm both correct and fast (polynomial-time) in all cases.
- While the decision problem is NP-complete, the optimization problem is NP-hard, its resolution is at least as difficult as the decision problem, and there is no known polynomial algorithm which can tell, given a solution, whether it is optimal (which would mean that there is no solution with a larger V, thus solving the NP-complete decision problem).
- There is a pseudo-polynomial time algorithm using dynamic programming.

Problème de sac-à-dos avec répétition

Soit $V_1(w)$ la plus grande valeur possible quand le poids du sac-à-dos est **égale à** w.

$$V_1(w) = \begin{cases} \max_{1 \leq i \leq n} v_i + V_1(w - w_i) & w > 0 \\ 0 & w = 0 \\ -\infty & w < 0 \end{cases}$$

La valeur optimale du sac-à-dos est le maxima de la fonction V_1 dans $[0, W]$.

Exercice 1 : sac-à-dos avec répétition

w_i	v_i
8	1
5	1
3	1

Table: Données ($W = 17$).

Exercice 1 : sac-à-dos avec répétition

w_i	v_i
8	1
5	1
3	1

Table: Données ($W = 17$).

w	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$V_1(w)$	0	–	–	1	–	1	2	–	2	3	2	3	4	3	4	5	4	5
$V_2(w)$	0	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5

Table: Valeurs des fonctions V_1 et V_2.

Une alternative pour le sac-à-dos avec répétition

Soit $V_2(w)$ la plus grande valeur possible quand le poids du sac-à-dos est **inférieur ou égale à** w.

$$V_2(w) = \begin{cases} \max(V_1(w), V_2(w-1)) & w \geq 0 \\ -\infty & w < 0 \end{cases}$$

La valeur optimale du sac-à-dos est $V_2(W)$.

Exercice 2 : sac-à-dos avec répétition

w_i	v_i
8	3
5	2
3	1

Table: Données ($W = 15$).

Exercice 2 : sac-à-dos avec répétition

w_i	v_i
8	3
5	2
3	1

Table: Données ($W = 15$).

w	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$V_1(w)$	0	0	0	1	1	2	2	2	3	3	4	4	4	5	5	6
$V_2(w)$	0	0	0	1	1	2	2	2	3	3	4	4	4	5	5	6

Table: Valeurs des fonctions V_1 et V_2.

Problème de sac-à-dos sans répétition

Soit $V_3(w, k)$ la plus grande valeur possible quand le poids du sac-à-dos est inférieur ou égal à w et ne contient qu'un sous-ensemble des objets de 1 à k.

$$V_3(w,k) = \begin{cases} \max(V_3(w - w_k, k-1) + v_k, V_3(w, k-1)) & w > 0 \text{ et } k > 0 \\ -\infty & w < 0 \text{ ou } k < 0 \\ 0 & \text{sinon} \end{cases}$$

Exercice 3 : sac-à-dos sans répétition

i	w_i	v_i	i	w_i	v_i
1	8	3	4	8	3
2	5	2	5	5	2
£	3	1	6	3	2

Table: Données ($W = 17$).

Exercice 3 : sac-à-dos sans répétition

i	w_i	v_i	i	w_i	v_i
1	8	3	4	8	3
2	5	2	5	5	2
£	3	1	6	3	2

Table: Données ($W = 17$).

$k \backslash w$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	0	0	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3
2	0	0	0	0	0	2	2	2	3	3	3	3	3	5	5	5	5	5
3	0	0	0	1	1	2	2	2	3	3	3	4	4	5	5	5	6	6
4	0	0	0	1	1	2	2	2	3	3	3	4	4	5	5	5	6	6
5	0	0	0	1	1	2	2	2	3	3	4	4	4	5	5	5	6	6
6	0	0	0	2	2	2	3	3	4	4	4	5	5	6	6	6	7	7

Table: Valeurs de fonction V_k