Cours de Tronc Commun Scientifique

Recherche Opérationnelle

La programmation dynamique

Frédéric Sur
École des Mines de Nancy

www.loria.fr/∼sur/enseignement/RO/

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

La programmation dynamique

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

Plus court chemin dans un graphe (1)

Donnée : un graphe orienté de sommets $\mathbf{1}, \mathbf{2}, \dots, \mathbf{N}$ & arcs valués par $d(\mathbf{i},\mathbf{j})$ (> 0 ou < 0) tel que la valuation totale des circuits soit > 0.

Problème : trouver la longueur du plus court chemin entre les sommets $\mathbf{i}$ et $\mathbf{j}$ ($\forall \mathbf{i}, \mathbf{j}$).
(Plus court chemin = de moindre coût)

Notations : $d^k_{i,j}$ longueur du plus court chemin entre $\mathbf{i}$ et $\mathbf{j}$ parmi ceux visitant des sommets intermédiaires $\mathbf{1}, \mathbf{2}, \dots, \mathbf{k}$,

$d^k_{i,j} = \infty$ si aucun tel chemin n'existe,

$d^0_{i,j} = d(\mathbf{i},\mathbf{j})$

$\rightarrow$ on cherche les $d^N_{i,j}$ ($\forall\, i,j$).

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

Plus court chemin dans un graphe (2)

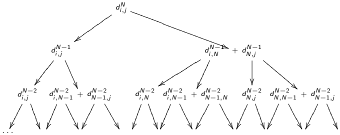
Propriété : pour $0 \leqslant k \leqslant N-1$

$$d^{k+1}_{i,j} = \min \left\{ d^k_{i,j},\ d^k_{i,k+1} + d^k_{k+1,j} \right\}$$

Preuve. De deux choses l'une :
– Soit le plus court chemin entre $\mathbf{i}$ et $\mathbf{j}$ parmi ceux visitant $\mathbf{1} \dots \mathbf{k+1}$ ne passe pas par $\mathbf{k+1}$.
– Soit il passe par $\mathbf{k}+1$ et est la concaténation du plus court chemin entre $\mathbf{i}$ et $\mathbf{k+1}$ et entre $\mathbf{k+1}$ et $\mathbf{j}$ parmi ceux visitant $\mathbf{1} \dots \mathbf{k}$. (il ne peut pas passer "plusieurs fois" par $\mathbf{k+1}$ car cela impliquerait un circuit, de valuation totale > 0 par hypothèse)

Remarque : trouver les chemins minimaux sur le graphe de taille N revient à les trouver sur des graphes de taille $k < N$.
$\rightarrow$ passage par la résolution de "problèmes plus petits".

Formulation top-down



Complexité :

– Récursivité naïve : $\mathcal{O}\left(\sum_{k=0}^{N} 3^k\right) = \mathcal{O}\left(3^N\right)$

À faire pour tous les (i,j) : $\mathcal{O}\left(N^2 3^N\right)$

– *Mémoïsation* : on garde en mémoire les $d_{i,j}^k$ déjà calculés.
Donc complexité en $\mathcal{O}\left(N^3\right)$ (mais coût récursivité...)

Formulation bottom-up

- On part du tableau des $d_{i,j}^0$.
  ($d_{i,j}^0 = d(i,j)$ si arc entre **i** et **j**, $d_{i,j}^0 = +\infty$ sinon)
- On calcule le tableau des $d_{i,j}^1$ grâce à :

$$d_{i,j}^1 = \min\left\{d_{i,j}^0,\ d_{i,1}^0 + d_{1,j}^0\right\}$$

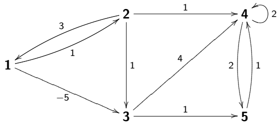
- On itère

$$d_{i,j}^{k+1} = \min\left\{d_{i,j}^k,\ d_{i,k+1}^k + d_{k+1,j}^k\right\}$$

C'est l'**algorithme de Floyd** (-Warshall) (-Roy) 1962.

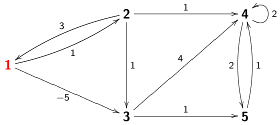
Complexité : $\mathcal{O}\left(N^3\right)$
+++ : pas de récursivité.
– : on calcule des valeurs pas forcément nécessaires pour les $d_{i,j}^N$.

Algorithme de Floyd : exemple



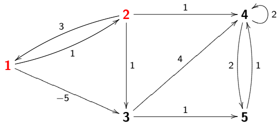
$$d^0 = \begin{pmatrix} \infty & 1 & -5 & \infty & \infty \\ 3 & \infty & 1 & 1 & \infty \\ \infty & \infty & \infty & 4 & 1 \\ \infty & \infty & \infty & 2 & 2 \\ \infty & \infty & \infty & 1 & \infty \end{pmatrix}$$

Algorithme de Floyd : exemple



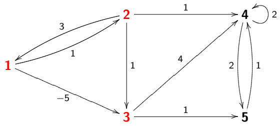
$$d^1 = \begin{pmatrix} \infty & 1 & -5 & \infty & \infty \\ 3 & 4 & -2 & 1 & \infty \\ \infty & \infty & \infty & 4 & 1 \\ \infty & \infty & \infty & 2 & 2 \\ \infty & \infty & \infty & 1 & \infty \end{pmatrix}$$

Algorithme de Floyd : exemple



$$d^2 = \begin{pmatrix} 4 & 1 & -5 & 2 & \infty \\ 3 & 4 & -2 & 1 & \infty \\ \infty & \infty & \infty & 4 & 1 \\ \infty & \infty & \infty & 2 & 2 \\ \infty & \infty & \infty & 1 & \infty \end{pmatrix}$$

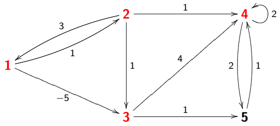
Algorithme de Floyd : exemple



$$d^3 = \begin{pmatrix} 4 & 1 & -5 & -1 & -4 \\ 3 & 4 & -2 & 1 & -1 \\ \infty & \infty & \infty & 4 & 1 \\ \infty & \infty & \infty & 2 & 2 \\ \infty & \infty & \infty & 1 & \infty \end{pmatrix}$$

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

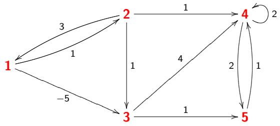
Algorithme de Floyd : exemple



$$d^4 = \begin{pmatrix} 4 & 1 & -5 & -1 & -4 \\ 3 & 4 & -2 & 1 & -1 \\ \infty & \infty & \infty & 4 & 1 \\ \infty & \infty & \infty & 2 & 2 \\ \infty & \infty & \infty & 1 & 3 \end{pmatrix}$$

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

Algorithme de Floyd : exemple



$$d^5 = \begin{pmatrix} 4 & 1 & -5 & -3 & -4 \\ 3 & 4 & -2 & 0 & -1 \\ \infty & \infty & \infty & 2 & 1 \\ \infty & \infty & \infty & 2 & 2 \\ \infty & \infty & \infty & 1 & 3 \end{pmatrix}$$

Question : Plus court chemin de **2** à **4** ?
→ Back-tracking. (cf TD, à faire en exercice)

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

La programmation dynamique

1 Introduction
- Algorithme de Floyd (-Warshall) (-Roy)

2 Problèmes d'optimisation et complexité
- Programmation dynamique et principe d'optimalité
- Un exemple standard. . .

3 Équation de Bellman

4 Conclusion

La programmation dynamique

Richard Bellman (1920-1984)
RAND Corporation

"Définition"

Programmation dynamique ($\sim$ 1953) "=" résolution d'un problème d'optimisation en le décomposant de manière récursive en sous-problèmes liés entre eux.

Programmation = établir des programmes (planning, prévision), comme dans *programmation linéaire*.
Dynamique = "I used it as an umbrella for my activities" (+ systèmes *dynamiques*. . .)

Remarque : les sous-problèmes sont de taille "juste un peu" plus petite que le pb original (pb liés).
Sinon : "diviser pour régner" (*divide and conquer*), la récursivité est alors possible (les problèmes ne sont pas liés).

Principe d'optimalité de Bellman

$\rightarrow$ Programmation dynamique utilisée pour les problèmes d'optimisation dans lesquels le *principe d'optimalité de Bellman* est vérifié :

"Une solution optimale pour le problème contient les solutions optimales pour tous les sous-problèmes."

Exemple 1 : plus court chemin dans un graphe.

Si le plus court chemin entre deux sommets X et Y est :
$X_0 = X \rightarrow X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow \dots X_{n-1} \rightarrow X_n = Y$,
alors le plus court chemin entre X_i et X_j ($i < j$) est :
$X_i \rightarrow X_{i+1} \rightarrow \dots \rightarrow X_{j-1} \rightarrow X_j$.

Exemple : problème du rendu de monnaie

Exemple 2 : quel est le nombre minimum N_n de pièces de 1, 2, 5, 10, 20 pour faire n centimes ?

Propriété :

$$N_n = 1 \quad \text{si } n \in \{1, 2, 5, 10, 20\}$$
$$N_n = \min_{k=1\dots n-1} (N_{n-k} + N_k) \quad \text{sinon.}$$

Preuve : si on découpe un ensemble optimal en 2, les deux tas réalisent le minimum sinon contradiction avec optimalité du 1er.

Résolution : N_7 au tableau.

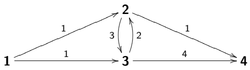
Remarque. "Autre" manière de voir :

$$N_n = 1 + \min\Big\{ N_{n-m},\ m \in \{1, 2, 5, 10, 20\} \Big\}$$

Plus court chemin dans un graphe ?

Le principe d'optimalité n'est pas toujours satisfait

Exemple : plus long chemin *simple* (*i.e.* ne contenant pas de cycle) dans un graphe.



Plus long chemin simple de **1** à **4** : **1**, **2**, **3**, **4**.

Mais : plus long chemin simple de **1** à **2** : **1**, **3**, **2**.

$\rightarrow$ Principe d'optimalité de Bellman non satisfait,
donc *programmation dynamique pas utilisable ici*...

Exemple d'application du principe d'optimalité

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

Un mois donné, une usine de fabrication de peintures doit produire 100t de peinture blanche, 50t de jaune, 30t de rouge, 20t de bleue, 20t de verte, 10t de noire.

Fabrication par brassage dans une cuve de 10t pendant 3 heures.

Au début et à la fin : cuve prête à produire du blanc.

Temps de nettoyage dépendent de la succession de couleurs :

↻	blanc	jaune	rouge	bleu	vert	noir
blanc	0	1	2	2	1	3
jaune	6	0	4	6	1	4
rouge	8	6	0	5	6	3
bleu	10	8	5	0	2	2
vert	12	10	8	4	0	2
noir	12	12	10	8	9	0

Question : Quel est le plan de production optimal ?

C'est un problème de voyageur de commerce !

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

Remarques :

1. si on commence une couleur, il faut la brasser jusqu'au bout,
2. on doit tout produire, donc on ne peut gagner du temps que sur la somme des durées de nettoyage.

C'est un pb de voyageur de commerce :
(ou Travelling Salesman Problem, TSP)

$\rightarrow$ Soit G le graphe orienté (complet) dont les sommets sont les couleurs et dont les arcs sont valués comme dans le tableau précédent, le pb est de trouver le chemin de durée minimale allant de blanc à blanc (sommet 1) en passant une et une seule fois par chaque sommet.

TSP par la programmation dynamique (1)

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

On cherche une formule récursive...
Le but est de minimiser :

$$C(x_1, S\backslash\{x_1\}) = \min_{(x_2, \dots x_N) \text{ circuit dans } S\backslash\{x_1\}} \sum_{i=1}^{N} c(x_i, x_{i+1})$$

où $x_1 = 1$ (blanc), $\forall\, i \neq j, x_i \neq x_j$, et $x_{N+1} = 1$ (blanc)
avec S l'ensemble des sommets.

Autrement écrit :

$$C(x_1, S\backslash\{x_1\}) =$$
$$\min_{x_2 \in S\backslash\{x_1\}} \left\{ c(x_1, x_2) + \min_{\substack{(x_3, \dots, x_N) \text{ circuit} \\ \text{dans } S\backslash\{x_1, x_2\}}} \sum_{i=2}^{N} c(x_i, x_{i+1}) \right\}$$

D'où :

$$C(x_1, S\backslash\{x_1\}) = \min_{x_2 \in S\backslash\{x_1\}} \Big\{ c(x_1, x_2) + C\big(x_2, S\backslash\{x_1, x_2\}\big) \Big\}$$

TSP par la programmation dynamique (2)

Conclusion : formule récursive / extension.
Soit $\mathcal{S} \subset \{1, 2, \dots N\}$, et $j \in \mathcal{S}$:

$$C(j, \{i\}) = c(j, i) + c(i, 1)$$

$$C(j, \mathcal{S}\backslash\{j\}) = \min_{i \in \mathcal{S}\backslash\{j\}} \left\{ c(j, i) + C\left(i, \mathcal{S}\backslash\{i, j\}\right) \right\}$$

est la longueur du plus court chemin commençant en j et parcourant tous les autres sommets de $\mathcal{S}$ avant d'aller en 1.

TSP par programmation dynamique

```
C(j,{i})=c(j,i)+c(i,1);
for (s=2;s<=N;s++)
   for all subsets S ⊂ {1,...,N} of size s
      for all j∈ S
         C(j,S)= min{ c(j,i)+C(i,S\{j}), i∈ S\{j}};
return C(1,{1,2,\dots,n});
```

Complexité : $\sum_{s=2}^{N} s^2 \binom{N}{s} = \mathcal{O}(\mathbf{N^2 2^N})$

TSP par la programmation dynamique (3)

Remarque : cet algo donne la *longueur* du circuit minimal.
Pour le chemin : back-tracking avec les $C(j, \mathcal{S})$ en mémoire.
(occupation mémoire exponentielle...)

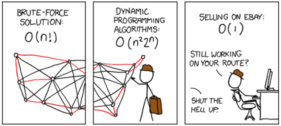
Autre méthode : "force brute" = énumérer tous les chemins possibles.
Complexité en $\mathcal{O}((N-1)!) = \mathcal{O}(N^N)$ (cf Stirling).

Avantage de la Prog. Dyn. : les sous-chemins du circuit minimal sous aussi minimaux, on ne teste donc pas *tous* les circuits possibles.

Quelques ordres de grandeurs :

N	6	10	50	100
N^N (FB)	$5 \cdot 10^4$	10^{10}	10^{85}	10^{200}
proc. 1GHz	$5 \cdot 10^{-5}$ s	10 s	10^{59} mds ans	10^{172} mds ans
$N^2 2^N$ (PD)	$2 \cdot 10^3$	10^5	10^{18}	10^{34}
proc. 1GHz	$2 \cdot 10^{-6}$ s	10^{-4}s	31 ans	10^8 mds ans

TSP par la programmation dynamique (4)



source : xkcd.com

Remarque : Programmation dynamique ne semble pas être d'une grande aide pour résoudre TSP...

Mauvaise nouvelle : il est probable qu'*aucun* programme ne soit d'une grande aide...

→ Question à 1,000,000$: **NP** $\neq$ **P ?**
(Millenium Prize, Clay Mathematics Institute)

La programmation dynamique

Équation de Bellman (1)

Elle régit un "système dynamique" sur $t = 0, 1, \dots, T$
$\rightarrow$ temps discret, horizon fini.

- x_t : état du système à l'instant t ;
- $a_t \in \Gamma(x_t)$ **(C1)** : action parmi celles possibles lorsque système en x_t ;
  il passe alors en $x_{t+1} = \phi(x_t, a_t)$ **(C2)** ;
- $F(x_t, a_t)$: "coût" à choisir l'action a_t si système en x_t ;
- $P_T(x_{T+1}) = \min_{\{a_t\}} \left\{ \sum_{t=0}^{T} F(x_t, a_t) \text{ t.q. } \textbf{(C1)} \text{ et } \textbf{(C2)} \right\}$
  "coût" global après T étapes lorsque l'on arrive en x_{T+1}.

Alors :

$$P_T(x_{T+1}) = \min_{a_T \in \Gamma(x_T) \text{ t.q. } x_{T+1} = \phi(x_T, a_T)} \left\{ F(x_T, a_T) + P_{T-1}(x_T) \right\}$$

(cf principe d'optimalité de Bellman)

Équation de Bellman (2)

$$P_T(x_{T+1}) = \min_{a_T \in \Gamma(x_T) \text{ t.q. } x_{T+1} = \phi(x_T, a_T)} \left\{ F(x_T, a_T) + P_{T-1}(x_T) \right\}$$

Généralisation :

$$P_t(x) = \min_{a \in \Gamma(y) \text{ t.q. } x = \phi(y, a)} \left\{ F(y, a) + P_{t-1}(y) \right\}$$

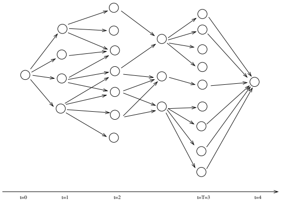
On suppose connaître les $P_0(x)$: résolution par récurrence.
$\rightarrow$ programmation dynamique !

Remarque : le problème d'optimisation est potentiellement compliqué à résoudre (par ex. $\Gamma(x)$ continu).

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

Équation de Bellman et plus court chemin

Cas discret.



Remarque : l'ensemble des "états" x_t peut être énorme par rapport à T (e.g. TSP).

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

La programmation dynamique

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion

Conclusion / Résumé

- *Programmation dynamique* : résolution de formules récursives de manière bottom-up (formulation standard) ou top-down ("mémoïsation").
- Complexité très réduite par rapport à implantation naïve, mais reste exponentielle si pb "vraiment difficile".
- Les problèmes d'optimisation qui satisfont le *principe d'optimalité de Bellman* (par ex. équation de Bellman) peuvent être traités par programmation dynamique
  → très fréquents en théorie du contrôle, économie mathématique, et en génie industriel, car les coûts/bénéfices s'additionnent par prise de décisions successives.

Référence bibliographique

- *Algorithms*, S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani. UC Berkeley 2006.

  Disponible sur internet, nombreux exercices.

Programmation dynamique
F. Sur - ENSMN
Introduction
Algorithme de Floyd (-Warshall) (-Roy)
Problèmes d'optimisation et complexité
Programmation dynamique et principe d'optimalité
Un exemple standard...
Équation de Bellman
Conclusion