EISTI Ecole Internationale des Sciences du Traitement de l'Information	**COURS DE PROGRAMMATION DYNAMIQUE**
Rédigé par : Hervé de Milleville	Ref : COU-*PRO-DYN*
Revu par : Jean-Paul Forest et Yannick Le Nir	Approuvé par :

1 Présentation

Optimiser une fonction à n variables est d'autant plus compliqué que n est grand. Un des méthodes pour simplifier consiste à résoudre le problème en résolvant des sous problèmes de dimension plus petite.
Une des méthodes parmi les plus connues est une technique algorithmique pour optimiser des sommes de fonctions monotones croissantes sous contrainte. Elle a été désignée par ce terme pour la première fois dans les années 1940 par Richard Bellman.
Le paragraphe qui suit provient du cours de programmation dynamique de Mr Djamal Rebaïne, professeur titulaire au Département d'informatique et de mathématique de l'Université du Québec à Chicoutimi Chicoutimi(Québec), Canada G7H 2B1

> *Pour la petite histoire, Bellman a choisi le terme programmation dynamique dans un souci de communication : son supérieur ne supportait ni le mot « recherche » ni celui de « mathématique ». Alors il lui a semblé que les termes « programmation » et « dynamique » donnaient une apparence qui plairait à son supérieur. En réalité, le terme programmation signifiait à l'époque plus planification et ordonnancement que la programmation au sens qu'on lui donne de nos jours. En un mot, la programmation dynamique est un ensemble de règles que tout un chacun peut suivre pour résoudre un problème donné*.

La programmation dynamique est basée sur le principe de Bellmann : toute sous solution d'une solution optimale est elle-même optimale.
En fait, ce principe n'est vérifié que si la fonction objectif est décomposable au sens suivant :
$f(x_1,...,x_n) = h_n\left(g_n(x_1,...,x_{n-1}), x_n\right)$ où h est une fonction de R^2 dans R ayant la propriété suivante :

$$\forall \beta, la\ \ fonction\ définie\ \ par\ \alpha \longrightarrow h_\beta(\alpha) = h(\alpha,\beta)\ est\ croissante$$

Si ce principe est vérifié alors : $Max_D\ f(x_1,...,x_n) = Max_{x_n}\ h_n\left(Max_{D^n_{x_n}}\ g_n(x_1,...,x_{n-1}), x_n\right)$

2 Théorème de Bellmann

2.1 Enoncé

Soit f une fonction de R^n dans R qui vérifie les propriétés suivantes :

- $\exists h_n\ et\ g_n\ deux\ fonctions\ de\ R^2 dans\ R\ et\ de\ R^{n-1} dans\ R\ telles\ que\ f(x_1,...,x_n) = h_n\left(g_n(x_1,...,x_{n-1}), x_n\right)$
- $\forall \beta, la\ \ fonction\ définie\ \ par\ \alpha \longrightarrow h_n^\beta(\alpha) = h_n(\alpha,\beta)\ est\ croissante$

alors $Max_D\ f(x_1,...,x_n) = Max_{x_n}\ h_n\left(\ Max_{D^n_{x_n}}\ g_n(x_1,...,x_{n-1}), x_n\right)$

2.2 Preuve

1) $Max_{x_n}\ h_n\left(\ Max_{D^n_{x_n}}\ g_n(x_1,...,x_{n-1}), x_n\right) \le Max_D\ f(x_1,...,x_n)$:

$$\forall x_n\ on\ note\ (x_1^*,...,x_{n-1}^*) \in D^n_{x_n} \subset R^{n-1}\ tel\ que\ g_n(x_1^*,...,x_{n-1}^*) = Max_{D^n_{x_n}}\ g_n(x_1,...,x_{n-1})$$

$$h_n\left(Max_{D_{x_n}}\ g_n(x_1,...,x_{n-1}), x_n\right) = h_n\left(g(x_1^*,...,x_{n-1}^*), x_n\right) = f\left(x_1^*,...,x_{n-1}^*, x_n\right) \le Max_D\ f\left(x_1,...,x_{n-1}, x_n\right)$$

$$\forall x_n\ h_n\left(Max\ g_{D^n_{x_n}}(x_1,...,x_{n-1}),x_n\right)\le Max_D\ f\left(x_1,...,x_{n-1},x_n\right)\Rightarrow$$
$$Max_{x_n}\ h_n\left(Max\ g_{D_{x_n}}(x_1,...,x_{n-1}),x_n\right)\le Max_D\ f\left(x_1,...,x_{n-1},x_n\right)$$

2) $Max_D\ f(x_1,...,x_n)\le Max_{x_n} h_n\left(Max_{D^n_{x_n}}\ g_n(x_1,...,x_{n-1}),x_n\right)$

$On\ note\ (x_1^*,...,x_n^*)\in R^n\ tel\ que\ \ f(x_1^*,...,x_n^*)=Max_D\ f(x_1,...,x_n)$
$f\ est\ décomposable\ donc\ f(x_1^*,...,x_n^*)=h_n\left(g_n(x_1,...,x_{n-1}),x_n\right).$
$Or\ pour\ x_n\ la\ \ fonction\ h_n^{x_n}:\alpha\longrightarrow h_n^{x_n}(\alpha)=h_n(\alpha,x_n)est\ croissante$
$$f(x_1^*,...,x_n^*)=h_n\left(g_n(x_1^*,...,x_{n-1}^*),x_n^*\right)\le h_n\left(Max_{D^n_{x_n^*}}\ g_n(x_1^*,...,x_{n-1}^*),x_n^*\right)\le Max_{x_n}\ h_n\left(Max_{D^n_{x_n}}\ g_n(x_1,...,x_{n-1}),x_n\right)$$

On en déduit $Max_D\ f(x_1,...,x_n)\le Max_{x_n} h_n\left(Max_{D^n_{x_n}}\ g_n(x_1,...,x_{n-1}),x_n\right)$

3 Application du principe de Bellman

En informatique, on connait le principe du diviser pour mieux régner. Il consiste à décomposer un problème en sous problèmes de plus petites tailles et ainsi de suite. Le principe de Bellman va permettre de résoudre le problème par décomposition. En revanche, dans le principe du "diviser pour mieux régner", la méthode est descendante tandis qu'avec le principe de Bellman, la méthode est montante : on résout d'abord les sous problèmes pour remonter au problème.

On suppose que la fonction à optimiser est complètement décomposable. On entend par là que la fonction g_n est elle-même décomposable et ainsi de suite.

Etape 1) on recense toutes les valeurs possibles de (x_1, $g_2(x_1)$). Pour x_1 donné, $g_2(x_1)$ est le maximum. C'est un cas particulier car il n'y a qu'une seule valeur, elle est donc la valeur optimale.
Etape i) ($i \geq 2$) On suppose que l'on connait les valeurs optimales g_i et on calcule les valeurs optimales g_{i+1} en appliquant le principe de décomposition. L'algorithme s'arrête à la fin de l'étape n-1.
Etape finale : on résout le dernier problème $Max_{x_n} h_n\left(Max_{D^n_{x_n}}\ g_n(x_1,...,x_{n-1}),x_n\right)$

4 Programmation dynamique discrète : Méthode des tableaux

On va exécuter un algorithme en n étapes.
L'étape i permet de résoudre l'optimisation qu'avec les variables x_1, …, x_i.
A chaque étape, on calcule un tableau formé de 3 colonnes et d'autant de lignes qu'il y a de possibilités différentes d'utiliser les ressources (contraintes).

- La première colonne contient la valeur d'utilisation des ressources,
- la deuxième colonne contient la valeur optimum de la fonction objectif pour cette utilisation des ressources
- la troisième colonne contient la valeur d'utilisation des ressources de l'étape précédente pour cet optimum (cette colonne permet à la fin de l'algorithme de faire un retour arrière pour trouver $x_1^*,...,x_n^*$. En fait, on les retrouve dans l'ordre $x_n^*,...,x_1^*$.