Chap 5. Paradigmes de conception d’algorithmes

Année universitaire :2023/2024

Plan du cours

I. **Introduction**

II. Les stratégies de résolution de problèmes

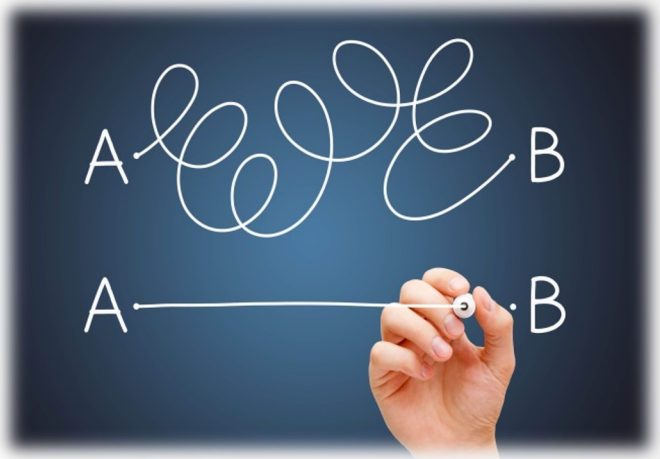
III. Paradigmes algorithmique

IV. Les heuristiques et métaheuristiques

Introduction

Un probleme P :

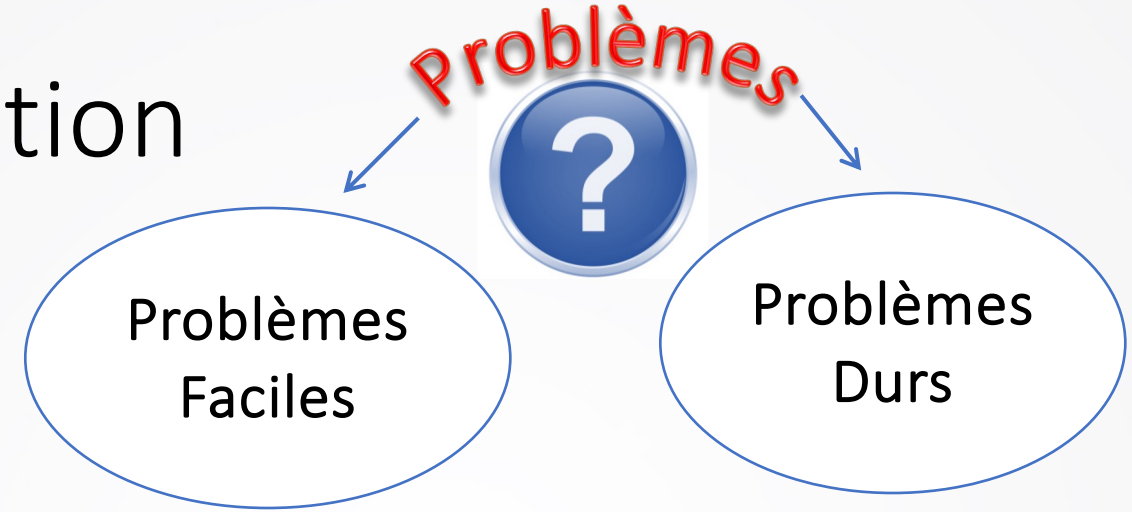
- Peut avoir une ou plusieurs solutions
- Peut avoir une solution exacte et/ou approchèe
- Peut être traité d’une ou plusieurs manières



Problème de taille n :

P(n) → Solution(s)

Introduction



Il existe d'**algorithmes exactes** de complexité raisonnable

Exemples :

- Recherche linéaire dans une liste
- Produit scalaire de deux vecteurs de taille n O(n)…

Il **n'existe pas d'algorithmes exactes** pour le résoudre.

Ou Il existe d'algorithmes exactes de complexité exponentiel.

- Il existe des **algorithmes d'approximation** avec un temps raisonnable.

Exemples : Bin Packing,Ordonnancement

Introduction

Les stratégies de résolution de problèmes peuvent être classéess en deux catégories :

- Ascendantes (Bottom-up)
- Descendantes (Top-down)

Paradigmes algorithmiques : méthodes de conception d’algorithmes se basant sur une stratégie

Méthodes de Résolution : Exactes, Heuristiques et Métaheuristiques

Plan du cours

I. Introduction

II. Les stratégies de résolution de problèmes

III. Paradigmes algorithmique

IV. Les heuristiques et métaheuristiques

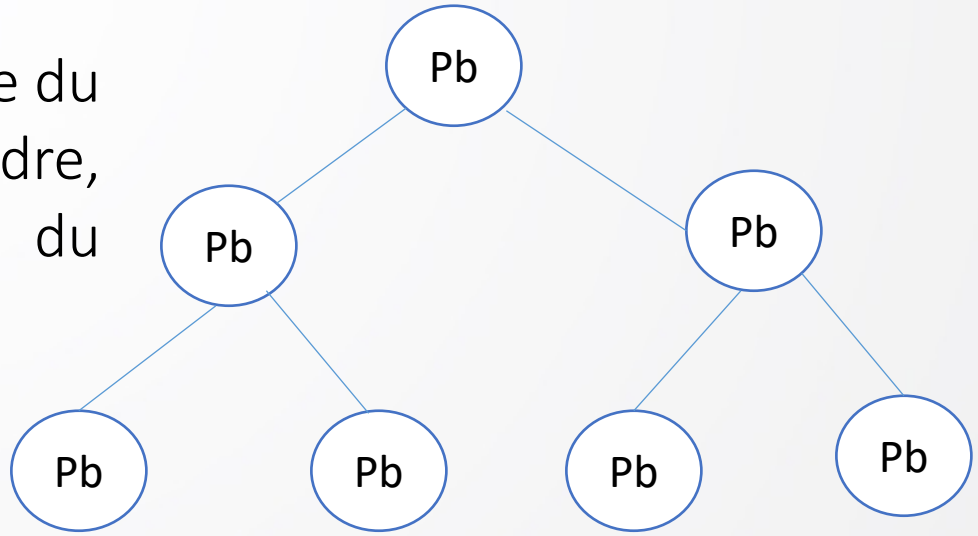
Les stratégies de résolution de problèmes

- Top-Down et Bottom-up sont des stratégies de traitement de l'information et de commande de la connaissance, utilisées pour la conception des logiciels, aussi dans d'autres domaines scientifiques.

Top-Down:

Approche descendante

Correspond à une décomposition progressive du système à construire (problème à résoudre, application, logiciel à concevoir …), allant du complexe au simple.



Les stratégies de résolution de problèmes

Top-Down:

- Dans le modèle top-Down on a un aperçu du système, il est formulé, sans entrer dans les détails pour n’importe quelle partie de celui-ci.
- Chaque partie du système est ensuite affinée par la concevoir de façon plus détaillée.
- Chaque nouvelle partie peut ensuite être raffiné par plus en détail.
- le modèle top-Down est souvent conçu avec l'aide de "boîtes noires" qui font qu'il est plus facile d’amener à son accomplissement, mais cela est insuffisant et non pertinent dans la compréhension des mécanismes élémentaires.

Les stratégies de résolution de problèmes

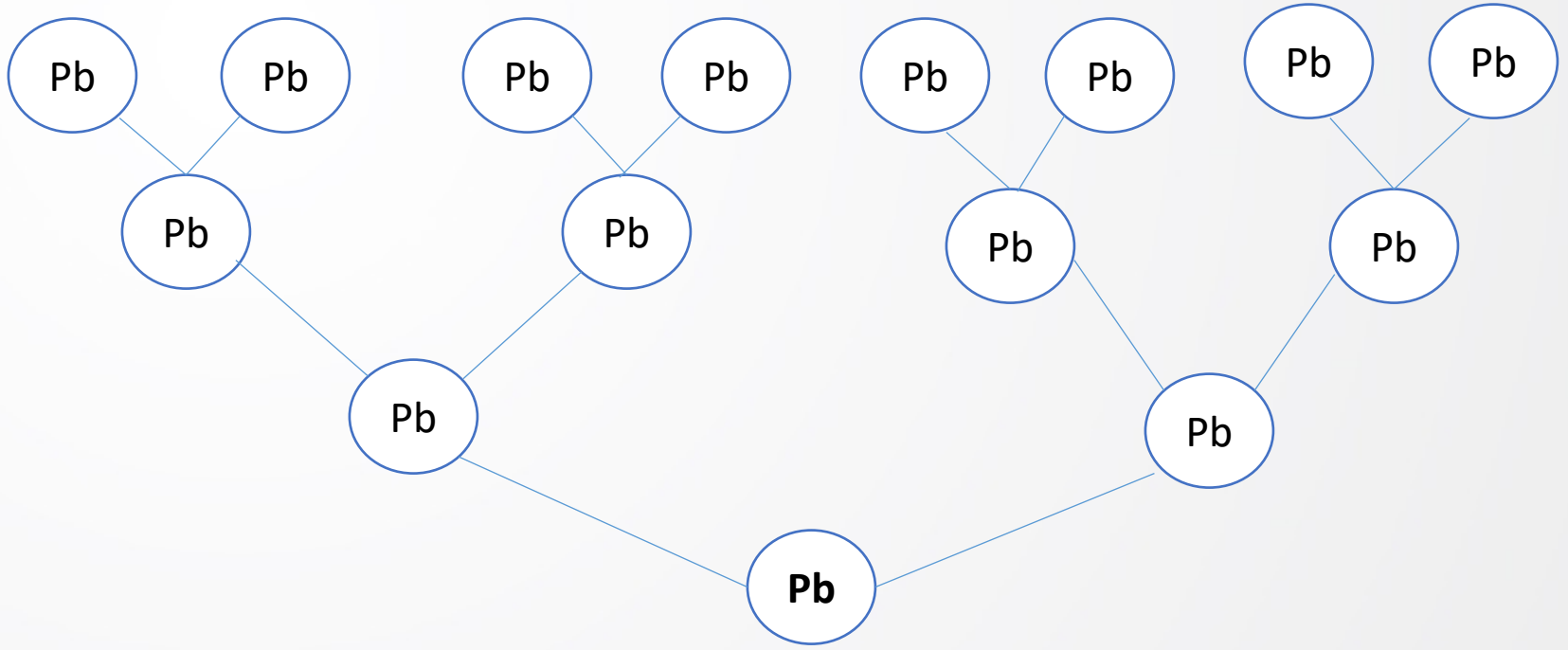
Top-Down:

- **Avantage** : On est certain que la complexité de ce qu'on étudie se réduit à chaque étape.
- **Inconvénient** 1 : la manière de décomposer n'a pas de raison d'être unique, il se peut qu'on ne choisisse pas la meilleure. Le problème se répète et se cumule à chaque nouvelle étape de décomposition.
- **Inconvénient** 2 : Le découpage d'un problème en tranches peut escamoter involontairement les questions transversales qui n'appartiennent spécifiquement ni à une tranche, ni à une autre. Pire : au moment où on examine chaque tranche, on peut de bonne foi croire que le problème transversal est du ressort de l'autre. Or, pour un problème complexe il ne semble guère possible de garder en tête simultanément tous les problèmes transversaux en suspens, sauf dans les cas où l'on sait déjà très bien formaliser pour des raisons d'habitude.

Les stratégies de résolution de problèmes

Bottom-up: Approche ascendante

- Correspond à la maîtrise progressive d'éléments simples, et que l'on combine pour cheminer vers une complexité de plus en plus grande.
- **du simple au complexe**



Les stratégies de résolution de problèmes

- Dans la conception de Bottom-up les parties individuelles du système sont décrites en détail.
- Les parties sont ensuite liés ensemble pour former de plus grands composants, qui sont à leur tour liés jusqu'à un système complet soit formé.
- Des stratégies fondées sur ce flux d'information ascendante semblent potentiellement nécessaire et suffisante, car ils sont basés sur la connaissance de toutes les variables qui peuvent affecter les éléments du système.
- Inconvénient : beaucoup d’essais et d’erreurs, et pas toujours dans les bonnes directions. On tâtonne.
- Mais en contrepartie on se familiarise avec les éléments de la résolution, on voit où on met les pieds, on acquiert des bases stables.

Les stratégies de résolution de problèmes

→ Cela suggère une approche où chaque méthode a son domaine d'usage optimal :

- le Top-down pour tout ce qu'on maîtrise à peu près dans les grandes lignes.
- Le Bottom-up pour explorer efficacement les Territoires inconnus (problème de grandes tailles).

Plan du cours

I. Introduction

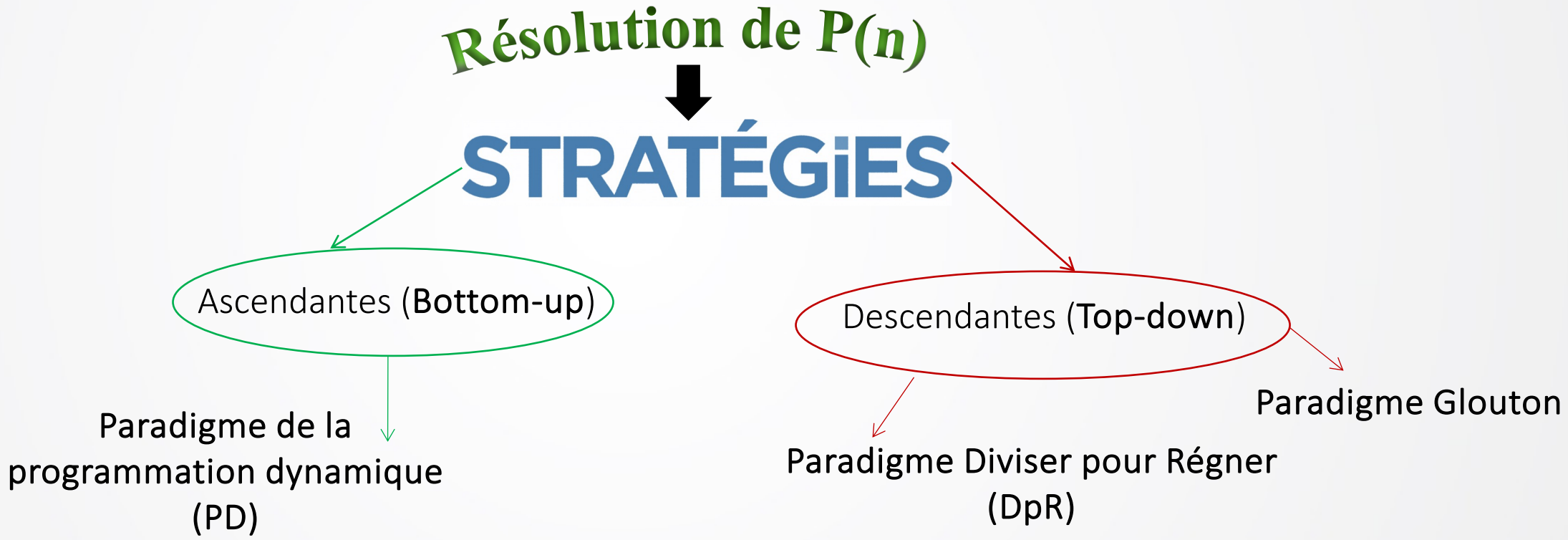
II. Les stratégies de résolution de problèmes

III. Paradigmes algorithmiquess

IV. Les heuristiques et métaheuristiques

Paradigmes algorithmique

Paradigmes algorithmiques : méthodes de conception d'algorithmes



Quelques paradigmes algorithmique

- Recherche exhaustive Appelée aussi Méthode Naïve

Principe : Examiner tous les cas possibles

Exemple : Méthode Branch-and-Bound La méthode est basée sur la Recherche Exhaustive mais permet d'atténuer l'explosion combinatoire

Quelques paradigmes algorithmique

- Pré-traitement : Pré-Conditionnement ou Preprocessing

Principe : organiser, d'ordonner, de structurer, . . . les données pour accélérer les traitements ultérieurs

Exemple : Tri, conversion

Quelques paradigmes algorithmique

- Méthode Incrémentale : Itérative, séquentielles

Principe : Les éléments de la donnée, ou des données, sont traités les uns après les autres

Exemple : Recherche Séquentielle d'un élément dans un ensemble, Sélection Séquentielle du minimum (ou du Maximum), Fusion séquentielle (ou simple) de deux suites triées (en listes ou en tableaux)

Quelques paradigmes algorithmique

- Transformation et réduction

Principe : Il s'agit de reformuler le problème avant de le résoudre ou de le ramener à un problème déjà connu

Exemple : Pour faire des opérations arithmétiques sur des chiffres romains, on les convertit en en décimal, on effectue les opérations en décimal puis on reconvertit les résultats en chiffres romains

Quelques paradigmes

- Programmation dynamique

Principe : Un algorithme de Programmation Dynamique résout chaque sous-sous-problème une seule fois et mémorise sa solution dans un tableau, une solution optimale du problème est obtenue à partir de solutions optimales de sous-problèmes

Exemple : La recherche des Plus Courts Chemins entre tous les couples de sommets d'un graphe valué, La recherche d'une Triangulation Minimale d'un polygone convexe, La construction d'Arbres Binaires Optimaux de Recherche

Quelques paradigmes

- Algorithmes gloutons : algorithmes gourmands (greedy), ou voraces

Principe : Ce sont des algorithmes itératifs construisant une solution S à un problème P pas à pas : partant de S = Ø, on construit S en extrayant parmi les éléments non encore choisis, qu'on peut appeler candidats, le meilleur élément possible sans remettre en cause ce choix.

Les algorithmes gloutons sont en général simples à concevoir et à programmer, ils utilisent souvent un tri préalable (prétraitement) sur les objets manipulés, leur complexité est raisonnable.

Exemple : FIFO, LIFO

Plan du cours

I. Introduction

II. Les stratégies de résolution de problèmes

III. Paradigmes algorithmique

IV. Les heuristiques et métaheuristiques

Les heuristiques et métaheuristiques

- Methodes exactes : ces méthodes donnent une garantie de trouver la solution optimale pour une instance de taille finie dans un temps limité et de prouver son optimalité.

Exemples : Branch and Bound, La méthode de coupes planes

Pour les problèmes durs, la complexité des algorithmes exactes est très élevée.

- Méthodes approchées : Heuristiques et métaheuristiques

Les heuristiques et métaheuristiques

Heuristique :

- une heuristique est un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale.
- Deux types d'heuristiques sont principalement utilisées : les heuristiques de construction (par exemple les méthodes gloutonnes), qui construisent itérativement une solution, et les heuristiques de descente, qui à partir d'une solution donnée cherchent un optimum local.

Les heuristiques et métaheuristiques

Les métaheuristiques :

Ces algorithmes sont plus complets et complexes qu'une simple heuristique, et permettent généralement d'obtenir une solution de très bonne qualité pour des problèmes durs.

Exemple : La recherche Tabou, Les algorithmes génétiques, Les algorithmes de colonies de fourmis, optimisation par essaim de particules

Les heuristiques et métaheuristiques

Heuristique	Métaheuristique
pas de garantie d'optimalité	pas de garantie d'optimalité
spécifique à un seul problème	Être généralisée(plusieurs problèmes)
construisant une seule solution	Construisant Un ou plusieurs solution
Problème classique	adaptables à tout type de problème d'optimisation
Recherche dépendent du problème(guider)	Recherche indépendant du problème