Algorithmes glouton et de retour sur trace. Exemples et Applications.

I. Problème d'optimisation

Définition 1 **Problème d'optimisation** [PAPA 7] Un problème d'optimisation est un problème pour lequel on cherche une solution qui :

- Satisfait certaines contraintes
- Est la meilleur possible, selon des critères bien definis

Exemple 2 [PAPA 7] La recherche d'un plus court chemin entre deux sommets s_1 et s_2 dans un graphe G est un problème d'optimisation, une solution doit :

- Être un chemin valide dans G, entre s_1 et s_2
- Minimiser le poids du chemin

Motivation 3 Cette leçon présente deux approches algorithmiques pour résoudre des problèmes d'optimisations :

- L'approche gloutone, qui est efficace, mais qui n'est pas assurée de résoudre exactement le problème
- Le retour sur trace, qui effectue un parcours optimisé de l'ensemble des solutions, mais qui peut avoir une complexité exponentielle dans le pire cas

II. Algorithmes gloutons

A. Principe des algorithmes gloutons

Définition 4 [PAPA 5] Un **algorithme glouton** construit une solution, morceau par morceau, en choisissant systématiquement le prochain morceau qui donne le plus grand bénéfice immédiat.

Exemple 5 L'algorithme de Dijkstra, qui résout le problème du plus court chemin en $O(|A| \times \log|S|)$, est un algorithme glouton basé sur une file de priorité, pour les graphes sans arcs de poids négatif.

Exemple 6 **Problème du rendu de monnaie** [TOR 9.3]

- **Entrée:** Un ensemble de n pièces de valeur respective $0 < v_1 < ... < v_n$ et une somme S à atteindre.
- **Sortie:** La distribution de pièce λ_i et p, avec: $S = \sum \lambda_i v_i$ et $p = \sum \lambda_i$.
- **Coût:** Le nombre de pièce choisies

Algorithme 7 [TOR Programme 9.7] On peut écrire un algorithme glouton pour résoudre ce problème en $O(S)$: il suffit de sélectionner à chaque fois la pièce de plus grande valeur sans que la somme choisie ne dépasse la somme voulue.

```
fonction rendue_de_monnaie(reste) :
    solution = []
    tant que (reste > 0):
        pièce = plus grand v_i <= reste
        reste -= pièce
        ajouter pièce à S
    renvoyer solution
```

Propriété 8 [TOR Théorème 9.2] Pour le système monétaire de la zone euro, alors l'algorithme glouton calcule un rendu de monnaie utilisant un nombre minimal d'éléments.
Les systèmes monétaires qui vérifient cette propriété sont dits **canoniques**.

Contre-exemple 9 Avec ces paramètres :

- $v_1 = 1, v_2 = n, v_3 = n + 1$
- S = 2n

Pour $n > 0$ quelconque, l'algorithme glouton fournie comme solution $S = (n + 1) + 1 + ... + 1$, et donc un rendu de n pièces, alors que la solution optimale est $S = n + n$, rendant ainsi 2 pièces.

DEV
Exemple 10 **Arbre couvrant minimal** [PAPA 5.1.3] On peut calculer l'arbre couvrant minimal avec une complexité $O(|A| \times \log|S|)$ à l'aide d'un algorithme glouton (Kruskal).

Exemple 11 **Codage de Huffman** [PAPA 5.2] Pour la compression de texte, on peut construire un arbre d'encodage de Huffman à l'aide d'un algorithme glouton.

Propriété 12 L'arbre de Huffman généré par l'algorithme glouton minimise, avec f_i la fréquence du caractère i et p_i sa profondeur

$$\sum_{i=1}^{n}(f_i * p_i)$$

et se calcul en $O(n \log(n))$.

B. Algorithme d'approximation

Définition 13 **Algorithme d'approximation** [PAPA 9.2] Soit un problème d'optimisation, et un algorithme $\mathcal{A}$ qui, étant donnée une instance I renvoie une solution de valeur $\mathcal{A}(I)$, et en notant $\mathrm{OPT}(I)$ la valeur de la solution optimale. Le facteur d'approximation de l'algorithme $\mathcal{A}$ est défini comme

$$\alpha_{\mathcal{A}} = \max_I \frac{\mathcal{A}(I)}{\mathrm{OPT}(I)}$$

Exemple 14 [PAPA 9.2.3] On peut utiliser les algorithmes gloutons de calcul d'arbre couvrant minimal pour établir une 2-approximation du problème de voyageur de commerce métrique.

Exemple 15 **Coloration de graphe** [TOR 9.3] Étant donné un graphe non orienté et non pondéré $G = (V, E)$, on appelle k -coloration de G, pour $k \in \mathbb{N}^*$, une application c qui associe à chacun des sommets de G un entier de $[0, k-1]$ de sorte que si deux sommets u et v sont voisins alors leur couleur sont différentes.

Algorithme 16 [TOR Programme 9.6] On peut colorier un graphe avec un algorithme glouton en $O(|S|^2)$, en prenant pour chaque sommet, la plus petite couleur disponible parmi celles des voisins :

```
fonction coloration(S,A):
  c = [0,...,0]
  Pour v dans S:
    c[v] <- min(c, voisins(v))
  renvoyer c
```

Exemple 17 **Couverture d'ensemble** [PAPA 5.4] Soit un ensemble B, le problème de couverture d'ensemble est un problème d'optimisation défini comme :

- **Entrée:** Une liste d'ensemble $S_1, ..., S_m$, avec $\forall i S_i \subseteq B$
- **Sortie:** Une liste de S_i telle que leur union soit B.
- **Coût:** Le nombre d'ensembles choisis

Propriété 18 [PAPA 5.4] S B contiens n éléments, et que la solution optimale utilise k ensembles, alors l'algorithme glouton consistant à prendre à chaque étape l'ensemble qui couvre le plus d'éléments restant utilise au plus $k \ln(n)$ ensembles.

Définition 19 **Minimum Local** [PAPA 9.3] Une solution est un minimum local si toute solution proche de cette dernière à un coût plus élevé. En résolvant le problème localement, un algorithme glouton peut trouver un minimum local, sans trouver nécessairement de minimum global.

Remarque 20 [PAPA 9.3.3] Face au problème du minimum local, plusieurs approches existent:

- **Ajout d'aléatoire**: exécuter à plusieurs reprises l'algorithme en rajoutant une notion d'aléatoire
- **Retour stimulé**: choisir occasionnellement un choix augmentant le coût total

III. Retour sur trace

Movitation 21 Même en relâchant l'optimalité, certains problèmes d'optimisation restent NP-complets. On cherche alors des algorithmes les plus optimisés possibles pour résoudre ces problèmes.

A. Principe du retour sur trace

Remarque 22 **Retour sur trace** [PAPA 9.1] Il est souvent possible de rejeter une solution en en observant seulement une petite parte. Par exemple, si une instance de SAT contient la clause $(x_1 \vee x_2)$, alors toute proposition contenant $x_1 = x_2 = \perp$ peut être immédiatement rejetée.

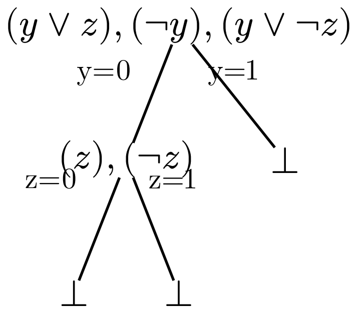
On peut alors se servir de cette observation pour éliminer rapidement des ensembles de solutions.

Algorithme 23 [PAPA 9.1.1]

```
commencer avec le problème P0
soit S = {P0} l'ensemble des sous-problèmes
tant que S non vide:
  choisir P dans S
  l'étendre en sous-problème P1, ..., Pk
  pour chaque Pi:
    si test(Pi) réussi, s'arrêter et renvoyer la solution
    si test(Pi) échoue, retirer Pi
    sinon test(Pi) incertain, rajouter Pi dans S
annoncer qu'il n'existe aucune solution
```

Remarque 24 Parcourir successivement toutes les possibilités à l'aide d'un retour sur trace revient à parcourir en profondeur un arbre représentant des solutions partielles, dont les nœuds sont des décisions, et les feuilles un succès ou un échec.

Exemple 25 **SAT** [PAPA 9.1] Pour vérifier si la formule $(y \vee z) \wedge (\neg y) \wedge (y \vee \neg z)$ est satisfiable ou non, nous pouvons parcourir les possibilités selon l'arbre ci-contre. En testant toutes les possibilités, nous pouvons montrer que cette formule est insatisfiable.



Exemple 26 **Problème des N-reines** [TOR 9.6.2] Le problème des N-reines consiste à placer N reines sur un échiquier $N \times N$ sans qu'elles soient en prise deux a deux. Le retour sur trace permet de résoudre ce problème, mais devient très lent très vite ($N \geq 30$).

Définition 27 **Mutabilité** Il est possible de séparer les structures construites par un retour sur trace en deux catégories:

- **Les structures mutables**: lors d'un retour, la structure est partiellement mise à jour en place
- **Les structures immuables**: la structure n'est dans ce cas jamais mise à jour

Remarque 28 **Complexité** Sur des problèmes NP-complets, l'algorithme 23 est exponentiel dans le pire cas. Il donne cependant un cadre générique pour résoudre des problèmes NP-complets, en réduisant l'ensemble des solutions à évaluer.

B. Séparation et évaluation

Intuition 29 Si on peut borner le coût minimal d'une solution à un sous-problème, et qu'on a déjà rencontré une meilleure solution que ce minimum, alors il est inutile d'explorer ce sous-problème, comme on ne trouvera que des moins bonnes solutions. On peut donc éliminer cette branche de l'arbre d'exploration.

Algorithme 30 **Séparation et évaluation** [PAPA 9.1.2]

```
commencer avec le problème P0
soit S = {P0} l'ensemble des sous-problèmes
meilleur_solution = ∞
tant que S non vide:
  choisir P dans S
  l'étendre en sous-problème P1, ..., Pk
  pour chaque Pi:
    si Pi est une solution: améliorer meiller_solution
    sinon:
      si coût_min(Pi) < meilleur_solution : ajouter Pi à S
renvoyer meilleur_solution
```

DEV

Exemple 31 **Problème du voyageur de commerce** [PAPA 9.1.2] On peut utiliser la méthode de séparation et évaluation pour calculer une solution exacte au problème du voyageur de commerce, en se servant d'arbre couvrant minimaux pour borner les calculs intermédiaires.

Remarque 32 L'algorithme `min-max` parcours l'arbre des solutions de manière similaire à un retour sur trace. On peut donc appliquer les méthodes de séparation et évaluation, en bornant les minimums et les maximums.

Définition 33 **Élagage $\alpha - \beta$** [TOR 9.7.2.2] Lorsque l'on calcule un minimum, on peut s'arrêter tout de suite dès qu'on est certain que la valeur de ce minimum sera inférieure ou égale à un maximum qui sera calculé au niveau supérieur. De même, un calcul de maximum peut être interrompu dès lors que la valeur de ce maximum sera supérieure ou égale à un minimum calculé au niveau supérieur.

Algorithmes glouton et de retour sur trace. Exemples et Applications.

I. Problème d'optimisation

1 Def Problème d'optimisation [PAPA 7]

2 Ex [PAPA 7]

3 Motiv

II. Algorithmes gloutons

A. Principe des algorithmes gloutons

4 Def [PAPA 5]

5 Ex

6 Ex Problème du rendu de monnaie[TOR 9.3]

7 Algo [TOR Programme 9.7]

8 Prop [TOR Théorème 9.2]

9 Contre-exemple [NAN]

10 Ex Arbre couvrant minimal[PAPA 5.1.3]

11 Ex Codage de Huffman[PAPA 5.2]

12 Prop

B. Algorithme d'approximation

13 Def Algorithme d'approximation [PAPA 9.2]

14 Ex [PAPA 9.2.3]

15 Ex Coloration de graphe[TOR 9.3]

16 Algo [TOR Programme 9.6]

17 Ex Couverture d'ensemble[PAPA 5.4]

18 Prop [PAPA 5.4]

19 Def Minimum Local[PAPA 9.3]

20 Rem [PAPA 9.3.3]

III. Retour sur trace

21 Movitation

A. Principe du retour sur trace

22 Rem Retour sur trace[PAPA 9.1]

23 Algo [PAPA 9.1.1]

24 Rem

25 Ex SAT [PAPA 9.1]

26 Ex Problème des N-reines [TOR 9.6.2]

27 Def Mutabilité

28 Rem Complexité

B. Séparation et évaluation

29 Intuition

30 Algo Séparation et évaluation[PAPA 9.1.2]

31 Ex Problème du voyageur de commerce[PAPA 9.1.2]

32 Rem

33 Def Élagage $\alpha-\beta$[TOR 9.7.2.2]

Bibliographie

[PAPA] S. Dasgupta & C. Papadimitriou & U. Vazirani, *Algorithms.*

[TOR] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen & L. Sartre, *MP2I MPI, Informatique Cours et exercices corrigés.*

Benjamin Voisin & Adrien Decosse