Recherche Opérationnelle
Optimisation combinatoire: Définition, Difficultés, Outils et Complexité

Sandra U. Ngueveu

INP-ENSEEIHT / LAAS-CNRS
sandra.ngueveu@toulouse-inp.fr - ngueveu@laas.fr

2019/2020

Définition

Soient :

- S un ensemble fini
- f une fonction qui attribue un coût $f(s)$ à chaque élément $s \in S$

Un problème combinatoire consiste à trouver l'élément s_0 de S qui minimise f.

Définition

Soient :

- S un ensemble fini
- f une fonction qui attribue un coût $f(s)$ à chaque élément $s \in S$

Un problème combinatoire consiste à trouver l'élément s_0 de S qui minimise f.

Cela revient à résoudre le problème :

$$\min f(s) \tag{1}$$

sous les contraintes (s.c.)

$$s \in S \tag{2}$$

où S est un ensemble discret

Définition

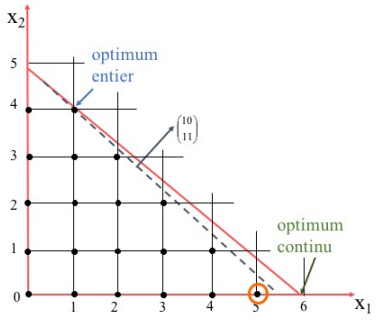
Cette définition est plus large qu'il n'y paraît au premier abord car elle englobe également :

- les problèmes de maximisation, car $\min f <=> \max(-f)$
- les problèmes de calcul d'une valeur optimale
- les problèmes de décision ou d'existence dont on attend une réponse du type "oui ou "non"

Elle peut être adaptée au cas multi-objectif, stochastique ou dynamique.

Difficulté de l'Optimisation Combinatoire

- Différence avec l'optimisation continue



- L'énumération complète est hors de question

- Apporter la solution à une instance particulière n'a aucun intérêt

Les outils de l'optimisation combinatoire sont empruntés de :

- Programmation linéaire (souvent en nombre entiers ou mixte)
- Programmation par contraintes (I.A.)
- Théorie des graphes
- Simulation, statistiques
- Algorithmique + complexité

Les outils de l'optimisation combinatoire sont empruntés de :

- Programmation linéaire (souvent en nombre entiers ou mixte)
- Programmation par contraintes (I.A.)
- Théorie des graphes
- Simulation, statistiques
- Algorithmique + complexité

Certains calculs peuvent être faits par des outils commerciaux d'optimisation, mais bien souvent il faut concevoir un algorithme car il n'existe pas de méthode générique performante quelque soit le problème ou l'instance en optimisation combinatoire.

Complexité d'un problème / Complexité d'un algorithme

Les problèmes d'Optimisation Combinatoire se répartissent en deux grandes classes :

- ceux qui sont résolus de manière optimale par des algorithmes polynomiaux (classe P)
- ceux dont la résolution optimale peut prendre un temps exponentiel dans le pire cas (classe NP)

La notion de problème *NP-complet/NP-difficile* permet de mieux comprendre pourquoi certains problèmes ne disposent toujours pas d'algorithmes efficaces à l'heure actuelle.

La complexité d'un problème corresponds à la complexité du meilleur algorithme capable de le résoudre.

Théorie de la complexité

Algorithme

Un algorithme est une suite d'opérations élémentaires qui, lorsqu'on lui fournit une instance d'un problème en entrée s'arrête après exécution de la dernière opération en nous renvoyant la solution.
(*Source : Optimisation Combinatoire - Vangelis Th Paschos - 2005*)

Définition, Difficultés, Outils et Complexité

Algorithme

Un algorithme est une suite d'opérations élémentaires qui, lorsqu'on lui fournit une instance d'un problème en entrée s'arrête après exécution de la dernière opération en nous renvoyant la solution.
(*Source : Optimisation Combinatoire - Vangelis Th Paschos - 2005*)

Opération élémentaire = opération simple (affectation de vars, tests, ...)

- 1 opération algébrique : +, -, *, /
- 1 test logique ou connecteurs logiques : ou, et, non, xor
- 1 test booléen : $<$, $>$, $=$, $\neq$
- ...

Evaluation d'un algorithme

Comment évaluer la qualité d'un algorithme ou comparer 2 algorithmes ?

Les deux critères les plus importants sont :

- le temps de calculs
- l'espace mémoire utilisé

Définition, Difficultés, Outils et Complexité

Evaluation d'un algorithme

Comment évaluer la qualité d'un algorithme ou comparer 2 algorithmes ?

Les deux critères les plus importants sont :

- le temps de calculs
- l'espace mémoire utilisé

D'autres critères, comme la difficulté d'implémentation, la fiabilité, ... peuvent rentrer en compte.

Certains résultats peuvent fortement dépendre des instances résolues et des données utilisées. Mais en général, ce que l'on veut mesurer en priorité c'est si l'algorithme gardera, même dans les pires cas, des temps de calcul raisonnables.

Evaluation d'un algorithme

Complexité [1]

La complexité est une fonction de la taille des données pour prédire l'ordre de grandeur de la variation des temps de calculs quand on passe de petits jeux d'essai de grands problèmes.

Cela permet aussi d'estimer la taille maximale des problèmes qui pourront être résolus en pratique.

1. rien à voir avec la difficulté de programmation ! !

Définition, Difficultés, Outils et Complexité

Evaluation d'un algorithme

Complexité [1]

La complexité est une fonction de la taille des données pour prédire l'ordre de grandeur de la variation des temps de calculs quand on passe de petits jeux d'essai de grands problèmes.

La complexité d'un algorithme A travaillant sur des données est une fonction qui exprime le nombre d'instructions exécutées :

- à l'ordre près
- dans le pire des cas,
- en fonction de la taille n des données.

1. rien à voir avec la difficulté de programmation ! !

Sandra U. Ngueveu (N7 - LAAS) Recherche Opérationnelle - CTD - APP S 2019/2020 9 / 21

Evaluation d'un algorithme

Il y a donc deux types d'algorithmes, en fonction de leur complexité :

- ceux dits polynomiaux
  - complexité de l'ordre d'un polynôme
  - ex : $logn$, $n^{0.5}$, $n \log n$, n^2, $O(n^{\log n})$, $O(n^{2.5})$, ...
- ceux dits exponentiels
  - vraie exponentielle au sens mathématique e, 2^n, ...
  - fonction comme factorielle n !, ou n^n, k^n, n^n, ...

Explosion Combinatoire

Taille-> Complexité	20	50	100	200	500	1000
10^3n	0.02 s	0.05 s	0.1 s	0.2 s	0.5 s	1 s
10^3n $\log_2$n	0.09 s	0.3 s	0.6 s	1.5 s	4.5 s	10 s
100 n^2	0.04 s	0.25 s	1 s	4 s	25 s	2 min
10 n^3	0.02 s	1 s	10 s	1 min	21 min	27 h
$n^{\log 2\, n}$	0.4 s	1.1 h	220 j	12500 ans	5.10^{10} ans	--
$2^{n/3}$	0.0001 s	0.1 s	2.7 h	$3.\ 10^6$ ans	--	--
2^n	1 s	36 ans	--	--	--	--
3^n	58 min	$2.\ 10^{11}$ ans	--	--	--	--
n!	77100 ans	--	--	--	--	--

Figure – Croissance du temps de calcul pour différentes complexités

Les cases non remplies ont des durées supérieures à 1000 miliards d'années.

Conséquence de la complexité d'un algorithme

A noter

- une exponentielle dépasse tout polynôme pour n assez grand.
- en pratique, il est rare d'avoir des complexités polynomiales d'ordre supérieur à n^5.
- les algorithmes exponentiels sont inexploitables en pratique pour les grands problèmes.

Définition, Difficultés, Outils et Complexité

Conséquence de la complexité d'un algorithme

A noter

L'énumération complète de solutions est exponentielle (ex à résoudre)

- remplir au maximum une disquette avec n fichiers ?
- affecter n personnes à n postes ?
- n coups aux échecs, k choix par coup ?

Sandra U. Ngueveu (N7 - LAAS) Recherche Opérationnelle - CTD - APP S 2019/2020 12 / 21

Comment résoudre un problème NP-difficile ?

- énumération complète (si très petits problèmes)
- énumération intelligente (problèmes de taille moyenne)
  - programmation dynamique (résolution de sous-problèmes emboités),
  - méthodes arborescentes (séparation en sous-cas dont beaucoup sont éliminés par des tests).
  - ...
- méthodes approchées (problèmes de grande taille) : bonnes solutions mais sans garantie d'optimalité.
  - (méta-)heuristiques
  - recherches locales
  - ...

Par ex, pour des problèmes de transport NP-difficiles (TSP, ...), seules des heuristiques peuvent traiter les très grandes instances (milliers de sommets).

Exemple classique

Il suffit parfois d'ajouter une contrainte/variable/donnée/variante pour qu'un problème passe de *polynomial* à *NP-difficile*.

Le pb d'affectation

Soient n produits et n machines. Fabriquer i coûte d_{ij} si i est affecté au à la machine j.
On veut affecter les produits aux machines pour tout traiter en minimisant le coût total.

... MODELE MATH ... ?

POLYNOMIAL ! ! !

Le pb d'affectation généralisée

Soient m produits, n machines. Fabriquer i dure a_{ij} et coûte d_{ij} si i est fabriqué par la machine j. La durée max de travail d'une machine j est b_j.
On veut affecter les produits aux machines pour tout traiter en minimisant le coût total.

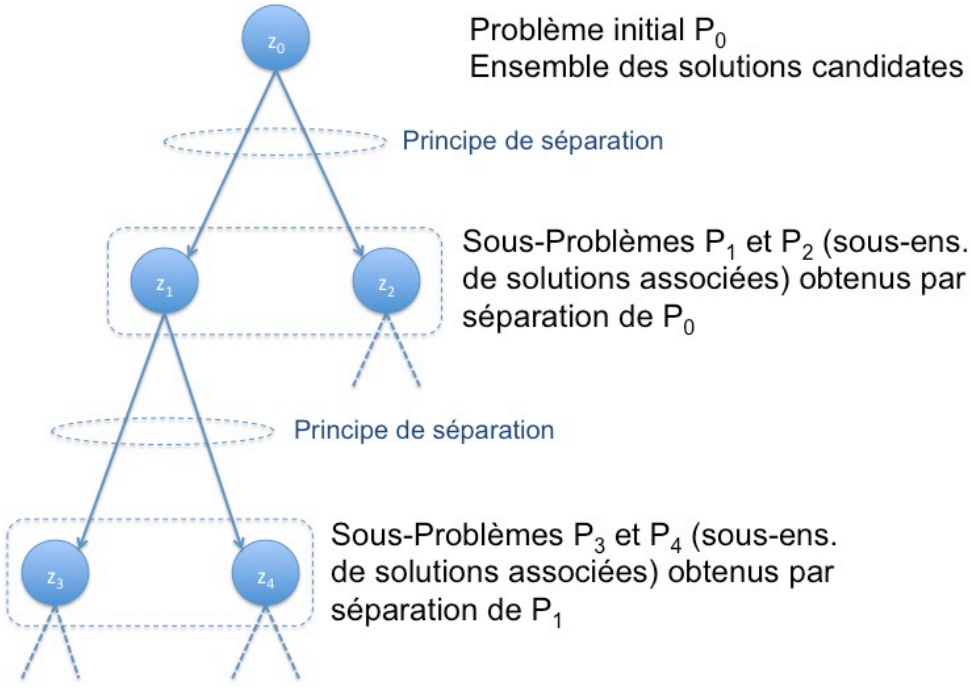
... MODELE MATH ... ?

NP-DIFFICILE ! ! !

Résumé de ce qui a été vu

- Définition de l'O.C.
- Relation entre O.C. et optimisation continue
- Problème vs instance : besoin d'algorithmes
- Comment évaluer un algorithme
- Bases de la théorie de la complexité
- Explosion combinatoire : pourquoi un PC plus puissant ne résouds rien

Procédures de séparation et évaluation pour PLNE



Procédures de séparation et évaluation pour PLNE

Principe : (schéma slide précédent)

- Séparer progressivement le problème en sous-problèmes traitables

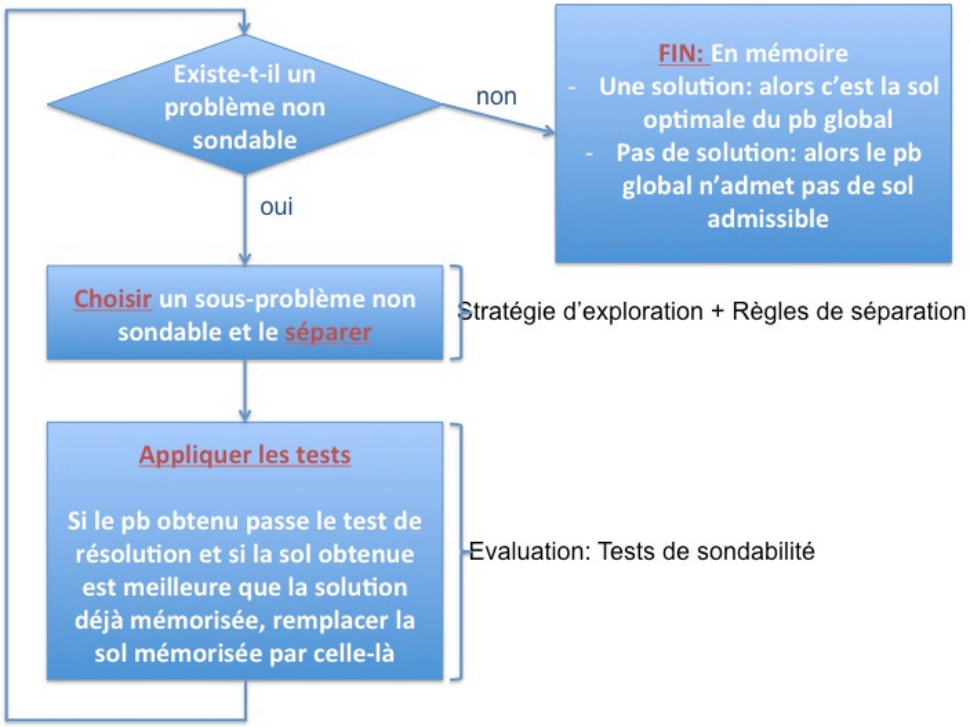
Objectif

- Enumérer intelligemment l'espace des solutions
- Hiérarchiser les ss-pbs sous forme d'arbre
- "tuer" des branches/nœuds au plus tôt lors de l'exploration de l'arbre

3 composantes :

- Règle de Séparation
- Evaluation/Tests de Sondabilité (TA, TO, TR)
- Stratégie d'exploration

Procédures de séparation et évaluation pour PLNE



Tests de Sondabilité

Un nœud est sondable si on peut répondre "OUI" à un des 3 tests suivants :

TA : Test d'admissibilité = montrer qu'il n'y a pas de sol admissibles

- ex1 : si contrainte du type $\geq b_i$, alors évaluer majorant G_i^k du membre de gauche. Si $G_i^k < b_i$ alors même dans le meilleur des cas on ne pourra pas valider cette contrainte, donc inutile de continuer sur cette branche, "tuer" le nœud correspondant
- ex2 : raisonnement inverse si contrainte du type $\leq b_i$
- ex3 : mq une relaxation du pb est déjà infaisable

Définition, Difficultés, Outils et Complexité

Tests de Sondabilité

Un nœud est sondable si on peut répondre "OUI" à un des 3 tests suivants :

TO : Test d'optimalité = mq la sol opt ne peut pas être au sein de ce nœud

- mq les coûts des sols du nœud seront toujours pires que celui d'une solution déjà connue
- ex : calcul de BINF (E_i) à comparer avec le coût d'une solution déjà connue (préalablement obtenue par la même PSE ou un autre algorithme)

Tests de Sondabilité

Un nœud est sondable si on peut répondre "OUI" à un des 3 tests suivants :

TR : Test de résolution = mq le ss-pb obtenu se résoud de manière triviale

- ss-pb à une seule variable
- ss-pb polynomial

Définition, Difficultés, Outils et Complexité

Tests de Sondabilité

Un nœud est sondable si on peut répondre "OUI" à un des 3 tests suivants :

Actions

- Un nœud sondable peut être éliminé.
- Un nœud non sondable doit être séparé en plusieurs ss-pbs suivant la règle de séparation prévue.

Une PSE efficace trouve un compromis entre la qualité des résultats fournis par les tests et l'investissement accepté en temps et volumes de calculs, car ces calculs doivent être faits pour chaque nœud exploré.

Sandra U. Ngueveu (N7 - LAAS) Recherche Opérationnelle - CTD - APP S 2019/2020 19 / 21

Principe de Séparation

- Aucune solution ne doit être perdue ou ajoutée
  - l'union des ss-pbs reforme le pb de départ
- Souvent un choix heuristique validé par des tests numériques
  - choix glouton (ex : le plus petit arc libre pour PVC)
  - choix le plus informant (ex : la variable présente dans le plus grd nbre de contraintes)
  - ...
- Le but est :
  - soit de précipiter la découverte d'une nouvelle solution
  - soit de détecter au plus vite des contradictions prouvant qu'un nœud est vide

Principe de Séparation

- Une séparation poussée à l'extrême revient à fragmenter en ss-ens à une solution, donc nbre exponentiel de nœuds explorés (énumération complète). Pour éviter cela, on évite de séparer le nœud-père en "trop" de nœuds fils et on compte sur les tests TA,TO,TR pour réduire l'arborescence en identifiant au plus tôt les nœuds qui ne valent pas la peine d'être séparés à nouveau.

Par convention, les nœuds sont souvent numérotés dans l'ordre de leur création.

Stratégie d'Exploration

Il s'agit de définir l'ordre suivant lequel les nœuds seront étudiés

PSEP

- Priorité au nœud qui donne la BINF de plus petite valeur
- Avantage : Les sol réalisables trouvées sont tt de suite de très bonne qualité
- Inconvénient : Requiert plus de mémoire (liste des nœuds non séparés)

PSES

- Priorité toujours "à gauche" (1er des noeuds-fils nouvellement créés)
- Avantage : Requiert moins d'espace mémoire, Trouve plus vite une solution réalisable
- Inconvénient : Globalement moins rapide que PSEP

Autres

- Probabilité sur le choix de la prochaine branche à explorer
- ...

Les tests numériques et les contraintes technologiques (espace mémoire, ...) permettent de valider/invalider le choix d'une stratégie d'exploration par rapport à une autre.