Optimisation combinatoire difficile

Janvier 2019

Introduction

Optimisation Combinatoire Difficile

C'est dans l'air du temps.

Optimisation Combinatoire Difficile

C'est dans l'air du temps.

Variables nominales / à domaines finis / NON continues / $\in \mathbf{N}$
$\Rightarrow$ *pas de dérivées partielles, pas de calcul matriciel, ...*

Optimisation Combinatoire Difficile

C'est dans l'air du temps.

Variables nominales / à domaines finis / NON continues / $\in \mathbf{N}$
$\Rightarrow$ *pas de dérivées partielles, pas de calcul matriciel, ...*

On veut une garantie de qualité des solutions
$\Rightarrow$ *pas de métaheuristiques.*

Optimisation Combinatoire Difficile

C'est dans l'air du temps.

Variables nominales / à domaines finis / NON continues / $\in \mathbf{N}$
$\Rightarrow$ *pas de dérivées partielles, pas de calcul matriciel, ...*

On veut une garantie de qualité des solutions
$\Rightarrow$ *pas de métaheuristiques.*

Problèmes pour lesquels
pas d'algorithme complet connu qui s'exécute en temps polynomial

Optimisation Combinatoire Difficile

C'est dans l'air du temps.

Variables nominales / à domaines finis / NON continues / $\in \mathbf{N}$
$\Rightarrow$ *pas de dérivées partielles, pas de calcul matriciel, ...*

On veut une garantie de qualité des solutions
$\Rightarrow$ *pas de métaheuristiques.*

Problèmes pour lesquels
pas d'algorithme complet connu qui s'exécute en temps polynomial
$\Rightarrow$ *pas de calcul de flot, plus difficile que prog. linéaire ...*

Optimisation Combinatoire Difficile

C'est dans l'air du temps.

Variables nominales / à domaines finis / NON continues / $\in \mathbf{N}$
$\Rightarrow$ *pas de dérivées partielles, pas de calcul matriciel, ...*

On veut une garantie de qualité des solutions
$\Rightarrow$ *pas de métaheuristiques.*

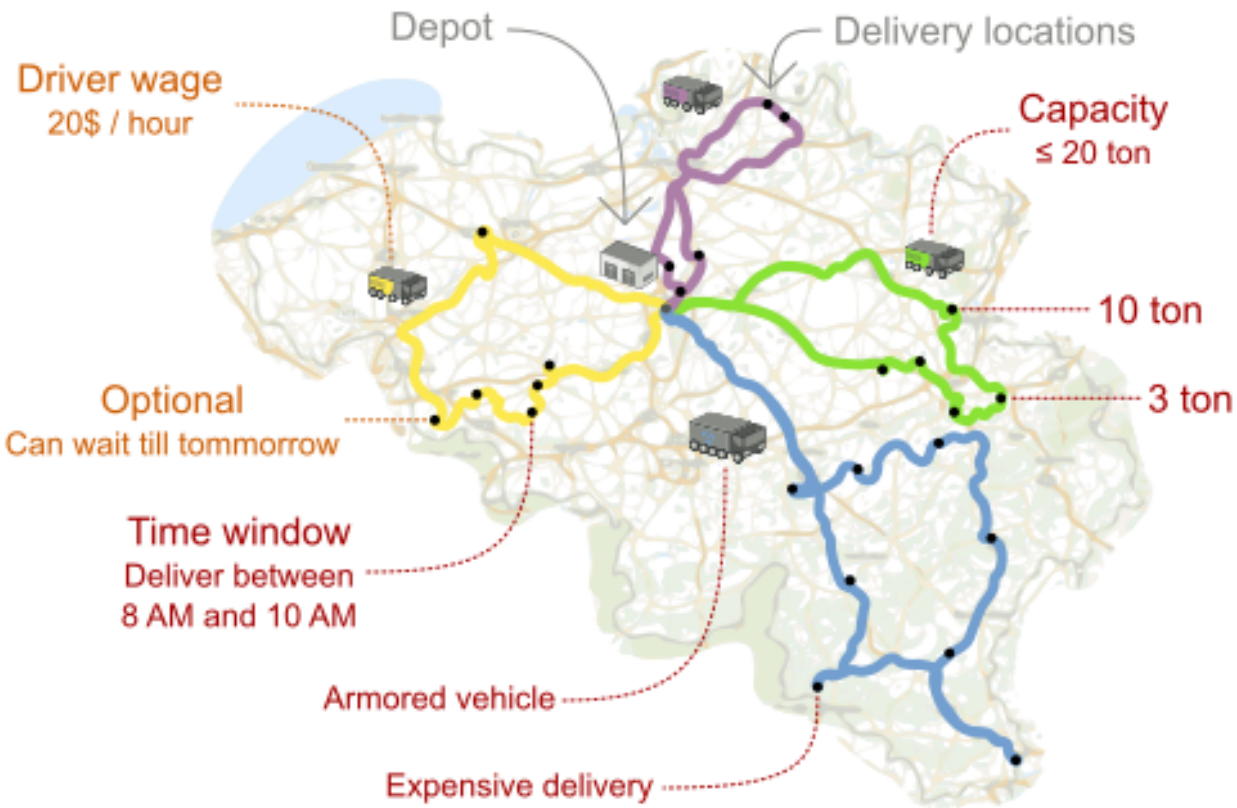
Problèmes pour lesquels
pas d'algorithme complet connu qui s'exécute en temps polynomial
$\Rightarrow$ *pas de calcul de flot, plus difficile que prog. linéaire ...*

Nombreux domaines d'application :

- Recherche opérationnelle
- Systèmes embarqués
- Machine learning
- Intelligence Artificielle
- ...

Exemples introductifs : Routage de véhicules

- Des clients à livrer à certaines heures
- Des livreurs et des camions, de capacités limitées
- Un dépôt

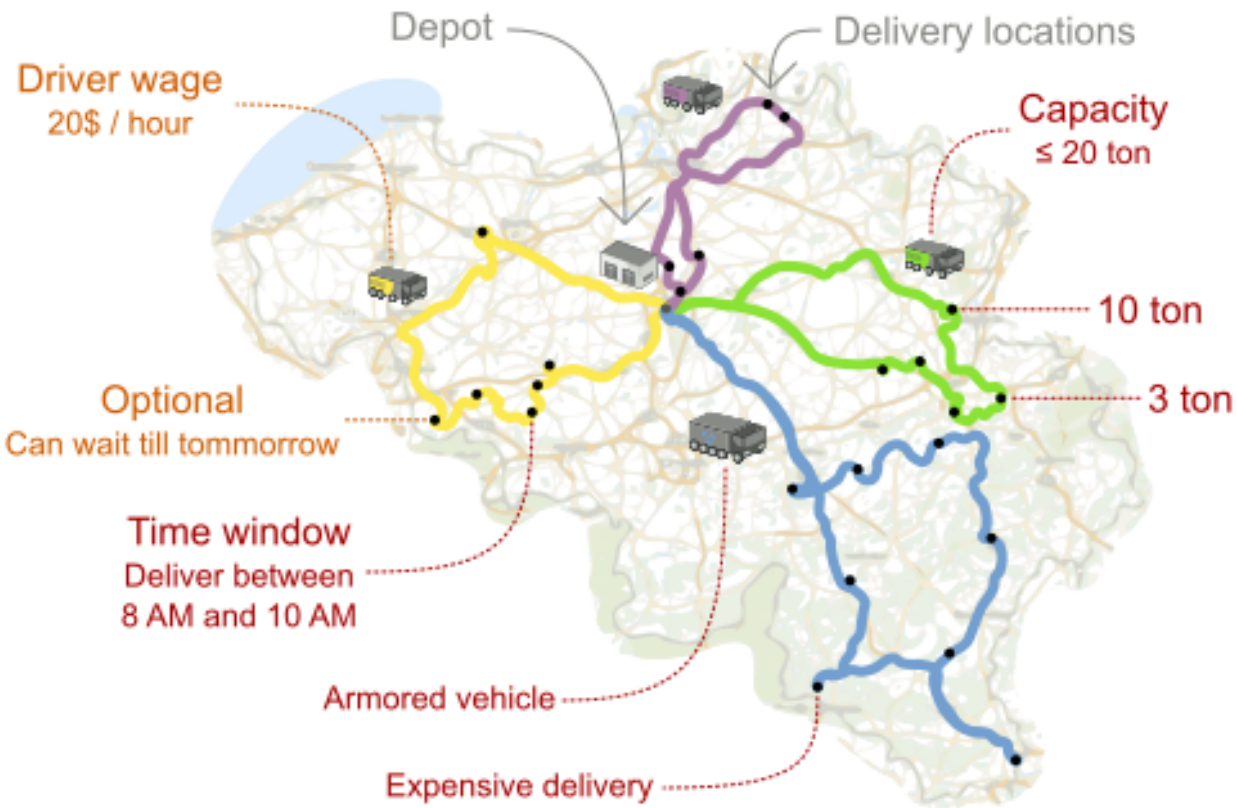


(Emprunté à optaplanner.org)

Exemples introductifs : Routage de véhicules

- Des clients à livrer à certaines heures
- Des livreurs et des camions, de capacités limitées
- Un dépôt

$\Rightarrow$ minimiser carburant / temps transport / nb. camions / retards...



(Emprunté à optaplanner.org)

Exemples introductifs : Routage de véhicules

- Des clients à livrer à certaines heures
- Des livreurs et des camions, de capacités limitées
- Un dépôt

$\Rightarrow$ minimiser carburant / temps transport / nb. camions / retards...

Un peu plus formellement :

- n points de livraisons, numérotés $1, \ldots, n$
- p camions initialement au dépôt, numéroté 0
- matrice de distances / coûts $d(i, j)$

Exemples introductifs : Routage de véhicules

- Des clients à livrer à certaines heures
- Des livreurs et des camions, de capacités limitées
- Un dépôt

$\Rightarrow$ minimiser carburant / temps transport / nb. camions / retards...

Un peu plus formellement :

- n points de livraisons, numérotés $1, \ldots, n$
- p camions initialement au dépôt, numéroté 0
- matrice de distances / coûts $d(i, j)$
- contraintes horaires :
  - livraison i dure t_i, à faire entre h_i et H_i
  - heures de départ et de retour max. h_0 et H_0
- contraintes de capacité :
  - chaque livraison nécessite capacité C_i
  - camions capacité totale C

Exemples introductifs : Routage de véhicules

- Des clients à livrer à certaines heures
- Des livreurs et des camions, de capacités limitées
- Un dépôt

$\Rightarrow$ minimiser carburant / temps transport / nb. camions / retards...

Un peu plus formellement :

- n points de livraisons, numérotés $1, \ldots, n$
- p camions initialement au dépôt, numéroté 0
- matrice de distances / coûts $d(i, j)$
- contraintes horaires :
  - livraison i dure t_i, à faire entre h_i et H_i
  - heures de départ et de retour max. h_0 et H_0
- contraintes de capacité :
  - chaque livraison nécessite capacité C_i
  - camions capacité totale C

$\Rightarrow$ Trouver un itinéraire pour chaque camion qui permette de tout livrer à temps et qui minimise les coûts / retards

Exemples introductifs : Routage de véhicules

Une solution =

permutation de l'ensemble des n points de livraisons,
$+p$ marqueurs indiquant les limites de chaque camion :

$$4, 8, \ldots, 5, \diamond, 2, 12, \ldots, 3$$
$$\uparrow$$
nouveau camion

A priori de l'ordre de $n! \times (n+1)^p$ solutions potentielles à explorer (mais nombreuses symétries). p est le nombre de camions

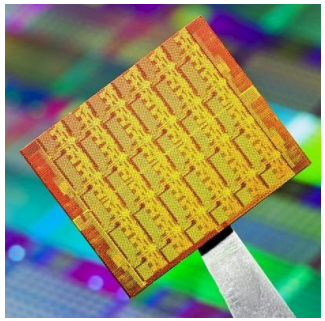
La plupart ne vérifient pas les contraintes.

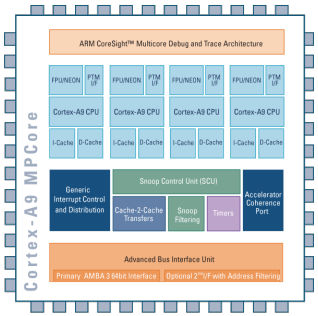
Point de vue «*informatique*» :
une solution = p listes disjointes d'éléments de $\{1, \ldots, n\}$

Exemples introductifs : MultiProcessor Systems on Chip

- Parallélisme
- Plusieurs mémoires / cœurs basse-fréquence

= basse consommation



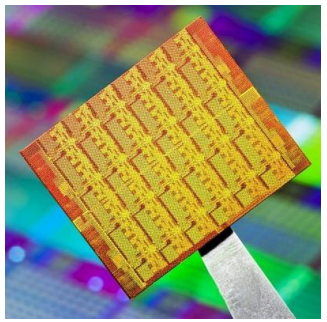


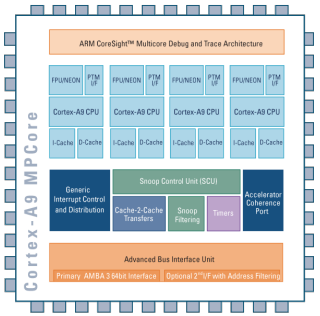
Exemples introductifs : MultiProcessor Systems on Chip

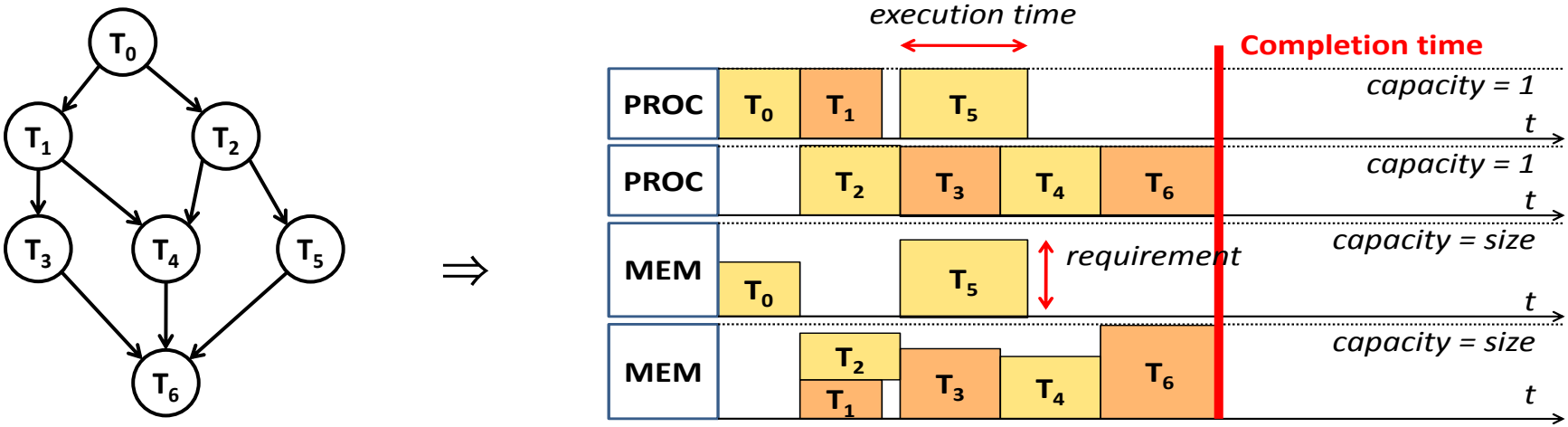
- Parallélisme
- Plusieurs mémoires / cœurs basse-fréquence

= basse consommation

- Nombreuses tâches,
  avec temps d'exécution, mémoire nécessaire,...
  avec contraintes de précédence







(Emprunté à Luca Benini – Université de Bologne)

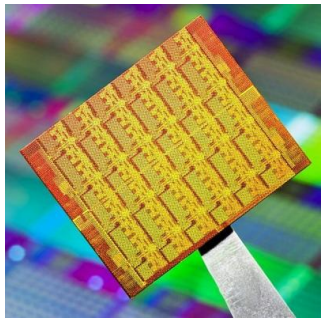
Exemples introductifs : MultiProcessor Systems on Chip

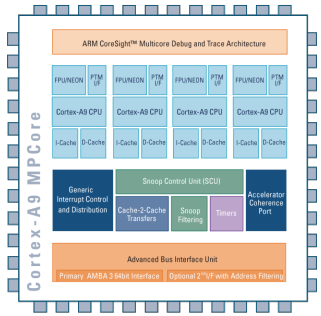
- Parallélisme
- Plusieurs mémoires / cœurs basse-fréquence

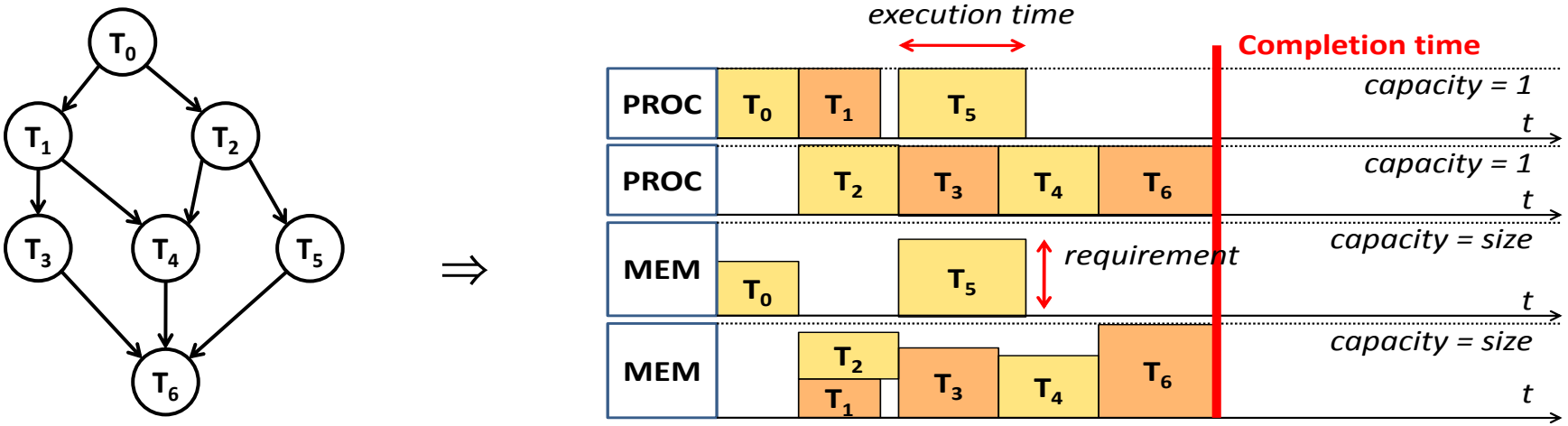
= basse consommation

- Nombreuses tâches,
  avec temps d'exécution, mémoire nécessaire,...
  avec contraintes de précédence

⇒ optimiser l'utilisation des ressources :
temps de calcul, consommation énergétique,...







(Emprunté à Luca Benini – Université de Bologne)

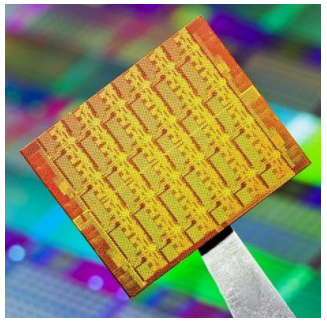
Exemples introductifs : MultiProcessor Systems on Chip

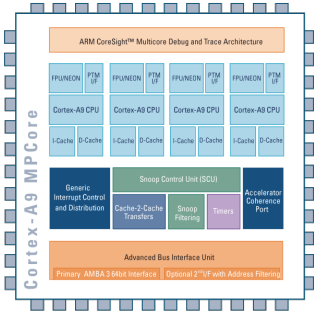
- Parallélisme
- Plusieurs mémoires / cœurs basse-fréquence

$=$ basse consommation

- Nombreuses tâches,
  avec temps d'exécution, mémoire nécessaire,...
  avec contraintes de précédence

$\Rightarrow$ optimiser l'utilisation des ressources :
temps de calcul, consommation énergétique,...





Plus formellement :

- n tâches numérotées de 1 à n, p cœurs identiques, q mémoires

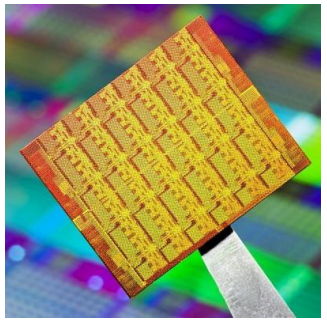
Exemples introductifs : MultiProcessor Systems on Chip

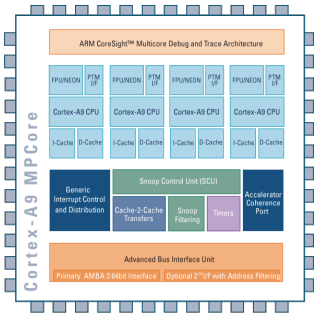
- Parallélisme
- Plusieurs mémoires / cœurs basse-fréquence

$=$ basse consommation

- Nombreuses tâches,
  avec temps d'exécution, mémoire nécessaire,...
  avec contraintes de précédence

$\Rightarrow$ optimiser l'utilisation des ressources :
temps de calcul, consommation énergétique,...





Plus formellement :

- n tâches numérotées de 1 à n, p cœurs identiques, q mémoires
- contraintes de précédences
  $i \prec j$: tâche i doit être finie avant que tâche j commence
- contraintes de ressources
  - chaque tâche i : durée d_i, mémoire nécessaire m_i
  - mémoire $j \Rightarrow$ capacité C_j

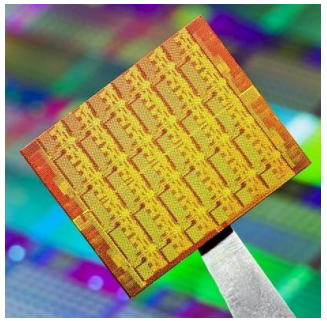
Exemples introductifs : MultiProcessor Systems on Chip

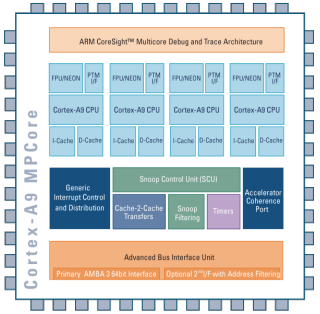
- Parallélisme
- Plusieurs mémoires / cœurs basse-fréquence

= basse consommation

- Nombreuses tâches,
  avec temps d'exécution, mémoire nécessaire,...
  avec contraintes de précédence

⇒ optimiser l'utilisation des ressources :
temps de calcul, consommation énergétique,...





Plus formellement :

- n tâches numérotées de 1 à n, p cœurs identiques, q mémoires
- contraintes de précédences
  $i \prec j$: tâche i doit être finie avant que tâche j commence
- contraintes de ressources
  - chaque tâche i : durée d_i, mémoire nécessaire m_i
  - mémoire j ⇒ capacité C_j

p^n * q^n

⇒ minimiser coût énergétique / temps de complétion / ...

Instances de problèmes d'Optimisation Combinatoire

Un ensemble fini $\mathcal{X}$ = objets / éléments / variables / ...

Instances de problèmes d'Optimisation Combinatoire

Un ensemble fini $\mathcal{X}$ = objets / éléments / variables / ...

Un ensemble de paramètres

- chaque objet peut avoir un poids, une durée, ...
- il peut y avoir des paramètres globaux :
  capacité des camions, nombre de processeurs, ...

Instances de problèmes d'Optimisation Combinatoire

Un ensemble fini $\mathcal{X}$ = objets / éléments / variables / ...

Un ensemble de paramètres
- chaque objet peut avoir un poids, une durée, ...
- il peut y avoir des paramètres globaux :
  capacité des camions, nombre de processeurs, ...

$\Rightarrow$ trouver un groupement / un ordre / une affectation / ...
des éléments de $\mathcal{X}$

Instances de problèmes d'Optimisation Combinatoire

Un ensemble fini $\mathcal{X}$ = objets / éléments / variables / ...

Un ensemble de paramètres
- chaque objet peut avoir un poids, une durée, ...
- il peut y avoir des paramètres globaux :
  capacité des camions, nombre de processeurs, ...

$\Rightarrow$ trouver un groupement / un ordre / une affectation / ...
des éléments de $\mathcal{X}$

$\mathcal{E}$ = l'espace de ces groupements / ordres / affectations / ...

$\mathcal{E}$ est très grand ! ! !

- $|\mathcal{E}| = p^{|\mathcal{X}|}$ si partition
- $|\mathcal{E}| = |\mathcal{X}|!$ si permutations

Instances de problèmes d'Optimisation Combinatoire

Un ensemble fini $\mathcal{X}$ = objets / éléments / variables / ...

Un ensemble de paramètres
- chaque objet peut avoir un poids, une durée, ...
- il peut y avoir des paramètres globaux :
  capacité des camions, nombre de processeurs, ...

$\Rightarrow$ trouver un groupement / un ordre / une affectation / ...
des éléments de $\mathcal{X}$

$\mathcal{E}$ = l'espace de ces groupements / ordres / affectations / ...

Temps pour énumérer $\mathcal{E}$ en fonction de $|\mathcal{X}|$ si on met $1\mu s$ par élément

$\lvert\mathcal{X}\rvert$	10	20	30	40	50
$\lvert\mathcal{E}\rvert = 2^{\lvert\mathcal{X}\rvert}$	1 ms	1 s	18 mn	13 jours	36 ans
$\lvert\mathcal{E}\rvert = 3^{\lvert\mathcal{X}\rvert}$	59 ms	58 mn	6.5 années	385 millénaires	...
$\lvert\mathcal{E}\rvert = \lvert\mathcal{X}\rvert!$	3.6 s	77 millénaires	...	...	...

Instances de problèmes d'Optimisation Combinatoire

On a aussi souvent des ***contraintes***

$\Rightarrow$ on parle de ***solution faisable*** :
groupement / ordre / affectation / ... de $\mathcal{E}$ qui satisfait les contraintes

$\mathcal{S}$ = l'ensemble des solutions faisables

(Si pas de contrainte : $\mathcal{S} = \mathcal{E}$.)

Instances de problèmes d'Optimisation Combinatoire

On a aussi souvent des ***contraintes***

$\Rightarrow$ on parle de ***solution faisable*** :
groupement / ordre / affectation / ... de $\mathcal{E}$ qui satisfait les contraintes

$\mathcal{S}$ = l'ensemble des solutions faisables

(Si pas de contrainte : $\mathcal{S} = \mathcal{E}$.)

Remarque

En général, $|\mathcal{S}| < |\mathcal{E}|$, et même $|\mathcal{S}| \ll |\mathcal{E}|$. Mais

- énumérer $\mathcal{S}$ reste souvent infaisable ;
- tester si $\mathcal{S} \neq \emptyset$ est même souvent difficile !

Instances de problèmes d'Optimisation Combinatoire

$v : \mathcal{E} \longrightarrow \mathbf{R}$: fonction de coût / valeur

⇒ On cherche une solution faisable S^* de valeur optimale :

$$S^* \in \underset{S \in \mathcal{S}}{\operatorname{argmax}}\{v(S)\} \quad \text{ou} \quad S^* \in \underset{S \in \mathcal{S}}{\operatorname{argmin}}\{v(S)\}$$

maximisation ↑ minimisation ↑

(On cherche un $S \in \mathcal{S}$ qui maximise / minimise l'«argument» $v(S)$.)

Instances de problèmes d'Optimisation Combinatoire

Une *instance* I d'un Pb. d'Optim. Comb. est donc caractérisée par

- ensemble d'objets / variables $\mathcal{X}$
- des paramètres

Instances de problèmes d'Optimisation Combinatoire

Une *instance* I d'un Pb. d'Optim. Comb. est donc caractérisée par

- ensemble d'objets / variables $\mathcal{X}$
- des paramètres

taille d'une instance I :
$|I|$ = taille de l'espace nécessaire pour stocker $\mathcal{X}$ et les paramètres

Instances de problèmes d'Optimisation Combinatoire

Une ***instance*** I d'un Pb. d'Optim. Comb. est donc caractérisée par

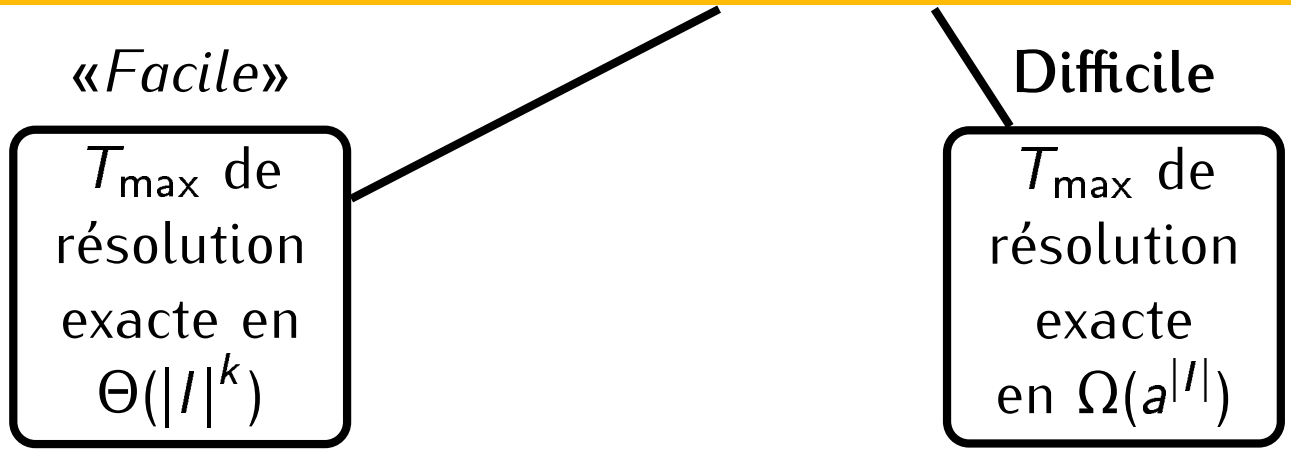
- ensemble d'objets / variables $\mathcal{X}$
- des paramètres

taille d'une instance I :
$|I|$ = taille de l'espace nécessaire pour stocker $\mathcal{X}$ et les paramètres

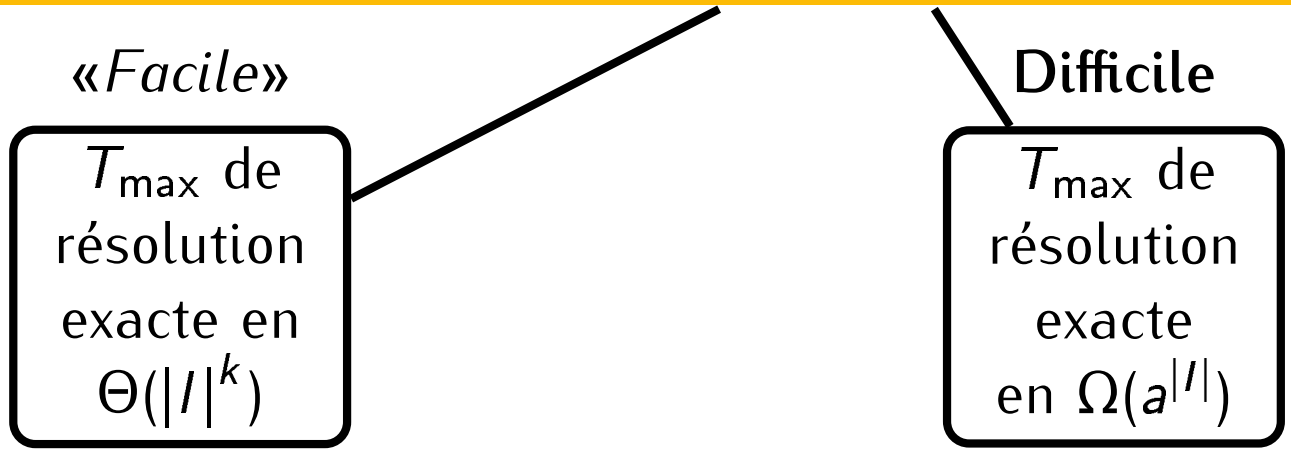
Un ***problème d'optimisation combinatoire*** est défini par :

- un ensemble d'instances $\mathcal{I}$
- pour chaque $I \in \mathcal{I}$: un ensemble $\mathcal{S}(I)$ de solutions faisables
- fonction de coût / valeur
  pour chaque I, chaque solution faisable S : $v(I, S) \in \mathbf{R}$

Optimisation Combinatoire : 2 catégories de problèmes

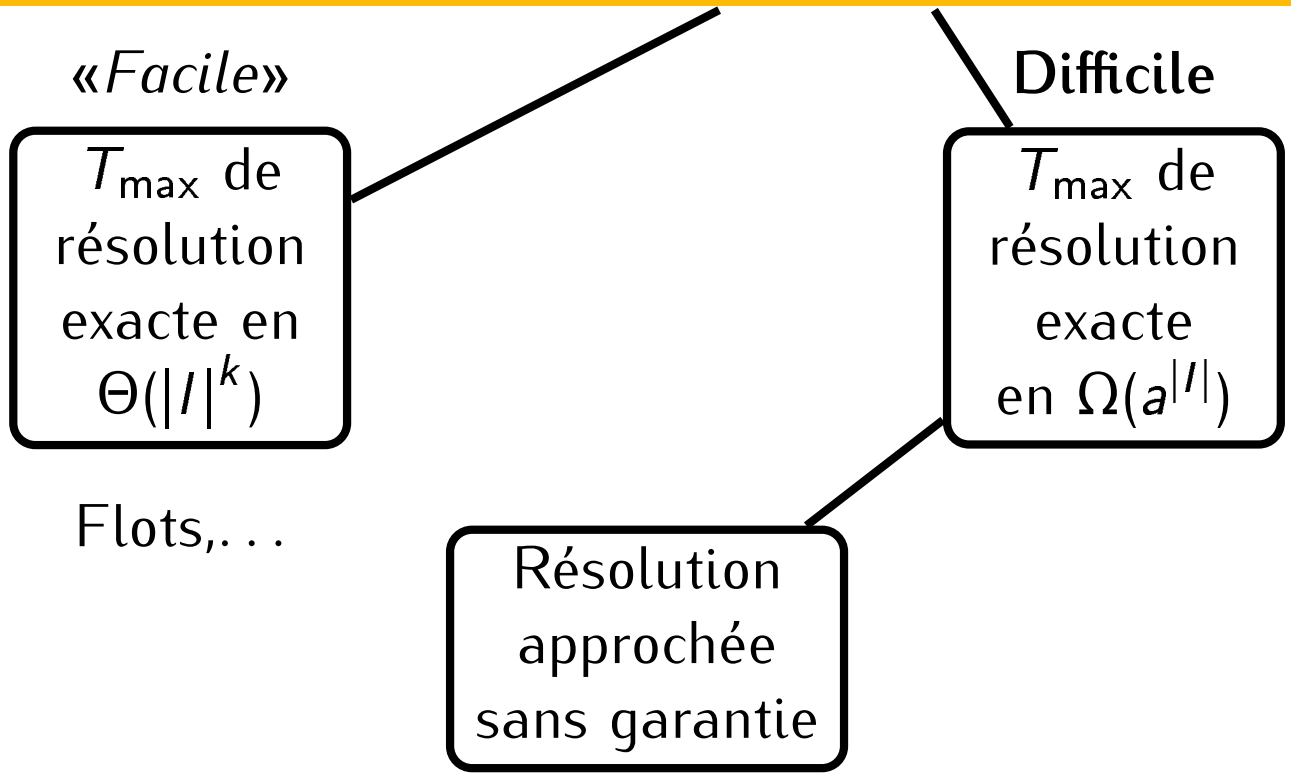


Optimisation Combinatoire : 2 catégories de problèmes

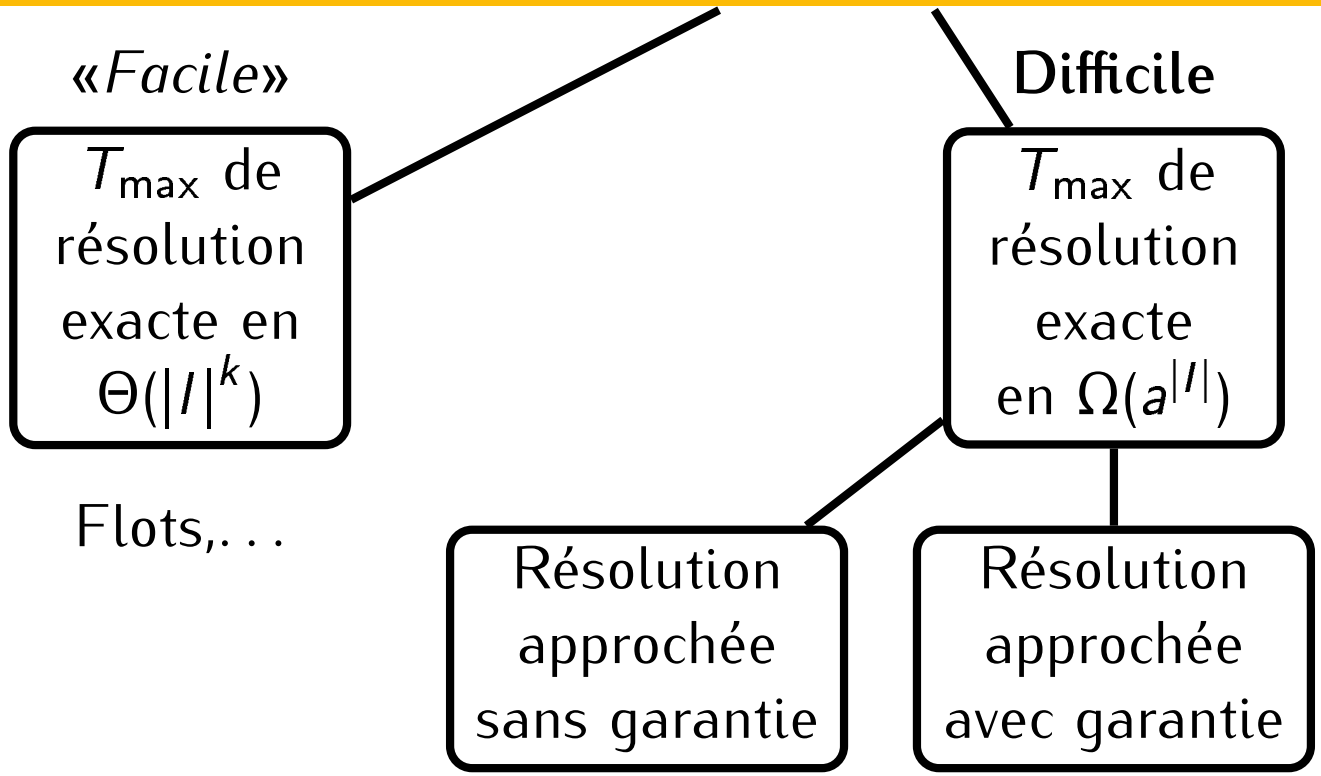


Flots,. . .

Optimisation Combinatoire : 2 catégories de problèmes



Optimisation Combinatoire : 2 catégories de problèmes



Optimisation Combinatoire : 2 catégories de problèmes