Optimisation en informatique

ESP (Ecole Supérieure Polytechnique) de Dakar

Alain Faye

1 – Introduction

Qu’est-ce un problème d’optimisation

- Recherche Opérationnelle
- Optimisation continue, discrète, mixte

Recherche Opérationnelle

Ensemble des méthodes et techniques scientifiques orientées
vers la **recherche** du meilleur choix dans les problèmes de décisions
se posant dans les grandes organisations publiques ou privées :
ordonnancement, gestion des ressources, gestion de la production, ….

Discipline incluant
mathématique, informatique, optimisation, optimisation discrète

Une étude RO passe par:

1. La modélisation

modélisation du problème réel par un modèle mathématique
Ceci demande souvent de faire des simplifications, des hypothèses

2. Recherche de solutions

mise en œuvre de méthodes numériques (calcul par ordinateur) pour résoudre le modèle

3. Confrontation des solutions à la réalité

Tests, retour éventuel au modèle pour le modifier, l'améliorer

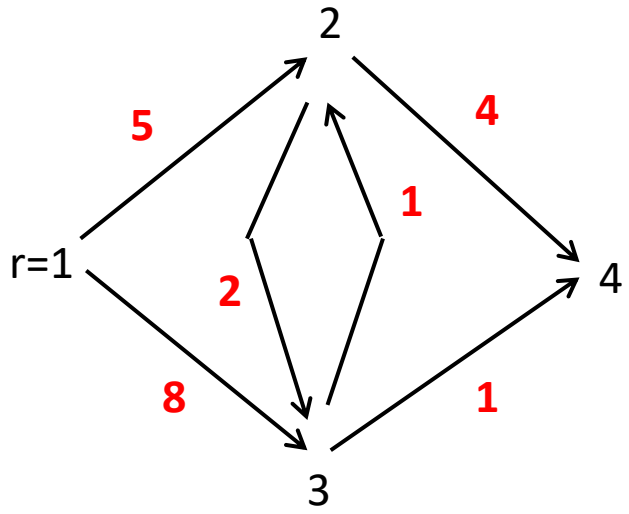
En RO les problèmes sont souvent discrets
Variables discrètes modélisant des entités indivisibles :
un nombre de voitures ou d'avions à construire par exemple

Optimisation combinatoire (Optimisation discrète)
- un objectif défini par une fonction objective
- trouver dans un ensemble discret (les solutions réalisables),
la *meilleure solution* relativement à la fonction objective.

Exemple 1 : Plus court chemin

- Problème du plus court chemin entre 2 nœuds d'un réseau représenté par un graphe dans lequel tout arc est valué par une longueur.
- Solution réalisable (admissible) : un chemin - ensemble d'arcs- qui relie les 2 nœuds.
- Longueur d'un chemin = somme des longueurs des arcs qui composent ce chemin.

Exemple1 plus courts chemins



Chemins de 1 à 4

Chemin 1, 2, 4	valeur 9
Chemin 1, 3, 4	valeur 9
Chemin 1, 3, 2, 4	valeur 13
Chemin 1, 2, 3, 4	valeur 8 ←optimal

Rappel : si les longueurs sont positives,
l'algorithme de **Dijkstra** permet de résoudre ce problème...

Exemple 2 : Achat de billets d'avion

Un homme d'affaires résidant à Paris (P) doit aller à Rome (R) le lundi et rentrer le mercredi pendant 5 semaines consécutives.

Le prix d'un billet aller-retour : 400 euros.

Réduction de 20% si au moins un week-end est inclus.

Aller simple : 65% du prix aller-retour.

Exemple 2 : Achat de billets d'avion

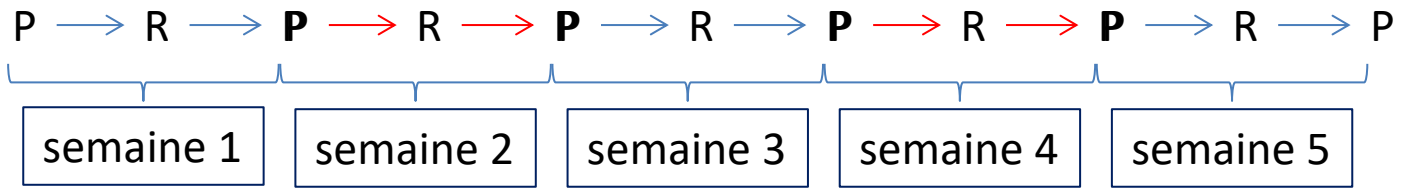
Un homme d'affaires résidant à Paris (P) doit aller à Rome (R) le lundi et rentrer le mercredi pendant 5 semaines consécutives.

Le prix d'un billet aller-retour : 400 euros.

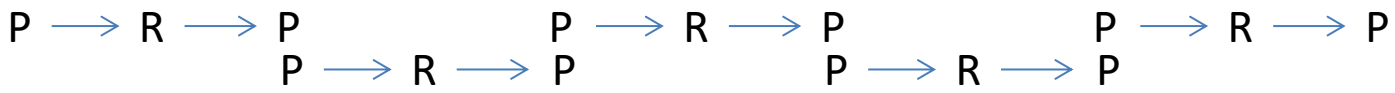
Réduction de 20% si au moins un week-end est inclus.

Aller simple : 65% du prix aller-retour.

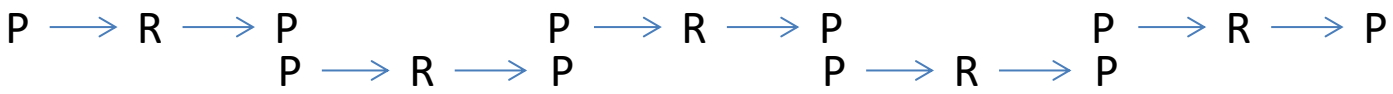
Il faut donc recouvrir les trajets P-R et R-P sur 5 semaines au moindre coût



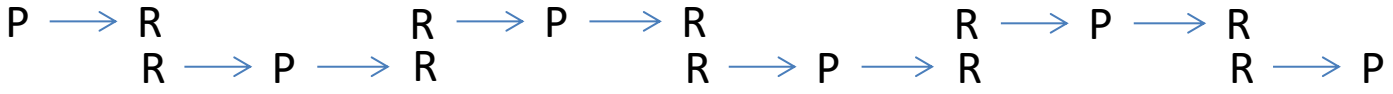
Sol.1 5 Aller-retour Paris-Rome coût = $5 \times 400 = 2000$ euros



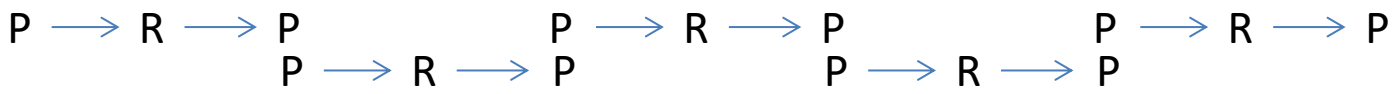
Sol.1 5 Aller-retour Paris-Rome coût = $5 \times 400 = 2000$ euros



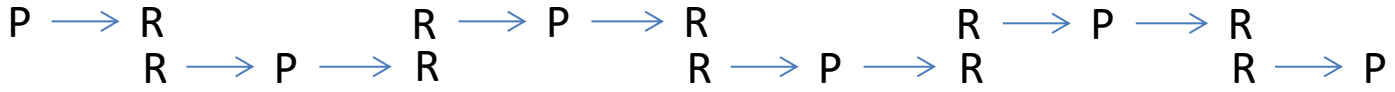
Sol.2 1 Aller Paris-Rome + 4 aller-retour Rome-Rome + 1 Aller Rome-Paris
Coût = $400(0{,}65+4 \times 0{,}8+0{,}65) = 1800$ euros



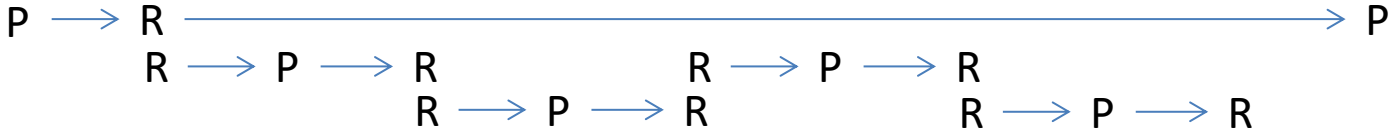
Sol.1 5 Aller-retour Paris-Rome coût = 5 × 400 = 2000 euros



Sol.2 1 Aller Paris-Rome + 4 aller-retour Rome-Rome + 1 Aller Rome-Paris
Coût = 400(0,65+4 × 0,8+0,65) = 1800 euros



Sol.3 1 Aller-retour Paris-Rome départ semaine1, retour semaine 5
+ 4 aller-retour Rome-Rome
Coût = 400(0,8+4 × 0,8) = 1600 euros



On ne peut pas faire mieux que la Solution 3. *Elle est* ***optimale***

Exemple 3 : Localisation de concentrateurs (notre exemple de base)

Données : terminaux (clients), sites potentiels pour concentrateurs, et

- distances (en km) sites-terminaux et coût d'un km de raccordement,
- coût d'installation d'un concentrateur sur un site.

Dist. (km)	S1	S2	S3	S4
T1	1	2	1	4
T2	6	1	6	3
T3	6	2	3	1
T4	3	3	7	10
T5	4	5	3	2

	S1	S2	S3	S4
Coût instal. (en Keuros)	110	200	100	100

Coût d'un km de raccordement : 15 Keuros

Exemple 3 : Localisation de concentrateurs (une première solution)

Dist. (km)	S1	S2	S3	S4
T1	1	2	1	4
T2	6	1	6	3
T3	6	2	3	1
T4	3	3	7	10
T5	4	5	3	2

	S1	S2	S3	S4
Coût instal. (en Keuros)	110	200	100	100

Coût d'un km de raccordement : 15 Keuros

Solution 1 : S1, S2 et S3 ouverts

On a intérêt à raccorder T1 à S1 (ou S3), T2 à S2, T3 à S2, T4 à S1 (ou S2) et T5 à S3

Coûts d'installation : 110+200+100 = 410

Coûts de raccordement : 15*(1+1+2+3+3) = 150

Coût total : 410+150=**560**

Exemple 3 : Localisation de concentrateurs (une deuxième solution)

Dist. (km)	S1	S2	S3	S4
T1	1	2	1	4
T2	6	1	6	3
T3	6	2	3	1
T4	3	3	7	10
T5	4	5	3	2

	S1	S2	S3	S4
Coût instal. (en Keuros)	110	200	100	100

Coût d'un km de raccordement : 15 Keuros

Solution 2 : S3 ouvert

==> On raccorde tous les clients à S3 !

Coût d'installation : 100

Coûts de raccordement : 15*(1+6+3+7+3) = 300

Coût total : 100+300=**400**

Résoudre un problème de R0

- Algorithme
- Modélisation et programmation mathématique

Trouver un algorithme

- Algorithme exact ou approché (heuristique)
- Complexité de l'algorithme
- Difficulté du problème à résoudre

Complexité des Algorithmes

Evaluer le temps d'exécution d'un algorithme

=

Donner un ordre de grandeur du nombre d'opérations élémentaires qu'il nécessite au pire des cas, exprimé en fonction de **la taille des données**

=

Calculer sa « complexité »

Complexité des Algorithmes

Notation (dite *de Landau*) :

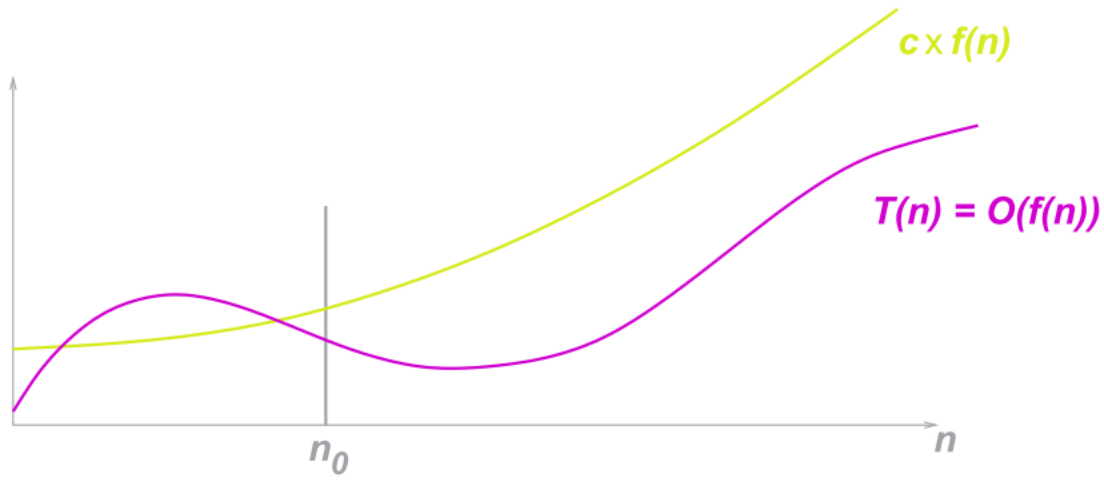
T(n)=*O*(f(n)) $\Leftrightarrow$ il existe c et n_0 tels que :

pour tout $n>n_0$, $T(n) \leq c*f(n)$

(En pratique, on a souvent $n_0=1$.)

Interprétation : en comportement asymptotique, T(n) ne croît pas plus vite que f(n)

Notation de Landau $O(f(n))$



Caractérise le comportement asymptotique (i.e. quand $n \to \infty$).

$$T(n) = O(f(n)) \text{ si } \exists c\ \exists n_0 \text{ tels que } \forall n > n_0,\ T(n) \leq c \times f(n)$$

Complexité des Algorithmes

Exemple : n est la taille du problème , un algorithme qui nécessite

$T(n)=5n^2+4n+3$ opérations au pire cas, est dit de complexité $O(n^2)$

(prendre $n_0=5, c=6$)

$T(n)=n^3+2n^2+4n+2$ complexité $O(n^3)$

(prendre $n_0=1, c=9$)

$T(n)=2^n+2n^2+n$ complexité $O(2^n)$

(prendre $n_0=1, c=4$)

Complexité des Algorithmes

Estimation du nombre d'opérations, étant donnés n et la complexité (à une constante multiplicative près, cachée dans la notation *O*()) :

n	**10**	**100**	**1000**
***O*(n)**	10	100	1000
***O*(n $\log_2$(n))**	33	664	9666
***O*(n^3)**	10^3	10^6	10^9
***O*(2^n)**	1024	$\approx 10^{30}$	$\approx 10^{300}$
***O*(n!)**	3628800	$\approx 10^{158}$	$\approx 10^{2567}$

polynomial

exponentiel

Complexité des Algorithmes

• Supposons qu'un algorithme de complexité n^2 soit capable de résoudre, en 1 heure, un problème de taille au maximum n=100 sur une machine M. Quelle est la nouvelle taille maximale n' résolue en 1 heure sur une machine M' deux fois plus rapide que M ?

• Même question avec un algorithme de complexité n^3

• Même question avec un algorithme de complexité 2^n

(Réponse : ≈ 141,126, puis 101)

Complexité des Algorithmes

- Les algorithmes polynomiaux bien plus efficaces que les exponentiels
- Est-il toujours possible de trouver un algorithme polynomial pour un problème ?

Complexité des problèmes

- Problème de décision: réponse par oui ou par non
- Problème d'optimisation

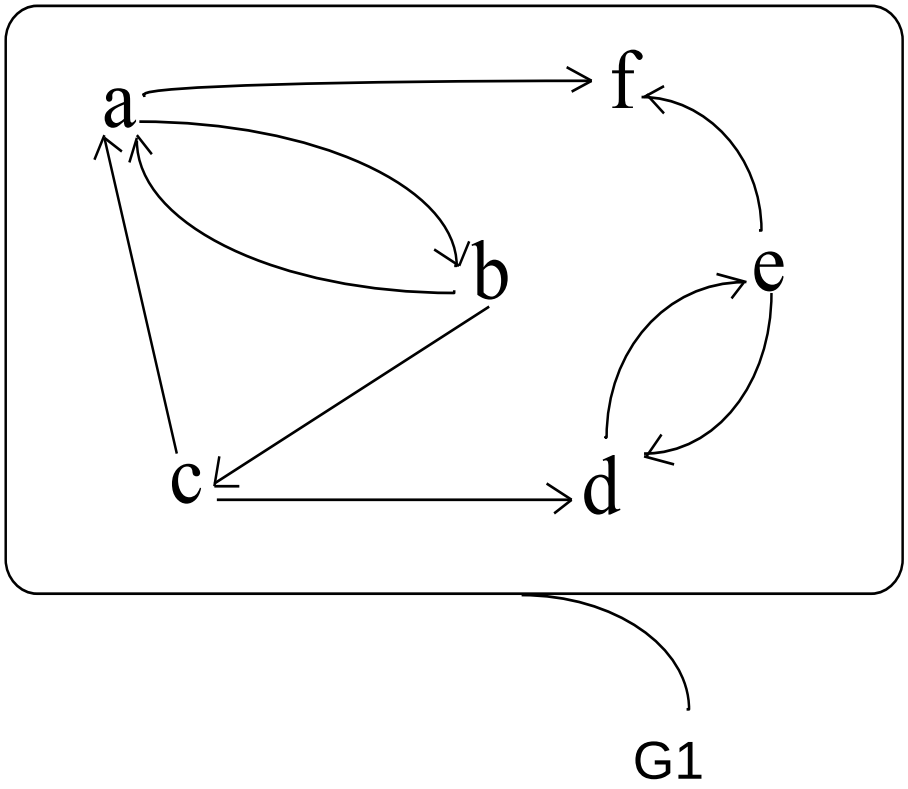
Classe *NP*

Un problème de décision est dans la classe *NP* si,
quand on sait que la réponse est OUI,
on peut facilement convaincre un tiers que c'est vrai.

(Mais on ne peut pas forcément trouver que la réponse est oui.)

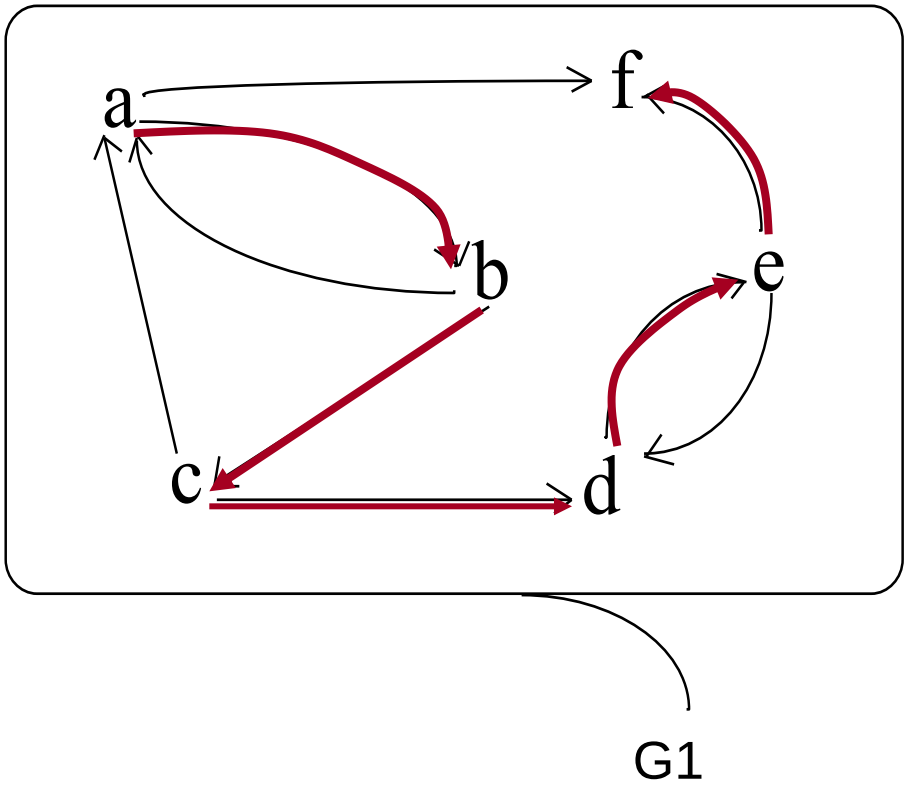
chemin hamiltonien :
chemin qui passe une fois et une seule par chaque sommet

Existe-t-il un chemin hamiltonien ?



chemin hamiltonien :
chemin qui passe une fois et une seule par chaque sommet

OUI
EXEMPLE (suite)
(a,b,c,d,e,f)



Exemple **si Carlos sait qu'il existe un cycle hamiltonien dans un graphe donné, il peut facilement vous en convaincre**

Mais si le graphe est grand, Carlos ne pourra pas savoir si ce cycle existe

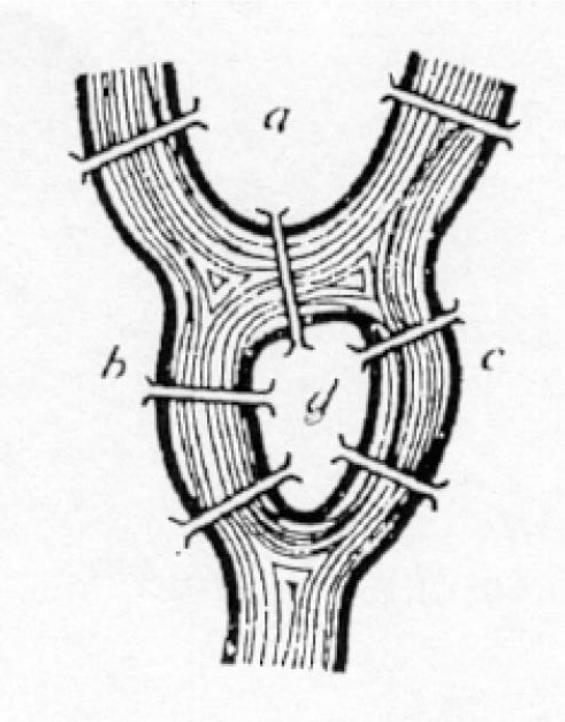
Classe *P*

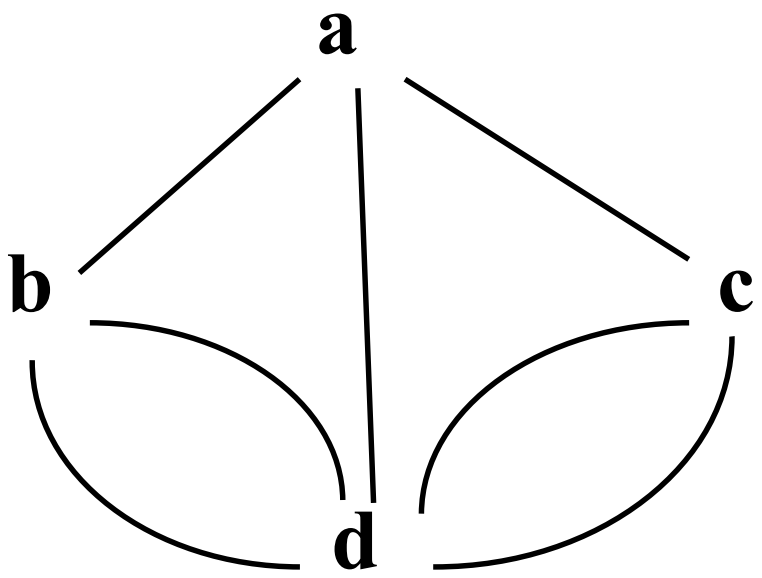
Un problème de *NP* est "facile" (polynomial) si on peut le résoudre par un algorithme "efficace" (temps polynomial en fonction de la taille de l'instance)

Exemple

L'existence d'un cycle eulérien dans un graphe

Euler (1736) : Peut-on se promener dans la ville en traversant chaque pont une et une seule fois ?





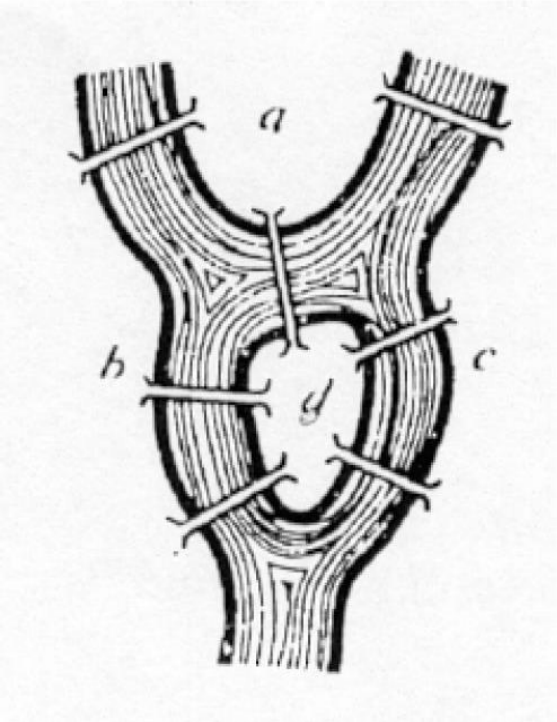
chaîne eulérienne : chaîne qui passe une fois et une seule par chaque arête

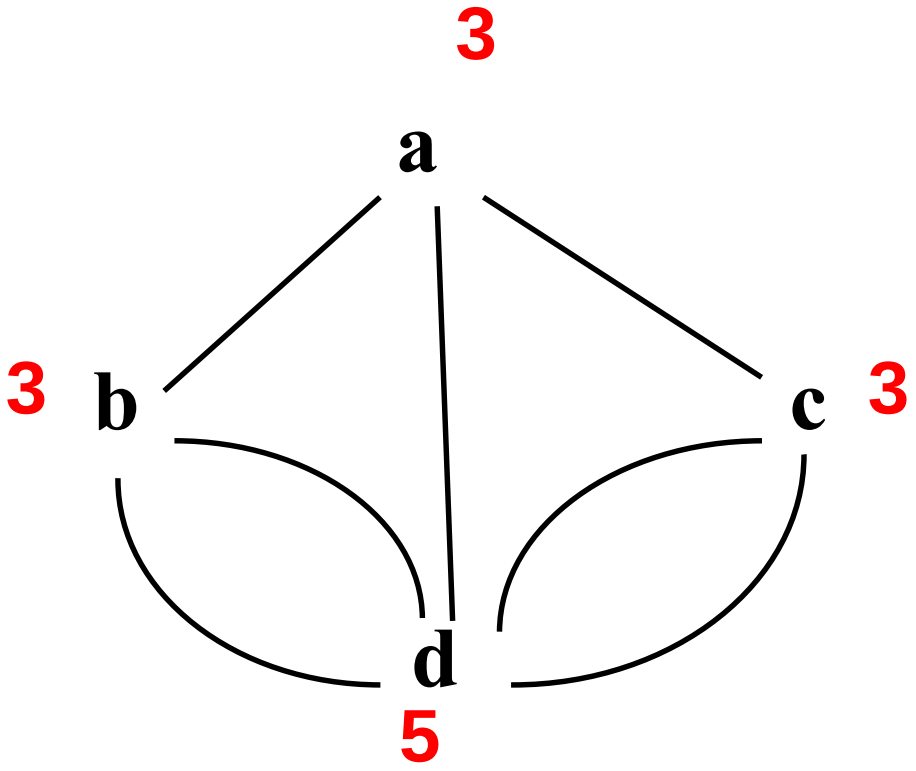
Théorème d'Euler

Un graphe connexe admet une chaîne eulérienne si et seulement si le nombre de sommets de degré impair est 0 ou 2

Peut-on se promener dans la ville en traversant chaque pont une et une seule fois ?

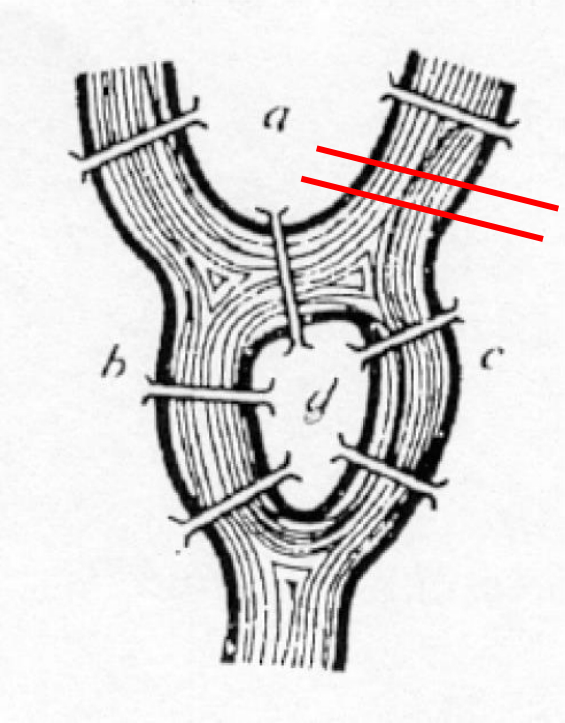
NON !

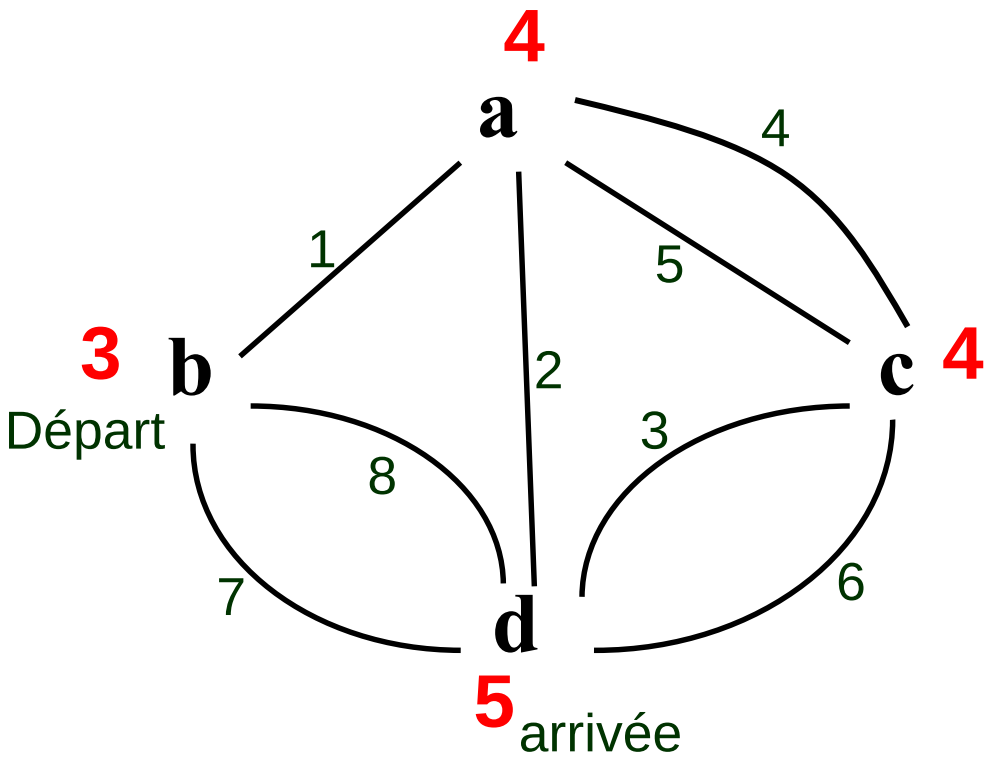




En ajoutant un pont:

OUI !



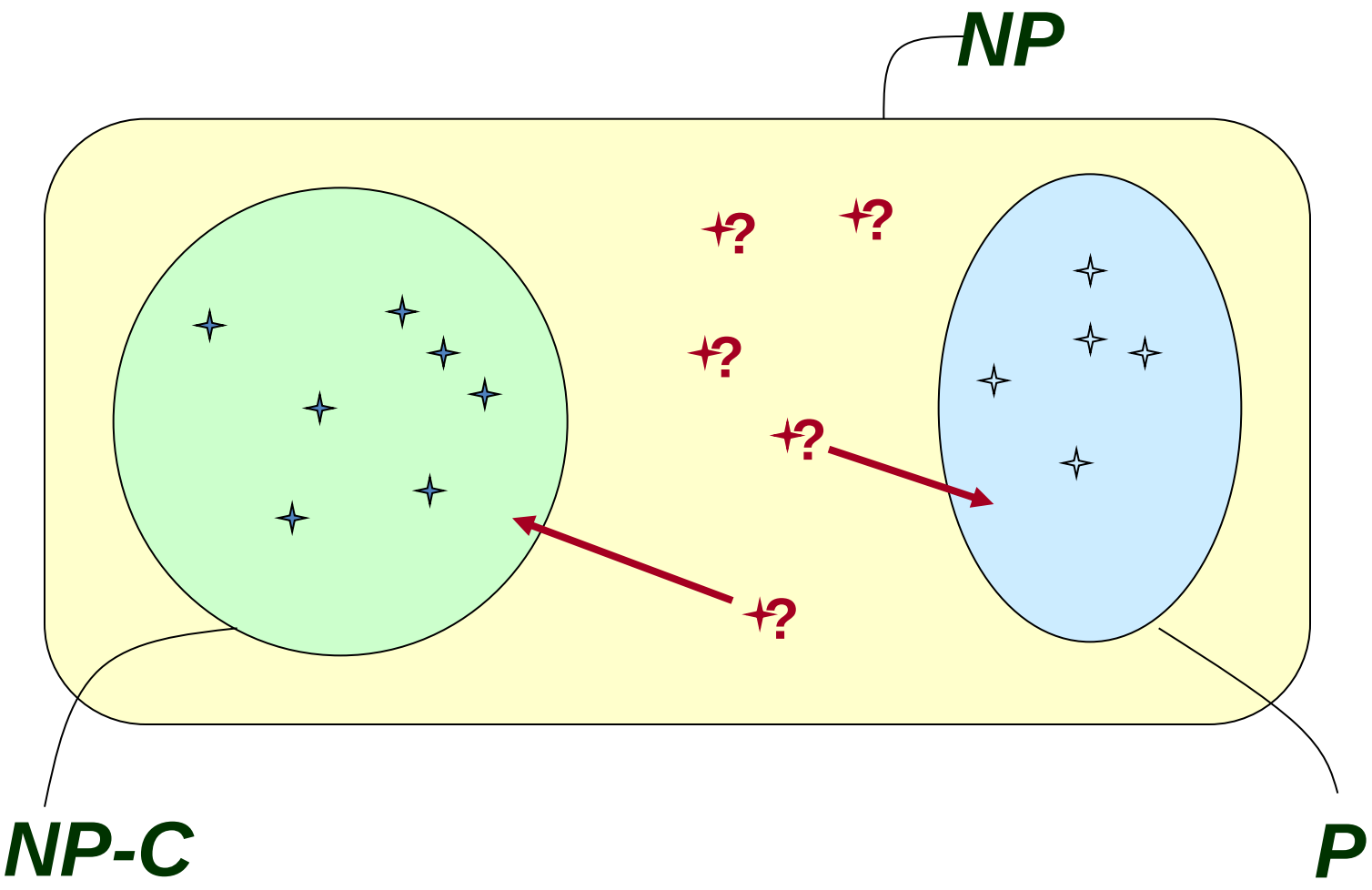


Un problème est "difficile" si les seules méthodes connues pour le résoudre exigent un temps de calcul exponentiel en fonction de la taille de l'instance

Exemple
L'existence d'un chemin hamiltonien dans un graphe

Classe *NP-C*

Un problème de *NP* est *NP*-complet si "savoir le résoudre efficacement" implique "savoir résoudre efficacement TOUS les problèmes de *NP*" .



Problèmes

? Problèmes non classés

Pour montrer qu'un problème Π est polynomial

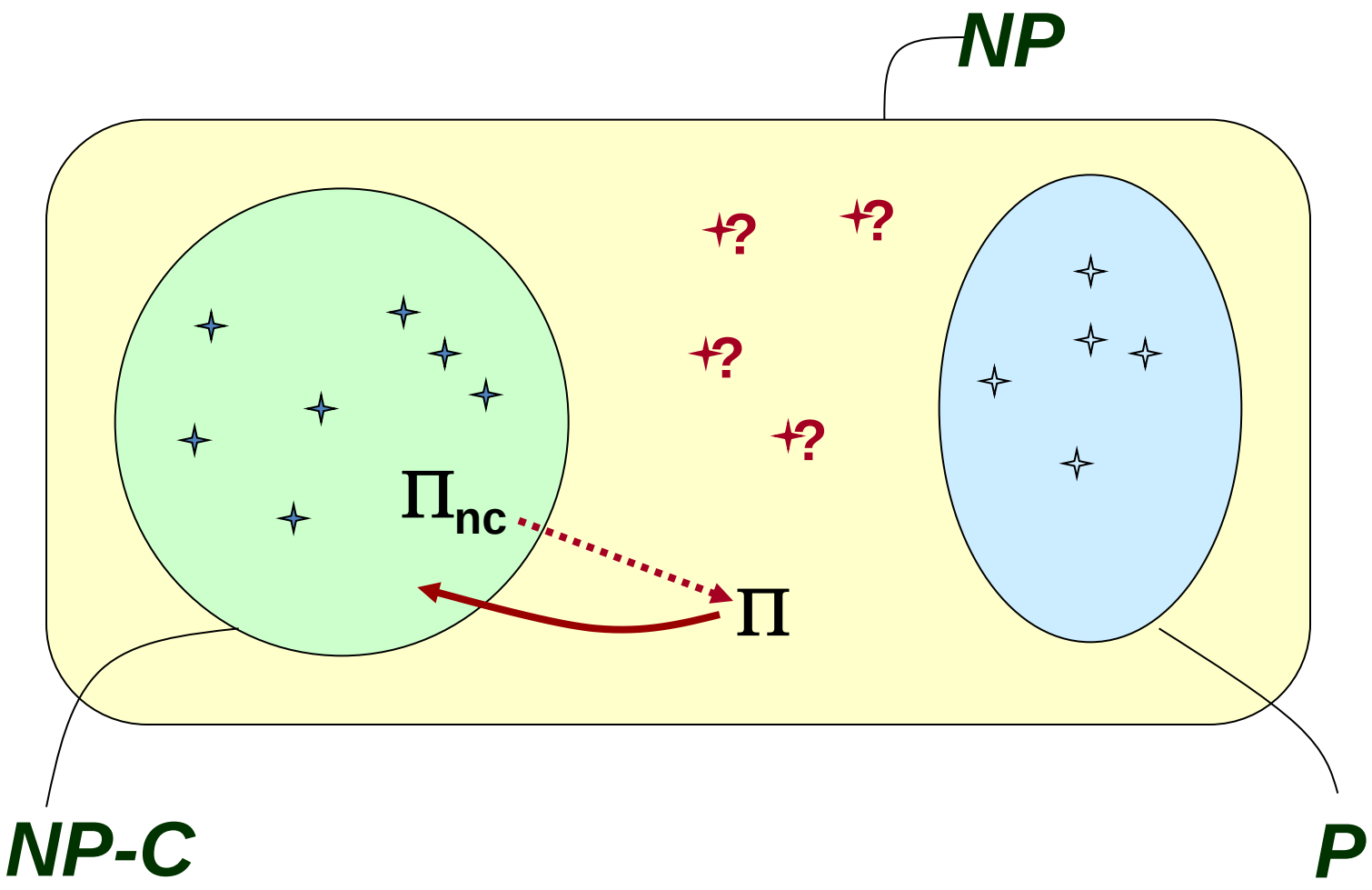
il faut trouver un algorithme pour le résoudre et prouver que cet algorithme s'exécute en un temps qui augmente de façon polynomiale en fonction de la taille de l'instance traitée

Pour montrer qu'un problème Π est *NP*-complet, on choisit un problème déjà connu pour être *NP*-complet, soit Π_{nc} , et on montre que Π_{nc} peut se "transformer" en Π.

Donc, si on savait résoudre Π, on saurait résoudre Π_{nc}.

Or, on ne sait pas résoudre Π_{nc} : donc il va sûrement être difficile de résoudre Π.

Π va, à son tour, être classé NP-complet.



Si on savait résoudre facilement Π on saurait résoudre aussi Π_{nc}; or on ne sait pas résoudre Π_{nc}

Π est donc sûrement difficile à résoudre

Les problèmes sont classés de façon incrémentale: la classe d'un nouveau problème est déduite de la classe d'un ancien problème.

L'établissement d'un "premier" problème *NP*-complet pour classer tous les autres s'est donc avéré nécessaire.

Le problème SAT

"satisfiabilité" d'une expression logique

Exemple

$(x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{y} \vee t) \wedge (y \vee \bar{z} \vee t) \wedge (x \vee z \vee \bar{t})$

x est vrai ou faux **x vrai $\Leftrightarrow$ $\bar{x}$ faux**

Peut-on affecter des valeurs vrai ou faux aux variables de telle façon que l'expression soit vraie ?

Exemple

une solution: x=vrai y=faux t=vrai z=vrai

Le théorème de Cook (1971)

Stephen Cook
a classé le
problème SAT
comme *NP*-complet

SAT est le premier problème
***NP*-complet connu**

Question ouverte

P* ≠ *NP

ou bien

P* = *NP

Pour prouver que *P* = *NP* il faudrait résoudre l'un des problèmes *NP*-complets avec un algorithme polynomial. Faire "tomber" un seul de ces problèmes dans la classe *P* ferait tomber l'ensemble de la classe *NP*

Problèmes d'optimisation

- **Problèmes d'optimisation**

Exemple : étant donné un graphe arc-valué, trouver un plus court chemin entre 2 sommets o et d.

étant donné un graphe sommet-valué, trouver un stable (sous-ensemble de sommets non reliés par une arête) de poids maximum.

Problème d'optimisation vers problème décision

• MAIS un problème d'optimisation peut toujours être transformé en un problème de décision (*version de décision d'un problème d'optimisation*) !

Exemple : étant donné un graphe arc-valué et un entier k, existe-t-il un chemin entre o et d de longueur ≤ k ?

étant donné un graphe sommet-valué et un entier k, existe-t-il un stable de poids ≥ k ?

Problèmes d'optimisation

On ne peut pas résoudre le problème de décision en temps polynomial

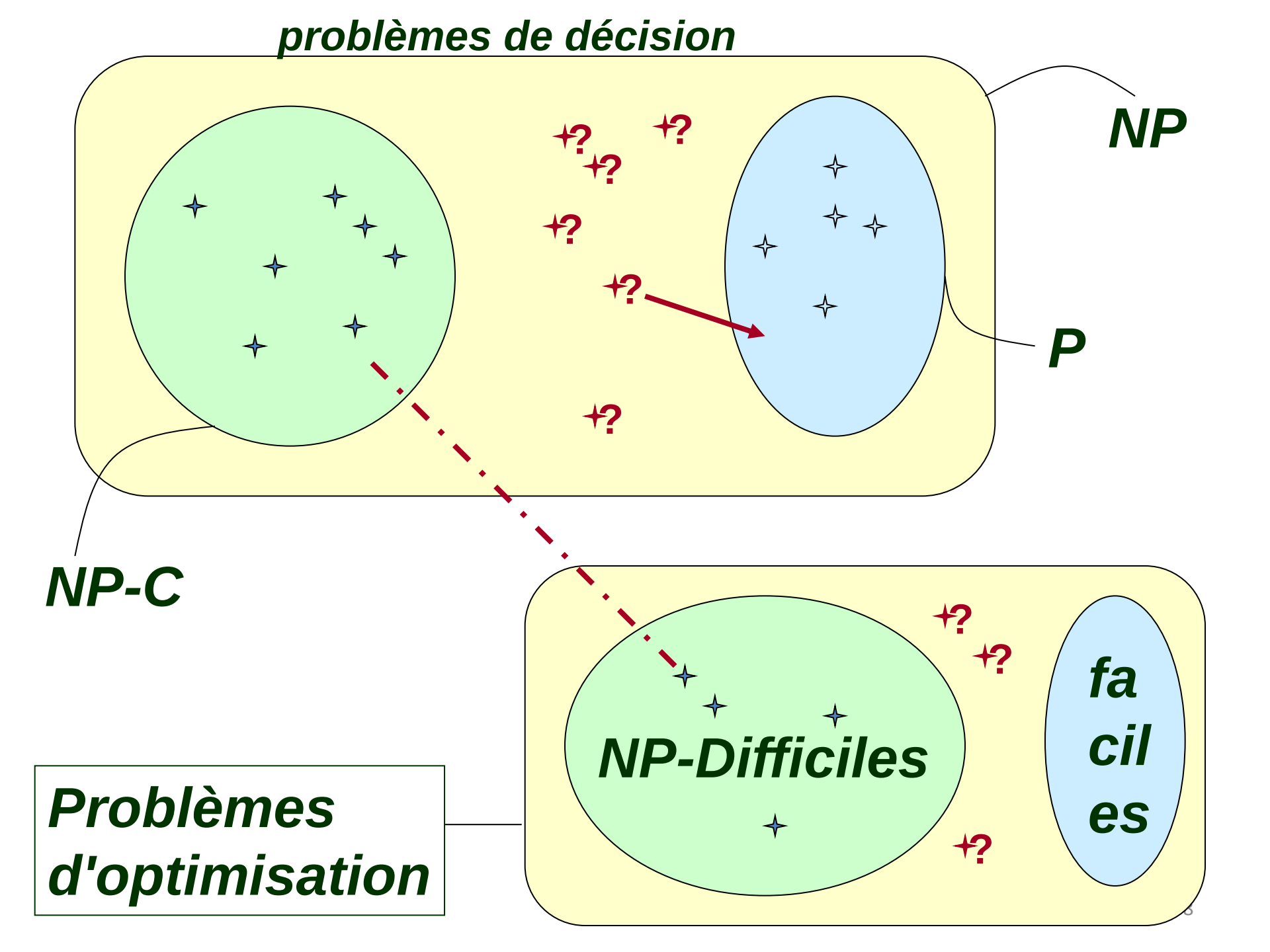
$\Rightarrow$

On ne peut pas résoudre le problème d'optimisation en temps polynomial

En effet, si on sait résoudre le problème d'optimisation en temps polynomial, il est facile de donner la réponse (oui ou non) au problème de décision

==> On peut se restreindre au problème de décision

pour étudier la difficulté d'un problème d'optimisation


problèmes de décision
NP
P
NP-C
NP-Difficiles
fa
cil
es
Problèmes
d'optimisation

Problèmes d'optimisation

Un problème d'optimisation est dit ***NP-difficile*** si le problème de décision associé est *NP-complet*.

Exercice: problème du stable maximum

Considérer le problème de décision noté P_{st}.
Montrer que le problème de décision est NP-complet. Pour cela, montrer que SAT se « réduit » à P_{st} .

Résoudre un problème classé difficile, est-ce possible ?

Si on sait qu'un problème donné est *NP-difficile*, que fait-on ?

1. Si on veut quand même une solution **optimale** (méthodes exactes) : elle sera coûteuse en temps de calcul, et probablement impossible à calculer à partir d'une certaine taille.
2. On peut chercher une solution avec une garantie a priori sur l'erreur (algorithmes d'approximation spécifiques), MAIS de telles méthodes n'existent pas pour tous les problèmes...
3. On peut chercher une solution admissible si elle existe, **sans garantie** sur la qualité de cette solution (méthodes heuristiques)...

Conclusion

- Complexité des problèmes
  - de décision: Polynomiaux , NP-complets
  - d'optimisation: Polynomiaux, NP-difficiles
- Si NP-difficile
  - résolution heuristique
  - résolution exacte