Master d'Informatique - Spécialité Androide
Module MAOA

Recherche Opérationnelle et Optimisation Combinatoire

Introduction

Pierre Fouilhoux

Sorbonne Université

2019-2020

Avant-propos

Cours de niveau dernière année d'étude (niveau Ingénieur / M2 Recherche)
sur la résolution exacte ou à garantie expérimentale
de problèmes d'Optimisation Combinatoire en Recherche Opérationnelle.

Face à un problème à traiter :
- tout d'abord effectuer une étude de complexité : peut-être existe-t-il un algorithme polynomial pour le traiter ?
- si le problème est NP-difficile, que faire ?

Une idée communément admise est qu'un problème NP-difficile ne peut être résolu exactement...
Ce cours va, au contraire, insister sur les méthodes permettant d'obtenir des solutions exactes ou à garantie expérimentale pour les problèmes NP-difficiles.

En effet, de nombreux problèmes engagent des sommes colossales ou mettent en place des infrastructures pour plusieurs décennies : il est important d'avoir une solution dont on peut prouver la qualité (**provably good solutions**).

Outils/Domaine

- les **solveurs entiers** (Cplex, Gurobi, Scip,...) qui ont une grande facilité d'utilisation pour la modélisation et une bonne capacité qui les place au premier plan des méthodes utilisées par les industriels : nous verrons leurs capacités et leurs limites en "ouvrant" le moteur de ces solveurs.
- le cadre de base de ces solveurs est le principe d'évaluation/branchement **Branch-and-Bound** qui nécessite d'étudier les techniques permettant d'obtenir une bonne évaluation du problème : souvent obtenues par méthode de **relaxation** linéaire, Lagrangienne, Dantzig-Wolfe,...
- le cadre ayant donné naissance à des solveurs suivant est celui de la **la programmation linéaire en nombres entiers** elle a permis ces dernières années de résoudre exactement des instances complexes et de grandes tailles de problèmes célèbres : problème du voyageurs de commerce, problème de coloration de graphes, tournées de véhicules... Nous étudieront les cadres pratiques et théoriques de cette approche.
- En particulier, nous insisterons sur la résolution de programmes à **nombres exponentiel de contraintes et de variables** : les algorithmes de coupes (Branch-and-Cut algorithm) et de génération de colonnes (Branch-and-Price).
- Nous verrons enfin comment sont renforcés ces formulations par l'ajout de contraintes qui se basent sur la structure combinatoire des problèmes d'Optimisation Combinatoire au travers l'**approche polyédrale**.

Plan du cours ROOC

Introduction

(Rappels) Résolution exacte ou à garantie expérimentale

A. PLNE compact et solveurs

(Rappels) Branchement et Evaluation (Branch-and-Bound)

B. Cas polynomiaux

C. Algorithmes de coupes (Branch-and-Cut)

D. Approches polyédrales

E. Caractérisation

(Hors-Programme) Décomposition et relaxation Lagrangienne

F. Génération de colonnes (Branch-and-Price)

Ouvrages

- "Pour avoir de bonnes bases"

C. H. Papadimitriou et Kenneth Steiglitz "Combinatorial Optimization : Algorithms and Complexity" Unabridged Edition, 1982.

M. Sakarovitch, "Optimisation combinatoire", Ed. Hermann, 1997 (non réédité).

- "Pour approfondir le cours"

"4 Bills book" : W. Cook, W. Cunningham, W. Pulleyblank et A. Schrijver ; Combinatorial Optimization, Wiley-Interscience (1997).

A.R. Mahjoub, "Approches Polyédrales en Optimisation Combinatoire", Optimisation combinatoire . 1 , concepts fondamentaux (2005) Hermes science publ. Lavoisier.

M. Minoux, "Programmation mathématique : théorie et algorithmes", Ed. Lavoisier, 2008 (1ère édition 1983).

L. Wolsey, Integer Programming, Wiley-Interscience, (1998).

Introduction

Qu'est-ce que la RO ?

Optimisation combinatoire et complexité : rappels sous forme de vulgarisation scientifique

Qu'est-ce que la RO ?
Définitions et historique
La RO en entreprise
La communauté scientifique RO
La pratique de la RO

Optimisation combinatoire et complexité : rappels sous forme de vulgarisation scientifique

Ce cours se base sur des problèmes d'Optimisation Combinatoire en Recherche Opérationnelle (RO).

Mais qu'est-ce que la RO ?

dans le monde académique ?
dans le monde industriel ?

Définitions et historique

La *Recherche Opérationnelle* (RO) :
ensemble des domaines scientifiques, des outils et des problèmes touchant aux questions d'ordre décisionnel (dit aussi stratégique) ou d'optimisation de systèmes complexes.

L'expression "systèmes complexes" est à prendre ici au sens le plus littéral du terme, c'est-à-dire difficile à comprendre pour un individu sans l'aide d'un modèle ou d'un ordinateur. Les problèmes sont rendus complexes par leur dimension qui peut être importante, mais surtout par leur structure qui peut-être par exemple combinatoire, concurrentielle, stochastique etc.

On peut citer en exemple pertinent : la recherche d'un itinéraire sur une carte, l'ordonnancement des tâches à accomplir en usine, la décision stratégique d'investissement d'une entreprise sur un marché concurrentiel,...

Historique rapide

Si l'on peut considérer que les problèmes de RO remontent aux "jeux mathématiques" des mathématiciens depuis le XVIème siècle (Blaise Pascal, Euler,...), l'utilisation de la RO dans la société remonte à la seconde guerre mondiale.

En effet, jusque là, les problèmes restaient gérés à un niveau humain, par des entreprises de petites tailles ou par des entreprises gérées selon des principes hiérarchisés qui simplifiaient les problèmes.

On fait traditionnellement remonter l'apparition de la RO à l'organisation, par l'armée rassemblée des Alliés, du débarquement en Normandie. A cette occasion, il fallait gérer au mieux l'implantation rapide de radars, d'acheminement de troupes, la gestion de leur alimentation, des contacts entre unité etc. L'armée alliée fit appel à des mathématiciens (et quelques premiers informaticiens) pour aider à ces décisions. Le mot "Opérationnel" viendrait donc du terme militaire.

En tout cas, le mot "Opérationnel" désigne plus sûrement l'aspect appliqué des décisions prises : il ne suffit pas de donner approximativement où placer des radars, il faut en indiquer avec précision le lieu et assurer du bon fonctionnement du système ! Les entreprises ont parfois ainsi une "division opérationnelle" ou "conseil stratégique" qui appuie les décisions de la direction.

Historique rapide

Si les premiers calculs numériques des années 40 ont été réalisés à la main (exemple l'algorithme du simplexe "déroulé à la main"), c'est bien entendu l'arrivée des ordinateurs qui permet à ce domaine d'avoir une véritable mise en pratique.

Les grands groupes industriels, souvent confrontés à la reconstruction d'après-guerre, emploient dans leur service de Rercherche et Développement (R&D) des chercheurs et des ingénieurs en RO. C'est le cas en France de la SNCF, d'EDF-GDF, du CNET (labo historique des télécoms français). C'est le cas partout dans le monde à diverses échelles.

Avec l'envolée industrielle et la généralisation de l'informatique, on peut constater un "creux" historique de la RO dans les années 70. C'est en fait à cette même époque que les chercheurs (souvent mathématiciens) inventent de nouveaux procédés de RO, s'appuyant sur la puissance des ordinateurs. La RO redevient "à la mode" après les premières crises économiques et écologiques des années 70-80. On peut constater même une réelle envolée depuis quelques années, renforcée encore par la crise économique actuelle (mieux gérer pour moins dépenser).

La RO en entreprise

La fédération européenne de RO s'amuse ainsi à définir la RO comme la

Science of better
Science de la bonne gestion
"Optimiser pour moins et mieux consommer"

Pour l'entreprise, la RO constitue un outil informatique pour aider à la gestion de l'entreprise : on parle alors de science de la gestion (Managment science ou Business managment tools).
La RO se retrouve donc sur une même ligne d'outils que les outils de comptabilité, de gestion de Base de données ou de gestion des systèmes d'information.
En fait, la RO utilise ces diverses sources de données pour aider aux décisions dans l'entreprise.

Au sein d'une entreprise

Les grands groupes industriels ont quasiment toujours un département RO. Il est la plupart du temps intégré au pôle R&D mais il est parfois intégré dans des pôles plus décisionnaires, comme les conseils de décisions stratégiques.

Les entreprises de transport ont évidemment un grand besoin de RO, pratiquemment toutes leurs décisions demandent des outils de RO. Par exemple, la Sncf possède une "culture RO" particulière créée et accumulée en interne depuis un siècle. On retrouve aussi cela dans l'armée, l'aviation civile etc. En revanche, certaines grosses entreprises ont jonglé avec leur service RO, le réduisant ou l'augmentant selon les périodes : cela s'explique par leurs réalités stratégiques souvent fluctuantes : par exemple, EDF n'a plus cherché à économiser l'énergie avec l'arrivée du nucléaire, même situation pour France-Télécom à l'arrivée de la fibre optique...

Certains départements RO en R&D d'entreprise sont parfois de véritables lieux de recherche académique, plus ou moins connectés aux réalités de l'entreprise. On ne peut que constater la fin progressive de ces départements au profit d'une gestion plus orientée. Ainsi, les départements RO deviennent des pourvoyeurs d'experts et de concepteur logiciel pour les autres départements de l'entreprises. Ils jouent ainsi le rôle de consultants internes, connaisseurs du "métier" de l'entreprise. Aussi, ils effectuent une réelle veille scientifique, permettant à l'entreprise d'avoir connaissance des nouvelles avancées techniques et de les acquérir rapidement si besoin.

Consultants RO

Avec la disparition progressive des services de recherche "pure" RO en entreprise, les chercheurs en R&D sont une interface entre les deux mondes industrielles et universitaires.
Les PME, parfois gestionnaires de problèmes complexes issus des nouvelles technologie par exemple, n'ont pas la dimension suffisante pour posséder leurs propres départements R&D ou RO. Elles font recours à des consultants extérieurs. Les grosses entreprises le font également (parfois même en dépit de la présence de compétences RO en leur sein).

Les consultants en RO sont demandés par les entreprises pour répondre à des besoins précis ou ponctuels. Ils sont parfois universitaires, ce qui est très répandu dans le monde anglo-saxon, mais bien plus rare en France.

Il existe de plus en plus de petites sociétés de service (SSII, consulting, conseil stratégique,...) qui proposent leurs compétences aux entreprises. Elles sont souvent spécialisés sur un type de problèmes (transport, gestion, planification, etc) ou basé sur un outil particulier (logiciel de résolution) : parmi elles en France Artelys, Eurodecision, Dynasys,...

Consultants RO

On peut les diviser en deux grandes catégories :

- les entreprises de décisions stratégiques qui louent les services d'un expert pour une décision précise ou une réorganisation d'un service : en général ce sont des personnes expérimentées qui sont ou ont été chercheurs.
- les entreprises de conception logicielle RO : elle propose de construire un outil informatique spécifique à l'entreprise, interfacé avec les systèmes d'information (SI) de l'entreprises. En général, cet outil demande un suivi sur le long terme qui permet à la PME d'avoir un contrat renouvelé régulièrement. La PME réutilise souvent le même noyau algorithmique d'un logiciel à l'autre.

Consultants RO

Parmi les enterprises spécialisées en RO, on peut mettre à part les entreprises qui proposent des logiciels très généraux pour la RO :

- d'une part, les entreprises qui ont une compétence scientifique élevée (souvent issus d'un travail universitaire). Il s'agit en général d'un (ou deux, rarement plus) algorithmes très compétitifs. C'est le cas de Cplex (PL, PLNE,...) qui a été racheté par ILOG puis par IBM ; de Xpress, racheté par Dash puis par..., etc. Ces entreprises, en général, ne vivent pas que de leurs produits logiciels, elles sont également une entreprise de consulting de très au niveau scientifique. C'est le cas aussi de LocalSolveur, outil basé sur les métaheuristique, créé initialement au sein de Bouygues et aujourd'hui géré par LocalSolver.
- d'autre part, certains produits logiciels vedettes contiennent, au sein d'une interface dédiée à l'entreprise, des outils de RO : c'est le cas des logiciels de SAP ou même d'Excel de Microsoft (qui contient un solveur PL).

La communauté scientifique RO

Une bonne part de la "communauté RO" est née au sein des départements R&D des grands groupes industriels. Ces chercheurs sont fréquemment devenus des universitaires, prenant des postes en informatique mais aussi en s'imposant comme une matière intégrée aux mathématiques, parfois même en économie ou en productique.

Depuis, il existe des équipes de RO dans ces 4 domaines universitaires, permettant ainsi une réelle richesse de cultures et d'approches. En fait, beaucoup d'universitaires déclarent généralement faire partie de la communauté RO mais également d'une autre communauté plus particulière en mathématique, informatique,... Par exemple, nous allons dans ce module aborder la RO sous l'angle de l'Optimisation Combinatoire qui est un domaine issu des mathématiques.

La RO s'exprime au travers de journaux scientifiques spécifiques : Operations Research, The journal of the Operational research society (royaume-uni), 4OR (France, Belgique, Italie), Rairo, European Journal of OR, Mathematics of OR, annals of OR, Computers and OR,... Et des journaux appartenant à des domaines scientifiques que l'on peut inclure dans l'OR : Mathematical Programming, Discrete Mathematics, Computational Optimization and Algorithm,... Et des journaux très spécialisés sur des domaines particuliers de la RO : Tranportation, Networks, Journal of Scheduling,...

La communauté scientifique RO

La RO se réunit au sein de fédération de la RO (société philantropique) : en France ROADEF (on en retrouve dans chaque pays : italie, UK, USA, Japon,...). Les fédérations européennes sont regroupés dans la fédération de fédération EURO. Idem au niveau international avec IFORS.

Il existe également des conférences sur les mêmes thèmes que les journaux (parfois avec le même nom) ou que les fédérations de RO. La conférence EURO rassemble jusqu'à plusieurs milliers participants ! C'est l'occasion d'écouter en un même lieu les chercheurs et les ingénieurs de l'industrie, des entreprises publiques, de l'armée et de l'université. Il existe aussi des conférences de RO spécialisées sur des thèmes précis : par ex Télécom : AlgoTel, Inoc.

En France, la RO est ainsi coordonnée par
- la ROADEF www.roadef.org
- un groupe de recherche du CNRS le GDR-RO gdro.lip6.fr

Faire ses études en RO

Les domaines qui sont à la base de la RO (algorithmique, théorie des graphes, optimisation continue,..) sont enseignés dans la plupart des formations d'informatique et de mathématiques appliquées.

La Recherche Opérationnelle en tant que matière apparaît dans des formations diverses : en formations universitaires en informatique et math-infos bien sûr mais aussi en licence de sciences économiques ou en écoles de commerce comme HEC ou l'ESSEC. Peu à peu la RO devient une matière clef des écoles d'ingénieurs où ce nom prend le pas sur les filières "Optimisation".

Docteur en RO ?
De nombreuses entreprises proposent chaque année des stages et de thèses CIFRE en RO : ces propositions s'ajoutent à l'offre académique plus classique. Ces stages et thèses proposent ainsi une formation par la pratique et par la recherche à cette matière aux nombreux aspects. Si tous les docteurs en RO ne deviennent pas chercheurs à l'université, le diplôme de docteur en RO est de plus en plus la porte d'entrée vers la R&D en RO dans les entreprises de consulting ou les grands groupes industriels.

Le "cadre" de la RO

On considère habituellement que la RO est finalement un ensemble de domaine scientifique, de problèmes classiques et d'outils.

Le cadre des problèmes de RO est assez large : il concerne tous les problèmes de décision et d'optimisation... Il est plus facile de dire ce qui ne relève pas de la RO :

- quand un problème est de structure simple mais de très grande taille : cela peut plutôt demander un outil de gestion de base de données ou de SI.
- quand le problème n'a pas de solutions évidentes, qu'on peut se demander s'il existe une solution, si la question exige des équations complexes : ce sont certainement des mathématiques
- si le problème demande une simplification de part sa nature physique ou biologique : il s'agit plutôt d'utiliser des compétences dans ces domaines,
- si le problème semble difficile mais ne concerne que des instances très très réduites, quasiment solvables à la main... il n'y a peut-être pas besoin de développer un modèle ou des outils (exemple un TSP à 3 villes...).

Les domaines scientifiques

Il est difficile de classer ou de lister exhaustivement tous les domaines de la RO. D'autant que ces domaines ont tous une réalité par eux-mêmes et on peut se demander quelle partie de ces domaines est dans la RO ou pas.

• Aspect "continu" :
Optimisation continue, PL,
Théorie des jeux continues,...
Optimisation stochastique
Programmation semi-définie
Simulation continue

• Aspect "discret" ou "combinatoire" :
Théorie des graphes
Algorithmique, algorithmique des graphes
Programmation dynamique
Optimisation combinatoire exacte ou à garanti d'approximation
Optimisation combinatoire heuristique
Processus markovien, file d'attente,...
Simulation discrète
Ordonnancement
PLNE, approches polyédrales
Programmation quadratique
Programmation par contraintes
Théorie des jeux algorithmique

• Plus général :
Complexité des problèmes, Décidabilité des problèmes,...

Les outils

Il existe de nombreux outils logiciels disponibles commercialement ou libres pour résoudre des problèmes précis ou des problèmes plus généraux.

• Logiciels d'optimisation (interactif ou modeleur)
Solveur linéaire (cplex, LP de Coin-OR, xpress, glpk, gurobi, excel...)
Solveur entier (cplex, xpress, glpk, gurobi, scip, Coin-Or,...)
Solveur quadratique (cplex, xpress)
Solveur continu, solveur pour programme semi-défini,...

• Logiciel de simulation
Système de file d'attente (qnap, simula,...)
Système continu

• Logiciel d'aide à la décision :
Outils à interface graphique de planification (SAP, produit ilog)...
Outils d'analyse des problèmes (probas)...

• Des API (souvent C++ ou java) d'algorithmes et de Structures de données
Algo de graphes : boost, lemon,...
Des frameworks permettant d'utiliser les solveurs dans un processurs algorithmiques plus complexes. Par exemple les méthodes de coupes ou de générations de colonnes : Scip, Concert technology (d'ILOG),...

Les grandes problématiques de la RO

En plus des domaines scientifiques qui sont plutôt théoriques, la RO se subdivise traditionnellement en problématiques qui se sont peu à peu constitués en véritables "problèmes". On peut ainsi considérer qu'il existe de véritables communautés scientifiques qui, sans avoir pourtant un bagage théorique commun, se retrouvent pour parler d'applications et, parfois, mettre en concurrence leurs outils algorithmiques pour résoudre les mêmes instances d'un même problème.

On peut ainsi considérer :
- network design (conception de réseaux de télécommunication)
- transport
- localisation (plant location)
- micro-électronique (VLSI design)
- bio-informatique, applications médicales
- productique (job-shop, flow-shop,...)
- ordonnancement (domaine à la limite problématique/problème...)
- optimisation financière
- chaîne logistique (supply chain) et création de lots (lot-sizing)
- optimisation "durable"
- ... et de nouvelles problématiques se créent régulièrement.

La pratique de la RO

Le spécialiste de RO "découvre" le problème de RO et doit y répondre en temps limité. L'universitaire découvre généralement des problèmes plus difficiles en ayant davantage de temps pour trouver des réponses. On peut considérer qu'il existe une démarche pour approcher et résoudre un problème de RO, démarche qui dépend fortement du temps disponible, mais qui globalement suit le schéma suivant.

• *Découvrir le problème*
En général, la découverte du sujet passe par des discussions orales, souvent contradictoires avec les différents intervenants. Il s'agit de cerner la/les questions au cours de réunion et de discussion autour d'un texte qui devient vite un "cahier des charges" du projet.
Il est important de cerner le projet dans sa globalité : certains problèmes n'existent que parce que les gens se refusent à changer le cadre du problème. Inversement, un problème peut être posé en demandant explicitement de ne pas toucher à ce qui n'est pas dans le cahier des charges... Il faut insister là-dessus : par exemple : impossibilité de recruter de nouveaux collaborateurs, d'acheter un véhicule supplémentaire etc.
L'autre aspect important est la collecte des données. Il est parfois impossible d'obtenir les données réelles du problème alors qu'elles conditionnent tout le problème ! En effet, si la dimension du problème est petite... ou si les données sont tellement correllées qu'elles simplifient le problème etc. On parle alors des degrés de liberté de l'énonce qui sont plus ou moins fermées.

La pratique de la RO

• *Modéliser*
L'étape de modélisation est primordiale. Elle nécessite d'avoir pris le recul nécessaire sur le sujet pour choisir l'outil de modélisation : UML, graphe, réseau de pétri, PL/PLNE, SDP, ordonnancement,... Attention, on utilise très souvent un modèle trop compliqué, ce qui revient à tenter d'utiliser un marteau-piqueur pour ouvrir une noix ! Or l'outil utilisé, même s'il donne un modèle valide, peut-être très lourd à manier (temps de calcul,...). On peut même commettre l'erreur d'utiliser un outil exponentiel pour résoudre un problème polynomial !
Souvent, lors de la conception d'un modèle, on doit utiliser des hypothèses simplificatrices (linéariser, négliger des paramètres, etc). Il est nécessaire de bien les indiquer dans le cahier des charges.

• *Valider le modèle*
Il est indispensable de "boucler" la démarche de modélisation en validant le modèle, les hypothèses, les dimensions et les données. C'est une étape parfois difficile, voire impossible si l'interlocuteur se refuse à signer le cahier des charges ou fait preuve de mauvaise foi etc.

La pratique de la RO

• *Evaluer la complexité et l'état de l'art*
Etape évidemment nécessaire, il faut étudier la complexité du problème obtenu après modélisation. C'est le moment de se rendre compte si le problème traité est difficile ou pas, si un outil classique est utilisable etc.
Cette étude se couple avec un état de l'art plus ou moins exhaustif des travaux déjà faits sur ce problème ou des problèmes proches. C'est l'occasion de découvrir comment a été traité ce problème etc.

• *Choisir d'un outil (s'il existe)*
Le choix de l'outil signifie à la fois choisir un algorithme théorique mais aussi rechercher s'il existe un outil informatique disponible : solveurs, architectures objets d'algorithmes, API d'algorithmes,...

• *Etude scientifique et algorithmique*
Sans commentaire.

La pratique de la RO

• *Expérimentation et retour d'expérience*
La phase de tests expérimentaux est souvent négligée car elle arrive toute à la fin de l'étude. Pour le partenaire, elle est pourtant essentielle. Elle gagne à être entammée dès le début de l'étude, par exemple en construisant une benchmark complète et validée d'instance permettant de bien évaluer les performances de l'outil produit.
Mener sa campagne d'expérimentations demandent d'avoir suffisamment d'instances de types divers et de tailles croissantes. Il faut dégager les performances, mais aussi les défauts et les limitations.
Il est aussi utile de mettre en évidence les caractéristiques fines des réactions de l'outil : on peut ainsi dégager des cas faciles (où le problème est parfois polynomial), indiquer la réaction de l'algorithme dans le cas où les données sortent du cadre initial (on parle de robustesse) ou dans le cas où les données sont non vérifiées (on parle de fiabilité).

La pratique de la RO

• *Intégration et vie logicielle*
Etape nécessaire pour l'entreprise, il faut intégrer l'outil dans le cadre des outils de l'entreprise. L'interfaçage avec les SI de l'entreprise n'est pas une chose simple, ni d'ailleurs l'obligation fréquente de permette une certaine ergonomie de l'outil qui ne s'adresse pas à un public initié à la RO ou au math (pas de $x_7 = 10...$) : prévoir un manuel d'utilisateur, des formations etc.
D'autre part, il faut souvent prévoir que l'outil aura une vie après votre expertise. Cela consiste à prévoir un manuel du concepteur pour permettre de reprendre votre code. Cela consiste aussi à garder en main les compétences nécessaires pour faire cette évolution (collaborateurs formés etc).

• *Plannification des projets de RO*
Un problème crucial pour les entreprises de conseils en RO : l'évaluation préalable du temps qui doit être consacré à la démarche globale d'analyse et de résolution du problème. Si ce temps est imposé à l'avance, tout est conditionné par cela. Par exemple, certains consultants ne se donnent que quelques jours : cela empêche pratiquement tout développement théorique ou pratique d'algorithmes et pousse à l'utilisation systématique de solveurs.

Conclusion

Sans sectarisme, la RO est un domaine transversal entre recherche et ingénieurie, entre maths et informatique, entre théorie pure et applications.

On dit qu'elle traite les problèmes de la cité (de la société), ce qui la montre utile à tous.

En effet, il est toujours intéressant d'avoir un système mieux géré, plus économe sur les ressources, plus écologique,... tout en respectant les contraintes naturelles du droit du travail et le respect des conditions de travail.

Qu'est-ce que la RO ?

Optimisation combinatoire et complexité : rappels sous forme de vulgarisation scientifique

- Le problème du sac-à-dos
- Le problème du stable
- Un problème combinatoire industriel
- L'optimisation combinatoire
- L'explosion combinatoire
- Cas polynomiaux et $\mathcal{NP}$-difficulté

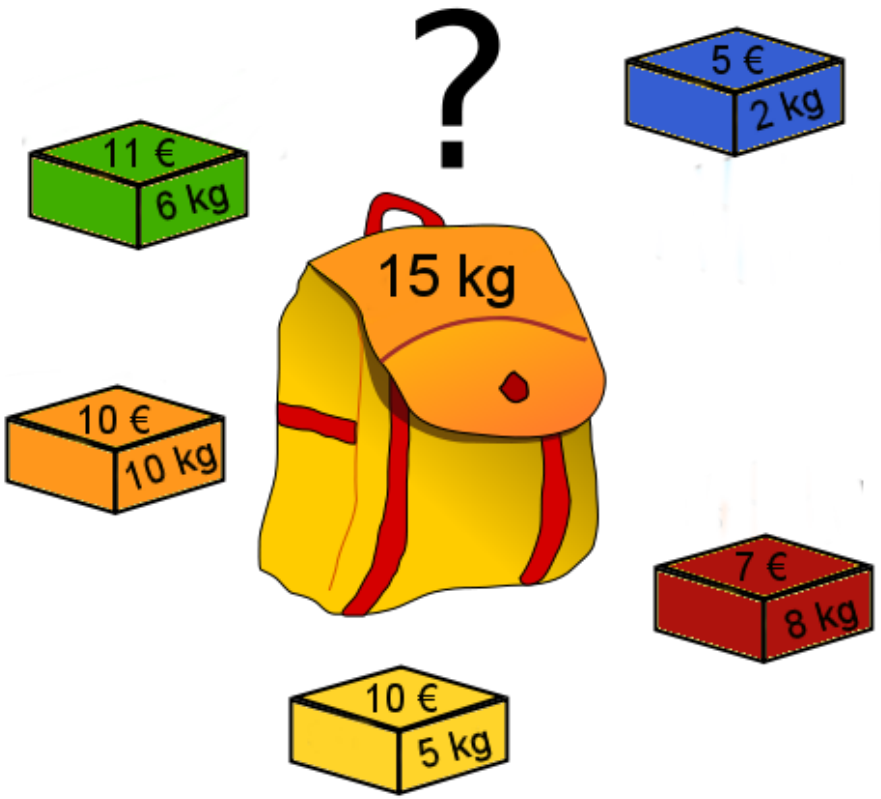
Un **problème d'optimisation combinatoire**

consiste à

trouver un plus petit (ou plus grand) élément
dans un ensemble fini valué.

Le problème du sac-à-dos

Quelles boîtes choisir
en maximisant le gain
sans dépasser 15kg ?



Définition

On considère n **objets**.
Chaque objet $i \in \{1, \ldots, n\}$
- de **gain** g_i
- de **poids** p_i

à ranger dans un sac-à-dos de **poids maximum** P.

Définition

On considère n **objets**.
Chaque objet $i \in \{1, \ldots, n\}$
- de **gain** g_i
- de **poids** p_i

à ranger dans un sac-à-dos de **poids maximum** P.

On veut trouver un sous-ensemble $S \subset \{1, \ldots, n\}$ tel que

Définition

On considère n **objets**.
Chaque objet $i \in \{1, \ldots, n\}$
- de **gain** g_i
- de **poids** p_i

à ranger dans un sac-à-dos de **poids maximum** P.

On veut trouver un sous-ensemble $S \subset \{1, \ldots, n\}$ tel que

son poids $\sum_{i \in S} p_i$ soit inférieur à P *(Contrainte de sac-à-dos)*

Définition

On considère n **objets**.
Chaque objet $i \in \{1, \ldots, n\}$
- de **gain** g_i
- de **poids** p_i

à ranger dans un sac-à-dos de **poids maximum** P.

On veut trouver un sous-ensemble $S \subset \{1, \ldots, n\}$ tel que

son poids $\sum_{i \in S} p_i$ soit inférieur à P *(Contrainte de sac-à-dos)*

son gain $\sum_{i \in S} g_i$ soit maximum. *(Fonction "objective")*

Comment représenter une solution du problème ?

Instance donnée par :

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Une solution $S \subset \{1, \ldots, n\}$ correspond à un **vecteur d'incidence** χ^S tel que $\chi^S[i] = \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{sinon.} \end{cases}$

$S_1 = \{1, 2, 5\}$

$\chi^{S_1} =$

1	1	0	0	1

Comment représenter une solution du problème ?

Instance donnée par :

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Une solution $S \subset \{1, \ldots, n\}$ correspond à un **vecteur d'incidence** χ^S tel que $\chi^S[i] = \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{sinon.} \end{cases}$

$S_1 = \{1, 2, 5\}$

$\chi^{S_1} =$

1	1	0	0	1

coût :	5	7		10	$= 22$
poids :	2	8		5	$= 15$

Comment reconnaître si un vecteur est solution ?

Comment reconnaître si un vecteur est solution ?

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Algorithme de reconnaissance :

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

$$S_2 = \{1, 4, 5\}$$

$$\chi^{S_2} = \begin{array}{|c|c|c|c|c|} \hline 1 & 0 & 0 & 1 & 1 \\ \hline \end{array}$$

Comment reconnaître si un vecteur est solution ?

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Algorithme de reconnaissance :

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

$$S_2 = \{1, 4, 5\}$$

$$\chi^{S_2} = \boxed{1}\ \boxed{0}\ \boxed{0}\ \boxed{1}\ \boxed{1}$$

$$\text{poids :} \quad 2 \qquad\qquad 6 \quad 5 \quad = 13$$

Solution de coût 26

Comment reconnaître si un vecteur est solution ?

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Algorithme de reconnaissance :

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

$$S_3 = \{1, 2, 4\}$$

$$\chi^{S_3} = \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 0 & 1 & 0 \\ \hline \end{array}$$

Comment reconnaître si un vecteur est solution ?

i	1	2	3	4	5	
gain g_i	5	7	10	11	10	
poids p_i	2	8	10	6	5	≤ 15

Algorithme de reconnaissance :

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

$$S_3 = \{1, 2, 4\}$$

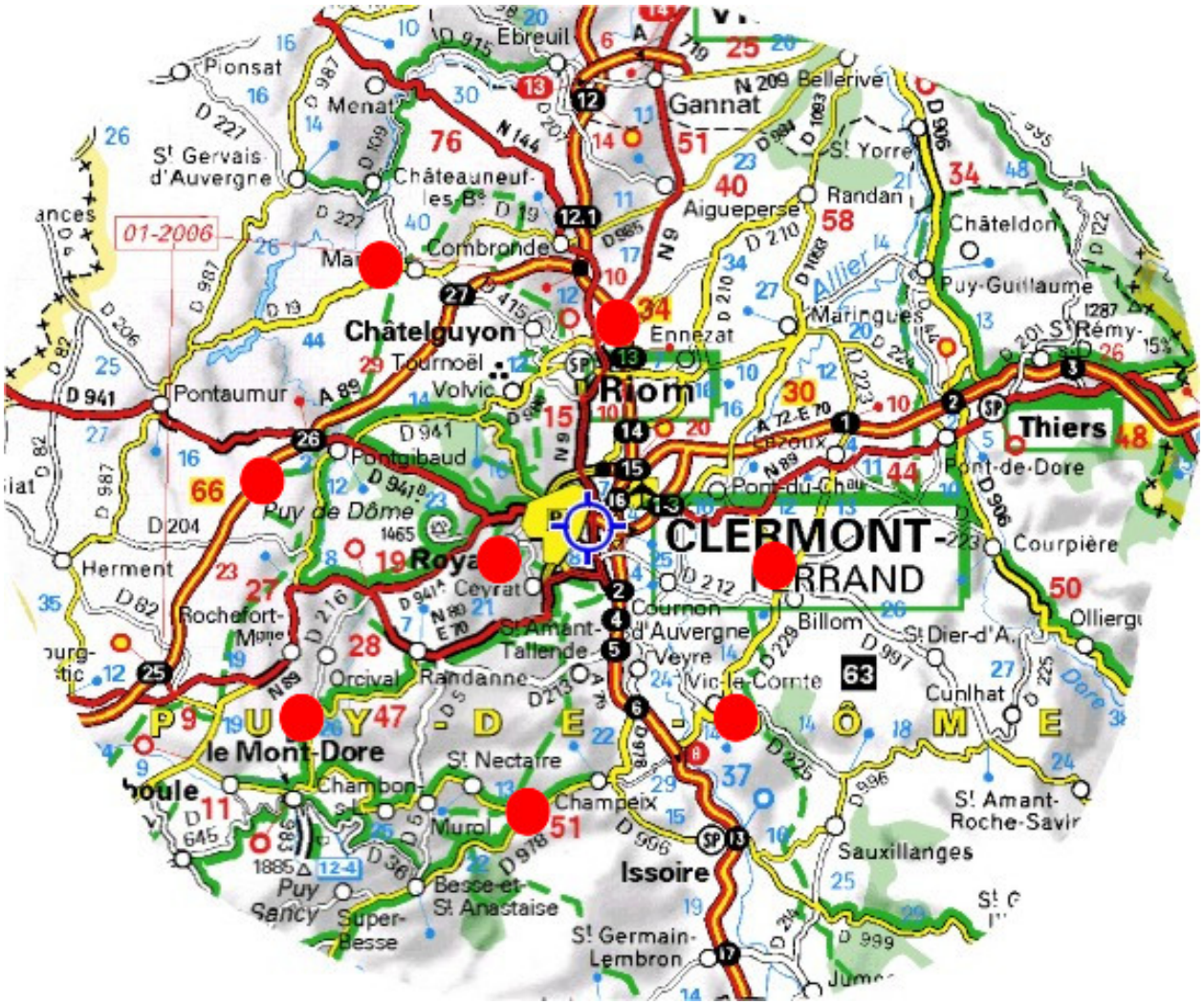
$$\chi^{S_3} = \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 0 & 1 & 0 \\ \hline \end{array}$$

$$\text{poids} : \quad 2 \quad 8 \qquad\qquad 6 \qquad\qquad = 16$$

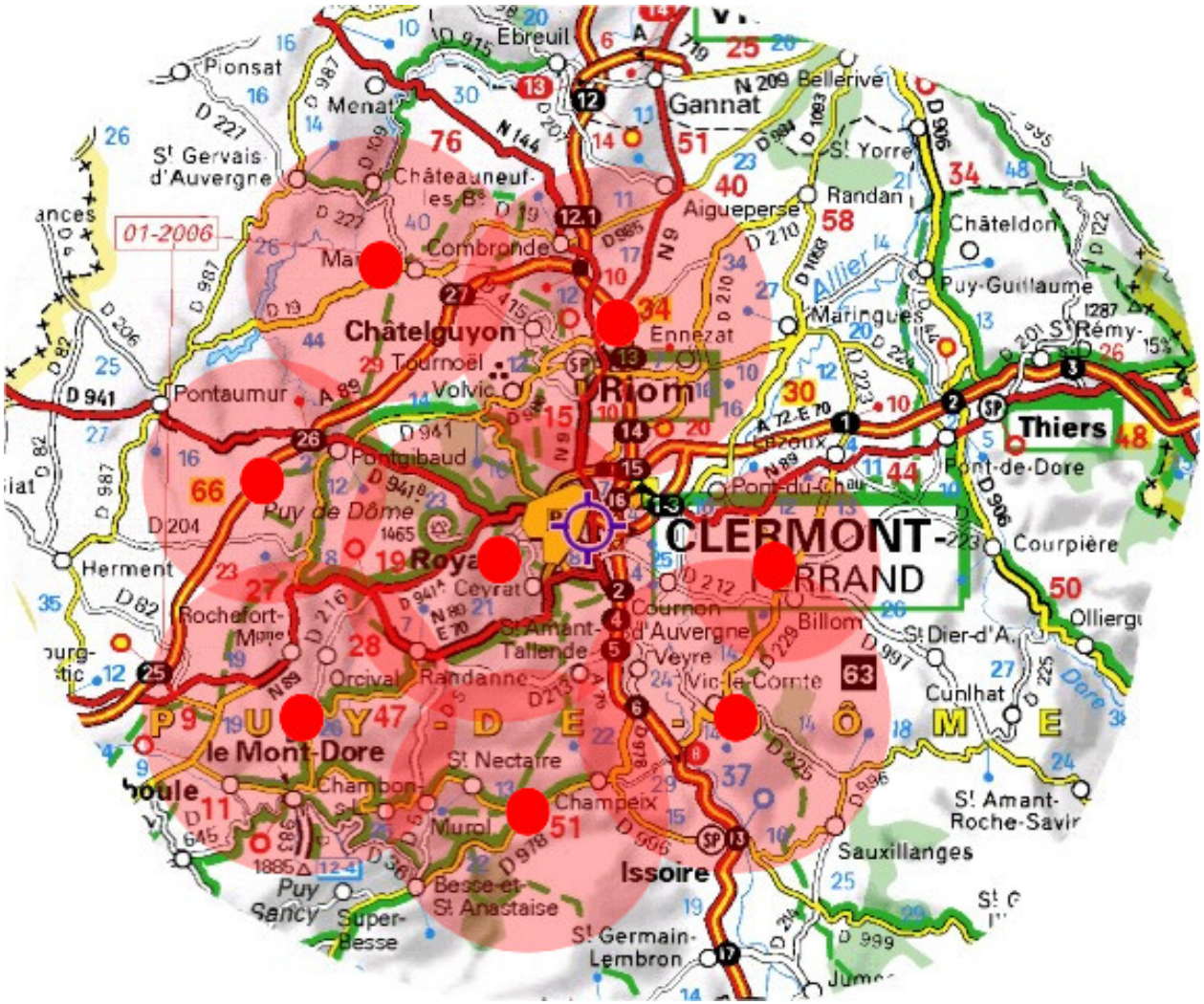
Pas solution

De grands sac-à-dos...

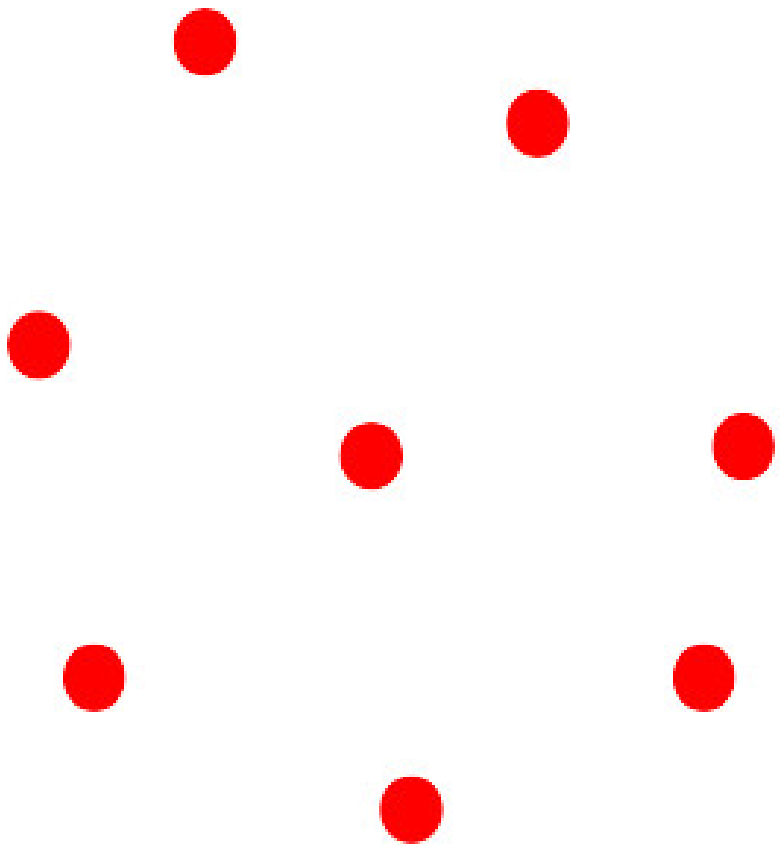
Interférence hertzienne



Couverture hertzienne

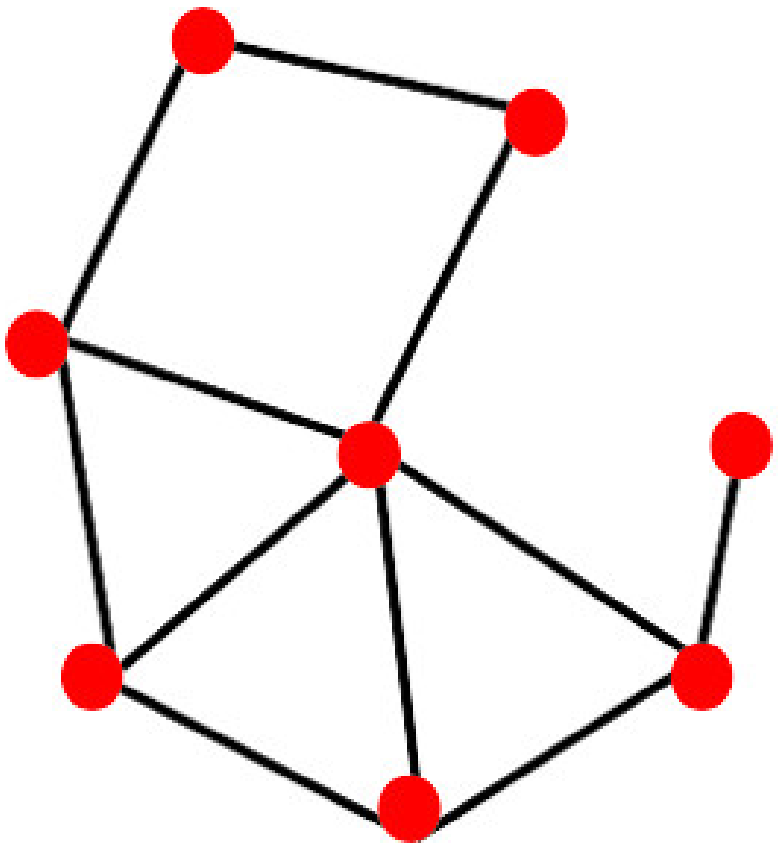


Modélisation par un graphe



Un graphe $G = (S, A)$
avec
S : ensemble des sommets

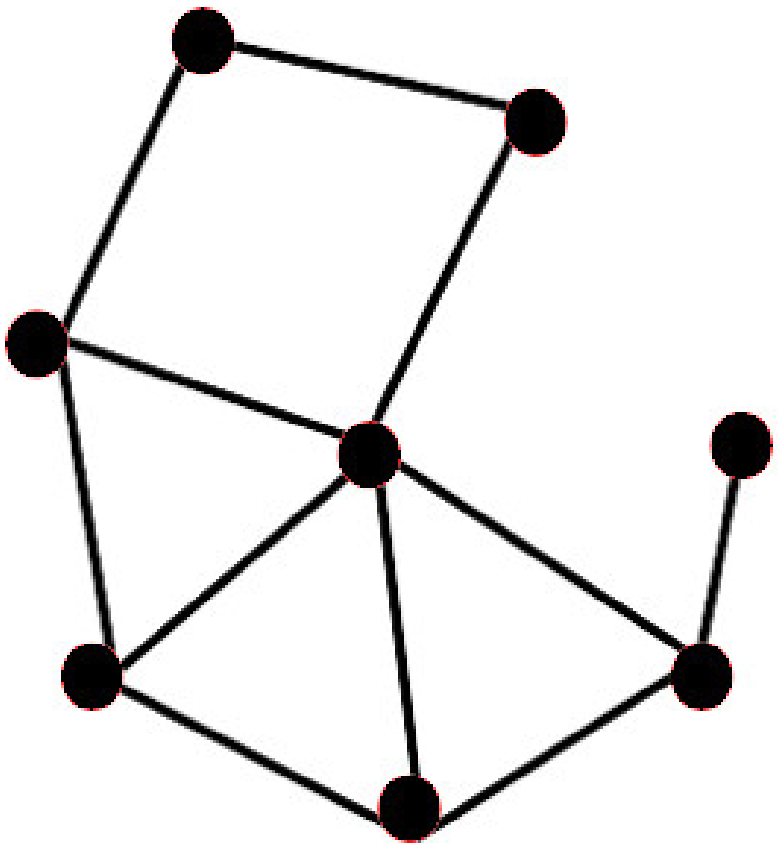
Modélisation par un graphe



Un graphe $G = (S, A)$
avec
S : ensemble des sommets
A : ensemble des arêtes

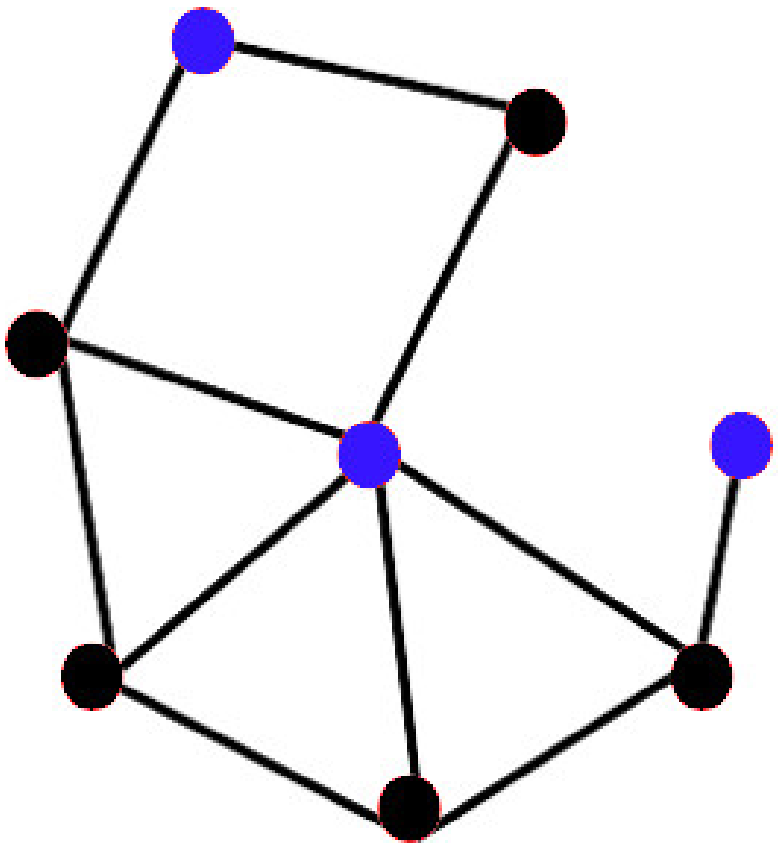
Une arête représente que le conflit entre deux arêtes trop proches : on ne peut leur donner la même fréquence.

Problème du stable



Un **stable** est un ensemble de sommet non reliés par des arêtes.

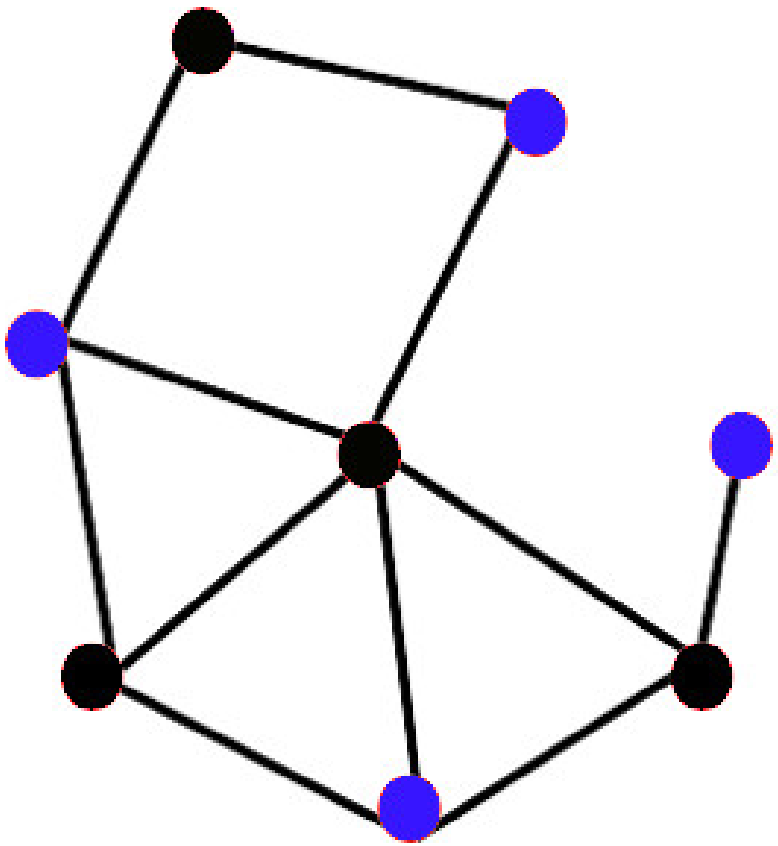
Problème du stable



Un **stable** est un ensemble de sommet non reliés par des arêtes.

Le **Problème du stable maximum** consiste à déterminer le stable ayant le plus de sommets possible.

Problème du stable

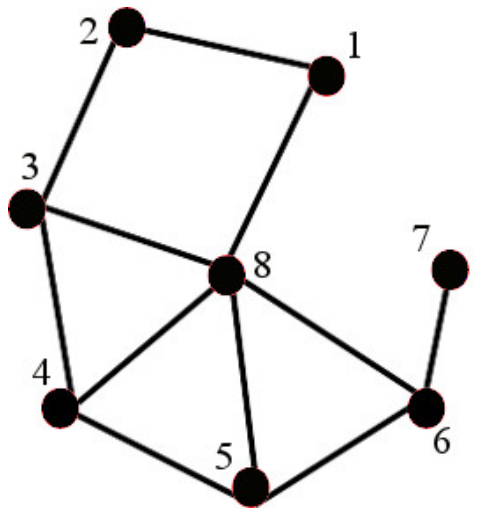


Un **stable** est un ensemble de sommet non reliés par des arêtes.

Le **Problème du stable maximum** consiste à déterminer le stable ayant le plus de sommets possible.

Comment représenter une solution du problème ?

Instance donnée par
un graphe avec
n sommets



Une solution $S \subset \{1, \ldots, n\}$
correspond à un
vecteur d'incidence χ^S
tel que $\chi^S[i] = \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{sinon.} \end{cases}$

$S_1 = \{1, 7, 8\}$

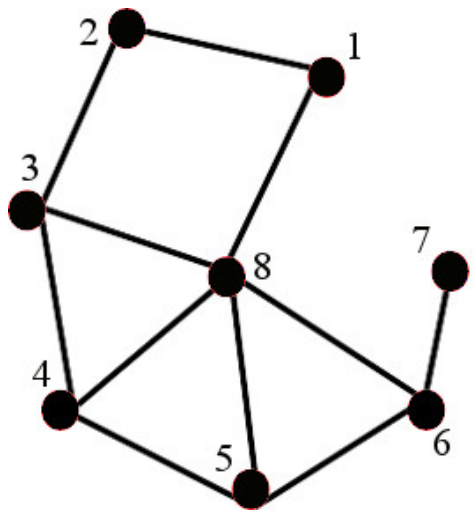
$\chi^{S_1} =$

i	1	2	3	4	5	6	7	8
	1	0	0	0	0	0	1	1

Comment reconnaître si un vecteur est solution ?

Algorithme de reconnaissance :

Pour i de 1 à n
  Pour j de 1 à n
    Si $\{i,j\}$ est une arête
      et si $\chi^S[i] = 1$ et $\chi^S[j] = 1$
    Alors STOP : Faux
  FinPour
FinPour
STOP : Vrai



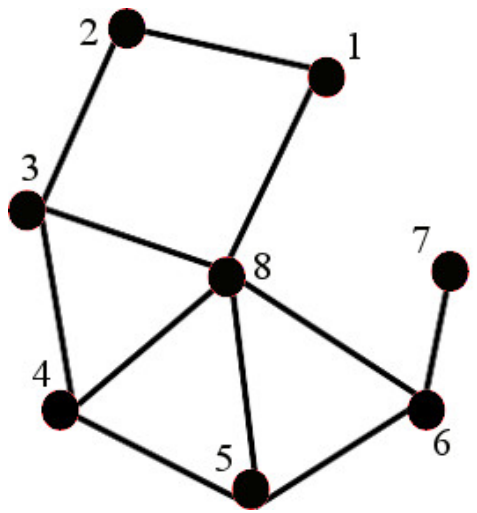
$S_1 = \{2, 7, 8\}$

i	1	2	3	4	5	6	7	8
$\chi^{S_1} =$	0	1	0	0	0	0	1	1

Comment reconnaître si un vecteur est solution ?

Algorithme de reconnaissance :

Pour i de 1 à n
  Pour j de 1 à n
    Si $\{i,j\}$ est une arête
      et si $\chi^S[i] = 1$ et $\chi^S[j] = 1$
    Alors STOP : Faux
  FinPour
FinPour
STOP : Vrai



$S_1 = \{2, 7, 8\}$

i	1	2	3	4	5	6	7	8
$\chi^{S_1} =$	0	1	0	0	0	0	1	1

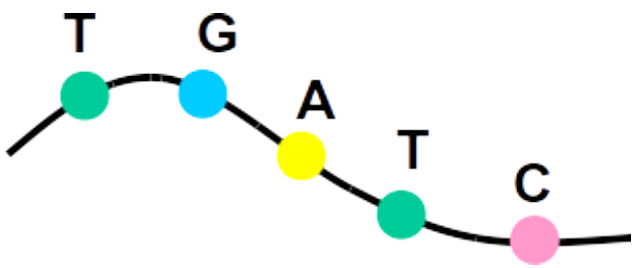
Solution (avec 3 sommets)

Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

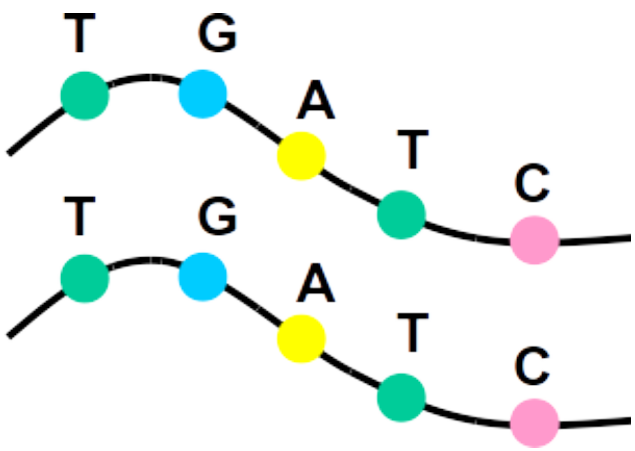


Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

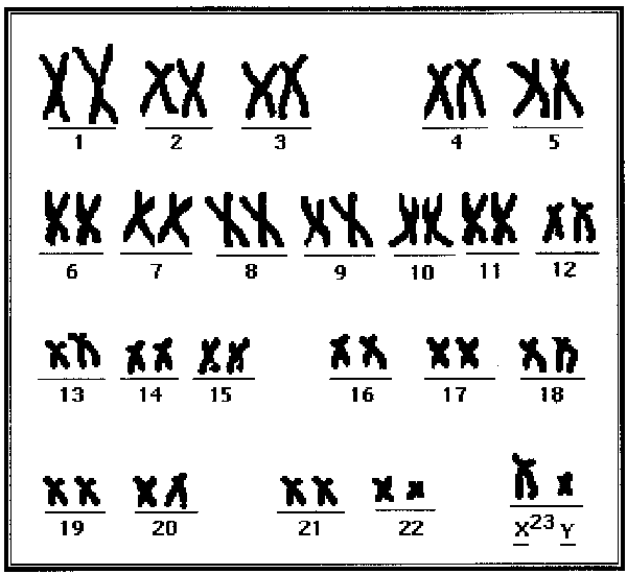


Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

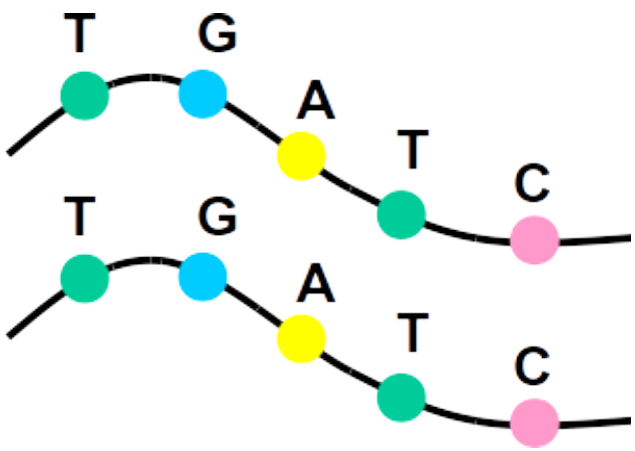


Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

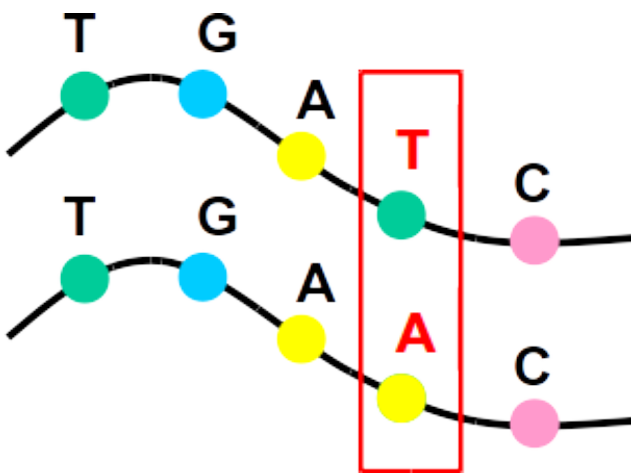


Combinatoire du génôme

L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).

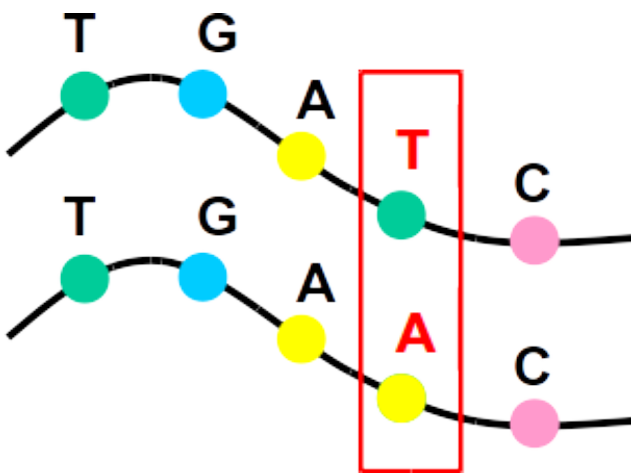


Combinatoire du génôme

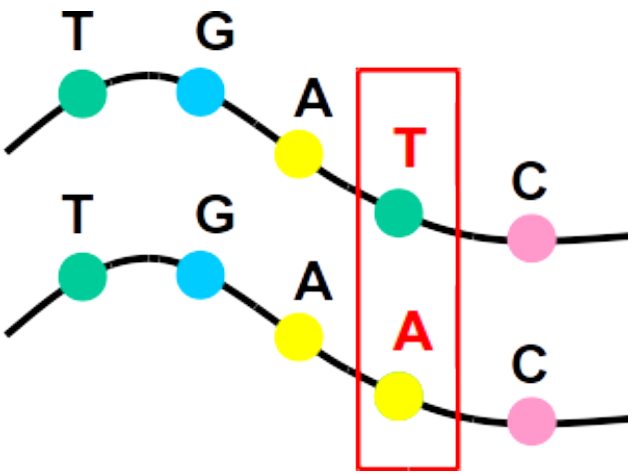
L'ADN d'un organisme est une **combinaison** des nucléotides A,C,G,T.

Les organismes diploïdes comme les humains ont deux branches d'ADN par chromosomes.

90% des variations entre les 2 branches sont simplement des différences d'un nucléotide isolé, appelé SNP (Single nucleotid polymorphism).



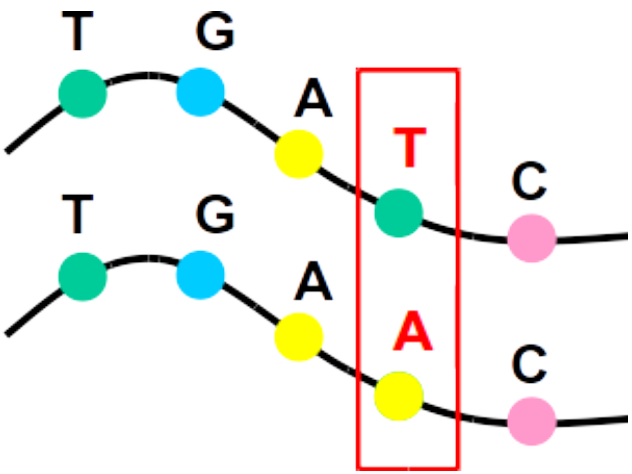
Combinatoire du génome



Lors du séquençage du génome, on doit trier parmi des petis fragments d'ADN qui ont été séquencé par des procédés bio-mécaniques.
Lors de l'opération, des SNPs faux ont été créés

On veut trier les fragments en omettant le moint de SNPs possible.

Combinatoire du génome

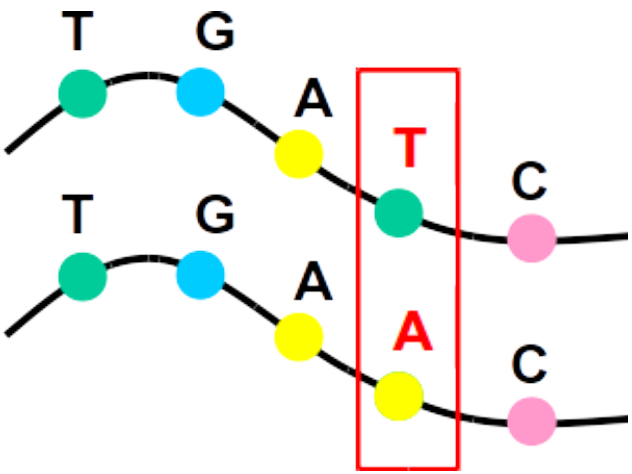


Lors du séquençage du génome, on doit trier parmi des petis fragments d'ADN qui ont été séquencé par des procédés bio-mécaniques.
Lors de l'opération, des SNPs faux ont été créés

On veut trier les fragments en omettant le moint de SNPs possible.

C'est un problème **d'optimisation combinatoire !**

Combinatoire du génome



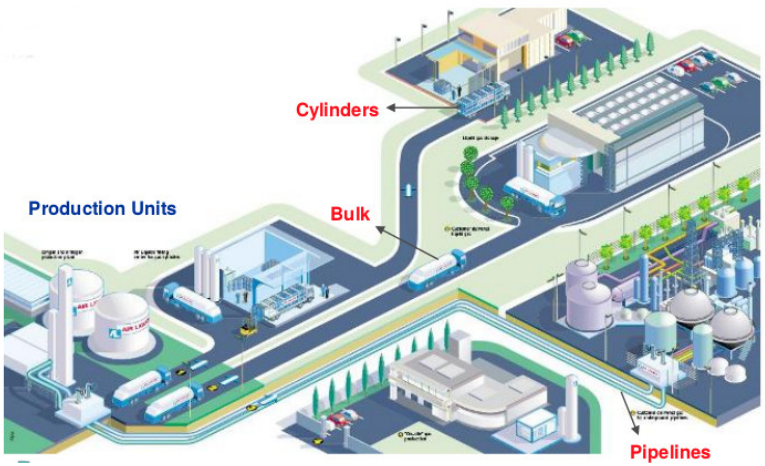
Lors du séquençage du génome, on doit trier parmi des petis fragments d'ADN qui ont été séquencé par des procédés bio-mécaniques.
Lors de l'opération, des SNPs faux ont été créés

On veut trier les fragments en omettant le moint de SNPs possible.

C'est un problème **d'optimisation combinatoire !**
Il a été montré que ce problème est exactement celui du **stable maximum**.

Lippert, R., Schwartz, R., Lancia, G., Istrail, S. : Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. Brief. Bioinform 3, 23–31 (2002)

Problème posé pour le Challenge Roadef 2016



L'entreprise **Air Liquide** propose ce problème réel :
Déterminer les tournées des camions d'oxygène liquide pour délivrer les hôpitaux
de manière à ce que :
- les cuves des hopitaux ne soient jamais vide ("monitoring" à distance)
- les tournées soient faisables dans la journée de travail du conducteur
- les coûts soient minimisés !

Problème d'optimisation combinatoire :

Etant donnés :
un ensemble fini d'éléments $\{1, ..., n\}$
avec un coût par élément $c_1, ..., c_n$

Problème d'optimisation combinatoire :

Etant donnés :
un ensemble fini d'éléments $\{1, ..., n\}$
avec un coût par élément $c_1, ..., c_n$

Et une famille $\mathcal{F}$ de sous-ensembles de $\{1, \ldots, n\}$
Ces sous-ensemble sont appelés **solutions**

Trouver un ensemble $F \in \mathcal{F}$ de poids $\sum_{i \in F} c_i$ maximum (ou minimum)

Reconnaissance

Sac-à-dos

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

Son temps d'exécution est proportionnel à n.

Reconnaissance

Sac-à-dos

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

Son temps d'exécution est proportionnel à n.

Reconnaissance Stable

Pour i de 1 à n

Pour j de 1 à n

Si $\{i, j\}$ est une arête

et si $\chi^S[i] = \chi^S[j] = 1$

Alors STOP : Faux

FinPour

FinPour

STOP : Vrai

Son temps d'exécution est proportionnel à n^2.

Reconnaissance

Sac-à-dos

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

Son temps d'exécution est proportionnel à n.

Reconnaissance Stable

Pour i de 1 à n

Pour j de 1 à n

Si $\{i, j\}$ est une arête

et si $\chi^S[i] = \chi^S[j] = 1$

Alors STOP : Faux

FinPour

FinPour

STOP : Vrai

Son temps d'exécution est proportionnel à n^2.

Les algorithmes **rapides** ont un temps d'exécution proportionnel à n ou à n^2 ou à n^3 ou à n^4,..., ou à n^{12},...
que l'on appelle **Algorithmes Polynomiaux**.

Reconnaissance

Sac-à-dos

$pds \leftarrow 0$

Pour i allant de 1 à n

$pds \leftarrow pds + p_i * \chi^S[i]$

FinPour

Si $pds \leq P$ **Alors** Vrai

Sinon Faux

Son temps d'exécution est proportionnel à n.

Reconnaissance Stable

Pour i de 1 à n

Pour j de 1 à n

Si $\{i, j\}$ est une arête

et si $\chi^S[i] = \chi^S[j] = 1$

Alors STOP : Faux

FinPour

FinPour

STOP : Vrai

Son temps d'exécution est proportionnel à n^2.

Les algorithmes **très très lents** ont un temps d'exécution proportionnel à 2^n, 3^n,... que l'on appelle **Algorithmes exponentiels**.

Les algorithmes **rapides** ont un temps d'exécution proportionnel à n ou à n^2 ou à n^3 ou à n^4,..., ou à n^{12},... que l'on appelle **Algorithmes Polynomiaux**.

Enumérer ?

Prenons un problème d'optimisation avec n éléments.
Supposons que l'on connaisse un algorithme **polynomial** (donc rapide) pour reconnaître une solution

Peut-on déterminer la meilleure solution par **énumération** ?

Enumérer ?

Prenons un problème d'optimisation avec n éléments.
Supposons que l'on connaisse un algorithme **polynomial** (donc rapide) pour reconnaître une solution

Peut-on déterminer la meilleure solution par **énumération** ?

Pour chaque sous-ensemble S d'éléments
  $\chi^S \leftarrow$ le vecteur d'incidence de S.
  Algorithme de réconnaissance pour χ^S.
  Si χ^S est solution
    Conserver chi^S si meilleure solution rencontrée.
FinPour
Retourner la meilleure solution rencontrée.

Enumérer ?

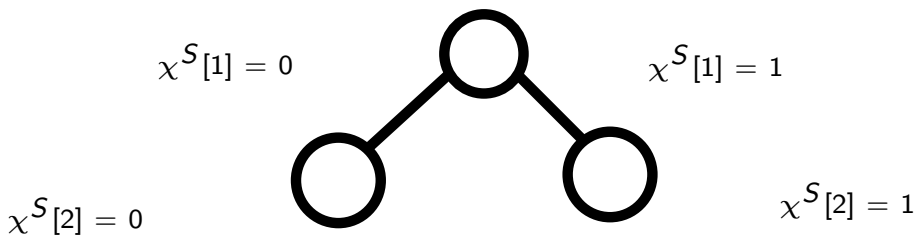
Prenons un problème d'optimisation avec n éléments.
Supposons que l'on connaisse un algorithme **polynomial** (donc rapide) pour reconnaître une solution

Peut-on déterminer la meilleure solution par **énumération** ?

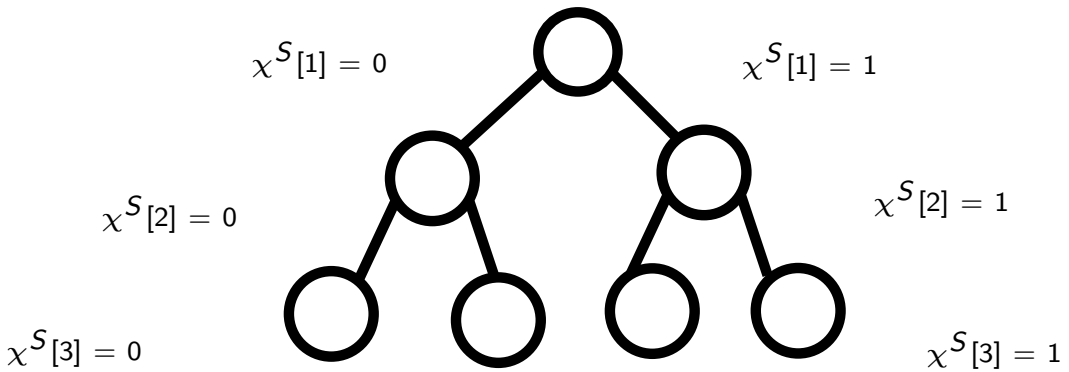
Pour chaque sous-ensemble S d'éléments
  $\chi^S \leftarrow$ le vecteur d'incidence de S.
  Algorithme de réconnaissance pour χ^S.
  Si χ^S est solution
    Conserver chi^S si meilleure solution rencontrée.
FinPour
Retourner la meilleure solution rencontrée.

Il y a 2^n **sous-ensembles** : Temps d'exécution **exponentiel**

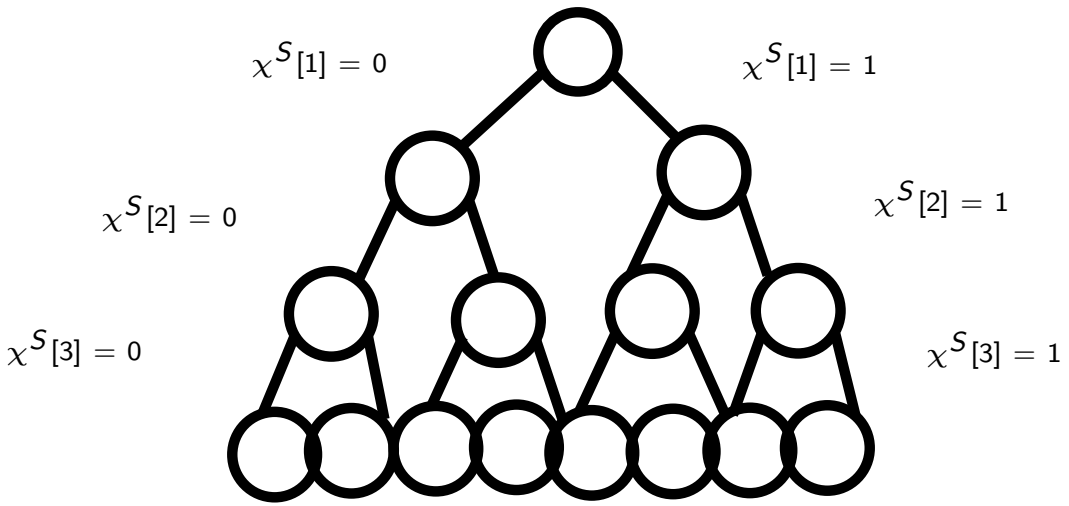
Explosion combinatoire



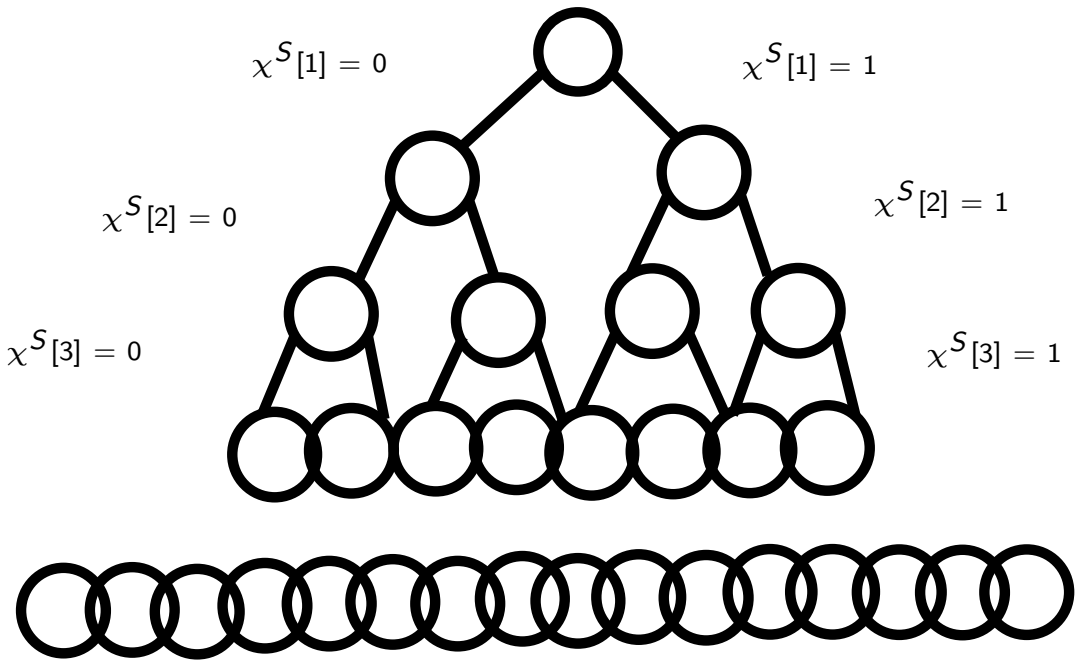
Explosion combinatoire



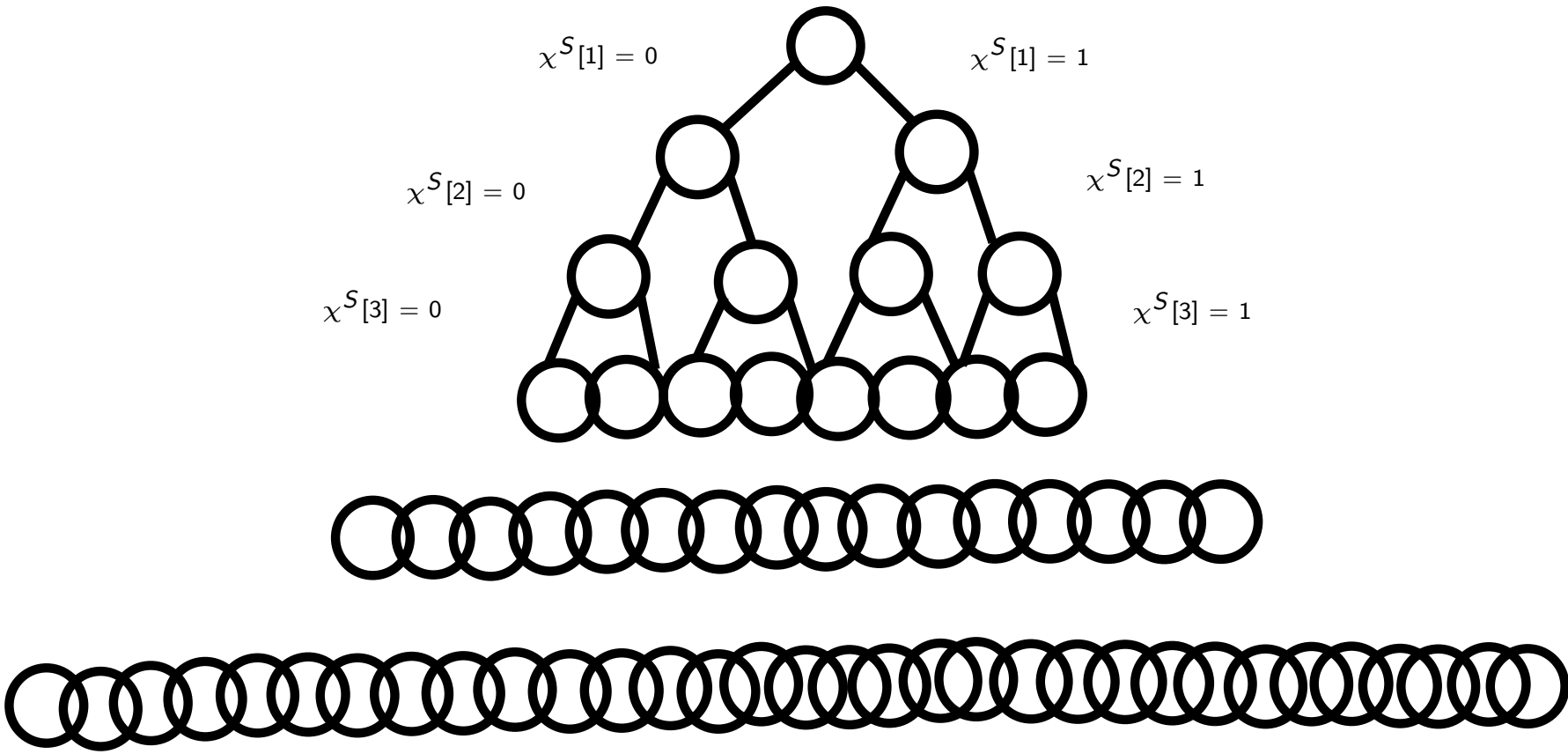
Explosion combinatoire



Explosion combinatoire



Explosion combinatoire



Explosion combinatoire

Prenons un des plus puissants calculateurs en 2015
Tianhe-2 (Chine)
33,86 pétaflops où 1 pétaflop représente le traitement de 10^{15} d'opérations par seconde (un million de milliards).

Supposons que l'algorithme de reconnaissance prennent $100n$ opérations élémentaires.
Ainsi pour $n = 10$, on peut traiter 33 860 milliards de sous-ensembles en 1 seconde !

	Tianhe-2			Ordinateur du futur
n	n^3	n^5	$100n2^n$	$100n2^n$
10	0,00...001 sec	0,00...001 sec		
20	0,00...001 sec	0,00...001 sec		
50	0,00...001 sec	0,00...001 sec		
60	0,00...001 sec	0,00000002 sec		
80	0,00...001 sec	0,0000009 sec		
100	0,00...001 sec	0,0000003 sec		
1000	0,00000003 sec	0,03 sec		

Explosion combinatoire

Prenons un des plus puissants calculateurs en 2015
Tianhe-2 (Chine)
33,86 pétaflops où 1 pétaflop représente le traitement de 10^{15} d'opérations par seconde (un million de milliards).

Supposons que l'algorithme de reconnaissance prennent $100n$ opérations élémentaires.
Ainsi pour $n = 10$, on peut traiter 33 860 milliards de sous-ensembles en 1 seconde !

	Tianhe-2			Ordinateur du futur
n	n^3	n^5	$100n2^n$	$100n2^n$
10	0,00...001 sec	0,00...001 sec	0,00...001 sec	
20	0,00...001 sec	0,00...001 sec	0,00000006 sec	
50	0,00...001 sec	0,00...001 sec	168,9 sec	
60	0,00...001 sec	0,00000002 sec	57,6 h	
80	0,00...001 sec	0,0000009 sec	9200 ans	
100	0,00...001 sec	0,0000003 sec		
1000	0,00000003 sec	0,03 sec		

Explosion combinatoire

Prenons un des plus puissants calculateurs en 2015
Tianhe-2 (Chine)
33,86 pétaflops où 1 pétaflop représente le traitement de 10^{15} d'opérations par seconde (un million de milliards).

Supposons que l'algorithme de reconnaissance prennent $100n$ opérations élémentaires.
Ainsi pour $n = 10$, on peut traiter 33 860 milliards de sous-ensembles en 1 seconde !

	Tianhe-2			Ordinateur du futur
n	n^3	n^5	$100n2^n$	$100n2^n$
10	0,00...001 sec	0,00...001 sec	0,00...001 sec	
20	0,00...001 sec	0,00...001 sec	0,00000006 sec	
50	0,00...001 sec	0,00...001 sec	168,9 sec	
60	0,00...001 sec	0,00000002 sec	57,6 h	
80	0,00...001 sec	0,0000009 sec	9200 ans	
100	0,00...001 sec	0,0000003 sec	$1,21.10^{10}$ ans	
1000	0,00000003 sec	0,03 sec	1.10^{282} ans	

Age de l'univers $13,7.10^9$ ans

Explosion combinatoire

Prenons un des plus puissants calculateurs en 2015
Tianhe-2 (Chine)
33,86 pétaflops où 1 pétaflop représente le traitement de 10^{15} d'opérations par seconde (un million de milliards).

Supposons que l'algorithme de reconnaissance prennent $100n$ opérations élémentaires.
Ainsi pour $n = 10$, on peut traiter 33 860 milliards de sous-ensembles en 1 seconde !

	Tianhe-2			Ordinateur du futur
n	n^3	n^5	$100n2^n$	$100n2^n$
10	0,00...001 sec	0,00...001 sec	0,00...001 sec	0,00...001 sec
20	0,00...001 sec	0,00...001 sec	0,00000006 sec	0,00...001 sec
50	0,00...001 sec	0,00...001 sec	168,9 sec	0,000001 sec
60	0,00...001 sec	0,00000002 sec	57,6 h	0,0001 sec
80	0,00...001 sec	0,0000009 sec	9200 ans	100 ans
100	0,00...001 sec	0,0000003 sec	$1,21.10^{10}$ ans	$1,21.10^{9}$ ans
1000	0,00000003 sec	0,03 sec	1.10^{282} ans	...

Age de l'univers $13,7.10^{9}$ ans

Explosion combinatoire

Enumérer 2^n solutions n'est donc pas un problème technique que des ordinateurs très puissants pourraient balayer.

Il est nécessaire de **contourner l'explosion combinatoire** par des outils mathématiques et algorithmiques.

Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Pour casser une noix, on peut :

Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

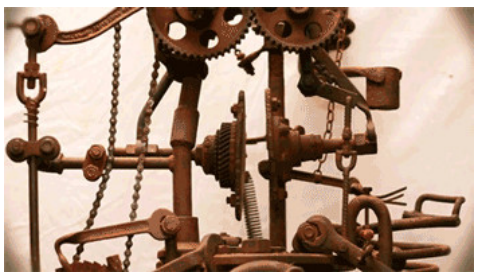
Pour casser une noix, on peut :

Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Pour casser une noix, on peut :



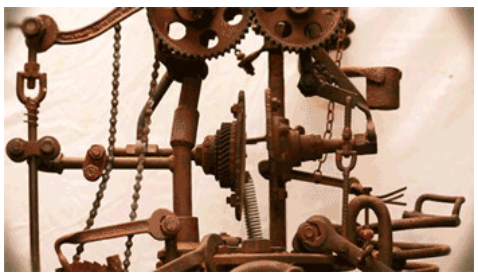


Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Pour casser une noix, on peut :



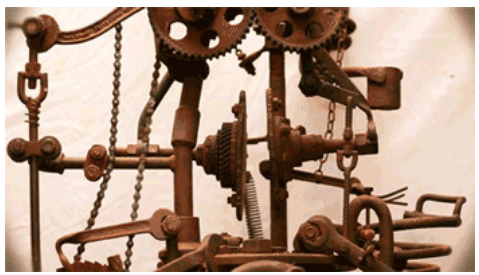



Complexité des problèmes

Ce n'est pas parce qu'on connaît un algorithme exponentiel pour résoudre un problème que le problème est difficile !

Pour casser une noix, on peut :






On cherche l'algorithme **le plus rapide** pour résoudre un problème !

Complexité des problèmes

Un problème est **de complexité exponentiel**
s'il existe un algorithme **exponentiel** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{EXP}$.

Complexité des problèmes

Un problème est **de complexité exponentiel**
s'il existe un algorithme **exponentiel** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{EXP}$.

Un problème est **de complexité polynomiale**
s'il existe un algorithme **polynomial** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{P}$.

Complexité des problèmes

Un problème est **de complexité exponentiel**
s'il existe un algorithme **exponentiel** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{EXP}$.

Un problème est **de complexité polynomiale**
s'il existe un algorithme **polynomial** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{P}$.

Donc $\mathcal{P}$ est **incluse** dans $\mathcal{EXP}$, on note $\mathcal{P} \subset \mathcal{EXP}$

Complexité des problèmes

Un problème est **de complexité exponentiel**
s'il existe un algorithme **exponentiel** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{EXP}$.

Un problème est **de complexité polynomiale**
s'il existe un algorithme **polynomial** pour le résoudre
$\Rightarrow$ la classe de problème $\mathcal{P}$.

Donc $\mathcal{P}$ est **incluse** dans $\mathcal{EXP}$, on note $\mathcal{P} \subset \mathcal{EXP}$

Question : Dans quelles classes de complexité sont les problèmes d'optimisation combinatoire ?

Problème $\mathcal{NP}$

On crée un classe un peu particulière...

Les problèmes d'optimisation combinatoire pour lesquels
on sait **reconnaître** par un algorithme polynomial
si un sous-ensemble d'éléments est solution

$\Rightarrow$ la classe de problème $\mathcal{NP}$ "Nondeterministic polynomial time"

Problème $\mathcal{NP}$

On crée un classe un peu particulière...

Les problèmes d'optimisation combinatoire pour lesquels
on sait **reconnaître** par un algorithme polynomial
si un sous-ensemble d'éléments est solution

$\Rightarrow$ la classe de problème $\mathcal{NP}$ "Nondeterministic polynomial time"

Sous cette hypothèse, on a vu qu'il **possible** d'utiliser un
algorithme d'énumération exponentiel

$\Rightarrow \mathcal{NP} \subset \mathcal{EXP}$

Problème $\mathcal{NP}$

On crée un classe un peu particulière...

Les problèmes d'optimisation combinatoire pour lesquels
on sait **reconnaître** par un algorithme polynomial
si un sous-ensemble d'éléments est solution

$\Rightarrow$ la classe de problème $\mathcal{NP}$ "Nondeterministic polynomial time"

Sous cette hypothèse, on a vu qu'il **possible** d'utiliser un
algorithme d'énumération exponentiel

$\Rightarrow \mathcal{NP} \subset \mathcal{EXP}$

De plus, un problème polynomial est directement $\mathcal{NP}$

$\Rightarrow \mathcal{P} \subset \mathcal{NP} \subset \mathcal{EXP}$

Et alors ?

$\mathcal{P}$ est-il égal à $\mathcal{NP}$?

Que l'on peut traduire par :

"Existe-t-il un algorithme polynomial pour résoudre les problèmes d'optimisation combinatoire ?"

Réponse :

Et alors ?

$\mathcal{P}$ est-il égal à $\mathcal{NP}$?

Que l'on peut traduire par :

"Existe-t-il un algorithme polynomial pour résoudre les problèmes d'optimisation combinatoire ?"

Réponse : On n'en sait rien !

A l'échelle de l'humanité, on ne connaît que cet algorithme d'énumération !

C'est un des 7 problèmes du "Millenium Prize Problems" du Clay Mathematics Institute of Cambridge, Massachussetts, doté d'un million de dollar !

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

Regardons un problème très simple : un sac-à-dos avec tous les objets de même poids.

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

Regardons un problème très simple : un sac-à-dos avec tous les objets de même poids.

La solution est :
- trier les n objets du plus chers au moins chers
- de prendre les objets un à un dans cet ordre tant que cela rentre dans le sac !

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

Regardons un problème très simple : un sac-à-dos avec tous les objets de même poids.

La solution est :
- trier les n objets du plus chers au moins chers
- de prendre les objets un à un dans cet ordre tant que cela rentre dans le sac !

Cet algorithme **glouton** est polynomial (de l'ordre de n^2).

Un sac-à-dos très simple : algorithme glouton

Est-ce que tous les problèmes d'Optimisation Combinatoires sont difficiles ?

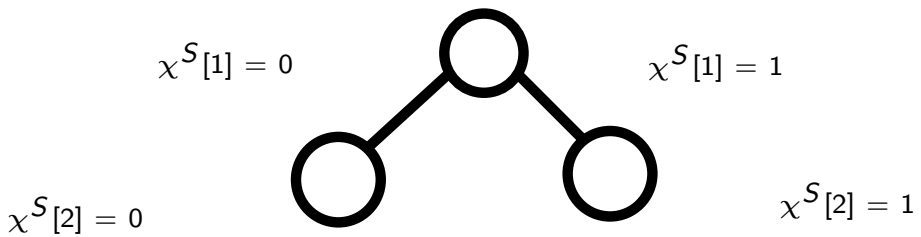
Regardons un problème très simple : un sac-à-dos avec tous les objets de même poids.

La solution est :
- trier les n objets du plus chers au moins chers
- de prendre les objets un à un dans cet ordre tant que cela rentre dans le sac !

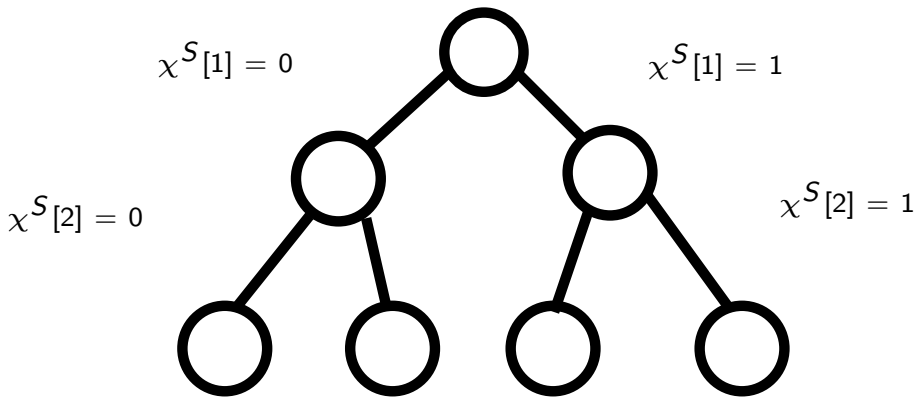
Cet algorithme **glouton** est polynomial (de l'ordre de n^2).

Est-ce que le cas général est lui-aussi polynomial ?

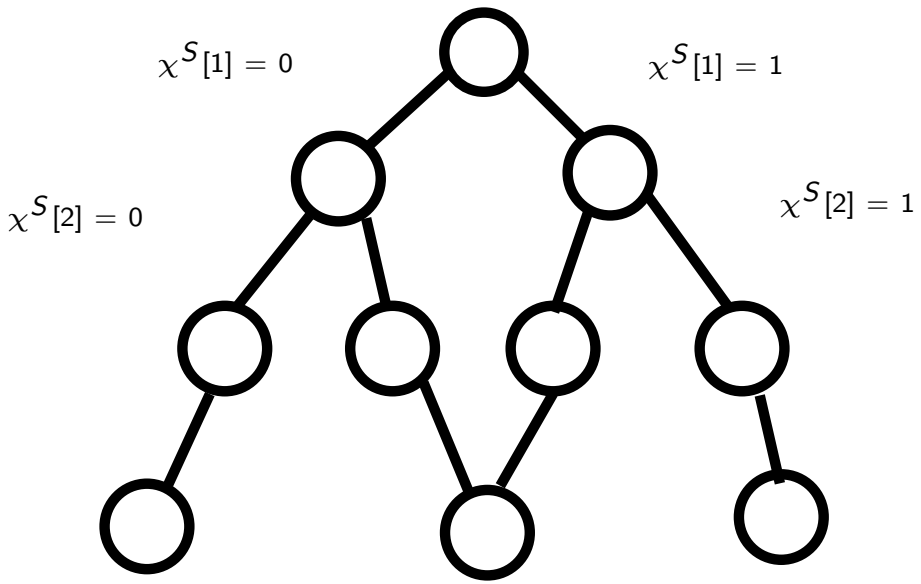
Programmation dynamique



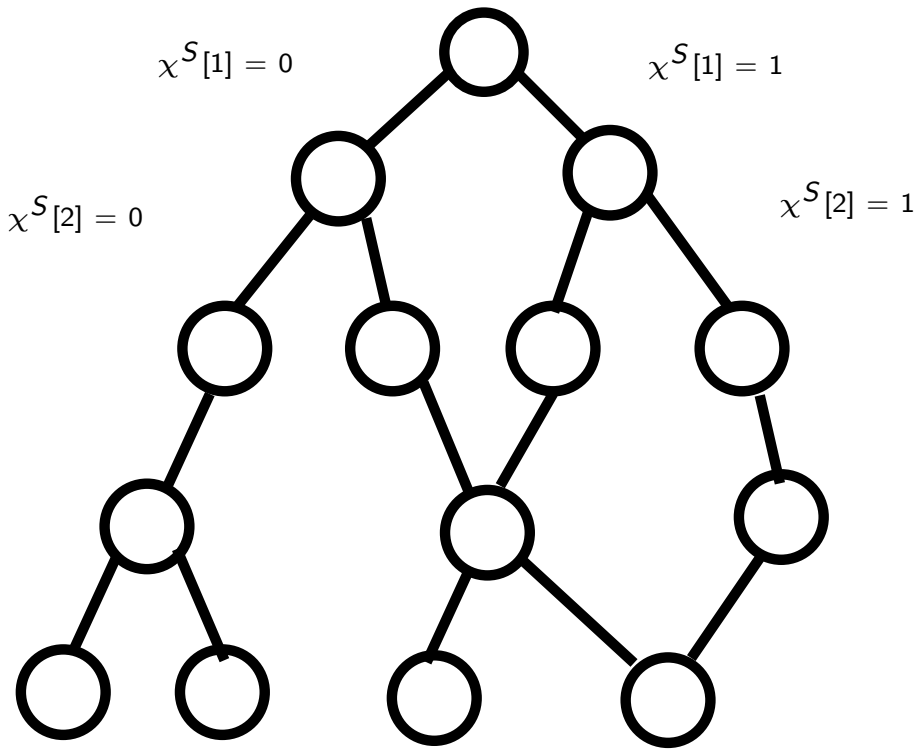
Programmation dynamique



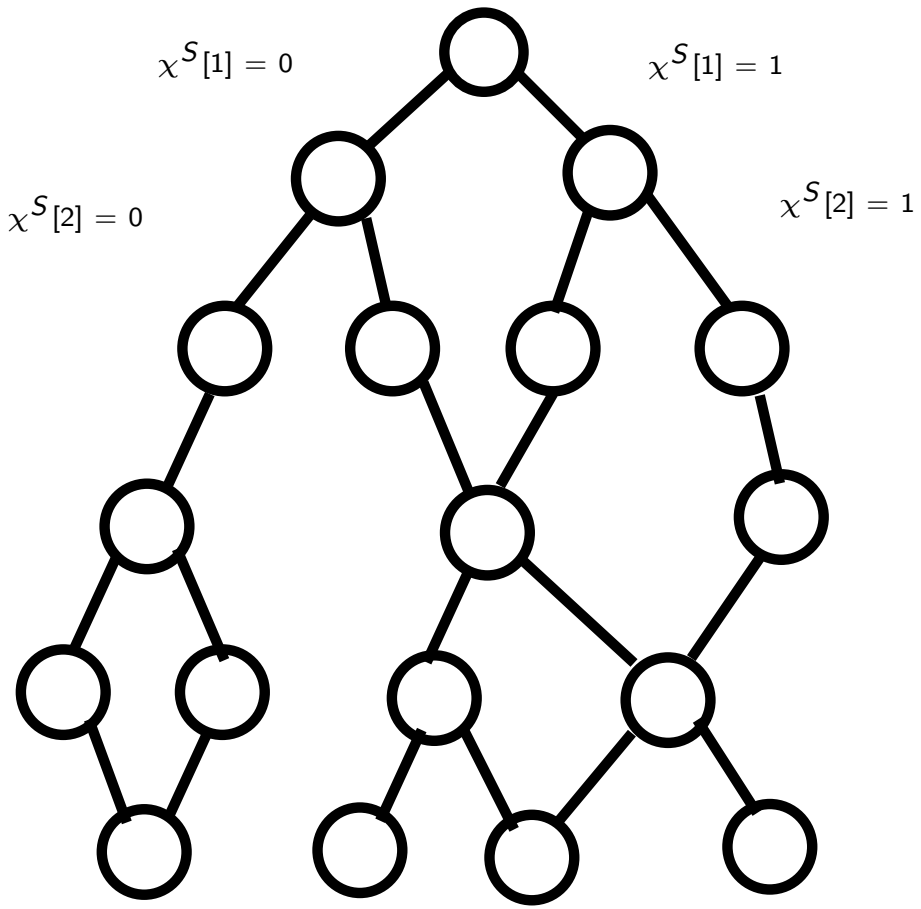
Programmation dynamique



Programmation dynamique



Programmation dynamique



Le “recoupement” freine l’explosion combinatoire

Cas polynomiaux

On connaît beaucoup de cas polynomiaux de problèmes d'optimisation combinatoire :

• Algorithme glouton :
- le problème du sac-à-dos avec tous les objets de même poids.

• Programmation dynamique :
- le problème de sac-à-dos si le sac n'est pas "trop grand"
- le problème du stable dans des graphes simples comme les arbres

• Parcours dans les graphes :
- comment relier tous les points d'un dessin avec une longueur minimale de traits (arbre couvrant minimum)

- ...

$\mathcal{NP}$-difficulté

Dans l'état des connaissances scientifiques, on ne sait pas s'il existe ou non un algorithme polynomial pour résoudre le problème du sac-à-dos en général !

$\mathcal{NP}$-**difficulté**

Dans l'état des connaissances scientifiques, on ne sait pas s'il existe ou non un algorithme polynomial pour résoudre le problème du sac-à-dos en général !

En fait on a pu prouver que ce problème était aussi difficile que tous les problèmes de la classe $\mathcal{NP}$!

On dit qu'un problèmes est $\mathcal{NP}$-**difficile** s'il est aussi difficile que tout problème de la classe $\mathcal{NP}$.

$\mathcal{NP}$-difficulté

Patron, je ne trouve pas d'algorithme polynomial pour le problème du sac-à-dos

Figure prise dans le "Garey et Johnson"

$\mathcal{NP}$-difficulté

Mais si vous pensez que je suis juste pas très doué...

Figure prise dans le "Garey et Johnson"

$\mathcal{NP}$-difficulté

... tous les autres ne le sont pas non plus !

Figure prise dans le “Garey et Johnson”

Problèmes $\mathcal{NP}$-difficiles

On connaît beaucoup de problèmes $\mathcal{NP}$-difficiles :

- le problème du sac-à-dos

- le problème du stable maximum

- les problèmes d'ordonnancement de tâches

- les problèmes de coloration de graphes

- ... et la plupart des problèmes de recherche opérationnelle venant de l'industrie.

Problème du plus court chemin

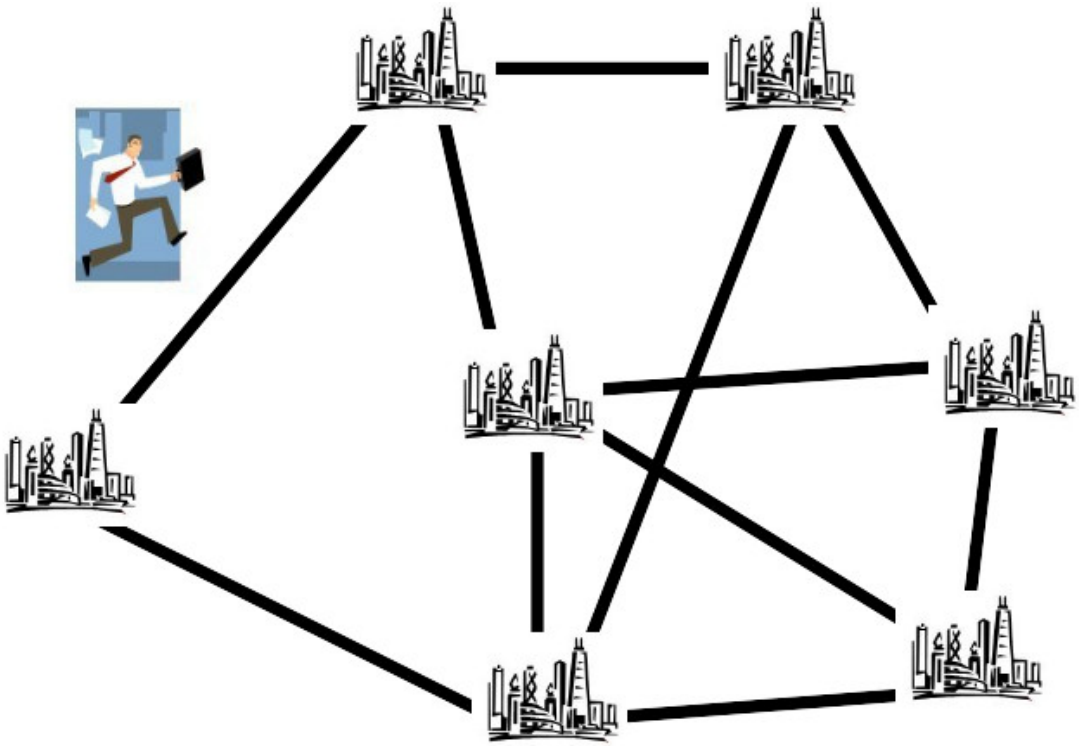
Aller d'un point à un autre au plus court sur une carte.

Problème du plus court chemin

Aller d'un point à un autre au plus court sur une carte.

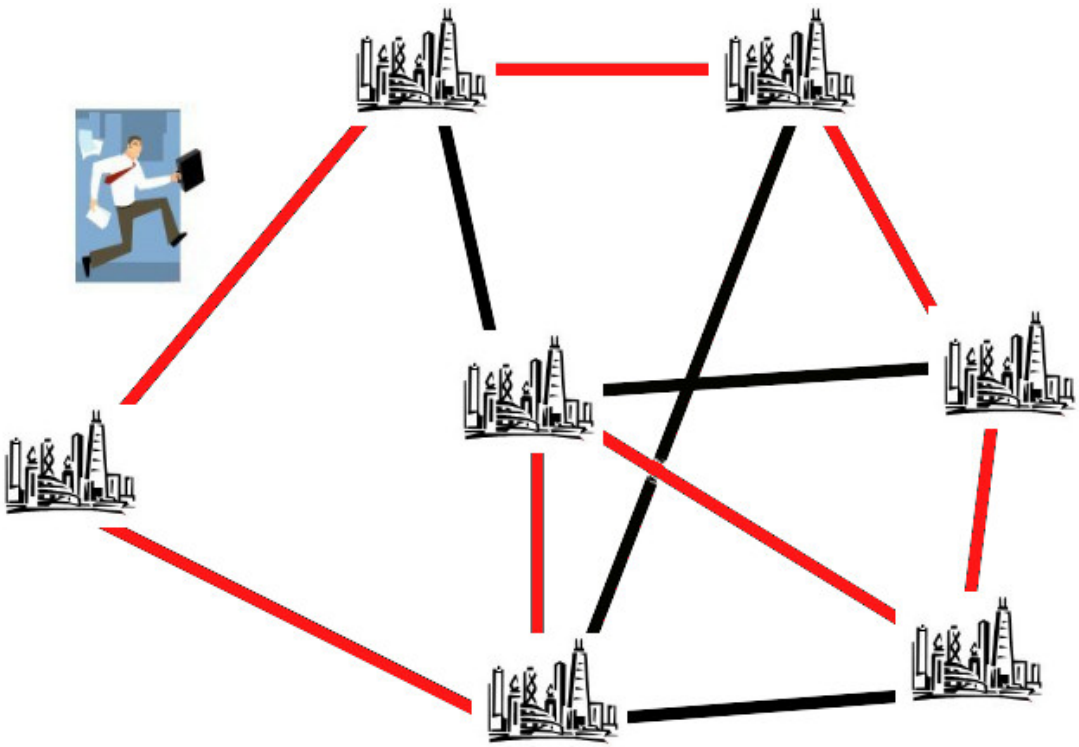
Quelle difficulté ?

Problème du voyageur de commerce



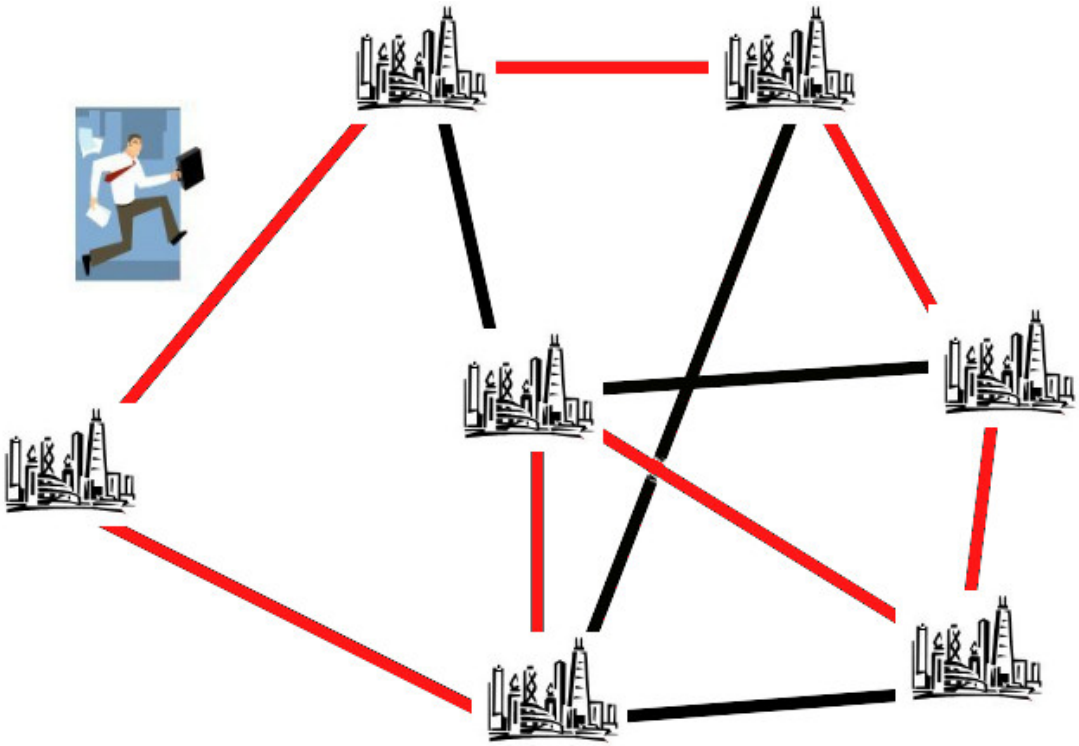
Partir d'un point et y revenir
en passant par tous les points
au plus court sur une carte.

Problème du voyageur de commerce



Partir d'un point et y revenir
en passant par tous les points
au plus court sur une carte.

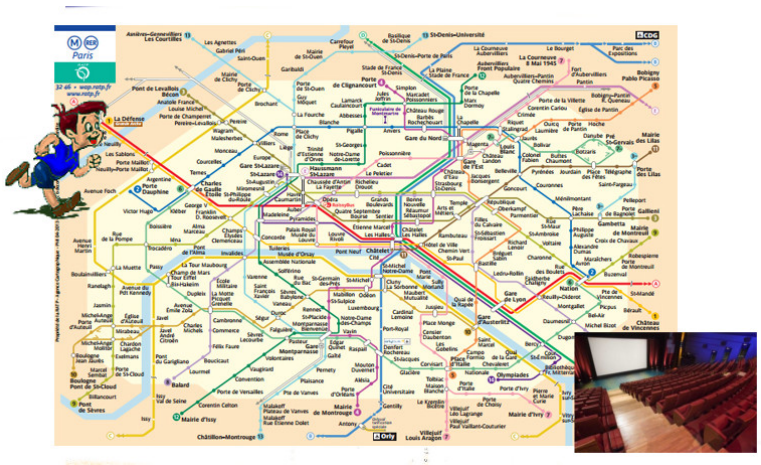
Problème du voyageur de commerce



Partir d'un point et y revenir
en passant par tous les points
au plus court sur une carte.

Quelle difficulté ?

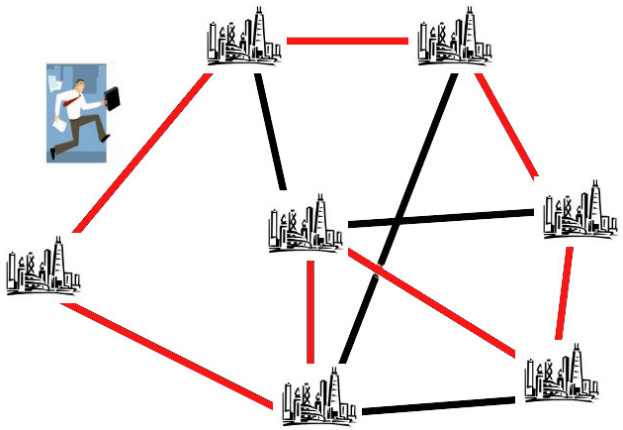
Plus court chemin



Polynomial

On sait le résoudre pour des millions de croisements !

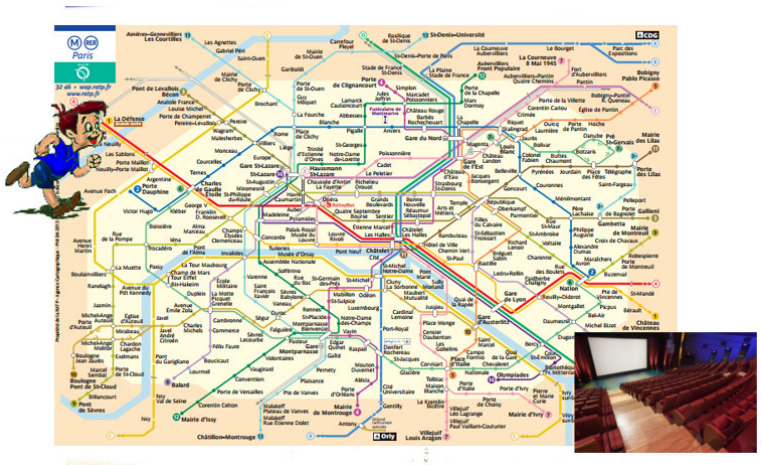
Voyageur de commerce



$\mathcal{NP}$-difficile

Il y a 20 ans, on ne savait pas dépasser quelques centaines de villes !

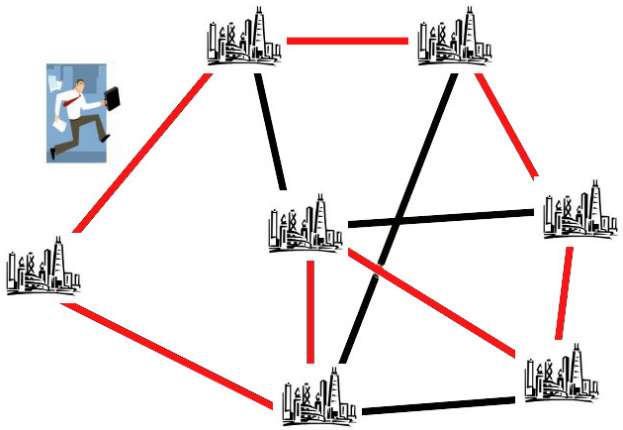
Plus court chemin



Polynomial

On sait le résoudre pour des millions de croisements !

Voyageur de commerce



$\mathcal{NP}$-difficile

Il y a 20 ans, on ne savait pas dépasser quelques centaines de villes !

Et pourtant, en 2003, des instances à 200 000 villes ont été résolues !

Et comment a-été résolu le problème du voyageur de commerce à 200 000 villes ?

En contournant l'explosion combinatoire...

C'est ce que nous allons voir !