Complexité et Algorithmique

- ISIMA, 2F2 & 2F3
- Christophe Duhamel & Philippe Lacomme

Plan du Cours

- Complexité algoritmique
- Conception d'algorithmes
- Problèmes combinatoires
- Problèmes d'ordonnancement
- Problèmes de transport

Planning du cours

- Complexité : 1 cours
- Ordonnancement : 4 cours, 4 TP
- Transport : 4 cours, 4 TP

	Cours	thème	TP 2F2	TP 2F3	thème
S37	14/09/10	complexité			
S38	21/09/10	ordo			
S39	28/09/10	ordo	28/09/10	01/10/10	ordo
S40	05/10/10	ordo	05/10/10	08/10/10	ordo
S41	12/10/10	ordo	12/10/10	15/10/10	ordo
S42	19/10/10	transport	19/10/10	22/10/10	ordo
S43	Toussaint				
S44	02/11/10	transport	02/11/10	05/11/10	transport
S45	09/11/10	transport	09/11/10	12/11/10	transport
S46	16/11/10	transport	16/11/10	19/11/10	transport
S47			23/11/10	26/11/10	transport
S49			30/11/09	02/12/09	transport

1. Complexité Algorithmique

- Motivations
- Définitions
- Principales classes de complexité
- Calcul de complexité

1.1 Motivations

- Concevoir un algorithme, c'est bien
- Connaître les limites de l'algo, c'est mieux !

- Limites :
  - Préconditions
  - Postconditions
  - Temps de calcul
  - Taille mémoire
  - Communications

1.1 Motivations

- Efficacité d'un algorithme
  - Les performances sont importantes lorsque l'algo constitue un goulot d'étranglement du système
    - Volumétrie importante (grande taille de données)
    - Nombre élevé d'appels (algo utilisé souvent)
    - Contraintes sur le temps de réponse
  - Contextes typiques
    - Systèmes embarqués, temps-réel
    - Logiciels de calcul
    - Fouille de données

1.1 Motivations

- Comparaison d'algorithmes
  - Plusieurs algos pour résoudre le même problème
  - Lequel est le meilleur ?
  - Selon quel critère ?
  - Dans quel contexte ?
  - Sur quel type de données ?

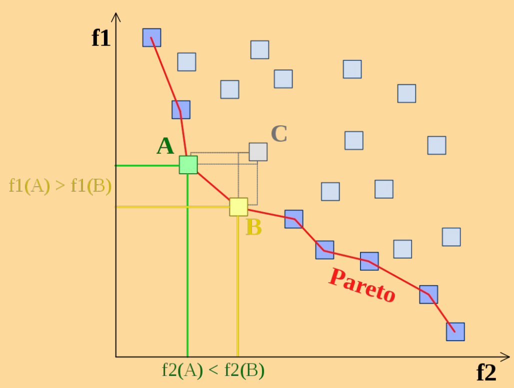
1.1 Motivations

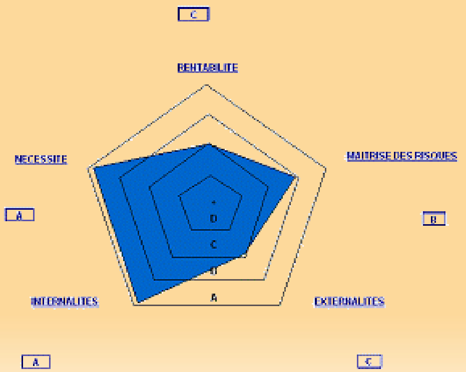
- Analyse a priori
  - Formulation mathématique pour le critère
  - + meilleure compréhension de l'algo
  - - obtention parfois complexe
- Analyse a postériori
  - Outils de profilage, de tests (unitaires, couverture...)
  - + utilisation simple
  - - algorithme = boîte noire, délai d'obtention
- Ici, analyse a priori

1.1 Motivations

- Trouver des critères d'analyse pertinents
  - Indépendants de la plateforme, du langage
  - Discriminants
  - Permettant une analyse rigoureuse

- Possibilité d'analyse multicritères
  - Dominance
  - Front de Pareto
  - Agrégation de critères...





1.1 Motivations

- Critères d'analyse
  - Temps de calcul → nombre d'opérations effectuées
  - Taille mémoire → quantité de mémoire consommée
  - Communications → nombre d'octets émis/reçus

- Typiquement : temps de calcul
- On va s'intéresser à la complexité temporelle des algorithmes

1.2 Définitions

- Analyse du temps de calcul : complexité temps
  - Nombre d'opérations élémentaires effectuées en fonction des données du problème
- Nécessite de définir
  - Une opération élémentaire
  - La taille du problème

1.2 Définition

- Opération élémentaire
  - Opération caractéristique de l'algorithme étudié
  - Exemples
    - Recherche dans un tableau (ZZ1) : accès, comparaison
    - Tri d'un tableau (ZZ1) : comparaison, déplacement
    - Produit scalaire : multiplication
    - Minimisation : évaluation d'un point, calcul du gradient
    - Calcul du PGCD : division, modulo
    - Enveloppe convexe : test d'un point

1.2 Définition

- Taille du problème
  - Grandeur caractéristique utilisée par l'algorithme

  - Exemples
    - Recherche dans un tableau (ZZ1) : taille du tableau
    - Tri d'un tableau (ZZ1) : taille du tableau
    - Produit scalaire : dimension
    - Minimisation : -
    - Calcul du PGCD : taille des nombres
    - Enveloppe convexe : dimension, nombre de points

1.2 Définition

- Trois types de complexité
  - Dans le pire des cas
    - Instance pathologique pour l'algo
    - Utilisé le plus souvent car donne une borne supérieure
  - Dans le meilleur des cas
    - Instance ”parfaite” pour l'algo
  - En moyenne
    - Pas d'hypothèse sur l'instance
    - Plus compliquée à calculer
    - Donne une idée plus proche du comportement réel

1.2 Définitions

- La complexité est donc une formule en fonction de la taille du problème
- Elle peut être compliquée
  - Exemple : $f(n, m) = 3\mathrm{n} + m^2 - 5m/n + 2\log(n+3) + m^{5/n}$
  - Précis mais peu pratique
- Ordre de grandeur
  - Expression simplifiée, montrant l'évolution
  - Même comportement limite que la formule
  - Simplifie les comparaisons et analyses

1.3 Classes de complexité

- Ordre de grandeur
  - Notation o : $f(x)=o(g(x))$ si $\lim_{x\to\infty}\frac{f(x)}{g(x)}=0$
    - f croît plus lentement que g
    - Exemples
      - $\sin(x)=o(x)$ $\quad 17x^2+\sqrt{3x-4}=o(x^4)$
  - Notation O : $f(x)=O(g(x))$ si $\exists k \, tq \, \forall x>x_0, f(x)<k\,g(x)$
    - f dominée par g à partir d'un certain point
    - Exemples
      - $\sin(x)=O(x)$ $\quad \sin(x)=O(1)$
  - O moins précis et plus simple à obtenir que o
  - O est suffisant en général

1.3 Classes de complexité

- Notations
  - Notation ~ : $f(x) \sim g(x)$ si $\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1$
    - f et g sont globalement similaires
    - Exemples
      - $17x^2 + \sqrt{3\mathrm{x} - 4} \sim 17x^2$
      - on se concentre sur le terme dominant
      - ne fonctionne pas pour les fonctions trigonométriques (Taylor !)

  - Autres notations
    - Notation Θ
    - Notation Ω

1.3 Classes de complexité

- On va rester sur O
  - On va chercher la fonction g la plus ”approchante”
    - $\exists k \, tq \, \forall x > x_0, f(x) < k\, g(x)$ et $\lim_{x \to \infty} \frac{f(x)}{g(x)} = k$
    - on se concentre sur le terme dominant dans f
    - on essaie de l'approcher au mieux
  - Exemple
    - $17x^2 + \sqrt{3\text{x} - 4} = O(x^2)$
    - $\sin(x) = O(1)$

1.3 Classes de complexité

- Principales classes de complexité
  - $O(1)$ constante
  - $O(\log x)$ superlinéaire (logarithmique, etc.)
  - $O(x)$ linéaire
  - $O(x \log x)$ n-log-n
  - $O(x^2)$ quadratique
  - $O(x^3)$ cubique
  - $O(2^x)$ exponentielle
  - $O(x!)$ factorielle

1.3 Classes de complexité

- Évolution de la quantité de calculs

n	$\log n$	n	$n \log n$	n^2	n^3	2^n	$n!$
10	$3,3$	10	$3,3 \times 10$	10^2	10^3	10^3	$3,6 \times 10^6$
10^2	$6,6$	10^2	$6,6 \times 10^2$	10^4	10^6	$1,3 \times 10^{30}$	$9,3 \times 10^{157}$
10^3	10	10^3	$1,0 \times 10^4$	10^6	10^9		
10^4	13	10^4	$1,3 \times 10^5$	10^8	10^{12}		
10^5	17	10^5	$1,7 \times 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2,0 \times 10^7$	10^{12}	10^{18}		

1.3 Classes de complexité

- Évolution du temps de calcul
  - Un problème de taille n
  - Un ordinateur d'une puissance de 10^9 flops

n	10^2	10^3	10^6
$O(n)$	$<1\ s$	$<1\ s$	$<1\ s$
$O(n^3)$	$<1\ s$	$1\ s$	$32\ ans$
$O(n^5)$	$10\ s$	$11\ jours$	$3.10^{13}\ ans$
$O(2^n)$	$3.10^{14}\ ans$	$10^{281}\ siècles$	$10^{3.10^5}\ siècles$

1.3 Classes de complexité

- Remarques
  - Pour des problèmes de petite taille, les termes constants ne doivent pas être négligés
  - Exemple
    - $0{,}00001\,n^3$ *vs.* n^2
  - Mais on compare toujours sur l'asymptotique...

- Conséquence
  - Rechercher l'algo offrant la meilleure complexité
  - Techniques pour réduire la complexité

1.4 Calcul de complexité

- Calcul de la complexité
  - Évaluer le nombre d'opérations élémentaires
    - Considérer chaque étape de l'algorithme
    - Faire la somme des appels
  - Déduire l'ordre de grandeur

  - Nécessite de connaître quelques formules
    - Sommations
    - Récurrences
    - Limites

1.4 Calcul de complexité

- Rappels sur les séries

1.4 Calcul de complexité

- Rappels sur les relations de récurrence

1.4 Calcul de complexité

- Rappels sur les limites

1.4 Calcul de complexité

- Exercices
  - Addition, multiplication matricielle
  - Recherche linéaire, dichotomique, interpolation
  - Tri insertion, sélection, rapide, histogramme

2. Conception d'Algorithmes

3. Problèmes NP-complets

- Complexité d'un problème
  - Souvent plusieurs algorithmes possibles
  - Complexité du meilleur algorithme associé
  - Exemples :
    - Recherche dans un tableau
      - O(n) si pas trié
      - O(log n) si trié
    - Tri dans un tableau
      - O(n lon g)
    - Résolution de systèmes linéaires
      - $O(n^2)$

3. Problèmes NP-complets

- Complexité d'un problème
  - Permet de connaître la classe d'un problème
  - Prouver qu'on ne peut faire mieux est complexe
  - Beaucoup de problèmes ouverts
    - Multiplication matricielle
    - Test de primalité
    - Programmation linéaire
    - Problèmes combinatoires...

3. Problèmes NP-complets

- Au final, 2 catégories de problèmes
  - Les problèmes ”faciles”
    - faible complexité : $O(1) \rightarrow O(n^k)$
    - existence d'algorithmes efficaces (pour k petit !)
  - Les problèmes ”difficiles”
    - forte complexité : $O(2^n)$, $O(n!)$
    - absence d'algorithmes efficaces
    - souvent avec une structure combinatoire

3. Problèmes NP-complets

- Problèmes de décision
  - Leur réponse est oui / non
  - Utilisés pour la définition des classes P et NP
  - Exemples
    - PREMIER(a) : le nombre a est-il un premier ?
    - CONNEXE(G) : G est-il connexe ?
    - PCC(G,p) : le chemin p est-il le plus court ?
  - Problèmes d'optimisation traités ”par la bande”

3. Problèmes NP-complets

- Classe **P** (polynômial)
  - Problèmes de décision qui peuvent être résolus par un algorithme fortement polynômial
  - Fortement polynômial = indépendant de la grandeur
  - Exemples
    - CONNEXE(G) : algo de parcours BFS
    - CHEMIN(G,o,d) : algo de parcours BFS
    - PREMIER(a) : algo de Aggarwal *et al.*

3. Problèmes NP-complets

- Classe **NP** (polynômial non-déterministe)
  - Problèmes de décision qui peuvent être résolu par un algorithme non-déterministe fortement polynômial
  - Définition plus compliquée que pour **P**
  - À chaque étape, l'algorithme a plusieurs choix
  - Réponse ”oui” : il existe une séquence de branchements tels que l'algorithme trouve
  - Réponse ”non” : quelque soit la séquence de branchements, l'algorithme ne trouve pas

3. Problèmes NP-complets

- Classe **NP**
  - Exemples
    - HAMILTONIEN(G,o,d) : existe-t-il un chemin hamiltonien de o à d dans G ?
    - COLORATION(G,k) : G peut-il être colorié avec k couleurs ?
    - SAT(f,x) : existe-t-il un vecteur x satisfaisant f ?
  - De petites modifications peuvent changer la classe
    - 3-SAT est de classe **NP**, 2-SAT est de classe **P**
    - EULERIEN(G,o,d) est de classe **P**

3. Problèmes NP-complets

- Relation **P** $\leftrightarrow$ **NP**
  - Trivialement **P** dans **NP**
    - ”qui peut le plus, peut le moins”
  - Question ouverte : **P = NP** ?
    - on cherche depuis 30 ans
    - un des 7 problèmes du Millennium Prize
    - à vous de répondre !

3. Problèmes NP-complets

- Classe **Indécidable**
  - Problèmes pour lesquels on ne connait pas on ne connaît pas d'algorithme qui réponde en un nombre fini d'étape
  - Exemple
    - HALT(P) : le programme P se termine-t-il ?
    - DIOPHANTE(E) : l'équation diophantienne E a-t-elle une solution ?

3. Problèmes NP-complets

- Classe **Co-NP**
  - Étant donné un problème, son complémentaire consiste à échanger les réponses
  - Exemple
    - COMPOSE(a) : le nombre a est-il composé ?
    - PREMIER(a) : le nombre a est-il premier ?
    - change radicalement la nature de la question (il existe / quelque soit)
  - Un problème est dans **Co-NP** si le complémentaire est dans **NP**

3. Problèmes NP-complets

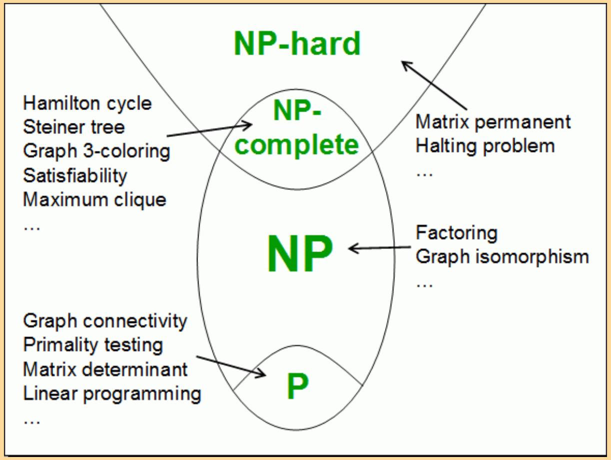
- Réduction
  - Deux problèmes A et P_2
  - Une tranformation f : A → B
    - un algorithme pour B résoud les instances de A
  - Si f est polynômiale
    - réduction polynômiale
    - si B est **P** alors A est **P**
- Classe **NP-difficile**
  - Si tout problème **NP** peut se réduire poly à A

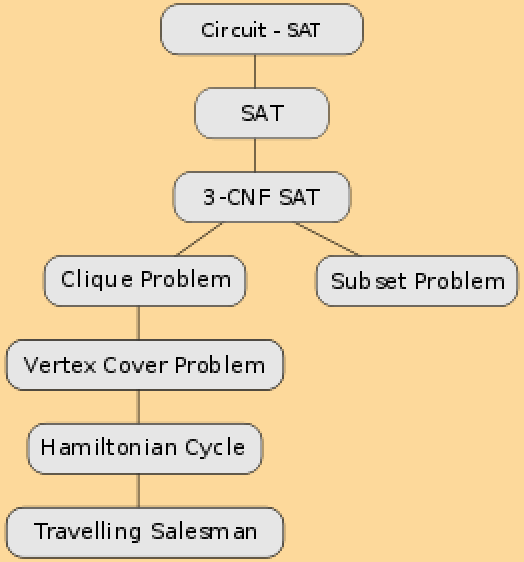
3. Problèmes NP-complets

- Classe **NP-complet**
  - Problème **NP-difficile** qui est dans **NP**
  - Intérêt
    - un algo pour un problème **NP-complet** peut, de fait, résoudre tous les problèmes **NP**
    - s'il existe un algo polynômial pour un problème **NP-complet**, alors **P** = NP
  - Théorème de Cook (1971) : SAT est **NP-complet**
  - Karp (1972) : 21 problèmes **NP-complets**

3. Problèmes NP-complets

- Classe **NP-complet**
  - Karp (1972) : 21 problèmes **NP-complets**
  - Garey & Johnson (1979) : livre-référence





3. Problèmes NP-complets

- SAT(f)
  - Une formule booléenne est une expression (à base de AND, OR, NOT et ()) de variables booléennes
  - Forme Normale Conjonctive
    - conjonction de clauses (disjonctions de litéraux)
    - ex : $f=(\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_2 \vee x_5) \wedge (\neg x_3 \vee x_4 \vee \neg x_5)$
  - Existe-t-il un vecteur satisfaisant la formule ?
    - ex : (F F F V V)

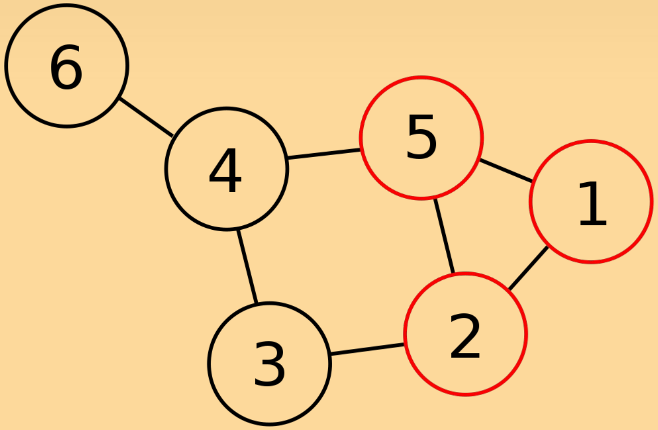
  - Problème fondamental de la classe **NP-complet**
  - Applications pratiques (logique, électronique...)

3. Problèmes NP-complets

- 3-SAT(f)
  - Idem à SAT avec des clauses d'au plus 3 litéraux
  - Exemple : $f(x)=(\neg x_1 \vee x_2 \vee x_3)\wedge(\neg x_2 \vee x_3)\wedge(x_1 \vee x_2 \vee \neg x_3)$
  - Existe-t-il un vecteur satisfaisant la formule ?
    - Exemple : (F F V)
  - Par contre 2-SAT est dans la classe **P**

3. Problèmes NP-complets

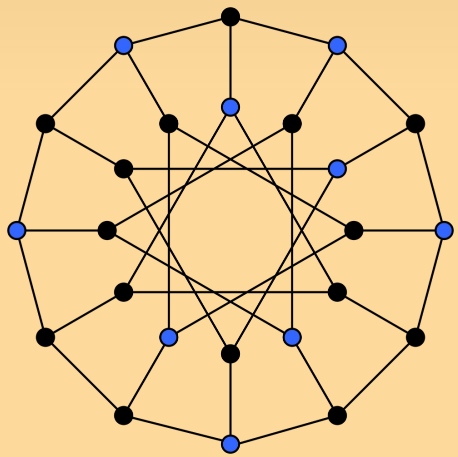
- CLIQUE(G,k)
  - On considère un graphe non orienté G=(V,E)
  - Sous-graphe complet



  - Existe-t-il une clique de cardinalité k ?
  - Par la suite, trouver une clique de card. maximum

3. Problèmes NP-complets

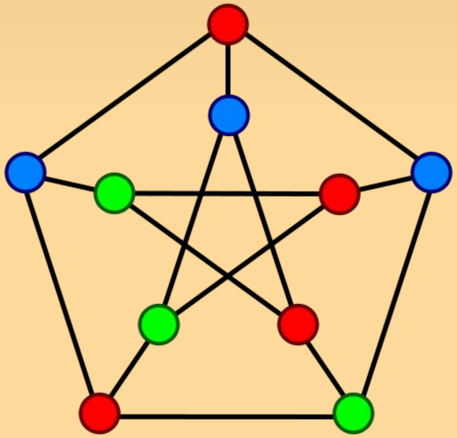
- STABLE(G,k)
  - On considère un graphe non orienté G=(V,E)
  - Ensemble de sommets 2 à 2 non adjacents



  - Existe-t-il un stable de cardinalité k ?
  - Aussi, trouver un stable de cardinalité maximum
  - Le stable et la clique sont complémentaires

3. Problèmes NP-complets

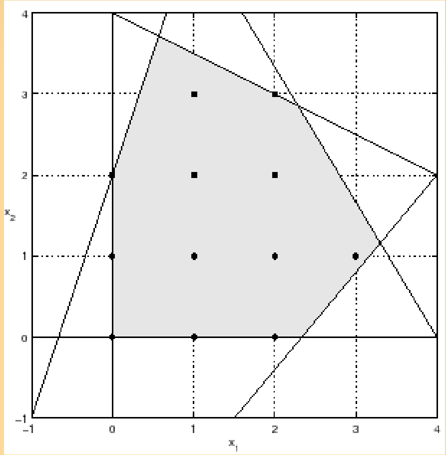
- COLORATION(G,k)
  - On considère un graphe non orienté G=(V,E)
  - Affecter une couleur à chaque sommet
    - deux sommets adjacents ont une couleur ≠



  - Existe-t-il une coloration avec k couleurs ?
  - Aussi, trouver le nombre minimal de couleurs
  - Note : 4 couleurs suffisent pour un graphe planaire

3. Problèmes NP-complets

- PLNE 0-1
  - On considère un PL avec des variables de décision
    - $x_i = 0 \rightarrow$ on ne prend pas la décision i
    - $x_i = 1 \rightarrow$ on prend la décision i



  - Le système admet-il une solution ?
  - Par la suite, trouver une solution optimale

  - PL est **P**, PLNE 0-1 est **NP-complet**

3. Problèmes NP-complets

- SUBSET-SUM
  - On a un ensemble de n entiers $E=\{e_1..e_n\}$
  - On a un entier S
  - Existe-t-il une partie de E non vide de somme S ?
  - Exemple : E={-3, -1, 2, 6}
    - pour S=0, pas de solution
    - pour S =1, E'={-1,2}
    - pour S=2, E'={2} ou {-3,-1,6}

3. Problèmes NP-complets

- SAC-A-DOS(c,p,P)
  - On considère n objets d'intérêt ci et de poids pi
  - On a une limite de poids total P
  - Existe-t-il une combinaison d'objets de poids P ?
  - Aussi, trouver un ensemble viable d'intérêt max
  - Exemple : O={(1,2)(8,10)(7,3)(4,1)}
    - P=12, O'={(1,2)(7,3)(4,1)}, intérêt = 12, poids = 6

3. Problèmes NP-complets

- Liens entre SUBSET-SUM et SAC-A-DOS
  - SAC-A-DOS peut résoudre SUBSET-SUM
    - On garde la contrainte de capacité
    - L'intérêt d'un entier est sa valeur
    - Si la solution de SAC-A-DOS sature la contrainte alors c'est une solution de SUBSET-SUM
    - Sinon SUBSET-SUM n'a pas de solution
  - SUBSET-SUM est un problème d'existence
  - SAC-A-DOS est un problème d'optimisation

3.1 Méthodes exactes

- Méthodes d'énumération
  - Forte complexité
  - Limitées à certains problèmes et à certaines tailles
- Programmation Linéaire en Nombres Entiers
  - Reprend le formalisme de la PL
  - Considère des variables entières, modélisant
    - des quantités entières
    - des décisions

3.1 Méthodes exactes

- Variables de décision $x_i \in \{0;1\}$
  - Ne peuvent prendre que deux valeurs : 0 ou 1
    - $x_i = 1$ on prend la décision i
    - $x_i = 0$ on ne prend pas la décision i
  - Modélise ”simplement” des situations booléennes
  - Toute variable entière peut s'exprimer en décision
    - $x \in \mathbb{N} \rightarrow x = \sum_{i=1\ldots k} 2^i y_i$ avec $y_i \in \{0,1\}$

3.1 Méthodes exactes

- Résolution d'un PLNE
  - Branch & Bound (séparation / évaluation)
  - On relâche le problème des contraintes d'intégrité
  - On résoud le problème relâché
  - Si la solution est fractionnaire
    - il existe au moins une variable fractionnaire
    - on effectue un branchement dessus
    - stratégie récursive

3.1 Méthodes exactes

- Stratégies évoluées
  - Branch & Cut (génération de coupes)
  - Branch & Price (génération de colonnes)
  - Relaxation lagrangienne
  - Méthodes de décomposition
    - Dantzig-Wolfe
    - Benders
    - décomposition lagrangienne, ...
- Programmation dynamique, ...

3.2 Méthodes approchées

- Les problèmes sont de classe NP
  - Temps de calcul explose sur les grosses instances
  - Se tourner vers des méthodes approchées
    - on ne garantit plus l'optimalité de la solution
    - on réduit fortement les temps de calcul
    - compromis entre le temps et la qualité

4. Problèmes d'ordonnancement

5. Problèmes de transport

- Transport de biens / personnes
-
- Composante essentielle de la société moderne
  - Charbon puis pétrole : essor du transport moderne
    - plus rapide : réduction des délais de transport
    - plus loin : accès à de nouveaux marchés, délocalisation
    - plus sûr : maîtrise des aléas
    - davantage : augmentation des quantités transportées
  - Massification des transports

5. Problèmes de transport

- Environnement fortement concurrentiel
  - Mécanisme de bourse au transport
  - Répondre aux offres de transport
    - Offrir les meilleures prestations de transport
      - Délais, coûts
    - Savoir rapidement si une offre est intéressante
      - Augmentation du bénéfice
      - Respect de la nouvelle demande
      - Respect des demandes existantes

5. Problèmes de transport

- Trois entités en jeu
  - Client : quantité, origine, destination, dates...
  - Véhicule : capacité, vitesse...
  - Réseau : urbain / routier, distances
- Dimensions
  - Nombre de demandes, de véhicules
- Prétraitement
  - Calcul du chemin entre chaque point
  - Réduction au sous-graphe des points

5. Problèmes de transport

- Exemples de problématiques
  - Organiser des tournées de transport (ex : la Poste)
    - collecte, livraison, collecte + livraison
    - gestion de la flotte de véhicules
    - satisfaire tous les clients
  - Organiser des lignes régulières (ex : bus, train)
    - point d'arrêt, fréquence
    - gestion de la flotte de véhicules
    - satisfaire le plus de clients possible

5. Problèmes de transport

- Problèmes de transport en Aide à la Décision
  - De problèmes simples
    - Plus court chemin
    - Problème de transport
    - Flot maximal, flot de coût minimal
  - À des problèmes difficiles
    - Voyageur de commerce
    - Tournées de véhicules
    - Transport à la demande

5. Problèmes de transport

- Plus Court Chemin (PCC)
  - Étant donné un graphe G=(S,A) muni de coût sur les arcs, trouver le plus court chemin d'un sommet origine o à un sommet destination d
  -
  - coût positifs : Dijsktra, Bellman
  - coût négatifs (sans cycle) : Bellman
  - problème polynômial

5. Problèmes de transport

- PCC : Dijsktra
  - 2 ensembles de sommets
    - S_1 : sommets validés
    - S_2 : sommets non traités
  - labels de distance : π

// initialisation
$S_1 \leftarrow \{\}$
$S_2 \leftarrow S$
$\pi_s \leftarrow 0$
$\pi_i \leftarrow \infty, \forall i \neq s$
// boucle d'expansion
tant que $S_2 \neq \{\}$
  $i \leftarrow \operatorname{argmin}_{j \in S_2}\{\pi_j\}$
  $S_1 \leftarrow S_1 \cup \{i\}$
  $S_2 \leftarrow S_2 - \{i\}$
  $\forall (i,j) \in A$
    si $\pi_j > \pi_i + c_{ij}$ **alors**
      $\pi_j \leftarrow \pi_i + c_{ij}$
    fin si
  fin forall
fin tant que

on peut ajouter une structure pour stocker le sommet qui connecte chaque sommet dans l'arbre des PCC

5. Problèmes de transport

- PCC : Bellman
  - L liste des sommets modif
    - propage le label aux voisins
  - labels de distance : π

on peut ajouter une structure pour stocker le sommet qui connecte chaque sommet dans l’arbre des PCC

// initialisation
$\pi_s \leftarrow 0$
$\pi_i \leftarrow \infty, \forall i \neq s$
$L \leftarrow \{s\}$

// boucle de correction
tant que $L \neq \{\}$
  choisir $i \in L$
  $\forall (i, j) \in A$
    si $\pi_j > \pi_i + c_{ij}$ **alors**
      $\pi_j \leftarrow \pi_i + c_{ij}$
      si $j \notin L$ **alors**
        $L \leftarrow L \cup \{j\}$
      fin si
    fin si
  fin forall
fin tant que

5. Problèmes de transport

- Présence de cycle négatif : pas d'optimalité
  - rechercher un PCC élémentaire
  - NP-difficile
  -
- Méthodes exactes
  - Programme linéaires en nombres entiers
  - Programmation dynamique

5. Problèmes de transport

- Problème de Transport
  - Étant donné un graphe biparti G=(S1,S2,A) muni de quantités sur les sommets et de coûts unitaires sur les arcs, trouver le flot de coût minimal permettant de router les entités de S1 vers S2

5. Problèmes de transport

- Problème de Flot Maximal
  - Étant donné un graphe G=(S,A) muni de capacité sur les arcs, trouver le flot maximal d'un sommet origine o à un sommet destination d
  - oroblème polynomial
  - algorithme de Ford-Fulkerson
  - algorithme de push-relabel

5. Problèmes de transport

- Problème de Flot de Coût Minimal
  - Étant donné un graphe G=(S,A) muni de capacité et de coût unitaire sur les arcs, trouver le flot de coût minimal routant q unités d'un sommet origine o à un sommet destination d
  - problème polynomial
  - algorithme de suppression de cycle négatif
  - algorithme de plus courts chemins successifs

5. Problèmes de transport

- Problème du Voyageur de Commerce
  - Travelling Salesman Problem (TSP)
  - Étant donné un graphe G=(S,A) muni de coût sur les arcs, trouver un cycle hamiltonien de coût minimal
  - problème NP-complet
  - méthodes exactes
    - programmation linéaire en nombres entiers, coupes
  - méthodes approchées
    - heuristiques, métaheuristiques

Le TSP

- Un des problèmes phares de la RO
  - Presque toutes les méthodes exactes et approchées ont été testées dessus
  - Permet donc de connaître le potentiel d'une nouvelle méthodes
  - Intérêt multiple
    - industriel : au coeur de presque toutes les problèmatiques de transport
    - académique : un des problèmes complexes le plus simples
    - ludique : simple à comprendre / visualiser

Le TSP

- Exemple
  - Concours dans les années 50

Le TSP

- À la base de nombreuses variantes
  - sur les distances
    - euclidiennes ou non
    - normes L_1, L_2, L_∞
    - symétrique, asymétrique
    - complet ou non
  - sur les contraintes
    - possibilité ou non de virages
    - précédence entre certains noeuds

5. Problèmes de transport

- Problème des Tournées de Véhicules
  - Étant donné un graphe G=(S,A) muni de coût sur les arcs,

5. Problèmes de transport

- Problème de Transport à la Demande
  - Étant donné un graphe G=(S,A) muni de coût sur les arcs,