Méthodes et modélisation pour l'optimisation

M1 informatique

2016–2017

INSTITUT D'ÉLECTRONIQUE
ET D'INFORMATIQUE
GASPARD-MONGE

Trouver des informations

- www-igm.univ-mlv.fr/~thapper/teaching/mmpo/
  - transparences
  - feuilles de td et de tp

- Responsable du cours
  - Johan Thapper <thapper@u-pem.fr>

Organisation

- Cours 6 x 2h
  - Johan Thapper
- TD/TP 6 x 2h
  - Johan Thapper — Groupe 2
  - Alfredo Hubard — Groupe 1

Contrôle des connaissances

- 1 examen (janvier)
  - sur les notions abordées aux cours, TD et TP
  - 1 feuille recto-verso manuscrite autorisée
  - téléphone, calculatrice, machine interdits
- Pour réussir à l'examen
  - Présence (pas seulement physique) au cours
  - Travail aux TD et TP indispensable

Acquis supposés

- Algèbre linéaire
  - élimination de Gauss
- Vocabulaire de la logique propositionnelle
  - variable Booléenne
  - forme normale conjonctive (CNF)
- Vocabulaire de la théorie des graphes
- Complexité (notions)
  - ordres de grandeur — temps polynomial et exponentiel
  - problèmes NP-difficiles — SAT, coloration de graphe, …

Modélisation et résolution d'un problème

Une entreprise veut installer la fibre optique entre ses bâtiment **A**, **B**, **C**, **D** et **E**. Le coût de chaque lien potentiel (point à point) est donné dans le tableau suivant :

	A	B	C	D	E
A		10	15	10	
B	10			7	18
C	15			15	
D	10	7	15		20
E		18		20	

Il suffit d'avoir un chemin entre chaque paire de bâtiments.

Minimiser le coût total de l'installation.

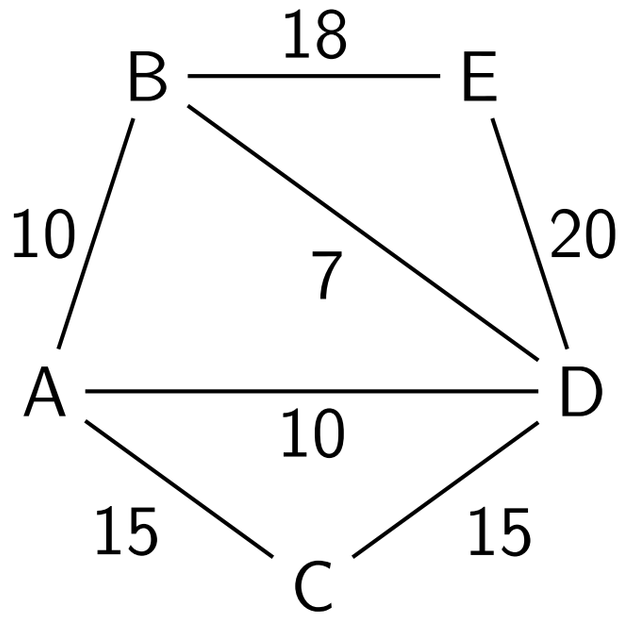
Problème concret

minimiser le coût

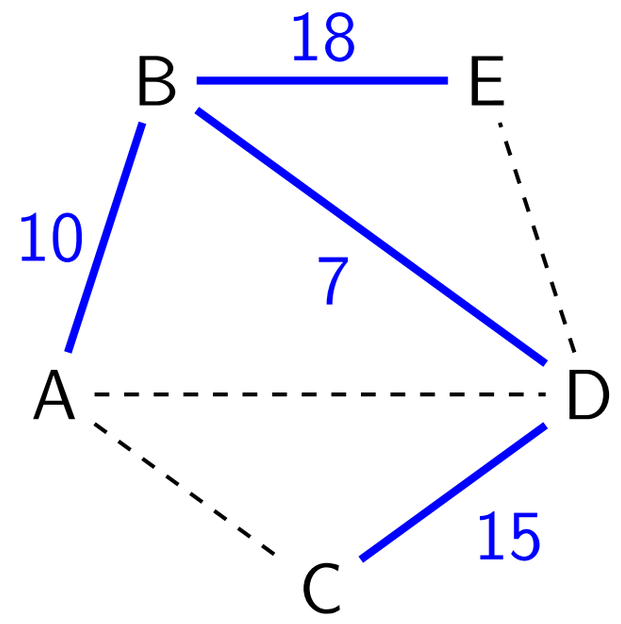
	A	B	C	D	E
A		10	15	10	
B	10			7	18
C	15			15	
D	10	7	15		20
E		18		20	

Problème abstrait

trouver un arbre couvrant minimal



algorithme : Kruskal



poser des lignes entre :

- A et B
- B et D
- B et E
- C et D

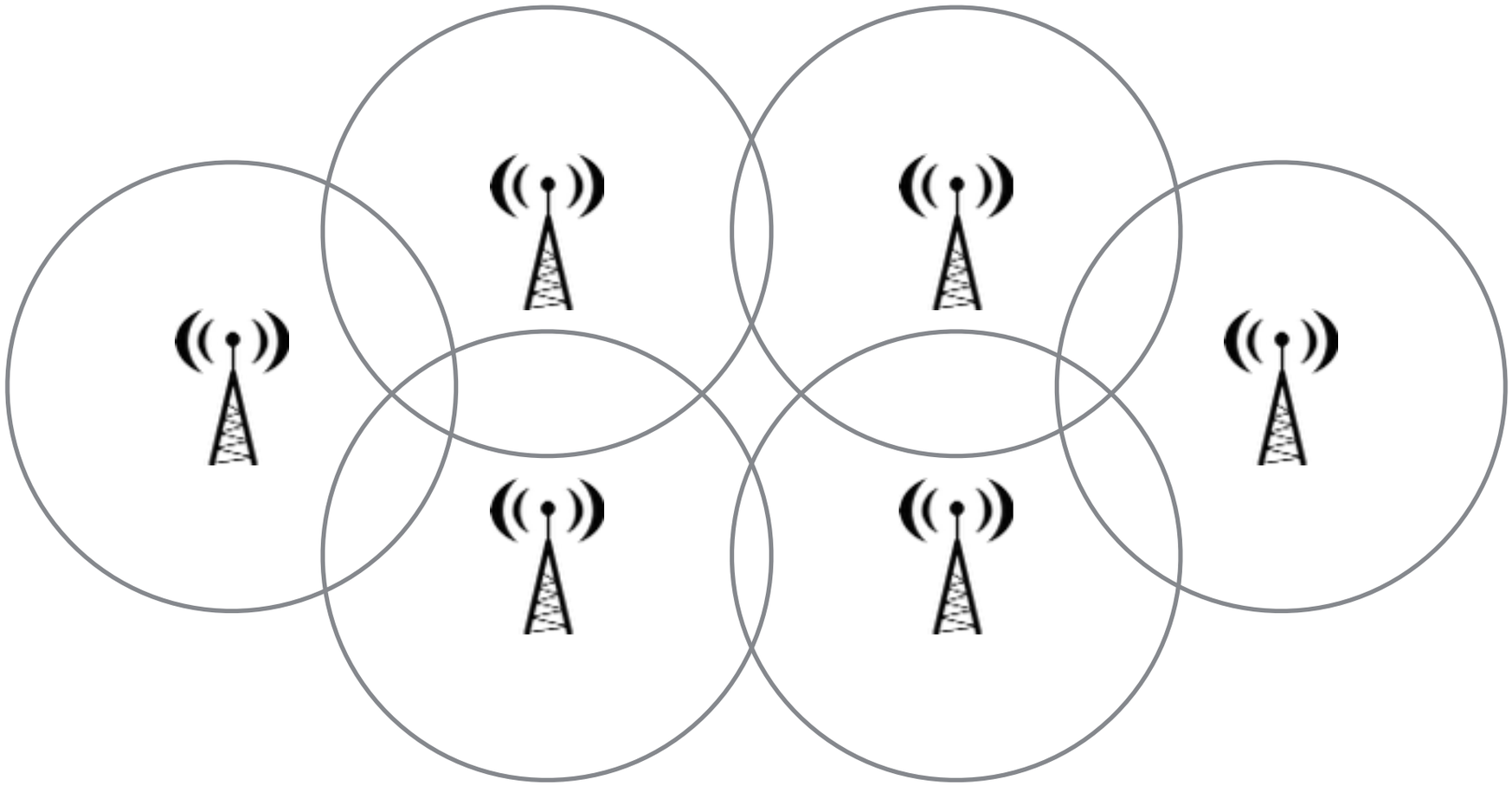
coût total : 10+7+18+15 = 50

Modélisation et résolution d'un problème

- Modélisation
  - Traduire un **problème concret** A (sous forme verbale) en un **problème abstrait** B (sous forme symbolique)
- Résolution
  - Résoudre B avec un algorithme / logiciel connu
- Récupération d'une solution
  - Récupérer une solution de B et l'interpreter comme une solution de A

Exemple : affectation de fréquence

Un nombre d'émetteurs est localisé comme dans la figure suivante :

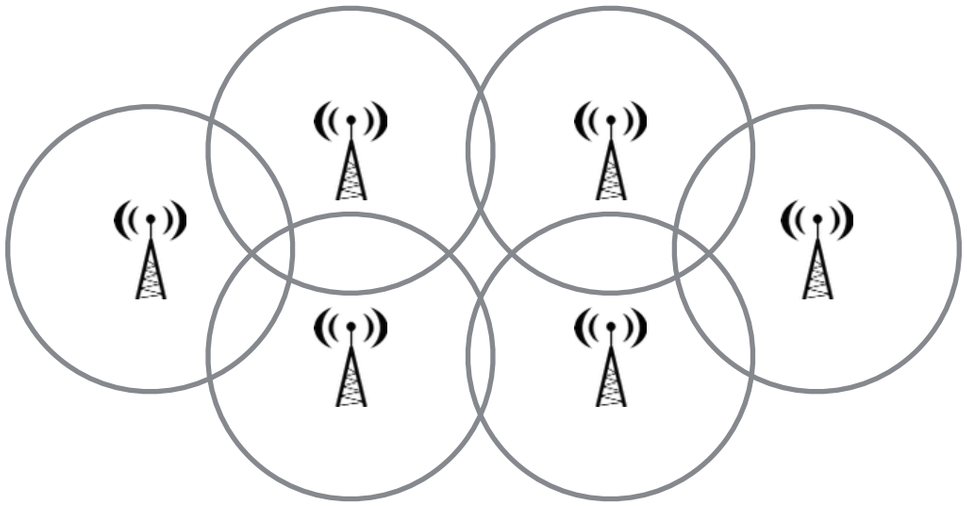


Si deux émetteurs sont trop proches, ils ne peuvent pas transmettre sur la même fréquence à cause d'interférences.

Minimiser le nombre de fréquences utilisées.

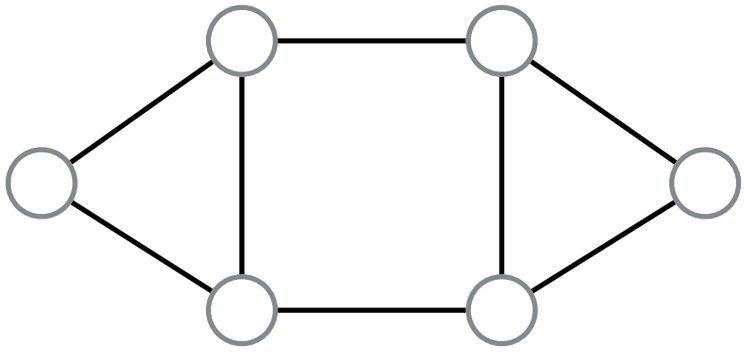
Problème concret

minimiser le nombre de fréquences



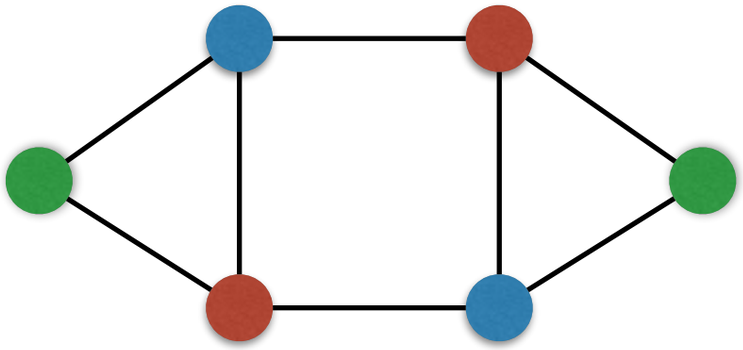
Problème abstrait

trouver une coloration de graphe
avec un nombre minimal de couleurs



sommet = émetteur

arête = interférence



couleur = fréquence

Solution

affectation :

fréquence 1 (X Mhz)

fréquence 2 (Y Mhz)

fréquence 3 (Z Mhz)

nombre minimal : 3

Exemple : emploi du temps

On veut affecter 6 tâches, T1, …, T6, aux créneaux horaires. Certaines tâches ne peuvent pas être affectées au même créneau à cause de conflits. Par exemple, deux tâches qui nécessitent la même machine.

Les conflits sont décrits dans le tableau suivant :

tâche	en conflit avec
T1	T2, T3, T5
T2	T1, T4, T6
T3	T1, T4, T5
T4	T2, T3, T6
T5	T1, T3
T6	T2, T4

Affecter les tâches en minimisant le nombre de créneaux horaires utilisés.

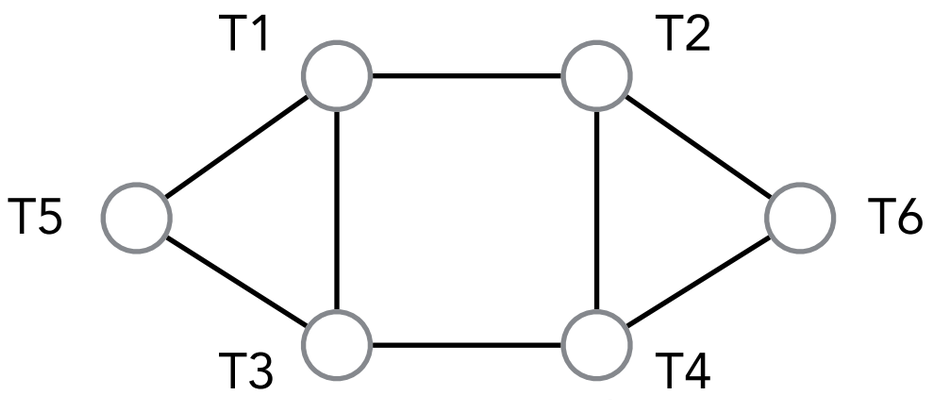
Problème concret

minimiser le nombre de créneaux

tâche	en conflit avec
T1	T2, T3, T5
T2	T1, T4, T6
T3	T1, T4, T5
T4	T2, T3, T6
T5	T1, T3
T6	T2, T4

Problème abstrait

trouver une coloration de graphe
avec un nombre minimal de couleurs



sommet = tâche

arête = conflit

Solution

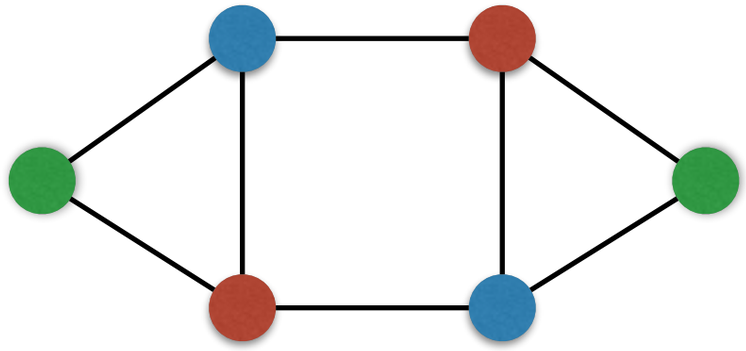
affectation :

créneau 1 : T1, T4

créneau 2 : T5, T6

créneau 3 : T2, T3

nombre minimal : 3



couleur = créneau

Modélisation

1. Décider d'un vocabulaire de modélisation
   - graphes, géométrie, logique, …
2. Trouver les variables
   - Souvent l'étape la plus difficile — il faut faire plusieurs exercices
   - Facilite la prochaine étape
3. Introduire des contraintes qui modélisent le problème
   - Vérifier que les solutions au problème B correspondent aux solutions au problème A
   - Sinon, vous n'avez pas les bonnes variables / contraintes

Exemple : programmation linéaire

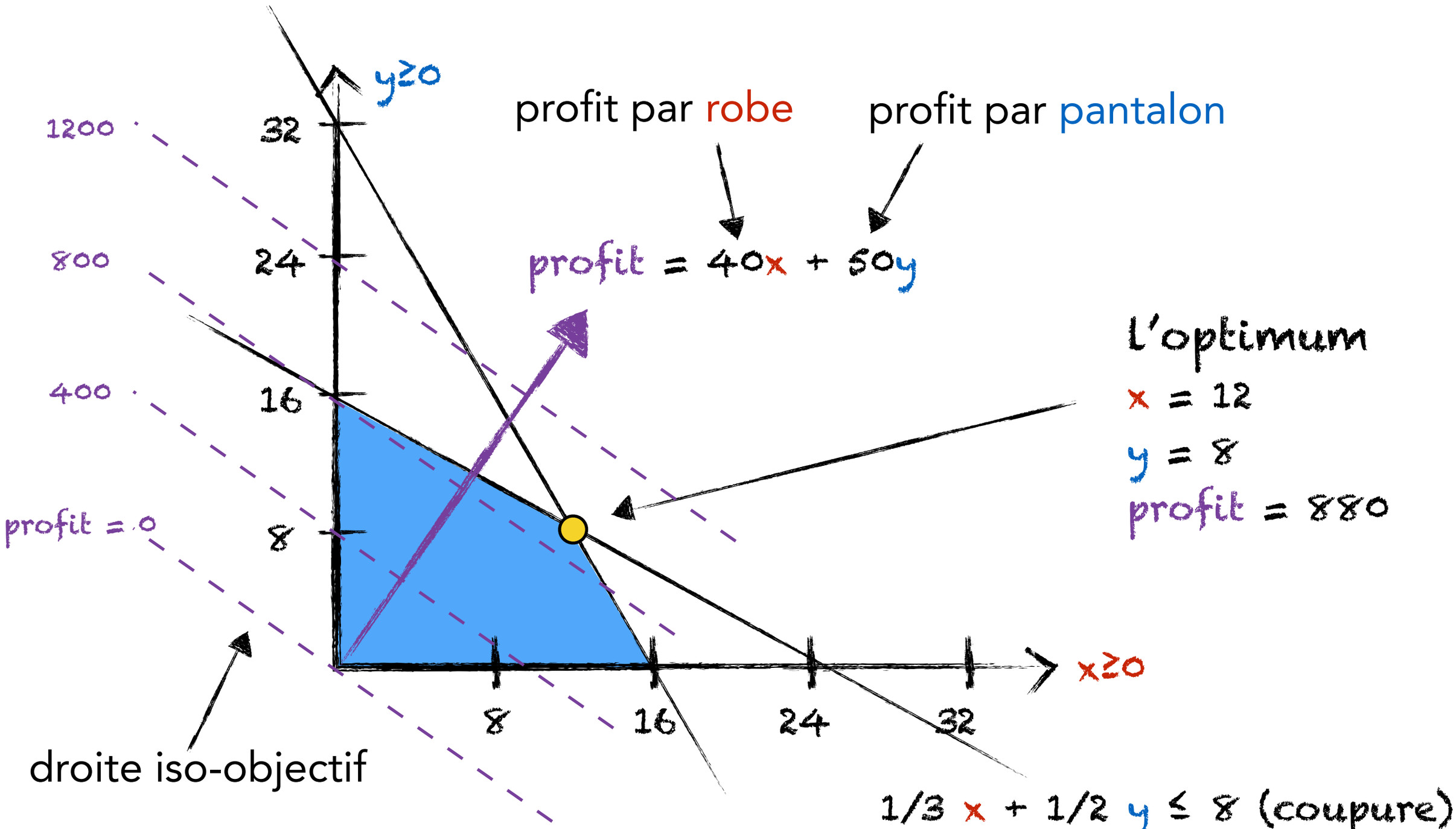
Une petite entreprise fabrique des robes et des pantalons. Elle dispose de deux machines : une machine pour couper le tissu et une machine pour la couture. Pour faire une robe, il faut une demi heure pour couper le tissu et 20 minutes pour la couture. Pour faire un pantalon, il faut 15 minutes de coupure et une demi heure pour la couture. Le profit d'une robe est de 40 euros et pour un pantalon 50 euros. L'entreprise fonctionne 8 heures par jour.

Maximiser le profit par jour.

Robes et pantalons — modélisation

1. Vocabulaire de modélisation
   - variables réelles et inégalités linéaires
2. Trouver les **variables**
   - Soit x le nombre de robes et y le nombre de pantalons par jour
3. Exprimer les **contraintes** en inégalités
   - x et y sont entiers et non-négatifs : $x, y \geq 0$
   - On coupe au plus 8 heures par jour : $1/2\ x + 1/4\ y \leq 8$
   - On coud au plus 8 heures par jour : $1/3\ x + 1/2\ y \leq 8$

$1/2\ x + 1/4\ y \leq 8$ (couture)



x — nombre de robes par jour

y — nombre de pantalons par jour

l'ensemble de solutions

La programmation linéaire

- Entrées
  - variables : $x_1, x_2, x_3, \ldots$
  - fonction objectif (linéaire) : $\max \quad x_1 + x_2 + 5x_3$
  - inégalités (linéaires) : $x_1 + 7x_2 - 5x_3 \leq 3$
- Sortie
  - une solution optimale : $x_1 = 3, x_2 = 0, \ldots$
  - ``pas de solution'' — s'il n'en existe aucune

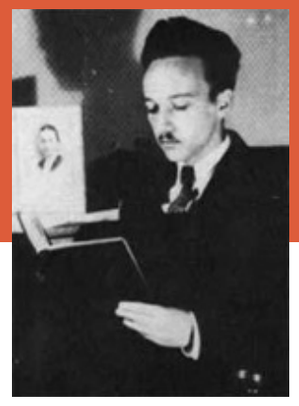
Résolution efficace en théorie et en pratique

- ***L'algorithme du simplexe*** (Dantzig 1947)
  - efficace et utilisé en pratique
  - exponentiel dans le pire des cas

- Algorithmes de complexité polynomiale
  - méthode de l'ellipsoïde (théorie)
  - méthodes de points intérieurs (théorie et pratique)

- Nous utilisons le solveur **lp_solve**
  - il y a un package pour votre distribution Linux préférée

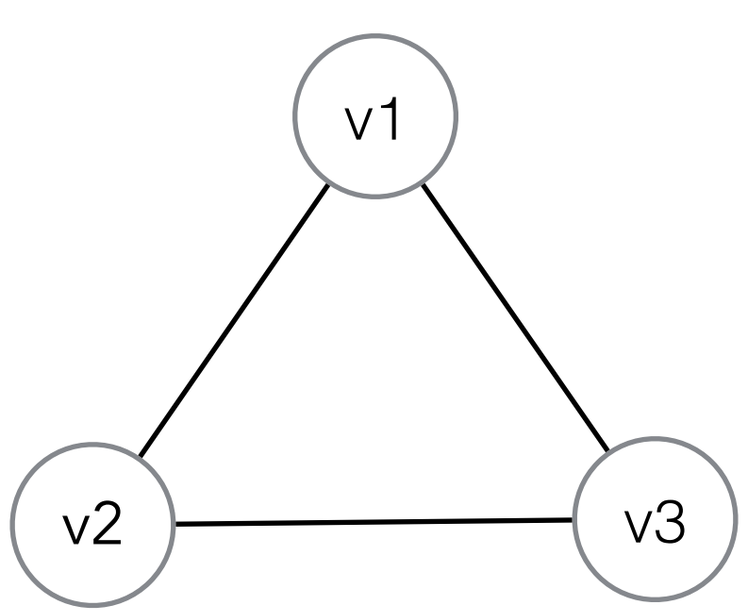
Un peu d'histoire



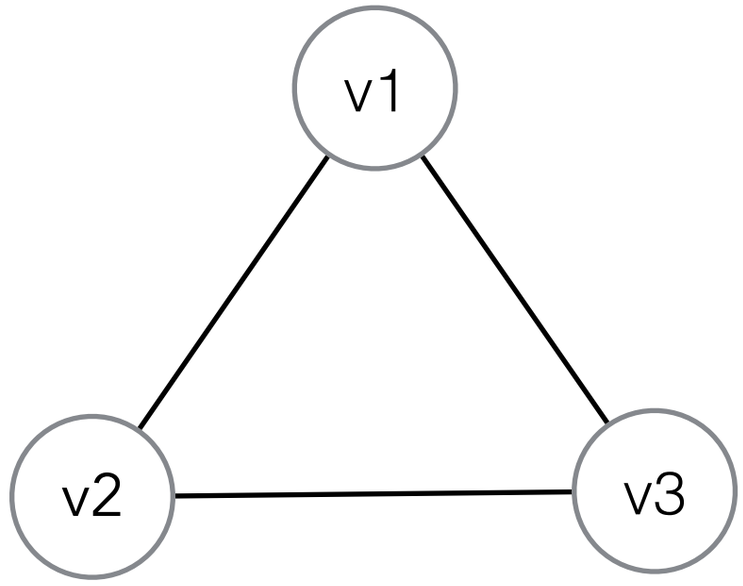
- Kantorovitch (~1939, URSS), Dantzig (1947, États-Unis)
- "Programmation" = planification
  - le terme date des années 1950 (e.g. programmation musicale)
- Applications importantes
  - planification, logistique, contrôle de la production dans de nombreux domaines : télécom, transport aérien, …
- Kantorovitch lauréat du *"Prix de la Banque de Suède en sciences économiques en mémoire d'Alfred Nobel"* (1975)
  - domaine : théorie de l'allocation optimale des ressources

Exemple : satisfaisabilité booléenne

Est-ce qu'il est possible de trouver une coloration du graphe ci-dessous avec deux couleurs telle que deux sommets partageant une arrête n'ont pas la même couleur ?



Triangle — modélisation



1. Vocabulaire de modélisation
   - variables booléennes et clauses CNF
2. Trouver les **variables**
   - la variable $x_{i,0}$ indique que le sommet vi prend la couleur 0
   - la variable $x_{i,1}$ indique que le sommet vi prend la couleur 1
3. Exprimer les **contraintes** en logique propositionnelle
   - un sommet doit prendre une couleur : $x_{i,0} \vee x_{i,1}$
   - un sommet ne peut prendre qu'une couleur : $\neg x_{i,0} \vee \neg x_{i,1}$
   - deux sommets ne prennent pas la même couleur : $\neg x_{i,0} \vee \neg x_{j,0}$

     $\neg x_{i,1} \vee \neg x_{j,1}$

$$x_{1,0} \vee x_{1,1} \ , \ x_{2,0} \vee x_{2,1} \ , \ x_{3,0} \vee x_{3,1}$$ *au moins* une couleur

$$\neg x_{1,0} \vee \neg x_{1,1} \ , \ \neg x_{2,0} \vee \neg x_{2,1} \ , \ \neg x_{3,0} \vee \neg x_{3,1}$$ *au plus* une couleur

$$\neg x_{1,0} \vee \neg x_{2,0} \ , \ \neg x_{1,1} \vee \neg x_{2,1}$$ v1 et v2 différentes

$$\neg x_{2,0} \vee \neg x_{3,0} \ , \ \neg x_{2,1} \vee \neg x_{3,1}$$ v2 et v3 différentes

$$\neg x_{3,0} \vee \neg x_{1,0} \ , \ \neg x_{3,1} \vee \neg x_{1,1}$$ v3 et v1 différentes

logiciel

Non simultanément satisfaisables !

Donc, pas de coloration du graphe avec deux couleurs

Satisfaisabilité booléenne

- Entrées
  - variables : $x_1, x_2, x_3, \ldots$
  - clauses : $x_1 \vee x_2$

    $\neg x_1 \vee x_2 \vee \neg x_3$
- Sortie
  - une solution satisfaisante $x_1 = \texttt{true}, x_2 = \texttt{false}, \ldots$
  - ``non satisfaisable" — s'il n'en existe aucune

Résolution quasi-efficace en pratique

- Il n'y a pas d'algorithme polynomial pour SAT
  - Il n'en existe aucun, si P ≠ NP !
- Mais il y a des solveurs qui fonctionnent bien en pratique

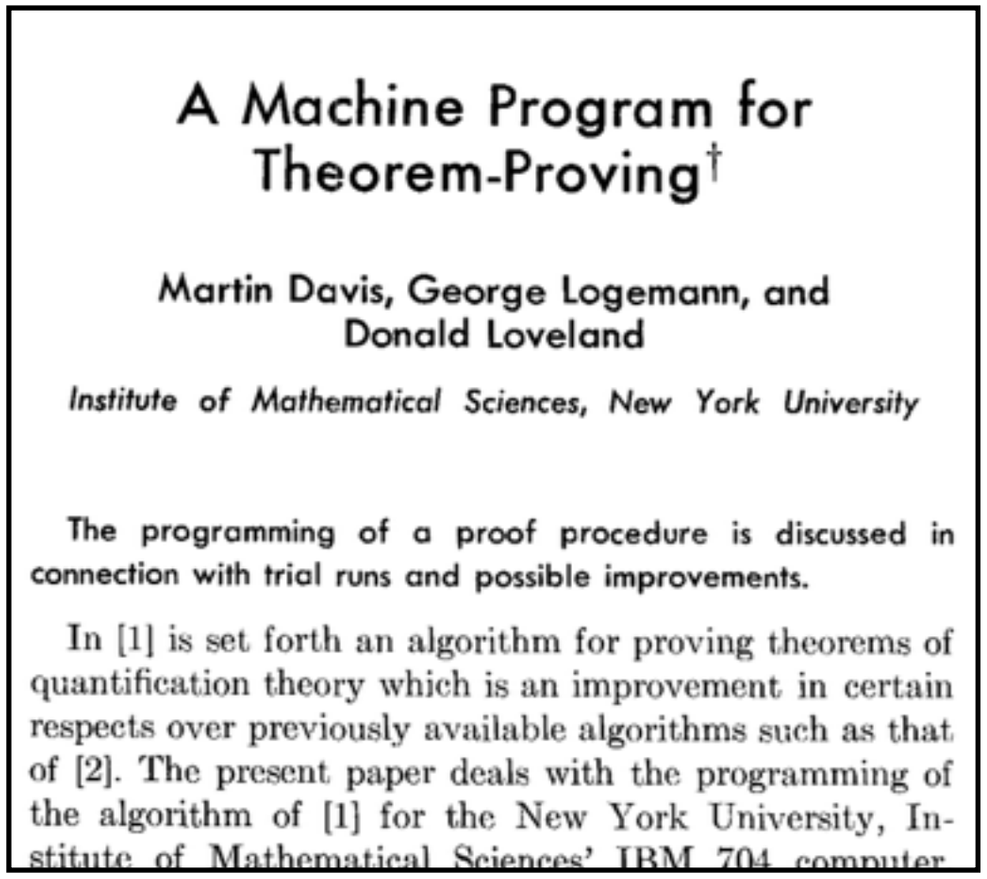
années	variables	clauses
1960-70	10	100
1980-95	100	1 000
1995-00	1 000	100 000
2000-10	100 000	1 000 000
2010-	> 1 000 000	> 5 000 000

- Nous utilisons le solveur **minisat**

```
$ sudo apt-get install minisat
```

Algorithme : DPLL

- **D**avis & **P**utnam (1960)
- Davis, **L**ogeman, **L**oveland (1962)
- Applications importantes
  - vérification formelle des circuits
  - planification (emploi du temps)
  - ordonnancement



A Machine Program for Theorem-Proving†

Martin Davis, George Logemann, and Donald Loveland

Institute of Mathematical Sciences, New York University

The programming of a proof procedure is discussed in connection with trial runs and possible improvements.

In [1] is set forth an algorithm for proving theorems of quantification theory which is an improvement in certain respects over previously available algorithms such as that of [2]. The present paper deals with the programming of the algorithm of [1] for the New York University, Institute of Mathematical Sciences' IBM 704 computer

- Application amusante (?)

  La plus grosse preuve de l'histoire des mathématiques

  - construction et vérification d'une preuve de 200To en utilisant 1 solveur SAT, 800 coeurs et 2 jours…

Contenu du cours

- Méthodes
  - L'algorithme du simplexe (programmation linéaire)
  - L'algorithme DPLL (SAT)
- Modélisation
  - Traduire un problème concret en un problème abstrait (PL ou SAT)
  - Reconnaitre deux problèmes qui sont ``les mêmes''
- Optimisation
  - Optimiser une fonction linéaire sous des contraintes linéaires
  - Problèmes d'emploi du temps
  - Coloration de graphe