

Chapitre I

Introduction

Claude-Guy Quimper



Qu'est-ce que l'optimisation combinatoire?

- C'est une branche de l'informatique (intelligence artificielle) et des mathématiques appliquées.
- L'optimisation combinatoire est la recherche d'une solution au coût minimal d'un problème dont l'espace des solutions est discret.

À quoi sert l'optimisation combinatoire?

- L'optimisation combinatoire est la science à la base de la recherche opérationnelle « qui recherche la meilleure façon d'opérer des choix en vue d'aboutir au résultat visé »¹.
- Dans notre économie, il est naturel de vouloir optimiser les processus industriels.
 - Diminuer les coûts de production.
 - Améliorer la fixation des prix.

¹ Source: Wikipédia, rubrique « Recherche opérationnelle »

En agriculture

- La fédération des producteurs de lait au Québec planifie les routes empruntées par ses camions-citernes pour collecter le lait dans les fermes et l'emmener dans les usines de traitement.

En transport

- UPS planifie sa collecte et sa livraison de colis afin de...
 - minimiser les délais
 - minimiser les virages à gauche!
 - minimiser les coûts en carburant
- Les experts en développement durable s'intéressent à l'optimisation combinatoire justement à cause de la diminution de la consommation des carburants

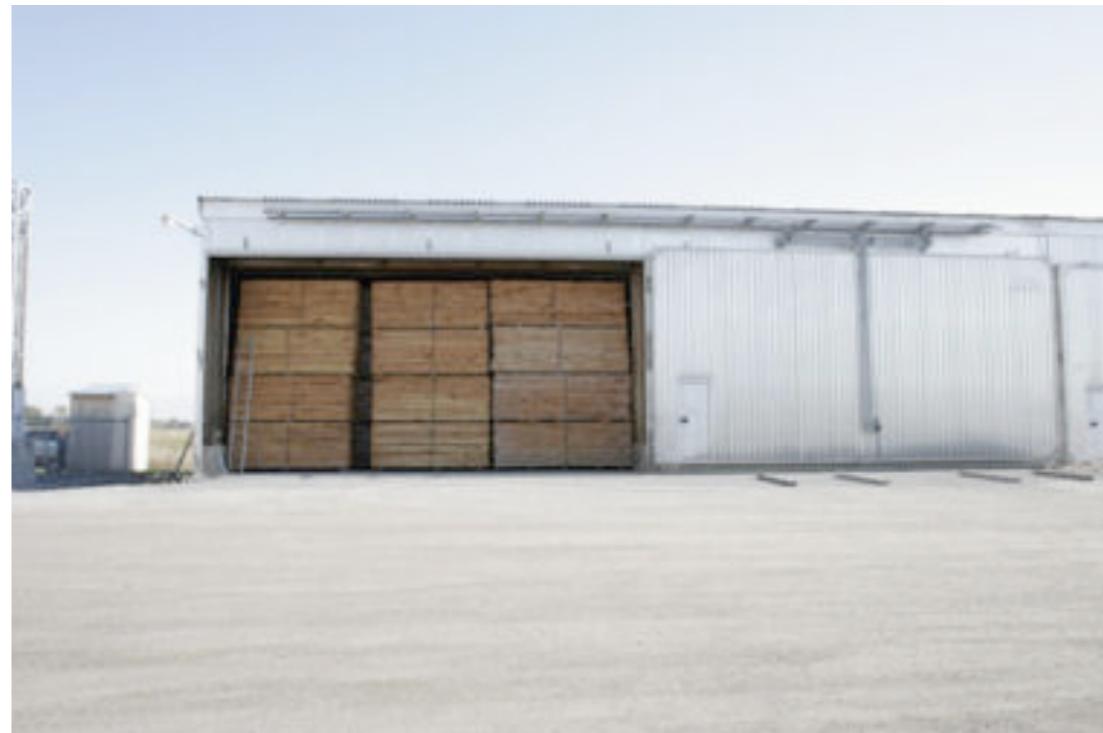
Toujours en transport

- Dans l'industrie du taxi:
 - Comment une flotte de taxis doit-elle se répartir sur le territoire pour maximiser le nombre de voyages?

Dans l'industrie forestière

- À l'Université Laval, le consortium de recherche Forac optimise les processus industriels de l'industrie du bois.
- Comment planifier le séchage et le rabotage du bois pour obtenir les produits qui seront les plus demandés?

Séchage du bois



Dans la gestion des ressources humaines

- Confection d'horaires de travail.
- Dans un restaurant ou un commerce de détail, on sait combien d'employés doivent travailler à chaque période de la journée.
- L'horaire d'un employé est soumis à une convention collective et les normes du travail.
- Comment minimiser le nombre d'heures travaillées tout en satisfaisant la demande?

Dans la gestion des ressources humaines

- Pour les compagnies aériennes, confectionner l'horaire de travail des agents de bord est un réel casse-tête.
- Non seulement il faut tenir compte de la demande et des normes du travail, mais il faut aussi tenir compte de la réalité géographique.

Dans les sports

- Confection des horaires de joutes pour une ligue sportive.
- Une équipe ne doit pas être à l'extérieur trop longtemps.
- Une équipe ne peut pas jouer plus de x fois par semaine.
- On doit minimiser les coûts de transport.
- En réalité, plusieurs autres variables entrent en compte dans la fonction objectif. Par exemple, une joute Canadian - Bruins rapporte beaucoup plus avant les séries qu'en milieu de saison. Il n'y a donc pas que les coûts de transports.

En informatique

- Y a-t-il un bogue dans cette fonction?

```
int puissanceDeux (int x) {  
    int c = 1;  
    while (x != 0) {  
        c = 2 * c;  
        x--;  
    }  
    return c;  
}
```

En informatique

- Il y a une possibilité de dépassement de la capacité d'un entier (*integer overflow*).
- Si x est négatif, la fonction retourne une réponse erronée après avoir bouclé 2^w fois où w est le nombre de bits dans un mot.

En informatique

- Pourrait-on avoir un programme qui détecte les bogues dans un code de programmation?
- Tous les bogues? Non.
- Plusieurs bogues? Oui!

En électronique

- Existe-t-il une entrée que l'on peut donner à un micro-processeur dont la sortie ne serait pas celle prévue?
- En 1994, Intel lance le processeur Pentium V. L'instruction FDIV (division flottante) est erronée.

Pentium IV $\frac{4195835}{3145727} = 1.333820449136241002$

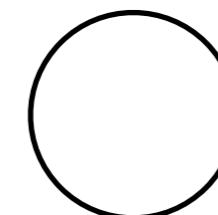
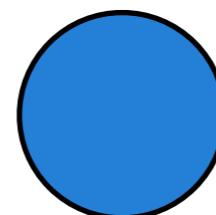
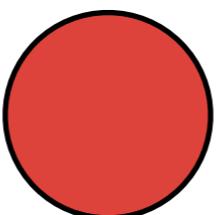
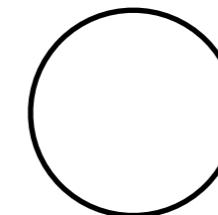
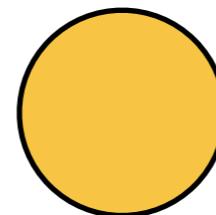
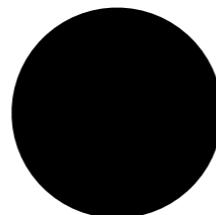
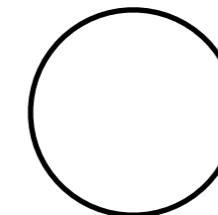
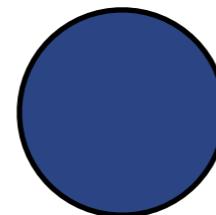
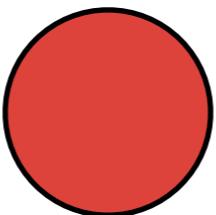
Pentium V $\frac{4195835}{3145727} = 1.333\textcolor{red}{739068902037589}$

- Intel a perdu 475M\$... selon Wikipédia.

Broderie

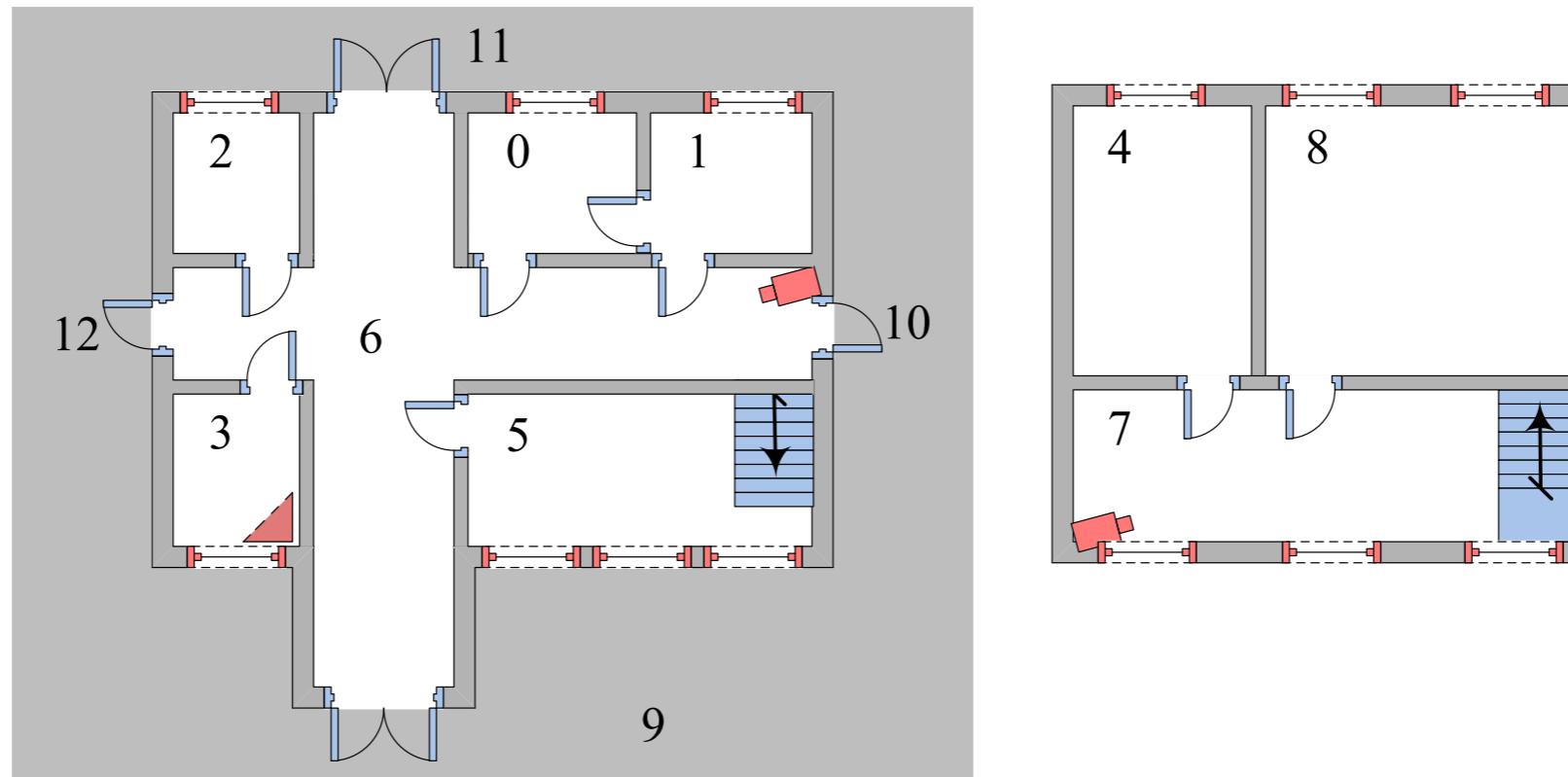
- Une machine broderie industrielle possède un nombre fixe de bobines de fil de couleurs différentes.
- Avant de commencer la broderie, il faut s'assurer que les fils de chaque couleur sont bien installés.
- Remplacer une bobine de fil par une autre peut prendre plusieurs minutes.
- Il est donc avantageux d'ordonner les tâches de broderie par couleur afin de broder conséutivement les logos partageant les mêmes couleurs.

Broderie



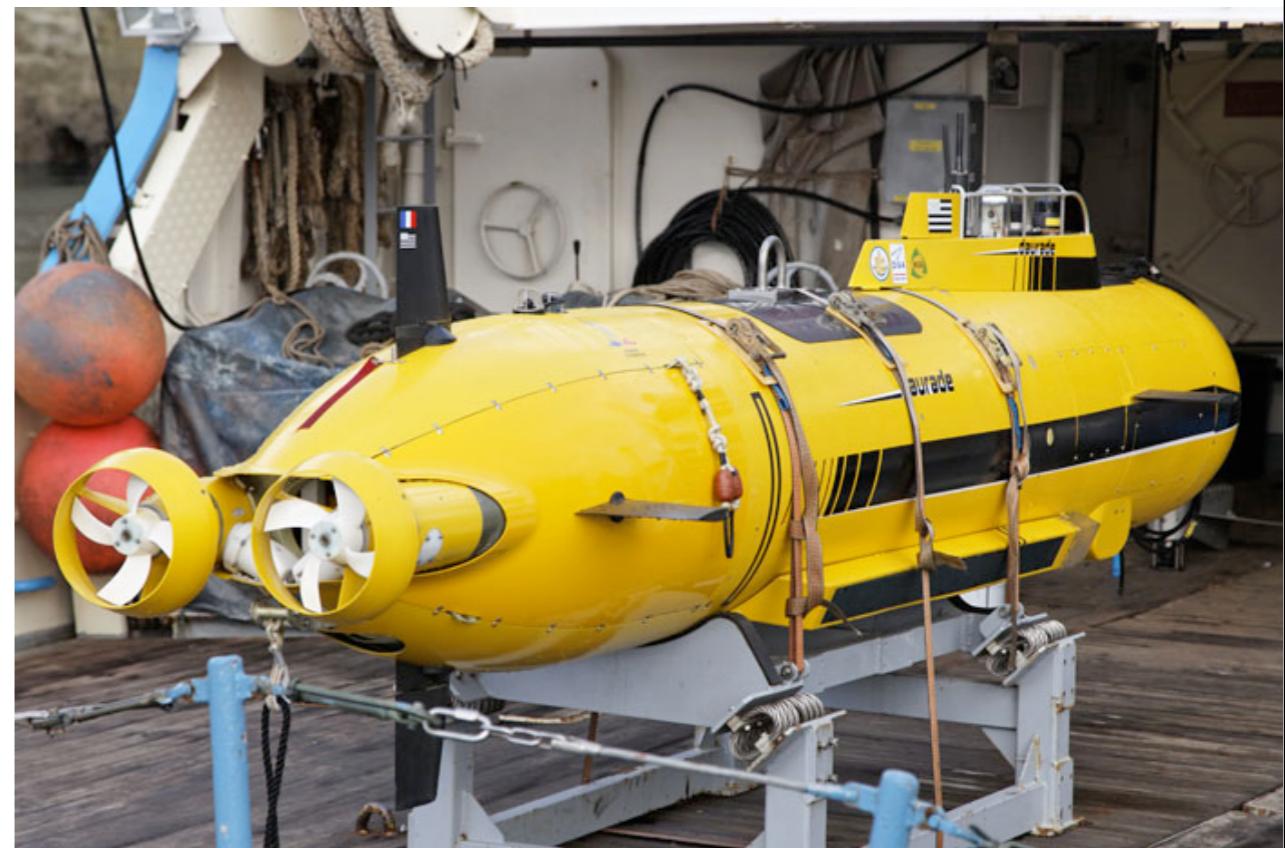
Planification des opérations de recherche et sauvetage

- On recherche une personne qui tente de s'évader de cet édifice. Dans quel ordre doit-on visiter les pièces?



A fictive building

Recherche de mines antisoumarins



Fond marin

c	r	f	f	f	f
c	r	f	f	r	r
c	r	r	r	r	r
r	r	r	r	r	r
f	f	f	f	f	f
f	f	f	f	f	f



complex



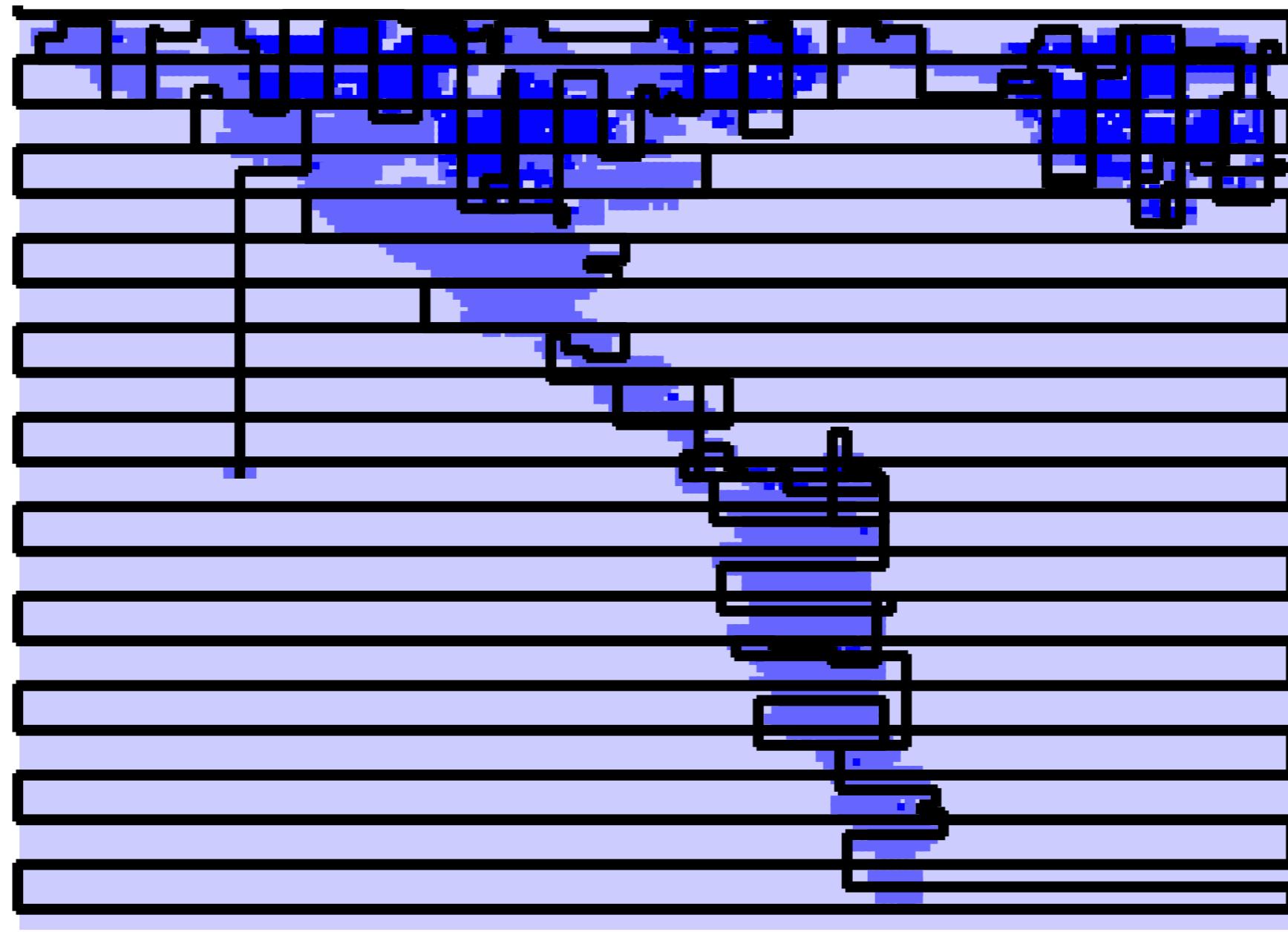
sand ripples



flat

- Le fond marin est divisé par une grille.
- Dans chaque case, on évalue la probabilité qu'il y ait une mine et le niveau de difficulté de détecter une mine à cet endroit.

Recherche de mines antisouamarins



NCSM Harry DeWolf



NCSM = Navire canadien de Sa Majesté

Retour sur la NP-Complétude

- En 1971, Stephen Cook souhaitait vérifier si un circuit électronique pouvait avoir un bogue. Il a modélisé ce problème sous forme du problème qu'il appela SAT pour *satisfiability*.

$$x_1 \vee \neg x_2 \vee x_3$$

$\neg x_1 \vee x_2 \vee x_4$ **Une instance SAT**

$$x_2 \vee x_3 \vee x_4$$

SAT

- Une variable booléenne peut prendre les valeurs *vrai* ou *faux*.
- Un littéral peut être une variable x_i ou sa négation $\neg x_i$.
- Une clause est une disjonction de plusieurs littéraux.
- Une solution est une affectation des variables aux valeurs *vrai* et *faux* de sorte à satisfaire chacune des clauses.

Exemple d'une instance SAT

$$x_1 \vee \neg x_2 \vee x_3$$

$$\neg x_1 \vee x_2 \vee x_4$$

$$x_2 \vee x_3 \vee x_4$$

- $x_1 = \text{vrai}, x_2 = \text{vrai}, x_3 = \text{vrai}, x_4 = \text{vrai}$ est une solution.
- $x_1 = \text{faux}, x_2 = \text{vrai}, x_3 = \text{vrai}, x_4 = \text{faux}$ est aussi une solution.
- Puisqu'il existe au moins une solution, on dit que cette instance est *satisfiable* ou *réalisable*.
- Lorsqu'il n'existe pas de solutions, on dit que l'instance n'est pas satisfiable ou n'est pas réalisable.

Difficulté de SAT

- Stephen Cook a modélisé son problème sous la forme d'un problème SAT, mais n'a pas réussi à concevoir un algorithme efficace pour le résoudre.
- En fait, depuis 1971, personne n'a réussi à concevoir un algorithme pouvant résoudre toute instance SAT en temps polynomial.
- Cette simple incapacité à résoudre le problème efficacement devrait nous convaincre que ce problème est difficile, mais il existe d'autres évidences de sa difficulté.

Que peut modéliser SAT?

- Cook voulait trouver les bogues dans un circuit électronique et s'est retrouvé avec une instance SAT à résoudre.
- Y a-t-il d'autres problèmes que l'on peut modéliser sous la forme d'une instance SAT?
- Réponse: Oui, beaucoup!

La classe NP

- Un **problème de décision** est un problème dont la réponse est soit *oui* ou *non*.
 - Ex.: Existe-t-il une solution à une instance SAT?
- Un **certificat** est une donnée prouvant que la solution à un problème de décision est *oui*.
 - Ex.: $x_1 = \text{faux}, x_2 = \text{vrai}, x_3 = \text{vrai}, x_4 = \text{faux}$
- Un **algorithme de vérification** prend en entrée une instance d'un problème de décision et une donnée et vérifie si cette donnée est un certificat valide.
 - Ex.: Un algorithme qui vérifie si une affectation rend vraie chaque clause d'une instance SAT.

La classe NP

- La classe NP est l'ensemble des problèmes de décision qui admettent un algorithme de vérification à temps polynomial.
- Le problème SAT est un problème NP-Complet, c'est-à-dire qu'il peut encoder tout problème de la classe NP.
- Autrement dit: si vous pouvez vérifier efficacement la solution d'un problème, c'est que ce problème peut être écrit sous la forme d'un problème SAT.

Conséquences du théorème de Cook.

- Si on peut écrire un programme qui résout SAT, ou tout autre problème NP-Complet, ce programme peut résoudre une multitude de problèmes.
- Faisons-le!

Le solveur

- Un solveur est un logiciel qui prend en entrée un problème et retourne une solution.
- S'il est possible de soumettre un problème NP-Complet à un solveur, alors ce logiciel peut résoudre tous les problèmes dans NP.
- Il existe différents types ou familles de problèmes. Un solveur est généralement spécialisé pour un type de problème.

Types de problèmes

- Nous étudierons deux types de problèmes
 - Les problèmes de satisfaction de contraintes
 - Les problèmes d'optimisation avec contraintes

Problème de satisfaction de contraintes

- Un problème de satisfaction de contraintes a trois composantes:
 - Des variables: X_1, X_2, \dots, X_n
 - Des domaines: $\text{dom}(X_1), \text{dom}(X_2), \dots, \text{dom}(X_n)$
 - Un domaine est un ensemble de valeurs candidates qui peuvent être affectées à une variable.
 - Il y a un domaine par variable.
 - Des contraintes: C_1, C_2, \dots, C_m

Problème d'optimisation avec contraintes

- Un problème d'optimisation avec contraintes inclut tous les éléments d'un problème de satisfaction.
- Il y a en plus une fonction objectif à minimiser ou à maximiser.
- Cette fonction objectif prend en paramètre des variables du problème et retourne un nombre réel.
- La valeur de la fonction objectif s'appelle la *valeur objective*.

Exemples de fonctions objectifs

- Dans la confection d'horaires de joutes de hockey, on veut minimiser le budget alloué au transport des équipes.
- Dans la confection d'horaires de travail, on veut minimiser les pertes causées par le salaire versé en trop lorsque trop d'employés travaillent simultanément et minimiser les pertes de revenus causées par un manque de personnel.

Minimisation vs Maximisation

- Pour certains problèmes, on veut minimiser la fonction objectif. Pour d'autres problèmes, on veut maximiser sa valeur
- Si on veut minimiser une fonction, on peut aussi maximiser la négation de cette fonction.

Qu'est-ce qu'une contrainte?

- Une contrainte est une condition que l'on impose à un sous-ensemble des variables.
- Ce sous-ensemble de variables s'appelle la « portée » de la contrainte.
- L'arité d'une contrainte est le nombre de variables dans sa portée.
- De façon formelle, une contrainte est un ensemble de tuples représentant les affectations valides des variables. Cet ensemble de tuples peut être défini implicitement ou explicitement.

La contrainte « ou »

- $\text{Ou}(X_1, X_2)$ (souvent dénoté $X_1 \vee X_2$)
 - Cette contrainte a une arité de deux variables.
 - Les affectations valides peuvent être énumérées:
 - $\{(f\ddot{a}ux, v\ddot{r}ai), (v\ddot{r}ai, f\ddot{a}ux), (v\ddot{r}ai, v\ddot{r}ai)\}$

La contrainte « Addition »

- $\text{Addition}(A, B, C)$
 - Cette contrainte a une arité de trois variables.
 - Elle est satisfaite lorsque la troisième variable est égale à la somme des deux premières.
 - Les affectations valides ne peuvent pas être énumérées explicitement puisqu'il y en a une infinité.
 - L'ensemble des affectations valides est donc le suivant.

$$\{(a, b, c) \mid a + b = c\}$$

La contrainte « Tableau »

- $\text{Tableau}(X_1, X_2, X_3, T)$
- Elle restreint les variables de la portée à prendre les valeurs d'une ligne du tableau T .
- Très utile pour exprimer ses propres contraintes.
- Dans l'exemple, on a $X_1 + X_2 = X_3 \bmod 3$
- Le tableau peut avoir un nombre de colonnes arbitraire, mais fixe.

T		
0	0	0
0	1	1
0	2	2
1	0	1
1	1	2
1	2	0
2	0	2
2	1	0
2	2	1

Notation

- Généralement, une contrainte s'écrit sous la forme d'un prédicat, c'est-à-dire avec le nom de la contrainte suivi des variables de la portée entre parenthèses.
Ex.: NomDeLaContrainte(X_1, X_2, X_3)
- Certaines contraintes peuvent être exprimées par des opérateurs.
Ex: Addition(A, B, C) $\Leftrightarrow A + B = C$

Exemples de contraintes courantes

$$A + B = C$$

$$AB = C$$

$$A < B$$

$$A - B = C$$

$$\frac{A}{B} = C$$

$$A \leq B$$

$$A \vee B$$

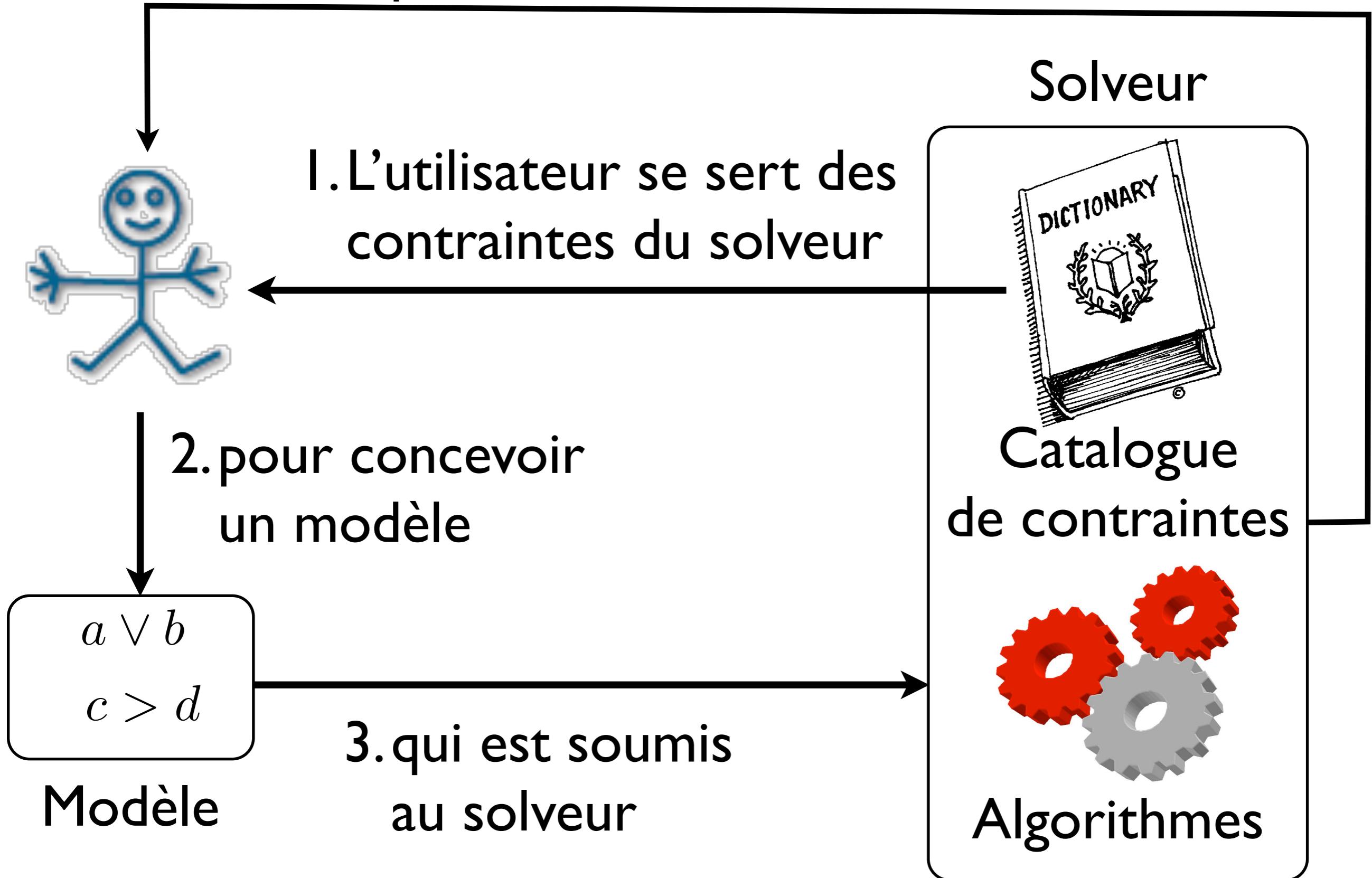
$$A \wedge B$$

$$A \neq B$$

Modélisation

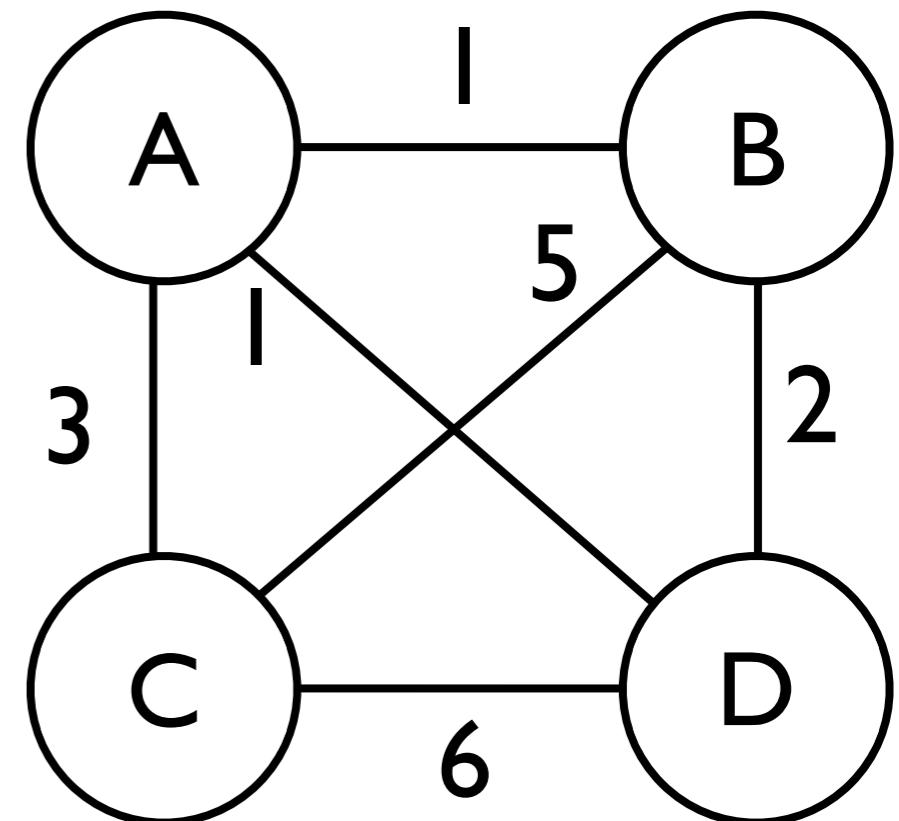
- Un solveur possède une collection de contraintes qui peuvent être utilisées pour modéliser un problème.
- Le rôle de l'utilisateur est donc d'écrire le problème de satisfaction de contraintes en utilisant les contraintes offertes par le solveur.
- Cette étape se nomme la modélisation.

4. qui retourne une solution



Le commis voyageur

- Un cycle est dit *hamiltonien* s'il visite tous les noeuds du graphe une et une seule fois.
- Le problème du commis voyageur consiste à trouver le cycle hamiltonien dont la somme des poids des arêtes est minimale.
- Les noeuds représentent des villes, les arêtes des routes, et la somme des poids la distance parcourue par le commis voyageur.



Comment modéliser le problème du commis voyageur?

- Quelles sont les variables?
- Quel est le domaine de chaque variable?
- De quelles variables dépend la fonction objectif?
- Quelles sont les contraintes?
- Quelle est la fonction objectif?

Les variables

- Soit X_i le noeud visité au temps i . La séquence X_0, X_1, X_2, X_3, X_0 est donc le chemin emprunté par le commis voyageur.
- Le domaine de X_i est l'ensemble des noeuds $\{A, B, C, D\}$
- Soit D_i la distance parcourue au j ème déplacement. Le domaine de D_i est l'ensemble des poids d'arêtes $\{1, 2, 3, 5, 6\}$

Les contraintes

Tableau($X_i, X_{i+1 \bmod 4}, D_i, T$)
pour $i = 0..3$

- Cette contrainte relie deux noeuds consécutifs avec la variable de distance correspondante.
 $X_i \neq X_j$ pour $0 \leq i < j \leq 3$
- Cette contrainte empêche de revisiter un noeud.

T		
A	B	I
A	C	3
A	D	I
B	A	I
B	C	5
B	D	2
C	A	3
C	B	5
C	D	6
D	A	I
D	B	2
D	C	6

La fonction objectif

- La fonction objectif est de minimiser la somme des distances.
- minimiser $D_0 + D_1 + D_2 + D_3$

Récapitulatif du modèle

$$\min D_0 + D_1 + D_2 + D_3$$

Sous les contraintes

$$\text{dom}(X_i) = \{A, B, C, D\} \quad \text{pour } i = 0..3$$

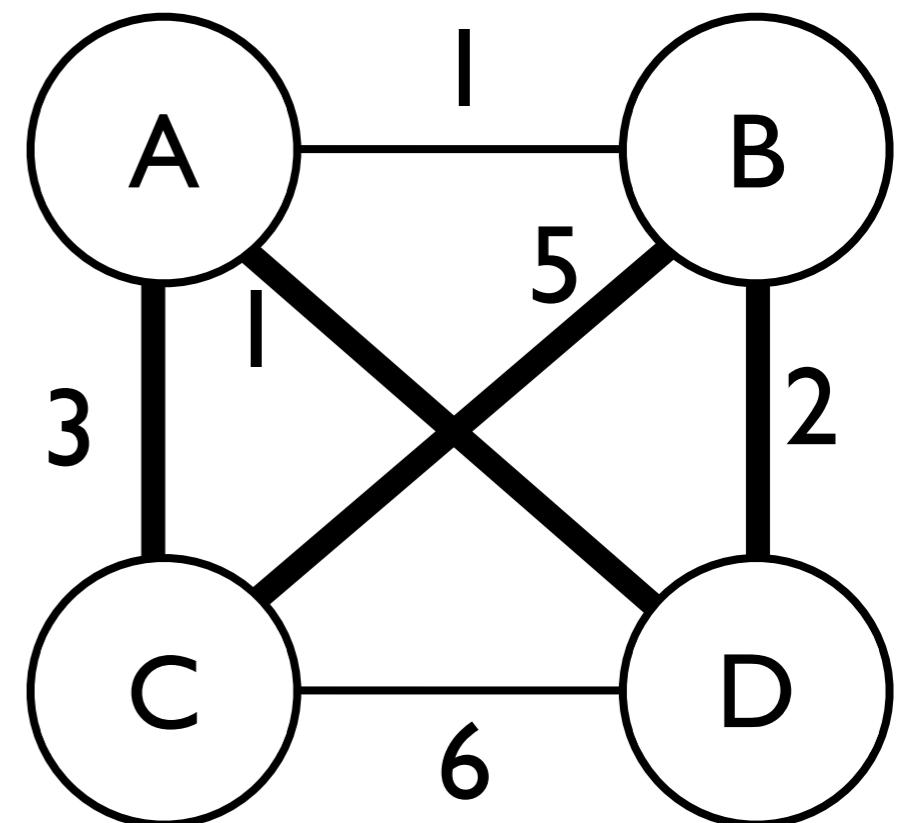
$$\text{dom}(D_i) = \{1, 2, 3, 5, 6\} \quad \text{pour } i = 0..3$$

$$\text{Tableau}(X_i, X_{i+1 \bmod 4}, D_i, T) \quad \text{pour } i = 0..3$$

$$X_i \neq X_j \quad \text{pour } 0 \leq i < j \leq 3$$

Solution de l'instance

- $X_0 = A, X_1 = D, X_2 = B, X_3 = C$
- $D_0 = 1, D_1 = 2, D_2 = 5, D_3 = 3$
- Valeur objective: 11



Caractéristiques d'un bon modèle

- De façon générale, on essaie de produire les modèles les plus compacts possible. La taille d'un modèle est donnée par:
 - Le nombre de variables;
 - Le nombre de contraintes;
 - La cardinalité des domaines.
- Si le modèle contient des contraintes tableau, on note aussi le nombre d'entrées dans les tableaux.
- On peut utiliser la notation asymptotique grand O pour exprimer ces quantités.
- Le modèle doit être de taille polynomiale.

Analyse du modèle du commis voyageur

- Soit n le nombre de noeuds dans le graphe.
- Nous avons un modèle avec
 - $2n$ variables (n variables X_i et n variables D_i);
 - n valeurs dans $\text{dom}(X_i)$ et $\frac{n(n-1)}{2}$ valeurs dans $\text{dom}(D_i)$ pour un total de $n^2 + \frac{n^2(n-1)}{2} \in \Theta(n^3)$ valeurs;
 - n contraintes tableau et $\frac{n(n-1)}{2}$ contraintes de différence pour un total de $\Theta(n^2)$ contraintes.
- Chaque contrainte tableau possède $n(n - 1) \in \Theta(n^2)$ entrées.

Conclusion

- L'optimisation combinatoire, c'est la recherche d'une solution optimale à un problème dans un espace de solutions discret.
- Elle est utilisée dans plusieurs industries.
- Un solveur pouvant résoudre un problème de satisfaction de contraintes ou un problème d'optimisation avec contraintes peut résoudre tous les problèmes dans NP.
- Il suffit de modéliser un problème avec les contraintes disponibles dans le catalogue du solveur.