Chapitre 1: Introduction

INF889B — Algorithmes d'optimisation combinatoire

Alexandre Blondin Massé

Université du Québec à Montréal

Hiver 2020

Plan

Présentation du cours

Informations générales

- **Trimestre**: Hiver 2020
- **Titre du cours**: Algorithmes d'optimisation combinatoire
- **Sigle**: INF889B
- **Département**: Informatique
- **Enseignant**: Alexandre Blondin Massé
- **Courriel**: blondin_masse.alexandre@uqam.ca
- **Site personnel**: http://lacim.uqam.ca/~blondin
- **Bureau**: PK-4525
- **Coordonnateur**: Alexandre Blondin Massé
- **Site du cours**: http://lacim.uqam.ca/~blondin/fr/inf889b
- **Plan de cours**: cliquer ici

Ancien cours: MAT7560

Description disponible sur le site de l'UQAM:

> *« Révision de l'algorithme du simplexe et de la théorie de la dualité. Problèmes classiques de l'optimisation combinatoire: flot maximum, couplage maximal, arbre minimal dans un graphe; problème du voyageur de commerce. Programmation linéaire en nombres entiers. Étude de la complexité des algorithmes introduits. »*

Concentré sur: optimisation convexe, programmation linéaire, programmation linéaire en nombres entiers

Nouveau cours: INF889B

Description disponible sur le site de l'UQAM:

> *« Modélisation d'un problème d'optimisation combinatoire. Optimisation exacte: solution naïve, séparation et évaluation progressive, algorithmes paramétrés. Optimisation convexe: programmes linéaires, algorithme du simplexe, théorie de la dualité, programmation linéaire en nombres entiers. Méthodes approximatives et métaheuristiques: recherche locale, recuit simulé, recherche taboue. Méthodes bio-inspirées: algorithmes évolutionnaires, colonies de fourmis, etc. Optimisation par apprentissage: survol des méthodes d'apprentissage automatique, intégration d'apprentissage dans des algorithmes d'optimisation combinatoire. »*

Plus général: approximation, méta-heuristiques, apprentissage automatique

Objectifs du cours

Se familiariser avec les méthodes d'optimisation combinatoire exactes, approximatives et adaptatives. Connaître leurs avantages, leurs limites, être en mesure de les implémenter et d'évaluer leur performance.

Plus spécifiquement

- Bien **modéliser** un problème d'optimisation combinatoire
- En utilisant de façon adéquate la **notation mathématique**
- Identifier les situations où des approches **exactes** sont insuffisantes
- Comprendre les stratégies algorithmiques **non exactes**
- Savoir analyser leur **complexité**
- Savoir **implémenter** ces algorithmes
- Évaluer la qualité d'une stratégie de façon **empirique**
- **Communiquer** à l'oral et à l'écrit ces différents aspects

Présentations

J'aimerais en savoir plus sur vous!

- Votre **nom**?
- Votre **programme** d'étude?
- Votre **directeur ou directrice** de recherche?
- Vos **objectifs**? Pourquoi ce cours?
- Des idées de **sujets** que vous aimeriez aborder?

Modalités d'évaluation (1/2)

Revue de littérature

- Conférence CPAIOR
- CPAIOR = *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*
- Choisir un article dans les **trois dernières éditions**
- Le résumer et le présenter devant la classe

Projet de session

- Doit être un « vrai » problème d'optimisation
- Idéalement un sujet lié à vos travaux de maîtrise/doctorat

Devoirs

- Énoncé et gabarit LaTeX
- Remise électronique avec code source

Modalités d'évaluation (2/2)

Revue de littérature (30%)

- Choix de l'article – 1 page (5%)
- Présentation orale – 20 minutes (15%)
- Résumé écrit – 2 pages (10%)

Projet de session (40%)

- Choix du sujet – 1 page (5%)
- Présentation du projet – 25 minutes (20%)
- Rapport final – 10 pages (15%)

Devoirs (30%)

- 3 devoirs
- 10% chacun

Contenu détaillé

1. **Chapitre 1**: Introduction. Présentation du cours, exemples.
2. **Chapitre 2**: Modélisation, algorithmes, espace combinatoire, ordre de grandeur, complexité, expérience.
3. **Chapitre 3**: Optimisation exacte. Solution naïve, séparation et évaluation progressive, algorithmes paramétrés.
4. **Chapitre 4**: Optimisation convexe. Programmation linéaire, algorithme du simplexe, théorie de la dualité, programmation linéaire en nombres entiers.
5. **Chapitre 5**: Approximation et méta-heuristiques. Recherche locale, recuit simulé, recherche taboue.
6. **Chapitre 6**: Méthodes bio-inspirées. Algorithmes évolutionnaires, colonies de fourmis, essaims de particules.
7. **Chapitre 7**: Optimisation avec apprentissage. Survol des méthodes d'apprentissage automatique, intégration d'apprentissage dans des algorithmes d'optimisation combinatoire.

Références (toutes optionnelles)

- Marek Cygan et al., *Parameterized Algorithms*, Springer, 2016, disponible en ligne
- Teofilo F. Gonzalez, *Handbook of Approximation Algorithms and Metaheuristics*, 2007.
- Fred Glover et Gary A. Kochenberger, *Handbook of metaheuristics*, 2003, ISBN 978-1-4020-7263-5.
- Vašek Chvátal, *Linear programming*, W. H. Freeman and Company, New York, 1983, 478 pp.
- Stephen Boyd et Lieven Vandenberghe, *Convex optimization*, notes disponibles en ligne
- Richard S. Sutton et Andrew G. Barto, *Reinforcement Learning: An Introduction*, second edition, MIT Press, Cambridge, MA, 2018, disponible en ligne

Politiques de l'UQAM

- Règlement 18 sur la tricherie et l'intégrité académique (plagiat): http://r18.uqam.ca/.
- Politique 16 contre le harcèlement sexuel: document pdf.

Exemples

Quelques problèmes d'optimisation

Question

Y a-t-il un problème d'optimisation qui vous intéresse?

Quelques problèmes d'optimisation

Question

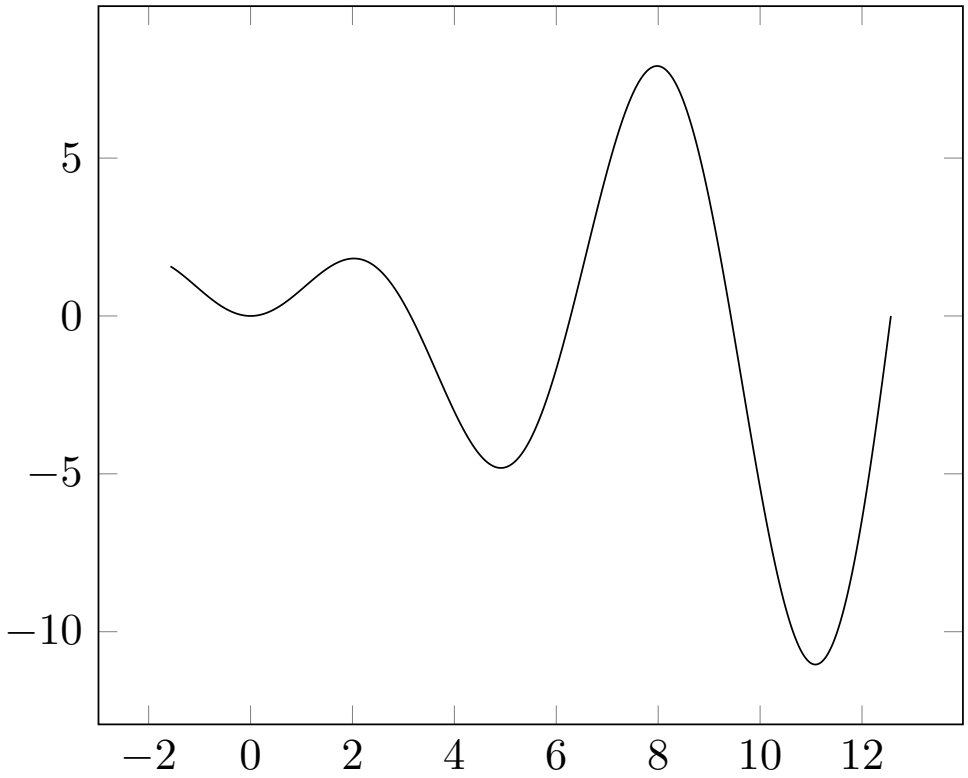
Y a-t-il un problème d'optimisation qui vous intéresse?

Autres exemples

- Optima d'une fonction réelle
- Distance entre deux nuages de points
- Plus court chemin dans un graphe
- Distance d'édition
- Traitement de tâche en temps minimal
- Allocation de ressources dans un réseau
- Déplacements dans une scène 2D ou 3D
- Désambiguïsation sémantique
- Joueur intelligent

Optima d'une fonction (1/2)

Considérez la fonction $f(x) = x \sin x$ sur l'intervalle $[-\pi/2, 4\pi]$:



- Quels sont ses **optima**?
- Comment les calcule-t-on?

Optima d'une fonction (2/2)

- On **dérive** la fonction:

$$\begin{aligned} f(x) &= x \sin x \\ f'(x) &= \sin x + x \cos x \end{aligned}$$

- Et on cherche ses **zéros**:

$$f'(x) = 0 \quad \Rightarrow \sin x + x \cos x = 0$$

Optima d'une fonction (2/2)

- On **dérive** la fonction:

$$\begin{aligned} f(x) &= x \sin x \\ f'(x) &= \sin x + x \cos x \end{aligned}$$

- Et on cherche ses **zéros**:

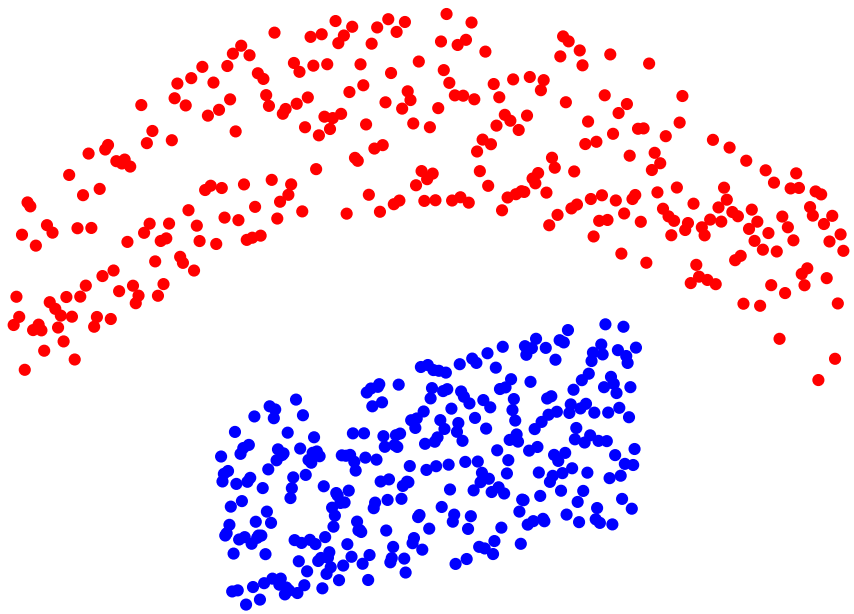
$$f'(x) = 0 \quad \Rightarrow \sin x + x \cos x = 0$$

- Un algorithme connu de Newton-Raphson: on choisit x_0 assez proche du zéro qui nous intéresse, puis on calcule la suite $(x_k)_{k \geq 0}$ définie par

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Distance entre deux nuages de points (1/2)

Considérez deux nuages de points X et Y:



- Comment calcule-t-on la **distance** entre X et Y?

Distance entre deux nuages de points (2/2)

Solution naïve

- On calcule

$$d = \min\{\text{dist}(x, y) \mid x \in X, y \in Y\}$$

- Complexité **temporelle**?

Distance entre deux nuages de points (2/2)

Solution naïve

- On calcule

$$d = \min\{\text{dist}(x, y) \mid x \in X, y \in Y\}$$

- Complexité **temporelle**? $O(|X||Y|)$

Autres solutions

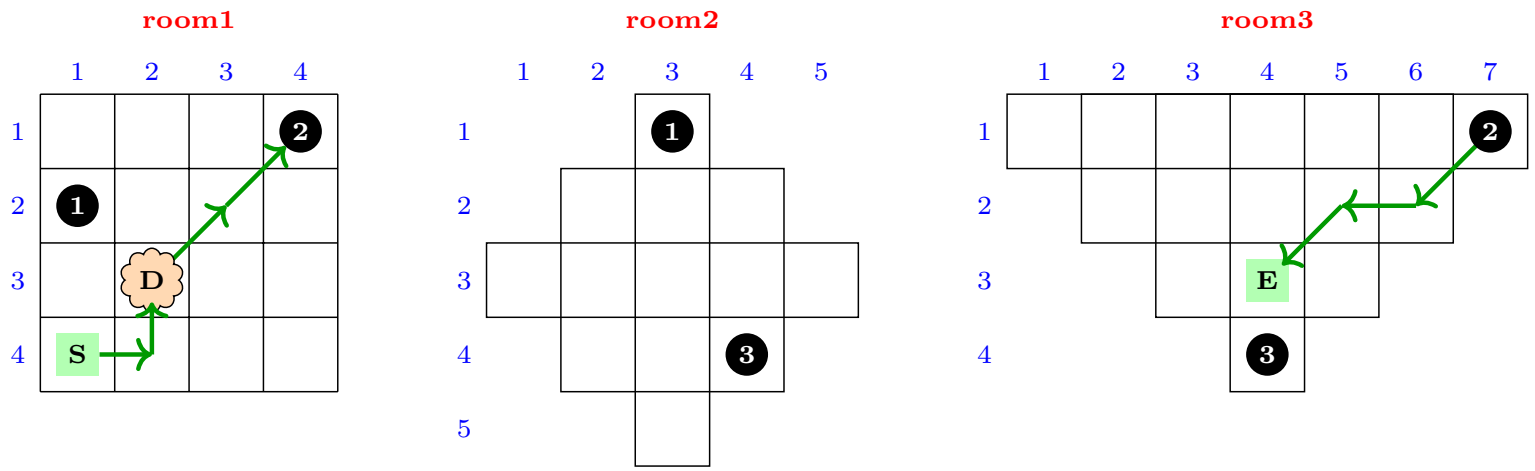
- Parfois, on s'intéresse à l'évolution de X et Y dans le temps
- Plusieurs **opérations** possibles: ajout de points, suppression de points, déplacement de points, etc.
- Alors il peut être intéressant d'effectuer des **précalculs**
- Calculer les enveloppes **convexes** et les mettre à jour
- Partitionner selon une **grille**
- Insérer dans un **arbre-kd**, etc.

Plus court chemin dans un graphe (1/2)

Plusieurs algorithmes connus:

- **Parcours en largeur**: lorsque le poids des arêtes est uniforme
- Algorithme de Dijkstra: lorsque les poids sont positifs ou nuls
- Algorithme A*: lorsque les poids sont positifs ou nuls, plus efficace que Dijkstra si on sait approximer la distance de façon « optimiste »
- Algorithme de Bellman-Ford-Moore: lorsque le graphe contient des poids négatifs
- Algorithme de Floyd-Warshall: plus courte distance pour toutes paires de sommets

Plus court chemin dans un graphe (2/2)



- Pièces en forme de **carré**, de **losange** ou de **triangle**
- Plus court chemin entre S et E
- Chaque pas horizontal ou vertical a un coût de 1
- Il existe des téléporteurs entre les pièces (coût de 3)
- Un objet spécial appelé diagonaliseur D permet de se déplacer en diagonal au coût de 1
- Calcul d'un plus court chemin entre deux **sommets** ou **états**?

Distance d'édition

Entre deux chaînes

- **Opérations**: insérer, supprimer, modifier une lettre
- **Applications**: bio-informatique, traitement des langues naturelles, traitement du texte, détection de plagiat, etc.

Entre deux graphes

- **Opérations**: insérer, supprimer, modifier un sommet ou une arête
- **Applications**: reconnaissance d'images, fouille de données, bio-informatique, etc.

Entre deux arbres

- **Opérations**: insérer, supprimer, modifier un noeud

Traitement de tâches (1/5)

Données

- Un ensemble de tâches J
- Un ensemble de machines identiques M
- Une fonction $t : J \to \mathbb{N}^*$ qui indique le temps nécessaire pour traiter chaque tâche
- **Objectif**: compléter toutes les tâches en un temps minimum

Exemple d'instance

- $J = \{j_1, j_2, j_3, j_4, j_5, j_6\}$
- $M = \{m_1, m_2, m_3\}$
- La fonction t est donnée par le tableau suivant

i	1	2	3	4	5	6
$t(j_i)$	3.0	2.0	3.0	4.0	2.0	1.0

Traitement de tâches (2/5)

Solution

- On cherche une **fonction** $S : J \to M$ qui associe à chaque tâche la machine qui la traitera
- **Exemple**: la solution S donnée par

i	1	2	3	4	5	6
$S(j_i)$	m_2	m_2	m_3	m_1	m_3	m_2

Symétries

- L'espace présente souvent des symétries
- Dans le cas ici, les machines sont supposées identiques
- Ainsi, la solution S' suivante est équivalente à S:

i	1	2	3	4	5	6
$S'(j_i)$	m_1	m_1	m_2	m_3	m_2	m_1

Traitement de tâches (3/5)

Question

- Quelle est la **taille** de l'espace des solutions?

Traitement de tâches (3/5)

Question

- Quelle est la **taille** de l'espace des solutions?

Réponse

- Nous avons $|M|$ choix par tâche
- Il y a $|M|!$ symétries
- Donc $|M|^{|J|}/|M|!$ solutions

Remarque

- La taille est **exponentielle** par rapport à $|J|$
- Généralement, M est assez petit, J est grand

Traitement de tâches (4/5)

Solution optimale

- Soit $\mathcal{S}$ l'ensemble des solutions possibles
- Soit $t : \mathcal{S} \to \mathbb{N}$ qui indique pour chaque solution S le temps total pris pour traiter la planification S
- On dit que $S \in \mathcal{S}$ est **optimale** si pour toute solution $S' \in \mathcal{S}$, on a $t(S) \leq t(S')$.

Questions

- Comment calcule-t-on $t(S)$ pour tout S?

Traitement de tâches (4/5)

Solution optimale

- Soit $\mathcal{S}$ l'ensemble des solutions possibles
- Soit $t : \mathcal{S} \to \mathbb{N}$ qui indique pour chaque solution S le temps total pris pour traiter la planification S
- On dit que $S \in \mathcal{S}$ est **optimale** si pour toute solution $S' \in \mathcal{S}$, on a $t(S) \leq t(S')$.

Questions

- Comment calcule-t-on $t(S)$ pour tout S?

$$t(S) = \max \left\{ \sum_{j \in S(m)} t(j) \;\middle|\; m \in M \right\}$$

Traitement de tâches (4/5)

Solution optimale

- Soit $\mathcal{S}$ l'ensemble des solutions possibles
- Soit $t : \mathcal{S} \to \mathbb{N}$ qui indique pour chaque solution S le temps total pris pour traiter la planification S
- On dit que $S \in \mathcal{S}$ est **optimale** si pour toute solution $S' \in \mathcal{S}$, on a $t(S) \leq t(S')$.

Questions

- Comment calcule-t-on $t(S)$ pour tout S?

$$t(S) = \max \left\{ \sum_{j \in S(m)} t(j) \;\middle|\; m \in M \right\}$$

- Comment peut-on vérifier si S est optimale?

Traitement de tâches (5/5)

Question

- Comment peut-on vérifier si S est **optimale**?

Exemple

- Prenons $S = (\{j_1, j_2\}, \{j_4, j_6\}, \{j_5, j_3\})$
- Alors $t(S) = \max\{3+2, 4+1, 2+3\} = 5$
- Est-ce qu'on peut faire mieux?

Traitement de tâches (5/5)

Question

- Comment peut-on vérifier si S est **optimale**?

Exemple

- Prenons $S = (\{j_1, j_2\}, \{j_4, j_6\}, \{j_5, j_3\})$
- Alors $t(S) = \max\{3+2, 4+1, 2+3\} = 5$
- Est-ce qu'on peut faire mieux?

$\rightarrow$ On peut calculer une borne inférieure

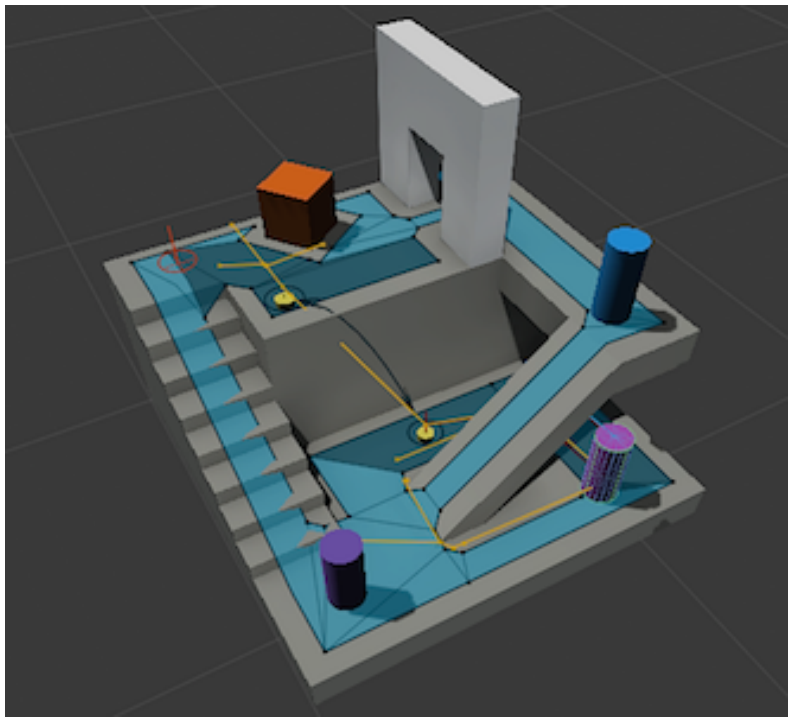
$\rightarrow$ Meilleur cas: utilisation optimale des machines

$\rightarrow$ Autrement dit, pour toute solution admissible S', on a

$$t(S') \geq \frac{\text{temps total des tâches}}{\text{nombre de machines}} = \frac{15}{3} = 5$$

$\rightarrow$ Ainsi, S est bien optimale

Déplacements dans une scène (1/2)



(Source: Unity3D)

- **Discrétisation** de l'espace possible
- Ajout de **points stratégiques** (*waypoints*)
- Distance sur **maillage**, **géodésiques**, etc.

Désambiguïsation sémantique

Trois **définitions** possibles du mot *bass* (exemple tiré de Wikipedia):

1. A type of fish
2. Tones of low frequency
3. A type of instrument

On s'intéresse aux deux **contextes** suivants:

- « I went fishing for some sea bass. »
- « The bass line of the song is too weak. »

Comment choisir la **bonne** définition?

Joueur intelligent

1989
(Source: MindYourDecisions)
VICTOR: joueur de V. Allis

1996-1997
(Source: ScientificAmerican)
Deepblue contre Kasparov

2017
(Source: BBC)
AlphaGo contre Lee Sedol

2018
(Source: TechCrunch)
AlphaStar contre Grzegorz Komincz