PyRat : cours 5

L'équipe PyRat

Jalons

Séance 1	...
Séance 2	...
Séance 3	Ramasser un unique morceau de fromage dans le labyrinthe
Séance 4	...
Séance 5	...
Séance 6	Ramasser plusieurs morceaux de fromage dans le labyrinthe
Séance 7	Gagner contre un adversaire

Au programme aujourd'hui

Complexité excessive

- La dernière séance, nous avons vu que le voyageur de commerce est trop complexe à résoudre pour un grand nombre de fromages
- Idée : utiliser un algorithme approché

Notions vues aujourd'hui

- Nous allons parler d'algorithmes gloutons
- Nous allons parler d'heuristiques
- Nous allons voir l'impact sur la complexité
- Nous allons voir l'impact sur la précision du résultat

Algorithme glouton et heuristiques

Principe

- On appelle **algorithme glouton** un algorithme qui fait une succession de décisions localement optimales dans l'espoir d'arriver à une solution globalement optimale
- *L'objectif en pratique est de considérablement réduire la complexité en sacrifiant la correction* : On parle d'**algorithme approché**

Heuristiques

- Pour prendre les décisions, on utilise des **heuristiques**
- Une heuristique est une stratégie locale espérée bonne
- Pour un même problème : de nombreuses heuristiques différentes
- La complexité du calcul de l'heuristique peut faire gagner sur la complexité totale
- *Si calculer votre heuristique coûte aussi cher que résoudre le problème global : aucun intérêt !*

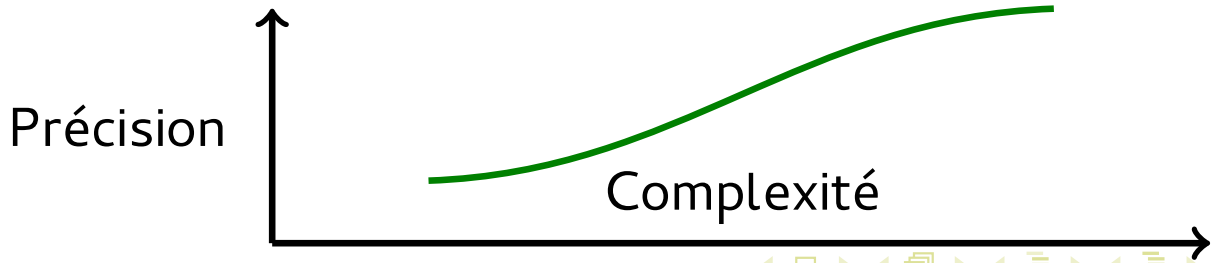
Exemple pour le voyageur de commerce

Exemples d'algorithmes approchés

- Aller systématiquement au voisin le plus proche
  - Complexité $\mathcal{O}(|V|)$
- Aller systématiquement aux k voisins les plus proches
  - Complexité $\mathcal{O}(|V|^k)$
- D'autres possibilités : A*...

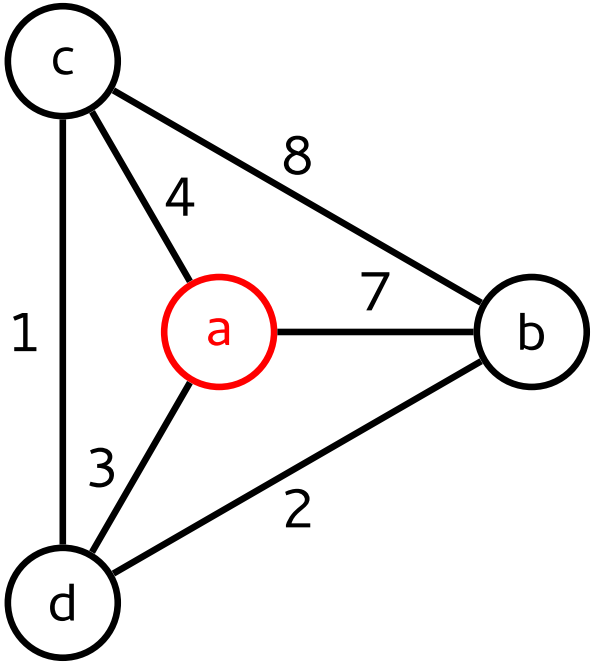
Compromis précision/complexité

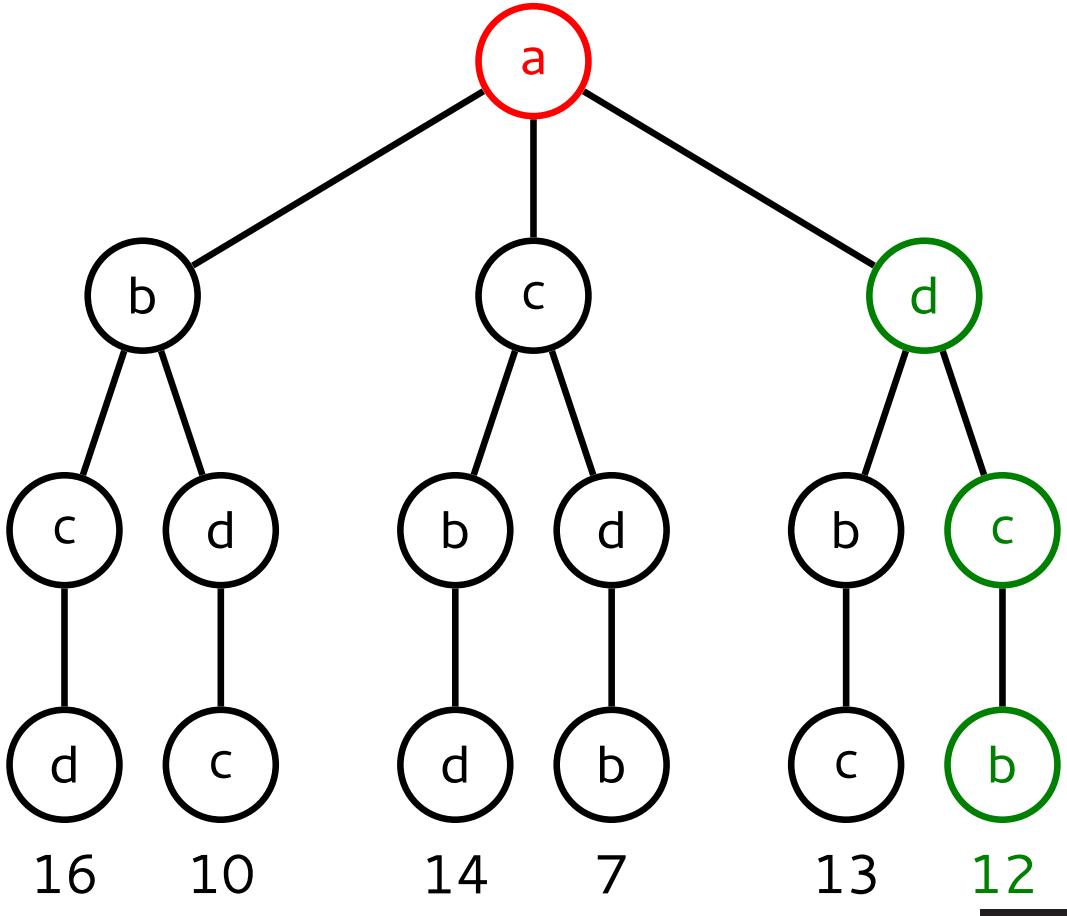
- Il faut réfléchir au problème :
  - Quel temps de calcul est disponible ?
  - Quelle perte de précision est acceptable ?



Exemple de résultat non optimal

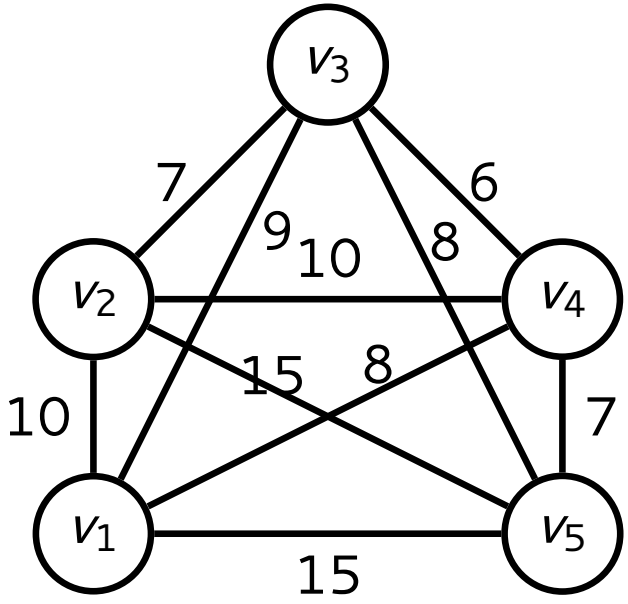
Aller systématiquement au plus proche voisin





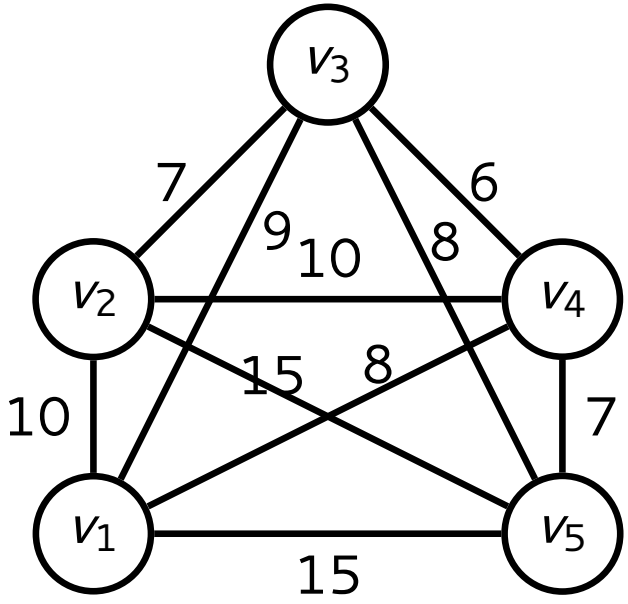
Exercice

Dérouler l'algorithme glouton du plus proche voisin pour résoudre le problème du voyageur de commerce à partir de v_1



Exercice

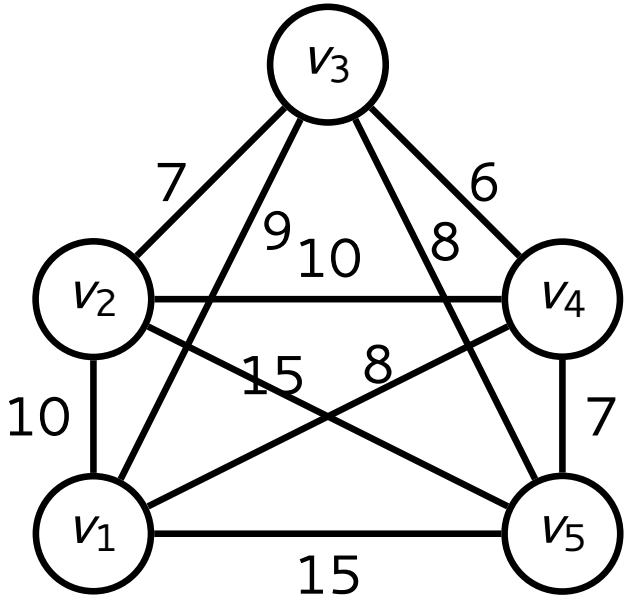
Dérouler l'algorithme glouton du plus proche voisin pour résoudre le problème du voyageur de commerce à partir de v_1



Tournée : $v_1 - v_4 - v_3 - v_2 - v_5$ de coût 36

Exercice

Dérouler l'algorithme glouton du plus proche voisin pour résoudre le problème du voyageur de commerce à partir de v_1



Tournée : $v_1 - v_4 - v_3 - v_2 - v_5$ de coût 36

Or par exemple la tournée : $v_1 - v_3 - v_2 - v_4 - v_5$ est de coût 33

Bon courage !

Quelques idées

- Tester différentes heuristiques ou algorithmes approchés
  - Regarder l'impact sur le temps de calcul
  - Regarder l'impact sur le résultat
- Comparer avec l'algorithme exhaustif
- Quels paramètres du labyrinthe sont les plus favorables aux algorithmes gloutons ?
- Peut-on sacrifier des fromages et minimiser grandement le nombre de déplacements ?