Méthodes arborescentes exactes et approchées

Complexité, Algorithmes Randomisés et Approchés

September 28, 2016

Problème d'optimisation

Définition :

- Nom du problème : P
- Paramètres génériques du problème (nombres, graphes, ...)
- Une caractérisation de ce qu'est une solution réalisable :
  - Une instance I de P est une instanciation des paramètres génériques
  - A chaque I correspond un ensemble de solutions réalisables $S(I)$.
- Une fonction objectif f

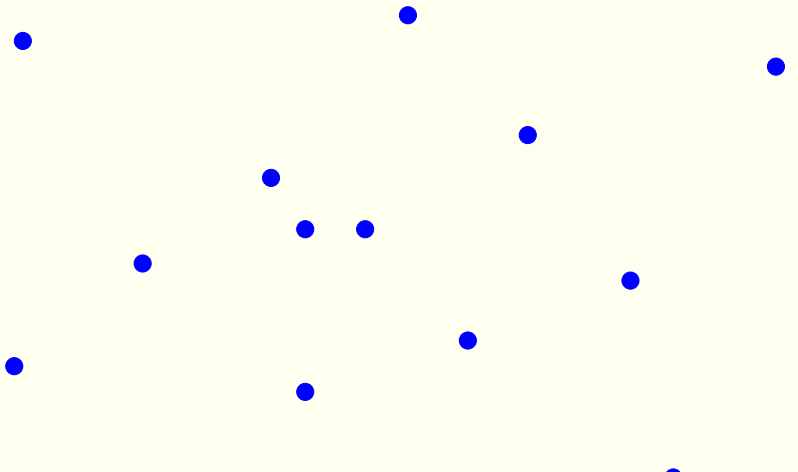
Résolution :

Déterminer un algorithme A qui, pour chaque instance I retourne une solution $s^*(I)$ de $S(I)$ t.q. :

problème de minimisation : $\forall s \in S(I), f(s^*(I)) \leq f(s)$, ou

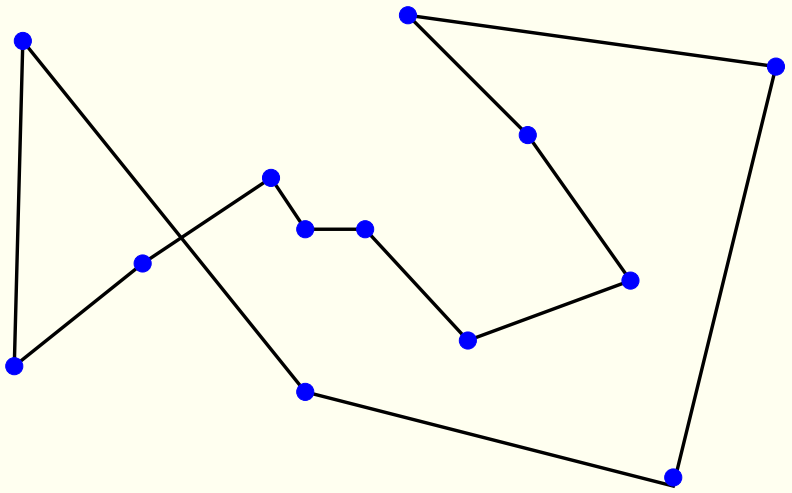
problème de maximisation : $\forall s \in S(I), f(s^*(I)) \geq f(s)$.

Exemple : le voyageur de commerce euclidien



Données : n points dans le plan

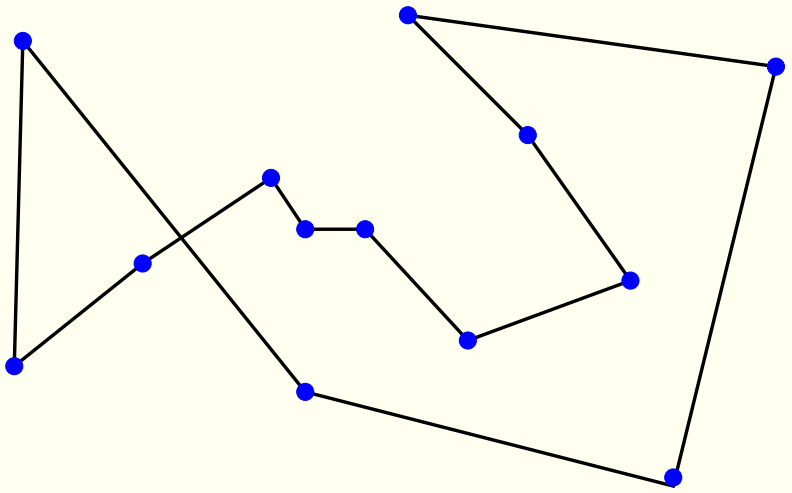
Exemple : le voyageur de commerce euclidien



Données : n points dans le plan

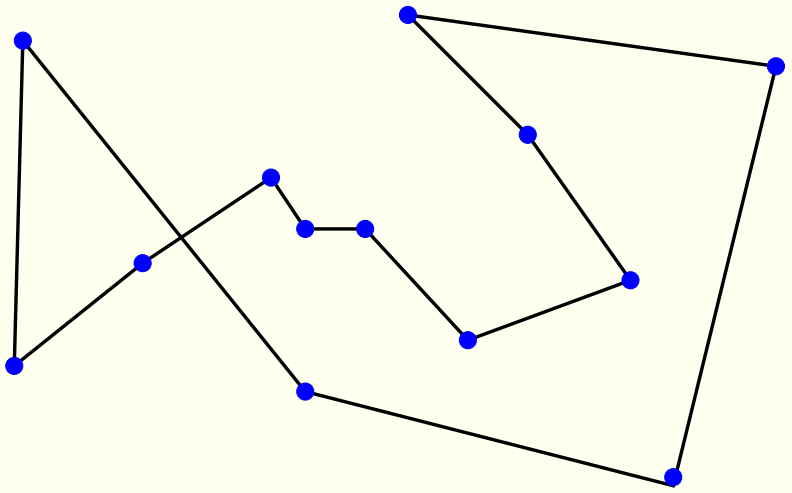
Solution réalisable : Un cycle hamiltonien dans le graphe complet sous-jacent K_n

Exemple : le voyageur de commerce euclidien



Données : n points dans le plan
Solution réalisable : Un cycle hamiltonien dans le graphe complet sous-jacent K_n
Fonction objectif : La longueur du cycle (que l'on souhaite minimiser)

Exemple : le voyageur de commerce euclidien



Données : n points dans le plan
Solution réalisable : Un cycle hamiltonien dans le graphe complet sous-jacent K_n
Fonction objectif : La longueur du cycle (que l'on souhaite minimiser)

Le problème de décision associé à ce problème est NP-complet.
$\Rightarrow$ ce problème est NP-difficile.

Trouver une solution optimale du TSP

Première idée : énumérer l'ensemble des solutions réalisables du TSP : $\frac{(n-1)!}{2}$ cycles possibles

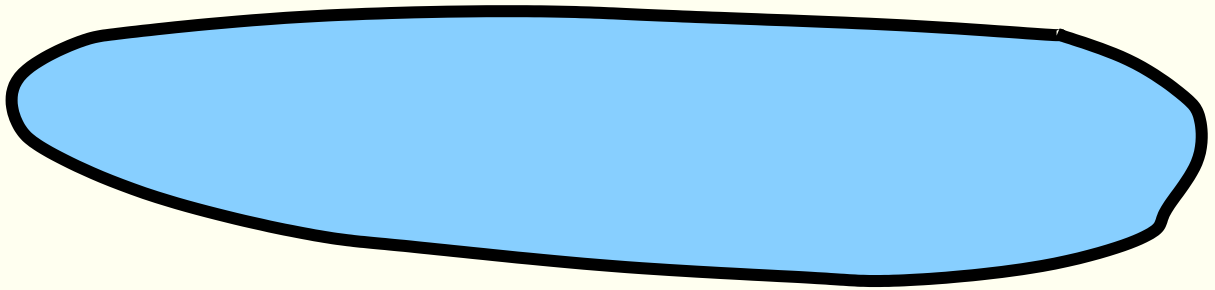
But : Trouver une solution optimale sans énumérer toutes les solutions.

“Branch-and-bound” (séparation-évaluation)

- “Branch” (brancher)
  - Diviser (partitionner) l’espace de recherche
    $\rightarrow$ Arbre d’énumération (ou arbre de recherche)
  - Explorer l’arbre de recherche

Brancher

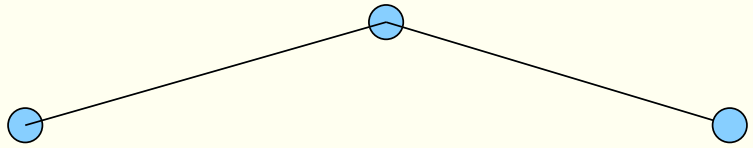
La racine de l'arbre représente l'ensemble des solutions.
Chaque sous-arbre représente une solution partielle.

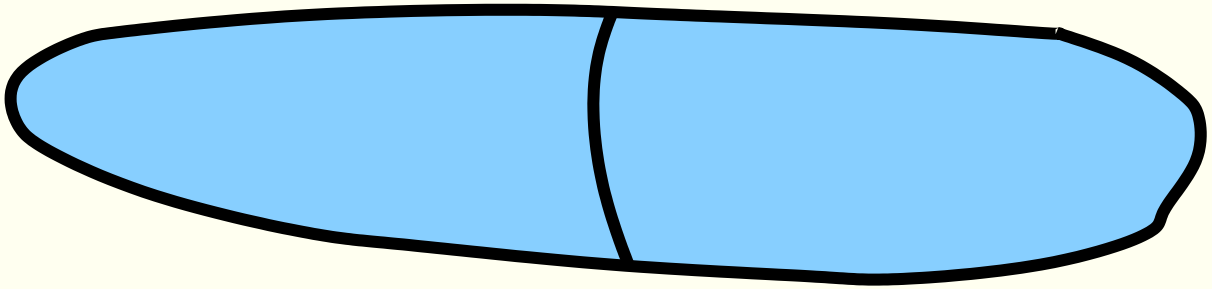


Espace des solutions

Brancher

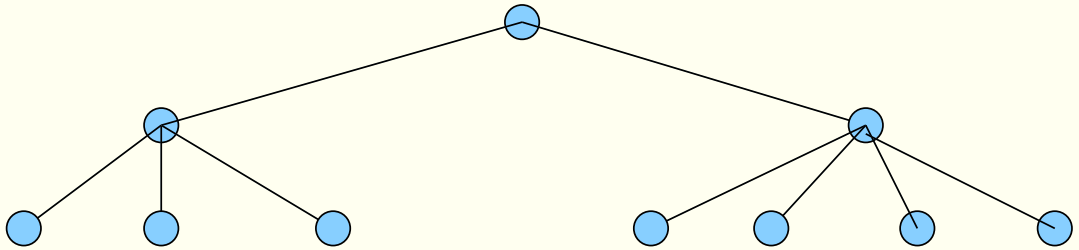
La racine de l'arbre représente l'ensemble des solutions.
Chaque sous-arbre représente une solution partielle.

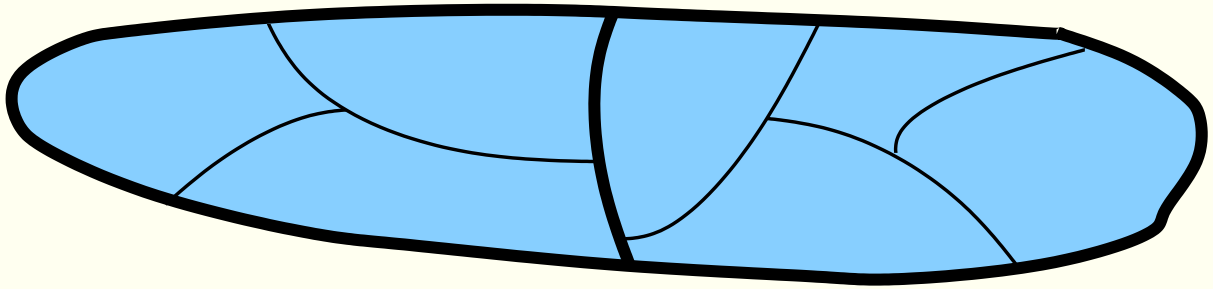




Brancher

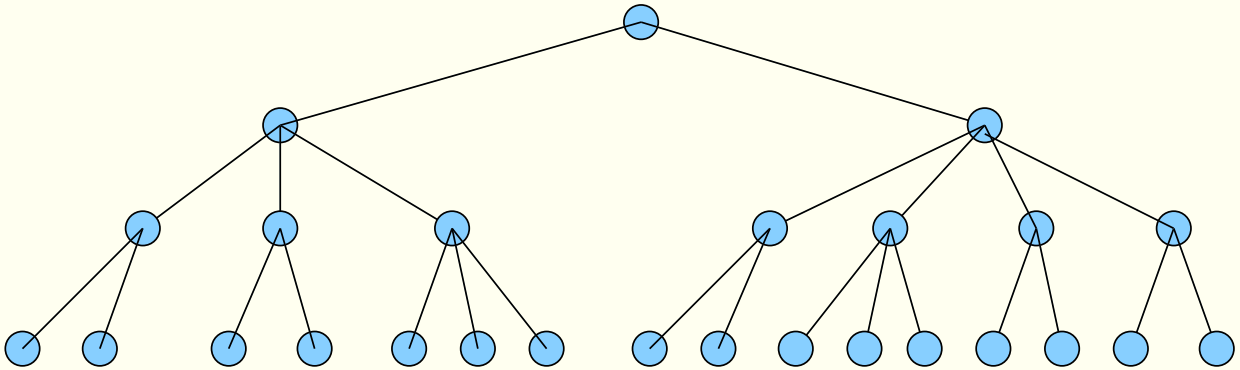
La racine de l'arbre représente l'ensemble des solutions.
Chaque sous-arbre représente une solution partielle.

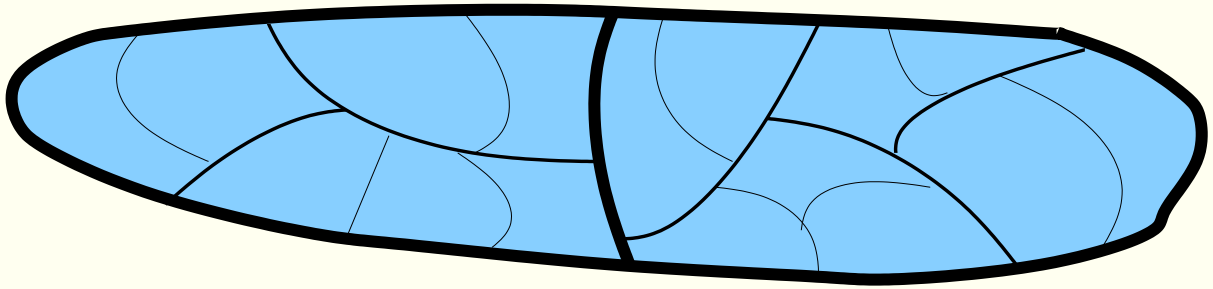




Brancher

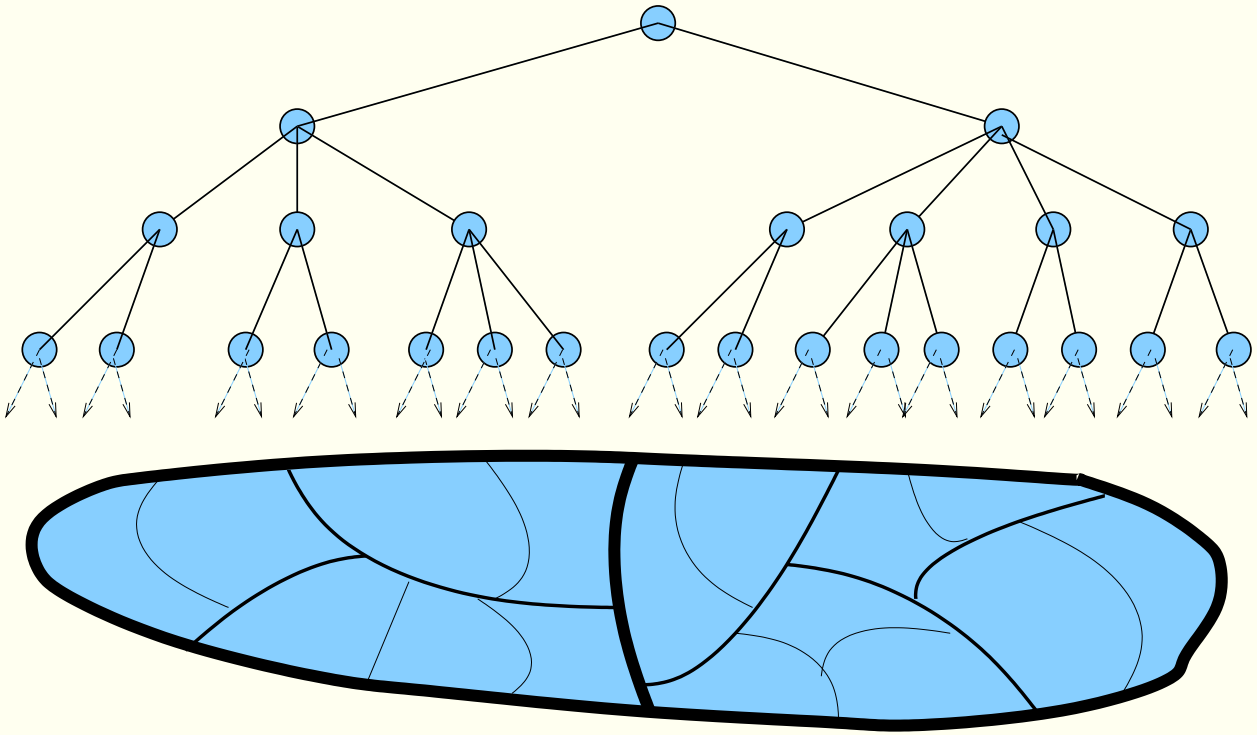
La racine de l'arbre représente l'ensemble des solutions.
Chaque sous-arbre représente une solution partielle.





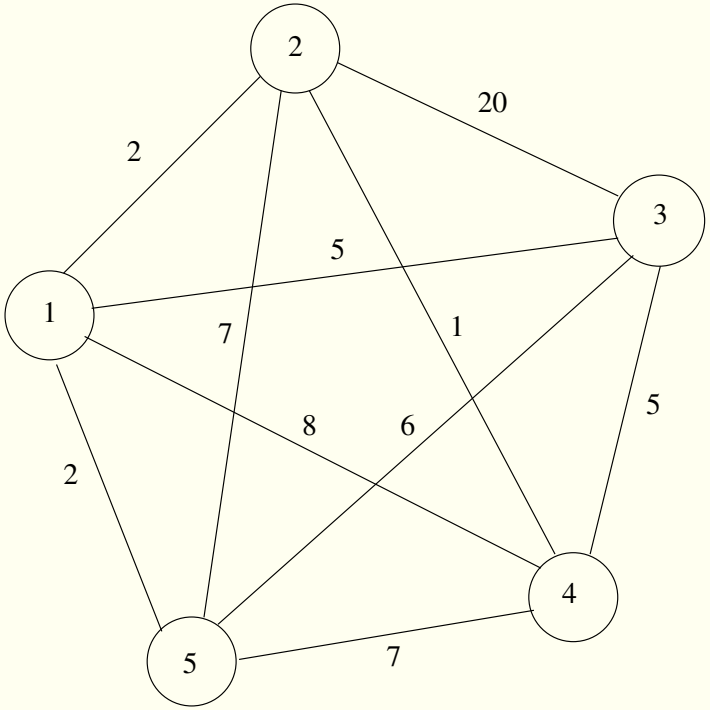
Brancher

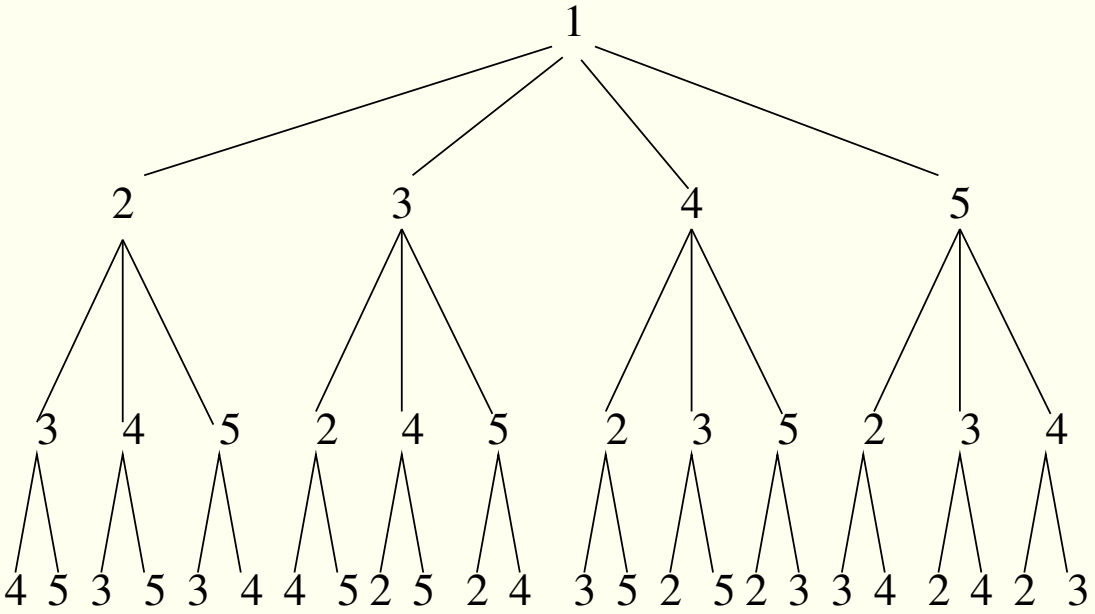
La racine de l'arbre représente l'ensemble des solutions.
Chaque sous-arbre représente une solution partielle.



Brancher : exemple sur le problème du TSP

Soit 1 le premier sommet du cycle.
On choisit au niveau i le i^{eme} sommet du cycle ($n - i$ choix).





“Branch-and-bound” (séparation-évaluation)

- “Branch” (brancher)
  - Diviser (partitionner) l’espace de recherche
    $\rightarrow$ Arbre d’énumération (ou arbre de recherche)
  - Explorer l’arbre de recherche

- “Bound” (borner) (présenté pour un pb de minimisation)
  - Borne supérieure de la valeur d’une solution optimale
  - Borne inférieure de la valeur d’un nœud (et de son sous-arbre)

Borner

Au noeud courant on a :

- une borne supérieure B_{sup} d'une solution optimale (par exemple le coût d'une solution réalisable que l'on a déjà rencontrée)
  $\rightarrow$ ce que l'on a déjà
- une borne inférieure B_{inf} du coût de toute solution issue du noeud courant (borne inf de toute solution du sous-arbre courant)
  $\rightarrow$ ce que l'on peut espérer avoir de mieux en explorant le sous-arbre

Si $B_{inf} > B_{sup}$ alors on "élague" : on n'explore pas le sous-arbre enraciné au noeud courant.

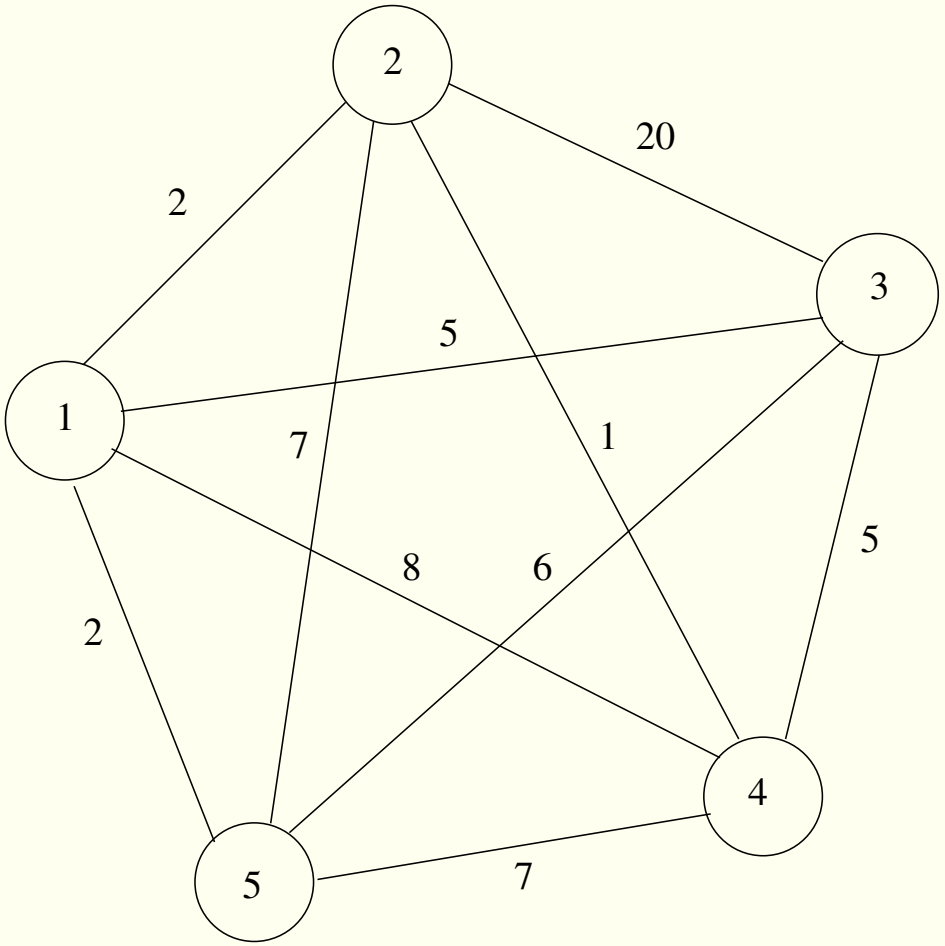
Borner : exemple sur le problème du TSP

Soit $G = (S, A)$ un graphe valué. Soit $i \in S$ un sommet. Soit $cout\_adj\_min1(i)$ le coût de la plus petite arête adjacente à i et $cout\_adj\_min2(i)$ le coût de la 2ème plus petite arête adjacente à i.

Propriété : Le coût d'un cycle hamiltonien de G est $\geq \frac{1}{2} \sum_{i=1}^{n} cout\_adj\_min1(i) + cout\_adj\_min2(i)$.

Borne inférieure du coût d'une solution dont les sommets S' forment un cycle partiel $(s_1, \ldots, s_k)$ = coût des arêtes du cycle partiel
$+\frac{1}{2}(cout\_adj\_min1(s_1) + cout\_adj\_min1(s_k))$
$+\frac{1}{2} \sum_{i \in S \setminus S'} (cout\_adj\_min1(i) + cout\_adj\_min2(i))$

Exemple



“Branch-and-bound” (séparation-évaluation)

Un algorithme de branch-and-bound pour un problème de minimisation est basé sur

- une procédure de branchement qui décompose le problème,

  et

- une borne inférieure pour éviter d’avoir à parcourir tout l’arbre.

Arbre d'énumération

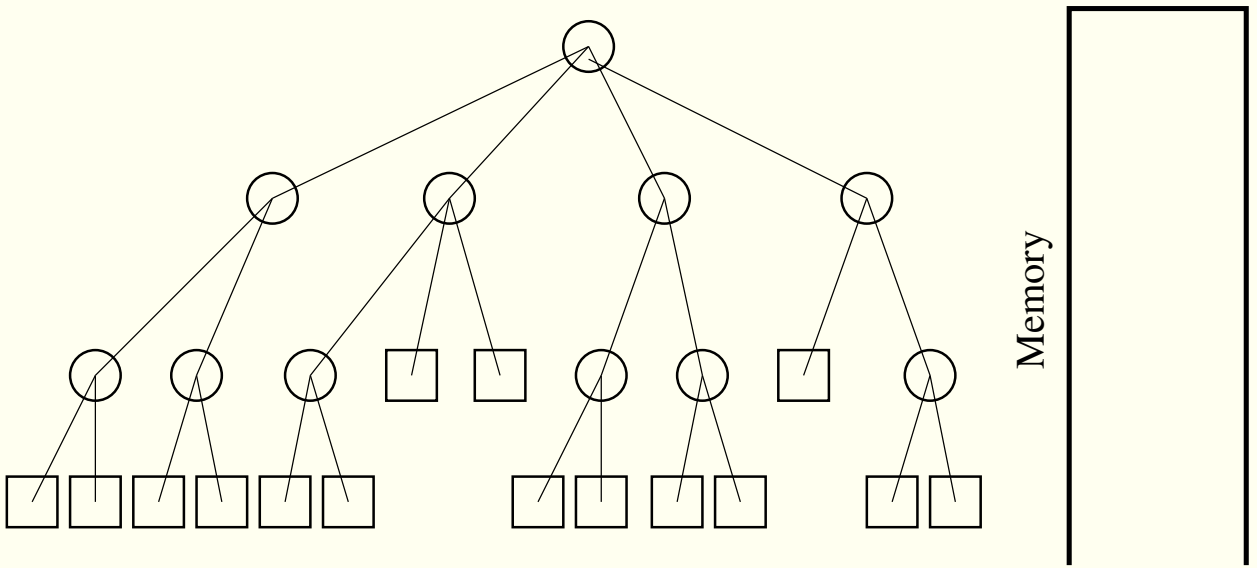
- L'arbre d'énumération n'est pas complètement stocké en mémoire.
- En effet, sa taille est proportionnelle à la taille de l'ensemble des solutions, qui est exponentielle.
- L'arbre d'énumération est exploré pour trouver la solution optimale.

Il y a deux façons classiques d'explorer l'arbre :

- Parcours en profondeur
- Parcours "le meilleur d'abord"

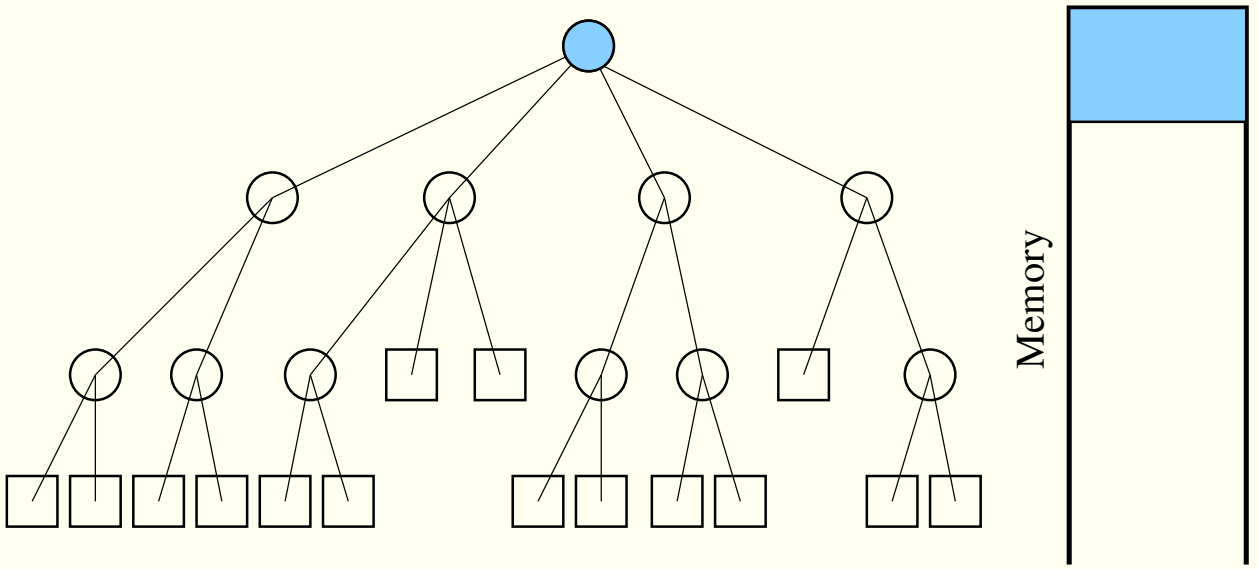
Explorer l'arbre d'énumération

Parcours en profondeur



Explorer l'arbre d'énumération

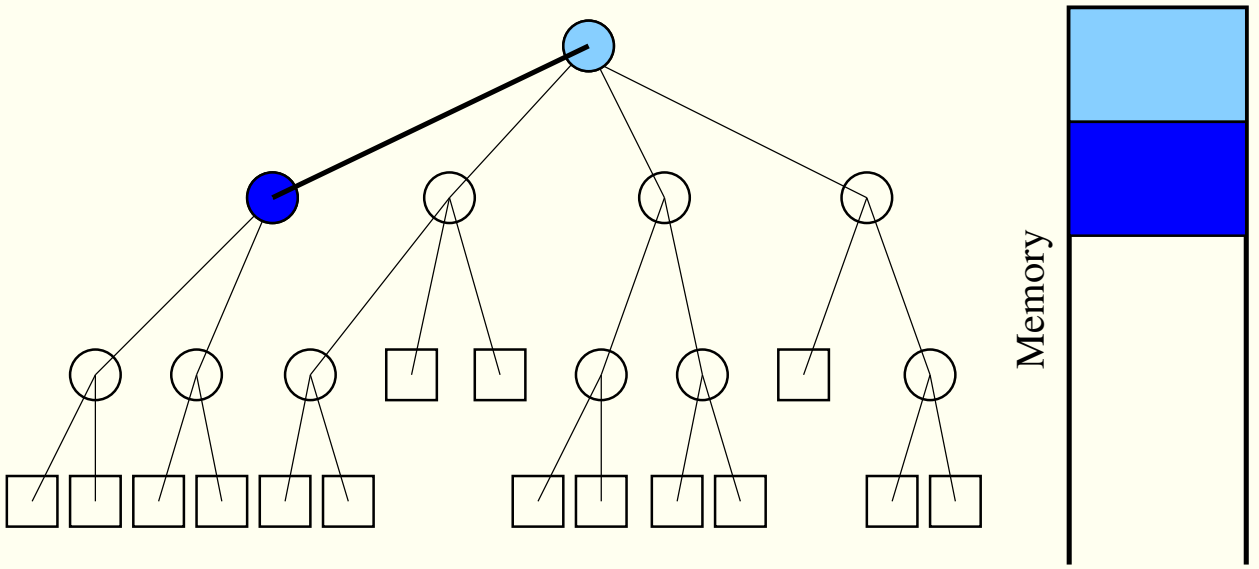
Parcours en profondeur



Charger le problème en mémoire

Explorer l'arbre d'énumération

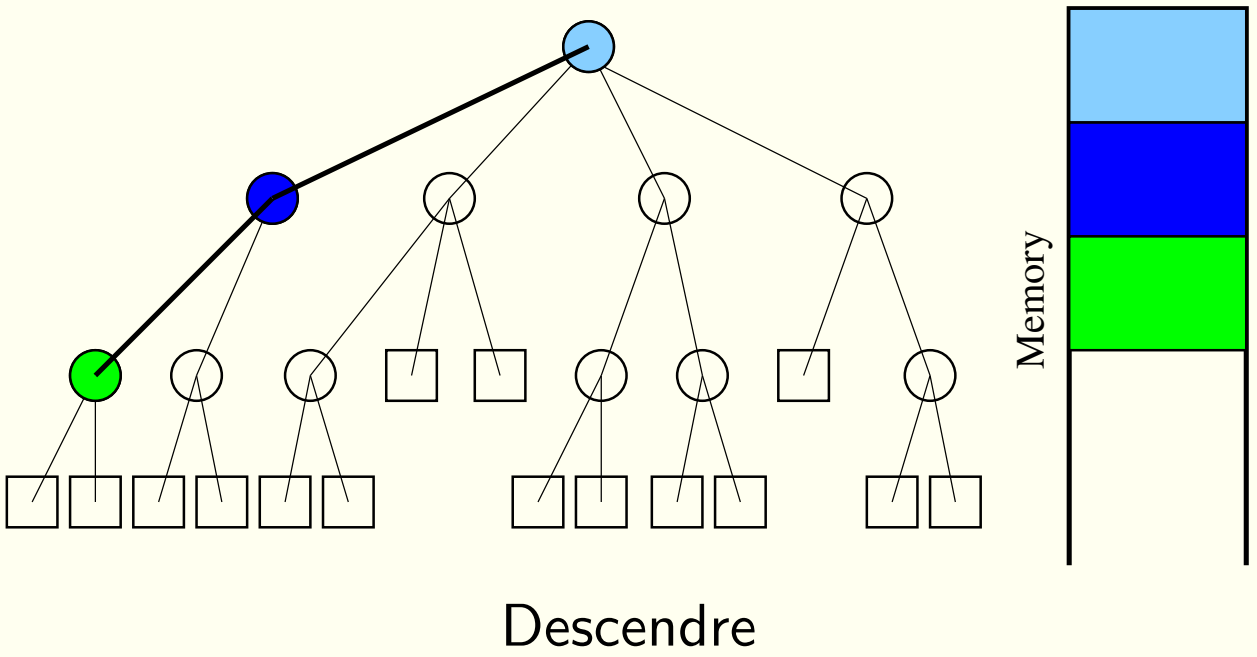
Parcours en profondeur



Première branche = premier sous-problème

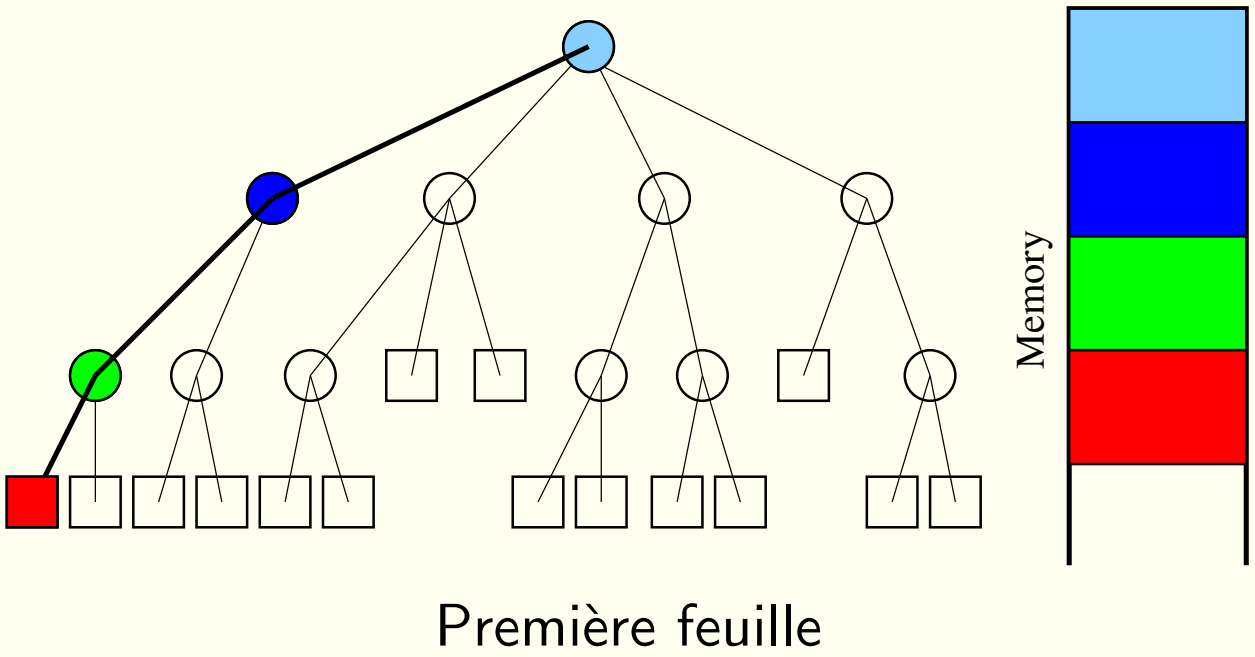
Explorer l'arbre d'énumération

Parcours en profondeur



Explorer l'arbre d'énumération

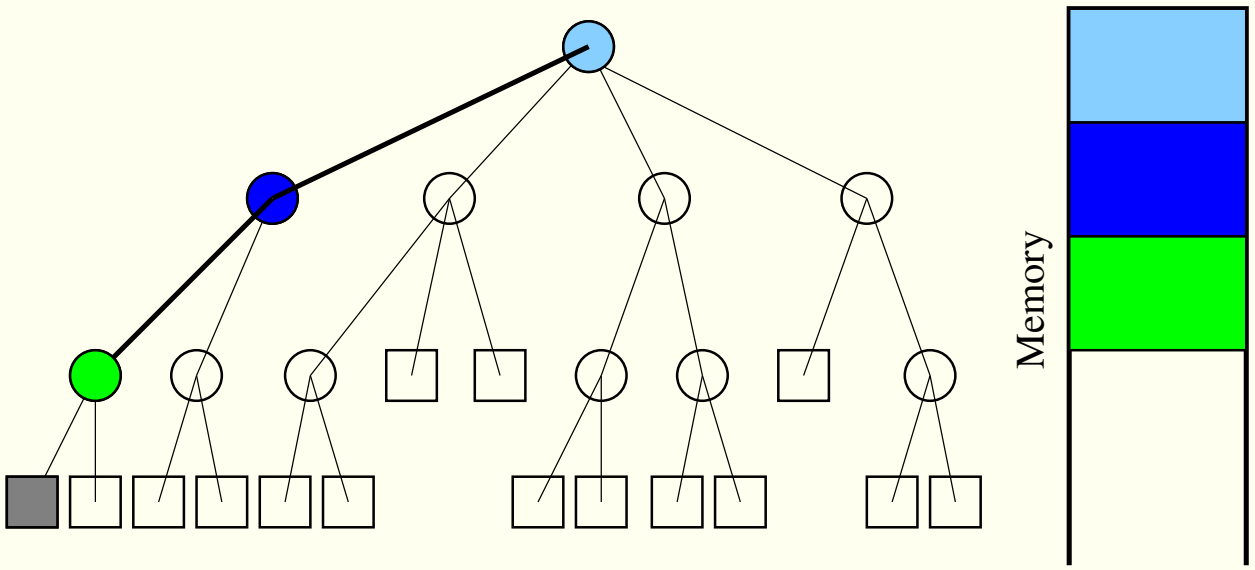
Parcours en profondeur



Première feuille

Explorer l'arbre d'énumération

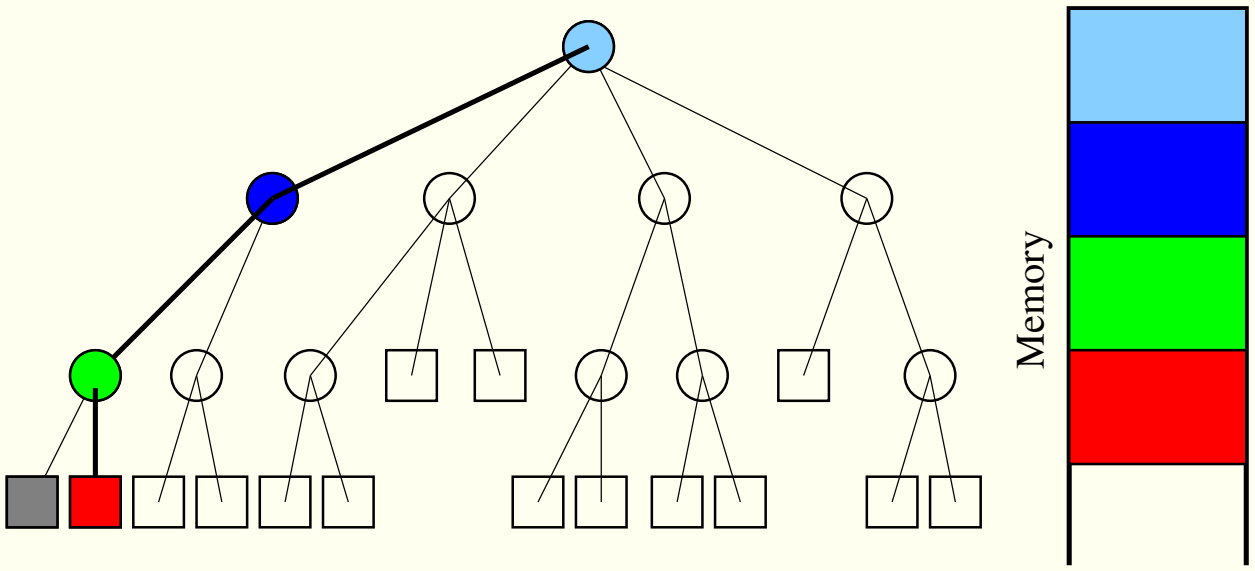
Parcours en profondeur



Premier retour en arrière

Explorer l'arbre d'énumération

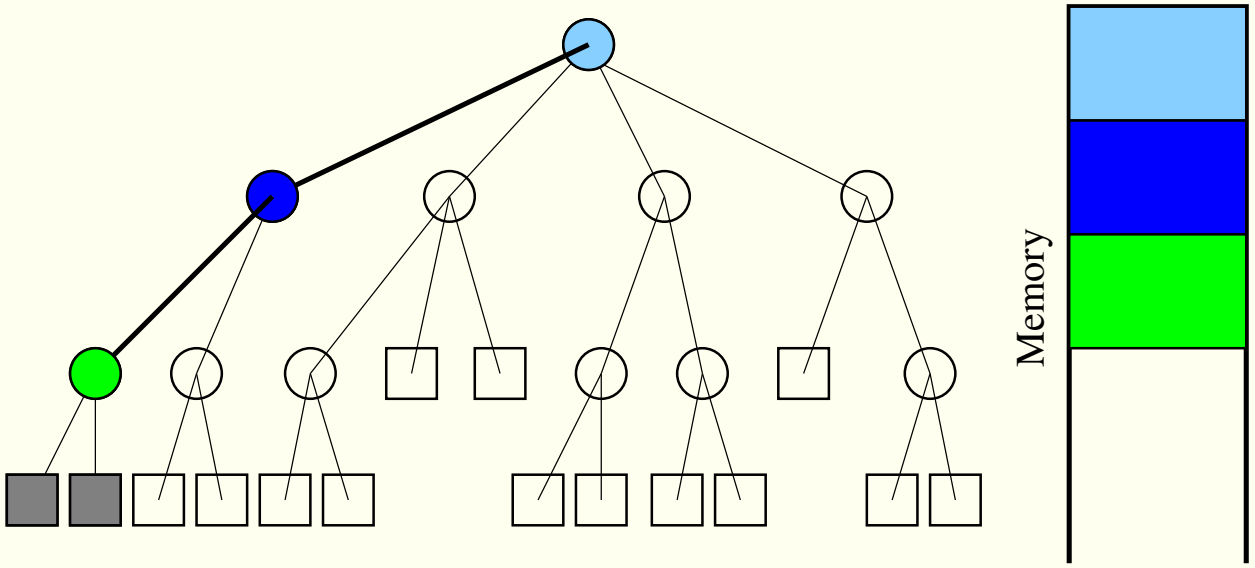
Parcours en profondeur



Seconde feuille

Explorer l'arbre d'énumération

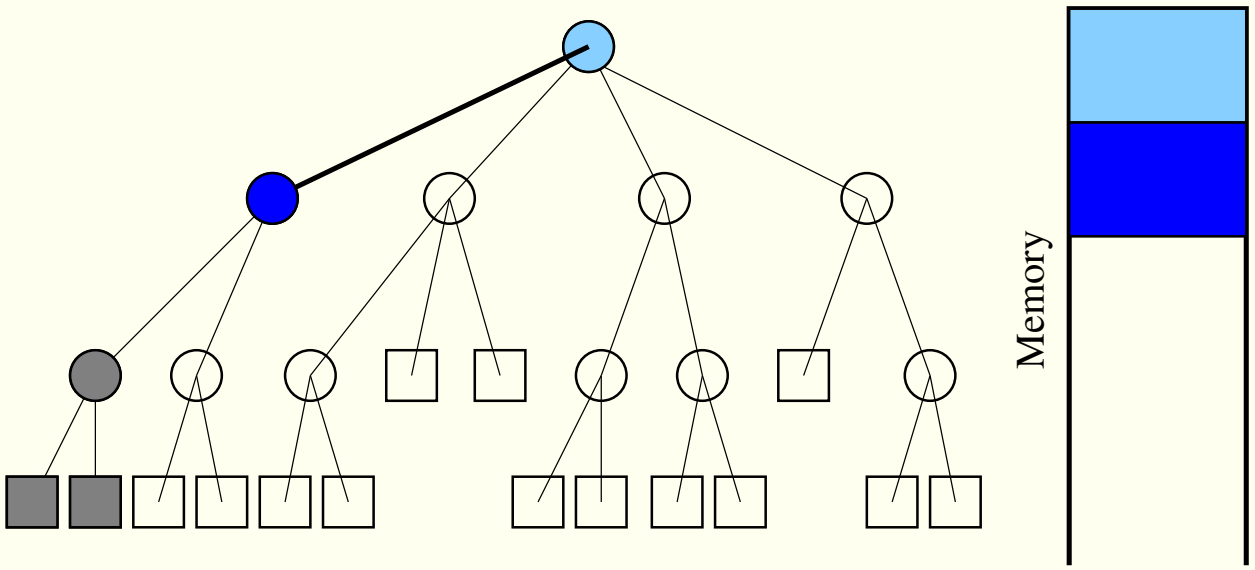
Parcours en profondeur



Deuxième retour en arrière

Explorer l'arbre d'énumération

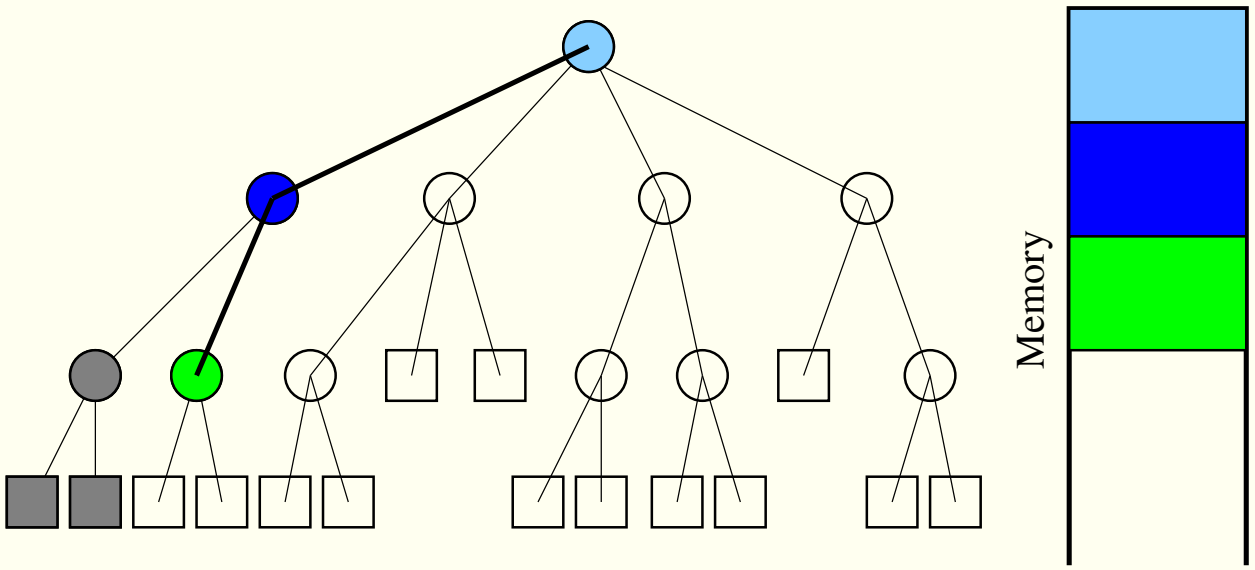
Parcours en profondeur



et retourner en arrière à nouveau

Explorer l'arbre d'énumération

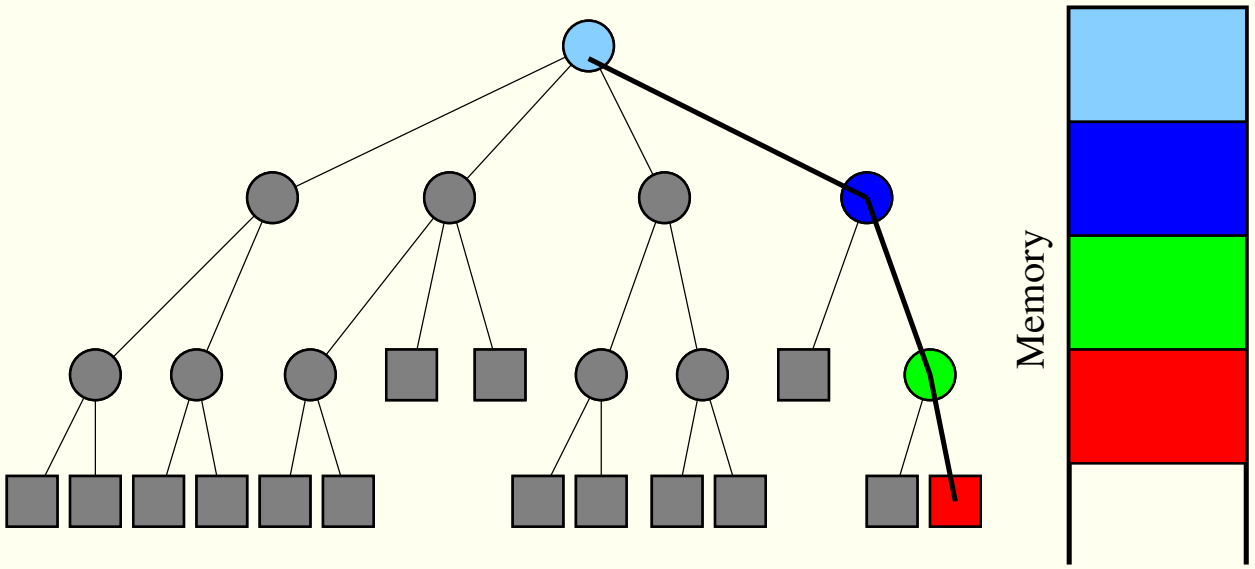
Parcours en profondeur



et ainsi de suite...

Explorer l'arbre d'énumération

Parcours en profondeur



...jusqu'à la dernière feuille

Complexité

- Complexité temporelle : généralement exponentielle en la taille du problème.
- Complexité en espace : en $O(hn)$ avec n taille du problème et h hauteur de l'arbre d'énumération (h est polynomial en n).