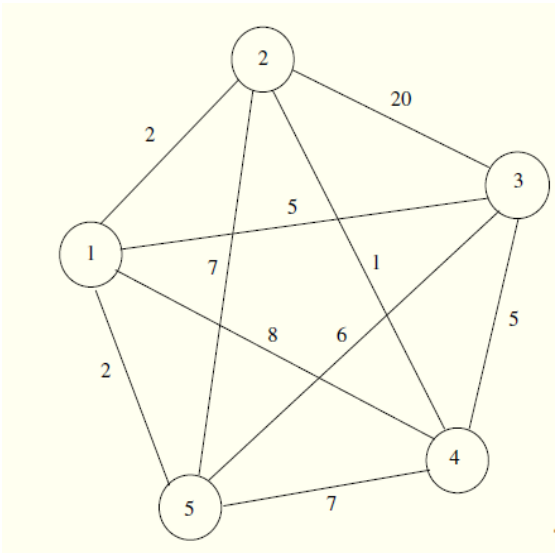
Méthodes arborescentes : rappel de cours et exemple sur le problème du voyageur de commerce.

Un algorithme de branch and bound (ou séparation-évaluation) permet de trouver une solution optimale à un problème d'optimisation, et ce (souvent) bien plus rapidement que si l'on examinait toutes les solutions réalisables possibles. Il repose sur deux principes :

Brancher : il s'agit de créer un arbre de recherche tel que au moins une feuille de cet arbre représente est une solution optimale. Cet arbre, s'il n'est pas élagué (nous verrons ce que cela signifie par la suite), peut contenir autant de feuilles qu'il existe de solution optimales au problème considéré. Cet arbre sera ensuite parcouru (puis élagué) jusqu'à ce que l'on soit sûr que l'on a trouvé une solution optimale.

Exemple sur le problème du voyageur de commerce (le but est ici de trouver un cycle hamiltonien de coût minimum, où le coût du cycle est la somme des coûts de ses arêtes) :

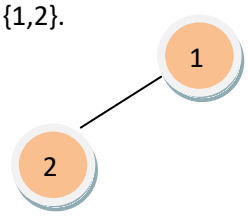
On cherche un tour (cycle hamiltonien) de coût minimum dans le graphe suivant :



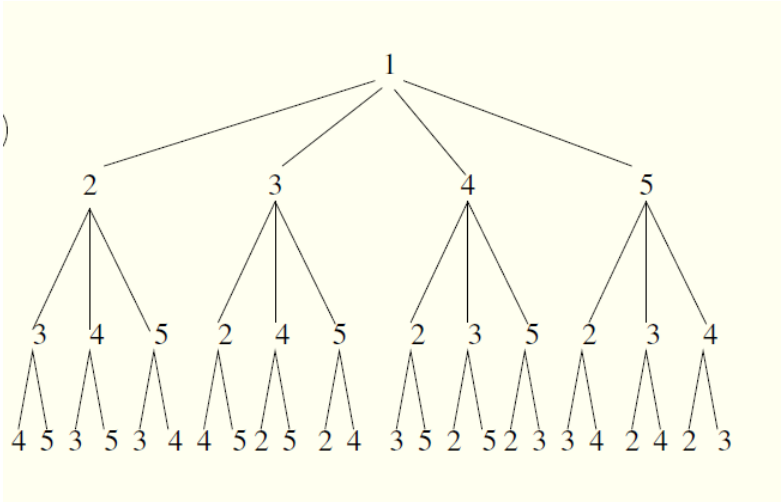
L'arbre de recherche considéré est le suivant :

- la racine représente le cas où le sommet 1 appartient au cycle retourné : elle représente donc l'ensemble des tours possibles (un cycle hamiltonien comportant une fois chaque sommet du graphe). J'indique à l'intérieur de chaque nœud de l'arbre le sommet du graphe qu'il représente.

- Chaque sommet représente un tour partiel. Par exemple, le nœud 2 ci-dessous représente ici le tour partiel (1,2) : le sous-arbre enraciné au sommet 2 contiendra tous les cycles hamiltoniens qui empruntent l'arête {1,2}.



- Un nœud a autant de fils qu'il y a de possibilité de poursuivre le tour partiel. L'arbre final est le suivant (on aurait pu ajouter à chaque feuille un seul fils indiquant le dernier sommet du tour, mais on peut aussi, quand on a (n-1) sommets dans le tour en déduire directement le dernier sommet du tour).



Avec un graphe à n sommets, cet arbre contiendrait (n-1) ! feuilles (chacune correspondant à un cycle hamiltonien). Nous allons maintenant voir comment trouver une solution optimale sans avoir à parcourir nécessairement tout cet arbre.

Borner :

On souhaite avoir, à chaque nœud de l'arbre, une borne inférieure de la valeur du coût de toute solution correspondant à une feuille du sous-arbre enraciné à cette feuille. Cette valeur sera comparée à une borne supérieure de la valeur d'une solution optimale (cette borne sup pourra tout simplement être la valeur d'une solution déjà rencontrée). Si la borne inf est supérieure à la borne sup, alors on ne parcourt pas le sous-arbre.

Retour à l'exemple sur le problème du voyageur de commerce :

A chaque nœud, la borne sup sera le coût de la meilleure solution rencontrée, et la borne inf sera

celle expliquée dans les transparents :

Soit $G = (S, A)$ un graphe valué. Soit $i \in S$ un sommet. Soit $min_1(i)$ le coût de la plus petite arête adjacente à i et $min_2(i)$ le coût de la 2ème plus petite arête adjacente à i.

Propriété : Le coût d'un cycle hamiltonien de G est $\geq \frac{1}{2} \sum_{i=1}^{n} min_1(i) + min_2(i)$.

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 $+\frac{1}{2}(min_1(s_1) + min_1(s_k)) + \frac{1}{2} \sum_{i \in S \setminus S'} (min_1(i) + min_2(i))$

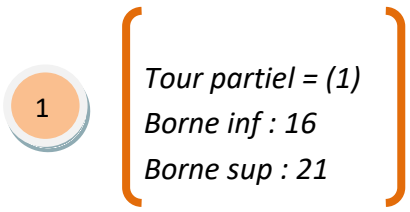
L'idée est que le coût de chaque arête est partagé équitablement entre ses deux extrémités. Le coût d'un cycle est alors la somme des coûts de ses sommets. Le coût minimal que peut avoir un sommet est donc la moitié de la somme des coûts des ses deux arêtes adjacentes de plus petit coût.

A chaque sommet de l'arbre de recherche, la borne inférieure considérée est ainsi le coût du cycle partiel (on emprunte dans tous les cas ces arêtes) plus le coût minimal que peut induire chacun des sommets restants (ce coût est égal à la moitié du coût d'une arête adjacente à i de coût minimum si i est une extrémité du tour partiel, et à la moitié de la somme des coûts des deux arêtes adjacentes de plus petit coût pour les autres sommets). Si cette borne n'est pas un entier, on prend la borne entière supérieure (car comme les coûts des arêtes sont des entiers, le tour optimal a un coût qui est un entier).

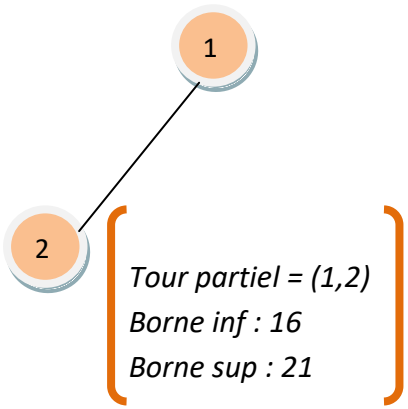
Exécution de l'algorithme de branch and bound :

On regarde un tour – au hasard le tour (1 ,2,4,5,3,1) – et on mémorise son coût (2+1+7+6+5=21 ici). Ce sera une borne supérieure au coût d'un tour optimal.

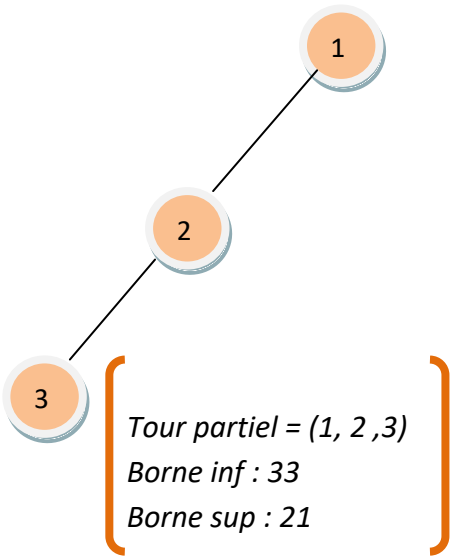
On commence alors à parcourir l'arbre de recherche (en calculant à chaque nœud la borne inférieure). J'indique à côté du nœud que l'on examine le tour partiel qu'il représente ainsi que la borne inf (calculée comme expliqué ci-dessus) et la borne sup (coût de la meilleure solution rencontrée jusqu'à présent).



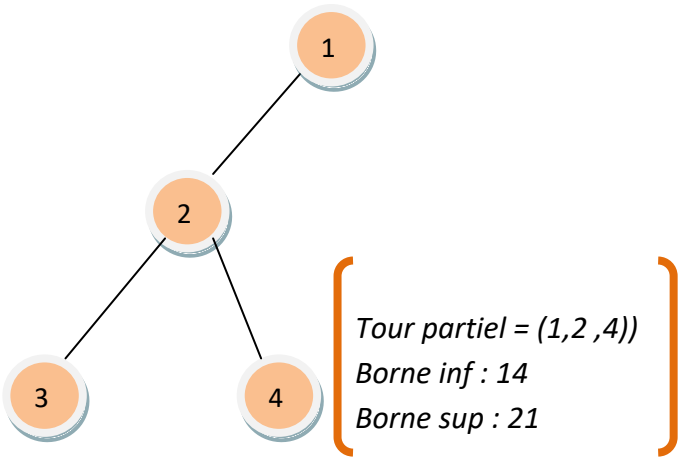
La borne inf est inférieure à la borne sup, on branche sur un fils et on examine le nœud créé :



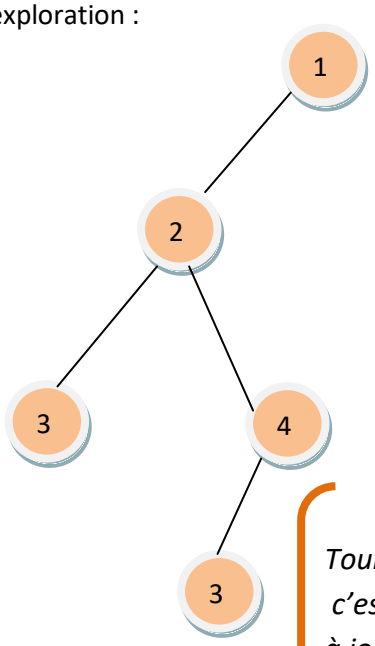
Comme la borne inf est inférieure à la borne sup (et que le nœud n'est pas une feuille), on continue l'exploration :



Comme la borne inf est supérieure à la borne sup, on n'explore pas ce sous-arbre (en effet, tout tour contenant le tour partiel (1 ,2 ,3), i.e. les arêtes {1,2} et {2,3}, a un coût d'au moins 33 et on a déjà trouvé un tour de coût 21 : il n'y aura pas de tour optimal dans ce sous-arbre. On continue alors l'exploration en retournant au nœud père du nœud élagué. Ici, on explore un autre de ses fils.



On continue l'exploration :



Tour partiel = (1,2 ,4,3,5,1) :
c'est une feuille de coût 16. On met à jour la meilleure solution rencontrée (qui était pour l'instant la solution initiale de coût 21).

Borne sup = 16

On a trouvé une solution de coût 16, la borne sup devient donc 16. On remonte ensuite aux nœuds 4, on élague, puis au nœud 2, où l'on élague aussi, et enfin à la racine, où l'on élague également car la borne inf (16) est égale à la borne sup. On arête donc l'exploration de l'arbre : le tour (1, 2, 4, 3, 5, 1) est un cycle hamiltonien de coût minimum.