Cours de Robotique 2

MENASRI Riad

14-11-2014

menasri.riad@gmail.com

Plan du cours

- Algorithme de recherche informés
- Techniques d'optimisation
- Contrôle/Commande

Recherche meilleur d'abord

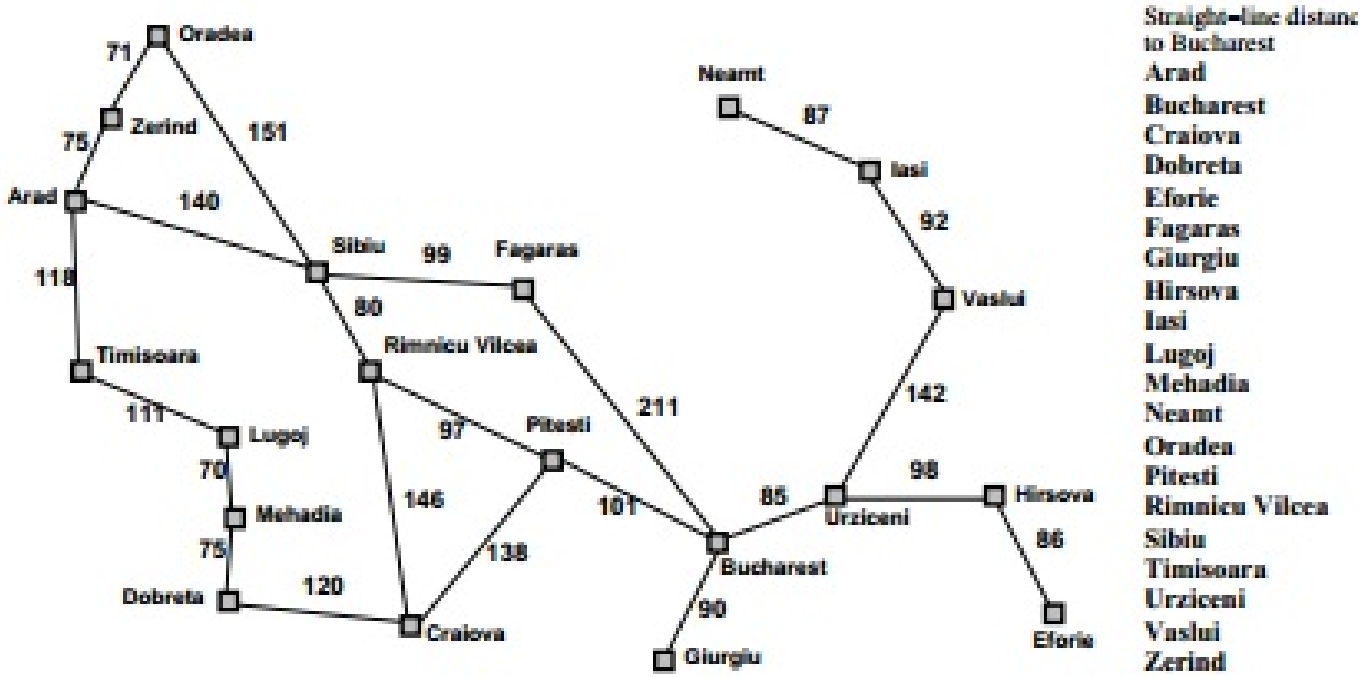
- Une stratégie est définie en choisissant un ordre dans lequel les états sont développées.
- Idée : Utiliser une fonction d'évaluation pour chaque noeud
  - mesure l'utilité d'un noeud.
- QUEING-FN : insérer le noeud en ordre décroissant d'utilité

Recherche meilleur d'abord

function BEST-FIRST-SEARCH(*problem*, EVAL-FN) **returns** a solution sequence
  inputs: *problem*, a problem
    Eval-Fn, an evaluation function

  Queueing-Fn ← a function that orders nodes by EVAL-FN
  return GENERAL-SEARCH(*problem*, *Queueing-Fn*)

Roumanie avec coût de chaque pas



Recherche gloutonne

- Fonction d'évaluation (heuristique) $h(n)$
- $h(n)$= estimation du coût de n vers l'état final
- Par exemple h_{dd}= distance directe entre la ville n et Bucharest.
- La recherche gloutonne développe le noeud qui paraît être le plus proche de l'état final
- function GREEDY-SEARCH(problem) returns a solution or failure, return BEST-SEARCH(problem,h)

Recherche gloutonne

- Incomplet(peut aller dans des boucles)
  - Complet si on ajoute test d'état répété.
- Non optimale

L'algorithme A^*

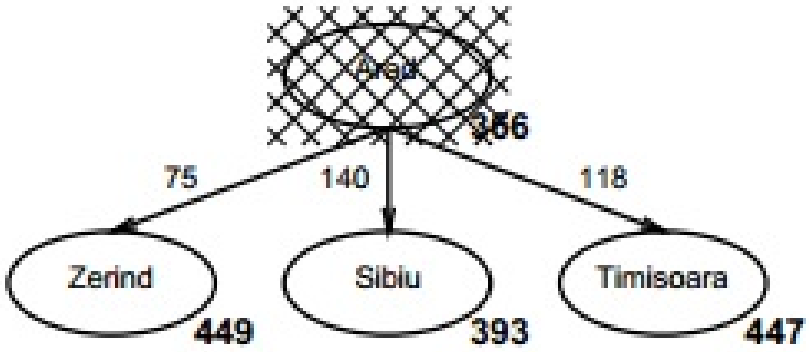
- Idée : Éviter de développer des chemins qui sont déjà chers
- Fonction d'évaluation : $f(n) = g(n) + h(n)$
  - $g(n)$= coût jusqu'à présent pour atteindre n
  - $h(n)$= coût estimé pour aller vers un état final
  - $f(n)$= coût total estimé pour aller vers un état final en passant par n
- A^* utilise une heuristique admissible :
  $h(n) <= h^*(n)$ où $h^*(n)$ est le vrai coût pour aller de n vers un état final
- Par exemple h_{dd} ne sur-estime jamais la vrai distance

L'algorithme A^*

Exemple : A^*

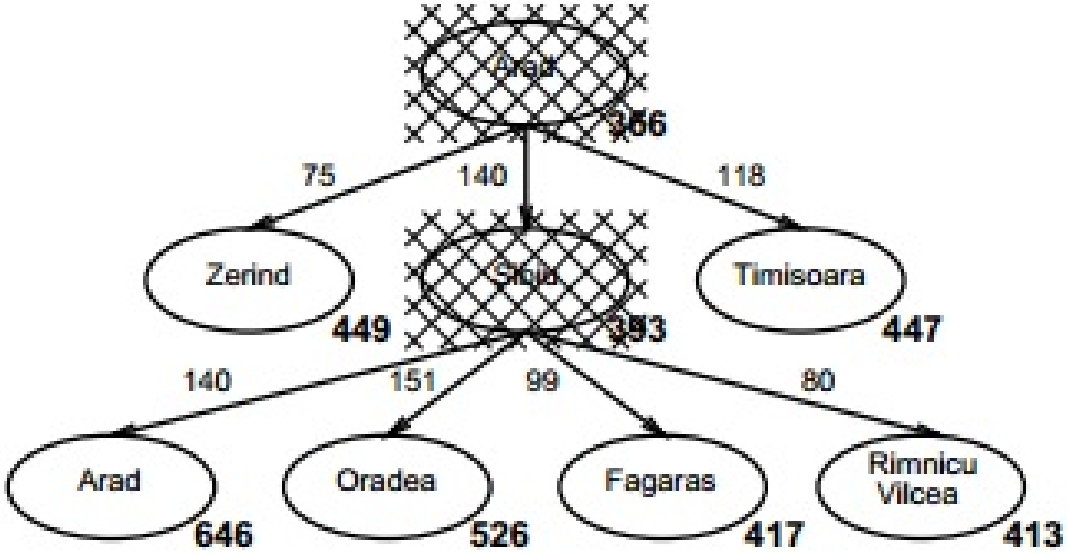
L'algorithme A^*

Exemple : A^*



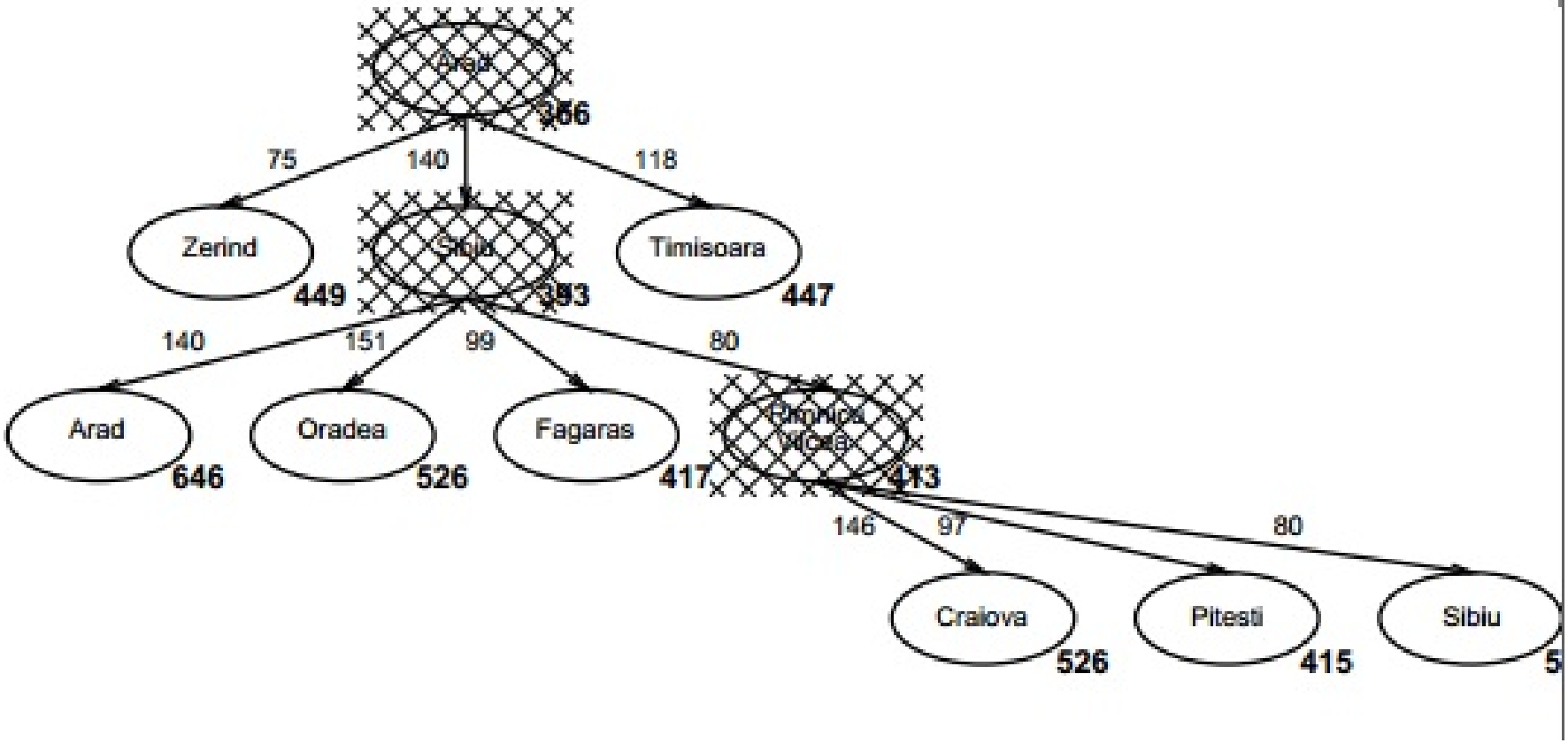
L'algorithme A^*

Exemple : A^*



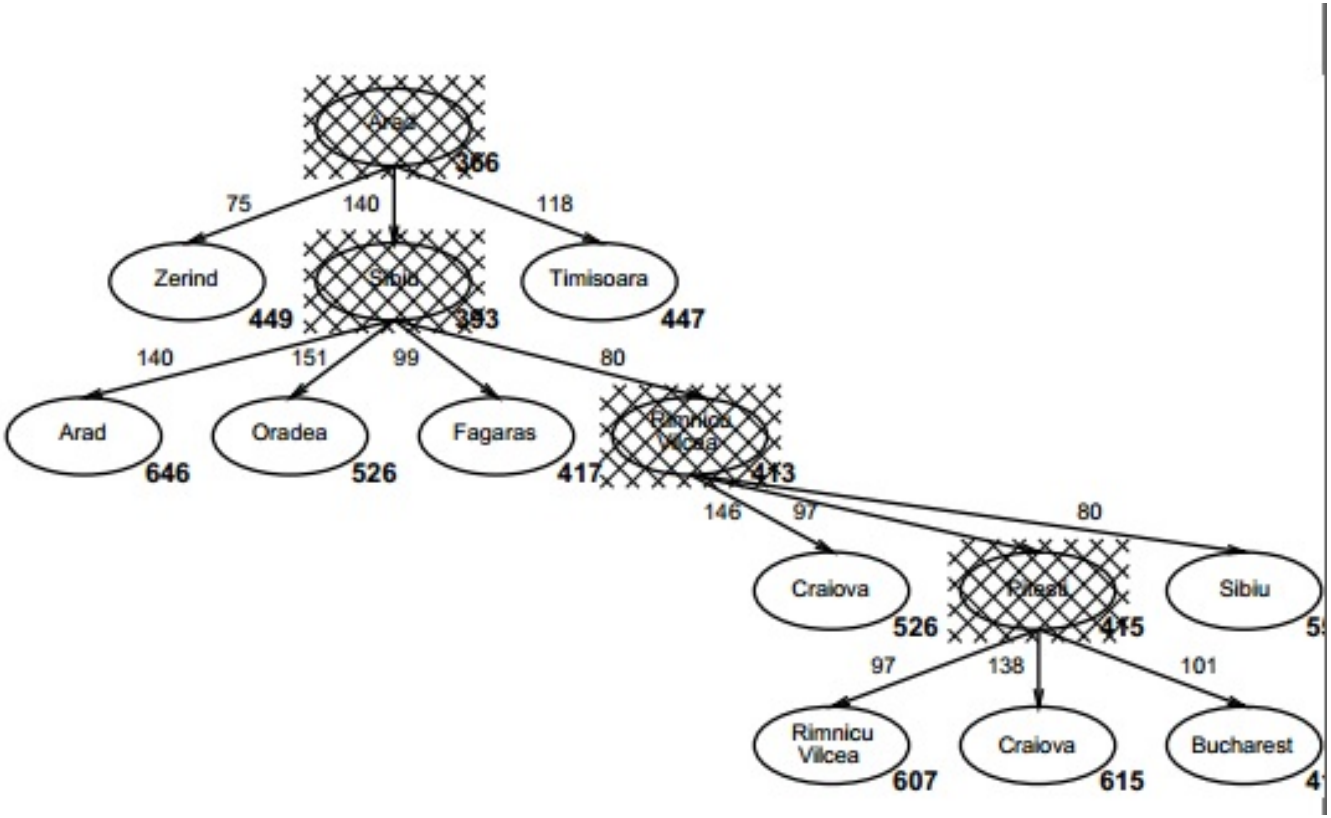
L'algorithme A^*

Exemple : A^*



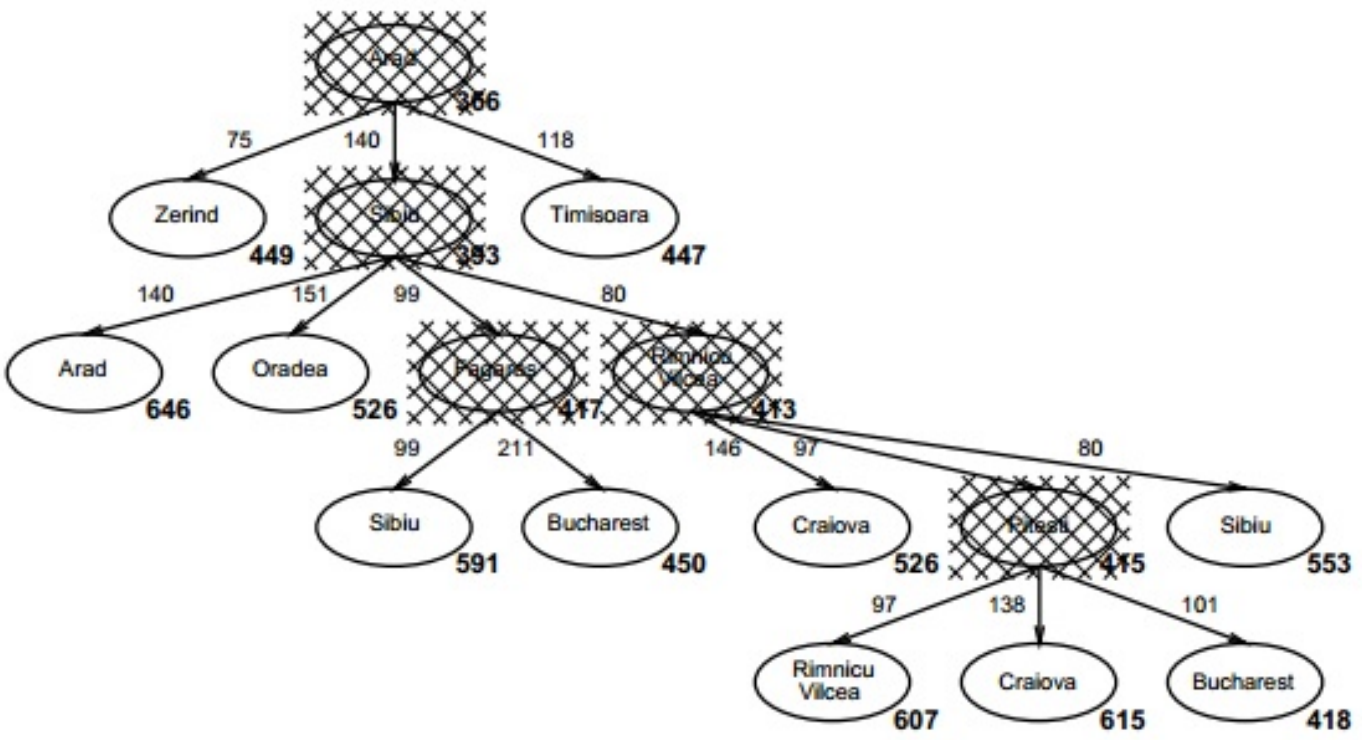
L'algorithme A^*

Exemple : A^*



L'algorithme A^*

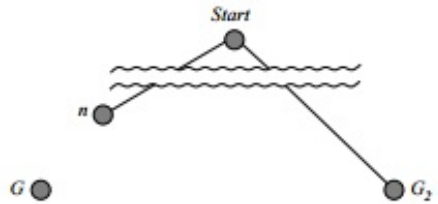
Exemple : A^*



L'algorithme A^*

Preuve que A^* est optimal

- Supposons qu'il y a un état final non optimal G_2 généré dans la liste des noeuds à traiter
- Soit n un noeud non développé sur le chemin le plus court vers un état final optimal G



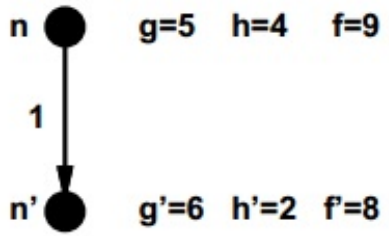
$$
\begin{aligned}
f(G_2) &= g(G_2) && \text{car } h(G_2) = 0 \\
&> g(G) && \text{car } G_2 \text{ n'est pas optimale} \\
&\geq f(n) && \text{car } h \text{ est admissible}
\end{aligned}
$$

$f(G_2) > f(n)$, donc A* ne va pas choisir G_2.

L'algorithme A^*

Quoi faire si f décroit ?

- Pour une heuristique admissible on peut avoir, que f décroît le long d'un chemin
- Par exemple, supposons que n' est successeur de n



- On perd de l'information
  - $f(n)$=9 : le vrai coût d'un chemin à travers n est >= 9
  - Donc le vrai coût d'un chemin à travers n' est aussi >=9

L'algorithme A^*

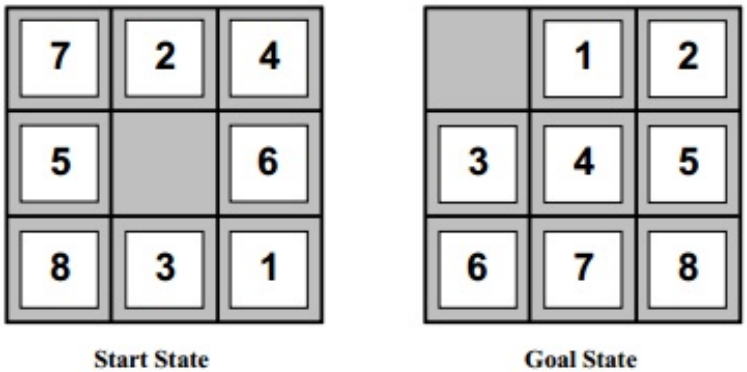
Modification de A^*

- Modification Pathmax pour A^*
- Au lieu de $f(n') = g(n') + h(n')$, on utilise $f(n') = max(g(n') + h(n'), f(n))$
- Avec pathmax, f ne décroit jamais le long d'un chemin

L'algorithme A^*

Exemples d'heuristiques admissibles : 8-puzzle

- $h_1(n)$= le nombre de pièces mal placées
- $h_2(n)$= la distance de Manhattan globale (la distance de chaque pièce en nombre de places de sa position finale)



- $h_1(S)$=8, $h_2(S)$=3+1+2+2+2+3+3+2=18

L'algorithme A^*

Comment trouver des heuristiques ?

- Des heuristiques admissibles peuvent être obtenues en considérant le coût exact d'une version simplifiée du problème
- Si les règles du 8-puzzle sont simplifiées de sorte qu'une pièce peut être déplacée partout, alors $h_1(n)$ donne la plus petite solution
- Si les règles du 8-puzzle sont simplifiées de sorte qu'une pièce peut être déplacée vers chaque place adjacente, alors $h_2(n)$ donne la plus petite solution

Exercice

Nous considérons un monde avec 4 pions (A,B,C,D) non superposables. Ils peuvent être arrangés dans n'importe quel ordre, sauf A qui ne peut pas être plus à droite que D. Par exemple, ABCD et CBAD sont deux états possibles du monde, tandis que DCBA et CDAB ne sont pas possibles. Le monde peut être manipulé par une action de la forme *echange(x,y)* qui échange les pions des positions x et y. Par exemple *echange(1,2)* transforme BCAD dans CBAD. Seules les actions *echange(1,2)*,*echange(2,3)* et *echange(2,4)* sont autorisées. Ils donnent un successeur uniquement si la situation atteinte est possible.

- Dessinez le graphe d'état.

Exercice

On suppose que l'état de départ est ADBC et l'état que l'on veut atteindre est CBAD. On suppose que chaque action coûte 1.

- Appliquer la recherche gloutonne. Ne considérez pas les noeuds déjà développés. En cas d'égalité choisissez un noeud à développer au hasard
- Appliquer la recherche A^* avec les mêmes suppositions.

Le voyageur de commerce

Le problème

Trouver le trajet de longueur minimale passant par toutes les villes et revenant au point de départ (distance euclidienne)

Le voyageur de commerce

Recherche exhaustive Il suffit de construire tous les chemins possibles et de calculer leurs longueur. Avec n villes il y a $(n-1)!/2$ chemins possibles. Avec 36 villes, on trouve : 516657398319307246483332566876160000000
Cette méthode est donc inutilisable, sauf si le nombre de villes est très petit

Branch and bound

En français : Algorithmes d'énumération par séparation et évaluation. voyons un premier exemple (problème de set partitionning) :

$$\min Z = 3X_1 + 7X_2 + 5X_3 + 8X_4 + 10X_5 + 4X_6 + 6X_7 + 9X_8$$

$$\begin{aligned} X_1 + X_2 &= 1 \\ X_3 + X_4 + X_5 &= 1 \\ X_5 + X_6 + X_7 &= 1 \\ X_7 + X_8 &= 1 \\ X_2 + X_4 + X_6 &= 1 \\ X_i &= 0 ou 1 \end{aligned}$$

On peut explorer l'arbre complet (256 noeuds)

Branch and bound

Évaluation

$z = 3X_1 + 7X_2 + 5X_3 + 8X_4 + 10X_5 + 4X_6 + 6X_7 + 9X_8$

$z = 3(X_1 + X_2) + 5(X_3 + X_4 + X_5) + 4(X_5 + X_6 + X_7) + ...$

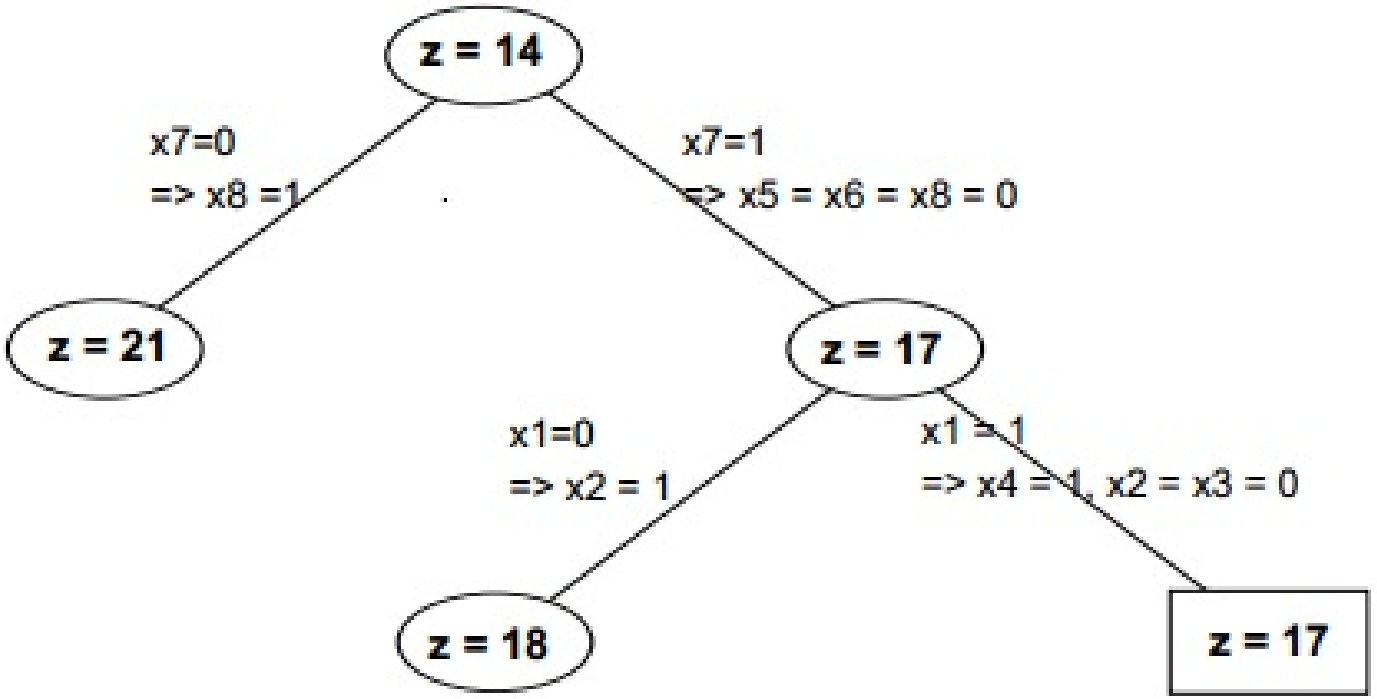
$z = 14 + 4X_2 + 3X_4 + X_5 + 7X_8$

On déduit donc que $min(z) >= 14$

Séparation : choix d'une variable

Variables de coût nul	Pénalité associée
$X_1 = 0$	4 car $X_2 = 1$
$X_3 = 0$	min(3,1)=1 car $X_4 = 1$ ou $X_5 = 1$
$X_6 = 0$	min(4,3)=3 car $X_2 = 1$ ou $X_4 = 1$
$X_7 = 0$	7 car $X_8 = 1$

Branch and bound



Branch and bound

On veut minimiser $z = f(x), x \varepsilon \omega$ ensemble des solutions réalisables (qui satisfont les contraintes). Il faut donc définir

- Une procédure de séparation : qui va choisir la prochaine variable à énumérer (pénalité, taille du domaine,...)
- Définir un minorant de z sur $S \subset \omega$ sous-ensemble de solution (on utilise le simplex ou des relations lagrangiennes)

L'objectif est de faire la plus grande coupure possible dans l'arbre de recherche. Cette méthode est donc une méthode exacte.

Exercice

Vous avez quatre objets à vendre (A, B, C, D) et quatre acheteurs $(1, 2, 3, 4)$. Chaque acheteur est prêt à payer un certain prix pour chaque objet comme indiqué dans le tableau ci-dessous (par exemple, l'acheteur 1 est prêt à payer 11 $ pour l'objet A). Utiliser la méthode branche and bound pour trouver à quel acheteur vous aller vendre chaque objet afin de maximiser votre profit sachant qu'un acheteur achète un objet et un seul. Vous devez expliciter la partie du graphe qui sera explorée en plus de fournir la solution. En plus indiquer clairement et complètement l'ordre de parcours du graphe au cours de l'exécution de l'algorithme.

Exercice

	1	2	3	4
A	11	11	8	15
B	14	15	10	5
C	3	3	4	15
D	10	10	20	10