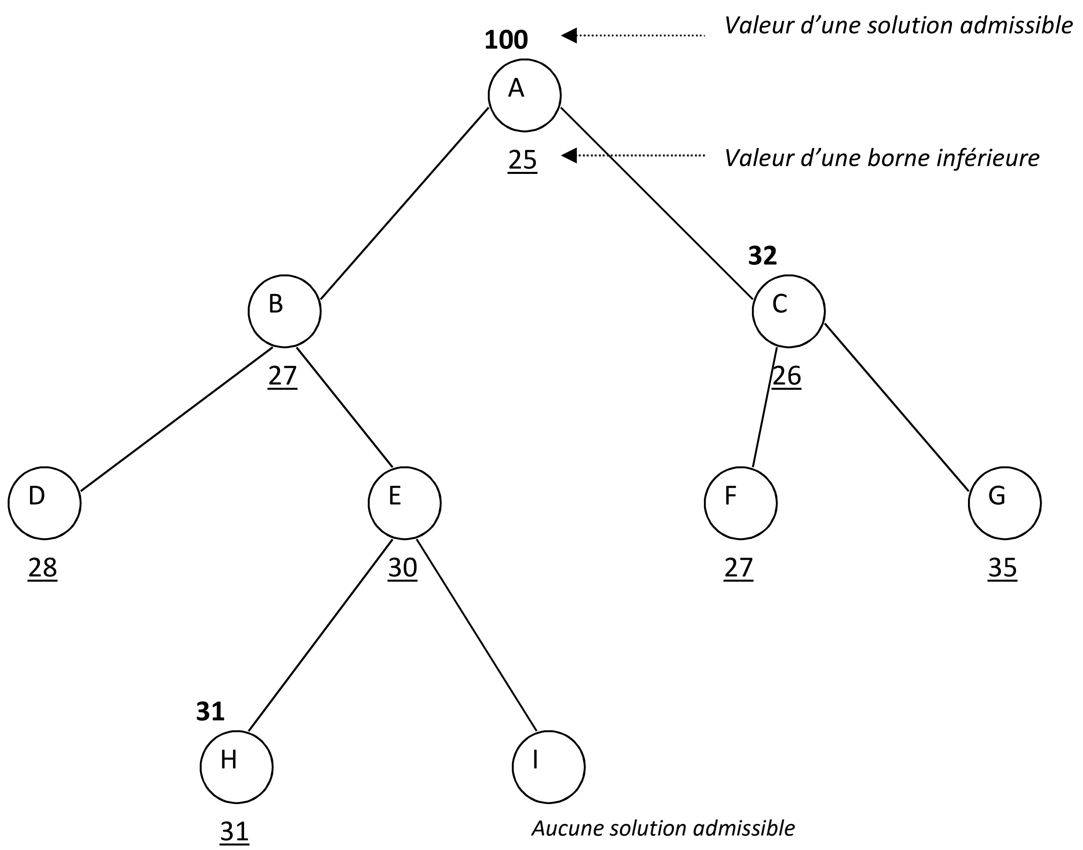
Fiche d'exercices

PLNE

Exercice 1 – Algorithme de Branch & Bound

Soit un problème de **minimisation** pour lequel on a commencé l'arborescence de recherche d'une solution optimale suivante, où les sommets sont arbitrairement notés A, B, …, I :



1) Quelle est la valeur de la meilleure solution admissible connue du problème, pour l'instant ?
2) Quelle est la valeur de la meilleure borne inférieure de la valeur optimale du problème, pour l'instant ?
3) Déduire des deux premières question un encadrement de la valeur optimale du problème de départ.
4) Pour chacune des feuilles, dire si elle peut être élaguée et pourquoi.

Exercice 2 – PLNE : Méthode de Dakin

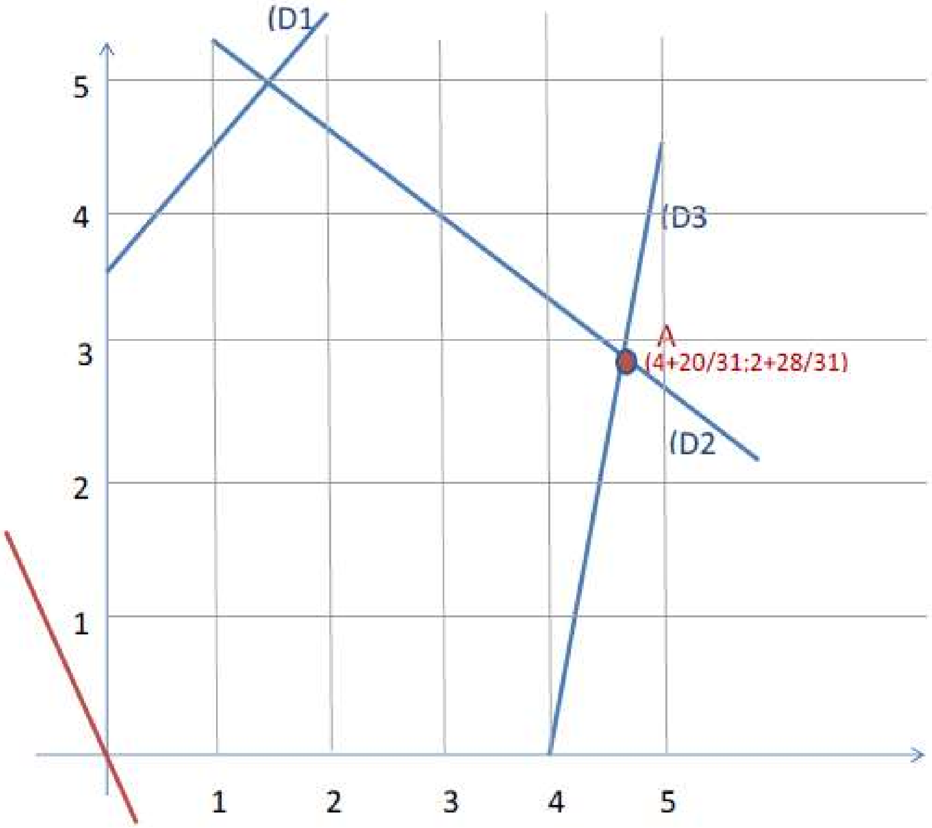
Cette méthode arborescente "meilleur d'abord" est utilisée pour résoudre des programmes en nombres entiers. On résout la relaxation continue du problème. La séparation se fait sur la variable qui a la plus grande partie fractionnaire dans la solution en cours : si $x = a + \frac{b}{c}$ (avec $b < c$) on sépare en $x \leq a$ et $x \geq a + 1$. Développer l'arborescence pour l'exemple suivant :

$$\begin{cases} \text{Max } z = 3x + 2y \\ s.c. \quad \left| \begin{array}{l} -2x + 2y \leq \ \ 7 \\ \ \ 2x + 3y \leq 18 \\ \ \ 9x - 2y \leq 36 \\ x \text{ et } y \geq 0 \text{ et entiers} \end{array} \right. \end{cases}$$

Les solutions des programmes linéaires continus à résoudre en chaque nœud seront trouvées directement de façon géométrique sur la figure donnée ci-dessous. La solution continue du problème à la racine est obtenue au point A.

Indication pour la suite : coordonnées des points à considérer successivement (mais pas obligatoirement dans cet ordre, et la liste n'est pas exhaustive) :

B : (4 ; 3+1/3) z=18+2/3 ; **C** : (4+4/9 ; 2) z=17+1/3 ; **D** : (4+1/2 ; 3) z=19+1/2 ; **E** : (3 ; 4) z=17



Exercice 3 – Sudoku

On désire résoudre, en utilisant la PLNE la grille de Sudoku suivant. Proposez une modélisation de ce problème et testez la en utilisant GLPK.

5				6		9		
		3					7	
	7						1	4
		8		5				
			2					
2						6		
			3			5		
	1		7					

Exercice 4 – Le problème du sac-à-dos

Pour embarquer dans l'avion, votre valise ne doit pas peser plus de B Kg. Il ne va pas être possible d'emporter les n objets que vous vouliez y placer et vous devez faire des choix. Vous avez pesé chaque objet et *pi* est le poids de l'objet i, $i = 1, \dots, n$. Pour optimiser le contenu de la valise, vous attribuez à chaque objet une note d'"utilité" entre 1 et 20 : c_i est la note attribuée à l'objet $i, i = 1, \dots, n$. Vous voulez maximiser l'utilité globale de la valise, égale à la somme des utilités des objets emportés, à l'aide de la recherche opérationnelle.

On définit les variables x_i , $i = 1, \dots, n$: si $x_i = 1$ si l'objet i est mis dans la valise, $x_i = 0$ sinon. Écrire le programme mathématique (P) à résoudre.

1. Écrire le programme si le poids maximum autorisé est *B* =17 kg et vous avez 4 objets de poids respectifs 3, 7, 9 et 6 kg et d'utilités respectives 8, 18, 20, 11.

On suppose que les objets sont fractionnables : le programme (P_R) est le programme (P) dans lequel on a remplacé $x_i \in \{0,1\}$ par $x_i \in [0, 1]$, $i = 1, \dots, n$. Le programme (P_R) (relaxation continue de (P)) se résout de la façon suivante :

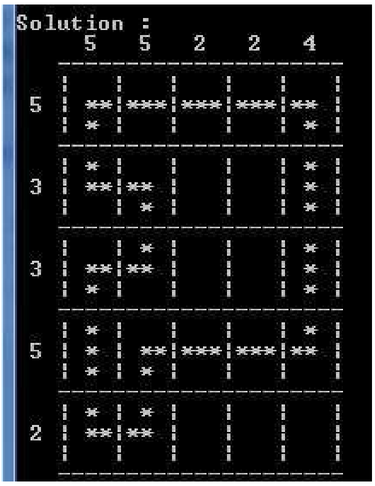
- On trie les ratios ${}^{c_i}/_{p_i}$ par ordre décroissant. Ensuite selon cet ordre, on met les objets dans le sac jusqu'à ce qu'on dépasse *B*. Le premier objet qui dépasse *B* est mis fractionnaire de façon à compléter le poids de la valise (jusqu'à B).
- Soit $imax$ tel que $\sum_{i=1}^{imax} p_i \leq B$ et $\sum_{i=1}^{ima} \quad p_i > B$. On a alors :
- $x_i = 1$ pour $i = 1, \dots, imax$; $x_{imax+1} = \frac{B - \sum_{i=1}^{imax} p_i}{p_{imax+1}}$; $x_i = 1$ pour $i = imax + 2, \dots, n$.

2. Quelle est la solution dans le cas de l'instance de la question (2.) si on suppose que les objets sont fractionnables ?
3. Résoudre le problème (P) pour l'exemple du (2.) par une méthode arborescente. En chaque nœud on obtient un majorant en résolvant la relaxation continue du problème en ce nœud. *Indications* : la séparation (en 2 sous-arbres) se fait sur la variable x_i non entière qui a la plus grande utilité : $x_i = 1$ ou $x_i = 0$. On déduit éventuellement des implications consécutives à chaque choix (c'est-à-dire qu'on fixe à 0 les variables associées à des objets qui ne pourront plus entrer dans la valise une fois que ce choix est fait).

<u>Exercice 5 – Le tour de la question</u> (Problème proposé par S. Elloumi)

Dans un numéro hors-série, les magazines La Recherche et Tangente proposent le problème suivant (p. 2), que nous appellerons problème du serpent. L'exemple proposé comporte une erreur (la trouver).

Le but étant de résoudre les 5 grilles proposées, vous pourrez essayer une résolution manuelle puis une modélisation par la PLNE, que vous mettrez en pratique à l'aide du solveur GLPK. Une visualisation de la solution sera proposée, sur le modèle suivant (grille de l'exemple) :



JEUX DE GRILLES

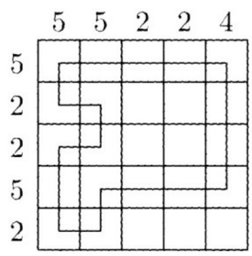
Tour de ville

RÈGLE DU JEU :
Retrouvez un circuit d'un seul tenant passant par le centre de certaines cases du plan. Le circuit est composé de segments horizontaux ou verticaux joignant les centres de deux cases voisines. Les nombres extérieurs à la grille indiquent le nombre de cases visitées (lors du tour) dans la ligne ou la colonne de l'indice.

Exemple

	5	5	2	2	4
5					
2					
2					
5					
2					

Solution



★ jeu 1

	3	4	3	3	5
4					
5					
3					
3					
3					

★★ jeu 2

	3	2	4	5	4
3					
5					
2					
5					
3					

★★ jeu 3

	4	4	3	3	4
3					
4					
5					
4					
2					

★★ jeu 4

	4	4	2	3	2	5
3						
4						
3						
2						
6						
2						

★★★ jeu 5

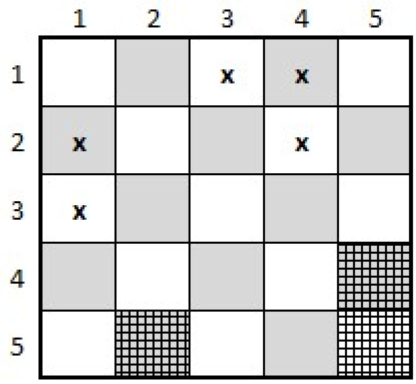
	4	5	4	5	5	3
4						
5						
5						
2						
5						
5						

Référence : « 5 jeux de circuits » Bernard Novelli et Martin Rivière Éditions Pôle, collection « À vous de jouer ». Disponible en librairie ou sur www.[illegible]

Solutions en page 98

Exercice 6 – Le problème des reines non dominantes

On souhaite placer sur un échiquier de taille *n*×*n* (pas nécessairement 8×8 comme les véritables échiquiers) exactement *n* reines, de sorte à maximiser le nombre de cases laissées libres. Une case n'est pas libre si elle est située sur la même ligne, colonne ou diagonale qu'au moins une des reines que l'on a placées. On notera *L*(*n*) le nombre maximal de cases libres. Le problème est relativement simple à résoudre « à la main » pour les petites valeurs de *n*. Ainsi, pour *n*=5, il est possible de laisser au maximum *L*(5)=3 cases libres sur l'échiquier (il existe plusieurs façons de disposer les 5 reines) :



Exemple de problème résolu pour n=5 : L(5) = 3.

Les croix représentent les reines, les cases libres sont hachurées.

Le but de cet exercice est d'essayer de résoudre ce problème pour des échiquiers de plus en plus grands. Trouvez une modélisation du problème sous la forme d'un PLNE (ou d'un PL-01). Puis vous pourrez essayer de mettre en œuvre la résolution exacte du problème pour différentes valeurs de *n*, en reportant à chaque fois le résultat obtenu pour *L*(*n*).

Exercice 7 – Programmation quadratique 0-1

On veut montrer que tout programme quadratique en variables 0-1 peut se ramener à un programme linéaire en 0-1. Pour cela, on va "linéariser" le programme en remplaçant chaque terme quadratique $x_i x_j$ par une seule variable 0-1 : z_{ij}.

1. Donner les contraintes à ajouter pour que le programme linéaire obtenu soit équivalent au programme quadratique initial, c'est-à-dire qu'on a bien $z_{ij} = x_i x_j$ pour tout *i* tout *j*.
2. Une telle transformation est-elle possible si les variables sont continues (dans $\mathbb{R}$) ?

Exercice 8 – Le problème du voyageur de commerce (TSP)

Définition du problème : un voyageur du commerce parisien doit programmer une tournée dans plusieurs villes. Il doit partir de Paris et retourner à Paris et visiter chaque ville exactement une fois. Il faut aussi que la distance parcourue au total soit la plus petite possible.

Voici les villes à visiter et les distances entre les différentes villes :

	Bordeaux	Lille	Lyon	Marseille	Paris	Rennes	Roubaix	Toulouse	Dunkerque
Bordeaux	0	796	482	640	583	449	801	244	876
Lille	796	0	685	995	212	562	11	900	79
Lyon	482	685	0	309	463	761	701	538	763
Marseille	640	995	309	0	766	921	1011	405	1074
Paris	583	212	463	766	0	338	229	676	292
Rennes	449	562	761	921	338	0	579	699	569
Roubaix	801	11	701	1011	229	579	0	908	91
Toulouse	244	900	538	405	676	699	908	0	969
Dunkerque	876	79	763	1074	292	569	91	969	0

Dans la littérature de la théorie des graphes, ce problème est celui de la recherche d'un cycle hamiltonien de longueur minimale. Le graphe non orienté considéré est tel que les sommets sont les villes et l'arête d'une ville i à une ville j est valuée par la distance de i à j.

1- Modéliser ce problème par un programme mathématique.
2- Implémenter le modèle obtenu en utilisant GLPK (les données sont fournies dans le fichier tsp.dat).
3- Vérifier ou corriger le modèle au vu des résultats fournis par GLPK.