CHAPTER 1

Programmation linéaire

1.1. Qu'est-ce que la programmation linéaire

1.1.1. Exemple : le problème du régime de Polly [1, p.3].

– Besoins journaliers :

Énergie: 2000 kcal

Protéines: 55g

Calcium: 800 mg

– Nourriture disponible

	Portion	Énergie (kcal)	Protéines (g)	Calcium (mg)	Prix/portion
Céréales	28g	110	4	2	3
Poulet	100g	205	32	12	24
Oeufs	2 gros	160	13	54	13
Lait entier	237cc	160	8	285	9
Tarte	170g	420	4	22	20
Porc et haricots	260g	260	14	80	19

Quels choix pour Polly ?

– Contraintes :

Céréales: au plus 4 portions par jour

Poulet: au plus 3 portions par jour

Oeufs: au plus 2 portions par jour

Lait: au plus 8 portions par jour

Tarte: au plus 2 portions par jour

Porc et haricots: au plus 2 portions par jour

PROBLEM 1.1.1. Polly peut-elle trouver une solution ?

Comment formaliser le problème ? (modélisation)

Qu'est-ce qui fait la spécificité du problème ?

Savez-vous résoudre des problèmes similaires ?

1.1.2. Forme standard d'un problème de programmation linéaire.

PROBLÈME. [**1**, p. 5]

DÉFINITION. Problème de programmation linéaire sous *forme standard :*
Maximiser :

$$z := \sum_{j=1}^{n} c_j x_j$$

Sous les contraintes :

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \text{ pour } i = 1, \ldots, m$$

$$x_j \geq 0, \text{ pour } j = 1, \ldots, n$$

Un choix des variables $(x_1, \ldots, x_n)$ est appelé *solution* du problème.
Une solution est *faisable* si elle vérifie les contraintes.
z est appelé *fonction objective*. À chaque solution elle associe une valeur.
Une solution est *optimale* si elle est faisable et maximize la fonction objective.

EXERCICE 1. Peut-on mettre sous forme standard les exemples précédents ?

1.1.3. Existence de solutions optimales ?

PROBLEM 1.1.2. [**1**, p. 7]
On considère les trois problèmes de programmation linéaire standard suivants, écrits avec la syntaxe du système de calcul formel `MuPAD` :

```
Chvatal7a := [ [   x1      <= 3,
                         x2 <= 7 ],
                3 +x1  +x2,
               NonNegative] :
Chvatal7b := [ [   x1  +x2 <= 2,
                -2*x1-2*x2 <= -10 ],
                 3*x1  -x2,
               NonNegative] :
Chvatal7c := [ [-2*x1  +x2 <= -1,
                   -x1-2*x2 <= -2 ],
                    x1  -x2,
               NonNegative] :
extra     := [ [   x1  +x2 <= 1 ],
                    x1  +x2,
               NonNegative] :
```

PROBLEM 1.1.3. Déterminer pour ces trois problèmes s'il y a des solutions optimales.

– Premier cas : une solution optimale unique
– Deuxième cas : pas de solution faisable
– Troisième cas : pas de solution optimale, car on peut faire tendre la fonction objective vers l'infini avec des solutions faisables.
– Quatrième cas : une infinité de solutions optimales.

1.2. Algorithme du simplexe

1.2.1. Premier problème.

PROBLEM 1.2.1. [**1**, p. 13]

```
Chvatal13 := [{2*x1 + 3*x2 +   x3 <= 5,
               4*x1 +   x2 + 2*x3 <= 11,
               3*x1 + 4*x2 + 2*x3 <= 8  },
               5*x1 + 4*x2 + 3*x3,
              NonNegative] :
```

Solution faisable ?
Amélioration de la solution ?

Introduction de variables d'écart :

```
5  = s1       + 2*x1 + 3*x2 +   x3
11 =    s2    + 4*x1 +   x2 + 2*x3
8  =       s3 + 3*x1 + 4*x2 + 2*x3
---------------------------------
z  =            5*x1 + 4*x2 + 3*x3
```

En augmentant x1 jusqu'à 5/2, on fait tomber s1 à zéro.
On transforme le système, pour se ramener à une situation similaire à la précédente :

```
5/2 = x1       + 3/2*x2 + 1/2*x3 + 1/2*s1
1   =    s2    -   5*x2            - 2*s1
1/2 =       s3 - 1/2*x2 + 1/2*x3 - 3/2*s1
---------------------------------------
z   = 25/2     - 7/2 x2 + 1/2*x3 - 5/2*s1
```

On augmente x3 jusqu'à 1, ce qui fait tomber s3 à 0 :

```
1 = x3       -   x2 - 3*s1 + 2*s3
2 =    x1    + 2*x2 + 2*s1 -   s3
1 =       s2 - 5*x2 - 2*s1
-------------------------------
z = 13       - 3*x2 -   s1 -   s3
```

Et maintenant, que fait-on ?

1.2.2. Variables d'écart.

PROBLEM 1.2.2. Est-ce que l'introduction de ces variables change le problème ?

1.2.3. Tableaux.

PROBLEM 1.2.3. [**1**, p. 19]

```
Chvatal19 := [[  x1 + 3*x2   + x3 <= 3,
                -x1         +3*x3 <= 2,
               2*x1 + 3*x2   - x3 <= 2,
               2*x1   - x2 + 2*x3 <= 4],
               5*x1 + 5*x2 + 3*x3,
              NonNegative] :
```

DÉFINITION. *Tableau initial :*

$$b_i = s_i + \sum_{j=1}^{n} a_{ij} x_j, \text{ pour } i = 1, \ldots, m$$

$$z = \sum_{j=1}^{n} c_j x_j$$

EXEMPLE. Tableau initial du problème précédent :

```
3 = s1         + x1 + 3 x2   + x3
2 =   s2       - x1        + 3 x3
2 =     s3   + 2 x1 + 3 x2   - x3
4 =       s4 + 2 x1   - x2 + 2 x3
--------------------------------
z = 0        + 5 x1 + 5 x2 + 3 x3
```

EXEMPLE 1.2.4. On peut l'abréger sous forme matricielle :

```
read("tableaux.mu") :
linopt : :Transparent(Chvatal19);
+-                                                    -+
|"linopt","restr",slk[1],slk[2],slk[3],slk[4],x3,x1,x2|
|                                                      |
|  "obj",    0,      0,     0,     0,     0,   3, 5, 5|
|                                                      |
| slk[1],    3,      1,     0,     0,     0,   1, 1, 3|
|                                                      |
| slk[2],    2,      0,     1,     0,     0,   3,-1, 0|
|                                                      |
| slk[3],    2,      0,     0,     1,     0,  -1, 2, 3|
|                                                      |
| slk[4],    4,      0,     0,     0,     1,   2, 2,-1|
+-                                                    -+
```

DÉFINITION. De manière générale, un *tableau* est un ensemble d'équations de la forme :

```
1 = x1       + 3/2 x2  - 1/2 x3 + 1/2 s4
2 =   s1     + 3/2 x2  + 3/2 x3 - 1/2 s4
3 =     s2   + 3/2 x2  + 5/2 x3 + 1/2 s4
2 =       s3   - 4 x2    + 3 x3     - s4
---------------------------------------
z = 5        - 5/2 x2 + 11/2 x3 - 5/2 s4
```

x_1, s_1, s_2, s_3 sont les variables *basiques ;* $\{x_1, s_1, s_2, s_3\}$ est la *base.*

x_2, x_3, s_4 sont les variables *non basiques.*

REMARQUE 1.2.5. Terminologie : on utilise dans ce cours les tableaux, plutôt que les *dictionnaires* utilisés par exemple dans [**1**]. La différence est minime : on fait juste passer les variables non basiques d'un côté ou de l'autre des équations. D'autre part, on utilise s_1, s_2, s_3, s_4 plutôt que x_4, x_5, x_6, x_7 comme noms pour les variables d'écarts.

Voici le dictionnaire correspondant au tableau précédent :

```
x1 = 1 - 3/2 x2  + 1/2 x3 - 1/2 x7
x4 = 2 - 3/2 x2  - 3/2 x3 + 1/2 x7
x5 = 3 - 3/2 x2  - 5/2 x3 - 1/2 x7
x6 = 2   + 4 x2    - 3 x3     + x7
--------------------------------
 z = 5 - 5/2 x2 + 11/2 x3 - 5/2 x7
```

REMARQUE 1.2.6. La caractéristique essentielle d'un tableau est que, connaissant les variables non-basiques, on peut immédiatement calculer les variables basiques et la fonction objective (d'où le terme de *dictionnaire*). Le calcul devient même immédiat si toutes les variables non-basiques sont nulles.

Les équations d'un tableau décrivent un sous-espace affine de $\mathbb{R}^{n+m}$.

Un point p de cet espace est caractérisé par ses coordonnées dans les variables non-basiques.

EXERCICE 2. Calculer directement le tableau correspondant aux variables non-basiques x_1, s_2, s_3 du programme linéaire `Chvatal13`.

EXERCICE 3. Soit t_1 et t_2 deux tableaux correspondant au même programme linéaire.

Que peut-on dire des deux sous-espaces affine de $\mathbb{R}^{n+m}$ qu'ils définissent ?

DÉFINITION 1.2.7. Le point de coordonnées $(0, \ldots, 0)$ dans les variables non-basiques est appellé *solution basique* du tableau.

Un tableau est *faisable* si la solution basique $(0, \ldots, 0)$ est une solution faisable.

De manière équivalente, un tableau est faisable si les constantes dans les équations du haut sont toutes positives ou nulles.

Revenons à l'exemple [**1**, p. 19] :

```
read("tableaux.mu") :
t :=linopt : :Transparent(Chvatal19) ;
t :=linopt : :Transparent : :userstep(t, slk[3], x3) ;
```

EXERCICE 4. [**1**, 2.1 p. 26]

Utilisez l'algorithme du simplexe pour résoudre les programmes linéaires suivants :

```
Chvatal26_21a :=
[[  x1  +x2+2*x3 <= 4,
  2*x1     +3*x3 <= 5,
  2*x1  +x2+3*x3 <= 7],
  3*x1+2*x2+4*x3,
 NonNegative] :
Chvatal26_21c :=
[[2*x1+3*x2 <= 3,
    x1+5*x2 <= 1,
  2*x1  +x2 <= 4,
  4*x1  +x2 <= 5],
  2*x1  +x2,
 NonNegative] :
```

EXERCICE 5. Essayez d'appliquer l'algorithme du simplexe aux programmes linéaires de l'exercice [**1**, p. 7] (cf. ci-dessus). Que se passe-t'il ?

1.3. Pièges et comment les éviter

1.3.1. Bilan des épisodes précédents. On a un algorithme qui marche sur quelques exemples.
Il faut vérifier trois points pour savoir s'il marche en général :

(1) Initialisation
(2) Itération
(3) Terminaison

1.3.2. Itération.

PROPOSITION. *Étant donné un tableau faisable, on peut toujours effectuer l'une des opérations suivantes :*

(1) Conclure que le système a une solution optimale unique, la calculer et la certifier ;
(2) Conclure que le système a une infinité de solutions optimales, les calculer et les certifier ;
(3) Conclure que le système est non borné, et le certifier en décrivant une demi-droite de solutions sur laquelle z prend des valeurs aussi grandes que voulu.
(4) Trouver une variable entrante, une variable sortante, et effectuer un pivot. Par construction, le tableau obtenu est équivalent au tableau précédent, et est encore faisable. De plus, z a *augmenté au sens large* (i.e. la constante z^* dans la nouvelle expression de z est supérieure ou égale à l'ancienne).

PROOF. Il suffit d'analyser le tableau faisable. Notons $S_1, \dots, S_m$ les variables basiques, $X_1, \dots, X_n$ les variables non-basiques, et $C_1, \dots, C_n, z^*$ les coefficients tels que $z = z^* + \sum C_i X_i$. Par exemple, dans le tableau final du problème1.2.1, on a $X_1 = x_2$, $X_2 = s_1$, $X_3 = s_2$, $S_1 = x_1$, $S_2 = x_3$, $S_3 = s_3$, $C_1 = -3$, $C_2 = -1$, $C_3 = -1$ et $z^* = 13$.

(1) Si $C_i < 0$, pour tout i, alors la solution basique du tableau, de coordonnées $X_1^* = \dots = X_n^* = 0$ est l'unique solution optimale. Vérifiez le en prouvant qu'une toute solution faisable quelconque de coordonnées $X_1, \dots, X_n$ donnant la même valeur $z = z^*$ à la fonction objective est égale à la solution basique du tableau.
(2) Si $C_i \leq 0$ pour tout i, la solution basique du tableau est optimale, et l'ensemble des solutions optimales est décrit par les inéquations linéaires du système et l'annulation des variables non-basiques X_i pour lesquelles on a $C_i < 0$. Les détails sont similaires au 1.
(3) Sinon, on peut prendre X_i, variable non-basique avec un coefficient $C_i > 0$. Si les équations du tableau n'imposent pas de limite sur X_i, le système est non borné : la demi-droite décrite par $(0, \dots, 0, X_i, 0, \dots, 0)$ pour $X_i \geq 0$ est composée de solutions faisables qui donnent des valeurs aussi grandes que voulu à z.
(4) Autrement, une des variables basiques S_j tombe à zéro, et on peut faire un pivot entre la variable entrante X_i et la variable sortante S_j. Par construction, la nouvelle solution basique correspond à une solution faisable $(0, \dots, 0, X_i, 0, \dots, 0)$ pour un $X_i \geq 0$. En particulier le nouveau tableau est faisable, et comme $C_i \geq 0$, la constante z^* a augmenté au sens large.

□

EXEMPLE. [**1**, p. 29] Système où z n'augmente pas strictement lors du pivot :

```
Chvatal29 :=  [[              2*x3 <= 1,
                - x1 + 3*x2 + 4*x3 <= 2,
                2*x1 - 4*x2 + 6*x3 <= 3],
                2*x1   - x2 + 8*x3,
                NonNegative] :
t0 := linopt : :Transparent(Chvatal29) ;
```

```
t1 := linopt : :Transparent : :userstep(t0, slk[1], x3);
t2 := linopt : :Transparent : :userstep(t1, slk[3], x1);
t3 := linopt : :Transparent : :userstep(t2, slk[2], x2);
t4 := linopt : :Transparent : :userstep(t3, x3, slk[1]);
```

REMARQUE. Lorsque z n'augmente pas, on est forcément dans une situation de dégénérescence : le pivot change le tableau, mais pas la solution basique décrite par le tableau.

1.3.3. Terminaison.

PROBLEM 1.3.1. Peut-on garantir que l'algorithme va finir par s'arrêter ?

THÉORÈME. *Si l'algorithme du simplexe ne cycle pas, il termine en au plus $C(n+m, m)$ itérations.*

PROOF. (Résumé)

Chaque itération correspond à un tableau faisable.

Un tableau faisable est entièrement caractérisé par le choix des variables basiques.

Il n'y a que $C(n+m, m)$ choix possibles de variables basiques. □

REMARQUE. L'algorithme ne peut cycler qu'en présence de dégénérescence.

Avec une stratégie incorrecte, l'algorithme du simplexe peut cycler éternellement :

EXEMPLE. [**1**, p. 31] Système cyclant en 6 itérations avec la stratégie :
- Choix de la variable entrante avec le coefficient dans l'expression de z le plus fort
- Choix de la variable sortante avec le plus petit index

```
Chvatal31 := [[0.5*x1 - 5.5*x2 - 2.5*x3 +  9*x4 <= 0,
               0.5*x1 - 1.5*x2 - 0.5*x3 +    x4 <= 0,
                   x1                           <= 1],
                10*x1  - 57*x2   - 9*x3 - 24*x4,
              NonNegative] :
t0 := linopt : :Transparent(Chvatal31);
t1 := linopt : :Transparent : :userstep(t0, slk[1], x1);
t2 := linopt : :Transparent : :userstep(t1, slk[2], x2);
t3 := linopt : :Transparent : :userstep(t2, x1, x3);
t4 := linopt : :Transparent : :userstep(t3, x2, x4);
t5 := linopt : :Transparent : :userstep(t4, x3, slk[1]);
t6 := linopt : :Transparent : :userstep(t5, x4, slk[2]);
```

Comment garantir que l'algorithme ne cyclera pas ?

1.3.3.1. *La méthode des perturbations.* L'algorithme du simplexe ne peut cycler qu'en présence de dégénérescence.

PROBLEM 1.3.2. Comment se débarasser des dégénérescences ?

Idée : supprimer les dégénérescences en perturbant légèrement le système !

EXEMPLE. [**1**, p. 34,35]

On introduit des constantes $\varepsilon_1 >> \cdots >> \varepsilon_n$.
Inconvénient : solution approchée, ou introduction de calcul symbolique

1.3.3.2. *La méthode du plus petit index.*

THÉORÈME. *L'algorithme du simplexe termine si, lorsqu'il y a ambiguïté sur le choix de la variable entrante ou sortante, on choisit toujours la variable de plus petit index.*

Cette méthode est simple et élégante.
Par contre, elle empêche toute stratégie pour faire converger l'algorithme plus vite.

1.3.3.3. *Méthodes intermédiaires.* Stratégie au choix, mais si z n'augmente pas pendant plus d'un certain nombre d'itérations, on bascule sur la stratégie du plus petit index jusqu'à ce que l'on soit sorti de la dégénérescence.

1.3.4. Initialisation. Pour le moment, l'algorithme du simplexe nécessite de partir d'un tableau faisable.

PROBLÈME. Dans le cas général, comment se ramener à un tableau faisable ?
Le système pourrait même ne pas avoir de solution !

EXEMPLE. [**1**, p. 39] Système P_1 :
Maximiser : $x_1 - x_2 + x_3$
Sous les contraintes :
$2x_1 - x_2 + 2x_3 \leq 4$
$2x_1 - 3x_2 + x_3 \leq -5$
$-x_1 + x_2 - 2x_3 \leq -1$
$x_1, x_2, x_3 \geq 0$

EXEMPLE. Introduction d'un *système* auxiliaire P_0 pour déterminer si P est faisable :
Maximiser : $-x_0$
Sous les contraintes :
$2x_1 - x_2 + 2x_3 - x_0 \leq 4$

$2x_1 - 3x_2 + x_3 - x_0 \leq -5$
$-x_1 + x_2 - 2x_3 - x_0 \leq -1$
$x_0, x_1, x_2, x_3 \geq 0$
Remarques :
– *P_0 est faisable (prendre x_0 suffisamment grand) ;*
– *Les solutions faisables de P correspondent aux solutions faisables de P_0 avec $x_0 = 0$;*
– *P est faisable si et seulement si P_0 a une solution faisable avec $x_0 = 0$.*
Étudions ce nouveau système :

```
Chvatal40 := [[ -x1   + x2 - 2*x3 - x0 <= -1,
                 2*x1 - 3*x2   + x3 - x0 <= -5,
                 2*x1   - x2 + 2*x3 - x0 <= 4],
                -x0,
                NonNegative] :
t0 :=linopt : :Transparent(Chvatal40) ;
t1 :=linopt : :Transparent : :userstep(t0, slk[2], x0) ;
t2 :=linopt : :Transparent : :userstep(t1, slk[1], x2) ;
t3 :=linopt : :Transparent : :userstep(t2, x0, x3) ;
```

Maintenant, nous savons que le système P est faisable.
En fait, en éliminant x_0 on obtient même un tableau faisable pour P !

Algorithme du simplexe en deux phases pour résoudre un problème P sous forme standard :
Phase I :

(1) Si $(0, \ldots, 0)$ est solution faisable de P, on passe directement à la phase II.
(2) Définir un problème auxiliaire P_0.
(3) Le premier tableau pour P_0 est infaisable.
(4) Le rendre faisable par un pivot approprié de x_0.
(5) Appliquer le simplexe habituel :
    (a) Si à une étape donnée, x_0 peut sortir de la base, le faire en priorité :
        En effet, il y a une solution faisable avec $x_0 = 0$, et on peut passer en phase II.
    (b) Si à une étape donnée on atteint une solution optimale :
        (i) Si x_0 n'est pas basique :
            Il y a une solution faisable avec $x_0 = 0$. On peut donc passer en phase II.
        (ii) Si x_0 est basique et $z_0 < 0$:
            P est infaisable, et on s'arrête.
        (iii) Sinon x_0 est basique et $z_0 = 0$:
            Situation impossible si on fait toujours sortir x_0 en priorité de la base.
(6) Tirer de P_0 un tableau faisable pour P ;

Phase II :

(1) Appliquer le simplexe habituel à partir du tableau donné par P_0.

EXERCICE. [**1**, ex 3.9a p. 44]
Maximiser $3x_1 + x_2$
Sous les contraintes :
$x_1 - x_2 \leq -1$
$-x_1 - x_2 \leq -3$
$2x_1 + x_2 \leq 4$
$x_1, x_2 \geq 0$

```
t0 :=linopt : :Transparent(Chvatal44_39a0)
t1 :=linopt : :Transparent : :userstep(t0, slk[2], x0)
t2 :=linopt : :Transparent : :userstep(t1, slk[1], x1)
t3 :=linopt : :Transparent : :userstep(t2, x0, x2)
t0 :=linopt : :Transparent(Chvatal44_39a)
t1 :=linopt : :Transparent : :userstep(t0, slk[1], x1)
t2 :=linopt : :Transparent : :userstep(t1, slk[2], x2)
t3 :=linopt : :Transparent : :userstep(t2, slk[3], slk[2])
```

1.3.5. Le théorème fondamental de la programmation linéaire.

THÉORÈME. *Tout programme linéaire P sous forme standard a l'une des propriétés suivantes :*

(1) Si P n'a pas de solutions optimales, alors P est infaisable ou non borné ;
(2) Si P a une solutions faisable, alors P a une solution basique faisable ;
(3) Si P a une solution optimale, alors P a une solution basique optimale.

1.4. Efficacité de l'algorithme du simplexe

Pour une discussion complète sur ce thème, nous renvoyons au livre de référence [**1**, 4. How fast is the simplex method], ainsi qu'à l'excellente Foire Aux Questions `http://rutcor.rutgers.edu/~mnk/lp-faq.html` pour les évolutions récentes.

EXERCICE. [**1**, 4.2 et 4.3, p. 53]

1.5. Le théorème de dualité

1.5.1. Motivation : estimer la valeur optimale de la fonction objective.

EXEMPLE. Maximiser : $z = 4x_1 + x_2 + 5x_3 + 3x_4$
Sous les contraintes :
$x_1 - x_2 - x_3 + 3x_4 \leq 1$
$5x_1 + x_2 + 3x_3 + 8x_4 \leq 55$
$-x_1 + 2x_2 + 3x_3 - 5x_4 \leq 3$
$x_1, x_2, x_3, x_4 \geq 0$

PROBLÈME. Borne inférieure sur la valeur optimale z^* ?
Borne supérieure sur la valeur optimale z^* ?

D'après la seconde contrainte :

$$z^* \leq 4x_1 + x_2 + 5x_3 + 3x_4 \leq \frac{25}{3}x_1 + \frac{5}{3}x_2 + 5x_3 + \frac{40}{3}x_4 \leq \frac{275}{3}$$

En utilisant la somme de la deuxième et troisième contrainte :

$$z^* \leq 4x_1 + 3x_2 + 6x_3 + 3x_4 \leq 58$$

PROBLÈME. Comment faire cela de manière systématique ?

On recherche des combinaisons linéaires des contraintes :
- y_1 fois la première contrainte : $x_1 - x_2 - x_3 + 3x_4 \leq 1$
- y_2 fois la seconde contrainte : $5x_1 + x_2 + 3x_3 + 8x_4 \leq 55$
- y_3 fois la troisième contrainte : $-x_1 + 2x_2 + 3x_3 - 5x_4 \leq 3$

Ce qui donne :

$$(y_1 + 5y_2 - y_3)x_1 + (-y_1 + y_2 + 2y_3)x_2 + (-y_1 + 3y_2 + 3y_3)x_3 + (3y_1 + 8y_2 - 5y_3)x_4$$

$$\leq y_1 + 55y_2 + 3y_3$$

Quelles sont les contraintes pour obtenir une borne sur z^* ?

Pour garder le sens des inégalités : $y_1, y_2, y_3 \geq 0$
Pour obtenir une majoration de $z = 4x_1 + x_2 + 5x_3 + 3x_4$:
$y_1 + 5y_2 - y_3 \geq 4$
$-y_1 + y_2 + 2y_3 \geq 1$
$-y_1 + 3y_2 + 3y_3 \geq 5$
$3y_1 + 8y_2 - 5y_3 \geq 3$
Si y_1, y_2, y_3 satisfont ces conditions, on obtient la borne $z \leq y_1 + 55y_2 + 3y_3$.
On veut donc minimiser $y_1 + 55y_2 + 3y_3$!
Par exemple, en prenant $y_1 = 0$ et $y_2 = y_3 = 1$, on retrouve l'inégalité $z \leq 58$.

1.5.2. Le problème dual.

DÉFINITION. Soit P un programme linéaire sous *forme standard :*
Maximiser :

$$z = \sum_{j=1}^{n} c_j \; x_j$$

Sous les contraintes :

$$\sum_{j=1}^{n} a_{ij} \; x_j \leq b_i, \text{ pour } i = 1, \ldots, m$$

$$x_j \geq 0, \text{ pour } j = 1, \ldots, n$$

Le *dual* de P est le problème :
Minimiser :

$$w = \sum_{i=1}^{m} b_i \ y_i$$

Sous les contraintes :

$$\sum_{i=1}^{m} a_{ij} \ y_i \geq c_j, \text{ pour } j = 1, \ldots, n$$

$$y_i \geq 0, \text{ pour } i = 1, \ldots, m$$

P est appelé problème *primal.*

PROPOSITION. *Si $x_1, \ldots, x_n$ est une solution faisable du problème primal et $y_1, \ldots, y_m$ une solution faisable du problème dual, alors $z \leq w$,* i.e.

$$\sum_{j=1}^{n} c_j \ x_j \leq \sum_{i=1}^{m} b_i \ y_i$$

DÉMONSTRATION. Il suffit d'appliquer les inégalités qui définissent les solutions faisables :

$$z = \sum_{j=1}^{n} c_j \ x_j \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} \ y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} \ x_j \right) y_i \leq \sum_{i=1}^{m} b_i \ y_i = w$$

□

En particulier :
- Si, comme dans l'exemple précédent, on connaît une solution faisable du problème dual, on obtient une borne sur le problème primal et réciproquement !
- Si on connaît une solution faisable du problème primal et une solution faisable du problème dual telles que $z = w$, *i.e.*

$$\sum_{j=1}^{n} c_j \ x_j = \sum_{i=1}^{m} b_i \ y_i,$$

alors on sait que ces deux solutions sont optimales !

EXERCICE. Prouver que les solutions faisables $x_1 = 0$, $x_2 = 14$, $x_3 = 0$, $x_4 = 5$ et $y_1 = 11$, $y_2 = 0$, $y_3 = 6$ du problème original et de son dual sont optimales.

La donnée de (y_1, y_2, y_3) donne un *certificat* de l'optimalité de la solution (x_1, x_2, x_3, x_4) :
Quelqu'un qui veut faire une vérification peut le faire quasiment sans calcul :
il suffit de tester que les solutions sont faisables et que que $z = w$.

PROBLÈME. Est-il toujours possible de trouver un tel certificat ?

La réponse est oui, et c'est le théorème central de la programmation linéaire.

1.5.3. Le théorème de dualité.

THÉORÈME. *Si le problème primal a une solution optimale* $(x_1^*, \ldots, x_n^*)$*, alors le problème dual a une solution optimale* $(y_1^*, \ldots, y_m^*)$ *telle que* $w^* = z^*$, i.e.

$$\sum_{j=1}^{n} c_j \; x_j^* = \sum_{i=1}^{m} b_i \; y_i^*.$$

Ce théorème nous assure de l'existence d'un certificat.

Mais y-a-t'il une technique pour le calculer ?

Oui, car la preuve va être *constructive* : son principe va précisément être de construire une solution optimale, en utilisant le tableau final obtenu par l'algorithme du simplexe.

EXEMPLE. Faisons un peu de magie sur notre exemple.

Le tableau initial est :

```
Chvatal54 :=
[[  x1   - x2   - x3 + 3*x4 <= 1,
  5*x1   + x2 + 3*x3 + 8*x4 <= 55,
   -x1  +2*x2 + 3*x3 - 5*x4 <= 3 ],
  4*x1   + x2 + 5*x3 + 3*x4,
 NonNegative] :
t0 :=linopt : :Transparent(Chvatal54)
```

L'algorithme du simplexe donne comme tableau final :

```
t1 :=linopt : :Transparent : :userstep(t0, slk[1], x4)
t2 :=linopt : :Transparent : :userstep(t1, slk[3], x2)
```

Ce calcul donne la solution optimale $(x_1^* := 0,\ x_2^* := 14,\ x_3^* := 0)$.

Ce calcul donne aussi un certificat, mais pour le vérifier, il faut refaire tout le calcul!

Sortons le lapin du chapeau ...

La variable y_1 est associée à la première contrainte, qui elle même est associée à la variable d'écart s_1. Hop, on prends pour y_1^* l'opposé du coefficient de s_1 dans l'expression de z dans le tableau final. De même pour y_2^* et y_3^* :

$$y_1^* := 11,\ y_2^* := 0,\ y_3^* := 6.$$

(y_1^*, y_2^*, y_3^*) est une solution faisable du problème dual.

Par «miracle», on obtient $w^* = z^*$.

On a donc pu lire le certificat voulu directement sur le tableau final !

Voyons maintenant pourquoi cela marche dans le cas général.

DÉMONSTRATION. Il suffit de construire une solution *faisable* $(y_1^*, \ldots, y_m^*)$ vérifiant $w^* = z^*$.

On applique l'algorithme du simplexe au problème initial, en introduisant comme d'habitude les variables d'écart $s_1, \ldots, s_m$. Dans le tableau final, z est de la forme

$$z = z^* + \sum_{j=1}^{n} \overline{c_j} \; x_j + \sum_{i=1}^{m} d_i \; s_i,$$

où les $\overline{c_j}$ et d_i sont des coeffs nuls pour les variables basiques, et négatifs pour les autres.

On pose comme dans l'exemple :

$$y_i^* := -d_i, \text{ pour } i = 1, \ldots, m.$$

Il ne reste plus qu'à vérifier que $(y_1^*, \ldots, y_m^*)$ est faisable et donne $w^* = z^*$.

C'est un calcul fastidieux mais direct ...

Pour une solution quelconque $(x_1, \ldots, x_n)$, on a par définition :

$$z = \sum_{j=1}^{n} c_j \; x_j$$

$$s_i = b_i - \sum_{j=1}^{n} a_{ij} \; x_j$$

En remplaçant dans l'expression ci-dessus, on obtient

$$\sum_{j=1}^{n} c_j \; x_j = z^* + \sum_{j=1}^{n} \overline{c_j} \; x_j - \sum_{i=1}^{m} y_i^* (b_i - \sum_{j=1}^{n} a_{ij} x_j)$$

$$\sum_{j=1}^{n} c_j \; x_j = z^* - \sum_{i=1}^{m} b_i \; y_i^* + \sum_{j=1}^{n} (\overline{c_j} + \sum_{i=1}^{m} a_{ij} \; y_i^*) \; x_j$$

Cette égalité étant vérifiée quel que soit le choix de $(x_1, \ldots, x_n)$, il doit y avoir égalité des coefficients des x_j de part et d'autre. On en déduit d'une part que

$$z^* = \sum_{j=1}^{n} b_i \; y_i^* = w^*,$$

comme voulu, et d'autre part que

$$\sum_{i=1}^{m} a_{ij} \; y_i^* = c_j - \overline{c_j} \geq c_j,$$

c'est-à-dire que $(y_1^*, \ldots, y_m^*)$ est une solution faisable du problème dual. □

1.5.4. Relations entre un problème et son dual.

PROPOSITION. *Le dual du dual d'un problème P est le problème P lui-même.*

EXERCICE. Vérifiez-le sur un exemple.

Il s'ensuit :

THÉORÈME. *On a les relations suivantes entre un problème P et son dual Q :*
P admet une solution optimale si et seulement si Q en admet une.
Si P est faisable, alors Q est borné ; si Q est faisable, alors P est borné.

REMARQUE. Un problème et son dual peuvent être simultanément infaisables !
Maximiser : $2x_1 - x_2$
Sous les contraintes :
$x_1 - x_2 \leq 1$
$-x_1 + x_2 \leq -2$
$x_1, x_2 \geq 0$

Le tableau suivant résume les possibilités (*nb* : un problème non borné est faisable !)

primal\dual	optimal	infaisable	non borné
optimal	possible	impossible	impossible
infaisable	impossible	possible	possible
non borné	impossible	possible	impossible

1.5.5. Notations matricielles.

EXERCICE 6. TODO !Introduire les notations matricielles. Vérifier que prendre le dual revient à transposer et à multiplier par -1. En déduire que le dual du dual de P est P. Redémontrer la proposition et le théorème en utilisant les notations matricielles.

1.5.6. Conditions de complémentarité des variables d'écart.

PROBLÈME. Supposons que l'on connaisse la solution optimale $(x_1^*, \ldots, x_n^*)$ du problème, mais pas le tableau final dans l'algorithme du simplexe. Peut-on retrouver la solution optimale $(y_1^*, \ldots, y_m^*)$ du problème dual de façon à obtenir un certificat ?

Pour voir cela, on va raffiner l'inégalité $w \geq z$ sur des solutions x_j et y_i faisables en utilisant les variables d'écart pour mesurer la différence $w - z$.

EXERCICE 7. On veut introduire des variables d'écart t_i pour le problème dual :

Donner une formule raisonable pour t_i.

Exprimer $w - z$ en fonction des $x_i,\ y_i,\ s_i,\ t_i$.

Par définition des variables d'écart s_i, on a

$$s_i = b_i - \sum_{j=1}^{n} a_{ij} x_j,$$

et donc

$$b_i = s_i + \sum_{j=1}^{n} a_{ij} x_j.$$

De même, par définition des variables d'écart t_j pour le problème dual, on a

$$t_j = \sum_{i=1}^{m} a_{ij} y_i - c_j,$$

que l'on utilise pour exprimer c_j

$$c_j = \sum_{i=1}^{m} a_{ij} y_i - t_j.$$

En remplaçant dans l'expression de $w - z$, on obtient

$$w - z = \sum_{i=1}^{m} b_i y_i - \sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} s_i y_i + \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i - \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j + \sum_{j=1}^{n} t_j x_j$$

Qui se simplifie en :

$$w - z = \sum_{i=1}^{m} s_i y_i + \sum_{j=1}^{n} t_j x_j.$$

PROBLÈME. Que peut-on déduire de cette égalité ?

THÉORÈME. *(Complémentarité des variables d'écart) Si $(x_1^*, \ldots, x_n^*)$ est solution optimale du problème primal et $(y_1^*, \ldots, y_m^*)$ est solution optimale du problème dual, alors :*

$$y_i^* = 0 \text{ ou } s_i^* = 0, \text{ pour tout } i = 1, \ldots, m;$$
$$x_j^* = 0 \text{ ou } t_j^* = 0, \text{ pour tout } j = 1, \ldots, n.$$

PROBLÈME. Et maintenant ? Comment utiliser ce théorème pour trouver $(y_1^*, \ldots, y_m^*)$?

EXERCICE. [**1**, p. 64-65]

THÉORÈME. *Si $(x_1^*, \ldots, x_n^*)$ est une solution basique non dégénérée, alors les équations que l'on tire du théorème de complémentarité ont une unique solution.*

Donc, lorsque la solution optimale du problème est non dégénérée, la technique que l'on a utilisée dans les exercices permet toujours d'obtenir un certificat, pour le prix de la résolution d'un système de m équations linéaires en m variables.

1.5.7. Interprétation géométrique de la dualité.

EXERCICE. Maximiser $x_1 + x_2$

Sous les contraintes

$2x_1 + x_2 \leq 14$

$-x_1 + x_2 \leq 8$

$2x_1 - x_2 \leq 10$

$x_1, x_2 \geq 0$.

Faire une figure dans le plan la région des solutions faisables.

Donner le problème dual.

Prendre $y_1 = y_2 = 1, y_3 = 0$. Donner l'inégalité sur les x_i correspondante, et représenter la région qu'elle délimite dans le plan.

Donner quelques solutions faisables du problème dual.

Tracer sur la figure les régions délimitées par les inégalités correspondantes.

Calculer la solution optimale du primal et du dual.

Les tracer sur la figure.

Essayer d'interpréter géométriquement les théorèmes que l'on a rencontrés.

1.5.8. Interprétation économique des variables duales.

PROBLEM 1.5.1. Modèle économique d'une usine dont on veut maximiser le profit.

Une papetterie produit et vend différents types de papier : du papier kraft vendu au rouleau, du papier recyclé vendu à la ramette et du papier velin vendu à la feuille. Pour celà, elle dispose en début de mois d'un certain stock de matière première : de l'eau (à l'hectolitre), du chlore (au litre) du bois (à la tonne), du vieux papier (au kilo), des fibres textiles (au ballot). Remplacer les stocks en fin de mois à un certain coût. Chaque type de papier nécessite une certaine proportion de chaque matière première. Par exemple, le chlore sert à blanchir le papier ; il n'y en a pas besoin

pour le papier kraft ; le papier velin est essentiellement produit à partir de bois et de fibres textiles, etc. Le but est de prévoir, pour le mois qui vient, quelle quantité de chaque papier il faut produire pour maximiser le profit de la papetterie.

Modéliser ce problème sous forme de programme linéaire sous forme standard.

---

x_j : quantité de produit j fabriquée

c_j : prix de vente unitaire du produit j

a_{ij} : quantité de ressource i consommée par unité de produit j fabriquée

b_i : limites sur la disponibilité de la ressource i

Maximiser :

$$z = \sum_{j=1}^{n} c_j x_j$$

Sous les contraintes :

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \text{ pour } i = 1, \ldots, m;$$

$$x_j \geq 0, \text{ pour } j = 1, \ldots, n.$$

Quelle dimension (au sens physique) ont les variables x_j , b_i , c_j , a_{ij} ?

On voudrait trouver une interprétation pour les variables y_i dans le problème dual. Quelle dimension physique ont-elles ? Qu'est-ce que cela suggère ?

Cela suggère que y_i mesure la valeur intrinsèque de la ressource i pour l'usine.

THÉORÈME. *S'il y a au moins une solution optimale $(x_1^*, \ldots, x_m^*)$ non dégénérée, alors il existe ε strictement positif tel que lorsque $|t_i| \leq \varepsilon$ pour tout i, le programme linéaire relaxé :*

Maximiser :

$$z = \sum_{j=1}^{n} c_j x_j$$

Sous les contraintes :

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i + t_i, \text{ pour } i = 1, \ldots, m;$$

$$x_j \geq 0, \text{ pour } j = 1, \ldots, n.$$

a une solution optimale, et la valeur optimale est

$$z^* + \sum_{i=1}^{m} y_i^* t_i$$

où z^* est la valeur optimale du problème original et $(y_1^*, \ldots, y_m^*)$ est la solution optimale du dual.

Autrement dit, on peut mesurer l'espérance de gain au voisinage d'une solution optimale lorsque l'on relaxe certaines des contraintes : y_i^* décrit le gain que l'usine peut espérer en augmentant la quantité de ressource i disponible.

1.5.9. Problèmes.

PROBLEM 1.5.2. Utiliser le théorème de dualité pour vérifier les solutions des problèmes de programmation linéaire que vous avez résolu jusqu'ici.

PROBLEM 1.5.3. Un bûcheron a 100 hectares de bois de feuillus. Couper un hectare de bois et laisser la zone se régénérer naturellement coûte 10 kF par hectares, et rapporte 50 kF. Alternativement, couper un hectare de bois, et replanter avec des pins coûte 50 kF par hectares, et rapporte à terme 120 kF. Sachant que le bûcheron n'a que 4000 kF en caisse au début de l'opération, déterminer la meilleure stratégie à adopter et le profit escomptable.

Maintenant, le bûcheron a aussi l'option d'emprunter pour augmenter son capital initial, et ce pour un taux d'intérêt total de $S\%$ sur la durée de l'opération. Alternativement, il peut décider d'investir son capital dans une autre activité rapportant $T\%$ sur la durée de l'opération. Déterminer, selon les valeurs de S et T, la meilleure stratégie à adopter.

PROBLEM 1.5.4. Pouvez vous interpréter les conditions de complémentarité des variables d'écart en termes économiques ?

PROBLEM 1.5.5. L'objectif est de démontrer l'un des sens du théorème d'interprétation économique des variables duales. L'autre sens est plus technique, et ne sera pas abordé ici ; voir les références pour les détails.

Soit z^* la valeur optimale du problème primal et $(y_1^*, \ldots, y_m^*)$ une solution optimale quelconque du problème dual. Montrer que pour toute solution faisable $(x_1, \ldots, x_n)$ du problème primal où l'on a relaxé chaque contrainte i de la quantité t_i, on a

$$\sum_{j=1}^{n} c_j x_j \leq z^* + \sum_{i=1}^{m} y_i^* t_i$$

DÉMONSTRATION. Exprimons le fait que $(x_1, \ldots, x_n)$ est solution faisable du problème avec les contraintes relaxées :

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i + t_i$$

Donc :

$$\sum_{i=1}^{m} y_i^* \left(\sum_{j=1}^{n} a_{ij}x_j \right) \leq \sum_{i=1}^{m} y_i^* b_i + \sum_{i=1}^{m} y_i^* t_i = w^* + \sum_{i=1}^{m} y_i^* t_i = z^* + \sum_{i=1}^{m} y_i^* t_i$$

On a trouvé le terme de droite voulu.

Reste à trouver le terme de gauche, ce que l'on fait avec une inversion de somme similaire à celle qui a été utilisée dans les démonstrations précédentes.

$$\sum_{i=1}^{m} y_i^* \left(\sum_{j=1}^{n} a_{ij}x_j \right) = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij}y_i^* \right) x_j \geq \sum_{j=1}^{n} c_j x_j$$

□

Problème. Construire un exemple montrant que la conclusion du théorème est fausse si l'hypothèse de non dégénérescence de la solution optimale est omise.

1.6. Applications

1.6.1. Jeux matriciels.

1.6.1.1. *Le jeu de Morra.* Règles du jeu (pour deux personnes, Louis et Claire).

À chaque tour, chaque joueur cache un ou deux pions, et essaye de parier, à voix haute, combien de pions l'autre joueur a caché. Si un seul des joueurs a parié la bonne solution, son score augmente d'autant de point qu'il y a de pions cachés en tout ; le score de l'autre joueur diminue du même nombre de points. Sinon, rien ne ce passe. Par exemple, si Claire cache 2 pions et parie 1 tandis que Louis cache 2 pions et parie 2, Louis gagne 4 points et Claire en perds 4.

Le but est de trouver une stratégie gagnante.

Exercice 8. Jouez !

À chaque étape, chaque joueur a le choix entre 4 actions :

- [1,1] : Cacher 1 pion, parier 1 pion
- [1,2] : Cacher 1 pion, parier 2 pions
- [2,1] : Cacher 2 pions, parier 1 pion
- [2,2] : Cacher 2 pions, parier 2 pions

Chacune de ces options est appelée *stratégie pure*.

PROBLÈME. Est-ce que suivre une stratégie pure est une stratégie raisonnable ?

Quelles autres stratégies ?

EXERCICE 9. Claire et Louis font un long match.

Stratégie de Claire : inconnue ; elle a joué c_1 fois [1,1], c_2 fois [1,2], c_3 fois [2,1] et c_4 fois [2,2].

Stratégie de Louis : lancer une pièce à chaque tour pour choisir entre [1,2] et [2,1].

Calculer les gains et pertes de Claire et Louis.

Résultat :

Gain de Louis : $(c_1 - c_4)/2$.

Perte moyenne maximale à chaque tour : $1/2$.

Une stratégie aléatoire de ce type est appellée stratégie mixte.

EXERCICE 10. Généralisation : on suppose que Louis se fixe une stratégie mixte. Caractérisez la meilleure stratégie de contre-attaque de Claire, c'est-à-dire celle qui minimise le gain moyen de Louis.

PROBLEM 1.6.1. Comment caractériser la meilleure stratégie mixte pour Louis ?

1.6.1.2. *Jeux matriciels.* Chaque matrice $A = (a_{ij})$ définit un jeu. À chaque tour, le joueur par Ligne (Louis) choisit une ligne i parmi les m lignes, et le joueur par Colonnes (Claire) choisit une colonne j parmi les n colonnes.

Le gain pour Louis est le coefficient a_{ij} :

- Si $a_{ij} \geq 0$, Louis reçoit a_{ij} de Claire
- Si $a_{ij} \leq 0$, Claire reçoit $-a_{ij}$ de Louis
- Si $a_{ij} = 0$, il n'y a pas d'échange

EXERCICE. Écrire la matrice pour le jeu de Morra.

Écrire la matrice pour le jeu papier/ciseaux/caillou/puits.

EXERCICE 11. Dans un long match, Louis adopte une *stratégie mixte*, en choisissant au hasard la stratégie pure i avec une probabilité fixée x_i. Claire joue selon une stratégie de son choix : à la fin du match, elle a joué c_j fois la stratégie pure j.

On note $N := \sum_i c_i$ et $y_i := \frac{c_i}{N}$. Calculer le gain moyen par tour pour Louis.

DÉFINITION. Les vecteurs $x := (x_1, \ldots, x_m)$ et $y := (y_1, \ldots, y_n)$ sont dit *stochastiques* :

$$x_i \geq 0 \text{ et } x_1 + \cdots + x_m = 1.$$

On considère $x := [x_1, \ldots, x_m]$ comme un vecteur ligne, et $y := [y_1, \ldots, y_n]^T$ comme un vecteur colonne, de façon à pouvoir écrire commodément le gain de Louis sous la forme :

$$xAy.$$

EXERCICE 12. Louis adopte une stratégie mixte donnée. Caractériser le gain au pire pour Louis.

Ici, x est constant. Cela peut se mettre sous la forme du programme linéaire en y :

$$\min_y \ xAy$$

Si Louis veut une bonne garantie pour maintenir ces gains hauts (ou ses pertes faibles), il peut chercher une stratégie mixte qui maximise la quantité $\min_y \ xAy$.

On appelle une telle stratégie mixte *optimale* ; son gain moyen vaut

$$\max_x \ \min_y \ xAy$$

PROBLEM 1.6.2. Est-ce que la stratégie mixte optimale est la meilleure stratégie ?

Comment calculer la stratégie optimale ?

Tel quel, le problème ne se met pas sous la forme d'un programme linéaire. On avait vu une astuce pour se débarasser d'un min dans les contraintes ; celle ci ne s'applique cependant que lorsque l'on prend le min d'un nombre fini d'expressions, alors qu'ici il y en a a priori autant que de choix de y.

PROPOSITION. *On peut toujours atteindre la quantité* $\min_y\ xAy$ *avec un* y *de la forme :*

$$(0, \dots, 0, 1, 0, \dots, 0).$$

Autrement dit :

$$\min_y\ xAy = \min_j \sum_{i=1}^{m} a_{ij}x_i.$$

Interprétation ?

DÉMONSTRATION. Clairement, pour une stratégie pure j donnée :

$$\min_y\ xAy \leq \sum_{i=1}^{m} a_{ij}x_i.$$

Maintenant, supposons que j_0 minimise $\sum_{i=1}^{m} a_{ij_0}x_i$:

$$\sum_{i=1}^{m} a_{ij}x_i \geq \sum_{i=1}^{m} a_{ij_0}x_i \text{ pour } j = 1, \dots, n.$$

Alors, si $y := (y_1, \dots, y_n)$ est un vecteur stochastique, on a :

$$xAy = \sum_{j=1}^{n}\sum_{i=1}^{m} x_i a_{ij} y_j = \sum_{j=1}^{n} y_j \left(\sum_{i=1}^{m} a_{ij}x_i\right) \geq \sum_{j=1}^{n} y_j \left(\sum_{i=1}^{m} a_{ij_0}x_i\right) = \left(\sum_{j=1}^{n} y_j\right)\left(\sum_{i=1}^{m} a_{ij_0}x_i\right).$$

Donc, comme voulu,

$$xAy \geq \sum_{i=1}^{m} a_{ij_0}x_i$$

□

EXERCICE 13. Formuler le problème de trouver une stratégie mixte optimale pour Louis comme un programme linéaire.

Supposons que Claire veuille aussi adopter une stratégie mixte optimale. Formuler de même son problème sous forme de programme linéaire.

THÉORÈME. *(Théorème minimax). Pour toute matrice $m \times n$ A, il existe un vecteur stochastique x^* de longueur m, et un vecteur stochastique y^* de longueur n tel que :*

$$\min_y \ x^* A y = \max_x \ x A y^*,$$

où le minimum est pris sur tout les vecteurs stochastiques y de longueur n, et le maximum est pris sur tout les vecteurs stochastiques x de longueur m.

Interprétation ?

DÉMONSTRATION. Application immédiate du théorème de dualité. □

DÉFINITION. Si A est interprétée comme un jeu, la *valeur du jeu* est la quantité :

$$\min_y \ x^* A y = \max_x \ x A y^*.$$

EXERCICE. Calculer la valeur du jeu de Morra et du jeu caillou/pierre/ciseaux/puit.

D'où vient cette particularité ?

1.6.1.3. *Stratégie cachée / stratégie révélée.*

PROBLEM 1.6.3. Est-ce que révéler sa stratégie à son adversaire, diminue l'espérance de gain ?

1.6.1.4. *Morra modifié.* Il n'est pas très pratique de devoir annoncer simultanément les paris.

PROBLEM 1.6.4. Est-ce que le jeu est modifié si Claire annonce toujours son pari en premier ?

EXERCICE. Faire l'analyse de ce nouveau jeu.

1.6.1.5. *Bluff et antibluff.* Jeu de poker avec trois cartes (jeu inventé et analysé par Kuhn en 1950).

A et B déposent chacun un pion, puis reçoivent chacun une carte.

Ensuite, A peut parier un pion supplémentaire ou passer.

De même pour B, puis pour A, jusqu'à ce que :

- Un pari est répondu par un passe : celui qui a parié gagne tous les pions ;
- Un pari est répondu par un pari ou un passe est répondu par un passe : Celui qui a la plus haute carte gagne tous les pions.

EXERCICE. Jouez !

Étant donné une distribution des cartes, décrire les stratégies pures pour A et B.

Décrire toutes les stratégies pures pour A et B.

Quelle est la taille de la matrice des gains ?

Y-a-t'il des stratégies que l'on peut éliminer d'office ?

Au final, on peut obtenir comme matrice de gain :

	124	124	314	324
112	0	0	$-\frac{1}{6}$	$-\frac{1}{6}$
113	0	$\frac{1}{6}$	$\frac{1}{3}$	$-\frac{1}{6}$
122	$-\frac{1}{6}$	$-\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
123	$-\frac{1}{6}$	0	0	$\frac{1}{6}$
312	$\frac{1}{6}$	$-\frac{1}{3}$	0	$-\frac{1}{2}$
313	$\frac{1}{6}$	$-\frac{1}{6}$	$-\frac{1}{6}$	$-\frac{1}{2}$
322	0	$-\frac{1}{2}$	$\frac{1}{3}$	$-\frac{1}{6}$
323	0	$-\frac{1}{3}$	$\frac{1}{6}$	$-\frac{1}{6}$

Stratégie mixte pour A : $[\frac{1}{3}, 0, 0, \frac{1}{2}, \frac{1}{6}, 0, 0, 0]$; stratégie mixte pour B : $[\frac{2}{3}, 0, 0, \frac{1}{3}]^T$.

EXERCICE. Prouver que ces stratégies sont optimales.

EXERCICE. Lorsque A a la carte 1 en main, calculer en quelles proportions il doit choisir entre les 4 stratégies élémentaires.

Résumé de la stratégie de A :

- Avec la carte 1 : mixer 1 et 3 en proportions 5 :1 ;
- Avec la carte 2 : mixer 1 et 2 en proportions 1 :1 ;
- Avec la carte 3 : mixer 2 et 3 en proportions 1 :1.

Pour A, bluffer ou contre-bluffer est rentable.

Résumé de la stratégie de B :

- Avec la carte 1 : mixer 1 et 3 en proportions 2 :1 ;

– Avec la carte 2 : mixer 1 et 2 en proportions 2 :1 ;
– Avec la carte 3 : toujours utiliser 4.

Pour B, bluffer est rentable, mais pas contre-bluffer.

1.7. Réseaux de transport

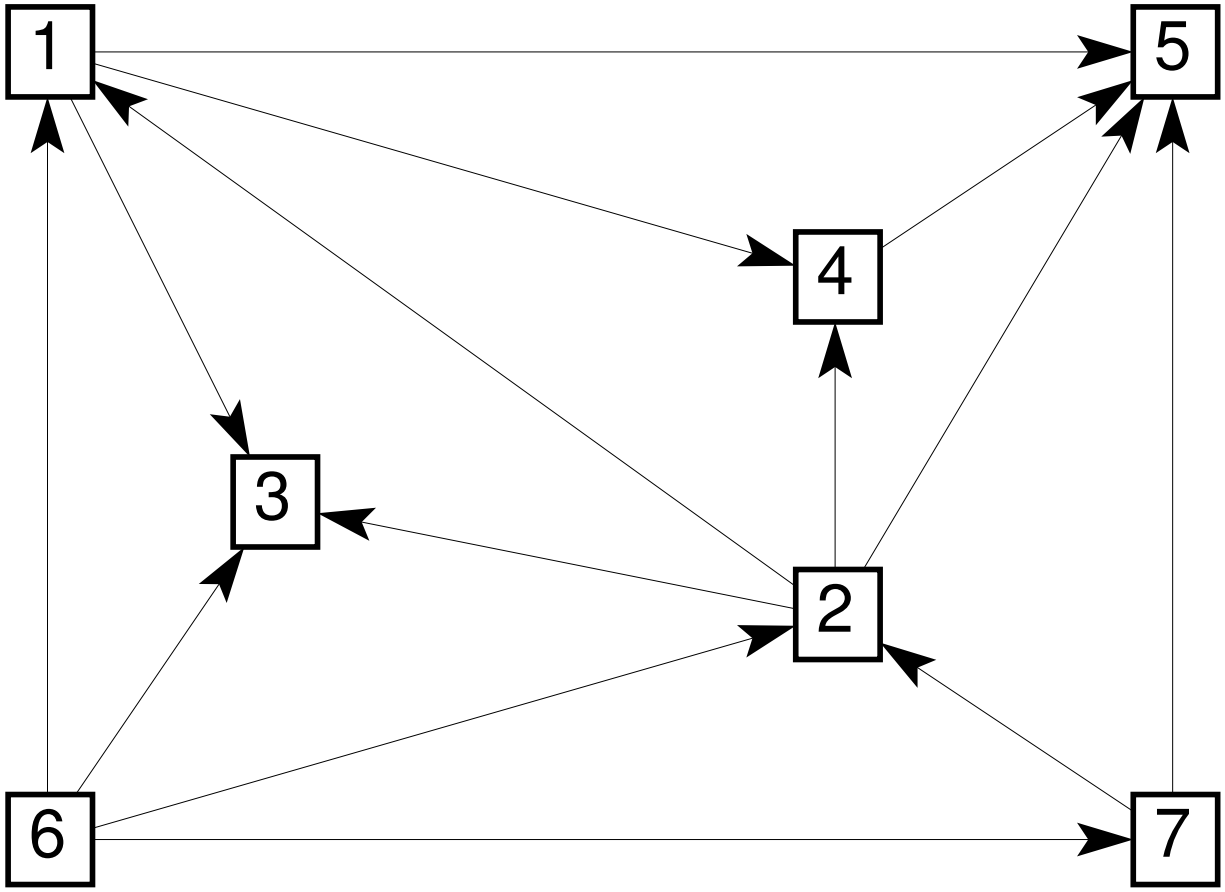
Objectif : étudier une certaine classe de problèmes de programmation linéaire sur laquelle l'algorithme du simplexe prends une forme simple et efficace.

1.7.1. Un exemple d'application.

EXEMPLE. Considérons le problème de transport d'électricité suivant.

Les noeuds sont des villes.

Les arcs sont des câbles électriques, à sens unique, reliant les villes.

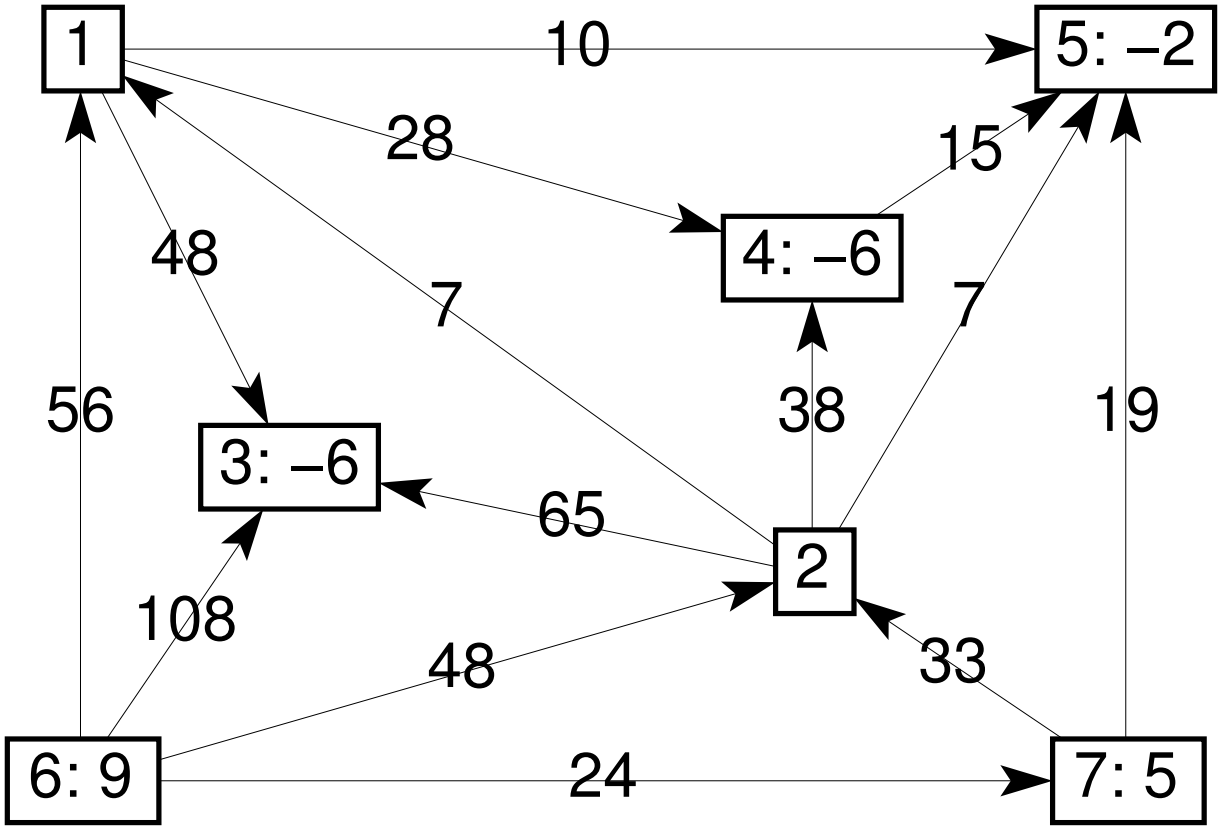


Sources (noeuds avec production) : 6 : 9 MW ; 7 : 5 MW

Puits (noeuds avec consommation) : 3 : 6 MW ; 4 : 6 MW ; 5 : 2 MW

Noeuds intermédiaires (noeuds sans production ni consommation) : 1,2

Il y a des pertes en lignes, donc transporter du courant a un coût. On le modélise par un coût par unité de courant transportée sur chaque arc entre la ville i et la ville j.



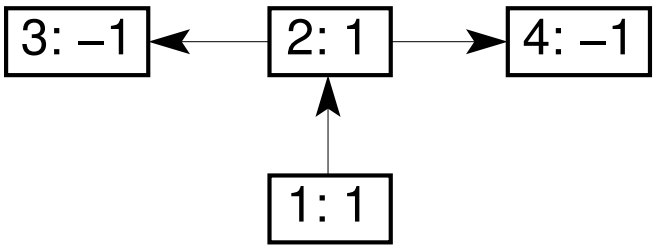
Répartition du flux de courant pour satisfaire la consommation au plus bas coût ?

Exercice. Mettre le problème précédent sous forme de problème de programmation linéaire. Qu'a-t'il de spécifique ?

- Consommation et production sont modélisées par des constantes b_i qui représentent la *demande*. On a pour les puits, $b_i > 0$, pour les sources $b_i < 0$, et pour les noeuds intermédiaires, $b_i = 0$; Attention, sur les graphiques, c'est l'offre qui est représentée ! Le signe est inversé !
- Le coût de transport unitaire le long d'un arc est modélisé par des constantes c_{ij} ;
- On modélise le flux de courant en mesurant par x_{ij} la quantité de courant transférée directement entre les villes i et j.

Remarque. Cette modélisation est *a priori* ambigue : dans le réseau suivant, $x_{12} = 1$, $x_{23} = 1$ et $x_{24} = 1$ peut représenter deux situations différentes :

- Transporter une unité de 1 vers 3 via 2, et une unité de 2 vers 4 directement ;
- Transporter une unité de 1 vers 4 via 2, et une unité de 2 vers 3 directement.



Comme le consommateur ne se soucie pas de l'origine du courant (un watt, c'est un watt), et comme le coût dans les deux situations est le même, on peut ignorer cette ambiguïté, et considérer que ces deux situations sont équivalentes.

1.7.2. Problème standard de transport.

DÉFINITION. Un problème comme le précédent est appelé *problème standard de transport.*

On impose les restrictions suivantes :
– La consommation totale est égale à la production totale ;
– Les arcs sont orientés, avec au plus un arc dans chaque sens ;
– Le réseau est connexe (cf. ci-dessous).

Ces restrictions permettent d'obtenir un algorithme plus simple et élégant. Nous verrons plus tard comment les contourner pour traiter des problèmes plus généraux.

Une solution d'un problème de transport peut être modélisé en introduisant pour chaque arc allant du noeud i au noeud j une variable x_{ij} qui mesure le flux le long de cet arc.

PROPOSITION. *Une solution décrite par les valeurs des x_{ij} est réalisable (faisable) si et seulement si :*
– *Chaque x_{ij} est positif (le flux est dans le sens de l'arc) ;*
– *Pour chaque noeud intermédiaire, le flux entrant est égal au flux sortant ;*
– *Pour chaque source, le flux sortant moins le flux entrant est égal à la production ;*
– *Pour chaque puit, le flux entrant moins le flux sortant est égal à la consommation ;*

Le *seulement si* est clair ; le *si* demanderait une vérification pour décider les détails de la réalisation : quel watt venant d'où se retrouve où au final.

Les solutions faisables sont donc décrites par un système d'équations de la forme :

$$x_{14} + x_{24} - x_{45} = 6.$$

(ici, il s'agit du sommet 4 dans notre exemple), et d'inégalités du type $x_{14} \geq 0$.

EXERCICE. Donner une solution faisable pour notre exemple.

REMARQUE. Pour des raisons de conventions, on note n le nombre de noeuds du réseau, et m le nombre d'arcs. C'est l'inverse de ce que l'on avait utilisé pour les problèmes de programmation linéaire généraux ...

DÉFINITION. La *matrice d'incidence* du réseau est une matrice A de taille $n \times m$. Les lignes sont indexées par les noeuds du réseau, et les colonnes par les arcs. Dans la cellule correspondant à un noeud k et un arc ij, on mets un coefficient valant :

– -1 si l'arc part du sommet ($i = k$),
– 1 si l'arc arrive au sommet ($j = k$),
– 0 sinon.

Là encore les notations ne sont pas parfaites : une paire ij indexe un arc, et donc une colonne, et non pas une cellule de la matrice...

Avec cette notation, et en notant b le vecteur colonne des b_i, x le vecteur colonne des x_{ij}, et c le vecteur ligne des c_{ij} on peut mettre le problème sous forme matricielle :

Minimiser : cx

Sous les contraintes : $Ax = b$ et $x \geq 0$.

EXERCICE. Écrire sous forme matricielle le problème correspondant à notre réseau :

$$x := \begin{bmatrix} x_{13} \\ x_{14} \\ x_{15} \\ x_{21} \\ x_{23} \\ x_{24} \\ x_{25} \\ x_{45} \\ x_{61} \\ x_{62} \\ x_{63} \\ x_{67} \\ x_{72} \\ x_{75} \end{bmatrix}, \; b := \begin{bmatrix} 0 \\ 0 \\ 6 \\ 6 \\ 2 \\ -9 \\ -5 \end{bmatrix}$$

$$A := \begin{bmatrix} -1 & -1 & -1 & 1 & & & & & 1 & & & & & & \\ & & & -1 & -1 & -1 & -1 & & & 1 & & & 1 & & \\ 1 & & & & 1 & & & & & & 1 & & & & \\ & 1 & & & & 1 & & -1 & & & & & & & \\ & & 1 & & & & 1 & 1 & & & & & & & 1 \\ & & & & & & & & -1 & -1 & -1 & -1 & & & \\ & & & & & & & & & & & 1 & -1 & -1 & \end{bmatrix}$$

$$c := \begin{bmatrix} c_{13} & c_{14} & c_{15} & c_{21} & c_{23} & c_{24} & c_{25} & c_{54} & c_{61} & c_{62} & c_{63} & c_{67} & c_{72} & c_{75} \end{bmatrix}$$

$$c := \begin{bmatrix} 48 & 28 & 10 & 7 & 65 & 38 & 7 & 56 & 48 & 108 & 24 & 33 & 19 \end{bmatrix}$$

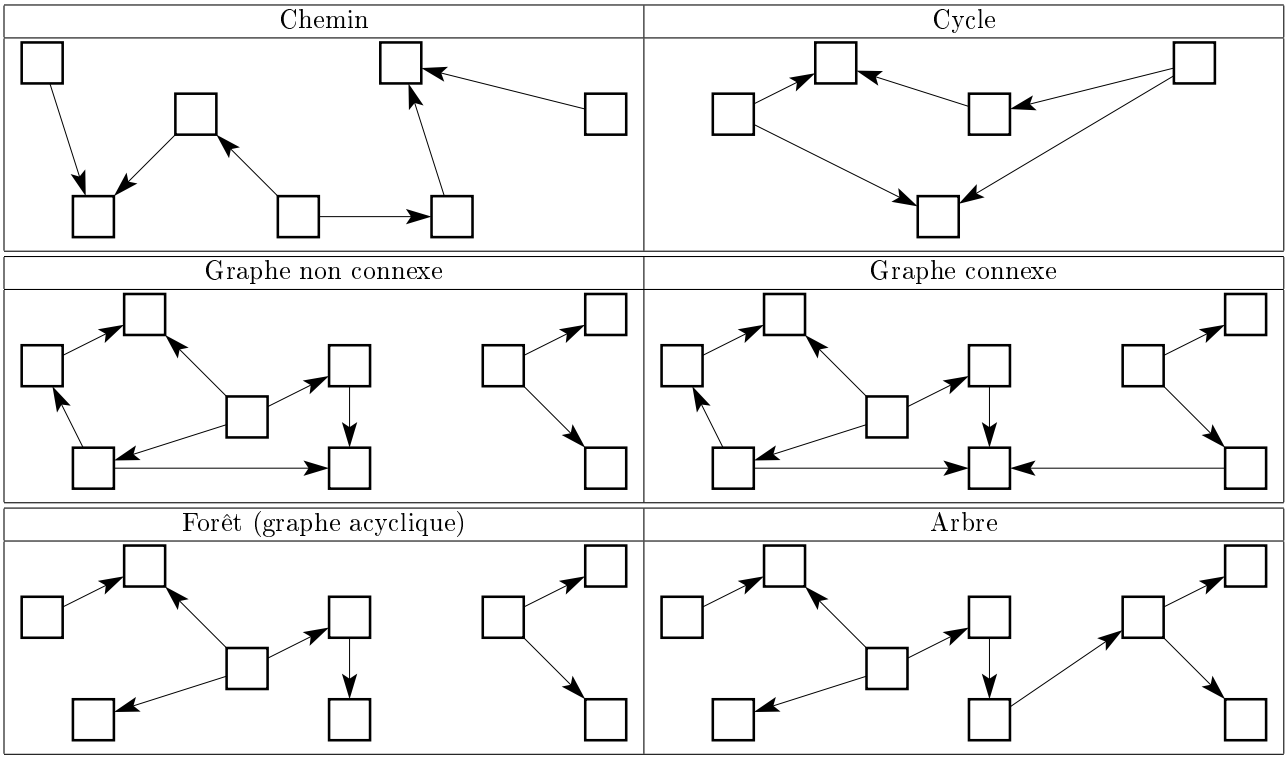
On peut vérifier que dans l´égalité $Ax = b$, la quatrième composante donne l'équation

$$x_{14} + x_{24} - x_{45} = 6,$$

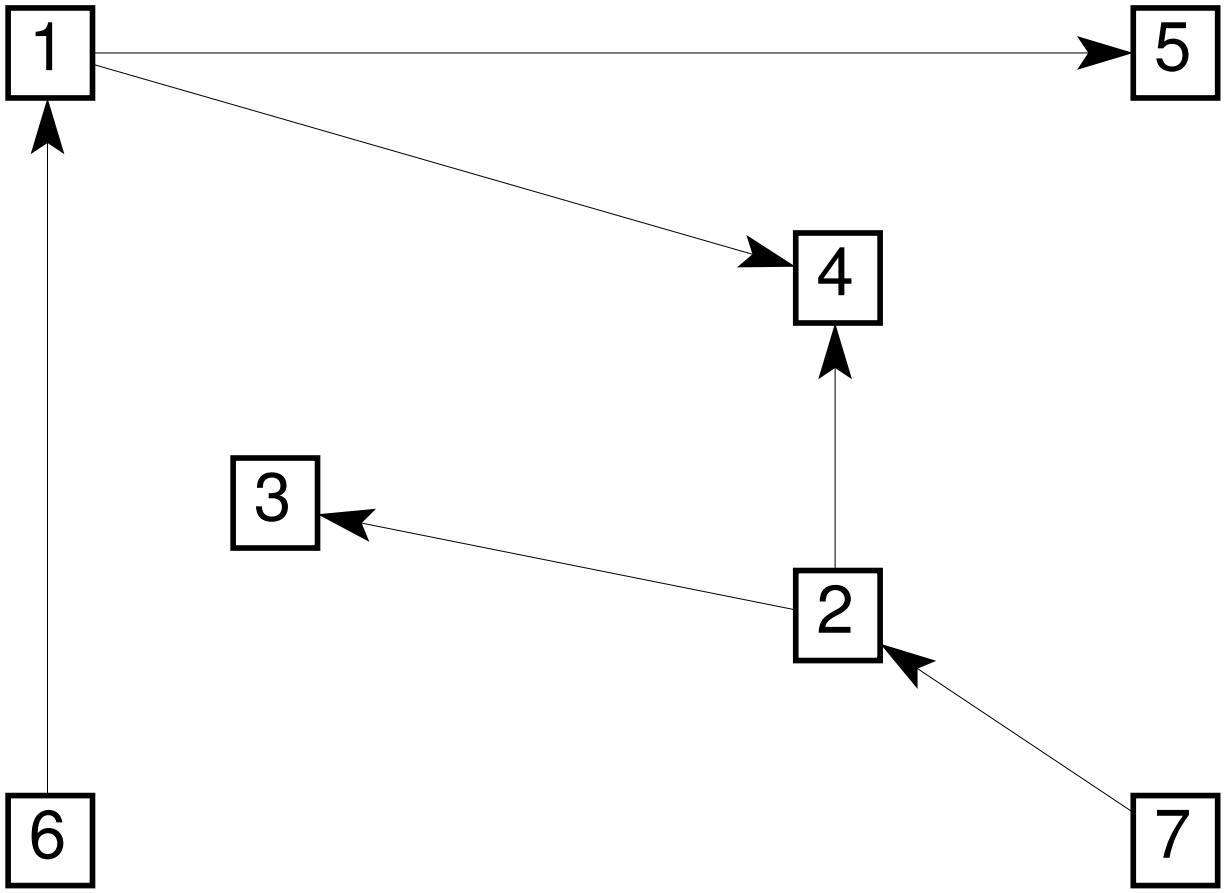
correspondant au sommet 4.

1.7.3. Solutions faisables arborescentes.

DÉFINITION. Quelques classes de graphes classiques :



Arbre couvrant du réseau :



EXERCICE. Supposez que seuls les arcs dans l'arbre couvrant précédent peuvent être utilisés. Y-a-t'il une solution ? Est-elle faisable ?

PROPOSITION. *Étant donné un arbre couvrant T, il y a une unique solution de transport pour satisfaire les contraintes de production et consommation en n'utilisant que les arcs de l'arbre couvrant.*

Formellement, il existe un unique vecteur $x := [x_{ij}]$ vérifiant :

$$Ax = b \text{ et } x_{ij} = 0 \text{ pour } ij \text{ n'appartenant pas à } T.$$

DÉFINITION. On appelle une telle solution *arborescente*.

Si de plus le vecteur x vérifie $x \geq 0$, c'est une *solution arborescente faisable*.

On dit aussi que l'arbre est *faisable*.

1.7.4. Algorithme du simplexe pour les réseaux, une motivation économique.

EXEMPLE. Description de l'algorithme sur le réseau précédent.

1.7.5. Démonstration algébrique de l'optimalité. Soit T un arbre.

On note $x := [x_{ij}]$ la solution correspondante.

Objectif : comparer le coût cx pour la solution x avec le coût $c\overline{x}$pour une autre solution faisable $\overline{x}$.

Soit $y := [y_1, \ldots, y_n]$ les prix à chaque noeuds pour la solution x.

Lors de l'application du simplexe, on a comparé le coût du transport c_{ij} d'une unité le long de l'arc ij par rapport à la différence de prix $y_j - y_i$ entre les noeuds i et j.

On pose $\overline{c_{ij}} := c_{ij} - (y_j - y_i)$, et $\overline{c} = [\overline{c_{ij}}]$ le vecteur ligne correspondant.

EXERCICE. Montrer que $\overline{c} = c - yA$.

LEMME. *On a* $c\overline{x} = cx + \overline{cx}$.

DÉMONSTRATION. On va utiliser le résultat de l'exercice pour reformuler le coût de $\overline{x}$ en fonction de $\overline{c}$:

$$c\overline{x} = (\overline{c} + yA)\overline{x} = \overline{cx} + yA\overline{x} = \overline{cx} + yb.$$

En particulier, $cx = \overline{c}x + yb$.

Comme en plus $\overline{c_{ij}} = 0$ si $ij \in T$ et $x_{ij} = 0$ si $ij \notin T$, $\overline{c}x = 0$, on a $cx = yb$.

Conclusion : $c\overline{x} = cx + \overline{cx}$. □

THÉORÈME. *Si* $\overline{c_{ij}} \geq 0$ *pour tout arc* ij*, alors la solution* x *est optimale.*

Exercice : finissez de le démontrer !

DÉMONSTRATION. Si $\overline{x}$ est une autre solution faisable, $x_{ij} \geq 0$. Donc $\overline{cx} = \sum \overline{c_{ij}} x_{ij} \geq 0$. □

1.7.6. Initialisation. Comment choisir un arbre de départ faisable ?

On va, comme pour le simplexe habituel, introduire un problème auxiliaire :

(1) Choisir un arbre couvrant T.

(2) Calculer la solution x correspondante.

(3) Si pour un arc ij de T on a $x_{ij} < 0$, la solution est infaisable. Ca n'est pas un problème : S'il n'existe pas, on ajoute un arc *artificiel* ji dans le réseau.
On met ji à la place de ij dans T.

(4) L'arbre obtenu est faisable dans le réseau modifié.

Problème : existe-t'il un arbre faisable dans le réseau modifié n'utilisant pas d'arc artificiel ?

On prend comme fonction de coût $w := \sum_{ij \text{ artificiel}} x_{ij}$.

De la sorte, si x est une solution faisable du problème original, alors $w = 0$.

On applique le simplexe. À la fin, on est dans l'une des situations suivantes :

(1) $w^* > 0$: Le problème original est infaisable.

(2) $w^* = 0$, et l'arbre final T_1 n'utilise aucun arc artificiel :
T_1 est une solution faisable du problème initial, comme voulu.

(3) $w^* = 0$, et l'arbre final T_1 utilise au moins un arc artificiel :
On a clairement $x^*_{ij} = 0$ pour tous les arcs artificiels.
Comme le réseau est connexe, on peut toujours échanger les arcs artificiels par d'autres arcs non artificiels, sans changer les x^*_{ij}.

1.7.7. Terminaison et cyclage. Comme dans le cas général, l'algorithme du simplexe pour les réseaux a les propriétés suivantes :

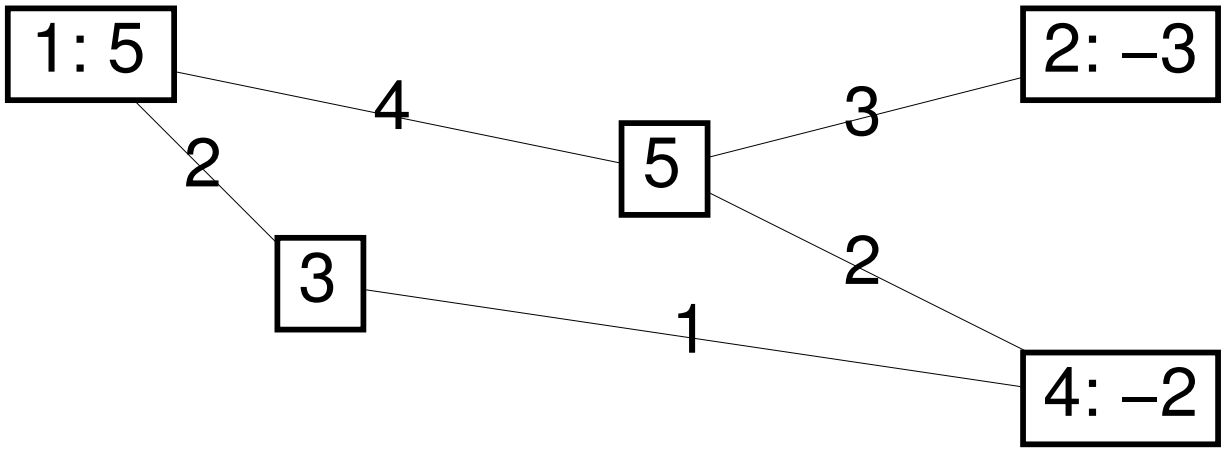
– Il peut y avoir des situations dégénérées (pivot ne changeant pas le coût) ;
– L'algorithme peut cycler, mais seulement en présence de dégénérescence ;
– S'il ne cycle pas, alors il termine ;
– Il y a des stratégies efficaces pour éviter les cycles.
– Même en évitant les cycles, la complexité au pire peut monter jusqu'à 2^n ;
– Dans la pratique, il ne cycle jamais ; la complexité est inférieure à n.

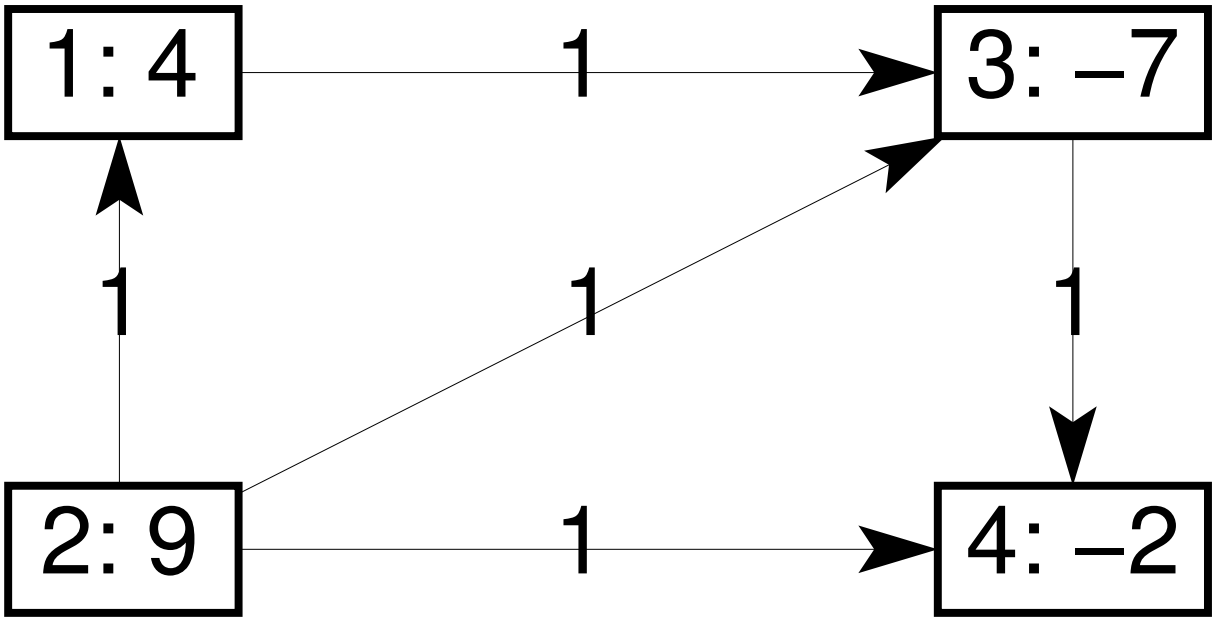
Pour les détails, nous renvoyons à [**1**, Ch. 19].

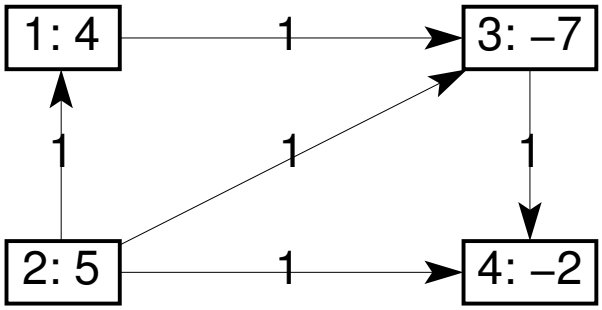
1.7.8. Comment contourner les restrictions ? Dans les exercices suivants, on cherche à contourner les restrictions sur les problèmes standards de transport.

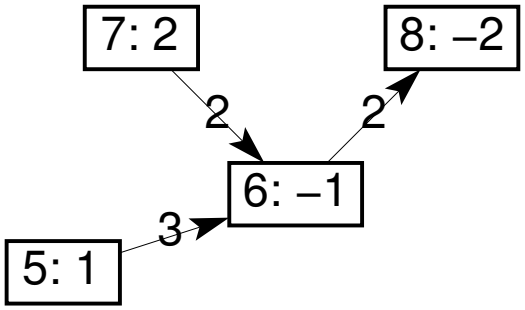
– Arcs orientés
– Égalité de la production et de la consommation
– Connexité
– Au plus un arc dans chaque sens entre deux sommets

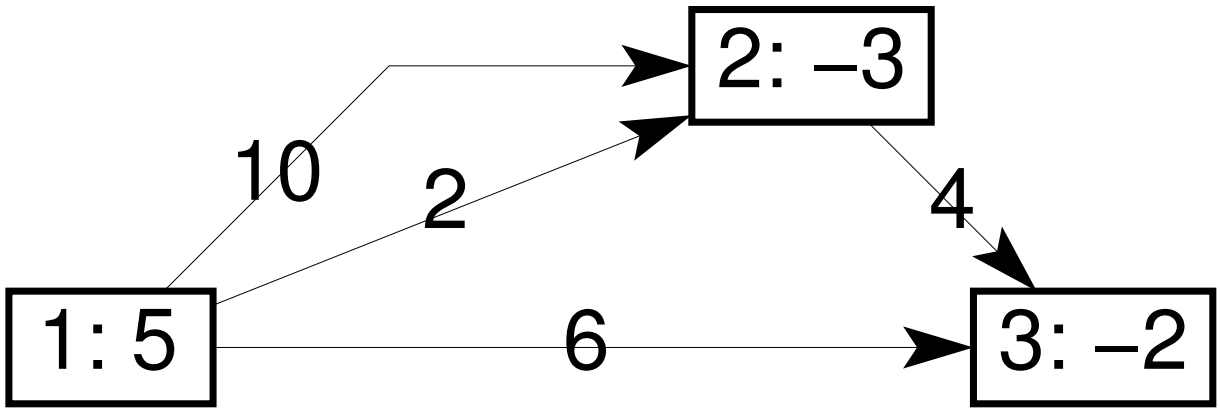
Exercice. Dans les problèmes suivants, on veut répartir au mieux le transport d'oranges via des réseaux ferroviaires, avec les productions et consommations indiquées sur les noeuds, et les coûts de transports indiqués sur les arcs. Pour chacun d'entre eux, indiquer si on peut le modéliser sous forme de problème standard de transport, et si oui, comment.


1: 5
2: –3
4
3
5
2
2
1
4: –2


1: 4
1
3: –7
1
1
1
2: 9
1
4: –2


1: 4
1
3: –7
1
1
1
2: 5
1
4: –2


7: 2
8: –2
2
2
6: –1
3
5: 1



1.8. Applications du simplexe des problèmes de transports

PROBLEM 1.8.1. Une usine de barrettes mémoire pour ordinateur doit faire face à une demande fluctuante dans le temps. On suppose que pour l'année à venir, la demande d_j pour chaque mois j est connue à l'avance (cette hypothèse vaut ce qu'elle vaut).

Pour adapter la production à la demande, l'usine a le choix entre plusieurs options :
– Stocker des barrettes d'un mois sur l'autre, sans limite, mais pour un coût unitaire de a.
– Produire, dans la limite de r barrettes par mois, pour un coût unitaire de b.
– Surproduire, dans la limite de s barrettes par mois, moyennant un coût unitaire plus élevé c.
À la fin de l'année, on veut de plus qu'il n'y ait plus aucun stock, afin de faciliter l'inventaire.

Évidemment, l'objectif est d'adapter la production au moindre coût.

Modéliser ce problème sous forme de problème de transport standard.

1.8.1. Un problème d'assignement.

EXERCICE. Répartition de cours entre plusieurs professeurs.

Dans le département de mathématiques d'une université aux USA, l'évaluation des enseignants par les étudiants a donné au cours des derniers semestres les résultats suivants :

Cours\Professeur	Bill	Yu	Luis	John	Hugh
Calculus 1	3	4	2,3	2,9	2,8
Differential Equations	2,25	3,2	3,7	1,9	4,4
Statistics	2,6	3,7	4,5	2.7	3,1
Calculus 2	3,9	4,1	2,6	3,9	2,4
Discrete maths	2,8	2,8	3,5	3,4	4,2

Dans un semestre, chaque cours est enseigné par un professeur, et chaque professeur enseigne un cours. Le chef du département veut répartir les cours du prochain semestre entre les professeurs de façon à exploiter au mieux leurs talents respectifs (ou minimiser la grogne des étudiants, au choix. . .). Il décide de prendre comme mesure de la qualité d'une répartition la moyenne sur chaque cours de la note du professeur qui l'enseigne.

Modéliser le problème, et indiquer comment on pourrait le résoudre.

PROBLÈME. Est-on sûr d'obtenir une solution entière ?

THÉORÈME. *(dit d'intégralité) Soit P un problème standard de transport où les contraintes sont entières (i.e. les b_i sont entiers). Alors :*

(1) *Si P a une solution, alors il a une solution à coefficients entiers ;*
(2) *Si P a une solution optimale, alors il a une solution optimale à coefficients entiers.*

DÉMONSTRATION. Une solution arborescente pour P a toujours des coefficients entiers !
En effet, la matrice d'incidence de l'arbre a des coefficients 1, -1 et 0, et on peut la mettre sous forme triangulaire avec des coefficients 1 et -1 sur la diagonnale. Du coup, lorsque l'on calcule le flux le long des arcs de l'arbre (ce qui revient à inverser la matrice), on obtient uniquement des flux entiers. □

Le théorème d'intégralité est assez simple. Alors quel est son intérêt ?

Le problème précédent est appellé *problème d'assignement*, et est essentiellement *combinatoire* (les variables sont discrètes).

Ce que dit fondamentalement le théorème d'intégralité, c'est que dans certains cas les méthodes de programmation linéaire peuvent être utilisées pour résoudre des problèmes purement combinatoire, ce qui est loin d'être trivial !

C'est le sujet de la *combinatoire polyhédrale*.

1.8.2. Quelques commentaires sur la programmation linéaire en coefficients entiers. Les problèmes de programmation linéaire en entiers (on impose que les solutions soit à coordonnées entières) sont notoirement difficile. Ils sont la plupart du temps NP-complets, et nécessitent la plupart du temps l'utilisation d'algorithme de backtrack (essai-erreur) qui ne sont pas polynomiaux.

Par contre, si par chance on peut les mettre sous forme de problèmes standards de transport, le théorème d'intégralité permet de les résoudre par l'algorithme du simplexe.

EXEMPLE. Un problème de sac-à-dos :

On a des objets de différentes tailles $l_1, \dots, l_n$ que l'on veut mettre dans un sac-à-dos de taille l. Évidemment le sac est trop petit, et l'on doit donc faire un choix. Le but est de remplir au maximum le sac-à-dos. Cela peut se mettre sous la forme :

Maximiser $\sum_{i=1}^{n} x_i l_i$, sous les contraintes $0 \leq x_i \leq 1$, x_i entier.

Peut-on le mettre sous forme de problème de transport ?

EXEMPLE. Le problème d'assignement cours/professeurs.

PROBLÈME. Est-ce que ce ce problème est polynomial ?

On note que l'algorithme du simplexe *n'est pas polynomial !*

Par contre, il existe un autre algorithme, dit de l'*ellipsoïde*, pour résoudre les problème de programmation linéaire qui est polynomial. Il est amusant de constater qu'en pratique il est moins efficace que l'algorithme du simplexe. Nous renvoyons au Chvatal pour une description complète de cet algorithme.

Toujours est-il que cela peut permettre de montrer que le problème d'assignement est polynomial.

1.8.3. Combinatoire polyhédrale. Pour des détails sur ce domaine, nous recommandons particulièrement la lecture de l'article sur le sujet dans le *handbook of combinatorics.*

L'idée générale de la combinatoire polyhédrale est la suivante :

- On part d'un problème d'optimisation combinatoire du type «maximiser une certaine quantité sur un ensemble discret E»
- On le transforme en problème de programmation linéaire, en assignant aux éléments de E des vecteurs, et en considérant le polyhèdre convexe qu'ils engendrent.
- Ce convexe peut être décrit par des inégalités (Comment ? c'est l'étape difficile !).
- On a alors un problème de programmation linéaire, que l'on peut résoudre.
- Si tout ce passe bien, le théorème d'intégralité garantit que la solution trouvée correspond bien à un des vecteurs extrémaux correspondant aux éléments de E.

Le théorème de dualité donne alors des relations de type min-max entre des problèmes combinatoires.

1.8.3.1. *Théorème de König (lemme des mariages).*

THÉORÈME. *On a un ensemble de n filles et n garçons que l'on veut marier ensemble. Dans notre grande magnanimité, on veut bien faire attention à ne pas marier deux personnes qui ne se connaissent pas.*

Si chaque fille connait exactement $k \geq 1$ garçons et chaque garçon connaît exactement k filles, alors on peut arranger n mariages de façon à ne pas marier des inconnus.

EXERCICE. Prouvez ce théorème en construisant le problème de transport qui va bien.

DÉMONSTRATION. On associe à chaque fille i une source r_i produisant une unité, et à chaque garçon un puit s_i consommant une unité, et on met un arc entre chaque couple r_i s_i se connaissant.

Indépendamment du coût sur les arcs, le problème est faisable :

Il suffit de mettre un flux de $\frac{1}{k}$ sur chaque arc.

Le théorème d'intégralité indique alors qu'il y a une solution entière.

Cette solution entière donne une façon d'organiser les mariages. □

1.8.3.2. *Matrices doublement stochastiques.*

DÉFINITION. Une matrice $X = [x_{ij}]$ de taille $n \times n$ est *doublement stochastique* si les coefficients x_{ij} sont positifs et si la somme des coefficients sur chaque ligne et chaque colonne vaut 1 :

$$\begin{bmatrix} 0,5 & 0,2 & 0,3 \\ 0,01 & 0,7 & 0,29 \\ 0,49 & 0,1 & 0,41 \end{bmatrix}.$$

X est une *matrice de permutation* si sur chaque ligne et chaque colonne il y a exactement un 1 et $n-1$ zéros :

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

La matrice précédente correspond à la permutation $3, 1, 2$.

Clairement une matrice de permutation est une matrice doublement stochastique.

Les matrices de permutations sont les matrices doublement stochastiques à coeffs entiers.

EXEMPLE. Dimension 2 : quelles sont les matrices doublement stochastiques ? quelles sont les matrices de permutations ?

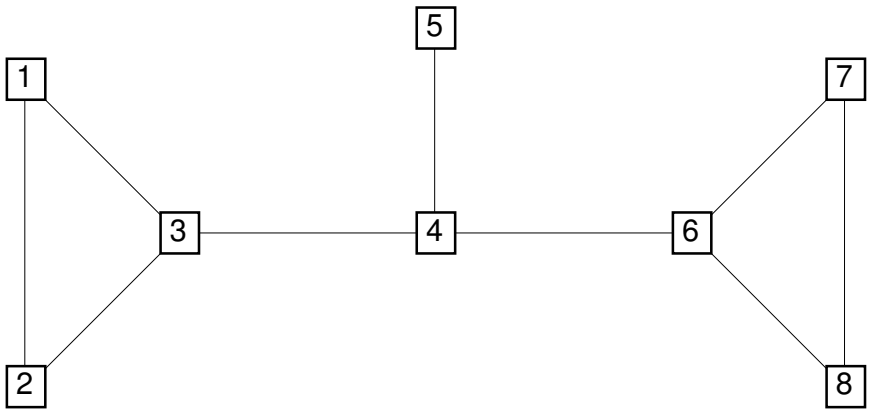
THÉORÈME. *(Birkhoff-Von Neumann) Toute matrice doublement stochastique est une combinaison linéaire convexe de matrices de permutations.*

EXERCICE 14. Écrire la matrice doublement stochastique ci-dessus comme combinaison linéaire convexe de matrices de permutations.

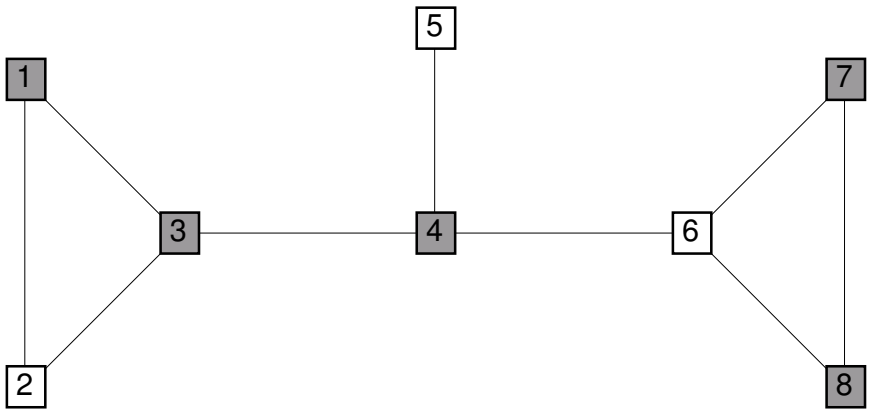
LEMME. *Pour toute matrice X doublement stochastique, on peut trouver une matrice de permutation Y de façon à ce que si $x_{ij} = 0$ alors $y_{ij} = 0$.*

EXERCICE 15. Démontrer ce lemme en utilisant un réseau de transport adéquat, et déduisez-en le théorème.

1.8.3.3. *Couvertures et couplages dans les graphes bipartis.* On va maintenant regarder une application de la programmation linéaire pour étudier des graphes non orientés comme le suivant :



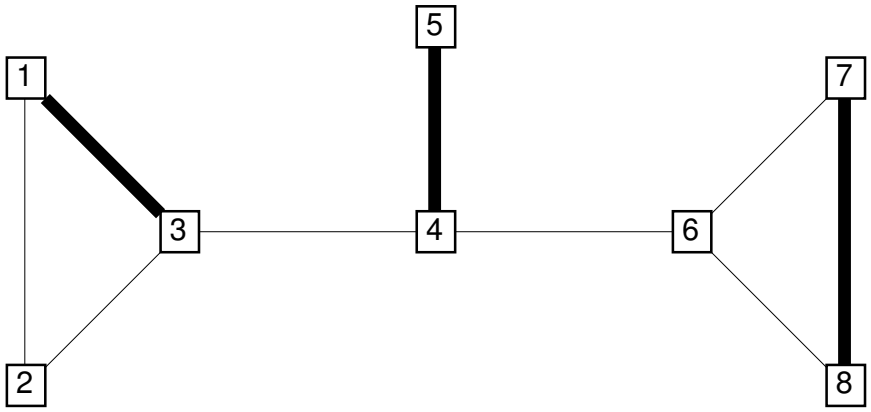
Une *couverture* C de ce graphe est un ensemble de sommets qui touchent toutes les arêtes, comme par exemple $C := \{1, 3, 4, 7, 8\}$:



EXEMPLE 1.8.2. On a 8 petits villages reliés par des routes. En cas d'accident de la route, on veut que les pompiers puissent intervenir rapidement. Le prefet impose que lorsqu'une route relie deux villages, il y ait une caserne de pompier dans au moins l'un des deux villages. Évidemment le budget est serré, donc on veut construire des casernes de pompier dans un nombre minimal de villages.

Modélisation : Chaque village est représenté par un sommet du graphe précédent, les arêtes représentant les routes. Résoudre notre problème revient à chercher une couverture de taille minimale du graphe.

Un *couplage* M de ce graphe est un ensemble d'arêtes qui ne se touchent pas, comme par exemple $M := \{\{1, 3\}, \{4, 5\}, \{7, 8\}\}$:



EXEMPLE 1.8.3. On veut loger un groupe de 8 personnes dans un hotel, avec des chambres simples et doubles. Pour minimiser les dépenses, on utiliser le maximum de chambres doubles. D'un autre côté on ne veut pas forcer deux personnes qui ne se connaissent pas bien à partager une chambre.

Modélisation : chaque sommet du graphe précédent représente une personne, et chaque arête relie deux personnes qui se connaissent bien. Résoudre notre problème revient alors à rechercher un couplage de taille maximale dans le graphe.

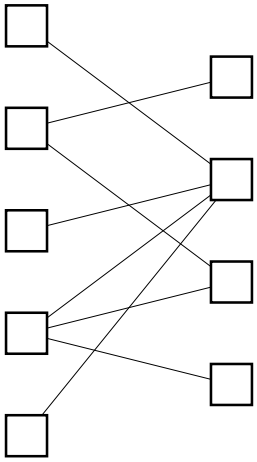
EXERCICE 16. Montrer que pour un couplage M et une couverture C d'un même graphe, on a toujours $|M| \leq |C|$.

DÉMONSTRATION. Comme C est une couverture, chaque arête de M devra être touchée par au moins un sommet dans C. De plus, M étant un couplage, chaque sommet de C touche au plus une arête de M. Donc, on a bien $|M| \leq |C|$. □

PROBLEM 1.8.4. Peut-on trouver M et C de façon à avoir égalité ?

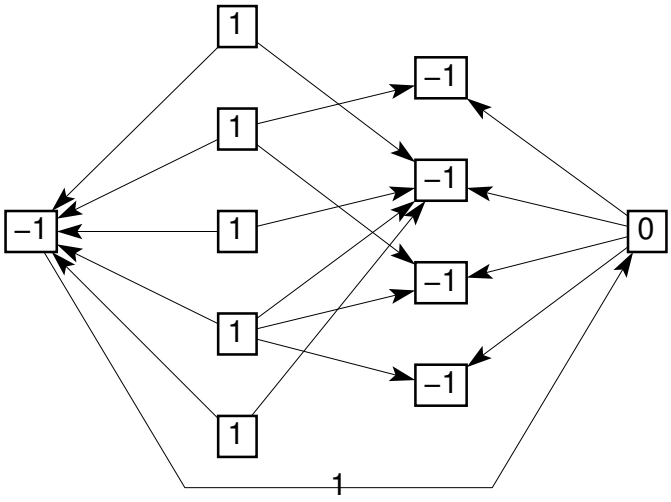
Dans notre exemple, non. Par contre, on va voir que pour certaines classes de graphe, cela va être vrai : on aura un théorème de dualité min-max. Comme par hasard, c'est une conséquence de la programmation linéaire.

On appelle *graphe biparti* un graphe dont on peut partitioner les sommets en deux paquets A et B de sorte que toutes les arêtes soient entre A et B :



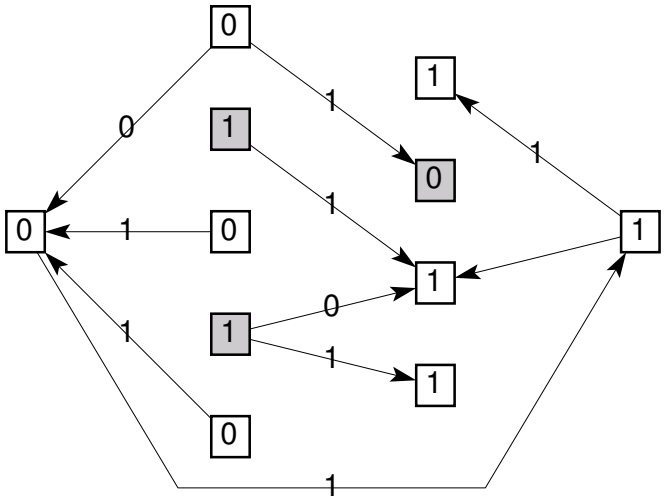
EXERCICE 17. On veut rechercher un couplage maximal du graphe précédent. Montrer comment on peut résoudre ce problème en utilisant un réseau de transport.

On peut par exemple introduire le réseau suivant.



Chaque solution entière du réseau correspond à un couplage M du graphe biparti (les arêtes sur lesquelles passent une unité). Le coût de cette solution est $4-|M|$. Donc minimiser ce coût revient à rechercher un couplage de taille max.

Voilà une solution arborescente optimale du réseau ; on a indiqué sur les sommets les prix relatifs, et sur les arêtes les quantités transportées :



La taille maximale d'un couplage M est donc 3.

On remarque que les sommets du graphe biparti de prix 1 à gauche et de prix 0 à droite (en grisé) forment une couverture optimale de taille 3 du graphe biparti.

PROBLEM 1.8.5. Est-ce une coïncidence ?

EXERCICE 18. Soit T une solution arborescente optimale pour le réseau associé à un graphe biparti quelconque. On définit M et C comme ci-dessus.

(1) Vérifier que si ij est une arête du graphe biparti, et si $i \notin C$, alors $j \in C$.
(2) En déduire que C est une couverture du graphe biparti.
(3) Vérifier que si i est dans C, alors i appartient à une des arêtes du couplage M.
(4) Vérifier que si ij est une des arêtes du couplage M, alors i et j ne sont pas simultanément dans C.
(5) En déduire que $|M| = |C|$.

THEOREM 1.8.6. *(König-Egerváry) Dans tout graphe biparti, la taille d'un couplage maximal est égale à la taille d'une couverture minimale.*

C'est une exemple typique où le théorème de dualité de la programmation linéaire donne un théorème min-max reliant deux problèmes combinatoires qui ne sont pas clairement reliés a priori.

1.9. Problèmes de transports avec limites de capacités

EXEMPLE. Modéliser un réseau routier par un réseau de transport sous forme standard n'est pas très réaliste : sur une autoroute donnée, on ne peut pas faire passer autant de camions que l'on veut !

Dans cette section, nous allons regarder une généralisation des problèmes de transports, dans lesquels on ajoutera des contraintes de capacités maximales.

Nous verrons rapidement que l'algorithme du simplexe et les résultats théoriques en découlant peuvent être étendus sans grosses difficultés. Puis nous étudierons quelques applications.

DÉFINITION. Problème de transport avec limites de capacités sous forme standard :

Minimiser : cx

Sous les contraintes : $Ax = b$ et $0 \leq x \leq u$,

où A est la matrice d'incidence d'un réseau.

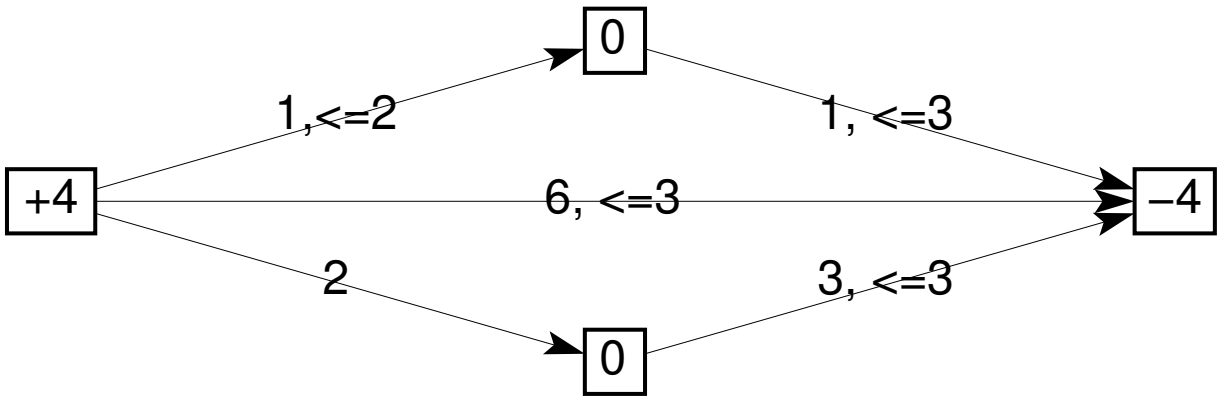
(u pour *upper bound*).

EXERCICE. Peut-on mettre le problème suivant sous forme standard ?

Minimiser : cx

Sous les contraintes : $Ax = b$ et $l \leq x \leq u$.

EXERCICE. Donner une solution optimale pour le problème suivant :



Comme on peut le constater dans l'exercice précédent, on ne peut pas toujours espérer trouver une solution optimale, ni même une solution faisable arborescente : c'est-à-dire telle qu'on utilise aucune arête en dehors d'un certain arbre T.

On va donc relâcher cette contrainte. Toute arête en dehors de l'arbre T devra être soit non utilisée, soit au contraire utilisée à pleine capacité :

DÉFINITION. Soit T un arbre couvrant du réseau.
Une solution x est T-*arborescente* si tout arc $ij \notin T$ on a :

$$x_{ij} = 0 \text{ ou } x_{ij} = u_{ij}.$$

Une solution x est *arborescente* s'il existe un arbre couvrant T tel que x est T-arborescente.

EXERCICE. Donner une solution arborescente au problème précédent.

EXERCICE. Étant donné un arbre T, a-t'on unicité de la solution arborescente vis-à-vis de cet arbre ?

EXERCICE. Montrer que si les b_i et les u_{ij} sont entiers, alors toute solution arborescente est entière.

Nous allons maintenant voir sur un exemple comment on peut adapter l'algorithme du simplexe pour les réseaux.

Dans le cas classique (sans limites de capacités), le principe était de faire rentrer dans l'arbre T une arête ij inutilisée ($x_{ij} = 0$) rentable ($y_i + c_{ij} < y_j$) de façon à pouvoir l'utiliser.

Ici, il y a un autre cas de figure : faire rentrer une arrête ij utilisée à pleine capacité ($x_{ij} = u_{ij}$) alors qu'elle n'est pas rentable ($y_i + c_{ij} > y_j$), de façon à pouvoir diminuer son utilisation.

EXEMPLE 1.9.1. Cf. [**1**, p.356-359].

THÉORÈME. *Soit x une solution T-arborescente telle que pour toute arête ij en dehors de l'arbre, on ait :*
– *soit $x_{ij} = 0$ et $y_i + c_{ij} \geq y_j$,*

– *soit* $x_{ij} = u_{ij}$ *et* $y_i + c_{ij} \leq y_j$.
Alors x *est une solution optimale.*

DÉMONSTRATION. La démonstration est très similaire à celle du cas sans limites de capacités.

On va considérer une autre solution faisable $\overline{x}$ du problème, et comparer les coûts cx et $c\overline{x}$ correspondants.

On pose $\overline{c_{ij}} := c_{ij} - (y_j - y_i)$, et $\overline{c} = [c_{ij}]$ le vecteur ligne correspondant.

$\overline{c_{ij}}$ mesure le coût relatif :
– $\overline{c_{ij}} = 0$ si ij est dans T
– $\overline{c_{ij}} \geq 0$ si $x_{ij} = 0$ (non rentable d'utiliser l'arc ij).
– $\overline{c_{ij}} \leq 0$ si $x_{ij} = u_{ij}$ (rentable d'utiliser l'arc ij à pleine capacité).
On note que dans les trois cas, $\overline{c_{ij}x_{ij}} \geq \overline{c_{ij}}x_{ij}$. Donc matriciellement $\overline{cx} \geq \overline{c}x$.

Comme précédemment on peut écrire $\overline{c}$ matriciellement sous la forme $\overline{c} = c - yA$.

De plus, x et $\overline{x}$ sont solutions faisables et vérifient donc toutes deux $Ax = b$.

On en déduit alors :

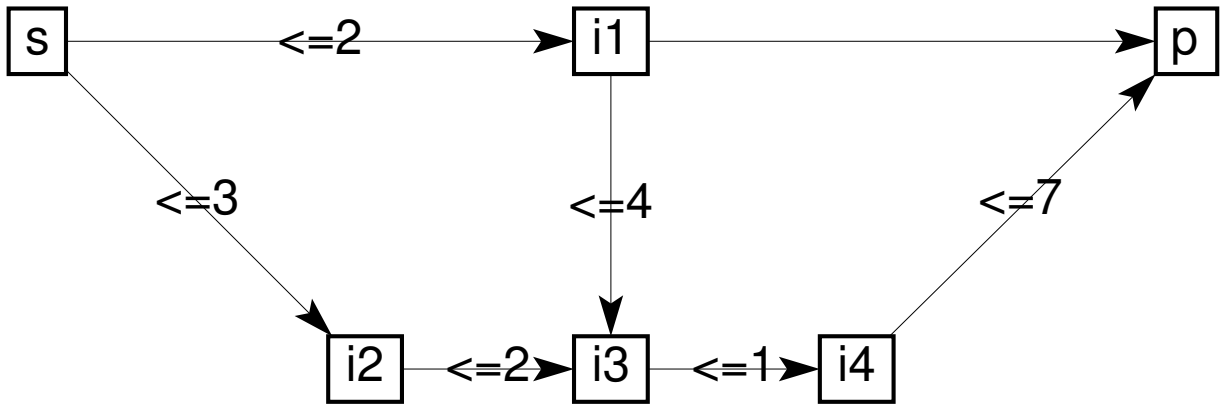
$$c\overline{x} = \overline{cx} + yA\overline{x} = \overline{cx} + yb \geq \overline{c}x + yb = \overline{c}x + yAx = cx.$$

□

1.10. Problèmes de flot maximum

1.10.1. Introduction.

DÉFINITION. *Problème de flot max :*
– Réseau avec source et puits
– Pas de coûts sur les arcs
– Contraintes de capacités sur les arcs
– Production et consommation nulle sur chaque noeud intermédiaire



Objectif :

Maximiser le *volume* du flot, c'est-à-dire la quantité transportée entre s et p.

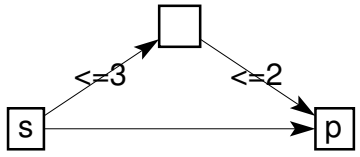
EXEMPLE. Un dimanche soir, maximiser le nombre de voitures allant d'Albertville à Lyon, en les répartissant entre les différentes routes possibles.

EXERCICE. Mettre le problème de flot dessiné ci-dessus sous forme de problème de transport standard avec limites de capacités

Clairement, cela se généralise à tout problème de flot max.

PROBLÈME. Que peut-on en déduire ?

– On a un algorithme de résolution (simplexe)
– Le problème de flot est polynomial
– On a un théorème d'intégralité :
  Si les contraintes de capacités sont entières (ou infinies), alors :
  – Soit le problème est non borné :



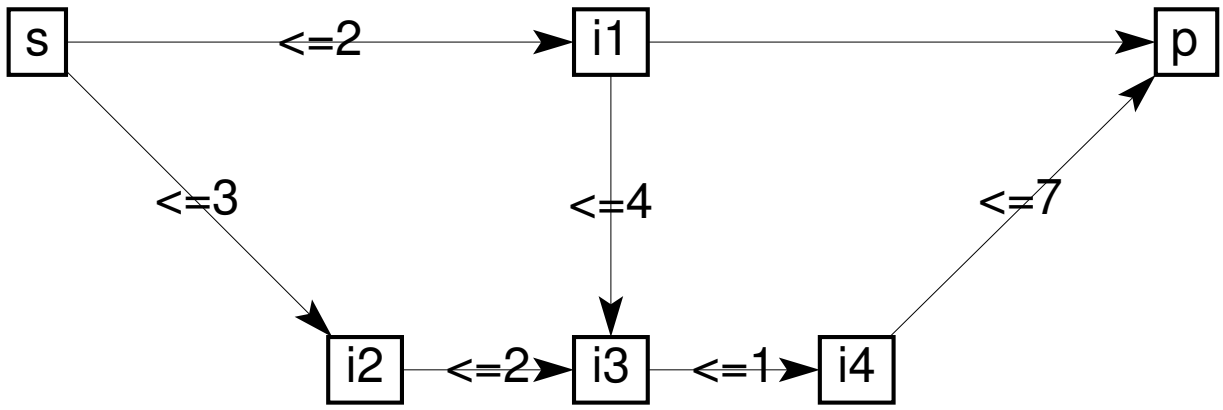
  – Soit le problème a une solution entière
– On doit bien avoir une dualité !

1.10.2. Dualité : le théorème flot max / coupe min.

DÉFINITION. Une *coupe* C dans un réseau est un ensemble de sommets du réseau contenant la source.

La capacité de la coupe C est la somme des capacités des arrêtes sortantes de C.

EXEMPLE. Dans notre réseau, la coupe $C = \{s\}$ est de capacité 5.



EXERCICE. Quelle est la capacité de la coupe $C = \{s, i_2, i_3\}$?
Que peut-on en déduire sur la valeur d'un flot ?

PROPOSITION. *Pour toute coupe C et tout flot F dans un réseau, la capacité $|C|$ de la coupe est supérieure au volume $|F|$ du flot : $|C| \geq |F|$.*

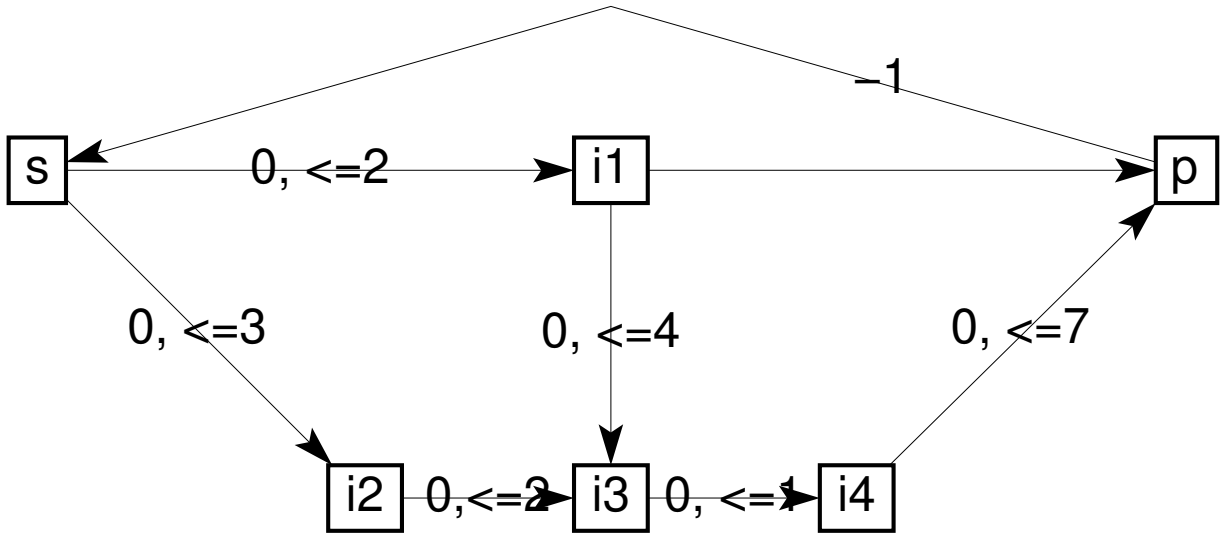
PROBLÈME. Que peut on espérer avoir ?

Une dualité et un théorème min-max, bien-sûr !

THÉORÈME. *(Coupe min-Flot max) Dans un réseau, le volume maximal d'un flot est égal à la capacité minimale d'une coupe.*

EXERCICE. Vérifiez-le dans notre exemple.

DÉMONSTRATION. On considère une solution F optimale du problème de flot obtenue avec le simplexe pour les réseaux avec limites de capacité.



On calcule les valeurs y_i en chaque sommets.
Les coûts sont de 0 partout sauf sur l'arc ps, où le coût est de -1.
Donc la valeur de y_i est :

– 0 si i est relié à s dans T,
– 1 si i est relié à t dans T.

On prend comme coupe C l'ensemble des sommets i avec $y_i = 0$.

Chaque arc ij sortant de C est «rentable» puisque $y_i + c_{ij} = 0 < 1 = y_j$.

Or, ij n'est pas dans l'arbre, et le flot est optimal.

Donc ij doit être utilisé à pleine capacité : $x_{ij} = u_{ij}$.

De même, tout arc ij entrant est non rentable, et n'est donc pas utilisé : $x_{ij} = 0$.

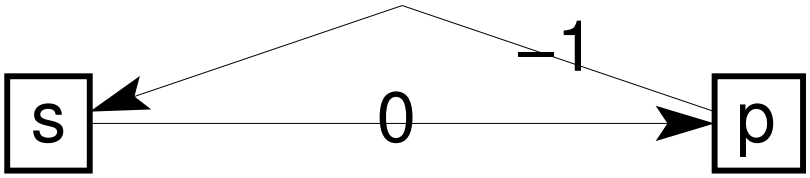
Conclusion : Le volume du flot F est égal à la capacité de la coupe C :

$$|F| = \sum_{ij \text{ sortant}} x_{ij} - \sum_{ij \text{ entrant}} x_{ij} = \sum_{ij \text{ sortant}} u_{ij} = |C| .$$

□

REMARQUE. Il y a quelques boulons à serrer.

(1) Le simplexe pourrait terminer en indiquant que le problème est non borné : *i.e.* il existe des flots de volume aussi grand que l'on veut :



Dans ce cas, il ne peut pas y avoir de coupe de capacité finie.

Donc le théorème reste valide.

(2) Le simplexe pourrait indiquer que le problème est non faisable.

En fait, non, puisque $x_{ij} = 0$, $\forall x_{ij}$ est solution.

(3) Si le flot max est de volume 0, il se pourrait que l'arbre ne contienne pas l'arc ps. Du coup, l'ensemble des sommets i tels que $y_i = 0$ ne serait pas forcément une coupe.

Un tel cas correspond en fait à une solution dégénérée qui est arborescente vis-à-vis de plusieurs arbres.

En fait, en faisant un pivot convenable, on peut toujours remettre ps dans l'arbre.

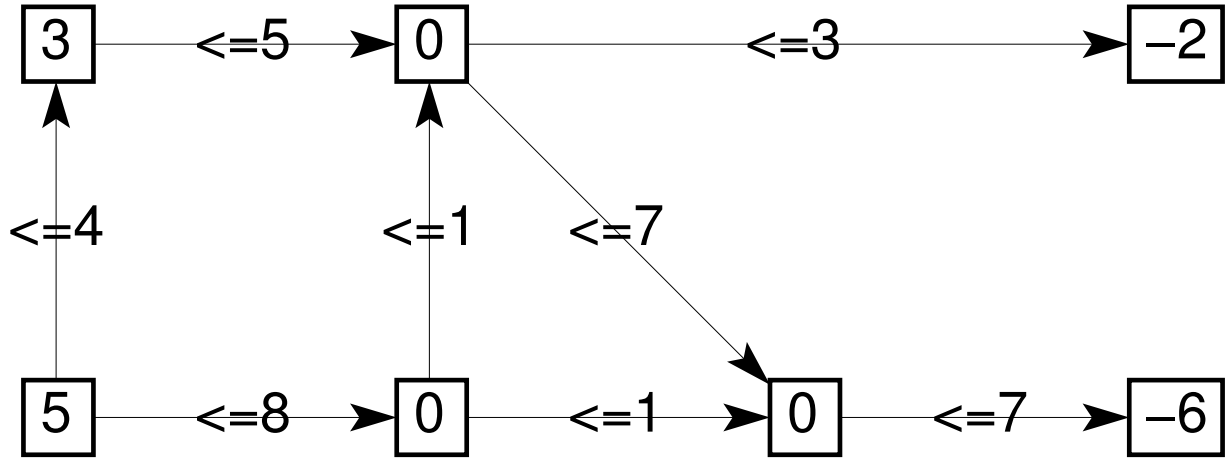
1.10.3. Applications. On a vu que les problèmes de flots était un cas particulier des problèmes de transport avec limites de capacités.

Quel est donc l'intérêt de considérer les problèmes de flots ?

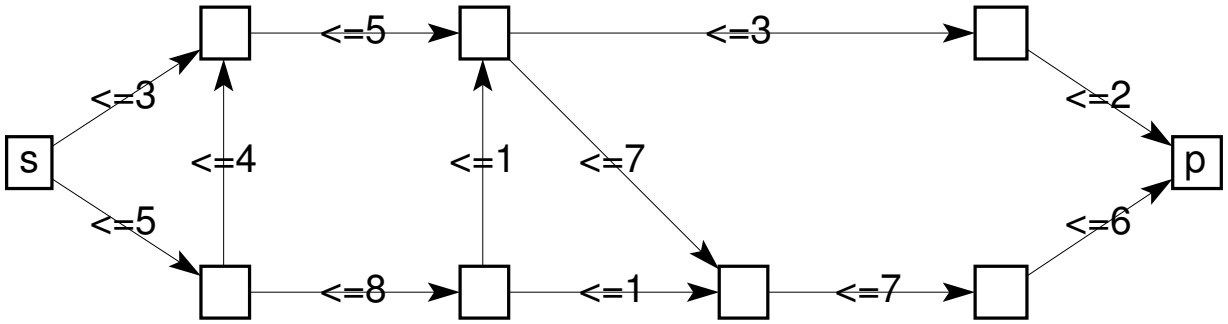
On a un algorithme (méthode du chemin augmentant) plus rapide que le simplexe.

1.10.3.1. *Trouver une solution faisable dans un problème de transport avec limites de capacités.*

EXEMPLE. On prend le problème de transport suivant, et on se demande s'il est faisable.



On peut le transformer en problème de flot, en oubliant les coûts, et en rajoutant une source, reliée convenablement aux producteurs, et un puits, relié convenablement aux consommateurs :



S'il existe un flot de volume 8, les arcs reliant s aux producteurs seront utilisés à pleine capacité, et de même pour les arcs reliant les consommateurs à t. Cela simule exactement les productions et consommations escomptées, donc le problème de réseau d'origine est faisable.

La réciproque est aussi clairement vraie : si le problème est faisable, alors il existe un flot de volume 8.

Dans notre cas, on en déduit que le problème n'est pas faisable. En effet, on peut trouver une coupe de capacité 7.

De manière générale, on peut toujours transformer un problème de transport avec limite de capacité en un problème de flot, de façon à déterminer s'il est faisable.

Cela donne un algorithme plus rapide que le simplexe pour la phase I de la résolution.

1.10.3.2. *Couplages dans les graphes bipartis.*

Exercice. Mettre sous forme de problème de flot le problème de rechercher un couplage max dans le graphe biparti suivant.

1.10.3.3. *Problème des mines à ciel ouvert.*

Problème. Des études géologiques ont permis de déterminer précisément la nature du sous-sol, et l'emplacement des gisements orifères à l'endroit ou l'on a décidé de creuser une mine à ciel ouvert. Certains gisements sont profonds, et il n'est pas clair qu'il soit rentable d'excaver tout le sol au-dessus pour y accéder.

Modèle : le sous-sol a été délimité en un certain nombre de blocs. Pour chaque bloc i, on connaît le coût C_i d'excavation, et le profit P_i que l'on peut escompter de son traitement.

Au final, on associe à chaque bloc i la quantité $w_i = C_i - P_i$. Si l'on ne considère pas les autres blocs, il est rentable de creuser i si et seulement si $w_i < 0$.

On veut déterminer quels blocs on doit creuser pour maximiser le profit total $-\sum_i w_i$ (ou autrement dit minimiser $\sum_i w_i$).

Maintenant, il y a des contraintes supplémentaires : si un bloc i est sous un bloc j, on ne peux pas creuser i sans creuser j !

On introduit un ordre partiel, de sorte que $i < j$ si pour creuser i on doit creuser j.

Comme on le verra, la forme des blocs, et le type d'ordre partiel n'est pas relevant.

EXEMPLE. On considère le sous-sol suivant :

Comment modéliser notre problème sous forme de problème de flot max ?

La modélisation des contraintes de précédences et un peu astucieuse !

On introduit le réseau suivant :

C'est la remarque suivante qui va faire marcher la machine :

REMARQUE. Soit C une coupe.

S'il existe deux blocs $i < j$, avec $i \in C$ et $j \in C$, alors C est de capacité infinie.

La réciproque est vraie.

Les coupes de capacité finie sont en correspondance avec les coupes respectant les contraintes.

Maintenant, on peut vérifier que la capacité d'une coupe finie vaut exactement

$$\sum_{i \in C,\ i \text{ bloc non rentable}} w_i - \sum_{i \notin C,\ i \text{ bloc rentable}} w_i.$$

Quitte à rajouter le terme constant $\sum_{i,\ i \text{ bloc}} w_i$, on est en train de calculer le profit lorsque l'on enlève les blocs i avec $i \in C$.

Résumé : Soit I un ensemble de blocs, et C la coupe $\{s\} \cup I$.

- Si I ne satisfait pas les contraintes, la capacité de C est infinie.
- Si I satisfait les contraintes, la capacité de C est l'opposé du profit.

Maximiser le profit revient à trouver une coupe min.

REMARQUE. En termes pédants : on peut résoudre par un algorithme de flot le problème de trouver une section finale de poids minimal dans un ordre partiel.

1.10.3.4. *Dualités chaînes/antichaînes dans les ordres partiels ; théorème de Dilworth.*

PROBLEM 1.10.1. [**1**, p. 338] Problème des visites guidées. Une compagnie propose 7 visites guidées dans la journée, notées a, b, c, d, e, f, g, dont les horaires et durées sont fixées. Si une visite (par ex. a) termine suffisament avant une autre (par exemple c), le guide de la première visite peut enchaîner sur la deuxième ; on notera alors $a \to c$. En l'occurence, voici tous les enchaînements possibles :

$a \to c,\ a \to d,\ a \to f,\ a \to g,\ b \to c,\ b \to g,\ d \to g,\ e \to f,\ e \to g.$

- Combien de guides faut-il de guides au minimum dans cet exemple ?
- Comment trouver le nombre minimum de guides nécessaires dans le cas général ?

DÉFINITION. Soit $P = (E, <)$ un ordre partiel.

Une *chaîne* C de P est un ensemble de sommets de P deux à-deux comparables :

$$x \in C \text{ et } y \in C \;\Rightarrow\; x < y \text{ ou } y < x.$$

Une *antichaîne* A de P est un ensemble de sommets deux-à-deux incomparables.

Une *couverture en chaînes* de P est un ensemble $C_1, \ldots, C_k$ de chaînes, de sorte que tout sommet de P est dans une unique chaîne C_i.

Une *couverture en antichaînes* de P est un ensemble $A_1, \ldots, A_k$ d'antichaînes, de sorte que tout sommet de P est dans une unique antichaîne A_i.

EXERCICE. Trouver dans l'ordre partiel P précédent :

(1) Une chaîne de taille maximale

(2) Une antichaîne de taille maximale

(3) Une couverture en chaînes de P de taille minimale

(4) Une couverture en antichaînes de P de taille minimale

Que remarquez vous ?

Y-aurait-il un théorème min-max reliant la taille de la plus grande chaîne et la taille de la plus petite couverture en antichaînes ? Et un autre reliant la taille de la plus grande antichaîne et celle de la plus petite couverture en chaînes ?

EXERCICE. Soit P un ordre partiel quelconque.

(1) Soit C une chaîne de P et $A_1, \ldots, A_k$ une couverture de P en antichaînes.
Montrer que $|C| \leq k$.

(2) Soit A une antichaîne de P et $C_1, \ldots, C_k$ une couverture de P en chaînes.
Montrer que $|A| \leq k$.

PROPOSITION. *Soit P un ordre partiel. La taille de la plus grande chaîne de P est égale à la taille de la plus petite couverture en antichaînes de P.*

EXERCICE. Prouvez-le !

Le théorème dans l'autre sens est plus difficile et bien plus profond. Il n'y a pas de construction élémentaire de l'antichaîne et de la couverture en chaîne idoine. On va en fait se ramener à la programation linéaire (surprise).

THÉORÈME. *(Dilworth) Soit P un ordre partiel. La taille de la plus grande antichaîne de P est égale à la taille de la plus petite couverture en chaînes de P.*

DÉMONSTRATION. On note n le nombre de sommets de P.

Choisir une couverture en chaîne de P est équivalent à sélectionner un certain nombre d'arcs dans P, de sorte que chaque sommet ait au plus un arc sortant de sélectionné, et un arc rentrant de sélectionné.

Remarque : s'il y a k chaînes, il y a $n-k$ arcs sélectionnés.

Cela ressemble à un problème de couplage maximal dans un graphe biparti.

On construit un graphe biparti B dans lequel chaque sommet x de P est dupliqué en $(x,1)$ et $(x,2)$.

Chaque fois que $x < y$ dans P, on relie $(x,1)$ et $(y,2)$.

Qu'est-ce qu'un couplage dans B ?

Un ensemble d'arcs de P vérifiant exactement les conditions voulues.

Une couverture de P en k chaînes correspond à un couplage de B de taille $n-k$.

Prenons une couverture de P de taille k minimale.

Cela donne un couplage de taille max $n-k$ de B.

Le théorème min-max pour les graphes bipartis indique qu'il y a une couverture de B de même taille : $n-k$ sommets de B qui touchent tous les arcs.

Dans P cela correspond à au plus $n-k$ sommets qui touchent tous les arcs.

Soit A l'ensemble des sommets restants qui est de taille au moins k.

Il ne peut pas y avoir d'arcs entre deux sommets de A.

Conclusion : A est une antichaîne de taille au moins k. □

EXERCICE. Suivez le déroulement de la preuve sur l'ordre partiel précédent

1.10.4. La méthode du chemin augmentant. Rechercher un couplage max dans un graphe biparti est un problème très classique.

Il existe en fait des techniques plus rapides que les algorithmes de flots pour cela.

L'une d'entre elles est la méthode du chemin augmentant.

1.10.4.1. *Cas général des graphes simples.*

EXEMPLE. Couplage de taille maximale dans le chemin P_6.

On peut construire un couplage de taille maximal progressivement à partir d'un couplage vide en rajoutant des arêtes une à une. Mais si on n'est pas parti correctement, on risque d'être bloqué avec un couplage qui est maximal, alors qu'il n'est pas de taille maximale. Il faut appliquer une transformation au couplage pour pouvoir l'agrandir.

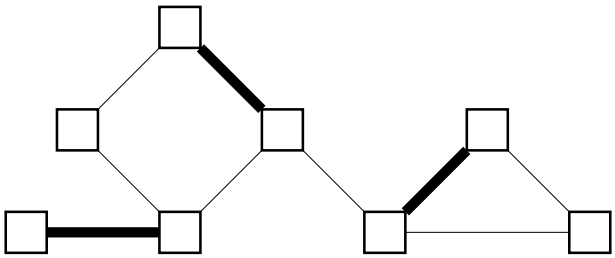
DÉFINITION. Soit M un couplage dans un graphe G.

Un sommet M-*saturé* (ou saturé) est un sommet touchant une arête du couplage.

Un chemin M-*alterné* de G est un chemin de G qui utilise alternativement des arêtes dans M et hors de M.

Un chemin M-*augmentant* est un chemin M-alterné commençant et finissant par des sommets non M-saturés.

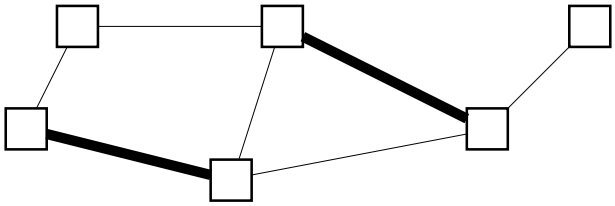
EXEMPLE. Regardons ce que cela donne avec le couplage suivant :



REMARQUE. Si M est un couplage, et C est un chemin M-augmentant, alors on peut construire un couplage M' strictement plus grand en utilisant C.

Cas particulier : lorsque le chemin M-augmentant consiste d'une seule arête reliant deux sommets non saturés, on est ramené au rajout d'une arête au couplage M'.

EXERCICE. Soit G le graphe

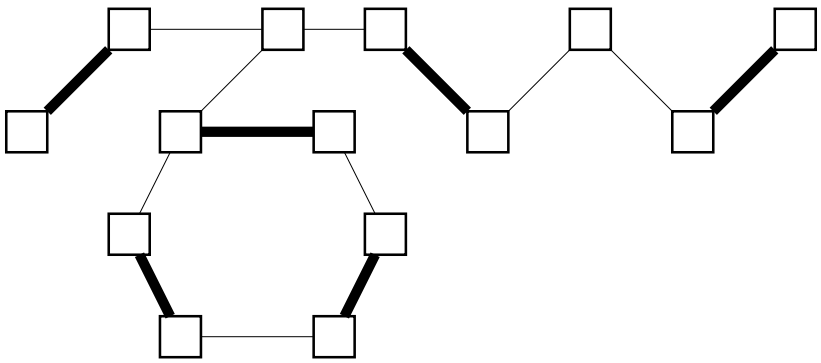


Le couplage M dessiné est maximal, mais pas de taille maximale. Trouvez un chemin M-augmentant et construisez le couplage M' correspondant.

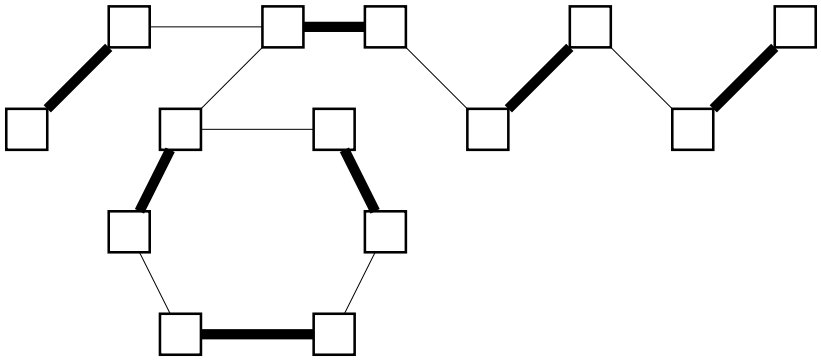
Est-ce que cette opération de chemin augmentant est suffisante ?

THÉORÈME. *(Berge 1957) Un couplage M est de taille maximale si et seulement s'il n'y a pas de chemin M-augmentant.*

DÉMONSTRATION. Soit M un couplage, comme dans l'exemple suivant :



On suppose qu'il existe un couplage M' de taille strictement plus grande.



On va construire un chemin M-augmentant.

On considère le graphe H obtenu par réunion de M et M'.

Les sommets de H sont de degré au plus 2.

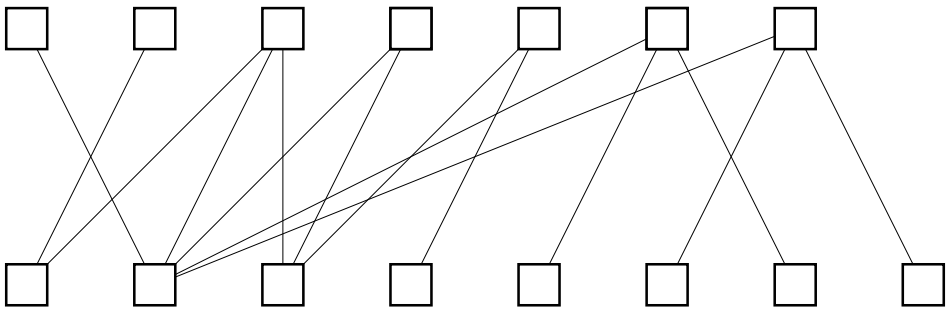
Ses composantes connexes sont donc des doubles arêtes ou des cycles et chemins dont les arêtes alternent entre M et M'. Dans un cycle ou une double arête il y a autant d'arêtes de M que de M'. Il y aura donc forcément un chemin qui contiendra une arête de plus de M' que de M. Ce chemin est un chemin M-augmentant. □

On déduit de ce théorème un algorithme (ou plutôt une méthode) :

ALGORITHME 1.10.2. *Recherche d'un couplage maximal dans un graphe :*

(1) Partir d'un couplage M quelconque (par exemple le couplage vide) ;
(2) Chercher un chemin M-augmentant ;
(3) S'il n'y en a pas, renvoyer M qui est maximal ; sinon réitérer avec $M := M'$.

EXERCICE. Appliquez cette méthode pour trouver un couplage maximal du graphe



Pour un graphe quelconque, l'étape 2 peut être difficile !

Par contre, pour un graphe biparti, il y a un algorithme.

1.10.4.2. *Recherche d'un chemin augmentant pour un couplage d'un graphe biparti.*

EXEMPLE. On va rechercher un couplage maximal dans le graphe biparti précédent, en montrant comment on peut trouver de manière systématique un chemin M-augmentant.

ALGORITHME 1.10.3. *Soit M un couplage d'un graphe biparti B entre les parties X et Y.*

Cet algorithme renvoie soit un chemin M-augmentant, soit une couverture du graphe de taille $|M|$, ce qui indiquera que le couplage M est de taille maximale.

Soit U l'ensemble des sommets de X non touchés par M, et V l'ensemble des sommets de Y non touchés par M.

On va rechercher par un parcours en largeur les chemins M-alternés allant de U à V.

S (resp. T) va représenter l'ensemble des sommets x de X (resp. Y) pour lesquels on aura déjà trouvé un chemin M-alterné allant de U à x.

Initialisation :: $S := U$, $T := \emptyset$;

Itération ::

Cas 1 :: *Il y a une arête reliant un sommet de S à un sommet de V. Cela donne un chemin M-augmentant que l'on renvoie*

Cas 2 :: *Il y a une arête reliant un sommet x de S à un sommet y de $Y - V$ avec $xy \notin /M$. Ce sommet est relié par une arête du couplage à un sommet w de X. Comme y et w sont reliés à U par un chemin M-alterné, on rajoute y à T et w à S.*

Cas 3 :: *$T \cup (X - S)$ est une couverture de taille $|M|$ du graphe (vérifiez le !) que l'on renvoie.*

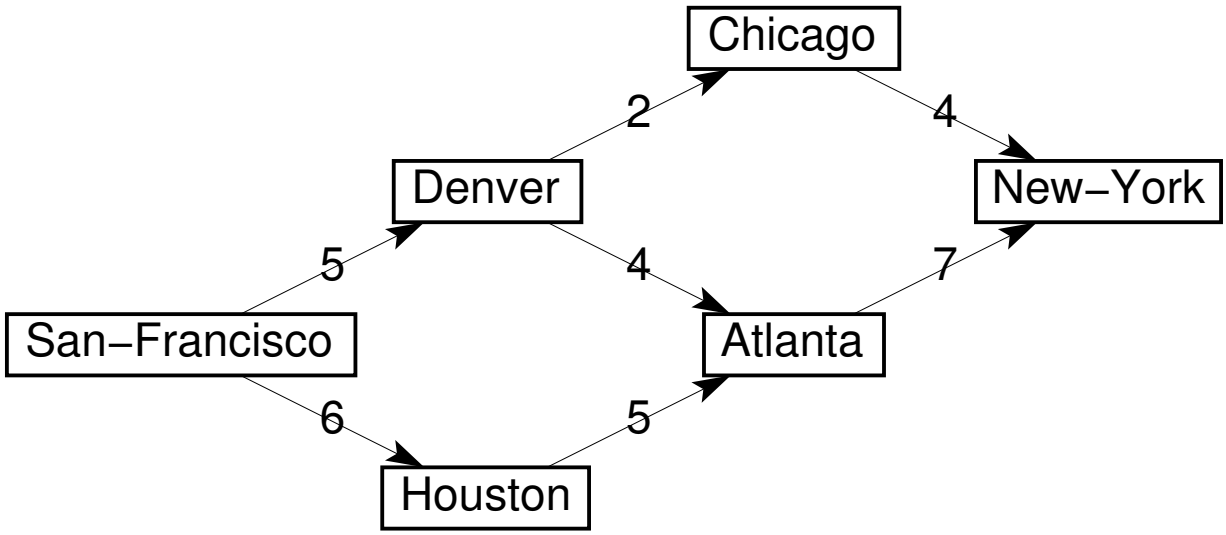
EXERCICE 19. Appliquer l'algorithme sur un graphe biparti de votre choix

Démontrer que $T \cup (X - S)$ est bien une couverture de taille $|M|$ du graphe.

Donner la preuve

1.10.5. Algorithme de Ford-Fulkerson. La méthode du chemin augmentant se généralise à la recherche de couplage max dans un graphe biparti valué, et en fait aussi aux problèmes de flots max. On ne va regarder son fonctionnement que sur un exemple, et on renvoie à [**1**, p. 369] pour les détails.

EXEMPLE. On veut transporter le plus grand nombre possible de voyageurs de San-Francisco à New-York, sachant qu'il ne reste que quelques places dans les avions entre les villes suivantes :



DÉFINITION. Soit R un réseau, et F un flot donné dans ce réseau.

Un chemin allant de la source s au puits p est *F-augmentant* si pour chaque arête ij du chemin on a :
– $x_{ij} < u_{ij}$ si l'arc ij est dans le même sens que dans le réseau
– $x_{ij} > 0$ si l'arc ij est dans le sens inverse du réseau.

À partir d'un chemin F-augmentant, on peut construire un nouveau flot F' qui sera de volume strictement plus gros.

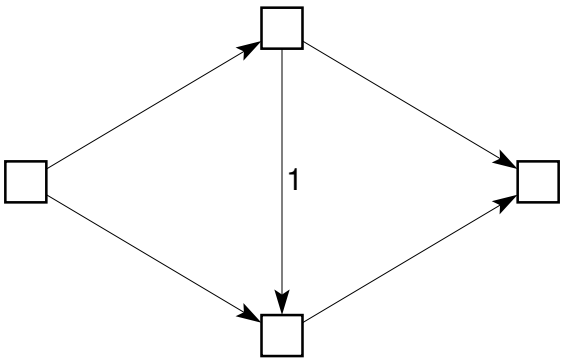
Le principe de l'algorithme de Ford-Fulkerson est de partir d'un flot F quelconque, et de l'améliorer itérativement en recherchant des chemins F-augmentant.

À chaque étape, la recherche d'un chemin F-augmentant se fait par un parcours en profondeur, de manière similaire à la recherche d'un chemin M-augmentant dans un graphe biparti. Si cette recherche échoue, elle dévoile une coupe de capacité égale au flot, ce qui donne un certificat d'optimalité du flot.

REMARQUE. On peut toujours initialiser l'algorithme avec un flot nul.

Si toutes les capacités sont entières et finies, chaque itération augmente le flot d'au moins 1. Cet algorithme ne peut donc pas cycler, et il termine en un nombre fini d'étapes.

Avec une mauvaise stratégie, et des capacités infinies ou non-entières, l'algorithme peut ne pas terminer.



Avec une stratégie convenable, cet algorithme est en fait polynomial, en $O\left(n^3\right)$, même si les capacités sont infinies ou non entières.

Pour les réseaux avec peu d'arcs, il y a des algorithmes plus compliqués qui permettent d'obtenir d'encore meilleurs bornes. Cf. [**1**, p. 369] pour les détails.

1.11. Méthodes alternatives au simplexe

Pour conclure ce chapitre sur la programmation linéaire, nous présentons rapidement quelques méthodes alternatives qui ont été développées pour résoudre les problèmes de programmation linéaire généraux. Nous nous contentons d'évoquer leur principe, leurs avantages et inconvénients, et donnons des références pour ceux qui voudraient en savoir plus.

1.11.1. Méthode de l'ellipsoïde.

Principe: On commence par utiliser la dualité pour se ramener à la recherche d'une solution faisable d'un système d'inéquations linéaires. On peut en fait se ramener par une perturbation convenable à la recherche d'une solution faisable d'un système d'inéquations linéaires strictes !

Si un tel système est faisable, le volume de l'ensemble des solutions peut alors être minoré par une quantité V qui dépends de la dimension de l'espace, et de la taille des coefficients dans le système linéaire.

On part d'un ellipsoïde E suffisamment gros pour contenir toutes les solutions faisables.

Si le centre de E est une solution faisable, on a terminé.

Sinon, on peut couper l'ellipsoïde en deux, et inclure ce demi-ellispoïde dans un ellipsoïde E' qui contient encore toutes les solutions faisables. On réitère avec $E := E'$.

À chaque itération, on a $V(E') < \alpha V(E)$, où $\alpha < 1$ est une constante qui ne dépends que de la dimension ; donc le volume de E décroît exponentiellement. Au bout d'un petit nombre d'itérations, si l'on a pas obtenu de solution faisable, on a $V(E) < V$, ce qui prouve que la système n'a pas de solution faisable.

Avantages:

- Algorithme polynomial

Inconvénients:

- Plus lent *en pratique* que l'algorithme du simplexe
- Ne donne pas certains résultats théoriques (dualité, géométrie des polyèdres)

Référence: [**1**, p. 443]

1.11.2. Méthode des points intérieurs.

Principe: Cette méthode est l'antithèse exacte du simplexe.

Le principe du simplexe est de ne considérer que des solutions basiques qui sont à la frontière du polyèdre, et d'utiliser de l'algèbre linéaire pour itérer parmi ces solutions.

Ici au contraire, la méthode ne considère que des solutions strictement à l'intérieur du polyèdre. L'idée est d'approximer les inégalités du système linéaire, qui forment fondamentalement des discontinuités, en barrières de potentiel, qui sont elles continues. Du coup, on peut utiliser les techniques d'optimisation non-linéaire continue, comme les méthodes de gradients.

Avantages:

- Rivalise avec le simplexe *en pratique*, suivant le type de problème rencontrés

Inconvénients:

- Ne donne pas de résultats théoriques (dualité, géométrie des polyèdres)

Références: [**2**]

Bibliographie

[1] V. Chvatal. Linear Programming.

[2] R. Vanderbie. Linear Programming ; Foundations and Extensions. http://www.princeton.edu/~rvdb/LPbook/index.html

[3] Linear Programming FAQ http://rutcor.rutgers.edu/~mnk/lp-faq.html