Recherche Opérationnelle

Programmation Linéaire

Benoît Darties

Université de Bourgogne - LE2I [1]

1. Supports de cours dispensés à l'ESIREM - Université de Bourgogne. La totalité de ce document a été rédigée uniquement à partir des connaissances de son auteur, et en utilisant un matériel personnel. Réutilisation partielle ou complète de ce document soumise à approbation l'auteur

On va manger des chips. T'entends ? DES CHIPS

Une alimentation équilibrée, le secret d'une scolarité réussie !

Pour qu'un étudiant réussisse sa scolarité, il a besoin d'un apport régulier en bières et en chips par semaine. Cet apport diffère selon sa formation :

- Pour un étudiant en SQR : 25 bières, 100 grammes de chips par semaine
- Pour un étudiant en SE : 12 bières et 230 grammes de chips par semaine

Etudiant qui réussit = argent !

Si un étudiant est correctement alimenté au long de sa formation, il réussira cette dernière et trouvera un travail. Il rapportera de la taxe professionnelle :

- Pour un étudiant en SQR : 150 euro en moyenne / an
- Pour un étudiant en SE : 110 euro en moyenne / an

Bien nourrir son cheptel, pour augmenter sa rentabilité

L'école dispose de 1000 bières et 7500 grammes de chips / semaine. Comment les répartir entre étudiants de SE et SQR de la façon la plus rentable possible ?

On va manger des chips. T'entends ? DES CHIPS

Une alimentation équilibrée, le secret d'une scolarité réussie !

Pour qu'un étudiant réussisse sa scolarité, il a besoin d'un apport régulier en bières et en chips par semaine. Cet apport diffère selon sa formation :

- Pour un étudiant en SQR : 25 bières, 100 grammes de chips par semaine
- Pour un étudiant en SE : 12 bières et 230 grammes de chips par semaine

Etudiant qui réussit = argent !

Si un étudiant est correctement alimenté au long de sa formation, il réussira cette dernière et trouvera un travail. Il rapportera de la taxe professionnelle :

- Pour un étudiant en SQR : 150 euro en moyenne / an
- Pour un étudiant en SE : 110 euro en moyenne / an

Bien nourrir son cheptel, pour augmenter sa rentabilité

L'école dispose de 1000 bières et 7500 grammes de chips / semaine. Comment les répartir entre étudiants de SE et SQR de la façon la plus rentable possible ?

On va manger des chips. T'entends ? DES CHIPS

Une alimentation équilibrée, le secret d'une scolarité réussie !

Pour qu'un étudiant réussisse sa scolarité, il a besoin d'un apport régulier en bières et en chips par semaine. Cet apport diffère selon sa formation :

- Pour un étudiant en SQR : 25 bières, 100 grammes de chips par semaine
- Pour un étudiant en SE : 12 bières et 230 grammes de chips par semaine

Etudiant qui réussit = argent !

Si un étudiant est correctement alimenté au long de sa formation, il réussira cette dernière et trouvera un travail. Il rapportera de la taxe professionnelle :

- Pour un étudiant en SQR : 150 euro en moyenne / an
- Pour un étudiant en SE : 110 euro en moyenne / an

Bien nourrir son cheptel, pour augmenter sa rentabilité

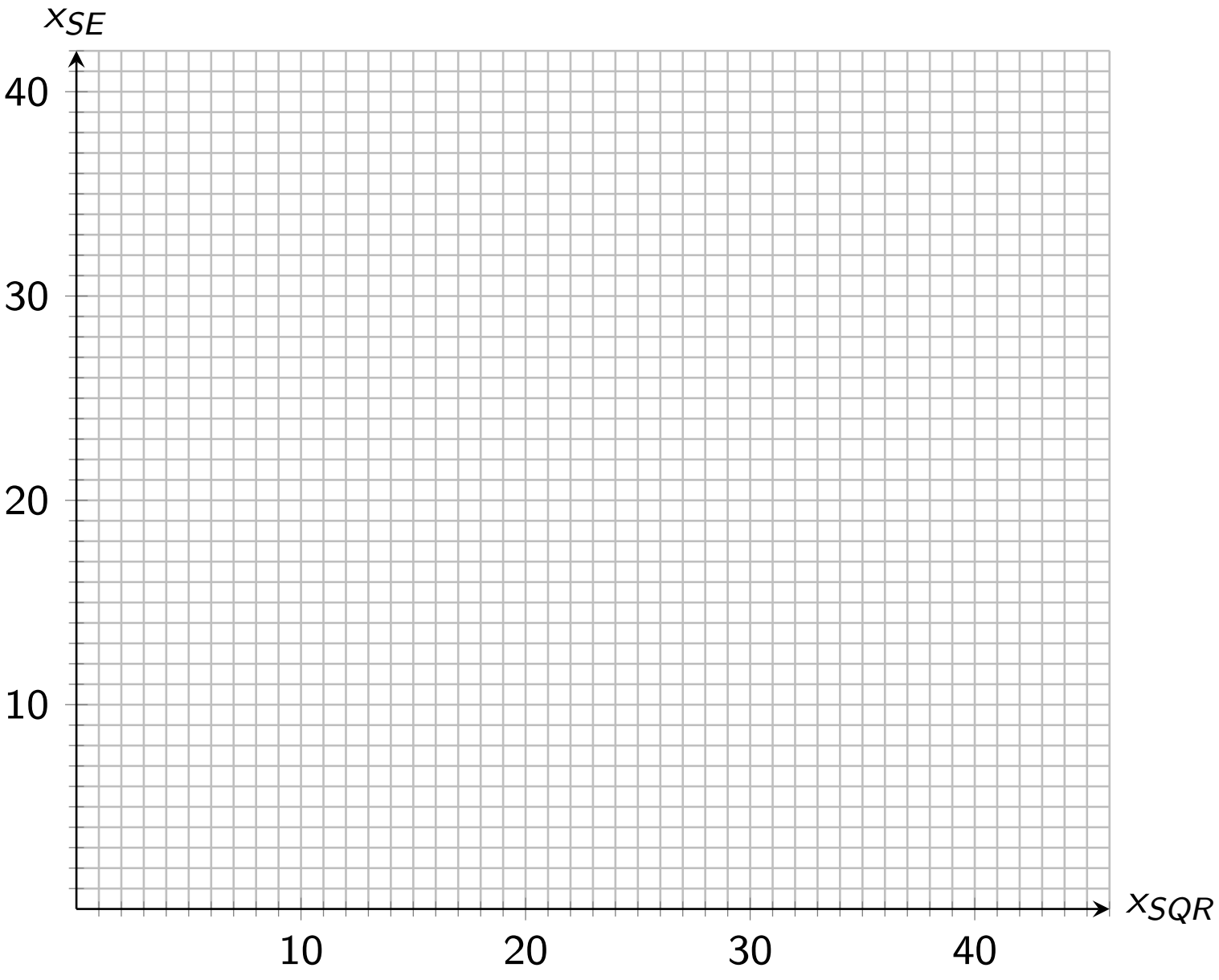
L'école dispose de 1000 bières et 7500 grammes de chips / semaine. Comment les répartir entre étudiants de SE et SQR de la façon la plus rentable possible ?

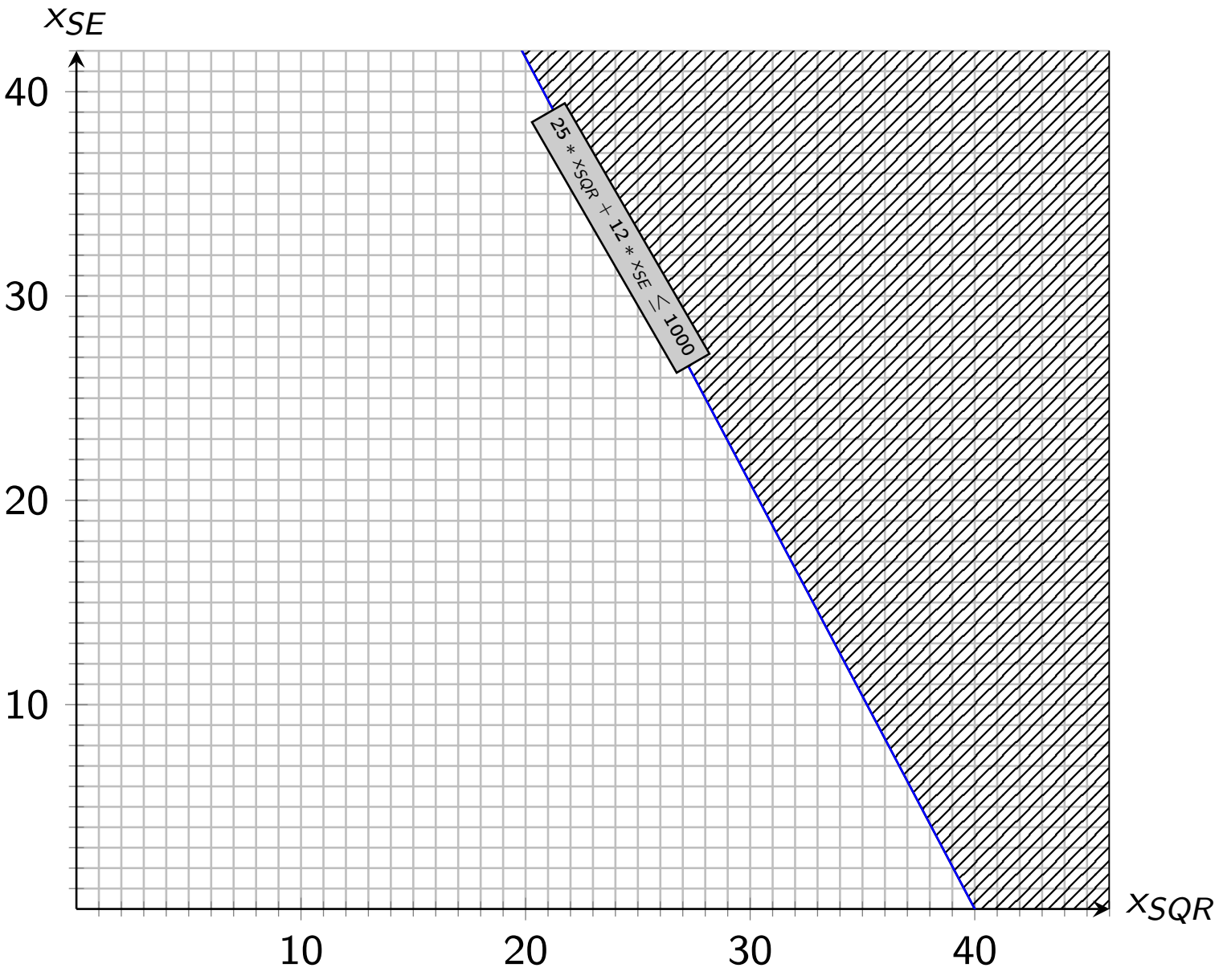
Problème sous la forme d'(in)équations linéaires

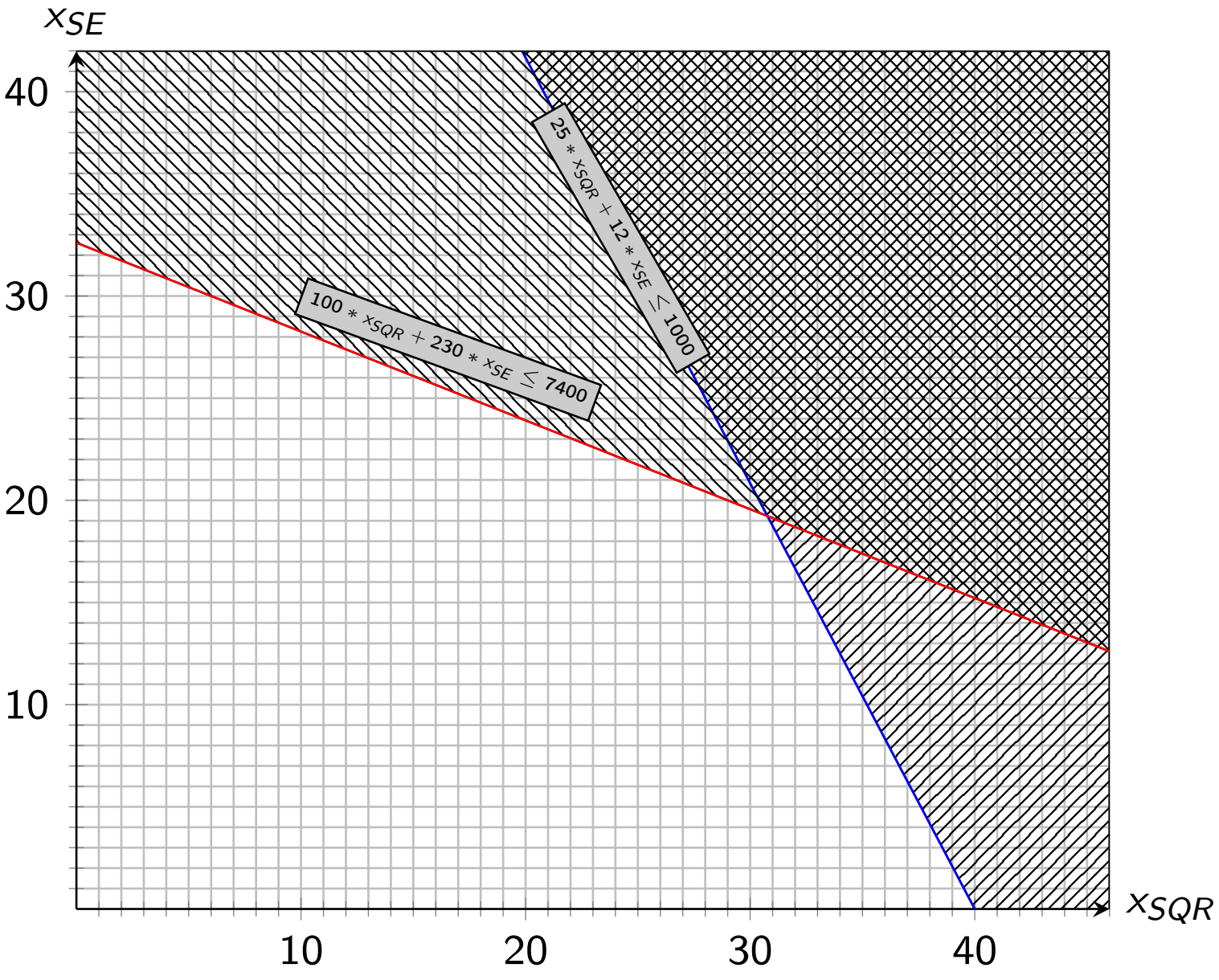
Ecrivons ce probleme sous la forme d'un programme linéaire :

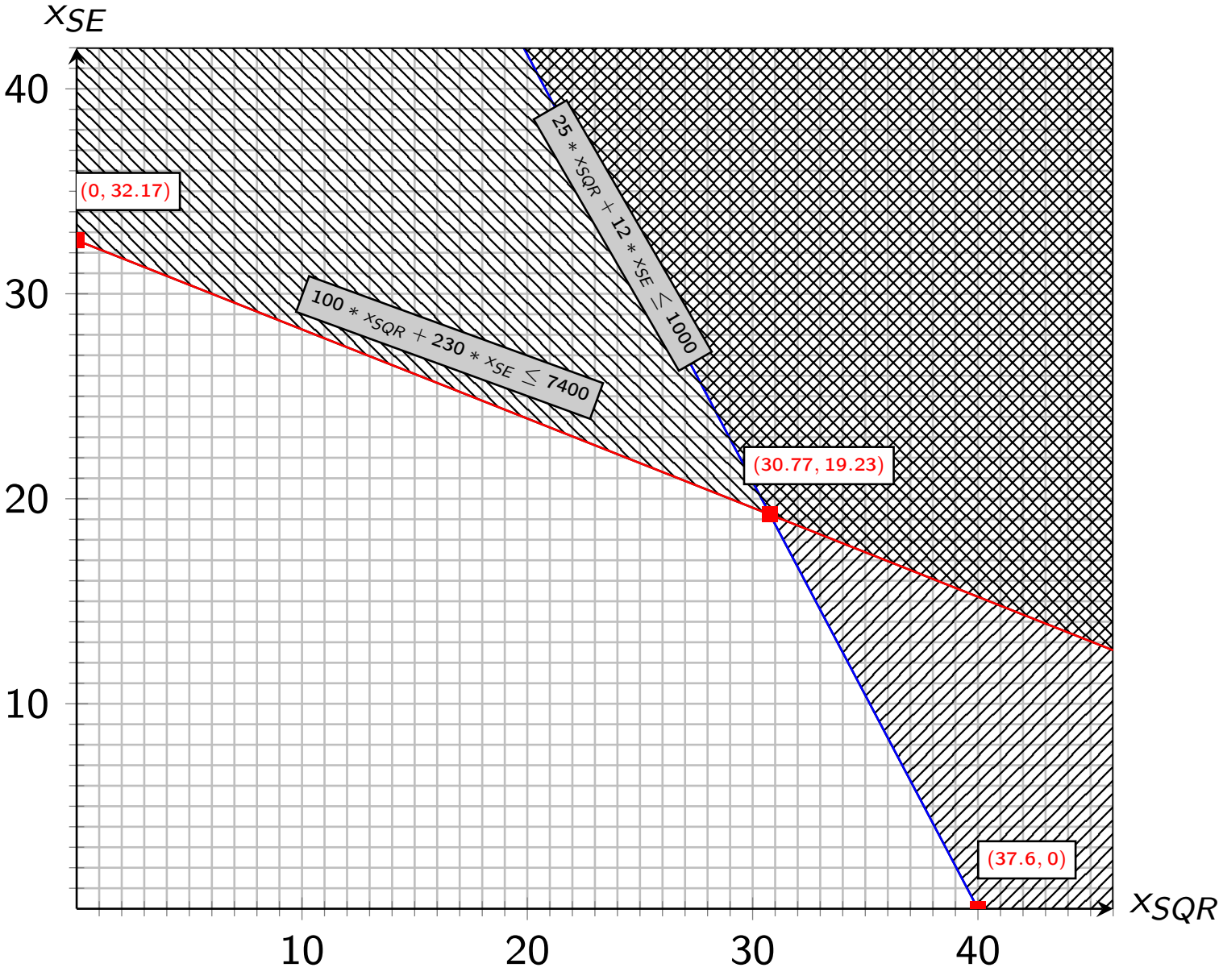
1. il nous faut des **variables de décision** :
   - x_{SQR} : nombre d'étudiants de SQR à nourrir
   - x_{SE} : nombre d'étudiants de SE à nourrir
2. il nous faut un **objectif** :
   - Maximiser $150 * x_{SQR} + 110 * x_{SE}$
3. il nous faut des **contraintes linéaires sur les variables** :
   - pour les bières : $25^* x_{SQR} + 12 * x_{SE} \leq 1000$
   - pour les chips : $100^* x_{SQR} + 230 * x_{SE} \leq 7500$

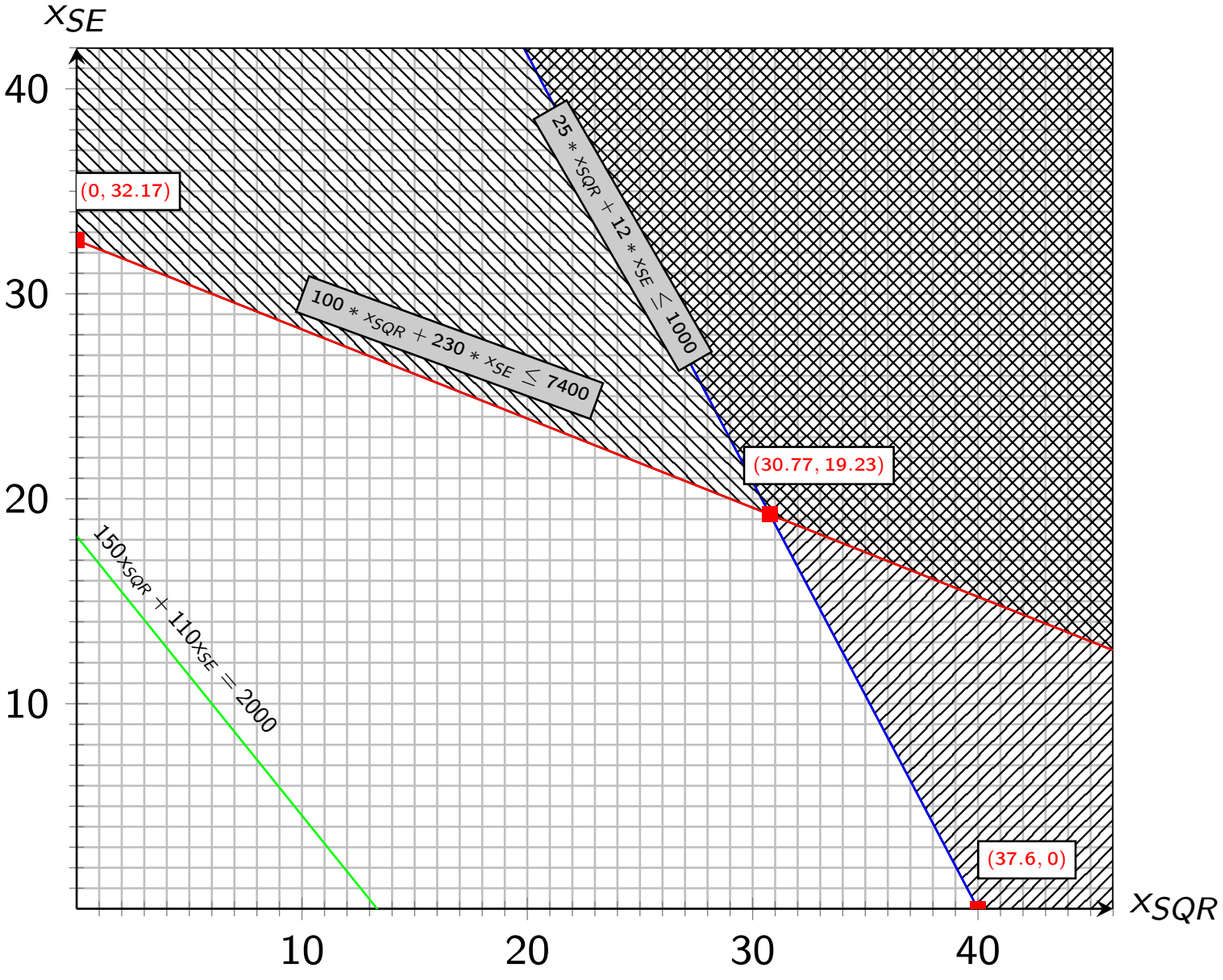
On obtient un problème général d'optimisation sous contraintes que l'on vient de formuler avec des (in)équations linéaires, qu'il ne reste *plus qu'à* résoudre...

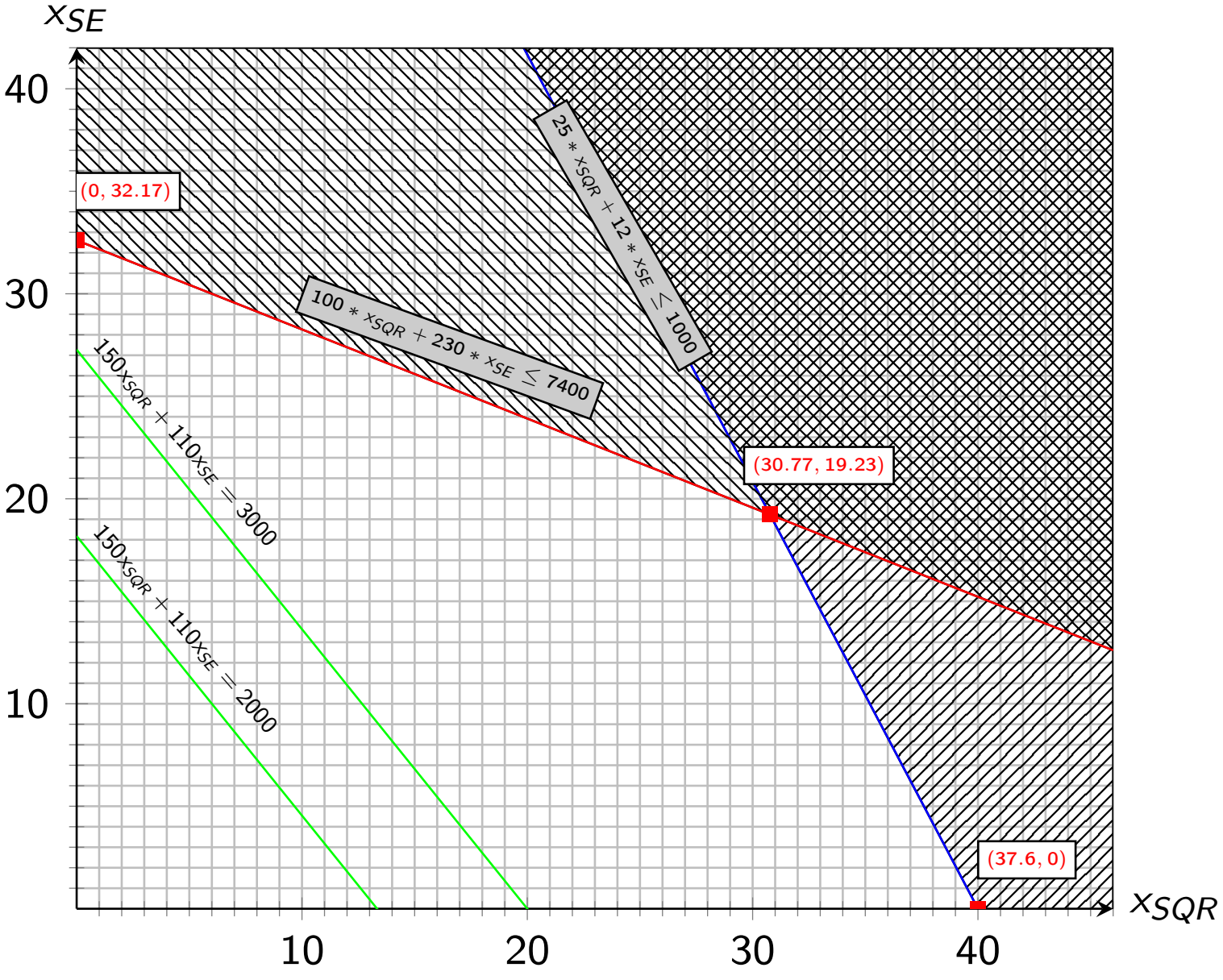

x_{SE}
40
30
20
10
10
20
30
40
x_{SQR}

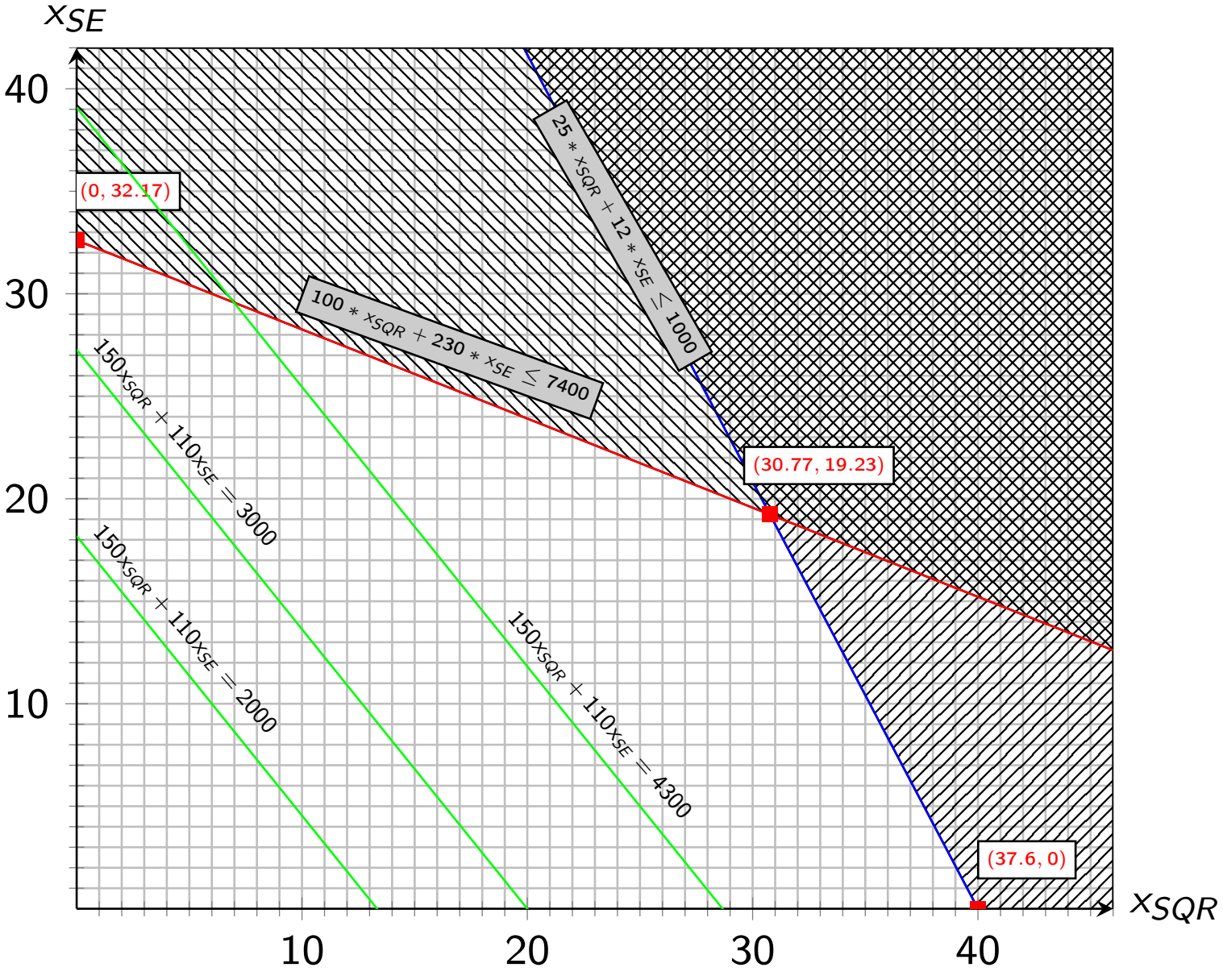

x_{SE}
40
30
20
10
$25 * x_{SQR} + 12 * x_{SE} \leq 1000$
10
20
30
40
x_{SQR}

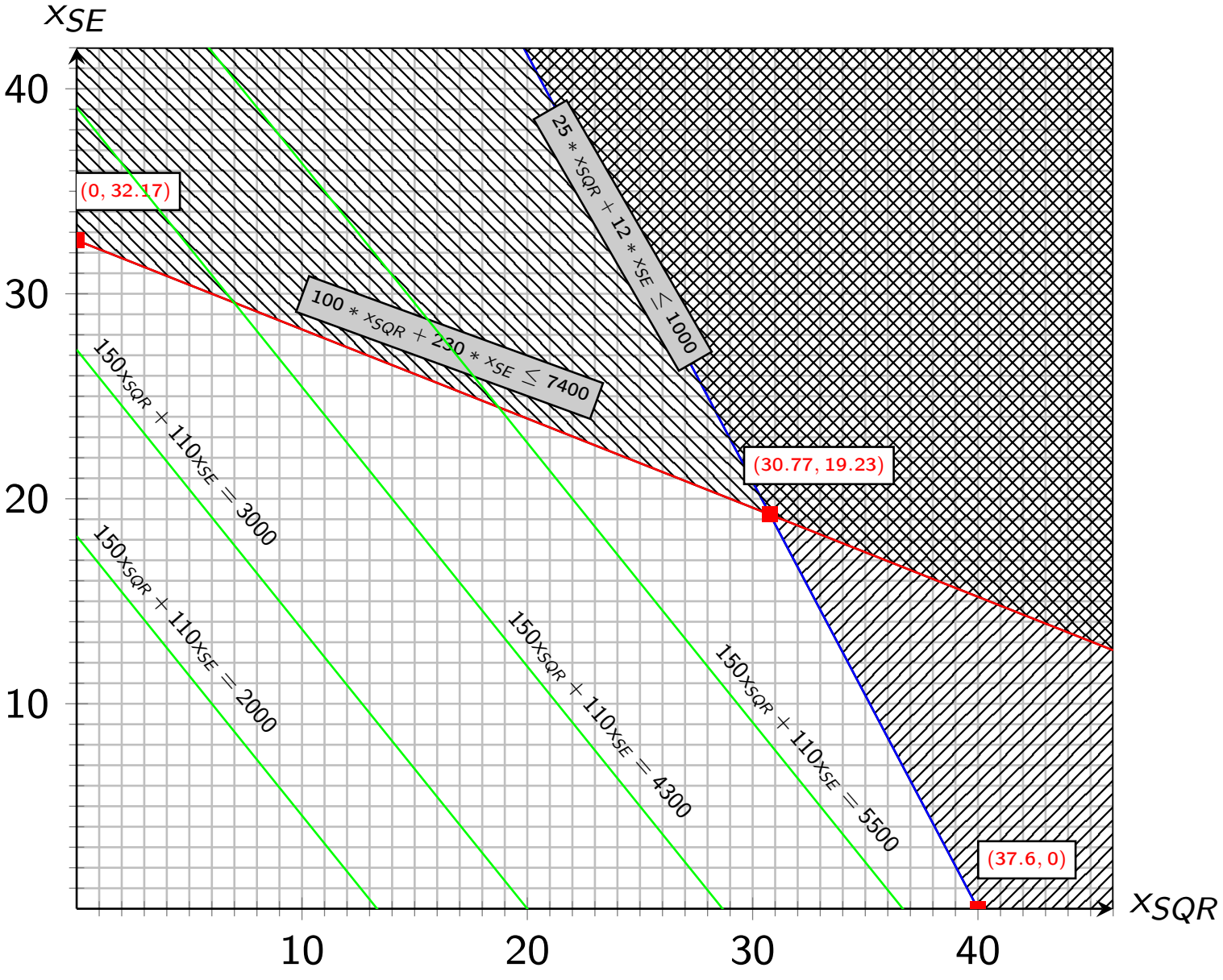

x_{SE}
40
30
20
10
10
20
30
40
x_{SQR}
$25 * x_{SQR} + 12 * x_{SE} \leq 1000$
$100 * x_{SQR} + 230 * x_{SE} \leq 7400$

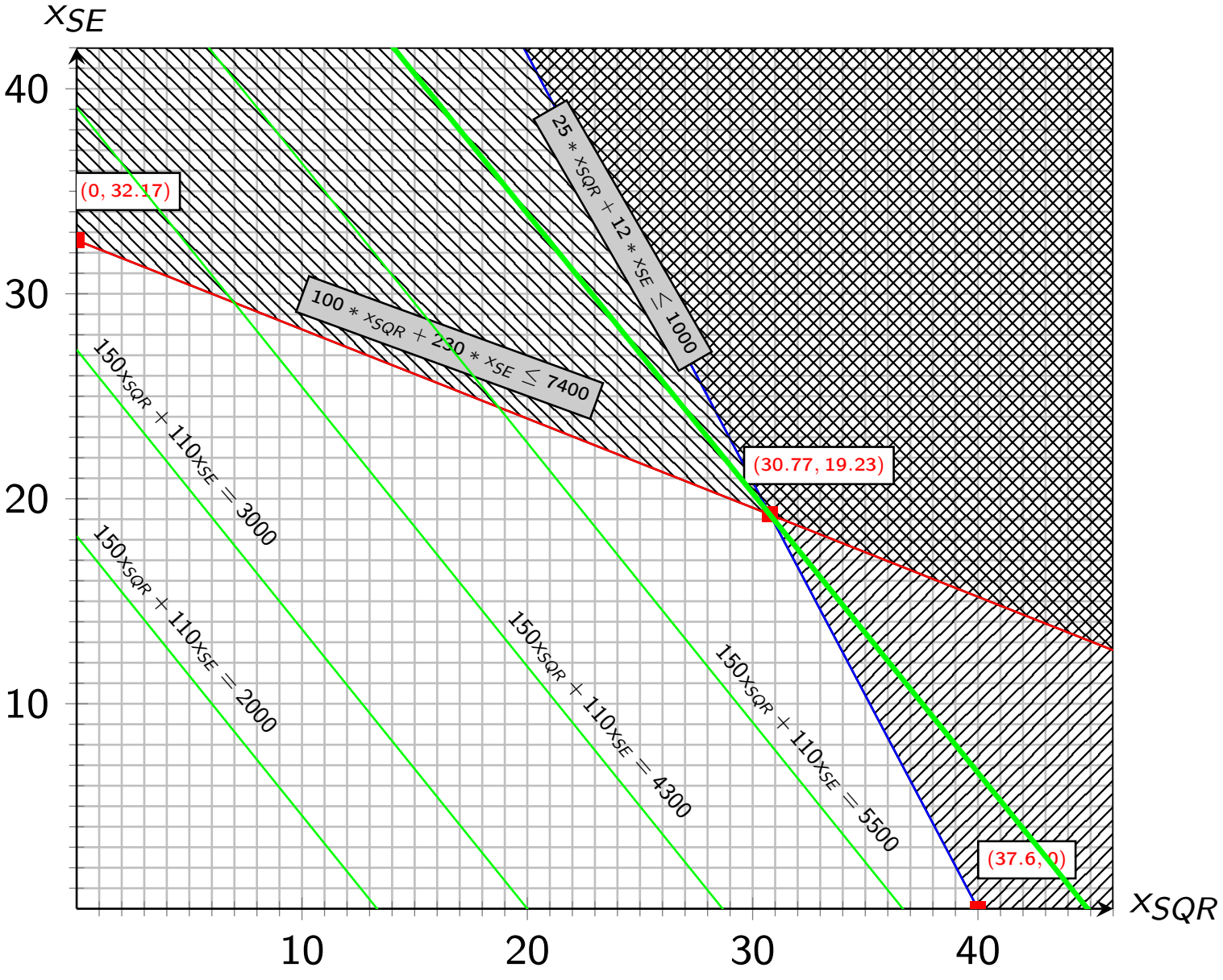

x_{SE}
40
30
20
10
(0, 32.17)
$100 * x_{SQR} + 230 * x_{SE} \leq 7400$
$25 * x_{SQR} + 12 * x_{SE} \leq 1000$
(30.77, 19.23)
(37.6, 0)
10
20
30
40
x_{SQR}


x_{SE}
40
30
20
10
(0, 32.17)
$100 * x_{SQR} + 230 * x_{SE} \leq 7400$
$25 * x_{SQR} + 12 * x_{SE} \leq 1000$
(30.77, 19.23)
$150x_{SQR} + 110x_{SE} = 2000$
(37.6, 0)
10
20
30
40
x_{SQR}


x_{SE}
40
30
20
10
(0, 32.17)
$100 * x_{SQR} + 230 * x_{SE} \leq 7400$
$25 * x_{SQR} + 12 * x_{SE} \leq 1000$
(30.77, 19.23)
(37.6, 0)
$150x_{SQR} + 110x_{SE} = 3000$
$150x_{SQR} + 110x_{SE} = 2000$
10
20
30
40
x_{SQR}


x_{SE}
40
30
20
10
(0, 32.17)
$25 * x_{SQR} + 12 * x_{SE} \leq 1000$
$100 * x_{SQR} + 230 * x_{SE} \leq 7400$
(30.77, 19.23)
$150x_{SQR} + 110x_{SE} = 3000$
$150x_{SQR} + 110x_{SE} = 2000$
$150x_{SQR} + 110x_{SE} = 4300$
(37.6, 0)
x_{SQR}
10
20
30
40

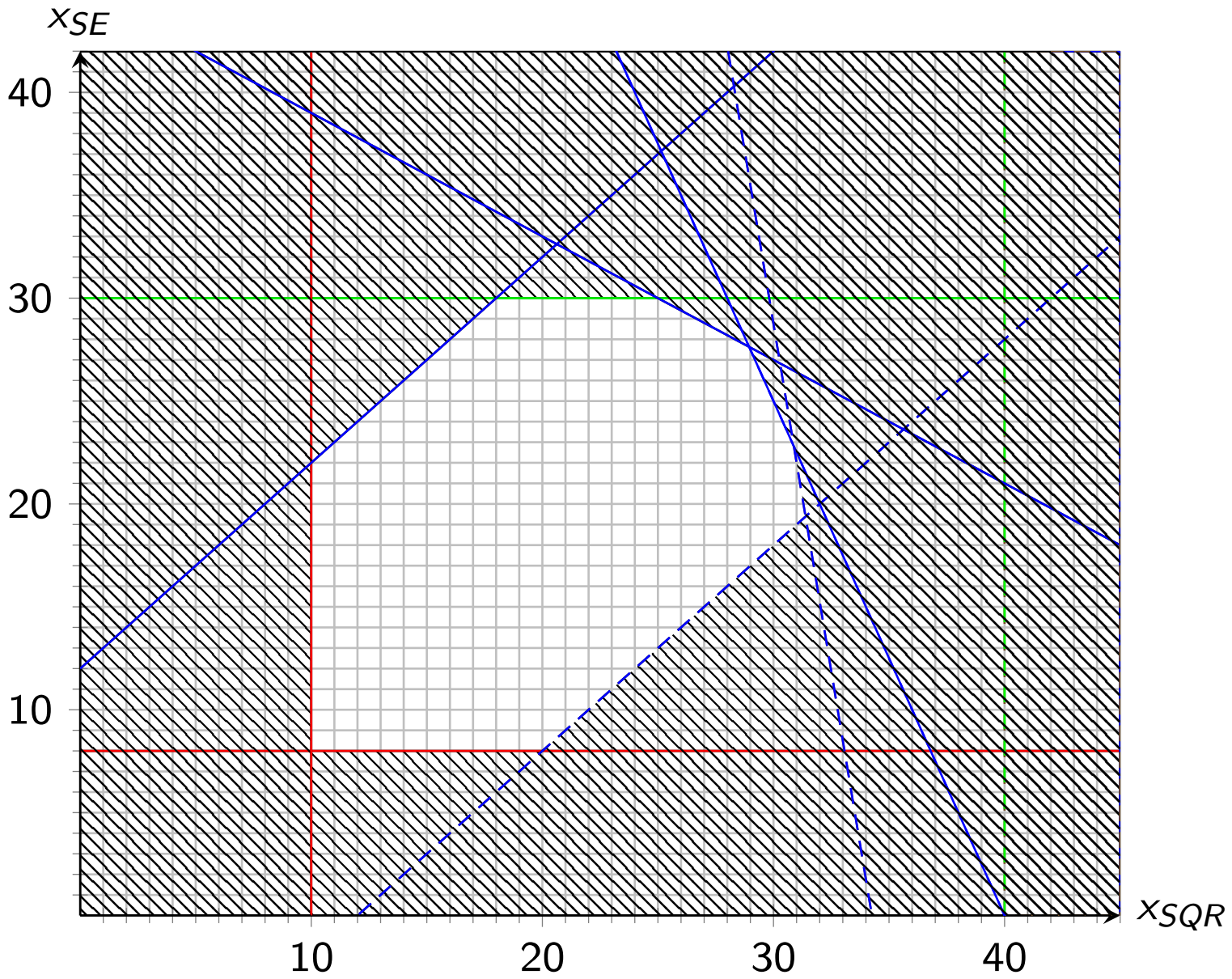

x_{SE}
40
30
20
10
(0, 32.17)
$100 * x_{SQR} + 230 * x_{SE} \leq 7400$
$25 * x_{SQR} + 12 * x_{SE} \leq 1000$
(30.77, 19.23)
(37.6, 0)
$150x_{SQR} + 110x_{SE} = 3000$
$150x_{SQR} + 110x_{SE} = 2000$
$150x_{SQR} + 110x_{SE} = 4300$
$150x_{SQR} + 110x_{SE} = 5500$
10
20
30
40
x_{SQR}

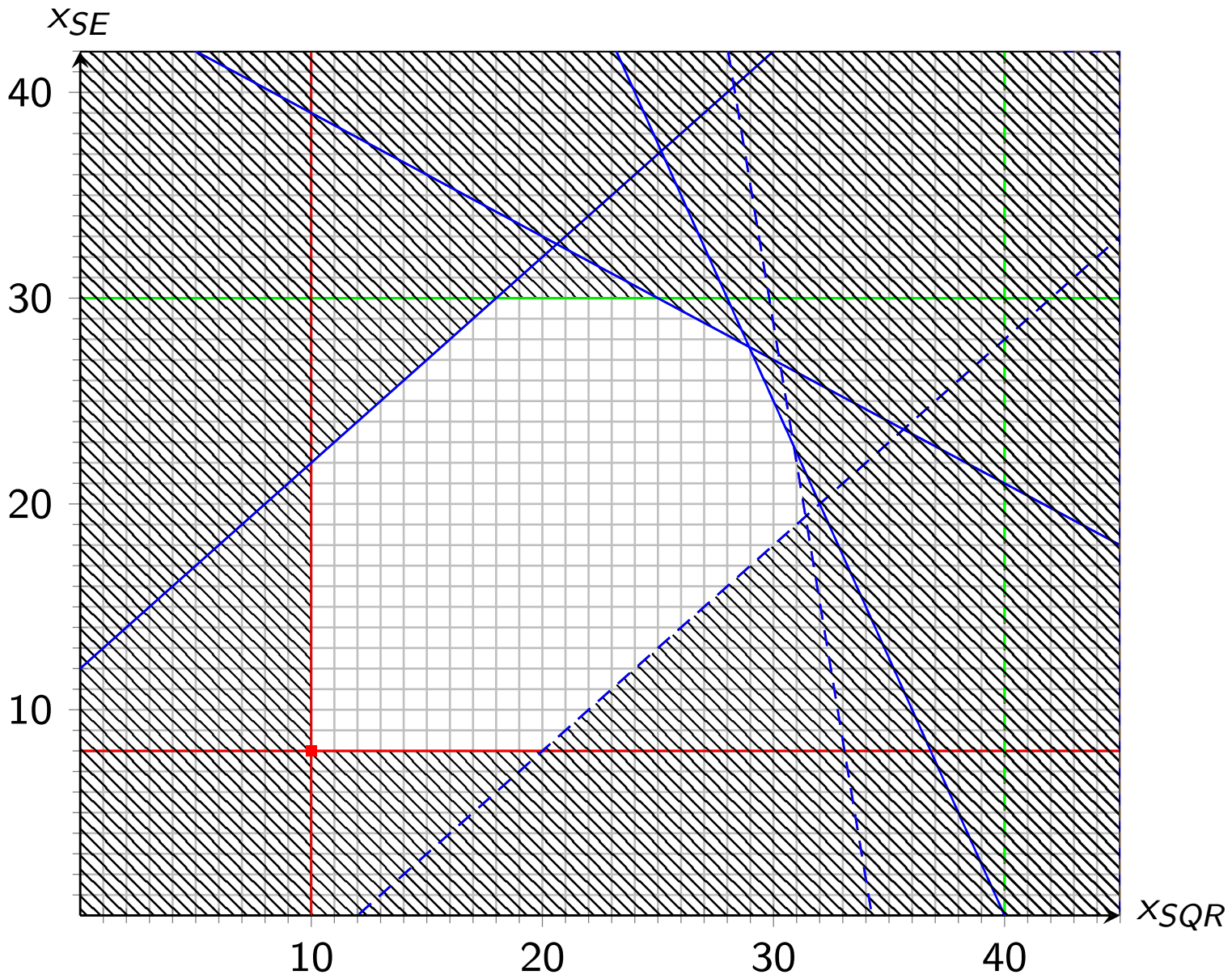

x_{SE}
40
30
20
10
(0, 32.17)
$25 * x_{SQR} + 12 * x_{SE} \leq 1000$
$100 * x_{SQR} + 230 * x_{SE} \leq 7400$
(30.77, 19.23)
$150x_{SQR} + 110x_{SE} = 3000$
$150x_{SQR} + 110x_{SE} = 2000$
$150x_{SQR} + 110x_{SE} = 4300$
$150x_{SQR} + 110x_{SE} = 5500$
(37.6, 0)
x_{SQR}
10
20
30
40

Avec quelques contraintes de plus

- L'école dispose de 800 bières, 9000 grammes de chips, 925 pizzas
- Un étudiant SQR : 20 bières, 120 grammes de chips, 27 pizzas
- Un étudiant SE : 8 bières, 200 grammes de chips, 4 pizzas
- Entre 10 et 40 étudiants max de SQR à former (contraintes du marché)
- Entre 8 et 30 étudiants max de SE à former (contraintes du marché)
- Un étudiant SQR rapporte 160 euro, un étudiant SE rapporte 50 euro
- Il faut un écart d'au plus 12 étudiants formés entre les deux filières

- Max $160 * x_{SQR} + 50 * x_{SE}$
  - $20 * x_{SQR} + 8 * x_{SE} \leq 800$ (bières)
  - $120 * x_{SQR} + 200 * x_{SE} \leq 9000$ (chips)
  - $27 * x_{SQR} + 4 * x_{SE} \leq 925$ (pizzas)
  - $10 \leq x_{SQR} \leq 40$ (contraintes marché SQR)
  - $8 \leq x_{SE} \leq 30$ (contraintes marché SE)
  - $x_{SE} - x_{SQR} \leq 12$ (écart max de 12)
  - $x_{SQR} - x_{SE} \leq 12$ (écart max de 12)


x_{SE}
40
30
20
10
10
20
30
40
x_{SQR}


x_{SE}
40
30
20
10
10
20
30
40
x_{SQR}

Vocabulaire associé à la PL

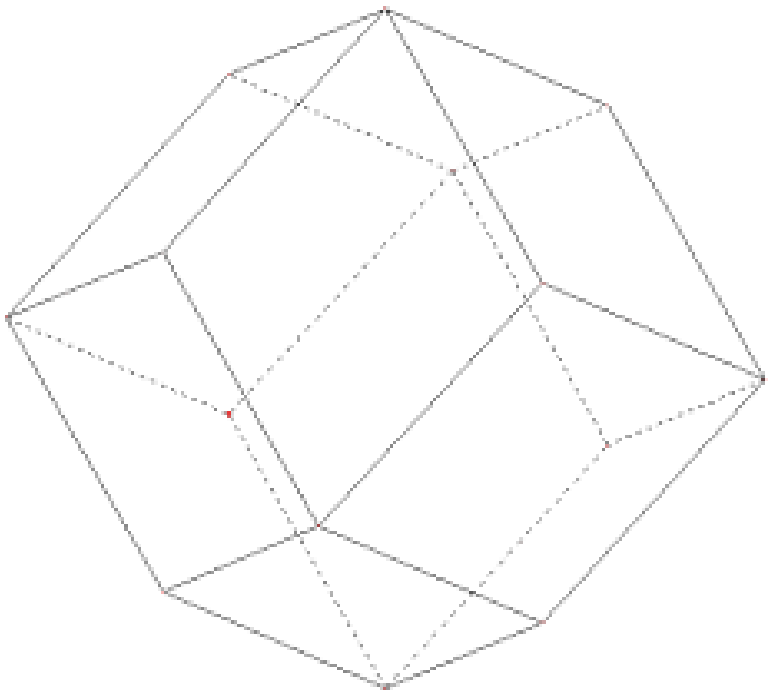
Quelques notions élémentaires

- x_i est une variable de décision du problème
- les contraintes sont des inégalités ou des égalités linéaires
- la fonction objectif est la fonction linéaire à maximiser
- Une solution $x = (x_1, \ldots, x_n)$ est réalisable si elle satisfait à toutes les contraintes

L'espace des solutions réalisables : le polyhèdre

- Chaque variable de décision x_i est une dimension du polyhèdre
  - deux variables : représentation 2D
  - trois variables : représentation 3D
  - n variables : espace n-dimensionnel
- Chaque contrainte est un plan de coupe dans l'espace

Polyhèdre convexe



Résoudre un programme linéaire

Utilisation d'un solveur de PL

- Méthodes de résolution générales et efficaces en théorie et en pratique
- Restreint aux **variables réelles**, contraintes linéaires, et objectifs linéaires
- Différents outils : Excel, GLPK, CPLEX, ...
- Plus le PL est grand (milliers de variables et de contraintes) plus il prendra de temps à être résolu.
- Mais généralement, résolution efficace

challenge : exprimer un problème sous forme d'équations

- 0, une, ou plusieurs formulations possibles.
- Trouver la bonne formulation.
- Le solveur se charge des calculs et de retourner le résultat
- exemple : expression du problème en python dans sage

Exemple de solveur : sagemath + cplex

programme linéaire, version in extenso :

- **Objectif** : Max $150 * x_{SQR} + 110 * x_{SE}$
- **contraintes** :
  - $25^* x_{SQR} + 12 * x_{SE} \leq 1000$ (bières)
  - $100^* x_{SQR} + 230 * x_{SE} \leq 7500$ (chips)
  - $x_{SQR} \geq 0$ et $x_{SE} \geq 0$

Routines

- `MixedIntegerLinearProgram(...)` : création d'un LP
- méthode `new_variable()` : création d'un jeu de variables

Expression du premier problème sous Sage

```
1  from sage.numerical.mip import MixedIntegerLinearProgram,MIPSolverException
2
3  # donnees de l'enonce
4  nbBieresDispo = 1000
5  nbChipsDispo = 7500
6  coutBieres_SQR = 25
7  coutChips_SQR = 100
8  coutBieres_SE = 12
9  coutChips_SE = 230
10 TaxeApprent_SQR = 150
11 TaxeApprent_SE = 110
12
13 # creation d'un probleme en Programmation lineaire
14 pb = MixedIntegerLinearProgram(maximization=True, solver='cplex')
15
16 # creation des variables x (tableau : nombres d'étudiants a nourir - SE  SQR -)
17 x = pb.new_variable(name='x', nonnegative=True)
18
19 # ajout des contraintes
20 pb.add_constraint(coutChips_SE* x['SE'] + coutChips_SQR* x['SQR'] <= nbChipsDispo)
21 pb.add_constraint(coutBieres_SE* x['SE'] + coutBieres_SQR* x['SQR'] <= nbBieresDispo)
22
23 #ajout d'un objectif a maximiser
24 pb.set_objective( TaxeApprent_SE*x['SE'] + TaxeApprent_SQR*x['SQR'])
25
26 #execution et recuperation des resultats
27 objectif = pb.solve()
28 print "nombre de SE a nourrir :", pb.get_values(x['SE'])
29 print "nombre de SQR a nourrir :", pb.get_values(x['SQR'])
30 print "Taxe d'apprentissage esperee :",objectif
```

Représentation matricielle

programme linéaire, version in extenso :

- **Objectif** : Max $150 * x_{SQR} + 110 * x_{SE}$
- **contraintes** :
  - $25^* x_{SQR} + 12 * x_{SE} \leq 1000$ (bières)
  - $100^* x_{SQR} + 230 * x_{SE} \leq 7500$ (chips)
  - $x_{SQR} \geq 0$ et $x_{SE} \geq 0$

programme linéaire, version matricielle :

objectif :

$$\max \quad \begin{pmatrix} 150 & 110 \end{pmatrix} \begin{pmatrix} x_{SQR} \\ x_{SE} \end{pmatrix}$$

contraintes :

$$\begin{pmatrix} 25 & 12 \\ 100 & 230 \end{pmatrix} \begin{pmatrix} x_{SQR} \\ x_{SE} \end{pmatrix} \leq \begin{pmatrix} 1000 \\ 7500 \end{pmatrix}$$

$$x_{SQR} \geq 0 \quad x_{SE} \geq 0$$

Rappel : multiplications de matrices

```
1  var('x1'); var('x2'); var('x3'); var('a'); var('b');
2  var('c'); var('d'); var('e'); var('f')
3
4  A = matrix([ [a,b,c] ])
5  X = matrix([ [x1],[x2],[x3] ])
6  print "A="
7  print A
8
9  print "\nX="
10 print X
11
12 print "\nA*X="
13 print A*X
14
15 B = matrix([ [a,b,c], [d,e,f] ])
16 print "\nB="
17 print B
18
19 print "\nB*X="
20 print B*X
```

```
A=
[a b c]

X=
[x1]
[x2]
[x3]

A*X=
[a*x1 + b*x2 + c*x3]

B=
[a b c]
[d e f]

B*X=
[a*x1 + b*x2 + c*x3]
[d*x1 + e*x2 + f*x3]
```

Représentation matricielle

- n variables de décision

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

- coût (ou profit)

$$c = \begin{pmatrix} c_1 & c_2 & \dots & c_n \end{pmatrix}$$

- matrice de format $m \times n$

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

- second membre

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

- **Représentation matricielle :**

$$\max z = cx$$

s.c.

$$\begin{array}{lcll} Ax & \left\{ \begin{array}{l} \leq \\ \geq \\ = \end{array} \right. & b & (1) \\ x & \geq & 0 & (2) \end{array}$$

Exercice : une histoire de covoiturage

Organisation du covoiturage Dijon-Toulouse

- Denis propose un trajet co-voiturage entre Dijon et Toulouse : 7 places dispo
- Il propose son trajet à 25 euro. Mais offre une réduction de 5 euro aux Geipouille

Contraintes de covoitureurs

- Les garcons de l'ESIROUM veulent au moins autant de filles d'AgroSoup qu'eux
- Les filles de l'ESIROUM ne veulent pas plus de 3 garçons de l'ESIROUM
- Les Geipouille veulent au moins 1 personne de l'ESIROUM et 1 personne d'AgroSoup
- Les filles de l'ESIROUM/AgroSoup veulent autant de Geipouille qu'elles
- Les garcons de l'ESIROUM veulent être au moins aussi nombreux que les garcons d'AgroSoup (peur de la concurrence)
- Les garcons Geipouille veulent être au moins autant que les filles Geipouille

Quel est l'équipage le plus rentable ?

Exercice : une histoire de covoiturage

Formulation In extenso

Variables : $X_{sexe_{ecole}}$, pour $sexe \in ['H','F']$ et pour $ecole \in ['ESIR','AS','Gpl']$
Objectif : Max $25.x_{H_{ESIR}} + 25.x_{F_{ESIR}} + 25.x_{H_{AS}} + 25.x_{F_{AS}}) + 20.x_{H_{Gpl}} + 20.x_{H_{Gpl}}$
Contraintes :

- $x_{H_{ESIR}} - x_{F_{AS}} \leq 0$ (au moins autant de F AgroSoup que H ESIROUM) :
- $x_{H_{ESIR}} \leq 3$ (au plus 3 H ESIROUM, sinon ils sont lourds)
- $x_{H_{ESIR}} + x_{F_{ESIR}} \geq 1$ (au moins un ESIROUM)
- $x_{H_{AS}} + x_{F_{AS}} \geq 1$ (au moins un ESIROUM)
- $x_{F_{AS}} - x_{H_{Gpl}} - x_{F_{Gpl}} = 0$ (autant de Geipouille que de filles AS/ ESIROUM)
- $x_{H_{ESIR}} - x_{H_{AG}} \geq 0$ (au moins autant d'H ESIROUM que d'H AgroSoup)
- $x_{H_{Gpl}} - x_{F_{Gpl}} \geq 0$ (au moins autant de H Geipouille que de F Geipouille)

Exercice : une histoire de covoiturage

Formulation Matricielle

$$\max \begin{pmatrix} 25 & 25 & 25 & 25 & 20 & 20 \end{pmatrix} \begin{pmatrix} x_{H_{ESIR}} \\ x_{F_{ESIR}} \\ x_{H_{AS}} \\ x_{F_{AS}} \\ x_{H_{Gpl}} \\ x_{H_{Gpl}} \end{pmatrix}$$

$$sc \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_{H_{ESIR}} \\ x_{F_{ESIR}} \\ x_{H_{AS}} \\ x_{F_{AS}} \\ x_{H_{Gpl}} \\ x_{H_{Gpl}} \end{pmatrix} \begin{matrix} \leq \\ \leq \\ \geq \\ = \\ \geq \\ \geq \end{matrix} \begin{pmatrix} 0 \\ 3 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Exercice : une histoire de covoiturage

La formulation est doublement fausse ! !

Erreur dans le raisonnement logique

- Les garçons de l'ESIROUM veulent ceci, veulent cela ...
- si on les dégage, leur revendications n'ont plus lieu de tenir !
- Les garçons d'AgroSoup n'ont aucune contrainte et rapportent un max
- Solution immédiate : 7 garcons d'AgroSoup, rien d'autre

Il est **difficile** - **mais pas impossible** d'exprimer la conjonction ('ou')

Erreur dans l'adéquation de l'outil

- Examinons les résultats . . .

Exercice : une histoire de covoiturage

La formulation est doublement fausse ! !

Erreur dans le raisonnement logique

- Les garçons de l'ESIROUM veulent ceci, veulent cela ...
- si on les dégage, leur revendications n'ont plus lieu de tenir !
- Les garçons d'AgroSoup n'ont aucune contrainte et rapportent un max
- Solution immédiate : 7 garcons d'AgroSoup, rien d'autre

Il est **difficile** - **mais pas impossible** d'exprimer la conjonction ('ou')

Erreur dans l'adéquation de l'outil

- Examinons les résultats . . .

Exercice : une histoire de covoiturage

La formulation est doublement fausse ! !

Erreur dans le raisonnement logique

- Les garçons de l'ESIROUM veulent ceci, veulent cela ...
- si on les dégage, leur revendications n'ont plus lieu de tenir !
- Les garçons d'AgroSoup n'ont aucune contrainte et rapportent un max
- Solution immédiate : 7 garcons d'AgroSoup, rien d'autre

Il est **difficile** - **mais pas impossible** d'exprimer la conjonction ('ou')

Erreur dans l'adéquation de l'outil

- Examinons les résultats . . .

Expression du problème de covoiturage sous Sage

```
1  from sage.numerical.mip import MixedIntegerLinearProgram,MIPSolverException
2
3  # donnees de l'enonce
4  nbPlaces = 7
5  prix = 25
6  prixGeipouille = 20
7  nbMaxHESIROUM = 3
8  nbMinESIROUM = 1
9  nbMinAgroSoup = 1
10
11 # creation d'un probleme en Programmation lineaire
12 pb = MixedIntegerLinearProgram(maximization=True, solver='cplex')
13 # creation des variables x (tableau : classes de covoitureurs)
14 x = pb.new_variable(name='x', nonnegative=True)
15
16 # ajout des contraintes
17 pb.add_constraint(x['H-ESIROUM'] - x['F-AgroSoup'] <= 0)
18 pb.add_constraint(x['H-ESIROUM'] <= nbMaxHESIROUM)
19 pb.add_constraint(x['H-ESIROUM'] + x['F-ESIROUM'] >= nbMinESIROUM)
20 pb.add_constraint(x['H-AgroSoup'] + x['F-AgroSoup'] >= nbMinAgroSoup)
21 pb.add_constraint(x['F-ESIROUM'] + x['F-AgroSoup'] - x['H-Geipouille'] - x['F-
       Geipouille']==0)
22 pb.add_constraint(x['H-ESIROUM'] - x['H-AgroSoup'] >= 0)
23 pb.add_constraint(x['H-Geipouille'] - x['F-Geipouille'] >= 0)
24 pb.add_constraint(x['H-AgroSoup'] + x['F-AgroSoup'] + x['H-ESIROUM'] + x['F-ESIROUM']
        + x['H-Geipouille'] + x['F-Geipouille']  <= nbPlaces)
25
26 #ajout d'un objectif a maximiser
27 pb.set_objective(prix*x['H-AgroSoup'] + prix*x['F-AgroSoup'] + prix*x['H-ESIROUM'] +
       prix*x['F-ESIROUM'] + prixGeipouille*x['H-Geipouille'] + prixGeipouille*x['F-
       Geipouille'] )
```

Exercice : une histoire de covoiturage

Résultats

- H/F ESIROUM : 1.75 / 0.0
- H/F AgroSoup : 1.75 / 1.75
- H/F Geipouille : 0.875 / 0.875

On ne peut pas découper des gens !

Limites de la PL

- La PL ne donne **généralement** pas de résultats en valeur entière
- Restrictions aux problèmes dont on veut des valeurs réelles
- Possibilité d'arrondir les valeurs, mais optimal non garanti
- Cependant : il est parfois possible de s'assurer que la PL retourne des valeurs entières

Utilisation de la PL pour les problèmes de flot

Problème de flots : contexte

- Problèmes de transport de quantités dans un réseau de distribution
- Une ou plusieurs source, vers une ou plusieurs destinations
- Plusieurs chemins contraints : capacité sur les liens, coût de transit, ...
- Quantités continues (nombres réels) ou discrètes (nombres entiers)

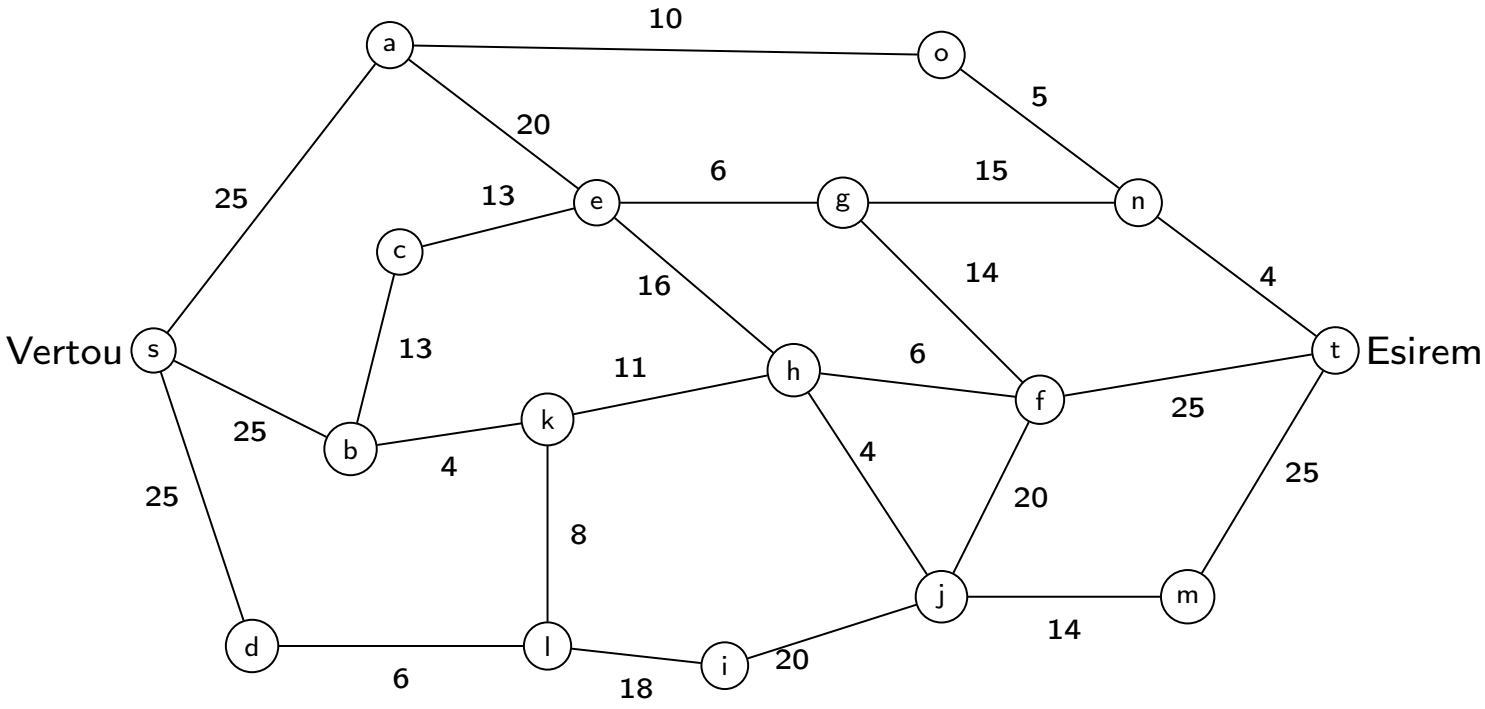
Modélisation et résolution

- Modélisation par des graphes et fonctions de coûts
- Objectifs : maximisation du flux, réduction des coûts
- Différents algorithmes de résolution : chaîne augmentante, ...
- Modélisation en PL très simple et efficace !

Organisation d'un trafic de choco BN

Flot maximum : modélisation et résolution

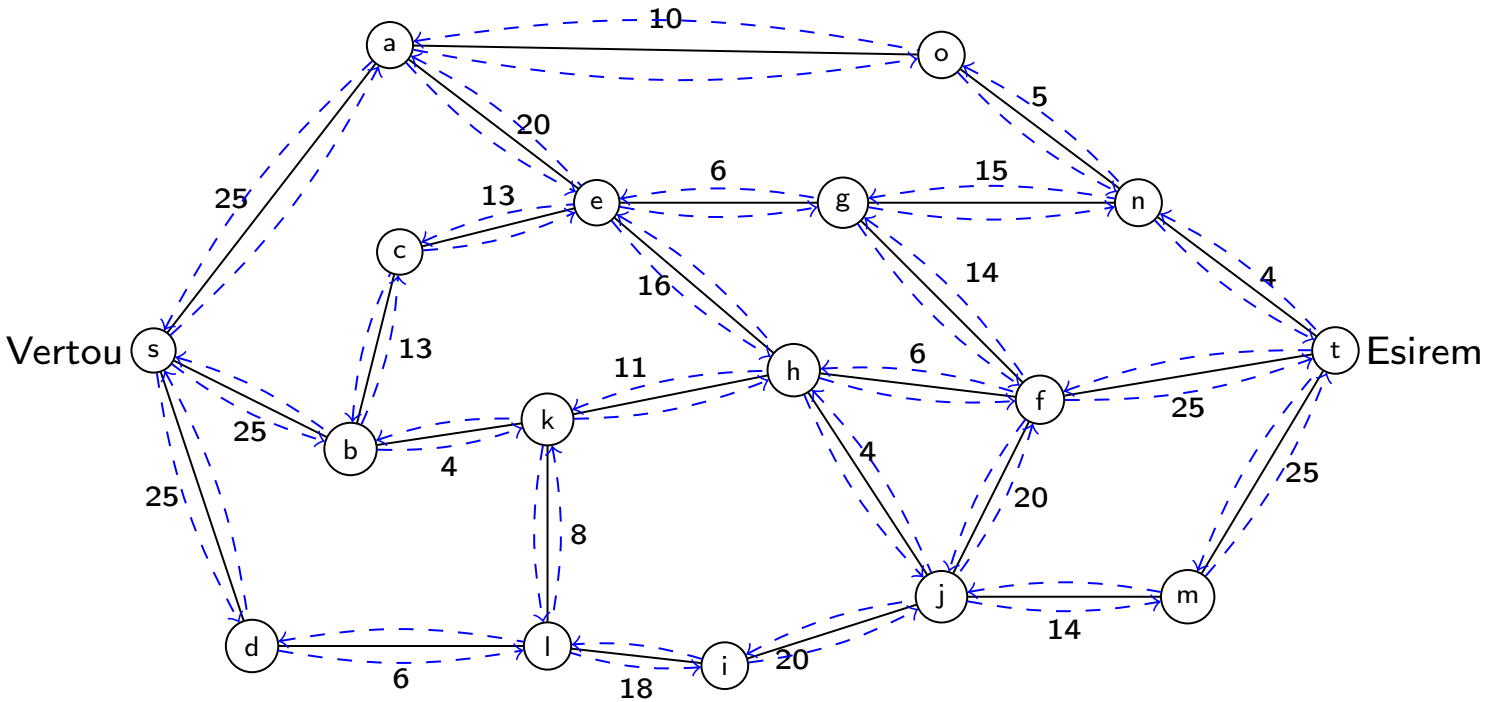
- Sevann a un contact à l'usine de Vertou pour "sortir" des choco-BN
- Mise en place d'un réseau d'intermédiaires et de sites de dépôt a, b, c, ..., Transport inter-sites assurés par intermédiaires, capacités de transport bornées.
- Quelle quantité maxi de chocho-BN peut recevoir l'ESIROUM par semaine ?



Organisation d'un trafic de choco BN

Flot maximum : modélisation et résolution

- Sevann a un contact à l'usine de Vertou pour "sortir" des choco-BN
- Mise en place d'un réseau d'intermédiaires et de sites de dépôt a, b, c, ..., Transport inter-sites assurés par intermédiaires, capacités de transport bornées.
- Quelle quantité maxi de chocho-BN peut recevoir l'ESIROUM par semaine ?



Organisation d'un trafic de choco BN

Données du problème

- Réseau modélisé par un graphe $G = (V, E)$ avec V ses noeuds, et E ses arêtes ; une fonction de poids $w : E \to N$; une source $s \in V$, une destination $t \in V$.

Formulation In extenso du problème

Variables : $x_{(i,j)}$: flot transitant sur l'arête $\{i, j\}$ dans le sens i vers j, $\quad \forall \{i, j\} \in E$
Objectif : $\max \sum_{u \in N(t)} x_{(u,t)}$
Contraintes :

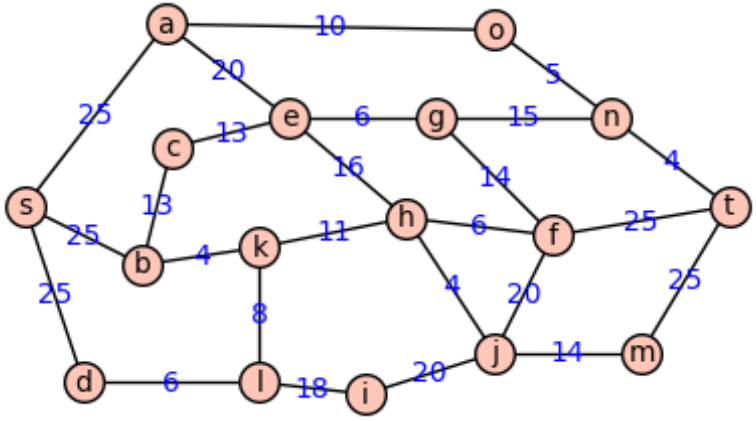
- Rien ne transite vers la source s : $\quad \sum_{u \in N(s)} x_{(u,s)} = 0$
- Rien ne transite depuis la destination t : $\quad \sum_{u \in N(t)} x_{(t,u)} = 0$
- Contrainte de capacité sur les liens : $\quad \forall \{u, v\} \in E \quad x_{(u,v)} + x_{(v,u)} \leq w(\{u, v\})$
- Loi de conservation des flots : ce qui rentre vers un nœud = ce qui en sort :

$$\forall u \in V - \{\{s, t\}\} \quad \sum_{v \in N(u)} x_{(v,u)} = \sum_{v \in N(u)} x_{(u,v)}$$

- Les quantités du flot sont toutes positives : $\quad \forall \{i, j\} \in E \quad x_{(i,j)} \geq 0$

Expression du reseau pour le problème de traffic de choco-BN sous Sage

```
1  G=graphs.EmptyGraph()
2
3  #ajout par (extremite1, extremite2, poids)
4  G.add_edge('s','a',25)
5  G.add_edge('s','b',25)
6  G.add_edge('s','d',25)
7  G.add_edge('a','o',10)
8  G.add_edge('a','e',20)
9
10 # methode plus rapide
11 G.add_edges([('c','e',13), ('b','c',13), ('b','k',4), ('d','l',6), ('e','g',6), ('e',
       'h',16), ('k','h',11), ('k','l',8), ('o','n',5), ('n','t',4), ('l','i',18), ('i
       ','j',20), ('j','m',14), ('m','t',25), ('g','n',15) , ('h','j',4), ('h','f',6),
        ('f','t',25), ('f','g',14), ('f','j',20)])
12 G.show()
```

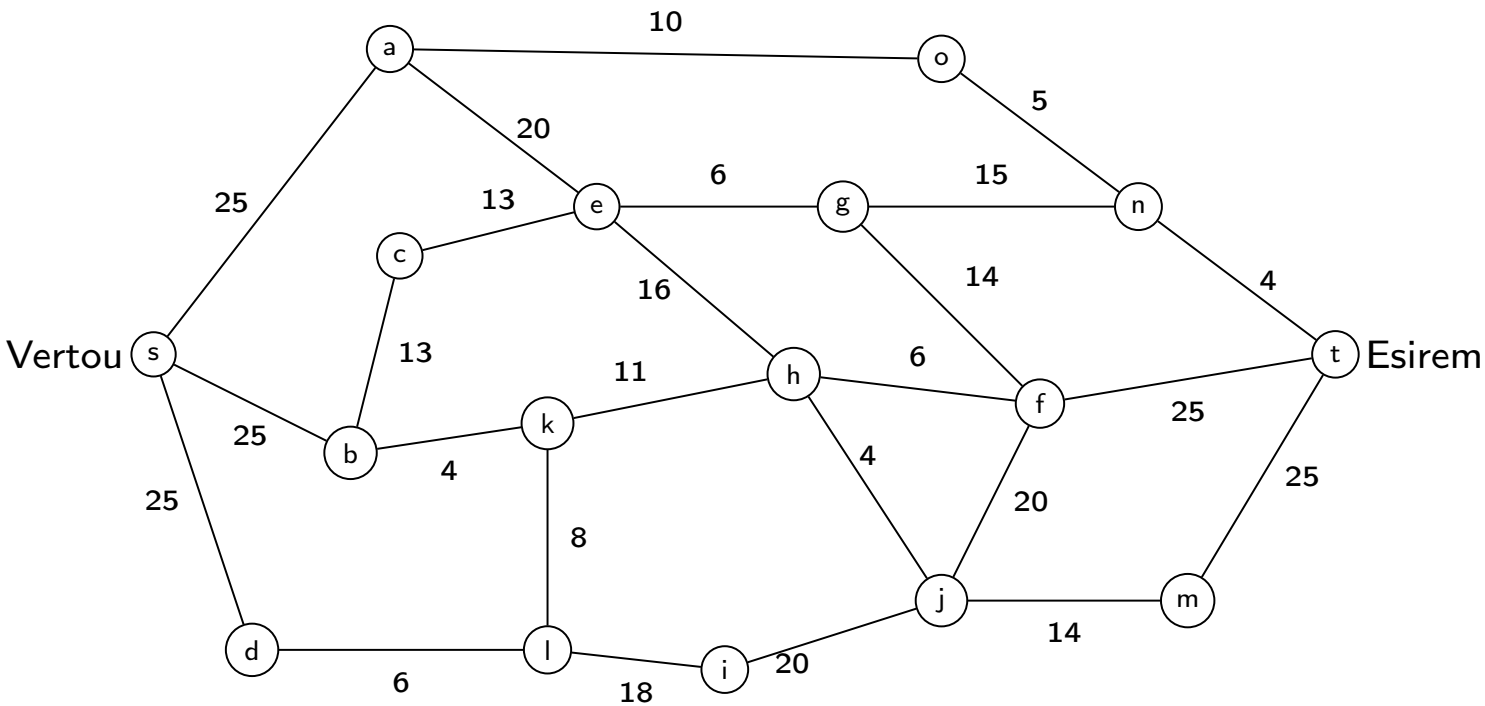


Expression du problème de traffic de choco-BN sous Sage

```
1  # definition du probleme
2  def LP_flotMax(G, src, dst) :
3      V = set(G.vertices())
4      E = set(G.edges())
5      # creation d'un probleme en Programmation lineaire
6      pb = MixedIntegerLinearProgram(maximization=True, solver='cplex')
7      # creation des variables x (tableau : classes de covoitureurs)
8      x = pb.new_variable(name='x', nonnegative=True)
9
10     # rien ne transite vers la source
11     pb.add_constraint(sum(x[(u,src)] for u in G.neighbors(src)) ==0 )
12
13     # rien ne transite depuis la destination
14     pb.add_constraint(sum(x[(dst, u)] for u in G.neighbors(dst)) ==0 )
15
16     # contrainte de capacite sur les liens
17     for e in E :
18         u = e[0];    v= e[1];    w = e[2];
19         pb.add_constraint(x[(u,v)] +x[(v,u)] <= w)
20
21     # loi de conservation des flots
22     for u in V - {src,dst} :
23         pb.add_constraint(sum(x[(v, u)] for v in G.neighbors(u)) == sum(x[(u,v)] for
               v in G.neighbors(u)))
24
25     pb.set_objective(sum(x[(u,dst)] for u in G.neighbors(dst)))
26     return [pb, x]
27
28 # appel de la fonction et execution du solveur
29 [pb, x] = LP_flotMax(G,'s','t')
30 objectif = pb.solve(log=0)
31 print "cout du flot :",objectif
```

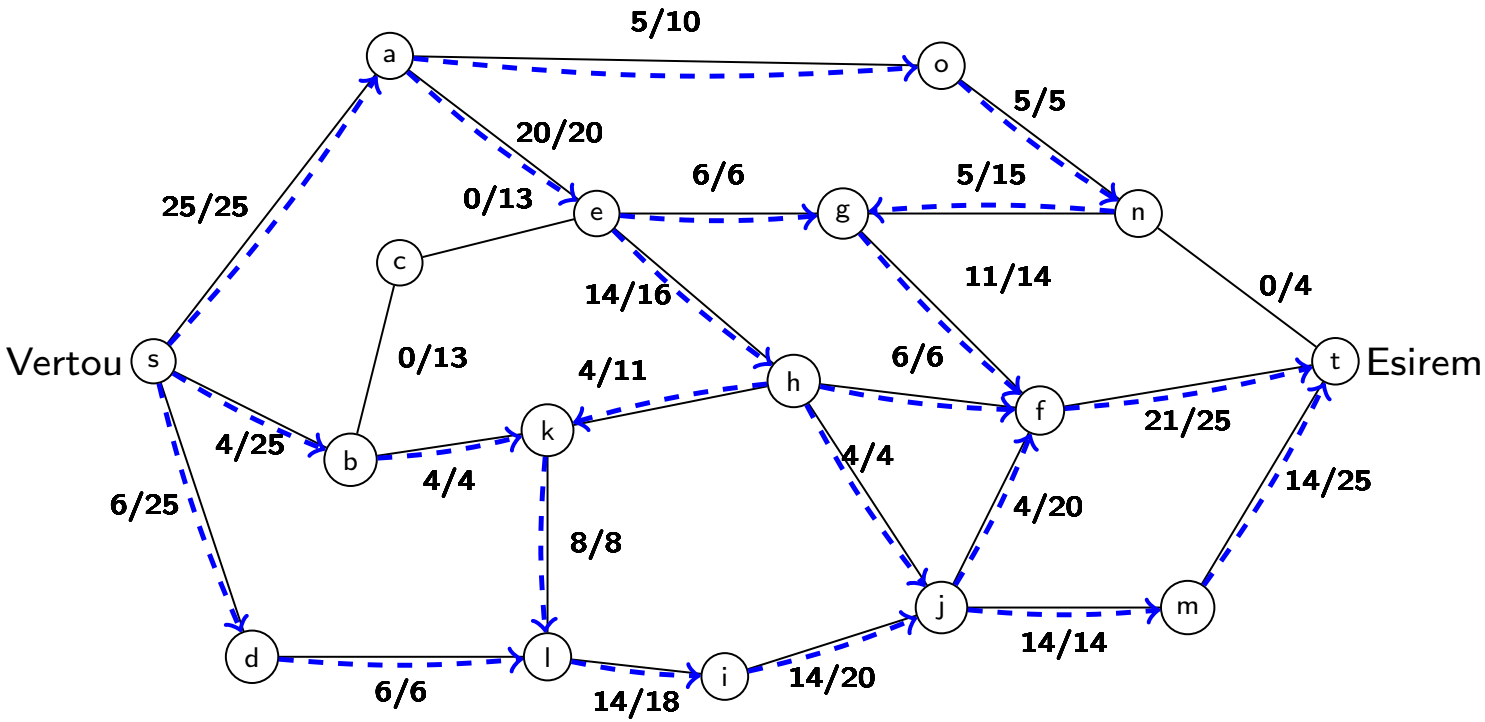
Récupération des valeurs des flots sur les arêtes :

```
1  for e in G.edges():
2      u = e[0];     v= e[1];     w = e[2];
3      if pb.get_values(x[(u,v)]) != 0 :
4          print "flot de ", u," vers ", v, ": ",  pb.get_values(x[(u,v)])
5      if pb.get_values(x[(v,u)]) != 0 :
6          print "flot de ", v," vers ", u, ": ",  pb.get_values(x[(v,u)])
```



Récupération des valeurs des flots sur les arêtes :

```
1  for e in G.edges():
2      u = e[0];     v= e[1];     w = e[2];
3      if pb.get_values(x[(u,v)]) != 0 :
4          print "flot de ", u," vers ", v, ": ",  pb.get_values(x[(u,v)])
5      if pb.get_values(x[(v,u)]) != 0 :
6          print "flot de ", v," vers ", u, ": ",  pb.get_values(x[(v,u)])
```



Tailles d'instances calculables en des temps raisonnables

Durée de résolution du problème

- dépend du nombre de variables et de contraintes de lignes et de colonnes, qui varient en fonction de la donnée en entrée
- Mais aussi de la nature du problème, sa formulation , ...
- Généralement : relativement impredictible, mais supposé polynomial !

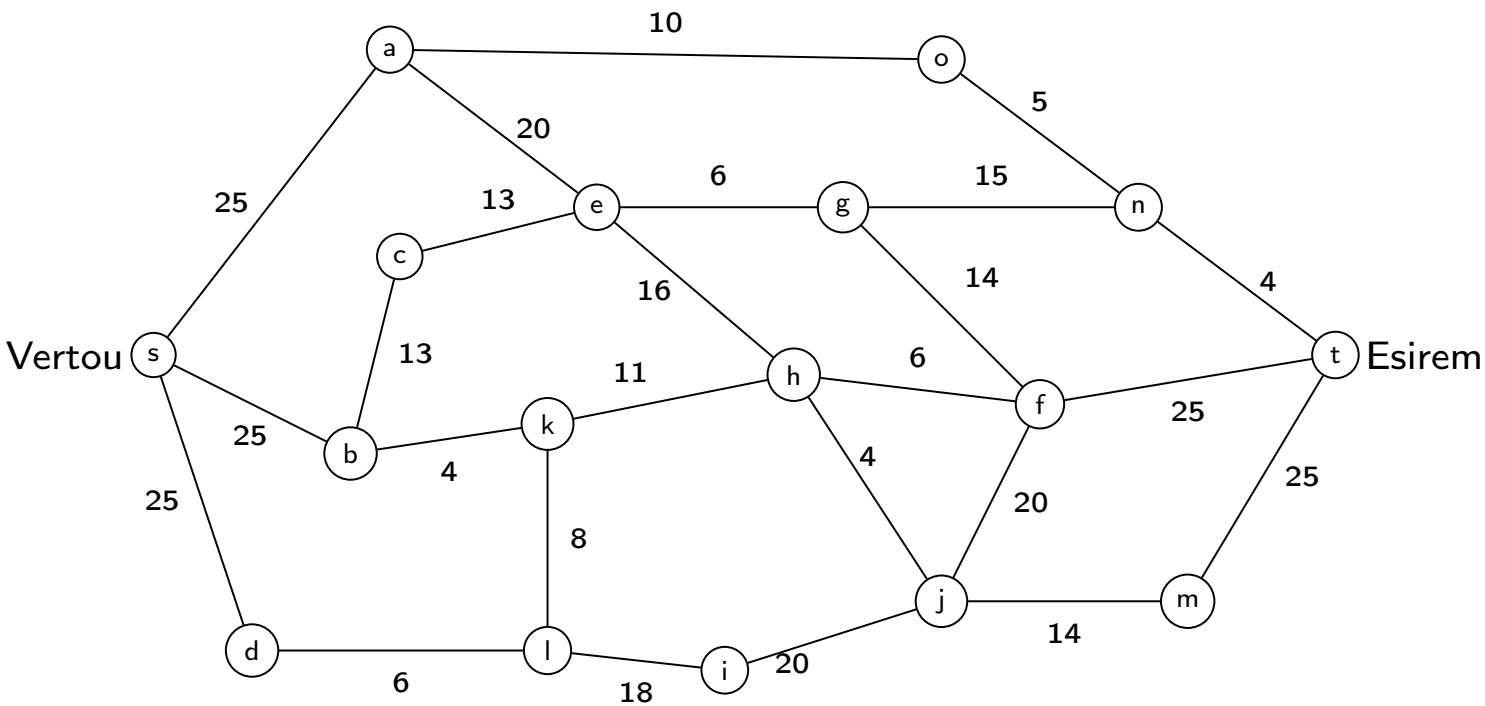
Pour le problème de flot

- Donnée : $G = (V, E)$ avec $|V| = n$ et $|E| = m$
- Nombre de variables : $2m$
- Nombre de contraintes : $2 + 2m + n - 2 = n + 2m$

Démantèlement d'un trafic de choco BN

Coupe minimum : Modélisation et résolution

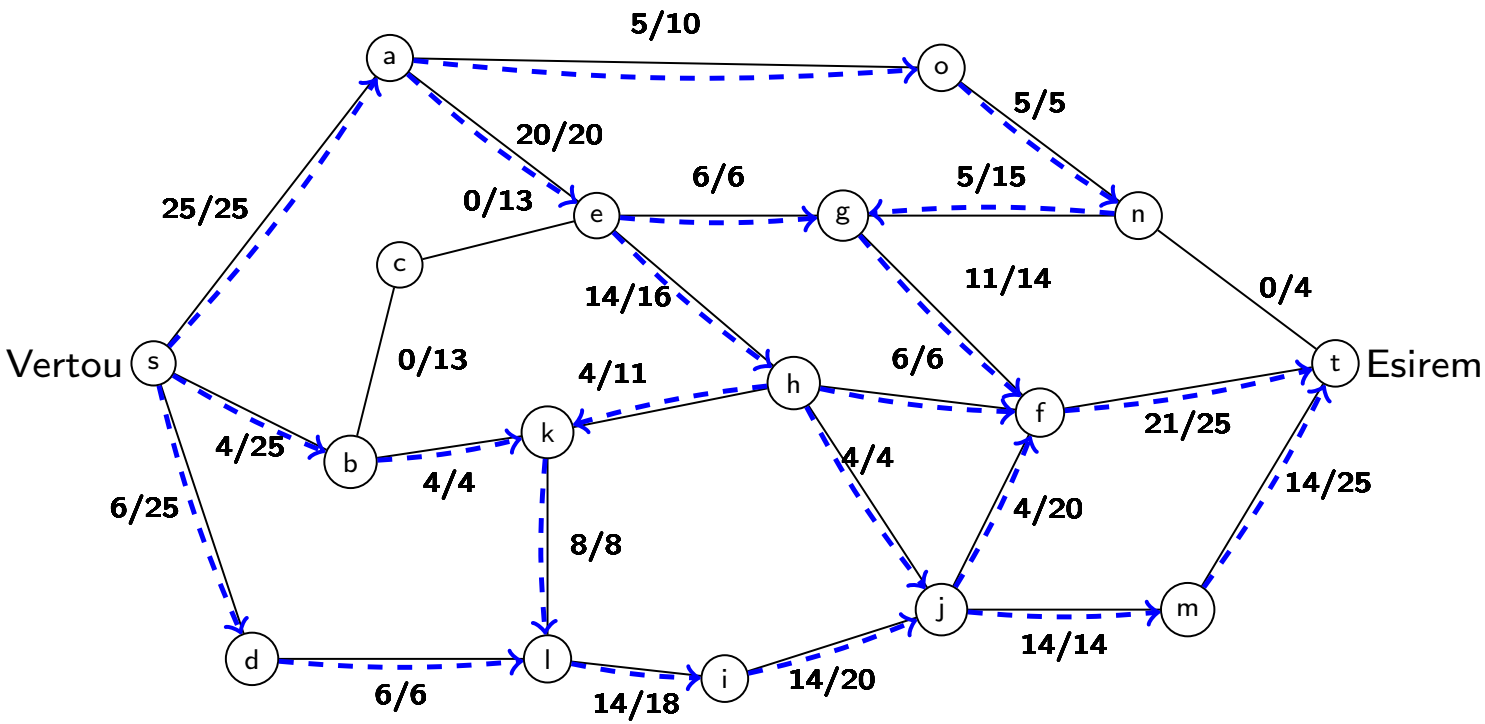
- Samuel, qui fournit des granolas à l'ESIROUM, est inquiet du commerce de Sevann et veut le faire tomber. Il veut soudoyer les intermédiaires : chaque intermédiaire veut bien arrêter de transporter des paquets s'il est payé du montant maximum de choco-BN qu'il pourrait transporter.
- Quelle minimum doit débourser Samuel pour stopper entièrement le traffic ?



Démantèlement d'un trafic de choco BN

Coupe minimum : Modélisation et résolution

- Samuel, qui fournit des granolas à l'ESIROUM, est inquiet du commerce de Sevann et veut le faire tomber. Il veut soudoyer les intermédiaires : chaque intermédiaire veut bien arrêter de transporter des paquets s'il est payé du montant maximum de choco-BN qu'il pourrait transporter.
- Quelle minimum doit débourser Samuel pour stopper entièrement le traffic ?

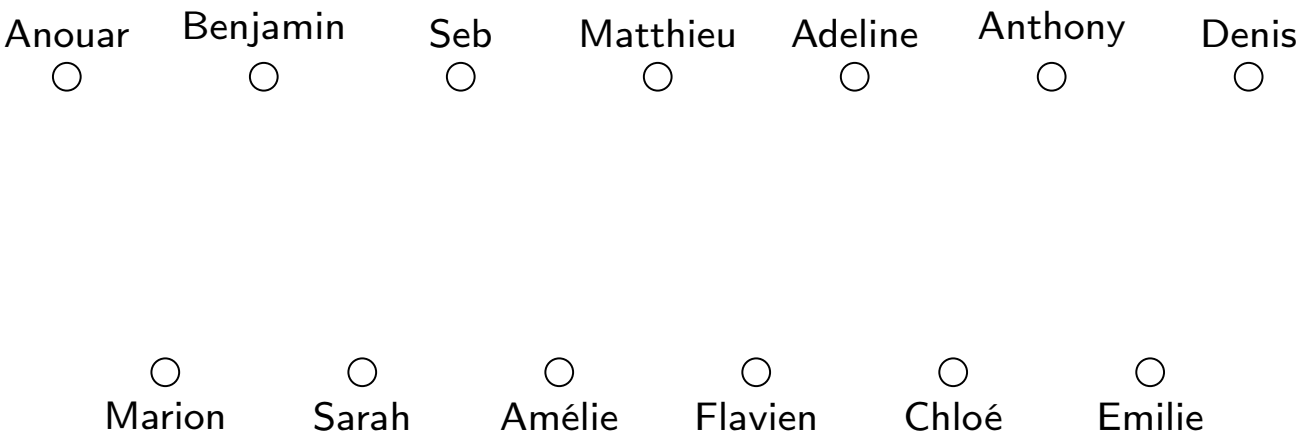


Problème dual

- Tout problème consistant à maximiser une quantité, possède un problème opposé consistant à minimiser une autre quantité, et réciproquement : c'est le problème dual
- Un problème dual est l'inverse du problème de base, dit problème primal
- Le problème dual peut être facile, ou difficile à exprimer selon les configurations
- Parfois, il est plus facile de passer par le problème dual

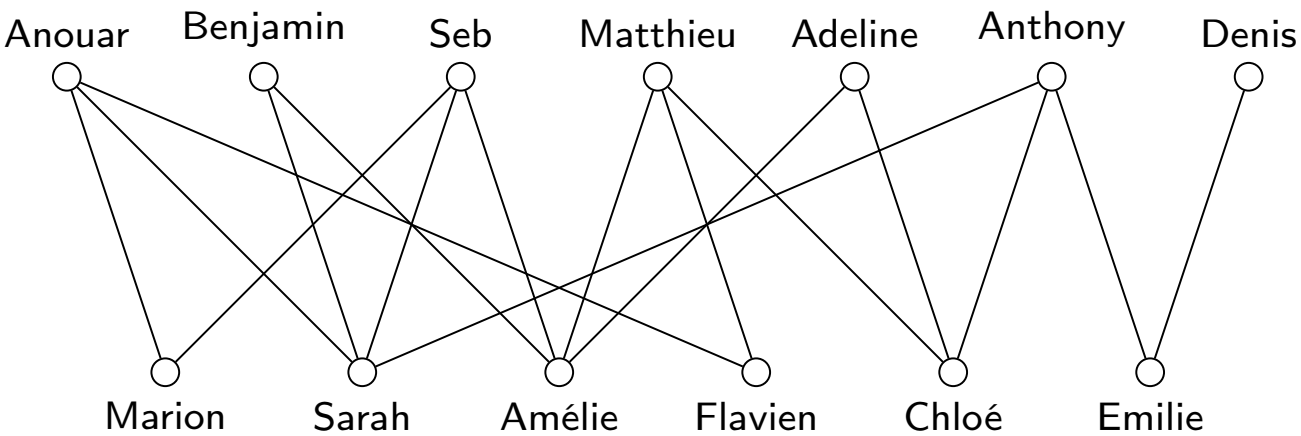
Une soirée speed-dating Esiroum-Agrosoup

- Les Esiroumiens et Agrosoupiens se rencontrent pendant 7 minutes sur le principe du speed-dating
- A la fin, on dresse un graphe des couples possibles
- Comment maximiser le nombre de couples que l'on peut créer ?



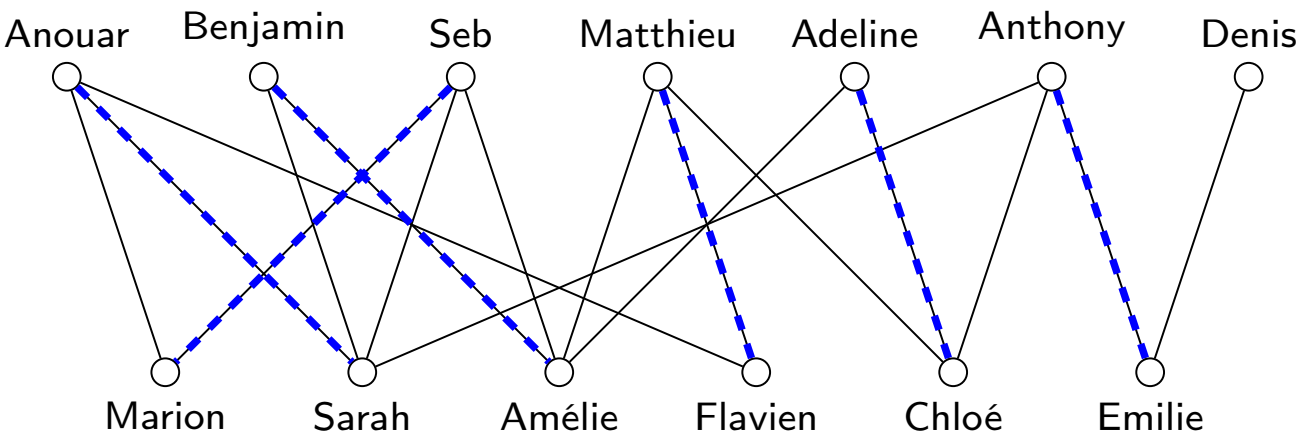
Une soirée speed-dating Esiroum-Agrosoup

- Les Esiroumiens et Agrosoupiens se rencontrent pendant 7 minutes sur le principe du speed-dating
- A la fin, on dresse un graphe des couples possibles
- Comment maximiser le nombre de couples que l'on peut créer ?



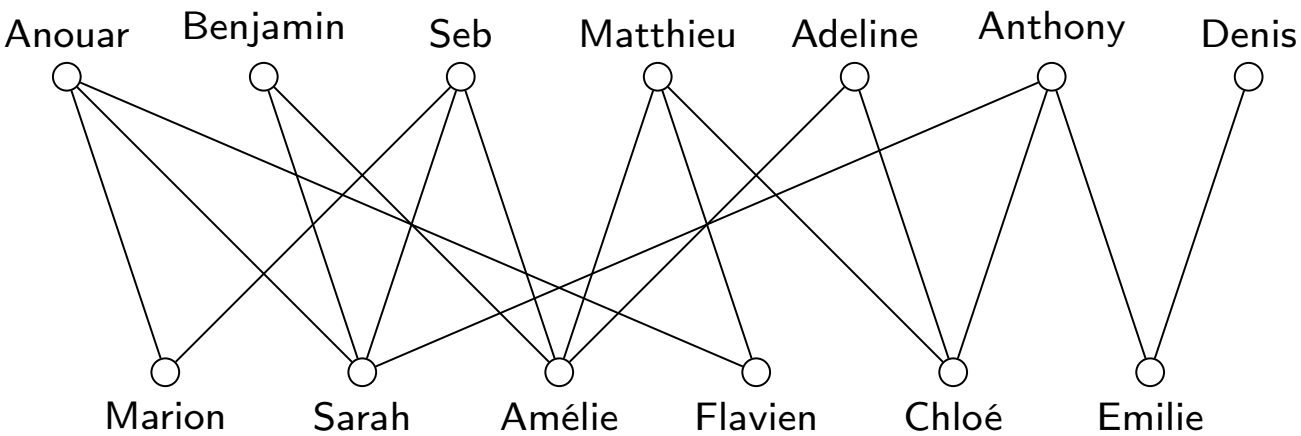
Une soirée speed-dating Esiroum-Agrosoup

- Les Esiroumiens et Agrosoupiens se rencontrent pendant 7 minutes sur le principe du speed-dating
- A la fin, on dresse un graphe des couples possibles
- Comment maximiser le nombre de couples que l'on peut créer ?



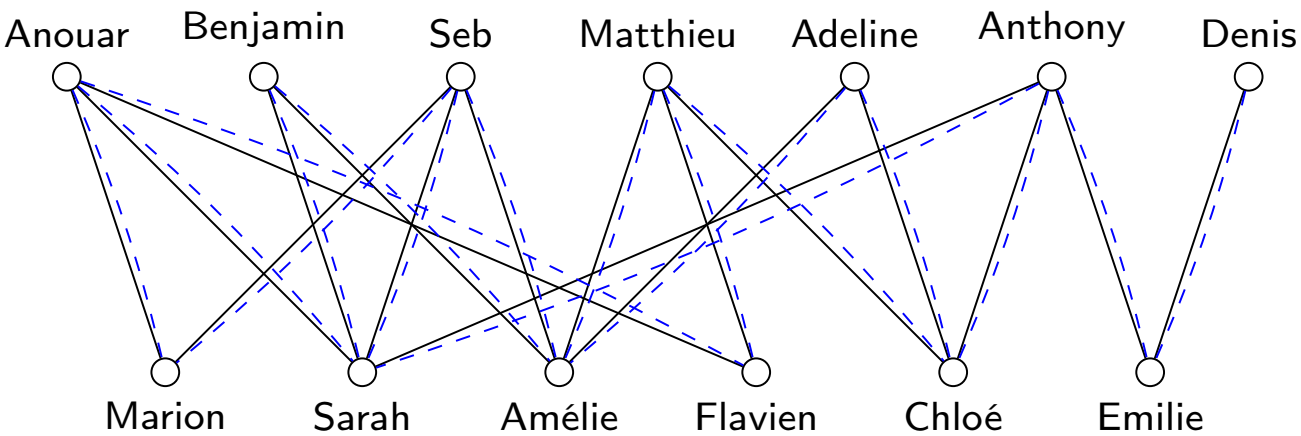
Une soirée speed-dating Esiroum-Agrosoup

- Les Esiroumiens et Agrosoupiens se rencontrent pendant 7 minutes sur le principe du speed-dating
- A la fin, on dresse un graphe des couples possibles
- Comment maximiser le nombre de couples que l'on peut créer ?



Une soirée speed-dating Esiroum-Agrosoup

- Les Esiroumiens et Agrosoupiens se rencontrent pendant 7 minutes sur le principe du speed-dating
- A la fin, on dresse un graphe des couples possibles
- Comment maximiser le nombre de couples que l'on peut créer ?



Une soirée speed-dating Esiroum-Agrosoup

Données du problème

- Réseau modélisé par un graphe biparti $G = (X, Y, E)$ avec $X \cup Y$ ses noeuds, et E ses arêtes

Formulation In extenso du problème

Variables : $x_{(i,j)}$: indique si l'arête $\{i, j\}$ doit être sélectionnée (=1) ou pas (=0)
Objectif : $\max \sum x_{(i,j)}$
Contraintes :

- Chaque personne u n'est impliquée que dans une relation au plus : $\forall u \in X \cup Y \quad \sum_{v \in N(u)} x_{u,v} \leq 1$
- Le domaine de chaque variable est entre 0 et 1 : $\forall \{u, v\} \in E \quad x_{u,v} \leq 1$

problème : formulation utilisant des variables binaires

- Les variables ne sont pourtant pas binaires, mais continues entre 0 et 1

Une soirée speed-dating Esiroum-Agrosoup

Données du problème

- Réseau modélisé par un graphe biparti $G = (X, Y, E)$ avec $X \cup Y$ ses noeuds, et E ses arêtes

Formulation In extenso du problème

Variables : $x_{(i,j)}$: indique si l'arête $\{i, j\}$ doit être sélectionnée (=1) ou pas (=0)
Objectif : $\max \sum x_{(i,j)}$
Contraintes :

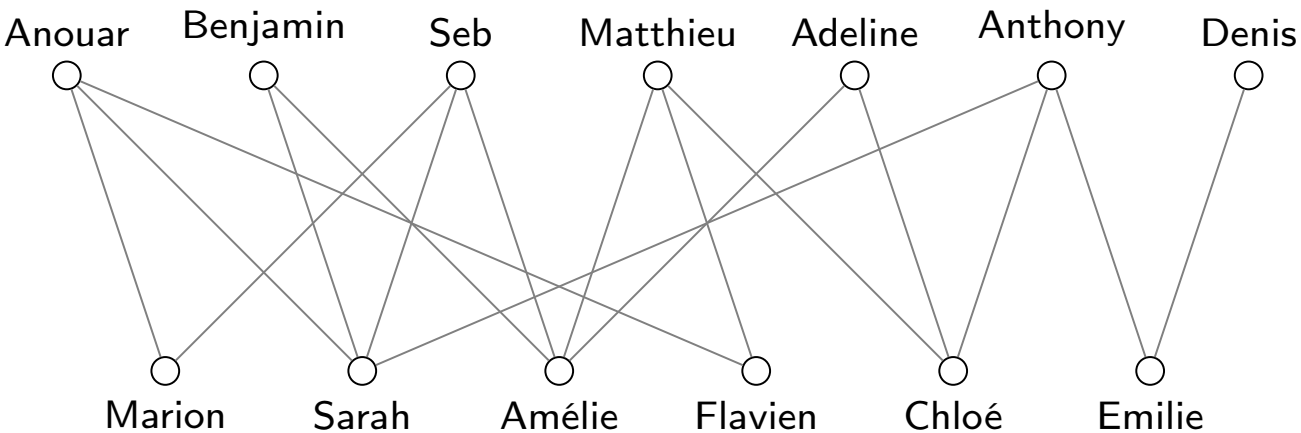
- Chaque personne u n'est impliquée que dans une relation au plus : $\forall u \in X \cup Y \quad \sum_{v \in N(u)} x_{u,v} \leq 1$
- Le domaine de chaque variable est entre 0 et 1 : $\forall \{u, v\} \in E \quad x_{u,v} \leq 1$

problème : formulation utilisant des variables binaires

- Les variables ne sont pourtant pas binaires, mais continues entre 0 et 1

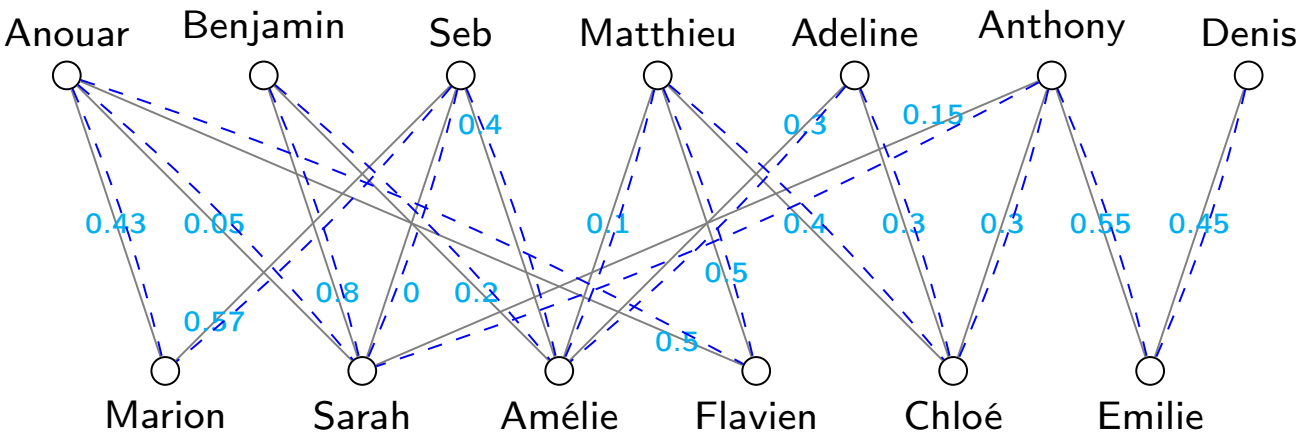
Une soirée speed-dating Esiroum-Agrosoup

- Les Esiroumiens et Agrosoupiens se rencontrent sur le principe du speed-dating
- A la fin des entretiens, on dresse un graphe des couples possibles
- Comment maximiser le nombre de couples que l'on peut créer ?

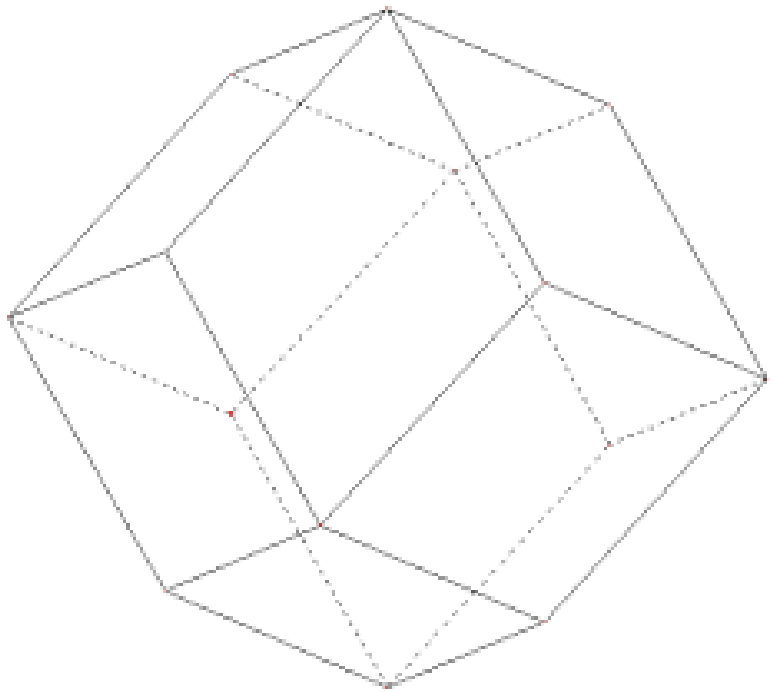


Une soirée speed-dating Esiroum-Agrosoup

- Les Esiroumiens et Agrosoupiens se rencontrent sur le principe du speed-dating
- A la fin des entretiens, on dresse un graphe des couples possibles
- Comment maximiser le nombre de couples que l'on peut créer ?



Rappel : les solutions sont des points du Polyhèdre



idée

- Solution(s) optimale(s) = coordonnées d'un angle
- Si les coordonnées de chacun des angles sont uniquement entières, alors n'importe quelle solution (optimale) sera composée de valeurs entières

Polyhèdre et matrice

relation polyhèdre - matrice

- le polyhèdre est décrit par la matrice
- si la matrice est totalement unimodulaire, alors les coordonnées de chacun des angles du polyhèdre sont entières

Definition

Matrice totalement unimodulaire Une matrice est totalement unimodulaire si et seulement si toute sous-matrice carrée possède un déterminant égal à 0, 1 ou -1

Représentation matricielle

programme linéaire, version matricielle :

contraintes :

$$\begin{pmatrix}
\textit{Anouar}: & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\textit{Benjamin}: & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\textit{Seb}: & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\textit{Matthieu}: & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\textit{Adeline}: & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
\textit{Anthony}: & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
\textit{Denis}: & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\textit{Marion}: & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\textit{Sarah}: & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\textit{Amlie}: & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\textit{Flavien}: & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\textit{Chlo}: & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
\textit{Emilie}: & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}
\begin{pmatrix}
x_{\textit{Ano:Mar}} \\
x_{\textit{Ano:Sar}} \\
x_{\textit{Ano:Fla}} \\
x_{\textit{Ben:Sar}} \\
x_{\textit{Ben:Ame}} \\
x_{\textit{Seb:Mar}} \\
x_{\textit{Seb:Sar}} \\
x_{\textit{Seb:Ame}} \\
x_{\textit{Mat:Ame}} \\
x_{\textit{Mat:Fla}} \\
x_{\textit{Mat:Chl}} \\
x_{\textit{Ade:Ame}} \\
x_{\textit{Ade:Chl}} \\
x_{\textit{Ant:Sar}} \\
x_{\textit{Ant:Chl}} \\
x_{\textit{Ant:Emi}} \\
x_{\textit{Den:Emi}}
\end{pmatrix}
\leq
\begin{pmatrix}
1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1
\end{pmatrix}$$

$$x_{\ldots} \geq 0$$

Quand on ne peut pas garantir que la matrice est totalement unimodulaire, mais que les variables doivent être entières :

Programmation Linéaire en nombres entiers !

The next chapter...