Stratégie de production

Optim

Paul Adenot, Etienne Brodu, Maxime Gaudin, Monica Golumbeanu,
Nor El Malki, Martin Richard, Yoann Rodière

19 octobre 2010

Table des matières

Résumé

L'objet de ce rapport est de présenter une solution capable de trouver une stratégie de production, menant à un contentement optimal des différent acteurs :

- Le comptable
- Le responsable des ateliers
- Le responsable des stocks
- Le responsable commercial

La démarche sera quant à elle incrémenta le puisque nous fournirons dans un premier temps des optimum très locaux, respectant un nombre limités de contraintes et de critères, puis nous combinerons ces différents résultat afin de converger vers une solution prenant en compte l'intérêt de chacun.

Première partie

Etude monocritère

1 Données

Soient :
- **T** la matrice des temps unitaires d'usinage d'un produit sur une machine (minutes) (*C.f. Table 1*).
- **Q** la matrice de quantité de matières premières par produit (*C.f. Table 2*).
- **S** la matrice des quantité maximum de matières premières (*C.f. Table 3*).
- **V** la matrice des prix de vente des produits finis (*C.f. Table 4*)
- **A** la matrice des prix d'achat des matières premières.
- **C** la matrice des coûts horaires des machines (*C.f. Table 5*).

1.1 Contraintes

Considérons :
- 6 produits identifiés chacun par une lettre $i \in A, B, C, D, E, F$
- 7 machines identifiée chacune par un nombre $j \in 1, 2, 3, 4, 5, 6, 7$
- 3 matières premières identifiée chacune par un nombre $k \in 1, 2, 3$
- n, vecteur colonne du nombre d'unités fabriquées pour chaque produit

L'ensemble de la chaine de production est régie par les contraintes suivantes :

- **Le nombre de produits usinés de type i :** Il doit être non nul

$$n_i \geq 0, \forall i \in A, B, C, D, E, F \tag{1}$$

- **Le temps d'occupation de chaque machine i :** Il doit être inférieur au temps de travail

$$\sum_{j=A}^{F} T_{j,i} \times n_j \leq 2 \times 8 \times 60 \times 5 = 4800, \forall i \in 1, 2, 3, 4, 5, 6, 7 \tag{2}$$

  soit un temps de travail en deux huit, 5 jours par semaine.
- **L'utilisation de chaque matière première i :** Elle doit être inférieure au stock

$$\sum_{j=A}^{F} Q_{i,j} \times n_j \leq S_i, \forall i \in 1, 2, 3 \tag{3}$$

1.1.1 Modélisation sous forme matricielle

Pour donner au problème une forme standard, nous allons le modéliser par des inéquations et des produits matriciels. Les contraintes C0, C1 et C2 se traduisent trivialement de la manière suivante :

$$A.n \leq b \tag{4}$$

Avec :

$$A = \begin{pmatrix} -I \\ \hline T^t \\ \hline Q \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline 4800 \\ 4800 \\ 4800 \\ 4800 \\ 4800 \\ 4800 \\ 4800 \\ \hline S^t \end{pmatrix}$$

Ce qui nous donne plus concrètement les matrices suivantes :

$$A = \left(\begin{array}{cccccc} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ \hline 8 & 15 & 0 & 5 & 0 & 10 \\ 0 & 1 & 2 & 15 & 7 & 12 \\ 8 & 1 & 11 & 0 & 10 & 25 \\ 2 & 10 & 5 & 4 & 13 & 7 \\ 5 & 0 & 0 & 0 & 10 & 25 \\ 5 & 5 & 3 & 12 & 8 & 0 \\ 5 & 3 & 5 & 8 & 0 & 0 \\ \hline 1 & 2 & 1 & 5 & 0 & 2 \\ 2 & 2 & 1 & 0 & 2 & 1 \\ 1 & 0 & 3 & 2 & 2 & 0 \end{array}\right), b = \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline 4800 \\ 4800 \\ 4800 \\ 4800 \\ 4800 \\ 4800 \\ 4800 \\ \hline 350 \\ 620 \\ 485 \end{array}\right)$$

2 Stratégie du point de vue *comptable*

En tenant compte des coût de fonctionnement des machines, du coût d'achat des mati1ères premières, du prix de vente des produits finis, et du temps d'usinage, le comptable de l'entreprise **Optim** cherche à maximiser les bénefices. Nous ne prendrons pas en compte la ≪ popularité ≫ du produit ou encore la répartition des unités produites les unes par rapport aux autres.

2.1 Modélisation

Soit n_i le nombre de produit i fabriqué.
Le coup fixe de production n'influant pas sur notre décision, nous considérerons uniquement le

coût variable de production.

Il est défini par la formule suivante :

$$CV(i) = n_i \times \left(\sum_{j=1}^{7} T_{i,j} \times \frac{C_{i,j}}{60} + \sum_{k=1}^{3} A_k \times Q_{k,i} \right)$$

Le chiffre d'affaire par produit est :

$$CA(i) = n_i \times V_i$$

Par conséquent le bénefice par produit se calcule de la manière suivante :

$$\begin{aligned} B(i) &= CA(i) - CV(i) \\ &= n_i \times \left(V_i - \sum_{j=1}^{7} T_{i,j} \times \frac{C_{i,j}}{60} + \sum_{k=1}^{3} A_k \times Q_{k,i} \right) \end{aligned}$$

La matrice permettant de calculer, à partir du vecteur colonne n du nombre de produits sortis de l'usine, le bénéfice d'*Optim* est donc la suivante :

$$\begin{aligned} M_B &= \left(V - \left(\left(T \times C^t \times \frac{1}{60} \right)^t + (A \times Q) \right) \right) \\ &\simeq \begin{pmatrix} 6.0667 & 11.9833 & 12.4333 & 9.5333 & 31.6500 & 27.9000 \end{pmatrix} \end{aligned}$$

Remarquons qu'elle donne explicitement le bénéfice unitaire pour chaque produit. instincitvement, le produit E est le plus intéressant.

Nous chercherons à maximiser la fonction linéaire correspondant à cette matrice, donc (pour prendre une forme plus standard) à minimiser son opposé.

2.2 Stratégie adoptée

En utilisant les outils Matlab, nous obtenons le résulat suivant :

```
1  N = linprog(Units, InfEqConstraints, InfEqValues);
```

$$n_{optimal} = \begin{pmatrix} 0.0000 \\ 20.4082 \\ 0.0000 \\ 0.0000 \\ 242.5000 \\ 94.1837 \end{pmatrix}$$

Du point de vue strictement comptable, le bénefice réalisable est 357.0919 ¤.

2.2.1 Interprétation

- Les produits (A), (C), (D) doivent être abandonnés.
- Le produit (E) doit être « surproduit » (en 242,5 exemplaires), tout comme le produit (F), mais dans une moindre mesure.

Ce résultat était (en partie) prévisible à partir de la matrice M_B, puisque le produit E est le plus rentable. Il faudra donc que l'entreprise **Optim**, pour optimiser le bénéfice, en produise le plus possible tout en utilisant intelligemment les matières premières et ressources humaines restantes.

3 Stratégie du point de vue du *Responsable d'atelier*

Le responsable d'atelier cherche à maximiser le nombre d'unités (toutes catégories confondues) produites sous les contraintes définies précedemment.
Autrement dit, seul la quantité de matières premières disponibles et le temps maximum de travail limitera la production.
Il n'intervient donc pour le moment aucune notion de bénéfice.

3.1 Modélisation

Soit N le nombre total de produits fabriqués. Trivialement, c'est la somme du nombre d'unités (A), (B), (C), (D), (E), et (F) produites :

$$N = \sum_{i=A}^{F} n_i \tag{5}$$

Par concéquent la matrice suivante représente la somme des différents produits :

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

La matrice des contraintes quant à elle, n'est pas modifiée. En effet, le responsable d'atelier d'**Optim** n'ajoute aucune contrainte à la production.

3.2 Stratégie adoptée

Instinctivement, il est évident qu'il faudra fabriquer au maximum l'unité consommant le moins de « temps machine » et de matières premières. On peux donc penser que (E) et (F) seront les produits les plus fabriqués.

En pratique, on obtient les résultats suivants :

```
1  N = linprog(Units, InfEqConstraints, InfEqValues);
```

$$M = \begin{pmatrix} 0 \\ 56.732 \\ 38.6928 \\ 0 \\ 182.4608 \\ 98.9216 \end{pmatrix}$$

On peut donc, au maximum, fabriquer 376.8072 unités, tous produits confondus.

3.2.1 Interprétation

Encore une fois :
- Les produits (A) et (D) doivent être abandonnés.
- Le produit (E) doit être « surproduit » (en 242,5 exemplaires), tout comme le produit (F), mais dans une moindre mesure.

Ceci confirme bien les résultats attendus, c'est à dire qu'il faut donner la priorité aux produits consommant le moins de ressources.
On remarque cependant qu'il est préférable de ne pas abandonner (C) cette fois-ci.

4 Stratégie du point de vue *commercial*

Le responsable commercial cherche à équilibrer le nombre d'unités de A, B, C (famille 1) et D, E, F (famille 2) afin que ces deux familles contiennent le même nombre d'unités (à ϵ unité(s) prés).
En prennant en compte les contraintes définies plus haut et y en ajoutant une nouvelle **contrainte d'équilibre** entre les deux familles de produits, il apparait que la meilleure solution serait de ne rien produire.
Encore une fois, cette solution n'est pas avantageuse.
Introduisons par conséquent une autre contrainte : nous allons essayer d'équilibrer les familles de produits tout en conservant un bénéfice maximal.

4.1 Modélisation

Soient :
- n_A le nombre de produits A usinés.
- n_B le nombre de produits B usinés.
- n_C le nombre de produits C usinés.
- n_D le nombre de produits D usinés.
- n_E le nombre de produits E usinés.
- n_F le nombre de produits F usinés.

Pour étudier l'équilibre entre les deux familles des produits on définira une fonction qui sera égale à la différence entre les quantités des produits provenant des deux familles, soit :

$$F = (n_A + n_B + n_C) - (n_D + n_E + n_F)$$

La contrainte sur l'équilibre va se traduire par l'essai de minimiser la fonction F, c'est à dire :

$$\begin{aligned} |F| &\leq \epsilon \\ \Leftrightarrow -\epsilon \leq F &\leq \epsilon \\ \Leftrightarrow -\epsilon \leq (n_A + n_B + n_C) - (n_D + n_E + n_F) &\leq \epsilon \\ Avec \quad \epsilon \to 0. & \end{aligned}$$

4.2 Stratégie adoptée

Comme indiqué plus haut nous allons essayer de trouver le nombre d'unités à produire de telle manière que :

1. Les famille 1 et 2 soient équilibrée
2. Le bénefice soit maximum

Pour cela :
- Nous fixons un certain bénéfice à atteindre
- Pour le bénéfice choisi, nous essayons de minimiser F.

Pour rester toujours dans une programmmation monocritère nous répèterons cette démarche un certain nombre de fois (pour des pourcentages différentes du benefice).
Nous pourrons donc tracer une courbe représentant la valeur de F en fonction du bénéfice atteint.

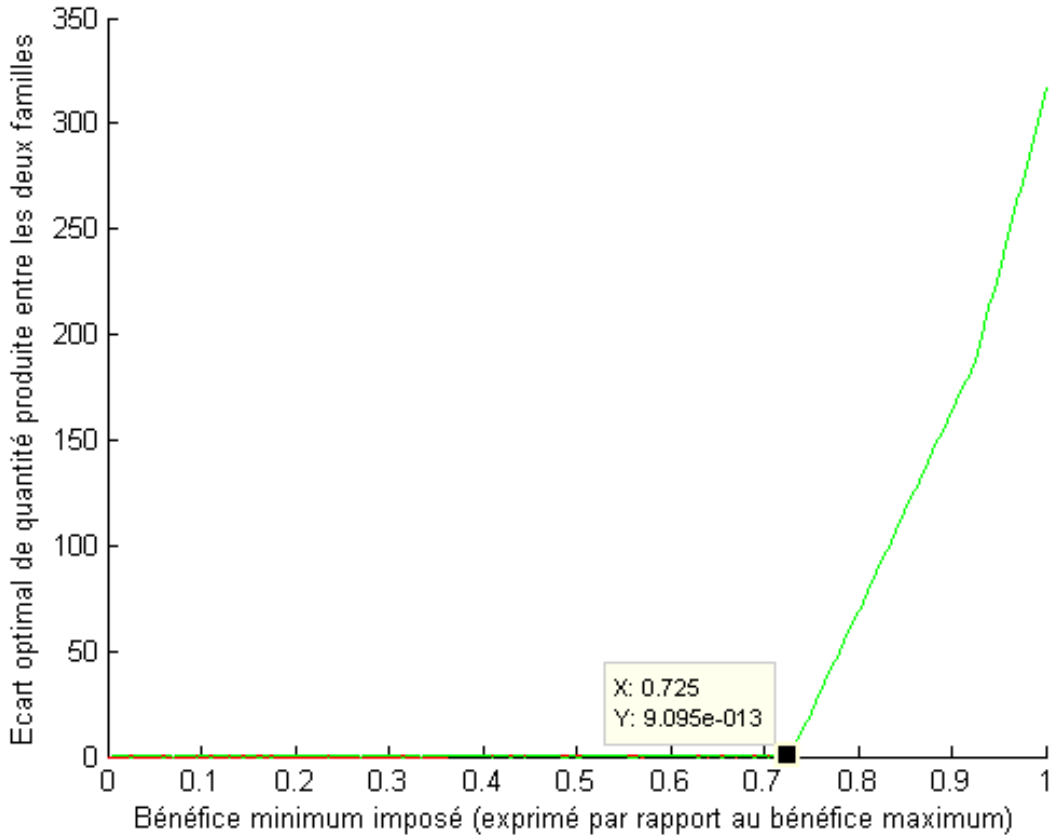
L'interprétation de cette courbe nous permettra de trouver le meilleur compromis.

Pour s'approcher de l'équilibre nous utilisons la fonction linprog de Matlab de la manière suivante :

```
1 linprog(F, [infEqConstraints;-Earnings], [infEqValues;-b]);
```

F correspond à la fonction qu'on essaie de minimiser. `[infEqConstraints;-Earnings]` et `[infEqValues;-b]` sont les matrices qui modelisent la prise en compte de toutes les contraintes générales auxquelles on a ajouté la contrainte de s'approcher d'un certain benefice. Pour obtenir une valeur de F qui s'approche de 0 nous effectuerons deux appels de linprog :
un en utilisant la fonction F et un en utilisant la fonction -F. Les contraintes restent les mêmes pour les deux appels.

Le graphe suivant est obtenu :



La courbe verte correspond aux valeurs minimales de F, la courbe rouge les valeurs minimales de -F.
On observe dès le début qu'on peut approcher les deux courbes de 0 différentes valeurs du benefice. Autrement dit, il est possible d'avoir un équilibre des benefices respectifs. Au bout d'un moment, les deux courbes commencent à s'écarter et l'équilibre est rompu.

C'est au point de rupture qu'est le meilleur rapport entre les deux contraintes (l'équilibre et le benefice obtenu).
Le meilleur compromis est donc obtenu autour de 72.5% du benefice maximum.
L'écart entre les deux familles de produits est : $9.095 * 10^{-13}$ ce qui est tres proche de 0 et donc en conformité avec notre objectif.

Les quantités de production pour chaque produit sont les suivantes :
- $n_A = 0{,}8888$
- $n_B = 114{,}9102$

- $n_C = 60{,}2321$
- $n_D = 0{,}2537$
- $n_E = 148{,}7430$
- $n_F = 27{,}0343$

Le benefice obtenu est de 7646,575 unités monétaires.

5 Stratégie du point de vue du *Responsable des stocks*

Le responsable des stocks cherche à minimiser la contenance du stock, *i.e.* le nombre de produits entreposés ajouté à la quantité de matières premières. Aucune contrainte particulière n'est ajoutée mais la recherche de l'optimum se fait sous les contraintes définies précédemment.

5.1 Modélisation

Soit $Stock(n_i)$ le nombre d'unités de stock nécessaires pour stocker les produits fabriqués et les matières premières nécessaires.
Cette fonction est évidemment la somme des produits fabriqués et de la quantité de matières premières nécessaire à leur fabrication.

Supposons qu'**un produit fabriqué correspond à une unité de stock.**

Ainsi, soient :
- n_i la quantité de produits usinés (pour chaque produit i)
- $Q_{j,i}$ la quantité de matière première par produit pour chaque produit i et chaque matière première j.

Alors :

$$Stock(n) = \sum_i (n_i + n_i \times \sum_j Q_{j,i}) \tag{6}$$

La représentation matricielle de cette fonction sera alors :

$$M_S = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} + (\begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \times Q) \tag{7}$$

5.2 Stratégie adoptée

Un résultat trivial est de ne fabriquer aucun produit :

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{8}$$

En effet, en l'absence de production, aucun stock n'est nécessaire. Ce résultat n'est pas satisfaisant.
Nous allons donc nous intéresser à un second critère : **les bénéfices de l'entreprise**.
Pour ce faire, nous allons nous utiliser les résultats obtenus en imposant un minimum de bénéfices.
En réalisant les calculs sur plusieurs échelons, nous obtenons un résultat linéaire par morceaux.
Le raisonnement adopté est similaire à celui développé dans la section 4.

5.2.1 Interprétation

De nombreuses valeurs sont équivalentes mathématiquement parlant et ne peuvent pas vraiment être départagées. Les choix possibles se situent dans la deuxième partie de la courbe :
- En dessous, les bénéfices sont trop bas.
- Au dessus, les besoins de stockage augmentent beaucoup plus que les bénéfices. Nous perdons alors de vu notre objectif initial, *i.e.* avoir le moins de stock possible.

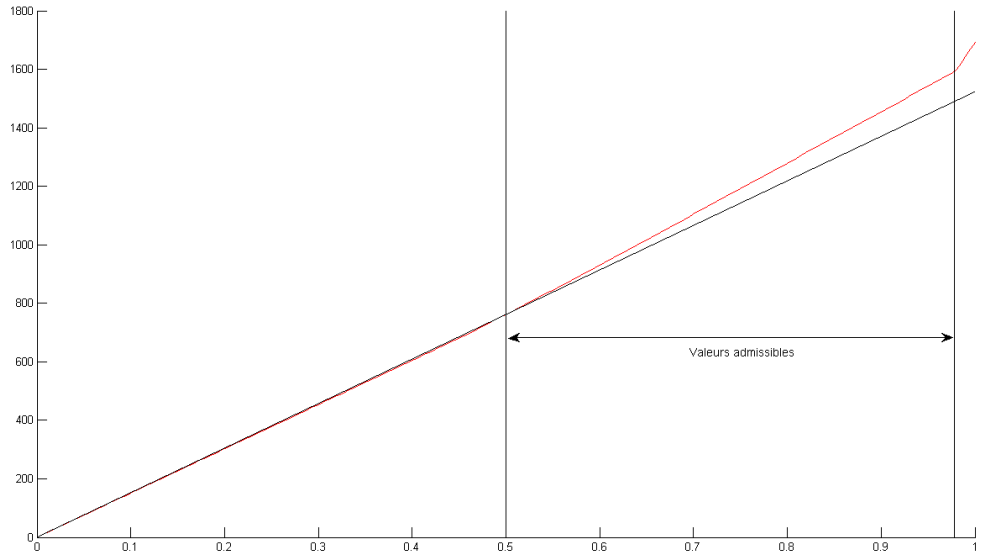


FIGURE 1 – Graphe de l'évolution des stocks en fonction du bénéfice

Nous obtenons donc une valeur comprise entre 50% et 98% de bénéfices. Pour 75% du bénéfice maximum, nous obtenons le nombre de produits suivants :

$$\begin{pmatrix} 1,91903382074088 \times 10^{-10} \\ 2,63753463514149 \times 10^{-10} \\ 1,89174897968769 \times 10^{-10} \\ 1,23691279441118 \times 10^{-10} \\ 124,634235411818 \\ 142,146305832495 \end{pmatrix} \tag{9}$$

Soit une quantité d'unités en stock de 1191,75640039347.

Cette étude de cas nous rappelle les limite d'une stratégie basée sur l'analyse d'un seul critère. En effet, en l'absence de contraintes assez fortes, le problème devient « indécidable » mathématiquement.

C'est pourquoi nous allons combiner les 4 études réalisées pour affiner nos stratégies et obtenir une solution :

- Mathématiquement claire
- Respectant un ensemble de contraintes plus grand, et par conséquent qui se concentre plus sur les contraintes propres à l'entreprise **Optim**.

6 Représentation graphique

Les graphiques ci-dessous représentent les résultats obtenus dans cette partie.

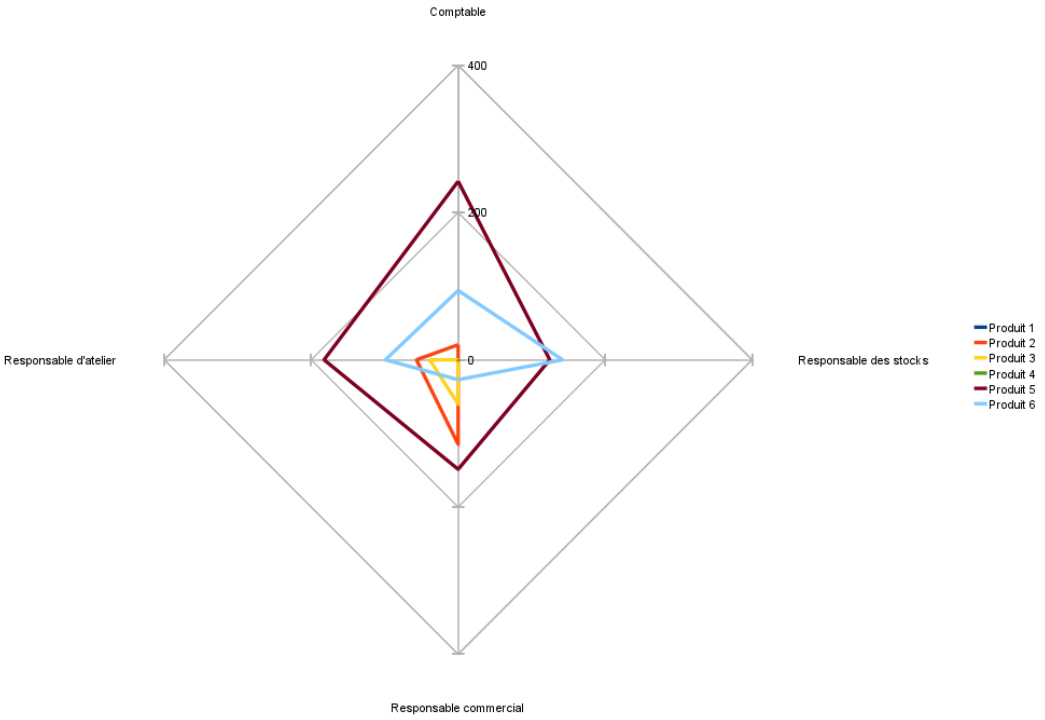


FIGURE 2 – Graphe radar représentant la quantité à produire par produit et par critère

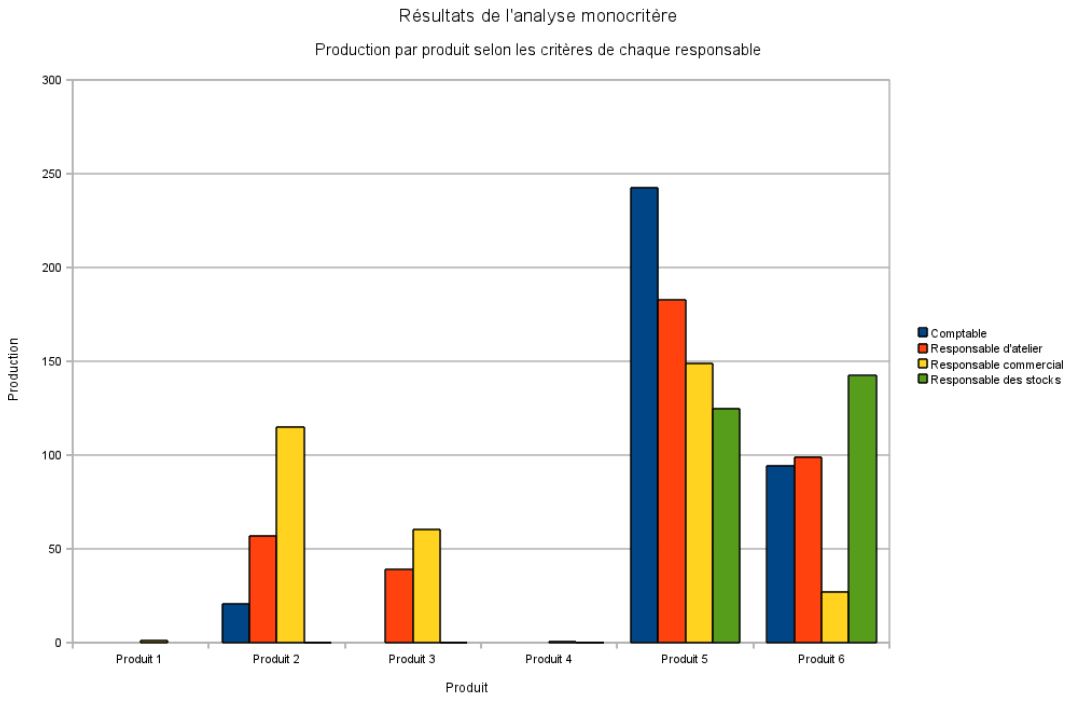


FIGURE 3 – Diagramme représentant la quantité à produire par produit et par critère

Deuxième partie

Programmation linéaire multicritère

7 Objectifs

L'objectif est de trouver une solution de compromis entre les différents responsables. Pour trouver une telle solution nous serons amenés à utiliser la programmation multicritère (PLM). Auparavant, dans la partie 1, nous avons trouvé un optimum pour chaque responsable indépendamment, ce qui nous conduit à un point de mire. Dans un monde parfait, ce point de mire respecterait les contraintes de chaque responsable. Nous devons donc voir si tel est le cas.

8 Recherche du point de départ

Si le point de mire est assez proche de l'ensemble des solutions acceptables, nous choisirons une solution proche de celle d'un responsable.

Sinon, nous allons calculer la satisfaction de chaque objectif, sachant qu'une solution a été retenue. Nous devrons alors définir des métriques, correspondant à cette satisfaction. Par exemple, pour le comptable, cette satisfaction sera exprimée par le ratio du bénéfice obtenue dans un solution par rapport au bénéfice maximal. Ensuite, nous choisirons comme point de départ la solution qui offre le plus de satisfaction à tout le monde, par exemple en utilisant une moyenne pondérée, dont la pondération sera basée sur *l'importance* de chaque critère.

9 Affinement de la solution

La solution trouvée précédemment peut sûrement être optimisée. Il peut être intéressant de perdre dans un critère, si cela nous fait gagner beaucoup dans un autre critère, d'autant plus si ce second critère est jugé plus *intéressant* que le premier.

10 Métriques utilisées

Cette section décrit les métriques utilisées pour caractériser une solution, du point de vue d'un cadre de l'entreprise. Les solutions pourront ainsi être comparées entre elles.

Comptable : La métrique utilisée sera le pourcentage du bénéfice par rapport au bénéfice maximum :

$$M_{Comptable} = \frac{B_S}{B_{max}} \times 100$$

Responsable d'atelier La métrique utilisée sera le pourcentage du nombre de produits fabriqués par rapport au nombre maximum :

$$M_{Atelier} = \frac{N_S}{N_{max}} \times 100$$

Responsable des stocks Pour élaborer la métrique de satisfaction pour le responsable des stocks nous opterons pour la fonction suivante, qui correspond à ce qui était précédemment annoncé dans la partie 1 (page 13).

Cette fonction est décrite par l'expression suivante :

$$M_{Stocks} = \begin{cases} \frac{x}{1192} \text{ si } x \in [0\,;\,1192] \\ x = 1 \text{ si } x \in [1192; 1559] \\ \frac{-x}{1192} + 1 + \frac{1559}{1192} \text{ si } x \in [1559\,;\,1691] \end{cases}$$

Cette fonction pourrait être justifiée par le fait que plus on s'éloigne des valeurs admises moins le responsable des stocks est satisfait. Les valeurs numériques présentes dans la formules viennent des résultats de la section 5.

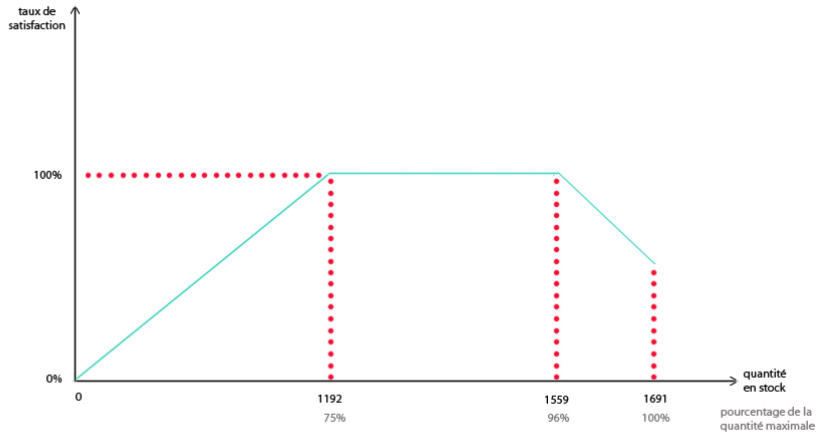


FIGURE 4 – Représentation graphique de la métrique pour le responsable des stocks.

Responsable commercial La métrique utilisée sera l'écart par rapport à un équilibre parfait. Si autant de produit de la famille de produit 1 (comprenant les produits A, B et C) que de la famille 2 (comprenant les produits D, E et F), la métrique sera à 100%. Si une seule famille de produit est fabriquée, la métrique devra valoir zéro.

Si F_1 (respectivement F_2 est le nombre de produit de la famille 1 (respectivement famille 2), la métrique sera :

$$M_{Commercial} = \left(1 - \frac{|F_1 - F_2|}{F_1 + F_2}\right) \times 100$$

11 Utilisation

Les résultats seront placés dans un tableau de ce type, qui qui permettra d'un seul coup d'œil de voir la meilleur des solutions. Ce tableau a été crée en calculant les métrique pour chaque solution des quatre responsables.

La case en rouge se lira par exemple :

« En suivant la volonté du responsable d'atelier, le comptable aura une satisfaction de 96.5498% »

	Comptable	Atelier	Stock	Commercial
Comptable	100.000%	94.7678%	88.9030%	11.4302%
Atelier	96.5498%	100.000%	77.7846%	50.6491%
Stock	74.1546%	70.8003%	99.9796%	0.00000%
Commercial	81.8653%	93.4330%	80.2626%	100.000%

On voit déjà que la solution du commercial contente, dans une certaine mesure, l'ensemble des responsables de l'entreprise, toutes les métriques sont supérieures à 80%.

12 Optimisation

Nous allons dans cette partie essayer de modifier les solutions précédentes pour trouver une solution qui tente de maximiser la satisfaction des différents responsables.

Le but premier d'une entreprise étant de faire du profit, et les meilleur manière de faire du profit étant souvent maximiser le bénéfice et d'avoir une bonne politique commerciale, nous allons donc favoriser ces critère (bénéfice et équilibrage des solutions), en leur appliquant un léger coefficient.

Nous allons par la même regarder l'évolution des autres critères lorsque ce coefficient évolue, c'est à dire recalculer les différentes métriques.

Il apparaît plausible qu'avoir un *stock non optimal* peut être acceptable si c'est pour avoir un bénéfice plus important, dans une certaine mesure. Une dégradation de la satisfaction du responsable des stock sera donc moins pénalisante, en terme de qualité de solution, qu'une baisse dans la satisfaction du commercial (produire sans vendre ne sert pas à grand chose, en terme de rentabilité).

De la même manière, produire un nombre de produit important peut être intéressant, mais cela ne doit pas aller à l'encontre du profit et de la mauvaise répartition de la production.

Nous pouvons donc donner des coefficients aux critères :

Responsable	Comptable	Atelier	Stocks	Commercial
Coefficient	1.2	0.8	0.8	1.2

TABLE 1 – Coefficients associés aux critères des différents responsables

Il faut cependant garder à l'esprit que ces coefficient devraient être ajustés par une plus grande étude du cas spécifique d'**Optim**, les informations dont nous disposons étant encore trop faibles pour effectuer une pondération efficace.

12.1 Méthodologie d'optimisation

Le procédé sera itératif, et tentera de contenter au mieux le comptable et le commercial (comme expliqué ci-dessus). Il s'agira donc de partir de la solution du responsable commercial (celle qui dispose des taux de satisfaction les plus haut), en essayant d'améliorer la métrique caractérisant la solution du responsable comptable, tout en restant dans le domaines des solutions possibles au vu des contraintes. Il faudra toutefois tâcher de ne pas trop dégrader les autres critères.

Une solution serait de *sacrifier* un peu d'équilibre dans les familles de produit pour fabriquer des produits plus rentables. D'après la partie 1, section 2, le produit qu'il faut fabriquer plus celui de type E.

12.2 Résultats

Les résultats sont assez peu intéressant, du fait de la faible latitude de changement des solutions, et parce que les produits en grande quantité dans la famille 1 ne sont pas fait à partir des même matériaux que le produit à produire dans la famille 2 (respectivement le produit B et le produit E).

L'algorithme, assez naïf, essaie d'enlever un produit de la famille 1 pour essayer de rajouter un produit de type E.

Ci dessous sont présenté les valeurs de satisfaction des quatre responsables , en enlevant du produit de type C, et en rajoutant un produit de type E.

Trois itérations sont possibles :

Itération	Comptable	Atelier	Stocks	Commercial
1	81.9901	93.4330	30.6925	99.4319
2	82.1148	93.4330	30.7764	98.8639
3	82.2396	93.4330	30.8603	98.2958

TABLE 2 – Résultats de l'exécution d'un algorithme d'optimisation

Nous choisirons donc la solution 3, qui est la plus équilibrée, à partir de la solution de départ que nous avons choisis.

Troisième partie

Analyse multicritère

Des deux parties précédentes, émergent 8 propositions de gestion de l'atelier. Le but de cette partie sera de sélectionner la meilleure solution en fonction des 4 critères présélectionnées.

Ces 4 critères sont :

g_1 : bénéfice

g_2 : équilibre commercial

g_3 : production

g_4 : gestion du stock

Pour réduire au maximum l'échéance, nous avons parallélisé au maximum les flux de travail. Nous avons donc dès le début du projet commencé à coder sous Matlab un algorithme de résolution indépendant des résultats des 2 premières parties.

13 Méthode choisie

La méthode de résolution choisie sera Électre III car elle donne plus de résultat qu'Électre I et Électre II. On pourras ainsi fournir au client la méthode sélectionné comme la plus optimale ainsi qu'une ou plusieurs méthodes alternatives.

14 Algorithme de la méthode

La résolution des méthodes d'Électre se base sur l'utilisation de deux matrices : la matrice de concordance et la matrice de discordance. Ces deux matrices contiennent des valeurs qui, respectivement, confirme ou infirme la supériorité d'une solution sur une autre.

14.1 Changement d'échelle

Avant de calculer ces matrices, les notes données dans la matrice de jugement peuvent variées suivant les coefficients attribués aux critère. Ainsi, si un critère à un coefficient 2 fois supérieur à un autre, les notes du coefficient le plus important seront répartie sur une échelle plus importante que celles du coefficient le moins important.

14.2 Matrices de concordances et de discordances

Les matrices de concordance et de discordance peuvent être calculé à partir de cette nouvelle matrice de jugement.

Soit n, le nombre de solutions, alors les matrices de discordances et de concordances sont carré de côté n.

Soit la $i$ème et la $j$ème solution, représenté par la $i$ème ligne et la $j$ème colonne.
En chaque case $\{i, j\}$ de la matrice de concordance, on trouve la valeur confirmant la supériorité de la solution i sur la solution j. Cette valeur est calculé comme la somme des poids des critères dans lesquels i domine j, divisé par la somme des poids de tous les critères.
À l'inverse, dans la matrice de discordance, les valeurs infirment la supériorité de i sur j. Elles sont calculés comme le plus grand écart positif entre les notes de j et les notes de i sur un critère donné.

14.3 Graphe de surclassement

À partir de ces deux matrices et en fonction des seuils de concordances et de discordance, on peut établir le graphe de surclassement. Pour qu'une solution i soit supérieur à une solution j, la concordance de la supériorité de i sur j doit être supérieur au seuil de concordance, et la discordance inférieur ou égal au seuil de discordance.

Le graphe de surclassement permet de visualiser les supériorités entres les solutions. On peut ainsi aisément se faire une idée du classement finale.

15 Mise en œuvre de la méthode

L'algorithme suivant est une implémentation de la méthode Électre III.
À partir de la matrice des jugements, il remet à l'échelle les notes des critères en fonction des poids.

```
for i = 1:n
        J(:,i) = ( (J(:,i) - (e/2)) * (w(i)/max(w))) + (e/2);
end;
```

Puis il calcule les matrices de concordance et de discordance.

```
for i = 1:n
    for j = 1:n
        if (i ~= j)
            C(i,j) = sum( (J(i,:)>=J(j,:)).*w )/sum(w);
            D(i,j) = max(max(J(j,:)-J(i,:)),0)/e;
        end;
    end;
end;
```

Pour finir, il établit la matrice des surclassement en fonction de ces deux dernières matrices.

```
S = ((C > s).*(D <= v));
```

Un graphe est ensuite généré à partir de cette matrice, pour faciliter la lecture du résultat. La meilleur solution apparait alors clairement en tête du graphe, et grâce à la méthode du classement et du classement inverse, on peut obtenir l'ordre des solutions.

16 Solution proposée

Dans un premier temps, sans pondérer les critères, on trie les solutions pour en tirer la plus avantageuse. Dans un second temps, on prendras en compte les poids des critères apporté par les résultats de la seconde partie.

16.1 Sans pondération

La matrice des jugements est la suivante :

	g_1	g_2	g_3	g_4
a	6	5	5	5
b	5	2	5	6
c	3	4	6	3
d	3	7	5	4
e	5	4	3	7
f	2	5	6	3
g	5	4	3	7
h	3	5	6	4

Après quelques tâtonnement, on utilise les seuils de concordance 0,7 et 0,9, et le seuil de discordance 0,4 pour générer un graphe de surclassement le plus clair possible, sans liens superflus.

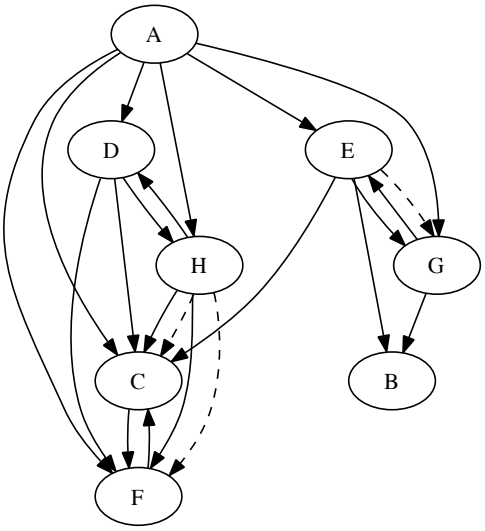


FIGURE 5 – Graphe des surclassement sans prise en compte des poids de chaque critère

Sur le graphe généré, on voit clairement que la la meilleur solution serait la solution A car c'est la seule qui n'est dominé par aucune autre. On voit également que certaines solutions sont équivalentes puisqu'elles se dominent entres elles. Les couples de solutions {D, H}, {E, G} et {C, F} sont des couples de solutions équivalentes. Les solutions F, C et B sont dominés, elles sont donc les moins intéressantes.

Hormis la solution A qui se démarque, il sera difficile d'établir un classement très juste car beaucoup de solutions sont équivalentes entre elles.

16.2 Avec pondération

Nous reprendrons les pondérations de la partie 2. Les pondérations sont :
g_1 : 1.2
g_2 : 1.2
g_3 : 0.8
g_4 : 0.8
La matrice des jugement après mise à l'echelle des notes devient :

	g_1	g_2	g_3	g_4
a	6	5	5	5
b	5	2	5.6	6.3
c	3	4	6.3	3.6
d	3	7	5	4.3
e	5	4	3.6	7.6
f	2	5	6.3	3.6
g	5	4	3	7.6
h	3	5	6.3	4.3

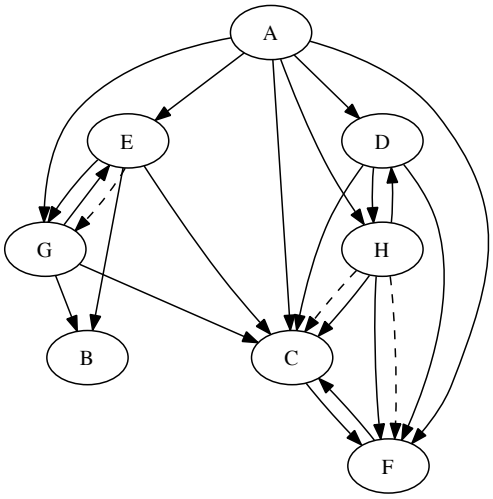


FIGURE 6 – Graphe des surclassement avec prise en compte des poids de chaque critère

On retrouve la solution A en tête, comme précédemment, et on voit que les positions n'ont pas changé, les surclassement sont presque identiques, seul un surclassement de G sur C est apparu.

17 Conclusion

La méthode d'Électre III permet de classer un grand nombre de solution suivant plusieurs critères de manière fiable. En pratique elle permet effectivement de tirer des groupes de solutions optimaux d'un ensemble vaste. Nous avons en effet réussi à isoler une solution optimale parmis les 8 proposés, mais il à été plus difficile d'établir un classement entre toutes les solutions.

Quatrième partie

Annexe

18 Code source

18.1 Mise en place des contraintes

```
%---------------------------------------------------
% RAW DATA
%---------------------------------------------------

% Indices - product mapping
a = 1;
b = 2;
c = 3;
d = 4;
e = 5;
f = 6;

Products = [a, b, c, d, e, f];

% Manufacturing time per machine, for each product. (Table 1)
% Usage : T(a,2), T(d,7), ...
T = [
        8, 0, 8, 2, 5, 5, 5;
        15, 1, 1, 10, 0, 5, 3;
        0, 2, 11, 5, 0, 3, 5;
        5, 15, 0, 4, 0, 12, 8;
        0, 7, 10, 13, 10, 8, 0;
        10, 12, 25, 7, 25, 0, 0
    ];

% Quantity of raw material needed for each product
% Usage : Q(1,a), Q(3,f), ...
Q = [
        1, 2, 1, 5, 0, 2;
        2, 2, 1, 0, 2, 1;
        1, 0, 3, 2, 2, 0
    ];

% Maximum stock of raw material
% Usage : S(1), S(2), ...
S = [
        350,   620,   485
    ];

% Sale price of each product
% Usage : V(a), V(f), ...
V = [
        20, 27, 26, 30, 45, 40
    ];

% Cost of each raw material
% Usage : A(1), A(2), ...
A = [
        3, 4, 2
    ];

% TODO
% Usage : C(1), C(4), ...
C = [
        2, 2, 1, 1, 2, 3, 1
    ];

% TODO
% Usage : J(a,2)
J = [
        6, 5, 5, 5;
        5, 2, 6, 7;
        3, 4, 7, 3;
        3, 7, 5, 4;
        5, 4, 3, 9;
        2, 5, 7, 3;
        5, 4, 2, 9;
        3, 5, 7, 4
    ];

% Weight of each factor.
```

```
% Usage : P(1), P(2), ...
P = [
        1.2, 1.2, 0.8, 0.8
    ];

%-------------------------------------------------
% BASE CONSTRAINTS
%-------------------------------------------------

InfEqConstraints = zeros(16,6);
InfEqValues = zeros(16,1);
OffSet = 0;

% Quantity of products must be >= 0
for i = Products,
    InfEqConstraints(i+OffSet,i) = -1;
    InfEqValues(i+OffSet) = 0;
end
OffSet = OffSet + 6;

% Work time is limited
for i = 1:7,
   for j = Products,
       InfEqConstraints(i+OffSet,j) = T(j,i);
   end
   InfEqValues(i+OffSet) = 2*8*60*5;
end
OffSet = OffSet + 7;

% We have limited stocks of raw material
for i = 1:3,
   for j = Products,
       InfEqConstraints(i+OffSet,j) = Q(i,j);
   end
   InfEqValues(i+OffSet) = S(i);
end
OffSet = OffSet + 3;
```

18.2 Stratégie comptable

```
function [ OptProduction, MaxEarnings ] = Comptable( )
%COMPTABLE Summary of this function goes here
%   Detailed explanation goes here

FetchData;

% This is the matrix corresponding to the earnings function
Earnings = (V - ( (T * C' ./ 60)' + (A * Q) ))

% We'll try to maximize the function, so we minimize the opposite
Earnings = -Earnings;

% Optimisation
OptProduction = zeros(size(Earnings,2),1)
OptProduction = linprog(Earnings, InfEqConstraints, InfEqValues);

% Effective earnings
MaxEarnings = -Earnings * OptProduction;
```

18.3 Stratégie atelier

```
function [ N ] = Atelier( )

FetchData;

% This is the matrix corresponding to the earnings function
Units = [1; 1; 1; 1; 1; 1];

% We'll try to maximize the function, so we minimize the opposite
Units = -Units;

```

```
% Optimisation
N = linprog(Units, InfEqConstraints, InfEqValues);
```

18.4 Stratégie commerciale

```
function [X1,Y1,NN1,X2,Y2,NN2] = ResponsableCommercial()
% Minimise la difference de production entre les familles des produits {A,B,C} et {D,E,F}
% pour toutes les valeurs possibles du benefice

%Prise en compte des constraintes generales
FetchData;

%La precision des iterations
steps=200;

%La fonction qu'on veut etudier et qui modelise la difference de production
%entre les deux familles des produits
F=[ones(1,3) -ones(1,3)];

%On approche F et -F de 0 et on cherche les minimums.
[X1,Y1,NN1]=linprog_earnings(0,1,steps,F,[InfEqConstraints;-F],[InfEqValues;0]);
[X2,Y2,NN2]=linprog_earnings(0,1,steps,-F,[InfEqConstraints;F],[InfEqValues;0]);

%Les valeurs obtenus pour F
Y1 = abs(Y1);
%Les valeurs obtenus pour -F
Y2 = abs(Y2);

%On construit le graphe des deux fonctions pour trouver la meilleure
%solution.
close;
hold on;
plot(X1,Y1,'Color',[1 0 0]);
plot(X2,Y2,'Color',[0 1 0]);
xlabel('Benefice minimum impose (exprime par rapport au benefice maximum)');
ylabel('Ecart optimal de quantite produite entre les deux familles');
hold off;


end
```

18.5 Stratégie stock

```
function result = get_remaining_stock(produits)
% Quantite de matiere premiere par produit :
MatPremParProduit = [
 1  2   1   5   0   2
 2  2   1   0   2   1
 1  0   3   2   2   0
];
%Quantite max de matiere premiere par produit
MatPremMax = [
    350 620 485
];

produits = produits';

result = MatPremMax - (produits * MatPremParProduit');

end
```

```
function result = compute_stock(input)
    remaining = get_remaining_stock(input);

    input = sum(input) + sum(remaining);

    if input < 1192 && input > 0
        result = (input / 1192) * 100;
    elseif input > 1192 && input < 1559
```

```
9           result = 100;
10      elseif input >= 1691 && input >= 1559
11          result = (-input/1192 + 1 + 1559/1192) * 100;
12      end
13  end
```

18.6 Optimisation multicritère

Calcul de la matrice de satisfaction :

```
1   result = ones(4);
2
3   % Qtt max comptable
4   QttProdMaxComptable = [0
5                          20.4082
6                          0
7                          0
8                          242.5
9                          94.1837];
10  % Qtt max atelier
11  QttProdMaxAtelier = [0
12                      56.732
13                      38.6928
14                      0
15                      182.4608
16                      98.9216];
17
18  % Qtt max commercial
19  QttProduitEquilibre = [0.8888
20                         114.9102
21                         60.2321
22                         0.2537
23                         148.7430
24                         27.0343];
25
26
27  % Qtt max stocks
28  QttMaxStock = [1.91903382074088e-10
29                 2.63753463514149e-10
30                 1.89174897968769e-10
31                 1.23691279441118e-10
32                 124.634235411818
33                 142.146305832495];
34
35  result(1,1:4) = compute_satisfaction(QttProdMaxComptable);
36  result(2,1:4) = compute_satisfaction(QttProdMaxAtelier);
37  result(3,1:4) = compute_satisfaction(QttMaxStock);
38  result(4,1:4) = compute_satisfaction(QttProduitEquilibre);
39
40
41  result
```

Calcul de l'optimisation :

```
1   function [ optimized ] = OptimisationPartie2( inputs )
2       % Quantite de matiere premiere par produit :
3       MatPremParProduit = [
4        1  2   1   5   0   2
5        2  2   1   0   2   1
6        1  0   3   2   2   0
7       ];
8       %Quantite max de matiere premiere par produit
9       MatPremMax = [
10          350 620 485
11      ];
12
13      result = ones(100,4);
14
```

```
15      stock = MatPremMax - (inputs' * MatPremParProduit');
16
17      % Pour chaque iteration, on enleve un produit de la famille 1, et
18      % on rajoute un produit de type E si on a les matieres premieres pour.
19      % A chaque fois, on recalcule les metriques.
20      for a=1:1:100
21          if stock(1,1) > MatPremParProduit(1,3)
22              if stock(1,2) > MatPremParProduit(2,3)
23                  if stock(1,3) > MatPremParProduit(3,3)
24                      % On ajoute un element au stock : il n'est plus
25                      % fabrique
26                      stock = stock + MatPremParProduit(1:3,3)';
27                      % On le retire en quantite
28                      inputs(3,1) = inputs(3,1) - 1;
29                  end
30              end
31          end
32           if stock(1,1) > MatPremParProduit(1,5)
33              if stock(1,2) > MatPremParProduit(2,5)
34                  if stock(1,3) > MatPremParProduit(3,5)
35                      % On fabrique un nouvel element : on me retire
36                      % du stock
37                      stock = stock - MatPremParProduit(1:3,5)';
38                      % On l'ajoute en quantite
39                      inputs(5,1) = inputs(5,1) + 1;
40                  end
41              end
42           end
43           % On recalcule les metriques
44          result(a,1:4) = compute_satisfaction(inputs);
45          %inputs'
46          %stock'
47      end
48
49      result
50      %stock'
51
52  end
```

Fonctions annexes :

```
1   function result = compute_satisfaction(inputs)
2
3   % Prix unitaires des produits
4   PuProduits = [20
5                 27
6                 26
7                 30
8                 45
9                 40];
10
11      % Qtt max comptable
12      QttProdMaxComptable = [0
13                             20.4082
14                             0
15                             0
16                             242.5
17                             94.1837];
18      % Qtt max atelier
19      QttProdMaxAtelier = [0
20                           56.732
21                           38.6928
22                           0
23                           182.4608
24                           98.9216];
25
26      % Qtt max commercial
27      QttProduitEquilibre = [0.888810041865781
28                             114.910246268340
29                             60.2321120650158
30                             0.253790723745169
31                             148.743054004414
32                             27.0343236470630];
```

```
33
34      % Qtt max stocks
35      QttMaxStock = [1.91903382074088e-10
36                     2.63753463514149e-10
37                     1.89174897968769e-10
38                     1.23691279441118e-10
39                     124.634235411818
40                     142.146305832495];
41
42      result = zeros(1,4);
43
44      result(1,1) = (sum((inputs .* PuProduits)) / sum((QttProdMaxComptable .* PuProduits)))
            * 100;
45      result(1,2) = (sum(inputs) / sum(QttProdMaxAtelier)) * 100;
46      result(1,3) = compute_stock(inputs);
47      result(1,4) = (1-abs(sum(inputs(1:3))-sum(inputs(4:6)))/sum(inputs)) * 100;
48  end
```

```
1   function result = compute_stock(input)
2       remaining = get_remaining_stock(input);
3
4       input = sum(input) + sum(remaining);
5
6       if input < 1192 && input > 0
7           result = (input / 1192) * 100;
8       elseif input > 1192 && input < 1559
9           result = 100;
10      elseif input >= 1691 && input >= 1559
11          result = (-input/1192 + 1 + 1559/1192) * 100;
12      end
13  end
```

18.7 Électre III

```
1   function [S1,S2,C,D] = electre3 (J,w,s1,s2,v)
2   % J = m x n judgment matrix
3   % w = n weight
4
5   n = size(J,1); % alternatives
6   m = size(J,2); % indicators
7   e = 10; % scale
8
9   graphname = 'graph.dot';
10
11  if (m ~= size(w))
12      error('Size␣doesn't␣match.');
13  end;
14
15  C = zeros(n);
16  D = zeros(n);
17
18  % APPLY NEW SCALE TO JUDGMENT MATRIX
19  for i = 1:m
20      J(:,i) = ( (J(:,i) - (e/2)) * (w(i)/max(w))) + (e/2)
21  end;
22
23  % EVALUATE AGREEMENT-DESAGREEMENT MATRICES
24  for i = 1:n
25      for j = 1:n
26          if (i ~= j)
27              C(i,j) = sum( (J(i,:)>=J(j,:)).*w )/sum(w);
28              D(i,j) = max(max(J(j,:)-J(i,:)),0)/e;
29          end;
30      end;
31  end;
32
33  % EVALUATE OUTCLASS MATRIX
34  S1 = ((C > s1).*(D <= v));
35  S2 = ((C > s2).*(D <= v));
36
37  % DRAW ORDER GRAPH
38  fid = fopen(graphname, 'w');
```

```
39  fprintf(fid, 'digraph␣G␣{\n');
40  [row1, col1] = find(S1);
41  [row2, col2] = find(S2);
42
43  for i = 1:n
44      fprintf(fid, [char(64+i), ';\n']);
45  end;
46  fprintf(fid, '\n');
47
48  k = size(row1,1);
49  for i = 1:k
50      fprintf(fid, [char(64+row1(i)), '␣->␣', char(64+col1(i)), ';\n']);
51  end;
52  fprintf(fid, '\n');
53
54  k = size(row2,1);
55  for i = 1:k
56      fprintf(fid, [char(64+row2(i)), '␣->␣', char(64+col2(i)), '[style␣=␣"dashed"];\n']);
57  end;
58  fprintf(fid, '\n');
59
60  fprintf(fid, '}');
61  fclose(fid);
```