Université de Lille

DM Modèles Linéaires

Les transistors d'Ornicar

2021-2022

Benedictus Kent **RACHMAT**

Hichem **Karfa**

L3 Informatique, Groupe 7

Rapport de DM Les transistors d'Ornicar

Question

Que peut-on dire d'un ensemble de transistors dont la dispersion vaut zéro ?

Rappel :

On définit la dispersion d'un ensemble de transistors T1, . . . , Tn, où Ti a un hfe égal à hi , et un vbe égal à vi, par la formule suivante :

```
D({T1, . . . , Tn}) = max( dh/MAXHFE , dv/MAXVBE)\newline
où \; \; dh = max (hi)_i − min (hi)_i \; et \; dv = max (vi)_i − min (vi)_i.
```

Conclusion :

Pour que la dispersion de l'ensemble soit de 0 il faut que dh et dv soient nuls, pour cela il faut que la valeur maximale et minimum des transistors soit égal, ce qui implique que tous les transistors ont les mêmes valeurs (ou qu'il n'y a qu'un transistor dans l'ensemble).

Exemple :

TRANS	hfe	vbe
T1	100	0.5
T2	100	0.5
T3	100	0.5

Question 1

Choix des paramètres

Pour déterminer les paramètres nous devons définir les données qui sont connues au début du programme.

Pour le choix des paramètres nous avons `MAXHFE` et `MAXVBE` qui représente la valeur maximum qu'hfe et vbe peut avoir.
La liste des transistors est représentée par un ensemble de transistors.
Chaque caractéristique d'un transistor est représenté par un paramètre indicé par cet ensemble, nous avons donc trois paramètres supplémentaires :

```
param hfe {TRANS};
param vbe {TRANS};
param type {TRANS} symbolic;
```

le paramètre type n'étant pas un entier il est désigné comme un symbole.

Choix des variables

Définissons maintenant les variables. Comme leur nom l'indique, ces données peuvent varier et ne sont pas connues au début du programme.

Il faut représenter la valeur minimum et maximum de hfe et vbe de l'ensemble des transistors.

Nous aurons donc quatre variables :

- `max_hfe`,`min_hfe`,`max_vbe` et `min_vbe`

Nous verrons comment les calculer dans la prochaine section (contrainte).

Pour calculer dh et dv nous aurons besoin de deux variables supplémentaires :

- `dh = max_hfe - min_hfe`
- `dv = max_vbe - min_vbe`

Choix des contraintes et résolution du problème

Revenons à nos variables `max_hfe`,`min_hfe` ,`max_vbe`, `min_vbe`.

Un des moyens de les calculer et d'utiliser directement la fonction `max` et `min` disponibles avec AMPL, mais ce faisant nous aurions à un moment des problèmes de linéarité des contraintes. Il faut donc trouver une autre solution permettant de calculer ces valeurs tout en ayant des contraintes linéaires :

la solution que nous avons opté est d'assigner une contrainte à ces valeurs, AMPL se chargera donc de trouver le maximum en respectant la contrainte.

Exemple :

Pour le maximum, on sait que `max_hfe` doit être supérieur ou égal à toutes les valeurs de hfe dans l'ensemble.

il faut alors le préciser dans AMPL dans une contrainte que nous nommons contrainte_max_vbe :

```
subject to contrainte_max_vbe {val in TRANS} :
    max_vbe >= vbe[val];
```

il suffit donc de faire la meme chose pour `max_hfe`. Concernant `min_hfe` et `min_vbe` il ne faut pas oublier d'inverser l'inégalité car on cherche cette fois-ci le minimum.

```
subject to contrainte_max_vbe {val in TRANS} :
    min_vbe <= vbe[val];
```

Pour la dispersion maximum nous allons avoir besoin de deux contraintes et d'une variable `dispersion_max`. Nous allons dire à AMPL comment calculer la dispersion pour hfe et vbe en indiquant que `dispersion_max` doit être supérieur ou égal a ces deux valeurs. `dispersion_max` sera donc assignée à deux contraintes et le solver devra les respecter toutes les deux.

```
subject to contrainte_dispersion_max_hfe {val in TRANS} :
    dispersion_max >= dh/MAXHFE;

subject to contrainte_dispersion_max_vbe {val in TRANS} :
    dispersion_max >= dv/MAXVBE;
```

Nous venons d'avoir une borne minimum pour la résolution du problème mais rien n'oblige AMPL de donner une valeur très grande qui n'est pas égale à nos deux dispersions.

Par exemple si la dispersion de hfe est de 0.5 et vbe et de 0.6 , `dispersion_max` Pourrait avoir une valeur de 1 et il respecterai tout de même les contraintes. Pour pallier à cela nous devons minimiser la valeur de `dispersion_max` pour qu'elle soit la plus petite possible tout en respectant les différentes contraintes.

```
minimize dispersion: dispersion_max;
```

Résultat et conclusion

Après chargement du modèle, nous lançons le solveur Minos et nous obtenons le résultat suivant :

```
 ampl: model annexe/exo1.ampl
MINOS 5.51: optimal solution found.
3 iterations, objective 0.6519189267
Adding  meminc=0.0998  to $minos_options might save time.
dispersion = 0.651919
```

La dispersion de l'ensemble des transistors est `0.651919` ce qui est un peu élevé. Nous verrons dans la prochaine question comment le réduire. Elle correspond donc à la dispersion des gains `hfe`.

Question 2

Choix des paramètres et des variables

Pour cette question nous nous sommes inspirée de l'annexe A. Dans l'annexe des étudiants pouvaient être affectés dans des créneaux avec une variable `est_affecte`. Cette problématique a beaucoup de points communs avec notre problème.

Pour exclure certains transistors nous avons choisi de créer une nouvelle variable `transistor_est_pris` cette variable sera indicée par les transistors et aura donc deux valeurs possibles :

```
 - 1 si le transistor n'est pas exclu
- 0 s'il est exclu
```

Nous aurons besoin aussi besoin d'un paramètre `E` indiquant le nombre de transistors à exclure.

Choix des contraintes et résolution du problème

Nous pouvons extraire deux contraintes de cette question.

Une permettant d'exclure `E` transistors au maximum et une autre permettant d'exclure les transistors.

Pour exclure les transistors nous utilisons donc la valeur de `transistor_est_pris`.

- Pour la recherche de `hfe_max` (et `vbe_max`) nous allons modifier la contrainte et ajouter seulement la multiplication de cette valeur par transistor_est_pris.

```
subject to contrainte_max_hfe {val in TRANS} :
max_hfe >= transistor_est_pris[val] * hfe[val];
```

Si transistor est pris vaut 1 cela ne change pas le calcul (1 * x = x). S'il n'est pas pris transistor est pris vaut 0 et le résultat sera de 0, la contrainte sera donc respectée de base qu'importe la valeur de hfe[val] et le maximum ne changera pas.

- Pour la recherche de `min_hfe` (et `vbe_min`) c'est un peu plus compliqué, multiplié la valeur de hfe[val] par `transistor_est_pris` ne suffit pas, en effet si cette valeur vaut 1 le résultat sera toujours bon, le problème vient dans le cas ou `transistor_est_pris` vaut 0, dans ce cas là le resultat sera de 0 et `min_hfe` sera donc contraint de positionner sa valeur à 0 sachant que le transistor est exclu, il faut donc trouver un moyen de ne pas impacter la contrainte dans le cas où le transistor n'est pas pris. Nous allons donc un peu modifier le calcul voici la nouvelle contrainte :

```
subject to contrainte_min_hfe {val in TRANS} :
  min_hfe <= transistor[val] * hfe[val] + (1 - transistor[val]) * MAXHFE;
```

Avec ce nouveau calcul, l'expression de droite sera égale à MAXHFE dans le cas où le transistor est exclu, le minimum d'hfe ne changera pas.

Il nous reste maintenant à trouver une contrainte permettant de n'exclure que `E` transistor, une idée pour cela est de faire la somme de (1 - la valeur transistor_est_pris) de chaque transistor

```
subject to contrainte_nb_max_exclusion:
    sum {val in TRANS} (1 - transistor_est_pris[val]) <= E;
```

Pour cet exercice on pouvait aussi inverser la valeur de `transistor_est_pris` et avoir une variable qui s'appellerai `transistor_est_exclu` à la place, les calculs seraient donc un peu différents mais le programme serait globalement le même. Nous avons choisi d'avoir une variable `transistor_est_pris` car cela était plus naturel pour nous.

Résultat et conclusion

Avant d'obtenir le résultat nous avons changé le solver pour Gurobi et nous avons obtenu les valeurs suivantes :

```
 ampl: model annexe/exo2.ampl
Gurobi 9.1.2: optimal solution; objective 0.5666726417
86 simplex iterations
1 branch-and-cut nodes
plus 5 simplex iterations for intbasis
dispersion = 0.566673

transistor [*] :=
 T1 1   T13 1   T17 1   T20 1   T24 1   T28 1   T31 0    T6 1
T10 1   T14 1   T18 1   T21 1   T25 1   T29 1   T32 1    T7 1
T11 0   T15 1   T19 1   T22 1   T26 1    T3 1    T4 1    T8 1
T12 1   T16 0    T2 1   T23 1   T27 1   T30 0    T5 1    T9 1
;
```

On a obtenu la dispersion (`0.5666726417`) ce qui est inférieur à la première question avec une différence de `0.085246285` Les transistors qui ont été exclus sont `T11, T16, T30, T31`.

T	HFE	VBE
T11	473.747183	0.620745
T16	477.658861	0.673297
T30	480.551455	0.530568
T31	446.039682	0.584432

On constate que les transistors avec les valeurs les plus extrêmes (ici en HFE) ont ont été exclus. Cela est logique car pour obtenir la dispersion on soustrait la valeur maximale et minimale qu'on divise par un paramètre. Donc pour minimiser la valeur de la dispersion Gurobi va exclure les valeurs les plus extrêmes, ici les valeurs maximales sont les valeurs les plus éloignées des autres valeurs, ce sont donc ces transistors qui vont être exclus.

dans le cas où il existe plusieurs valeurs différentes, la dispersion obtenue ne peut être que plus grande qu'à la question précédente car en éliminant certains transistors on ne peut que diminuer la distance entre ces derniers.

Question 3

Pour cette question il va falloir trouver un critère permettant de séparer les transistors en plusieurs paquets voici les différentes solutions que nous avons trouvées :

V1)

- première version, on pourrait trier en fonction du type de transistor, PNP ou NPN, le problème est qu'il n'y a que deux types de transistors pour 3 paquets, et en fonctionnant comme cela on pourrait avoir des paquets inégalement repartis et avoir une grande dispersion sur un paquet et une petite sur un autre, cette méthode a donc beaucoup trop d'inconvénient.

V2)

- Pour minimiser les paquets on pourrait se concentrer que sur hfe ou vbe en minimisant la valeur maximale d'un paquet, pour regrouper les paquets on pourrait prendre la valeur la plus haute et la valeur la plus base de l'ensemble des transistors pour le premier paquet, puis la 2e valeur la plus haute et la deuxième la plus basse sur un autre, cette manière de faire simplifie le problème et permet d'avoir des paquets de même longueurs.

V3)

- Une dernière solution est de minimiser la somme de la dispersion de chaque paquet pour garder une dispersion totale basse, pour regrouper les paquets nous allons séparer les transistors en trois groupes avec pour le premier groupe les X plus petites valeurs de HFE ou VBE, pour le second les Y valeurs intermédiaires et pour le dernier groupe les T transistor avec les valeurs les plus élevées de hfe, en faisant comme cela nous réduisons énormément la distance entre la valeur maximale et minimale de la dispersion de chaque paquet tout en ayant des paquets assez équivalents en dispersion, nous allons donc utiliser cette version pour la suite des exercices.

Question 4

Choix des paramètres et des variables

Pour diviser les transistors par groupes, nous avons donc besoin d'un paramètre `P` qui représentera le nombre de paquets. Nous avons aussi besoin d'un ensemble de paquets que nous nommons GROUP, finalement nous aurons besoin du nombre total de transistor, pour indiquer combien de transistor doit-il y avoir par paquets (nous verrons cela plus précisément dans la prochaine section).

Les valeurs maximales et minimales des transistors seront maintenant indicées par le groupe exemple :
`var max_hfe {g in GROUP}` .

Nous avons choisi de changer la variable dispersion pour avoir une dispersion par groupe, cela permettra d'afficher la dispersion par groupe.

Le point de plus important est la façon de pouvoir représenter des paquets dans un groupe. Notre solution a été d'avoir une valeur binaire, `gtransistor` indicé par un transistor et un groupe

- si `gtransistor` = 1 , le transistor est dans le groupe.
- si `gtransistor` = 0 , le transistor n'est pas dans le groupe.

Choix des contraintes et résolution du problème

Nous avons pensé à trois grandes contraintes pour pouvoir résoudre il faudrait :

- Calculer le `max_hfe/vbe` et `min_hfe/vbe` en prenant en compte le groupe dans lequel est le transistor.
- Fixer un transistor dans un groupe pour qu'il soit dans un et un seul groupe de transistor.
- Fixer un transistor dans un groupe pour qu'il soit dans un et un seul groupe de transistor.

Modélisation :

Dans le même état d'esprit que `est_exclu` une expression permet de séparer les transistors dans des groupes en multipliant la valeur du transistor(hfe / vbe) par `gtransistor` .

```
subject to contrainte_max_hfe {g in GROUP, val in TRANS}  :
    max_hfe [g] >= gtransistor[g,val] * hfe[val];
```

Pour récupérer le minimum on fait attention à la valeur que doit prendre le transistor quand il n'est pas dans le groupe :

```
subject to contrainte_max_vbe {g in GROUP, val in TRANS} :
    max_vbe [g] >= gtransistor[g,val] * vbe[val];
```

Pour fixer un transistor dans un groupe, il faut vérifier que la somme gtransistor[g,val] pour chaque groupe soit égal à 1.

Il existe différents moyens de vérifier que les groupes ont un nombre de transistors égale. Par exemple en contraignant le nombre de transistors d'un paquet à être plus petit que le cardinal de l'ensemble des paquets par le nombre de paquets + 1.

Résultat et conclusion

à partir des données ci-dessous, nous pouvons voir que le deuxième groupe a la dispersion la plus faible et le troisième à la plus grande dispersion, bien que ce dernier à une dispersion un peu différente la dispersion reste globalement la même avec comme avantage d'avoir une dispersion basse.

```
 ampl: model annexe/exo4.ampl
Gurobi 9.1.2: optimal solution; objective 0.5818244
3678 simplex iterations
47 branch-and-cut nodes
plus 15 simplex iterations for intbasis
:   max_hfe   max_vbe    min_hfe    min_vbe  dispersion    :=
1   480.551   0.678436   380.632    0.511903   0.166533
2   188.987   0.627137    89.4001   0.505002   0.165978
3   369.713   0.699465   220.125    0.505002   0.249314
;

gtransistor [*,*] (tr)
:     1   2   3    :=
T1    0   0   1
T10   0   1   0
T11   1   0   0
T12   1   0   0
T13   0   1   0
T14   0   0   1
T15   1   0   0
T16   1   0   0
T17   0   1   0
T18   0   1   0
T19   1   0   0
T2    0   1   0
T20   0   0   1
T21   0   1   0
T22   1   0   0
T23   0   1   0
T24   0   0   1
T25   0   1   0
T26   0   1   0
T27   1   0   0
T28   0   0   1
T29   0   0   1
T3    1   0   0
T30   1   0   0
T31   1   0   0
T32   0   0   1
T4    0   0   1
T5    0   1   0
T6    0   0   1
T7    0   1   0
T8    1   0   0
T9    0   0   1;
```

Nous avons organisé et trié les groupes par ordre croissant pour mieux visualiser le résultat :

Groupe 1 (Valeurs plus élevées)	Groupe 2 (Valeurs plus basse)	Groupe 3
T15 380.631659	T13 89.400099	T9 220.125258
T3 388.349744	T10 101.105138	T28 230.998222
T22 392.307125	T26 121.435795	T32 242.383656
T19 400.910157	T18 131.327049	T24 250.582394
T12 414.865893	T21 131.701247	T29 304.763147

Groupe 1 (Valeurs plus élevées)	Groupe 2 (Valeurs plus basse)	Groupe 3
T27 423.506906	T25 156.313272	T6 330.622531
T8 429.403684	T5 159.318987	T14 338.676073
T31 446.039682	T7 160.220311	T4 345.246256
T11 473.747183	T23 164.524036	T20 354.565503
T16 477.658861	T2 172.445927	T1 369.713399
T30 480.551455	T17 188.986802	

Cette solution permet donc de réduire la dispersion totale, qui passe à `0.5818244` sans pour autant exclure un certain nombre de transistors.

Question 5

Pour la question 5 il s'agit maintenant de ne plus découper les transistors en paquet. à la place on veut avoir des pairs de transistor, une solution est alors d'affectés deux transistors dans une paire :

```
subject to contrainte_transistor_pair {val in TRANS} :
    sum {(p1,p2) in COUPLE} pair[val,p1,p2] = 1;
```

Pour le critère ici l'objectif est différent, Nous avons choisi de maximiser la somme des transistors affectés dans une paire, en effet on ne cherche plus à maximiser la dispersion en priorité, la quantité prime donc sur la qualité. il faut aussi penser à gérer la dispersion des paquets, chaque paquet doit avoir une dispersion inférieure ou égale à 0.12 il faut donc ajouter une contrainte pour cela :

```
subject to contrainte_dispersion {(p1,p2) in COUPLE}:
    dispersion [p1,p2] <= DISPERSION_MAX;
```

Une des difficultés majeures a été de gérer la modélisation avec un minimum de contraintes, mais nous n'avons pas réussi à résoudre cette contrainte néanmoins nous comprenons notre erreur notre modélisation n'est pas assez performante pour pouvoir gérer les différents critères. Même en étant bloqué nous comprenons l'idée générale de cette question.