CHAPITRE 15 (BONUS)

Jeux et casse-tête

15.1 Introduction

Les applications de la programmation linéaire n'échappent pas aux jeux et aux énigmes. Ce chapitre présente trois applications amusantes, dont une bien connue. Pour certains de ces problèmes, il n'y a pas de fonction-objectif, ou alors elle est sans objet. Le but de la résolution est de trouver une solution satisfaisant un ensemble de contraintes particulières correspondant à l'énoncé.

Les problèmes 15.2 et 15.3 sont deux des nombreux problèmes de la page Web de Martin Chlond (*http://www.chlond.demon.co.uk/academic/puzzles.html*). Cette page web présente aussi les solutions formulées à l'aide du langage de modélisation Mosel de FICO (il s'agit du langage associé au solveur XPRESS).

On trouvera dans ce chapitre un jeu de grille et de jetons que l'on peut résoudre à la main, un partage équitable et difficile de tonneaux de vins et, pour finir, le problème très connu des *n* reines.

15.2 Grille et jetons

15.2.1 Problème

Partant de la grille carrée de taille 4×4 recouverte par seize jetons de la figure 15.1, comment enlever six jetons en laissant un nombre pair de jetons dans chaque ligne et chaque colonne de la grille ?

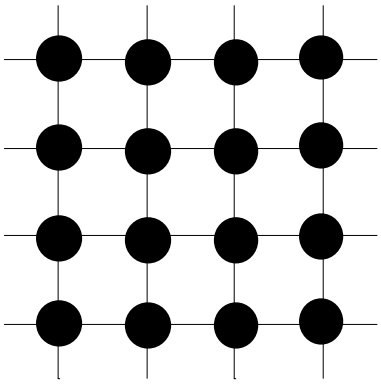


Figure 15.1 – Grille de départ

15.2.2 Modélisation

La constante *n* représente le nombre de jetons maximal que l'on peut placer en colonne ou en ligne. *P* désigne le nombre de jetons à laisser sur la grille. Une variable binaire x_{ij} est fixée à 1 si la case (*i*,*j*) est couverte par un jeton, 0 sinon. Sur la grille finale, *P* jetons doivent être positionnés, contrainte (1).

$$(1) \quad \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = P$$

Pour vérifier qu'il y a un nombre pair de jeton par ligne ou par colonne, deux types de variables qui représentent la demi-somme des jetons sont nécessaires. NL_i (resp. NC_i) représente la moitié de la somme des jetons de la ligne *i* (resp. de la colonne *i*). Pour être sûr d'obtenir un nombre de jetons pair, il suffit que ces deux types de variables soient entiers. Les contraintes (2) et (3) permettent d'exprimer ces demi-sommes pour les lignes et les colonnes. Les contraintes (4), (5) et (6) définissent le domaine d'appartenance des variables. Comme il a été mentionné en introduction de ce chapitre, il n'y a pas de fonction-objectif pour ce problème, mais seulement un ensemble de contraintes à satisfaire.

$$(1) \quad \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = P$$

$$(2) \quad \forall i = 1\ldots n : \sum_{j=1}^{n} x_{ij} = 2.NL_i$$

$$(3) \quad \forall j = 1\ldots n : \sum_{i=1}^{n} x_{ij} = 2.NC_j$$

$$(4) \quad \forall i = 1\ldots n, \forall j = 1\ldots n : x_{ij} \in \{0,1\}$$

$$(5) \quad \forall i = 1\ldots n : NL_i \in \mathbb{N}$$

$$(6) \quad \forall j = 1\ldots n : NC_j \in \mathbb{N}$$

15.2.3 Traduction en Excel

Le classeur *C15-Jetons* contient la feuille de calcul qui traduit le modèle mathématique en Excel. La feuille est organisée de la façon suivante. Le premier bloc va servir à vérifier le nombre de jetons à laisser sur la grille et le nombre de jetons actuellement présents sur la grille. Le tableau suivant présente la grille avec les cellules B7:E10 qui seront les variables x_{ij}. Pour chaque colonne et chaque ligne de cette grille, nous avons rajouté les informations sur le nombre de jetons présents, une variable correspondant à NL_i ou NC_j et une cellule qui sera le double de la variable précédente.

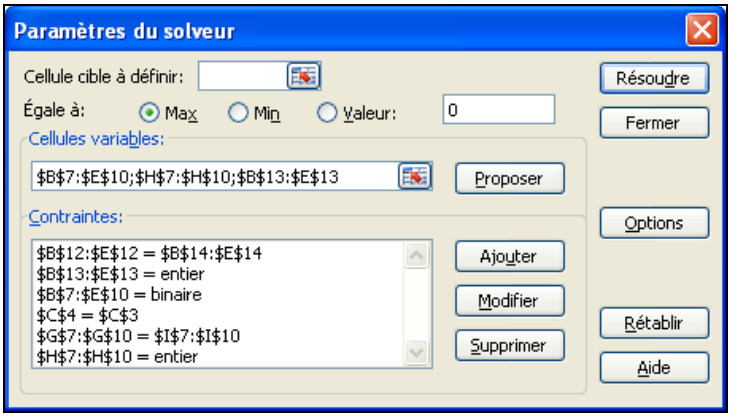
	A	B	C	D	E	F	G	H	I
1	C15-Jetons : Grille et jetons								
2									
3	Jetons à laisser		10						
4	Jetons restants		10						
5									
6	Grille	1	2	3	4		Jetons	Entier	Entier x 2
7	1	1	1	1	1		4	2	4
8	2	1	1	0	0		2	1	2
9	3	1	0	0	1		2	1	2
10	4	1	0	1	0		2	1	2
11									
12	Jetons	4	2	2	2				
13	Entier	2	1	1	1				
14	Entier x 2	4	2	2	2				

Les formules de la feuille sont alors les suivantes :

- *Nombre de jetons sur la grille.* La cellule C4 reçoit la formule "=SOMME(B7:E10)".
- *Nombre de jetons par ligne et par colonne.* La cellule G7 reçoit la formule "=SOMME(B7:E7)" qui compte le nombre de jetons dans la ligne courante. Cette formule est recopiée dans les cellules G8:G10. La cellule B12 reçoit la formule "=SOMME(B7:B10)" qui compte le nombre de jetons par colonne. Cette formule est étendue aux cellules C12:E12. L'ensemble de ces formules correspond aux membres de gauche des contraintes (2) et (3).
- *Nombre pair de jetons par ligne et colonne.* La formule "=2*H7" est placée dans la cellule I7 et recopiée dans les cellules I8:I10. De même la formule "=2*B13"est placée en B14 et recopiée en C14:E14. Ces formules correspondent aux seconds membres des contraintes (2) et (3).

Il reste maintenant à mettre en place les informations de la boite de dialogue du solveur. Pour ce problème, il n'y a pas de cellule cible puisqu'il n'y a pas d'objectif. Ceci est parfaitement autorisé par le solveur, qui va se contenter de trouver une solution vérifiant toutes les contraintes. Les cellules variables sont les cellules de la grille B7:E10 (variables x_{ij}), les cellules H7:H10 (variables NL_i) et les cellules B13:E13 (variables NC_j).

La première ligne dans le bloc des contraintes correspond aux contraintes (3). La seconde ligne impose aux variables NC_j d'être entières, contrainte (6), la troisième ligne aux variables x_{ij} d'être binaires, contrainte (4) et la quatrième ligne contraint le nombre de jetons qu'il devra rester sur la grille au final, contrainte (1). La cinquième ligne correspond aux contraintes (2) et la dernière ligne aux contraintes (5). Comme d'habitude, il ne faut pas oublier d'activer les options du solveur : *Modèle supposé linéaire* et *Supposé non-négatif*.



15.2.4 Résultats

La résolution Excel est immédiate. La grille présentée dans la feuille Excel permet de visualiser le placement des jetons rapidement (une cellule avec 1 correspond à la présence d'un jeton et avec 0 à l'absence de jeton). Cette solution est représentée sur la figure 15.2, mais elle n'est pas unique : il en existe d'autres pouvant se déduire par symétrie ou permutations de lignes ou de colonnes.

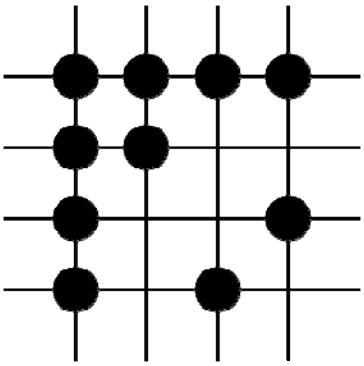


Figure 15.2 – Solution de l'énigme

15.3 Les tonneaux de vins

15.3.1 Problème

Un fermier décide de vider sa cave avant de vendre ses terres et prendre sa retraite. Sans enfants, il décide tout de même de partager ses 46 tonneaux de vin entre ses 5 neveux. Sur les 45 premiers tonneaux, 9 sont pleins, 9 aux trois-quarts pleins, 9 à moitié plein, 9 pleins au quart et 9 sont vides. D'où cinq taux de remplissage différents. Le dernier tonneau est rempli d'un très bon vin qu'il désire céder au plus intelligent de ses neveux. Il le cédera donc à celui qui saura partager le vin des 45 premiers tonneaux de façon que les neveux repartent chacun avec le même nombre de tonneaux et le même volume de vin. Il ajoute deux difficultés supplémentaires : les nombres de tonneaux alloués à chaque neveu et ayant le même taux de remplissage doivent être tous différents, et chaque neveu doit avoir au moins un tonneau de chaque taux !

15.3.2 Modélisation

Le nombre de neveux est noté n et le nombre de tonneaux de types différents noté T. Un ensemble de constantes r_t représente les différents remplissages des tonneaux, r_1 correspondra aux tonneaux pleins, r_2 aux tonneaux pleins aux trois-quarts, etc. Les constantes auront donc les valeurs suivantes : $r_1 = 1$, $r_2 = 0{,}75$, $r_3 = 0{,}5$, $r_4 = 0{,}25$ et $r_5 = 0$. Des variables x_{it} préciseront le nombre de tonneaux de type t que recevra le neveu i. Chaque neveu va recevoir neuf tonneaux (contrainte (1)) et les neuf tonneaux de chaque type doivent trouver preneur (contrainte (2)).

$$(1) \quad \forall i = 1 \ldots n : \sum_{t=1}^{T} x_{it} = 9$$

$$(2) \quad \forall t = 1 \ldots T : \sum_{i=1}^{n} x_{it} = 9$$

Le partage doit être équilibré en termes de volume de vin. Chacun des neveux doit donc recevoir l'équivalent de quatre tonneaux et demi en vin. La contrainte (3) permet de vérifier cet équilibre dans la distribution.

$$(3) \quad \forall i = 1 \ldots n : \sum_{t=1}^{T} r_t \times x_{it} = 4{,}5$$

La dernière difficulté ajoutée par le fermier est exprimée par deux contraintes (4) et (5). Elle nécessite l'introduction de coefficients c_1 à c_5 qui ont les valeurs 10 000, 1 000, 100, 10 et 1. L'idée de la contrainte (4) est de réaliser le produit scalaire p_i du vecteur des coefficients et du vecteur du nombre de tonneaux de différents types pour un neveu i. La contrainte (5) créé ensuite un ordre entre les différents produits scalaires, en imposant qu'il y ait au moins une unité de différence entre deux produits scalaires. La contrainte (6) impose aux variables x_{it} d'être entières et comprises entre 1 et 5.

(1) $\forall i=1\ldots n: \sum_{t=1}^{T} x_{it} = 9$

(2) $\forall t=1\ldots T: \sum_{i=1}^{n} x_{it} = 9$

(3) $\forall i=1\ldots n: \sum_{t=1}^{T} r_t \times x_{it} = 4{,}5$

(4) $\forall i=1\ldots n: \sum_{t=1}^{T} c_t . x_{it} = p_i$

(5) $\forall i=2\ldots n: p_i - p_{i-1} \geq 1$

(6) $\forall i=1\ldots n, \forall t=1\ldots T: x_{it} \in \{1,2,3,4,5\}$

(7) $\forall i=1\ldots n: p_i \geq 0$

15.3.3 Traduction en Excel

Le classeur *C15-Tonneaux* décrit le problème. La première partie présente les informations dont nous allons avoir besoin pour le calcul des volumes attribués et les coefficients pour les contraintes (4) et (5). On trouve en haut à droite, le nombre total de tonneaux attribué à chaque neveu et le volume de vin qu'il devra obtenir au final. Le tableau central contient le nombre de tonneaux de chaque type que nous allons attribuer à chaque neveu (les variables x_{it}). Pour chaque ligne on va retrouver le calcul du nombre de tonneaux affecté, le volume de vin contenu dans ces tonneaux, les coefficients p_i et le membre de gauche de la contrainte (5).

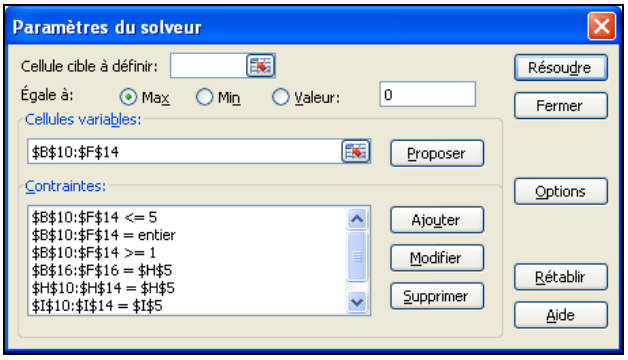
	A	B	C	D	E	F	G	H	I	J	K
1	C15-Tonneaux : Les tonneaux de vins										
2											
3		Tonneaux						Nombre de	Volume		
4		Plein	3/4 plein	1/2 plein	1/4 plein	vide		chaque type	par neveu		
5	Volume	1	0,75	0,5	0,25	0		9	4,5		
6	Coefficient	10000	1000	100	10	1					
7											
8		Tonneaux									
9	Neveux	Plein	3/4 plein	1/2 plein	1/4 plein	vide		Total	Volume	Pi	Pi-Pi-1
10	1	1	2	3	2	1		9	4,5	12321	
11	2	1	3	1	3	1		9	4,5	13131	810
12	3	2	1	3	1	2		9	4,5	21312	8181
13	4	2	2	1	2	2		9	4,5	22122	810
14	5	3	1	1	1	3		9	4,5	31113	8991
15											
16	Total	9	9	9	9	9					

Les formules de la feuille sont les suivantes :

- *Nombre de tonneaux affecté à chaque neveu.* Il est calculé par la formule "=SOMME(B10:F10)" que l'on place dans la cellule H10 et que l'on recopie dans les cellules H11:H14.

- *Affectation de tous les tonneaux.* On calcule le nombre de tonneaux de chaque type affecté par la formule "=SOMME(B10:B14)" placée en B16 et recopiée en C16:F16.
- *Calcul du volume de vin par neveu.* Pour ce calcul, on place dans la cellule I10 la formule "=SOMMEPROD(B10:F10;$B$5:$F$5)" et on la copie dans I11:I14.
- *Calcul des coefficients p_i.* La formule "=SOMMEPROD(B10:F10;$B$6:$F$6)" est placée dans la cellule J10 et recopiée dans les cellules J11:J14.
- *Calcusl pour les contraintes* (5). On calcule la valeur de $p_i - p_{i-1}$ par la formule "=J11-J10"que l'on place dans la cellule K11 et on recopie cette formule dans les cellules K12:K14.

Il ne reste plus qu'à remplir la boite de dialogue du solveur. Comme dans le problème précédent, il n'y a pas de fonction-objectif donc pas de cellule cible. Les cellules variables sont dans B10:F14. Dans le bloc des contraintes, les trois premières lignes correspondent aux contraintes (6). Les deux lignes suivantes sont les contraintes (2) et (1) respectivement. La ligne du bas correspond aux contraintes (3) sur le volume de vins affecté à chaque neveu. Enfin, les contraintes (5) sont décrites par l'expression K11:K14 $\geq$ 1, mais cette expression est masquée dans la copie-écran. Attention à ne pas oublier de mettre les options pour le solveur : *Modèle supposé linéaire* et *Supposé non-négatif*.



15.3.4 Résultats

Le tableau 15.1 indique le nombre de tonneaux de chaque type pour chacun des neveux.

Tableau 15.1 – Distribution des tonneaux

	Tonneaux				
Neveu	Plein	3/4	Moitié	1/4	Vide
1	1	2	3	2	1
2	1	3	1	3	1
3	2	1	3	1	2
4	2	2	1	2	2
5	3	1	1	1	3

15.4 Les *n* reines

15.4.1 Problème

Sur un échiquier de taille $n \times n$, comment placer un nombre maximal de reines tel qu'aucune des reines ne puisse en attaquer une autre ? On rappelle qu'une reine peut attaquer toutes les cases de la colonne, de la ligne et des diagonales où elle est placée.

15.4.2 Modélisation

Le nombre de cases du côté de l'échiquier sera noté n. Une variable binaire c_{ij} vaudra 1 si la case (i,j) de l'échiquier est couverte par une reine, 0 sinon. La fonction-objectif (1) consiste à maximiser le nombre de reines placées sur l'échiquier. Il ne doit pas y avoir de prise en ligne et en colonne (contraintes (2) et (3)), c'est-à-dire qu'au plus une case est occupée par une reine, en ligne et en colonne.

$$(1) \quad \text{Max} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}$$

$$(2) \quad \forall i=1\ldots n: \sum_{j=1}^{n} c_{ij} \le 1$$

$$(3) \quad \forall j=1\ldots n: \sum_{i=1}^{n} c_{ij} \le 1$$

Le même type de contraintes doit être mis en place pour les diagonales comprenant au moins deux cases. La difficulté réside uniquement dans le jeu de balayage des indices. Les contraintes (4) à (7) traitent successivement des diagonales allant de haut en bas pour chaque ligne, puis de haut en bas pour chaque colonne, de bas en haut pour chaque ligne et enfin de bas en haut pour chaque colonne.

$$(4) \quad \forall j=1\ldots n-1: \sum_{k=j}^{n} c_{k-j+1,k} \le 1$$

$$(5) \quad \forall i=2\ldots n-1: \sum_{k=i}^{n} c_{k,k-i+1} \le 1$$

$$(6) \quad \forall j=2\ldots n: \sum_{k=1}^{j} c_{k,j-k+1} \le 1$$

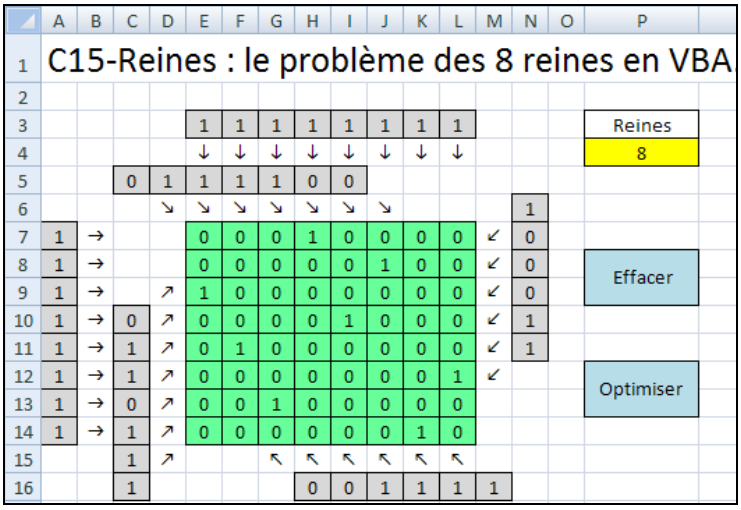
$$(7) \quad \forall i=2\ldots n-1: \sum_{k=i}^{n} c_{k,n-k+i} \le 1$$

Soit par exemple les contraintes (6) qui examinent les diagonales de bas en haut en partant de la ligne 1 : pour la ligne 3 (j = 3), la contrainte se traduit par $c_{13} + c_{22} + c_{31} \le 1$. Le modèle complet reprend les contraintes (1) à (7) en y ajoutant les contraintes (8) pour les variables binaires c_{ij}.

$$(8) \quad \forall i=1\ldots n, \forall j=1\ldots n: c_{ij} \in \{0,1\}$$

15.4.3 Traduction en Excel

Comme Excel n'offre pas de fonctions pour calculer des sommes sur des diagonales, nous avons utilisé VBA pour éviter une saisie fastidieuse des formules et du modèle. Le classeur *C15-Reines* contient la feuille de calcul et le code VBA développé. La feuille est organisée de la façon suivante. On retrouve le dessin de l'échiquier et autour les blocs qui permettront de calculer les formules pour les lignes, les colonnes et les diagonales. En haut à droite on retrouve l'objectif, le nombre de reines à placer sur l'échiquier.



Pour résoudre ce problème, nous allons écrire deux macros *EFFACER_SOLUTION* et *RESOUDRE_PROBLEME* qui vont non seulement placer les formules, mais aussi remplir la boite de dialogue du solveur et résoudre le problème. Ces macros sont associées aux boutons *Effacer* et *Optimiser* de la feuille de calcul.

Le code VBA étant assez long, nous allons détailler chaque macro séparément et même les différentes parties de la macro principale. On commence par activer l'option *Explicit* qui impose la déclaration de toutes les variables. Cette option est conseillée car sinon les variables non déclarées sont implicitement de type *Variant*, ce qui ralentit les calculs. La première macro *EFFACER_SOLUTION* efface le contenu des cellules de l'échiquier.

```
'RESOLUTION DU PROBLEME DES 8 REINES
'La taille est mise en constante (N=8) pour améliorer la lisibilité
'mais elle ne doit pas être modifiée car il est trop difficile de
'dessiner automatiquement les tableaux en cas de taille variable.

Option Explicit

Sub EFFACER_SOLUTION()
   Range("E7:L14") = 0
End Sub
```

La macro *RESOUDRE_PROBLEME* va placer les formules dont nous aurons besoin, mais aussi ajouter les contraintes et toutes les informations à la boite de dialogue du solveur avant de lancer la résolution. Pour cette macro, nous déclarons une constante de taille, *N*=8, mais attention, changer cette valeur entraine le changement de beaucoup de formules et de plages de cellules. Il nous faut aussi deux variables *E* et *S* de type *Range* pour nommer l'échiquier et pour les différentes sommes. Une variable *F* de type chaîne de caractères servira à stocker les formules que nous allons créer. Enfin, quelques autres variables sont nécessaires pour les indices et le code retour du solveur.

```
Sub RESOUDRE_PROBLEME()

Const N = 8                           'Taille de l'échiquier
Dim E As Range, S As Range            'Echiquier et plage de sommes
Dim F As String                       'Chaîne les formules de sommes
Dim i As Long, j As Long, k As Long 'Indices pour échiquier et plages
Dim Result As Integer                 'Code retour solveur
```

Dans un premier temps, on définit la plage correspondant à l'échiquier avec l'instruction *Set* et nous désactivons la mise à jour de l'écran avec la propriété *ScreenUpdating*, sinon la macro qui modifie les cellules de la feuille de calcul provoque un scintillement. Les quatre lignes suivantes servent pour réinitialiser le solveur, placer les options habituelles, définir la cellule cible et son sens d'optimisation, spécifier les cellules variables et imposer aux variables d'être binaires.

```
'Définit échiquier + options, objectif, variables binaires du solveur
Set E = Range("E7:L14")
Application.ScreenUpdating = False
Call SolverReset
Call SolverOptions(AssumeLinear:=True, AssumeNonNeg:=True)
Call SolverOK(SetCell:="P4", MaxMinVal:=1, ByChange:="E7:L14")
Call SolverAdd(CellRef:="E7:L14", Relation:=5)
```

Le code permettant d'ajouter les contraintes (2) et (3) pour les lignes et les colonnes est assez simple. Décrivons le principe pour les lignes, le traitement des colonnes étant similaire. La variable *S* est d'abord mise en équivalence avec la plage des sommes de lignes, puis une boucle *For* balaye les indices *i* des lignes pour ajouter une formule calculant la somme des cellules de la ligne. Cette formule est une chaîne de caractères contenant "=SUM(" puis la référence de plage pour la ligne *i* et une parenthèse fermante. La référence est obtenue avec la propriété *Address* : "Range(E(i,1), E(i,8)).Address".

La formule est ici introduite dans une cellule de la feuille avec la propriété *Formula* qui exige des noms anglais pour les fonctions. Une autre propriété, *FormulaLocal*, permet d'utiliser la langue de la version d'Excel utilisée ("=SOMME") mais elle donne un code VBA moins portable. À l'issue de la boucle *For*, on ajoute la contrainte dans la boite de dialogue du solveur avec un appel à *SolverAdd*. La partie gauche de la contrainte correspond aux cellules contenant les formules de somme que l'on vient de calculer, la relation est "≤" et le membre de droite est 1 comme l'indiquent les contraintes (2).

```
'Formules et contraintes pour les lignes
Set S = Range("A7:A14")
For i = 1 To N
  S(i).Formula = "=SUM(" & Range(E(i, 1), E(i, 8)).Address & ")"
Next i
Call SolverAdd(CellRef:="A7:A14", Relation:=1, FormulaText:=1)

'Formules et contraintes pour les colonnes
Set S = Range("E3:L3")
For j = 1 To N
  S(j).Formula = "=SUM(" & Range(E(1, j), E(8, j)).Address & ")"
Next j
Call SolverAdd(CellRef:="E3:L3", Relation:=1, FormulaText:=1)
```

Pour les diagonales, le principe reste le même, sauf qu'il faut utiliser les bons indices pour chacune des diagonales, comme dans les expressions des contraintes (4) à (7).

```
'Diagonales descendant à droite en partant de la ligne 1
Set S = Range("C5:I5")
For j = 1 To 7
  F = "="
  For k = j To N
    F = F & "+" & E(k - j + 1, k).Address
  Next k
  S(j).Formula = F
Next j
Call SolverAdd(CellRef:="C5:I5", Relation:=1, FormulaText:=1)

'Diagonales descendant à droite en partant de la colonne 1
Set S = Range("H16:M16")
For i = 2 To N - 1
  F = "="
  For k = i To N
    F = F & "+" & E(k, k - i + 1).Address
  Next k
  S(8 - i).Formula = F
Next i
Call SolverAdd(CellRef:="H16:M16", Relation:=1, FormulaText:=1)

'Diagonales descendant à gauche en partant de la ligne 1
Set S = Range("C10:C16")
For j = 2 To N
  F = "="
  For k = 1 To j
    F = F & "+" & E(k, j - k + 1).Address
  Next k
  S(j - 1).Formula = F
Next j
Call SolverAdd(CellRef:="C10:C16", Relation:=1, FormulaText:=1)
```

```
'Diagonales descendant à gauche en partant de la colonne 1
Set S = Range("N6:N11")
For i = 2 To N - 1
  F = "="
  For k = i To N
    F = F & "+" & E(k, N - k + i).Address
  Next k
  S(i - 1).Formula = F
Next i
Call SolverAdd(CellRef:="N6:N11", Relation:=1, FormulaText:=1)
```

Une fois l'ensemble des formules placées dans la feuille et les contraintes dans la boite de dialogue du solveur, on peut lancer la résolution avec l'appel à *SolverSolve* en désactivant la fenêtre de fin d'exécution usuelle. Pour être sûr que la résolution a fonctionné correctement, on teste la valeur de *Result* et on affiche le message correspondant.

```
'Résout le PL en 0-1 et affiche le code-retour
Result = SolverSolve(UserFinish:=True)
Select Case Result
  Case 0:    MsgBox ("Optimisation réussie")
  Case 4:    MsgBox ("Optimum non borné, vérifiez les données")
  Case 5:    MsgBox ("Infaisable, vérifiez les données")
  Case Else: MsgBox ("Erreur solveur code " & Result)
End Select

End Sub
```

15.4.4 Résultats

Il existe de nombreuses solutions à ce problème et la résolution, d'un ordinateur à l'autre peut différer. La figure 15.3 donne la position des huit reines sur l'échiquier comme indiqué par les valeurs de variables c_{ij}.

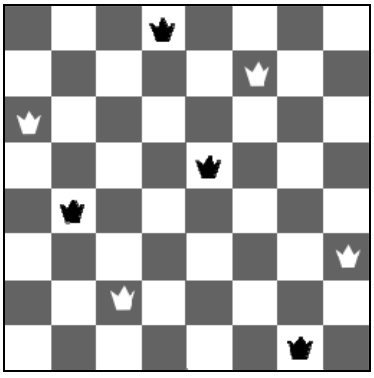


Figure 15.3 – Les huit reines

15.5 Références et compléments

On peut trouver de nombreuses références sur les énigmes, jeux et casse-tête [Clarke 1954], [Dudeney 1917], [Kraitchik 1942]. La page web de Martin Chlond (référence au début de ce chapitre) est une très bonne introduction à ces problèmes particuliers. Sur cette page, certains modèles manquent d'explications pour la compréhension de chaque contrainte, mais les commentaires nécessaires peuvent être demandés à Martin Chlond par courrier électronique.

Le lecteur intéressé plus particulièrement par les problèmes de logique est renvoyé à Friant [Friant 1986] et L'Hospitalier [L'Hospitalier 1998]. Ces deux livres présentent les démarches de résolution généralement utilisées et expliquent comment construire de tels problèmes.