

# **Rapport TP d'optimisation**

**Réalisé par**

**SEKKAT Mohammed Amine**

**BELMAMOUN Yassine**

L’ensemble du code est présent à l’adresse :

<https://github.com/yassinepample/TP_optimisation>

**1/ Séance 1 : optimisation continue et optimisation approchée**

*L'executable de cette partie se trouve dans MAIN.m*

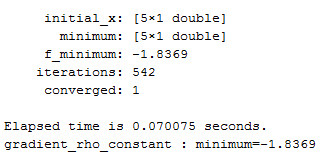
***1.1 Optimisation sans contraintes***

***1.1.1 Méthode du gradient***

**a)** La fonction **GradResults** permet de minimiser une fonction par la méthode du gradient à pas fixe.

Nous la testons ici avec différentes valeurs de Rho. On obtient les résultats suivants :

* Rho = 0.01: Minimum = -1.8369



* Pour Rho = 0.1, il n’y pas de convergence
* Rho = 0.5, il n’y pas de convergence

On s'apercoit donc que la convergence n'est pas toujours assurée en prenant une méthode du gradient à Rho constant.

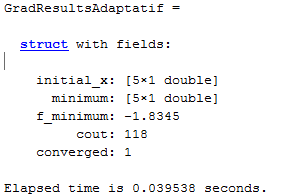
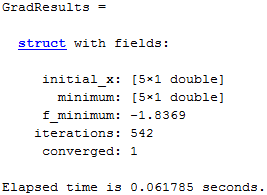
**b)** L'algorithme se trouve dans le fichier gradient\_rho\_adaptatif.m

**c)** Avec 'coût' représentant dans le cas adaptatif le nombre d'appel de la fonction, et en partant d'un pas initial Rho0 de 0.01, on a les résultats suivants :

**Algorithme Rho Constant = 0.01**

**Algorithme Rho Adaptatif : Rho0 = 0.01**

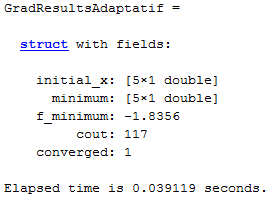
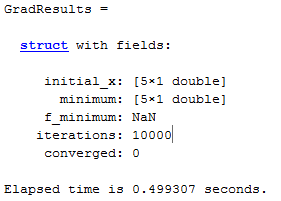




**Algorithme Rho Constant = 0.1**

**Algorithme Rho Adaptatif : Rho0 = 0.1**

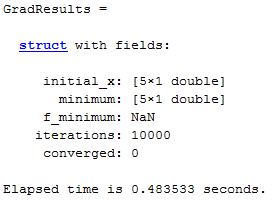
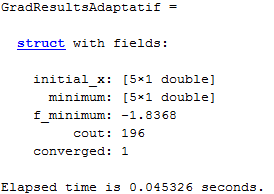
****

****

**Algorithme Rho Constant = 0.5**

**Algorithme Rho Adaptatif : Rho0 = 0.5**

****

** **

On remarque alors que la convergence est toujours assurée dans le cas de rho adaptatif, avec un temps de calcul plus faible.

**1.1.2 Utilisation des routines d’optimisation de la Toolbox Optimization. Méthode de Quasi-Newton (version BFGS).**

En utilisant la méthode de quasi Newton en laissant Matlab calculer un gradient approché (option par défaut), on obtient les résultats suivants:

|  |  |  |  |
| --- | --- | --- | --- |
| Méthode Rho constant  =0.01 | Méthode Rho Adaptatif  Rho0=0.01 | Quasi-Newton sans fournir de gradient | Quasi-Newton en fournissant le gradient |
| -1.8369 | -1.8345 | -1.8370 | -1.8370 |

|  |  |
| --- | --- |
| **Methodes de Quasi-Newton** | |
| Sans renseigner le gradient | En renseignant le gradient |
|  |  |

Les méthodes de quasi-Newton présentent moins d'itérations et d'appels à la fonction, mais présentent néanmoins un temps de calcul plus important, étant donné le calcul de l'inverse d'une matrice dans le processus d'optimisation.

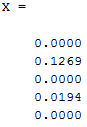
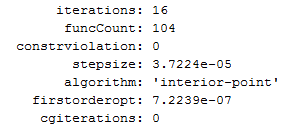
La fonction f1 est quadratique : il serait possible de calculer explicitement l'expression dans R5 et à partir de là se ramener à un problème de minimisation où on peut expliciter la solution par Fritz-Jones.

**1.2 Optimisation sous contraintes**

**1.2.1 Optimisation à l’aide de routines Matlab**

En utilisant l'algorithme SQP de Matlab, nous obtenons les résultats suivants:

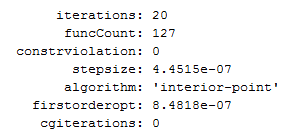
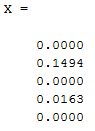
**F1**

****



**Valeur\_Min = -0.1385**

**F2**

****

Elapsed time is 4.294305 seconds.

**Valeur\_Min = 2 x 10^-6**

**1.2.2 Optimisation sous contraintes et pénalisation**

1) On prend une fonction C1 pour être sûr qu'elle soit semi continue inférieur.

Cette fonction de pénalisation est positive et telle que B(U)=0 si U appartient à Uad.

La fonction est implémentée dans le fichier Beta.m, et le script lançant l'optimisation de la fonction de pénalisation pour f1 et f2 se trouve dans MAIN2.m .

**1.2.3 Méthodes duales pour l’optimisation sous contraintes**

1) Le lagrangien de la fonction est défini dans le fichier lagrangien.m.

qui peut s'écrire sous forme matricielle par :

2) L'algorithme d'Uzawa est implémenté dans le fichier MAIN2.m .

On trouve avec cette méthode

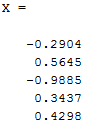
**F1min =-0.1381**

**1.3 Optimisation non convexe - Recuit simulé**

a) La fonction f4 est implémentée dans le fichier f4.m .

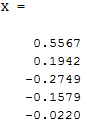
b) On utilise ici BFGC avec fminunc, en prenant différentes valeurs initiales.

* U0 = [0,0,0,0,0]



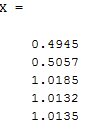
FVAL = -7.6563

* U0 = [0.5;0.5;0.5;0.5;0.5];



FVAL = -1.3731

* U0 = [0.5;0.5;1;1;1];



FVAL = 105.4410

c) Etant donné que plus T est faible, plus la probabilité p = exp(−(J(y)−J(xi))/T) de choisir un point "Moins bon" est faible, il convient de choisir un T petit. On fixe ici T = 50.

Par ailleurs, on diminue le nombre d'itérations à la même température afin de converger plus rapidement : On fixe L = 10.

Avec ces paramètres, nous verifions aisément que nous convergeons toujours vers le même point, à savoir le minimum global

**X = -1.5041; 0.4583; -0.6201; 0.9569; -0.8089**

**FVAL = -10.7979**

**1.4 Application Synthèse d’un filtre à réponse impulsionnelle finie**

1) En minimisant le critère J comme problème d'optimisation sans contrainte par la méthode de Quasi Newton, et en prenant comme H initial le vecteur 2\*ones(30,1), on obtient :

Jmin = 0.7669

Ce résultat dépend de la valeur initial de H: en prenant Hinitial = ones(30,1), on trouve Jmin = 1.

2) Le problème peut être reformulé de la manière suivante :

Pour

**2/ Séances 2 et 3 : optimisation discrète et optimisation multi-objectif**

**2.1 Rangement d’objets (optimisation combinatoire)**

On note n le nombre de boîtes qui est égal au nombre d’objets par hypothèse du problème.

On considère ici la matrice X(i, j)= 1 si l'objet j est dans la boîte i.

1. La condition pour que la boite i contienne un objet et un seul s'écrit:

Par ailleurs, la condition pour que l'objet j ne soit que dans une seule boite s'écrit:

**2.** Le problème s'écrit ici comme un problème linéaire en nombre entier qui s'écrit de manière à minimiser la fonction de déplacement total:

constante définie du problème, notée .

Dans un premier temps, il convient de transformer la matrice X en vecteur de taille n². Celui ci s'écrit :

X =

n

n

n

n

n

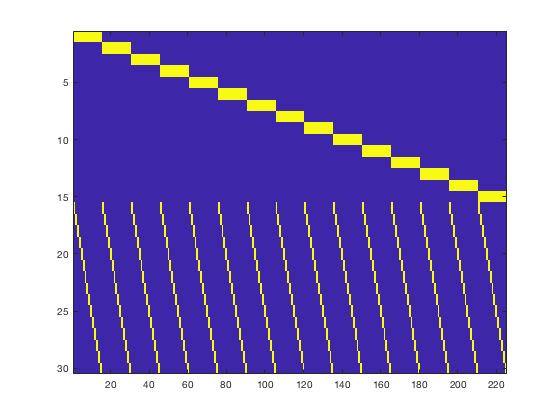
Les contraintes s'écrivent de la forme A\_eq.X = B\_eq avec A\_eq qui s'écrit de la manière suivante dans le cas 3x3

n²&²

A=

2n

La représentation à l’aide de la fonction « imagesc » de la matrice des contraintes d'égalité est de la forme suivante dans le cas n=15.



B\_eq est le vecteur de taille n^2 et ne contenant que des 1.

En transformant la matrice ( ) i,j en vecteur, le problème devient :

sous contrainte A\_eq.X = B\_eq.

Le résultat que nous obtenons (fval) est le suivant :

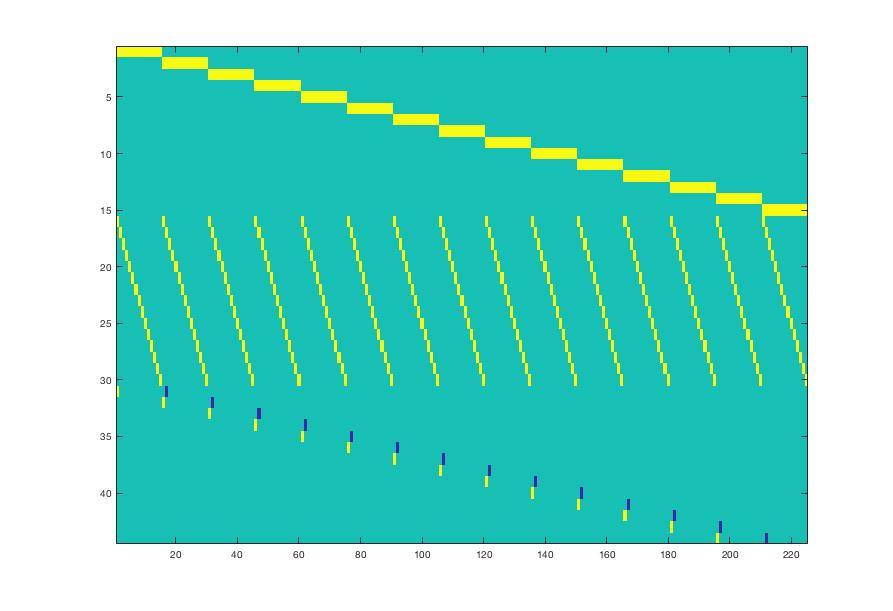
**15.3776**

Le programme est implémenté dans le fichier **TP2/partie\_1/question\_2.m**

**3-** La condition pour que l’objet n°1 se situe dans la boîte située juste à gauche de la boîte contenant l’objet n°2.

Ajouter cette contrainte revient à enrichir la matrice de contraintes afin de prendre en compte la condition que

On ajoute donc à la matrice construite précédemment la matrice suivante dans le cas n=3 par concaténation des lignes :



Le programme est implémenté dans le fichier **TP2/partie\_1/question\_3.m**

On obtient : **Dmin =15.56**

**4-** cela signifie que l'objet 3 se trouve dans une boite i et l'objet 4 ne se trouve pas dans une boite à droite de l'objet 3: l'objet 3 est donc à droite de l'objet 4.

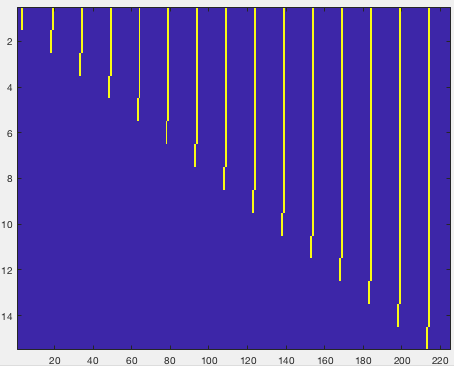
cela signifie que l'objet 3 ne se trouve pas dans une boite à gauche de l'objet 4: l'objet 3 est donc à droite de l'objet 4.

Alors ni l'objet 3 ni l'objet 4 ne sont dans les boites i et (i+k): Si l'objet 3 n'est pas dans la boite i, alors l'objet 4 n'est pas a droite du 3.

Donc la condition traduisant la contrainte que la boîte contenant l’objet n°3 se situe à droite de la boîte contenant l’objet n°4 est bien xi,3 + xi+k,4 ≤ 1∀i, ∀k > 0.

Ainsi, la condition s'écrit :

La matrice de contrainte qui intègre cette contrainte est de la forme suivante :



Le programme est implémenté dans le fichier **TP2/partie\_1/question\_4.m**

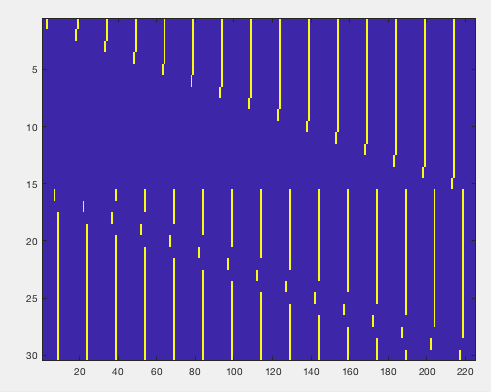
On obtient : **Dmin =** **15.901379**

**5-** Modélisation pour traduire la contrainte que la boîte contenant l’objet n°7 se situe à côté de la boîte contenant l’objet n°9:

ou

Celle ci s'écrit :

La matrice de contrainte qui intègre cette contrainte est de la forme suivante :



Le programme est implémenté dans le fichier **TP2/partie\_1/question\_5.m**

On obtient : **Dmin =** **15.9048**

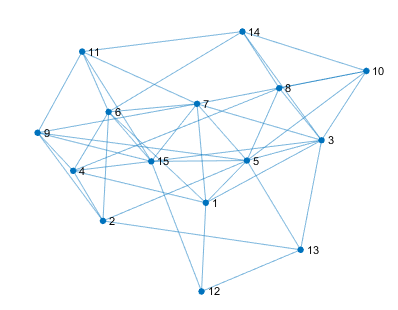
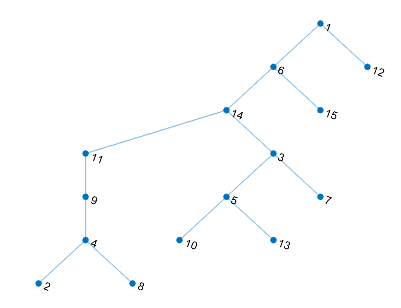
**2.2 Communication entre espions (optimisation combinatoire)**

On modélise le lien entre les espions par de taille n² tel que si les espions i et j communiquent, et 0 sinon.

Notons que .

La fonction a minimiser est donc .

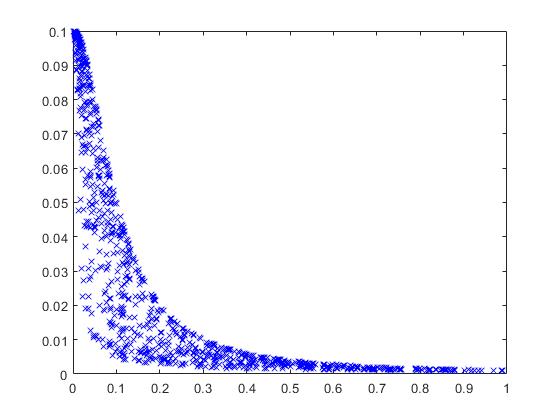
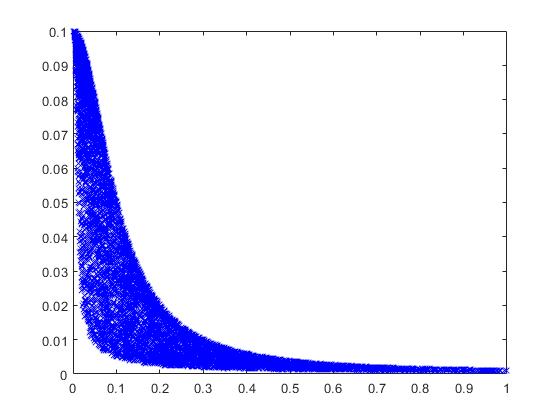
Afin d'avoir un problème linéaire, nous passons au logarithme: la fonction à minimiser devient :



En raisonnant par événement contraire, on obtient comme valeur minimale :

**2.3 Dimensionnement d’une poutre (optimisation multiobjectif)**

Il convient ici de générer N couples (a,b) de solutions réalisables, puis d'approximer le front de pareto par la courbe obtenue à partir des solutions de rang 1. Dès lors, une fois les solutions réalisables obtenues, on détermine le front de pareto par domination des solutions.

 **N=1000 N=10 000**

**2.4 Approvisionnement de changer (optimisation combinatoire)**

Nous cherchons à minimiser les dépenses de l’entreprise en terme d’utilisation d’engins (location, acheminement, remise).

Le problème s’étale sur une durée de n=99 semaines.

Nous modélisons le problème de la manière suivante :

Les lignes représentant les semaines et les 3 colonnes sont définies comme suit :

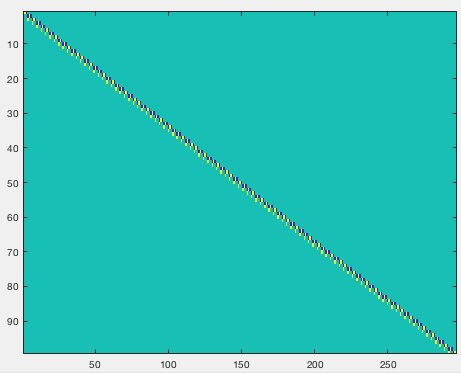
* La première colonne étant le nombre d’engins présent au début de la semaine i sur le chantier.
* La deuxième colonne étant le nombre d’engins acheminer au début de la semaine i.
* La troisième colonne étant le nombre d’engins remis à la société de location au début de la semaine i.

Nous transformations la matrice X en un vecteur de la façon suivante :

Le coût a payé par l’entreprise à l’agence de location est défini comme le produit suivant:

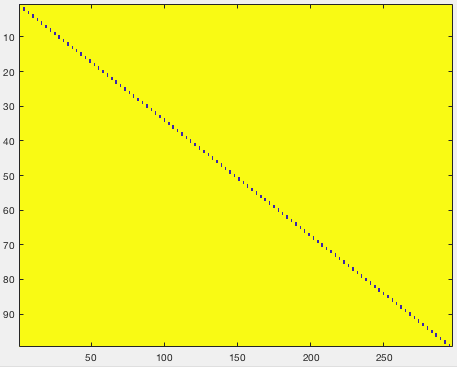
Il existe deux contraintes :

* Contrainte de conservation du nombre d’engins (Egalité) :



*Cette contrainte d’égalité consiste à dire que le nombre d’engins présents à la semaines (n+1) est égale au nombre d’engins présents à la semaine n + le nombre d’engins apportés à la semaine (n+1) moins le nombre d’engins remis à la semaine (n+1).*

* Contrainte de respect du nombre d’engins nécessaire par semaine (Inégalité) :



*Cette contrainte consiste à vérifier que le nombre d’engins présents à la semaine n est supérieur ou égale.*

Remarque importante : Il faut noter que le cout total de cet algorithme ne prend pas en compte les frais d’acheminement à la semaine 1 (36\*800=28,800) et les frais de remise à dernière semaine (173\*1200=207,600).

On obtient donc :

**fval = 3,112,400**

Le cout tôtal à payer par l’entreprise à l’agence de location des engins est donc de :

**Coût = fval + 207,600 + 28,800 = 3,348,800**

L’implementation du code se situe au niveau du fichier : **TP2/partie\_3/approvisionnement\_chantier.m**

Voici le résultat qui permet de voir qu’il est plus intéressant d’un point de vue finanier de garder des engins sur le chantier quitte à ne pas les utiliser (temporairement) plutôt que de les rendre et les re-louer lorsqu’il y en a besoin à nouveau.

Les semaines 14 à 18 sont un bon exemple pour comprendre ce qui se passe.

|  |  |  |  |
| --- | --- | --- | --- |
| Semaine | Engins présents | Engins nécessaires | Engins non-utilisés |
| 1 | 36 | 36 | 0 |
| 2 | 61 | 61 | 0 |
| 3 | 84 | 84 | 0 |
| 4 | 87 | 87 | 0 |
| 5 | 121 | 121 | 0 |
| 6 | 121 | 88 | 33 |
| 7 | 121 | 66 | 55 |
| 8 | 121 | 69 | 52 |
| 9 | 121 | 71 | 50 |
| 10 | 121 | 78 | 43 |
| 11 | 121 | 116 | 5 |
| 12 | 149 | 149 | 0 |
| 13 | 149 | 130 | 19 |
| **14** | **158** | **158** | **0** |
| **15** | **158** | **143** | **15** |
| **16** | **158** | **151** | **7** |
| **17** | **158** | **148** | **10** |
| **18** | **180** | **180** | **0** |
| 19 | 170 | 170 | 0 |
| 20 | 165 | 165 | 0 |
| 21 | 158 | 158 | 0 |
| 22 | 131 | 130 | 1 |
| 23 | 131 | 96 | 35 |
| 24 | 131 | 82 | 49 |
| 25 | 131 | 80 | 51 |
| 26 | 131 | 81 | 50 |
| 27 | 131 | 117 | 14 |
| 28 | 131 | 113 | 18 |
| 29 | 131 | 124 | 7 |
| 30 | 131 | 131 | 0 |
| 31 | 131 | 110 | 21 |
| 32 | 131 | 103 | 28 |
| 33 | 131 | 131 | 0 |
| 34 | 126 | 126 | 0 |
| 35 | 126 | 121 | 5 |
| 36 | 126 | 101 | 25 |
| 37 | 126 | 118 | 8 |
| 38 | 126 | 95 | 31 |
| 39 | 126 | 85 | 41 |
| 40 | 126 | 82 | 44 |
| 41 | 126 | 91 | 35 |
| 42 | 126 | 67 | 59 |
| 43 | 126 | 75 | 51 |
| 44 | 126 | 118 | 8 |
| 45 | 145 | 145 | 0 |
| 46 | 145 | 138 | 7 |
| 47 | 150 | 150 | 0 |
| 48 | 150 | 147 | 3 |
| 49 | 150 | 120 | 30 |
| 50 | 150 | 102 | 48 |
| 51 | 150 | 146 | 4 |
| 52 | 150 | 130 | 20 |
| 53 | 166 | 166 | 0 |
| 54 | 166 | 160 | 6 |
| 55 | 167 | 167 | 0 |
| 56 | 164 | 153 | 11 |
| 57 | 164 | 160 | 4 |
| 58 | 164 | 145 | 19 |
| 59 | 164 | 164 | 0 |
| 60 | 137 | 137 | 0 |
| 61 | 134 | 110 | 24 |
| 62 | 134 | 133 | 1 |
| 63 | 134 | 107 | 27 |
| 64 | 134 | 75 | 59 |
| 65 | 134 | 95 | 39 |
| 66 | 134 | 132 | 2 |
| 67 | 134 | 116 | 18 |
| 68 | 134 | 117 | 17 |
| 69 | 134 | 134 | 0 |
| 70 | 134 | 129 | 5 |
| 71 | 134 | 134 | 0 |
| 72 | 134 | 108 | 26 |
| 73 | 141 | 141 | 0 |
| 74 | 155 | 155 | 0 |
| 75 | 151 | 151 | 0 |
| 76 | 125 | 111 | 14 |
| 77 | 125 | 125 | 0 |
| 78 | 112 | 106 | 6 |
| 79 | 112 | 80 | 32 |
| 80 | 112 | 91 | 21 |
| 81 | 112 | 92 | 20 |
| 82 | 112 | 90 | 22 |
| 83 | 112 | 95 | 17 |
| 84 | 112 | 102 | 10 |
| 85 | 112 | 87 | 25 |
| 86 | 112 | 106 | 6 |
| 87 | 112 | 100 | 12 |
| 88 | 112 | 112 | 0 |
| 89 | 149 | 149 | 0 |
| 90 | 171 | 171 | 0 |
| 91 | 171 | 157 | 14 |
| 92 | 173 | 173 | 0 |
| 93 | 173 | 137 | 36 |
| 94 | 173 | 134 | 39 |
| 95 | 173 | 141 | 32 |
| 96 | 173 | 165 | 8 |
| 97 | 173 | 149 | 24 |
| 98 | 173 | 173 | 0 |
| 99 | 173 | 129 | 44 |