Rapport du DM d'OPTI2

Adrien Blassiau
Corentin JUVIGNY
Jiahui XU

29 novembre 2019

1 Modélisation du problème (MOON)

On décrit tout d'abord le problème de manière formelle sous la forme d'un **problème d'optimisation** avec le quadruplet (I, S, M, O).

Instances : Soit une grille $\Omega = (\omega_{ij})_{1 \leq i \leq h, 1 \leq j \leq w}$ d'entiers positifs, un entier n, n triplets $(M_i, H_i, W_i)_{i \in [\![1;N]\!]}$ où M_i,H_i et W_i sont des entiers tels que :

$$1 \leq H_i \leq H \tag{1}$$
$$1 \leq W_i \leq W \tag{2}$$
$$0 \leq M_i \tag{3}$$

Solutions : Un ensemble $I \subset [\![1; N]\!]$ et $\forall i \in I$, $l_i, c_i \in \mathbb{N}$ tels que :

$$1 \leq l_i \leq h - H_i + 1 \tag{C1}$$
$$1 \leq c_i \leq w - W_i + 1 \tag{C2}$$

et

$$c_j \geq c_i + W_i \tag{C3}$$
$$\text{ou} \quad l_j \geq l_i + H_i \tag{C4}$$
$$\text{ou} \quad c_i \geq c_j + W_j \tag{C5}$$
$$\text{ou} \quad l_i \geq l_j + H_j \tag{C6}$$

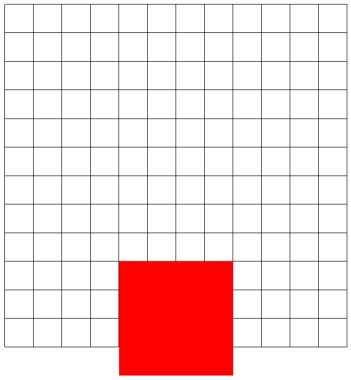
De petites illustrations de ces contraintes sont disponibles en Figure 1 et Figure 2.

Mesure :

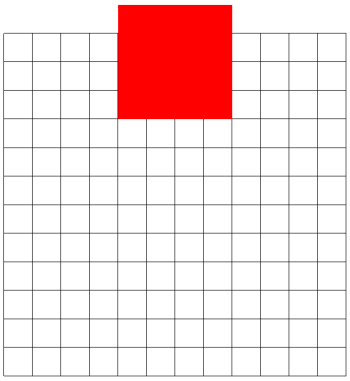
$$\sum_{i \in I} min \left(M_i, \sum_{l=l_i}^{l_i+H_i-1} \sum_{c=c_i}^{c_i+W_i-1} \omega_{lc} \right) \tag{4}$$

Objectif : max

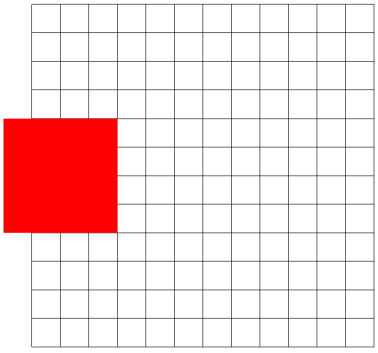
On peut maintenant décrire le **problème de décision** associé à ce problème d'optimisation : soit x une instance du problème et k un entier, déterminons si $M(x) \geq k$ où M est la mesure définie plus haut.



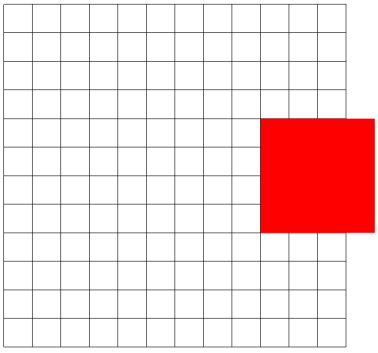
(a) C1 bas



(b) C1 haut

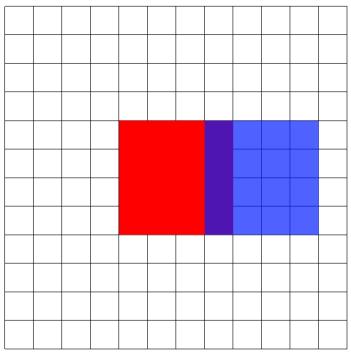


(c) C2 gauche

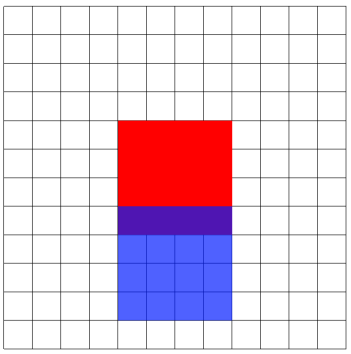


(d) C2 droite

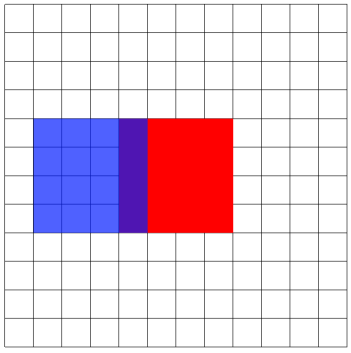
FIGURE 1 – Contraintes de non dépassement de la grille



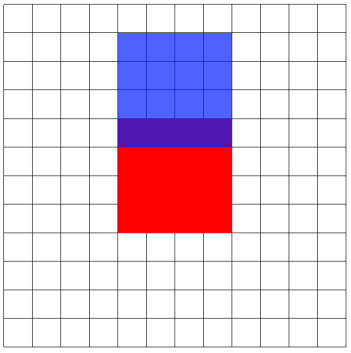
(a) C3



(b) C4



(c) C5



(d) C6

FIGURE 2 – Contraintes de non superposition de deux publicités

2 Étude de la complexité

TODO

3 Algorithme Exact

On propose un algorithme exact qui permet de résoudre tout type d'instance. Cet algorithme est subdivisé en deux algorithmes :

— le premier (**Algorithm 1**) permet de traiter toutes les instances mais à une complexité élevée
— le second (**Algorithm 2**) à une bonne complexité mais permet de ne traiter que les instances où tout les annonceurs demandent une seule case de publicité.

3.1 Premier sous algorithme

On propose le premier algoithme suivant :

Algorithm 1 Renvoie une solution optimale au problème MOON pour n'importe quelle instance.

Require: $n = 1$
Ensure: La solution et le profit renvoyés sont les meilleurs possible.
  if On a essayé de placer tout les annonceurs. **then**
    On retient le profit courant pc.
    if Si pc est meilleur que le meilleur profit trouvé pour le moment. **then**
      On garde en mémoire pc comme étant le meilleur profit courant ainsi que la solution associée qui devient la meilleure solution courante.
      On retourne le meilleur profit et la meilleur solution.
    else
      On retourne le meilleur profit et la meilleur solution.
    end if
  else
    while On n'a pas itéré sur toutes les cases de la grille. **do**
      On essaye de placer l'annonceur n sur la case courante.
      On appelle récursivement la fonction sur le prochain annonceur n+1 qui renvoie la meilleur solution et le meilleur profit courant.
      On enlève l'annonceur n de la grille.
    end while
    On retourne le meilleur profit et la meilleur solution.
  end if

Cette algorithme teste toutes les arrangement de n éléments d'une séquence de $w * h$ éléments donc réalise A^n_{w*h} itérations. Or, on rappelle que :

$$A^n_p = \frac{n!}{(n-p)!} \tag{5}$$

Le nombre d'itérations effectuées est donc exponentiel. **Cet algorithme à donc une complexité en temps exponentielle**.

Sur la plus grosse instance de test qui comporte une grille de taille 100 et 10 annonceurs à placer, on a $A^{10}_{100} = 62815650955529472000$ soit **soixante-deux trillions huit cent quinze billiards six cent cinquante billions neuf cent cinquante-cinq milliards cinq cent vingt-neuf millions quatre cent soixante-douze mille cas à tester**!
En supposant que l'on fasse 2 milliards d'opérations pas seconde, ce calcul nous prendra **995 ans** ...
On a donc mis en place un algorithme plus efficace mais qui ne marche que sur un type d'instance bien particulière.

3.2 Deuxième sous algorithme

On propose le deuxième algorithme suivant :

Algorithm 2 Renvoie une solution optimale au problème MOON pour une instance où les annonceurs ne demandent qu'une seule case de publicité.

Ensure: La solution et le profit renvoyés sont les meilleurs possible.
On associe à chaque case de la grille p des coûts une clé qui indique ses coordonnées.
On redimensionne p sous la forme d'un vecteur.
On trie p par coût décroissant.
On associe à chaque case du vecteur v des prix des annonceurs une clé qui indique sa coordonnée.
On trie v par coût décroissant.
L'annonceur courant n est celui de coût le plus élevé donc le premier élément de v.
while On n'a pas itéré sur toutes les cases de la grille p triée. **do**
  if On a essayé de placer tout les annonceurs. **then**
    On retourne la solution et le profit courant obtenus qui sont les meilleurs.
  else
    On essaye de placer l'annonceur n sur la case courante.
    if On a réussi à placer l'annonceur. **then**
      On continue avec n+1.
    end if
  end if
end while
On retourne la solution et le profit courant obtenus qui sont les meilleurs.

Cette algorithme est bien plus efficace que le précédent.
On réalise tout d'abord un parcours et un tri de la matrice des coûts et du vecteur des prix des annonceurs, on a donc une complexité en $\mathcal{O}((h*v)log(h*v)+nlog(n))$.
Ensuite, dans le pire des cas on itère sur toutes les cases de la matrice sans réussir à placer d'annonceur donc la complexité est en $\mathcal{O}((h*v)log(h*v))$.
Au final, la complexité est en $\mathcal{O}((h*v)log(h*v)+nlog(n))$. **Cet algorithme à donc une complexité en temps polynomiale.**

3.3 Programmation linéaire

On écrit le programme linéaire modélisant notre problème. Il comporte **6 contraintes**, **4 variables** ($P_i, y_i^{lc}, z_i^{lc}, c_i$) et **7 constantes** $n, h, w, M_i, H_i, W_i, \omega_{lc}$.

$$\max \quad z = \sum_{i=1}^{n} P_i$$

s.c.

$$P_i \leq M_i \qquad \forall i \in [\![1;n]\!] \quad \textbf{(C1)}$$

$$P_i \leq \sum_{l=1}^{h}\sum_{c=1}^{w} \omega_{lc} y_i^{lc} \qquad \forall i \in [\![1;n]\!] \quad \textbf{(C2)}$$

$$\sum_{i=1}^{n} y_i^{lc} \leq 1 \qquad \forall l \in [\![1;h]\!], \quad c \in [\![1;w]\!] \quad \textbf{(C3)}$$

$$\sum_{l=1}^{h}\sum_{c=1}^{w} z_i^{lc} \leq 1 \qquad \forall i \in [\![1;n]\!] \quad \textbf{(C4)}$$

$$z_i^{lc} \leq y_i^{lc} \qquad \forall i \in [\![1;n]\!], \quad \forall l \in [\![1;h]\!], \quad c \in [\![1;w]\!] \quad \textbf{(C5)}$$

$$y_i^{l'c'} \leq \sum_{l=l'-H_i+1}^{l'} \sum_{c=c'-W_i+1}^{c'} z_i^{lc} \qquad \forall i \in [\![1;n]\!], \quad \forall l' \in [\![1;h]\!], \quad c' \in [\![1;w]\!] \quad \textbf{(C6)}$$

$$z_i^{lc} \leq c_i \qquad \forall i \in [\![1;n]\!], \quad \forall l \in [\![1;h]\!], \quad c \in [\![1;w]\!] \quad \textbf{(C7)}$$

$$\sum_{l=1}^{h}\sum_{c=1}^{w} y_i^{lc} \leq W_i * H_i * c_i \qquad \forall i \in [\![1;n]\!] \quad \textbf{(C8)}$$

$$n \in \mathbb{N}^+$$

$$h \in \mathbb{N}^+$$

$$w \in \mathbb{N}^+$$

$$P_i \in \mathbb{N}^+ \qquad \forall i \in [\![1;n]\!]$$

$$M_i \in \mathbb{N}^+ \qquad \forall i \in [\![1;n]\!]$$

$$H_i \in \mathbb{N}^+ \qquad \forall i \in [\![1;n]\!]$$

$$W_i \in \mathbb{N}^+ \qquad \forall i \in [\![1;n]\!]$$

$$\omega_{lc} \in \mathbb{N}^+ \qquad \forall l \in [\![1;h]\!], \quad c \in [\![1;w]\!]$$

$$y_i^{lc} \in [\![0;1]\!] \qquad \forall i \in [\![1;n]\!], \quad \forall l \in [\![1;h]\!], \quad c \in [\![1;w]\!]$$

$$z_i^{lc} \in [\![0;1]\!] \qquad \forall i \in [\![1;n]\!], \quad \forall l \in [\![1;h]\!], \quad c \in [\![1;w]\!]$$

$$c_i \in [\![0;1]\!] \qquad \forall i \in [\![1;n]\!]$$