COLLÈGE DE FRANCE
1530

Schémas d'approximation

Claire Mathieu

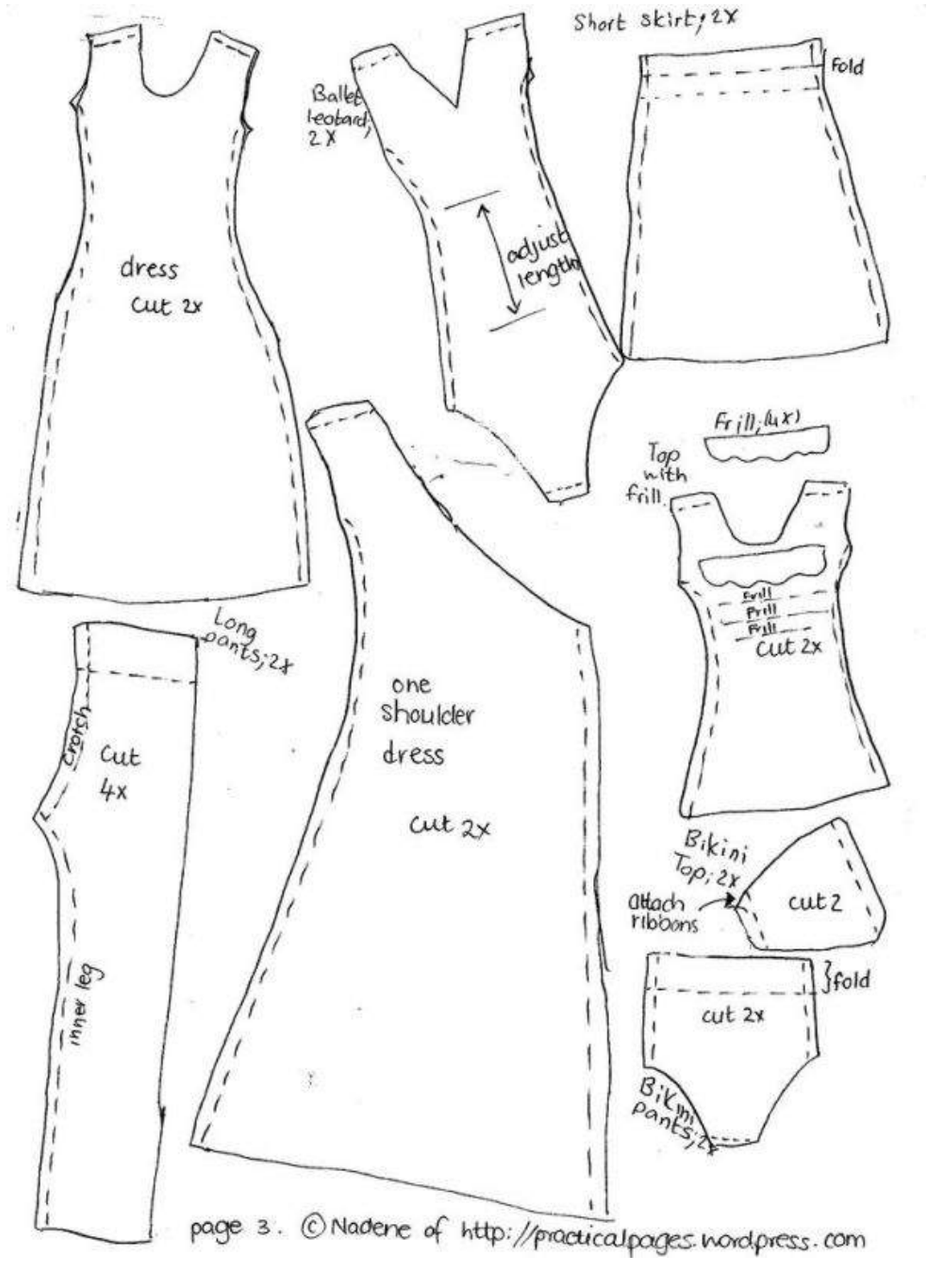
Découpe de tissu

Coupe maximum dans le plan

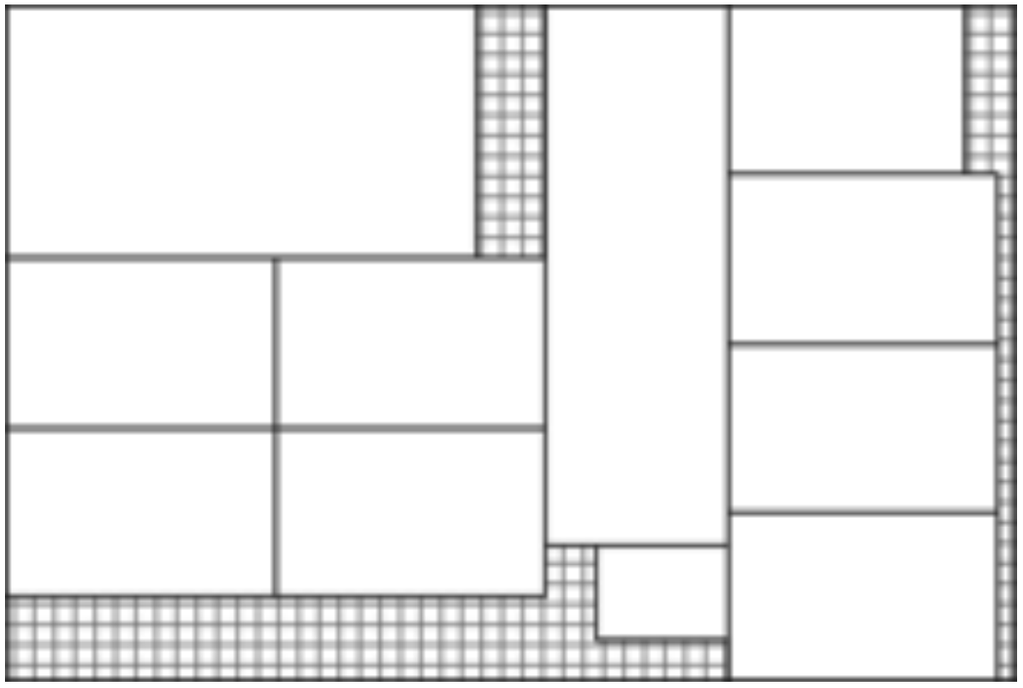
Disques disjoints

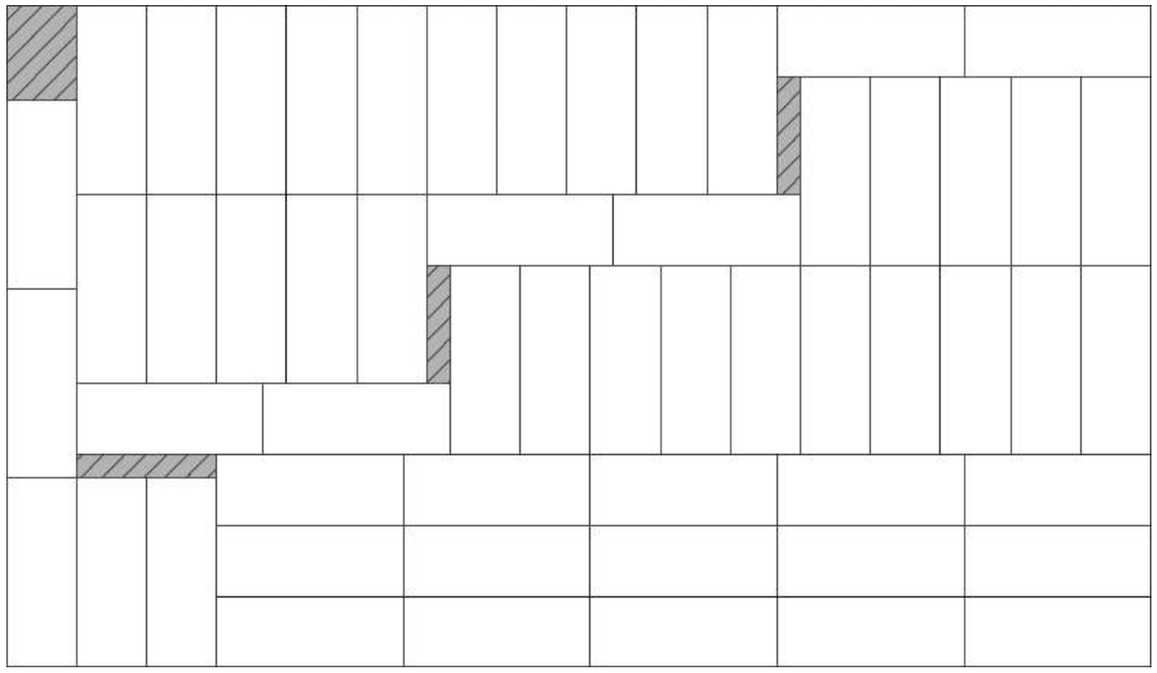
Découpe de tissu

tissu

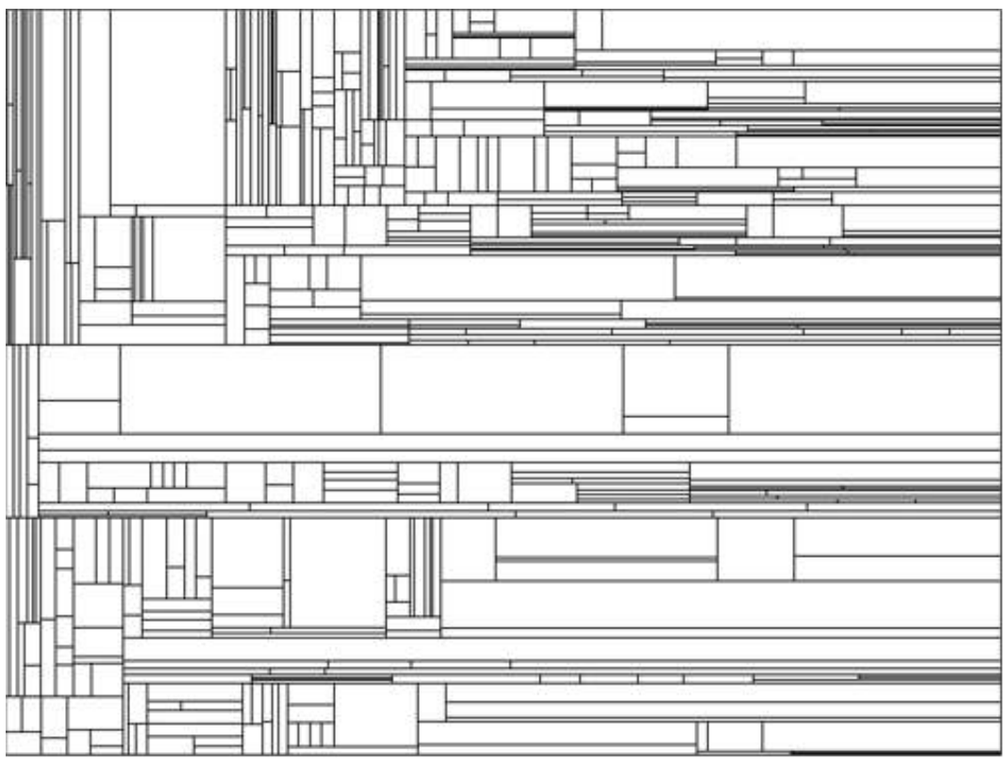


patron



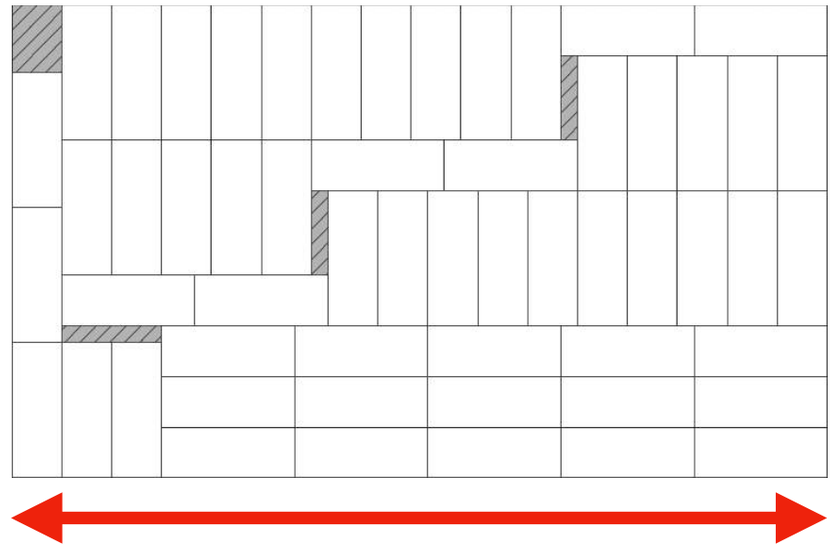


longueur à minimiser



Comment découper des rectangles dans un rouleau de tissu pour utiliser le moins de longueur possible

NP-difficile
Résolu quotidiennement
et approximativement



Mais comment font-ils ?

Algorithme d'approximation

Comment découper
en temps raisonnable
des rectangles
dans un rouleau de tissu
pour utiliser
le moins de longueur possible,
ou presque

Parenthèse :
comment ne pas concevoir les algorithmes

Comment ne pas concevoir les algorithmes

Algorithme
apparemment “raisonnable”

- Utilisons la programmation linéaire, parce que c’est à la mode
- Ajoutons des conditions supplémentaires, parce que ça renforce le programme
- Ajoutons une étape d’optimisation, parce que ça ne peut qu’améliorer la solution

Comment ne pas concevoir les algorithmes

Algorithme
apparemment “raisonnable”

- Utilisons la programmation linéaire, parce que c’est à la mode
- Ajoutons des conditions supplémentaires, parce que ça renforce le programme
- Ajoutons une étape d’optimisation, parce que ça ne peut qu’améliorer la solution

Mes critiques

- Exploration dans le vide
- Pas de direction
- Complexité inutile

Il n’y a pas de raison que ça marche
Même si ça marche, on ne pourra pas le démontrer
parce qu’on ne comprend pas ce qu’on fait

Comment ne pas concevoir les algorithmes

Algorithme
apparemment “raisonnable”

- **Utilisons la programmation linéaire, parce que c’est à la mode**
- **Ajoutons des conditions supplémentaires, parce que ça renforce le programme**
- **Ajoutons une étape d’optimisation, parce que ça ne peut qu’améliorer la solution**

Mes critiques

- **Exploration dans le vide**
- **Pas de direction**
- **Complexité inutile**

Il n’y a pas de raison que ça marche
Même si ça marche, on ne pourra pas le démontrer
parce qu’on ne comprend pas ce qu’on fait

Ce qui manque :
des exemples
pour développer l’intuition
et guider la démarche algorithmique

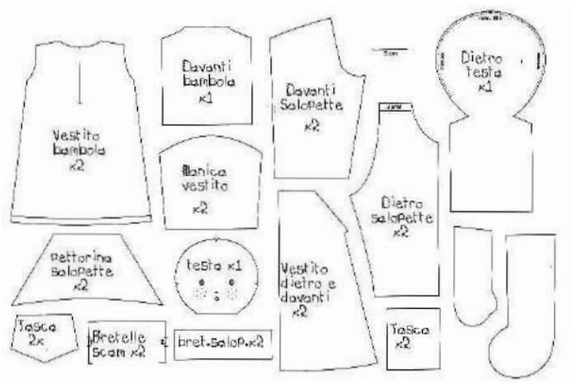
Comment concevoir les algorithmes

**“Rendez les choses
aussi simples que possible,
mais pas plus simple”**

“Aussi simples que possible” =
pour chaque étape de l’algorithme,
il faut pouvoir répondre à la question
“Pourquoi fait-on cela ?”
non pas par
“Ça ne fait pas de mal et ça peut toujours être utile”
mais avec un exemple :
“Dans l’exemple suivant, sans cette étape, l’algorithme ne marcherait pas”

Comment ne pas concevoir les algorithmes

Une famille d'exemples intéressants ⟷ **Algorithme optimisé pour ces exemples**



il faut rester simple :
ne pas optimiser sur des cas particuliers

La bonne question n'est pas:
"Comment traiter cet exemple de façon optimale ?"
mais
" Comment traiter cet exemple d'une façon adéquate

et qui ait une chance de se généraliser ?"

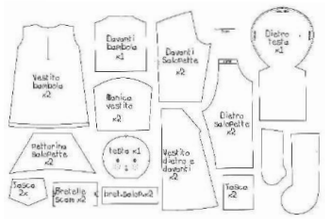
"Rendez les choses

aussi simples que possible,

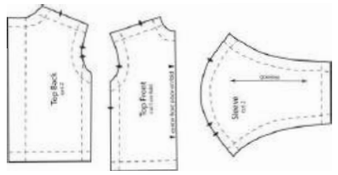
mais pas plus simple"

L'algorithmique, science collaborative

Une famille d'exemples intéressants ⟷ **Algorithme simple pour ces exemples**



2e famille d'exemples très différents ⟷ **Algorithme simple pour ces autres exemples**



Non, ça ne peut pas marcher parce que dans tel cas bla bla bla

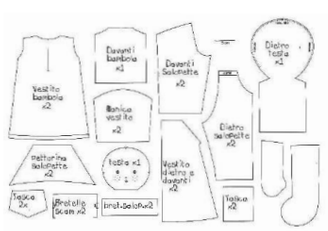
mais non, ça ne suffit pas à cause de tel autre cas

Et si on faisait bla bla bla ?

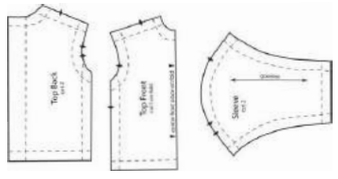
Mais alors, il suffirait de faire bla bla bla pour résoudre ce cas ?

Une méthode de conception d'algorithme

Une famille d'exemples intéressants ⟷ **Algorithme simple pour ces exemples** ⟷ **Analyse**



2e famille d'exemples très différents ⟷ **Algorithme simple pour ces autres exemples** ⟷ **Analyse**



Combinaison d'exemples ⟷ **Combinaison des algorithmes**

Ici il y a de la marge pour introduire de la complexité si nécessaire

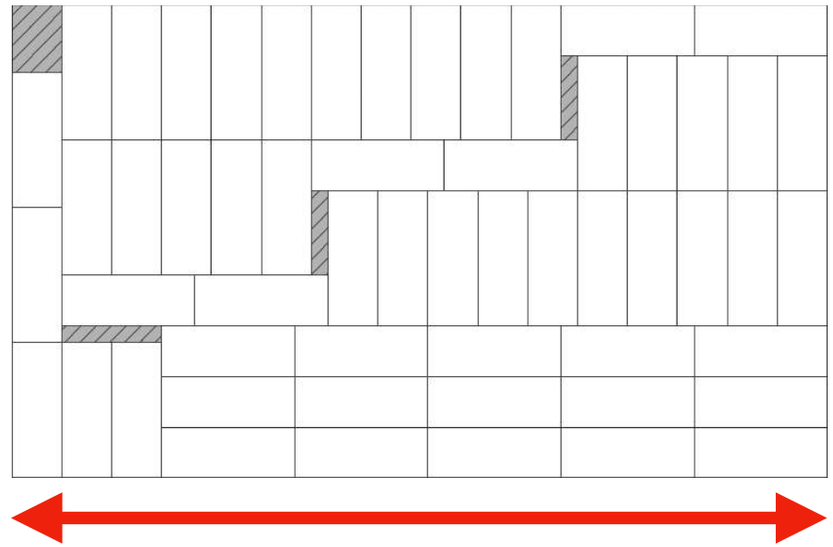
Cas général

Algorithme pour le cas général

Théorème

Fin de parenthèse

NP-difficile
Résolu quotidiennement
et approximativement



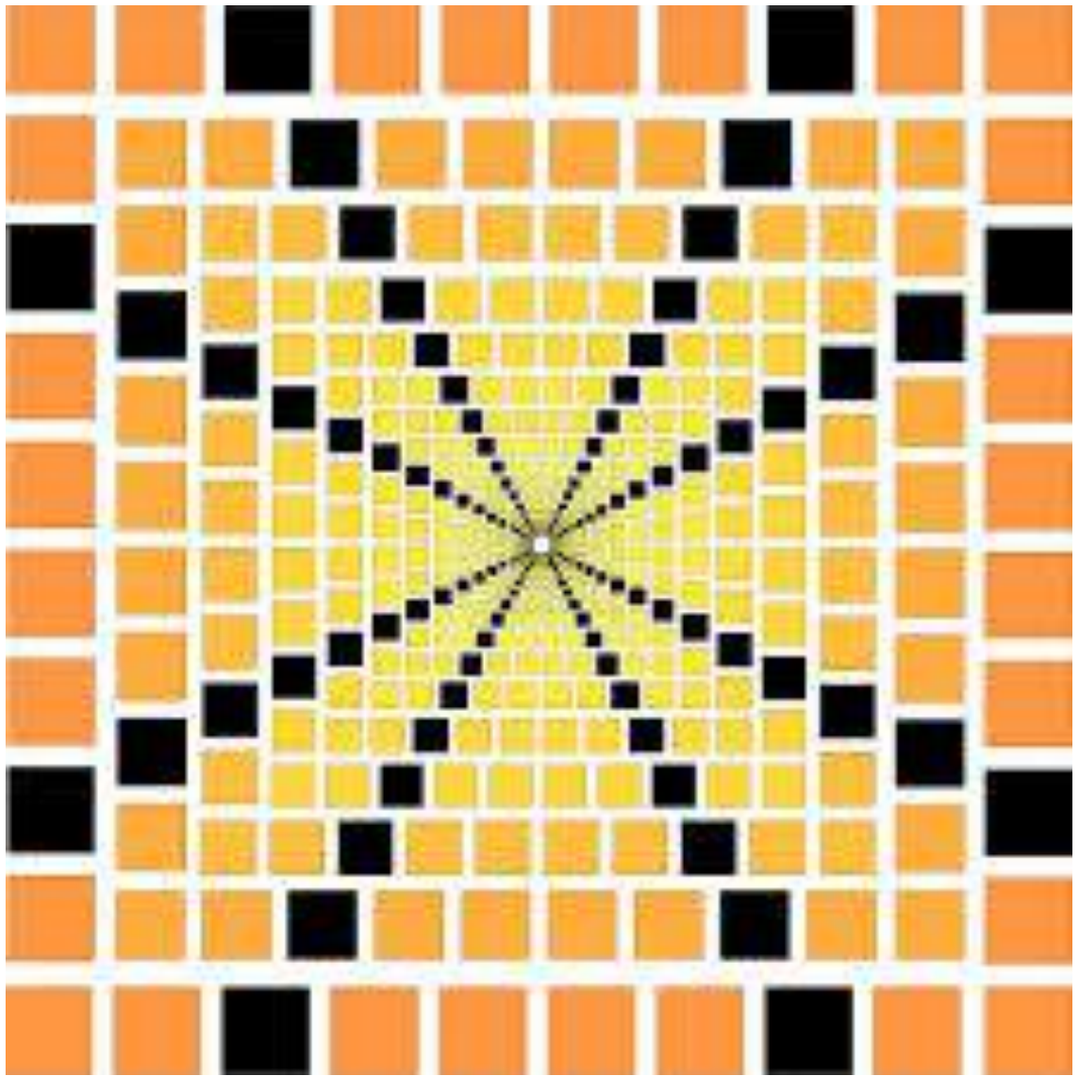
Mais comment font-ils ?

Algorithme d'approximation

Comment découper
en temps raisonnable
des rectangles
dans un rouleau de tissu
pour utiliser
le moins de longueur possible,
ou presque

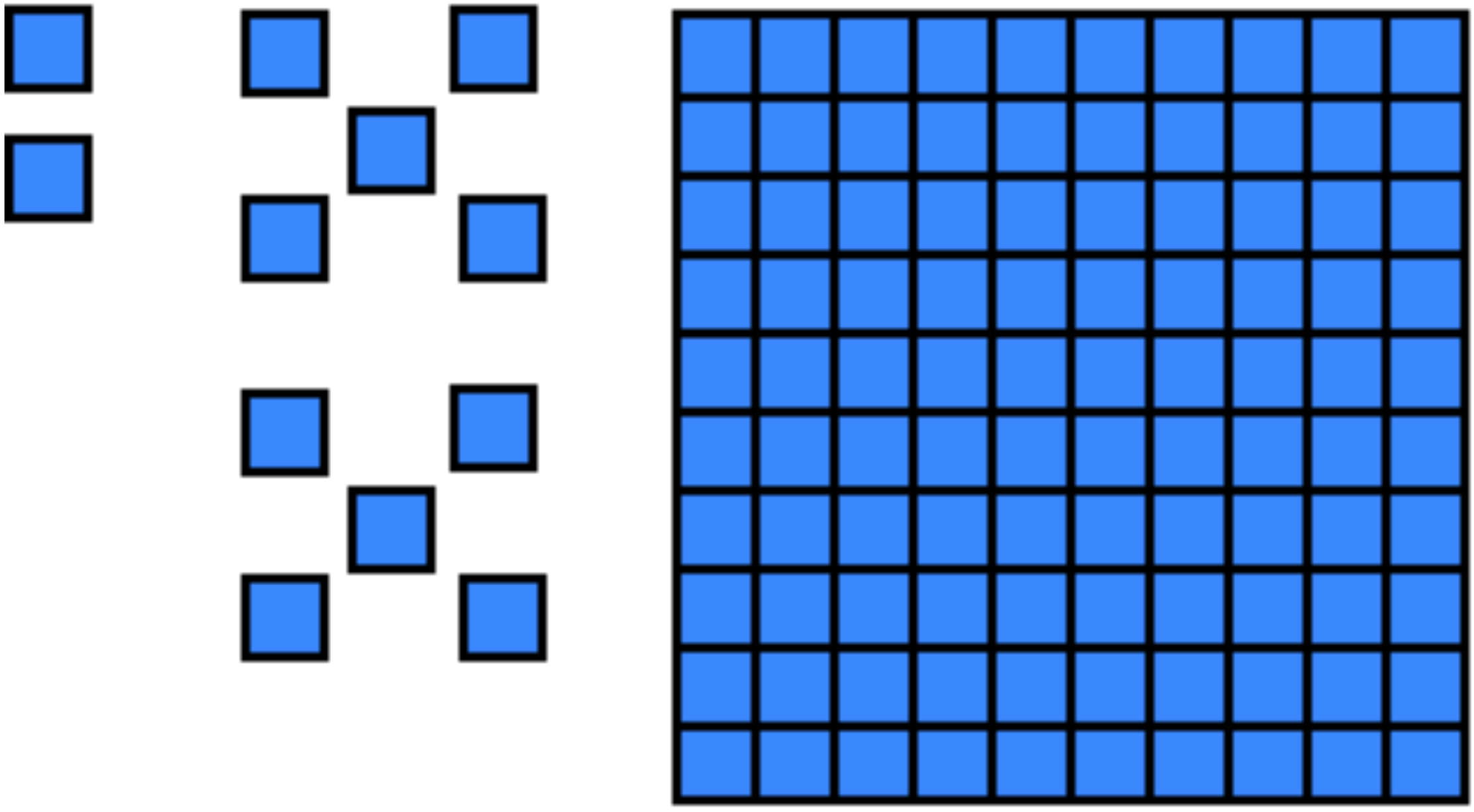
Application : découper des carrés dans une bande de tissu

Exemple intéressant :
beaucoup de petits carrés

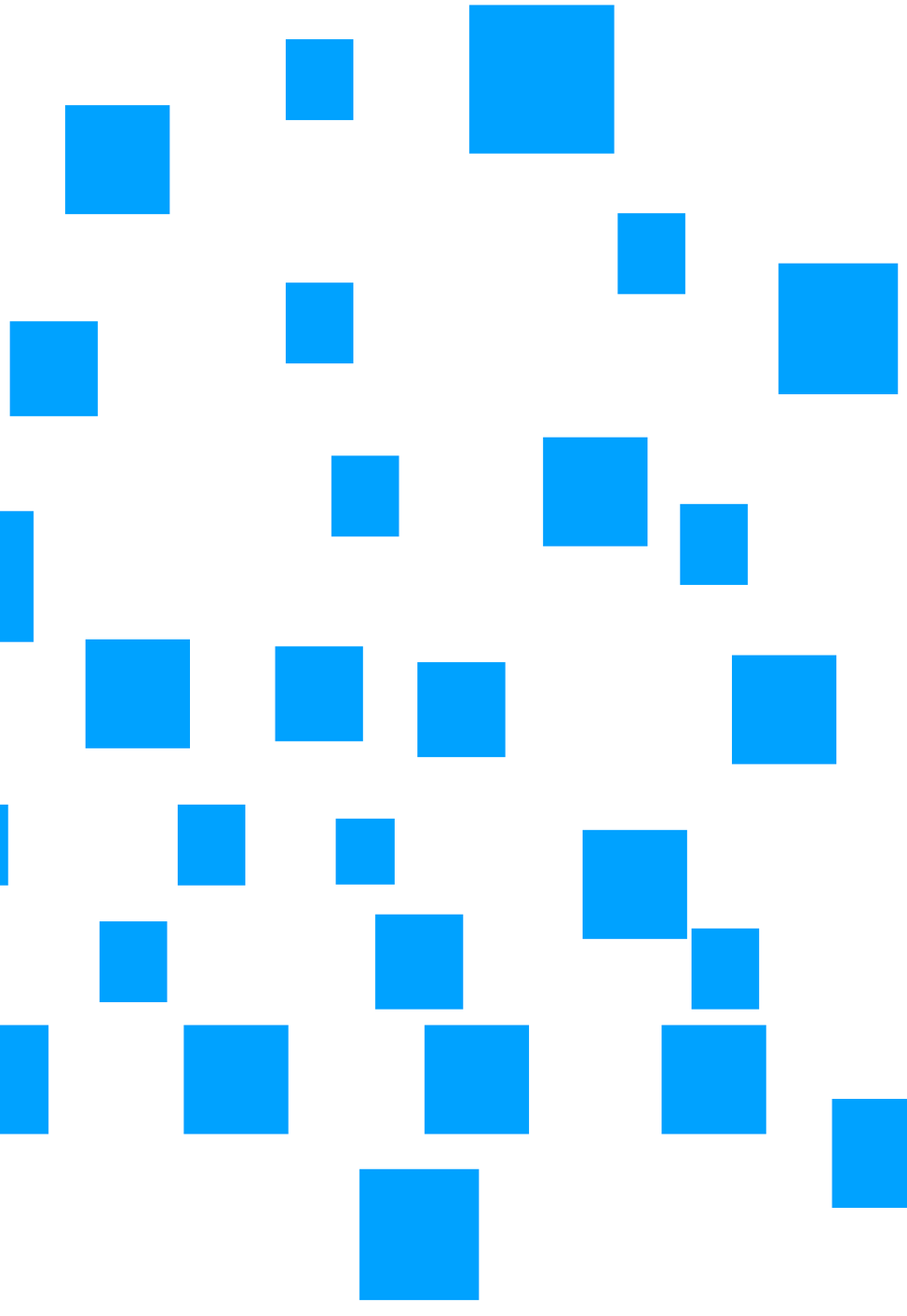


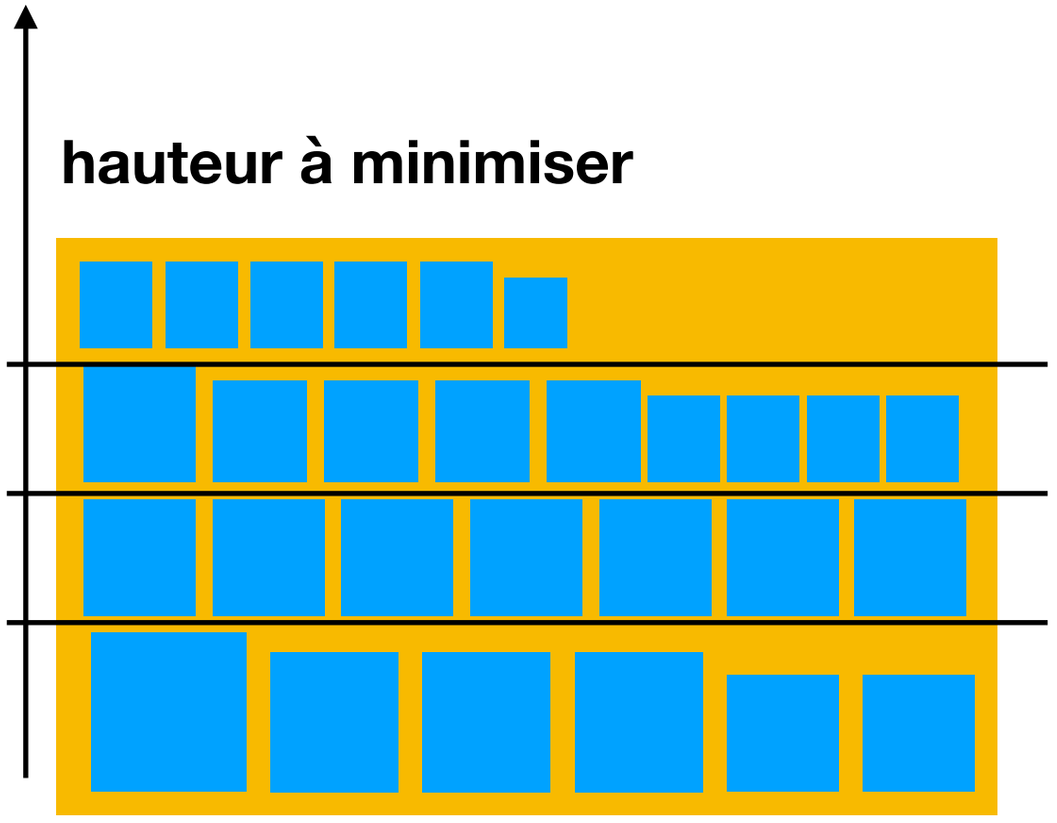
Mais comment font-ils ?

Beaucoup de petits carrés

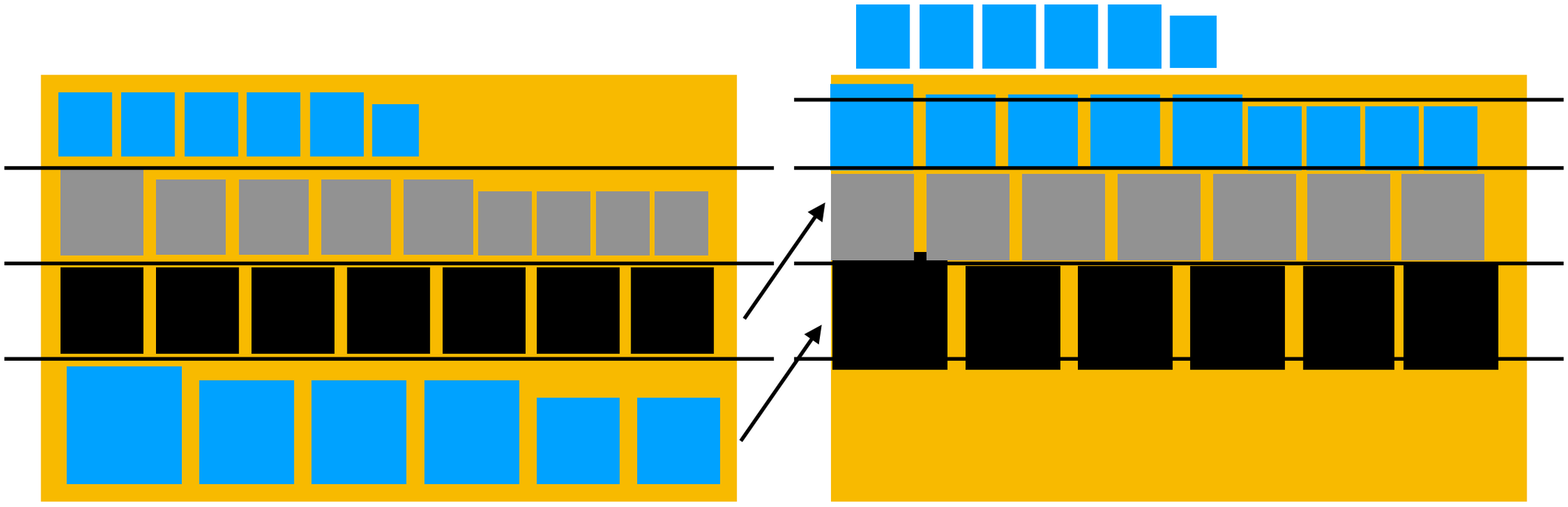


Algorithme des étagères



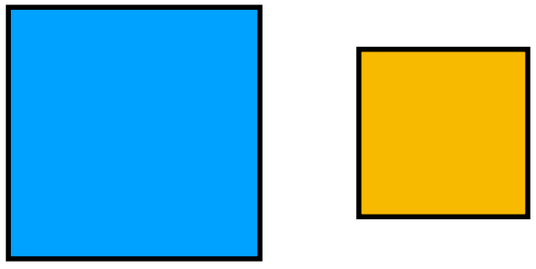


Analyse



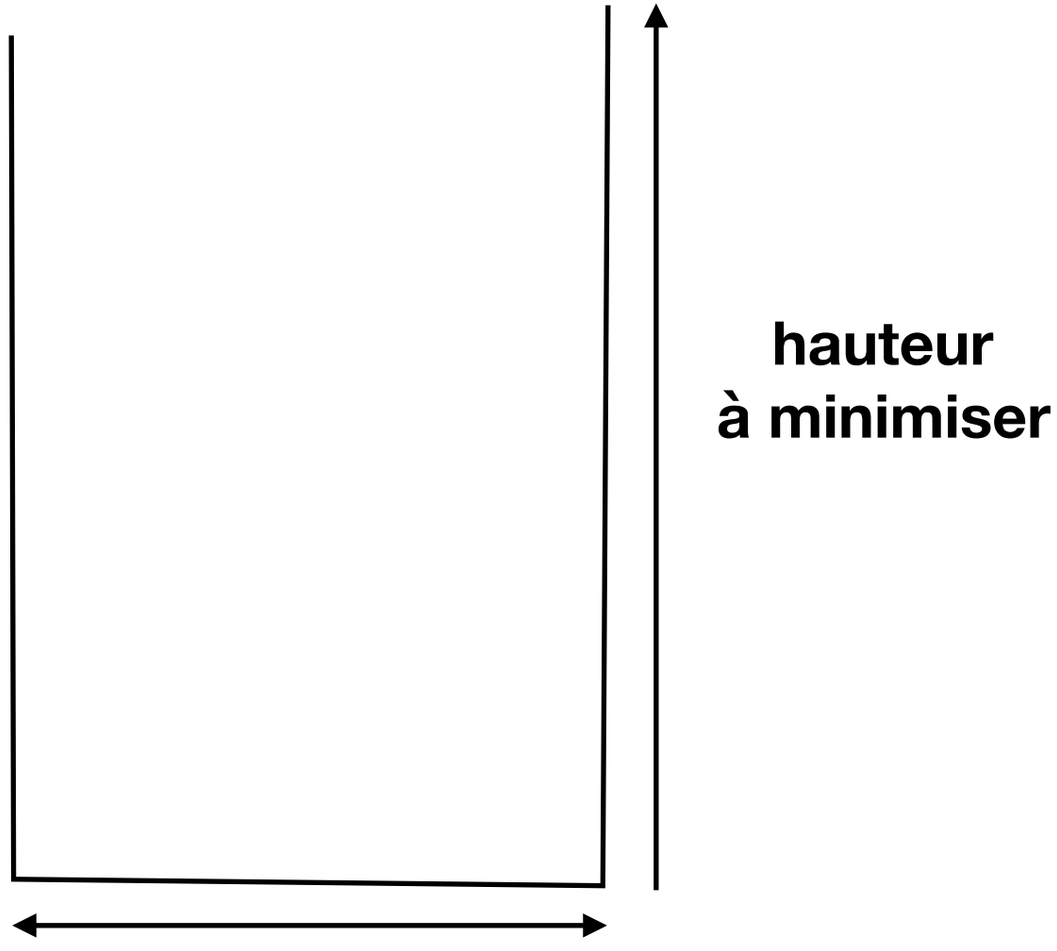
Les carrés du premier niveau recouvrent (presque) le deuxième niveau
Les carrés du deuxième niveau recouvrent (presque) le troisième niveau
La surface est (presque) toute couverte :
(presque) optimal !

Exemple très différent :
quelques gros carrés



carré
3*3

carré
2*2

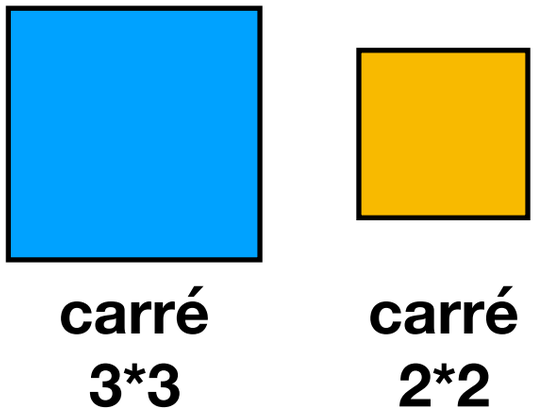


hauteur
à minimiser

7

100 carrés 3*3 et 45 carrés 2*2

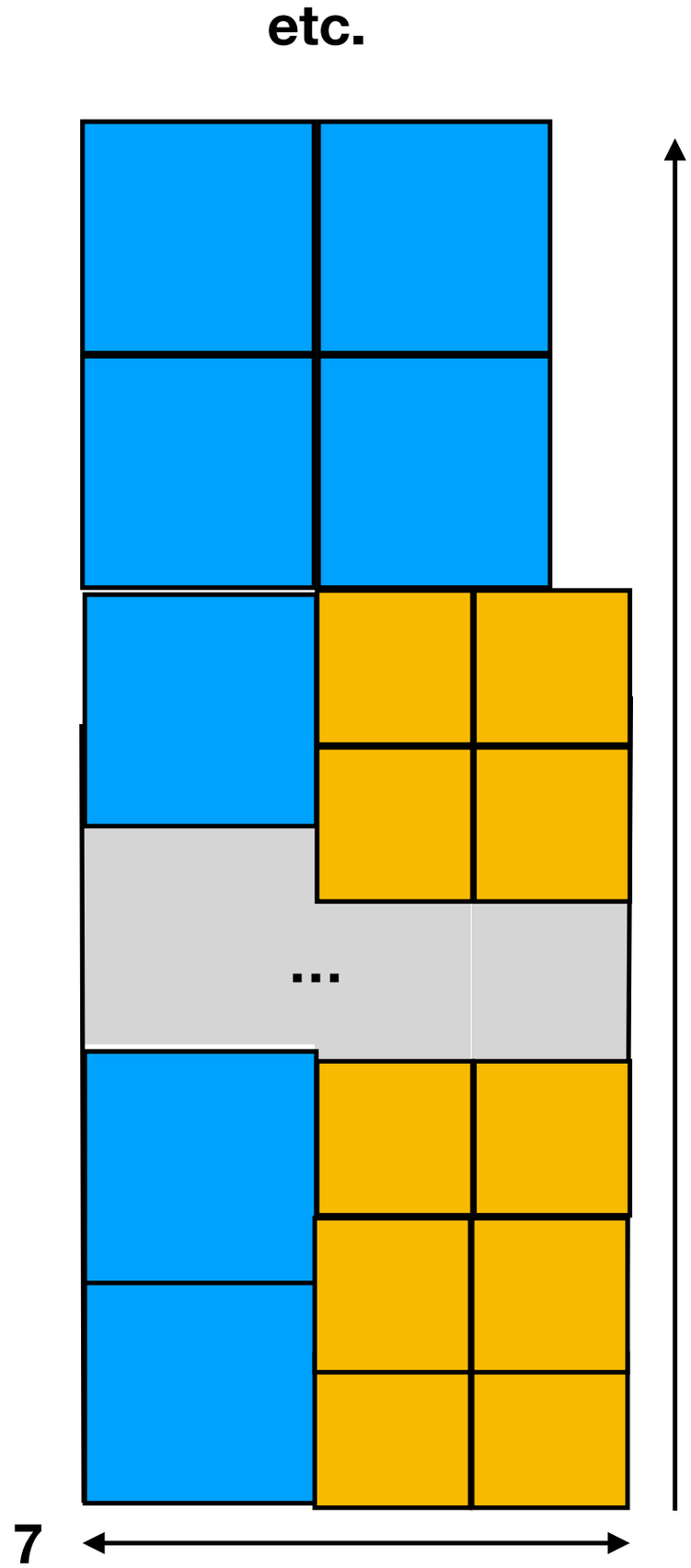
Exemple très différent :
quelques gros carrés



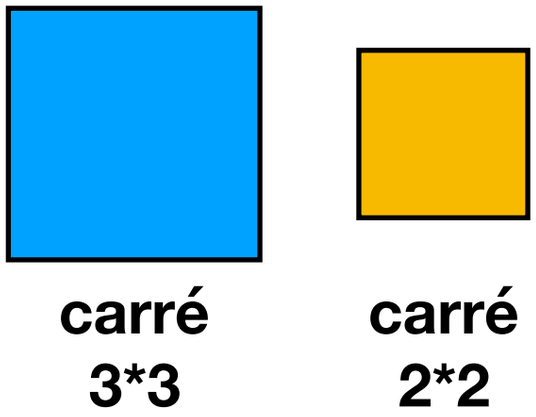
carré 3*3

carré 2*2

100 carrés 3*3 et 45 carrés 2*2



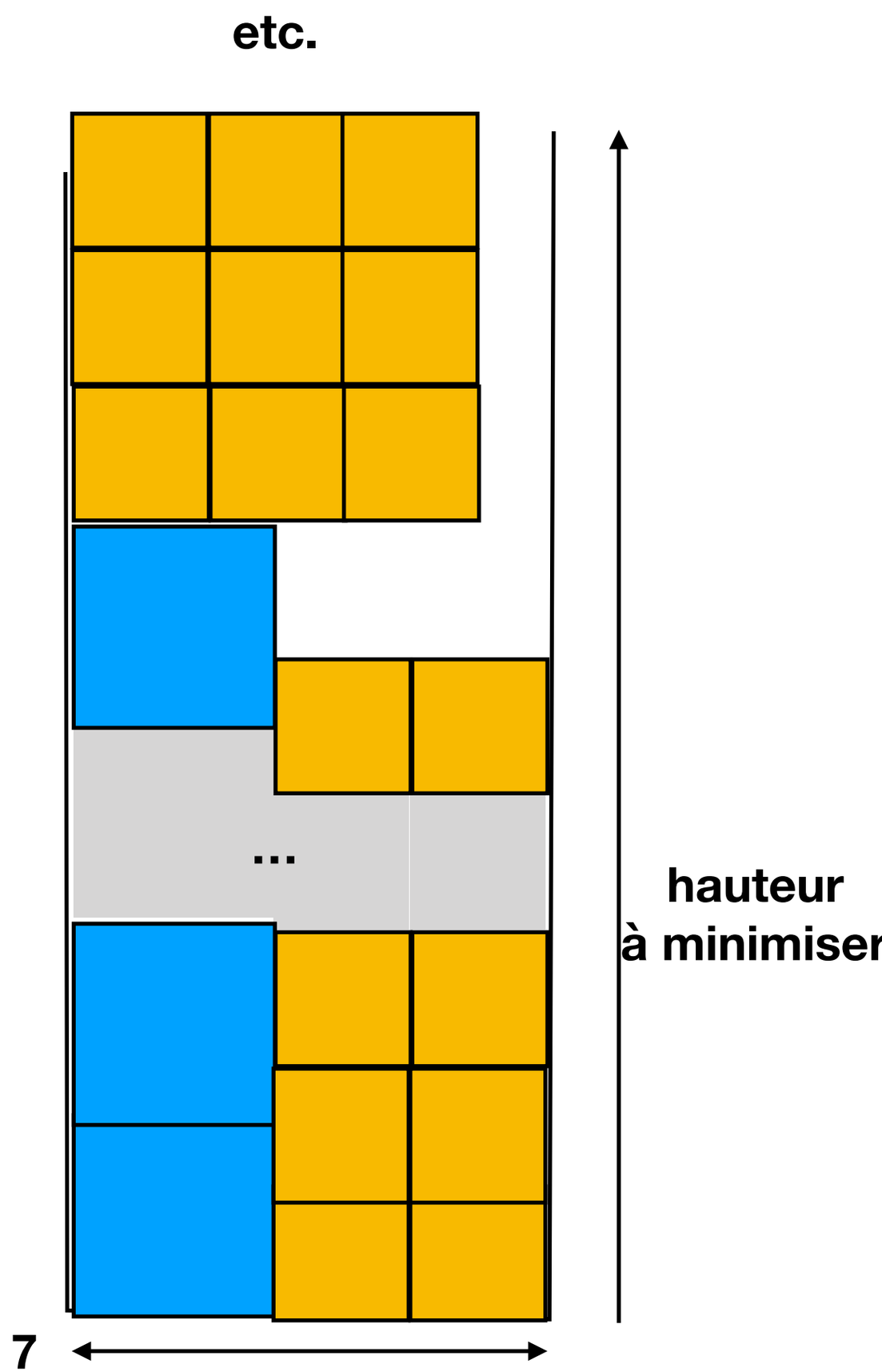
Exemple très différent :
quelques gros carrés



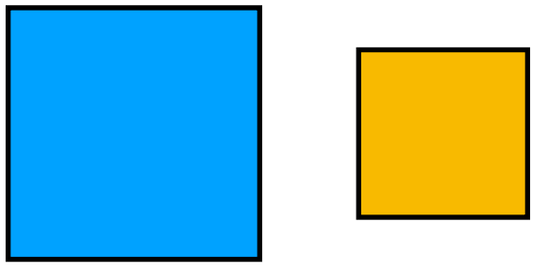
carré
3*3

carré
2*2

25 carrés 3*3 et 100 carrés 2*2



Exemple très différent :
quelques gros carrés



carré 3*3

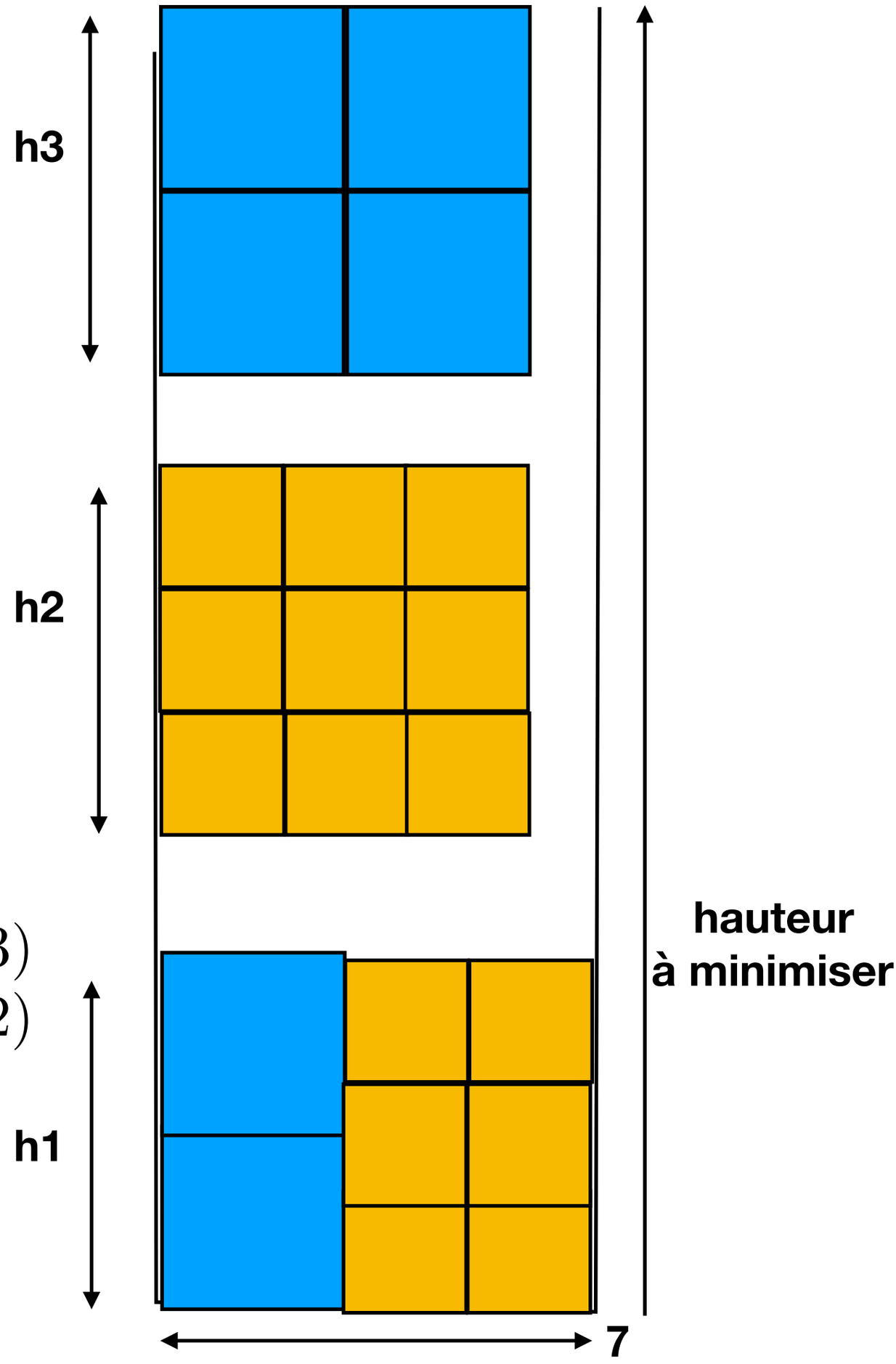
carré 2*2

un certain nombre de carrés 3*3
et
un certain nombre de carrés 2*2

$$\min(h_1 + h_2 + h_3)$$

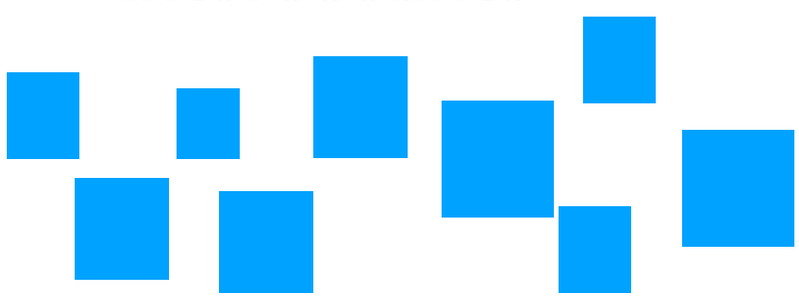
$$\begin{cases} h_1 + h_3/2 & \geq 3(\text{nbr carrés } 3 \times 3) \\ h_1/2 + h_2/3 & \geq 2(\text{nbr carrés } 2 \times 2) \end{cases}$$

Programme linéaire
Solution en nombres réels
on arrondit



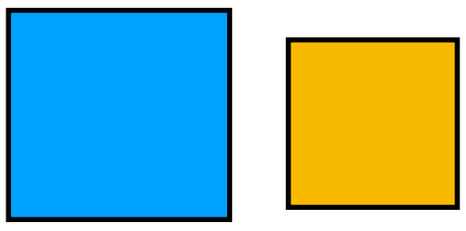
Appliquer la méthode

1e famille d'exemples intéressants ⟷ **Algorithme simple pour ces exemples** ⟷ **Analyse**



Algorithme des étagères — **OPT > Surface**

2e famille d'exemples très différents ⟷ **Algorithme simple pour ces autres exemples** ⟷ **Analyse**



carrés 3*3 et 2*2

Programme linéaire puis arrondi — **OPT > Programme linéaire**

Combinaison d'exemples ⟷ **Combinaison des algorithmes**

Comment combiner les deux idées ?

h3

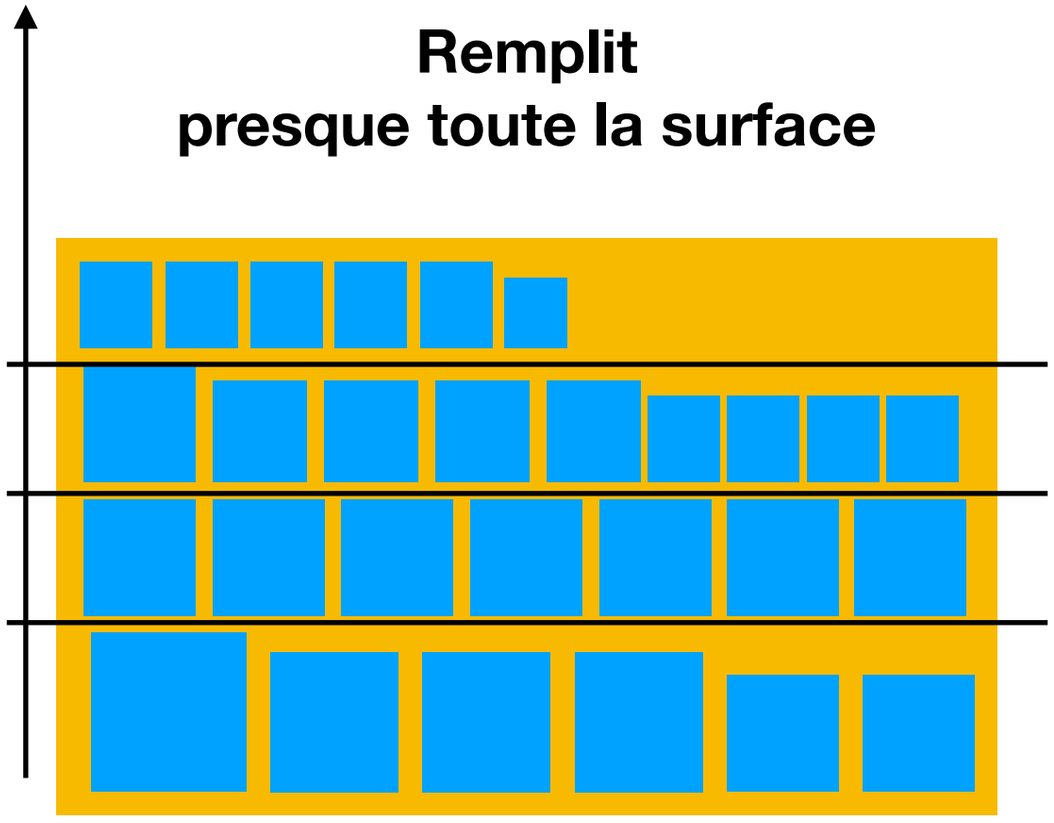
h2

h1

7

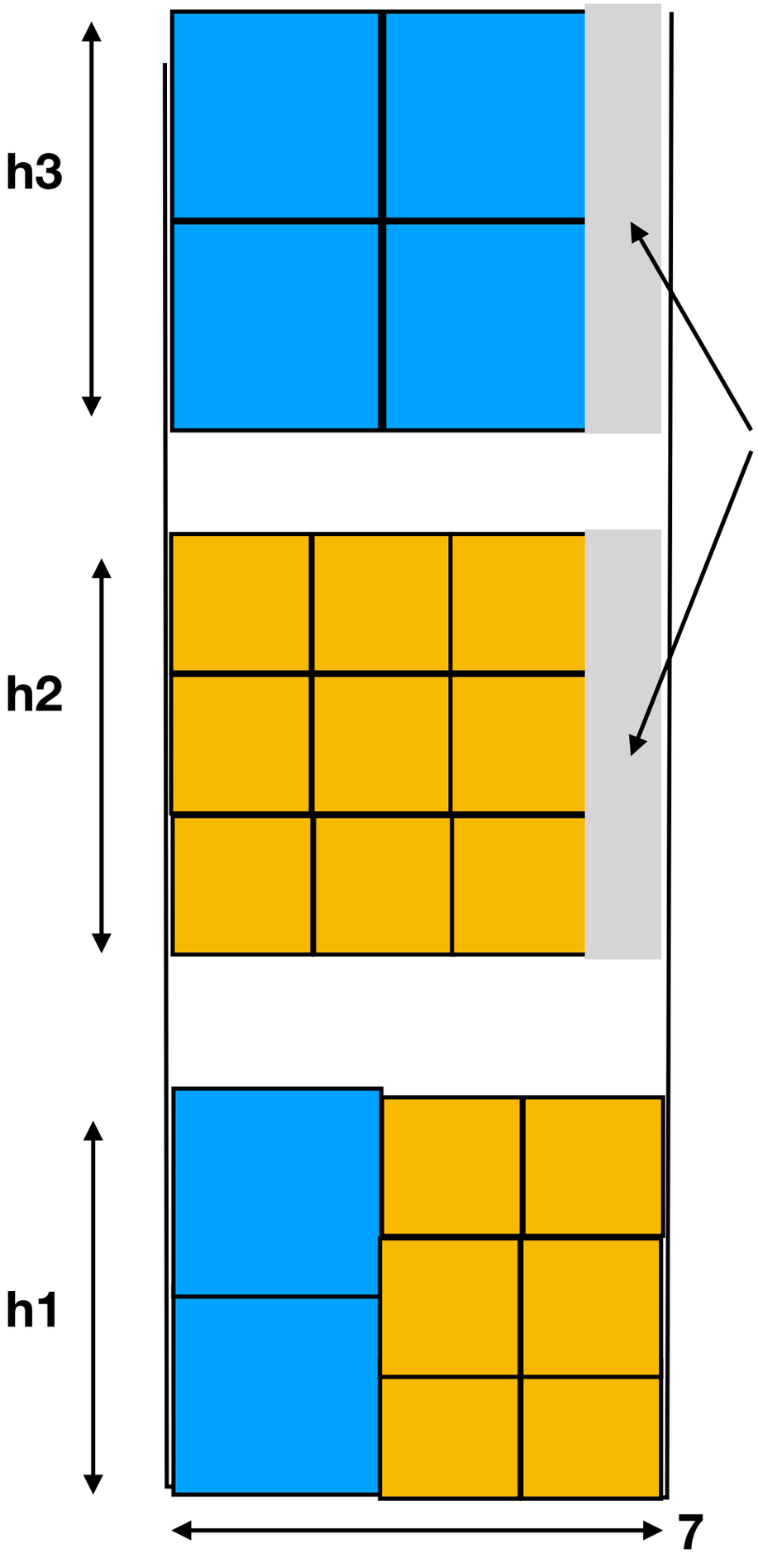
Programme linéaire
+ arrondis

Remplit
presque toute la surface



L’algorithme de ma grand-mère

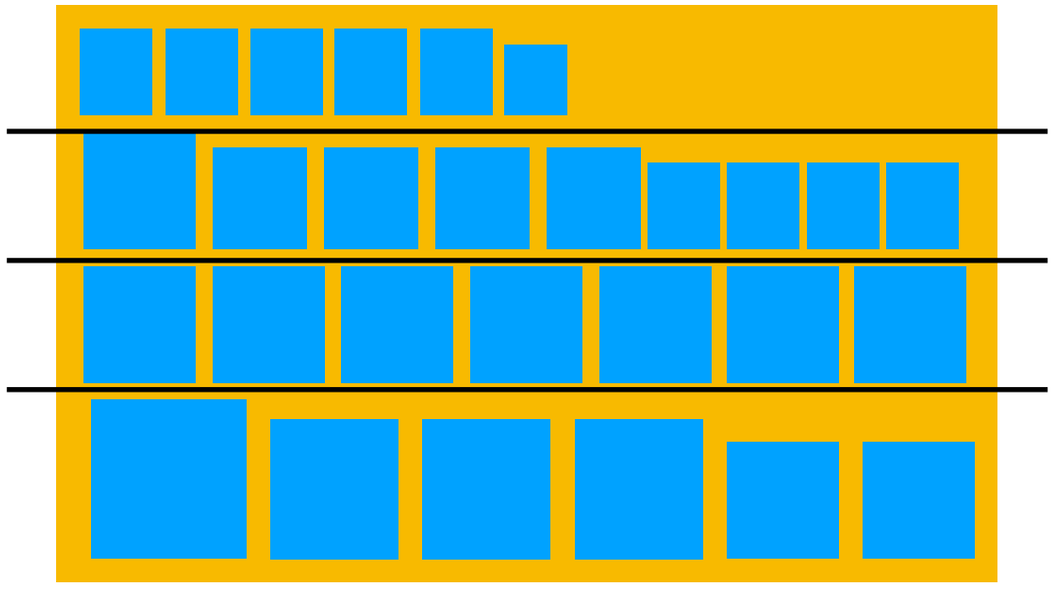
On place d'abord les grosses pièces
puis on utilise les chutes de tissu
pour les petites pièces

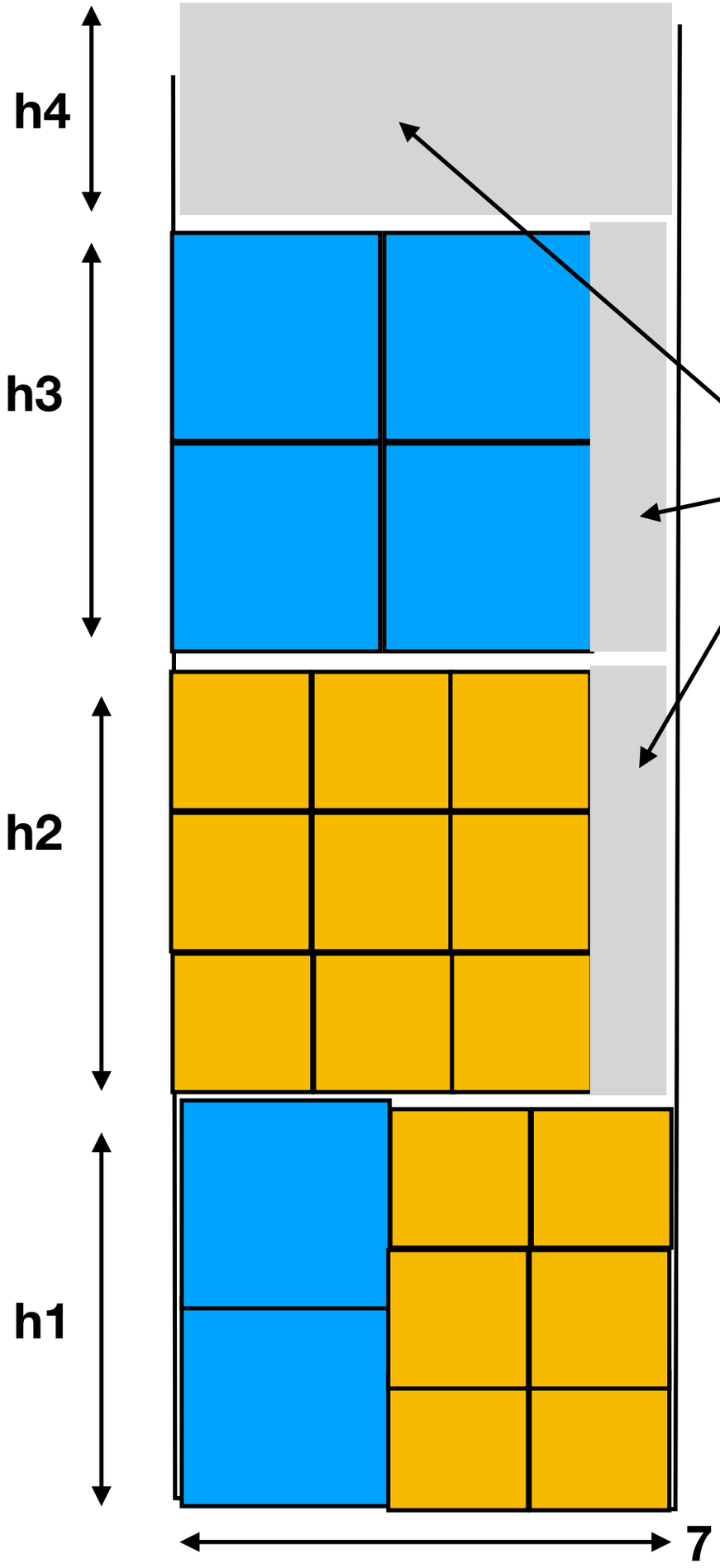


Programme linéaire
+ arrondis

zones libres pour insérer des petits carrés par l'algorithme des étagères

Remplit
presque toute la surface





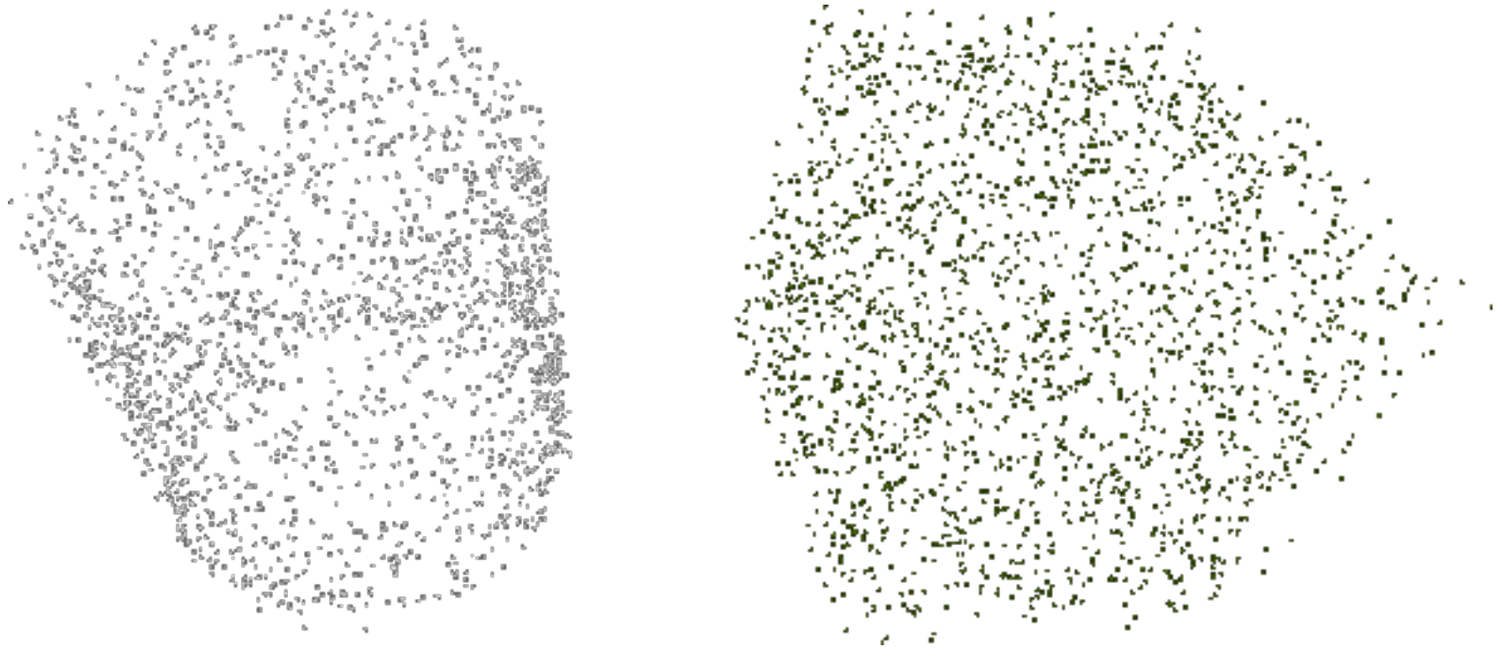
zones libres pour insérer des petits carrés par l'algorithme des étagères

$$\min(h_1 + h_2 + h_3 + h_4)$$

$$\begin{cases} h_1 + h_3/2 & \geq 3(\text{nbr carrés } 3 \times 3) \\ h_1/2 + h_2/3 & \geq 2(\text{nbr carrés } 2 \times 2) \\ h_2 + h_3 + 7h_4 & \geq (\text{surface petits carrés}) \end{cases}$$

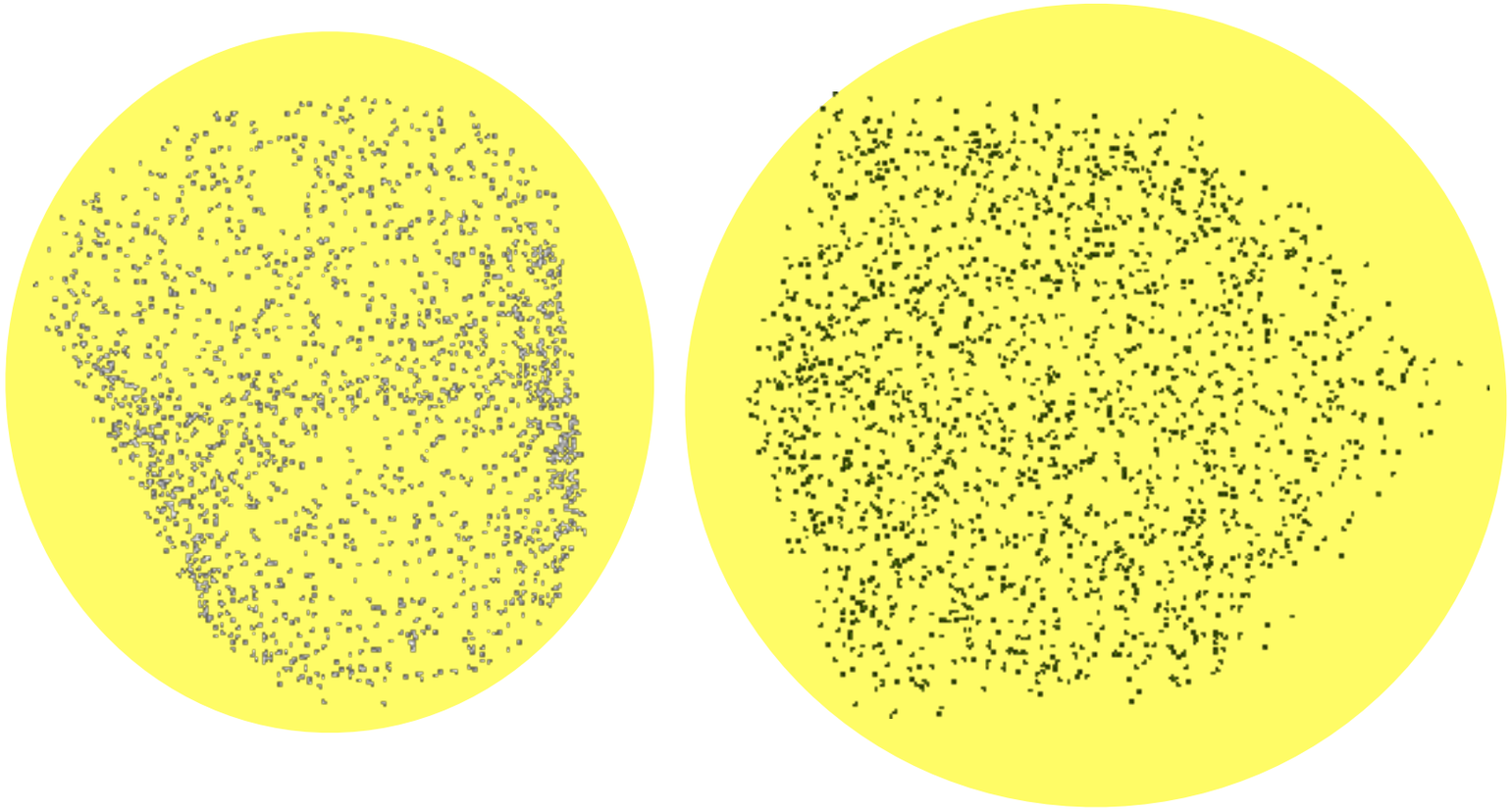
Coupe maximum dans le plan

Entrée du problème :
On a des points

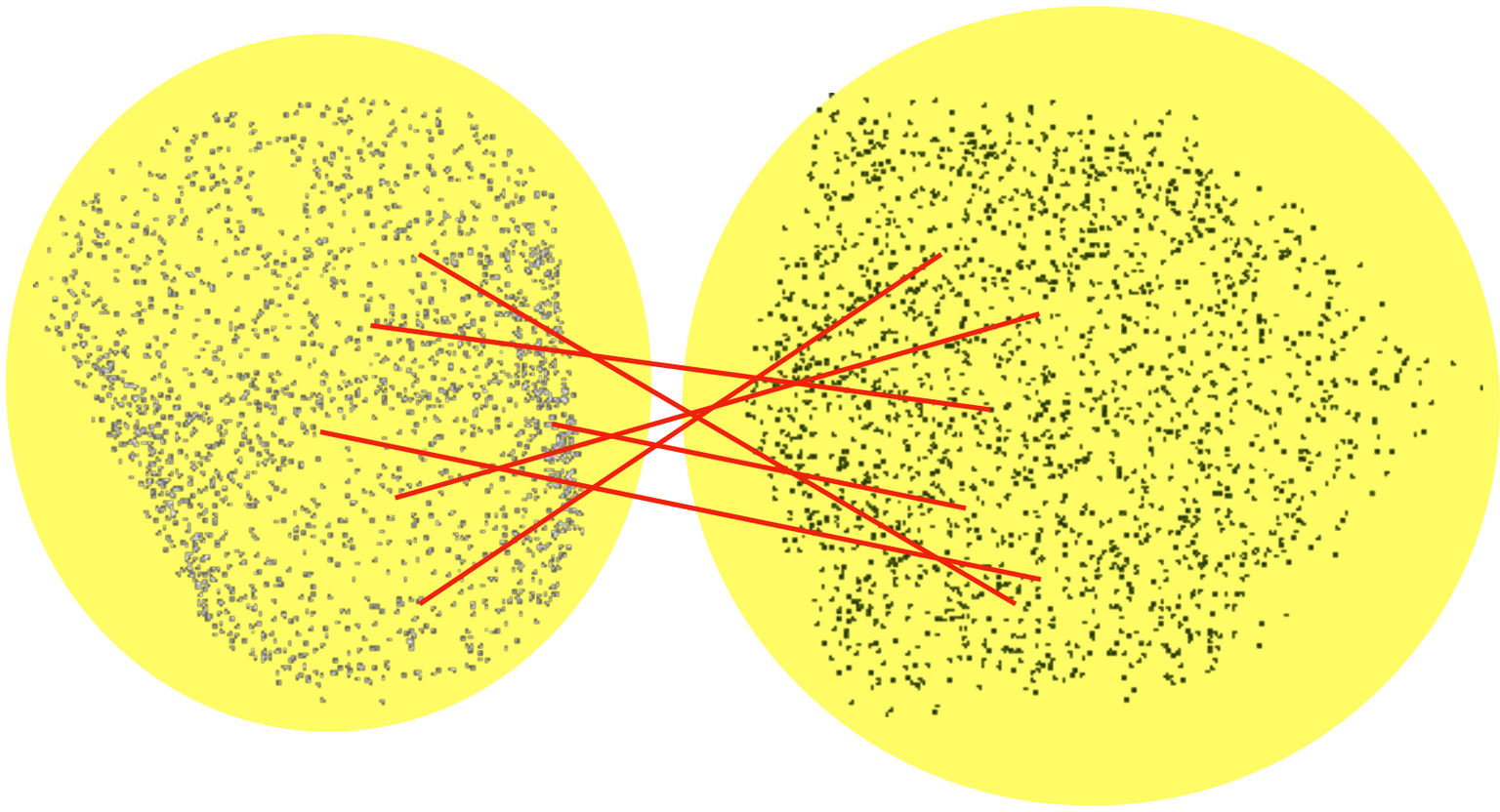


Maxcut

Résultat :
On veut les couper
en deux parties
à bonne distance
l’une de l’autre



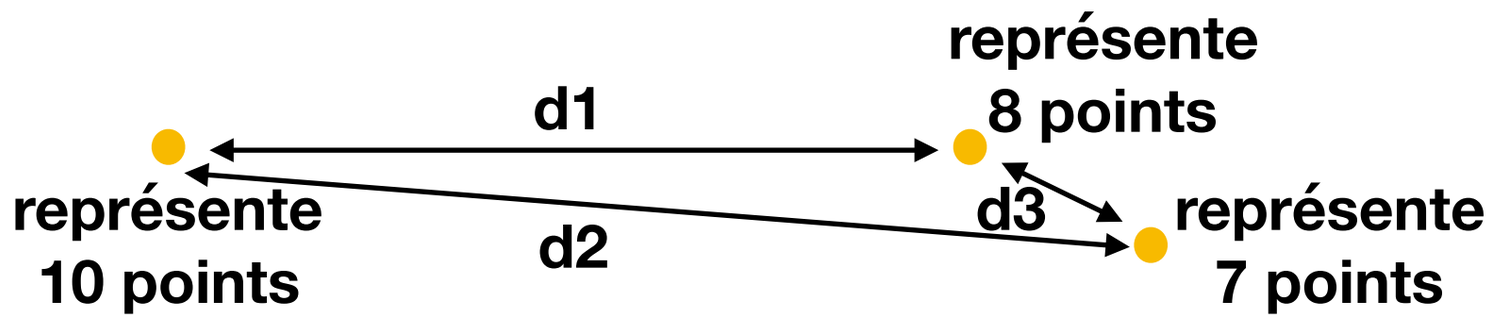
NP-difficile



Maxcut :
maximiser la somme des distances
entre les deux parties

Exemple : presque tous les points sont aux mêmes endroits

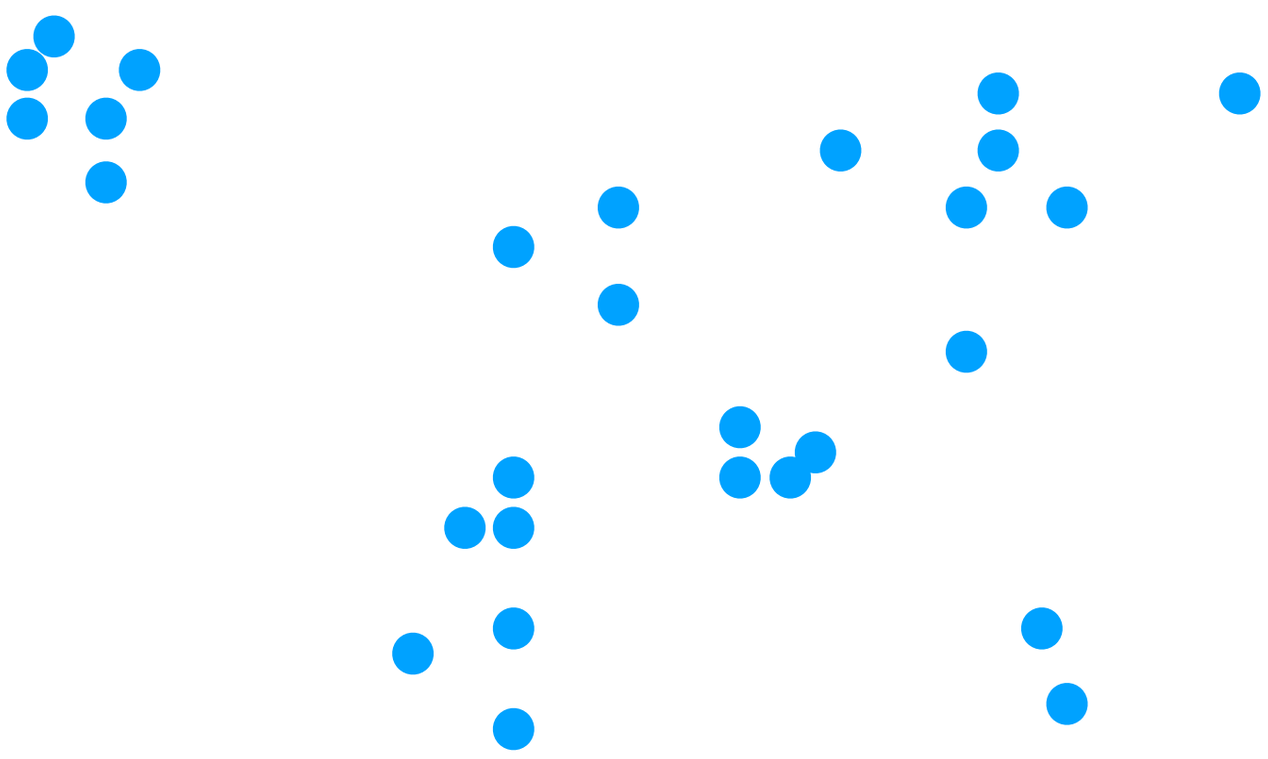
Idée : on arrondit les positions



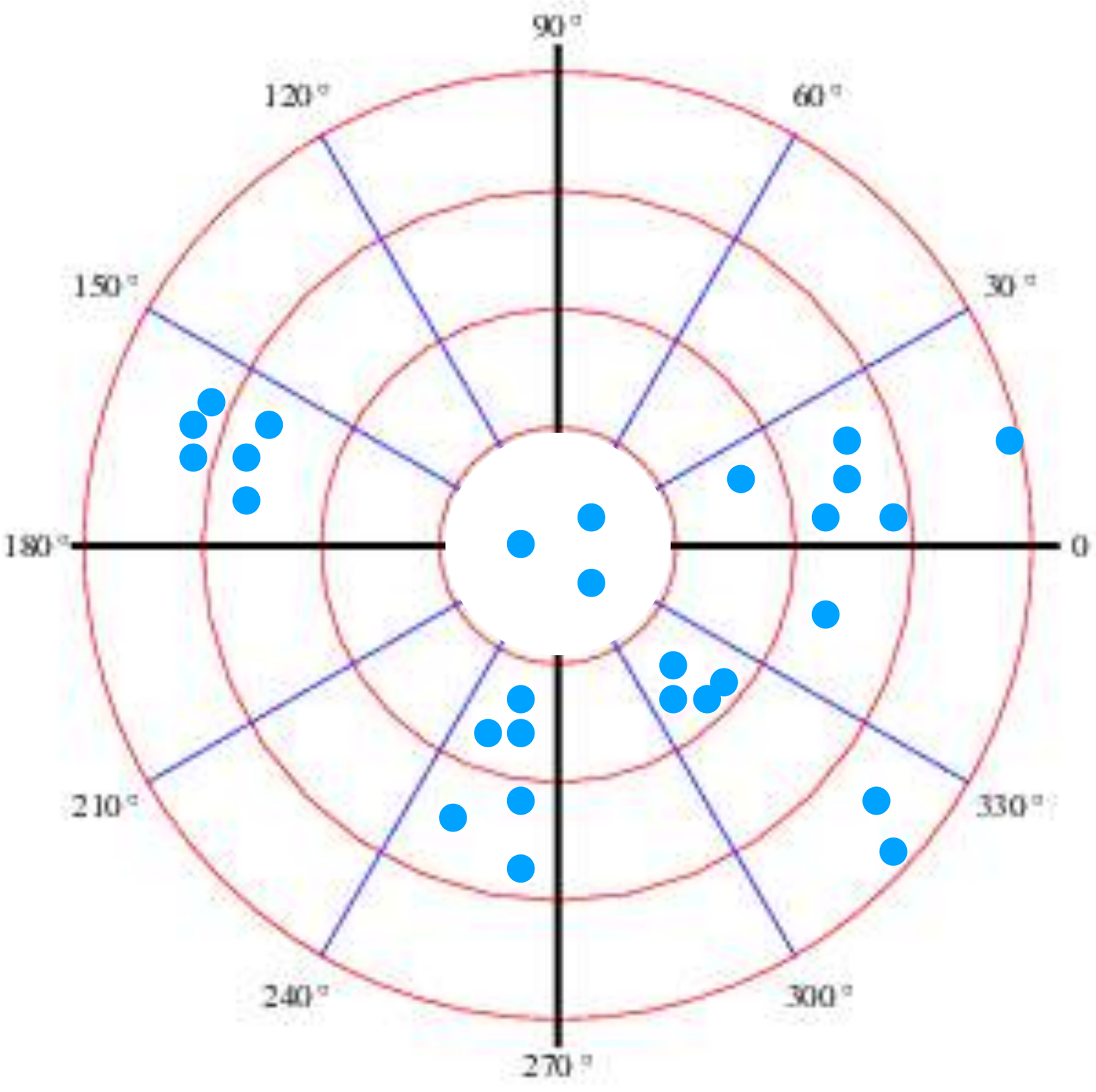
10*8*d1+10*7*d2
ou
10*7*d2+8*7*d3
ou
10*8*d1+8*7*d3

Algorithme :
on regarde les 3 possibilités
et on choisit celle
qui donne le maximum

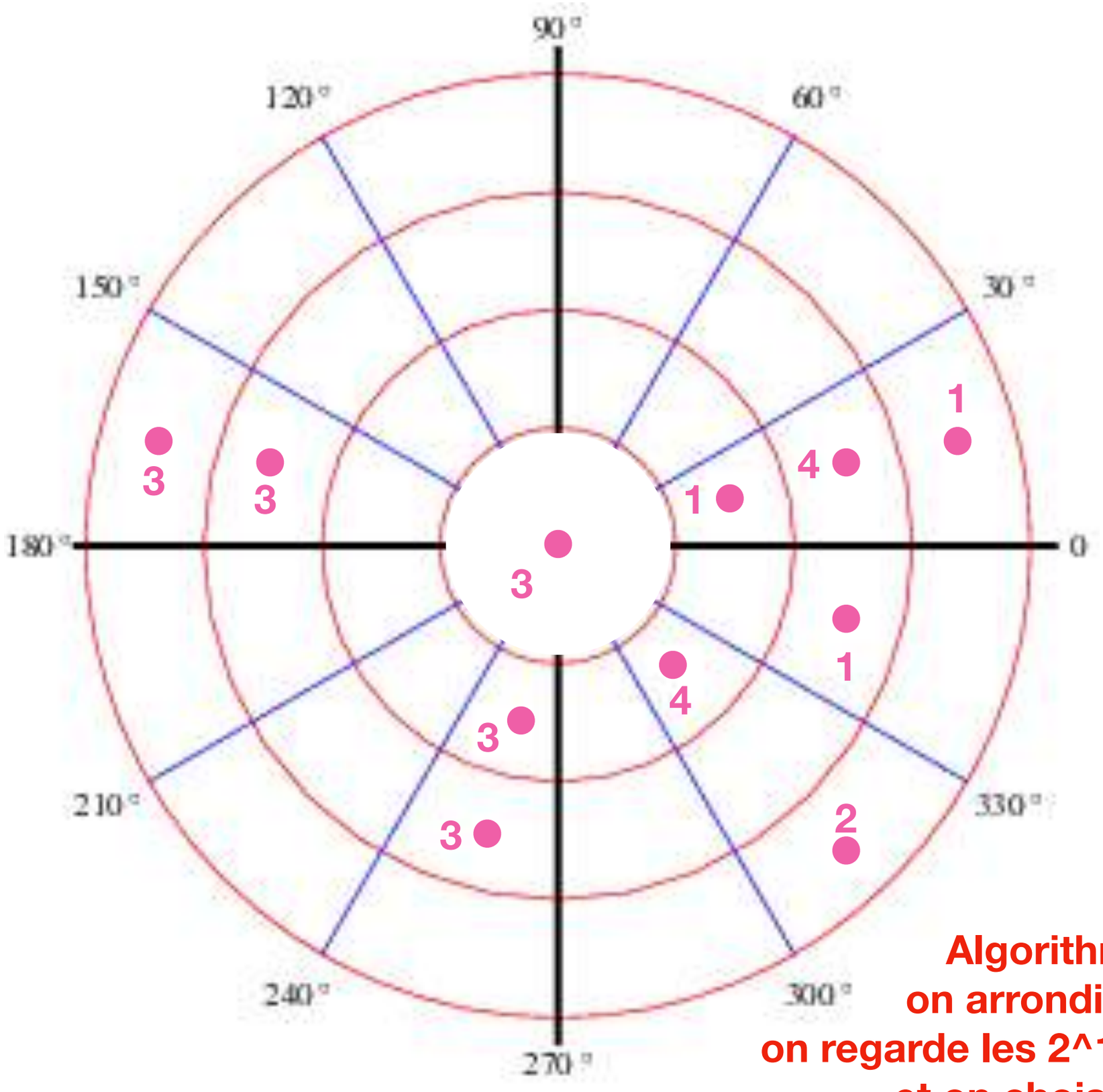
Comment arrondir les positions en général ?



Idée : coordonnées polaires



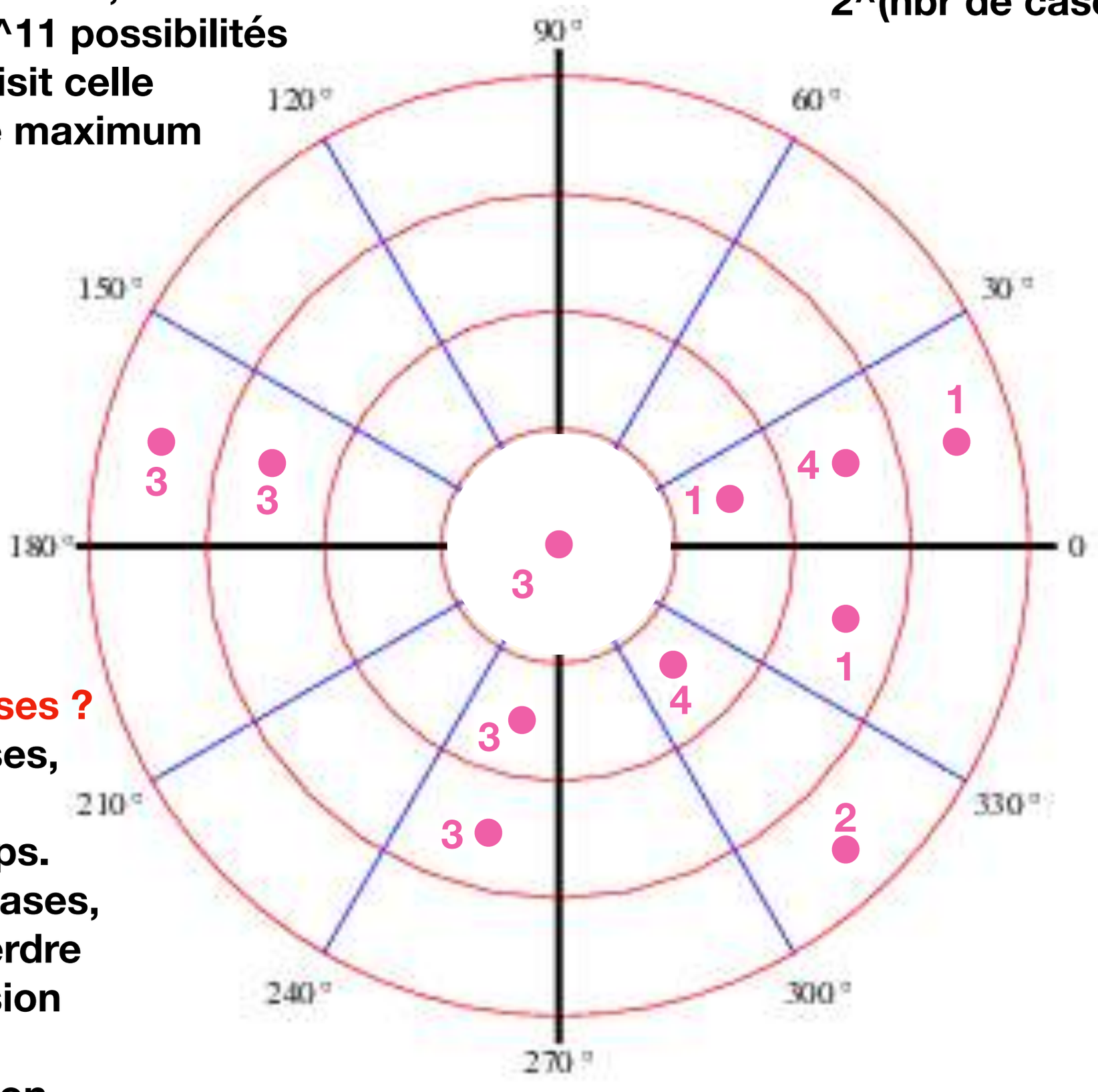
Positions après arrondi



Algorithme :
on arrondit, puis
on regarde les 2^11 possibilités
et on choisit celle
qui donne le maximum

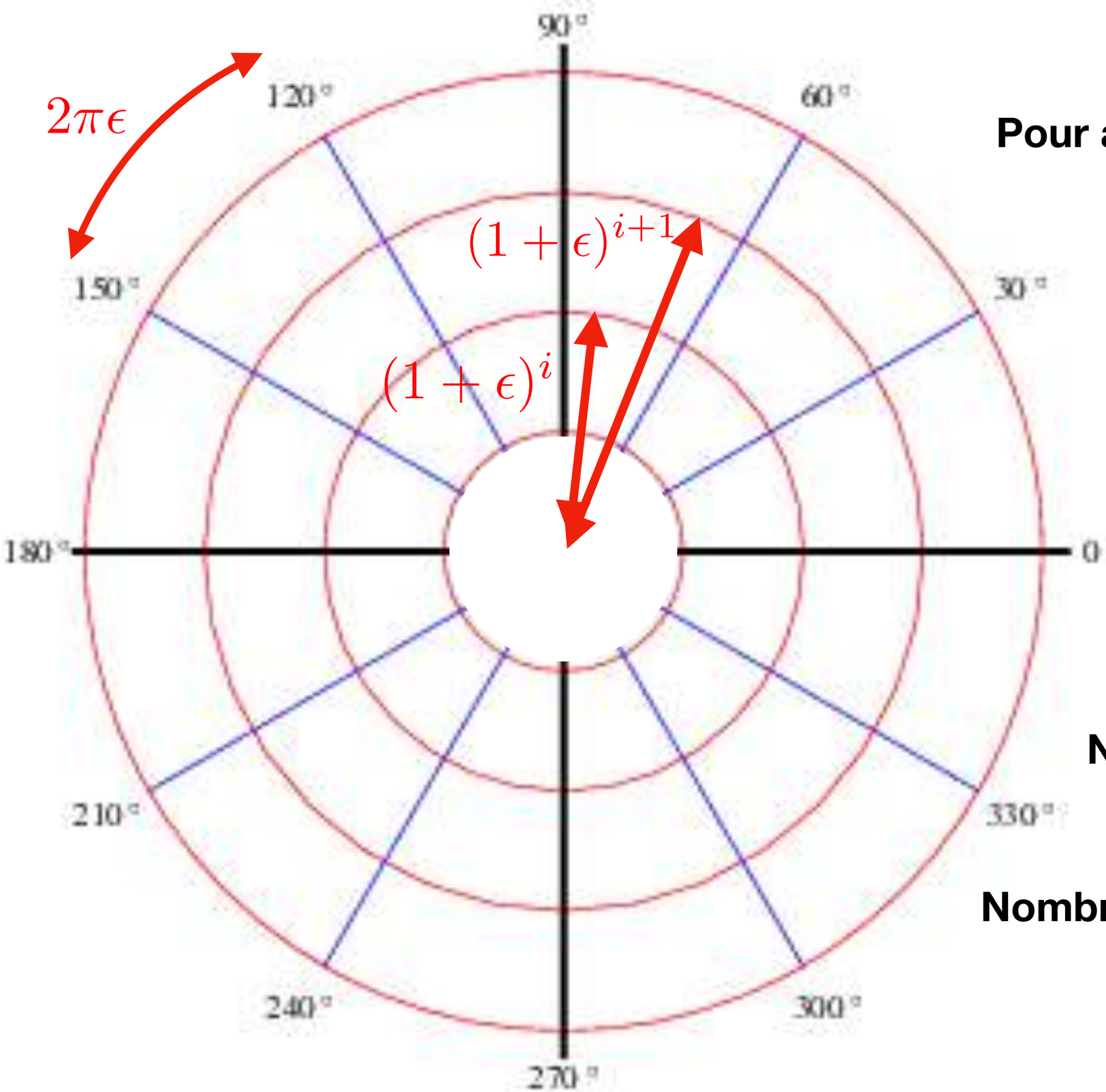
Algorithme :
après arrondi,
on regarde les 2^11 possibilités
et on choisit celle
qui donne le maximum

Combien de possibilités en général ?
2^(nbr de cases)



Combien de cases ?
Si trop de cases,
ça prend
trop longtemps.
Si trop peu de cases,
l'arrondi fait perdre
trop de précision
et
l'approximation
n'est pas bonne.

La bonne réponse :
ni trop, ni trop peu



Pour avoir erreur relative de ϵ

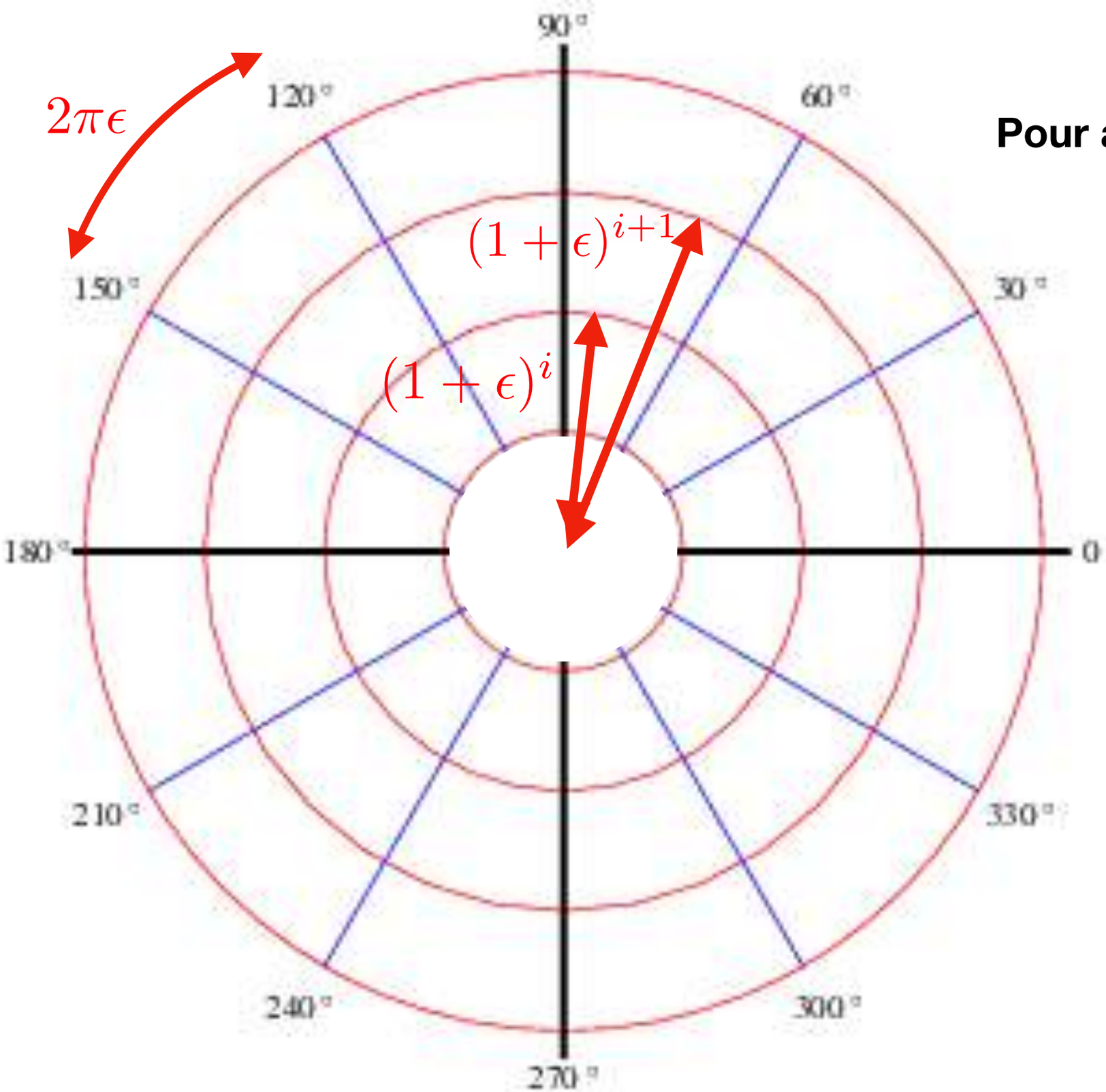
Nombre de cases : $\ln n/\epsilon^2$

Nombre de possibilités : $n^{O(1/\epsilon^2)}$

Temps “raisonnable” …

La bonne réponse :

ni trop, ni trop peu



Pour avoir erreur relative de ϵ

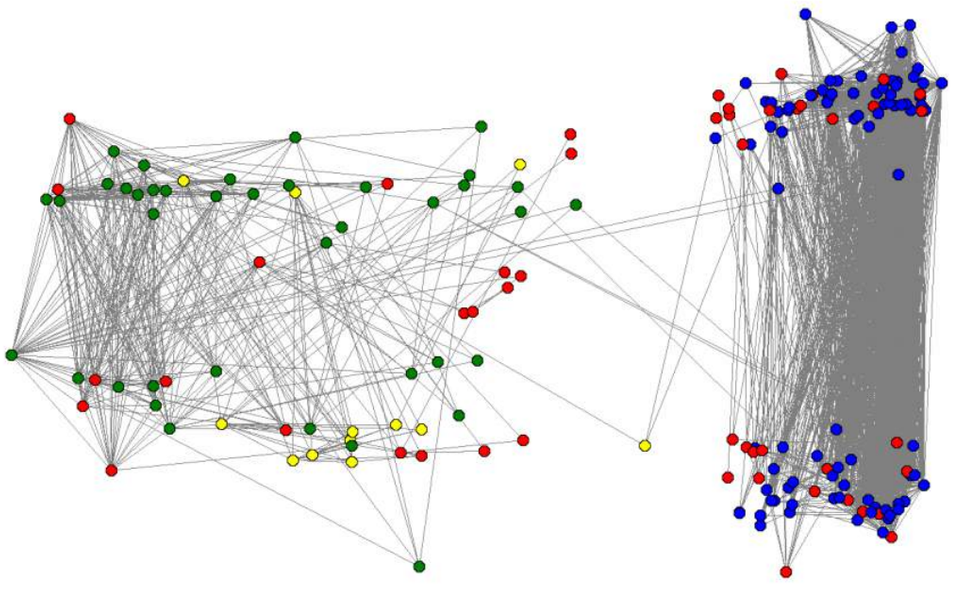
Preuve :

OPT > (n/2)*(diamètre)

Algorithme dans le plan euclidien :
on regarde les coordonnées polaires
par rapport au centre de gravité
on arrondit les positions
on regarde toutes les possibilités
et on choisit celle
qui donne le maximum

Théorème

Si l'échelle de l'arrondi est bien choisie
alors le résultat a une erreur relative au plus ϵ
et le temps de calcul est au plus $n^{O(1/\epsilon^2)}$



Le cas euclidien en 2 dimensions
n'est qu'un exemple :
on peut généraliser dans
un espace métrique quelconque !

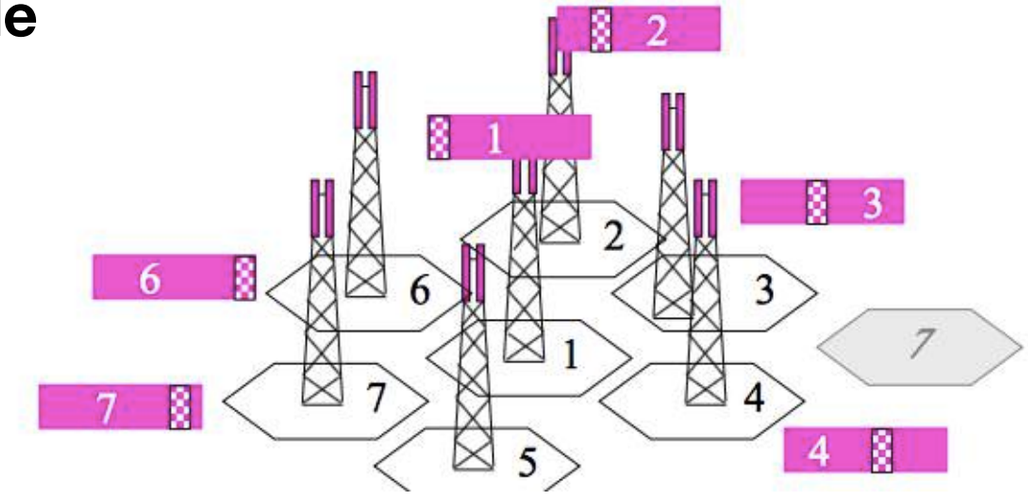
Idée : remplacer coordonnées polaires
par échantillon aléatoire.

Disques disjoints

Donner une même bande de fréquence à un ensemble de sommets d'un graphe

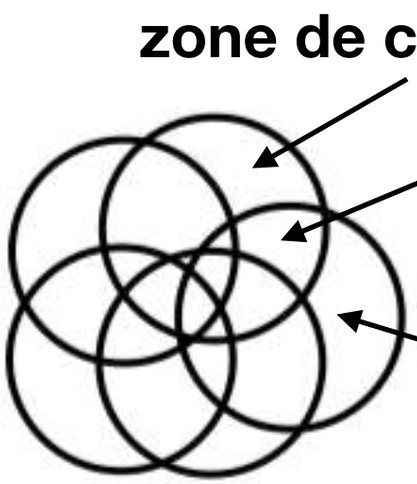
Allocation de fréquences pour téléphonie mobile

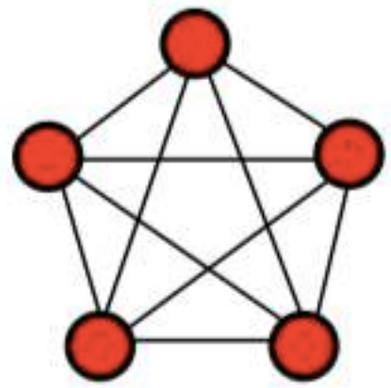
Réseau cellulaire GSM
cellules géographiques (hexagonales)
tour de communication dans la cellule
bandes de fréquences différentes
pour éviter interférences
entre cellules voisines



Répartition des fréquences

Assignation de fréquences dans un réseau sans fil



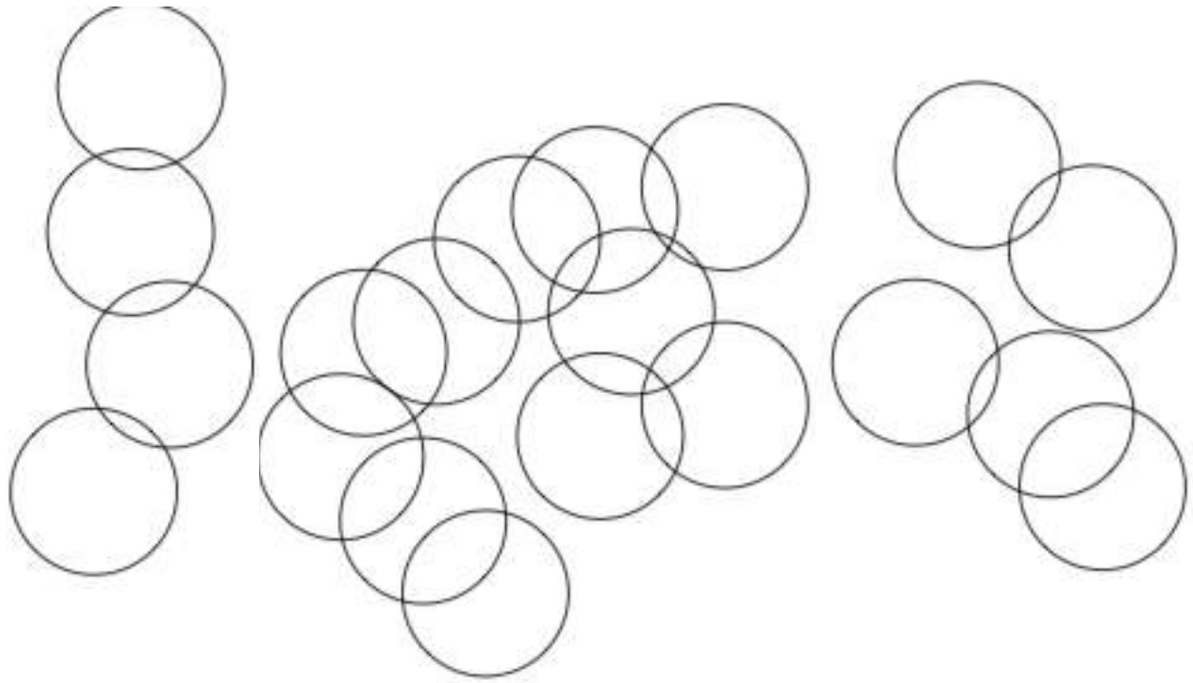


sommet =
zone de communication
arête =
interférences possibles

pas d’arêtes entre sommets qui reçoivent une même couleur

Entrée :
ensemble de disques de rayon 1

Sortie :
sous-ensemble
de disques disjoints
de cardinal maximum

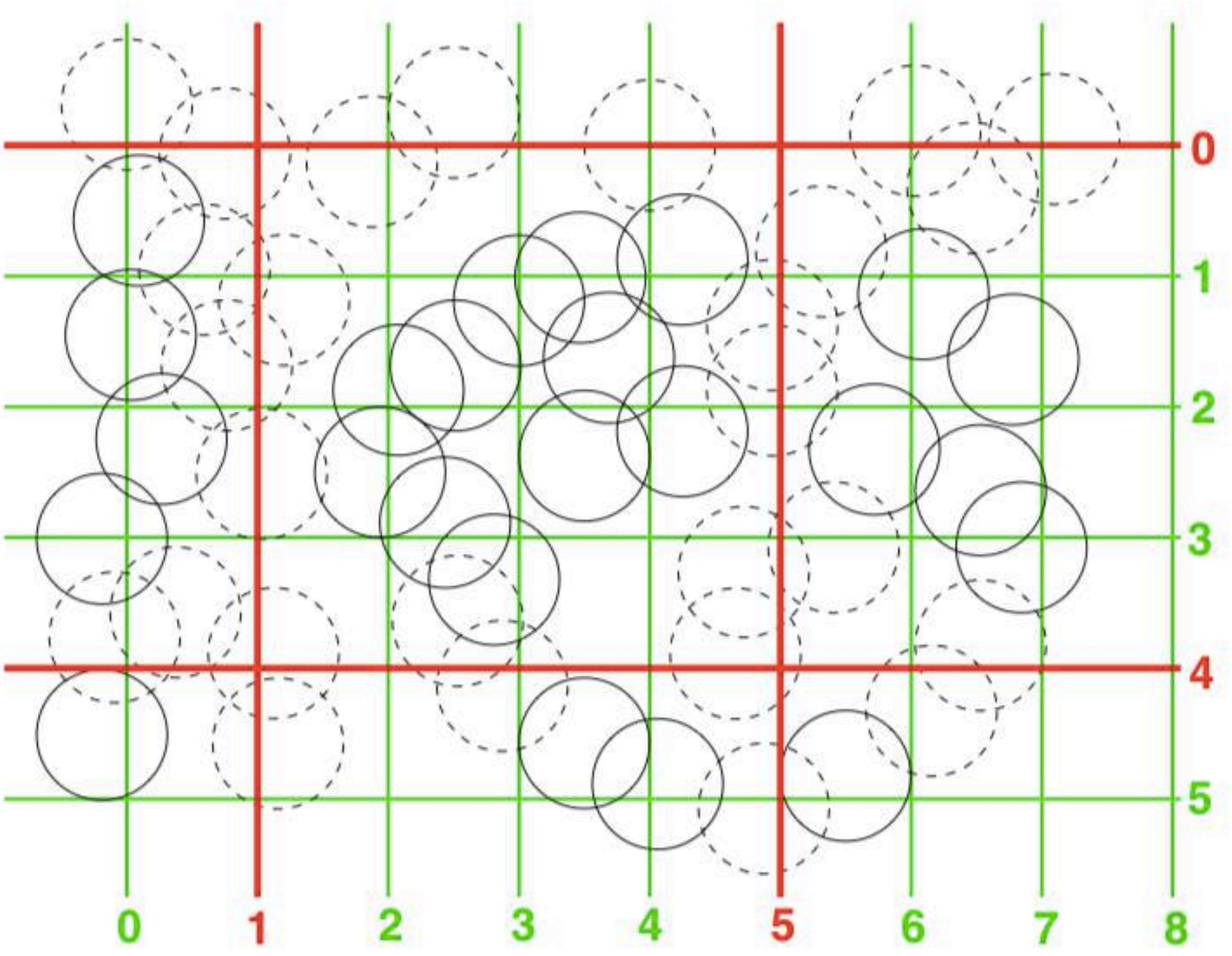


Problème NP-difficile

Exemple :
si tous les disques sont dans un rectangle 10*10
Algorithme :
il suffit de regarder tous les sous-ensembles
d'au plus 100 disques
pour en trouver un de disques disjoints
et de cardinal maximum
Analyse :
OPT < 100

Algorithme

Enlever les disques qui touchent les lignes rouges
Dans chaque carré rouge, regarder toutes les possibilités



Analyse :
Si les lignes rouges sont suffisamment distantes les unes des autres, en moyenne enlever les disques ne changera guère OPT.
Si elles ne sont pas trop distantes, regarder toutes les possibilités dans un carré ne prend pas trop de temps.
Choix :
ni trop ni trop peu

Conclusion

Exemples et algorithmes

Non, ça ne peut pas marcher parce que exemple bla bla bla

Et si on faisait algo bla bla ?

mais non, à cause de tel autre exemple

Mais alors, il suffirait de faire algo bla bla bla pour résoudre ce cas ?

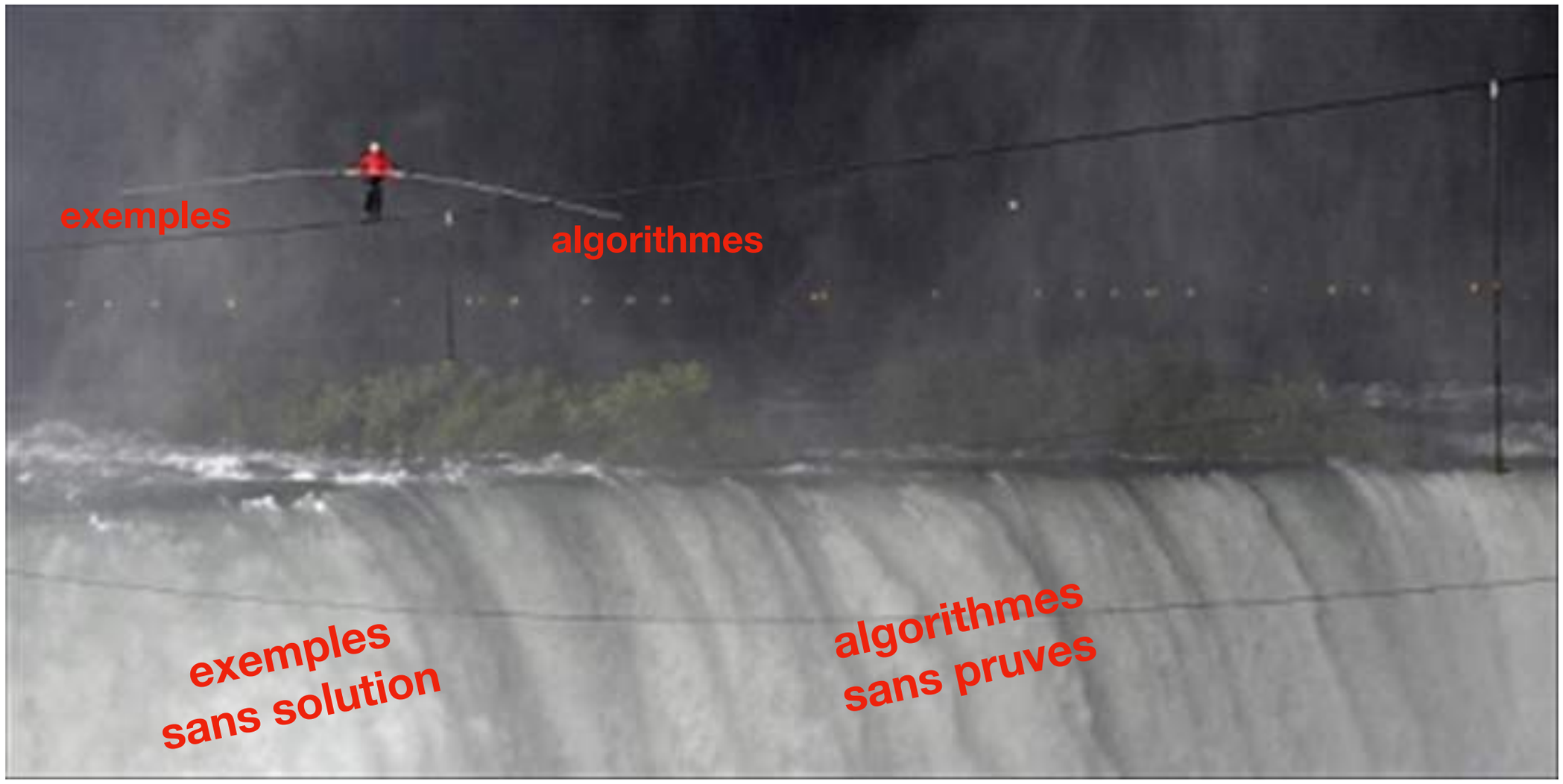
Le joueur colonne essaie de trouver de bons algorithmes

Le joueur ligne essaie de trouver des exemples difficiles

tps, qualité	algo 1	algo 2	algo 3
exemple 1	15s, 90%	1s, 50%	60s, 100%
exemple 2	20s, 85%	1s, 50%	70s, 100%
exemple 3	30s, 85%	10s, 50%	80s, 100%
exemple 4	35s, 80%	20s, 50%	> 99s

Théorie des jeux

Les algorithmes d’approximation sont l’art du compromis


exemples
algorithmes
exemples
sans solution
algorithmes
sans pruves