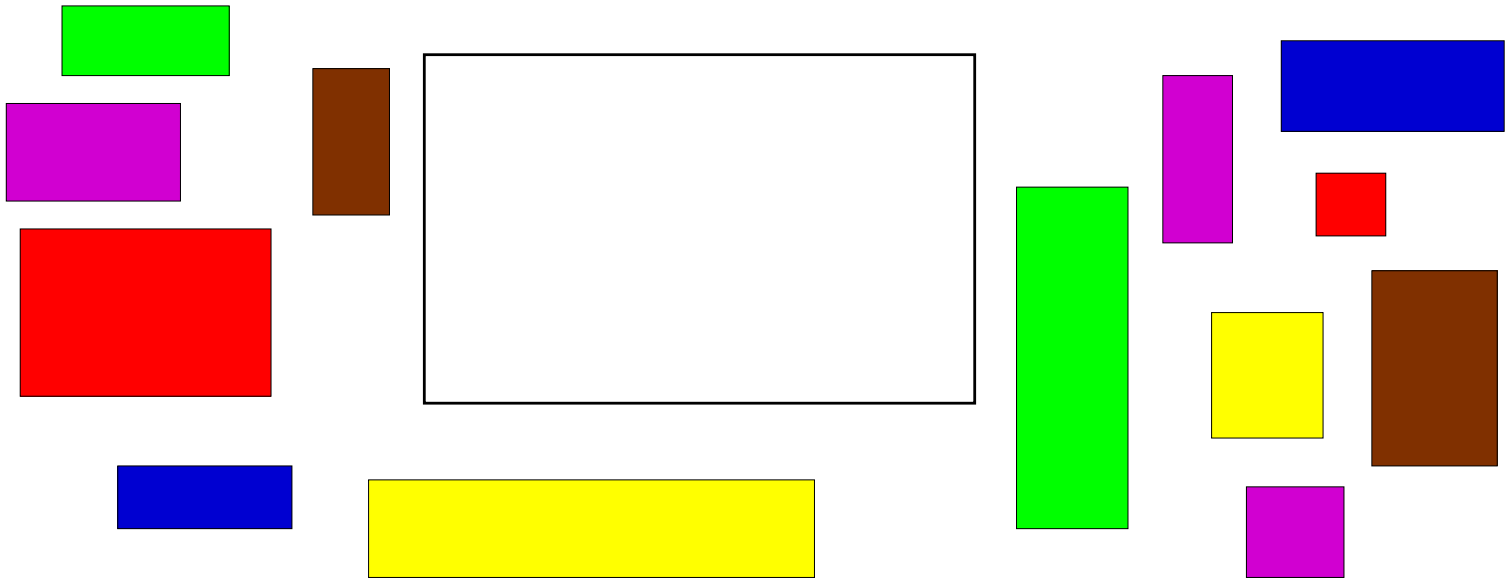
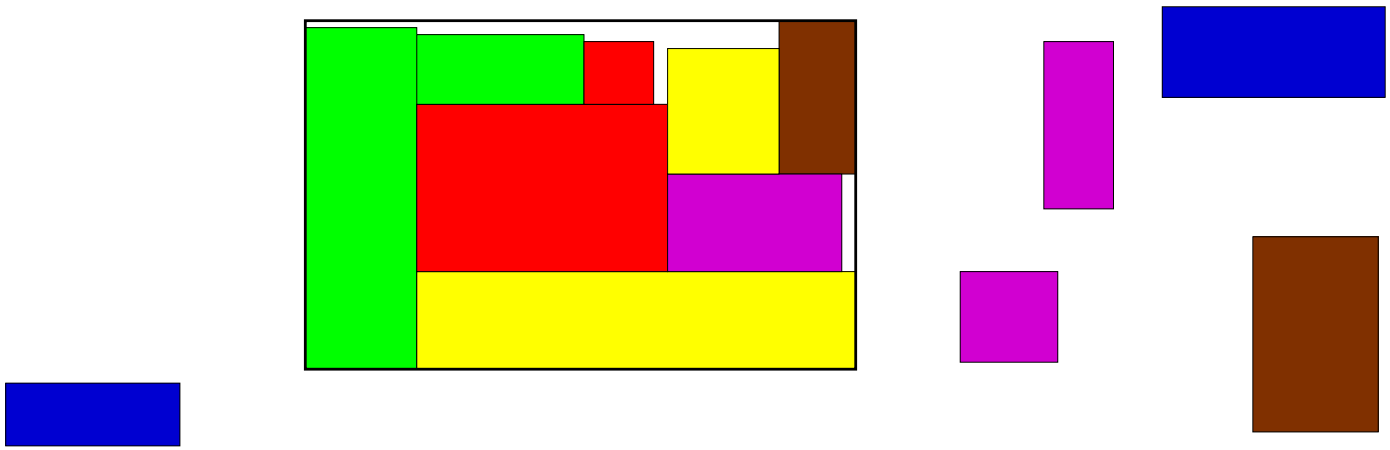
Packing 2D.



Données

- Une boite de dimensions $L \times H$,
- n objets rectangulaires de largeurs $l_1, \ldots, l_n$ et de hauteurs $h_1, \ldots, h_n$.

Packing 2D.

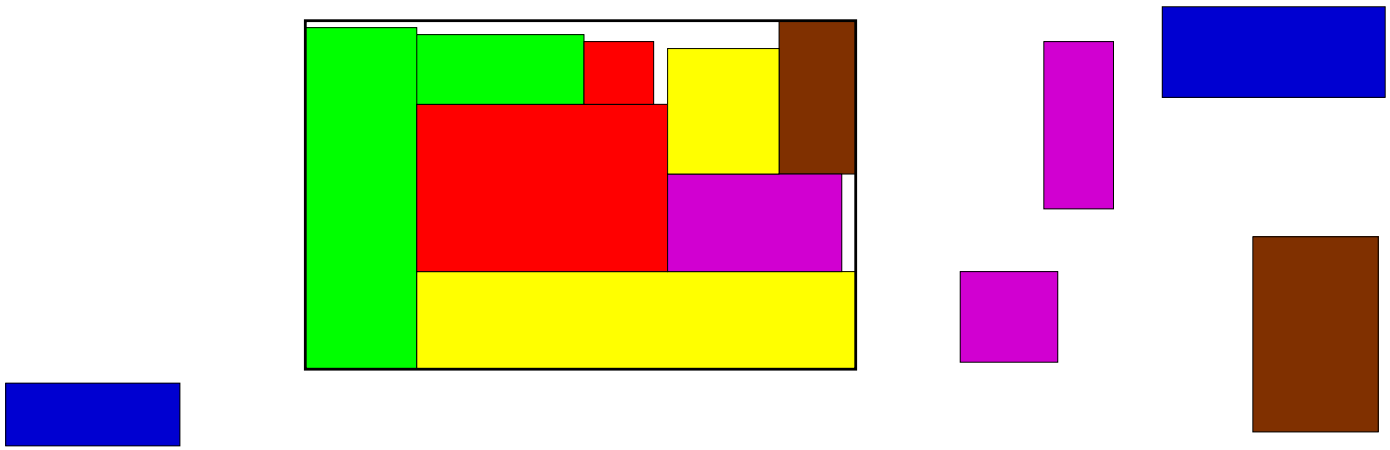


Problème

- Remplir *au mieux* la boite avec des objets.
- **Contrainte** : pas de chevauchement entre objets dans la boite.

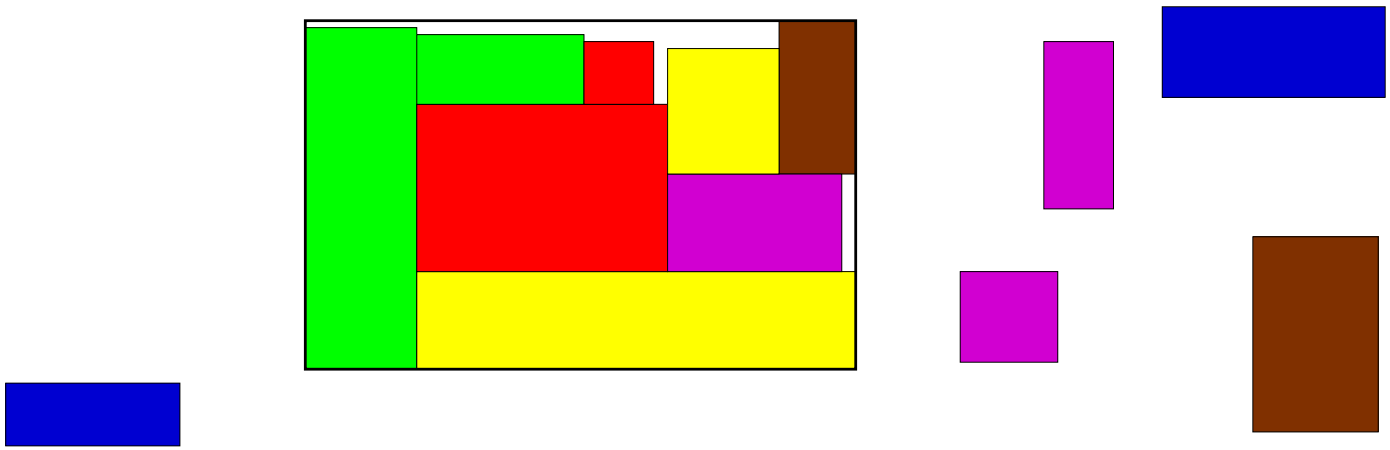
Ce problème s'apparente au sac à dos à deux dimensions.

Packing 2D.



Placement orthogonal : les bords des objets sont parallèles aux bords de la boite (rotation de 90 degrés possible).

Packing 2D.

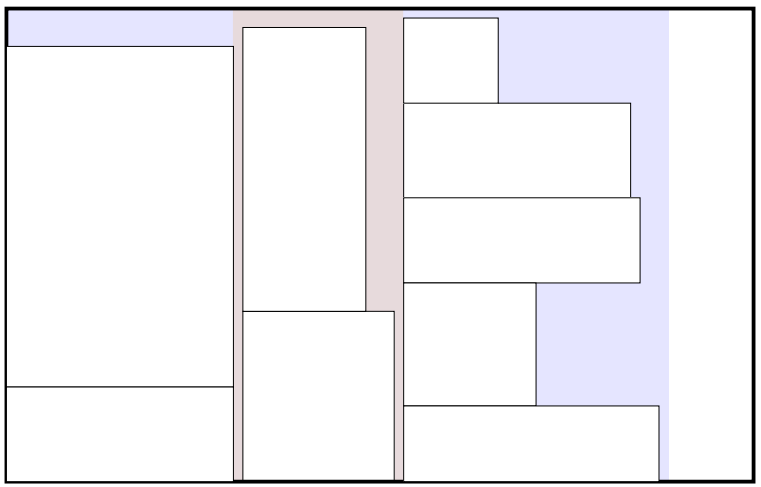


Placement optimal : maximise la surface occupée.

Les méthodes de recherche d'un placement optimal ont des coûts *exponentiels*.

Packing 2D : méthode incomplète.

Remplissage par bandes (strips).

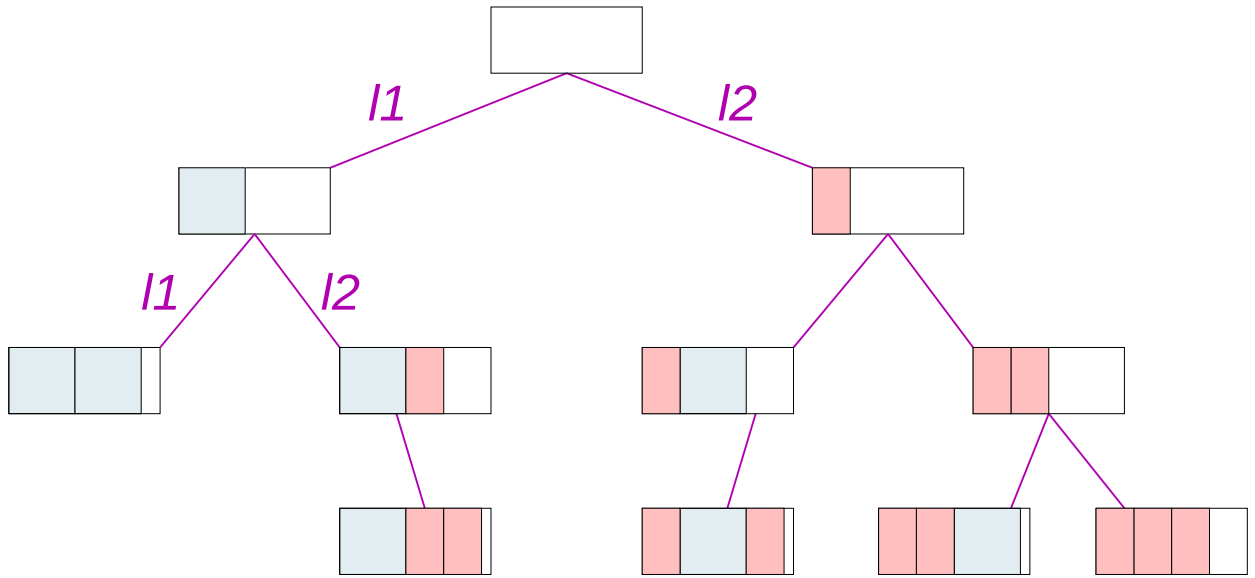


Pas de garantie de l’optimalité.

Pisinger, D. : Heuristics for the container loading problem. Eur. J. Oper. Res. 141(2), 382–392 (2002).

Recherche d'un remplissage.

Recherche énumérative sur la largeur des bandes.



L'algorithme explore deux largeurs pour chaque bande ($k = 2$).

(pour simplifier on a considéré ici uniquement les largeurs $l1$ et $l2$)

Approche ascendante.

Function `remplir_boite`(l, O, h, k)

Data: l : longueur , O : ensemble d'objets, h : hauteur de la boite, k : largeur de l'exploration

begin

$B := \emptyset$;

$C :=$ selectionner_k_largeurs_valides (l, O, k);

while $C \neq \emptyset$ **do**

extraire e de C;

$S :=$ remplir_bande (e, O, h);

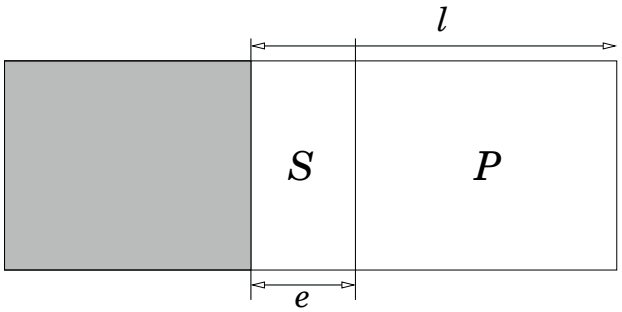
$P :=$ remplir_boite ($l - e, O \backslash$objets(S), h, k);

if $S \cup P$ *est meilleur que* B **then**

$B := S \cup P$;

return B;

end



Approche ascendante : la solution est construite en remontant.

La fonction renvoie le meilleur remplissage de l'espace $l \times h$.

Approche descendante.

Function `remplir_boite`(l, O, h, k, P)

Data: l : longueur , O : ensemble d'objets, h : hauteur de la boite, k : largeur de l'exploration,
P : remplissage courant

begin

if P est meilleur que $BestP$ **then**
$BestP := P$;

$C :=$ selectionner_k_largeurs_valides (l, O, k);

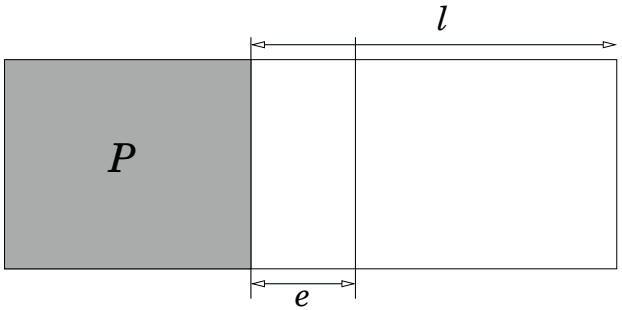
while $C \neq \emptyset$ **do**

extraire e de C;

$S :=$ remplir_bande (e, O, h);

remplir_boite ($l - e$, $O \backslash$objets(S), h, k, $P \cup S$);

end



Le remplissage est construit en descendant.

On mémorise dans $BestP$ le meilleur remplissage rencontré pendant l'exploration.

Note : $BestP$ peut être une variable globale ou un paramètre/résultat.

Remplissage d'une bande : approche glouton.

Remplissage d'une bande de largeur fixée e :

- initialiser la bande à vide (hauteur nulle),
- parcourir les objets libres et les ajouter à la bande chaque fois que c'est possible (i.e. compte tenu de la largeur de la bande et de la hauteur restante ; envisager les deux orientations de l'objet).

Le remplissage de la bande n'est pas toujours optimal.

Recherche d'un remplissage : améliorations.

[1**] remplir les bandes de façon optimale avec un sac à dos,

[2*] choix pertinent des largeurs de bandes (les dimensions qui apparaissent souvent par exemple),

[3*] propriété de coupure : le remplissage partiel ne peut pas produire un meilleur remplissage,

[4***] construction des bandes dans les deux sens (verticales/horizontales),

[5***] mémorisation des solutions partielles dans une table de hashage,

[6***] imbrication des bandes les unes dans les autres,

[7***] remplissage en découpant en sous-rectangles le conteneur.

les $*$ indiquent la difficulté

les parties grisées sont facultatives

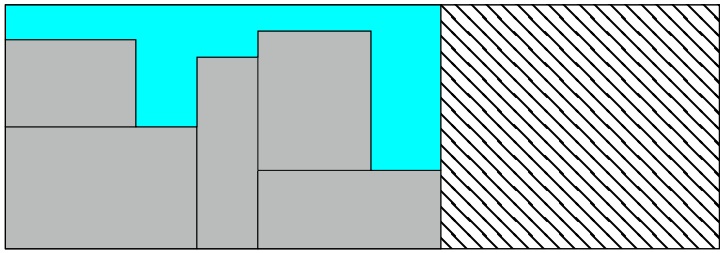
Amélioration : remplissage des bandes.

Remplissage d'une bande de largeur fixée : sac à dos.

- sélectionner les *objets candidats* et leurs *orientations*
- calculer un remplissage optimal consiste à résoudre un problème de *sac à dos* :
  - capacité : hauteur de la boite,
  - poids des objets : hauteurs des objets,
  - gains associés aux objets : surfaces des objets.

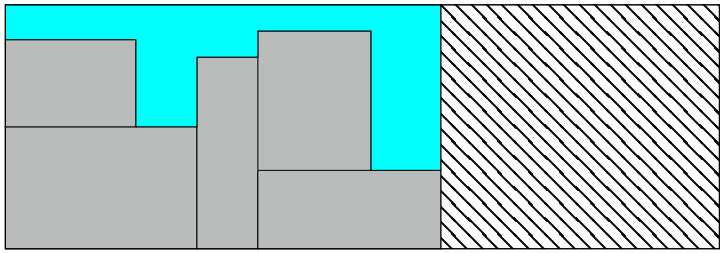
Amélioration : propriété de coupure.

- évaluer l’espace perdu dans le remplissage courant

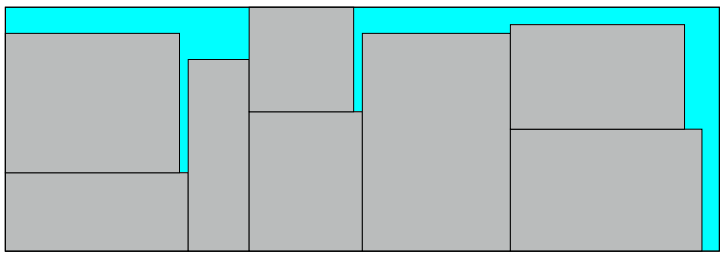


Amélioration : propriété de coupure.

- évaluer l'espace perdu dans le remplissage courant



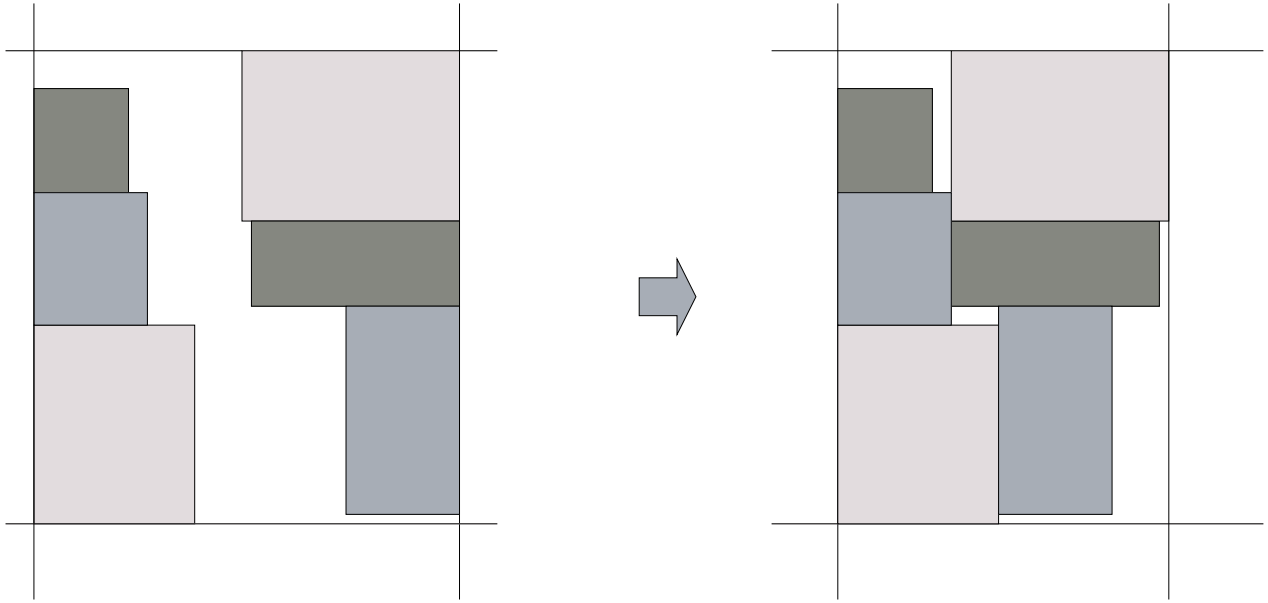
- comparer avec l'espace perdu dans le meilleur remplissage



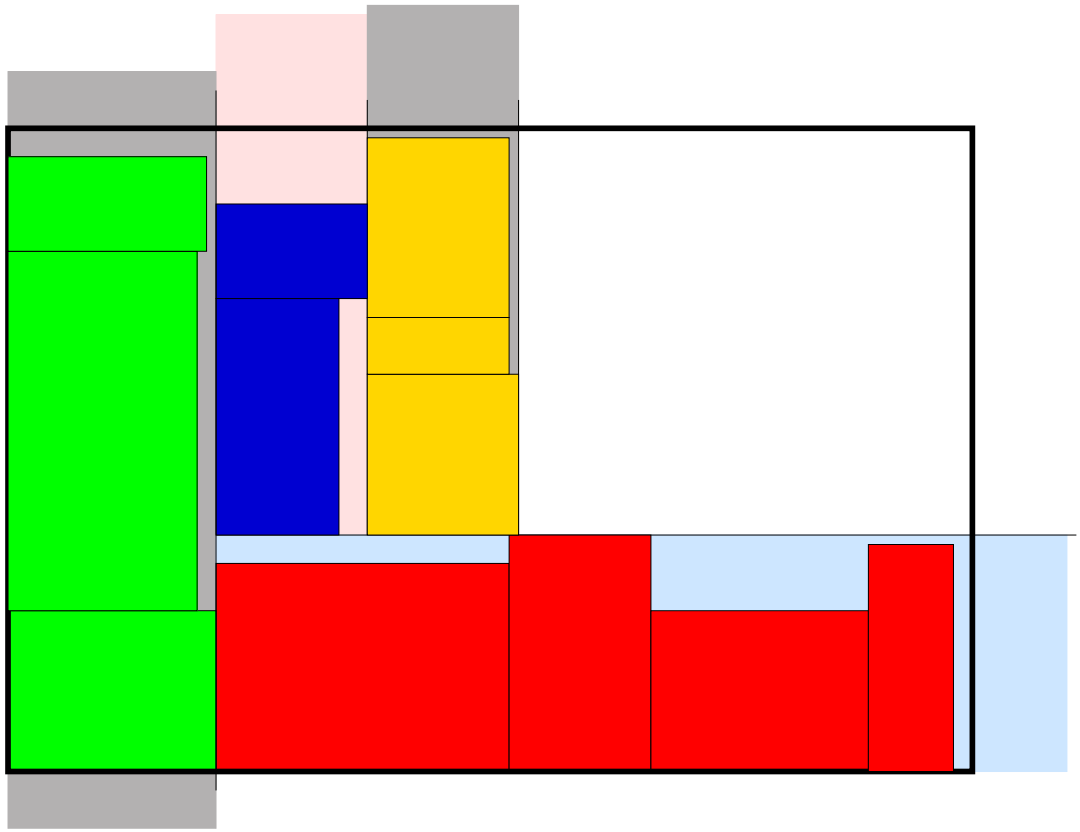
- si l'espace perdu est trop important on ne pourra jamais obtenir une solution meilleure

Amélioration : imbrication a postériori.

Imbriquer les bandes deux par deux.

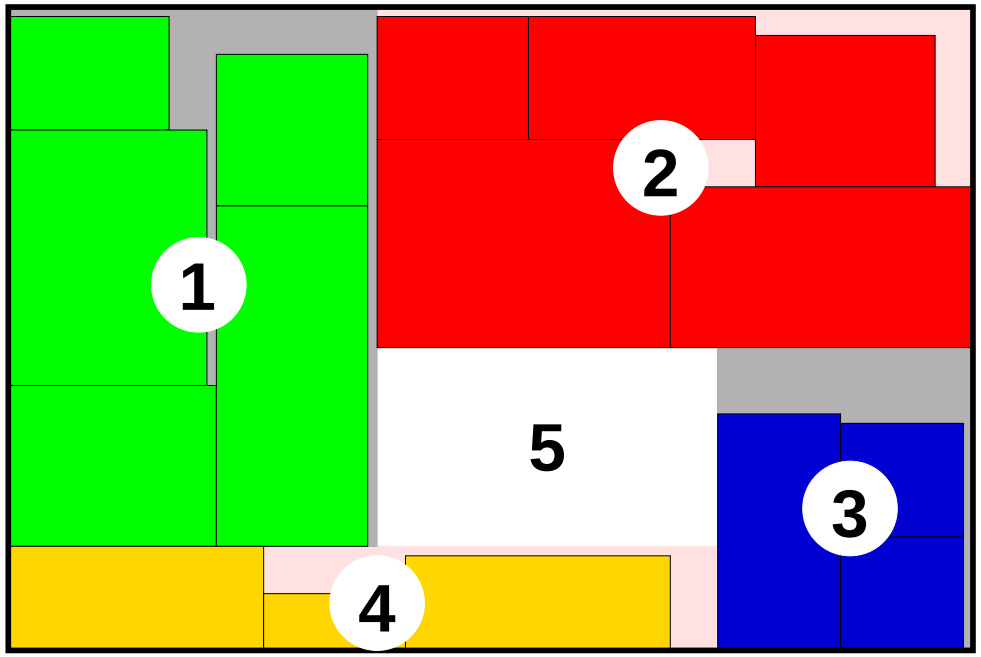


Amélioration : bandes verticales et horizontales.



Amélioration : découpage du conteneur.

Découpage de l’espace en bandes ou en boites.



Projet : première partie.

- générateur aléatoire de problèmes,
- calcul d'une solution
  - structures de données,
  - lecture et allocations,
  - calcul d'un remplissage (approche glouton pour les bandes),

A rendre la semaine du 22 février 2016

Projet : deuxième partie.

- calcul d'une solution
  - sac à dos pour le remplissage des bandes,
  - reconstruction et sauvegarde des solutions,
- visualisation des solutions avec EZ-Draw :
  `http://pageperso.lif.univ-mrs.fr/~edouard.thiel/ez-draw/index.html`
- document de présentation.

A rendre avant le 1er avril 2016

Annexe : syntaxe des problèmes.

```
<largeur boite> <hauteur boite>

<nombre d'objets n>
<nom1> <l1> <h1>
   .
   .
   .
<nom1> <ln> <hn>
```

Noms : chaînes de caractères (permettent de donner un nom, une couleur, un type,...)

Annexe : syntaxe des solutions.

```
<largeur conteneur> <hauteur conteneur>

<nombre d'objets n>
<nom1> <l1> <h1> <x1> <y1>
   .
   .
   .
<nomn> <ln> <hn> <xn> <yn>
```

Pour chaque objet : sa largeur, sa hauteur, sa position sur l'axe des x et sa position sur l'axe des y

Annexe : organisation des programmes.

- modules (lecture.c,...),
- fichiers d'entête (lecture.h)
  - macros,
  - définitions de types,
  - déclarations de variables **extern**,
  - prototypes.
- méthodes courtes, claires et lisibles.

Annexe : évaluation.

- Document (6 points)
- Projet (10 points)
- Validation/expérimentation (4 points)