Packing 2D.

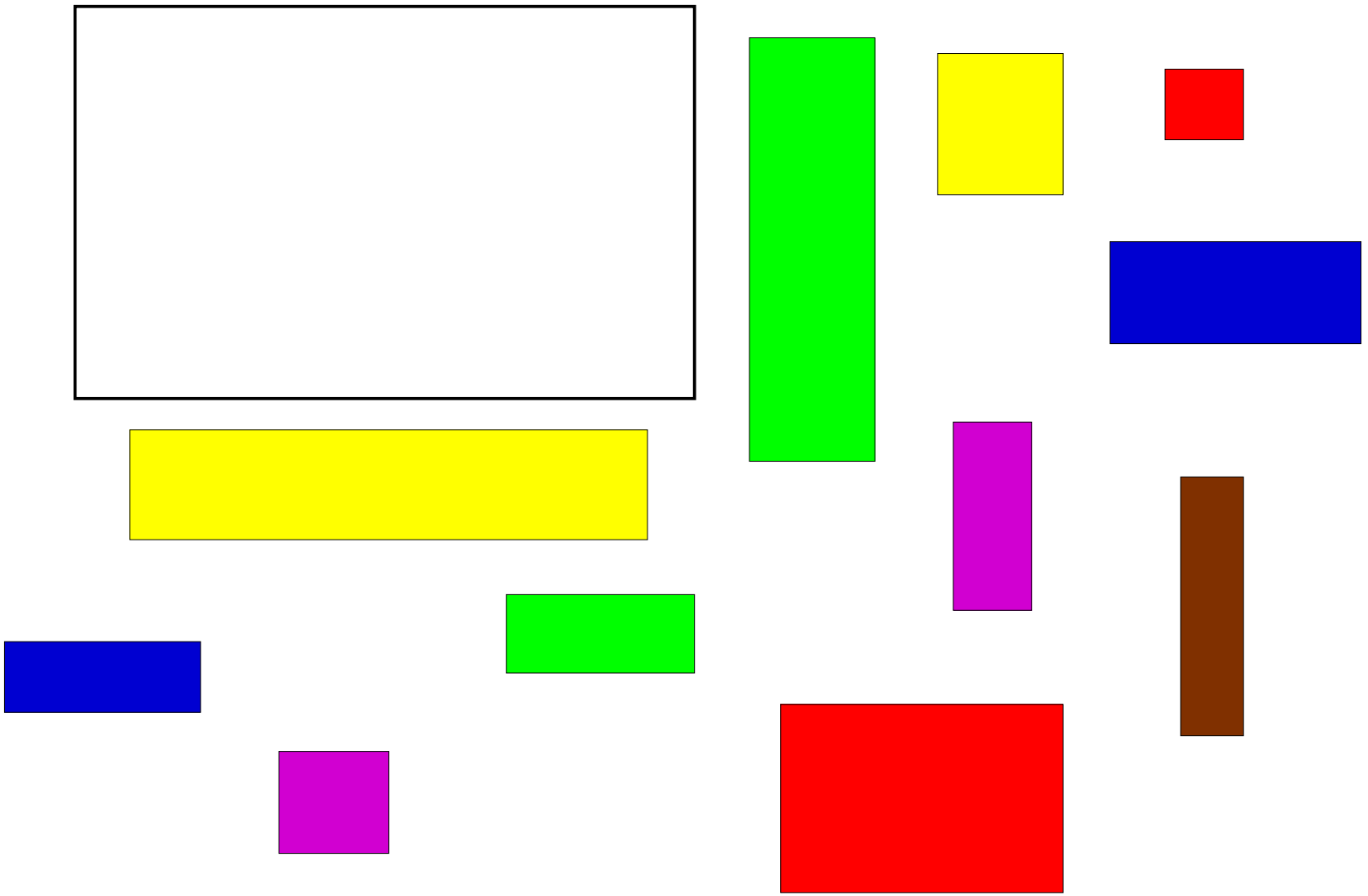
Données.

- Un conteneur C de dimension $L \times H$,
- des objets rectangulaires $x_1, \ldots, x_n$ de dimensions $\langle l_1, h_1 \rangle, \ldots, \langle l_n, h_n \rangle$.

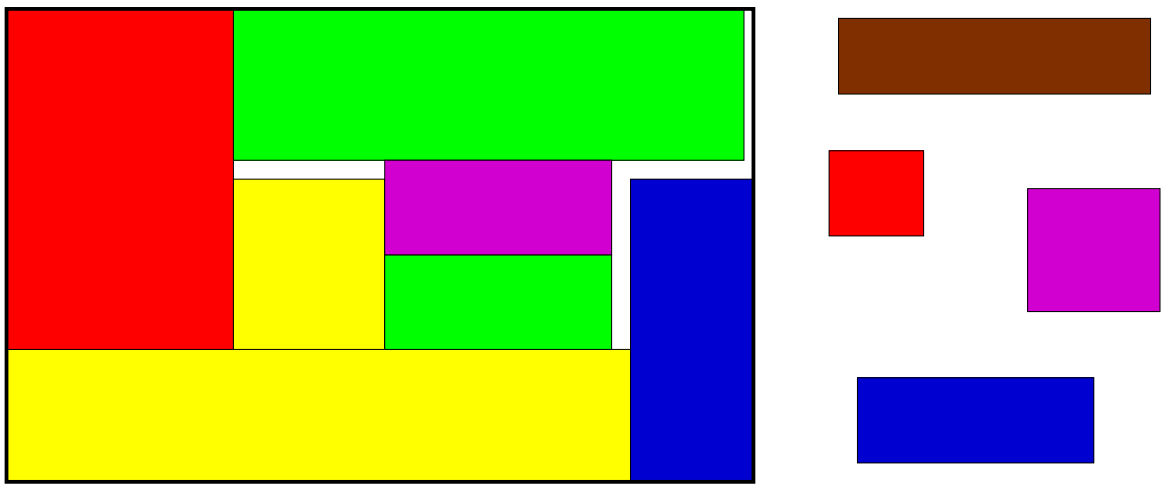
Placement orthogonal : les objects sont dans le conteneur avec les bords parallèles aux bords du conteneur.

Problème. Remplir *au mieux* le conteneur avec les objets.

Packing 2D : exemple.



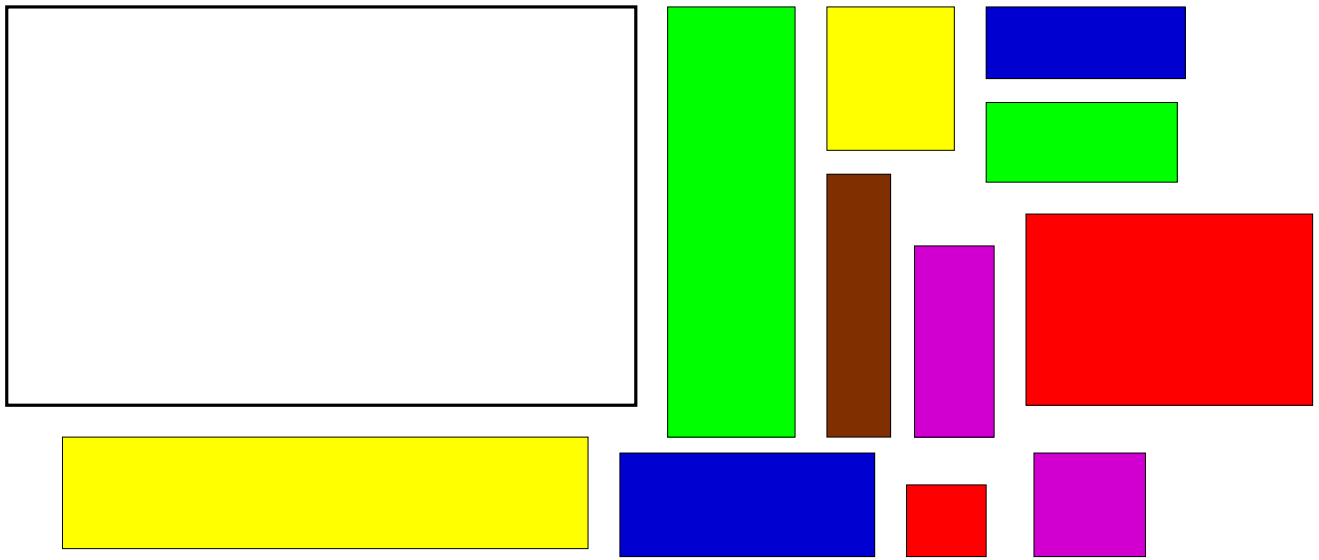
Packing 2D : exemple.



Un *placement optimal* (i.e. qui maximise la surface occupée).

Les méthodes de recherche d'une solution optimale
ont des coûts *exponentiels*.

Packing 2D : recherche solution optimale.

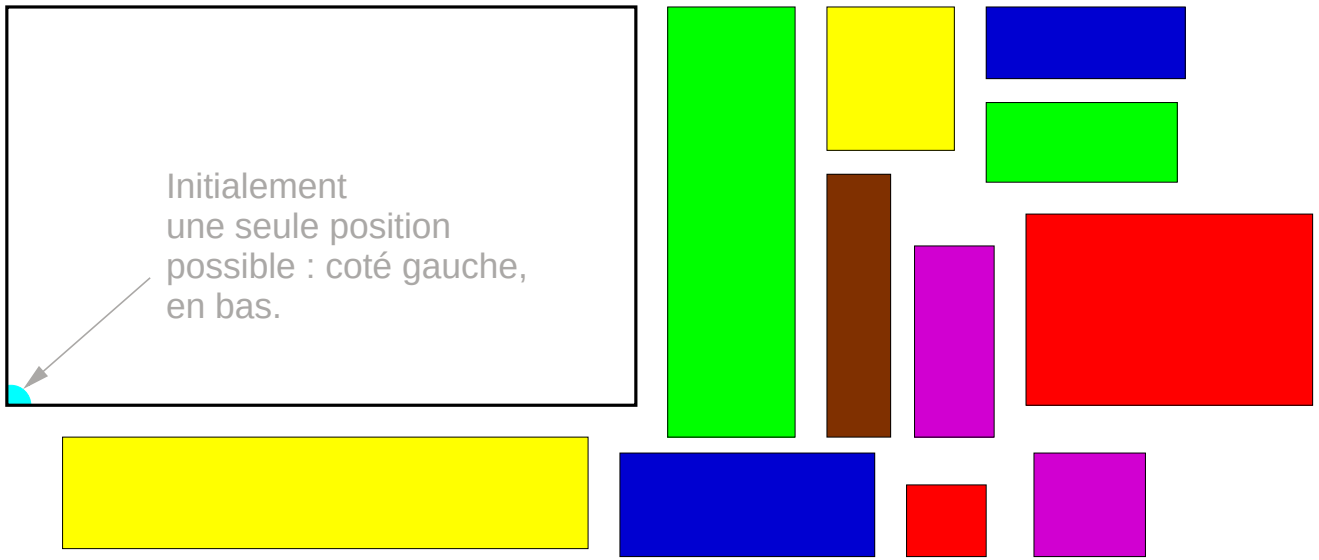


Placement des objets

- le plus à gauche possible
- le plus bas possible

Chaque objet est en appui sur la gauche du conteneur ou la face droite d'un autre objet.
Chaque objet est en appui sur le bas du conteneur ou la face supérieure d'un autre objet.
Si il existe une solution au problème alors il en existe une de ce type.

Packing 2D : recherche solution optimale.

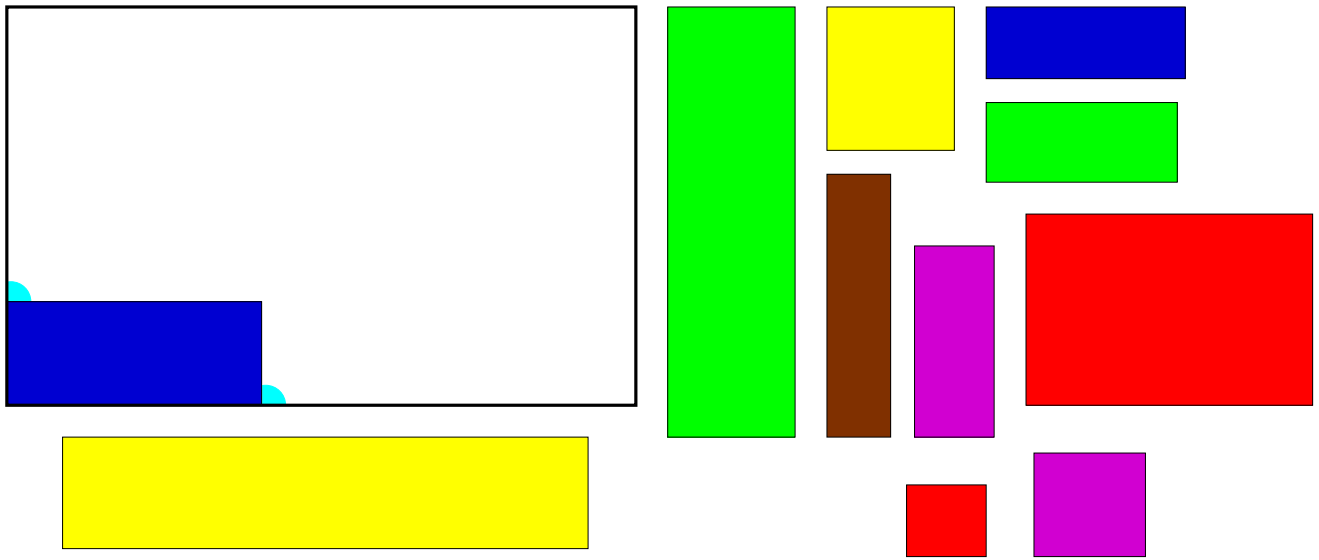


Recherche : explorer toutes les possibilités

- toutes les positions
- tous les objets

niveau 1 : une position, n objets $\Rightarrow 1 \times n$ choix

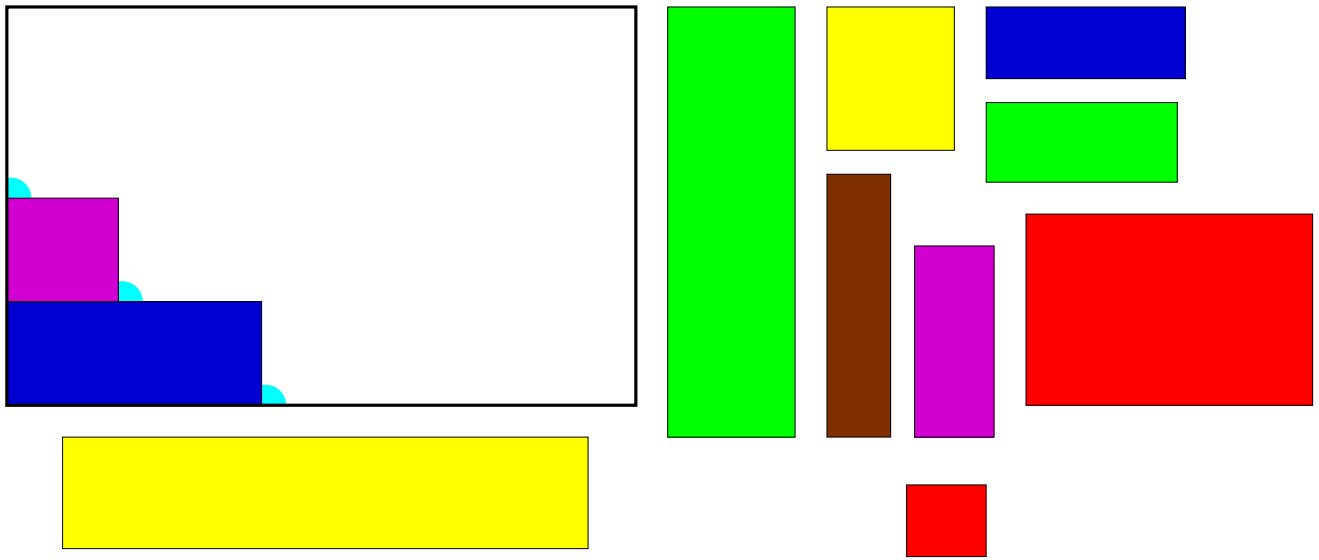
Packing 2D : recherche solution optimale.



Ajout d'un objet $\longrightarrow$ création d'une nouvelle position

niveau 2 : 2 positions, $n-1$ objets $\Rightarrow 2 \times (n-1)$ choix

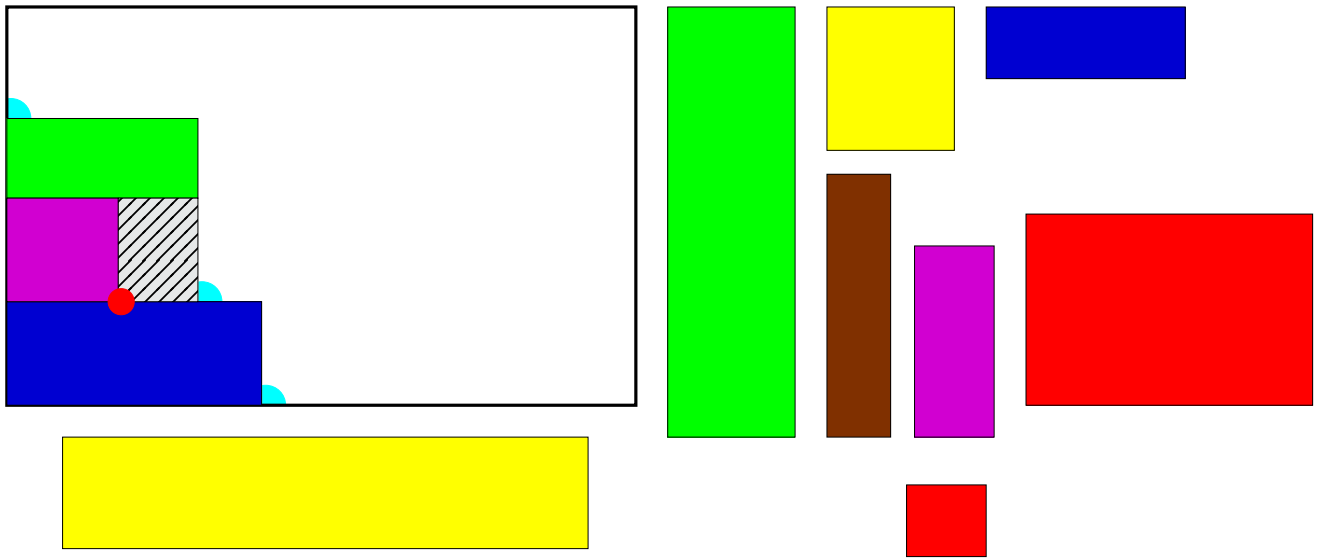
Packing 2D : recherche solution optimale.



niveau 3 : 3 positions, $n-2$ objets $\Rightarrow 3 \times (n-2)$ choix

Pourquoi envisager toutes les positions pour tous les objets ?

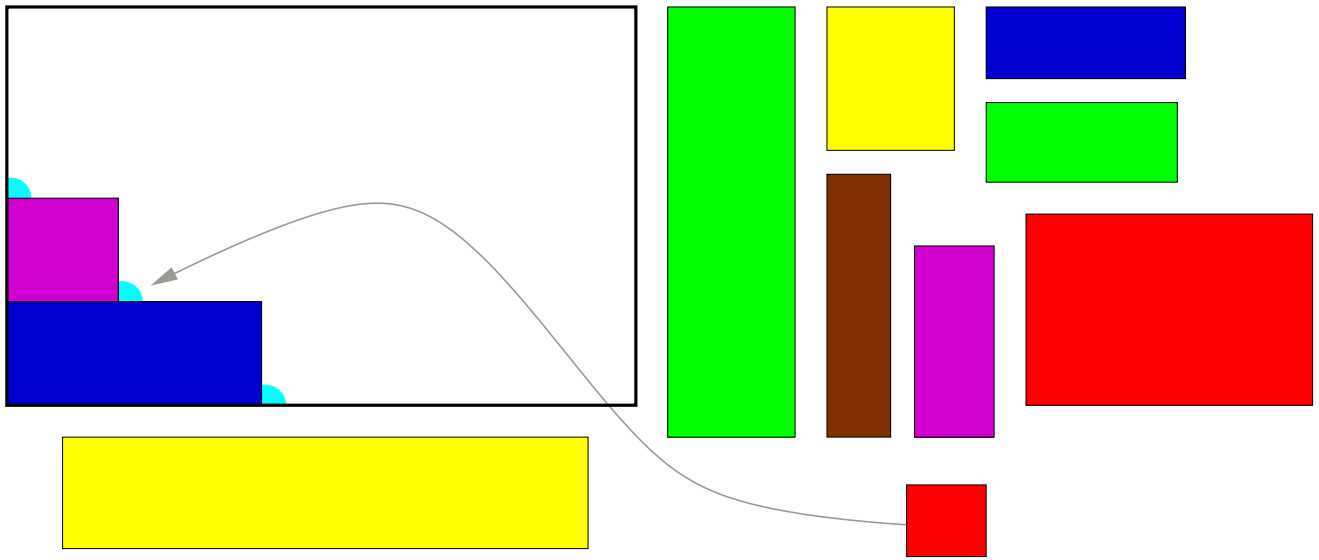
Packing 2D : recherche solution optimale.



Positions possibles : angles entrants de l'enveloppe

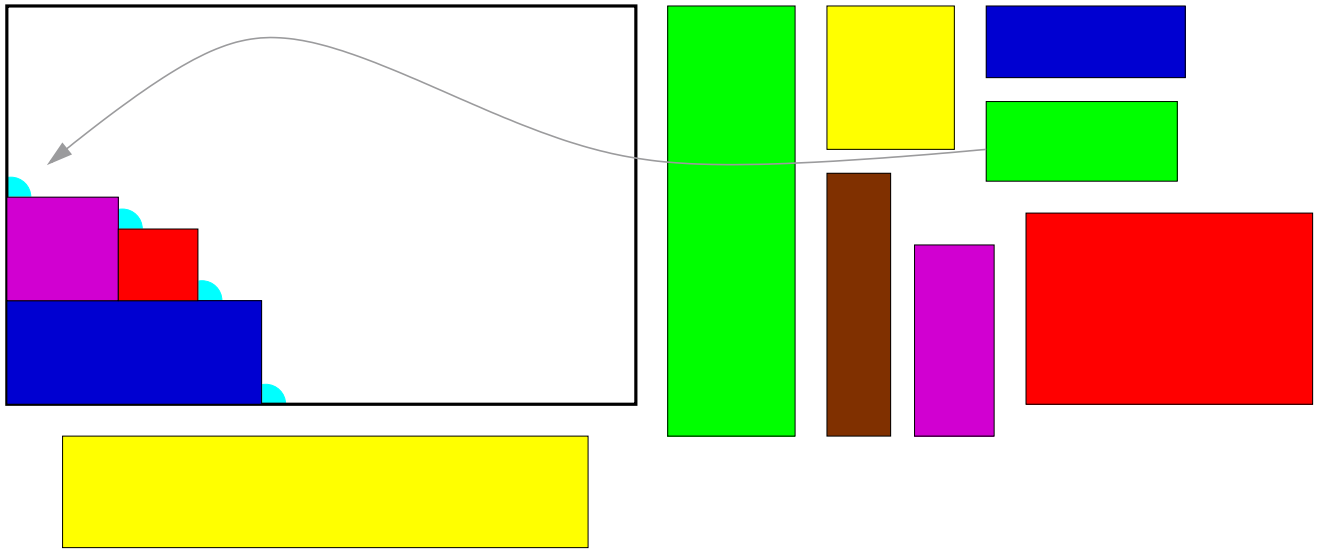
Pour placer un objet à la position du point rouge il fallait le faire avant

Packing 2D : recherche solution optimale.



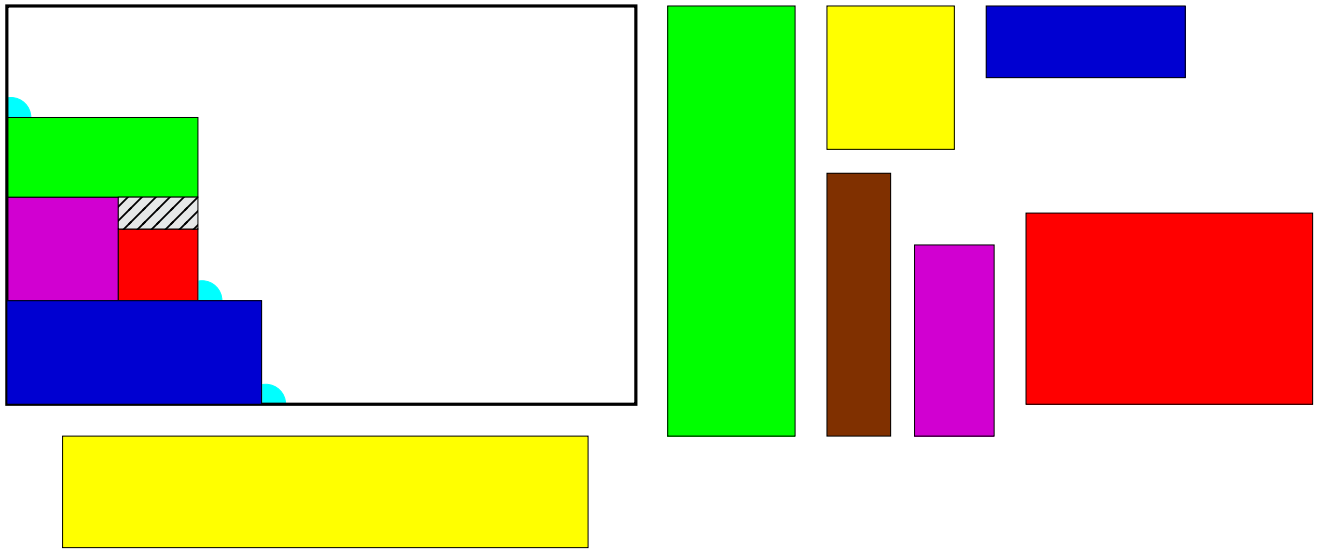
Placement de l’objet rouge

Packing 2D : recherche solution optimale.



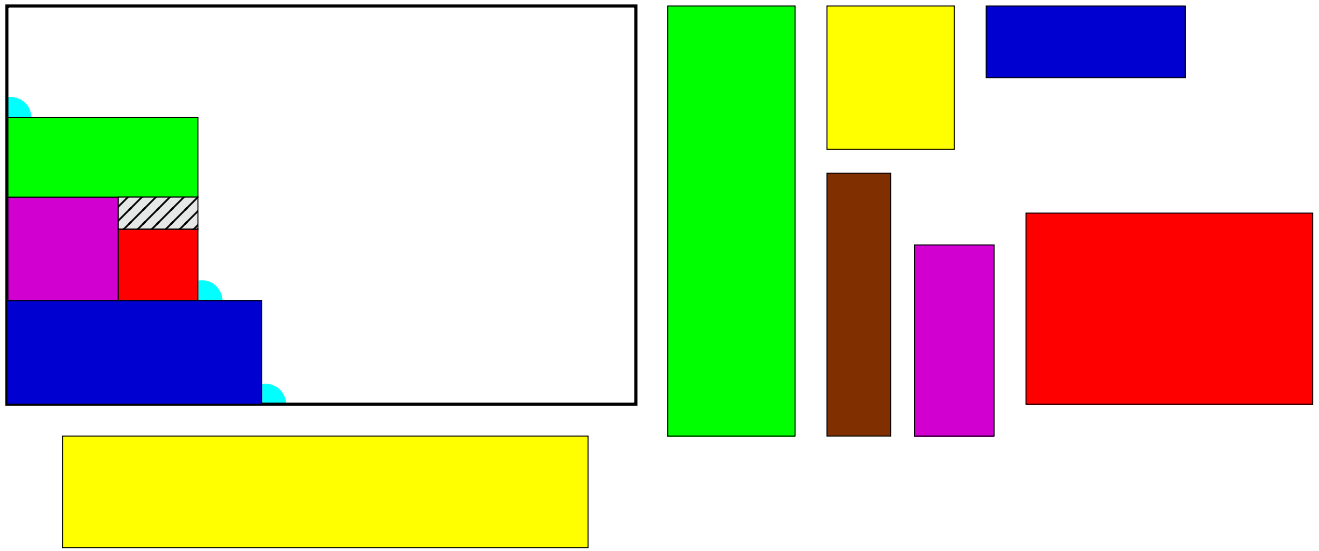
Placement de l’objet vert

Packing 2D : recherche solution optimale.



Placement du $i$ème objet : $i \times (n - i + 1)$ choix possibles

Packing 2D : recherche solution optimale.

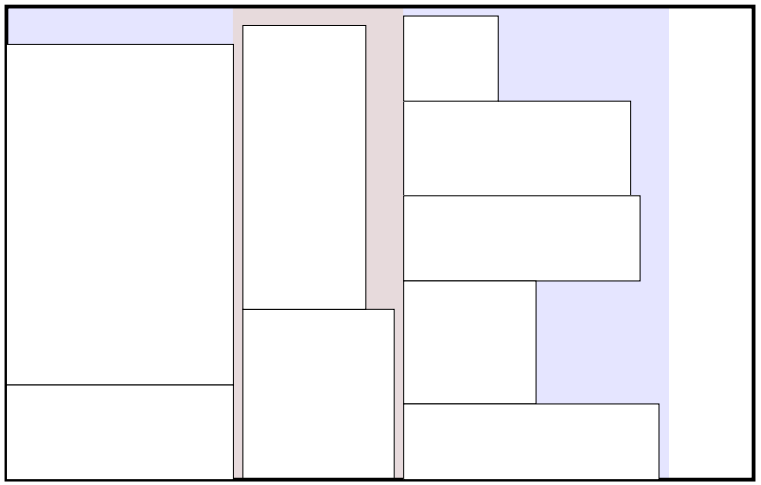


Au final on a donc un coût

$$C(n) \leq \prod_{i=1}^{n}(i \times (n-i+1)) = n! \times n! \backsimeq O(n^{2n})$$

Packing 2D : méthode incomplète.

Remplissage par bandes (strips).

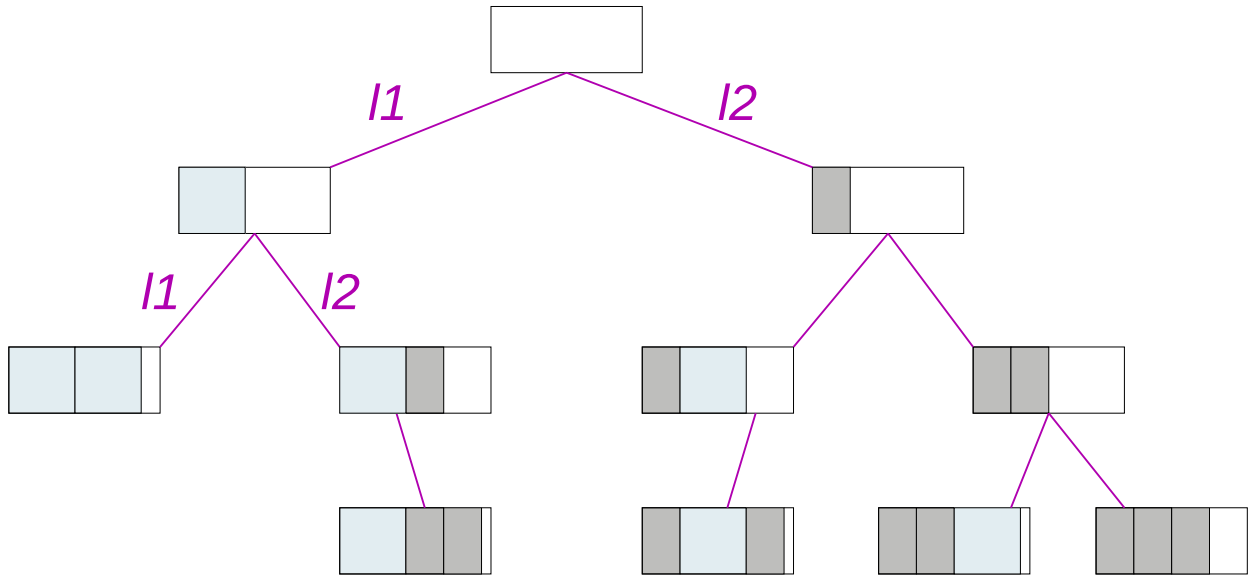


Ne garantit pas l’optimalité des solutions.

Recherche d'un remplissage.

Recherche énumérative sur k largeurs de bandes.

(on considère des largeurs correspondant aux dimensions des objets)



Exemple avec $k = 2$ et deux dimensions différentes en tout et pour tout.

Approche ascendante.

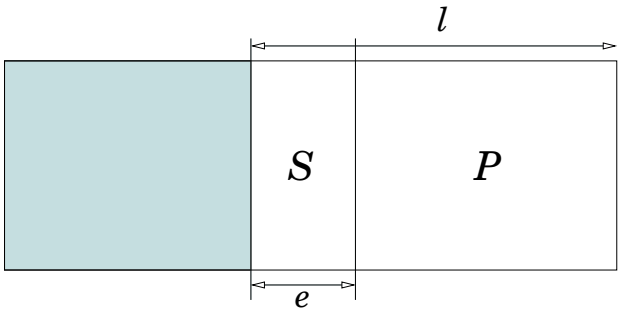
Function `remplir_conteneur`(l, O, h, k)

Data: l : longueur , O : ensemble d'objets, h : hauteur du conteneur, 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_conteneur ($l - e, O \setminus obj(S), h, k$);
  - **if** $S \cup P$ *est meilleur que* B **then**
    - $B := S \cup P$;
- **return** B;

end



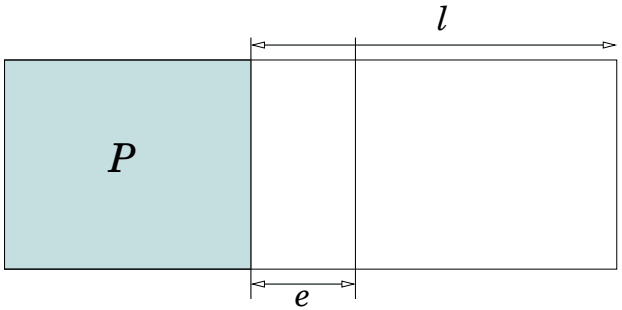
Approche ascendante : la solution est construite en remontant.

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

Approche descendante.

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

Data: l : longueur , O : ensemble d'objets, h : hauteur du conteneur, 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_conteneur ($l - e, O \setminus obj(S), h, k, P \cup S$);

end

Le remplissage du conteneur 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.

Recherche d’un remplissage.

Remplissage d’une bande de largeur fixée : approche glouton.

- 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. qu’on ne dépasse pas la hauteur du conteneur).

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

Recherche d'un remplissage : améliorations.

[1] remplir les bandes de façon optimale : sac à dos $(**)$,

[2] favoriser des dimensions qui apparaissent le plus souvent $(*)$,

[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] mémoriser les solutions partielles dans une table de hashage $(***)$,

[6] imbriquer les bandes les unes dans les autres au fur et à mesure $(***)$.

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

les $*$ indiquent la difficulté

les parties grisées sont facultatives

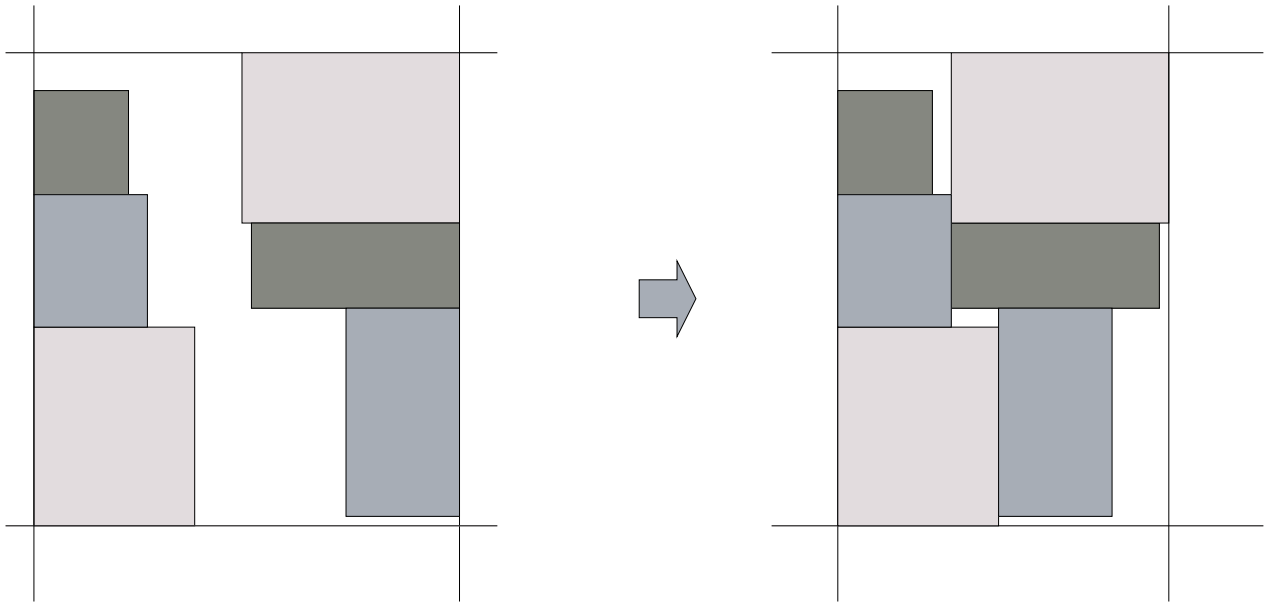
Amélioration : remplissage des bandes.

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

- sélectionner les *objets candidats* et leurs *orientations*
- calculer un remplissage optimal en utilisant un *sac à dos* :
  - capacité : hauteur du conteneur
  - poids : hauteurs des objets
  - gains : surfaces des objets

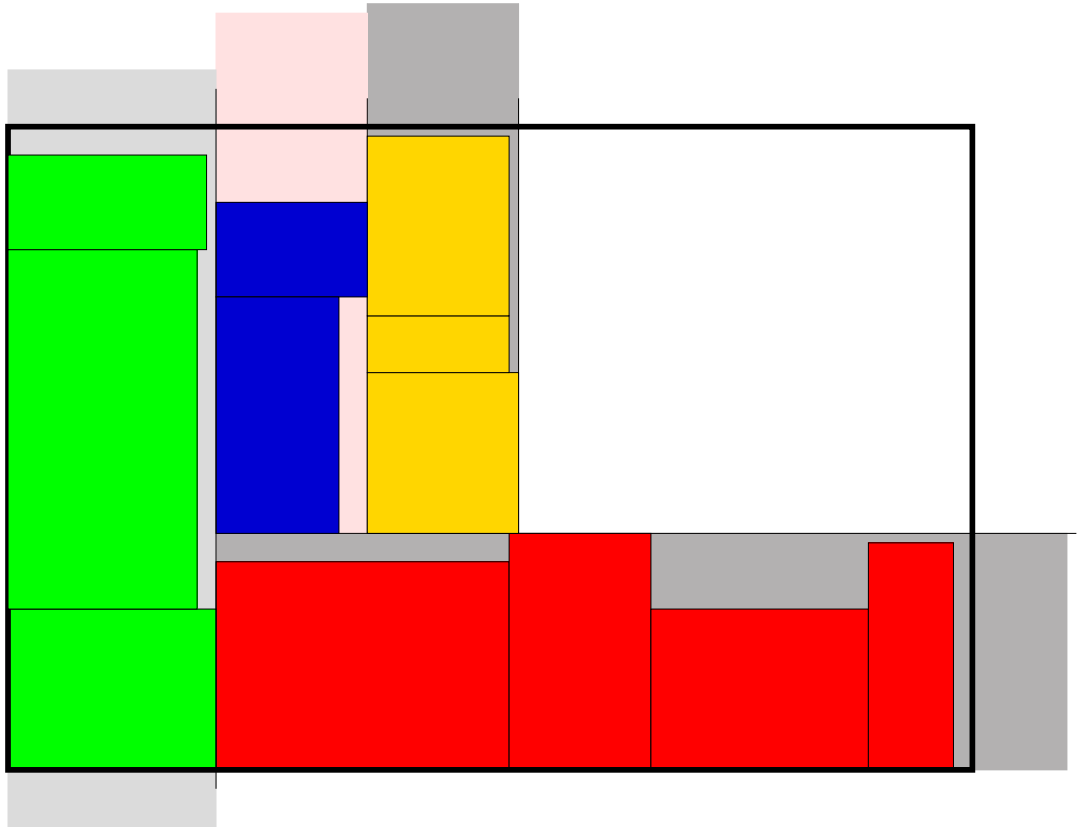
Amélioration : imbrication a postériori.

Imbriquer les bandes deux par deux.



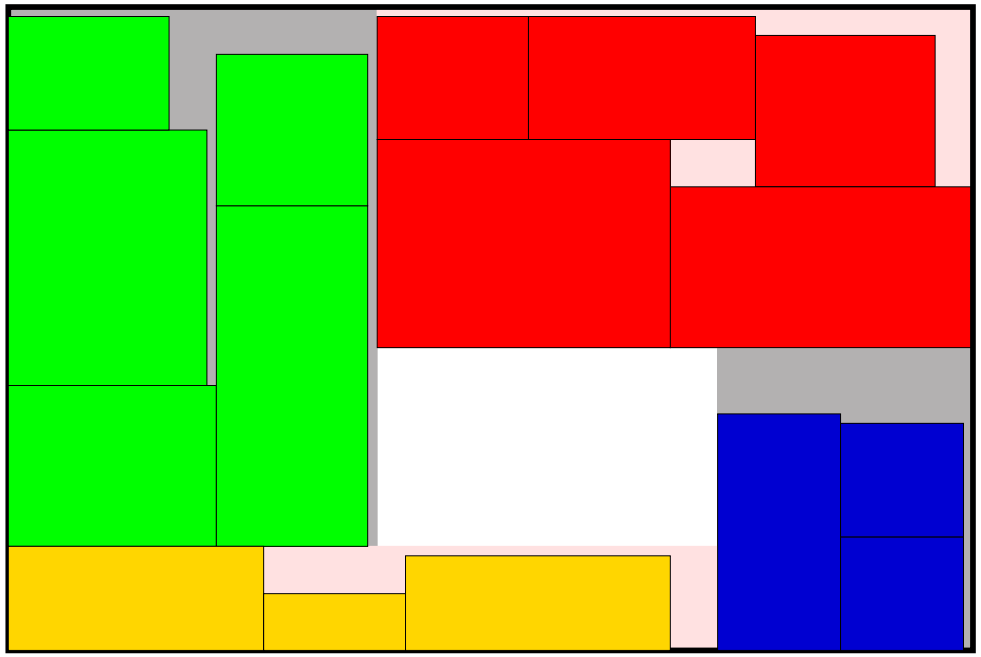
Amélioration : découpage en bandes.

Bandes horizontales ou verticales.



Amélioration : découpage du conteneur.

Découpage en bandes et en rectangles.



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 10 février 2014

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 28 mars 2014

Annexe : syntaxe des problèmes.

```
<largeur conteneur> <hauteur conteneur>

<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, son abcisse et son ordonnée

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.