Résolution de niveaux du Sokoban

Nom, Prénom

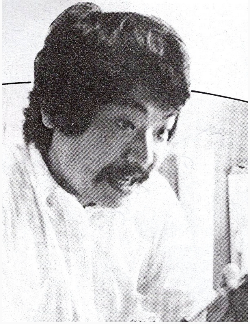
25 mai 2023

Numéro de candidat

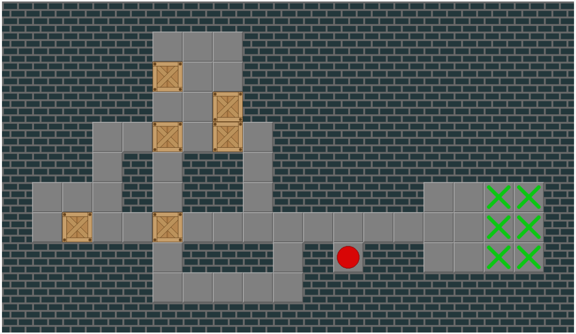
Plan

Le jeu du Sokoban

Le jeu du Sokoban



Hiroyuki Imabayashi

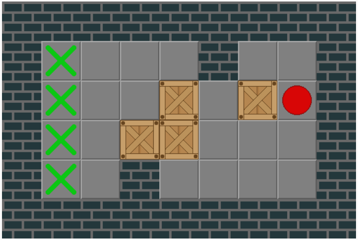


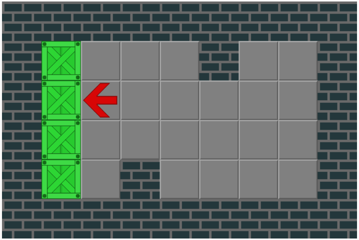
XSokoban

Problème **PSPACE-complet**

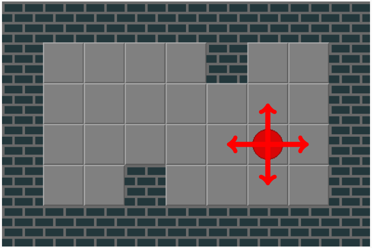
But du jeu

Déplacements



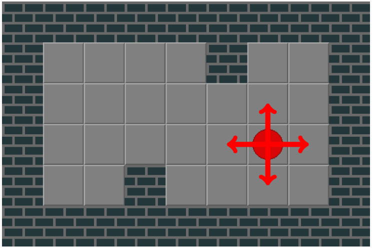


Règles

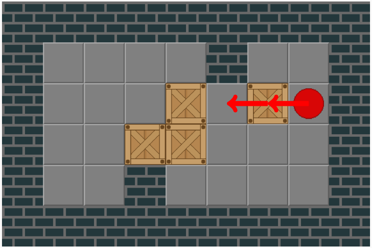


Déplacements autorisés

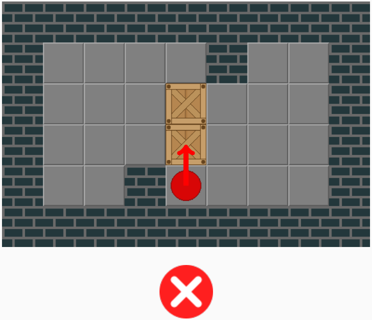
Règles

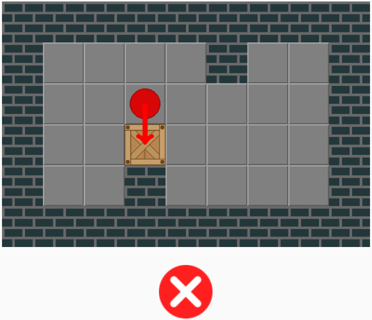


Déplacements autorisés

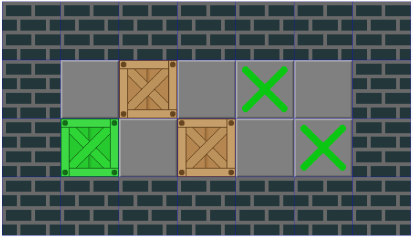


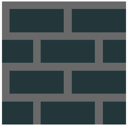
Règles



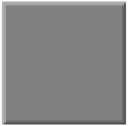


Tuiles

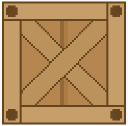




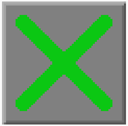
Mur



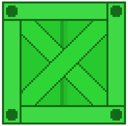
Sol



Caisse



Cible



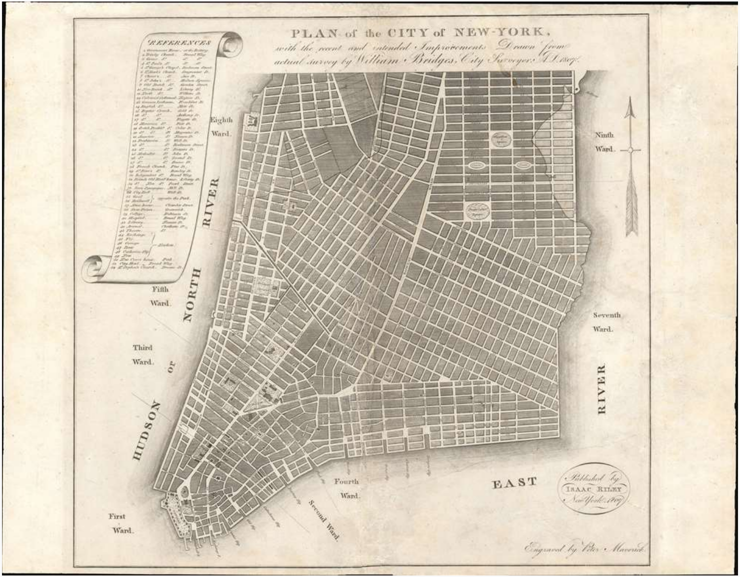
Caisse sur une cible

Quelles stratégies adopter pour trouver une solution le plus rapidement possible à un niveau de Sokoban ?

```
Welcome to sokoshell - Version 1.0-SNAPSHOT
Type 'help' to show help. More help for a command with 'help command'
sokoshell>
```

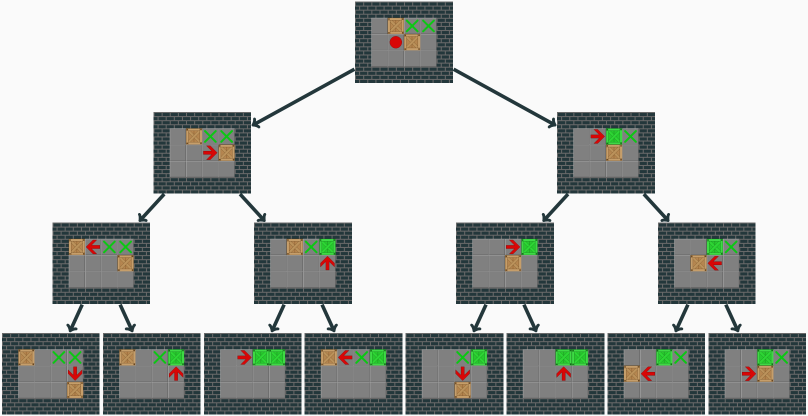
Lien avec le thème de l'année

Lien avec le thème de l'année

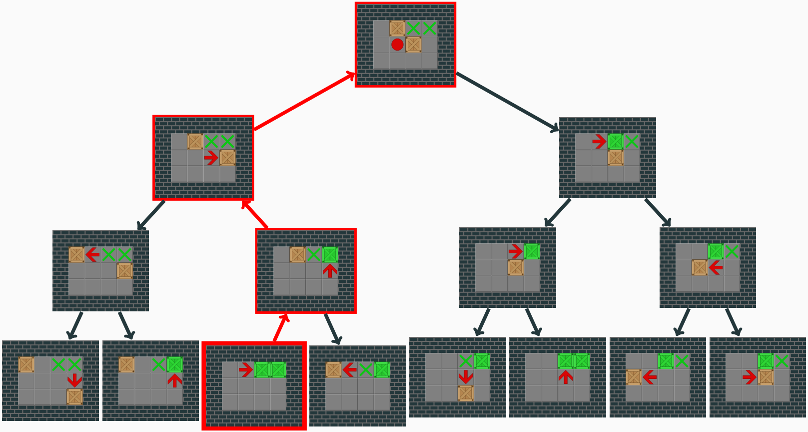


Principe de résolution

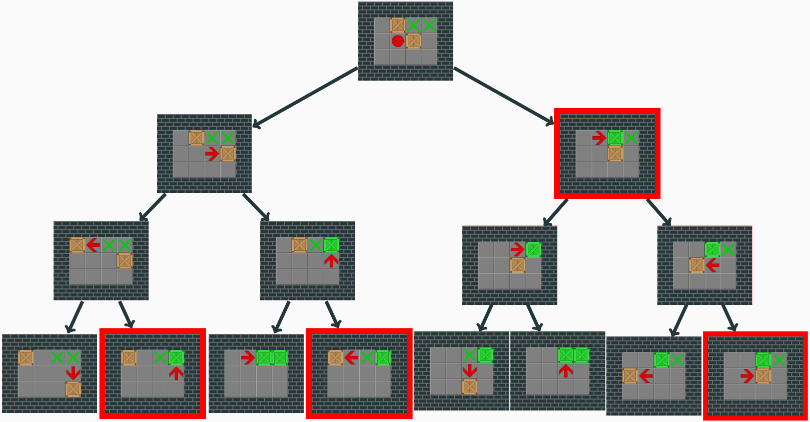
Arbre des états



Arbre des états



Arbre des états



Calcul du *hash* d'un état - Hash de Zobrist

Propriétés du **XOR** :

1. a **XOR** $a = 0$
2. **XOR** commutatif, associatif
3. **XOR** préserve l'aléatoire

Initialisation :

$$T = \begin{array}{cc} \text{caisse} & \text{joueur} \\ \left(\begin{array}{cc} 6357 & 5742 \\ -1378 & 42 \\ \vdots & \vdots \\ 93268 & -278 \end{array}\right) \end{array} \begin{array}{c} \text{case} \\ 0 \\ 1 \\ \vdots \\ wh-1 \end{array}$$

Calcul du *hash* d'un état - Hash de Zobrist

Usage : $(c_1, ..., c_n)$ n caisses et p position du joueur :

$h = \underset{i=0}{\overset{n}{\textbf{XOR}}}\, T[c_i][0]\,\textbf{XOR}\, T[p][1]$

Calculer le hash d'un état à l'aide de son parent : $c_i \to c_i', p \to p'$

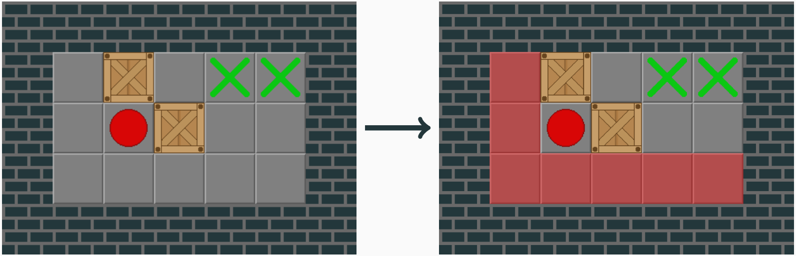
$h' = h\,\textbf{XOR}\, T[c_i][0]\,\textbf{XOR}\, T[c_i'][0]\,\textbf{XOR}\, T[p][1]\,\textbf{XOR}\, T[p'][1]$

Réduction de l’espace de recherche

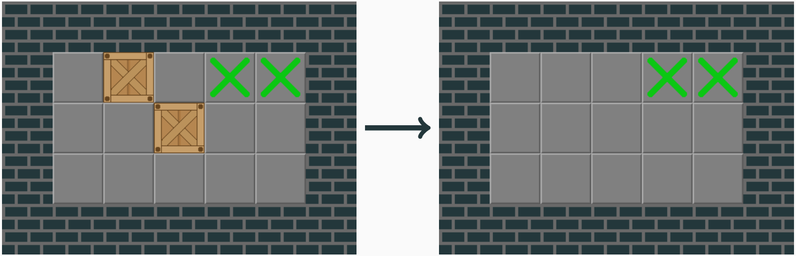
Réduction de l'espace de recherche

Analyse statique

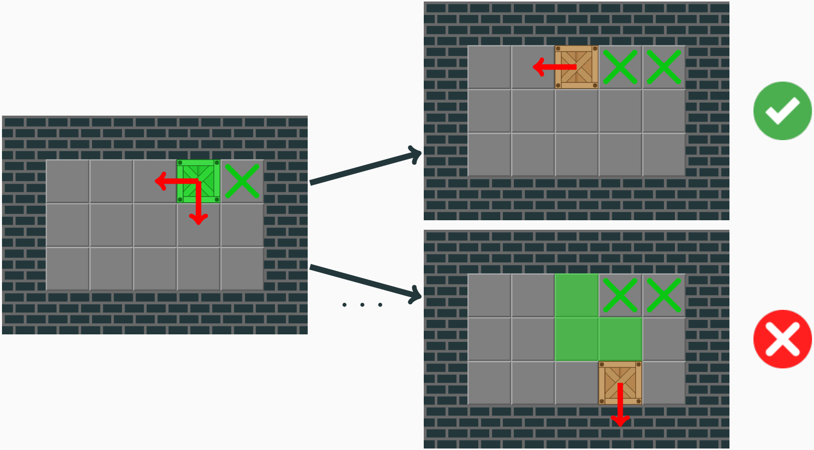
Détection des positions mortes *(dead positions)*



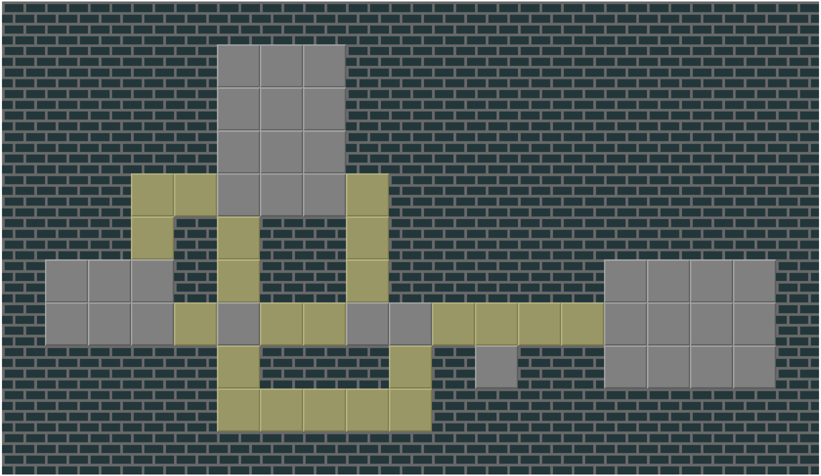
Détection des positions mortes *(dead positions)*



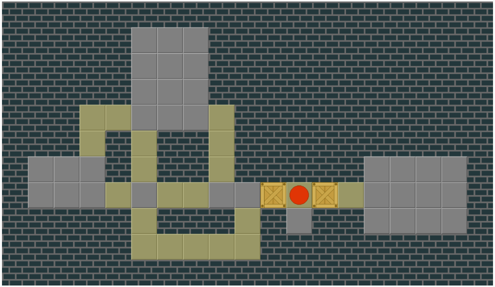
Détection des positions mortes *(dead positions)*

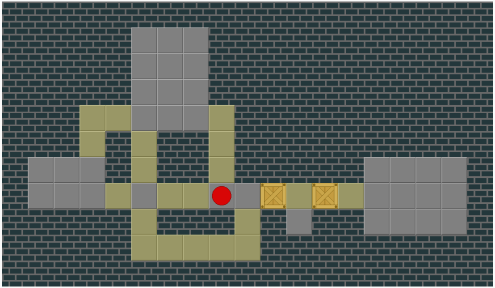


Détection de tunnels

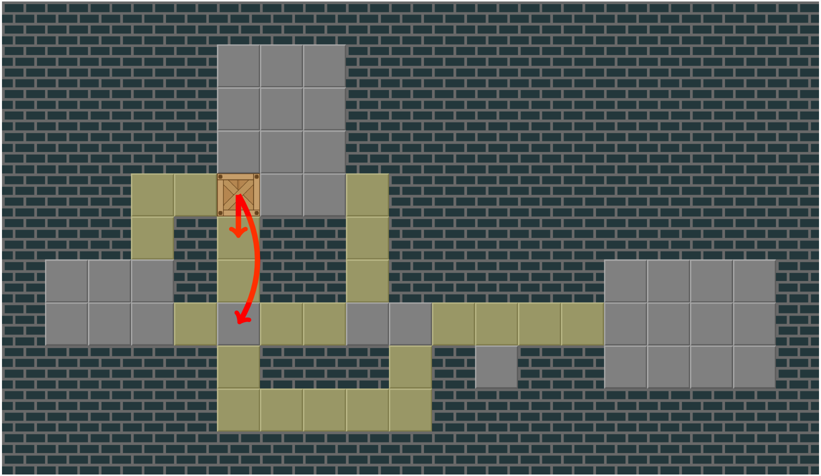


Détection de tunnels

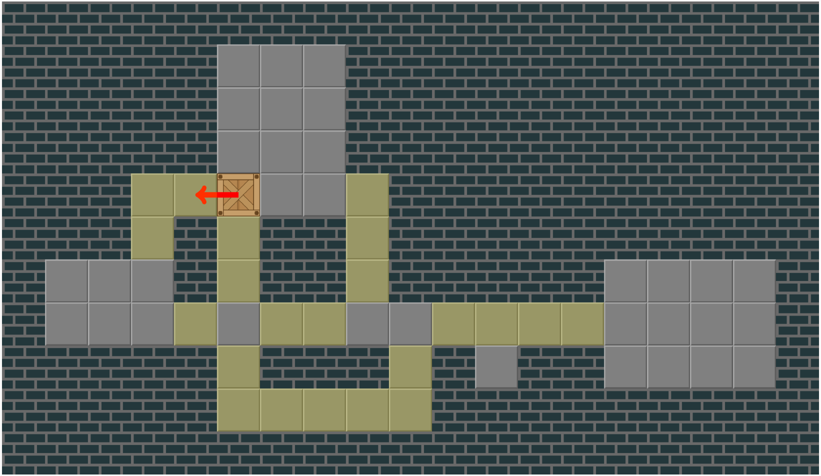




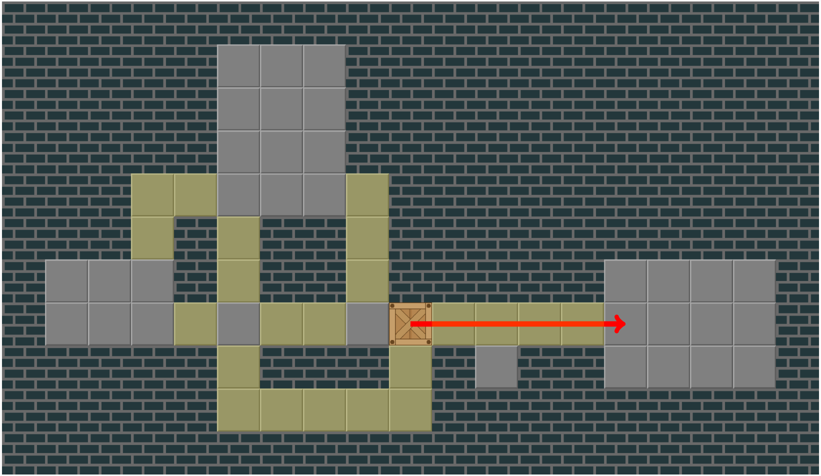
Détection de tunnels



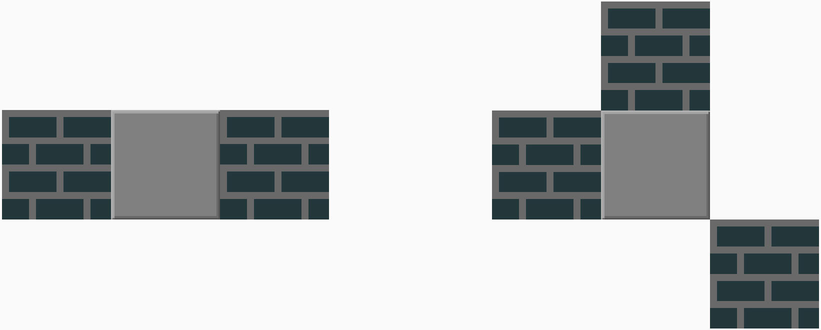
Détection de tunnels



Détection de tunnels

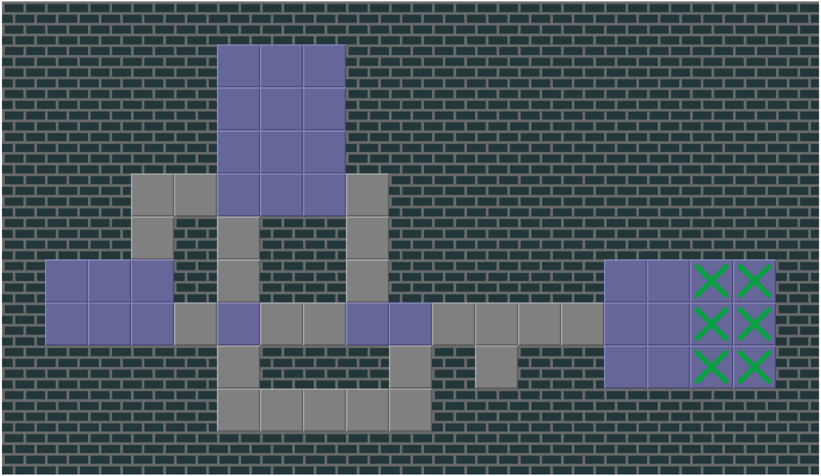


Détection de tunnels

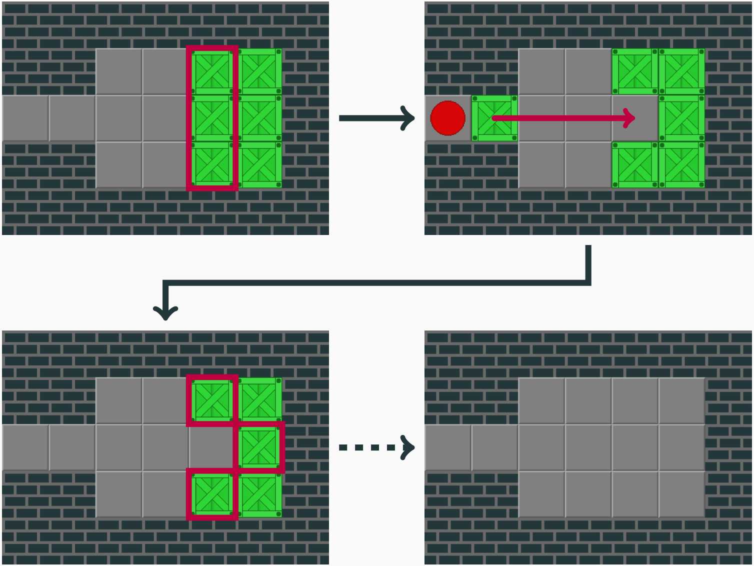


Composition d'un tunnel

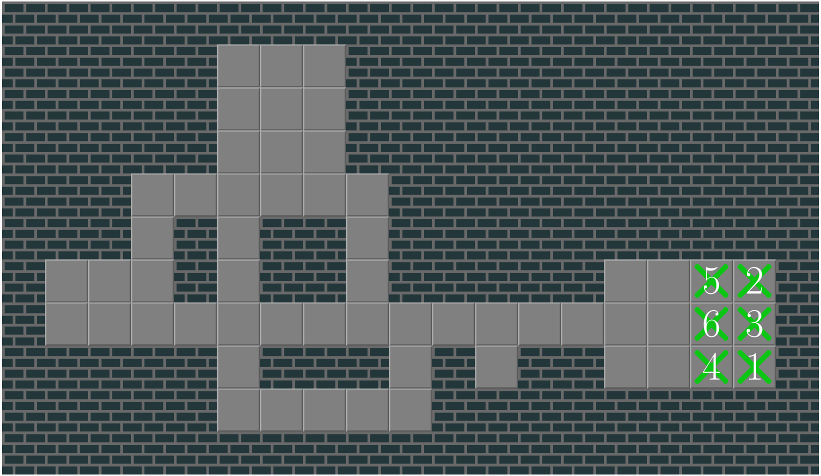
Salles et ordre de rangement *(packing order)*



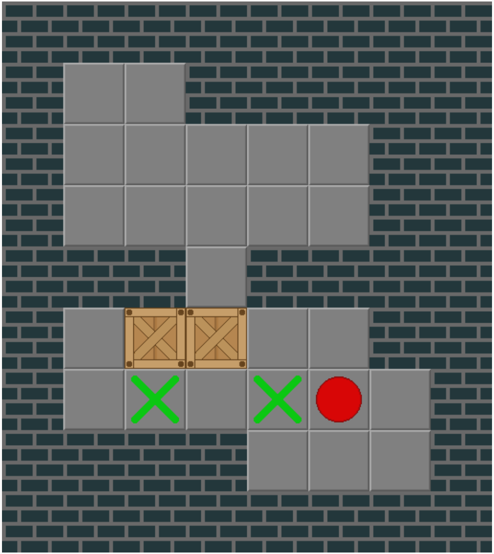
Salles et ordre de rangement *(packing order)*



Salles et ordre de rangement *(packing order)*



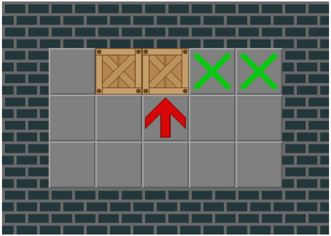
Salles et ordre de rangement *(packing order)*



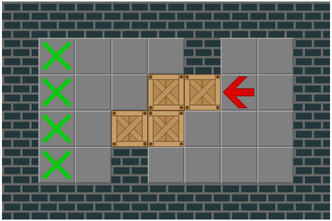
Réduction de l'espace de recherche

Analyse dynamique

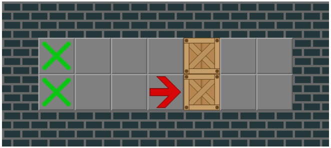
Détection d'impasses *(deadlocks)*



(a) *Freeze deadlock n°1*

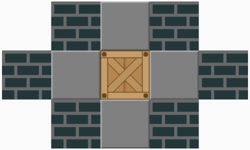


(b) *Freeze deadlock n°2*

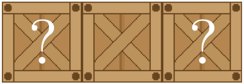


(c) *PI Corral deadlock*

(a) *Règle n°1*

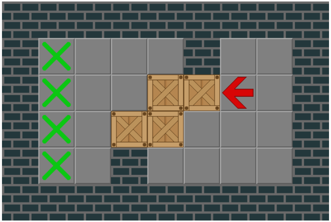


(b) *Règle n°2*

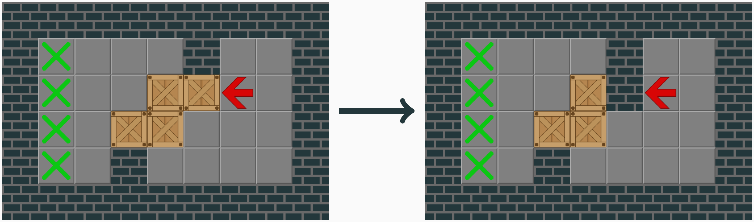


(c) *Règle n°3*

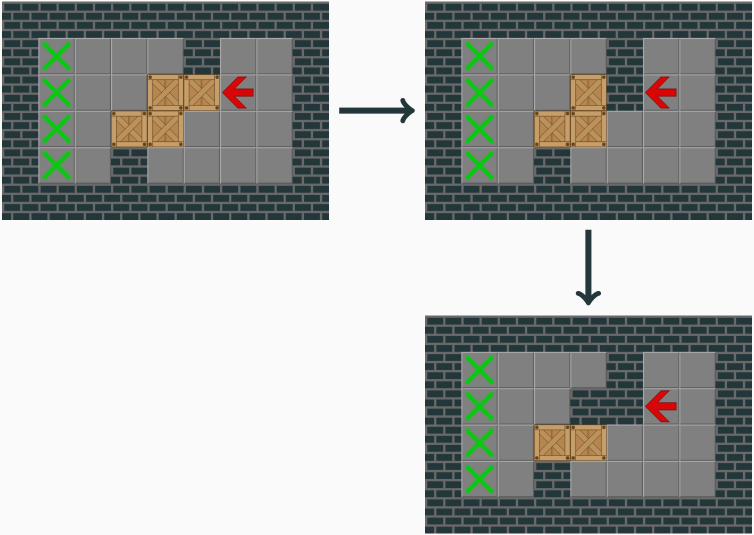
Détection de *freeze deadlocks*



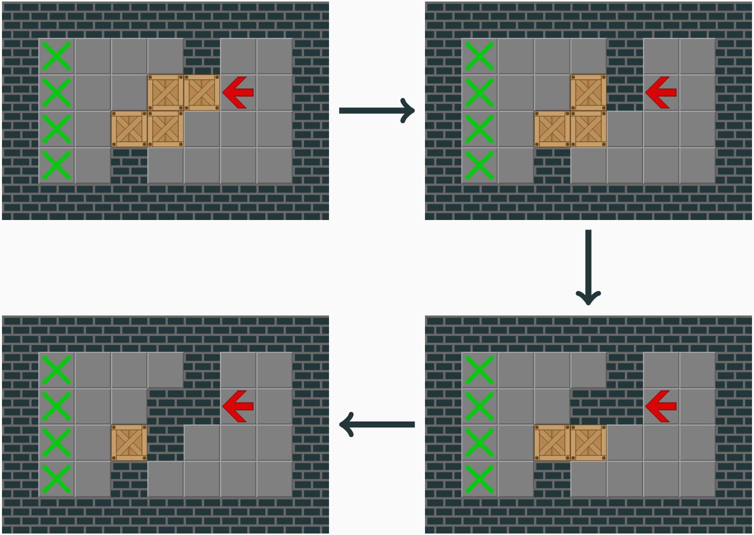
Détection de *freeze deadlocks*



Détection de *freeze deadlocks*

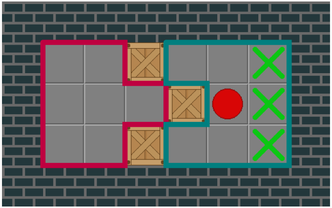


Détection de *freeze deadlocks*

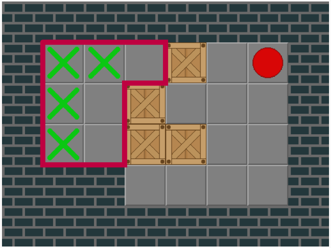


Gelée!

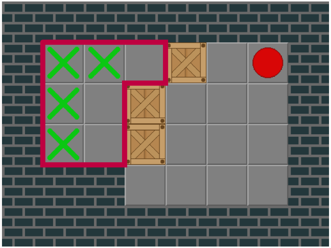
Détection de *PI Corral deadlocks*



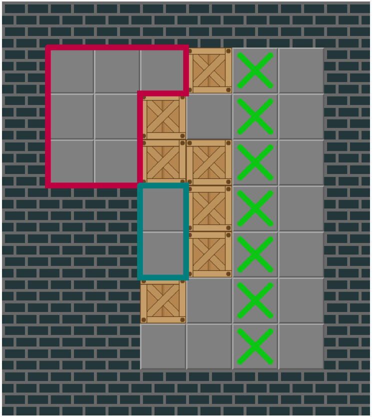
(a) *Corral*



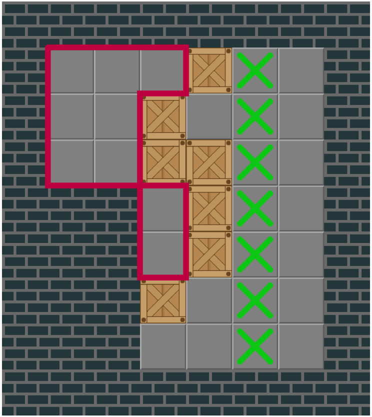
(b) *I Corral*



(c) *PI Corral*



Deux *I-Corrals*

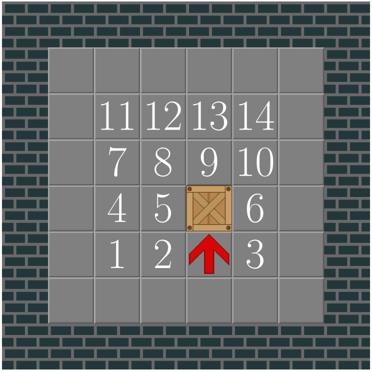


Un multi *PI-Corrals*

Détection de *PI Corral deadlocks*

Brian Damgaard : émonde d'au moins **20%** l'arbre de recherche !

Table de *deadlocks*



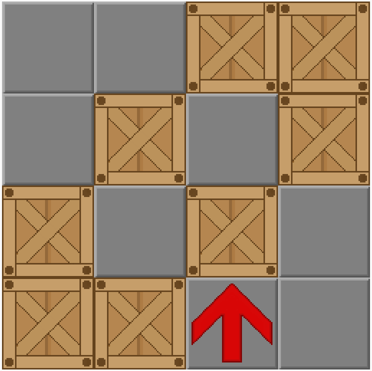
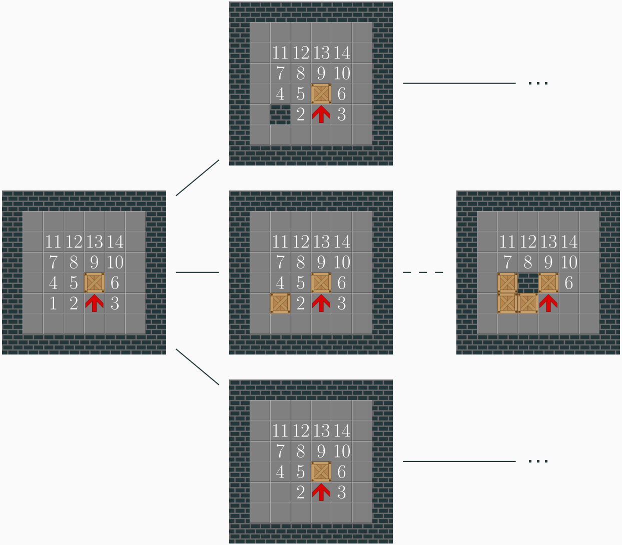
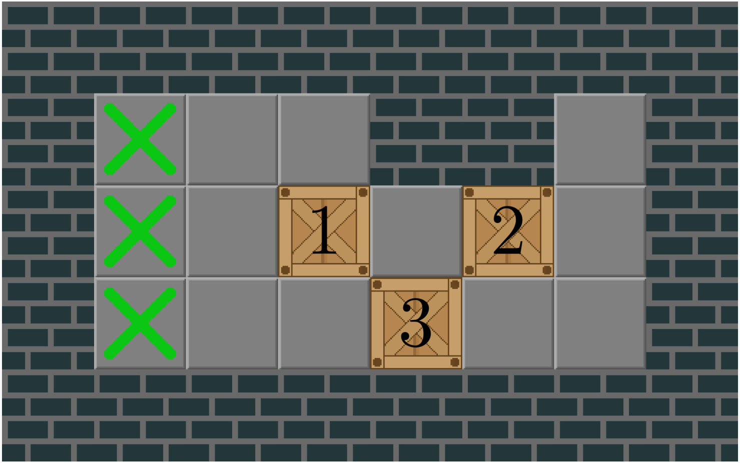


Table de *deadlocks*

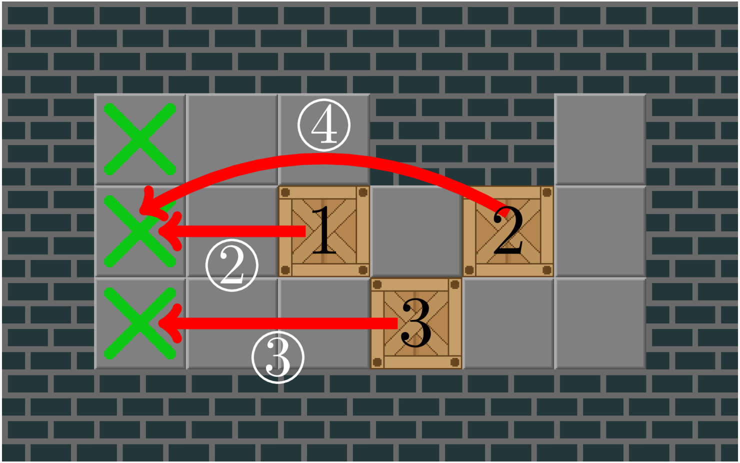


Recherche dirigée par une heuristique

Heuristique simple *(Simple Lower Bound)*

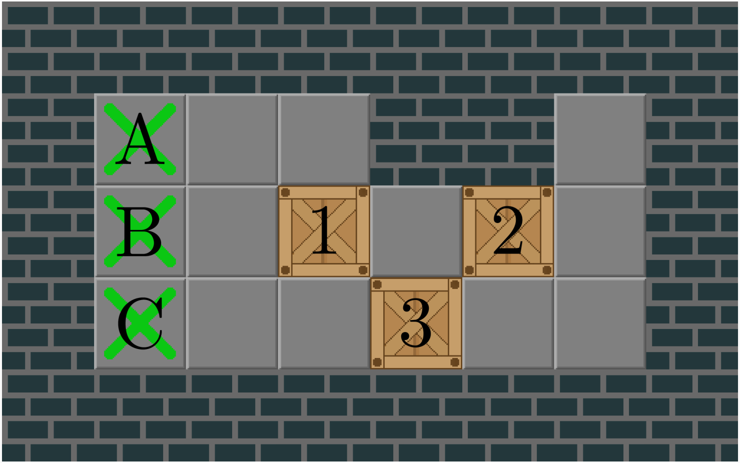


Heuristique simple *(Simple Lower Bound)*

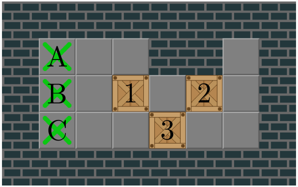


$$\boxed{2 + 4 + 3 = \mathbf{9}}$$

Heuristique gloutonne *(Greedy Lower Bound)*



Heuristique gloutonne *(Greedy Lower Bound)*

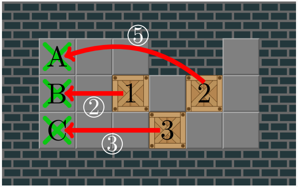


$1 \rightarrow A$	3
$1 \rightarrow B$	2
$1 \rightarrow C$	3
$2 \rightarrow A$	4
$2 \rightarrow B$	4
$2 \rightarrow C$	5
$3 \rightarrow A$	5
$3 \rightarrow B$	4
$3 \rightarrow C$	3

$\xrightarrow{\text{Tri}}$

$1 \rightarrow$ B	**2**
$1 \rightarrow A$	3
$1 \rightarrow C$	3
$3 \rightarrow$ C	**3**
$2 \rightarrow B$	4
$3 \rightarrow B$	4
$2 \rightarrow A$	5
$2 \rightarrow C$	5
$3 \rightarrow$ A	**5**

Heuristique gloutonne *(Greedy Lower Bound)*



$$\boxed{2 + 3 + 5 = \mathbf{10}}$$

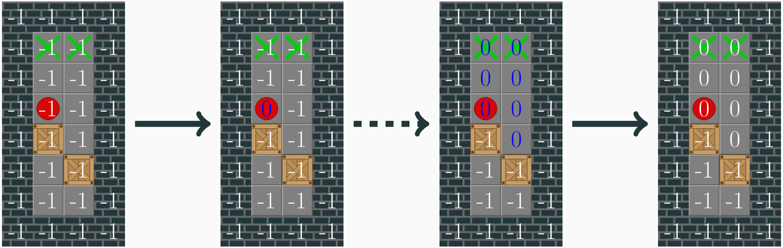
$1 \rightarrow A$	3
$1 \rightarrow B$	2
$1 \rightarrow C$	3
$2 \rightarrow A$	4
$2 \rightarrow B$	4
$2 \rightarrow C$	5
$3 \rightarrow A$	5
$3 \rightarrow B$	4
$3 \rightarrow C$	3

$\xrightarrow{\text{Tri}}$

$\mathbf{1 \rightarrow B}$	**2**
$1 \rightarrow A$	3
$1 \rightarrow C$	3
$\mathbf{3 \rightarrow C}$	**3**
$2 \rightarrow B$	4
$3 \rightarrow B$	4
$2 \rightarrow A$	5
$2 \rightarrow C$	5
$\mathbf{3 \rightarrow A}$	**5**

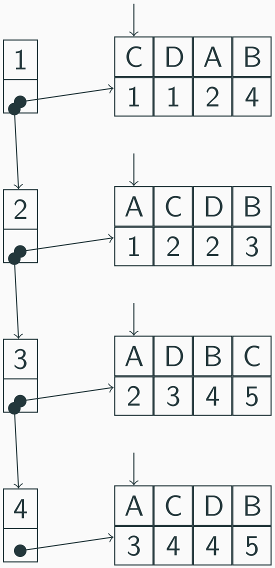
Optimisations

Parcours de graphes : démarquer tous les noeuds en $\mathcal{O}(1)$



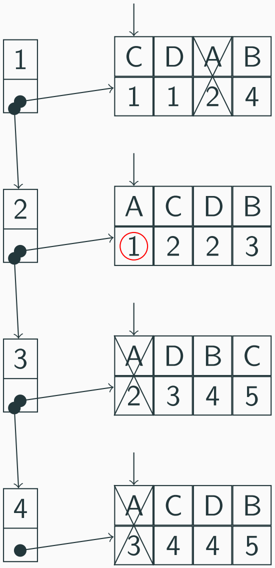
$m = 0$ $m = 0$ $m = 0$ $m = 1$

Greedy Lower Bound en $\mathcal{O}(n^2)$



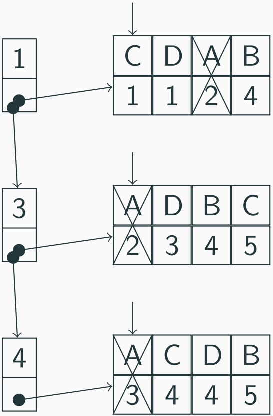
$h =$

Greedy Lower Bound en $\mathcal{O}(n^2)$



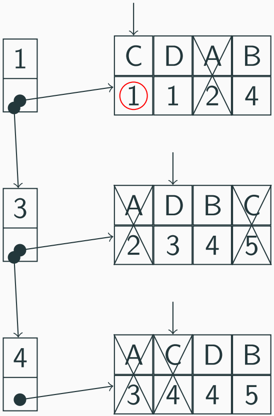
$h = 1+$

Greedy Lower Bound en $\mathcal{O}(n^2)$



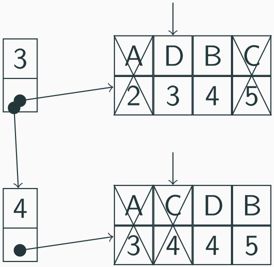
$h = 1+$

Greedy Lower Bound en $\mathcal{O}(n^2)$



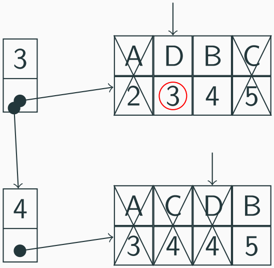
$h = 1 + 1 +$

Greedy Lower Bound en $\mathcal{O}(n^2)$



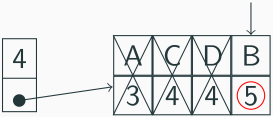
$h = 1 + 1 +$

Greedy Lower Bound en $\mathcal{O}(n^2)$



$h = 1 + 1 + 3 +$

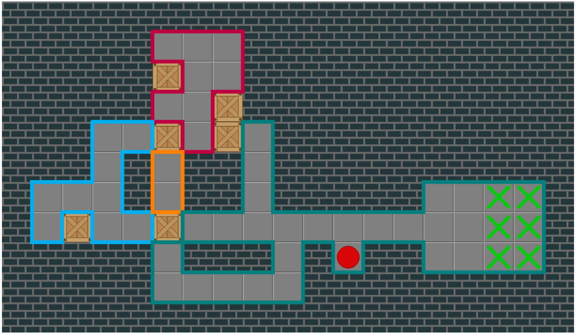
Greedy Lower Bound en $\mathcal{O}(n^2)$



$$h = 1 + 1 + 3 + 5 = 10$$

Calcul des *corrals* en $\mathcal{O}(wh)$

Utilisation de *Union-Find* : partition de $[\![0; wh-1]\!]$.



Calcul des *corrals* en $\mathcal{O}(wh)$

```
procedure CORRAL(x, y)
    if not solid(x,y) then
        createSingleton(x, y)
    else
        if solid(x-1, y) and solid(x,y-1) then
            createSingleton(x, y)
        else if not solid(x-1, y) and solid(x,y-1) then
            addToCorral(x-1,y, x,y)
        else if solid(x-1, y) and not solid(x,y-1) then
            addToCorral(x,y-1, x,y)
        else
            addToCorral(x-1,y, x,y)
            union(x,y-1, x,y)
        end if
    end if
```

Résultats

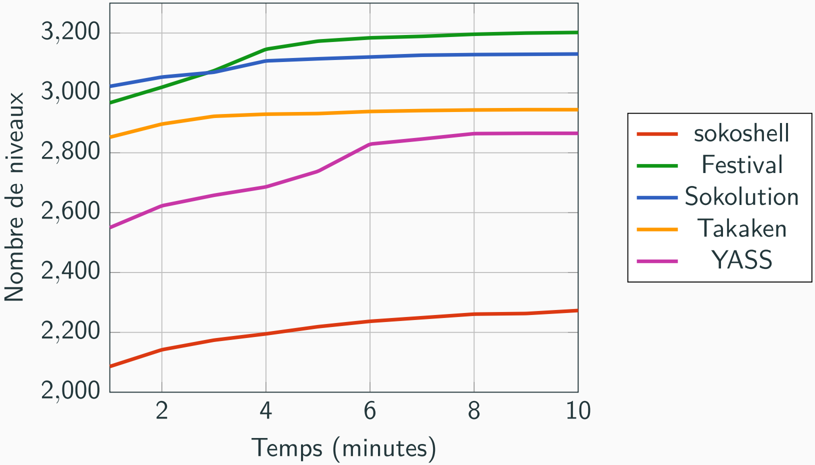
Nombre de niveaux résolus

Limite de temps : 10 min. Limite de RAM : 32 Gb.

Ensemble de niveaux	XSokoban	*Large test suite*
Nombre de niveaux	90	3272
A*	11	2204
fess0	15	2273
Festival (Yaron Shoham)	90	3202
Sokolution (Florent Diedler)	90	3130
Takaken (Ken'ichiro Takahashi)	90	2944
YASS (Brian Damgaard)	89	2865

Statistiques

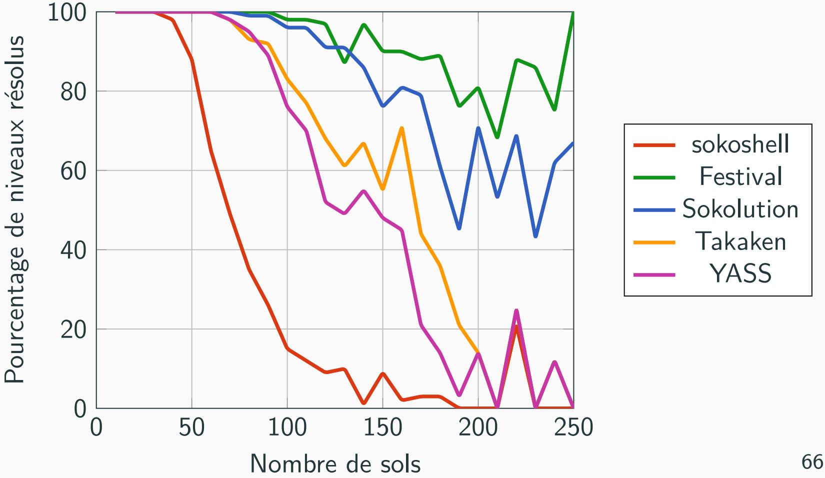
Nombre de niveaux résolus (cumulés) en fonction du temps



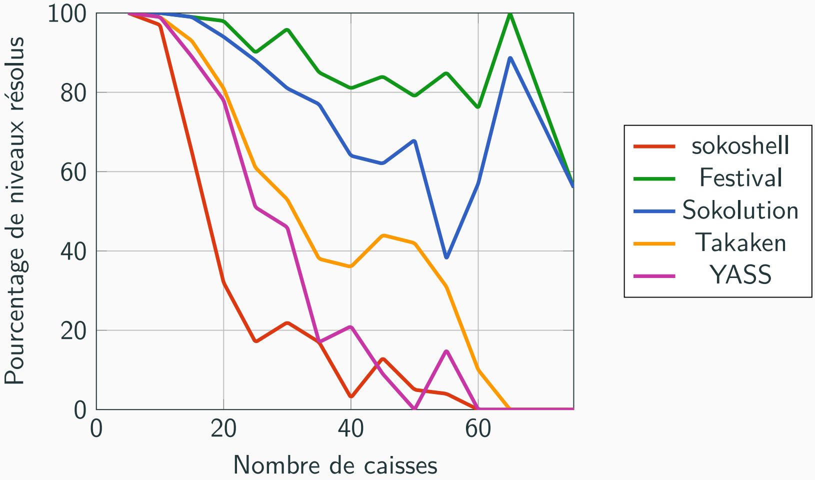
Statistiques

Temps moyen passé par niveaux

Solveur	A*	fess0	Festival	Sokolution	Takaken	YASS
Temps moyen	3min 28s	3min 16s	3s	2s	7s	24s



Statistiques



Annexe

Tableau des complexités