

Résolution de niveaux du Sokoban

Nom, Prénom

16 mai 2023

Numéro de candidat

Le jeu du Sokoban

Principe de résolution

Réduction de l'espace de recherche

- Analyse statique

- Analyse dynamique

Recherche dirigée par une heuristique

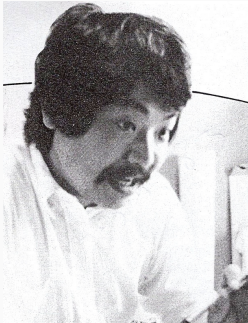
Optimisations

Résultats

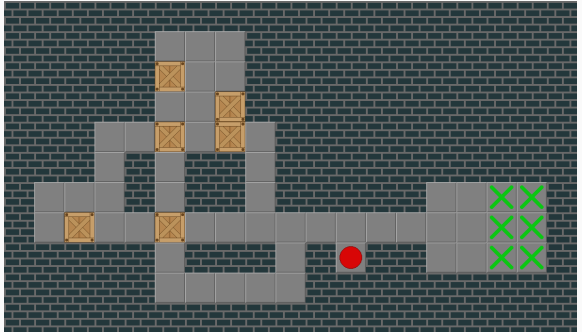
Annexe

Le jeu du Sokoban

Le jeu du Sokoban



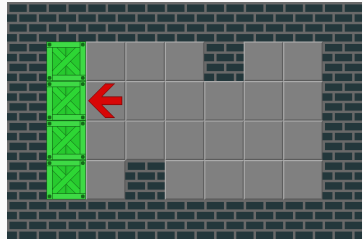
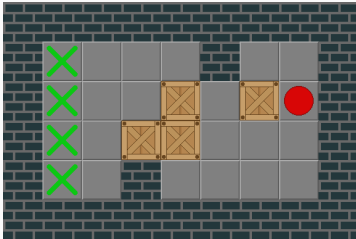
Hiroyuki Imabayashi

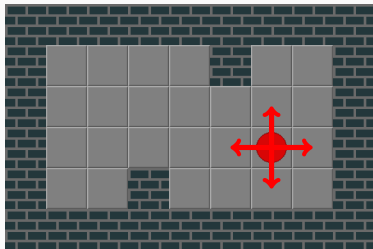


X Sokoban

But du jeu

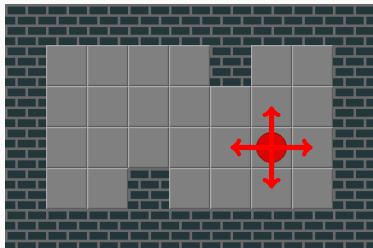
Déplacements



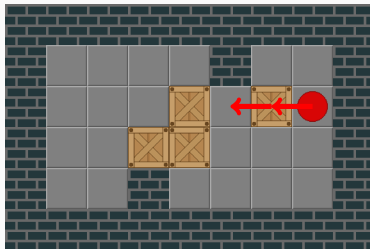


Déplacements autorisés

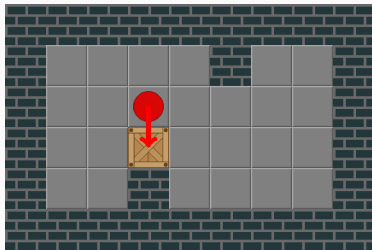
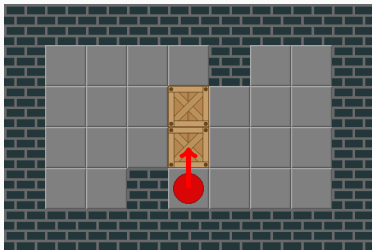
Règles



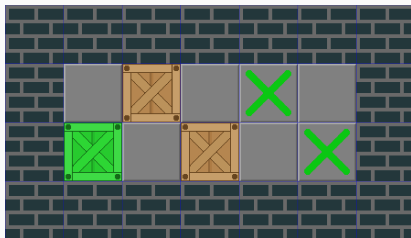
Déplacements autorisés



Règles



Tuiles



Mur



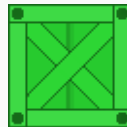
Sol



Caisse



Cible



Caisse sur une cible

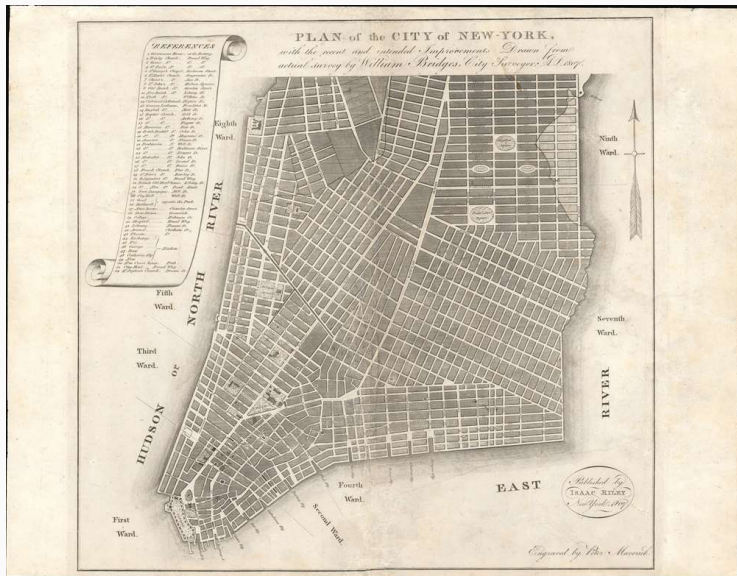
Quelles stratégies adopter pour trouver une solution le plus rapidement possible à un niveau de Sokoban ?

```
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

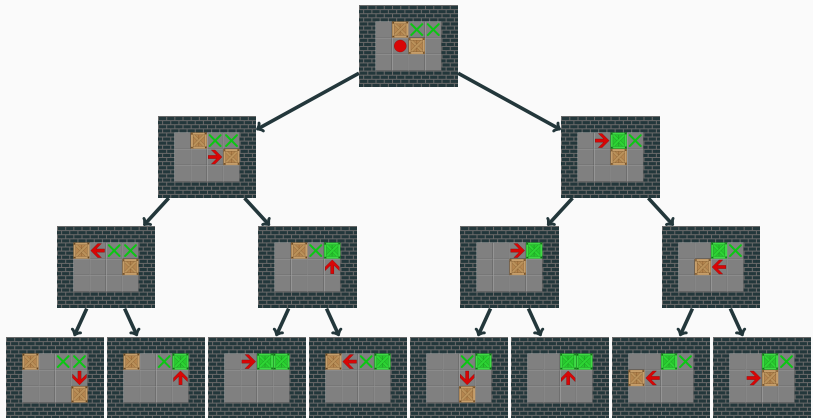


Lien avec le thème de l'année

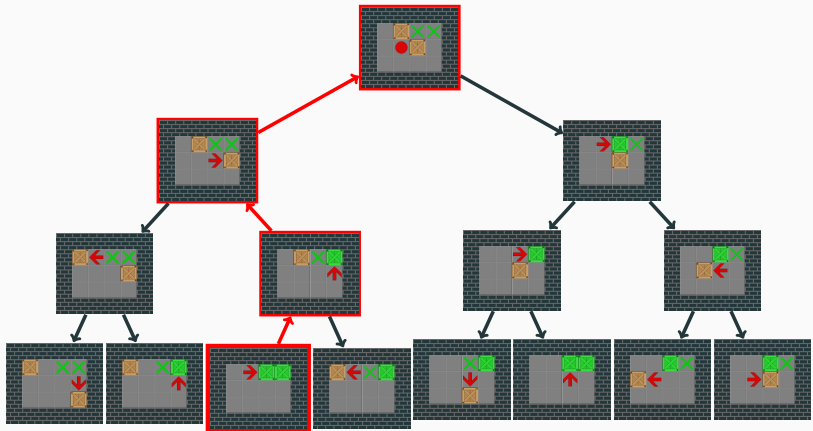


Principe de résolution

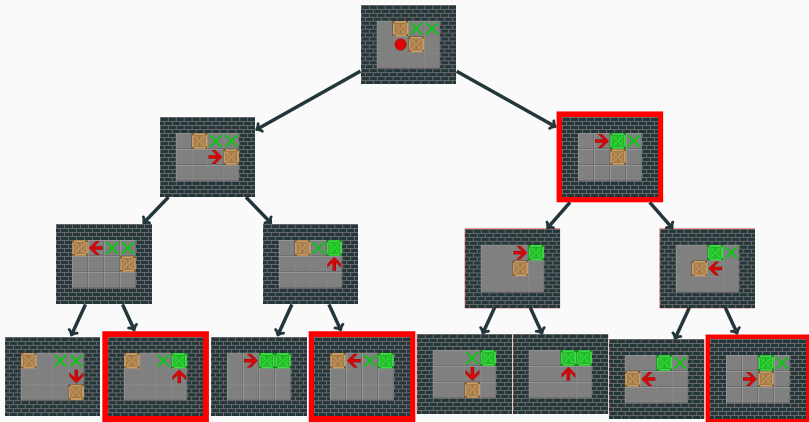
Arbre des états



Arbre des états



Arbre des états



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

Propriétés du **XOR** :

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

Initialisation :

$$T = \begin{matrix} & \begin{matrix} \text{caisse} & \text{joueur} & \text{case} \end{matrix} \\ \begin{pmatrix} 6357 & 5742 \\ -1378 & 42 \\ \vdots & \vdots \\ 93268 & -278 \end{pmatrix} & \begin{matrix} 0 \\ 1 \\ \vdots \\ wh - 1 \end{matrix} \end{matrix}$$

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

Usage : (c_1, \dots, c_n) n caisses et p position du joueur :

$$h = \mathbf{XOR}_{i=0}^n T[c_i][0] \mathbf{XOR} T[p][1]$$

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

$$h' = h \mathbf{XOR} T[c_i][0] \mathbf{XOR} T[c'_i][0] \mathbf{XOR} T[p][1] \mathbf{XOR} T[p'][1]$$

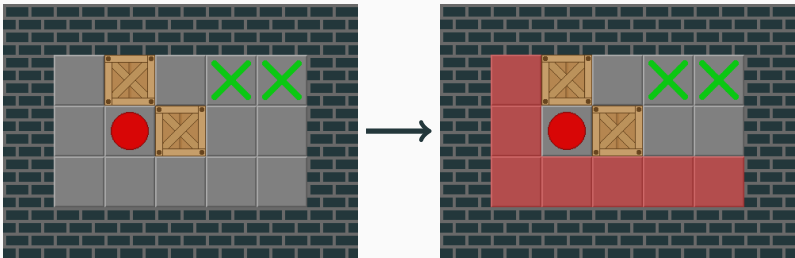
Réduction de l'espace de recherche

Réduction de l'espace de recherche

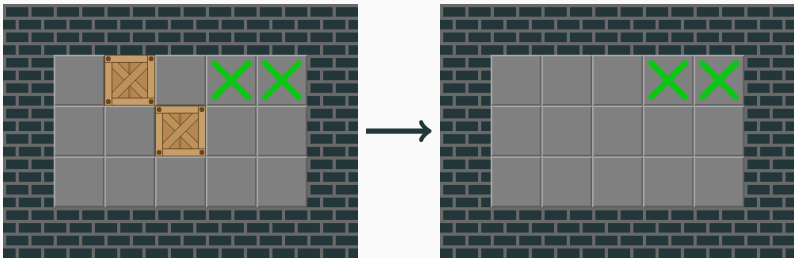


Analyse statique

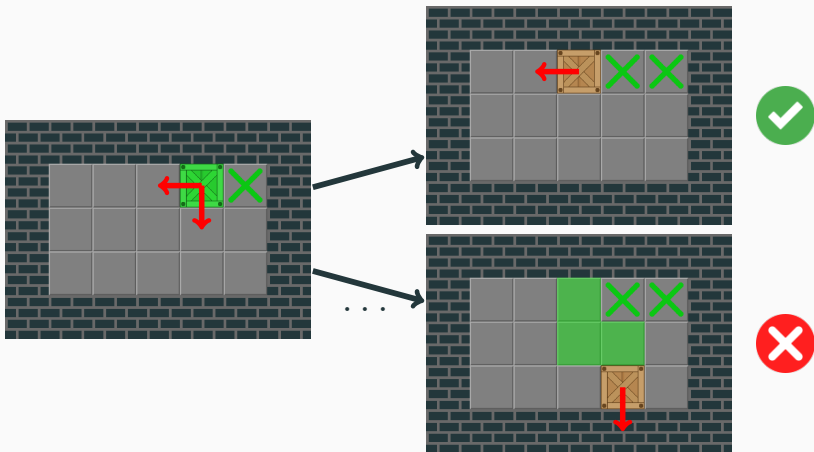
Détection des positions mortes (*dead positions*)



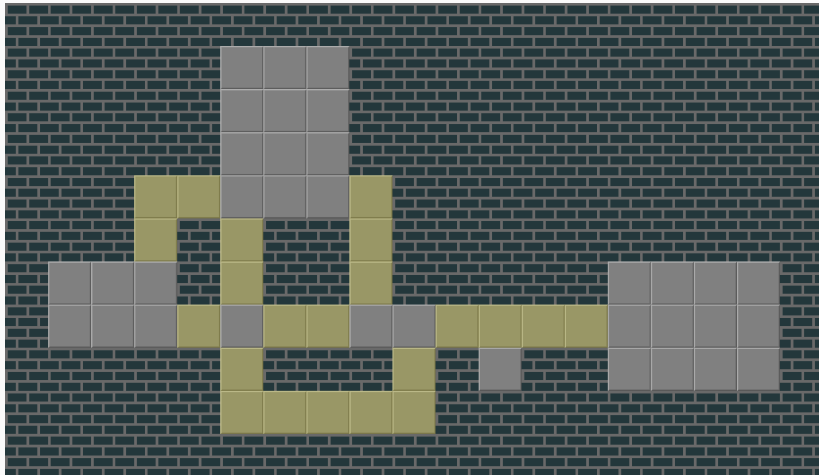
Détection des positions mortes (*dead positions*)



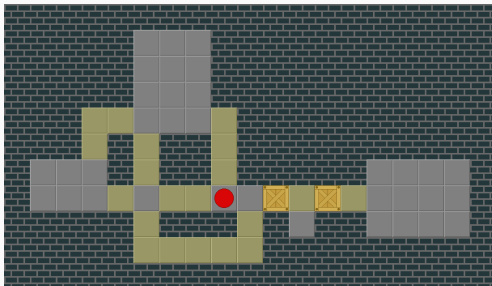
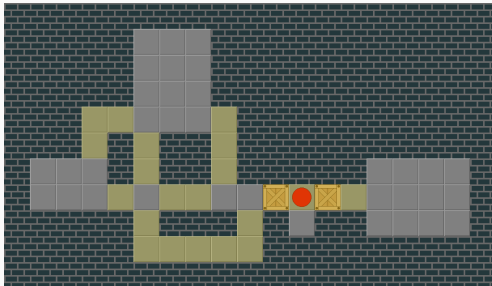
Détection des positions mortes (*dead positions*)



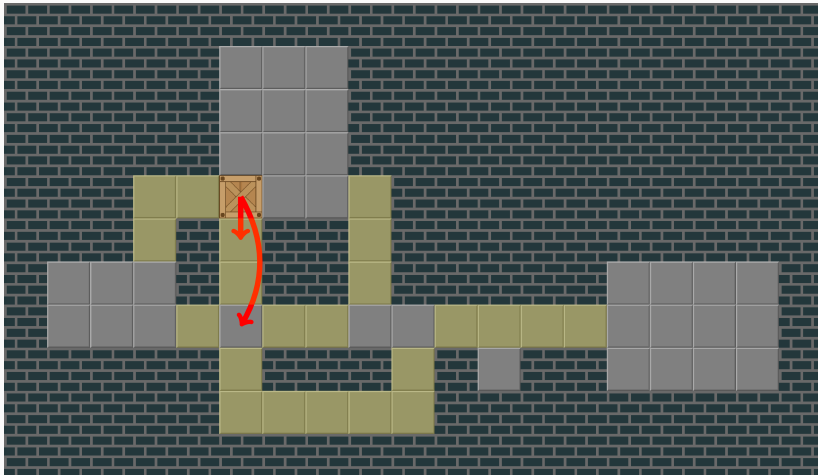
Détection de tunnels



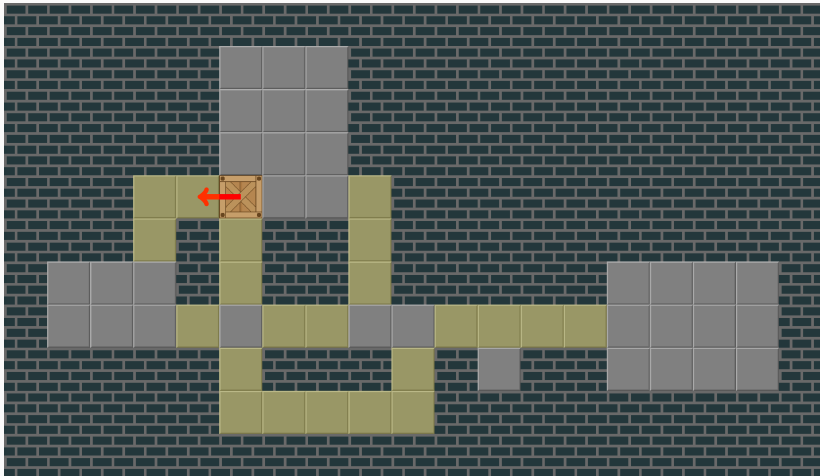
Détection de tunnels



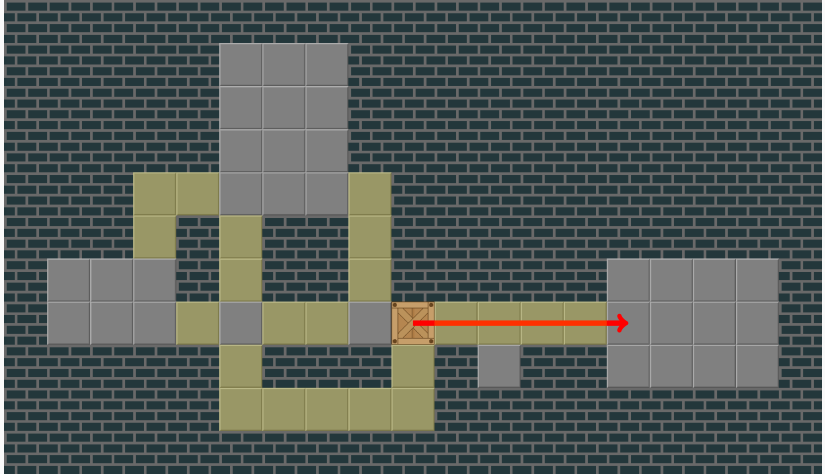
Détection de tunnels



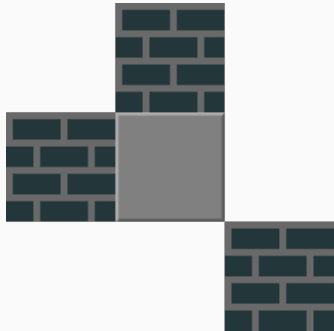
Détection de tunnels



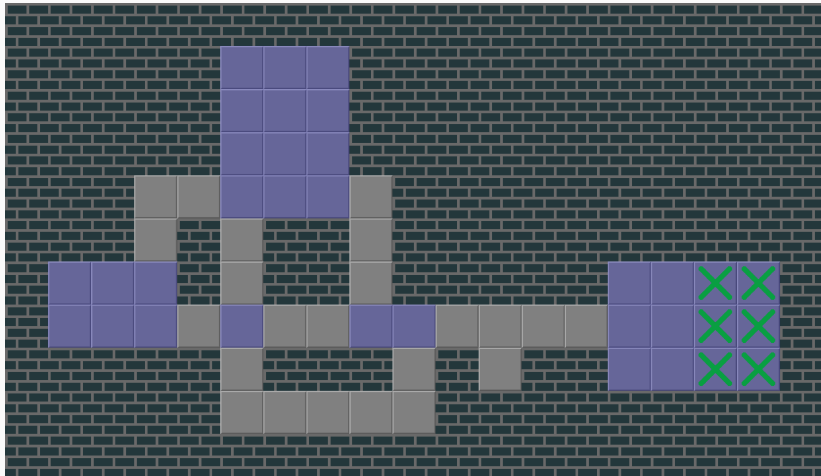
Détection de tunnels



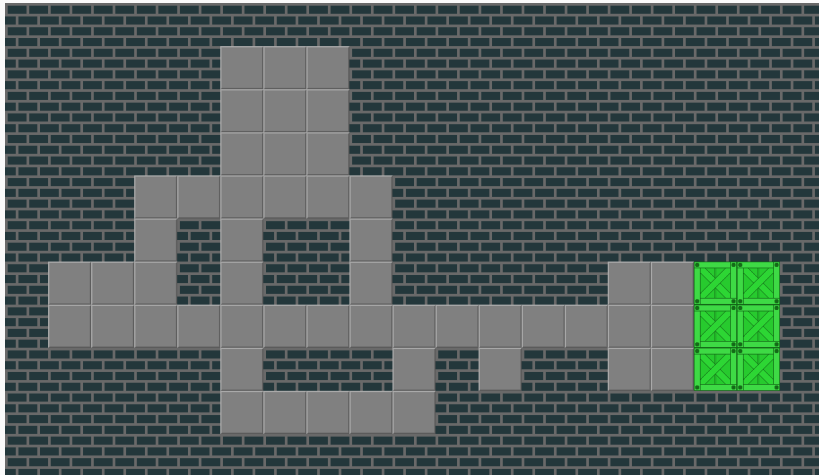
Détection de tunnels



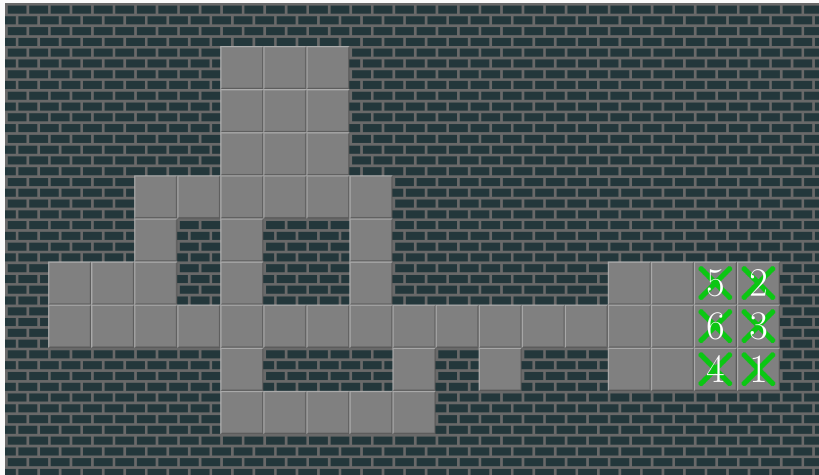
Salles et ordre de rangement (*packing order*)



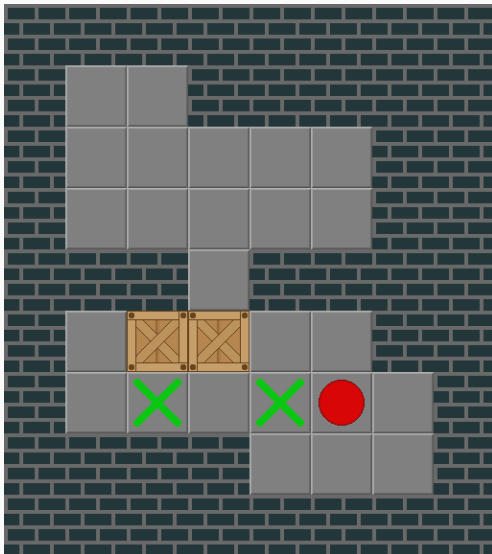
Salles et ordre de rangement (*packing order*)



Salles et ordre de rangement (*packing order*)



Salles et ordre de rangement (*packing order*)

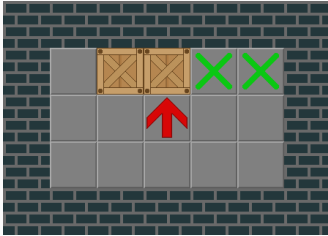


Réduction de l'espace de recherche

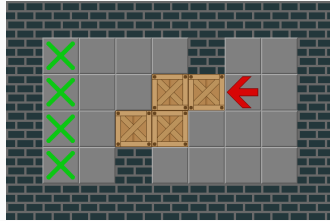


Analyse dynamique

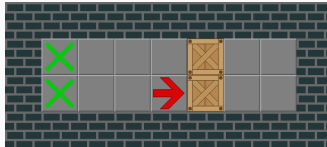
Détection d'impasses (*deadlocks*)



(a) *Freeze deadlock n°1*



(b) *Freeze deadlock n°2*

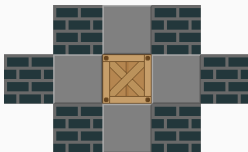


(c) *PI Corral deadlock*

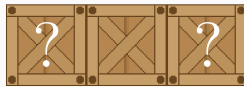
Détection de *freeze deadlocks*



(a) Règle n°1

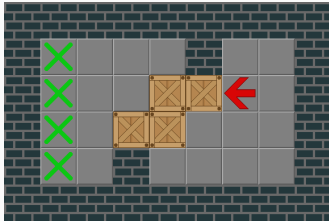


(b) Règle n°2

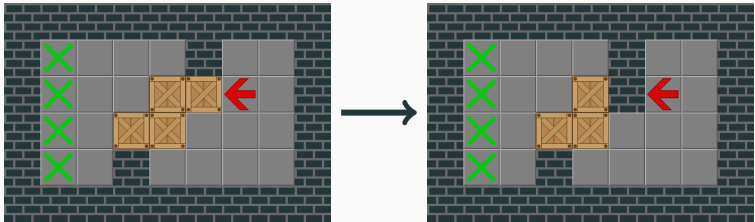


(c) Règle n°3

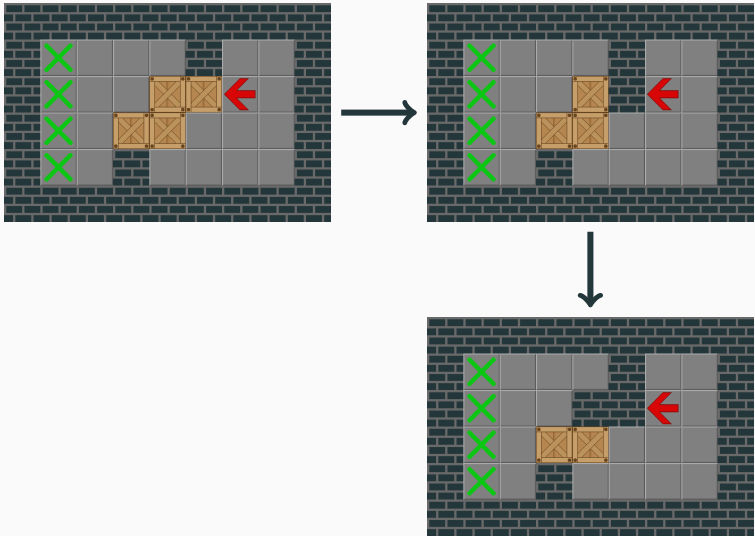
Détection de *freeze deadlocks*



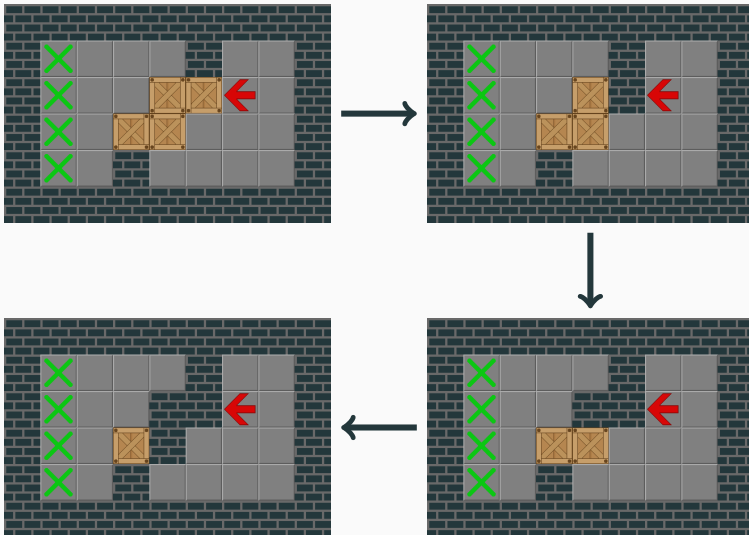
Détection de *freeze deadlocks*



Détection de *freeze deadlocks*

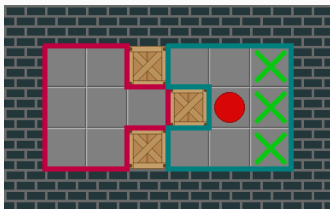


Détection de *freeze deadlocks*

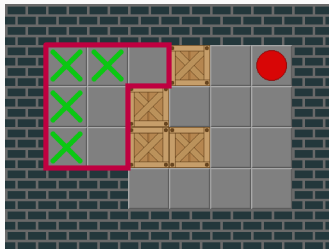


Gelée!

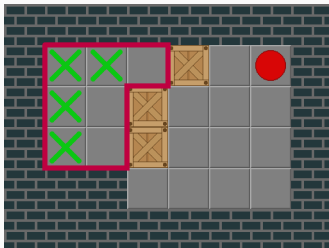
Détection de *PI Corral* deadlocks



(a) *Corral*



(b) *I Corral*



(c) *PI Corral*

Détection de *PI Corral* deadlocks

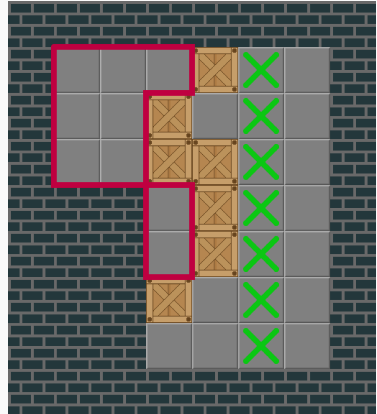
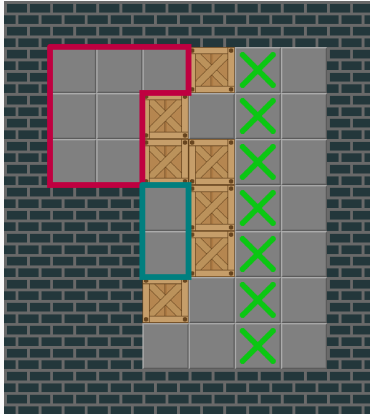


Table de *deadlocks*

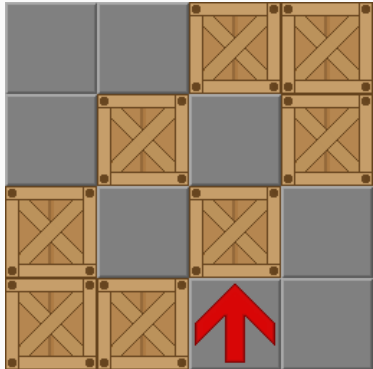
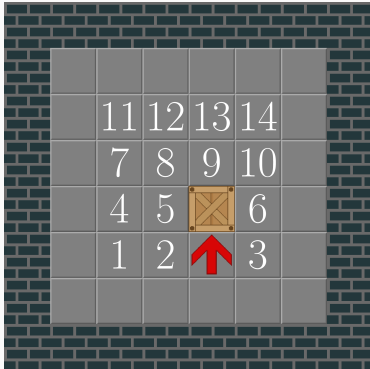
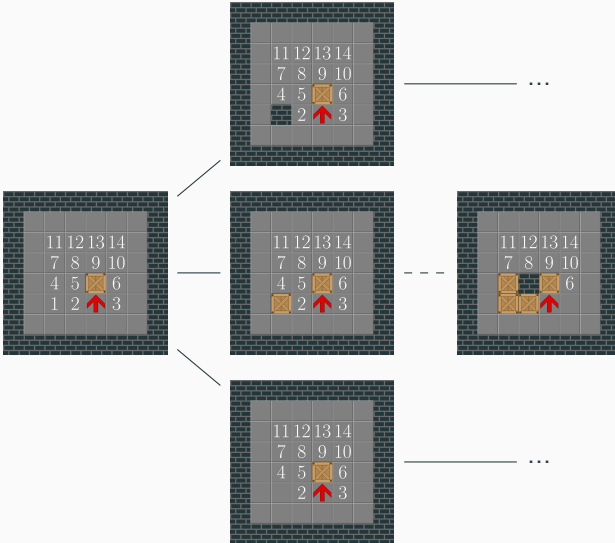


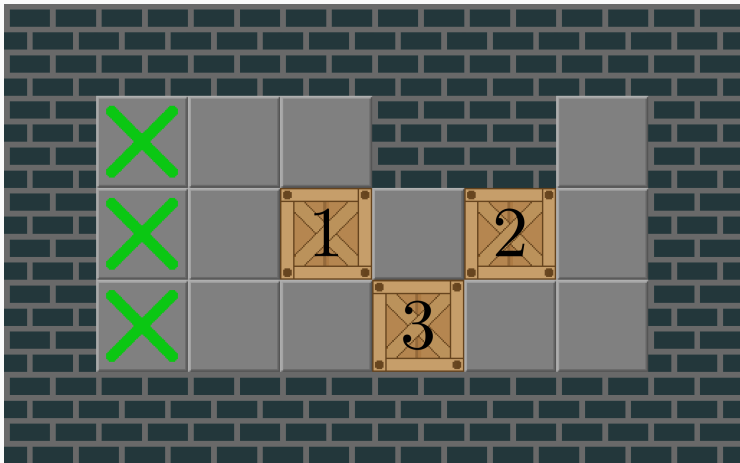
Table de *deadlocks*



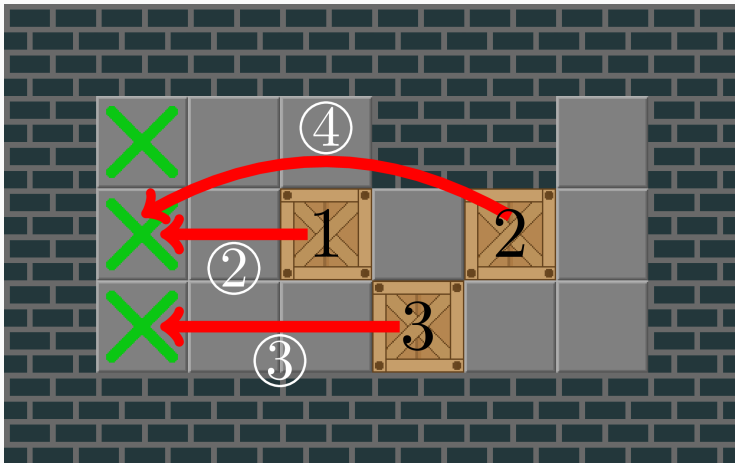
Recherche dirigée par une heuristique



Heuristique simple (*Simple Lower Bound*)

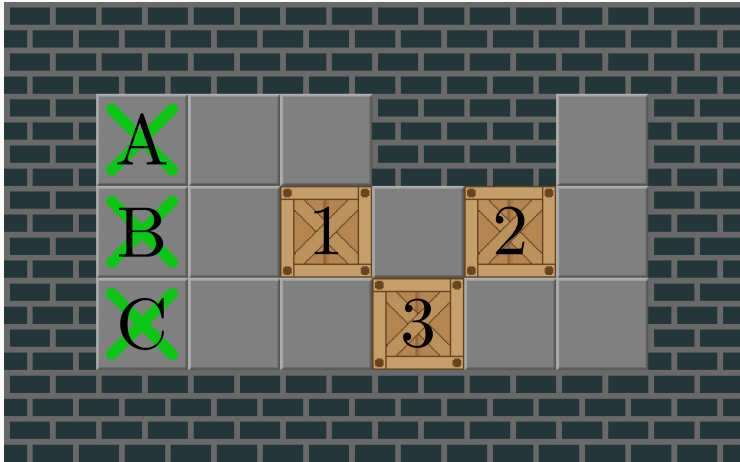


Heuristique simple (*Simple Lower Bound*)

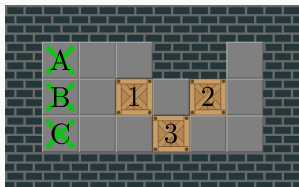


$$2 + 4 + 3 = 9$$

Heuristique gloutonne (*Greedy Lower Bound*)



Heuristique gloutonne (*Greedy Lower Bound*)

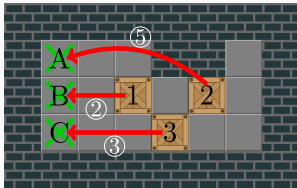


$1 \rightarrow A$	3
$1 \rightarrow B$	2
$1 \rightarrow C$	3
$2 \rightarrow A$	4
$2 \rightarrow B$	4
$2 \rightarrow C$	5
$3 \rightarrow A$	5
$3 \rightarrow B$	4
$3 \rightarrow C$	3



$1 \rightarrow B$	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*)



$$2 + 3 + 5 = 10$$

$1 \rightarrow A$	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

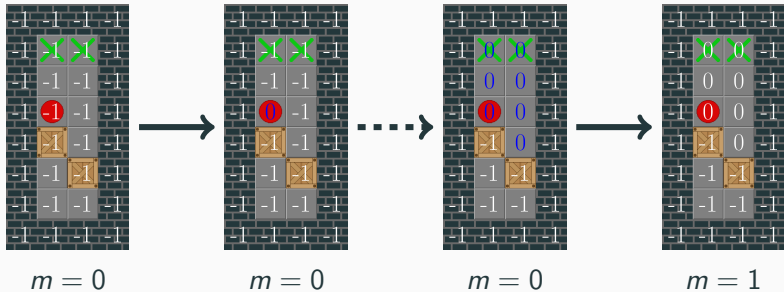


$1 \rightarrow B$	2
$1 \rightarrow A$	3
$1 \rightarrow C$	3
$3 \rightarrow C$	3
$2 \rightarrow B$	4
$3 \rightarrow B$	4
$2 \rightarrow A$	5
$2 \rightarrow C$	5
$3 \rightarrow A$	5

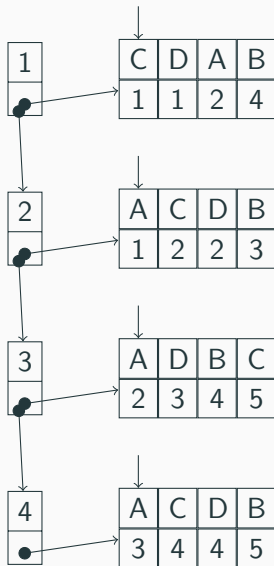
Optimisations



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

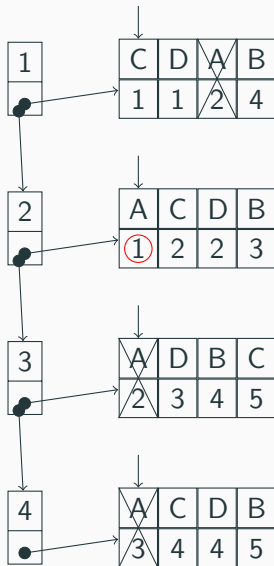


Greedy Lower Bound en $\mathcal{O}(n^2)$



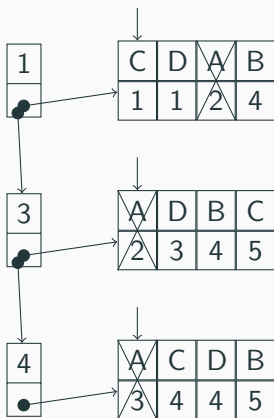
$h =$

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



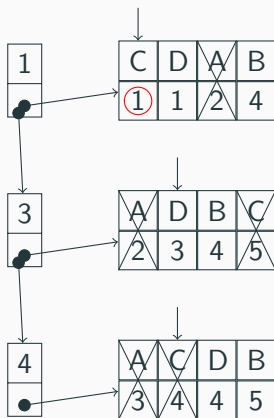
$$h = 1 +$$

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



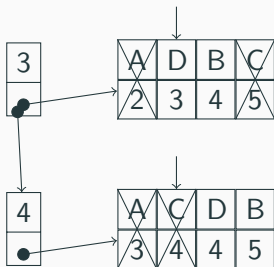
$$h = 1 +$$

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



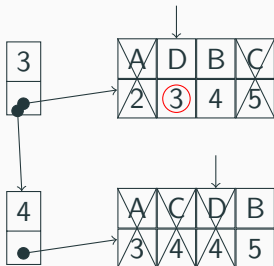
$$h = 1 + 1 +$$

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



$$h = 1 + 1 +$$

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



$$h = 1 + 1 + 3 +$$

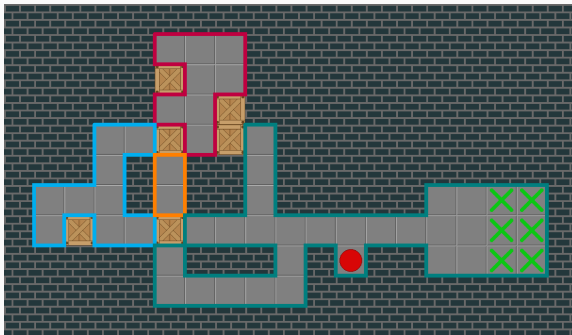
Greedy Lower Bound en $\mathcal{O}(n^2)$



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

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

Utilisation de *Union-Find* : partition de $\llbracket 0; wh - 1 \rrbracket$.



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

```
1: procedure CORRAL( $x, y$ )
2:   if not solid( $x, y$ ) then
3:     createSingleton( $x, y$ )
4:   else
5:     if solid( $x-1, y$ ) and solid( $x, y-1$ ) then
6:       createSingleton( $x, y$ )
7:     else if not solid( $x-1, y$ ) and solid( $x, y-1$ ) then
8:       addToCorral( $x-1, y, x, y$ )
9:     else if solid( $x-1, y$ ) and not solid( $x, y-1$ ) then
10:      addToCorral( $x, y-1, x, y$ )
11:    else
12:      addToCorral( $x-1, y, x, y$ )
13:      union( $x, y-1, x, y$ )
14:    end if
15:  end if
```

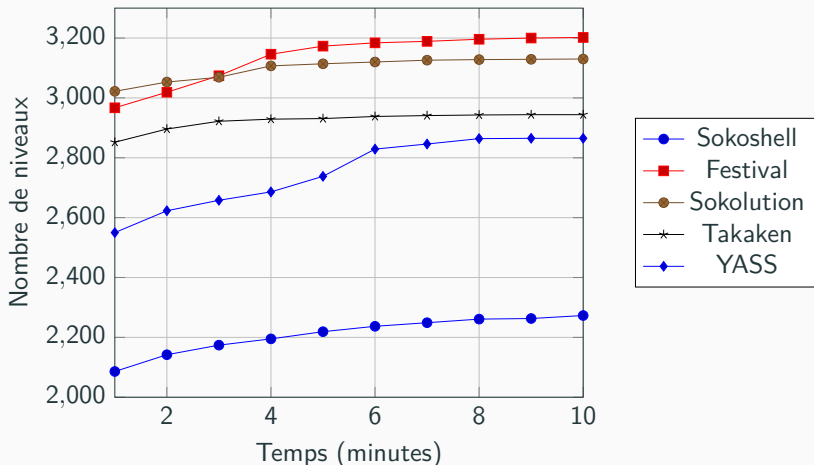
Résultats



Nombre de niveaux résolus

Collection	Nombre de niveaux	A*	fess0	Festival	Sokolution	Takaken	YASS
XSokoban	90	11	15	90	90	90	89
Large test suite	3272	2204	2273	3202	3130	2944	2865

Nombre de niveaux résolus (cumulés) en fonction du temps



Temps moyen passé par niveaux

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

Annexe



Tableau des complexités