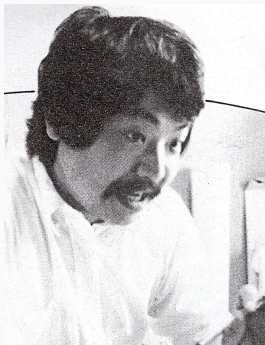


Résolution de niveaux du Sokoban

Carrez Valentin

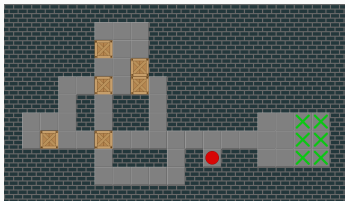
Candidat n°31593

Le jeu du Sokoban

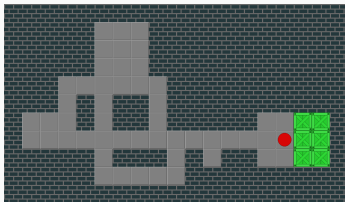


<https://shmuplations.com/wp-content/uploads/2022/03/thinkingrabbit04.jpg>

Hiroyuki Imabayashi



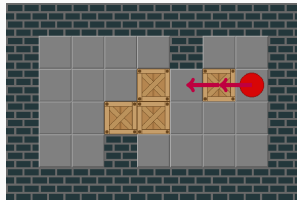
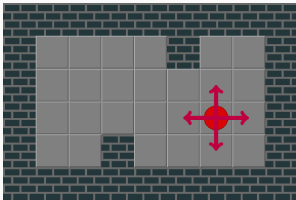
XSokoban 1



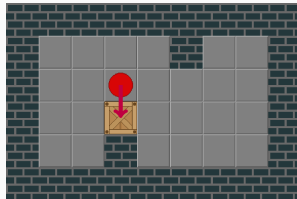
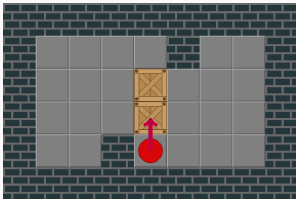
XSokoban 1 résolu

Problème **PSPACE-complet**

Règles



Déplacements autorisés



Déplacements interdits

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

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

Principe de résolution

Réduction de l'espace de recherche

Analyse statique

Analyse dynamique

Recherche dirigée par une heuristique

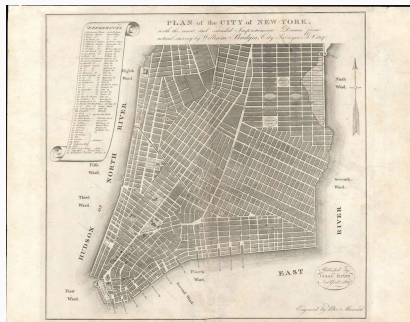
Résultats

Annexe

Lien avec le thème de l'année



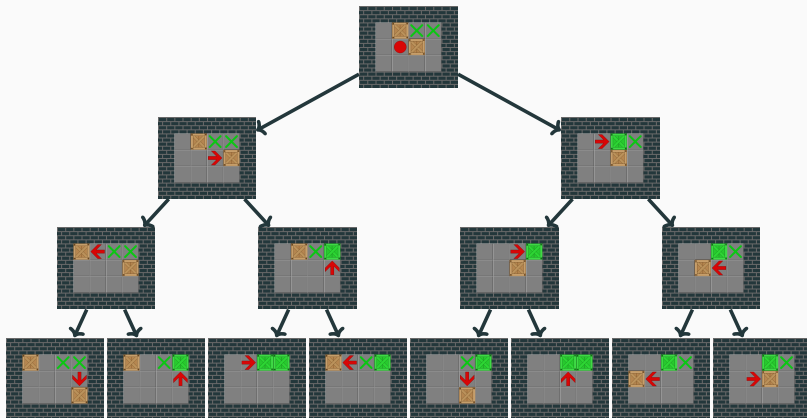
Source : *Indiana Jones et les Aventuriers de l'arche perdue* (scène de fin), Steven Spielberg, 1981
<https://pbs.twimg.com/media/EyjVShEVEAAQZjK.jpg>



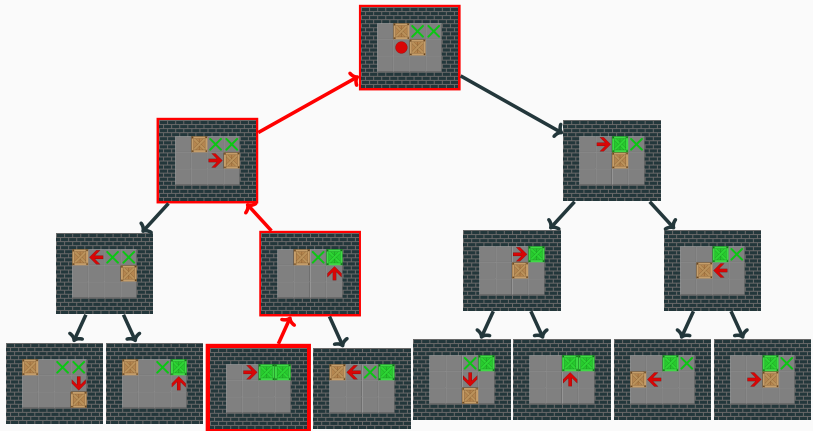
Source : <https://www.geographicus.com/mm5/graphics/00000001/L/NewYork-bridgesmaverick-1807.jpg>

Principe de résolution

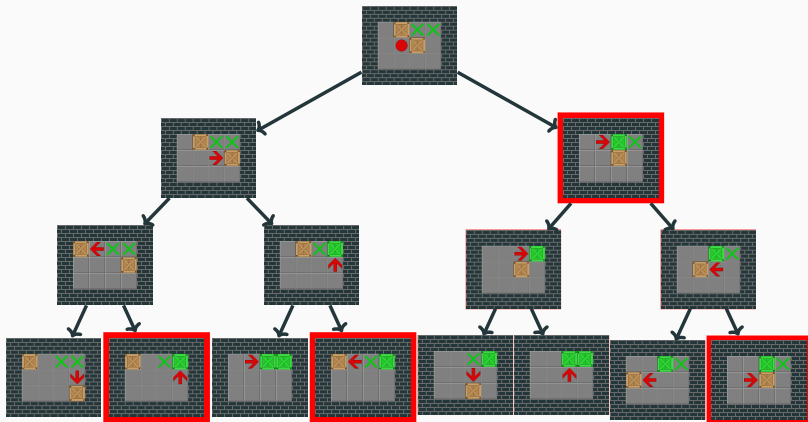
Arbre des états



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 :

$$T = \begin{matrix} & \begin{matrix} \text{caisse} & \text{joueur} & \text{case} \end{matrix} \\ \begin{pmatrix} 6357 & 31593 \\ -1378 & 42 \\ \vdots & \vdots \\ 93268 & -278 \end{pmatrix} & \begin{matrix} 0 \\ 1 \\ \vdots \\ wh - 1 \end{matrix} \end{matrix}$$

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

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

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

en $\mathcal{O}(n)$

- **Connaissant le hash de l'état parent** : $c_i \rightarrow c'_i, p \rightarrow p'$

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

en $\mathcal{O}(1)$

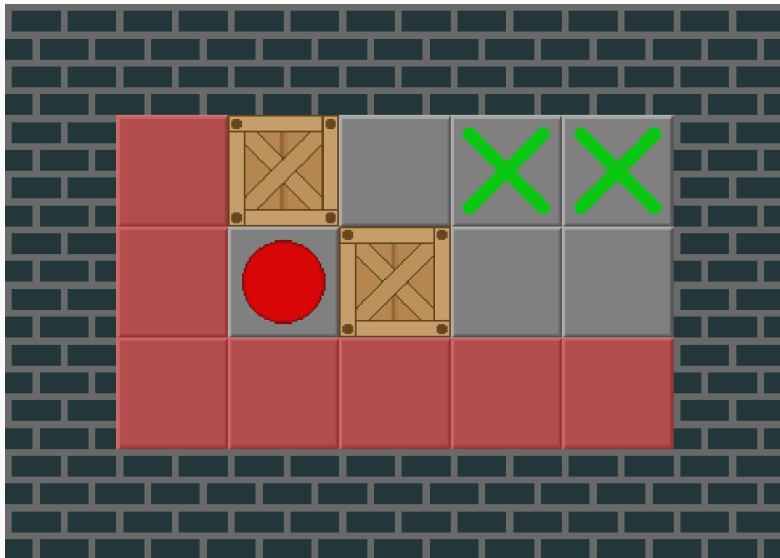
Réduction de l'espace de recherche

Réduction de l'espace de recherche

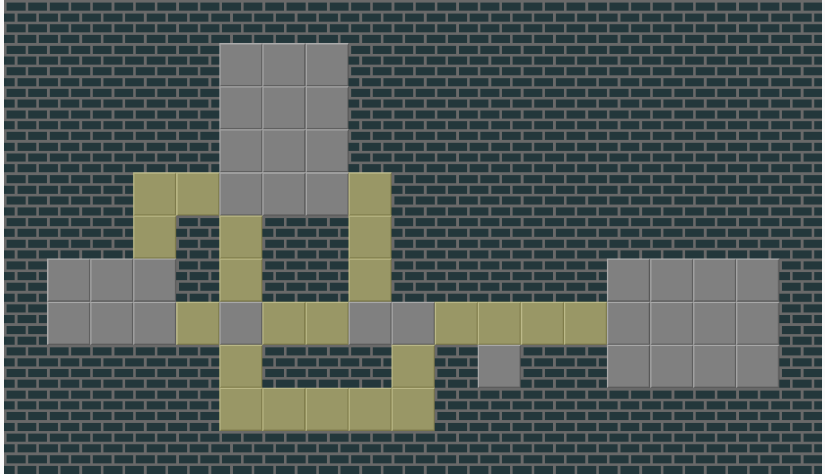


Analyse statique

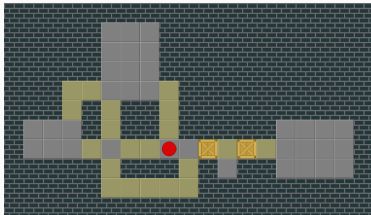
Détection des positions mortes (*dead tiles*)



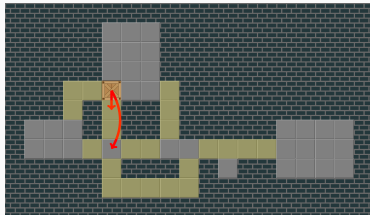
Détection de tunnels



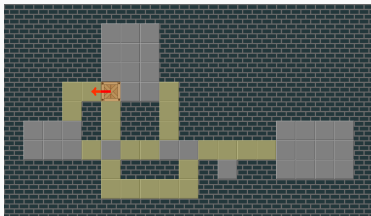
Détection de tunnels



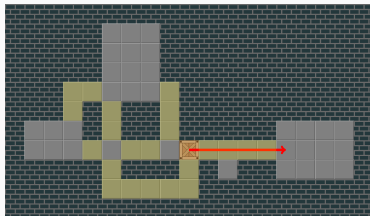
Au plus une caisse



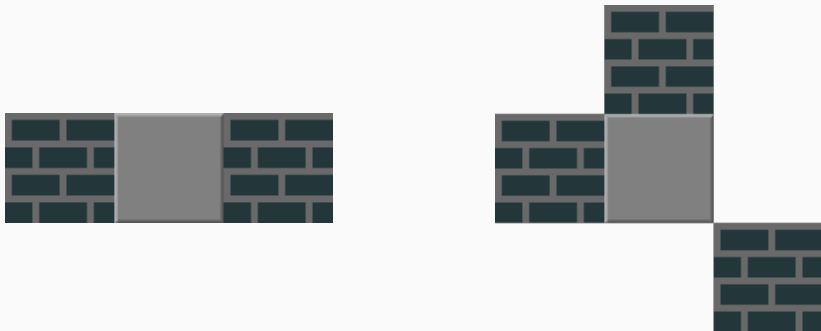
Deux états fils



Coin \Rightarrow un état fils

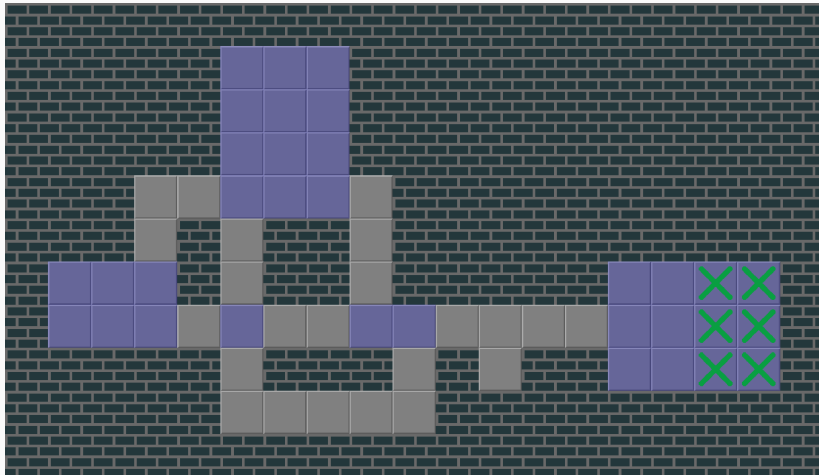


Tunnel *oneway*

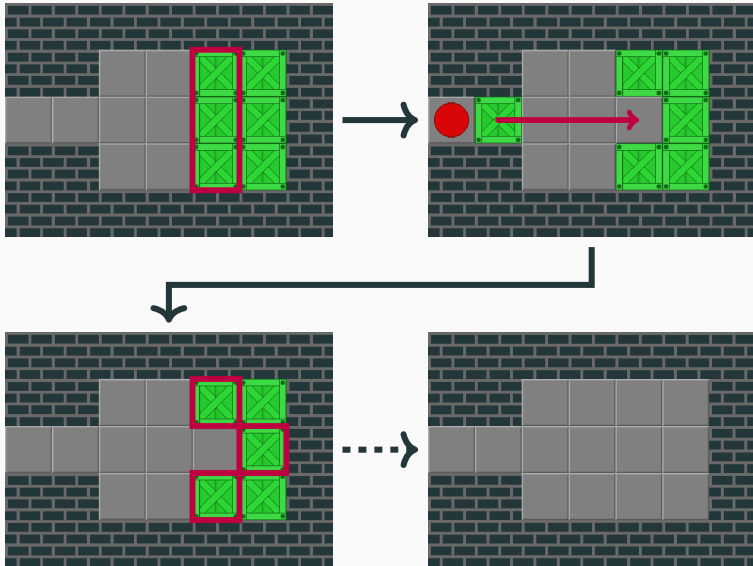


Composition d'un tunnel

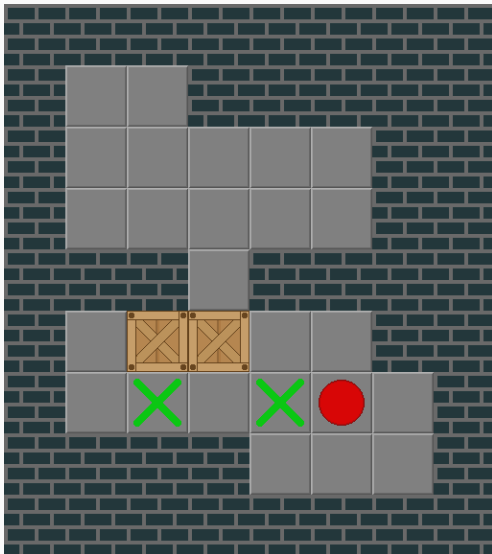
Salles et ordre de rangement (*packing order*)



Salles et ordre de rangement (*packing order*)



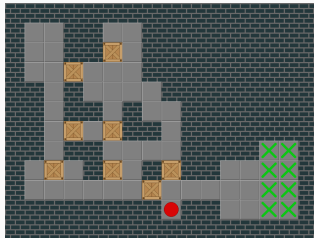
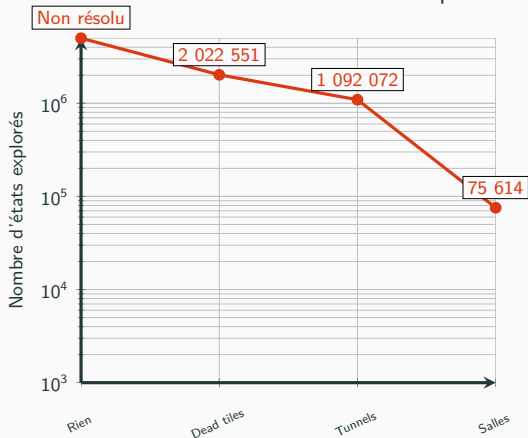
Salles et ordre de rangement (*packing order*)



Résultats intermédiaires

Niveaux résolus dans *XSokoban* : 1 / 90 (+ 0)

Niveau 24 du pack *Boxxle* :

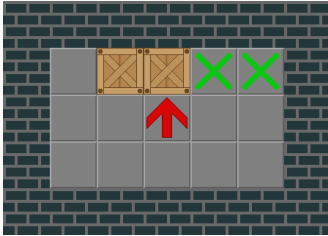


Réduction de l'espace de recherche

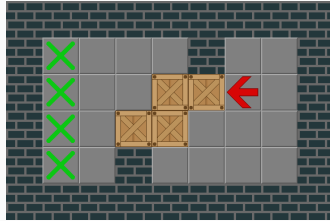


Analyse dynamique

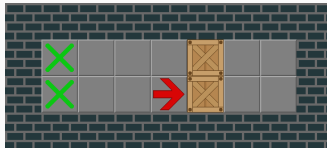
Détection d'impasses (*deadlocks*)



(a) *Freeze deadlock n°1*



(b) *Freeze deadlock n°2*

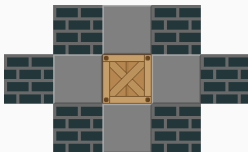


(c) *PI Corral deadlock*

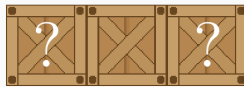
Détection de *freeze deadlocks*



(a) Règle n°1

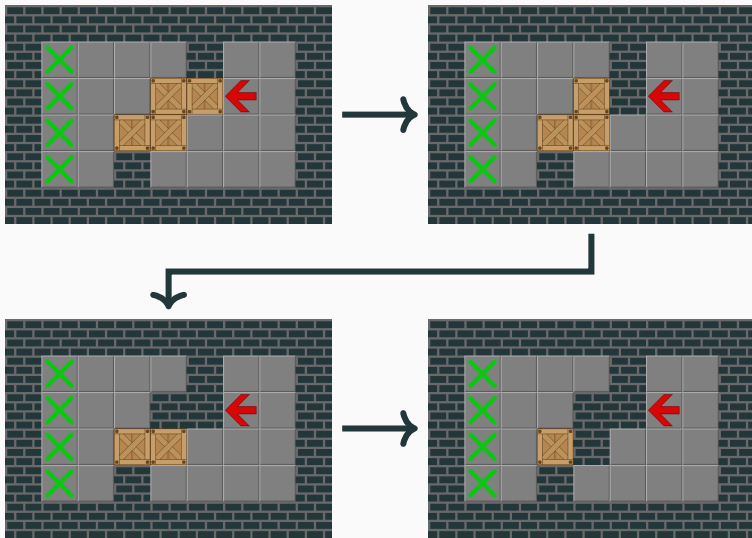


(b) Règle n°2



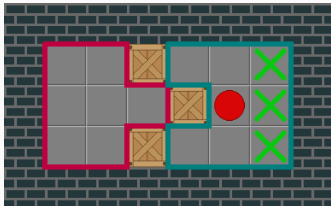
(c) Règle n°3

Détection de *freeze deadlocks*

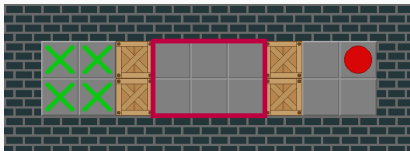


Gelée!

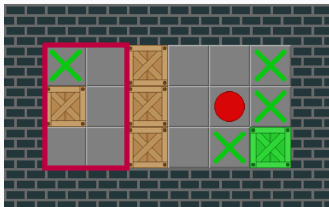
Détection de *PI Corral* deadlocks



(a) *Corral*



(b) *I Corral*



(c) *PI Corral*

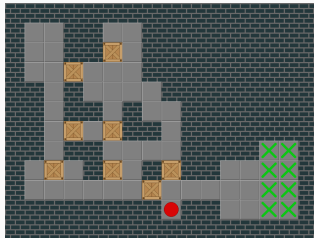
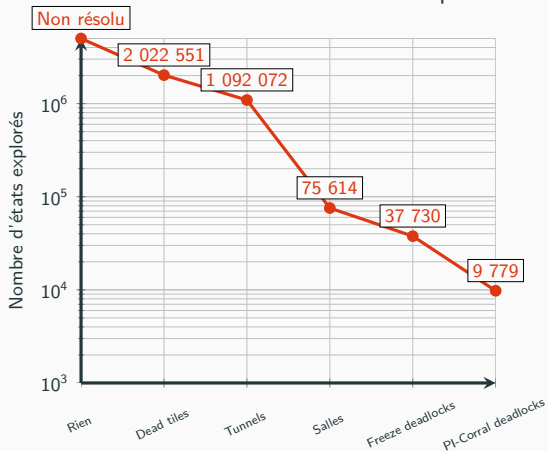
d

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

Résultats intermédiaires

Niveaux résolus dans *XSokoban* : 6 / 90 (+ 4)

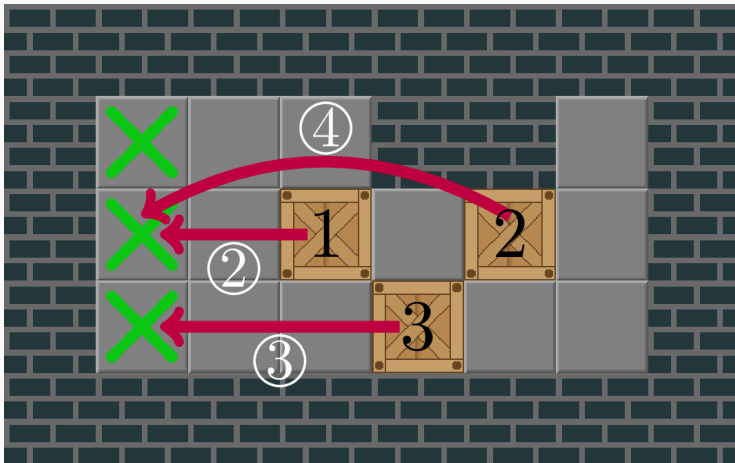
Niveau 24 du pack *Boxxle* :



Recherche dirigée par une heuristique

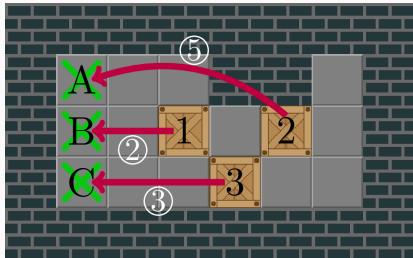
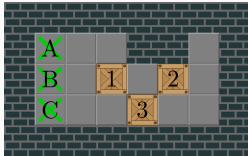


Heuristique simple (*Simple Lower Bound*)



$$2 + 4 + 3 = 9$$

Heuristique gloutonne (*Greedy Lower Bound*)



$$2 + 3 + 5 = 10$$

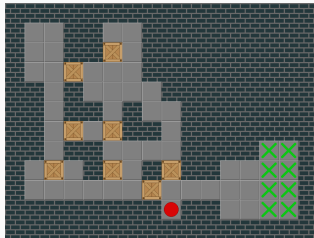
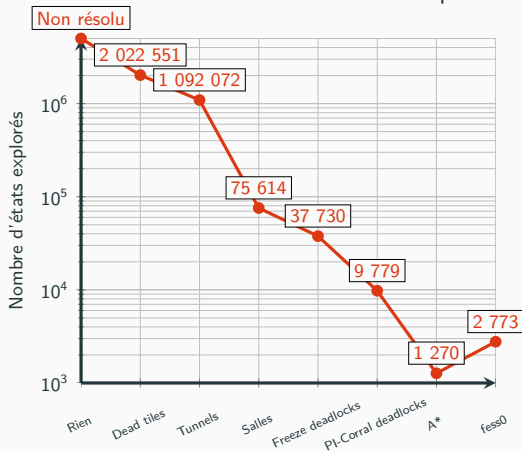
Caisse → Cible	Distance
1 → B	2
1 → A	3
1 → C	3
3 → C	3
2 → B	4
3 → B	4
2 → A	5
2 → C	5
3 → A	5

- FESS : algorithme utilisé par Festival, meilleur solveur.
- Ordre de priorité :
 - maximiser le nombre de caisses rangées.
 - minimiser le nombre de *corral*.
 - minimiser l'heuristique précédente.

Résultats intermédiaires

Niveaux résolus dans *XSokoban* : 15 / 90 (+ 5 + 4)

Niveau 24 du pack *Boxxle* :



Résultats



Nombre de niveaux résolus

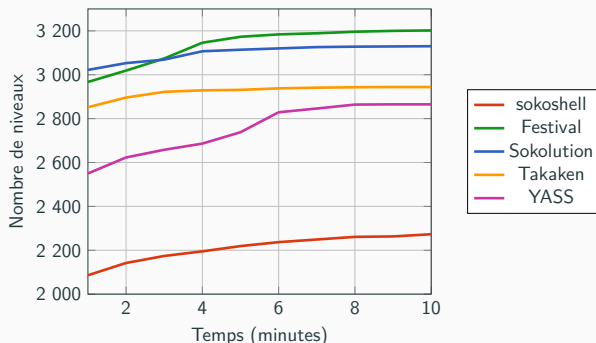
Limite de temps : 10 min. Limite de RAM : 32 Gio.

Ensemble de niveaux	XSokoban	<i>Large test suite</i>
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

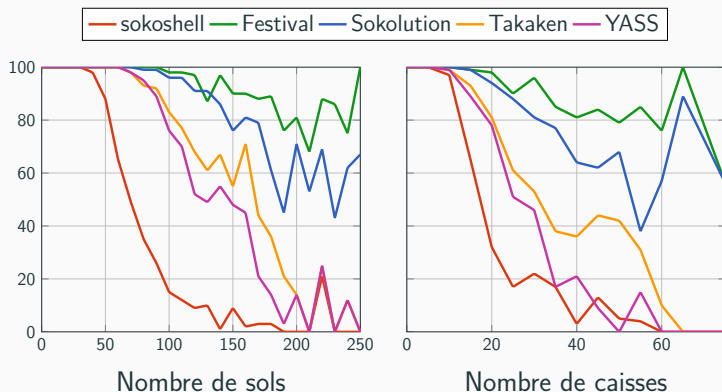
Temps moyen passé par niveaux

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

Nombre de niveaux résolus (cumulés) en fonction du temps



Pourcentage de niveaux résolus selon la composition des niveaux



Annexe



Tableau des complexités - Statique

c nombre de caisses, C nombre de cibles, w longueur et h largeur du niveau, t nombre de tunnels, r nombre de salles, N nombre d'états dans la liste des états à explorer.

Statique	
<i>Dead tiles</i>	$\mathcal{O}((wh)^2)$
Détection des tunnels	$\mathcal{O}((wh)^2)$
Propriété <i>oneway</i> des tunnels	$\mathcal{O}(twh)$
Détection des salles	$\mathcal{O}((wh)^2)$
<i>Packing order</i>	$\mathcal{O}(rcwh)$
Précalcul des distances cibles-caisses	$\mathcal{O}(wh(Cwh + C \log C))$

Tableau des complexités - Dynamique

c nombre de caisses, C nombre de cibles, w longueur et h largeur du niveau, t nombre de tunnels, r nombre de salles, N nombre d'états dans la liste des états à explorer.

Dynamique	
<i>Freeze deadlocks</i>	$\mathcal{O}(c)$
Détection des <i>corrals</i>	$\mathcal{O}(wh)$
PI- <i>corral deadlocks</i>	Exponentielle
Table de <i>deadlocks</i>	$\mathcal{O}(1)$
Recherche des états enfants	$\mathcal{O}(crwh)$
Ajout des états enfants (A^*)	$\mathcal{O}((wh)^2 + \log N)$
Ajout des états enfants (fess0)	$\mathcal{O}(c + (wh)^2 + \log N)$

Table de *deadlocks*

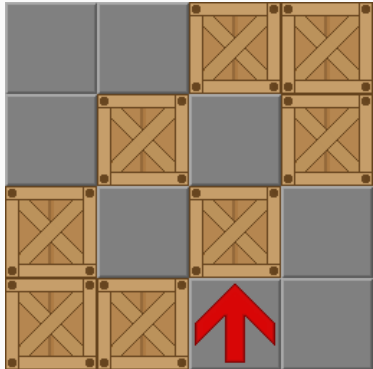
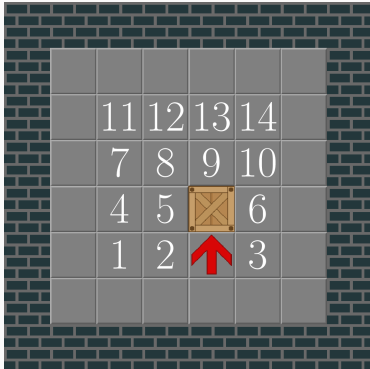
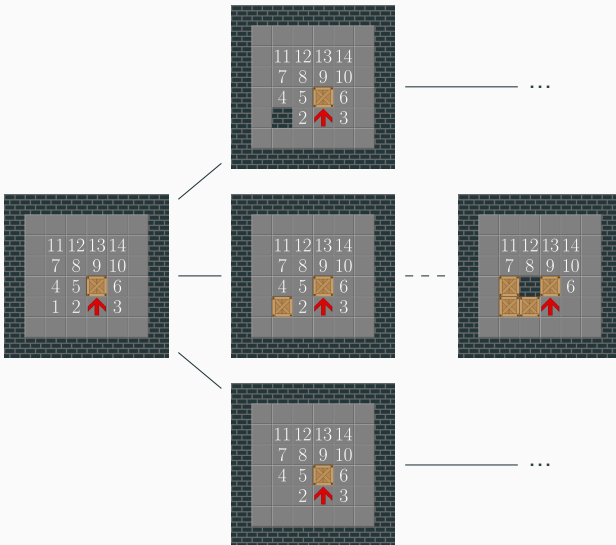
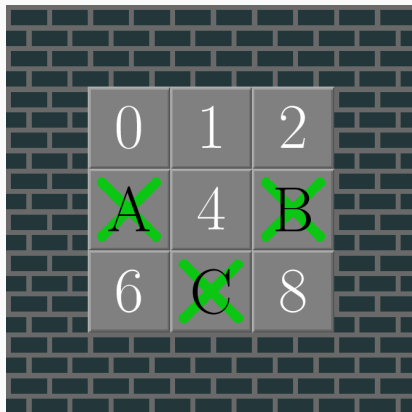


Table de *deadlocks*



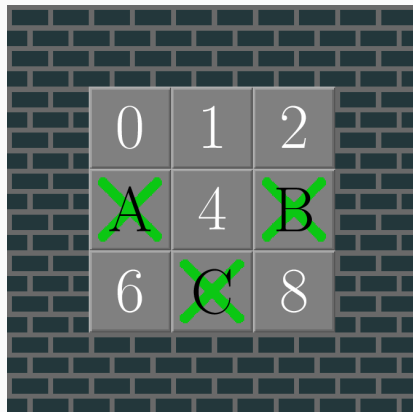
Précalcul des distances caisses-cibles

Case	Distances		
	A	B	C
0	1	3	3
1	2	2	2
2	3	1	3
3	0	2	2
4	1	1	1
5	2	0	2
6	1	3	1
7	2	2	0
8	3	1	1

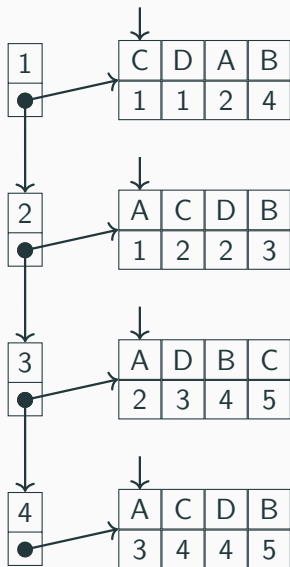


Précalcul des distances caisses-cibles

Case	Distances triées		
0	A : 1	B : 3	C : 3
1	A : 2	B : 2	C : 2
2	B : 1	A : 3	C : 3
3	A : 0	B : 2	C : 2
4	A : 1	B : 1	C : 1
5	B : 0	A : 2	C : 2
6	A : 1	C : 1	B : 3
7	C : 0	A : 2	B : 2
8	B : 1	C : 1	A : 3

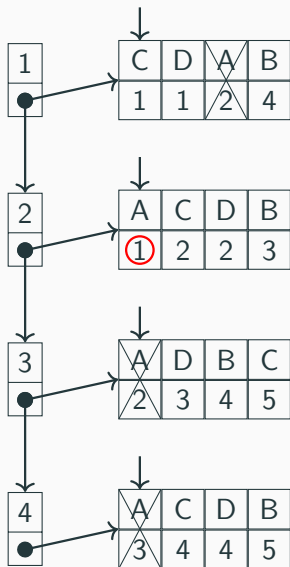


Greedy Lower Bound en $\mathcal{O}(n^2)$



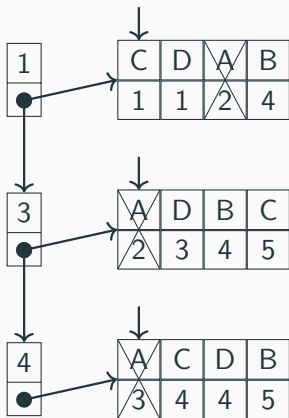
$h =$

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



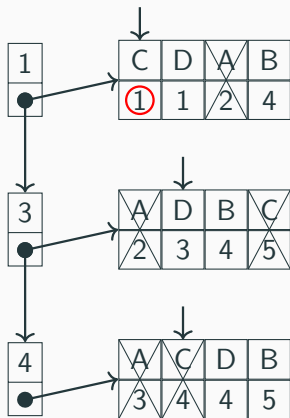
$$h = 1 +$$

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



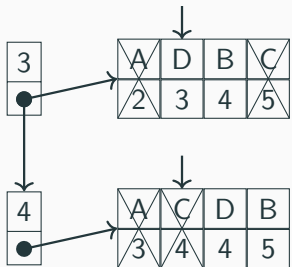
$$h = 1 +$$

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



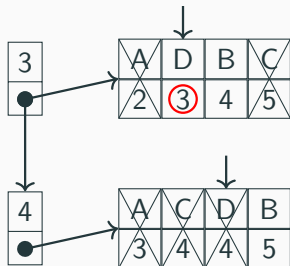
$$h = 1 + 1 +$$

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



$$h = 1 + 1 +$$

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



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

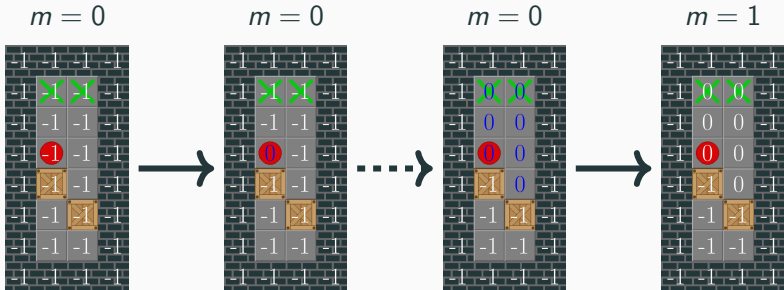
Greedy Lower Bound en $\mathcal{O}(n^2)$



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

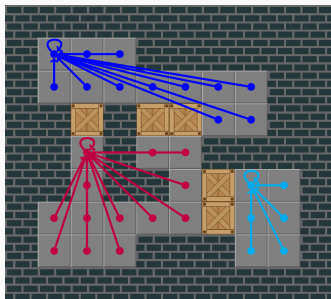
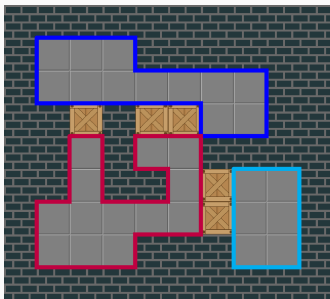
Parcours de graphes : démarquer tous les nœuds en $\mathcal{O}(1)$

nœud marqué ssi valeur = m

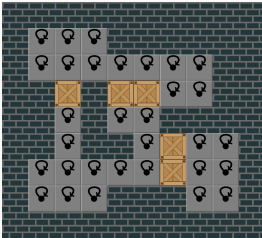


Calcul des *corrals* en $\mathcal{O}(wh)$

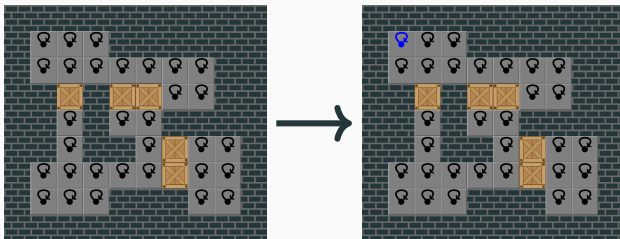
Utilisation de *Union-Find* : partition de $\llbracket 0; wh - 1 \rrbracket$.



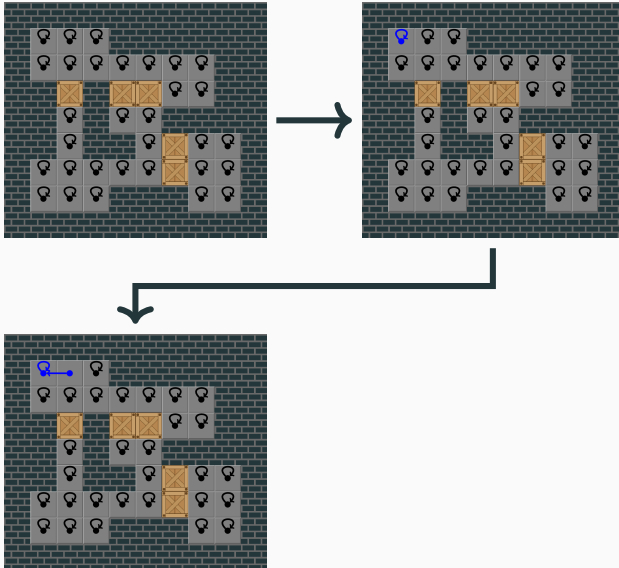
Calcul des *corrals* en $\mathcal{O}(wh)$



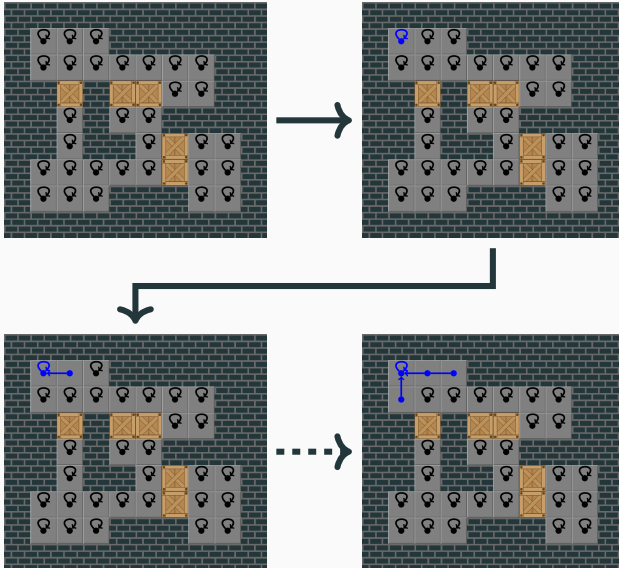
Calcul des *corrals* en $\mathcal{O}(wh)$



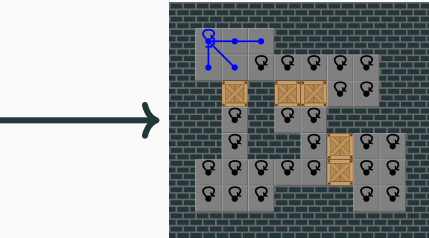
Calcul des *corrals* en $\mathcal{O}(wh)$



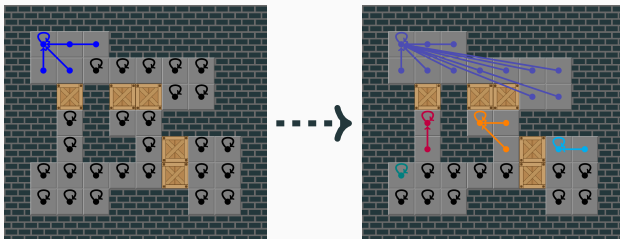
Calcul des *corrals* en $\mathcal{O}(wh)$



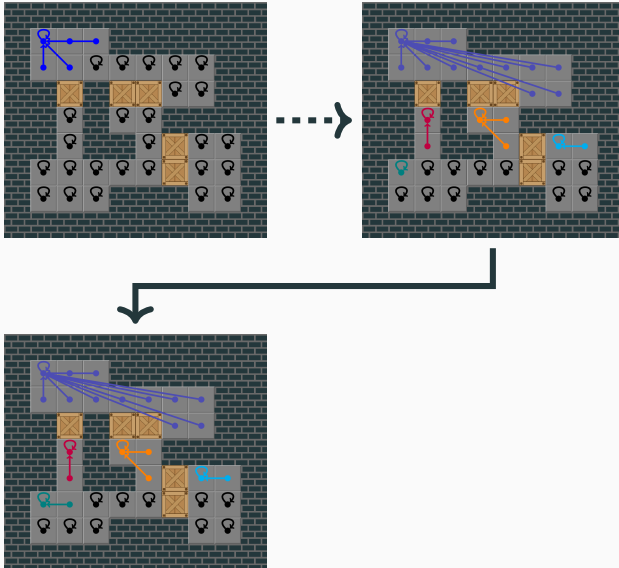
Calcul des *corrals* en $\mathcal{O}(wh)$



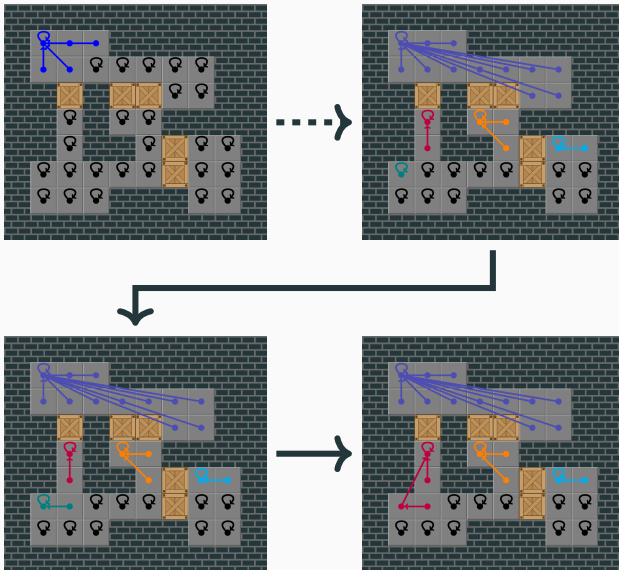
Calcul des *corrals* en $\mathcal{O}(wh)$



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



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



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

