Graphs for hierarchical RL macro planning: A Sokoban testbed

Mathieu Orhan and Bastien Déchamps

4 December 2019

Summary

1. Introduction
   - Sokoban challenges
   - Objectives of the project
2. Approach
   - Graph embedding
   - Node vs. Graph centered
   - Reinforcement learning framework
3. Experiments & results
   - Simple levels
   - Harder levels
   - Generalization capacities
4. Conclusion & Work directions

Sokoban, a very challenging environment

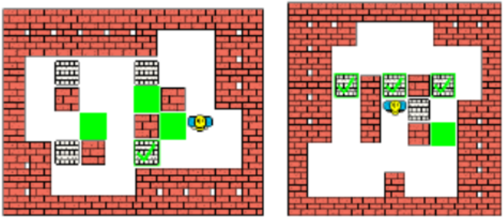


Figure – Median human solving time is 43 min. (left), 49 min. (right) [2]

Sokoban is **hard** !

- NP-Hard [1]
- Require far-sighted planning abilities
- Many irreversible moves that lose the game
- Often, parts of the solution are unique

Sokoban, an interesting testbed

- Excellent environment to test hierarchical planning abilities
- Great variety of level difficulties, ranging from trivial to impossible instance for solvers
- Levels can be procedurally generated [4]
- Large databases already exists [6]

Are graphs well suited to model this problem ?

Objectives

Are graphs well suited to model this problem ?

Main objective

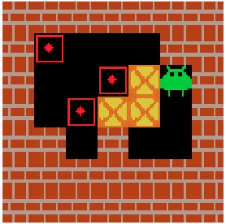
Experiment with **graph modeling** and **graph neural networks** using the **Sokoban** environment

Additional objectives :

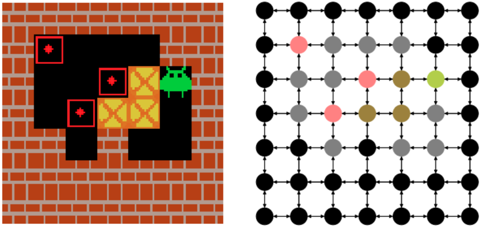
- Test generalization capabilities
- Experiment with curriculum learning

Summary

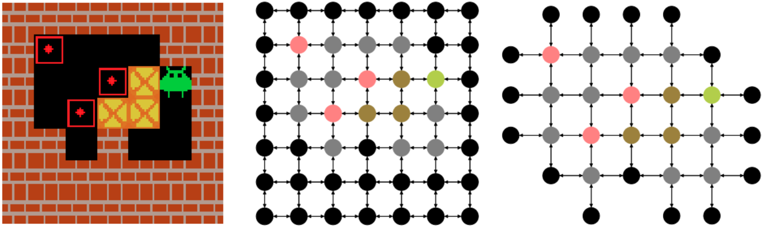
Embedding : Represent a level by a graph



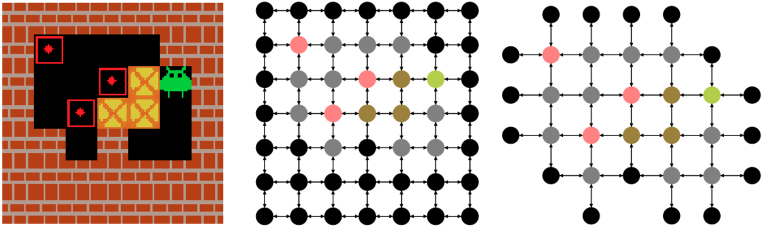
Embedding : Represent a level by a graph



Embedding : Represent a level by a graph

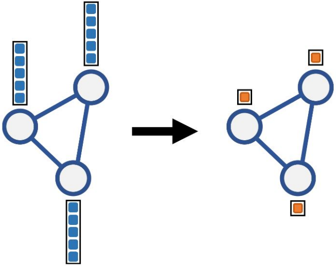


Embedding : Represent a level by a graph



- **Node features** : node nature (player, box, wall, ...) + normalized position
- **Edge features** : direction (up, down, right or left)

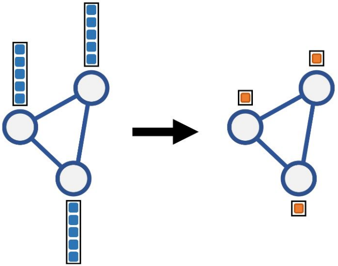
Node-centered :



- State-action value is predicted for each node
- Non-legal moves are masked
- **Node regression** task

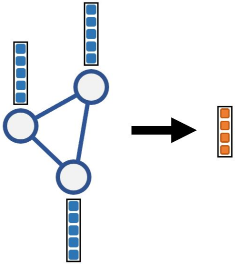
Different approaches

Node-centered :



- State-action value is predicted for each node
- Non-legal moves are masked
- **Node regression** task

Graph-centered :



- State-action value is predicted for each direction
- **Graph regression** task

Message Passing Scheme

Message passing : **EdgeConv** [5]

$$x_i' = \square_{j \in \mathcal{N}(i)} h_\Theta \left([x_i, x_j - x_i] \right)$$

- $\square$ aggregation function (max, $\sum$, ...)
- h_Θ 2-layers MLP with ReLU activations

Message passing : **EdgeConv** [5]

$$x_i' = \mathop{\square}_{j \in \mathcal{N}(i)} h_\Theta \left([x_i, x_j - x_i] \right)$$

- $\square$ aggregation function (max, $\sum$, ...)
- h_Θ 2-layers MLP with ReLU activations

Graph-centered $\rightarrow$ **Max Pooling** :

$$\forall j \in \{1, ..., H\}, \quad r_j = \max_{i \in \{1,...,N\}} \left(x_i^D \right)_j$$

where x_i^D feature vector of node i of the last EdgeConv layer.

Model

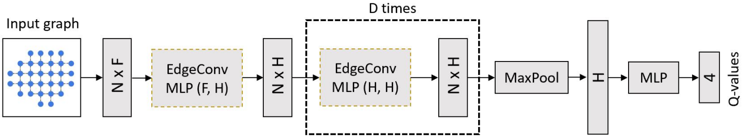


Figure – Model used for the graph-centered approach.

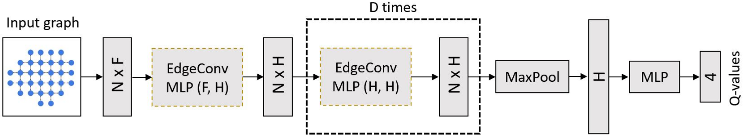


Figure – Model used for the graph-centered approach.

- **Input** : batch of graph state (G_i)
- **Output** : batch of Q-values $(Q(G_i, a)) \in \mathbb{R}^{B\times 4}$
- D EdgeConv layers with their own parameters but could be shared.

Algorithm 1 DQN with Target Network and Experience Replay

Require: policy net $\mathcal{M}_P$, target net $\mathcal{M}_T$, replay buffer $\mathcal{B}$, data $\mathcal{D}$

  for $s_0 \in \mathcal{D}$ **do**

    while episode is not terminated **do**

      Sample action a_t using ϵ-greedy exploration

      Take a_t, observe (s_{t+1}, r_t)

      Push (s_t, a_t, s_{t+1}, r_t) into $\mathcal{B}$

      if $\mathcal{B}$ is large enough **then**

        Sample a batch $(S_t, A_t, S_{t+1}, R_t) \sim \mathcal{B}$

        $Q_p = \mathcal{M}_P(S_t, A_t)$

        $Q_e = R_t + \gamma \max_A \mathcal{M}_T(S_{t+1}, A)$

        Backpropagate $\mathcal{L}(Q_p, Q_e)$

    Update $\mathcal{M}_T$ at regular intervals using $\mathcal{M}_P$ weights

Replay buffer

Advantages :

- Decorrelate the samples
- Stabilize the training
- Allow to reuse samples
- Allow for the rarest experiences to be stored

We experimented variants to sample **non-uniformly** trajectories

Exploration strategy

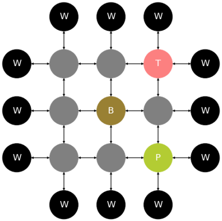
We used ϵ-**greedy** strategy to explore.

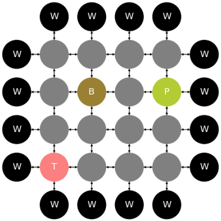
- Very common approach and baseline
- Linear annealing, from $\epsilon = 1.0$ to $\epsilon = 0.1$, following [3].
- Improved by ignoring some moves

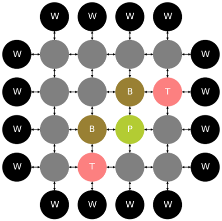
1. Introduction
   - Sokoban challenges
   - Objectives of the project
2. Approach
   - Graph embedding
   - Node vs. Graph centered
   - Reinforcement learning framework
3. Experiments & results
   - Simple levels
   - Harder levels
   - Generalization capacities
4. Conclusion & Work directions

Very simple levels

Dummy levels on 5×5, 6×6 grids with 1 or 2 boxes





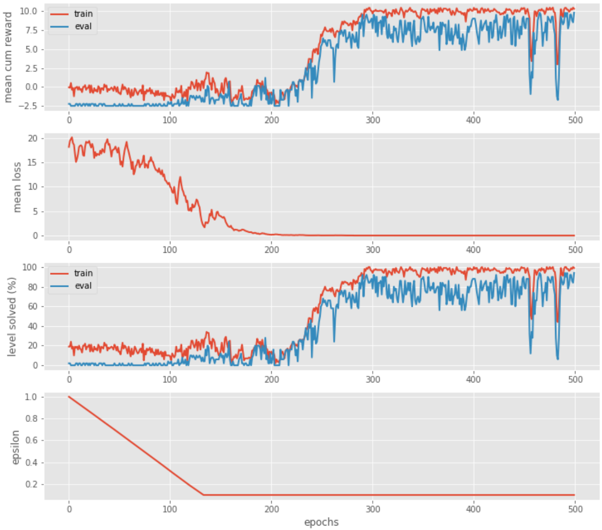


5×5, 1 box

- 5×5 with 1 box
- $H = 32$
- $D = 2$
- 100 train levels
- 50 test levels

5×5, 1 box

- 5×5 with 1 box
- $H = 32$
- $D = 3$
- 100 train levels
- 50 test levels



5×5, 1 box

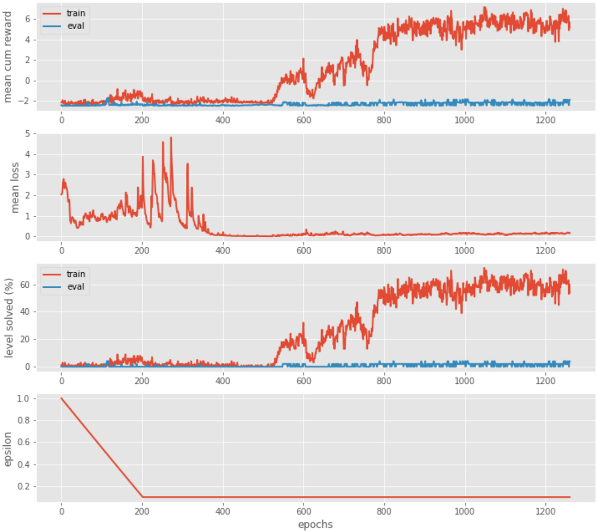
- 5×5 with 1 box
- $H = 32$
- $D = 4$
- 100 train levels
- 50 test levels

6×6, 1 box

- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels

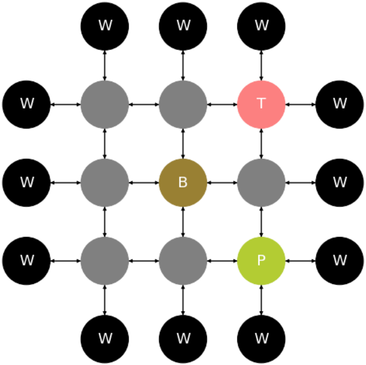
6×6, 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



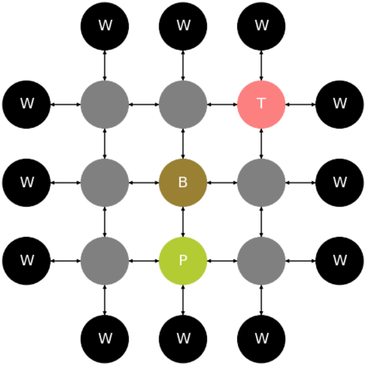
Example : 5×5

- 5×5 with 1 box
- $H = 32$
- $D = 2$
- 100 train levels
- 50 test levels



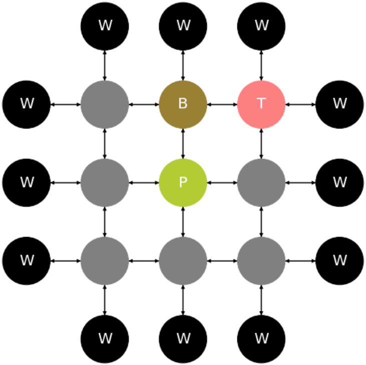
Example : 5×5

- 5×5 with 1 box
- $H = 32$
- $D = 2$
- 100 train levels
- 50 test levels



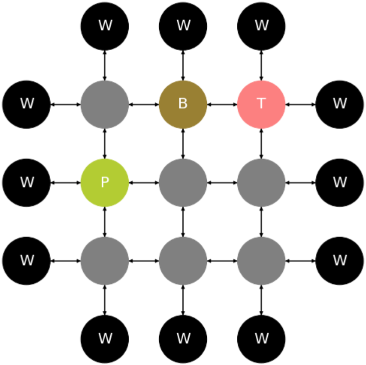
Example : 5×5

- 5×5 with 1 box
- $H = 32$
- $D = 2$
- 100 train levels
- 50 test levels



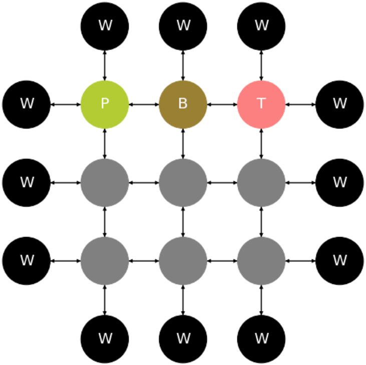
Example : 5×5

- 5×5 with 1 box
- $H = 32$
- $D = 2$
- 100 train levels
- 50 test levels



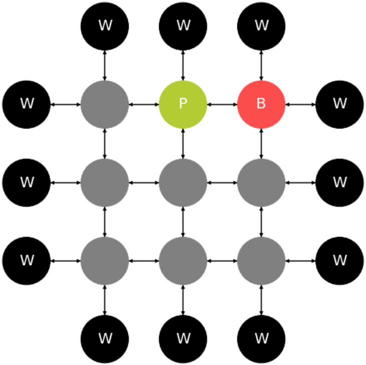
Example : 5×5

- 5×5 with 1 box
- $H = 32$
- $D = 2$
- 100 train levels
- 50 test levels



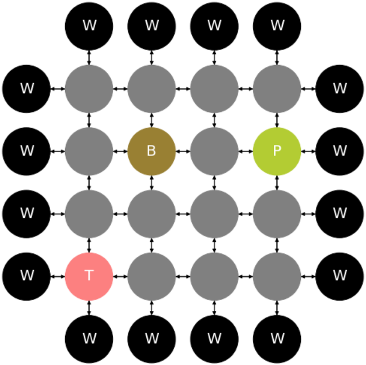
Example : 5×5

- 5×5 with 1 box
- $H = 32$
- $D = 2$
- 100 train levels
- 50 test levels

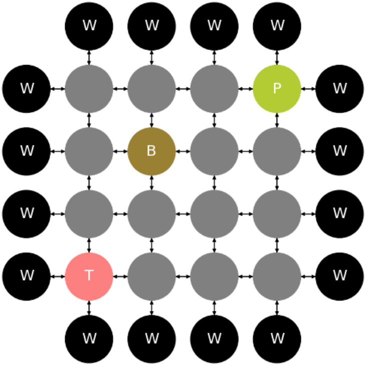


Example : 6×6 with 1 box

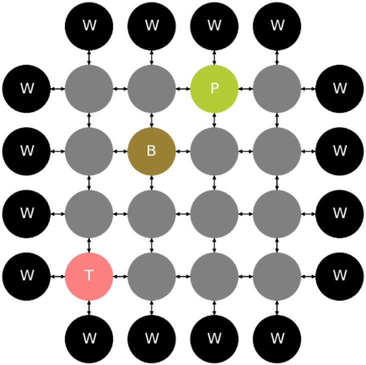
- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



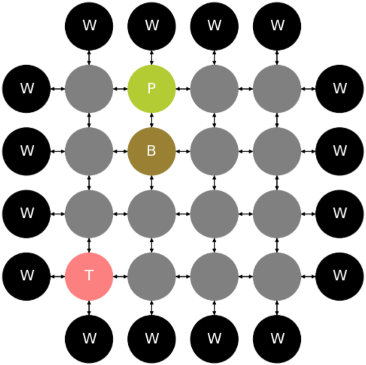
- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels

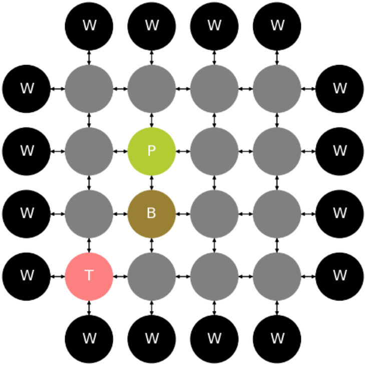


- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



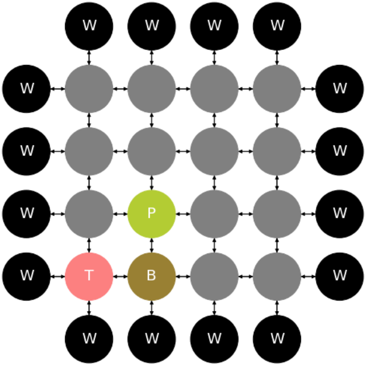
Example : 6×6 with 1 box

- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



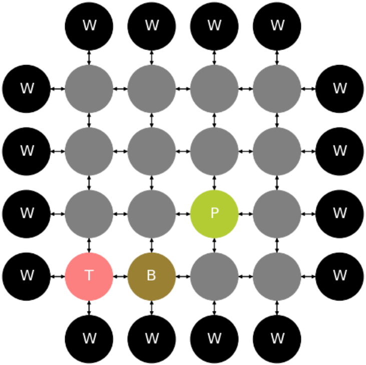
Example : 6×6 with 1 box

- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



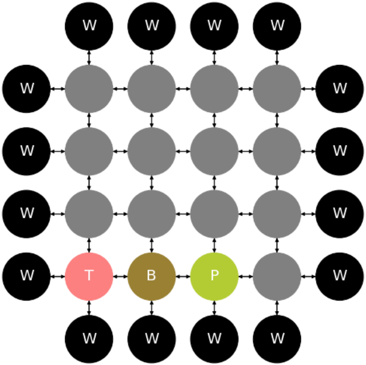
Example : 6×6 with 1 box

- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



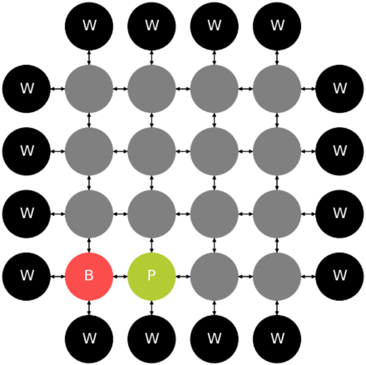
Example : 6×6 with 1 box

- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



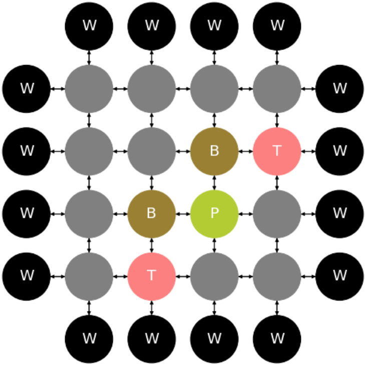
Example : 6×6 with 1 box

- 6×6 with 1 box
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



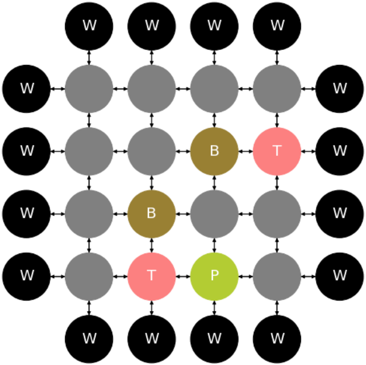
Example : 6×6 with 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



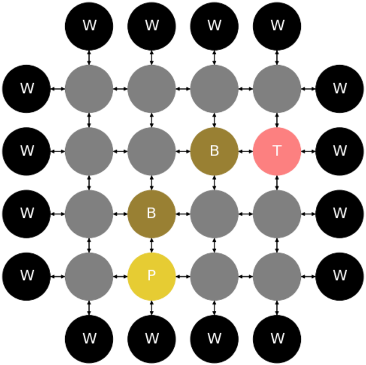
Example : 6×6 with 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



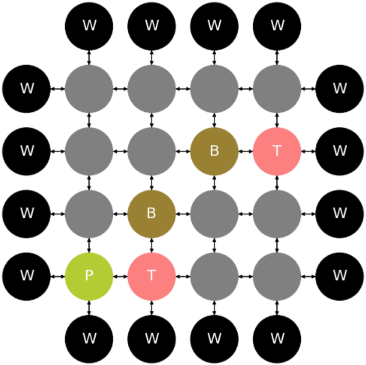
Example : 6×6 with 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



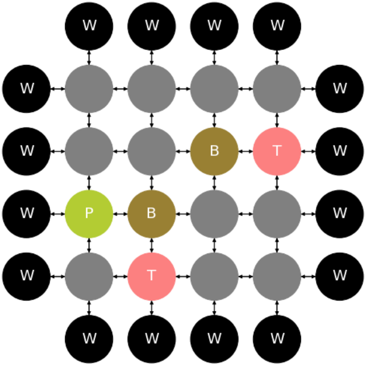
Example : 6×6 with 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



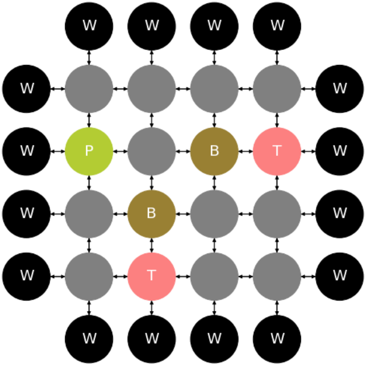
Example : 6×6 with 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



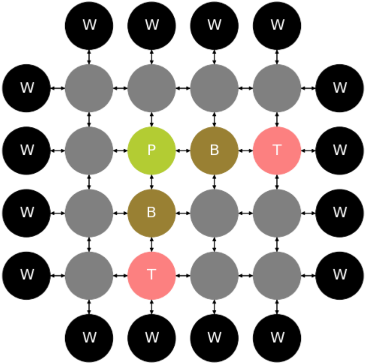
Example : 6×6 with 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



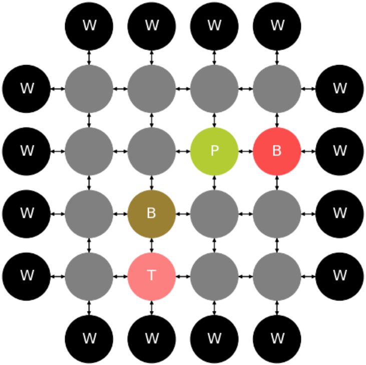
Example : 6×6 with 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



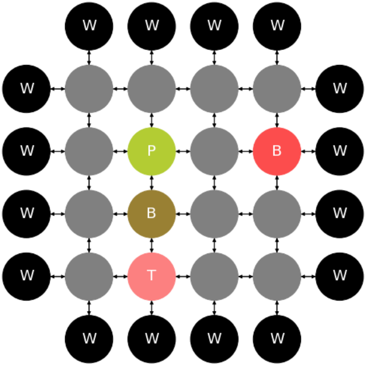
Example : 6×6 with 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels

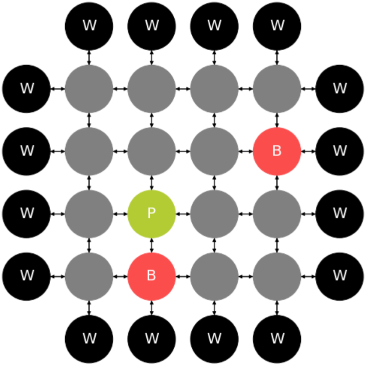


Example : 6×6 with 2 boxes

- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



- 6×6 with 2 boxes
- $H = 256$
- $D = 5$
- 100 train levels
- 50 test levels



... but still very easy for human players ...

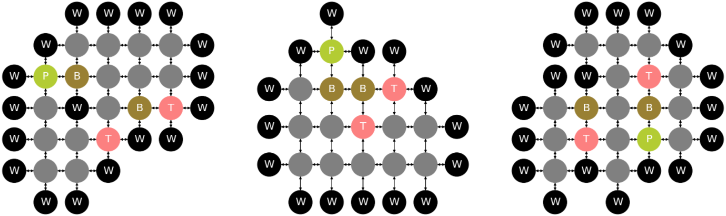
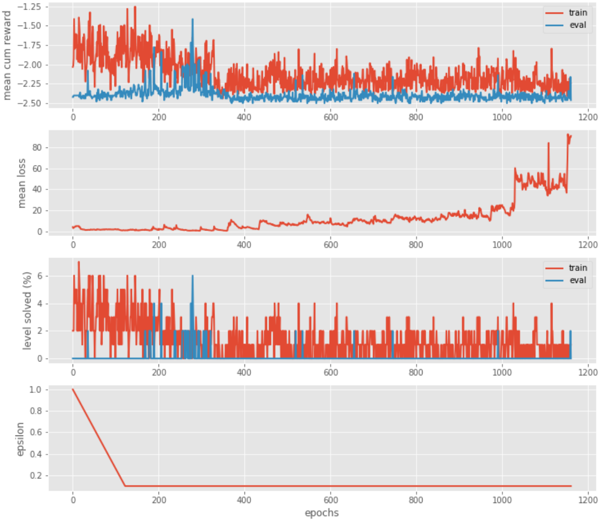


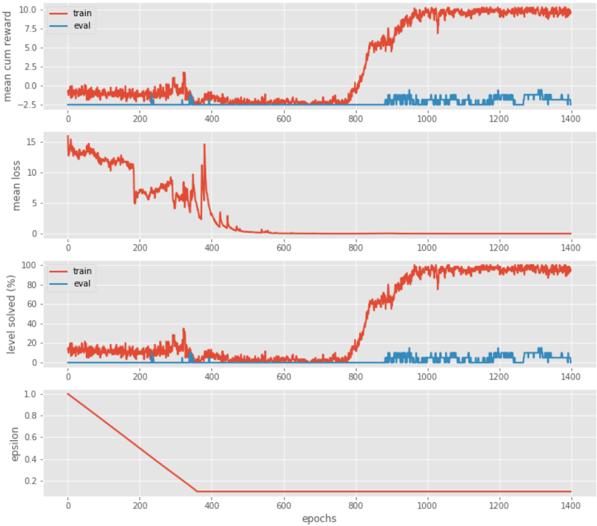
Figure – "Easy" levels from `gym-sokoban`

Harder levels

- "Easy" levels with 2 boxes
- $H = 256$
- $D = 6$
- 100 train levels
- 50 test levels



Experiment 1 : Train same network with different types of dummy levels.



Training set :

- 20 levels 5×5
- 20 levels 6×6
- 20 levels 7×7

Experiment 2 : Apply trained models to harder levels.

		Tested on			
		5×5 (1)	6×6 (1)	6×6 (2)	easy
Trained on	5×5 (1)	–	0.02	0	0
	6×6 (1)	0.26	–	0.01	0.01
	6×6 (2)	0.14	0.04	–	0.01
	mixed	**0.46**	**0.15**	0	0.01

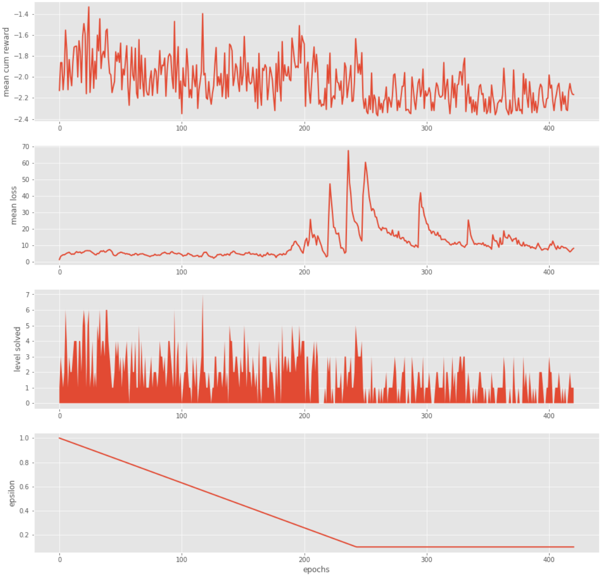
Table – Fraction of level solved for generalization experiences.

→ **existing but poor generalization capacities...**

Improvement attempts

Prioritized Replay Memory

- "Easy" levels with 2 boxes
- $H = 256$
- $D = 6$
- 100 train levels
- 100 test levels
- $\alpha = 0.6$
- $\beta = 0.1$



Summary

To summarize

- We designed from scratch a first algorithm to solve the Sokoban
- A lot of experiments and tuning were required to have results
- We successfully solve dummy levels, but fail for real Sokoban levels

Work directions

- A different modeling of the input graph...

Work directions

- A different modeling of the input graph...
- Put more priors into the graph model

Work directions

- A different modeling of the input graph...
- Put more priors into the graph model
- Test better suited reinforcement learning algorithms

Work directions

- A different modeling of the input graph...
- Put more priors into the graph model
- Test better suited reinforcement learning algorithms
- Curriculum learning

D. Dor and U. Zwick.
Sokoban and other motion planning problems.
Computational Geometry, 13(4) :215–228, 1999.

P. Jarusek and R. Pelánek.
Difficulty rating of sokoban puzzle.
volume 222, pages 140–150, 01 2010.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller.
Playing atari with deep reinforcement learning, 2013.

M.-P. B. Schrader.
gym-sokoban.
https://github.com/mpSchrader/gym-sokoban, 2018.

Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon.
Dynamic graph cnn for learning on point clouds.
ACM Transactions on Graphics, 38(5) :1–12, Oct 2019.

T. Weber, S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, R. Pascanu, P. Battaglia, D. Hassabis, D. Silver, and D. Wierstra.
Imagination-augmented agents for deep reinforcement learning, 2017.