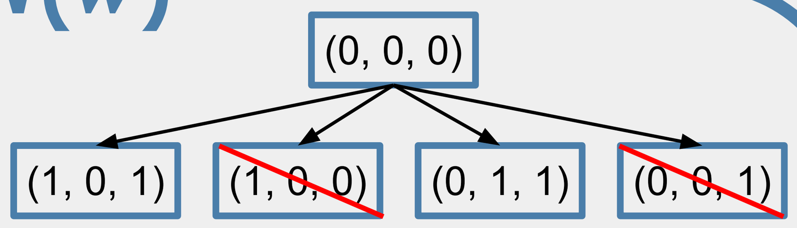
Hierarchical Width-Based Planning and Learning

Miquel Junyent, Vicenç Gómez, Anders Jonsson

Universitat Pompeu Fabra, Barcelona, Spain

Background: IW(*w*)



Example for w=1 with 3 binary features and 4 actions

- **Breadth-first search** (BrFS)
- States factored into **features** $\phi(s)$
- **Prunes states** that are **not novel**
- A state is **novel** if it has a **feature tuple of size *w*** that is **new in the search**
- **Complexity exponential in *w***, but independent of $|\mathcal{S}|$
- Most classical planning benchmarks present a **low width** with **single atom goals**
- In practice:

# Domains	# Inst.	Inst. IW(1)	Inst. IW(2)
37	37,921	37.0%	88.2%

  - Problems have **higher width** (no single-atom goal tasks)
  - IW(*w*) is mostly used with **w=1** due to computational constraints

Hierarchical IW (HIW)

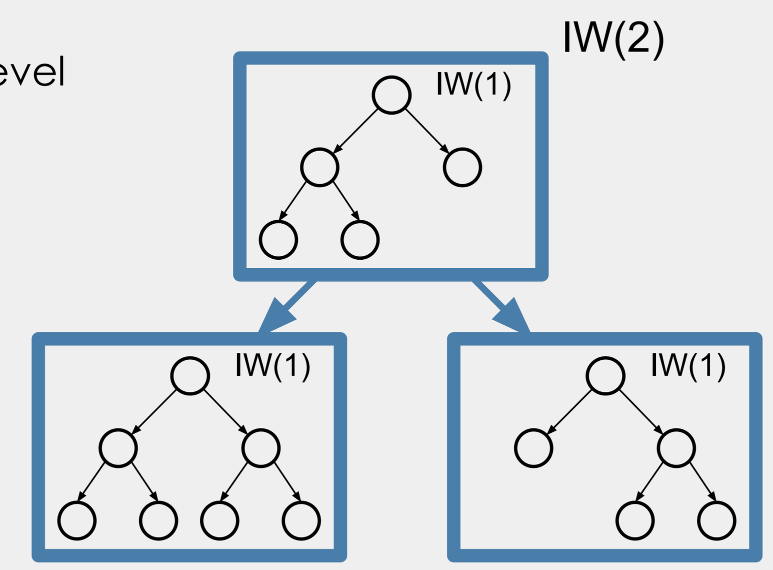
- Blind search methods require two components:
  - **Successor function**: given a state and an action, returns a successor state (e.g., simulator)
  - **Stopping condition**: tells us when to stop the search (e.g., goal is met)

```
Q = Queue(root)
While Q not empty:
  s = PopFirst(Q)
  For each action a:
    x = GenerateSuccessor(s,a)
    Append(Q, x)
    If ShouldStop(x):
      return
```

Example BrFS

- Our **hierarchical approach to blind search**:
  - Considers two sets of features F_h and F_ℓ
  - Modifies:
    - **High-level successor function**: each call triggers a low-level search
    - **Low-level stopping condition**: stops the low-level search when a state s that maps to a different $\phi_h(s)$ is found
  - We can pause and **resume low level searches**
  - Allows for **many levels of abstraction**
  - Accepts **different planners** at each level

Hierarchical IW



- **IW(w) at the different levels**
- For instance:
  - IW(2) at high-level
  - IW(1) at low-level

  } HIW(2, 1)
- In general: $\mathrm{HIW}(w_h, w_\ell)$
- HIW can solve problems of width $w_h + w_\ell$

Features:
- 1-D position **(low level)**
- Having the key **(high level)**

This problem has **width 2**, but it can be solved by **HIW(1,1)**

Complexity Results

- Let $N(n,d,w)$ denote the **maximum amount of novel nodes** that $\mathrm{IW}(w)$ generates in a problem with $n = |F|$ features of domain size $d = |D|$
- Two basic premises:
  - A feature **has one value** at a time
  - A feature value **appears in several tuples** simultaneously
- Recursive formula:

$$N(n,d,0) = 1, \longrightarrow \text{Only the initial state is novel}$$

$$N(n,d,n) = d^n, \longrightarrow \text{All states are novel}$$

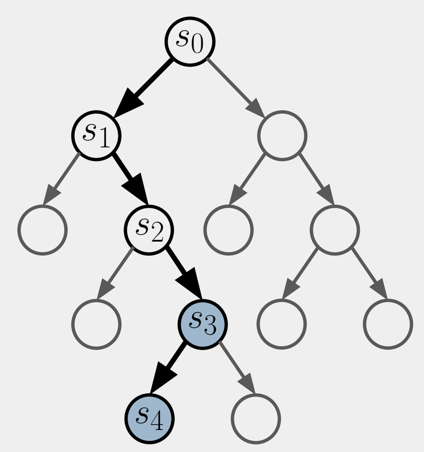
$$N(n,d,w) = \underbrace{(d-1)N(n-1,d,w-1)}_{\text{States novel due to one feature } f} + \underbrace{N(n-1,d,w)}_{\text{States novel due to other features different than } f}.$$

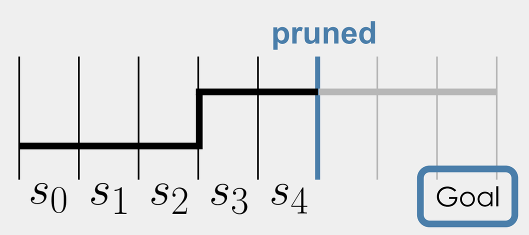
- General formula, for $0 \leq w < n$:

$$N(n,d,w) = \sum_{k=0}^{w} \left[\binom{n-1-k}{w-k} d^k (d-1)^{w-k} \right]$$

HIW in Classical Planning

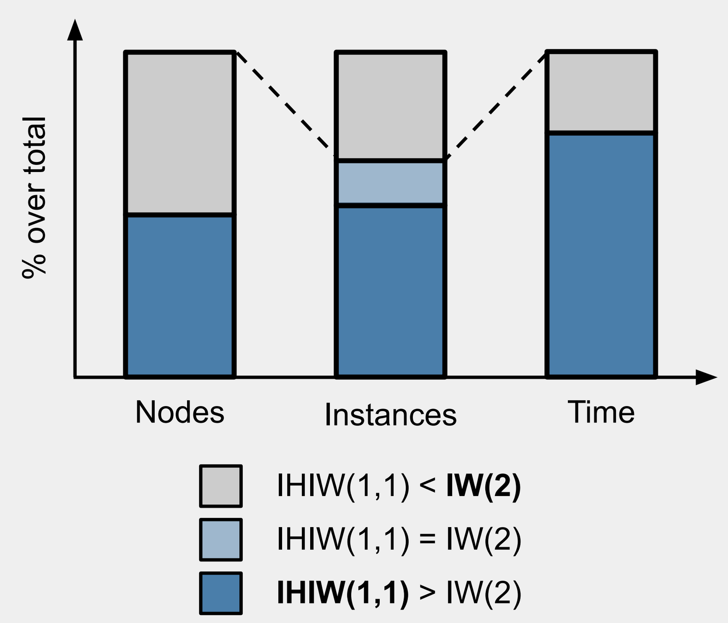
- **Hypothesis**: features that **only change once** in a branch before being pruned are good candidates
- **Incremental HIW(1,1)**:
  - Iteratively **run HIW(1,1)**
  - **Add one feature** to F_h at each iteration
  - **Discover new features** when necessary
  - **Reuse** the search **tree** among iterations





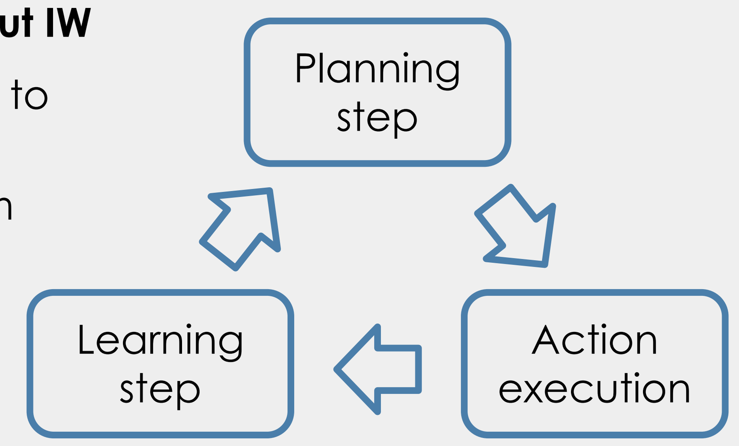
Results in Classical Planning

- Single goal instances
- Budget of 10K nodes
- We report:
  - Solved instances (%)
  - Avg. nodes (solved)
  - Avg. time in s (solved)
- **IHIW > IW(1) in 31/36** domains
- Compared to IW(2):
  - **>= in 24/36** domains
  - Uses **less nodes in 12/24**
  - Solves it **faster in 18/24**



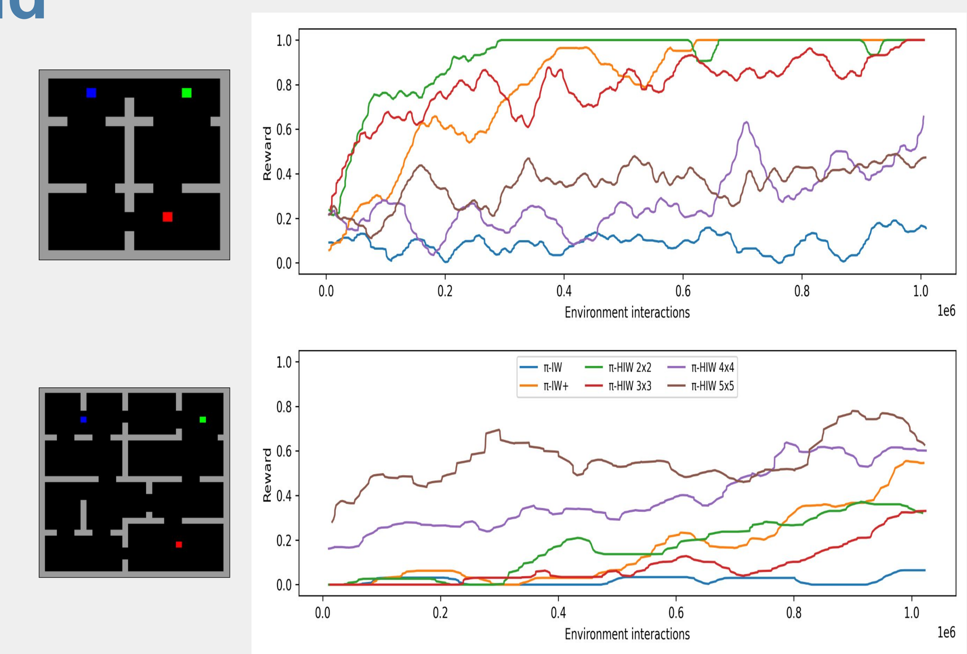
π-HIW: Planning & Learning

- We integrate HIW with a **policy learning** scheme:
  - **High-level** planner: **Count-Based Rollout IW**
    - Selects high-level nodes according to $p \propto \exp(1/\tau(c+1))$
    - Prunes nodes using a mapping from novel tuples to unpruned nodes
  - **Low-level** planner: **π-IW modified**
    - Tree counts for tie-breaking
    - Value function

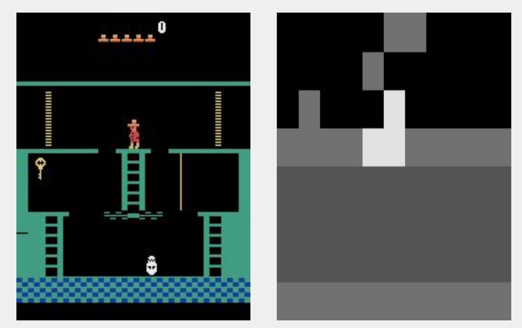


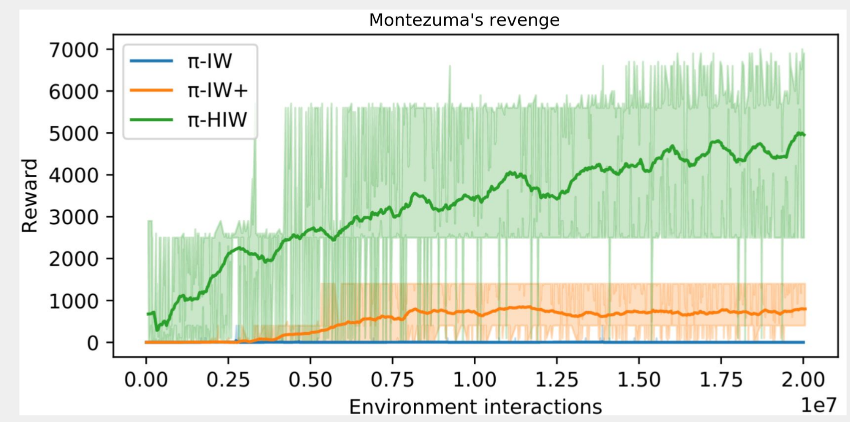
Results in gridworld

- Two sparse reward tasks
- The episode terminates:
  - when the agent picks the key and reaches the door (r = +1)
  - when hitting a wall (r = -1)
  - after 200 / 500 steps (r = 0)

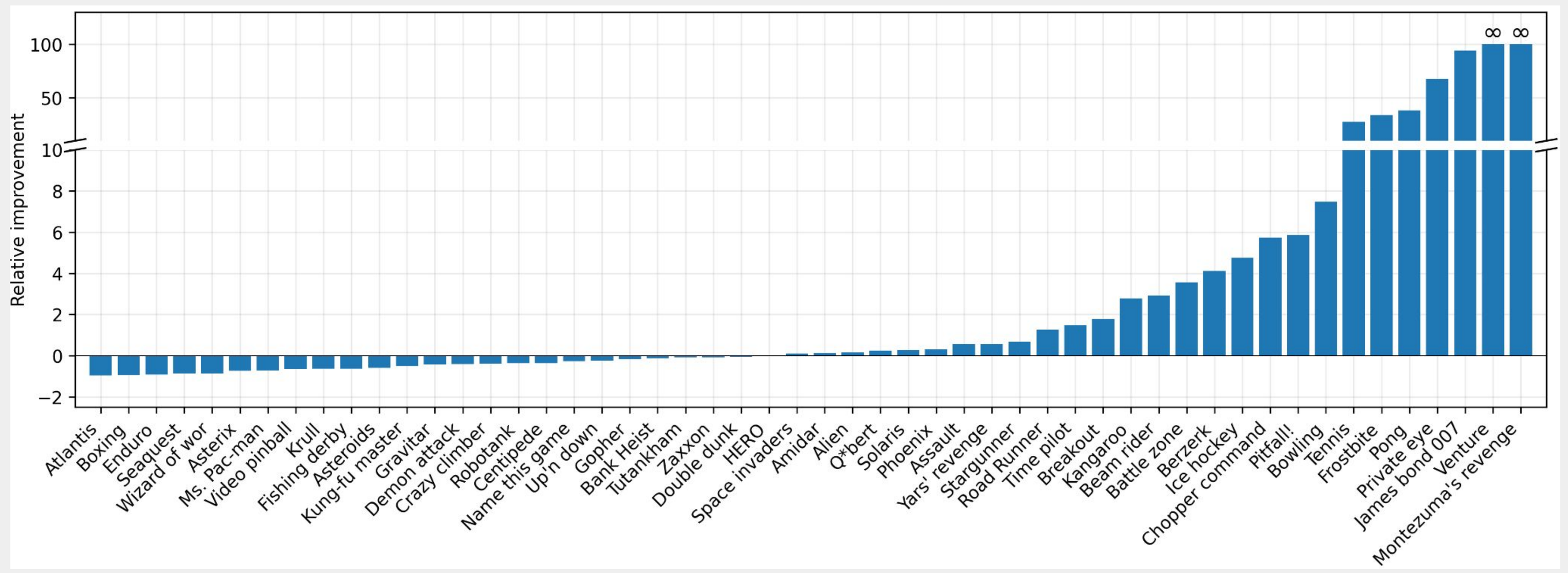


Results in Atari games





Features:
- Neural network activations **(low level)**
- Downsampling **(high level)**



miquel.junyent@upf.edu

Universitat Pompeu Fabra Barcelona