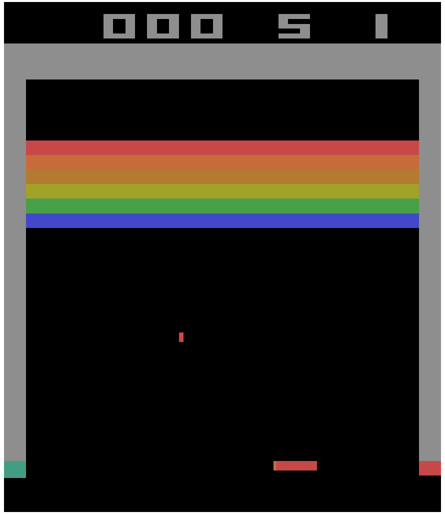
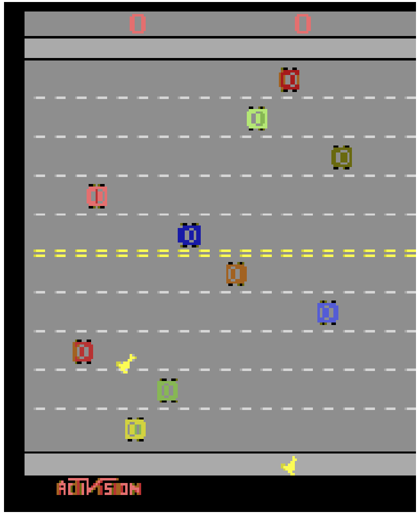
Width-based Lookaheads with Learnt Base Policies and Heuristics Over the Atari-2600 Benchmark

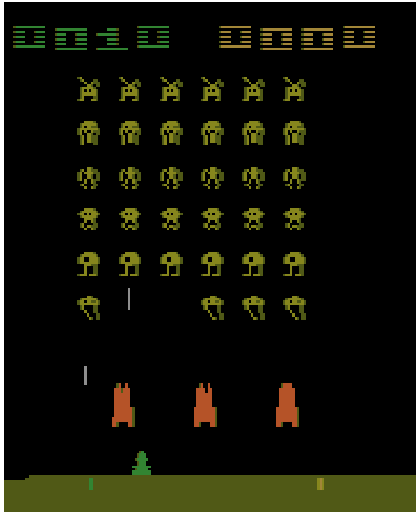
Stefan O'Toole, Nir Lipovetzky, Miquel Ramirez, Adrian R. Pearce

THE UNIVERSITY OF MELBOURNE

Atari Benchmark

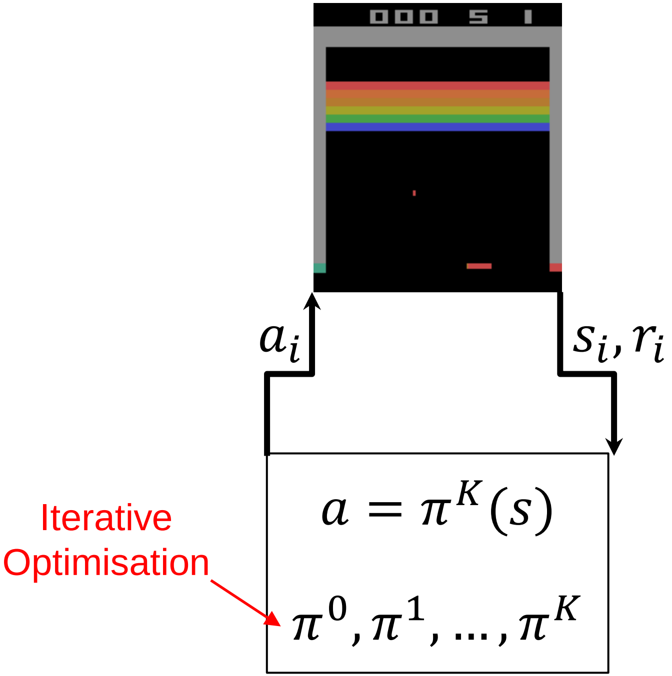




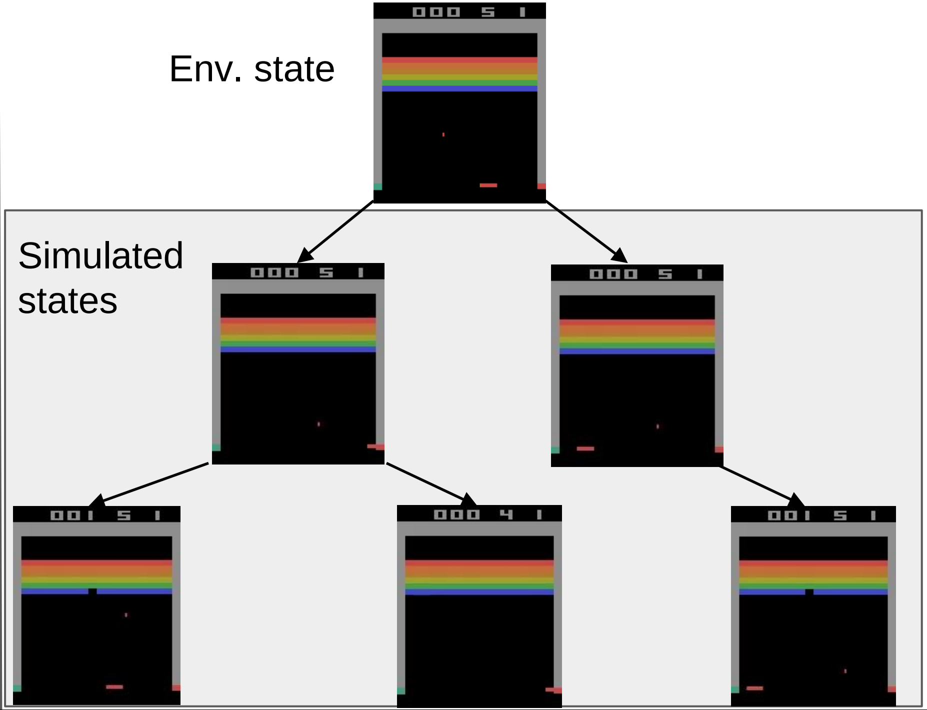


Playing Atari

Reinforcement Learning



Lookaheads



Research Questions

How can lookahead and learning methods be combined to create sample efficient algorithms?

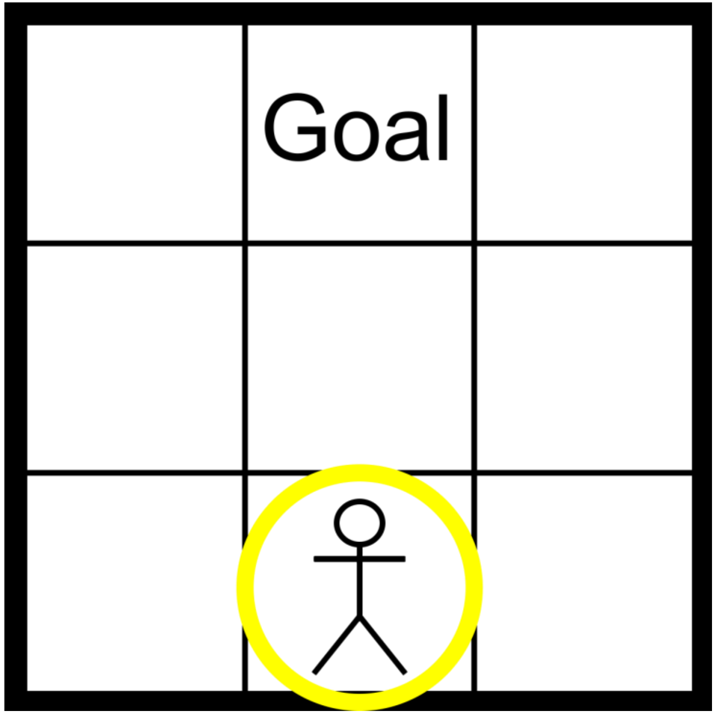
- We analyse the existing π-IW and AlphaZero algorithms
- Our alg. outperforms previous best alg. in 32/53 games

What are the structural properties of the transition system that are good predictors of a certain algorithm's performance?

- We present a taxonomy for comparing Atari results

Width-Based Lookaheads

Environment



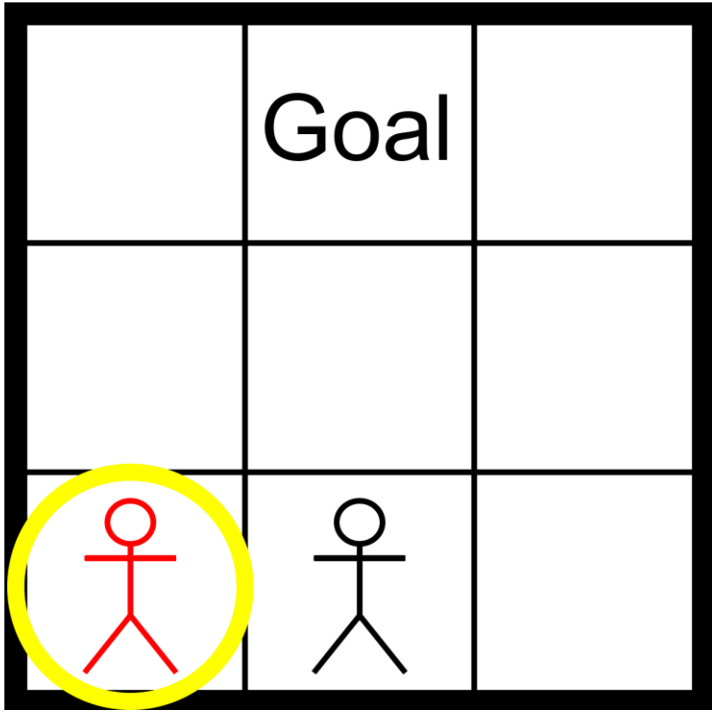
Lookahead

Feature table

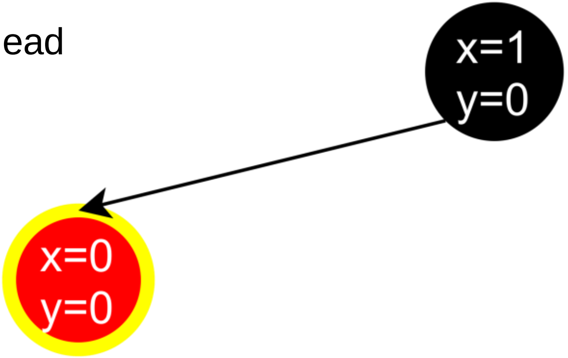
x	0	~~1~~	2
y	~~0~~	1	2

Width-Based Lookaheads

Environment



Lookahead

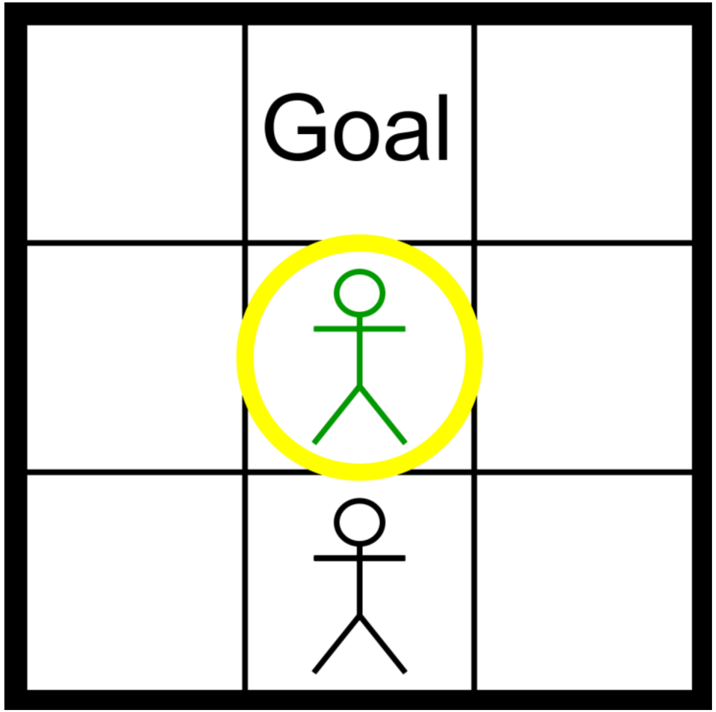


Feature table

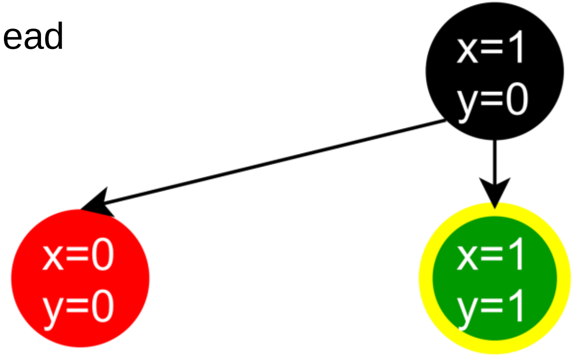
x	~~0~~	~~1~~	2
y	~~0~~	1	2

Width-Based Lookaheads

Environment



Lookahead

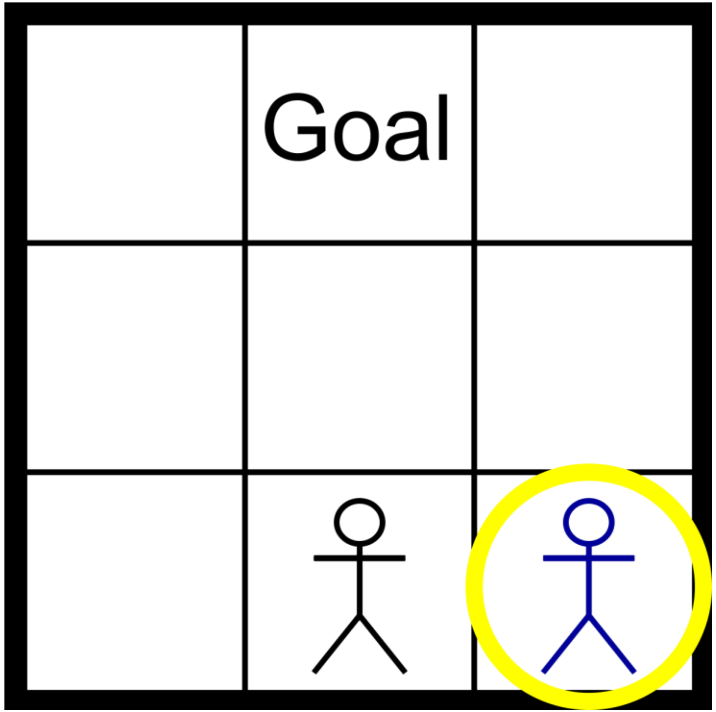


Feature table

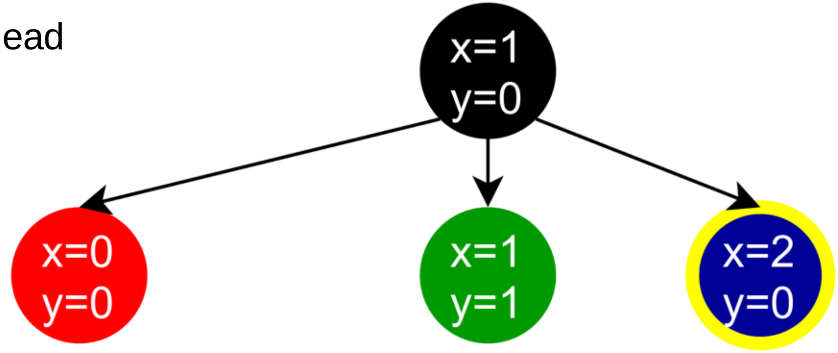
x	~~0~~	~~1~~	2
y	~~0~~	~~1~~	2

Width-Based Lookaheads

Environment



Lookahead

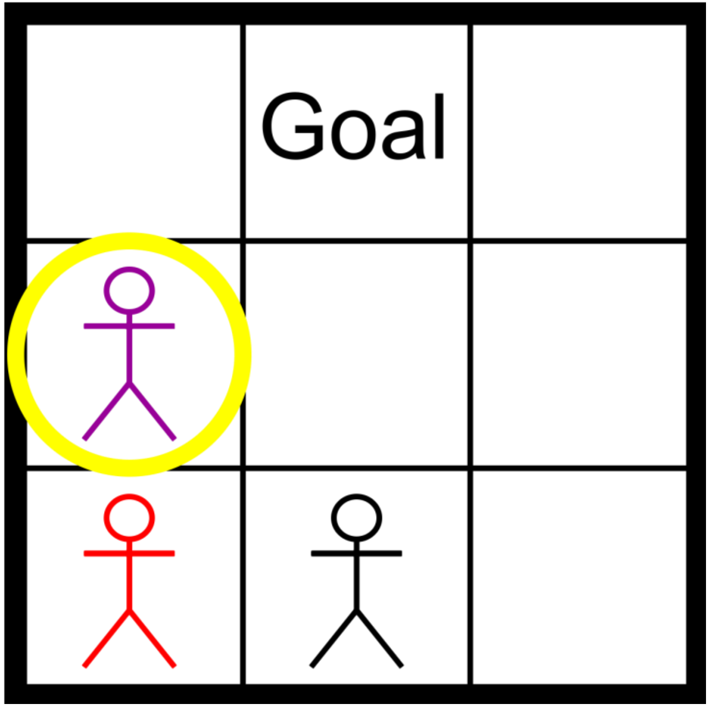


Feature table

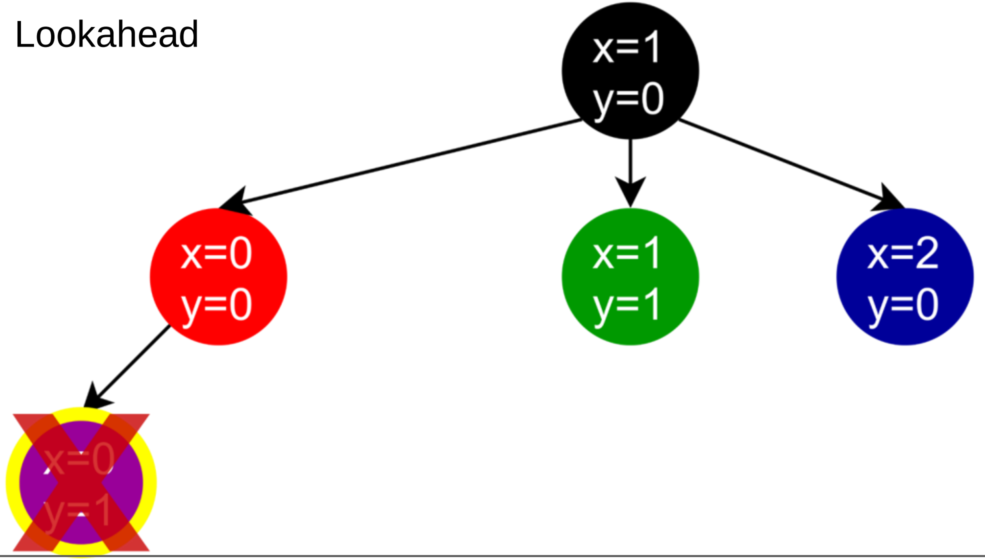
x	~~0~~	~~1~~	~~2~~
y	~~0~~	~~1~~	2

Width-Based Lookaheads

Environment



Lookahead

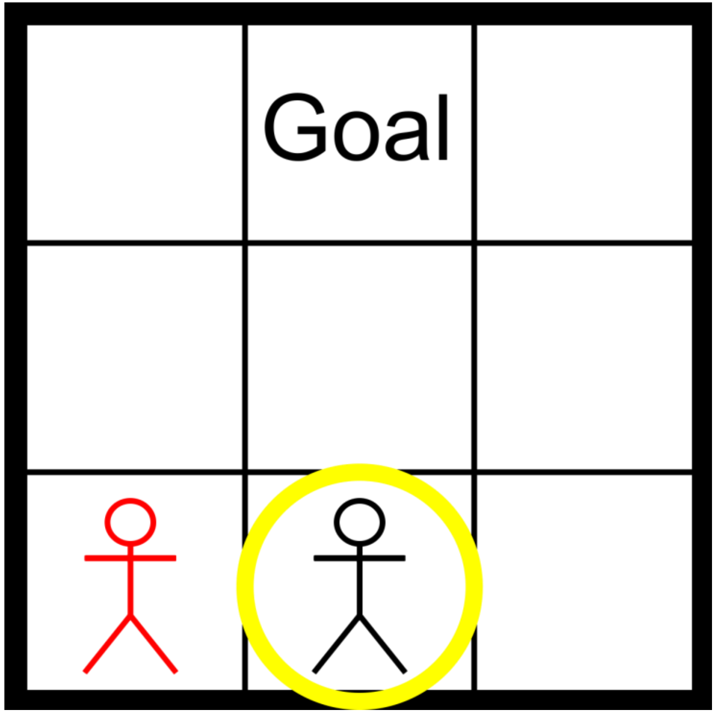


Feature table

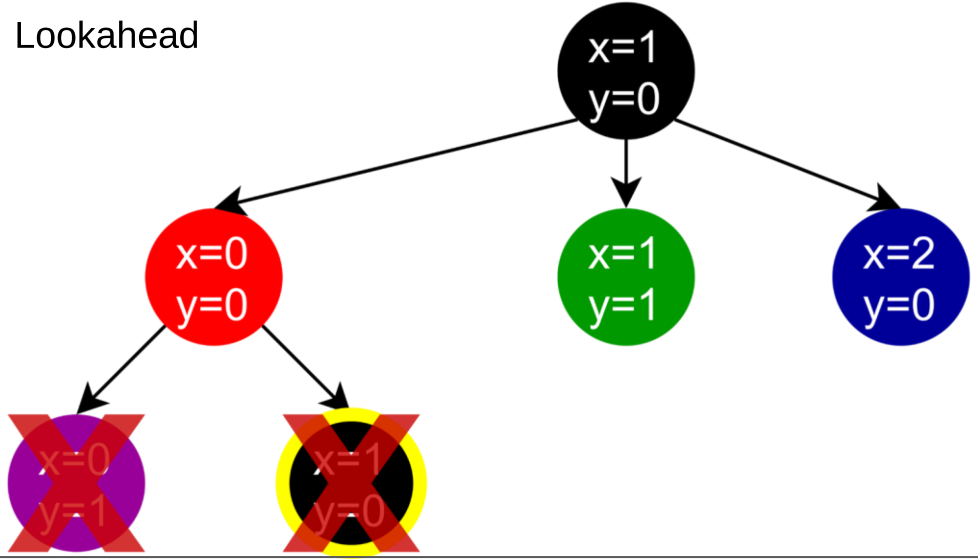
x	0	1	2
y	0	1	2

Width-Based Lookaheads

Environment



Lookahead

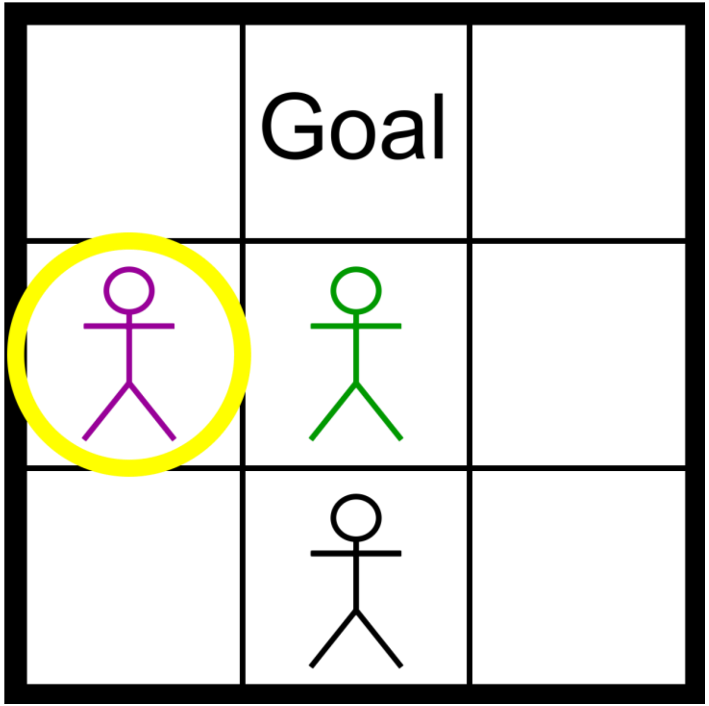


Feature table

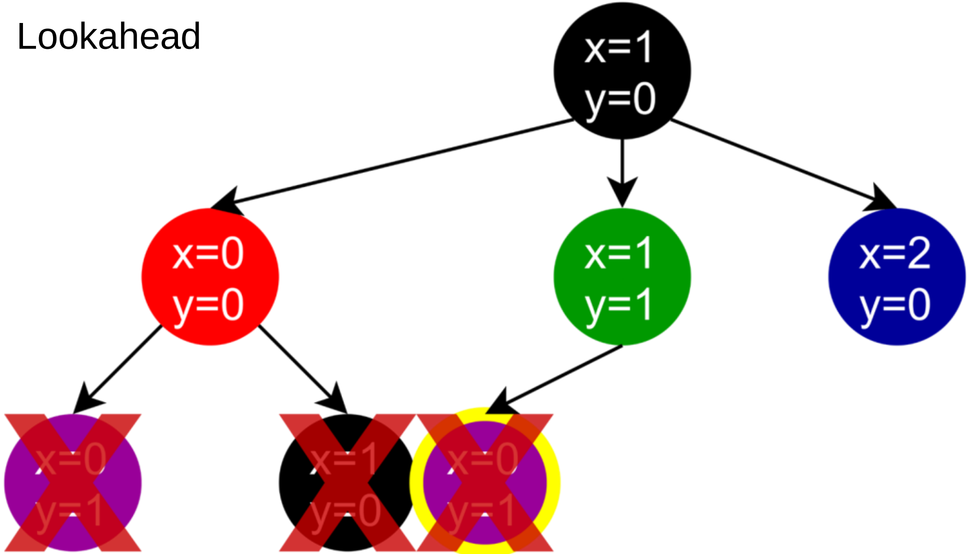
x	~~0~~	~~1~~	~~2~~
y	~~0~~	~~1~~	2

Width-Based Lookaheads

Environment



Lookahead

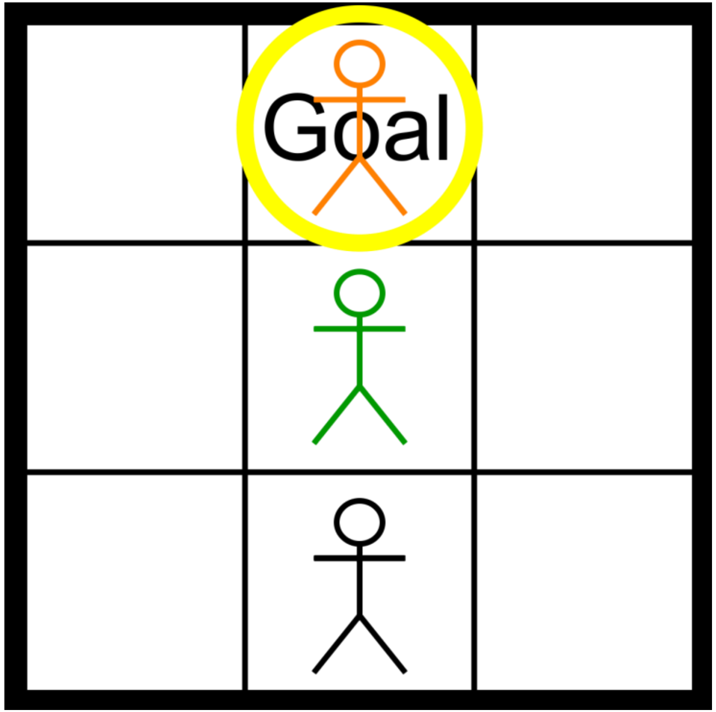


Feature table

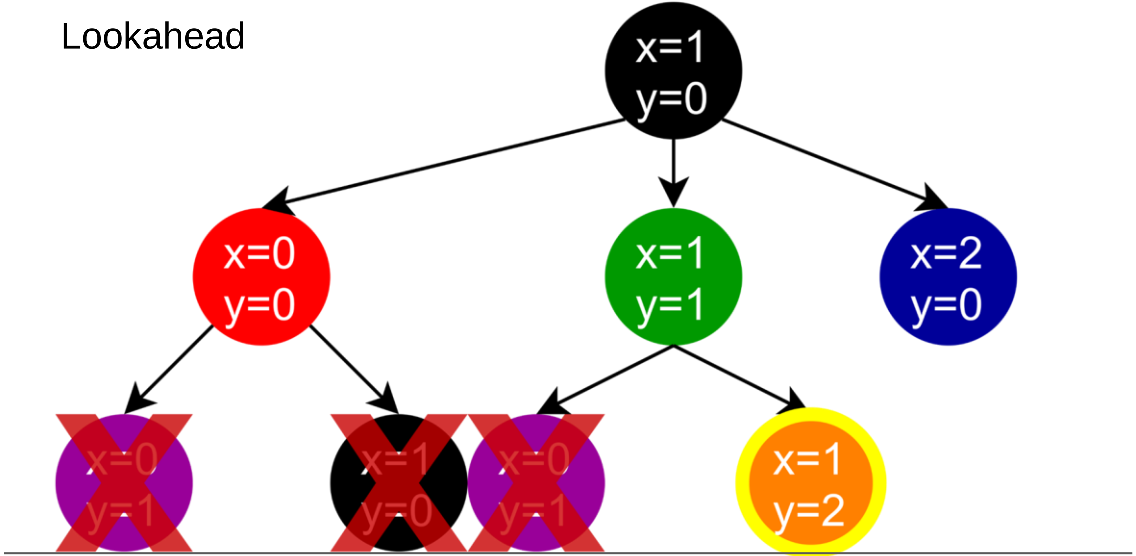
x	~~0~~	~~1~~	~~2~~
y	~~0~~	~~1~~	2

Width-Based Lookaheads

Environment



Lookahead



Feature table

x	0	1	2
y	0	1	2

Width-Based Lookaheads

IW(1) prunes search states according the following novelty measure.

Definition 1 - Novelty for IW(1)[2]

A state s in the search is regarded as novel iff any feature of s has not previously been generated.

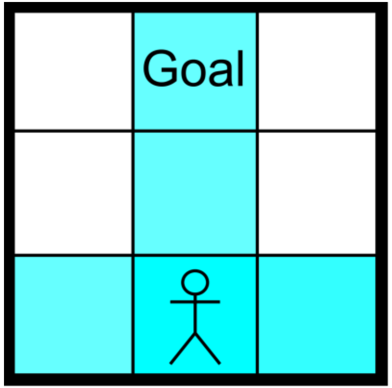
2: Nir Lipovetzky and Héctor Geffner. Width and serialization of classical planning problems. In Proc. of ECAI, 2012.

Width-Based Lookaheads

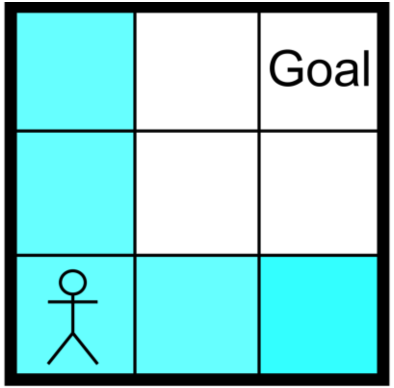
Theorem 1 - Optimality of IW(1)

IW(1) will find the shortest path to every feature along a width 1 trajectory from the initial state in time linear in the number of features.

Width 1 Problem



Width 2 Problem



Width-Based Lookaheads

Rollout-IW(1)[3] (RIW) is a depth-first search that prunes rollouts states that are not considered novel.

Definition 2 - Novelty for RIW(1)
A newly generated state s at depth d in the search is regarded as novel iff any feature of s has not previously been generated at depth ≤ d.

3: Wilmer Bandres, Blai Bonet, and Hector Geffner. Planning with pixels in (almost) real time. In Proc. of AAAI., 2018.

Related Work

- AlphaZero[4] learns a policy and value network used within MCTS
- π-IW(1)[5] learns policy network, becomes base policy for Rollout-IW(1) lookahead.
- π-IW(1)+[6] introduces value network, to backup of rewards
- π-HIW(n,1)[6] combines lookahead algorithms π-IW(1)+ and IW(n).

4: David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and others. Mastering the game of Go without human knowledge. Nature, 2017.

5: Miquel Junyent, Anders Jonsson, and Vicenç Gómez. Deep policies for width–based planning. In Proc. of ICAPS, 2019.

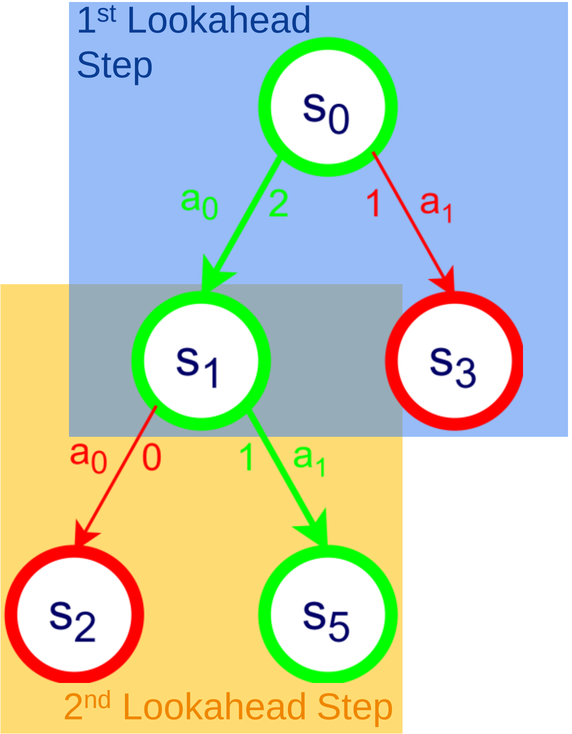
6: Miquel Junyent, Vicenç Gómez, and Anders Jonsson. Hierarchical width-based planning and learning. In Proc of ICAPS, 2021.

New Planning & Learning Alg.

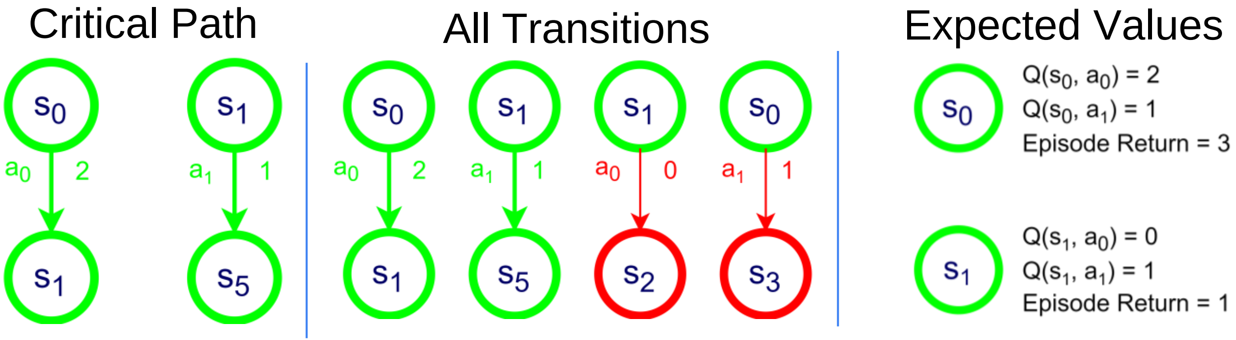
Novelty guided Critical Path Learning (N-CPL)

- Width-based planning and learning algorithm based on RIW(1)
- Runs on a single vCPU
- Learns policy and value networks

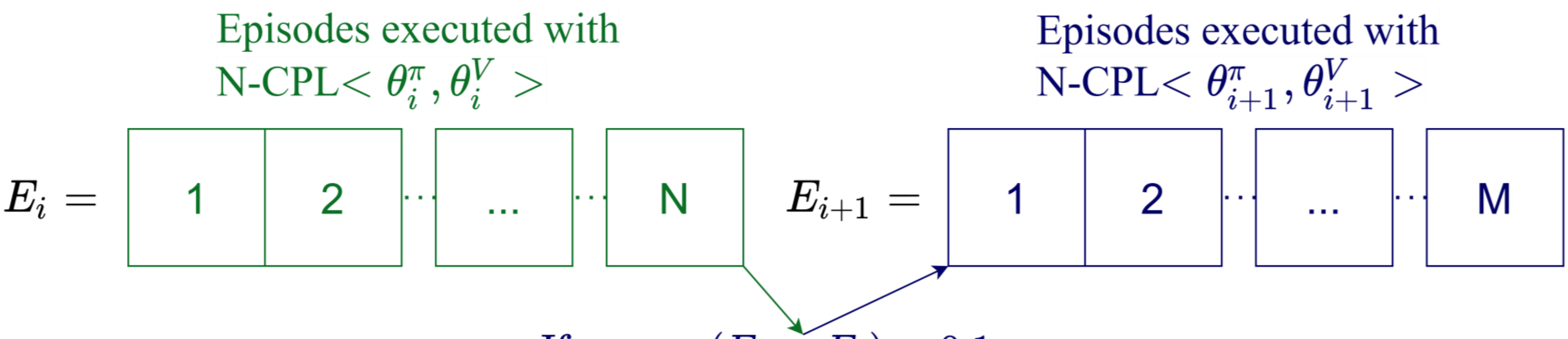
Training Data



- Critical path = transitions selected by the agent.
- N-CPL uses critical path for training value and policy networks.



Learning Schedule



If $\texttt{t\_test}(E_{i-1}, E_i) < 0.1$:

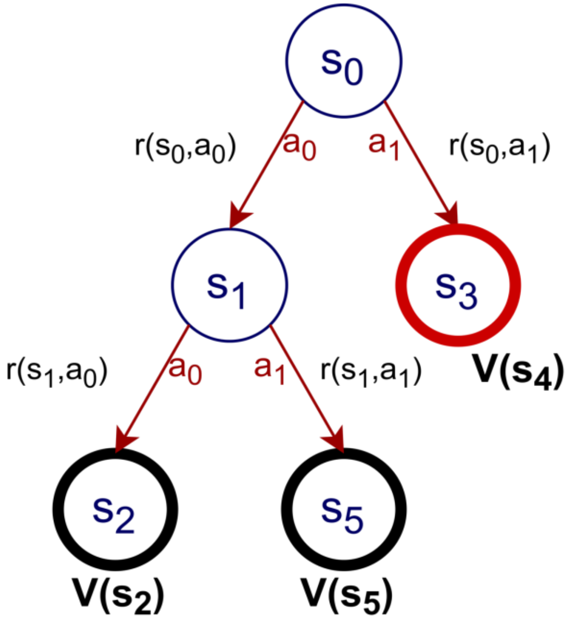
$$< \theta_{i+1}^\pi, \theta_{i+1}^V > = < \theta_{i-1}^\pi, \theta_{i-1}^V >$$

else:

$$< \theta_{i+1}^\pi, \theta_{i+1}^V > = < \theta_i^\pi, \theta_i^V > + < \delta_i^\pi, \delta_i^V >$$

Cost-to-go heuristics

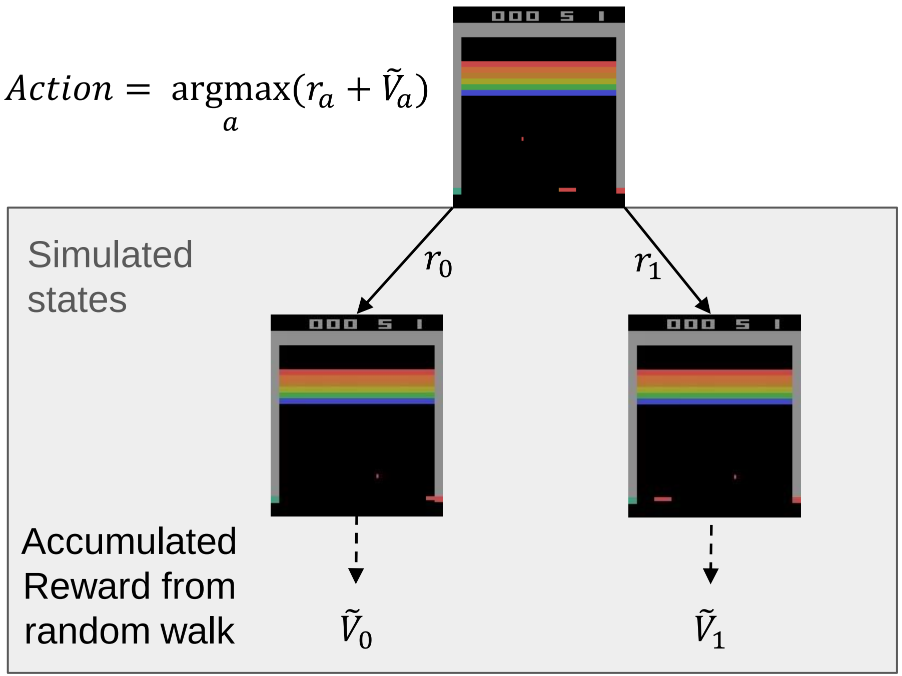
The value function is used as a cost-to-go heuristic at non-terminal leaf nodes.



A taxonomy for Atari Games

- We introduce a definition for a Sparse Meaningful Reward Function (SMRF)
- A game has a SMRF when there is no statistical difference in returns from RTDP with random walks of k times steps vs a random policy
- Analysis includes separating games according to SMRF definition and branching factor

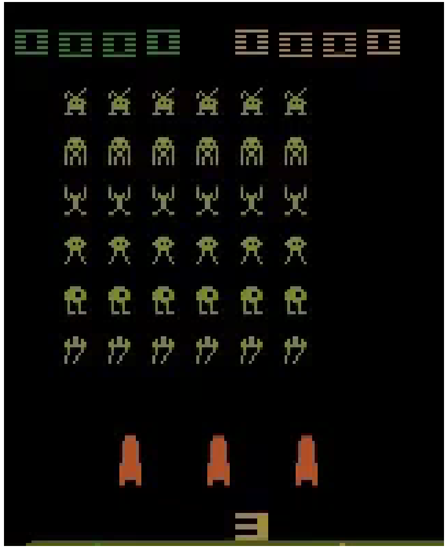
Real-Time Dynamic Programming (RTDP)



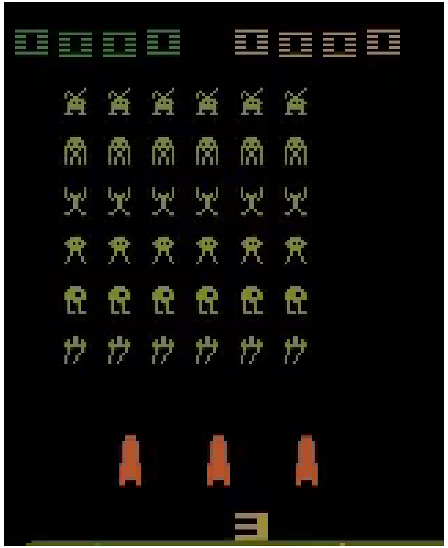
A taxonomy for Atari Games

Not a SMRF

Random Policy

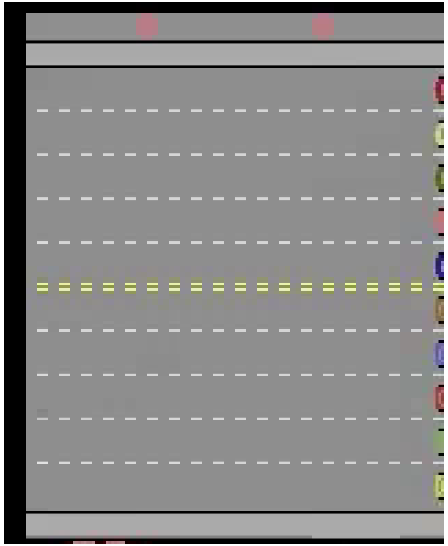


RTDP

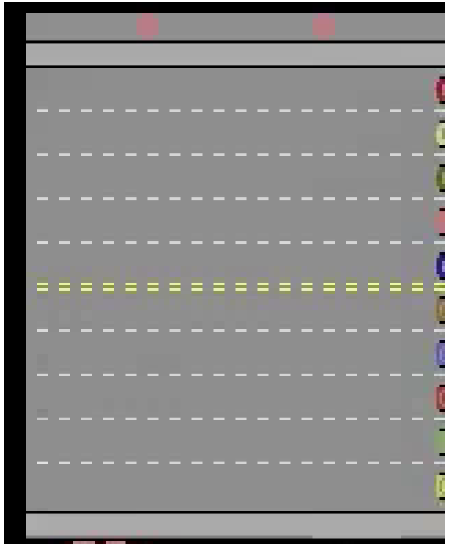


SMRF

Random Policy



RTDP



Planning & Learning without Novelty

CPL only prunes states at the planning horizon.

	CPL vs π-HIW(n,1)
Overall Benchmark (53 Games)	**CPL better** in **51%** Games
SMRF games (12 Games)	**CPL better** in **58%** Games
Large Branching Factor Games (33 Games)	**CPL better** in **64%** Games

Novelty Pruning

Even with a simplified feature set and novelty definition, pruning can improve performance.

	N-CPLvs CPL
Overall Benchmark (53 Games)	**N-CPL better** in **66%** games
SMRF games (12 Games)	**CPL better** in **58%** games
Large Branching Factor (33 Games)	**N-CPL better** in **66%** games

Comparison with Model-Free RL

Note the difference in evaluation settings

Setting	N-CPL	Rainbow
Frame skip	15	4
Simulation Budget	20M sim. calls (300M Frames)	50M sim. calls (200M Frames)
Train Time	~3 days (vCPU)	~10 days (GPU)
Training Data	0.2M sim. calls (3M Frames)	50M sim. calls (200M Frames)
Starts	-	Random no-ops
Loss of Life Signal	No	Yes
Max. ep. Length	1,200 sim. calls (18,000 frames)	27,000 sim. calls (108,000 frames)

	N-CPL vs Rainbow
Overall Benchmark (51 Games)	**Rainbow better in 51%** games
SMRF games (11 Games)	**N-CPL better** in **55%** games
Large Branching Factor (32 Games)	**N-CPL better** in **66%** games

Key takeaways

1. Separating games according to branching factor and Sparse Meaningful Reward Functions provide useful insights into the behaviour of Lookahead Algorithms

2. Simpler novelty definition and features over the raw pixel values perform very strongly

3. Novelty pruning very often further improves performance

4. Learning from critical path transitions with a methodological learning schedule outperforms previous lookahead methods