Guiding Search with Generalized Policies for Probabilistic Planning

William Shen[1], Felipe Trevizan[1] , Sam Toyer[2] ,
Sylvie Thiébaux[1] and Lexing Xie[1]

1 Australian National University

2 Berkeley UNIVERSITY OF CALIFORNIA

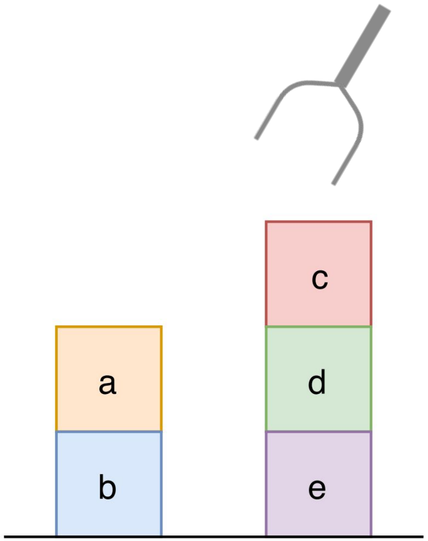
Motivation

- Action Schema Networks (ASNets)
  - Pro: Train on limited number of small problems to learn local knowledge, and generalize to problems of any size
  - Con: Suboptimal network, poor choice of hyperparameters, etc.
- Monte-Carlo Tree Search (MCTS) and UCT
  - Pro: Very powerful in exploring the state space of the problem
  - Con: Requires a large number of rollouts to converge to the optimum
- Combine UCT with ASNets to get the best of both worlds, and overcome their shortcomings.

Stochastic Shortest Path (SSP)

An SSP is a tuple $\langle S, s_0, G, A, P, C\rangle$

- finite set of states S → s = {on(a, b), on(c, d), ...}
- initial state $s_0 \in S$
- set of goal states $G \subseteq S$
- finite set of actions A → pickup, putdown, stack, unstack
- transition function $P(s' \mid a, s)$ → pickup(a) => 0.9: SUCCESS
  0.1: FAILURE
- cost function $C(s, a) \in (0, \infty)$ → for most problems, c(s, a) = 1
- **Solution to a SSP**: stochastic policy $\pi(a \mid s) \in [0, 1]$
  - SSPs have a deterministic optimal policy π^*



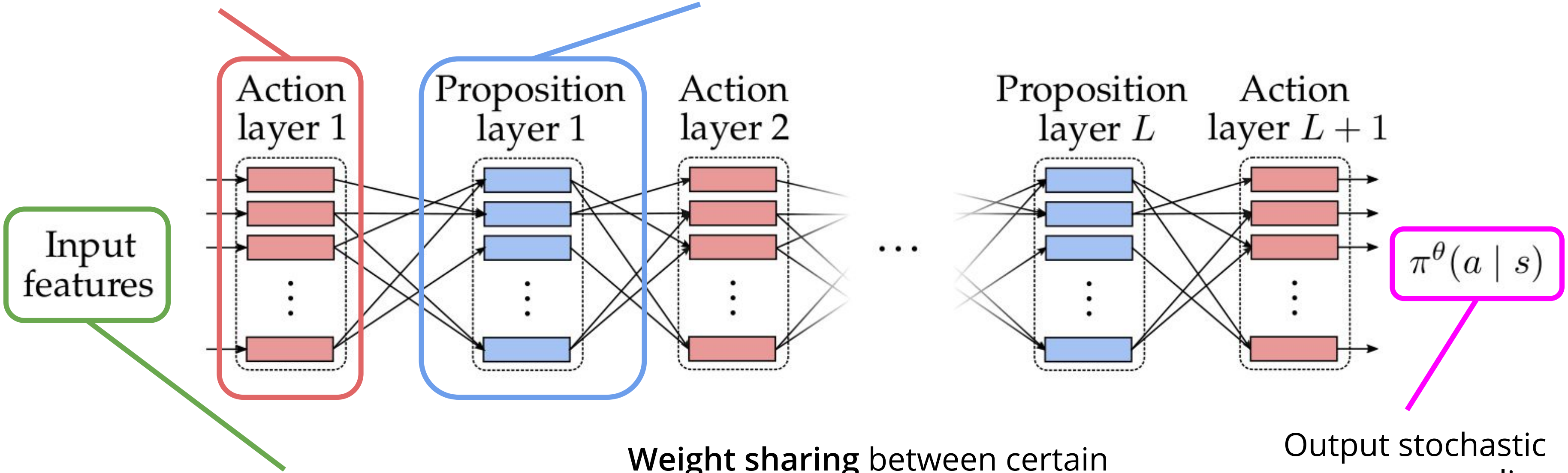
Action Schema Networks (ASNets)

Toyer et al. 2018. In AAAI

Action module for each ground action

Proposition module for each ground predicate

Sparse connections - only connect modules that *affect* each other.



Output stochastic policy

Proposition truth values, goal information (LM-Cut features)

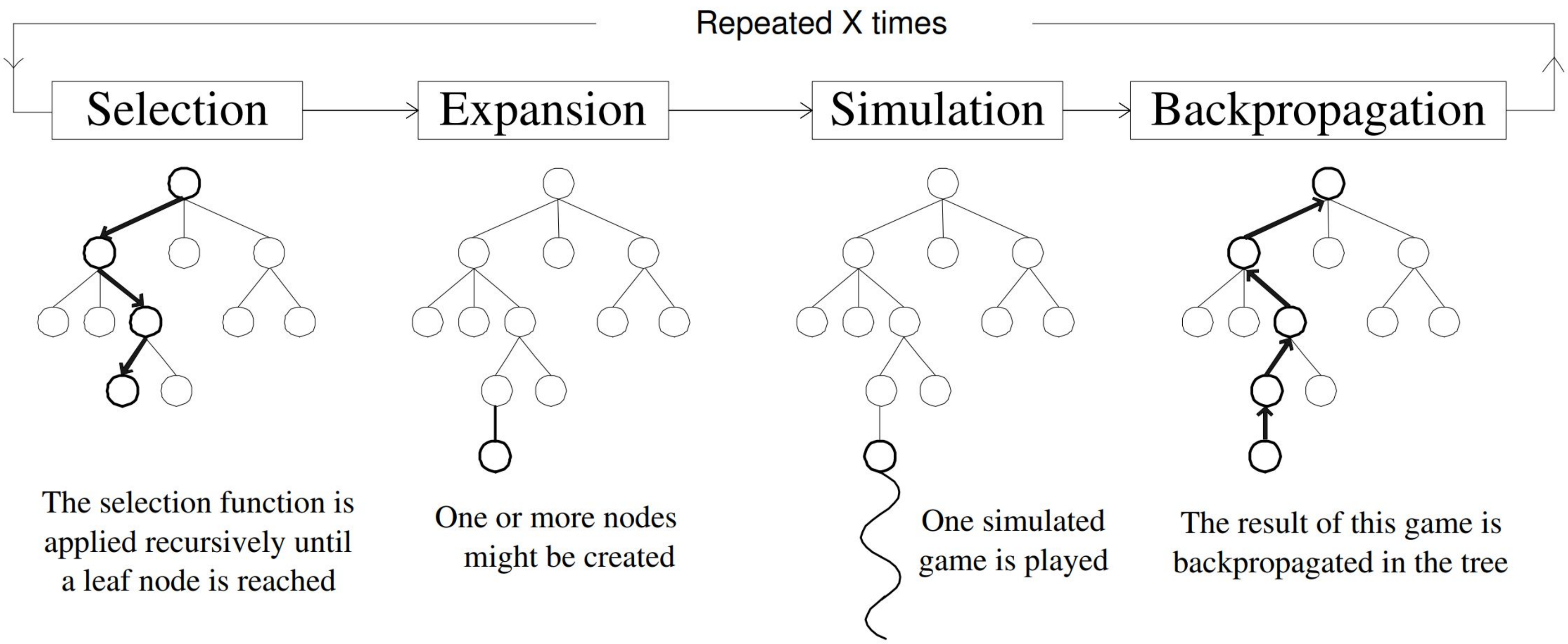
Weight sharing between certain modules in the same layer. Scale up to problems with any number of actions and propositions.

Action Schema Networks (ASNets)

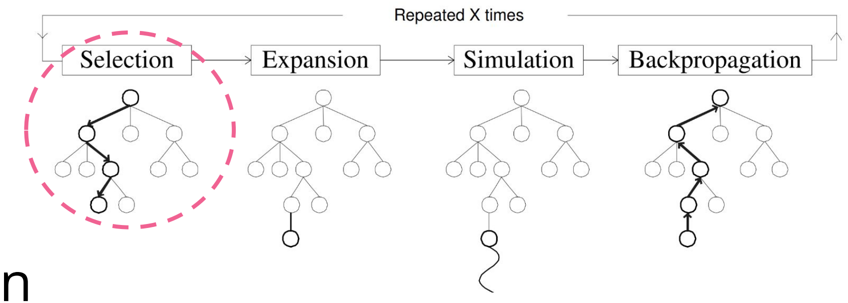
- Pros: Learns a generalized policy for a given planning domain
  - Policy can be applied to any problem in the domain
  - Learns domain-specific knowledge
  - ASNets learn a ‘trick’ to easily solve every problem in the domain
  - Train on small problems, scale up to large problems without retraining
- Cons:
  - Fixed number of layers, limited receptive field
  - Poor choice of hyperparameters, undertraining/overtraining
  - Unrepresentative training set
  - No generally applicable ‘trick’ to solve problems in a domain

Monte-Carlo Tree Search (MCTS)

Sample and score trajectories



Selection Phase



- Balance exploration and exploitation
  - Upper Confidence Bound 1 Applied to Trees (UCT)

Number of times state has been visited.

Bias (free parameter)

Proxy for state

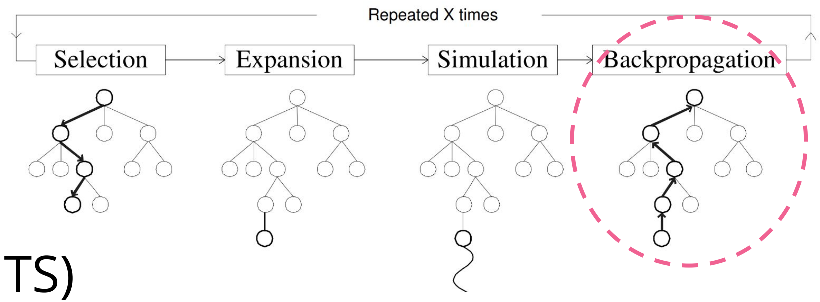
$$\mathrm{UCB1}(n_d, n_c) = B \cdot \overbrace{\sqrt{\frac{\log C^k(n_d)}{C^k(n_c)}}}^{\textbf{Exploration}} - \overbrace{Q^k(n_c)}^{\textbf{Exploitation}}$$

Proxy for action in state

Number of times action has been applied in state

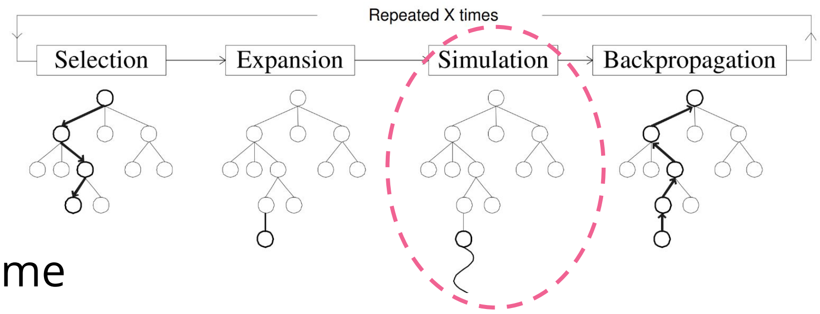
Estimate of cost to reach goal

Backpropagation Phase



1. Trial-Based Heuristic Tree Search (THTS)
   (Keller & Helmert. 2013. ICAPS)
   - Ingredient-based framework to define trial-based heuristic search algorithms

2. **Dynamic Programming UCT** (DP-UCT)
   - Uses Bellman backups
     - Known transition function
   - UCT* - variant where trial length is 0
     - Baseline algorithm



Simulation Phase

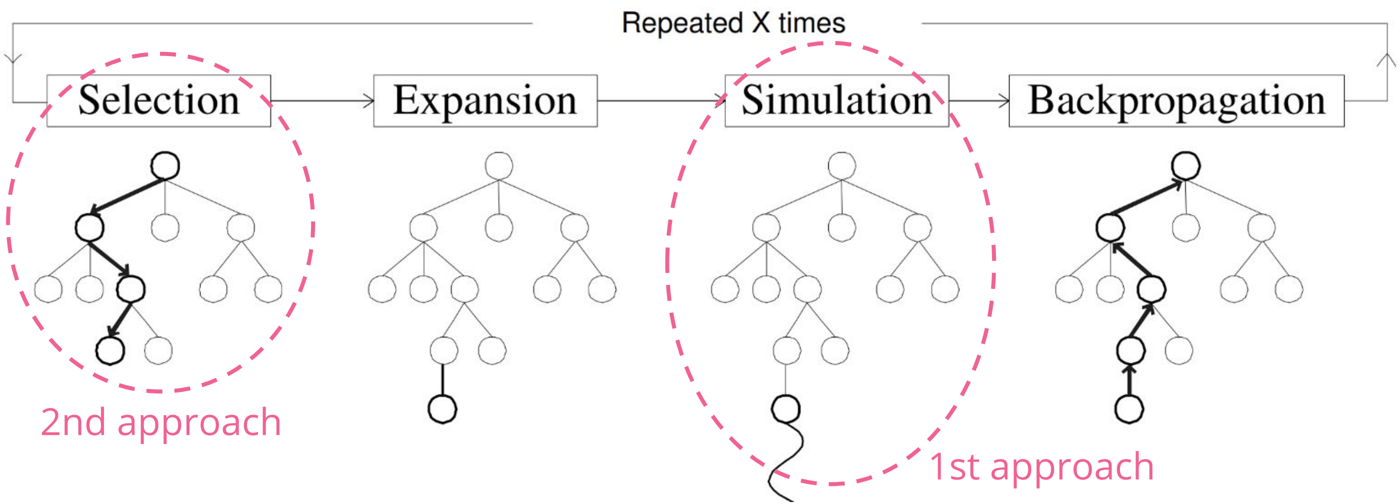
- THTS alternates between action and outcome selection using the heuristic function

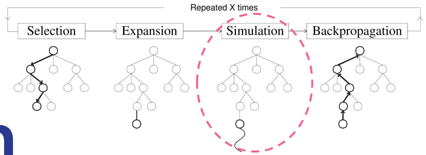
- Re-introduce the **Simulation Phase**:
  - Perform rollouts using the **Simulation Function**
  - Traditional MCTS algorithms use a random simulation function

- **Why?** Current heuristics are not quite informative because of dead ends.
  - Underestimate probability of reaching dead end
  - Very optimistic about avoiding dead ends

Combining ASNets and UCT

1. Learn what an ASNet has not learned
2. Improve suboptimal learning
3. Robust to changes in the environment or domain





Using ASNets as a Simulation Function

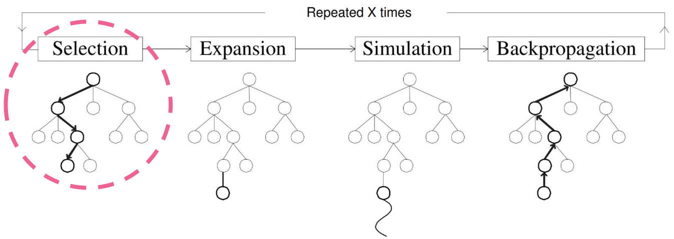
- **Max-ASNet:** select action in the policy with the highest probability
- **Stochastic-ASNet:** sample an action in the policy using the probability distribution
- Not very robust if policy is uninformative/misleading

$$\pi(s) = \begin{cases} 0.4 : \text{stack}(a, b) \\ 0.1 : \text{stack}(a, d) \\ 0.2 : \text{put-down}(a) \\ 0.3 : \text{stack}(a, c) \end{cases}$$

Max-ASNet: `argmax π(a|s)`

Stochastic-ASNet: sample from `π(s)`

Using ASNets in UCB1



- Need to maintain balance between exploration and exploitation
- Add exploration bonus that converges to zero as action applied infinitely often - more robust

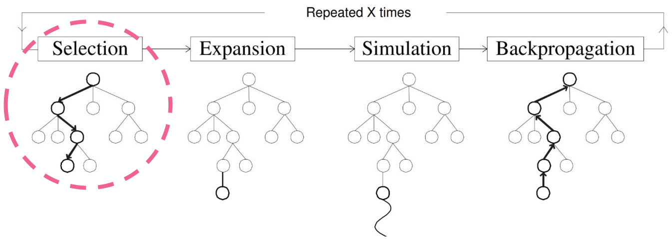
Influence Constant

Probability of applying action in state

$$\textsc{Simple-ASNet}(n_d, n_c) = \frac{M \cdot \pi(n_c)}{C^k(n_c)} + \text{UCB1}(n_d, n_c)$$

Number of times action has been applied in state

Using ASNets in UCB1



- In Simple-ASNets, a network's policy is only considered after all actions have been explored at least once
- **Ranked-ASNet** action selection:
    - Select unvisited actions by their probability (ranking) in the policy
- Focus initial stages of search on actions an ASNet suggests

$$\pi(s) = \begin{cases} 0.4 : \text{stack(a, b)} & \textbf{1st} \\ 0.1 : \text{stack(a, d)} & \textbf{4th} \\ 0.2 : \text{put-down(a)} & \textbf{3rd} \\ 0.3 : \text{stack(a, c)} & \textbf{2nd} \end{cases}$$

Evaluation

- **Three experiments**
  - Each designed to test whether we can achieve the 3 goals
  - Maximize the quality of the search in the limited computation time
- **Recall our goals**
  - Learn what ASNets have not learned
  - Improve suboptimal learning
  - Robust to changes in the environment or domain

Improving on the Generalized Policy

Objectives:

- *Learn what we have not learned*
- *Improve suboptimal learning*

- **Exploding Blocksworld** - extension of Blocksworld with dead-ends and probabilities
- Very difficult for ASNets
  - Each problem may have its own 'trick'
  - Training set may not be representative of test set
- Can the limited knowledge learned by the network help UCT?

Improving on the Generalized Policy

Coverage over 30 runs for a subset of problems

Planner/Prob.	p02	p04	p06	p08
ASNets	10/30	0/30	19/30	0/30
UCT*	9/30	11/30	28/30	5/30
Ranked ASNets (*M* = 10)	6/30	10/30	25/30	4/30
Ranked ASNets (*M* = 50)	10/30	**15/30**	27/30	**10/30**
Ranked ASNets (*M* = 100)	12/30	10/30	29/30	4/30

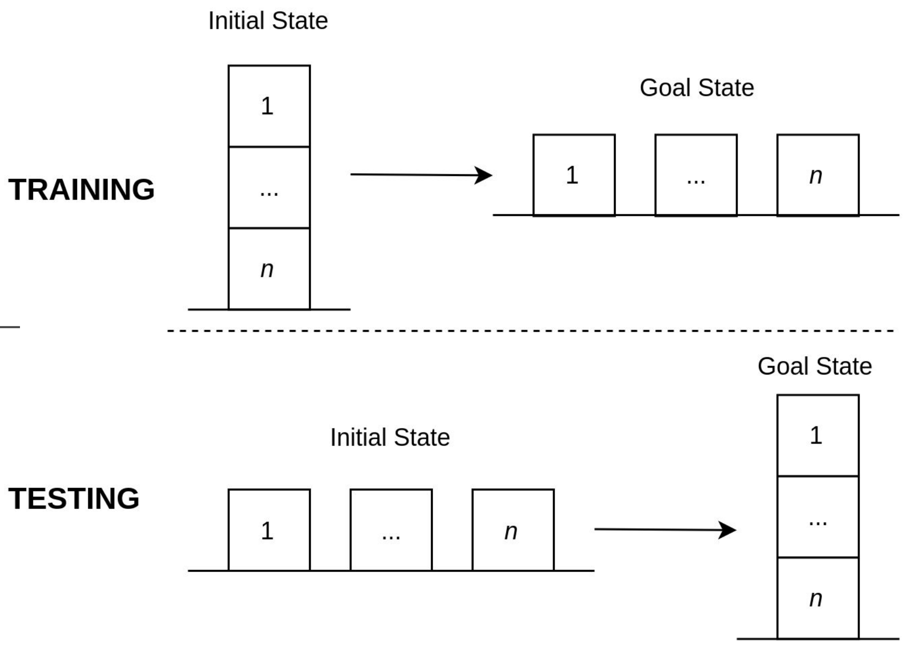
For results for full set of problems, please see our paper.

Combating an Adversarial Training Set

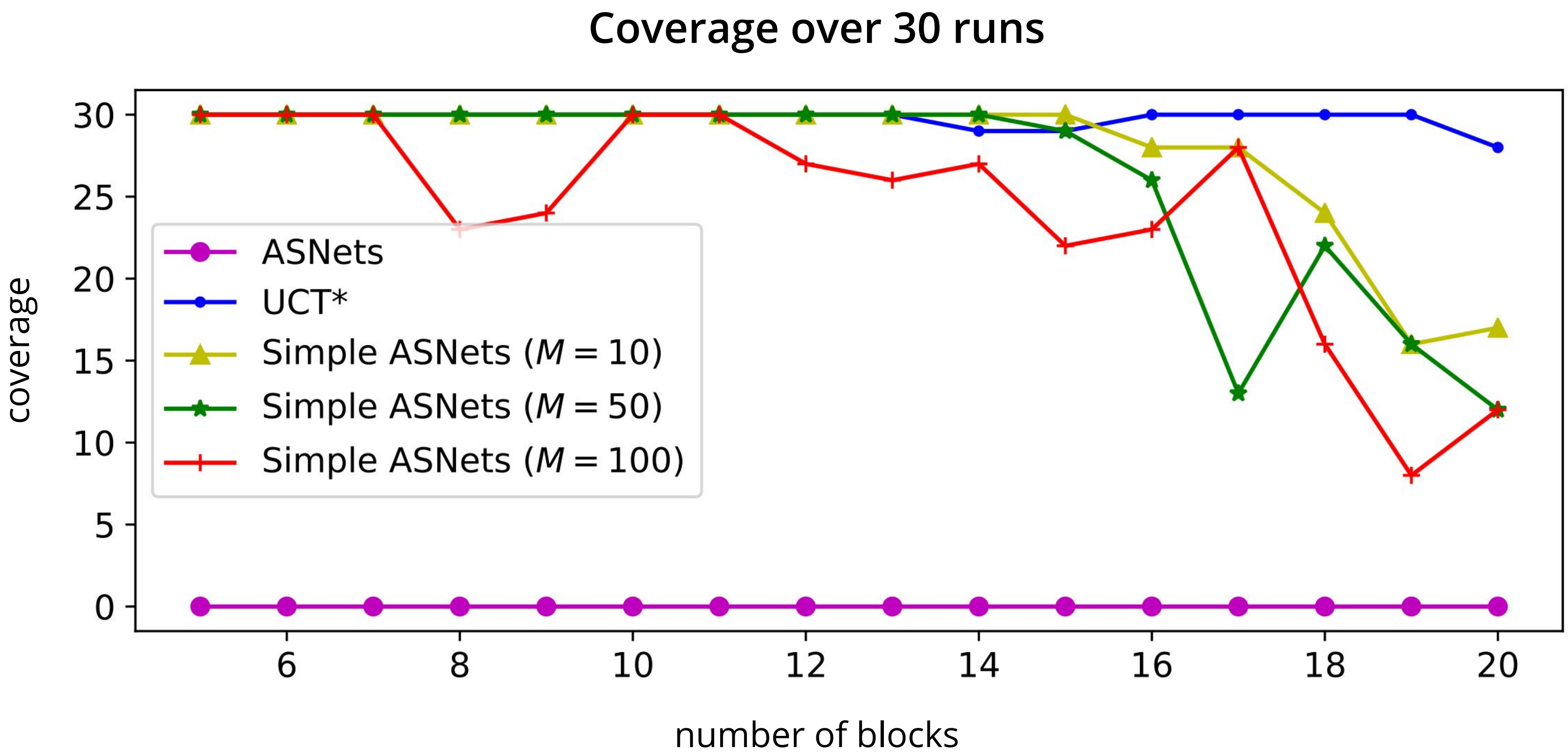
<u>Objectives:</u>

- *Learn what we have not learned*
- *Robust to changes in the environment or domain*

- Train network to unstack blocks
- Test network to stack blocks
- Worst-case scenario for inductive learners

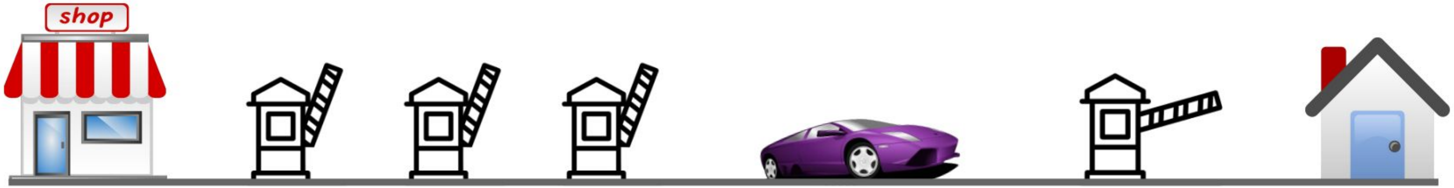


Combating an Adversarial Training Set



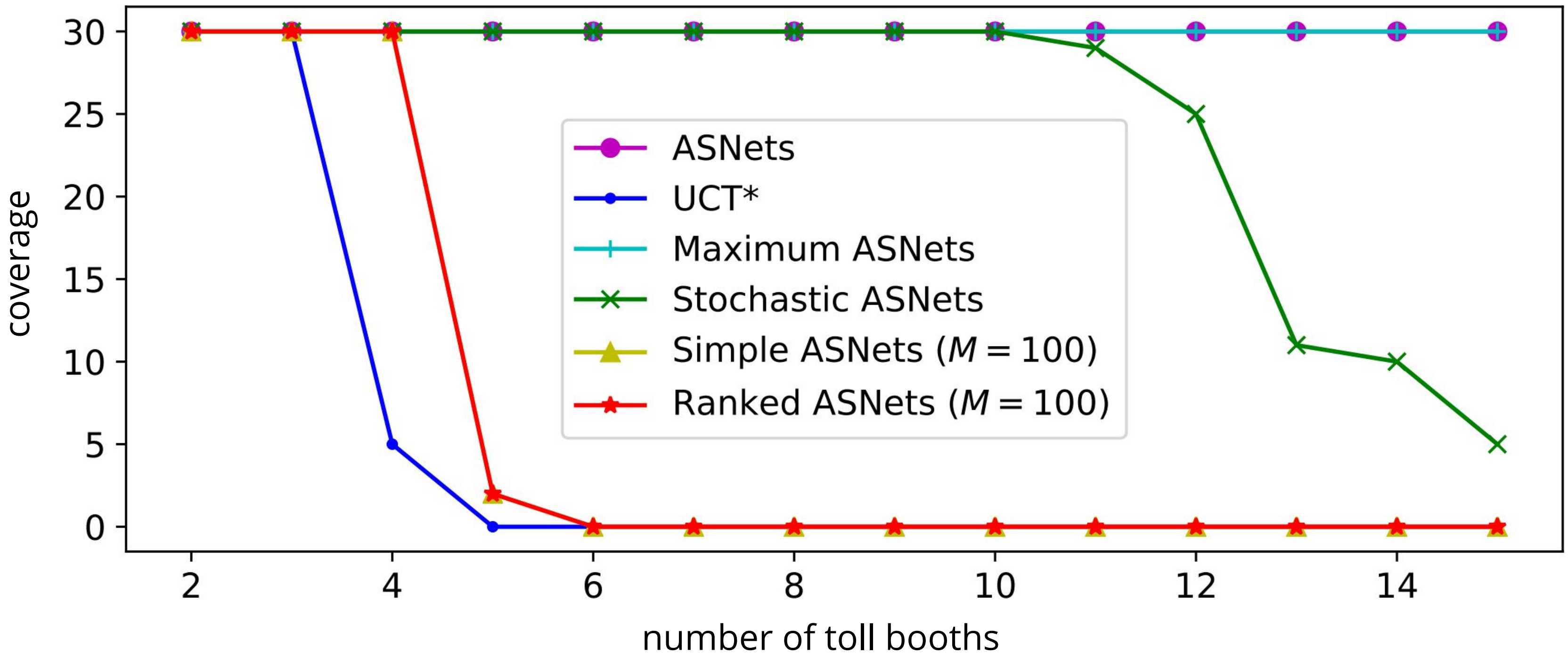
Exploiting the Generalized Policy

- **CosaNostra Pizza** - new domain introduced by Toyer et al. (2018)
  - Probabilistically interesting (has dead ends)
  - Optimal policy: pay toll operator only on trip to customer
- ASNets is able to learn the ‘trick’ to pay the toll operator only on the trip to the customer, and scales up to problems of any size
- Challenging for SSP heuristics (determinization, delete relaxation)
- Requires extremely long reasoning chains



Exploiting the Generalized Policy

Coverage over 30 runs



Conclusion and Future Work

- Demonstrated how to leverage generalized policies in UCT
    - **Simulation Function**: Stochastic and Max ASNets
    - **Action Selection**: Simple and Ranked ASNets
- Initial experimental results showing efficacy of approach
- Future Work
    - 'Teach' UCT when to play actions/arms suggested by ASNets
    - Automatically adjust influence constant *M*, mix ASNet-based simulations with random simulations
    - Interleave training of ASNets with execution of ASNets + UCT

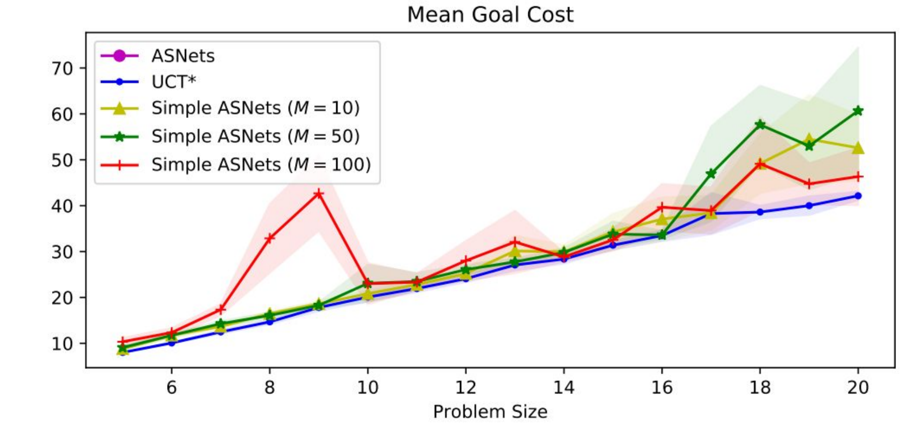
Thanks!

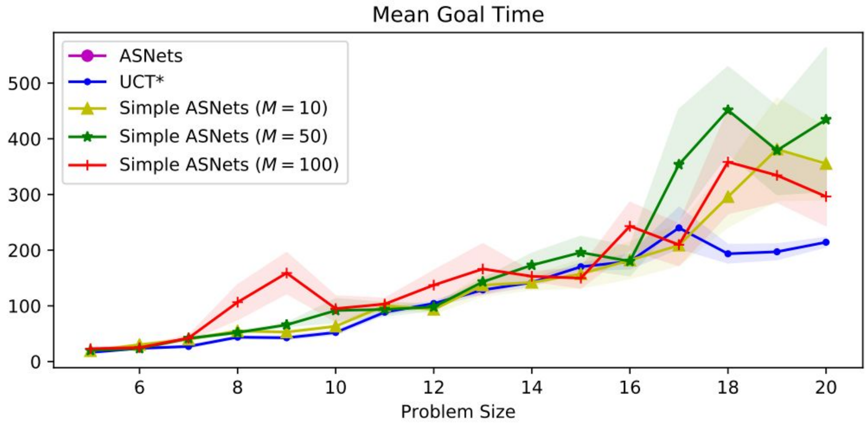
Any Questions?

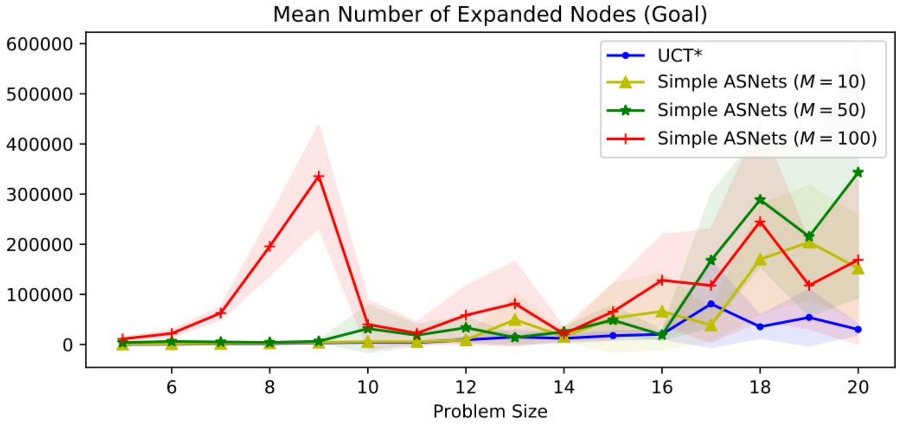
References

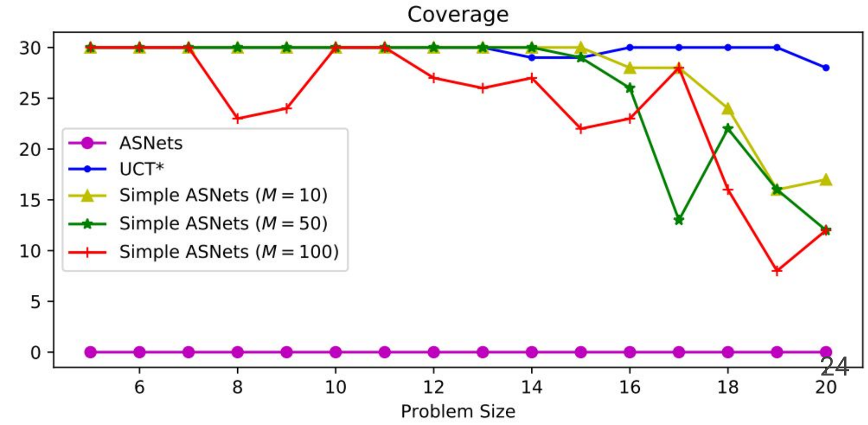
- **MCTS Diagram**: Monte-Carlo tree search in backgammon on ResearchGate
- **CosaNostra Pizza Diagram**: ASNets presentation on GitHub
- **ASNets and associated diagrams**: Toyer, S.; Trevizan, F.; Thiebaux, S.; and Xie, L. 2018. Action Schema Networks: Generalised Policies with Deep Learning. In AAAI.
- **Trial Based Heuristic Tree Search:** Keller, T., and Helmert, M. 2013. Trial-Based Heuristic Tree Search for Finite Horizon MDPs. In ICAPS.
- **Triangle Tireworld**: Little, I., and Thiebaux, S. 2007. Probabilistic Planning vs. Replanning. In ICAPS Workshop on IPC: Past, Present and Future

Stack Blocksworld - Additional Results







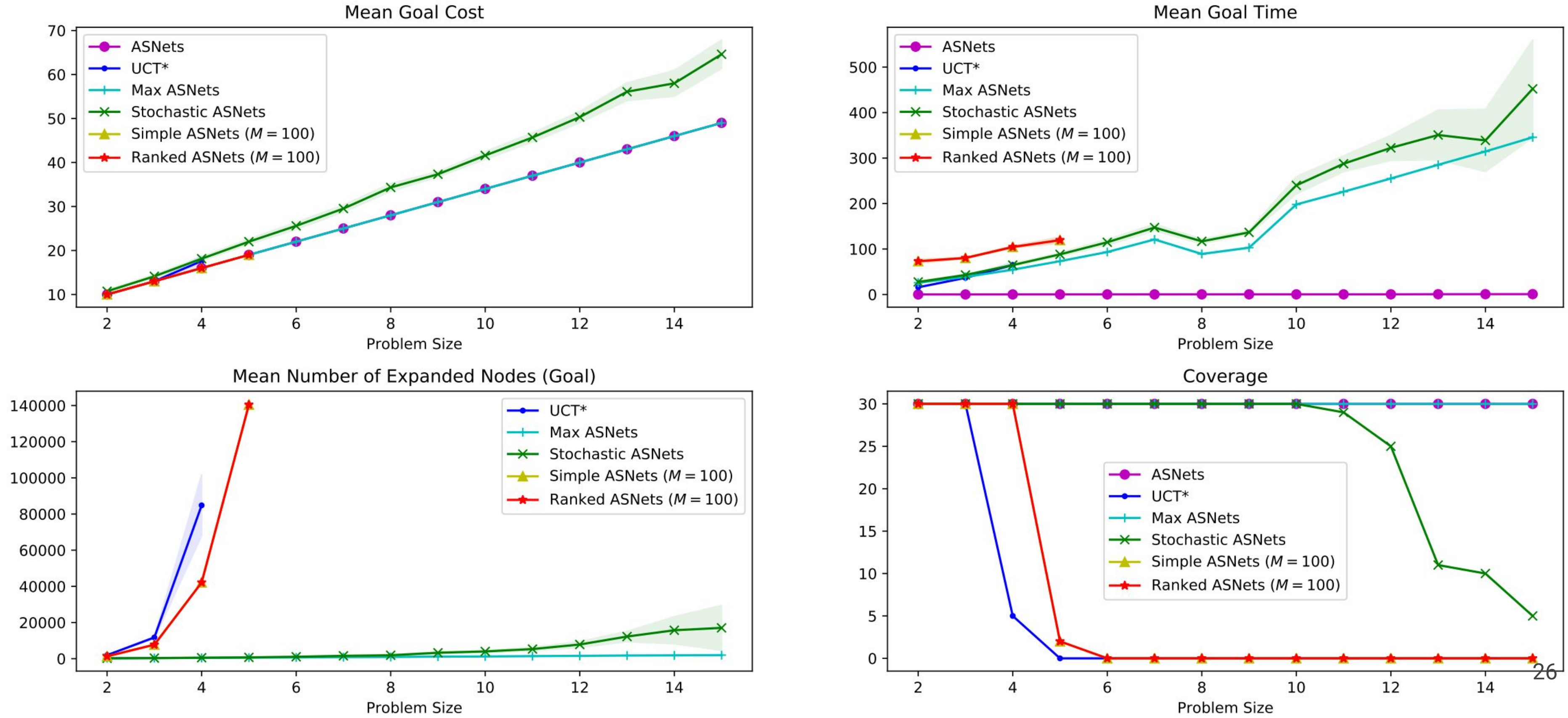


Exploding Blocksworld - Additional Results

Planner/Prob.	p01	p02	p03	p04	p05	p06	p07	p08
ASNets	16/30 8.0 ± 0.0 0.18 ± 0.14s	10/30 12.0 ± 0.0 0.17 ± 0.01s	6/30 10.0 ± 0.0 0.2 ± 0.04s	-	30/30 6.0 ± 0.0 0.19 ± 0.07s	19/30 12.0 ± 0.0 0.42 ± 0.12s	-	-
UCT*	26/30 10.92 ± 0.52 102.51 ± 5.24s	9/30 18.22 ± 1.62 175.01 ± 16.24s	13/30 25.23 ± 8.86 222.27 ± 88.77s	11/30 14.55 ± 0.63 136.46 ± 6.75s	30/30 6.13 ± 0.19 36.51 ± 2.4s	28/30 13.93 ± 0.8 132.36 ± 8.11s	30/30 13.0 ± 0.73 107.11 ± 6.95s	5/30 36.4 ± 5.09 335.87 ± 54.56s
Ranked ASNets $M = 10$	25/30 10.96 ± 0.48 100.21 ± 6.01s	6/30 17.0 ± 3.45 164.77 ± 34.89s	11/30 30.0 ± 13.64 280.25 ± 135.07s	10/30 14.4 ± 0.6 125.74 ± 11.93s	30/30 6.0 ± 0.0 38.11 ± 1.17s	25/30 13.6 ± 0.83 113.56 ± 8.11s	30/30 12.07 ± 0.14 116.36 ± 1.4s	4/30 35.0 ± 7.58 340.82 ± 75.18s
Ranked ASNets $M = 50$	23/30 11.04 ± 0.58 94.17 ± 6.51s	10/30 17.6 ± 2.85 166.29 ± 27.91s	14/30 35.71 ± 7.87 352.14 ± 78.66s	15/30 14.4 ± 0.46 123.06 ± 5.75s	30/30 6.0 ± 0.0 38.85 ± 1.15s	27/30 13.33 ± 0.76 127.69 ± 7.59s	30/30 12.07 ± 0.14 102.57 ± 1.38s	10/30 38.6 ± 0.97 374.93 ± 12.01s
Ranked ASNets $M = 100$	25/30 11.04 ± 0.48 105.26 ± 4.83s	12/30 17.33 ± 2.44 167.75 ± 24.5s	14/30 28.43 ± 6.54 259.18 ± 65.16s	10/30 14.6 ± 0.69 126.61 ± 6.41s	30/30 6.0 ± 0.0 39.41 ± 1.08s	29/30 13.38 ± 0.74 111.66 ± 7.15s	30/30 12.33 ± 0.28 103.56 ± 3.16s	4/30 36.5 ± 9.14 344.06 ± 93.88s

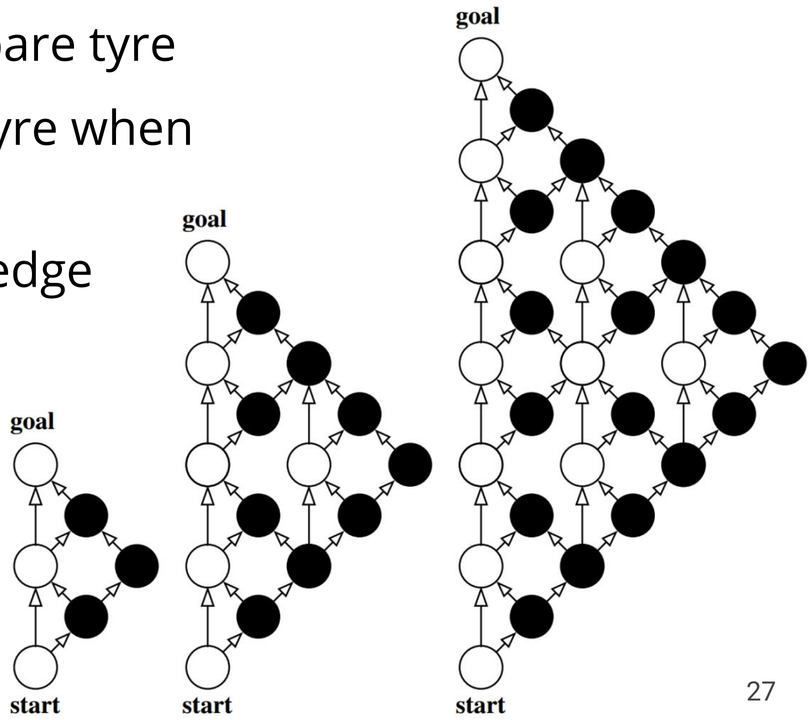
1st line is coverage, 2nd and 3rd lines of each cell show the mean cost and mean time to reach a goal, respectively, and their associated 95% confidence interval.

CosaNostra Pizza - Additional Results

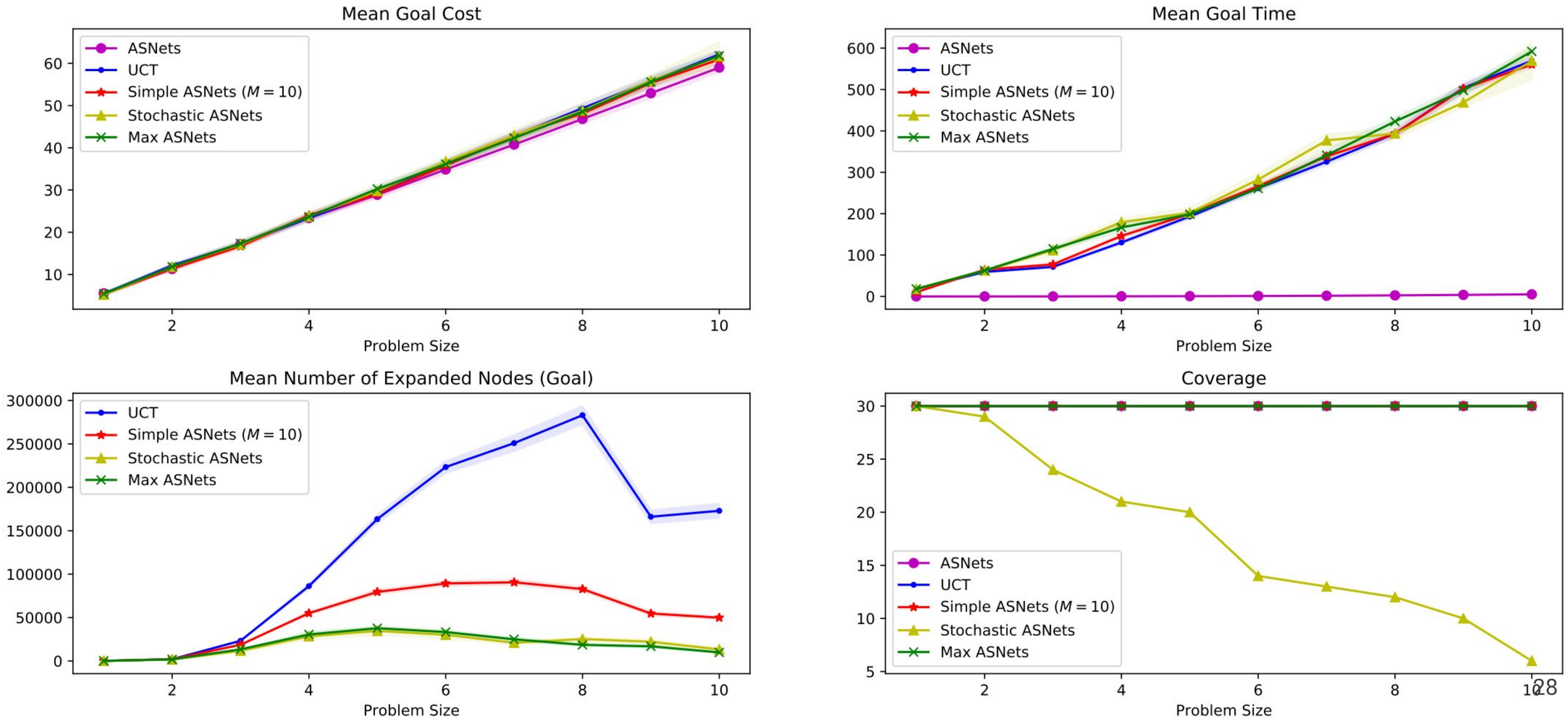


Triangle Tireworld

- One-way roads, goal is navigate from start to the goal
- Black nodes indicate locations with a spare tyre
- 50% probability that you will get a flat tyre when you move from one location to another
- Optimal policy is to navigate along the edge of the triangle to avoid dead ends



Triangle Tireworld - Results



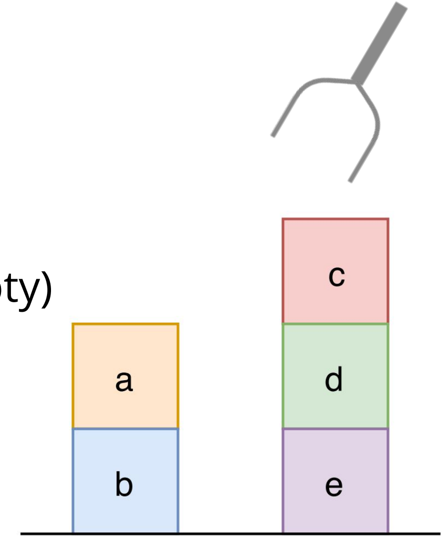
Action Schema Networks (ASNets)

- Neural Network Architecture inspired by CNNs
- Action Schemas

unstack ?x ?y

PRE: (on ?x ?y) ∧ (clear ?x) ∧ (handempty)

EFF: (not (on ?x ?y)) ∧ (holding ?x) ∧ (not (handempty)) ∧ ...



- Sparse Connections
  - “Action *a* affects proposition *p*”, and vice-versa
  - Only connect action and proposition modules if they appear in the action schema of the module.

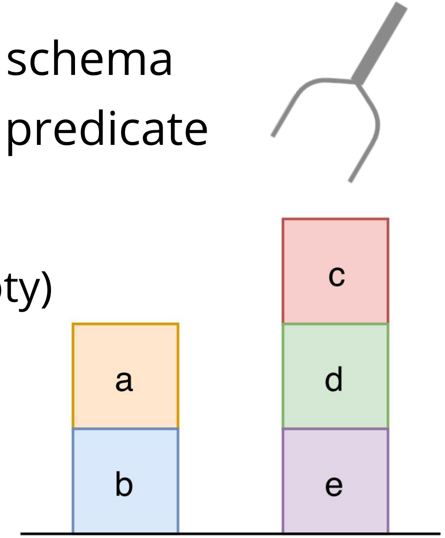
Action Schema Networks (ASNets)

- **Weight sharing.** In one layer, share weights between:
  - Action modules instantiated from the same action schema
  - Proposition modules that correspond to the same predicate

unstack ?x ?y

PRE: (on ?x ?y) ∧ (clear ?x) ∧ (handempty)

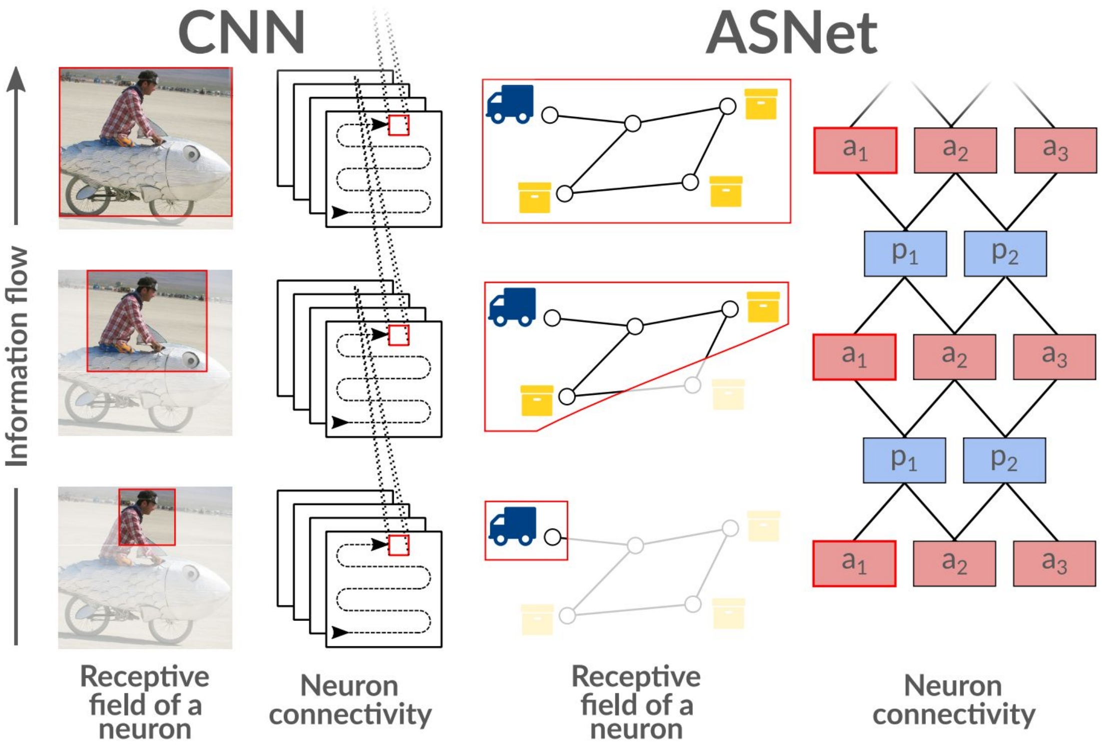
POST: (not (on ?x ?y)) ∧ (holding ?x) ∧ (not (handempty)) ∧ ...



Action modules for (unstack a b), (unstack c d), etc. share weights

Proposition modules for (on a b), (on c d), (on d e), etc. share weights

Action Schema Networks (ASNets)



How to overcome fixed receptive field? Use search!