Specifying Goals to Deep Neural Networks with Answer Set Programming

Forest Agostinelli, Rojina Panta, Vedant Khandelwal
University of South Carolina

Rojina Panta

Vedant Khandelwal

Outline

- Overview
- Heuristic function training
- Goal specification and reaching
- Results
- Future work

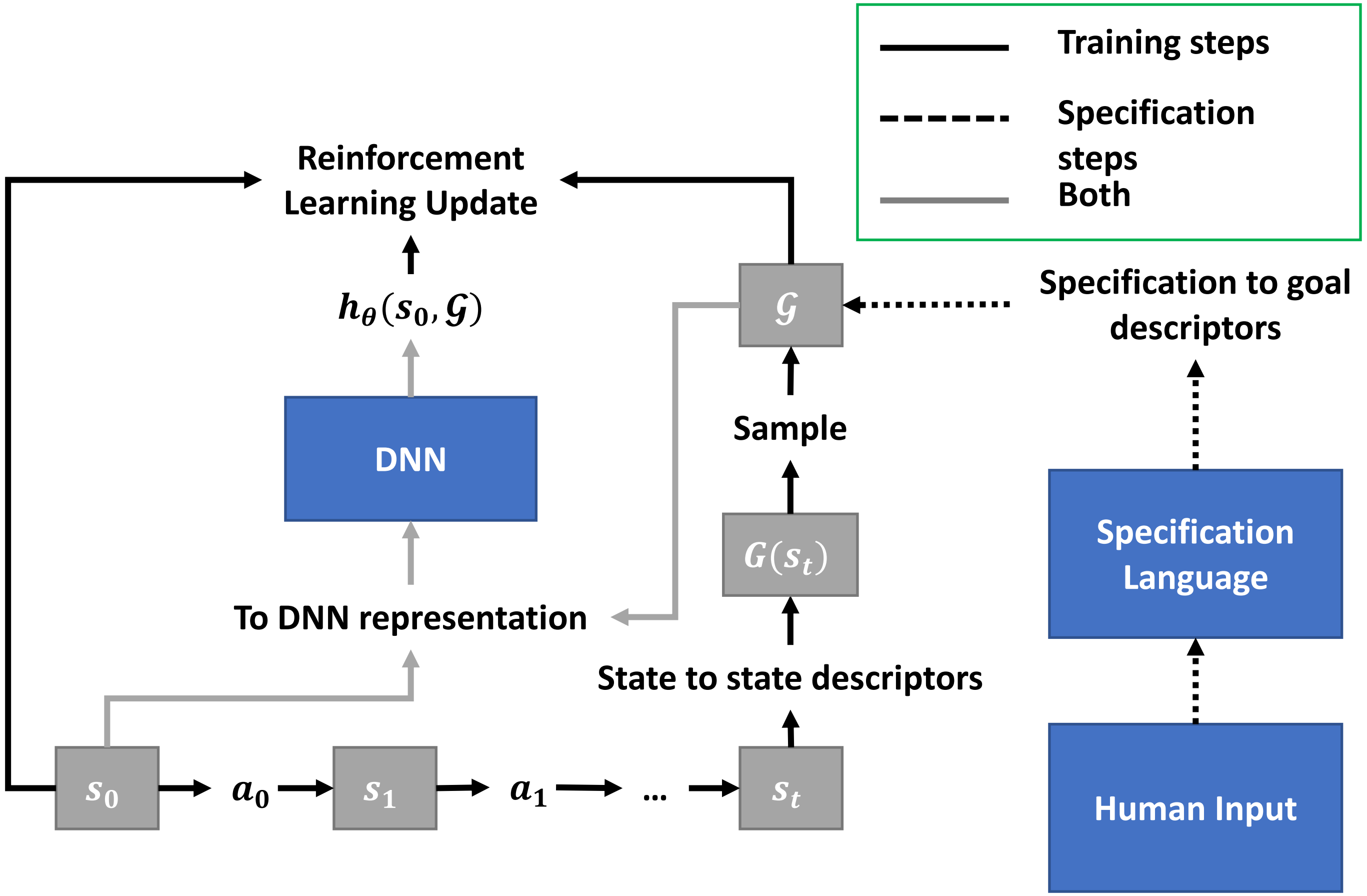
Motivation

- Deep reinforcement learning methods, such as DeepCubeA, can learn domain-specific heuristic functions in a largely domain-independent fashion
- Limitations
  - The goal is pre-determined
    - Specifying a new goal requires re-training the entire DNN
  - Hindsight experience replay can be used to generalize over start states and goal states
    - Must know the exact goal state, which is not always feasible
    - Cannot define a set of goal states using a high-level specification language
- **Desired solution**
  - **High-level specifications**
    - It should be possible to specify a goal (a set of states), without knowing the elements in the set
    - This will allow us to discover new states by finding a path to a currently unknown state that meets a given specification
  - **Flexible specification language**
    - The specification language should be able to represent diverse goals
  - **Goal agnostic training**
    - The training process should not have to be given any information about the goals it will see during testing
    - No re-training necessary
- Can be applied to specifying goals in applications such as chemical synthesis, quantum circuit design, manufacturing

Agostinelli, Forest, et al. "Solving the Rubik's cube with deep reinforcement learning and search." Nature Machine Intelligence 1.8 (2019): 356-363.

Solution Overview

- In our work
  - State descriptors: assignments of values to variables
  - Specification language: Answer set programming (ASP)
  - ASP will be used to describe goals at a high-level using formal logic and an answer set solver will be used to find assignments that represent the a subset of the goal



Outline

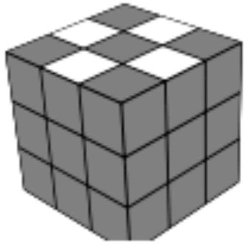
- Overview
- Heuristic function training
- Goal specification and reaching
- Results
- Future work

State Representation

- In a given pathfinding domain, there are V variables
  - A variable, x_i, can be assigned a single value from its (variable) domain, $D(x_i)$
- An assignment is an assignment is a set of assignments of values to variables $\{x_i = v_i\}$
  - All $v_i \in D(v_i)$
  - If x_i is not in the assignment then it is unassigned
- An assignment is a complete assignment iff all variables have been assigned values
- A state is a complete assignment
- For example, for the Rubik's cube, variables are stickers and values are their colors

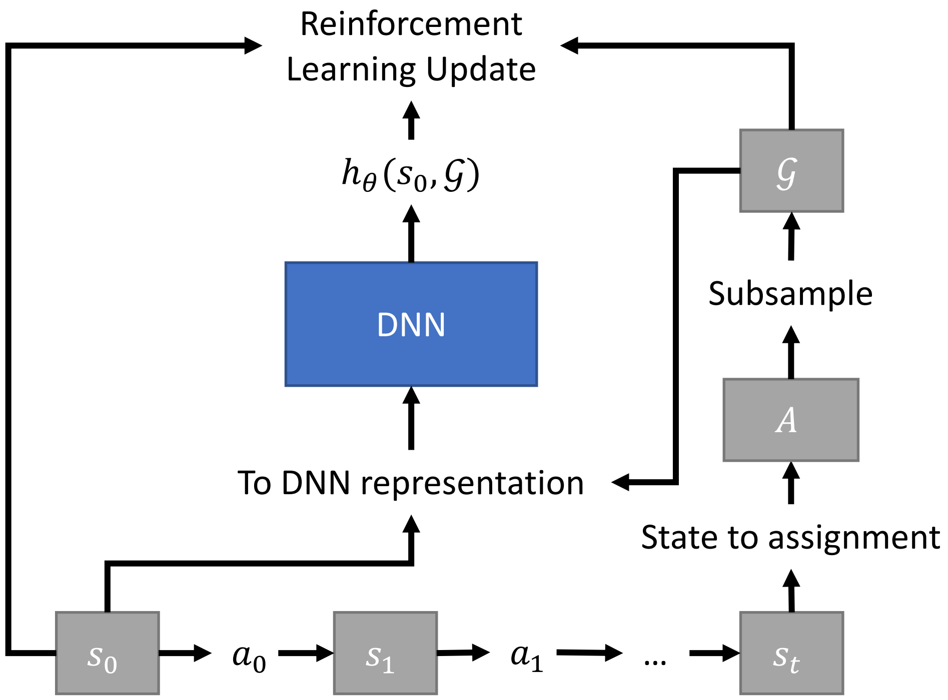
Goal Representation

- An assignment is a partial assignment iff at least one variable has not been assigned a value
- A goal is a complete or partial assignment
- An assignment, A, represents a set of states, $\mathcal{S}_A$
  - A complete assignment always represents a set of states of size 1
- A state, s, is in $\mathcal{S}_A$ iff $A \subseteq s$
  - In other words, all assignments in A are present in s
  - An empty assignment represents the set of all possible states
- For example, a visualization of an assignment for the “white cross” pattern for the Rubik’s cube and a state that is in the set of states represented by this assignment



Training

- Generate a start state
- Take a random walk whose length is somewhere between 0 and T
  - Future work could use artificial curiosity
- Convert the end state to its representation as an assignment
- Subsample to obtain a goal
- Convert this representation into one suitable for the DNN
  - One-hot representation
  - Graph
  - Etc.
- RL Update
  - $L(\theta) = \left(\min_{a}(c^{a}(s) + h_{\theta^{-}}(T(s,a),\mathcal{G})) - h_{\theta}(s,\mathcal{G})\right)^2$



Experiments

- ASP will be used to find assignments; therefore, we compare our method, DeepCubeA$_g$, to other methods capable of finding paths to goals that can be represented as assignments
- 500-1,000 test start and goal pairs
- 200 second time limit to solve test states
- **DeepCubeA$_g$**
  - Batch A* search
- **DeepCubeA**
  - Predefined goal
  - Batch A* search
- **Fast Downward Planner**
  - A* search
  - Goal count heuristic, fast forward heuristic, causal graph heuristic
- **PDBs**
  - Predefined goal
  - IDA* search

Agostinelli, Forest, et al. "Obtaining approximately admissible heuristic functions through deep reinforcement learning and A* search." *ICAPS PRL Workshop*. 2021.
Li, Tianhua, et al. "Optimal search with neural networks: Challenges and approaches." *Proceedings of the International Symposium on Combinatorial Search*. Vol. 15. No. 1. 2022.

Performance

- Canon: Canonical goal states
- Rand: Random assignment selected as goal
  - Can be as small as the empty assignment
  - Methods that require a pre-definied goal cannot be applied to this scenario without considerable overhead
- PDBs+: Also includes group theory knowledge
- DeepCubeA$_g$ consistently outperforms fastdownard in terms of percentage of states solved

Puzzle	Solver	Path Cost	% Solved	% Opt	Nodes	Secs	Nodes/Sec
RC (Canon)	PDBs$^+$	**20.67**	**100.00%**	**100.0%**	**2.05E+06**	**2.20**	**1.79E+06**
	DeepCubeA	21.50	**100.00%**	60.3%	6.62E+06	24.22	2.90E+05
	DeepCubeA$_g$	22.03	**100.00%**	35.00%	2.44E+06	41.99	5.67E+04
	FastDown (GC)	-	0.00%	0.0%	-	-	-
	FastDown (FF)	-	0.00%	0.0%	-	-	-
	FastDown (CG)	-	0.00%	0.0%	-	-	-
RC (Rand)	DeepCubeA$_g$	15.22	**99.40%**	-	1.91E+06	32.24	5.19E+04
	FastDown (GC)	7.18	32.80%	-	2.67E+06	13.79	1.41E+05
	FastDown (FF)	6.49	31.20%	-	4.87E+05	13.83	2.93E+04
	FastDown (CG)	7.85	33.80%	-	1.12E+06	11.62	5.81E+04
15-P (Canon)	PDBs	**52.02**	**100.00%**	**100.0%**	**3.22E+04**	**0.002**	**1.45E+07**
	DeepCubeA	52.03	**100.00%**	99.4%	3.85E+06	10.28	3.93E+05
	DeepCubeA$_g$	**52.02**	**100.00%**	**100.0%**	1.81E+05	2.61	6.94E+04
	FastDown (GC)	36.75	0.80%	0.80%	9.05E+07	102.11	8.66E+05
	FastDown (FF)	52.75	80.80%	24.80%	2.92E+06	42.11	6.93E+04
	FastDown (CG)	41.95	4.40%	1.20%	2.00E+07	80.58	2.47E+05
15-P (Rand)	DeepCubeA$_g$	**33.98**	**100.00%**	-	**1.11E+05**	**1.60**	**6.16E+04**
	FastDown (GC)	14.92	38.00%	-	1.61E+07	18.77	5.46E+05
	FastDown (FF)	32.66	89.20%	-	1.24E+06	17.39	5.65E+04
	FastDown (CG)	20.45	51.20%	-	3.90E+06	21.41	1.20E+05
24-P (Canon)	PDBs	**89.41**	**100.00%**	**100.00%**	8.19E+10	4239.54	**1.91E+07**
	DeepCubeA	89.49	**100.00%**	96.98%	6.44E+06	19.33	3.34E+05
	DeepCubeA$_g$	90.47	**100.00%**	55.24%	**3.38E+05**	**5.22**	6.48E+04
	FastDown (GC)	-	0.00%	0.00%	-	-	-
	FastDown (FF)	81.00	1.01%	0.40%	2.68E+06	89.84	2.91E+04
	FastDown (CG)	-	0.00%	0.00%	-	-	-
24-P (Rand)	DeepCubeA$_g$	66.28	**99.60%**	-	3.10E+05	4.91	6.16E+04
	FastDown (GC)	9.86	10.00%	-	9.54E+06	11.88	4.27E+05
	FastDown (FF)	26.35	26.00%	-	5.99E+05	19.57	2.41E+04
	FastDown (CG)	13.75	12.60%	-	1.42E+06	14.42	6.85E+04
Sokoban	DeepCubeA	32.88	**100.00%**	-	**5.01E+03**	2.71	1.84E+03
	DeepCubeA$_g$	**32.02**	**100.00%**	-	1.80E+04	0.95	1.79E+04
	FastDown (GC)	31.94	99.80%	-	3.17E+06	5.93	5.85E+05
	FastDown (FF)	33.15	**100.00%**	-	2.92E+04	**0.32**	**7.49E+04**
	FastDown (CG)	33.12	**100.00%**	-	4.43E+04	0.51	7.25E+04

Outline

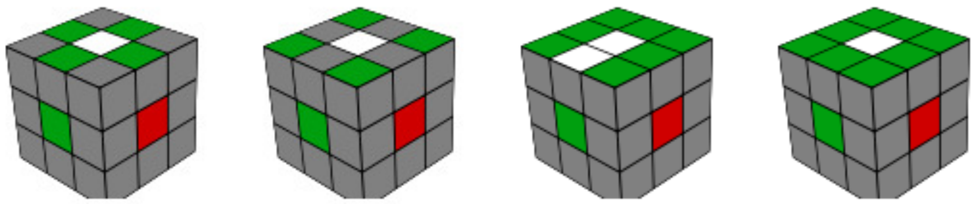
- Overview
- Heuristic function training
- Goal specification and reaching
- Results
- Future work

Answer Set Programming

- An answer set program (ASP) is a set of sentences in first order logic that defines a set of stable models (also known as answer sets)
  - We obtain assignments from stable models
- ASP solvers, such as clingo, can also make use of choice rules, aggregates, and classical negation
- $\alpha(\Pi)$ is the set of all possible assignments that can be obtained from Π
- A candidate state is a state that is a superset of some assignment in $\alpha(\Pi)$
- A goal state is a state that is in $\alpha(\Pi)$
- **Monotonic specification:** All candidate states are goal states
- **Non-monotonic specification:** Some candidate states are not goal states

ASP Specifications: Rubik's Cube Example

- Define basic background knowledge
  - Colors, faces, cubelets
  - Constraints: Cannot have two stickers of the same color on the same cubelet, cannot have two stickers from the same cubelet on opposite faces
- Given basic background knowledge, specifications often only require a few lines of code
  - `face_same(F) :- face_col(F, FCol), #count{Cbl : onface(Cbl, FCol, F)}=9.`
  - `canon_solved :- #count{F : face_same(F)}=6.`
- Our specifications contain combinations of common patterns
  - Note: the training procedure is unaware of what the specification will be at test time



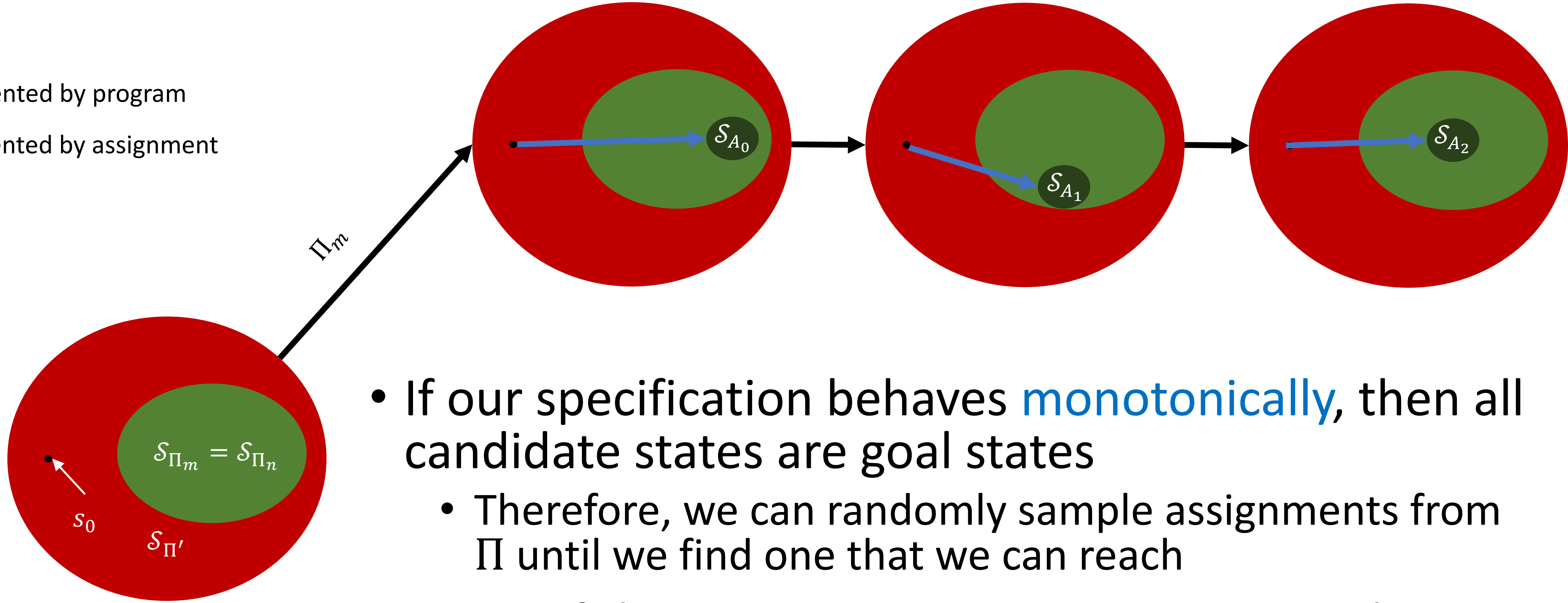
(a) Cross (b) X (c) Cup (d) Spot

Goal Reaching: Monotonic Specification

Π: Answer set program

$\mathcal{S}_\Pi$: set of states represented by program

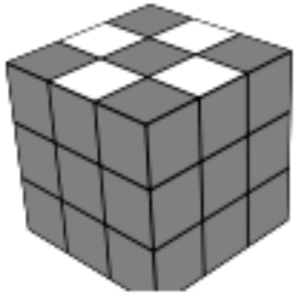
$\mathcal{S}_\text{A}$: set of states represented by assignment



- If our specification behaves monotonically, then all candidate states are goal states
  - Therefore, we can randomly sample assignments from Π until we find one that we can reach
- Some of these assignments may represent the empty set
- The answer set solver (we use clingo) used is agnostic to the cost of a shortest path

Handling Non-Monotonicity

- If negation as failure is used in a program, Π, then Π can exhibit non-monotonic behavior
  - A logic program is non-monotonic if some atoms that were previously derived can be retracted by adding new knowledge
  - Therefore, we can have a state that is a candidate state but not a goal state
- For example, a white cross with no yellow stickers on the white face
  - The assignment for this specification is just a white cross
  - However, there can be a state that is a specialization of this assignment, but has yellow on the white face

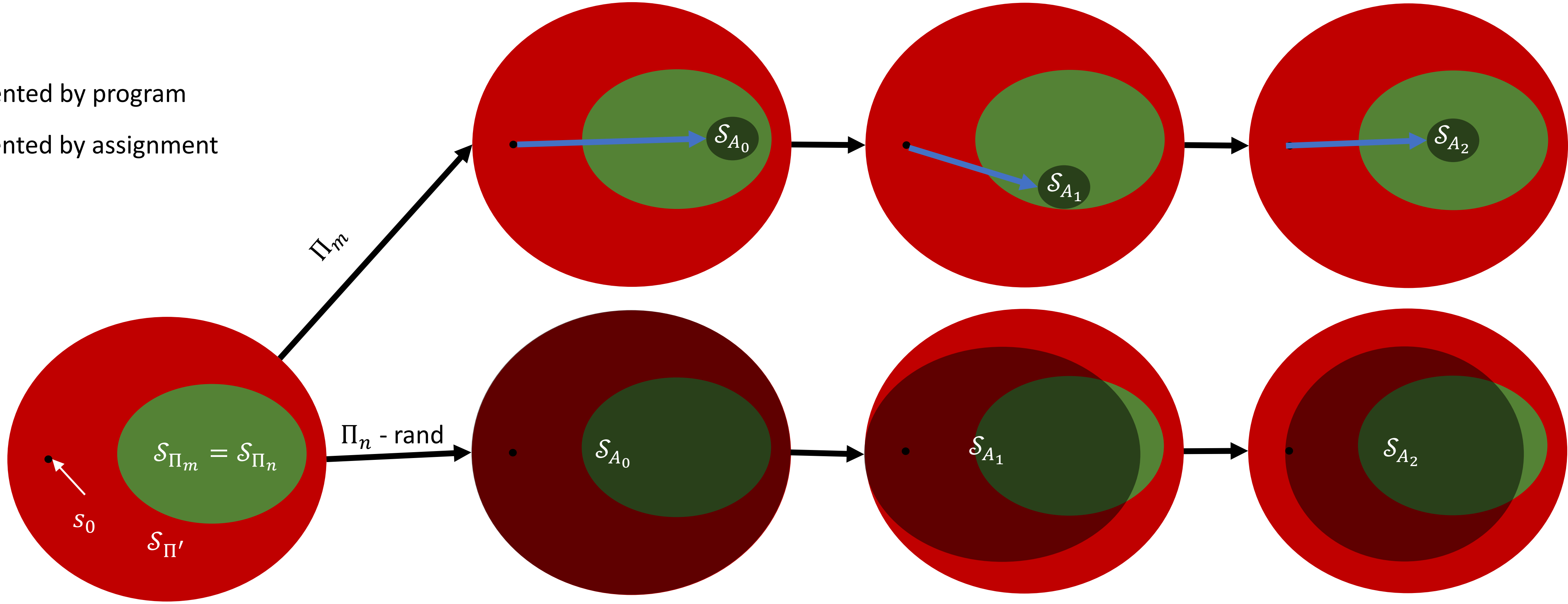


Goal Reaching: Non-monotonic

Π: Answer set program

$\mathcal{S}_\Pi$: set of states represented by program

$\mathcal{S}_A$: set of states represented by assignment



To reduce the size of candidate states while ensuring there is still at least one goal state, find another minimal assignment, A_2, such that

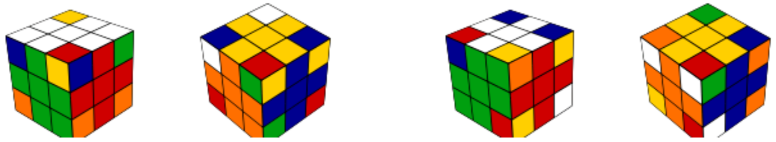
$$A \subset A_2$$
$$A_2 \in \alpha(\Pi)$$

Outline

- Overview
- Heuristic function training
- Goal specification and reaching
- Results
- Future work

Results

Goal	Path Cost	% Solved	# Models	Model Time	Search Time
Rubik's Cube (Canon)	24.41	100%	1	0.37	4.39
Rubik's Cube (Cross6)	13.11	100%	1	0.41	2.14
Rubik's Cube (Cup4)	24.33	100%	42.5	34.65	374.11
Rubik's Cube (CupSpot)	17.99	100%	27.68	38.66	241.08
Rubik's Cube (Checkers)	23.85	100%	1	0.49	4.2
Sokoban (Immov)	35.15	100%	6.37	6.83	16.16
Sokoban (BoxBox)	33.77	88%	1.89	0.58	6.08
Sokoban (AgentInBox)	34.42	77%	1.26	0.38	4.09



(a) Example 1 (b) Example 2

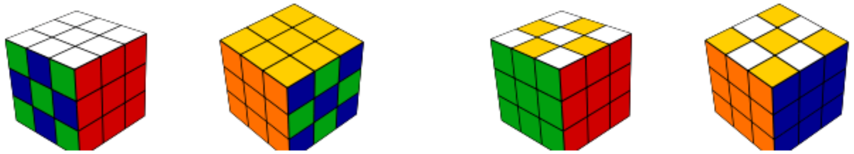
Cross6

(a) Example 1 (b) Example 2

Cup4

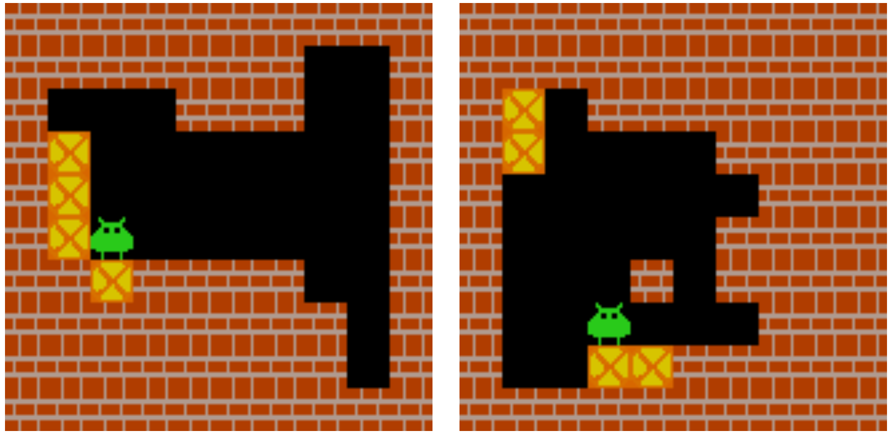
(a) Example 1 (b) Example 2

CupSpot



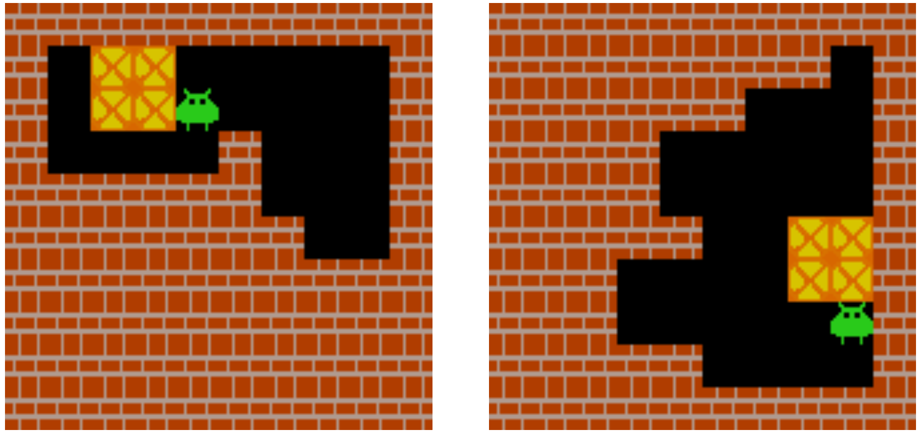
Checkers

Results

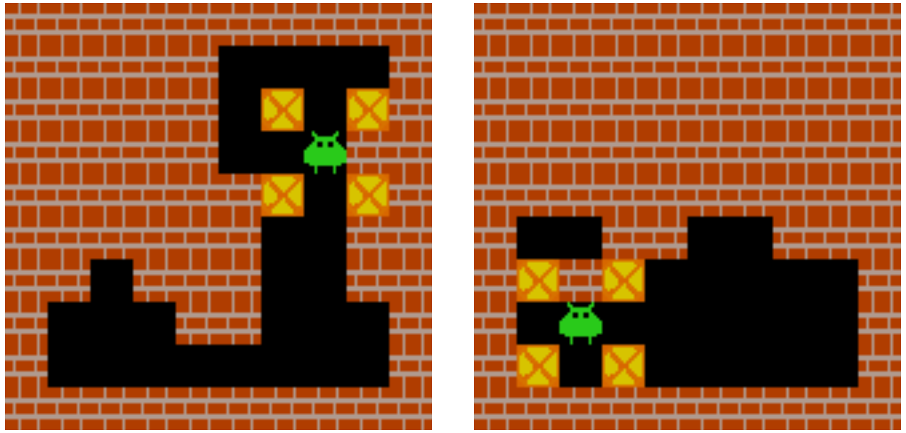


All boxes are immoveable

Goal	Path Cost	% Solved	# Models	Model Time	Search Time
Rubik's Cube (Canon)	24.41	100%	1	0.37	4.39
Rubik's Cube (Cross6)	13.11	100%	1	0.41	2.14
Rubik's Cube (Cup4)	24.33	100%	42.5	34.65	374.11
Rubik's Cube (CupSpot)	17.99	100%	27.68	38.66	241.08
Rubik's Cube (Checkers)	23.85	100%	1	0.49	4.2
Sokoban (Immov)	35.15	100%	6.37	6.83	16.16
Sokoban (BoxBox)	33.77	88%	1.89	0.58	6.08
Sokoban (AgentInBox)	34.42	77%	1.26	0.38	4.09



A box of boxes



Boxes at the four corners of the agent

Outline

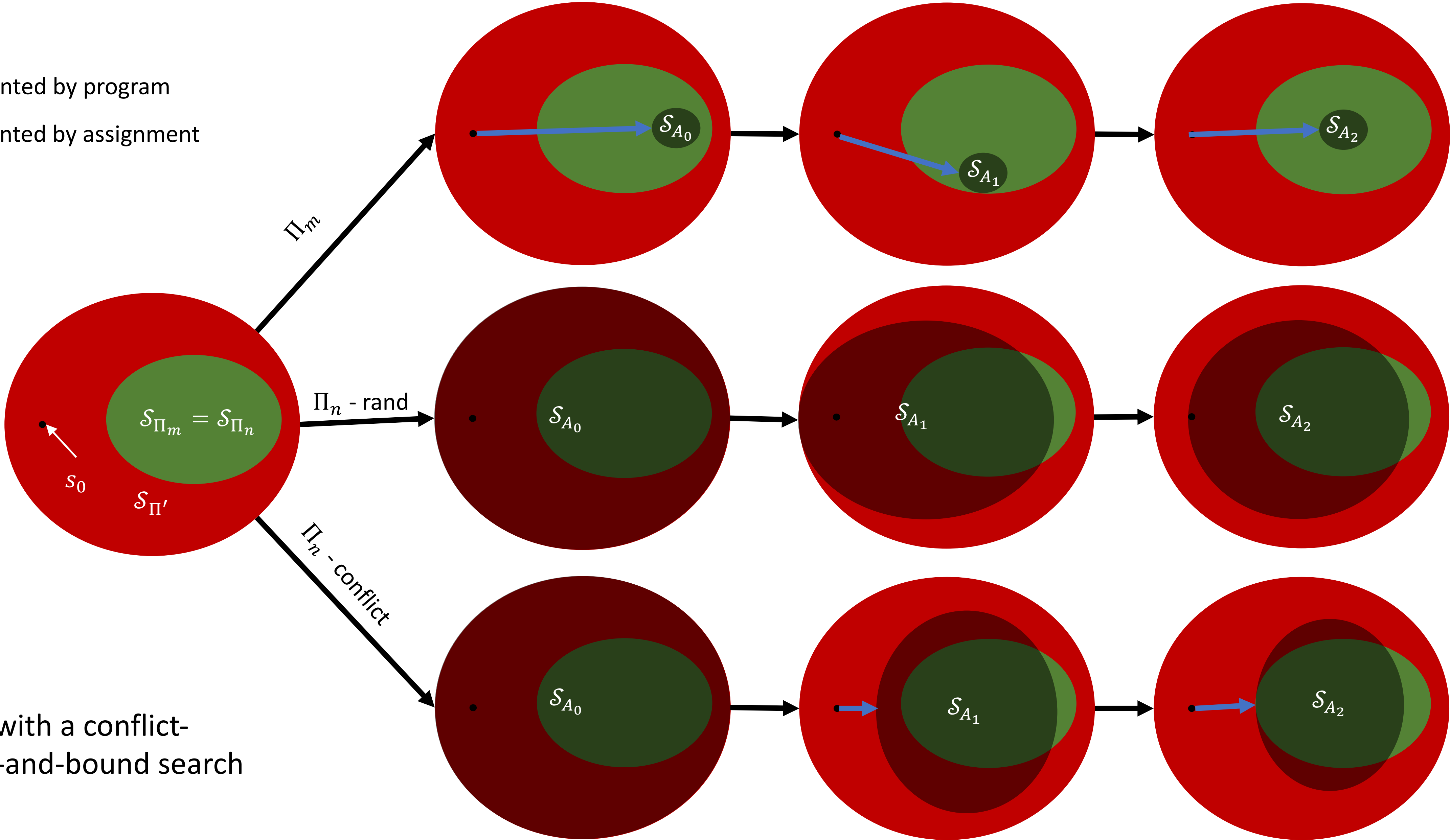
- Overview
- Heuristic function training
- Goal specification and reaching
- Results
- Future work

Goal Reaching: Non-monotonic

Π: Answer set program

$\mathcal{S}_\Pi$: set of states represented by program

$\mathcal{S}_\text{A}$: set of states represented by assignment

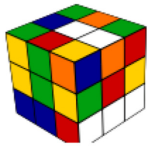


Combine this with a conflict-driven branch-and-bound search

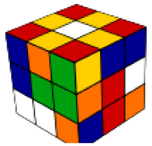
Results

Goal	SpecOp	Cost	%Solve	#Itr	#Assign	%reach	%not goal	$\frac{\text{Secs}}{\text{Spec}}$	$\frac{\text{Secs}}{\text{Path}}$	Secs
RC:∀diffCtrW	-	11.54	70	**3.34**	**33.43**	7.68	**0**	12.77	7.5	564.94
RC:¬∃sameCtrW	Rand	1.67	99	7.2	63.02	87.84	69.06	**0.06**	1.04	95.46
	Conflict	**1.26**	**100**	5.43	36.31	**99.34**	52.36	**0.06**	**0.07**	**5.98**
24p:r0SumEven	-	24.55	**100**	9.24	92.4	**100**	**0**	**0.2**	0.23	42.52
24p:¬r0SumOdd	Rand	3.16	**100**	4.27	33.6	**100**	38.71	**0.2**	**0.03**	6.64
	Conflict	**2.51**	**100**	**4.06**	**31.6**	**100**	22.13	0.21	0.04	**6.58**
24p:∀rSumEven	-	83.71	**100**	9.19	91.9	50.41	**0**	0.88	1.77	250.18
24p:¬∃rSumOdd	Rand	17.07	**100**	10.23	92.05	99.98	85.51	**0.1**	**0.08**	21.72
	Conflict	**12.87**	**100**	**8.66**	**77.1**	**100**	79.72	0.11	**0.08**	**17.08**

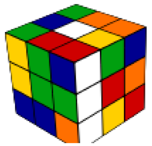
All stickers on the white face are different than the center sticker



Start



Mono: path cost 12



Non-mono: path cost 1

All rows sum to an even number

12	22	6	9	5
7	1	19	2	17
16	13	4	20	21
11	15	10	3	8
	14	18	24	23

Start

17	10	20	5	22
1	6	14	15	16
12	13	23		8
11	3	9	4	7
18	19	2	21	24

Mono: path cost 93

12	22	6	9	5
7	1	19	2	17
16	13	4	20	21
11	15	10		8
14	18	24	3	23

Non-mono: path cost 4

Questions?

- Code
  - Code available on GitHub
  - https://github.com/forestagostinelli/SpecGoal

Rojina Panta

Vedant Khandelwal

Email: foresta@cse.sc.edu
Website: https://cse.sc.edu/~foresta/

Agostinelli, F., Panta, R., & Khandelwal, V. Specifying Goals to Deep Neural Networks with Answer Set Programming. *ICAPS 2024*
Agostinelli, Forest. "A Conflict-Driven Approach for Reaching Goals Specified with Negation as Failure." *ICAPS 2024 HAXP Workshop*