Learning Features and Abstract Actions for Computing Generalized Plans

Blai Bonet[1] Guillem Francès[2] Hector Geffner[3]

[1]Universidad Simón Bolívar, Caracas, Venezuela
[2]University of Basel, Basel, Switzerland
[3]ICREA & Universitat Pompeu Fabra, Barcelona, Spain

Planning and Generalized Planning

- Planning is about solving **single planning instances**

- **Generalized planning** is about solving **multiple instances** at once

For example:

- Achieve goal $on(x, y)$ in Blocksworld (any number of blocks, any configuration)

- Go to target location (x^*, y^*) in empty square grid of any size

- Pick objects in grid (any number and location, any grid size)

Srivastava et al. 2008, B. et al. 2009, Hu & De Giacomo 2011, Illanes & McIlraith 2017, ...

Example: Plan for $clear(x)$ using Right Abstraction

- Get $clear(x)$, for designated block x, on **any Blocksworld instance**

- **Features:** $F = \{H, n\}$ where
  - H is **Boolean feature** that tells whether gripper is holding a block
  - n is **numerical feature** that counts number of blocks above x

- **Abstract actions:** $A_F = \{\text{Pick-above-}x, \text{Put-aside}\}$ given by
  - Pick-above-$x = \langle \neg H,\, n > 0;\; H,\, n{\downarrow} \rangle$
  - Put-aside $= \langle H;\; \neg H \rangle$

- **Solution:** If $\neg H \wedge n > 0$ then Pick-above-x ; If H then Put-aside

- **Computed** with off-the-shelf FOND planner

Can we learn the RIGHT abstraction?

Features for Generalized Planning: Requirements

Features required for solving collection $\mathcal{Q}$ of instances:

- Must be **general**; i.e. well defined on any state for any instance in $\mathcal{Q}$
- Must be **predictable**; i.e. effects of actions on features is predictable
- Must **distinguish** goal from non-goal states

Solving all instances in $\mathcal{Q}$ is mapped to solving **single FOND problem** over the features

FOND = Fully Observable Non-Deterministic

Learning Features and Abstract Actions using SAT

- **Input:**

– $\mathcal{S} =$ sample of **state transitions** (s, s') for some instances in $\mathcal{Q}$

– $\mathcal{F} =$ **pool of features** computed from primitive predicates in $\mathcal{Q}$

Learning Features and Abstract Actions using SAT

- **Input:**
  - $\mathcal{S}$ = sample of **state transitions** (s, s') for some instances in $\mathcal{Q}$
  - $\mathcal{F}$ = **pool of features** computed from primitive predicates in $\mathcal{Q}$

- **Propositional variables:**
  - $selected(f)$ for each feature f (f in "final set" F iff $selected(f)$ is true)
  - $D_1(s, t)$ iff selected features **distinguish states** s and t in sample $\mathcal{S}$
  - $D_2(s, s', t, t')$ iff selected features **distinguish transitions** (s, s') and (t, t') in $\mathcal{S}$

Learning Features and Abstract Actions using SAT

- **Input:**
  - $\mathcal{S}$ = sample of **state transitions** (s, s') for some instances in $\mathcal{Q}$
  - $\mathcal{F}$ = **pool of features** computed from primitive predicates in $\mathcal{Q}$

- **Propositional variables:**
  - $selected(f)$ for each feature f (f in "final set" F iff $selected(f)$ is true)
  - $D_1(s, t)$ iff selected features **distinguish states** s and t in sample $\mathcal{S}$
  - $D_2(s, s', t, t')$ iff selected features **distinguish transitions** (s, s') and (t, t') in $\mathcal{S}$

- **Formulas:**
  - $D_1(s, t) \iff \bigvee_f selected(f)$ (for f that make s and t different)
  - $D_2(s, s', t, t') \iff \bigvee_f selected(f)$ (for f that make (s, s') and (t, t') different)

Learning Features and Abstract Actions using SAT

- **Input:**
  - $\mathcal{S}$ = sample of **state transitions** (s, s') for some instances in $\mathcal{Q}$
  - $\mathcal{F}$ = **pool of features** computed from primitive predicates in $\mathcal{Q}$
- **Propositional variables:**
  - $selected(f)$ for each feature f (f in "final set" F iff $selected(f)$ is true)
  - $D_1(s, t)$ iff selected features **distinguish states** s and t in sample $\mathcal{S}$
  - $D_2(s, s', t, t')$ iff selected features **distinguish transitions** (s, s') and (t, t') in $\mathcal{S}$
- **Formulas:**
  - $D_1(s, t) \iff \bigvee_f selected(f)$ (for f that make s and t different)
  - $D_2(s, s', t, t') \iff \bigvee_f selected(f)$ (for f that make (s, s') and (t, t') different)
  - $\neg D_1(s, t)$ (for s and t such **only one** is goal)

Learning Features and Abstract Actions using SAT

- **Input:**
  - $\mathcal{S}$ = sample of **state transitions** (s, s') for some instances in $\mathcal{Q}$
  - $\mathcal{F}$ = **pool of features** computed from primitive predicates in $\mathcal{Q}$
- **Propositional variables:**
  - $selected(f)$ for each feature f (f in "final set" F iff $selected(f)$ is true)
  - $D_1(s, t)$ iff selected features **distinguish states** s and t in sample $\mathcal{S}$
  - $D_2(s, s', t, t')$ iff selected features **distinguish transitions** (s, s') and (t, t') in $\mathcal{S}$
- **Formulas:**
  - $D_1(s, t) \iff \bigvee_f selected(f)$ (for f that make s and t different)
  - $D_2(s, s', t, t') \iff \bigvee_f selected(f)$ (for f that make (s, s') and (t, t') different)
  - $\neg D_1(s, t)$ (for s and t such **only one** is goal)
  - $\bigwedge_{t'} D_2(s, s', t, t') \implies D_1(s, t)$ (for each (s, s') and t in $\mathcal{S}$)

Learning Features and Abstract Actions using SAT

- **Input:**
  - $\mathcal{S}$ = sample of **state transitions** (s, s') for some instances in $\mathcal{Q}$
  - $\mathcal{F}$ = **pool of features** computed from primitive predicates in $\mathcal{Q}$

- **Propositional variables:**
  - $selected(f)$ for each feature f (f in "final set" F iff $selected(f)$ is true)
  - $D_1(s, t)$ iff selected features **distinguish states** s and t in sample $\mathcal{S}$
  - $D_2(s, s', t, t')$ iff selected features **distinguish transitions** (s, s') and (t, t') in $\mathcal{S}$

- **Formulas:**
  - $D_1(s, t) \iff \bigvee_f selected(f)$ (for f that make s and t different)
  - $D_2(s, s', t, t') \iff \bigvee_f selected(f)$ (for f that make (s, s') and (t, t') different)
  - $\neg D_1(s, t)$ (for s and t such **only one** is goal)
  - $\bigwedge_{t'} D_2(s, s', t, t') \implies D_1(s, t)$ (for each (s, s') and t in $\mathcal{S}$)

- **Guarantee:** Theory $T(\mathcal{S}, \mathcal{F})$ is SAT iff there is **sound and complete** abstraction relative to sample $\mathcal{S}$ (abstraction is easily obtained from model)

Pool of Features

- Pool $\mathcal{F}$ obtained from primitive and newly defined predicates in $\mathcal{Q}$

- Numerical and Boolean features n and f defined from **unary predicates** $q(\cdot)$:
  - $n(s) = |\{x : s \models q(x)\}|$ (cardinality of set)
  - $f(s) = |\{x : s \models q(x)\}| > 0$ (whether set is empty or not)

- New unary predicates obtained with **concept grammar**

- "Distance notion" also defined with concept grammar using binary predicates

- Feature f has $cost(f)$ given by its "concept complexity"

- MaxSAT solver minimizes $\sum_{f:selected(f)} cost(f)$

We look for most economical abstraction!

Computational Workflow

For solving generalized problem $\mathcal{Q}$:

1. Sample set of transition $\mathcal{S}$ from some instances in $\mathcal{Q}$
2. Compute pool of features $\mathcal{F}$ from primitive predicates, grammar, and bounds
3. **MaxSAT** to find model of theory for $(\mathcal{S}, \mathcal{F})$ of min cost $\sum_{f:selected(f)} cost(f)$
4. Decode SAT model to extract abstraction
5. Solve abstraction with off-the-shelf FOND planner

Experimental Result: $\mathcal{Q}_{gripper}$

- **Training:** 2 instances with 4 and 5 balls respectively
- **Features learned ($|\mathcal{S}| = 403, |\mathcal{F}| = 130$):**
  - $X =$ "whether robby is in target room"
  - $B =$ "number of balls not in target room"
  - $C =$ "number of balls carried by robby"
  - $G =$ "number of empty grippers (available capacity)"
- **Abstract actions learned:**
  - Drop-ball-at-target $= \langle C > 0,\ X;\ C\!\downarrow, G\!\uparrow\rangle$
  - Move-to-target-fully-loaded $= \langle \neg X,\ C > 0,\ G = 0;\ X\rangle$
  - Move-to-target-half-loaded $= \langle \neg X,\ B = 0,\ C > 0,\ G > 0;\ X\rangle$
  - Pick-ball-not-in-target $= \langle \neg X,\ B > 0, G > 0;\ B\!\downarrow, G\!\downarrow, C\!\uparrow\rangle$
  - Leave-target $= \langle X,\ C = 0,\ G > 0;\ \neg X\rangle$
- FOND solved in 171.92 secs; MaxSAT time is 0.01 secs
- Plan solves instances for **any number of grippers, any number of balls**

Experimental Result: $\mathcal{Q}_{reward}$

- Pick rewards in grid with **blocked cells** (from *Towards Deep Symbolic RL, Garnelo, Arulkumaran, Shanahan, 2016*)

- STRIPS instances with predicates: $reward(\cdot), at(\cdot), blocked(\cdot), adj(\cdot, \cdot)$

- **Training:** 2 instances 4×4, 5×5, diff. dist. of blocked cells and rewards

- **Features learned ($|\mathcal{S}| = 568, |\mathcal{F}| = 280$):**

– $R =$ "number of remaining rewards"

– $D =$ "min distance to closest reward along unblocked path"

- **Abstract actions learned:**

– Collect $= \langle D = 0,\ R > 0;\ R{\downarrow}, D{\uparrow} \rangle$

– Move-to-closest-reward $= \langle R > 0,\ D > 0;\ D{\downarrow} \rangle$

- FOND solved in 1.36 secs; MaxSAT time is 0.01 secs

- Plan solves **any grid size, any number of rewards, any dist. of blocked cells**

Summary and Future

- **Inductive framework** for generalized planning that mixes **learning** and **planning**
  - Learner needs to learn abstraction (not plans)
  - Planner uses abstraction, transformed, to compute general plans

- **Number of samples** is small as learner only identifies features to be tracked

- Unlike purely learning approaches:
  - Features and policies are **transparent**
  - **Scope and correctness** of plans can be formally characterized

- Relation to **dimensionality reduction** and **embeddings** in ML/Deep Learning

FOND translator: `https://github.com/bonetblai/qnp2fond`