Learning Sketches for Decomposing Planning Problems into Subproblems of Bounded Width

Dominik Drexler\textsuperscript{1}  
Jendrik Seipp\textsuperscript{1}  
Hector Geffner\textsuperscript{2,1}

June 21, 2022

\textsuperscript{1}Linköping University, Linköping, Sweden,  
\textsuperscript{2}ICREA & Universitat Pompeu Fabra, Barcelona, Spain
Two important questions in planning (and RL) are:

1. What is a good language for representing the subgoal structure of planning tasks?
   → Policy sketches [Bonet and Geffner, 2021]
2. How to learn common subgoal structure of a family of tasks?
   → In this paper
Policy Sketches

- Policy sketches (sketches) are simple and powerful [Drexler et al., 2021]
- Sketch splits problems into subproblems of bounded width in such a way that problems become solvable in polynomial time by the SIW$_R$ algorithm
- Semantics in terms of what subgoal to achieve
- Not so much: more complex languages such as HTN or LTL
Outline

- Example
- Sketches
- Learning sketches of width $k$
- Experimental results
### Example Sketch for the Delivery Domain

#### Features $\Phi$

- $H$: holding a package?
- $n$: number of undelivered packages
- $p$: distance to nearest package
- $t$: distance to target cell

#### Rules $R_\Phi$

$$\{ n > 0 \} \mapsto \{ n \downarrow \} ; \text{deliver misplaced package}$$
## Example Sketch for the Delivery Domain (cont.)

### Rules $R_\Phi$; 2-width sketch

\[ \{n > 0\} \mapsto \{n\downarrow\} \quad ; \text{deliver misplaced package} \]

### Rules $R_\Phi$; 1-width sketch

\[ \{\neg H\} \mapsto \{H\} \quad ; \text{pick pkg} \]
\[ \{H, n > 0\} \mapsto \{H?, n\downarrow\} \quad ; \text{deliver pkg} \]

### Rules $R_\Phi$; 0-width sketch or general policy [Francès et al., 2021]

\[ \{\neg H, p > 0\} \mapsto \{p\downarrow, t?\} \quad ; \text{go to nearest pkg} \]
\[ \{\neg H, p = 0\} \mapsto \{H\} \quad ; \text{pick it up} \]
\[ \{H, t > 0\} \mapsto \{t\downarrow\} \quad ; \text{go to target} \]
\[ \{H, n > 0, t = 0\} \mapsto \{H?, n\downarrow, p?\} \quad ; \text{deliver pkg} \]
Syntax and Semantics of Sketches

• Syntax:
  - **Sketch rule** has form \( C \mapsto E \)
  - For **Boolean feature** \( p \) and **numerical feature** \( n \), we can have
    - \( p, \neg p, n > 0, n = 0 \) in \( C \)
    - \( p, \neg p, p?, n\uparrow, n\downarrow, n? \) in \( E \)

• Semantics:
  - State pair \((s, s')\) **satisfies** sketch rule \( C \mapsto E \) if
    1. \( s \) satisfies \( C \), and
    2. \((s, s')\) satisfied \( E \)
Sketch Width

- Sketch $R$ splits problem $P$ in $Q$ into collection of subproblems $P[s, G_R(s)]$ where
  - initial state $s$ is reachable state $s$ in $P$, and
  - (sub) goal states $G_R(s) = \{s' \mid (s, s')$ satisfies sketch rule or $s'$ is goal$\}$

- **Width of problem** $w(P[s, G])$ is exploitable measure for difficulty of achieving goal $G$ from initial state $s$ [Lipovetzky and Geffner, 2012]

- **Width of sketch** $R$ over $Q$ is $\max\{w(P[s, G_R(s)]) \mid s \in P, P \in Q\}$

- **Theorem**: Any $P$ in $Q$ solvable with $\exp(k)$ resources if sketch has width $k$ and sketch is terminating
Example Sketch for the Floortile Domain

Features $\Phi$
- $n$: number of painted tiles
- $S$: state is solvable?

Rules $R_\Phi$
$$\{S, n > 0\} \mapsto \{n\downarrow\} \quad ; \text{deliver misplaced package}$$

Theorem
The sketch $R_\Phi$ for the Floortile domain is terminating and has width 2.
Learning Sketches as Combinatorial Optimization

- **Given:**
  - Planning tasks $P_1, \ldots, P_n$
  - Feature pool $\mathcal{F}$
  - Sketch width $k$
  - Maximum number of rules $m$

- **Find:** sketch $R_\Phi$ over features $\Phi \subseteq \mathcal{F}$ with $m$ rules that
  1. results in subproblems $P[s, G_R(s)]$ of width $\leq k$,
  2. is acyclic in each $P_i$ (approximation of termination), and
  3. has minimum feature complexity, i.e., $\sum_{f \in \Phi} \text{complexity}(f)$
Learning Sketches as Combinatorial Optimization: Details

- Select $R_\Phi$ consisting of $m$ rules
  - **Construct rules:** $\text{cond}(i, f, v), \text{eff}(i, f, v)$, use unique $v$, implies $\text{select}(f)$
  - **Ensure compatibility:** $\text{sat_rule}(s, s', i)$ iff $(s, s')$ compatible with rule $i$
- Ensure that $R_\Phi$ is terminating
  - **Ensure termination:** collection of rules $i = 1, \ldots, m$ is terminating
- Ensure that $R_\Phi$ has sketch width $\leq k$
  - **Select subgoal tuples:** $\forall t \text{subgoal}(s, t)$, each alive $s$ has some subgoal $t$
  - **Select subgoal states:** $\text{subgoal}(s, t)$ iff $\land_{s'} \text{subgoals}(s, t, s')$
  - **Ensure compatible rule:** $\text{subgoals}(s, t, s')$ implies $\forall_{i=1, m} \text{sat_rule}(s, s', i)$
  - **Ensure deadend free:** $\text{sat_rule}(s, s'', i)$ implies $\forall t : d(s, t) < d(s, s'') \text{subgoal}(s, t)$
  - **Ensure optimal width:** $\text{sat_rule}(s, s', i)$ implies $\forall t : d(s, t) \leq d(s, s') \text{subgoal}(s, t)$
- Implementation as answer set program in Clingo [Gebser et al., 2012]
**Experimental Results of Learning Sketches**

**Table 1:** Learning results for width bound $k = 1$, maximum feature complexity of 8, time limit of 7 days, and memory limit of 384 GiB.

| Domain       | Memory | Time  | $|\mathcal{P}|$ | States | $|\mathcal{F}|$ | max. feature complexity | $|\Phi|$ | $|R|$ |
|--------------|--------|-------|-------------|--------|-------------|-------------------------|--------|------|
| Blocks-clear | 1      | 4     | 1           | 22     | 233         | 4                       | 1      | 1    |
| Blocks-on    | 9      | 105   | 1           | 22     | 1011        | 4                       | 2      | 2    |
| **Childsnack** | **122** | **228k** | **3** | **792** | **629** | **6** | **4** | **5** |
| Delivery     | 17     | 521   | 1           | 96     | 474         | 4                       | 2      | 2    |
| Gripper      | 3      | 60    | 1           | 28     | 301         | 4                       | 2      | 2    |
| Miconic      | 1      | 5     | 1           | 32     | 119         | 2                       | 2      | 2    |
| Reward       | 1      | 4     | 1           | 12     | 210         | 2                       | 1      | 1    |
| Spanner      | 3      | 22    | 1           | 74     | 424         | 5                       | 1      | 1    |
| **Visitall** | **1**  | **1** | **1**       | **3**  | **10**      | **2**                   | **1**  | **1** |
# Experimental Results of Testing the Learned Sketches

Table 2: Testing results for time limit 30 minutes and 6 GiB memory.

<table>
<thead>
<tr>
<th>Domain</th>
<th>( w = 0 )</th>
<th>( w = 1 )</th>
<th>( w = 2 )</th>
<th>LAMA</th>
<th>BFWS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S</td>
<td>T</td>
<td>S</td>
<td>T</td>
<td>S</td>
</tr>
<tr>
<td>Blocks-clear (30)</td>
<td>30</td>
<td>3</td>
<td>30</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>Blocks-on (30)</td>
<td>30</td>
<td>3</td>
<td>30</td>
<td>6</td>
<td>30</td>
</tr>
<tr>
<td>Childsnack (30)</td>
<td>–</td>
<td>–</td>
<td>30</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>Delivery (30)</td>
<td>–</td>
<td>–</td>
<td>30</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>Gripper (30)</td>
<td>30</td>
<td>4</td>
<td>30</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>Miconic (30)</td>
<td>–</td>
<td>–</td>
<td>30</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>Reward (30)</td>
<td>30</td>
<td>4</td>
<td>30</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>Spanner (30)</td>
<td>30</td>
<td>3</td>
<td>30</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>Visitall (30)</td>
<td>26</td>
<td>1360</td>
<td>30</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>#Domains solved (9)</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
Formal Properties of Learned Sketches

- Sketch width $\leq k$ only guaranteed for training instances $P_1, \ldots, P_n$
- However, sketch width $\leq k$ across family of tasks $Q$ was proven
## Learned Sketch for the Visitall Domain

### Features $\Phi$
- $n$: number visited locations

### Rules $R_\Phi$

<table>
<thead>
<tr>
<th>${} \mapsto {n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>; visit a new location</td>
</tr>
</tbody>
</table>

### Theorem

*The sketch $R_\Phi$ for the Visitall domain is acyclic and has width 1.*
Learned Sketch for the Childsnack Domain

**Features $\Phi$**

- $sk$: number of *sandwiches* at the *kitchen*,
- $ua$: number of *unserved* and *allergic* children,
- $gfs$: number of *gluten-free sandwiches*, and
- $s$: number of *served* children.

**Rules $R_\Phi$**

- $\emptyset \mapsto \{gfs\uparrow\}$; make gluten free sandwiches
- $\emptyset \mapsto \{sk\downarrow\}$; move sandwiches from kitchen on tray
- $\emptyset \mapsto \{ua\downarrow\}$; serve gluten-free sandwich to allergic children
- $\{ua = 0\} \mapsto \{sk\uparrow\}$; make any sandwich if all allergic children are served
- $\{ua = 0\} \mapsto \{s\uparrow\}$; serve arbitrary sandwich if all allergic children are served

**Theorem**

*The sketch $R_\Phi$ for the Childsnack domain is acyclic and has width 1.*
• Sketches with bounded width ensure poly time solutions and hence only possible for tractable domains
• Learning implementation in Clingo does not scale up in all domains, e.g., Barman, Schedule, Floortile, Driverlog
• Feature pool assumes first-order language to describe states (PDDL)
Conclusions and Future Work

• First general method for learning how to decompose planning problems into subproblems with a polynomial complexity that is controlled with a parameter

• Future work:
  • From sketches to hierarchies
  • From PDDL inputs/states to other state languages

