Planning with Incomplete User Preferences and Domain Models

Tuan Anh Nguyen

Graduate Committee Members:
Subbarao Kambhampati (Chair)
Chitta Baral
Minh B. Do
Joohyung Lee
David E. Smith

MOTIVATION

Automated Planning Research:

- Actions
  - Preconditions
  - Effects
    - Deterministic
    - Non-deterministic
    - Stochastic
- Initial situation
- Goal conditions
- What a user wants about plans
- Find a (best) plan!

In practice…

- Action models are not available upfront
  - Cost of modeling
  - Error-prone
- Users usually don't exactly know what they want
  - Always want to see more than one plan

Planning with incomplete user preferences and domain models

Preferences in Planning – Traditional View

- Classical Model: “Closed world” assumption about user preferences.
  - All preferences assumed to be fully specified/available

Full Knowledge of Preferences

Two possibilities

- If no preferences specified —then user is assumed to be *indifferent*. Any single feasible plan considered acceptable.
- If preferences/objectives are specified, find a plan that is optimal w.r.t. specified objectives.

Either way, solution is a *single* plan

Preferences in Planning—Real World

- Real World: Preferences **not** fully known

Full Knowledge of Preferences is *lacking*

Unknown preferences

- For all we know, user may care about every thing --- the flight carrier, the arrival and departure times, the type of flight, the airport, time of travel and cost of travel…

Partially known

- We know that users cares only about travel time and cost. But we don't know how she combines them…

Domain Models in Planning – Traditional View

- Classical Model: "Closed world" assumption about action descriptions.
  - Fully specified preconditions and effects
  - Known exact probabilities of outcomes

Full Knowledge of domain models

```
pick-up
:parameters (?b - ball ?r - room)
:precondition
   (and (at ?b ?r) (at-robot ?r) (free-gripper))
:effect
   (and (carry ?b) (not (at ?b ?r)) (not (free-gripper)))
```

Domain Models in Planning – (More) Practical View

- Completely modeling the domain dynamics
  - Time consuming
  - Error-prone
  - Sometimes impossible

- What does it mean by planning with incompletely specified domain models?
  - Plan could fail! Prefer plans that are more likely to succeed…
  - How to define such a solution concept?

Problems and Challenges

- **Incompleteness representation**
- **Solution concepts**
- **Planning techniques**

Dissertation overview

"Model-lite" Planning

Preference incompleteness

- Representation
- Solution concept
- Solving techniques

Domain incompleteness

- Representation
- Solution concept
- Solving techniques

Dissertation overview

“Model-lite” Planning

Preference incompleteness

- **Representation:** two levels of incompleteness
  - User preferences exist, but totally unknown
  - Partially specified
    - Complete set of plan attributes
    - Parameterized value function, unknown trade-off values
- **Solution concept:** plan sets
- **Solving techniques:** synthesizing high quality plan sets

Domain incompleteness

- **Representation**
  - Actions with possible preconditions / effects
  - Optionally with weights for being the real ones
- **Solution concept:** “robust” plans
- **Solving techniques:** synthesizing robust plans

Dissertation Overview

"Model-lite" Planning

Preference incompleteness

- **Representation:** two levels of incompleteness
  - User preferences exist, but totally unknown
  - Partially specified
    - Full set of plan attributes
    - Parameterized value function, unknown trade-off values
- **Solution concept:** plan sets with quality
- **Solving techniques:** synthesizing quality plan sets

⇨

- Distance measures w.r.t. base-level features of plans (actions, states, causal links)
- CSP-based and local-search based planners

⇨

- IPF/ICP measure
- Sampling, ICP and Hybrid approaches

Dissertation overview

"Model-lite" Planning

Preference incompleteness	Domain incompleteness

Publication

- *Domain independent approaches for finding diverse plans*. **IJCAI (2007)**
- *Planning with partial preference models*. **IJCAI (2009)**
- *Generating diverse plans to handle unknown and partially known user preferences*. **AIJ 190 (2012)**

(with Biplav Srivastava, Subbarao Kambhampati, Minh Do, Alfonso Gerevini and Ivan Serina)

Publication

- *Assessing and Generating Robust Plans with Partial Domain Models*. **ICAPS-WS (2010)**
- *Synthesizing Robust Plans under Incomplete Domain Models*. **AAAI-WS(2011), NIPS (2013)**
- *A Heuristic Approach to Planning with Incomplete STRIPS Action Models*. **ICAPS (2014)**

(with Subbarao Kambhampati, Minh Do)

Planning with Incomplete Domain Models

REVIEW: STRIPS

- *Predicate set **R***: clear(x – object), on-table(x – object), on(x – object, y – object), holding(x – object), hand-empty

- *Operators **O***:
  - **Name** (signature): pick-up(x – object)
  - **Preconditions:** hand-empty, clear(x)
  - **Effects:** ~hand-empty, holding(x), ~clear(x)

- A single complete model!

Planning Problem with STRIPS

- Set of typed objects $\{o_1, \ldots, o_k\}$
  - Together with predicate set P, we have a set of grounded propositions F
  - Together with operators O, we have a set of grounded actions A
- Initial state: $I \in F$
- Goals: $G \subseteq F$

Planning Problem with STRIPS (2)

- **Find**: a plan π achieves G starting from I: $\gamma(\pi, I) \supseteq G$.

- Transition function:
  - $\gamma(\langle a \rangle, s) = s \cup Add(a) \setminus Del(a)$ for applying $a \in A$ in $s \subseteq F$ s.t. $Pre(a) \subseteq s$.
  - Applying $\pi = \langle a_1, \dots, a_n \rangle$ at state s: $\gamma(\pi, s) = \gamma(\langle a_n \rangle, \gamma(\langle a_2, \dots, a_{n-1} \rangle, s))$

Incomplete Domain Models

- Predicate set $\boldsymbol{R}$: clear(x – object), on-table(x – object), on(x – object, y – object), holding(x – object), hand-empty, light(x – object), dirty(x – object)
- Operators $\boldsymbol{O}$
  - Name (signature): pick-up(x – object)
  - Preconditions: hand-empty, clear(x)
  - Possible preconditions: light(x)
  - Effects: ~hand-empty, holding(x), ~clear(x)
  - Possible effects: dirty(x)

Incompleteness in deterministic domains

≠

Stochastic domains

- Incomplete domain $\widetilde{\boldsymbol{D}} = \langle \boldsymbol{R}, \boldsymbol{O} \rangle$
  - At “schema” level with typed variables (no objects)
  - With K “annotations”, we have 2^K possible complete models, **one of which is the true model**.

Planning Problem with Incomplete domain

- Set of typed objects $\{o_1, \dots, o_k\}$
  - Together with predicate set P, we have a set of grounded propositions F
  - Together with operators O, we have a set of grounded actions A
- Initial state: $I \in F$
- Goals: $G \subseteq F$
- **Find**: a plan π "achieves" G starting from I
  - An ill-defined solution concept!
  - Need a definition for "goal achievement"

Transition Function

- Under $\widetilde{\boldsymbol{D}}$, applying $\boldsymbol{\pi}$ in $\boldsymbol{s}$ results in a set of possible states:

$$\boldsymbol{\gamma}(\boldsymbol{\pi}, \boldsymbol{s}) = \bigcup_{\boldsymbol{D}_i \in \ll \widetilde{\boldsymbol{D}} \gg} \boldsymbol{\gamma}^{\boldsymbol{D}_i}(\boldsymbol{\pi}, \boldsymbol{s})$$

- The probability of ending up in $\boldsymbol{s}' \in \boldsymbol{\gamma}(\boldsymbol{\pi}, \boldsymbol{s})$ is equal to

$$\sum_{\boldsymbol{D}_i \in \ll \widetilde{\boldsymbol{D}} \gg,\, \boldsymbol{s}' = \boldsymbol{\gamma}^{\boldsymbol{D}_i}(\boldsymbol{\pi}, \boldsymbol{s})} \boldsymbol{Pr}(\boldsymbol{D}_i)$$

where $\boldsymbol{Pr}(\boldsymbol{D}_i)$ is the probability of $\boldsymbol{D}_i$ being the true model.

Transition Function

$\underline{\boldsymbol{\gamma^D(\langle a \rangle, s)}}$:

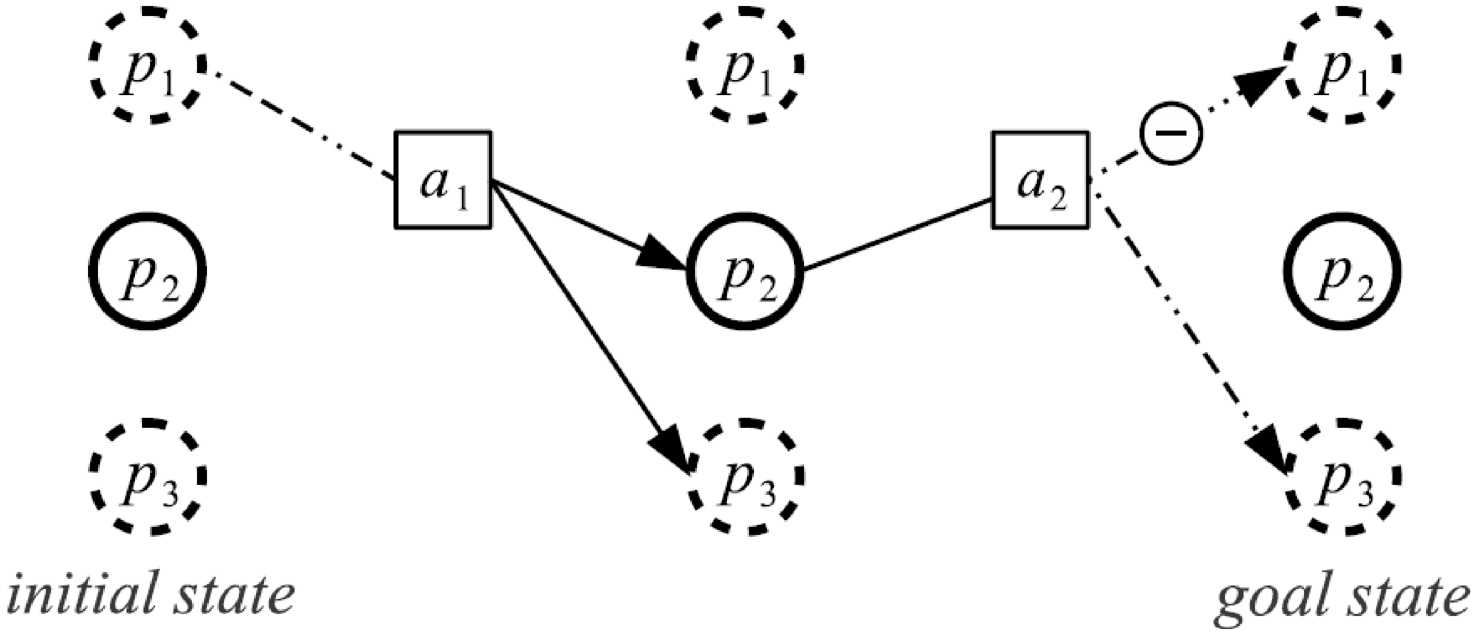
- STRIPS Execution (SE):

$$\gamma_{SE}^{D}(\langle a \rangle, s) = \begin{cases} s \setminus Del^D(a) \cup Add^D(a), & if\ Pre^D(a) \subseteq s \\ s_\perp = \{\perp\}, & otherwise \end{cases}$$

$$\perp \notin F \qquad Pre^D(a) \nsubseteq s_\perp, G \nsubseteq s_\perp$$

- Generous Execution (GE):

$$\gamma_{GE}^{D}(\langle a \rangle, s) = \begin{cases} s \setminus Del^D(a) \cup Add^D(a), & if\ Pre^D(a) \subseteq s \\ s, & otherwise \end{cases}$$



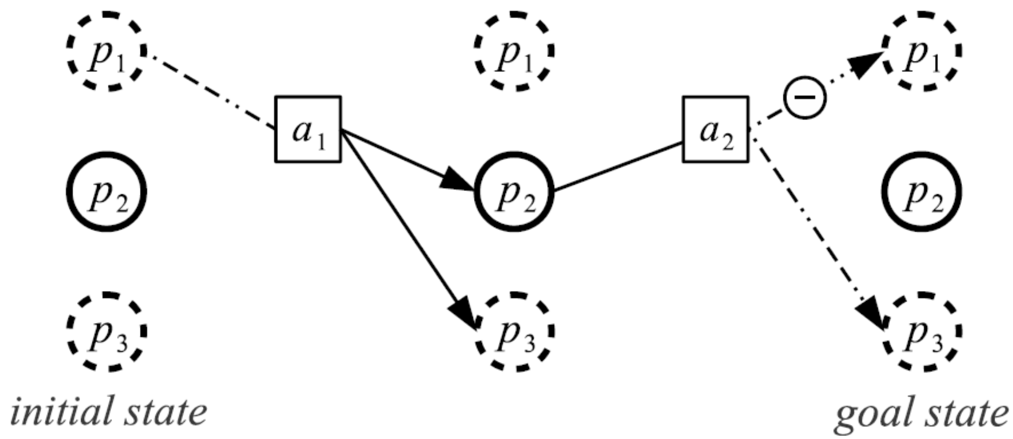
Candidate models	1	2	3	4	5	6	7	8
a_1 **relies on** p_1	yes	yes	yes	yes	no	no	no	no
a_2 **adds** p_3	yes	yes	no	no	yes	yes	no	no
a_2 **deletes** p_1	yes	no	yes	no	yes	no	yes	no
Plan status – GE semantics	succeed	succeed	fail	fail	succeed	succeed	succeed	succeed
Plan status – SE semantics	fail	fail	fail	fail	succeed	succeed	succeed	succeed

- Proposition set $F = \{p_1, p_2, p_3\}$
- Initial state $I = \{p_2\}$
- Goal $G = \{p_3\}$

A Measure for Plan Robustness

- Naturally, we prefer plan that succeeds in as many complete models as possible

$$R(\pi) = \frac{|\Pi|}{2^K}$$



$$\boldsymbol{R_{SE}(\pi) \leq R_{GE}(\pi)}$$

$$\mathrm{R_{GE}}(\pi) = 6/8$$

$$\mathrm{R_{GE}}(\pi) = 4/8$$

Candidate models	1	2	3	4	5	6	7	8
a_1 relies on p_1	yes	yes	yes	yes	no	no	no	no
a_2 adds p_3	yes	yes	no	no	yes	yes	no	no
a_2 deletes p_1	yes	no	yes	no	yes	no	yes	no
Plan status – GE semantics	succeed	succeed	fail	fail	succeed	succeed	succeed	succeed
Plan status – SE semantics	fail	fail	fail	fail	succeed	succeed	succeed	succeed

A bit more general…

- Predicate set ***R***: clear(x – object), on-table(x – object), on(x – object, y – object), holding(x – object), hand-empty, light(x – object), dirty(x – object)
- Operators ***O***
  - Name (signature): pick-up(x – object)
  - Preconditions: hand-empty, clear(x)
  - Possible preconditions: light(x) with a weight of 0.8
  - Effects: ~hand-empty, holding(x), ~clear(x)
  - Possible effects: dirty(x) with an unspecified weight
- Treat weights as probabilities with random variables
- Robustness measure:

$$\boldsymbol{R(\pi)} \stackrel{\text{def}}{=} \sum_{\boldsymbol{D_i} \in \langle\langle \widetilde{\boldsymbol{D}} \rangle\rangle : \boldsymbol{\gamma}^{\boldsymbol{D_i}}(\boldsymbol{\pi}, \boldsymbol{I}) \models \boldsymbol{G}} \mathbf{Pr}(\boldsymbol{D_i})$$

Content

- A measure for plan quality
  - Robustness of plan $R(\pi) \in [0,1]$
- **Plan robustness assessment**
  - Reduced to weighted model counting
  - Complexity
- Synthesizing robust plans
  - Compilation approach
  - Heuristic search approach

Plan Robustness Assessment

- Computation:
  - Given $\widetilde{D}$, $\tilde{P} = \langle F, A, I, G \rangle$, a plan π
  - Construct a set of correctness constraints $\boldsymbol{\Sigma(\pi)}$ for the execution of π:
    - State transitions caused by actions are correct.
    - The goal G is satisfied in the last state.
  - Then: $R(\pi)$ is computed from the *weighted model count* of $\boldsymbol{\Sigma(\pi)}$

$$I \xrightarrow{\pi} G$$

$$\bigvee_{C_p^i \leq k \leq i-1, p \in \widetilde{Add}(a_k)} \boldsymbol{p_{a_k}^{add}}$$

$$\boldsymbol{\Sigma(\pi)}$$

$$\boldsymbol{p_{a_i}^{pre}} \Rightarrow \bigvee_{C_p^i \leq k \leq i-1, p \in \widetilde{Add}(a_k)} \boldsymbol{p_{a_k}^{add}}$$

$$\boldsymbol{p_{a_m}^{del}} \Rightarrow \bigvee_{C_p^i \leq k \leq i-1, p \in \widetilde{Add}(a_k)} \boldsymbol{p_{a_k}^{add}}$$

$$\boldsymbol{p_{a_i}^{pre}} \Rightarrow (\boldsymbol{p_{a_m}^{del}} \Rightarrow \bigvee_{C_p^i \leq k \leq i-1, p \in \widetilde{Add}(a_k)} \boldsymbol{p_{a_k}^{add}})$$

Plan Robustness Assessment

- **Complexity**

  The problem of computing $R(\pi)$ for a plan π to a problem $\langle \widetilde{D}, I, G \rangle$ is #P-complete.

- Membership:
  - Have a Counting TM non-deterministically guess a complete model, and check the correctness of the plan.
  - The number of accepting branches output: the number of complete models under which the plan succeeds.
- Completeness:
  - There exists a counting reduction from the problem of counting satisfying assignments for Monotone-2-SAT problem to Robustness-Assessment (RA) problem

Content

- A measure for plan quality
  - Robustness of plan $R(\pi) \in [0,1]$
- Plan robustness assessment
  - Reduced to weighted model counting
  - Complexity
- Synthesizing robust plans
  - **Compilation approach**
  - Heuristic search approach

Compilation Approach

- The realization of possible preconditions / effects is determined by unknown variables p_a^{pre}, p_a^{add}, p_a^{del}
  - Thus, can be compiled away using "conditional effects"
    - If $p_a^{pre} = true$ then p is a precondition of a.
  - Domain incompleteness ➔ State incompleteness
    - Conformant probabilistic planning problem!

Compiled “pick-up”

```
pick-up
:parameters (?b - ball ?r - room)
:precondition (and )
:effect (and
  (when (and (at ?b ?r) (at-robot ?r) (free-gripper)
             (light ?b) ( p^{pre}_{pick-up} ) ( q^{add}_{pick-up} ))
        (and (carry ?b)(not (at ?b ?r))(not (free-gripper))
             dirty ?b)))
  (when (and (at ?b ?r) (at-robot ?r) (free-gripper)
             (light ?b) ( p^{pre}_{pick-up} ) ( nq^{add}_{pick-up} ))
        (and (carry ?b)(not (at ?b ?r))(not (free-gripper))))
  (when (and (at ?b ?r) (at-robot ?r) (free-gripper)
             ( np^{pre}_{pick-up} ) ( q^{add}_{pick-up} ))
        (and (carry ?b)(not (at ?b ?r))(not (free-gripper))
             (dirty ?b)))
  (when (and (at ?b ?r) (at-robot ?r) (free-gripper)
             ( np^{pre}_{pick-up} ) ( nq^{add}_{pick-up} ))
        (and (carry ?b)(not (at ?b ?r))(not (free-gripper))))
```

Compilation: Experimental Results

- Using Probabilistic-FF planner (Domshlak & Hoffmann, 2006)

ρ	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$
0.1	32/10.9	36/26.2	40/57.8	44/121.8	48/245.6
0.2	32/10.9	36/25.9	40/57.8	44/121.8	48/245.6
0.3	32/10.9	36/26.2	40/57.7	44/122.2	48/245.6
0.4	$\perp$	42/42.1	50/107.9	58/252.8	66/551.4
0.5	$\perp$	42/42.0	50/107.9	58/253.1	66/551.1
0.6	$\perp$	$\perp$	50/108.2	58/252.8	66/551.1
0.7	$\perp$	$\perp$	$\perp$	58/253.1	66/551.6
0.8	$\perp$	$\perp$	$\perp$	$\perp$	66/550.9
0.9	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

Incomplete Logistics domain

Synthesizing Robust Plans under Incomplete Domain Models (NIPS 2013)

- Normally fails with large problem instances

Content

- A measure for plan quality
  - Robustness of plan $R(\pi) \in [0,1]$
- Plan robustness assessment
  - Reduced to weighted model counting
  - Complexity
- Synthesizing robust plans
  - Compilation approach
  - **Heuristic search approach**

Approximate Transition Function

- Not explicitly maintain set of resulting states

$$\gamma(\pi, s) = \bigcup_{D_i \in \langle\langle \widetilde{D} \rangle\rangle} \gamma^{D_i}(\pi, s)$$

- Successor state:

$$\tilde{\gamma}_{SE}(\langle a \rangle, s) = s \cup Add(a) \cup \widetilde{Add}(a) \setminus Del(a), if\ Pre(a) \subseteq s$$

  - Possible delete effects might not take effects!
- Recursive definition for $\tilde{\gamma}_{SE}(\pi, s)$

Completeness: *Any solution in the complete STRIPS action model exists in the solution space of the problem with incomplete domain.*
Soundness*: For any plan returned under incomplete STRIPS domain semantics, there is one complete STRIPS model under which the plan succeeds.*

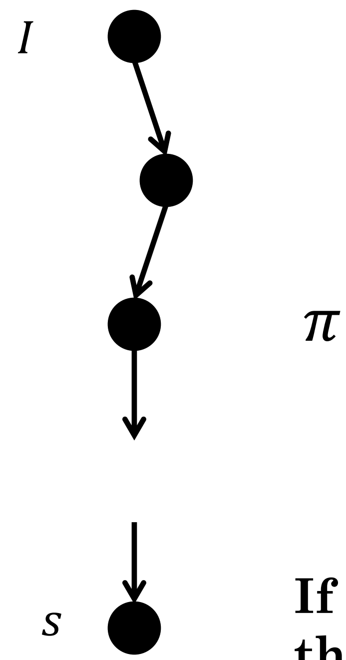
Anytime Approach for Generating Robust Plans

1. Initialize: $\delta = 0$
2. **Repeat**
   - Find plan π s.t. $R(\pi) > \delta$ (Stochastic)
   - If plan found: $\delta = R(\pi)$

   Until time bound reaches
3. Return π and $R(\pi)$ if plan found

Use of Upper Bound

- Reduce exact weighted model counting



If $(G \subseteq s)$ and $U(\pi) > \delta$
then $wmc(\pi)$

$\boldsymbol{U(\pi) \geq wmc(\pi)}$ (Upper bound for $R(\pi)$)

Use of Lower Bound

- How to…
  - Compute $h(s, \delta) = |\tilde{\pi}|$

Find: $\tilde{\boldsymbol{\pi}}$ s.t. $\mathrm{wmc}(\pi_k \circ \tilde{\pi}) > \delta$

Avoid invoking $wmc(\circ)$ during the construction of $\tilde{\pi}$!

Find: $\tilde{\boldsymbol{\pi}}$ s.t. $\mathrm{L}(\pi_k \circ \tilde{\pi}) > \delta$

$\mathbf{L}(\boldsymbol{\pi}) \leq \boldsymbol{wmc}(\boldsymbol{\pi})$ (Lower bound for $R(\pi)$)

LOWER BOUND FOR $R(\pi)$

$$\bigvee_{C_p^i \le k \le i-1, p \in \widetilde{Add}(a_k)} p_{a_k}^{add}$$

$$\Sigma(\pi)$$

$$p_{a_i}^{pre} \Rightarrow \bigvee_{C_p^i \le k \le i-1, p \in \widetilde{Add}(a_k)} p_{a_k}^{add}$$

$$p_{a_m}^{del} \Rightarrow \bigvee_{C_p^i \le k \le i-1, p \in \widetilde{Add}(a_k)} p_{a_k}^{add}$$

$$p_{a_i}^{pre} \Rightarrow (p_{a_m}^{del} \Rightarrow \bigvee_{C_p^i \le k \le i-1, p \in \widetilde{Add}(a_k)} p_{a_k}^{add})$$

LOWER BOUND FOR $R(\pi)$

$$\bigvee_{C_p^i \leq k \leq i-1, p \in \widetilde{Add}(a_k)} p_{a_k}^{add}$$

$$\neg p_{a_i}^{pre} \vee \bigvee_{C_p^i \leq k \leq i-1, p \in \widetilde{Add}(a_k)} p_{a_k}^{add}$$

$\boldsymbol{\Sigma(\pi)}$

$$\neg p_{a_m}^{del} \vee \bigvee_{C_p^i \leq k \leq i-1, p \in \widetilde{Add}(a_k)} p_{a_k}^{add}$$

$$\neg p_{a_i}^{pre} \vee \neg p_{a_m}^{del} \vee \bigvee_{C_p^i \leq k \leq i-1, p \in \widetilde{Add}(a_k)} p_{a_k}^{add}$$

$\boldsymbol{w(\neg p) = 1 - w(p)}$

$\Sigma(\pi)$ as a set of clauses with positive literals.

LOWER BOUND FOR $R(\pi)$

Given positive clauses c, c' : $\Pr(c \mid c') \geq \Pr(c)$

Given $\Sigma(\pi) = \{c_1, c_2, \dots, c_n\}$:

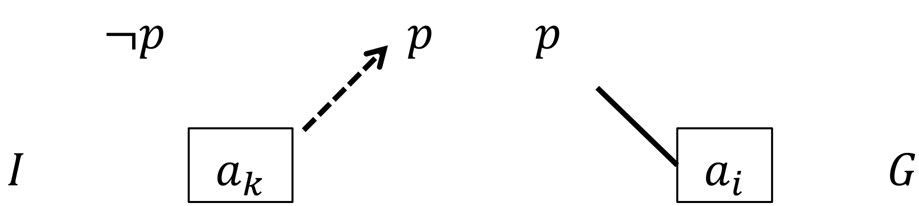
$$
\begin{aligned}
wmc\big(\Sigma(\pi)\big) &= \Pr(c_1 \wedge c_2 \wedge \cdots \wedge c_n) \\
&= \Pr(c_1)\Pr(c_2|c_1)\dots\Pr(c_n|c_{n-1}, \dots, c_1) \\
&\geq \prod_{c_i \in \Sigma(\pi)} \Pr(c_i)
\end{aligned}
$$

(Equality holds when all clauses are independent)

$$L(\pi) = \prod_{c_i \in \Sigma(\pi)} \Pr(c_i) \leq R(\pi)$$

UPPER BOUND FOR $R(\pi)$

- $\Sigma(\pi) = \{c_1, \dots, c_n\}$
- An (trivial) upper bound:
  - $U(\pi) \geq \min_{c_i \in \Sigma(\pi)} \Pr(c_i)$
- A much tighter bound:



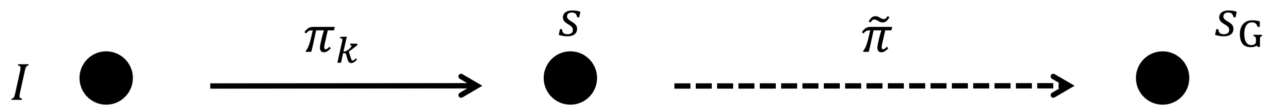
- Clauses contains variables corresponding to one specific predicate
- Thus, $\Sigma(\pi)$ is highly decomposable into "connected components"

$$\bigvee_{C_p^i \leq k \leq i-1, p \in \widetilde{Add}(a_k)} p_{a_k}^{add}$$

$$U(\pi) = \prod_j \min_{c \in \Sigma_j} \Pr(c)$$

$\{x_1, x_2\}$ $\{x_2, x_3\}$ $\{x_4, x_5\}$

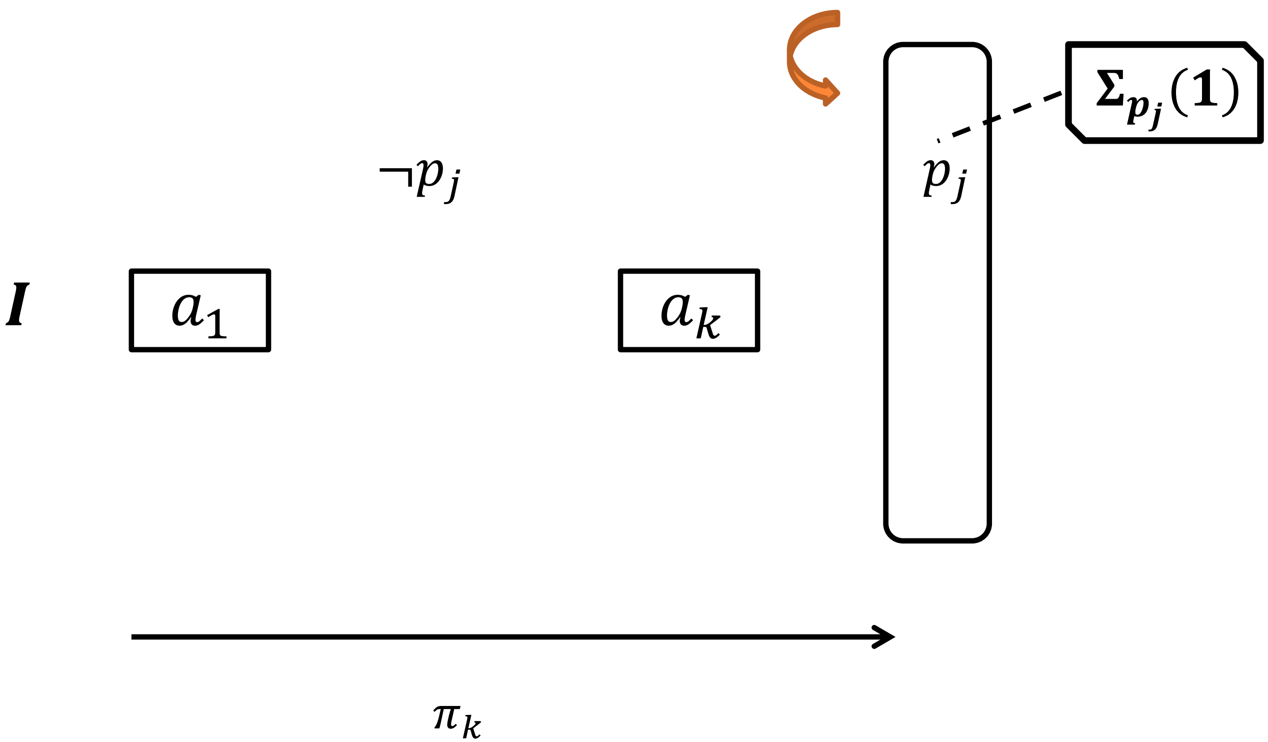
FIND: $\tilde{\pi}$ S.T. $L(\pi_k \circ \tilde{\pi}) > \delta$



- Build relaxed planning graph
  - Ignoring known & possible delete effects
- Propagate clauses for propositions and actions
- Extract relaxed plan

Relaxed Planning Graph
Propositional Layer L_1

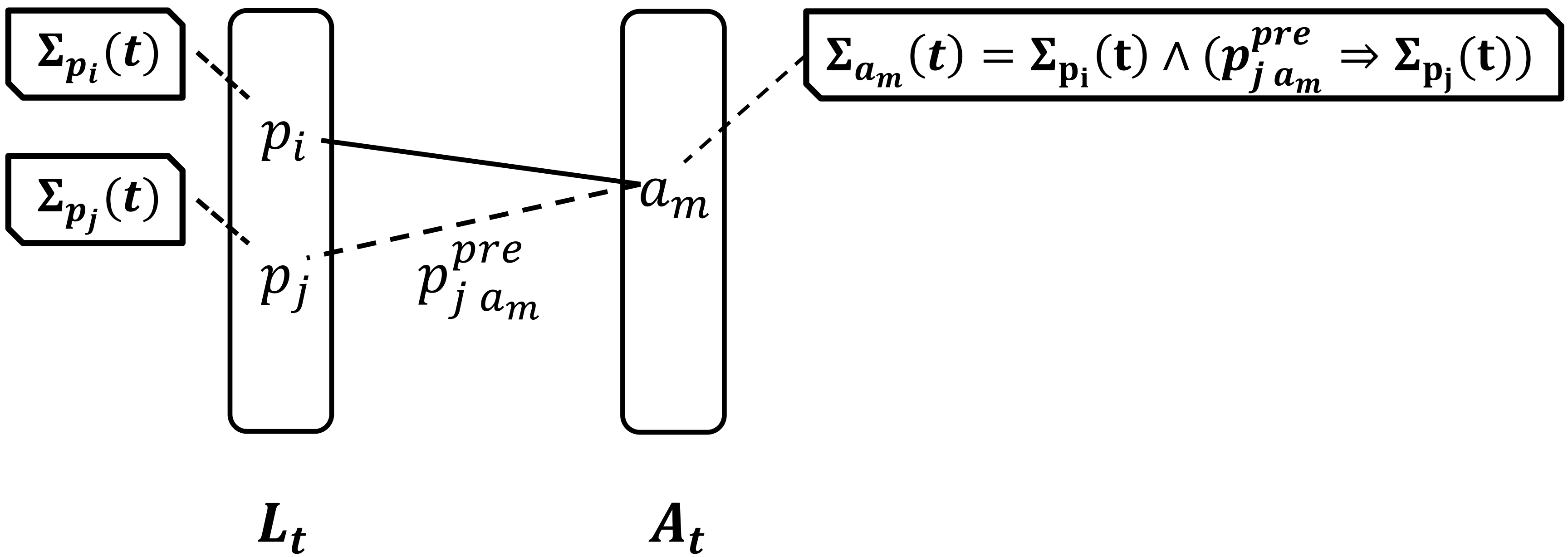
$$L_1 = s_{k+1} = \tilde{\gamma}_{SE}(\pi_k, I)$$



Establishment constraints (if needed) and protection constraints for p_j at state s_{k+1}

Relaxed Planning Graph
Action Layer A_t

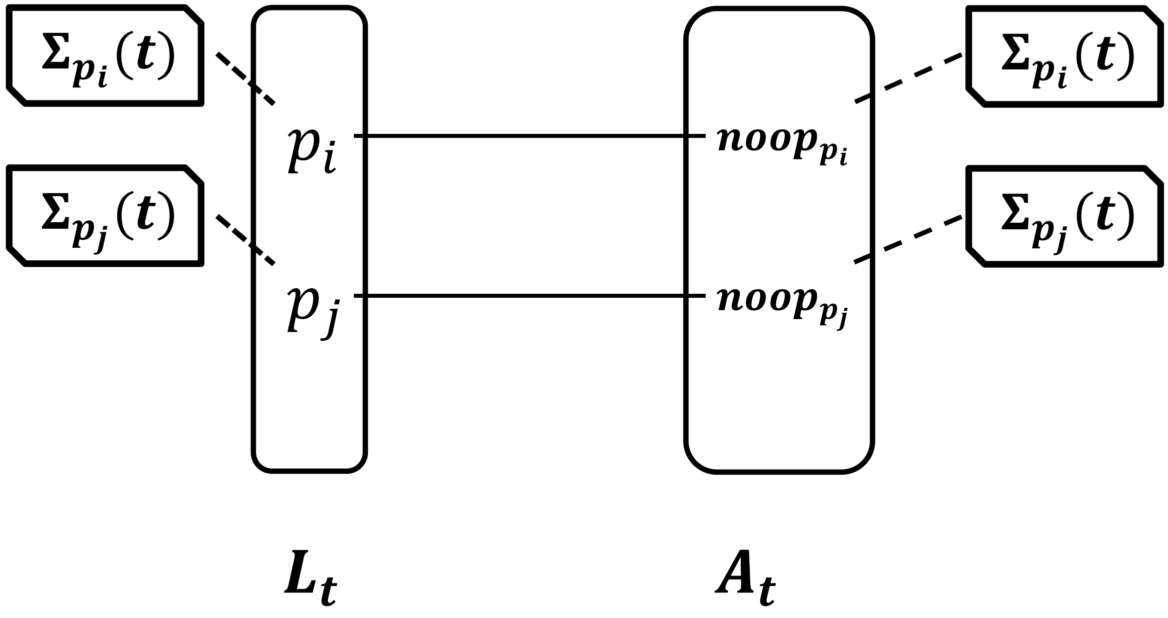
$$A_t = \{a \mid a \in A, Pre(a) \subseteq L_t\} \cup \{noop_p \mid p \in L_t\}$$



Relaxed Planning Graph

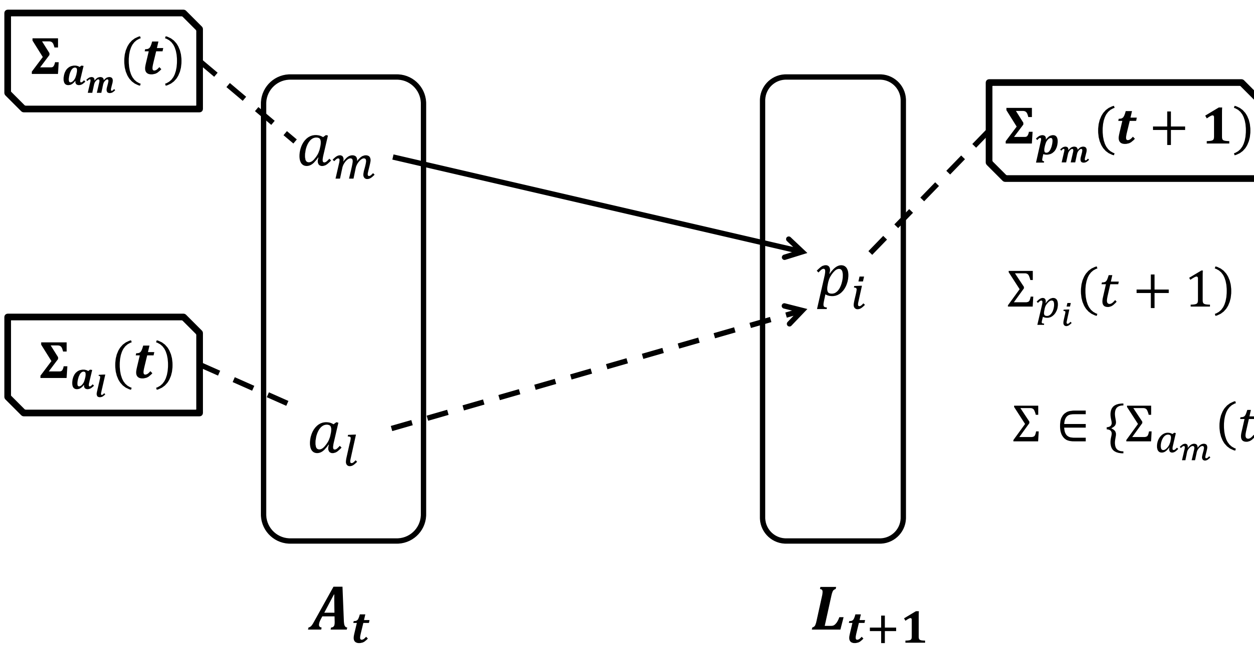
Action Layer A_t

$$A_t = \{a \mid a \in A, Pre(a) \subseteq L_t\} \cup \{noop_p \mid p \in L_t\}$$



Relaxed Planning Graph
Propositional Layer L_{t+1}

$$L_{t+1} = \{p \mid a \in A_t, p \in Add(a) \cup \widetilde{Add}(a)\}$$



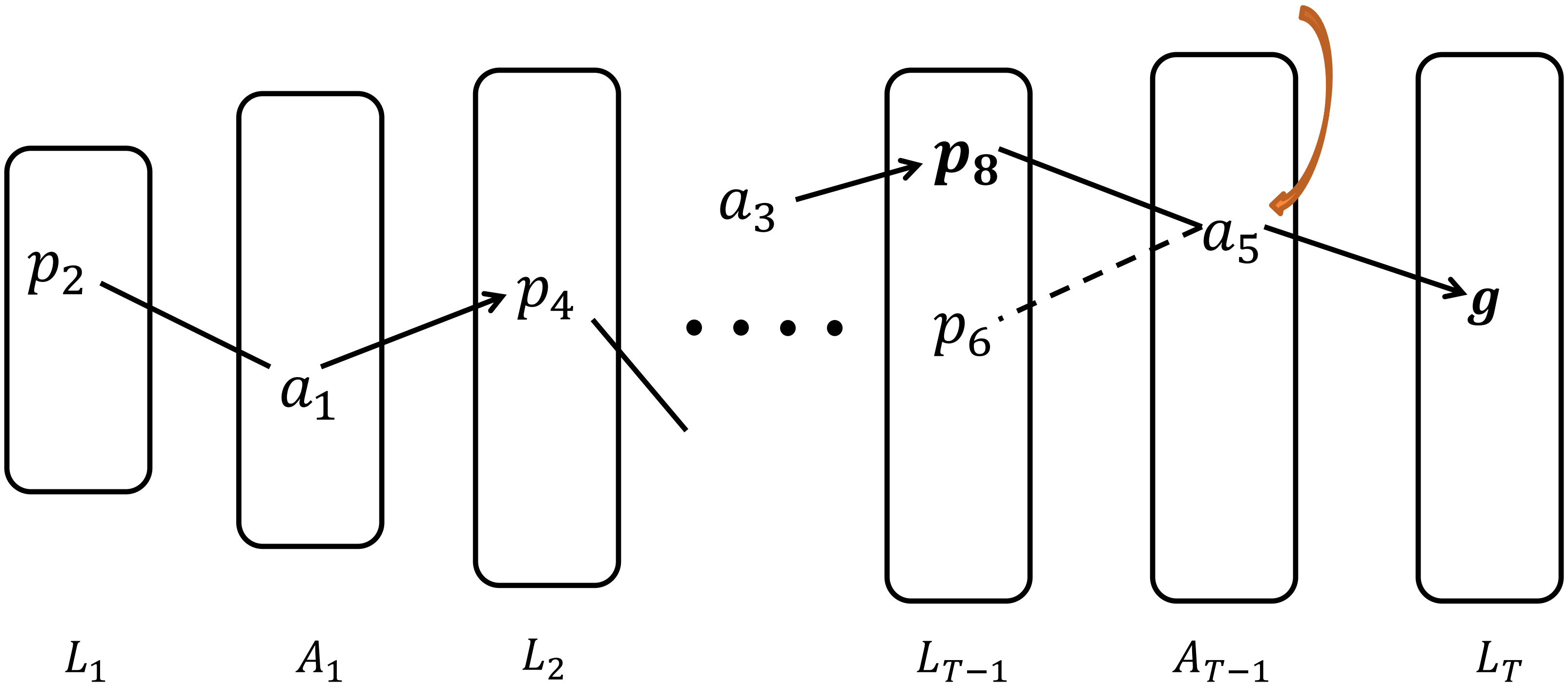
$$\Sigma_{p_i}(t+1) = argmax_{\Sigma}\, l(\Sigma \wedge \Sigma_k)$$

$$\Sigma \in \{\Sigma_{a_m}(t), p_{i\,a_l}^{pre} \Rightarrow \Sigma_{a_l}(t)\}$$

Relaxed Plan Extraction
Overview

$$\boldsymbol{\Sigma_{p_i}(t+1) = argmax_{\Sigma}\, l(\Sigma \wedge \Sigma_k)}$$

Best supporting action
for g at layer T

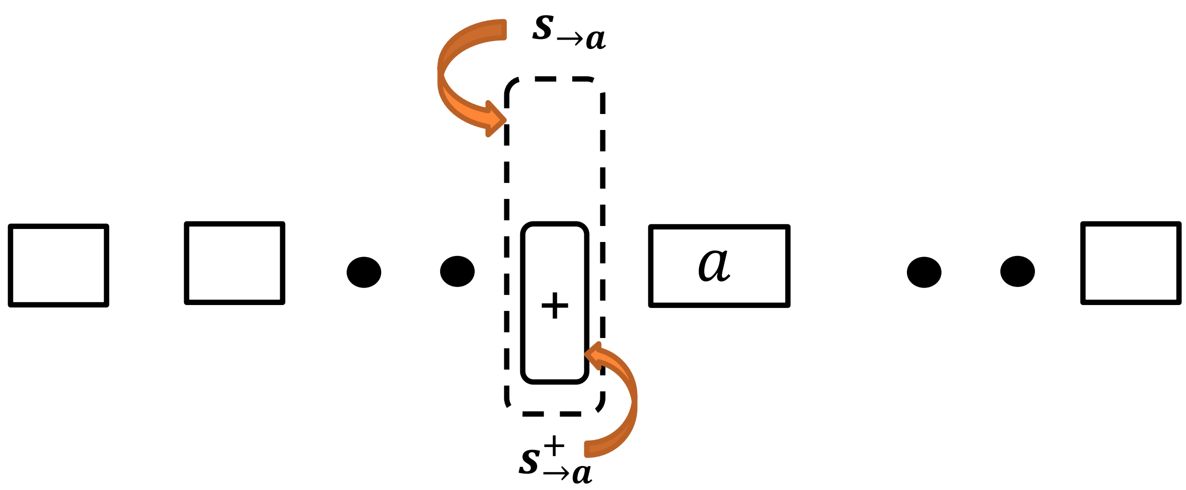


- $\tilde{\pi}$ in total order
- Succeed when:
  - All know preconditions are supported
  - $l(\Sigma_k \wedge \Sigma_{\pi'}) > \delta$

Relaxed Plan Extraction

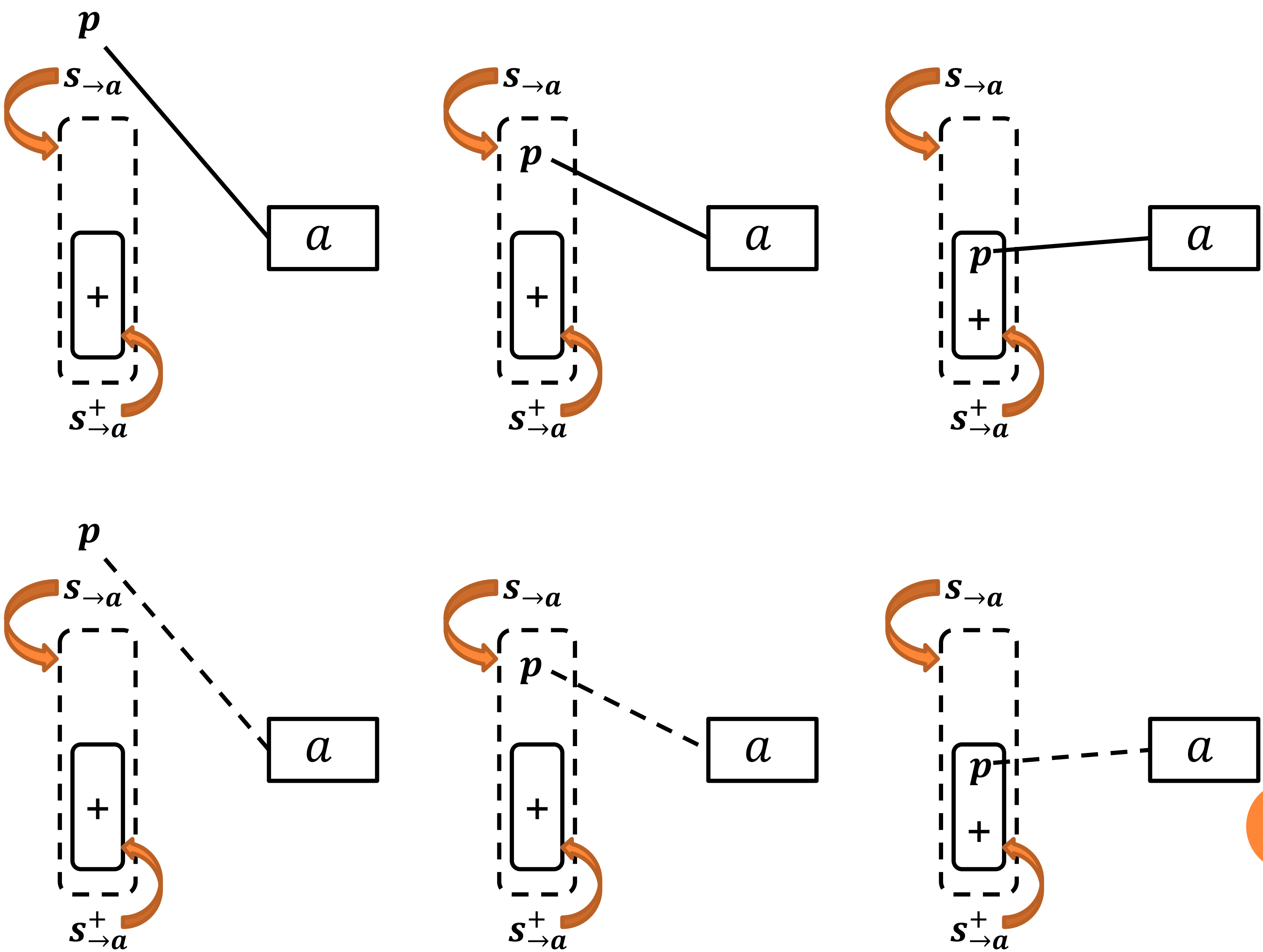
When to insert actions?

- A supporting action a_{best} is inserted only if needed
- Depending on:
  - Relation between: subgoal and "relaxed plan state"
  - Robustness of the current $\tilde{\pi}$ and $\tilde{\pi} \cup \{a_{best}\}$



Relaxed Plan Extraction

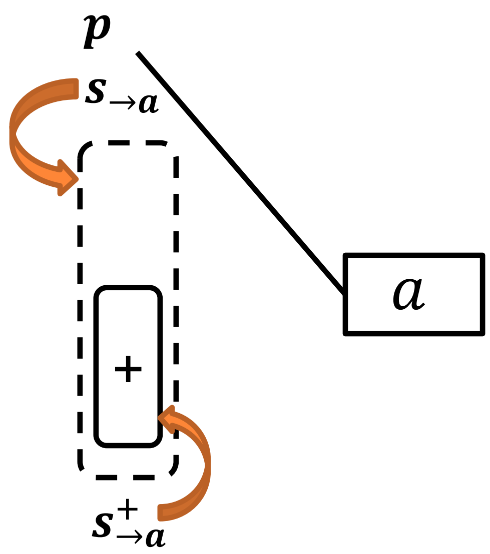
Subgoal v.s RP state



Relaxed Plan Extraction

- $p \in Pre(a), p \notin s_{\rightarrow a}$: insert a_{best} into $\tilde{\pi}$

No actions in π_k and $\tilde{\pi}$ supporting this subgoal

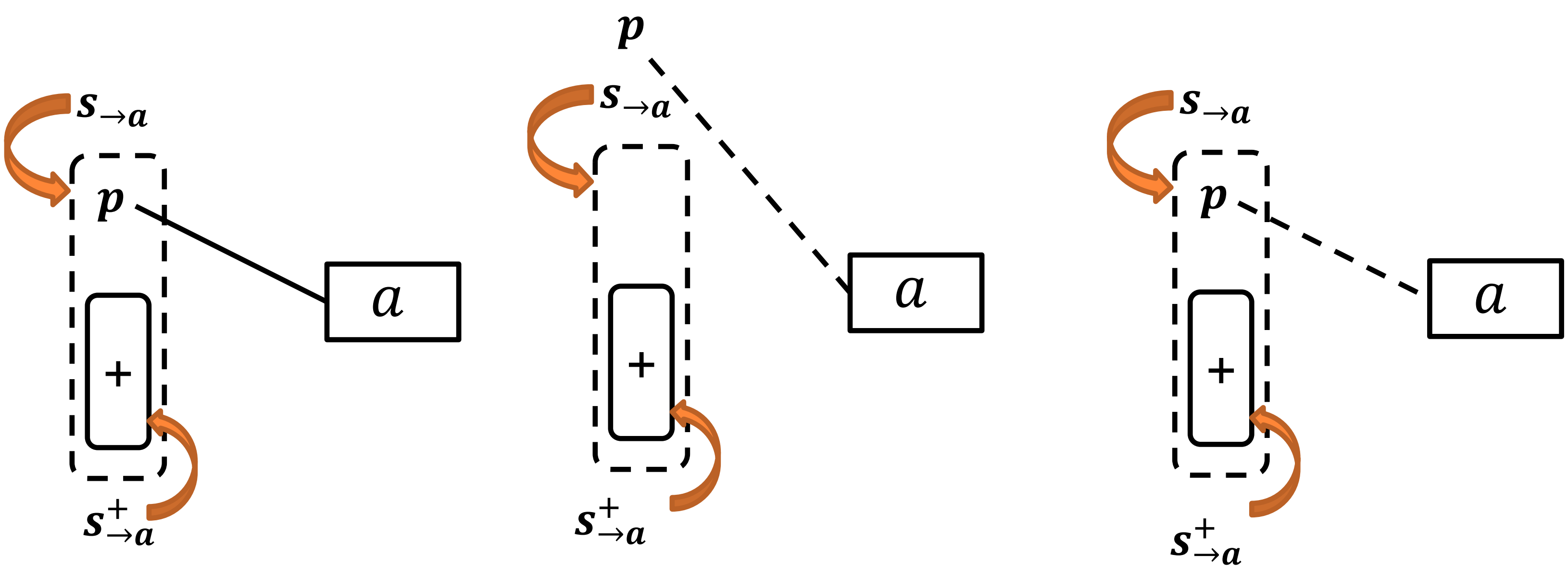


- This type of subgoal makes the relaxed plan "incomplete"

Relaxed Plan Extraction

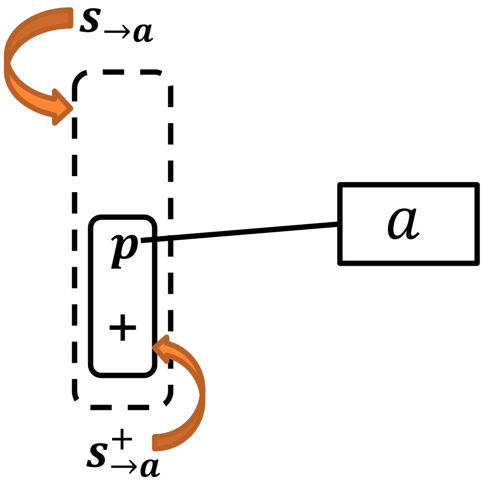
- For these subgoals, supporting actions inserted if the insertion increases the robustness of the current relaxed plan.

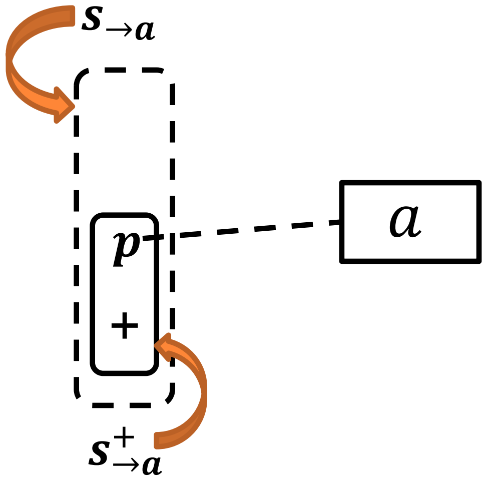
$$l(\Sigma_\pi \wedge \Sigma_{\tilde{\pi} \cup \{a_{best}\}}) > l(\Sigma_\pi \wedge \Sigma_{\tilde{\pi}})$$



Relaxed Plan Extraction

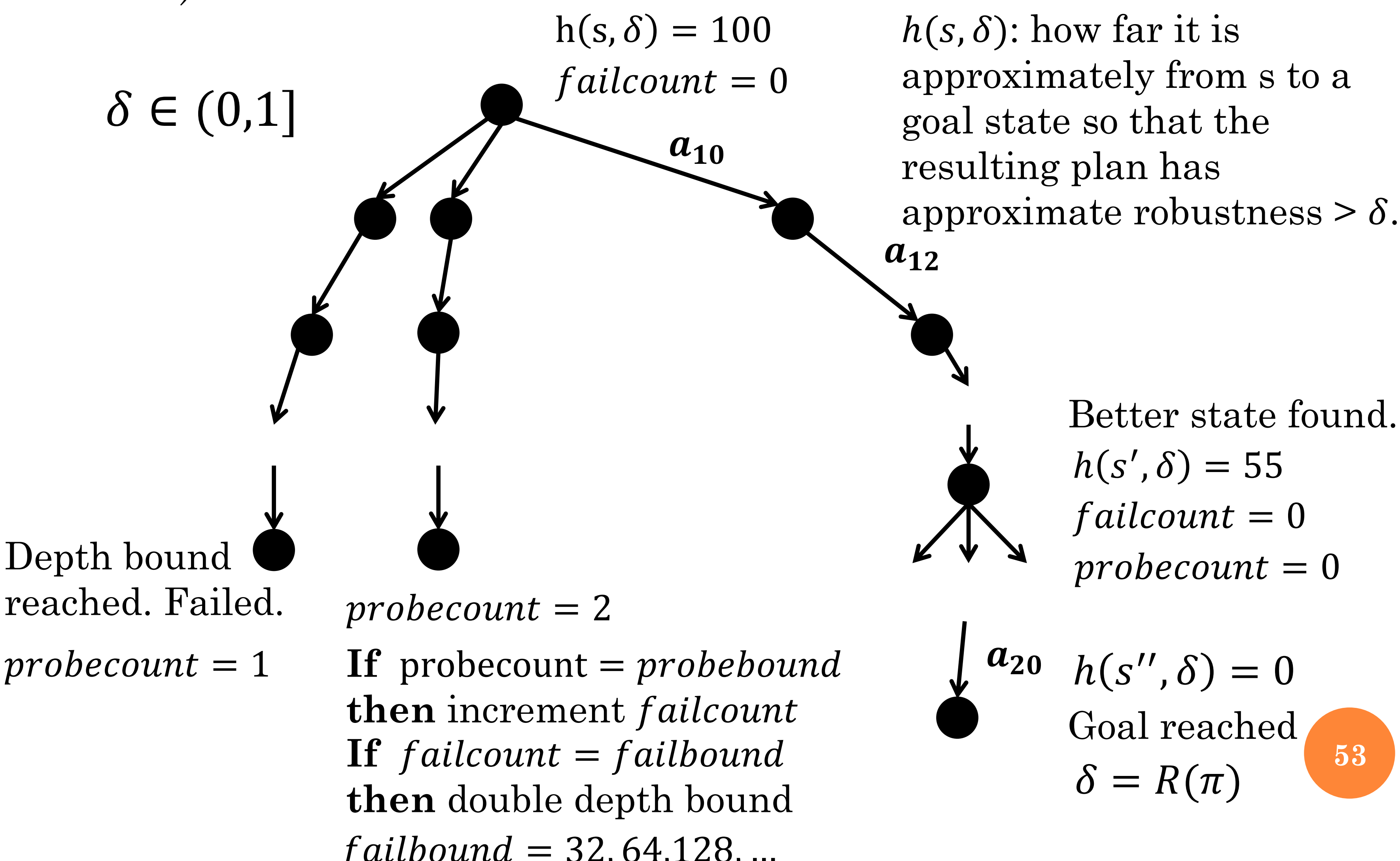
- For these subgoals, no supporting actions needed!





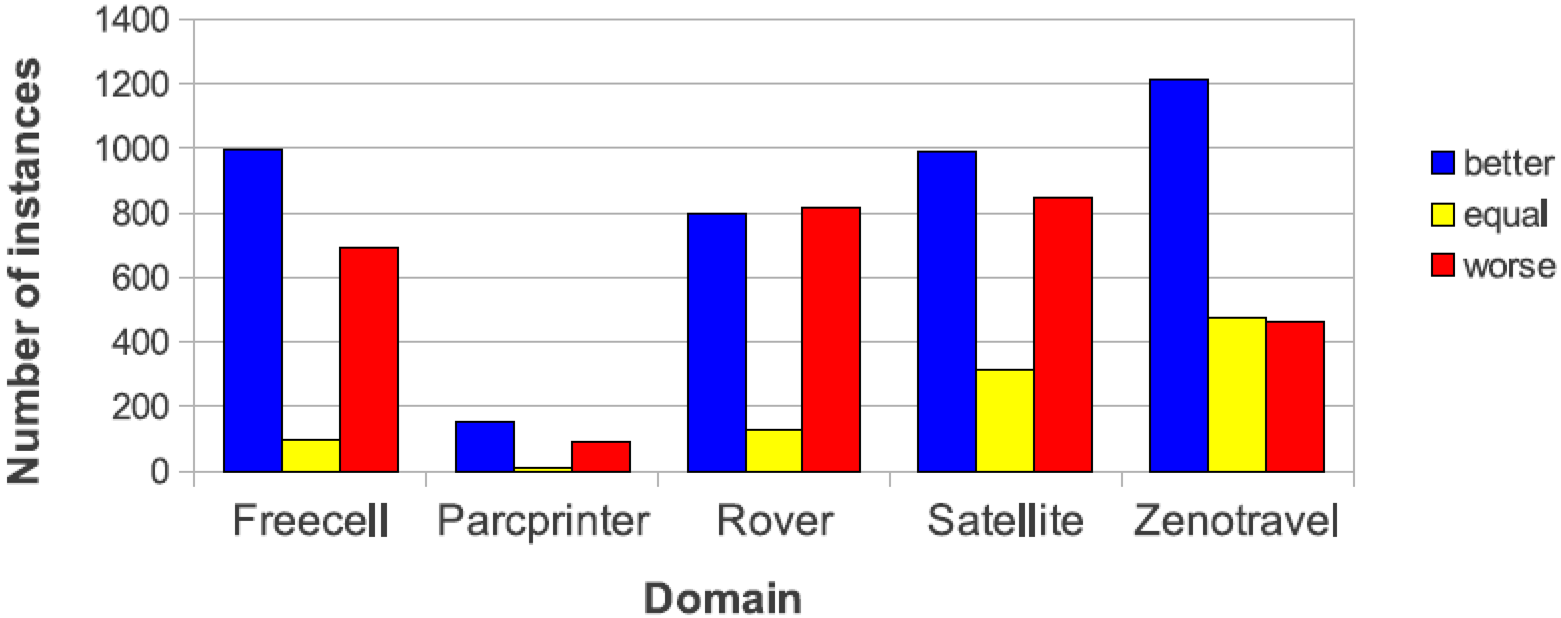
Find π s.t. $R(\pi) > \delta$: Search Algorithm

- Stochastic local search with failed bounded restarts (Coles et al., 2007)



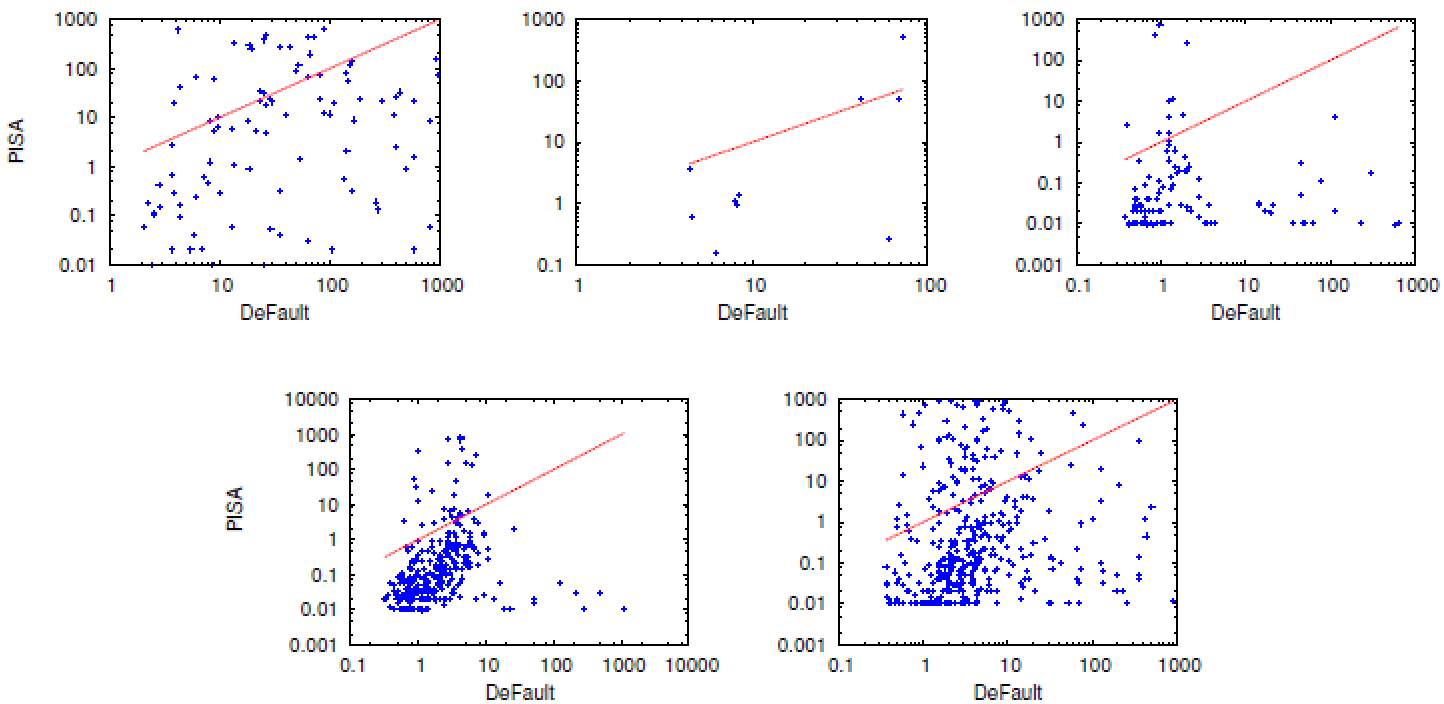
Experimental Results

- Domains:
  - Zenotravel, Freecell, Satellite, Rover (215 domains x 10 problems = 2150 instances)
  - Parc Printer (300 instances)



Number of instances for which PISA produces better, equal and worse robust plans compared to DeFault.

Experimental Results



Total time in seconds (log scale) to generate plans with the same robustness by PISA and DeFault.

Dissertation overview

"Model-lite" Planning

Preference incompleteness

- **Representation:** two levels of incompleteness
  - User preferences exist, but totally unknown
  - Partially specified
    - Full set of plan attributes
    - Parameterized value function, unknown trade-off values
- **Solution concept:** plan sets
- **Solving techniques:** synthesizing high quality plan sets

Domain incompleteness

- **Representation**
  - Actions with possible preconditions / effects
  - Optionally with weights for being the real ones
- **Solution concept:** "robust" plans
- **Solving techniques:** synthesizing robust plans

DISSERTATION OVERVIEW

"Model-lite" Planning

Preference incompleteness

Publication

- *Domain independent approaches for finding diverse plans*. **IJCAI (2007)**
- *Planning with partial preference models*. **IJCAI (2009)**
- *Generating diverse plans to handle unknown and partially known user preferences*. **AIJ 190 (2012)**

(with Biplav Srivastava, Subbarao Kambhampati, Minh Do, Alfonso Gerevini and Ivan Serina)

Domain incompleteness

Publication

- *Assessing and Generating Robust Plans with Partial Domain Models*. **ICAPS-WS (2010)**
- *Synthesizing Robust Plans under Incomplete Domain Models*. **AAAI-WS (2011), NIPS (2013)**
- *A Heuristic Approach to Planning with Incomplete STRIPS Action Models*. **ICAPS (2014)**

(with Subbarao Kambhampati, Minh Do)

Thank you!
Q & A