Comparison of State-Space and Plan-Space Planning

Manuela Veloso

Carnegie Mellon University
Computer Science Department

Planning - Fall 2001

Outline: Selecting ACTIONS

- Planning algorithms
- Comparison of planning algorithms
- Learning in planning
- Planning, execution, and learning
- Behavior-based (reactive) planning
- Action selection in multiagent systems

Several Planning Algorithms

- TWEAK (Chapman 87), SNLP (McAllester & Rosenblitt 91 , UCPOP (Penberthy and Weld 92)
- NONLIN (Tate 76), O-PLAN (Tate), SIPE (Wilkins 88)
- Prodigy2.0 (Minton et al. 87), Prodigy4.0 (Veloso et al. 90)
- UNPOP, Planning and acting (McDermott 78)
- Reactive planning (Georgeff & Lansky 87), (Firby 87), (Hendler & Sanborn 87)
- Action and time (Allen 84) (Dean & McDermott 87)

Several Planning Algorithms

- Walksat, Satplan (Selman et al. 92, Kautz & Selman 92, 96)
- Flecs (Veloso & Stone 95)
- Graphplan (Blum & Furst 95)
- MBP (Cimatti, Roveri, Traverso 98)
- UMOP (Jensen & Veloso 00)
- More at planning competitions - AIPS'98, AIPS'00

Plan-Space Partial-Order Nonlinear Planning

SNLP Planning Algorithm *McAllester & Rosenblitt 91*

1. Terminate if the goal set is empty.

2. Select a goal g from the goal set and identify the plan step that needs it, S_{need}.

3. Let S_{add} be a step (operator) that adds g, either a new step or a step that is already in the plan. Add the causal link $S_{add} \xrightarrow{g} S_{need}$, constrain S_{add} to come before S_{need}, and enforce bindings that make S_{add} add g.

4. Update the goal set with **all** the preconditions of the step S_{add}, and delete g.

5. Identify *threats* and resolve the conflicts by adding ordering or bindings constraints.

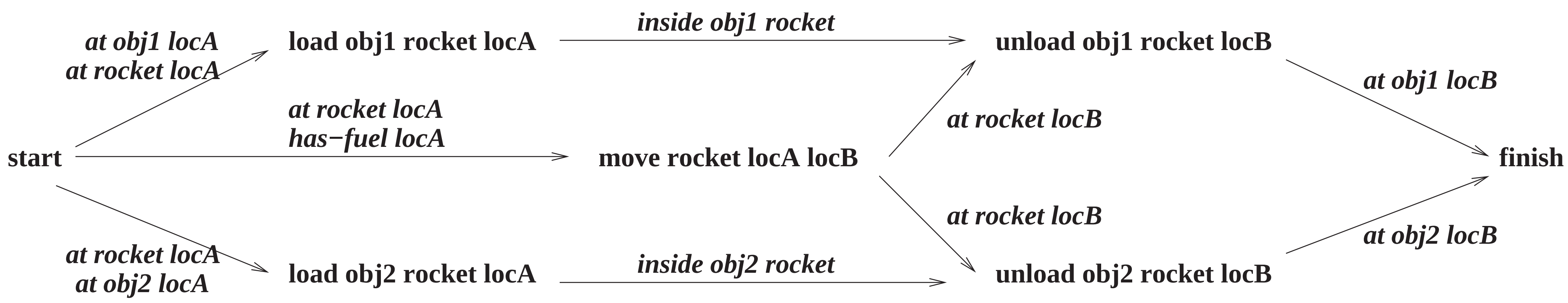
- A step S_k threatens a causal link $S_i \xrightarrow{g} S_j$ when it occurs between S_i and S_j, and it adds or deletes p.

- Resolve threats by using *promotion*, *demotion*, or *separation*.

Plan-space Planning

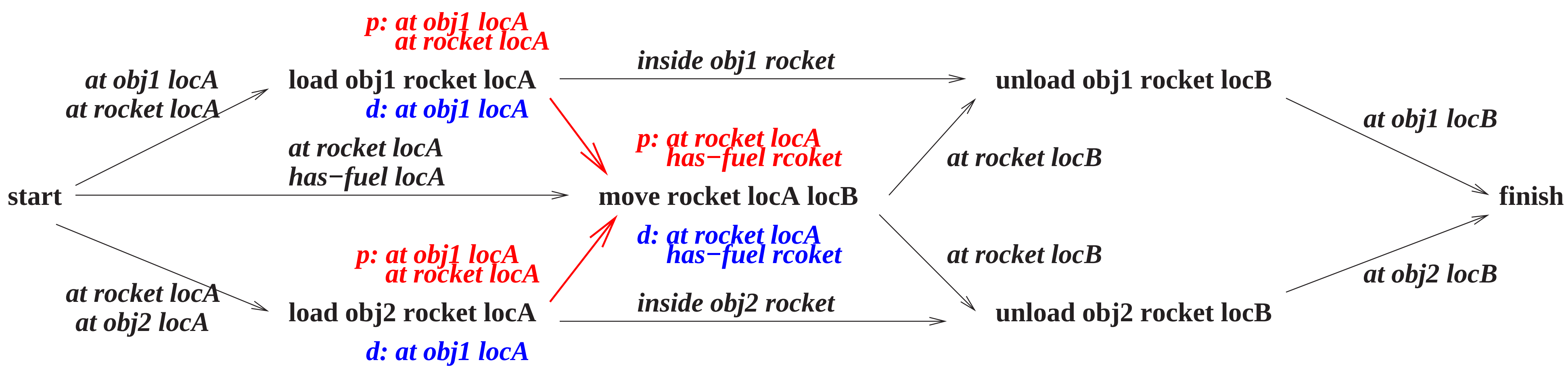
- Complete, sound, and optimal.
- Optimal handling of goal orderings.

Rocket Domain - Linking



Example − LINKING

Rocket Domain - Threats



Example − THREATS

Comparison of Planning Algorithms

- Complete nonlinear state-space planning
- Plan-space planning
- Graphplan
- Satplan
- And more

> Is there a *universally* best planning algorithm?

State-space and Plan-space

- Planning is NP-hard.
- Two different planning approaches: state-space and plan-space planning

	State-space	*Plan-space*
Commitments in plan step orderings	Yes	No
Therefore, suffer with goal orderings	Yes	No
Therefore, handle goal interactions	Poorly	Efficiently

Step Ordering Commitments

WHY?

Use of the STATE of the world while planning

In Prodigy4.0 advantages include:

- Means-ends analysis - plan for goals that reduce the differences between current and goal states.
- Informed selection of operators - select operators that need less planning work than others.
- State useful for learning, generation and match of conditions supporting informed decisions.
- Helpful for generating anytime planning - provide *valid,* executable, plans at any time.

Facts and Goals

- FACTS:
  - Partial-order planners are perceived as generally more efficient than total-order planners.
  - MANY results supporting this claim.
- HOWEVER:
  - Planning as search implies necessarily a series of commitments during search.
  - Partial-order planners do search.

Facts and Goals

- GOALS:
  - Identify commitments in a partial-order planner.
  - Understand the implications of such commitments.
  - Provide clear demonstration of exemplary domains where total-order planners perform better than partial-order planners.

Parallel between Commitments

Operator Polish	Operator Drill-Hole
preconds: ()	preconds: ()
adds: polished	adds: has-hole
deletes: ()	deletes: polished

Goal: *polished* and *has-hole* Initial state: empty Prodigy4.0	Goal: *polished* and *has-hole* Initial state: *polished* SNLP
- plan for goal *polished* - select *Polish* • order *Polish* as first step - plan for goal *has-hole* - select *Drill-Hole* • order *Drill-Hole* $\succ$ *Polish* • *polished* deleted, backtrack - *Polish* $\succ$ *Drill-Hole*	- plan for goal *polished* - select *Initial* state • link *Initial* to *polished* - plan for goal *has-hole* - select *Drill-Hole* • link *Drill-Hole* to *has-hole* • threat - relink *polished* - select *Polish* - link *Polish* to *polished* - *Polish* $\succ$ *Drill-Hole*

Serializability and Linkability

- A set of subgoals is *serializable* (Korf):
  - If there exists some ordering whereby they can be solved sequentially,
  - without ever violating a previously solved subgoal.
- Easily serializable, laboriously serializable
- A set of subgoals is *easily linkable*:
  - If, independently of the order by which the planner links these subgoals to operators,
  - it never has to undo those links.
  - Otherwise it is *laboriously linkable*.

Easily Linkable Goals

```
operator A_i            operator A_*
   preconds ()             preconds ()
   adds     g_i            adds     g_*
   deletes  ()             deletes  g_i, ∀i
```

operator A_i
- preconds ()
- adds g_i
- deletes ()

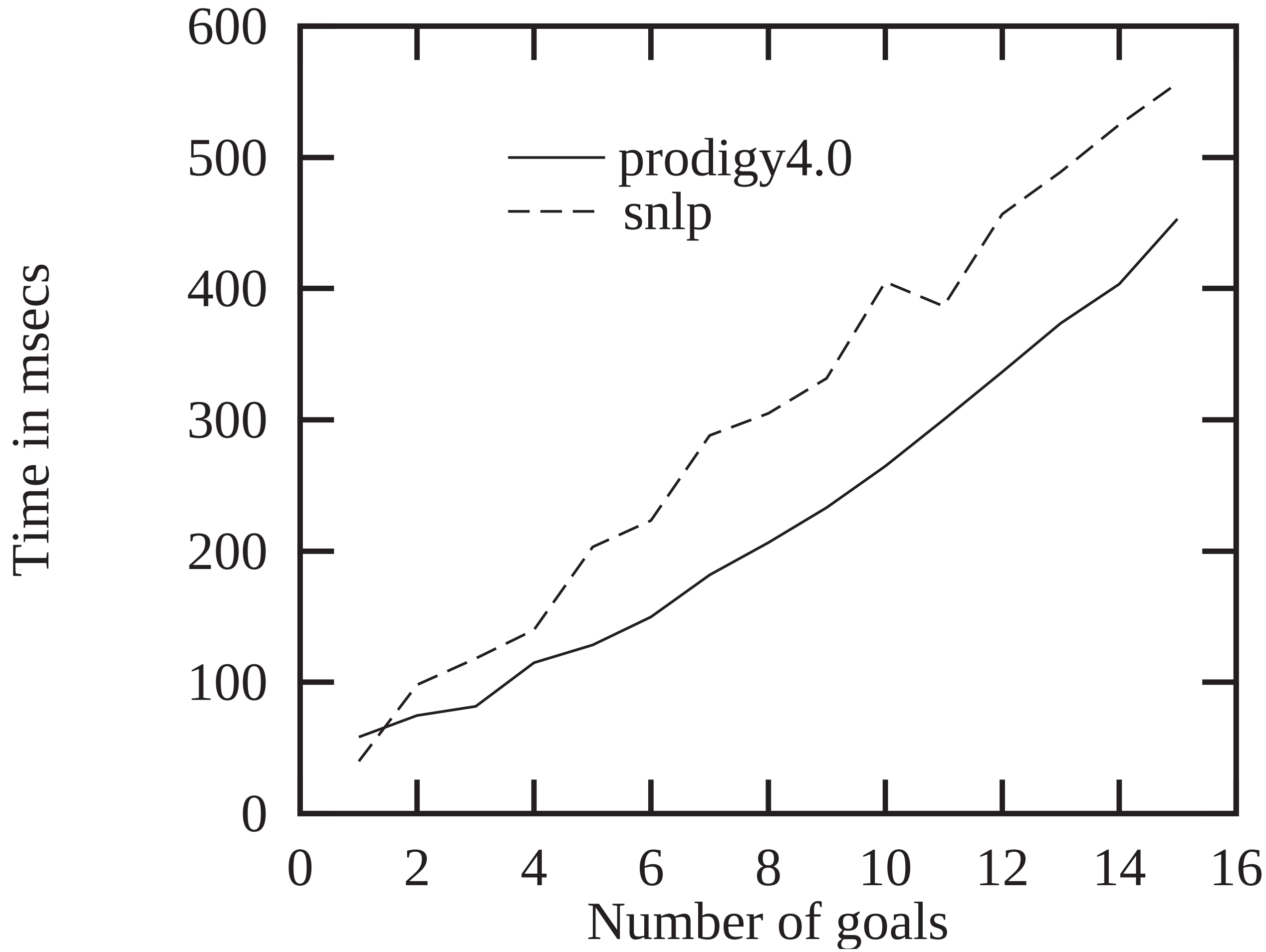
operator A_*
- preconds ()
- adds g_*
- deletes $g_i, \forall i$

Initial state: g_1, g_2, g_3, g_4, g_5

Goal statement: $g_2, g_5, g_4, g_*, g_3, g_1$

Plan: $A_*, A_2, A_5, A_4, A_3, A_1$

Easily Linkable Goals



Laboriously Linkable Goals

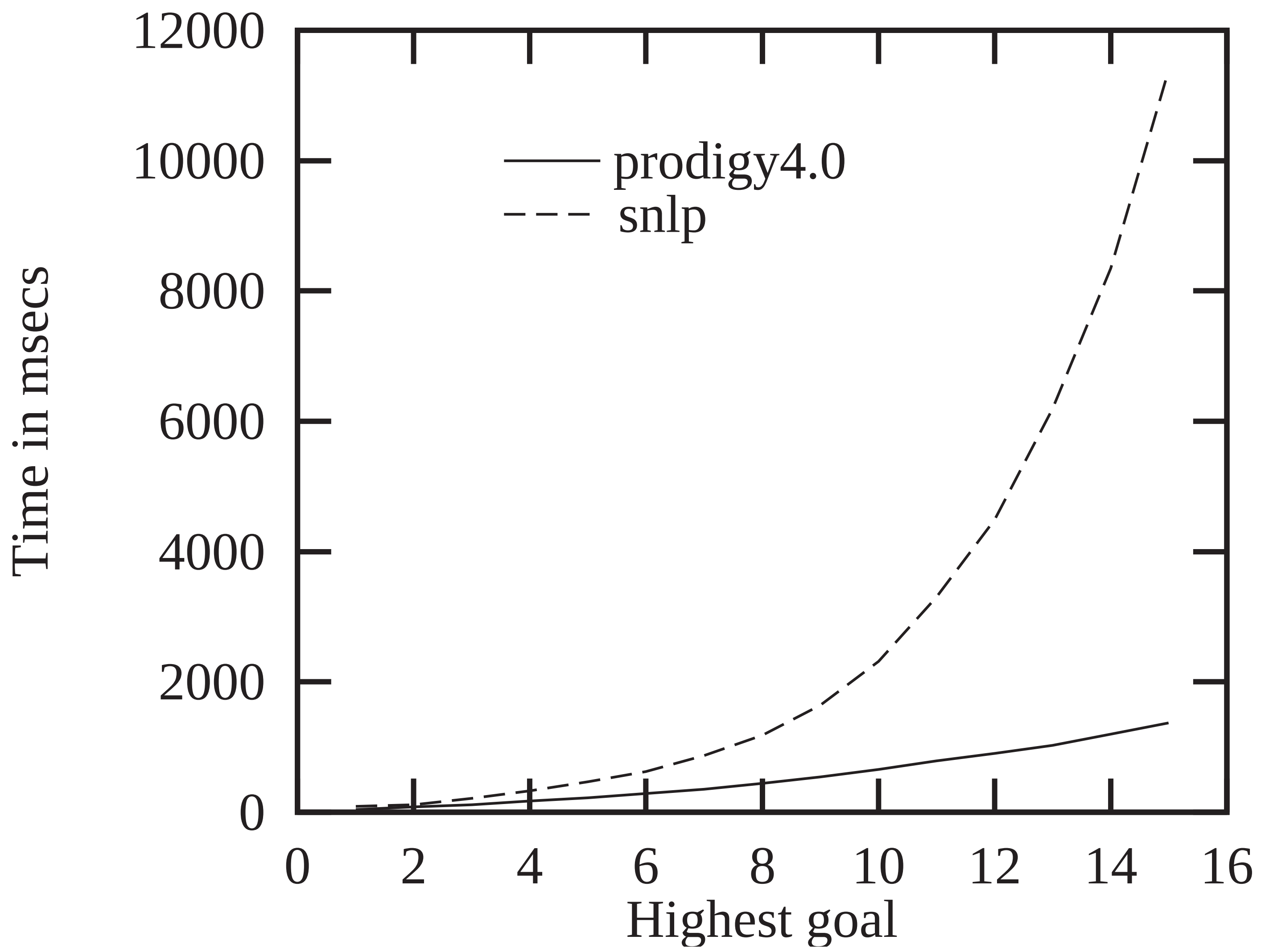
`operator` A_i		`operator` A_*	
`preconds`	g_*, g_{i-1}	`preconds`	()
`adds`	g_i	`adds`	g_*
`deletes`	g_*	`deletes`	()

`Initial state:` g_*

`Goal statement:` g_*, g_5

`Plan:` $A_1, A_*, A_2, A_*, A_3, A_*, A_4, A_*, A_5, A_*$

Laboriously Linkable Goals



Multiple Linking Alternatives

`operator` A_i

`preconds` $g_j, \forall j < i$

`adds` $g_i, g_j, \forall j < i - 1$

`deletes` g_{i-1}

`operator` A_5
`pre` g_4, g_3, g_2, g_1
`add` g_5, g_3, g_2, g_1
`del` g_4

`operator` A_4
`pre` g_3, g_2, g_1
`add` g_4, g_2, g_1
`del` g_3

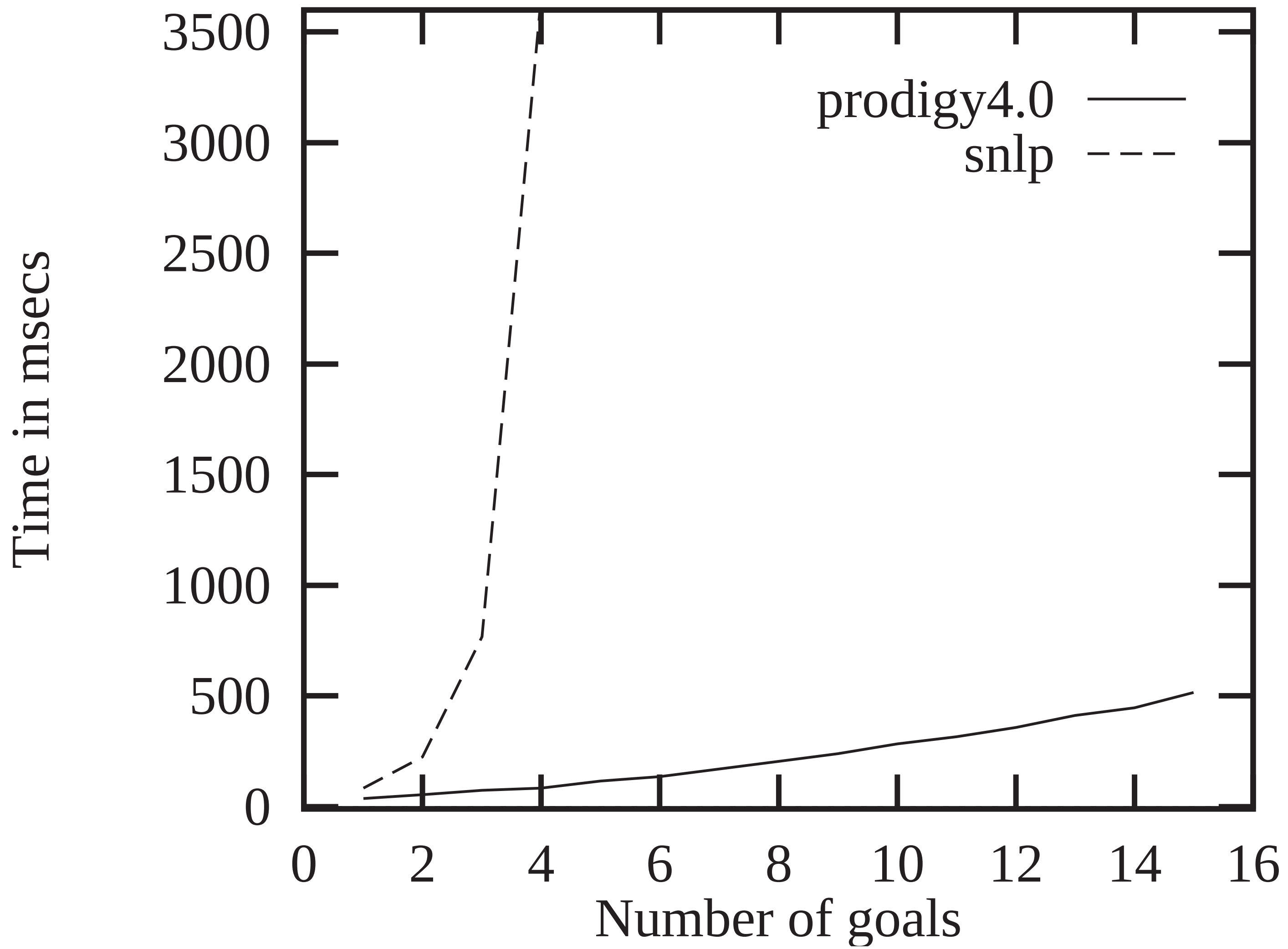
`operator` A_3
`pre` g_2, g_1
`add` g_3, g_1
`del` g_2

`Initial state:` g_1, g_2, g_3, g_4

`Goal statement:` g_2, g_5, g_4, g_3, g_1

`Plan:` A_5, A_4, A_3, A_2, A_1

Empirical Results - Multiple Linking



Summary – Comparison of Planners

- Similar empirical comparison results for other planning algorithms (we'll see later).

- **There is not a planning strategy that is universally better than the others.**

- Even for a particular planning algorithm: **There is no single domain-independent search heuristic that performs more efficiently than others for all problems or in all domains.**

Learning is challenging and appropriate for **ANY** planner.