Scale-up Revolution in Automated Planning

(Or 1001 Ways to Skin a Planning Graph for Heuristic Fun & Profit)

Subbarao Kambhampati

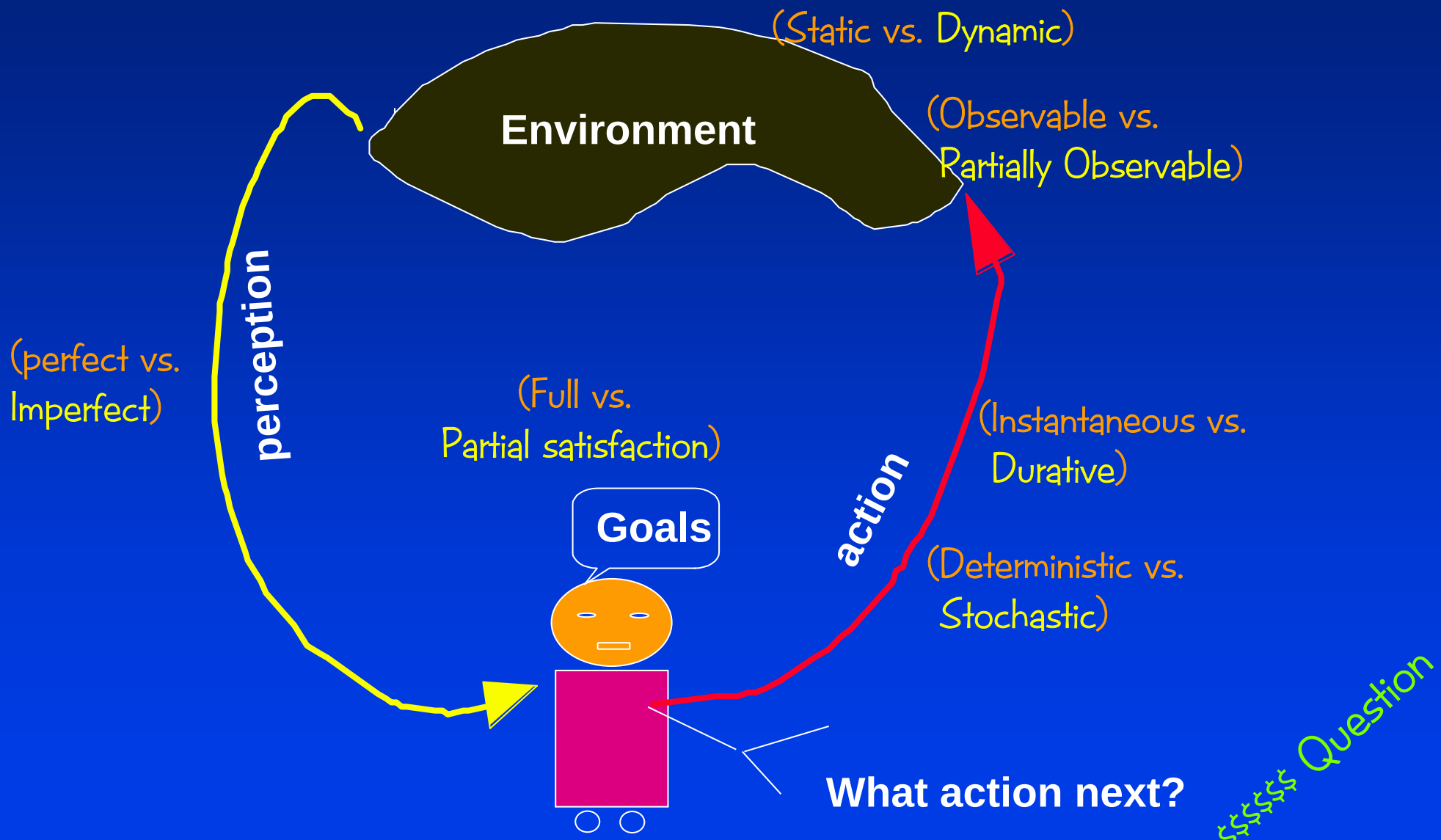
ASU Arizona State University

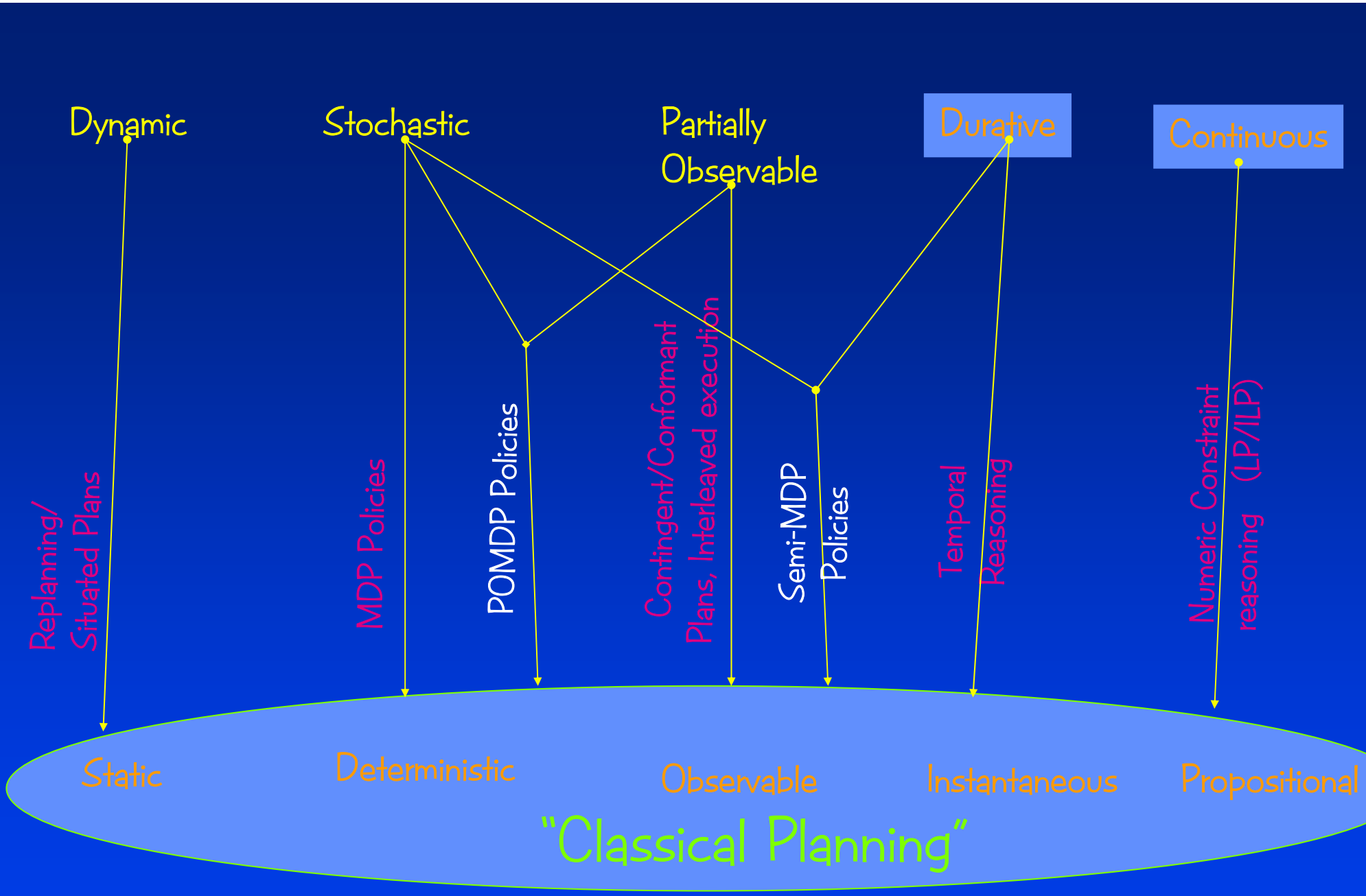
http://rakaposhi.eas.asu.edu

With thanks to all the Yochan group members

RMIT; 2/17/06

The Many Complexities of Planning




Dynamic
Stochastic
Partially Observable
Durative
Continuous
Replanning/ Situated Plans
MDP Policies
POMDP Policies
Contingent/Conformant Plans, Interleaved execution
Semi-MDP Policies
Temporal Reasoning
Numeric Constraint reasoning (LP/ILP)
Static
Deterministic
Observable
Instantaneous
Propositional
"Classical Planning"

Transition System Models

A transition system is a two tuple <S, A>

Where

S is a set of "states"

A is a set of "transitions"

each transition a is a subset of SXS

--If a is a (partial) function then deterministic transition

--otherwise, it is a "non-deterministic" transition

--It is a stochastic transition

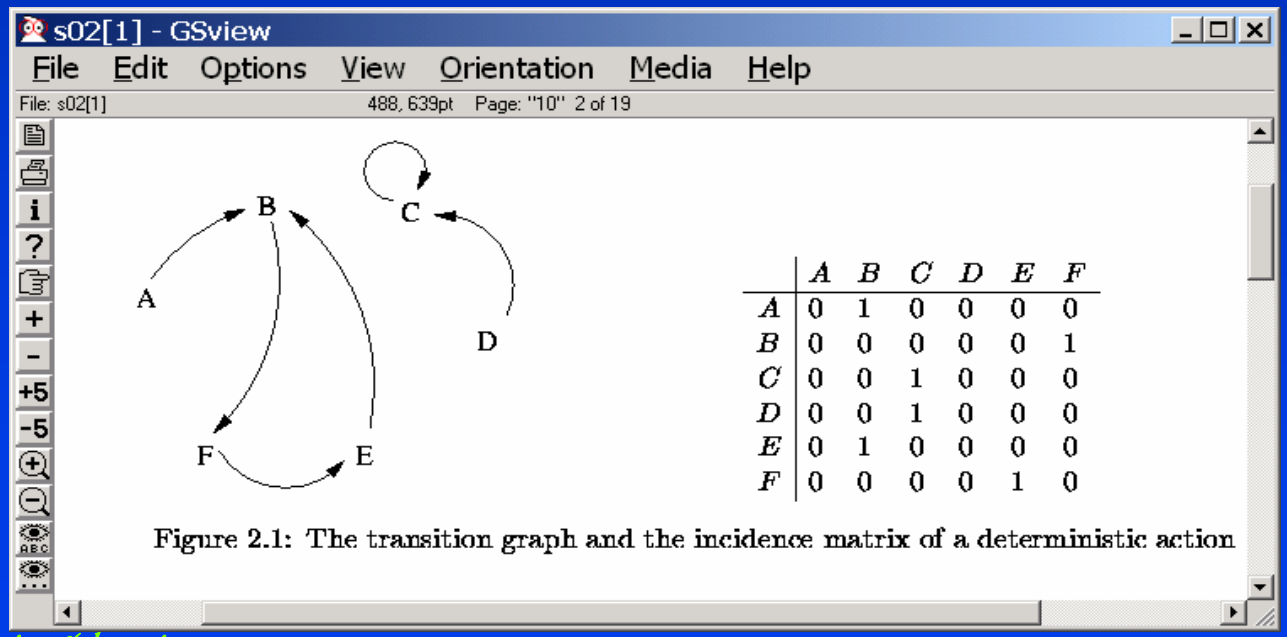
If there are probabilities associated with each state a takes s to

--Finding plans becomes is equivalent to finding "paths" in the transition system

Each action in this model can be Represented by incidence matrices (e.g. below)

The set of all possible transitions Will then simply be the SUM of the Individual incidence matrices

Transitions entailed by a sequence of actions will be given by the (matrix) multiplication of the incidence matrices



	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	0	0	0	1
C	0	0	1	0	0	0
D	0	0	1	0	0	0
E	0	1	0	0	0	0
F	0	0	0	0	1	0

Figure 2.1: The transition graph and the incidence matrix of a deterministic action

Transition system models are called "Explicit state-space" models

In general, we would like to represent the transition systems more compactly

e.g. State variable representation of states.

These latter are called "Factored" models

State Variable (Factored) Models

- Planning systems tend to use factored models (rather than direct transition models)
  - World is made up of states which are defined in terms of state variables
    - Can be boolean (or multi-ary or continuous)
  - States are *complete assignments over state variables*
    - So, k boolean state variables can represent how many states?
  - Actions change the values of the state variables
    - Applicability conditions of actions are also specified in terms of partial assignments over state variables

Blocks world

State variables:

Ontable(x) On(x,y) Clear(x) hand-empty holding(x)

Initial state:

Complete specification of T/F values to state variables

--By convention, variables with F values are omitted

Goal state:

A partial specification of the desired state variable/value combinations

--desired values can be both positive and negative

Init:
Ontable(A),Ontable(B),
Clear(A), Clear(B), hand-empty

Goal:
~clear(B), hand-empty

Pickup(x)
Prec: hand-empty,clear(x),ontable(x)
eff: holding(x),~ontable(x),~hand-empty,~Clear(x)

Putdown(x)
Prec: holding(x)
eff: Ontable(x), hand-empty,clear(x),~holding(x)

Stack(x,y)
Prec: holding(x), clear(y)
eff: on(x,y), ~cl(y), ~holding(x), hand-empty

Unstack(x,y)
Prec: on(x,y),hand-empty,cl(x)
eff: holding(x),~clear(x),clear(y),~hand-empty

PDDL Standard

```
(:action pick-up
        :parameters (?ob1)
        :precondition (and (clear ?ob1)
                           (on-table ?ob1)
                           (arm-empty)
                           (block ?ob1))
        :effect
        (and (not (on-table ?ob1))
               (not (clear ?ob1))
               (not (arm-empty))
               (holding ?ob1)))
```

PDDL standard under continual extension
(02) Support for time/durative actions
(04) Support for stochastic outcomes
(06) Support for soft constraints /preferences

```
(define (problem sussman-anomaly)
  (:domain blocks-world)
  (:objects A B C)
  (:init (block a) (block b) (block c)
   (on-table a) (on-table b) (on c a)
   (clear b) (clear c) (arm-empty))
  (:goal (and (on a b) (on b c))))
```

Progression:

An action A can be applied to state S iff the preconditions are satisfied in the current state

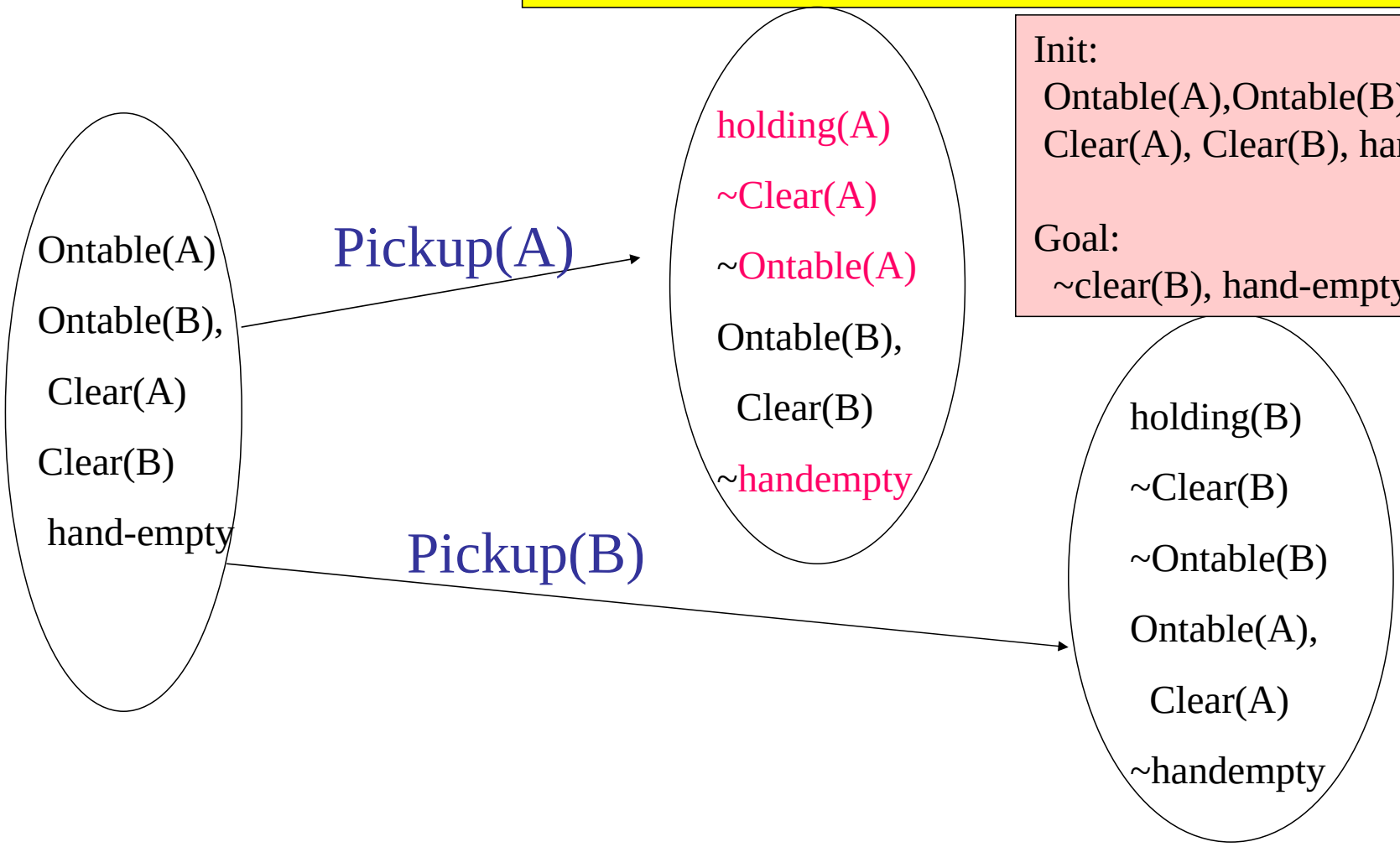
The resulting state S' is computed as follows:

--every variable that occurs in the actions effects gets the value that the action said it should have

--every other variable gets the value it had in the state S where the action is applied

Init:
Ontable(A),Ontable(B),
Clear(A), Clear(B), hand-empty

Goal:
~clear(B), hand-empty

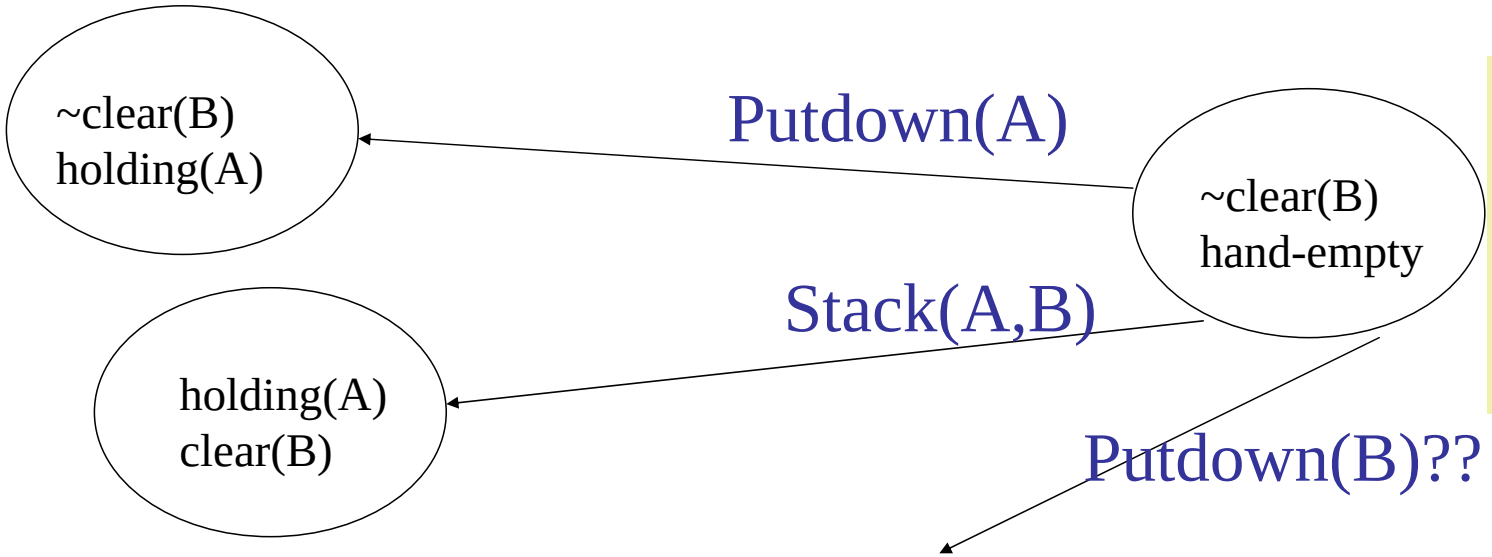


Regression:

Init:
Ontable(A),Ontable(B),
Clear(A), Clear(B), hand-empty

Goal:
~clear(B), hand-empty

A state S can be regressed over an action A
(or A is applied in the backward direction to S)
Iff:
--There is no variable v such that v is given different values by the effects of A and the state S
--There is at least one variable v' such that v' is given the same value by the effects of A as well as state S
The resulting state S' is computed as follows:
-- every variable that occurs in S, *and does not occur* in the effects of A will be copied over to S' with its value as in S
-- every variable that occurs in the precondition list of A will be copied over to S' with the value it has in in the precondition list



Termination test:
Stop when the state s' is *entailed by* the initial state s_I

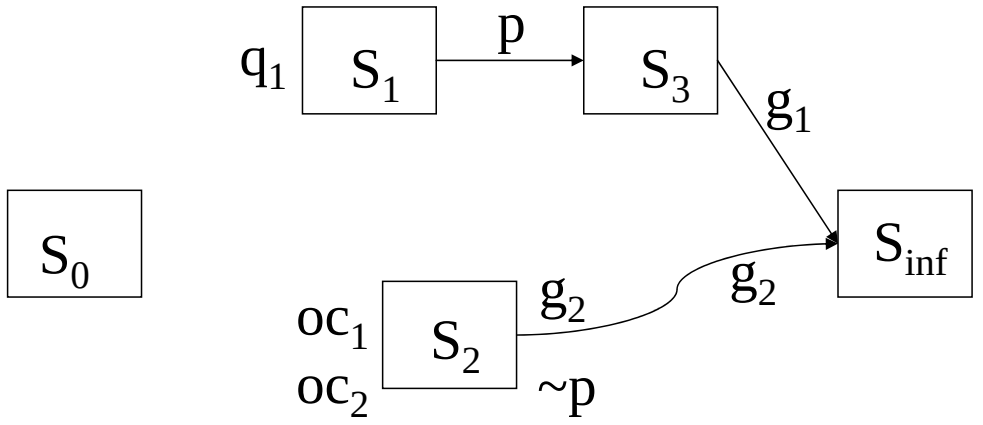
**Same entailment dir as before..*

POP Algorithm

1. *Plan Selection*: *Select a plan P from the search queue*
2. *Flaw Selection*: Choose a flaw f (open cond or unsafe link)
3. *Flaw resolution*:
   If f is an open condition,
      choose an action *S* that achieves *f*
   If f is an unsafe link,
      choose promotion or demotion
   Update P
   Return NULL if no resolution exist
4. *If* there is no flaw left, *return* P

1. Initial plan:

2. Plan refinement (flaw selection and resolution):

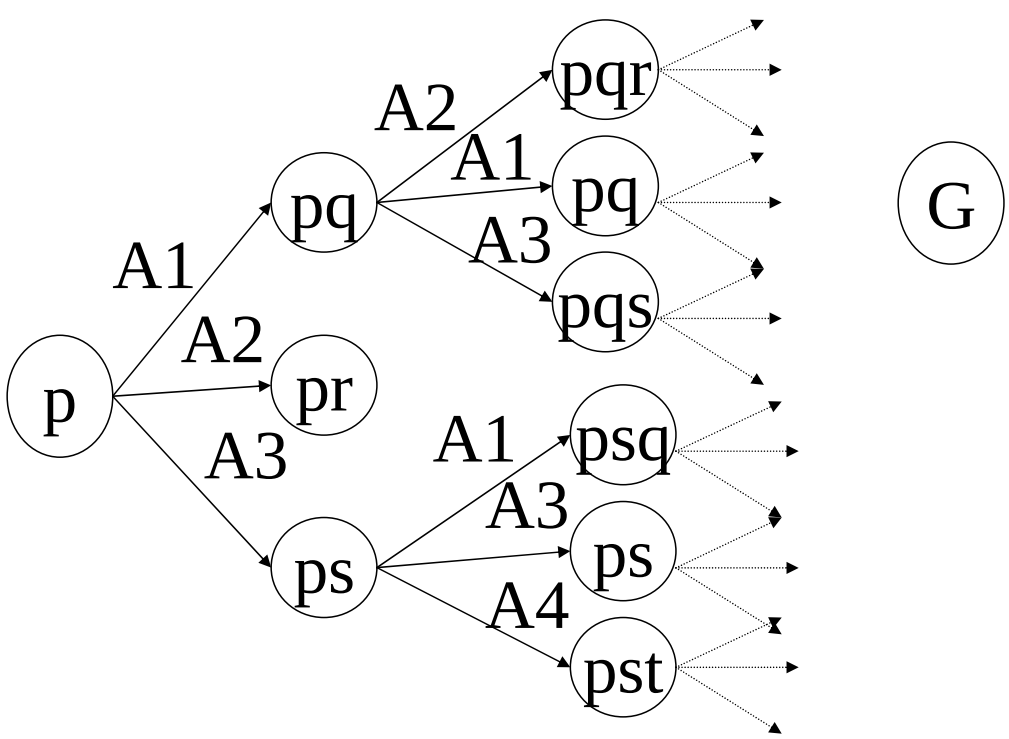


Choice points

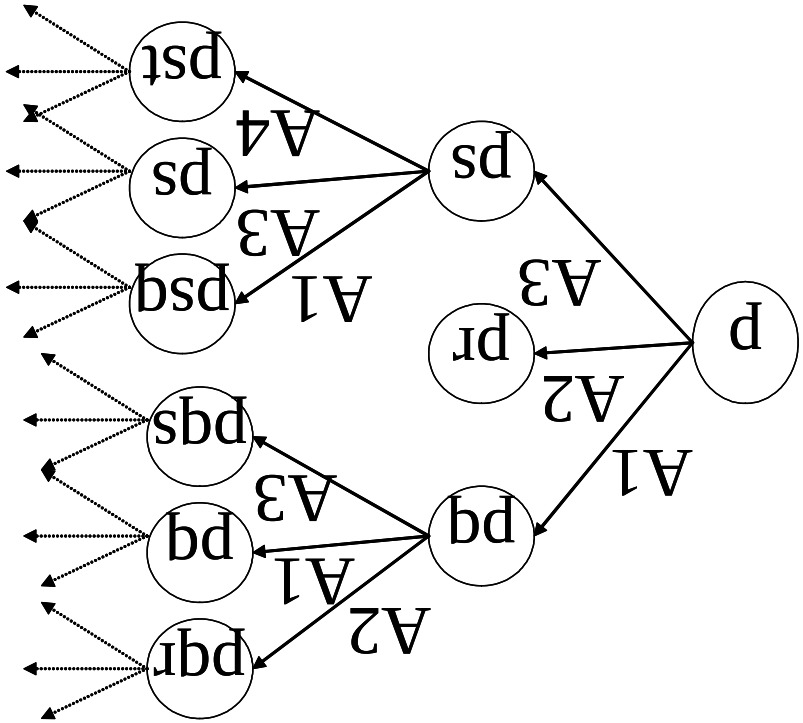
- Flaw selection *(open condition? unsafe link? Non-backtrack choice)*
- Flaw resolution/Plan Selection *(how to select (rank) partial plan?)*

Search & Control

Progression Search



Regression Search



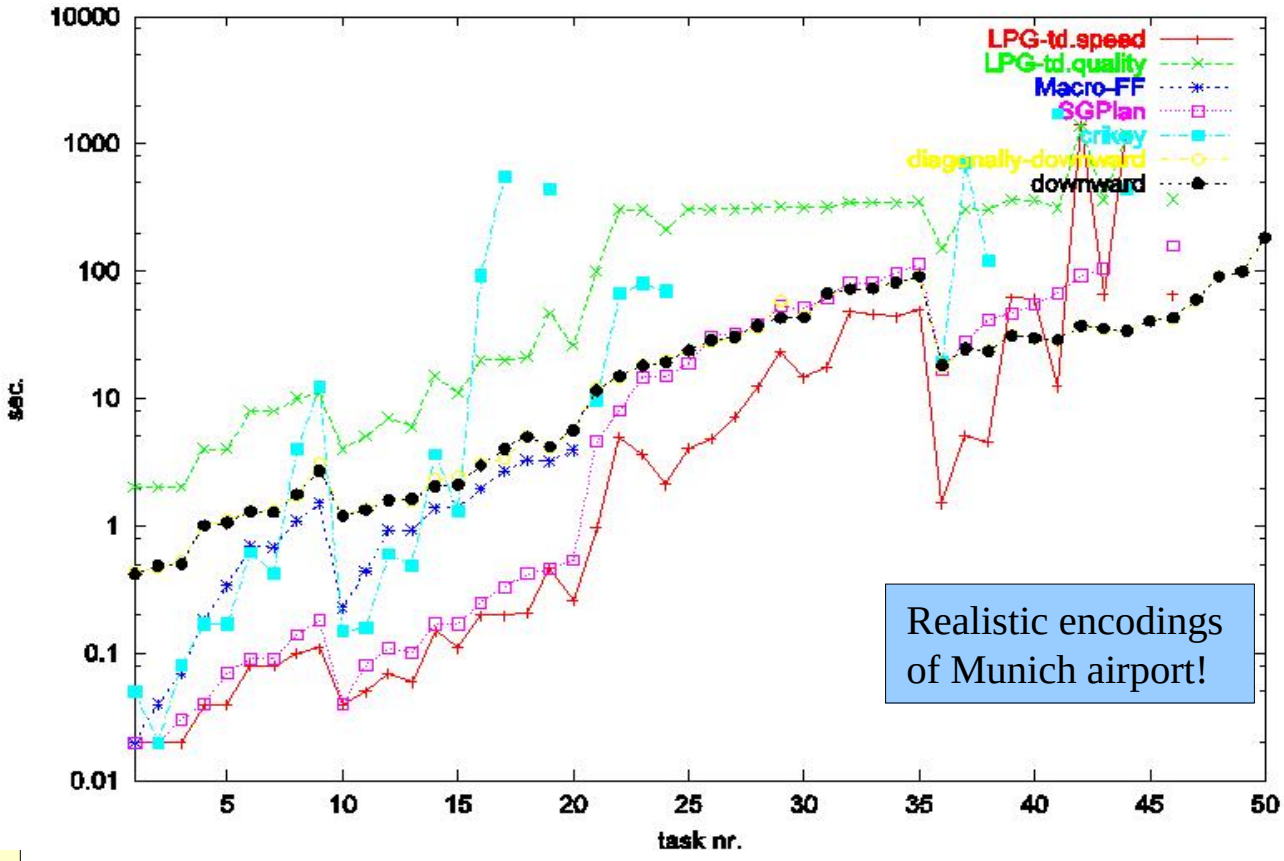
Which branch should we expand?

..depends on which branch is leading (closer) to the goal

Scalability of Planning

- Before, planning algorithms could synthesize about 6 – 10 action plans in minutes

- Significant scale-up in the last 6-7 years
  - Now, we can synthesize 100 action plans in seconds.



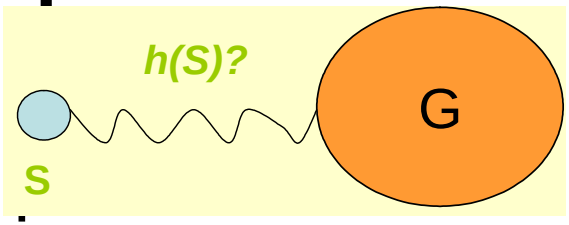
Realistic encodings of Munich airport!

Problem is Search Control!!!

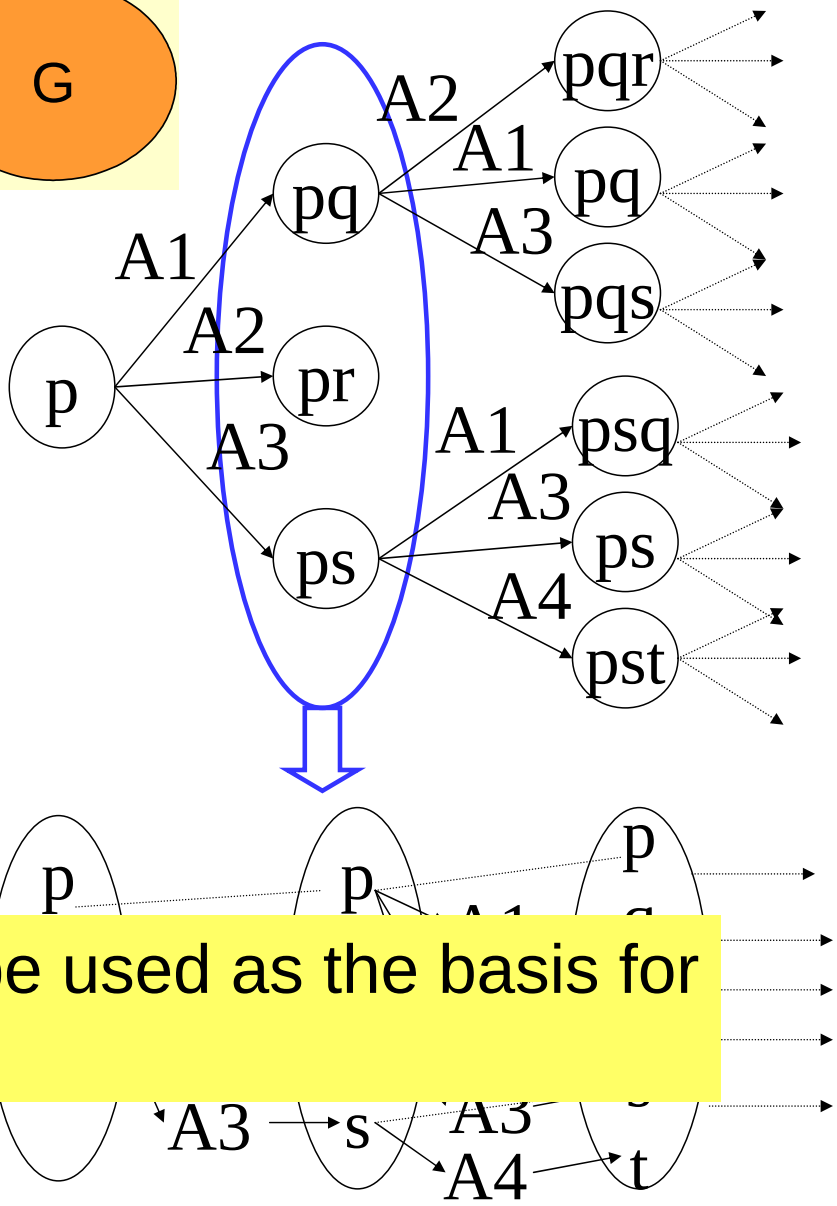
The primary revolution in planning in the recent years has been **domain-independent heuristics** to scale up plan synthesis

…and now for a ring-side retrospective ☺

Planning Graph and Projection



- Envelope of Progression Tree (Relaxed Progression)
  - Proposition lists: Union of states at k^{th} level
  - Mutex: Subsets of literals that cannot be
    y legal state
- L
  r



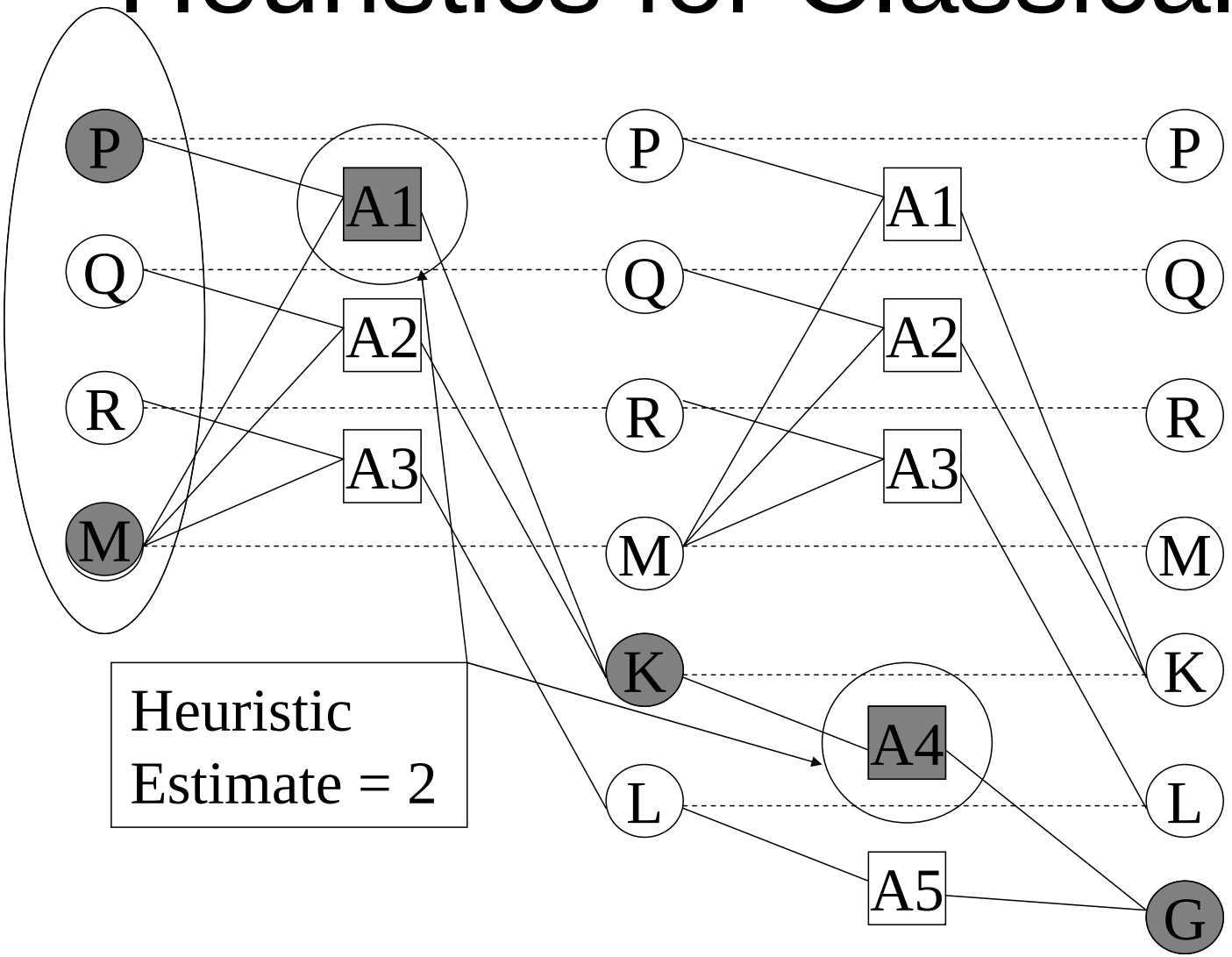
Planning Graphs can be used as the basis for heuristics!

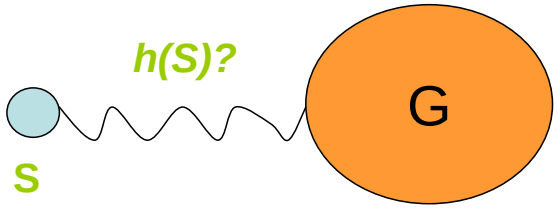
[Blum&Furst, 1995] [ECP, 1997]

1001 ways to skin a planning graph for heuristic fun & profit

- Classical planning
  - AltAlt (AAAI 2000; AIJ 2002); RePOP (IJCAI 2001); AltAlt^{p} (JAIR 2003)
  - Serial vs. Parallel graphs; Level and Adjusted heuristics; Partial expansion
- Over-subscription planning
  - Sapa^{Mps}; AltAlt^{Mps} (AAAI 2004; ICAPS 2005; IJCAI 2005)
  - Subgoal set selection using cost-sensitive relaxed plans
- Metric Temporal Planning
  - Sapa (ECP 2001; AIPS 2002; JAIR 2003); Sapa^{Mps} (IJCAI 2005)
  - Propagation of cost functions; Phased relaxation
- Nondeterministic Conformant/Conditional Planning
  - CAltAlt (ICAPS 2004); POND (AAAI 2004 wkshp)
  - Multiple graphs; Labelled uncertainty graphs; State-agnostic graphs
- Stochastic Conformant planning
  - Monte Carlo Labelled uncertainty graphs [ICAPS 2006]

Heuristics for Classical Planning

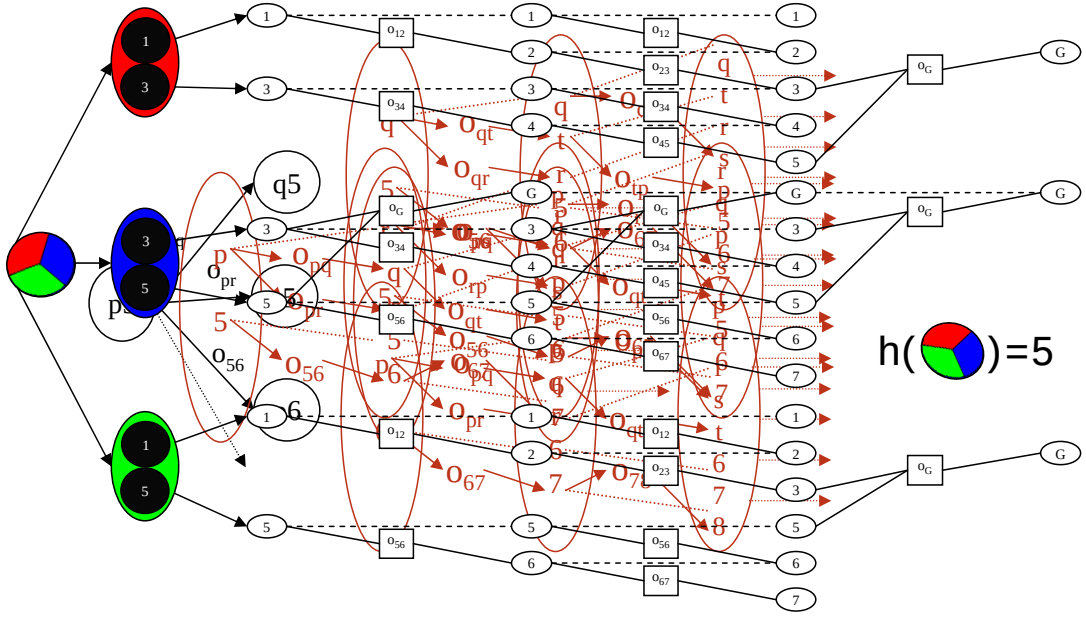




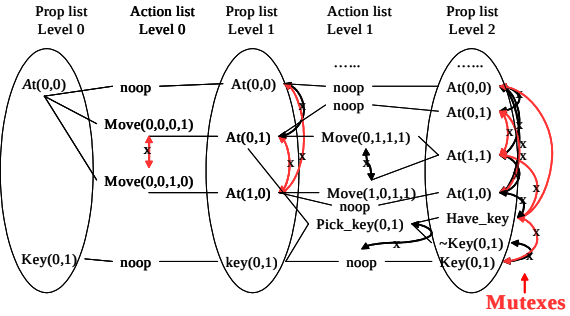
Relaxed plans are solutions for a relaxed problem

Planning Graphs for heuristics

- Construct planning graph(s) at each search node
  - Extract relaxed plan to achieve goal for heuristic



Cost of a Set of Literals



Degree of –ve interaction

$$\Delta(S) = h_{lev}(S) - h_{max}(S)$$

Admissible

h(S) = $\sum p \in S$ lev({p})

Sum

h(S) = lev(S)

Set-Level

Partition-k

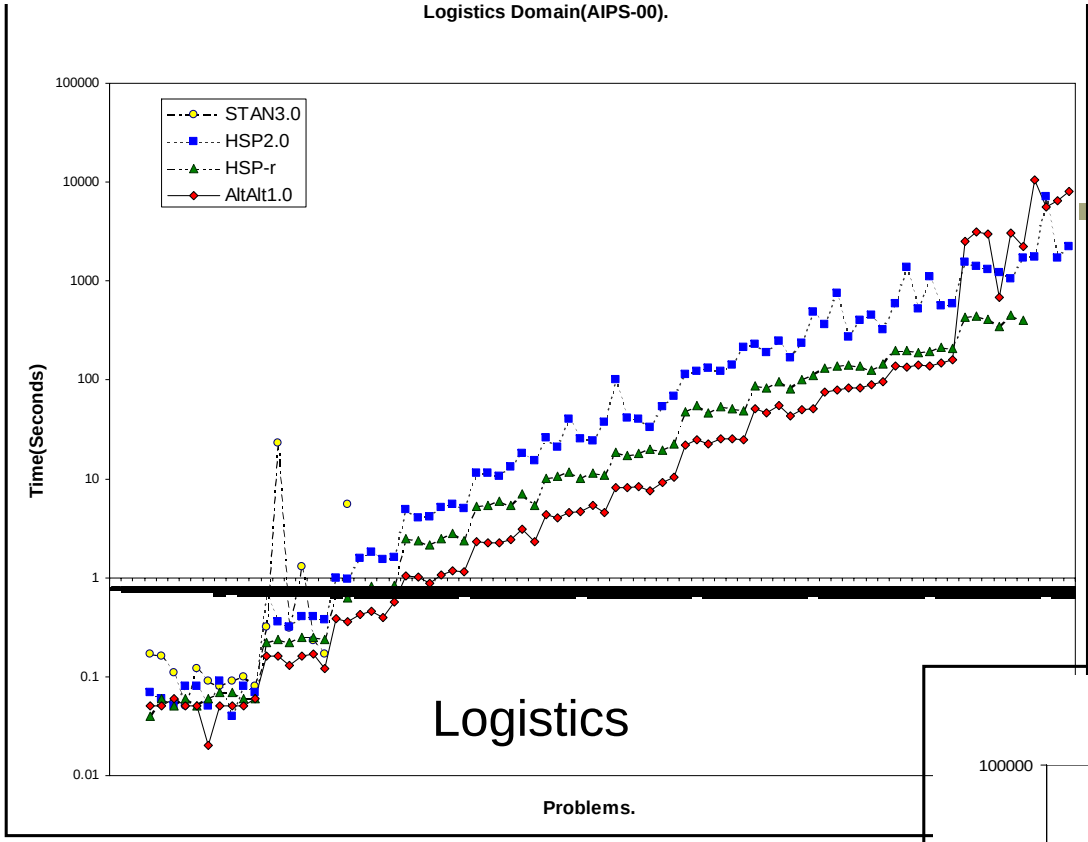
Adjusted Sum

Combo

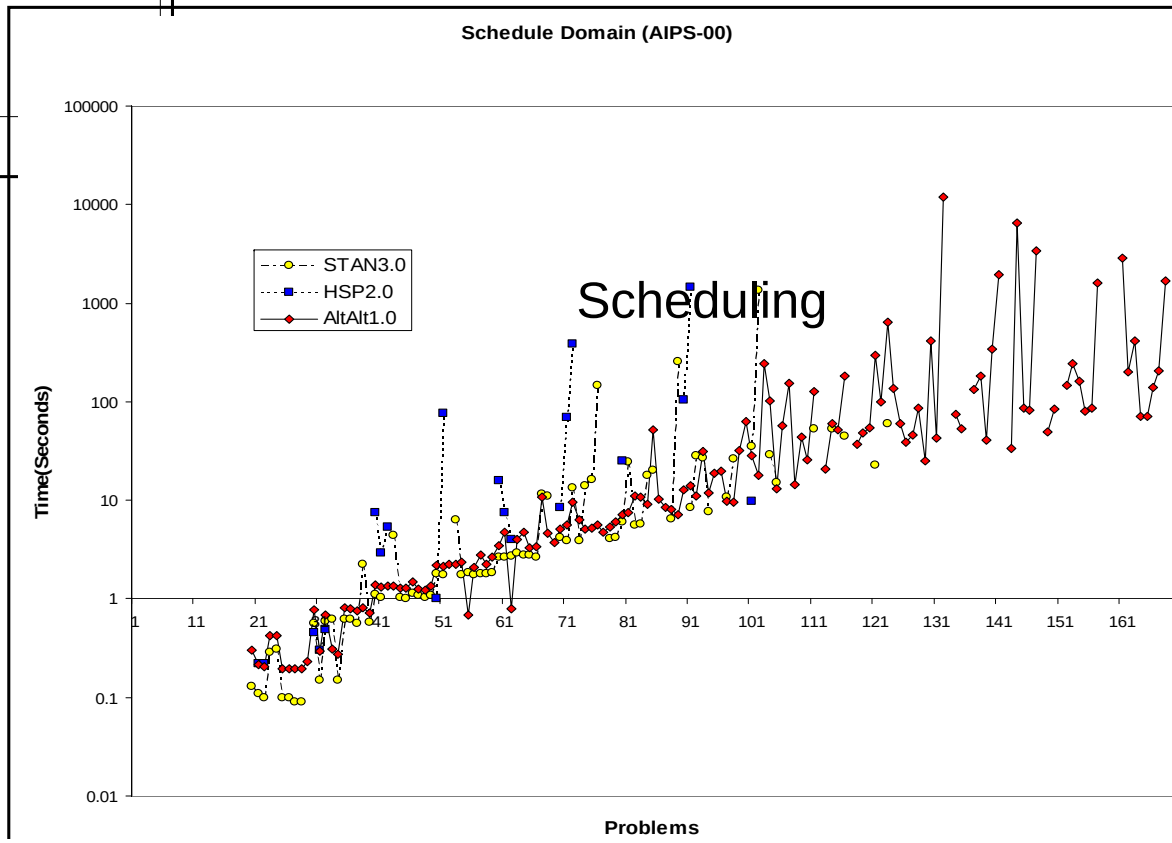
Set-Level with memos

Relaxed plan + Degree of –ve int

- lev(p) : index of the first level at which p comes into the planning graph
- lev(S): index of the first level where all props in S appear non-mutexed.



Combining the best of Planning Graphs and State Search ideas



Problem sets from IPC 2000

Planning graphs for heuristic estimation

A planning graph, once constructed, is a rich source of information about the problem. For example, *a literal that does not appear in the final level of the graph cannot be achieved by any plan*. This observation can be used in backward search as follows: any state containing an unachievable literal has a cost $h(n) = \infty$. Similarly, in partial-order planning, any plan with an unachievable open condition has $h(n) = \infty$.

This idea can be made more general. We can estimate the cost of achieving any goal literal as the level at which it first appears in the planning graph. We will call this the **level cost** of the goal. In Figure 11.12, $Have(Cake)$ has level cost 0 and $Eaten(Cake)$ has level cost 1. It is easy to show (Exercise 11.9) that these estimates are admissible for the individual goals. The estimate might not be very good, however, because planning graphs allow several actions at each level whereas the heuristic counts just the level and not the number of actions. For this reason, it is common to use a **serial planning graph** for computing heuristics. A

LEVEL COST

SERIAL PLANNING GRAPH

serial graph insists that only one ac
by adding mutex links between ev
extracted from serial graphs are of

To estimate the cost of a con
max-level heuristic simply takes
missible, but not necessarily very
independence assumption, returns the sum of the level costs of the goals; this is inadmissible but works very well in practice for problems that are largely decomposable. It is much more accurate than the number-of-unsatisfied-goals heuristic from Section 11.2. For our problem, the heuristic estimate for the conjunctive goal $Have(Cake) \wedge Eaten(Cake)$ will be $0 + 1 = 1$, whereas the correct answer is 2. Moreover, if we eliminated the $Bake(Cake)$ action, the estimate would still be 1, but the conjunctive goal would be impossible. Finally, the **set-level** heuristic finds the level at which all the literals in the conjunctive goal appear in the planning graph without any pair of them being mutually exclusive. This heuristic gives the correct values of 2 for our original problem and infinity for the problem without $Bake(Cake)$. It dominates the max-level heuristic and works extremely well on tasks in which there is a good deal of interaction among subplans.

MAX-LEVEL

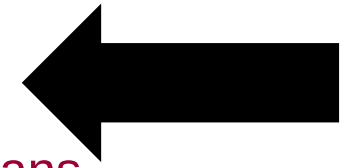
LEVEL SUM

SET-LEVEL

Nguyen *et al.* (2001) give a very thorough analysis of heuristics derived from planning graphs. Our discussion of planning graphs is based partly on this work and on lecture notes by Subbarao Kambhampati.

1001 ways to skin a planning graph for heuristic fun & profit

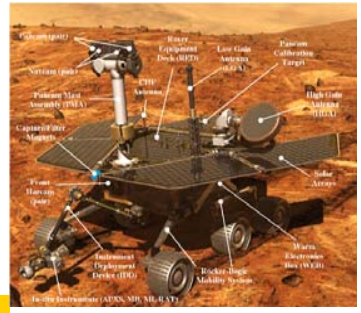
- Classical planning
  - AltAlt (AAAI 2000; AIJ 2002); RePOP (IJCAI 2001); AltAlt^{p} (JAIR 2003)
  - Serial vs. Parallel graphs; Level and Adjusted heuristics; Partial expansion
- Over-subscription planning
  - Sapa^{Mps}; AltAlt^{Mps} (AAAI 2004; ICAPS 2005; IJCAI 2005)
  - Subgoal set selection using cost-sensitive relaxed plans
- Metric Temporal Planning
  - Sapa (ECP 2001; AIPS 2002; JAIR 2003); Sapa^{Mps} (IJCAI 2005)
  - Propagation of cost functions; Phased relaxation
- Nondeterministic Conformant/Conditional Planning
  - CAltAlt (ICAPS 2004); POND (AAAI 2004 wkshp)
  - Multiple graphs; Labelled uncertainty graphs; State-agnostic graphs
- Stochastic Conformant planning
  - Monte Carlo Labelled uncertainty graphs [ICAPS 2006]



PG Heuristics for Oversubscription (Partial Satisfaction) Planning

[AAAI-04,ICAPS-05;IJCAI-05]

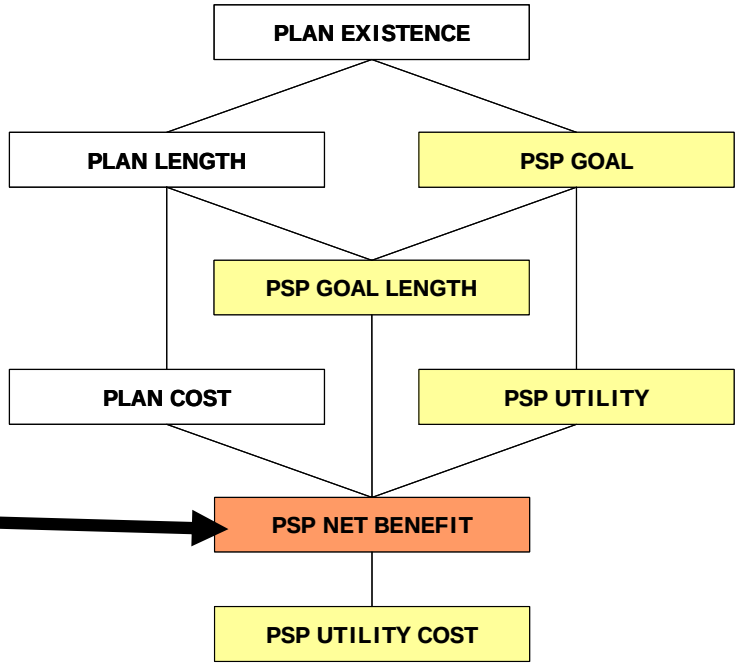
In many real world planning tasks, the agent often has more goals than it has resources to accomplish.



Example: Rover Mission Planning (MER) involved

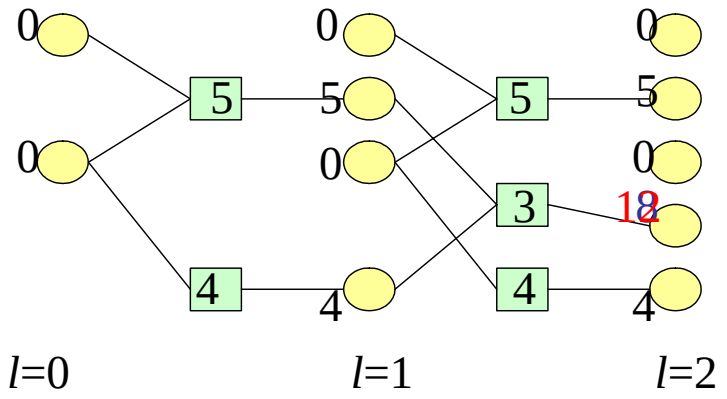
Need automated support for Over-subscription/Partial Satisfaction Planning

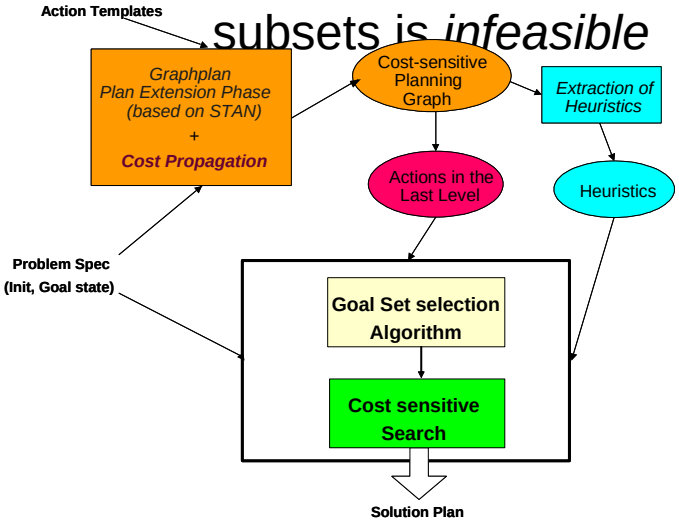
Actions have execution costs, goals have utilities, and the objective is to find the plan that has the highest net benefit.



Adapting PG heuristics for PSP

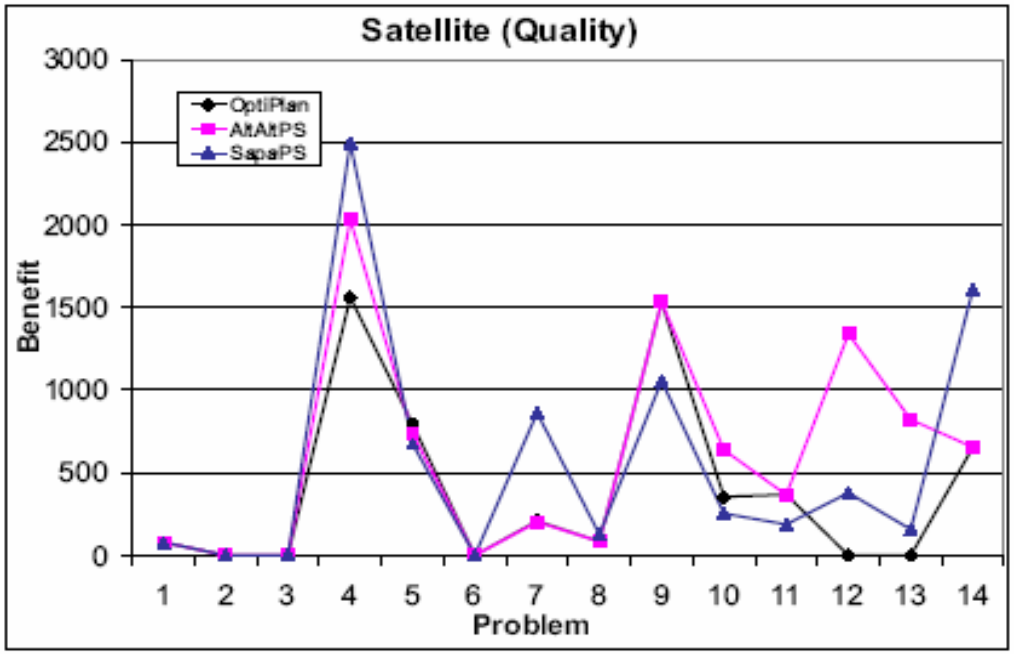
- **Challenges:**
  - **Need to propagate costs on the planning graph**
  - The exact set of goals are not clear
    - Interactions between goals
    - Obvious approach of considering all 2^n goal subsets is *infeasible*

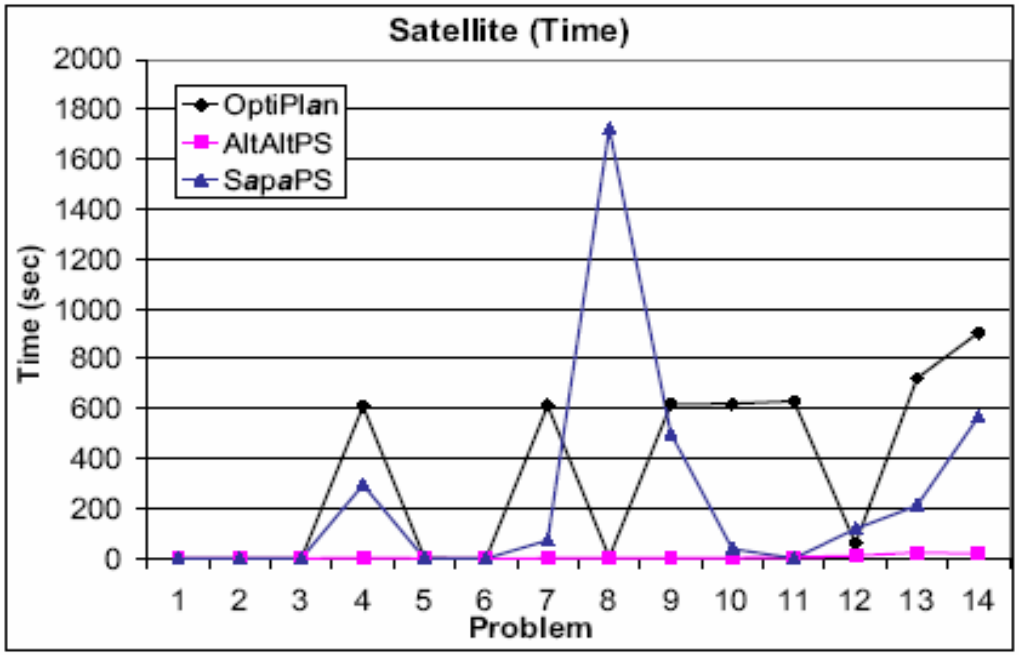




- **Idea:** Select a subset of the top level goals upfront
- Challenge: Goal interactions
  - Approach: Estimate the net benefit of each goal in terms of its utility minus the cost of its relaxed plan
    - Bias the relaxed plan extraction to (re)use the actions already chosen for other goals

Some Empirical Results for AltAlt^{ps}





Exact algorithms based on MDPs don't scale at all

[AAAI 2004]

PSP+MTP=SAPA$^{\mathcal{M}ps}$

[IJCAI 2005]

- In MTP, PSP will involve
  - Partial Degree of satisfaction
    - If you can't give me 1000$, give me half at least
    - Need to track costs for various intervals of a numeric quantity ☹
  - Delayed Satisfaction
    - If you submit the homework past the deadline, you will get penalty points

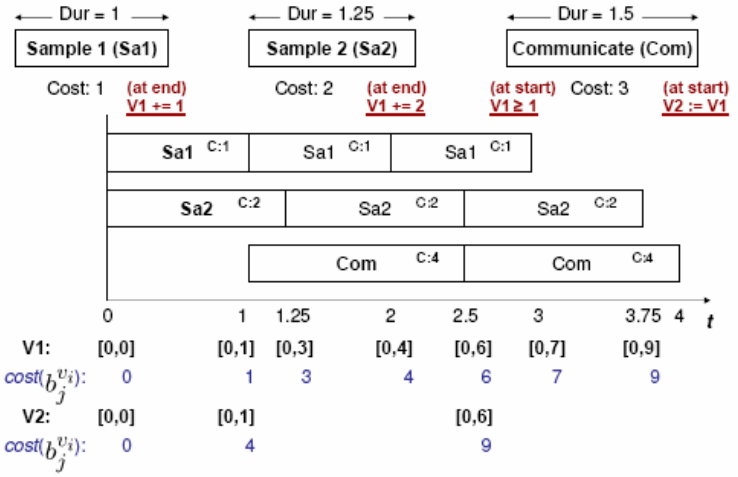


Figure 3: The RTPG for our example. Our actions are defined above it.

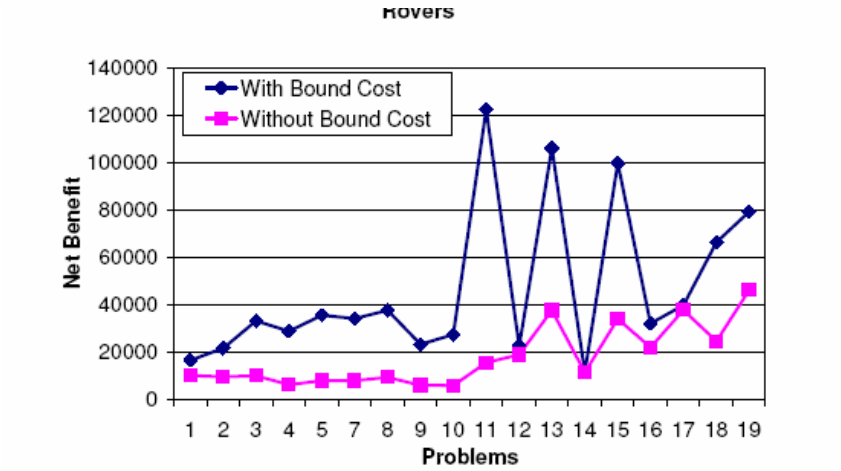
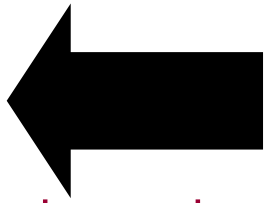


Figure 4: Comparison of utilities for our rovers domain

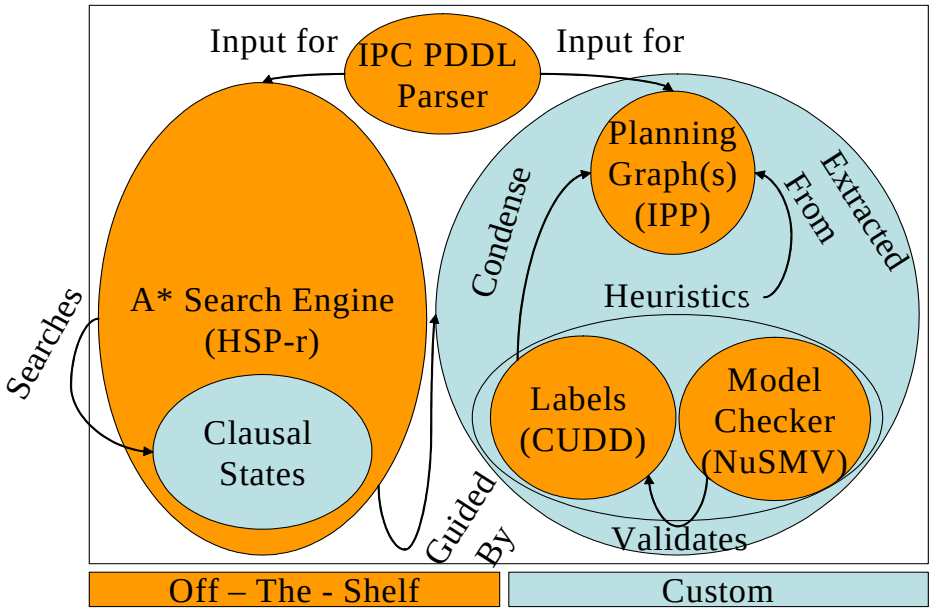
1001 ways to skin a planning graph for heuristic fun & profit

- Classical planning
  - AltAlt (AAAI 2000; AIJ 2002); RePOP (IJCAI 2001); AltAlt^{p} (JAIR 2003)
  - Serial vs. Parallel graphs; Level and Adjusted heuristics; Partial expansion
- Over-subscription planning
  - Sapa^{Mps}; AltAlt^{Mps} (AAAI 2004; ICAPS 2005; IJCAI 2005)
  - Subgoal set selection using cost-sensitive relaxed plans
- Metric Temporal Planning
  - Sapa (ECP 2001; AIPS 2002; JAIR 2003); Sapa^{Mps} (IJCAI 2005)
  - Propagation of cost functions; Phased relaxation
- Nondeterministic Conformant/Conditional Planning
  - CAltAlt (ICAPS 2004); POND (AAAI 2004 wkshp)
  - Multiple graphs; Labelled uncertainty graphs; State-agnostic graphs
- Stochastic Conformant planning
  - Monte Carlo Labelled uncertainty graphs

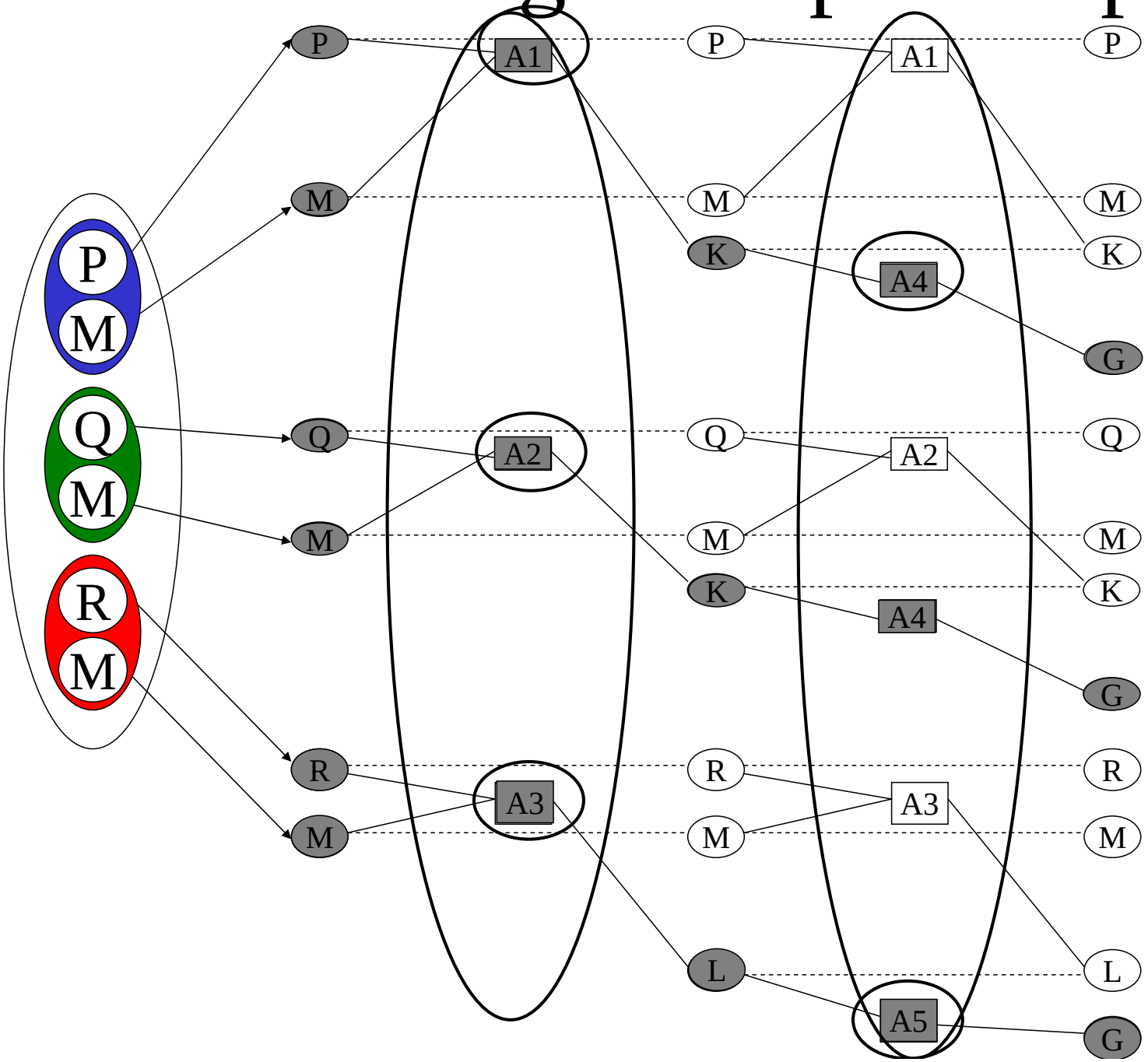


III. PG Heuristics for Belief Space Conformant/Conditional Planning

- Partially known initial state
- Actions with non-deterministic effects
- Need to search in Belief Space
  - Belief States are sets of world states (2^S)
    - Even more need for search control
      - Represented as formulas over fluents (implemented as BDDs)



Using Multiple Graphs

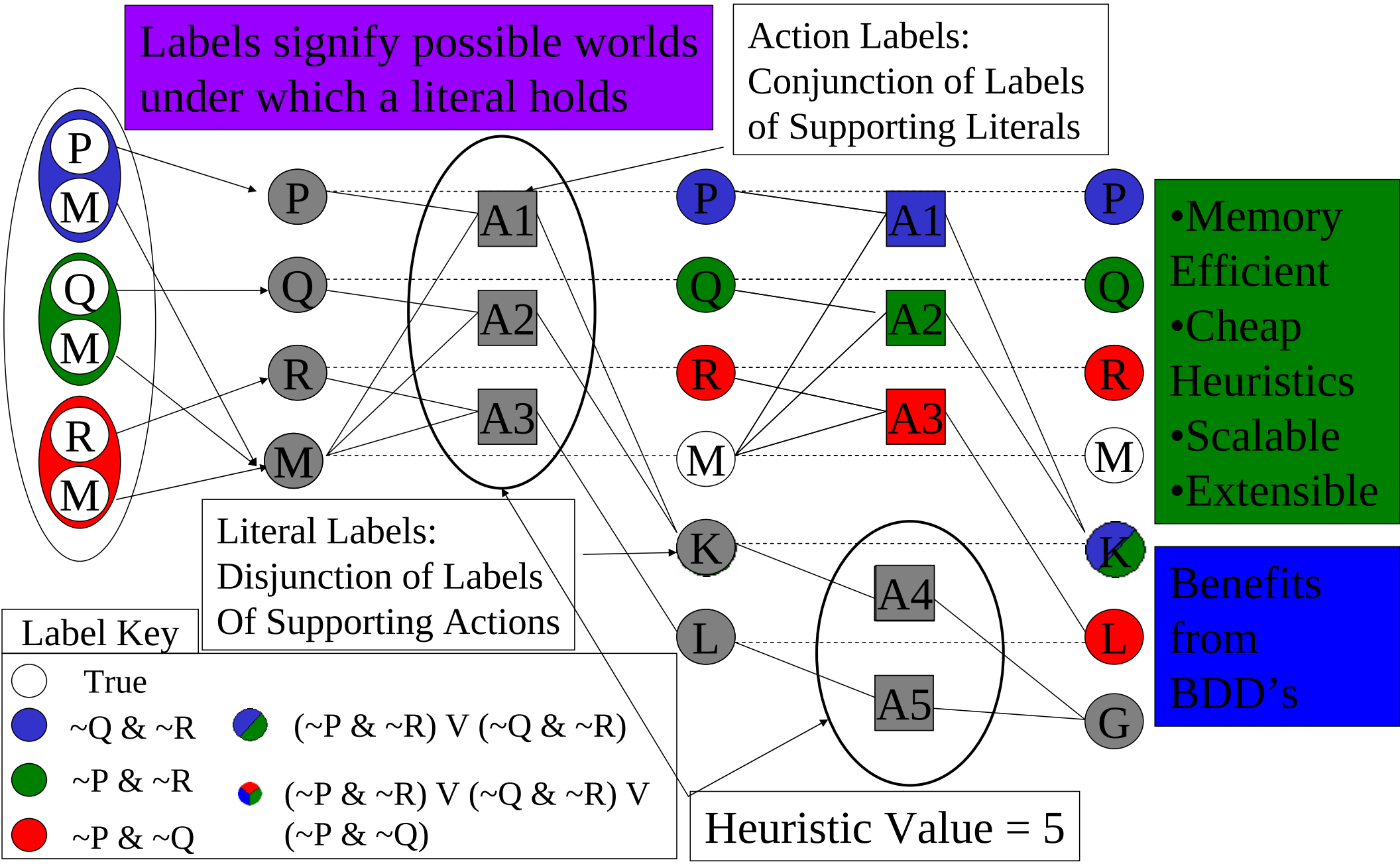


- Same-world Mutexes

- Memory Intensive
- Heuristic Computation Can be costly

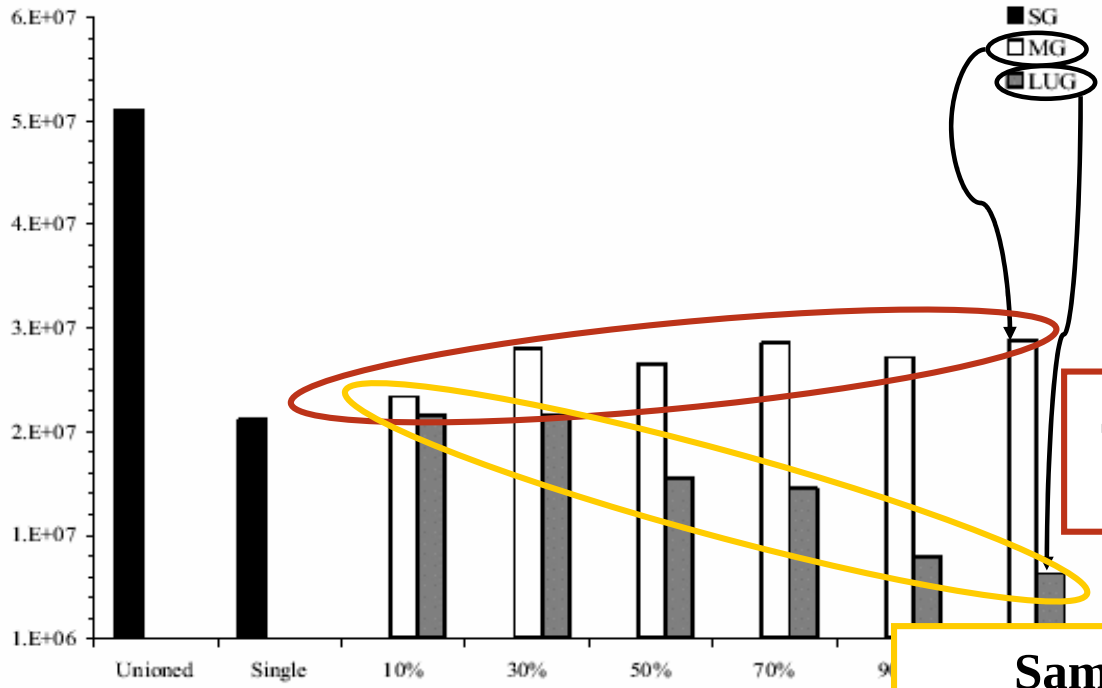
Unioning these graphs a priori would give much savings …

Using a Single, Labeled Graph



Comparison of Planning Graph Types

[Bryce *et.al*, 2005]



LUG saves Time

Sampling more worlds, Increases cost of MG

Sampling more worlds, Improves effectiveness of LUG

Figure 15: Total Time (ms) to solve all problems when sampling worlds to use in heuristic computation.

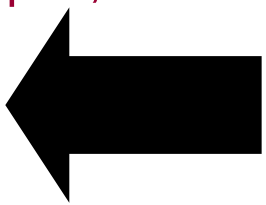
[AAAI 2005]

State Agnostic Graphs

- Labelled graphs handle “state uncertainty” using labels on the PG elements
- But the same idea can be use to handle “search uncertainty”
  - We can compute a labelled graph that gives us reachability information from any set of states—including the set of *all reachable states*
    - Such state agnostic graphs do “all pairs shortest path” analysis (as against single source shortest path analysis done by normal PG).

1001 ways to skin a planning graph for heuristic fun & profit

- Classical planning
  - AltAlt (AAAI 2000; AIJ 2002); RePOP (IJCAI 2001); AltAlt^{p} (JAIR 2003)
  - Serial vs. Parallel graphs; Level and Adjusted heuristics; Partial expansion
- Over-subscription planning
  - Sapa^{Mps}; AltAlt^{Mps} (AAAI 2004; ICAPS 2005; IJCAI 2005)
  - Subgoal set selection using cost-sensitive relaxed plans
- Metric Temporal Planning
  - Sapa (ECP 2001; AIPS 2002; JAIR 2003); Sapa^{Mps} (IJCAI 2005)
  - Propagation of cost functions; Phased relaxation
- Nondeterministic Conformant/Conditional Planning
  - CAltAlt (ICAPS 2004); POND (AAAI 2004 wkshp)
  - Multiple graphs; Labelled uncertainty graphs; State-agnostic graphs
- Stochastic Conformant planning
  - Monte Carlo Labelled uncertainty graphs



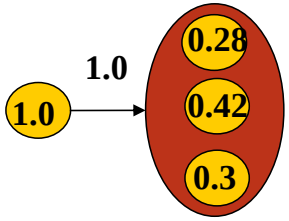
Probabilistic Planning

- 2 views on solution criteria:
  1. Plan with optimal p(G) given length bound k
     - Addressed by SAT/CSP techniques or k steps of POMDP value iteration
     - NP^{PP}-Complete – finite search space
  2. Plan with p(G) no less than τ
     - Addressed by heuristic search, POCL planners
     - Undecideable – infinite search space
  - Iterated values of k in 1 can address 2
  - Strictly optimizing p(G) or cost (without constraints) can lead to infinite or zero length plans, respectively

Heuristic Search for Conformant PP (CPP)

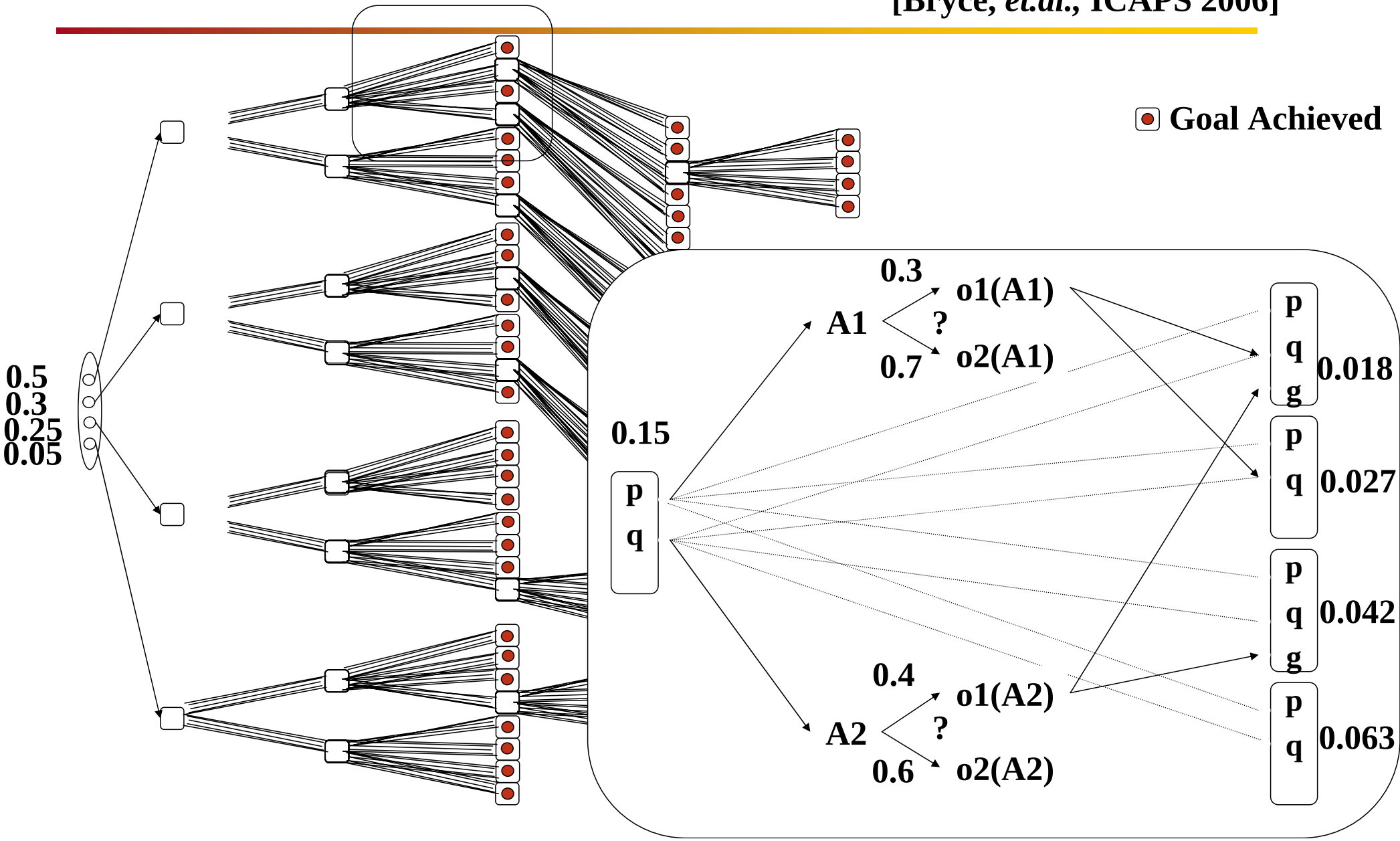
[Bryce, *et.al.*, ICAPS 2006]

- Forward-chaining A* search in space of probabilistic belief states
  - Terminal nodes: $p(G) \geq \tau$
  - Compute h-value with novel relaxed plan heuristic



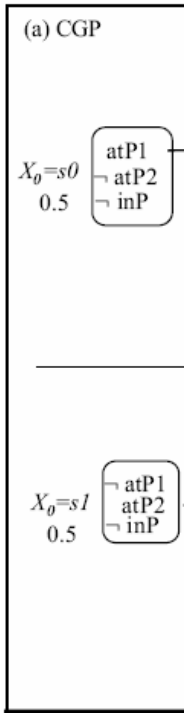
CPP reachability heuristics

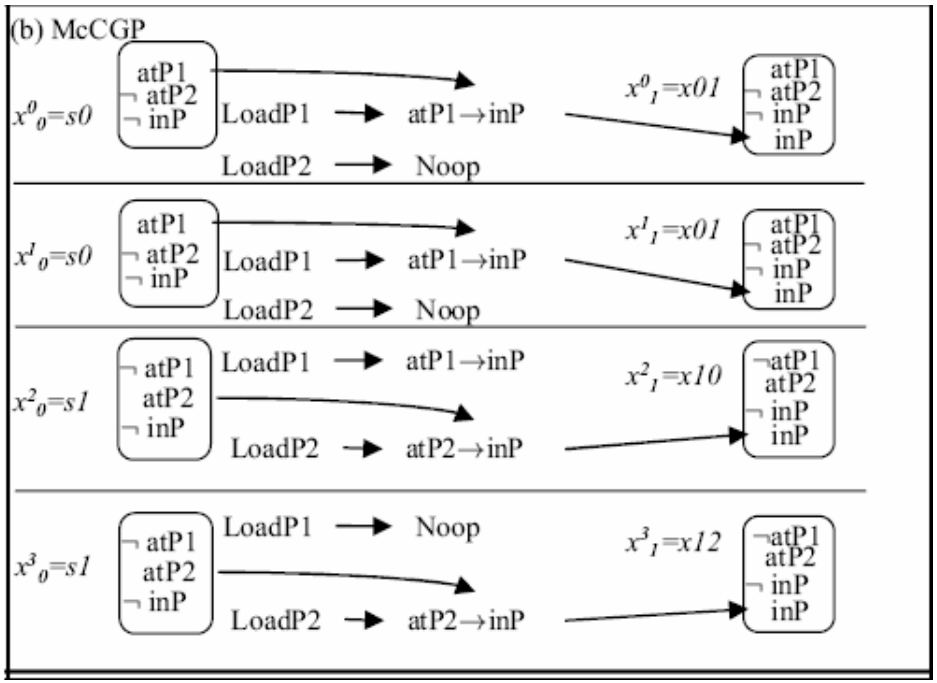
[Bryce, *et.al.*, ICAPS 2006]

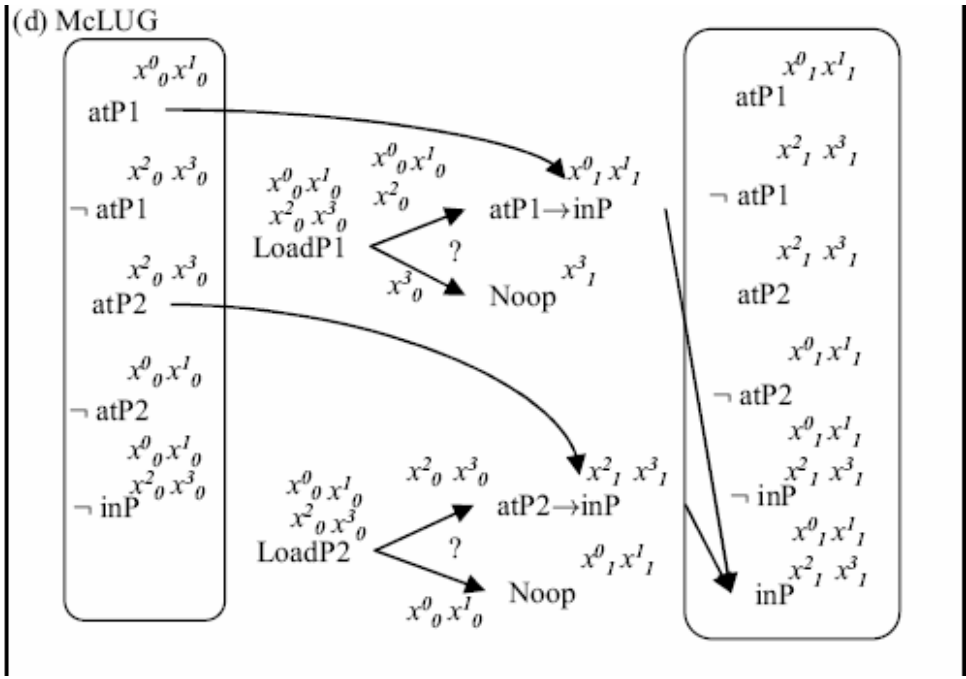


Reachability for Stocha

$$\text{LoadP1} = \langle\{\}, \{\langle 0.8, \{\text{atP1}\rightarrow\text{inP}\}\rangle, \langle 0.2, \{\text{Noop}\}\rangle\}\rangle$$
$$\text{LoadP2} = \langle\{\}, \{\langle 0.8, \{\text{atP2}\rightarrow\text{inP}\}\rangle, \langle 0.2, \{\text{Noop}\}\rangle\}\rangle$$



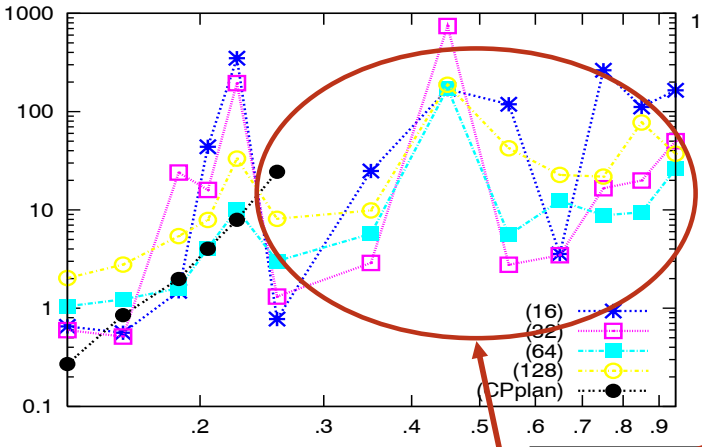




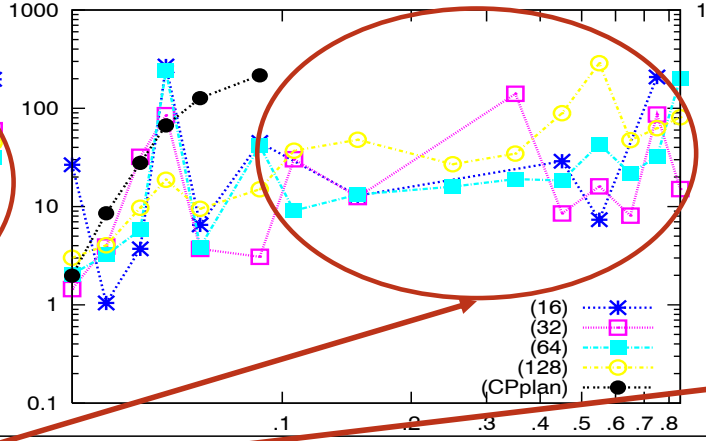
Logistics Domain Results

[Bryce, *et.al.*, submitted]

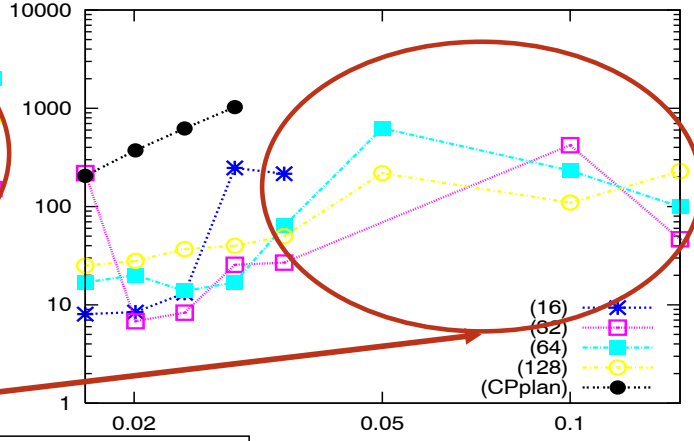
P2-2-2 time (s)



P4-2-2 time (s)

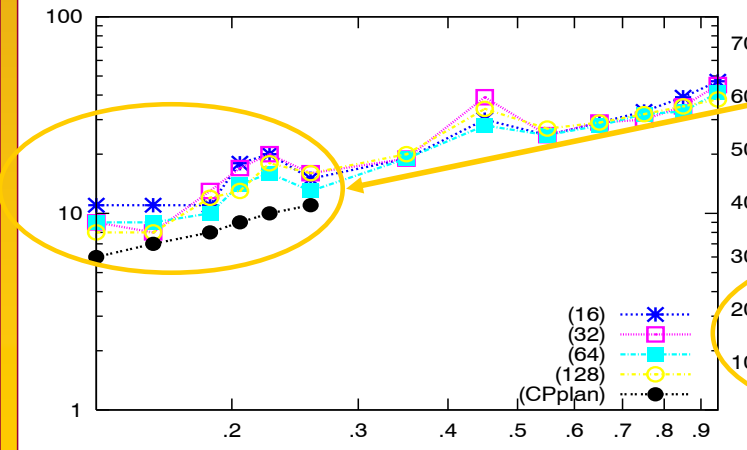


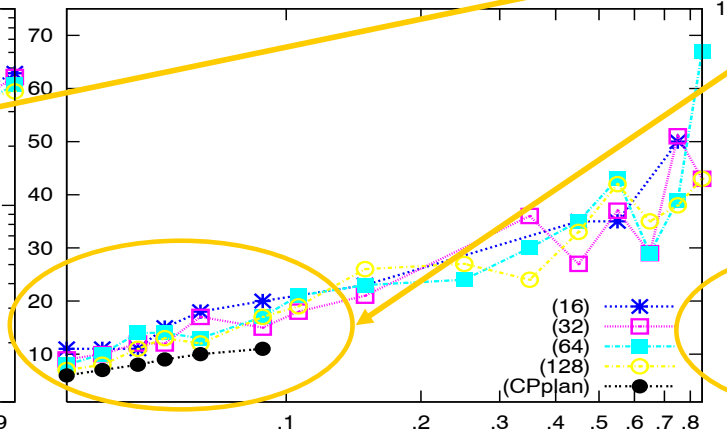
P2-2-4 time (s)



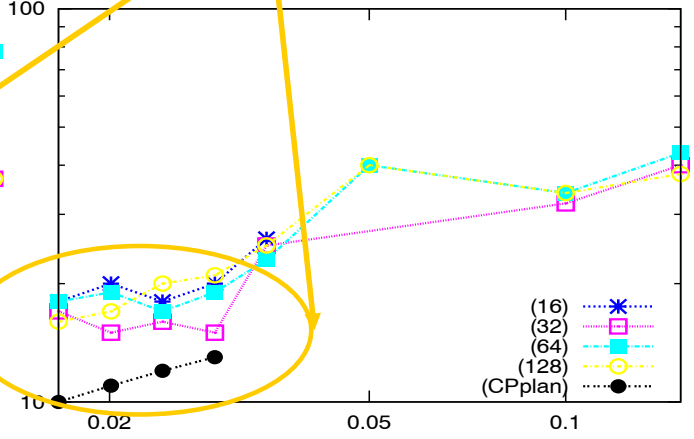
Scalable, w/ reasonable quality

P2-2-2 length





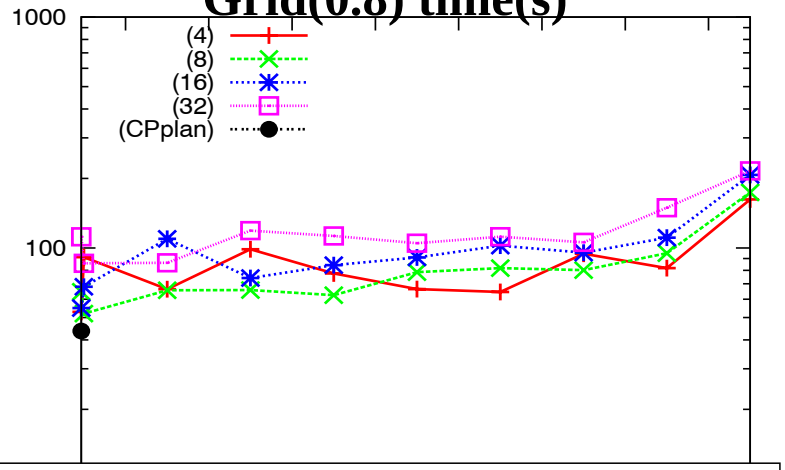
P2-2-4 length



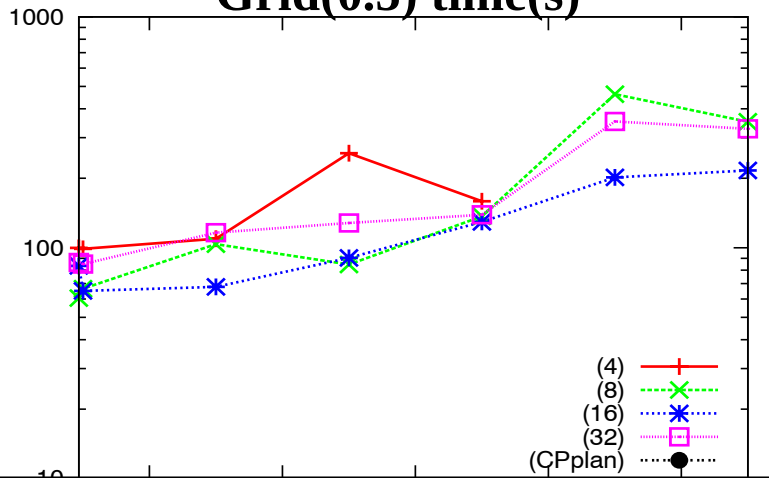
Grid Domain Results

[Bryce, *et.al.*, submitted]

Grid(0.8) time(s)



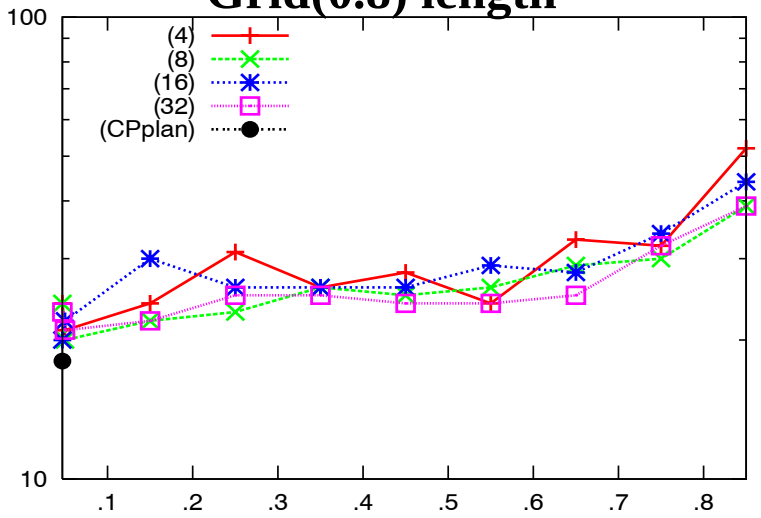
Grid(0.5) time(s)

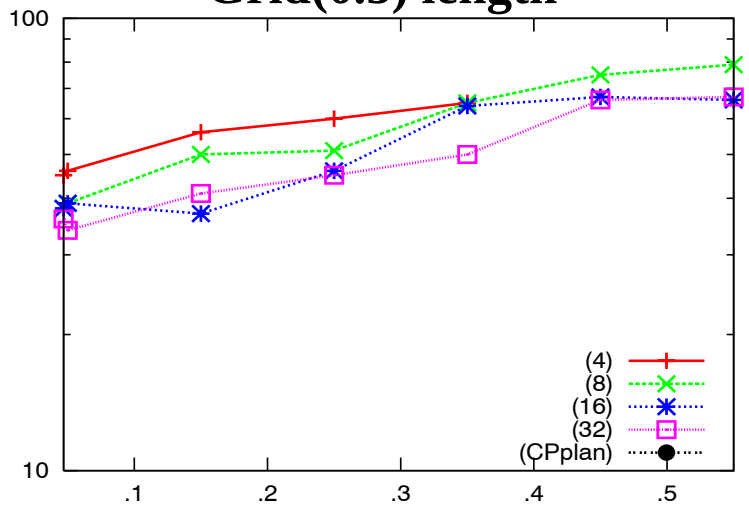


Again, good scalability and quality!

Need More Particles for broad beliefs

Grid(0.8) length





- PG Variations
  - Serial
  - Parallel
  - Temporal
  - Labelled
  - State Agnostic
  - Monte-Carlo Labelled

- Propagation Methods
  - Level
  - Mutex
  - Cost
  - Label

Versatility of PG Heuristics

- Planning Problems
  - Classical
  - Resource/Temporal
  - Conformant/Conditional
  - Over-subscription
  - Stochastic

- Planners
  - Regression
  - Progression
  - Partial Order
  - Graphplan-style