Audio and pptx versions available at
http://rakaposhi.eas.asu.edu/tmp/onr-ipam

Incomplete Domain Models, Uncertain Users, Unending Planning & Open Worlds

Model-Lite Planning for Autonomy in the presence of Humans

Subbarao Kambhampati

Arizona State University

Funding from
- ONR Model-lite Planning (PM: Behzad Kamgar-Parsi)
- ONR Information Integration (PM: Don Wagner)
- ONR MURI (PM: Tom McKenna)

The “Boxers” or “Briefs” Qn of this workshop…

- Bio-inspired folks
- Pro-human folks
- Mechano-philes

When there is an elephant in the room, *introduce* it…

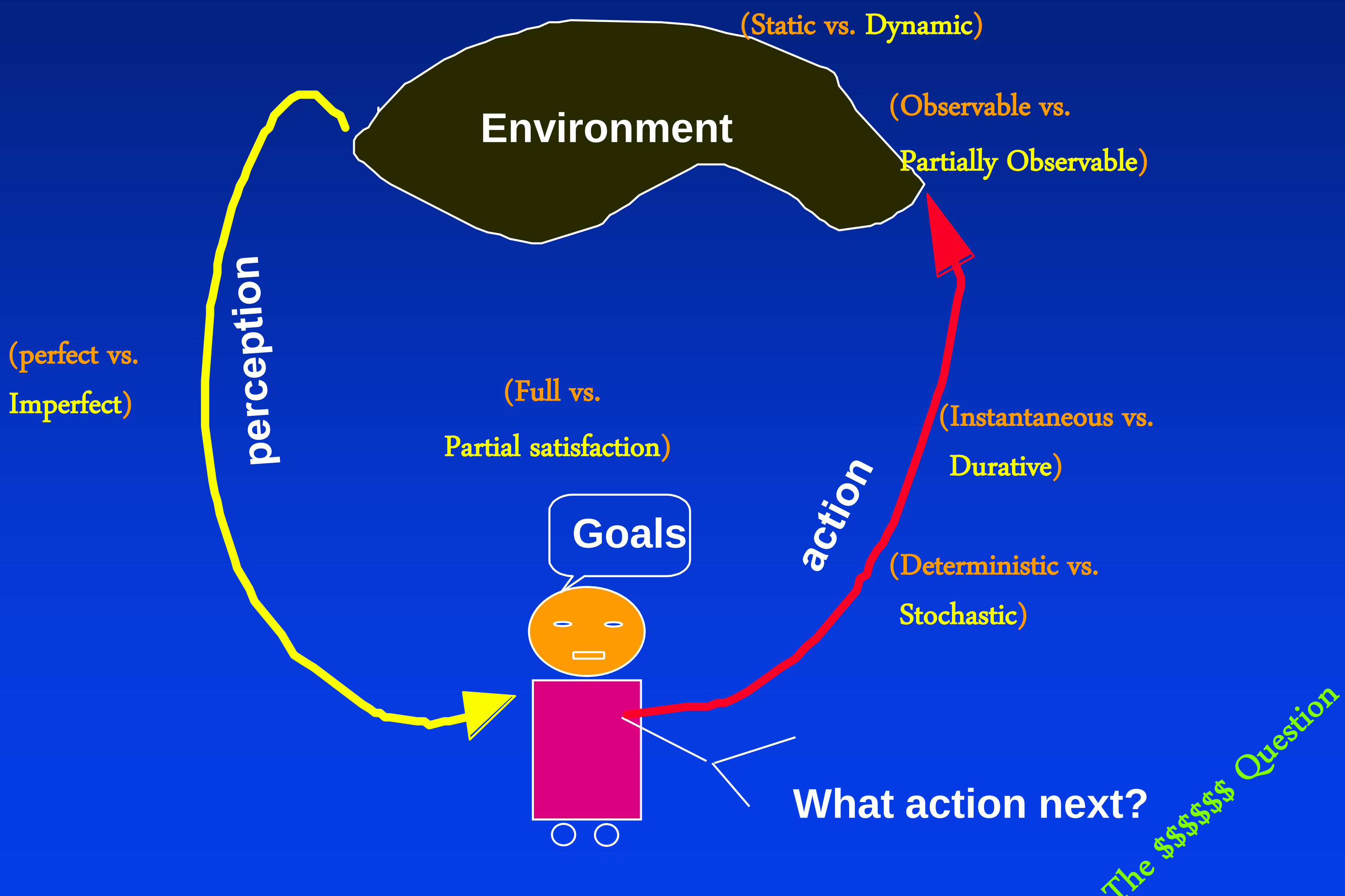
-Randy Pausch

Irritated/Resigned

Mechano-phile? ~~Bio-Inspired?~~

- Don’t like ants, spiders or other bugs…
  - Not sure they can teach me much about high-level planning
- Not crazy about humans either..
- ..am stuck with humans in the loop
  - So, a Buddhist middle way

Planning Involves Deciding a Course of Action to achieve a desired state of affairs



"Classical" Planning

```
(: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)))
```

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

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

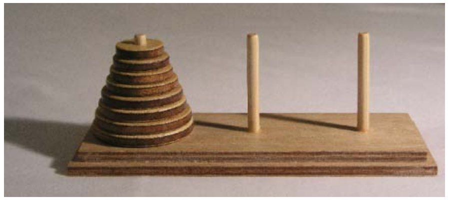
Pickup(A)

holding(A)
~Clear(A)
~Ontable(A)
Ontable(B),
Clear(B)
~handempty

Pickup(B)

holding(B)
~Clear(B)
~Ontable(B)
Ontable(A),
Clear(A)
~handempty

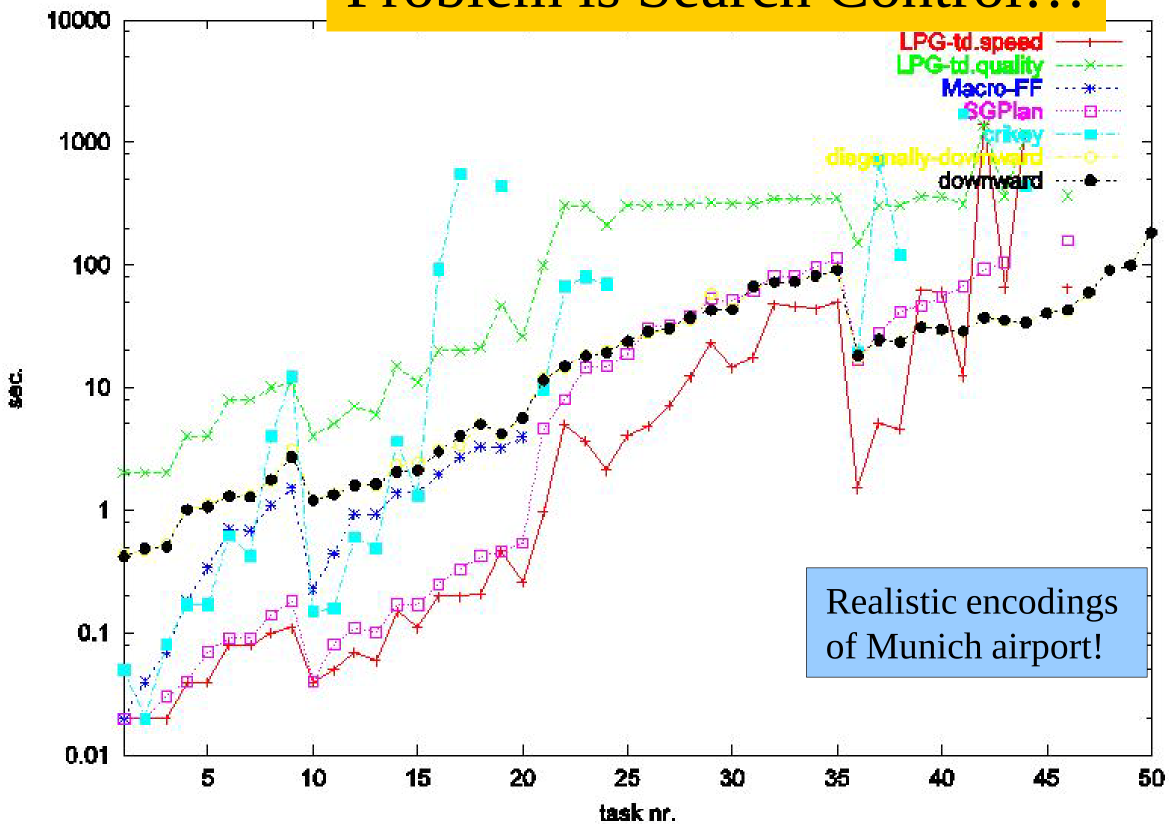
P-Space Complete



Scalability was the big bottle-neck… We have figured out how to scale synthesis..

Problem is Search Control!!!

- Before, planning algorithms could synthesize about 6 – 10 action plans in minutes
- Significant scale-up in the last decade
  - Now, we can synthesize 100 action plans in seconds.

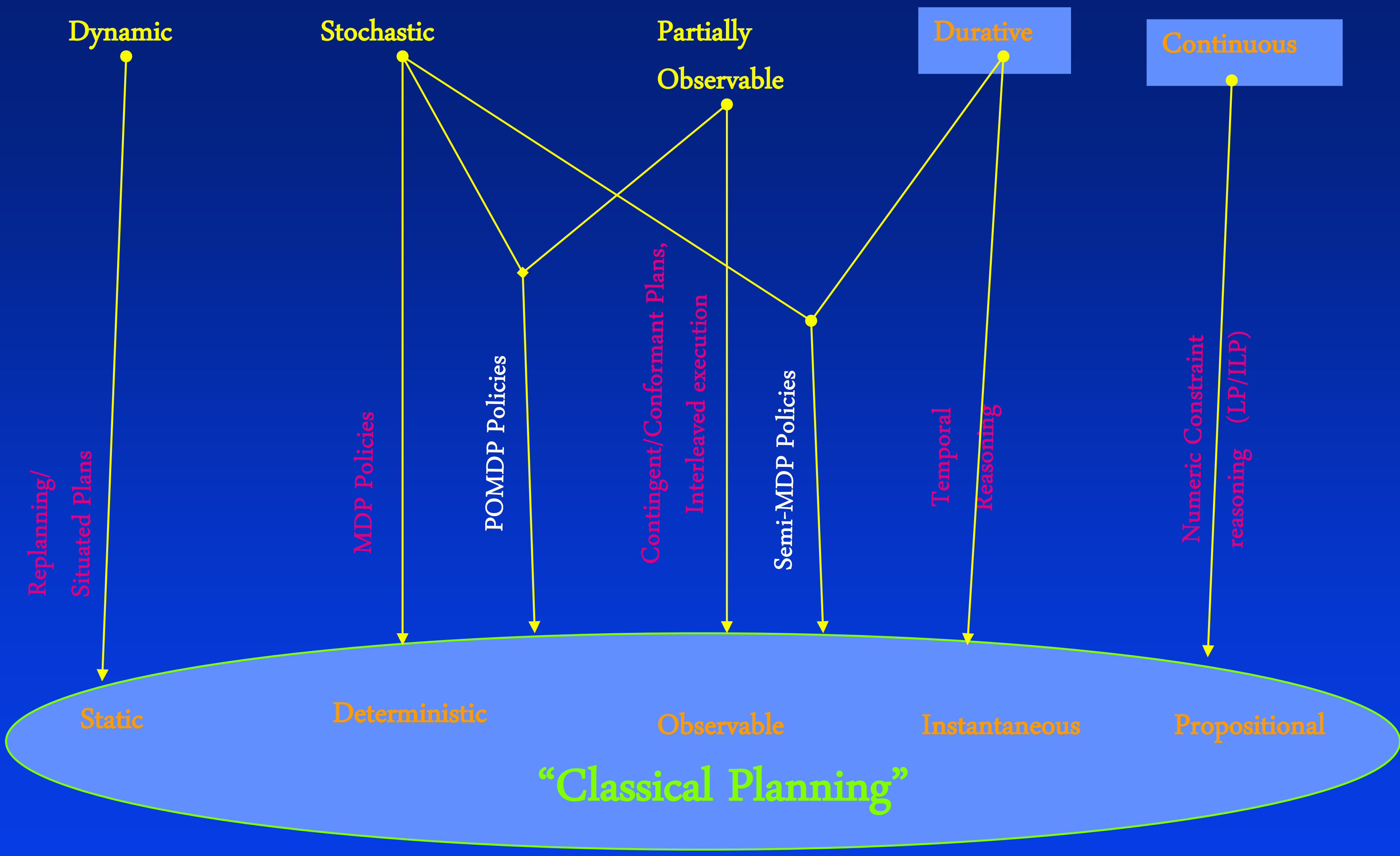


Realistic encodings of Munich airport!

The primary revolution in planning in the recent years has been **methods to** scale up plan synthesis

..and we have done our fair bit...

So, what next?



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"

Assumption: Complete Models

→ ~~Complete~~ Action Descriptions (fallible domain writers)

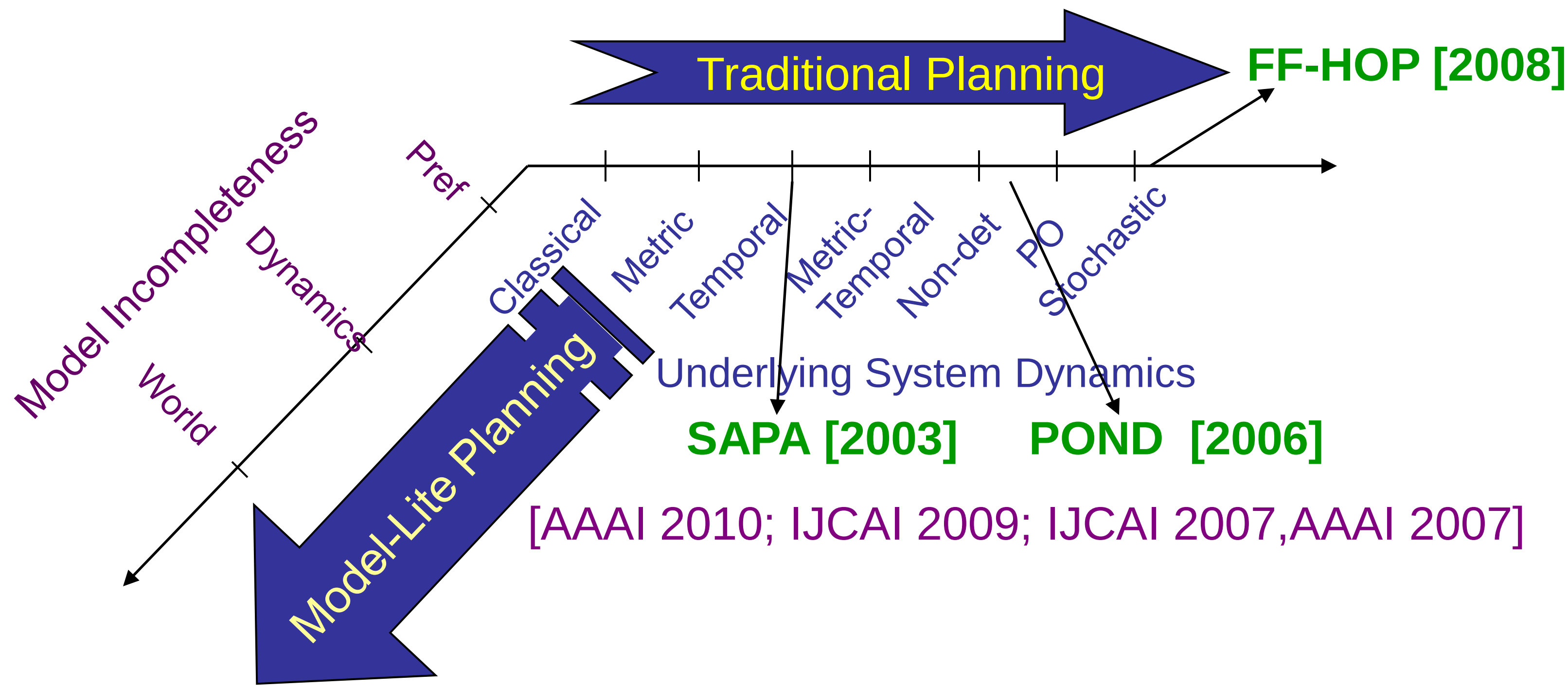
→ ~~Fully Specified~~ Preferences (uncertain users)

→ ~~All objects~~ in the world known up front (open worlds)

→ ~~One-shot~~ planning (continual revision)

Planning is no longer a pure inference problem ☹

☹ But humans in the loop can ruin a really a perfect day ☹



Effective ways to handle the more expressive planning problems by exploiting the deterministic planning technology

Learning is not the (sole) answer..

- A tempting way to handle incompleteness is to say that we should wait until the full model is obtained
  - Either through learning
  - Or by the generosity of the domain writer..
- Problem: Waiting for complete model is often times not a feasible alternative
  - The model may never become complete…
  - We need to figure out a way of maintaining incomplete models, and planning with them (pending learning..)

Challenges of Handling Incompleteness

1. Circumscribing the incompleteness

2. Developing the appropriate solution concepts

3. Developing planners capable of synthesizing them

4. Life-long Planning/Learning to reduce incompleteness
   – Commitment-sensitive Replanning

There are known knowns; there are things we know that we know. There are known unknowns; that is to say, there are things that we now know we don’t know. But there are also unknown unknowns; there are things we do not know we don’t know.

Challenges of Human-in-the-Loop Planning

1. Circumscribing the incompleteness

2. Developing the appropriate solution concepts

3. Developing planners capable of synthesizing them

4. Life-long Planning/Learning to reduce incompleteness
   - Commitment-sensitive Replanning

Tough Problems

Our Contributions

- Preference incompleteness
  - Unknown Preferences [IJCAI 2007]
  - Partially known Preferences [IJCAI 2009]
- Model incompleteness
  - Robust plan generation [ICAPS Wkshp 2010]
- World/Object incompleteness
  - OWQG [IROS 2009; BTAMP 2009; AAAI 2010]

Model-Lite Planning

Preferences in Planning – Traditional View

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

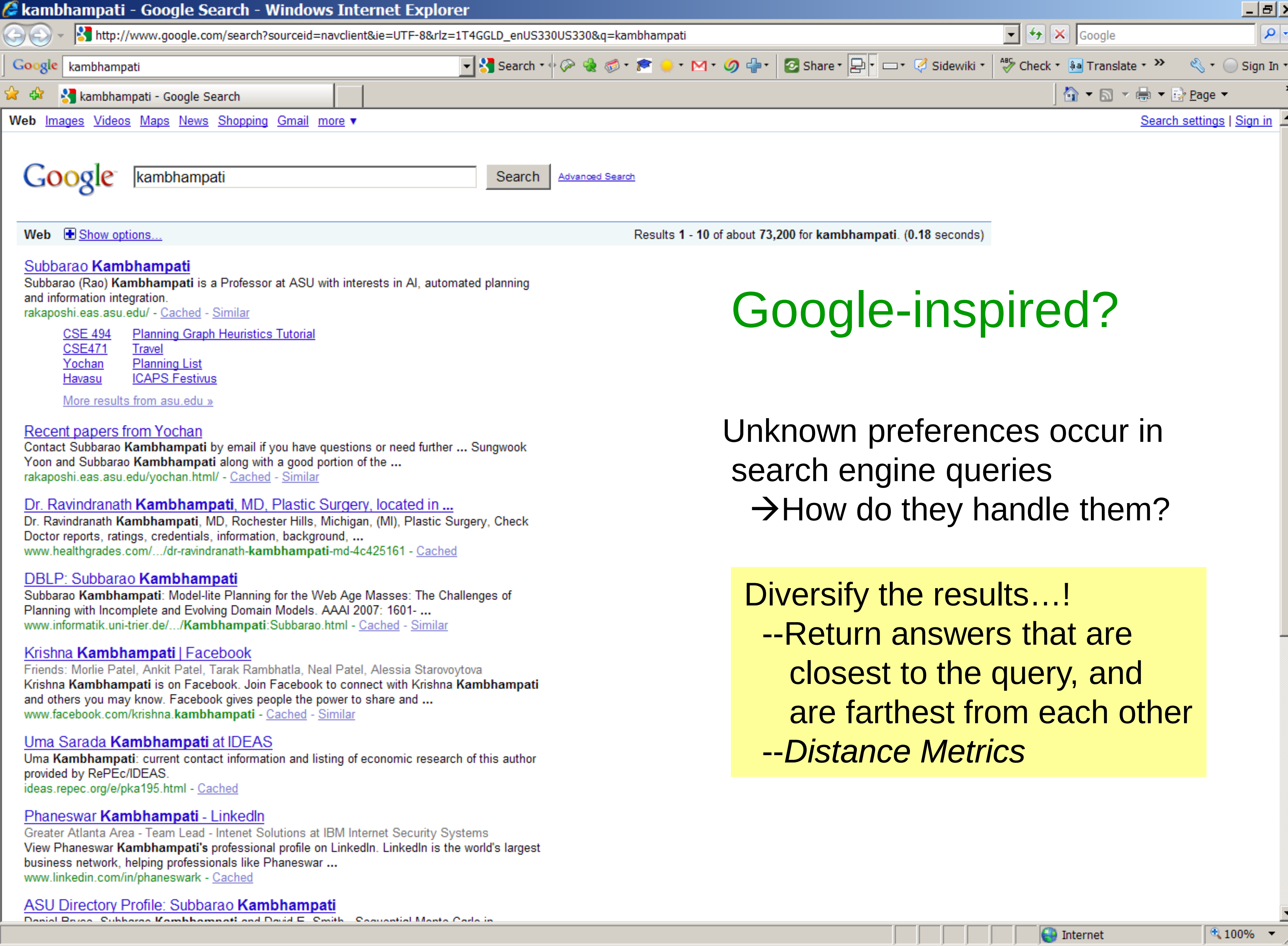
Full Knowledge of Preferences

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

Human in the Loop: Unknown & Partially Known Preferences



Google-inspired?

Unknown preferences occur in search engine queries
→How do they handle them?

Diversify the results…!
--Return answers that are closest to the query, and are farthest from each other
--*Distance Metrics*

Handling Unknown & Partially Known Preferences

- **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…
  - Best choice is to return a *diverse* set of plans [IJCAI 2007]
    - Distance measures between plans

Generating Diverse Plans

- Formalized notions of bases for plan distance measures
- Proposed adaptation to existing representative, state-of-the-art, planning algorithms to search for diverse plans
  - Showed that using action-based distance results in plans that are likely to be also diverse with respect to behavior and causal structure
  - LPG can scale-up well to large problems with the proposed changes

[IJCAI 2007]

- *d*DISTANT*k*SET
  - Given a distance measure δ(.,.), and a parameter k, find k plans for solving the problem that have guaranteed minimum pair-wise distance *d* among them in terms of δ(.,.)

Distance Measures

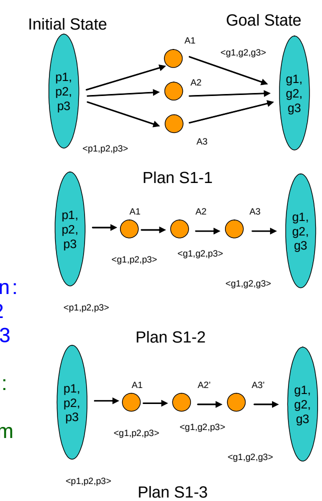
- In what terms should we measure distances between two plans?
  - The actions that are used in the plan?
  - The behaviors exhibited by the plans?
  - The roles played by the actions in the plan?
- Choice may depend on
  - The ultimate use of the plans
    - E.g. Should a plan P and a non-minimal variant of P be considered similar or different?
  - What is the source of plans and how much is accessible?
    - E.g. do we have access to domain theory or just action names?

Compute by Set-difference

•Action-based comparison: S1-1, S1-2 are similar, both dissimilar to S1-3; with another basis for computation, all can be seen as different

•State-based comparison: S1-1 different from S1-2 and S1-3; S1-2 and S1-3 are similar

•Causal-link comparison: S1-1 and S1-2 are similar, both diverse from S1-3



Generating Diverse Plans with Local Search

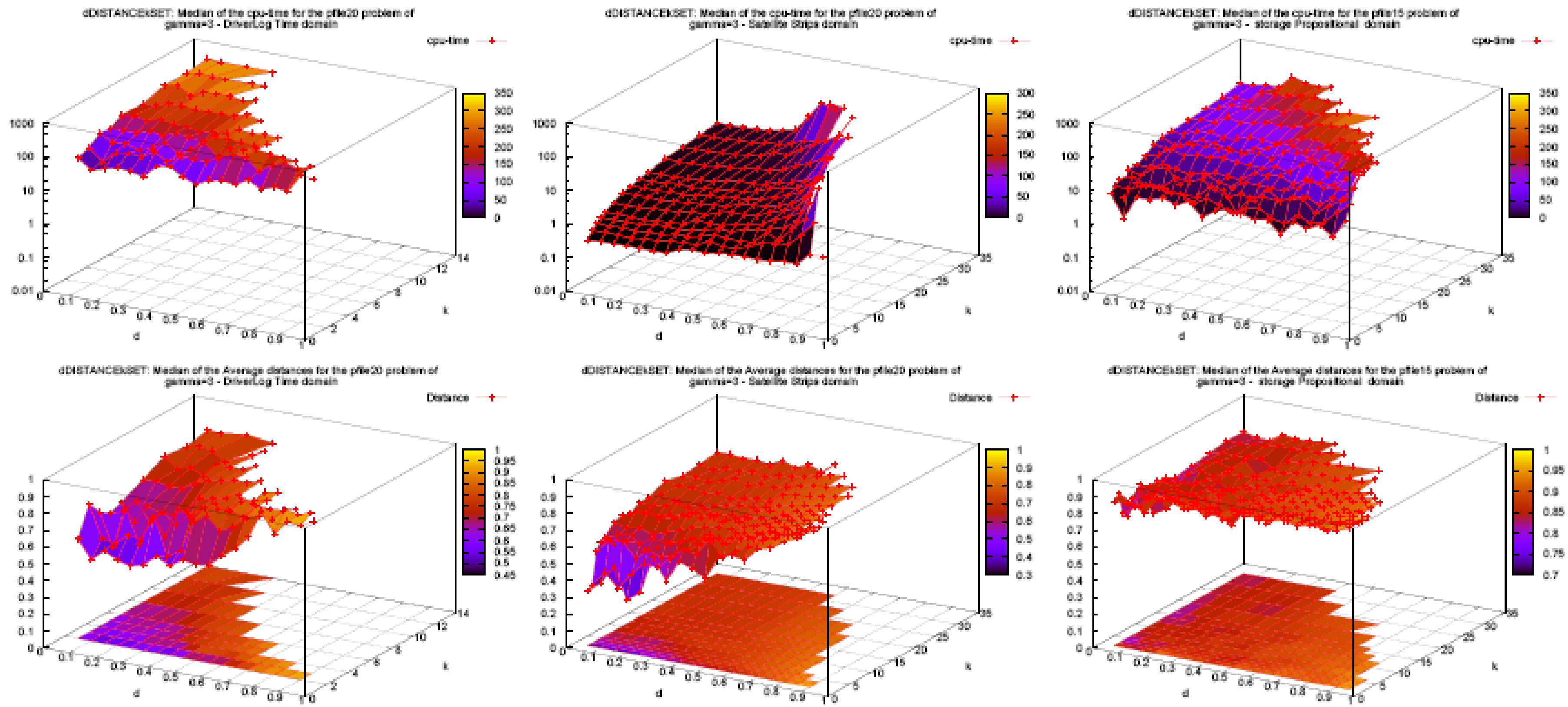


Figure 2: Performance of LPG-d (CPU-time and plan distance) for there problems in DriverLog-Time, Satellite-Strips and Storage-Propositional.

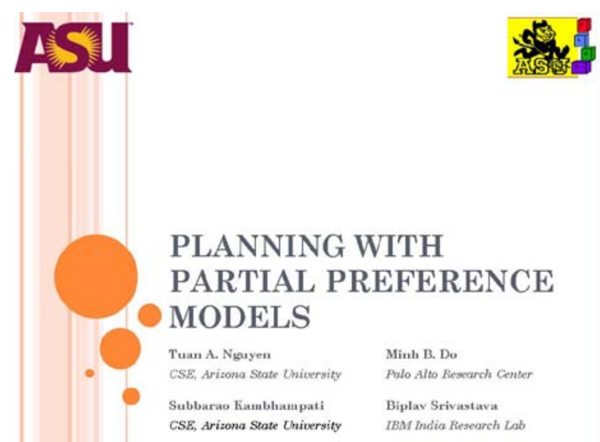
LPG-d solves 109 comb.
Avg. time = 162.8 sec
Avg. distance = 0.68
Includes d<0.4,k=10; d=0.95,k=2

LPG-d solves 211 comb.
Avg. time = 12.1 sec
Avg. distance = 0.69

LPG-d solves 225 comb.
Avg. time = 64.1 sec
Avg. distance = 0.88

Unknown & Partially Known Preferences

- **Partially known**
  - We may know that user cares only about makespan and cost. But we don't know how she combines them..
- Returning a diverse set of plans may not be enough
  - *They may not differ on the attributes of relevance..*
- Focus on spanning the pareto set..



Modeling Partially Known Objectives

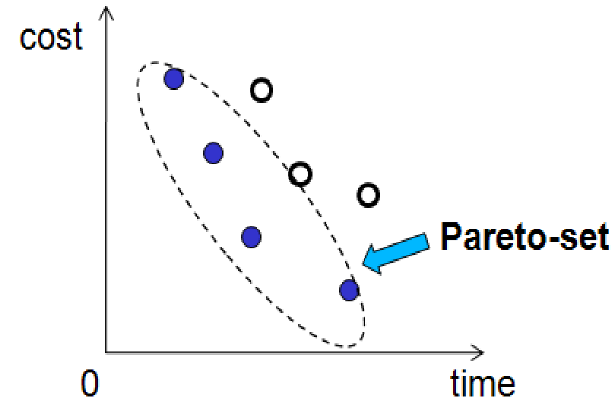
- The user is interested in minimizing two objectives (say makespan and execution cost of plan *p*: *time*(*p*), *cost*(*p*).)
- The quality of plan *p* is given by *cost function*:

$$f(p, w) = w \times time(p) + (1 - w) \times cost(p) \quad (w \in [0,1])$$

  - $w \in [0,1]$ represents the trade-off between two competing objectives.

measure developed in OR community (Carlyle, 2005).

Handling Partially Known Preferences

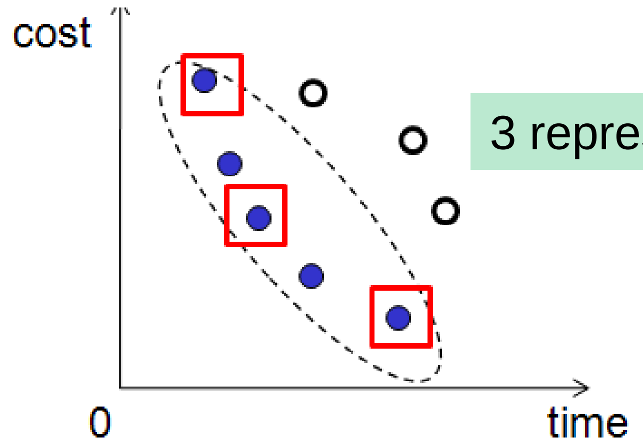


- **View it as a Multi-objective optimization**
  - Return the Pareto optimal set of plans (and let the user select from among them)
- Two problems
  - [Computational] Computing the full pareto set can be too costly
  - [Comprehensional] Lay users may suffer information overload when presented with a large set of plans to choose from
- Solution: Return *k* representative plans from the Pareto Set
  - Challenge 1: How to define "representative" robustly?
  - Challenge 2: How to generate representative set of plans efficiently?

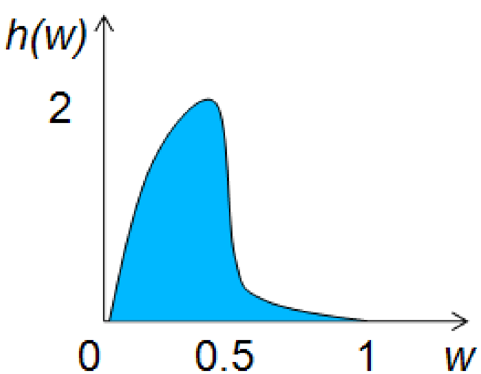
Measuring Representativeness: ICP

$$f(p,w) = w \times time(p) + (1-w) \times cost(p) \quad (w \in [0,1])$$

$$ICP(\mathcal{P}) = \sum_{i=1}^{k} \int_{w_{i-1}}^{w_i} h(w)(w \times t_{p_i} + (1-w) \times c_{p_i}) dw$$



3 representative plans



(Fast plan seems better)

Handling Partial Preferences using ICP

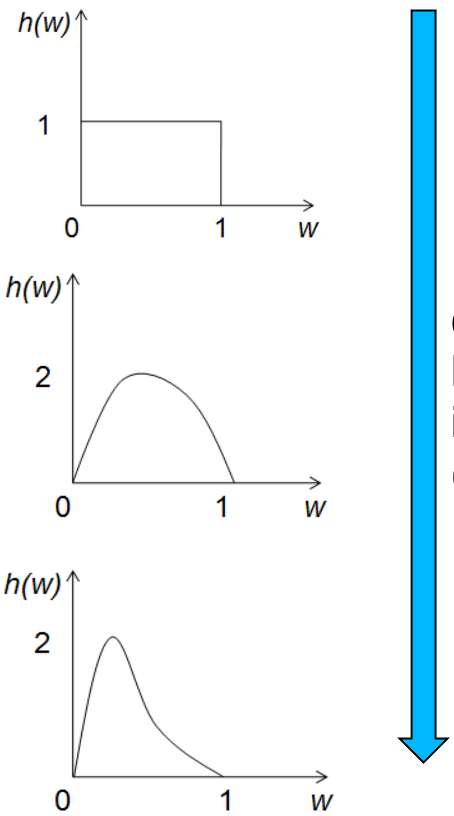
Problem Statement:

- Given
  - the objectives O_i,
  - the vector w for convex combination of O_i
  - the distribution $h(w)$ of w,
- Return a set of k plans with the minimum ICP value.

- Solution Approaches:
  - Sampling: Sample k values of w, and approximate the optimal plan for each value.
  - ICP-Sequential: Drive the search to find plans that will improve ICP
  - Hybrid: Start with Sampling, and then improve the seed set with ICP-Sequential
  - [Baseline]: Find k diverse plans using the distance measures from [IJCAI 2007] paper; LPG-Speed.

Learning Planning Preferences

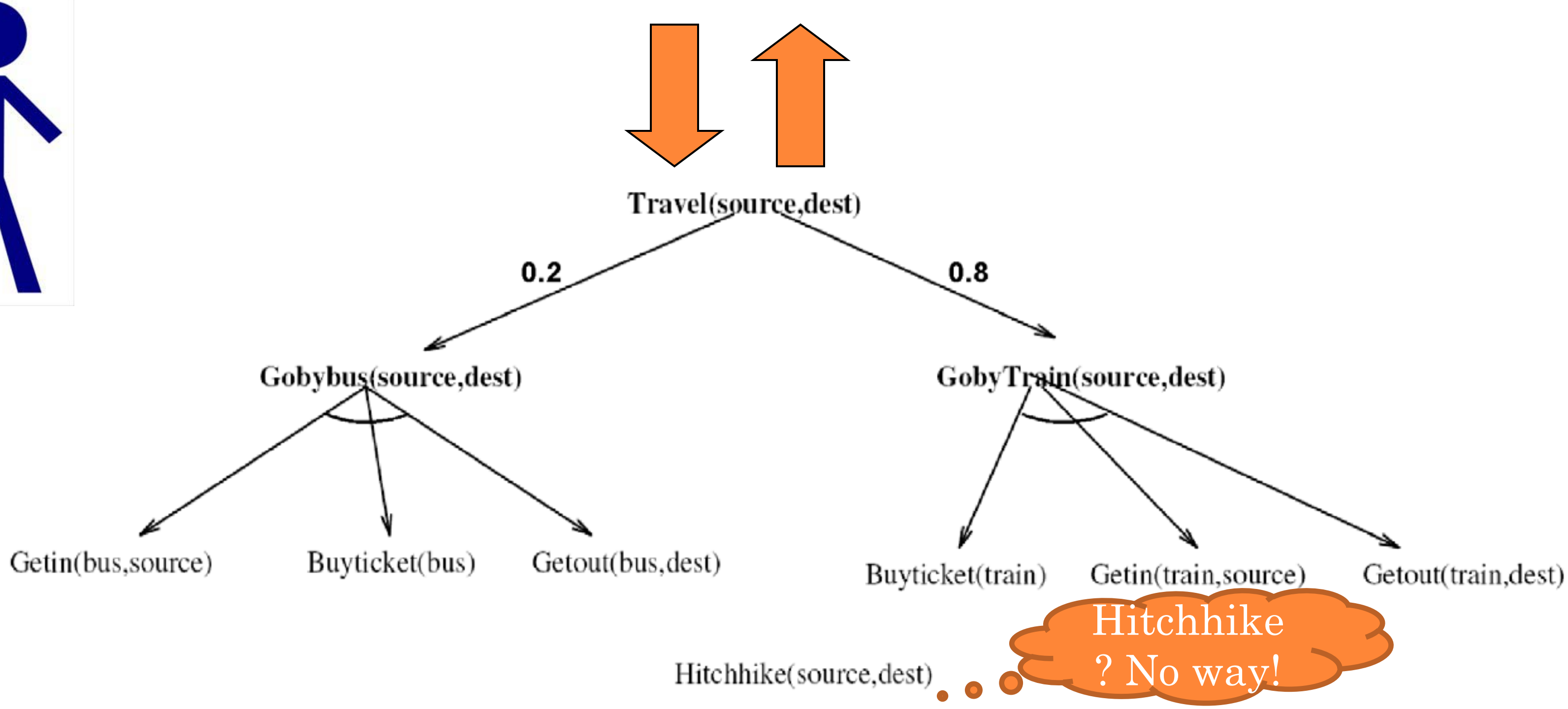
- We can learn to improve the preference model by revising the *h(w)* after every few iterations (through user interaction)



Revising distribution h(w) over iterations (Bayesian learning..)

LEARNING PLAN PREFERENCES
From Observed Executions

- P_{bus}: Getin(bus, source), Buyticket(bus), Getout(bus, dest) 2
- P_{train}: Buyticket(train), Getin(train, source), Getout(train, dest) 8
- P_{hike}: Hitchhike(source, dest) 0

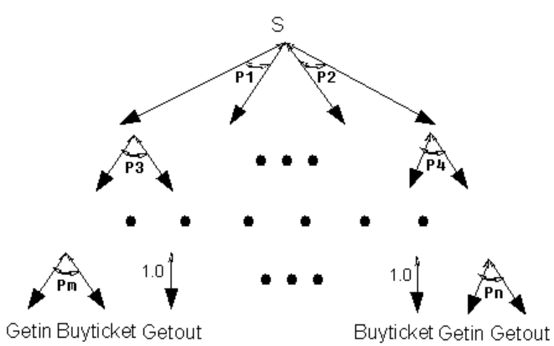


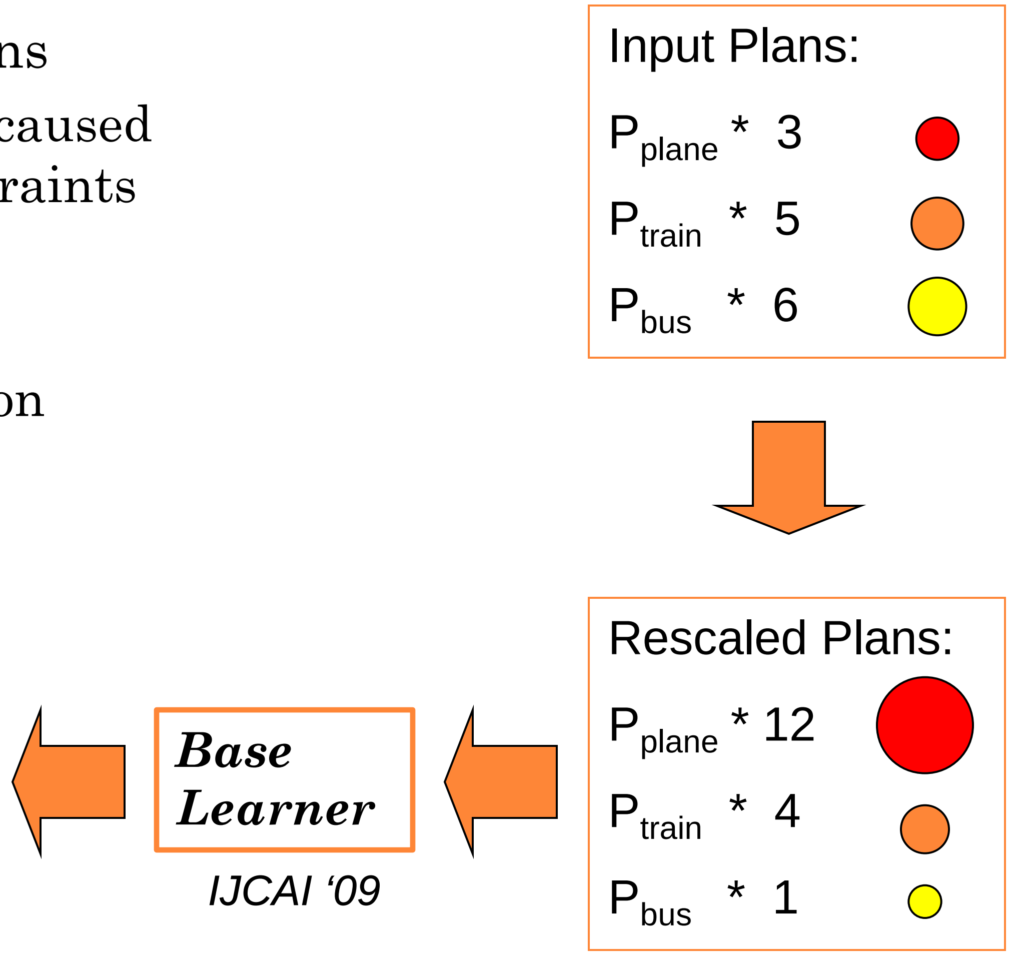
[IJCAI 2009]

LEARNING USER PLAN PREFERENCES OBFUSCATED BY FEASIBILITY CONSTRAINTS

- Rescale observed plans
  - Undo the filtering caused by feasibility constraints
- Base learner
  - Acquires true user preferences based on adjusted plan frequencies

User Preference Model





[ICAPS 2009]

Our Contributions

Preference incompleteness

- Unknown Preferences [IJCAI 2007]
- Partially known Preferences [IJCAI 2009]

Model incompleteness

- Robust plan generation [ICAPS Wkshp 2010]

World/Object incompleteness

- OWQG [IROS 2009; BTAMP 2009; AAAI 2010]

There are known knowns; there are things we know that we know. There are known unknowns; that is to say, there are things that we now know we don't know. But there are also unknown unknowns; there are things we do not know we don't know.

Planning with partial domain models: Motivation

- Planning, in traditional perspective, assumes a completely specified domain model
  - We know exactly the conditions and effects of action execution
  - Stochastic models also assume completeness ("known" probabilities)

```
(: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)))
```

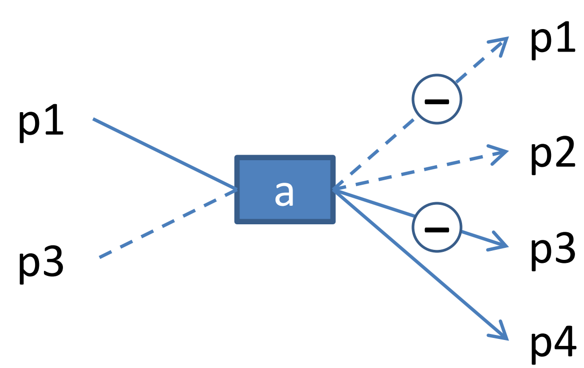
- Domain modeling is a laborious, error-prone task
  - So much so that there is a Knowledge Engineering track for ICP
- Action descriptions have to be seen as "nominal"
  - May have missing preconditions and effects…
- Sometimes, the domain modeler may be able to annotate the action with sources of incompleteness
  - Possible preconditions/effects

Can the planner exploit such partial knowledge?

Deterministic Partial Domain Models

- We consider planning with deterministic, but incompletely specified domain model
- Each action **a** is associated with *possible* precond and effects (in addition to the normal precond/eff):
  - **PreP(a)** ***[p]:*** set of propositions that **a** *might* depend on during execution
  - **AddP(a)** ***[p]:*** : set of propositions that **a** *might* add after execution
  - **DelP(a)** ***[p]:*** : set of propositions that **a** *might* delete after execution

<u>Example</u>: An action **a** that is known to depend on **p1**, add p4 and delete **p3**. In addition, it might have **p3** as its precondition, might add **p2** and might delete **p1** after execution.



Solution Concept: Robust Plans

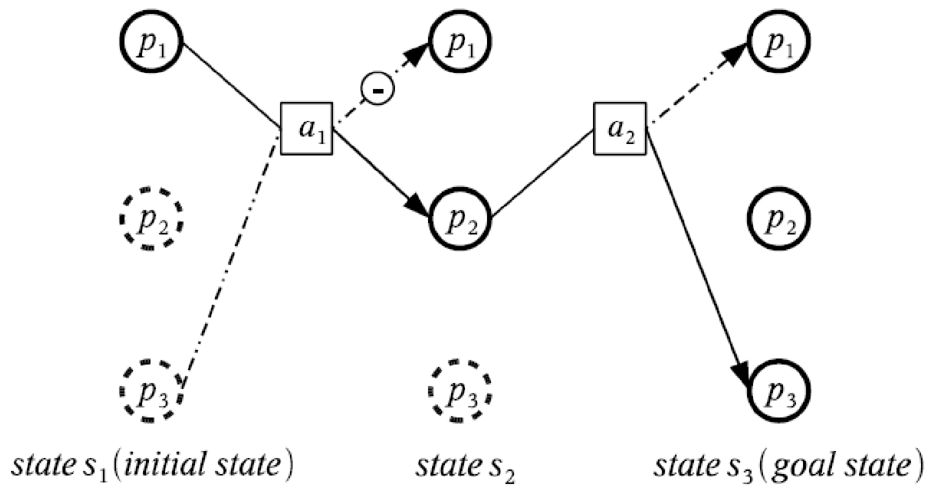
- Solution concept:
  - Robust plan
  - Plan is highly robust if executable in large number of most-likely candidate models
- Robustness measure
  - Set of candidate domain models **S** (consistent with the given deterministic partial domain model **D**)
  - A complete but unknown domain model **D***
    - Can be any model in **S**

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

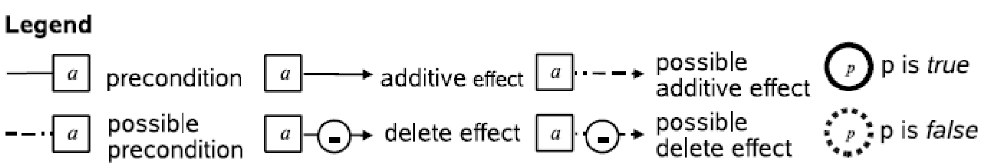
$|\Pi|$ Number of candidate models with which the plan succeeds

$$K = \sum_{a} \mathrm{PreP}(a) + \mathrm{AddP}(a) + \mathrm{DelP}(a)$$

Easily generalized to consider model likelihood



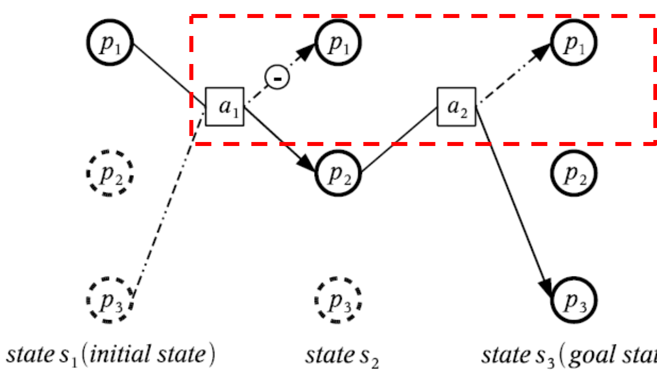
Candidate models of plan	1	2	3	4	5	6	7	8
a_1 relies on p_3	yes	yes	yes	yes	no	no	no	no
a_1 deletes p_1	yes	yes	no	no	yes	yes	no	no
a_2 adds p_2	yes	no	yes	no	yes	no	yes	no
Plan status	fail	fail	fail	fail	succeed	fail	succeed	succeed



Robustness value: 3/8

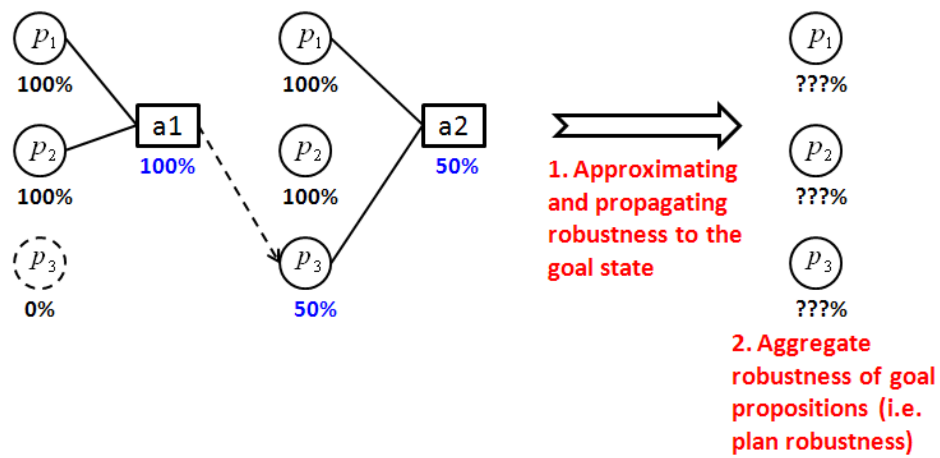
Assessing Plan Robustness

- Number of candidate models: exponentially large. Computing robustness of a given plan is hard!!!
  - Exact and approximate assessment.
- **Exact methods:**
  - (Weighted) Model-counting approach:
    - Construct logical formulas representing *causal-proof* (Mali & Kambhampati 1999) for plan correctness
    - Invoke an exact model counting approach



"If **p1** is realized as a delete effect of **a1**, then it must be an additive effect of **a2**."

- **Approximate methods:**
  - Invoke *approximate* model counting approach
  - Approximate and propagate action robustness
    - Can be used in generating robust plans



Generating Robust Plans

- Compilation approach: Compile into a *(Probabilistic) Conformant Planning* problem
  - One "unobservable" variable per each possible effect/precondition
    - Significant initial state uncertainty
  - Can adapt a probabilistic conformant planner such as POND [JAIR, 2006; AIJ 2008]
- Direct approach: Bias a planner's search towards more robust plans
  - Heuristically assess the robustness of partial plans
    - Need to use the (approximate) robustness assessment procedures

D. Bryce et al. / Artificial Intelligence 172 (2008) 685–715

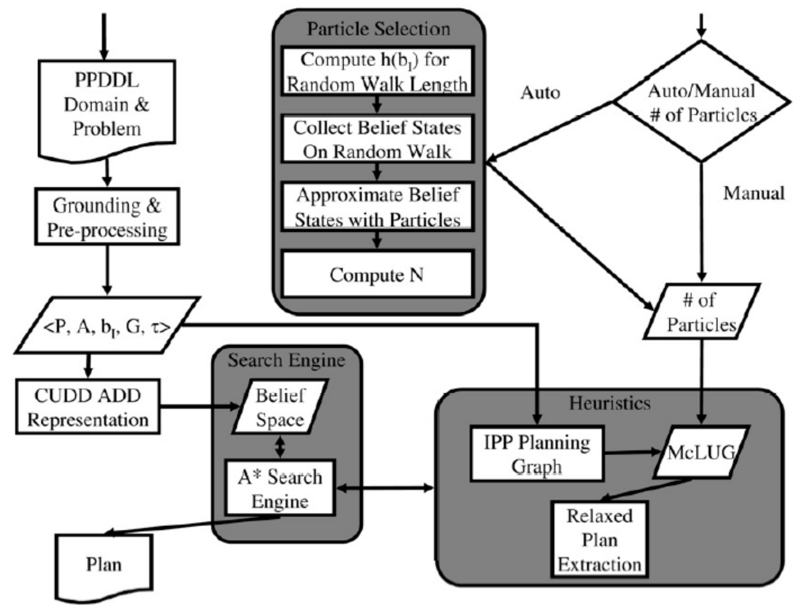
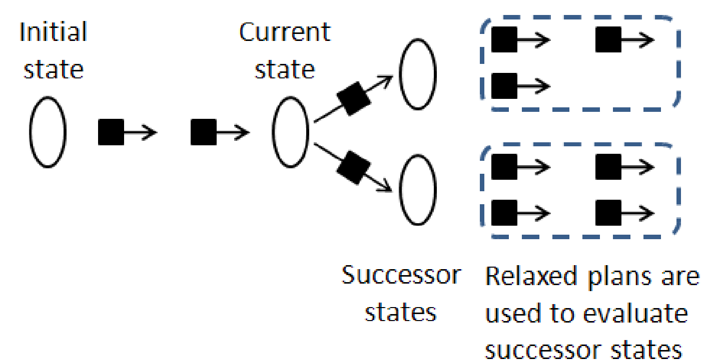


Fig. 6. *POND* architecture.

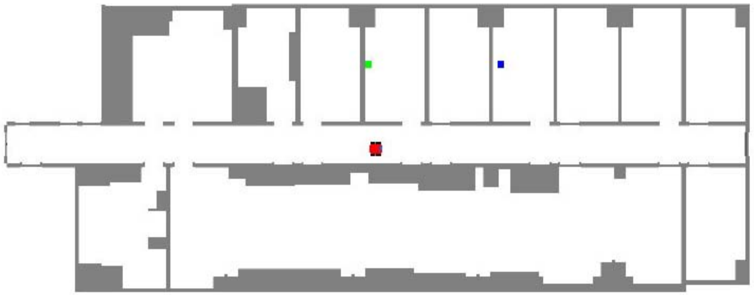


[ICAPS 2010 Wkshp on Planning under Uncertainty]

Our Contributions

- Preference incompleteness
  - Unknown Preferences [IJCAI 2007]
  - Partially known Preferences [IJCAI 2009]
- Model incompleteness
  - Robust plan generation [ICAPS Wkshp 2010]
- World/Object incompleteness
  - OWQG [IROS 2009; BTAMP 2009; AAAI 2010]

Urban Search and Rescue



- Human-Robot team
- Robot starts the beginning of the hallway
- Human is giving higher level knowledge
- Hard Goal: Reach the end of the hallway
- Wounded people are in rooms
- Soft Goal: Report locations of wounded people

Planning Support for USAR

- Good News: Some aspects of existing planning technology are very relevant
    - Partial Satisfaction
    - Replanning & Execution Monitoring
- Bad News: Incomplete Model / Open World
    - Unknown objects
        - Don't know where injured people are
    - Goals specified in terms of them
        - If the robot finds an injured person, it should report their location …

How do you make a deterministic closed-world planner believe in opportunities sans guarantees?

Open World Quantified Goals

Partial Satisfaction Planning (PSP)

Sensing and Replanning

Planner

CLOSED WORLD

Robot

OPEN WORLD

Under Sensing
Closed World Model

Limited Sensing
Planner guides robot in a limited way

Over Sensing
Robot senses its way through the world

Handling Open World

- Extreme Cases
  - If the robot assumes "closed world", it will just go to the end of the corridor.
  - If the robot insists on "closing" the model before doing planning, it will do over-sensing.
- Need a way of combining sensing and planning
  - Information on unknown objects
  - Goals conditioned on these objects

Open World Quantified Goals (OWQGs)

- Goals that allow for the specification of additional information
  - To take advantage of opportunities

```
(:open  (forall ?r – room            Quantified Object(s)
   (sense ?p – person                Sensed Object
      (looked_for ?p ?r)             Closure Condition
    (and (has_property ?p wounded)
         (in ?p ?r))                 } Quantified Facts
(:goal
   (and (reported ?p wounded ?r)
   [100] - soft))))                  } Quantified Goal
```

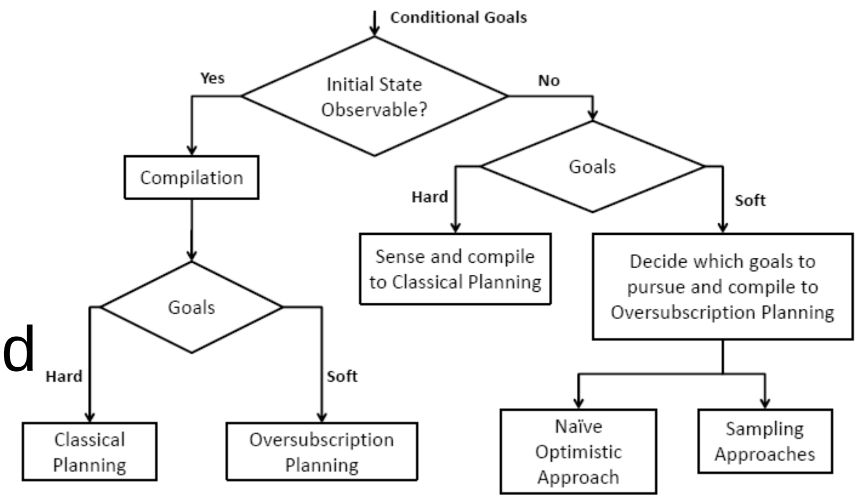
OWQGs as Conditional Rewards

Robot needs to
  sense wounded people
  before reporting them

Planner has to deal with open world

Naïve idea: Ask Robot to look
  everywhere (high sensing cost)

--Need to sense for those conditional goals
  whose antecedents are likely to hold



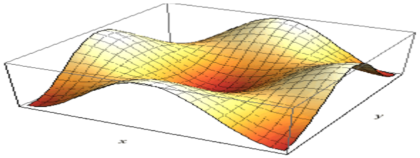
Conditional Goals can be compiled
  down when the world model is complete

$$\hat{\mathcal{G}}_c = \arg\max_{\hat{\mathcal{G}}^i_c \subseteq \mathcal{G}_c} \mathbf{E}_{\mathbf{P}\sim\Psi} \mathcal{B}(G_o \cup [\mathcal{G}^i_c \setminus \mathbf{P}]) - \mathcal{S}(\mathcal{G}^i_c)$$

[ACM TIST 2010; AAAI, 2010; IROS 2009; BTAMP 2009]

Planning with OWQGs

- Bias the planner's model
- Endow the planner with an **optimistic view**
  – Assume existence of objects and facts that may lead to rewarding goals
    - e.g. the presence of an injured human in a room
  – Create **runtime objects**
  – Add to the planner's database of ground objects
- Plans are generated over this reconfigured **potential** search space



Partial Satisfaction Planning (PSP)

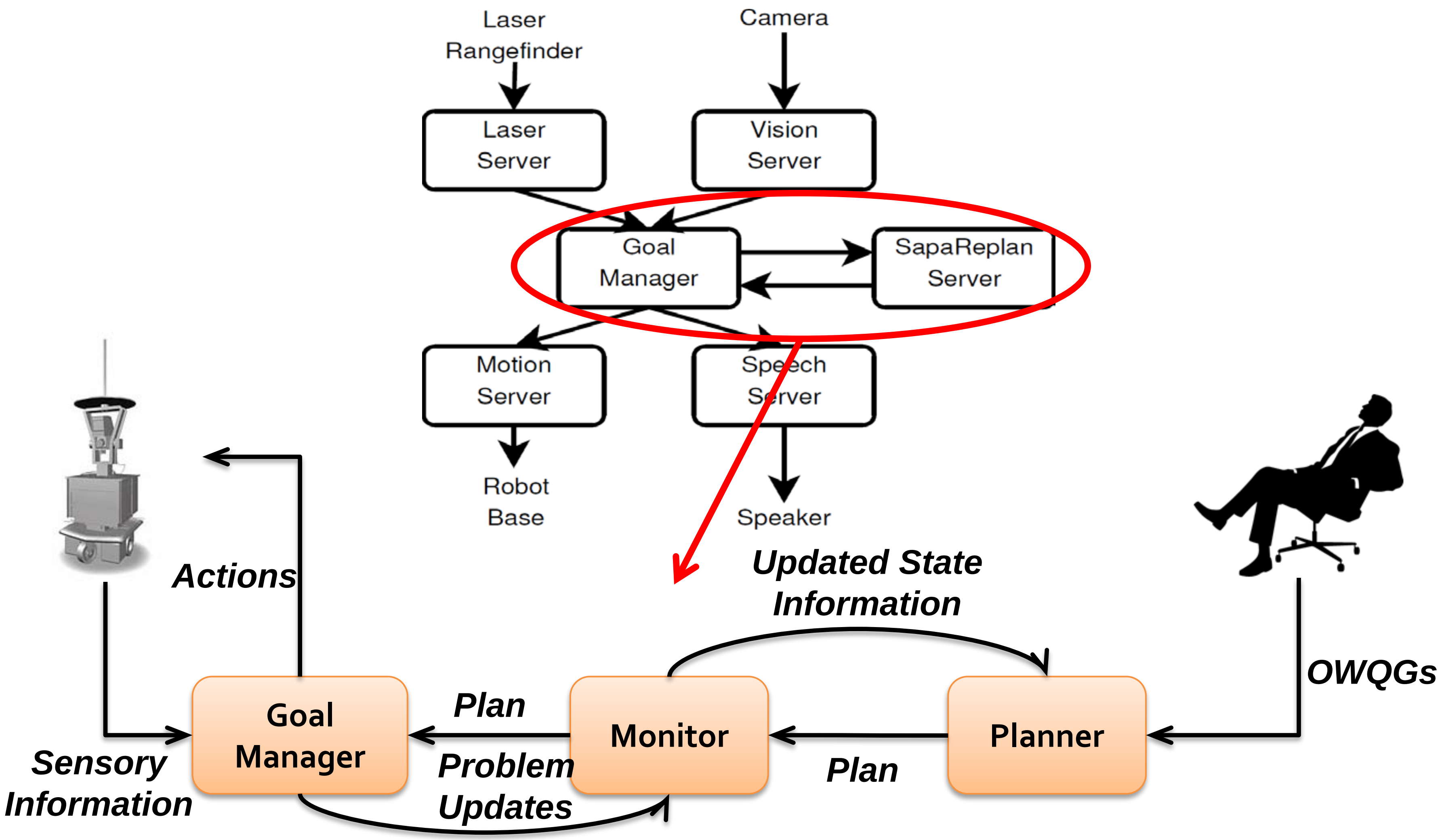
- Soft Goals
  - Allows planner to model "bonus" goals
- Quantification of Goals
  - Cannot possibly satisfy **all** possible groundings
  - Constrained by metric resources (time etc.)
- Net Benefit
  - Sensing is costly
  - Must be balanced with goal-achievement reward

Preferences and PSP in Planning
Benton, Baier, Kambhampati (Tutorial)

Replanning and Execution Monitoring

- Sensing is expensive …
  - Cannot be done at every step
- Planner needs to direct the architecture on:
  - when to sense
  - what to sense for
- Planning to sense in a goal-directed manner
  - Output all actions up to (and including) any action that results in "closing" the world
    - Obtaining information about unknown objects

Putting It All Together



Challenges of Human-in-the-Loop Planning

1. Circumscribing the incompleteness

2. Developing the appropriate solution concepts

3. Developing planners capable of synthesizing them

4. Life Long Planing/Learning to reduce incompleteness

Partial Solutions for Human-in-the-Loop Planning

Can exploit Deterministic Planning technology!

1. Circumscribing the incompleteness
   - Preference components; possible precond/effect annotations; OWQG
2. Developing the appropriate solution concepts
   - Diverse plans; Robust plans; Partial sensing plans
3. Developing planners capable of synthesizing them
   - Can adapt existing planners toward these solution concepts
4. Life Long Planning/Learning to reduce incompleteness
   - Learning preferences *h(.)* through interactions; learning model conditions through execution
     - [Tutorial on Learning in Planning AI MAG 2003; Learning preferences as HTNs IJCAI 2009; ICAPS 2009]

Model-Lite Planning:
Planning is more than pure inference over completely specified models!