Preferences and Partial Satisfaction in Planning

J. Benton, Jorge Baier, Subbarao Kambhampati

Domain-Independent 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)

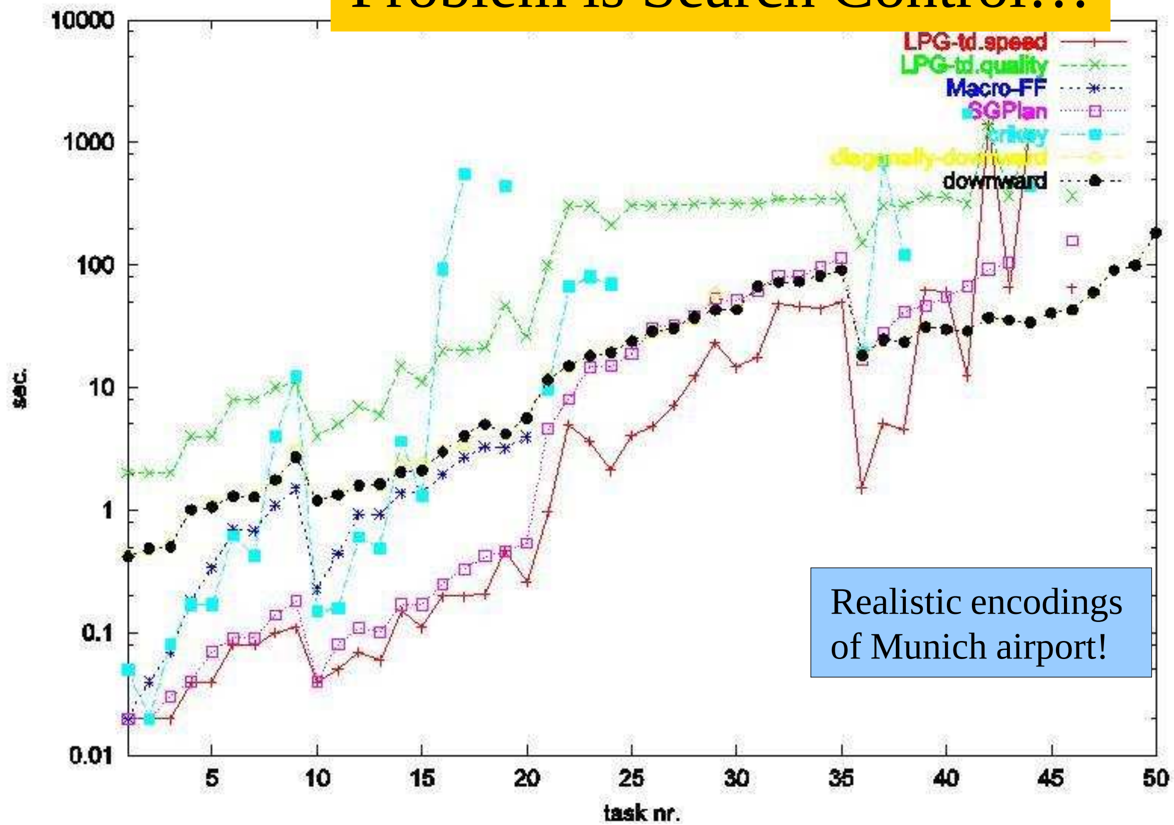
holding(B)
~Clear(B)
~Ontable(B)
Ontable(A),
Clear(A)
~handempty

P-Space Complete

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

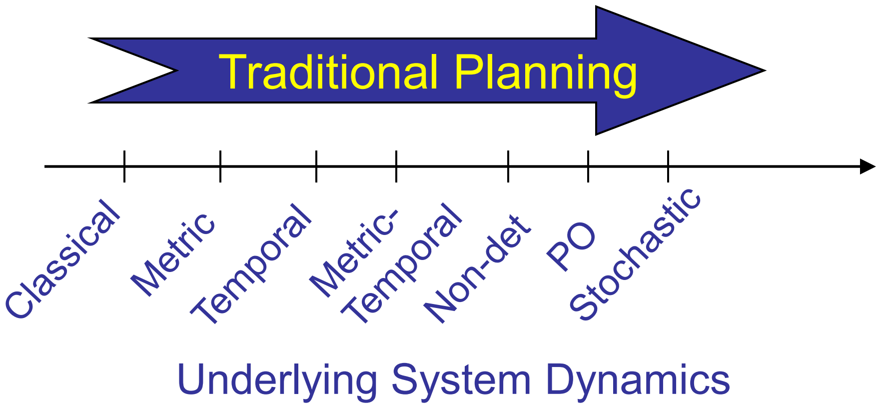
Problem is Search Control!!!

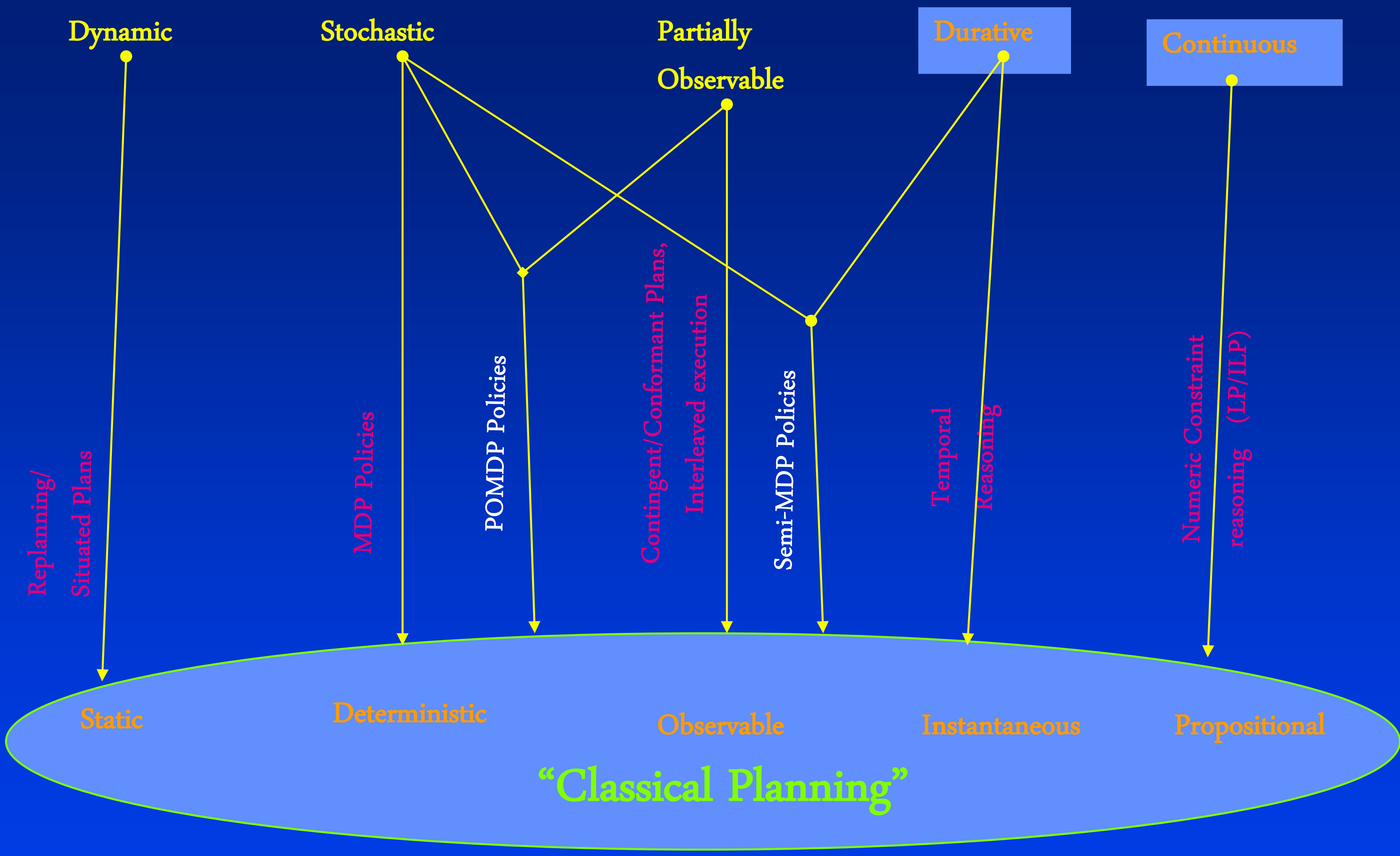
- Before, planning algorithms could synthesize about 6 – 10 action plans in minutes
- Significant scale-up in the last 6-7 years
    - Now, we can synthesize 100 action plans in seconds.



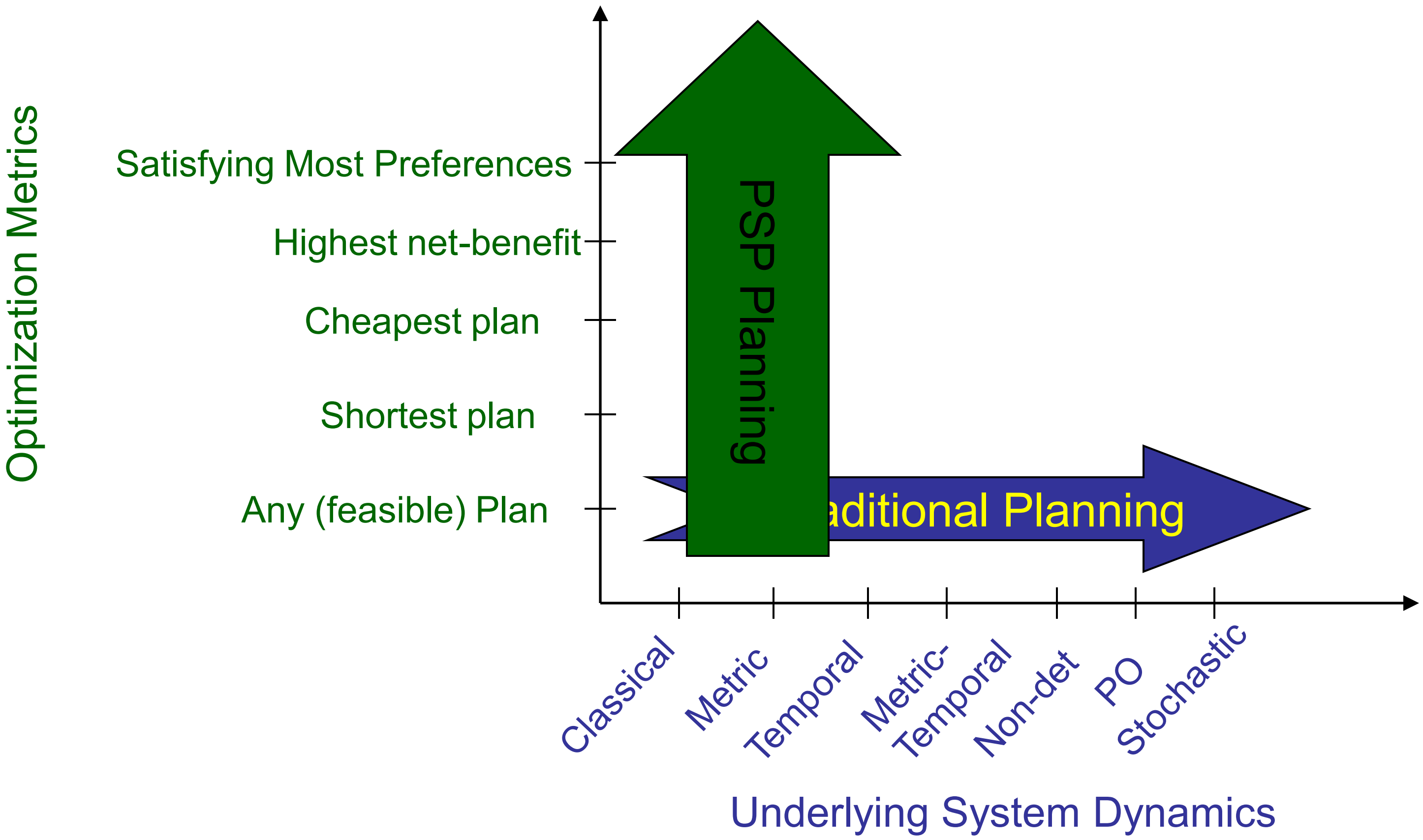
The primary revolution in planning in the recent years has been **methods to** scale up plan synthesis

What should we be doing next?





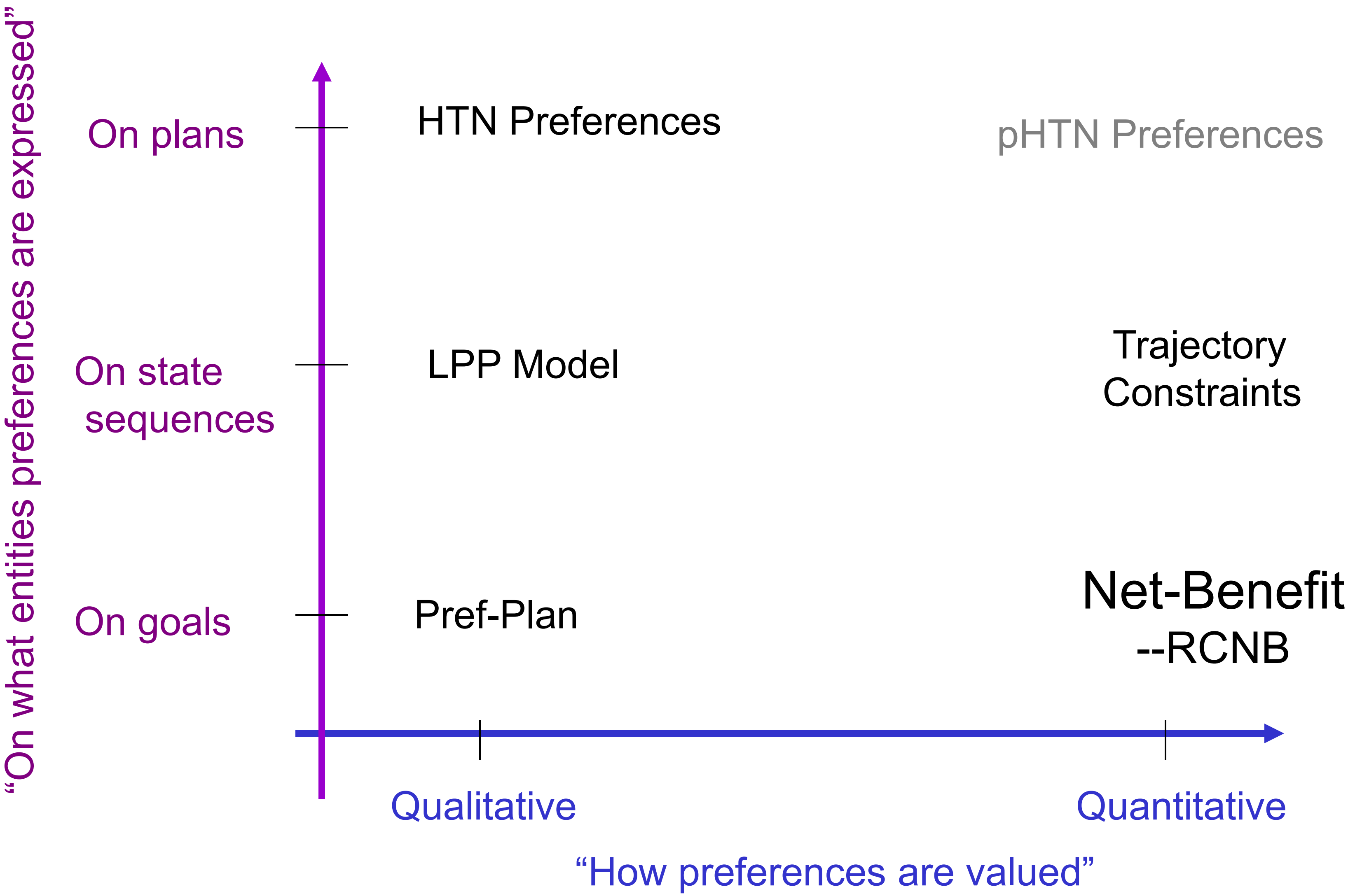
Dynamic
Stochastic
Partially Observable
Durative
Continuous
Replanning/ Situated Plans
MDP Policies
POMDP Policies
Contingent/Conformant Plans, Interleaved execution
Semi-MDP Policies
Temporal Reasoning
Numeric Constraint reasoning (LP/ILP)
Static
Deterministic
Observable
Instantaneous
Propositional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


Optimization Metrics
Satisfying Most Preferences
Highest net-benefit
Cheapest plan
Shortest plan
Any (feasible) Plan
PSP Planning
aditional Planning
Classical
Metric
Temporal
Metric-
Temporal
Non-det
PO
Stochastic
Underlying System Dynamics

Example Applications

- Mars rover, maximizing scientific return with limited resources (Smith, 2004)
- UAVs attempting to maximize reconnaissance returns given fuel constraints
- Logistics problems with time and resource constraints
- Search and rescue scenarios with human-robot-planner communications and replanning (Talamadupula et al., 2010)
- Manufacturing with multiple job requests and deadlines (Ruml et al., 2005)
- Many benchmarks in ICP were originally meant to be PSP (e.g. Satellite domain)

Dimensions of Variation



Challenges

- Representation
  - Languages for expressing preferences
    - That account for preference interactions
  - Compilability
    - Is it possible to compile preferences of one type into another?

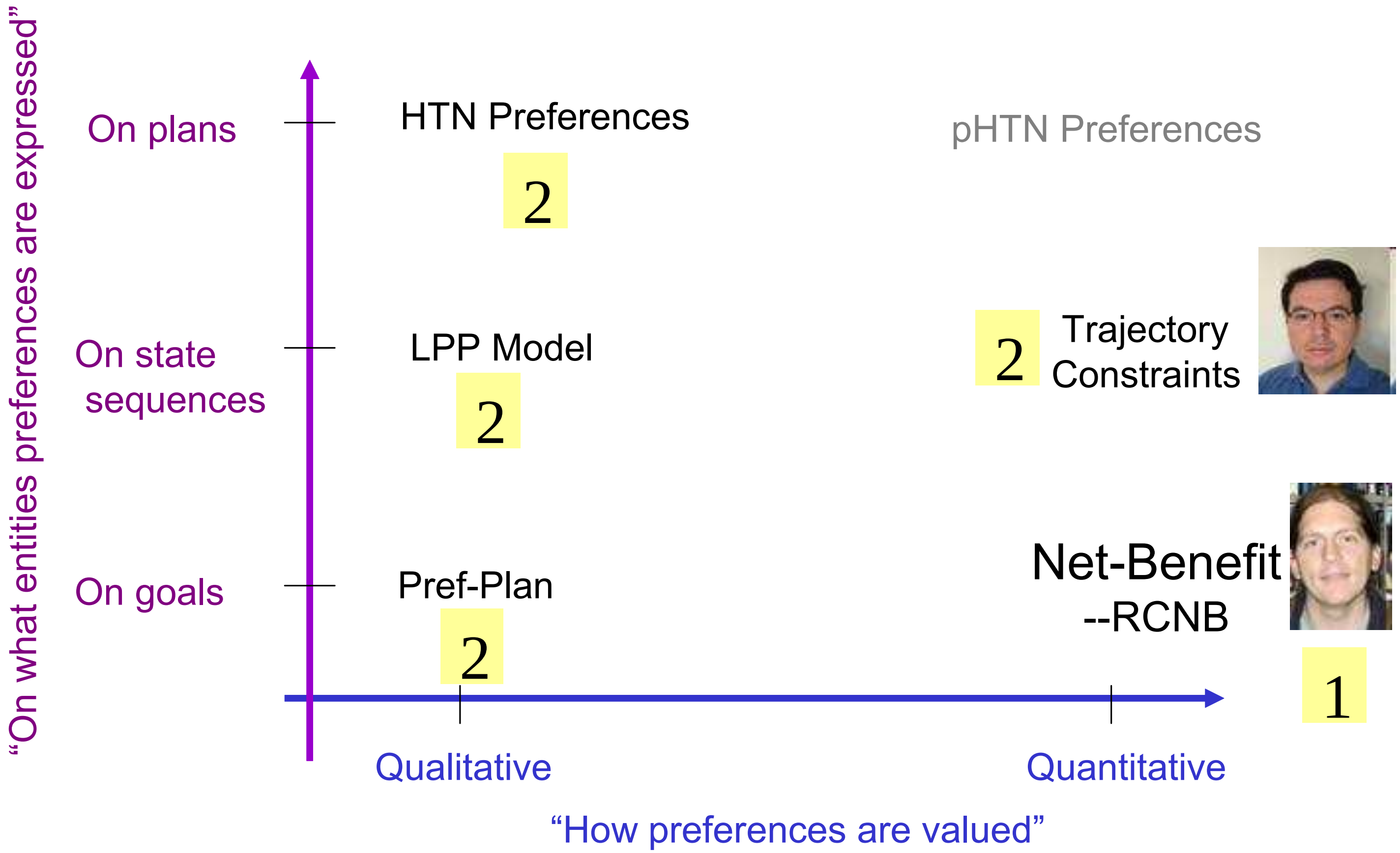
- Synthesis
  - Evaluating plan quality
  - Synthesizing plans with high quality
    - Optimal plans/Pareto Optimal plans
  - Explaining planner decisions

- Acquisition
  - Handling incompletely specified preferences
    - Preference uncertainty
  - Learning preferences

Tutorial Outline

- Planning for net benefit
- Break
- Trajectory Constraints and Preferences
- Qualitative Preferences
- HTN Planning with Preferences
- Handling Partial / Unknown Preference Models

Dimensions of Variation



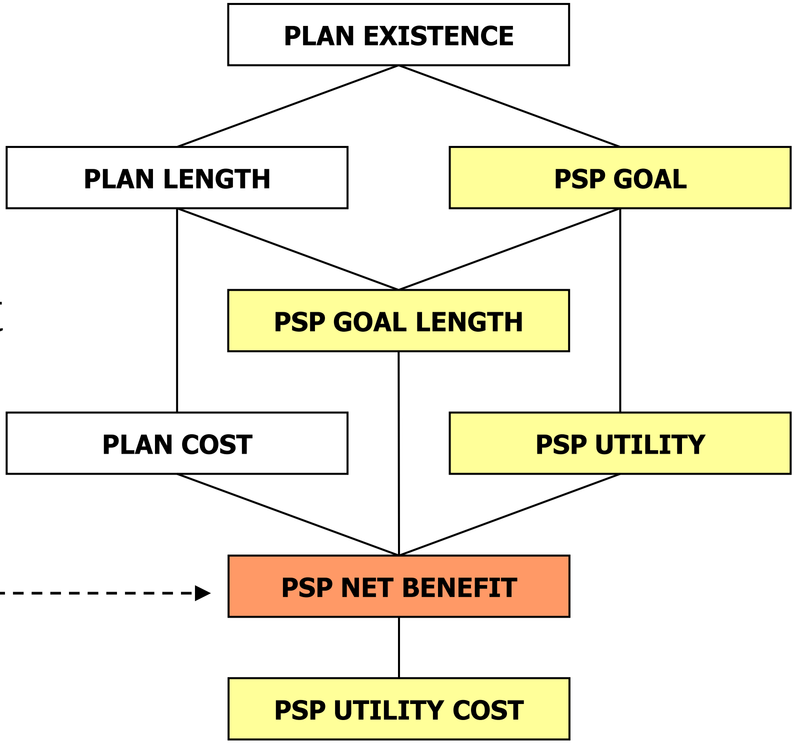
Tutorial Outline

- Planning for net benefit
- Break
- Trajectory Constraints and Preferences
- Qualitative Preferences
- HTN Planning with Preferences
- Handling Partial / Unknown Preference Models

Taxonomy

Maximize the Net Benefit

Actions have execution costs, goals have utilities, and the objective is to find the plan that has the highest net benefit.
→ easy enough to extend to mixture of soft and hard goals



PSP Net Benefit

- A PSP planning instance is a tuple

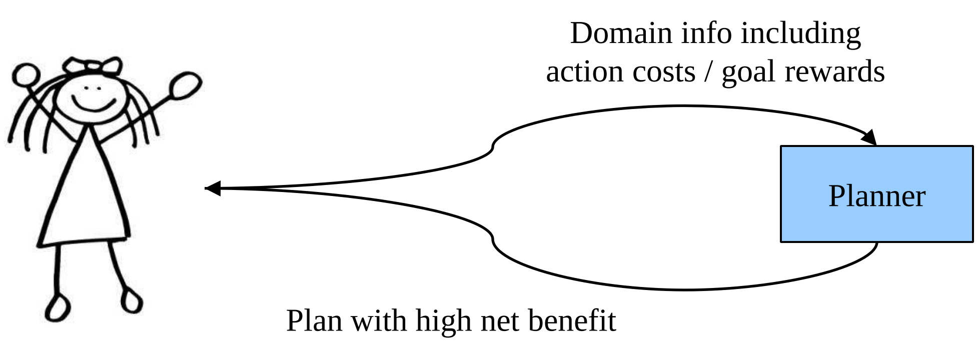
$$I = (S, s_0, O, G, c(a \in O), r(G' \subseteq S))$$

S	=	a set of states
$s_0 \in S$	=	initial state
O	=	set of operators
$G \subseteq S$	=	set of goal states
$c(a \in O)$	=	action cost function
$r(G \subseteq S)$	=	goal state reward function

Task: Find a sequence of operators $a_1, a_2, \ldots, a_n \in O$ that will produce the best net benefit state $g \in G$ when applied to s_0. Where net benefit is defined as **$r(G) - \Sigma\, c(a_i)$**.

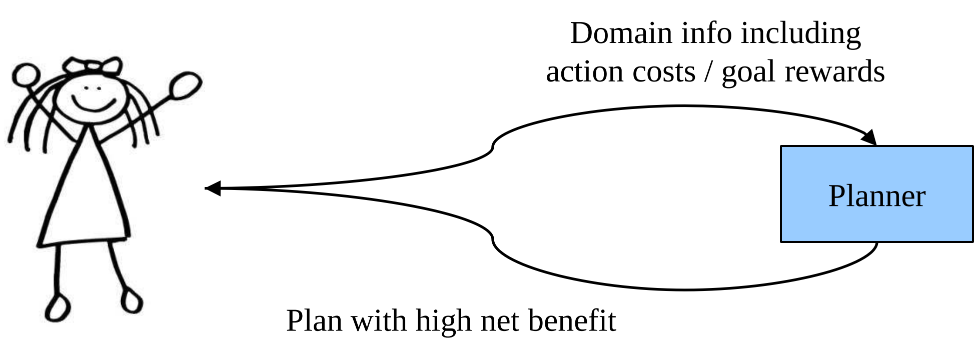
One Metric to Rule Them All: Net Benefit

- Reward is a function of the final state
  - » Goal achievement grants reward to the user
  - » Negative reward (i.e., penalty) for failing to achieve goals
- User models action costs and goal rewards that seem fitting to the domain



One Metric to Rule Them All: Net Benefit

- Reward is a function of the final state
    - » Goal achievement grants reward to the user
    - » Negative reward (i.e., penalty) for failing to achieve goals
- User models action costs and goal rewards that seem fitting to the domain
- What if cost and reward are not on the same metric?
    - Resource Constrained Net Benefit
        - Given a fixed, limited resource (e.g., battery) find the best *net benefit* plan



General Additive Independence Model

[Bacchus & Grove, 1995; Do et al., 2007]

- Goal Cost Dependencies come from the plan
- Goal Utility Dependencies come from the user

Utility over sets of dependent goals

$$S \subseteq G \longrightarrow f(S) \in R$$

$$U(G) = \sum_{S \subseteq G} f(S)$$

Util: 20

$f(So) = 20$

Util: 50

$f(Sh) = 50$

Util: 300

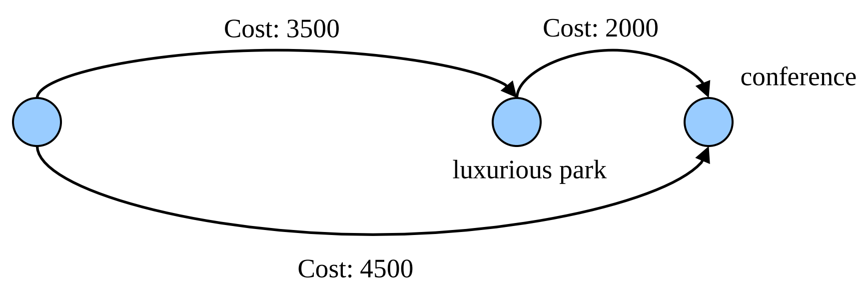
$f(\{So, Sh\}) = 230$

$$U(\{So, Sh\}) = 20 + 50 + 230 = 300$$

The Planning Dilemma

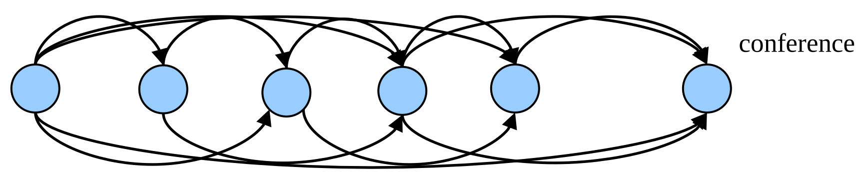
– Cost-dependencies on plan benefit among goals

Goals: G1:(at student conference) G2:(visited luxurious_park)
Reward: 6000 Reward: 600



Net benefit
G1 & G2:
6600 – 5500 = 1100
G1:
6000 – 4500 = 1500
G2:
600 - 3500 = -2900
(null):
0 - 0 = 0

– Impractical to find plans for all 2^n goal combinations



Net Benefit in PDDL 3.0

- The Planning Domain Description Language (PDDL)
  - Standard for the International Planning Competitions (IPC)
  - PDDL 3.0 added preferences
    - » "Simple Preferences" – Fragment of PDDL 3.0
      - Can compile to *Net Benefit*
    - » IPC 2006 had one strictly *Net Benefit* domain
    - » IPC 2008 had an optimal *Net Benefit* track

PDDL 3.0 – "Simple Preferences"

- "Simple Preferences" as *net benefit*
  - Action costs

```
(:action open-new-stack
   :parameters (?open ?new-open - count)
   :precondition (and (stacks-avail ?open) (not (making-product))
                      (next-count ?open ?new-open))
   :effect (and (not (stacks-avail ?open))
                (stacks-avail ?new-open)
                (increase (total-cost) (stack-cost)))
   )
```

  - Soft Goals

```
(preference d-o1-p1 (delivered o1 p1))
(preference d-o1-p2 (delivered o1 p2))
```

  - Specify *reward*, maximize *net benefit*

```
(:metric maximize (- 30 (+(total-cost)
(* (is-violated d-o1-p1) 20)
(* (is-violated d-o1-p2) 10))
```

"violation cost" = reward

Various Substrates for *Net Benefit*

- MDP
  - Optimal
- Integer Programming
  - Bounded-optimal (optimal in plan length *k*)
- MaxSAT
  - Bounded-optimal (optimal in plan length *k*)
- Heuristic Search
  - Optimal
  - Anytime Optimal (asymptotically reach optimal)
  - Satisficing (no optimality guarantees)

Scalability Improves

Optimization Methods: MDP

[Sanchez & Kambhampati 2005]

- No probability
  - Deterministic MDP
- Prevent repeated reward collection
  - Bad idea: Make every state for which any subset of the holds hold into a sink state using a summed reward of the subset (reify achievement)
    - What if achieving goal g2 requires passing through states with g1 already achieved
  - Good idea: Create a proposition "done" and an action "finish" that has "done" as an effect and is applicable in any state. "done" with no applicable actions and reward equal to the sum of goal rewards.
- Can find optimal policy

Optimization Methods: Integer Programming

[van den Briel et al., 2004]

- Optiplan / iPUD
  - Encode planning graph
  - Use binary variables
  - $V(p) = \{0,1\}$: **p** is goal
- Constraints:
  - »$V(a) = 1 \rightarrow V(Pre(a)) = 1$
    - »If an action *a*'s conditions are satisfied, require the action
  - »$V(p) = 1 \rightarrow \Sigma\, V(a) \geq 1$; p in Effect(a)
    - »If an action gives a proposition, require that proposition
  - »$V(p) = 1$: **p** is in initial state

  Objective function for classical planning: **minimize** $\Sigma\, V(a)$
- IP Encoding for OSP
  - **maximize** $\Sigma\, V(g).U(g) - \Sigma\, V(a).C(a)$
- Bounded-length optimal

Optimization Methods: Weighted MaxSAT

[Russell & Holden 2010]

- Extend SATPLAN (Kautz, et al. 1999)
  - Encode planning graph as a SAT problem
- Max(Σ Achieved Rewards – Σ Action Costs) =

  Min (Possible Reward - Σ Unachieved Rewards) – Σ Action Costs
  - Weigh violated clauses:
    - For action a clause: "~a" violated with cost c(a)
    - For goal set g clause: "~g" violated with cost r(g)
- Beats IP approach in scalability
- Bounded-length optimal

Optimization Methods

- MDP model: Optimal
- IP model: Bounded-optimal
- MaxSAT model: Bounded-optimal

Scalability Improves

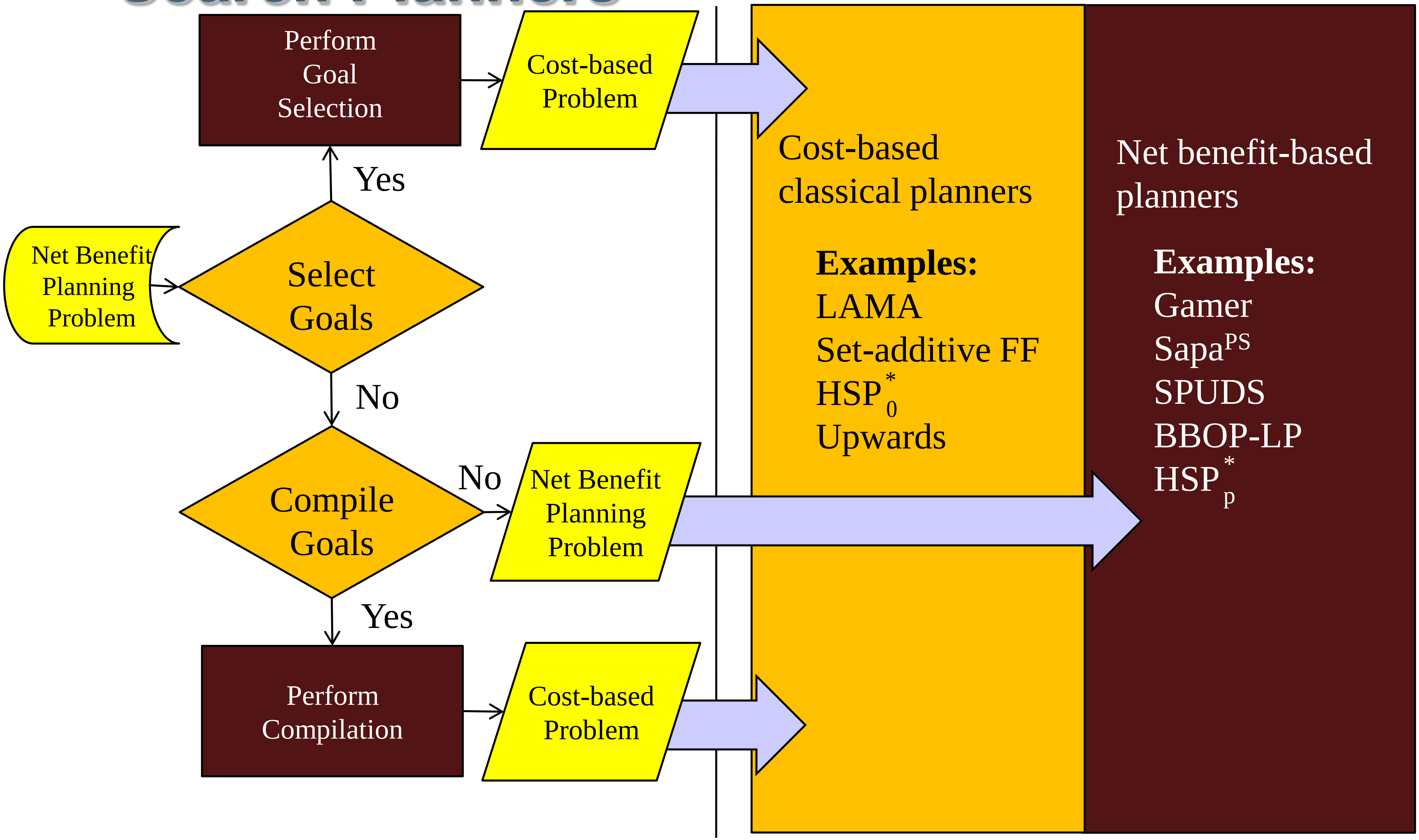
Optimization Methods

- MDP model: Optimal
- IP model: Bounded-optimal
- MaxSAT model: Bounded-optimal

Scalability Improves

- Big Problem: These methods fail to scale as well as modern heuristic planners
  - Can we leverage the benefits of current state-of-the-art planners to handle the partial satisfaction *net benefit* planning problems?

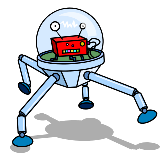
How to Leverage Modern Heuristic Search Planners

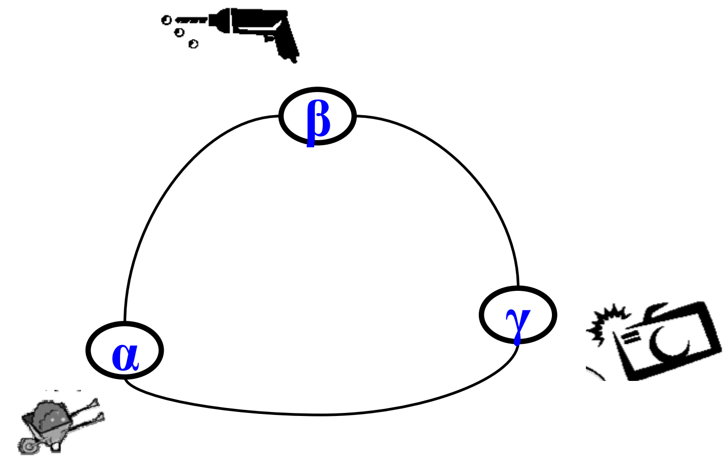


Preliminaries: Planning Problem

Actions:

Move(α,β)
Sample(Soil,α)
Sample(Rock,β)
Take(Picture,γ)





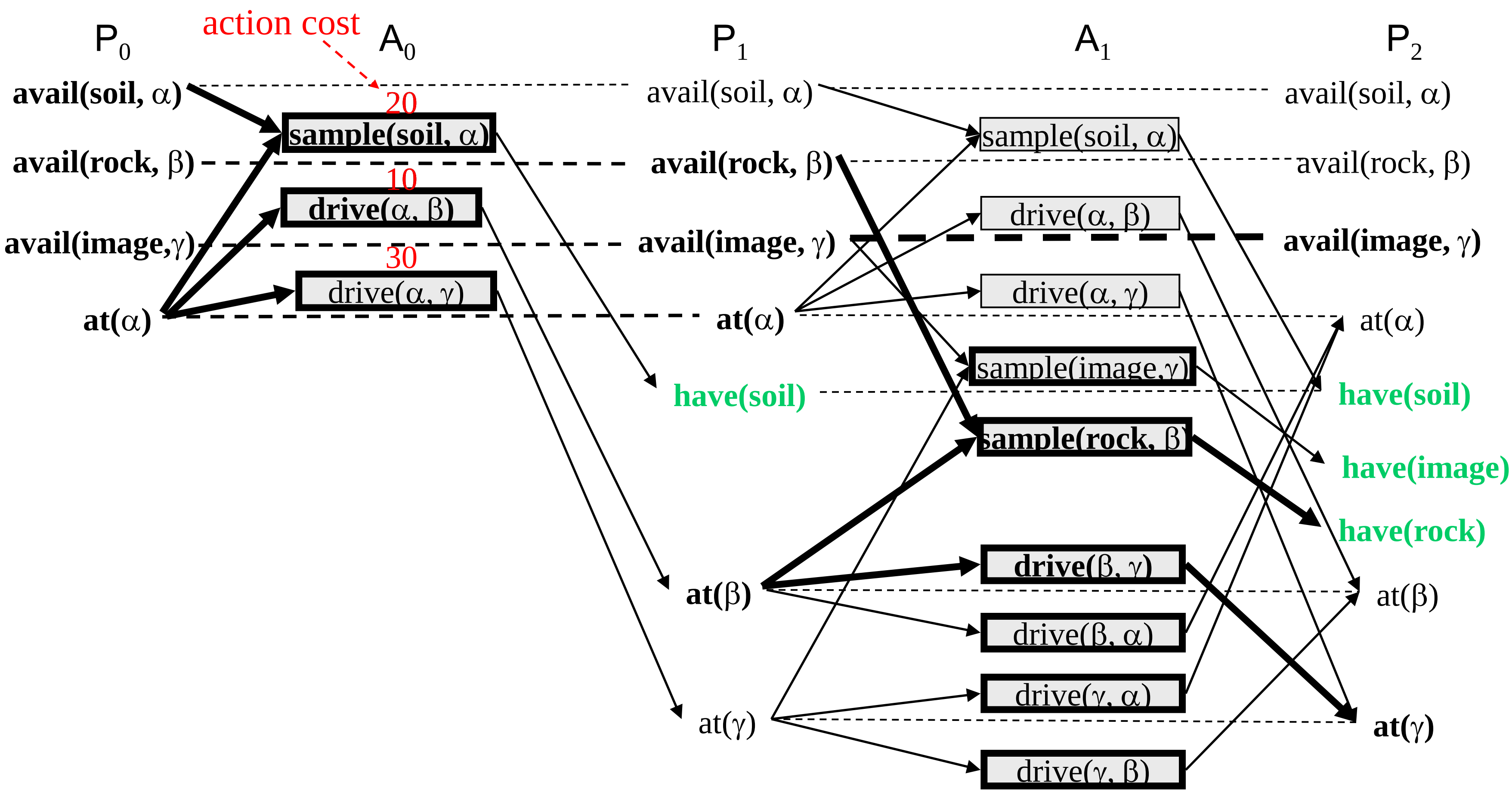
[Bryce & Kambhampati, 2006]

- Planning Problem in STRIPS:
  - Domain:
    - » Set of binary literals representing world state
      - At(Rover,α), HaveImage(γ)
    - » Actions: preconditions → effects
      - Move(α,β): At(Rover,α) → At(Rover,β)
  - Initial state: fully specified
    - » At(Rover,α), Available(Soil,α), Available(Rock,β), Visible(Image,γ)
  - Goal state: partially specified
    - » Have(Soil), Have(Rock), Have(Image)
- *Soft*-goals with utilities:

  U(Have(Soil)) = 20, *U(Have(Rock))* = 50, *U(Have(Image))* = 30

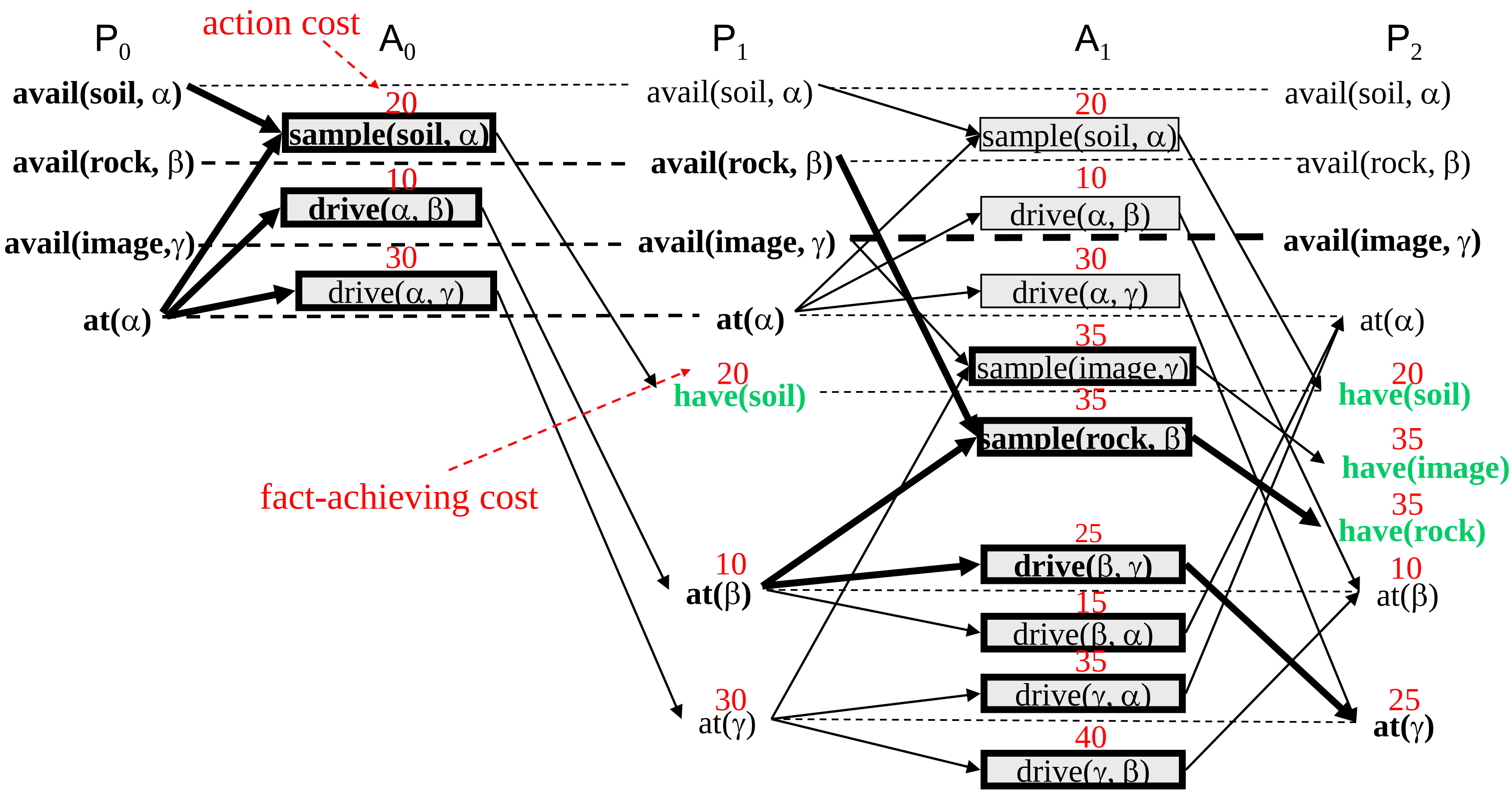
Sum Cost Propagation on the Relaxed Planning Graph (RPG)

[Do & Kambhampati, 2002]



Sum Cost Propagation on the Relaxed Planning Graph (RPG)

[Do & Kambhampati, 2002]



Using a Cost-based Classical Planner

Select Goals Up-Front

Each selected goal becomes a hard goal

AltAltPS (2004 / 2005)
Smith's Orienteering Approach (2004)
Garcia-Olaya et al.'s Orienteering Approach (2008)

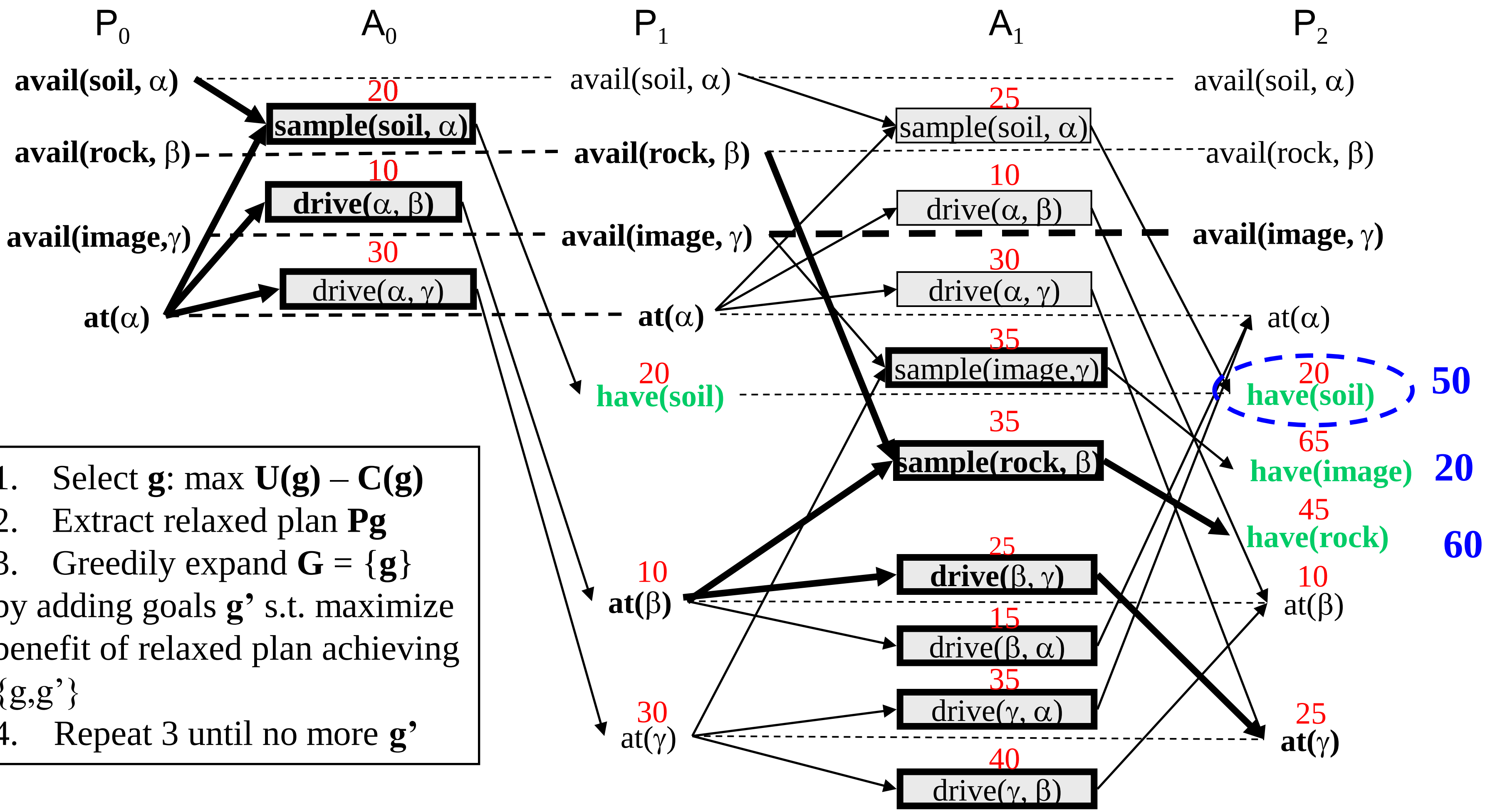
Compile the Net Benefit Problem

Each soft goal set becomes a set of actions and hard goal

Keyder & Geffner Compilation (2007 / 2009)

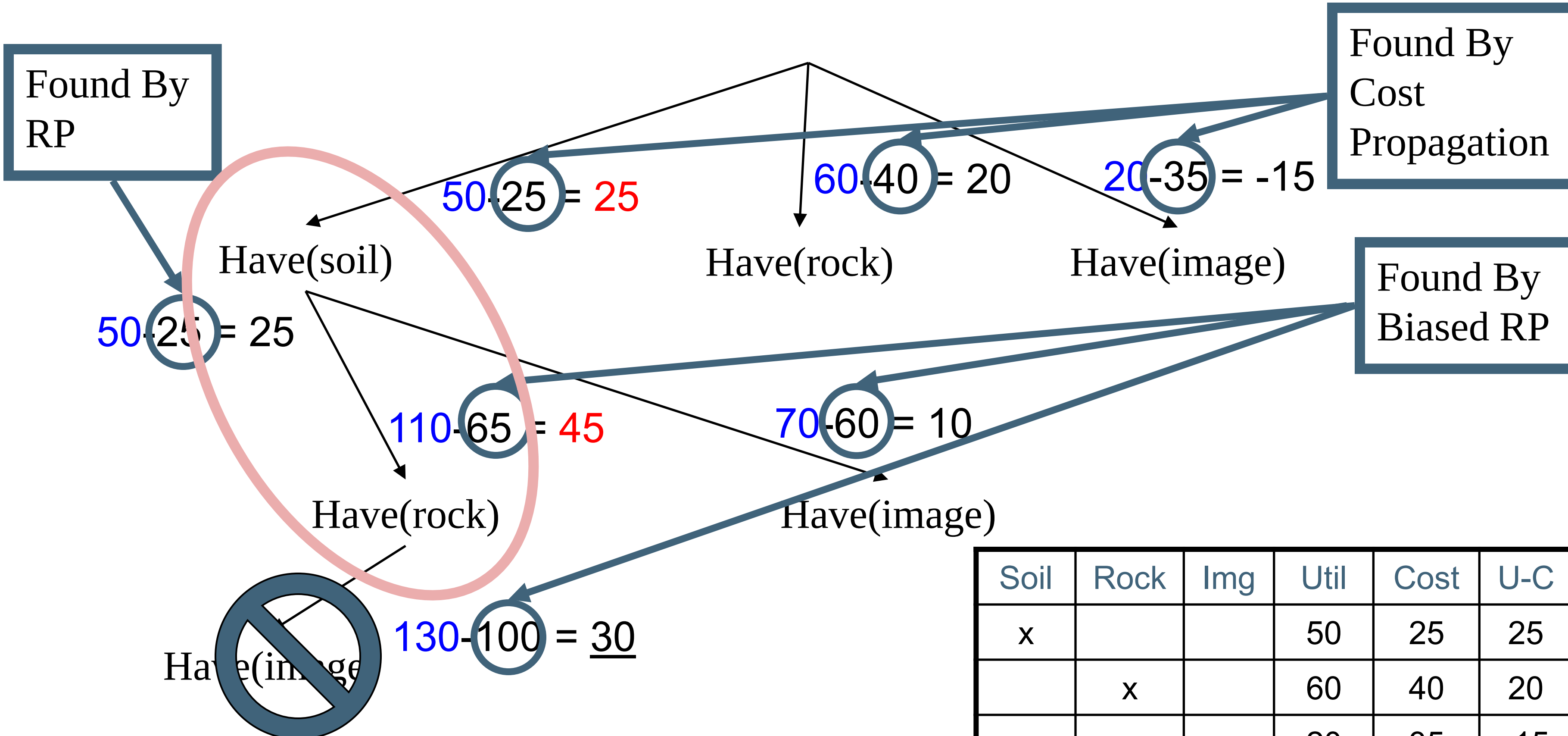
AltAltPS: Goal Selection using Propagated Cost

[van den Briel et al., 2004; Sanchez 2005]



1. Select **g**: max **U(g)** – **C(g)**
2. Extract relaxed plan **Pg**
3. Greedily expand **G** = {**g**} by adding goals **g'** s.t. maximize benefit of relaxed plan achieving {g,g'}
4. Repeat 3 until no more **g'**

AltAltPS: Goal Set Selection



Soil	Rock	Img	Util	Cost	U-C
x			50	25	25
	x		60	40	20
		x	20	35	-15
x	**x**		**110**	**65**	**45**
x		x	70	60	10
x	x	x	130	100	30

Does AltAltPS work?

Problem: Relaxed problem ignores negative interactions:

- Might cause us to choose mutex goals
- or goals that produce poor quality plans

Does AltAltPS work?

Problem: Relaxed problem ignores negative interactions:

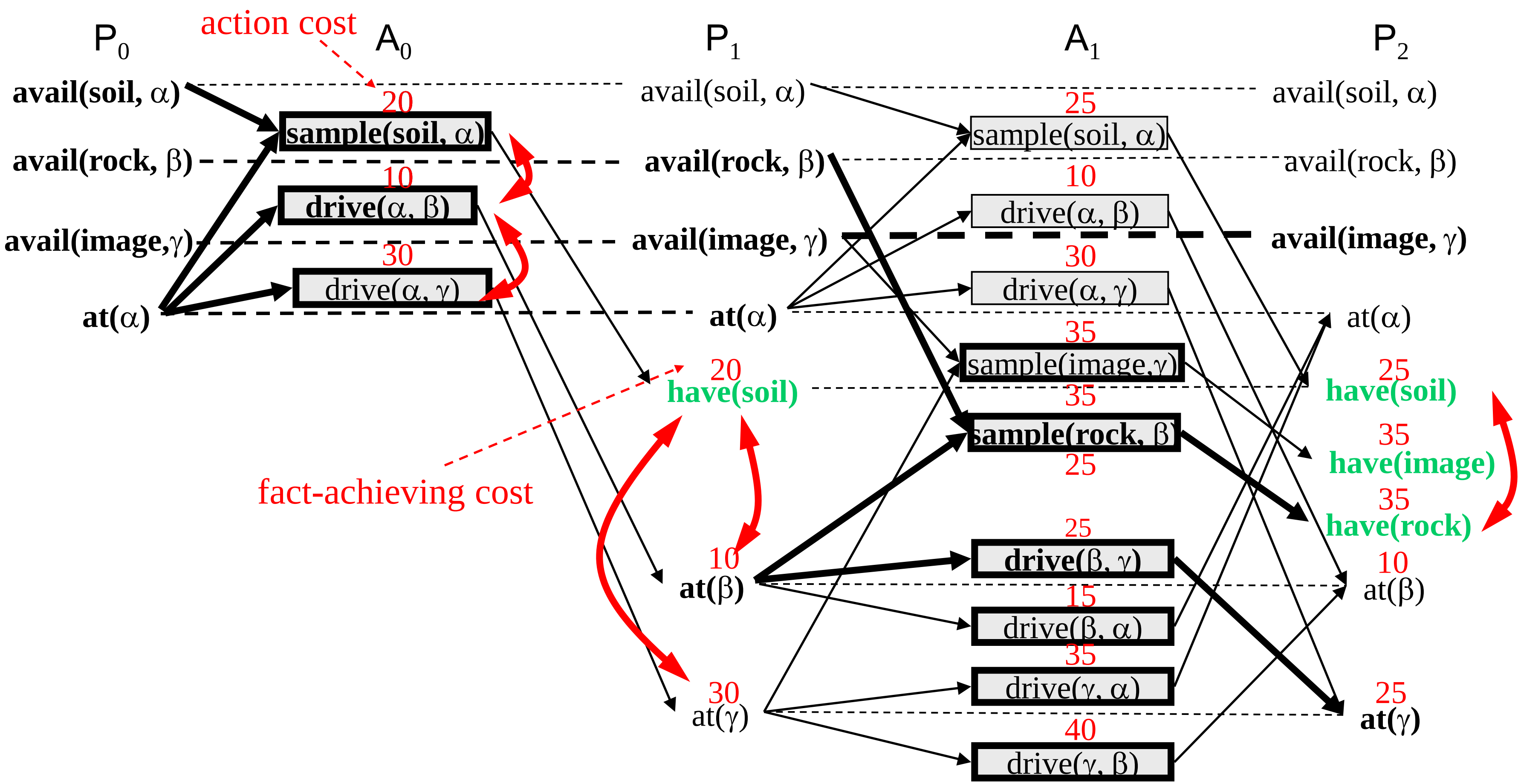
- Might cause us to choose mutex goals
- or goals that produce poor quality plans

- Potential solution:
  - Use mutex analysis in RPG for negative interactions
    - » Use "propagated" mutexes from static binary mutexes
    - » Add *penalty* cost for goal sets that involve more mutual exclusions for achievement
      - $\mathbf{Max}_{(g1,\ g2)}$ {lev(g1, g2) – max(lev(g1), lev(g2)) }
        - Distance between first appearance of one of the goals and the level in which the goals are not mutex (infinity if this never happens)
      - *Penalty cost is the highest mutex subgoal cost to the goals*
    - » Incrementally add goals based on estimate over extracted relaxed plan

AltWlt: Improve AltAlt^{PS} with Mutexes

Mutexes in Planning Graph

[Sanchez & Kambhampati, 2005]



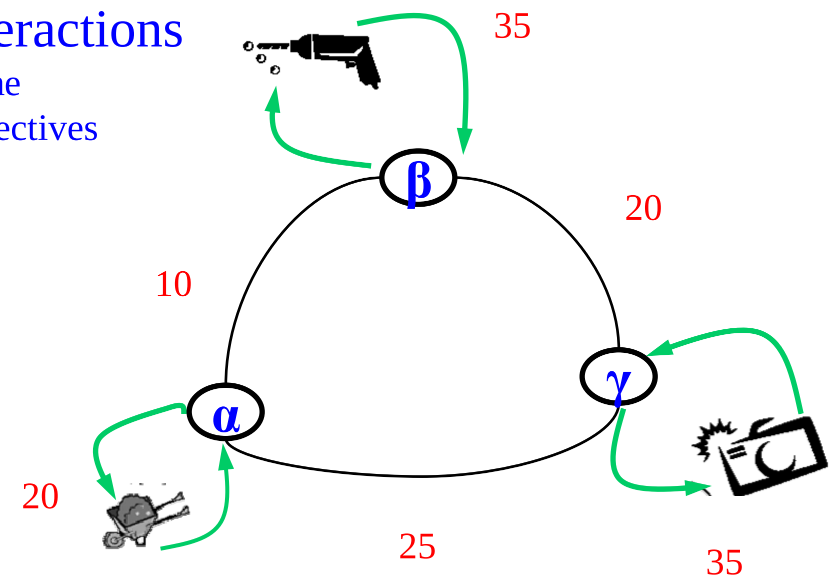
PG + Orienteering Problem

[Smith, 2004]

» Improve with more negative & positive interactions
- **Negative:** can not move to two locations at the same time
- **Positive:** move to one location can achieve multiple objectives

Orienteering Problem (variation of TSP):
- Set of linked cities
- Reward for visiting each city
- Maximize reward with limited "gas"

» Suitable for "Transportation" domains



1. **Cost-propagation**: estimate cost to do experiment at each location
2. **OP**: use path-planning to build the orienteering graph
3. **Solve OP** and use the results to select goals and goal orderings

PG + OP: Generalization

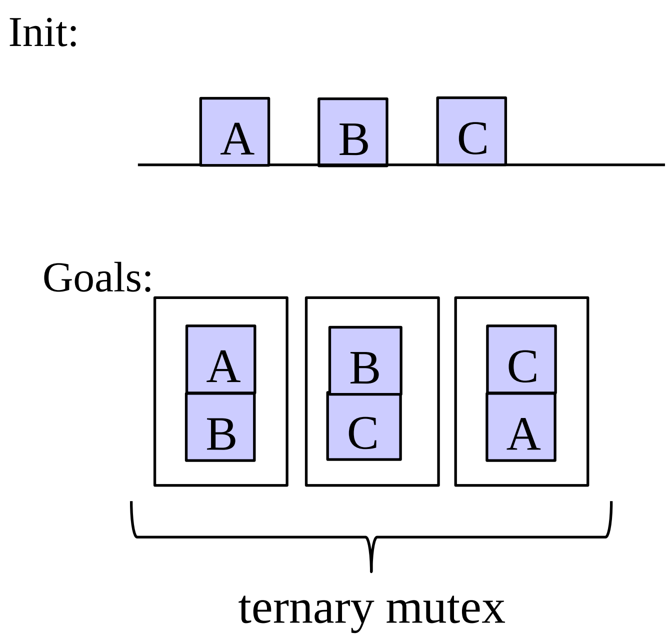
[Smith, 2004]

- Abstraction: select subset **L** of *critical* literals (basis set)
  - » Based on relaxed plan analysis
- Build *state-transition graph* **G** based on **L** (project the state-space on **L**)
  - – Set **G** as an orienteering graph
- Based on solving **OP** and relaxed plan at each node, select:
  1. Beneficial goal (sub)set **S**
  2. Order in which goals in **S** need to be achieved

➢ Planning search guided by goal ordering received from solving OP

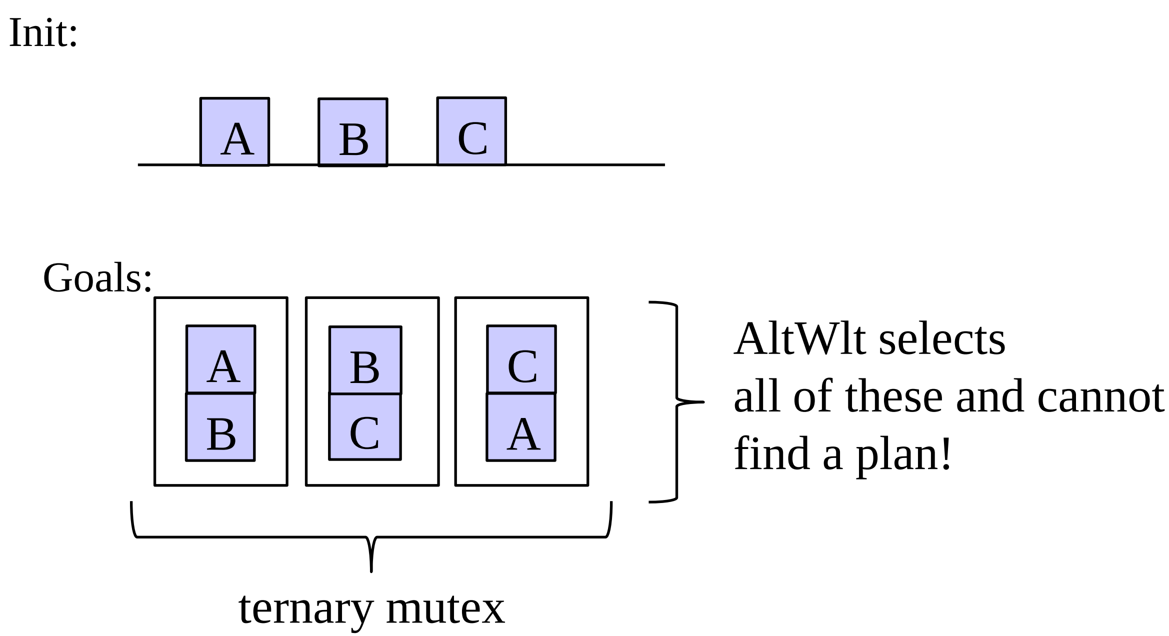
Goal Selection: Bad News

- Easy to have n-ary mutexes in “non-transportation” domains
  – Example: Blocksworld



Goal Selection: Bad News

- Easy to have n-ary mutexes in “non-transportation” domains
  - Example: Blocksworld



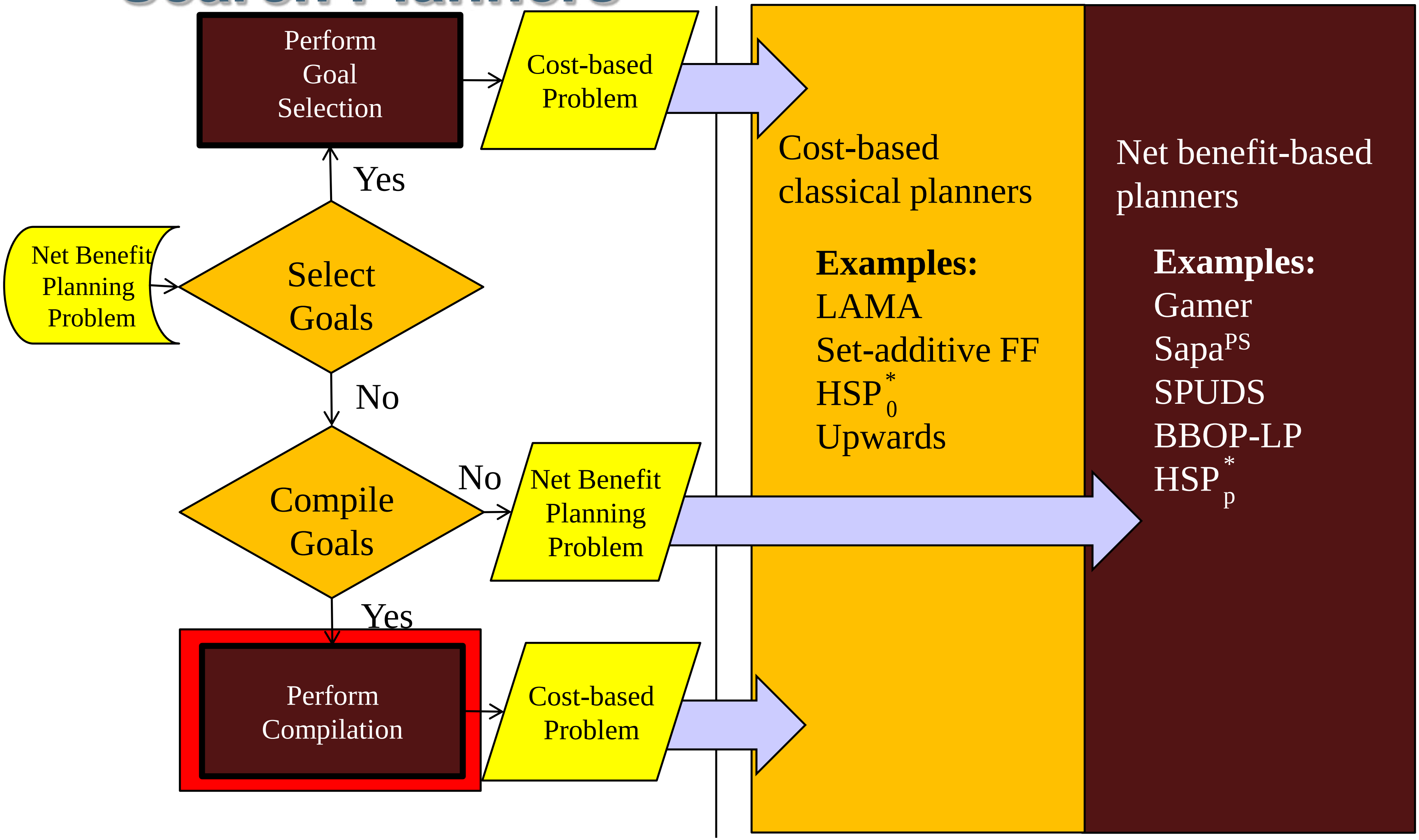
HSP^*_p using IDA* Goal Selection

[Haslum, 2008]

- Optimal planner
- Generates a minimization version of the problem
- Regression using a cost-propagation heuristic
  - For each goal set, find a lower bound on cost using the heuristic
  - Perform IDA* search on the best looking goal set (based on *net benefit*)
  - For each IDA* iteration, update the cost bound (monotonically increases)
    - If there exists a goal set that appears to have a better potential *net benefit*, switch to searching on that goal set

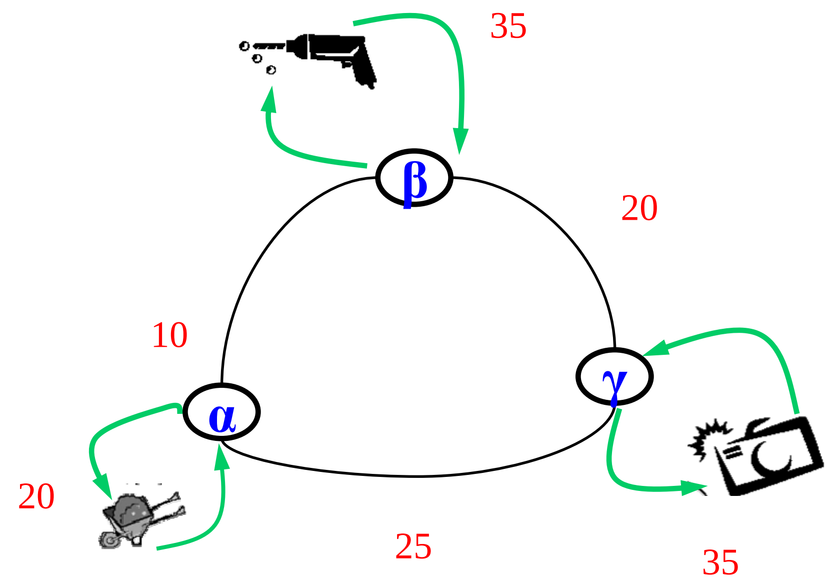
Set Cost	{} 0	{soil} 50	{rock} 60	{image} 20	{soil, rock} 110	{soil, image} 70	{image, rock} 80	{soil, rock, image} 130
R	0	20	45	60	55	80	100	100
NB	0	30	15	-40	55	-10	-20	30

How to Leverage Modern Heuristic Search Planners



Soft Goal Compilation

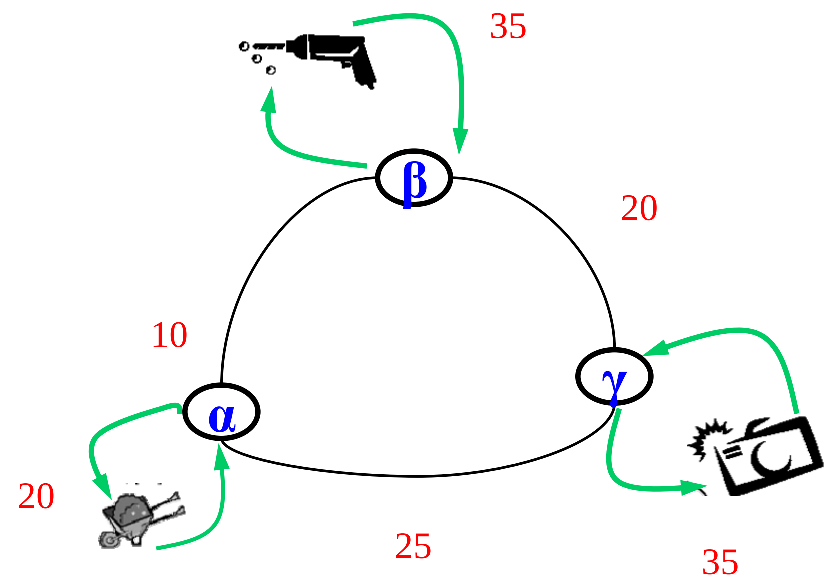
What's the first step to making a soft goal problem into an equivalent hard-goal problem?



Soft Goal Compilation

Make some hard goals…

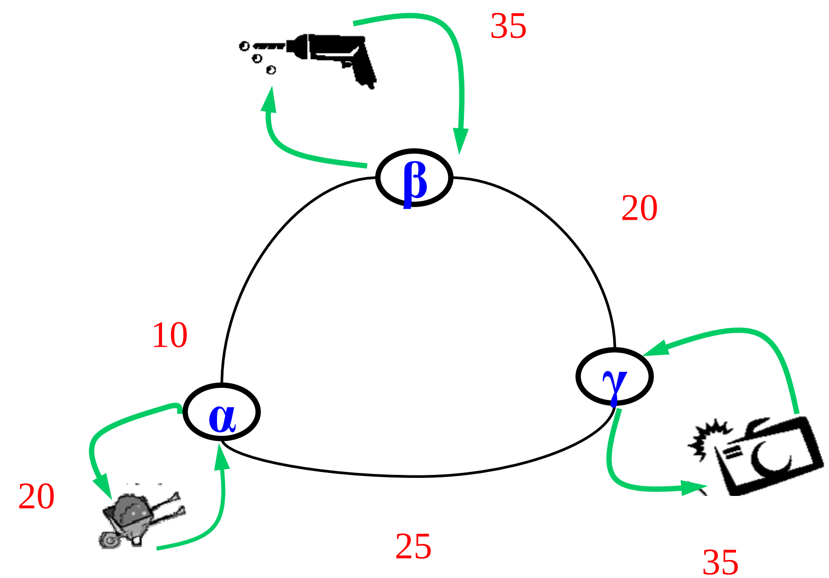
Soft Goal	Hard Goal
Have(soil)	Have(soil)’
Have(rock)	Have(rock)’
Have(image)	Have(image)’



Soft Goal Compilation

Make some hard goals…

Soft Goal	Hard Goal
Have(soil)	Have(soil)’
Have(rock)	Have(rock)’
Have(image)	Have(image)’



Evaluation actions that give hard goal version:

forgo-have(soil)	Pre: ~Have(soil)	*claim-have(soil)*	Pre: Have(soil)
forgo-have(rock)	Pre: ~Have(rock)	*claim-have(rock)*	Pre: Have(rock)
forgo-have(image)	Pre: ~Have(image)	*claim-have(image)*	Pre: Have(image)

Soft Goal Compilation

Max-to-min

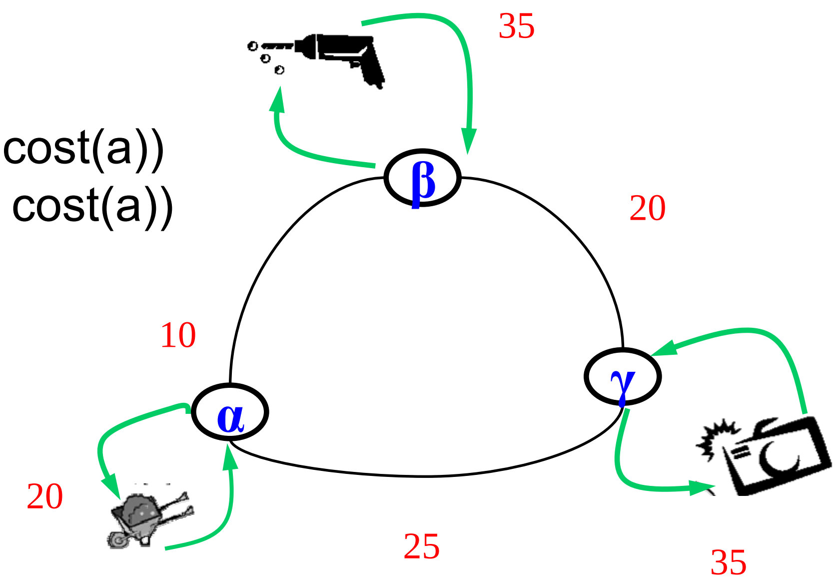
Reward for all not achieved

Net benefit

= max (Σ reward(g) - Σ cost(a))

= max (possible reward - Σ reward(~g)) - Σ cost(a))

= min (Σ reward(~g) – possible reward) + Σ cost(a))



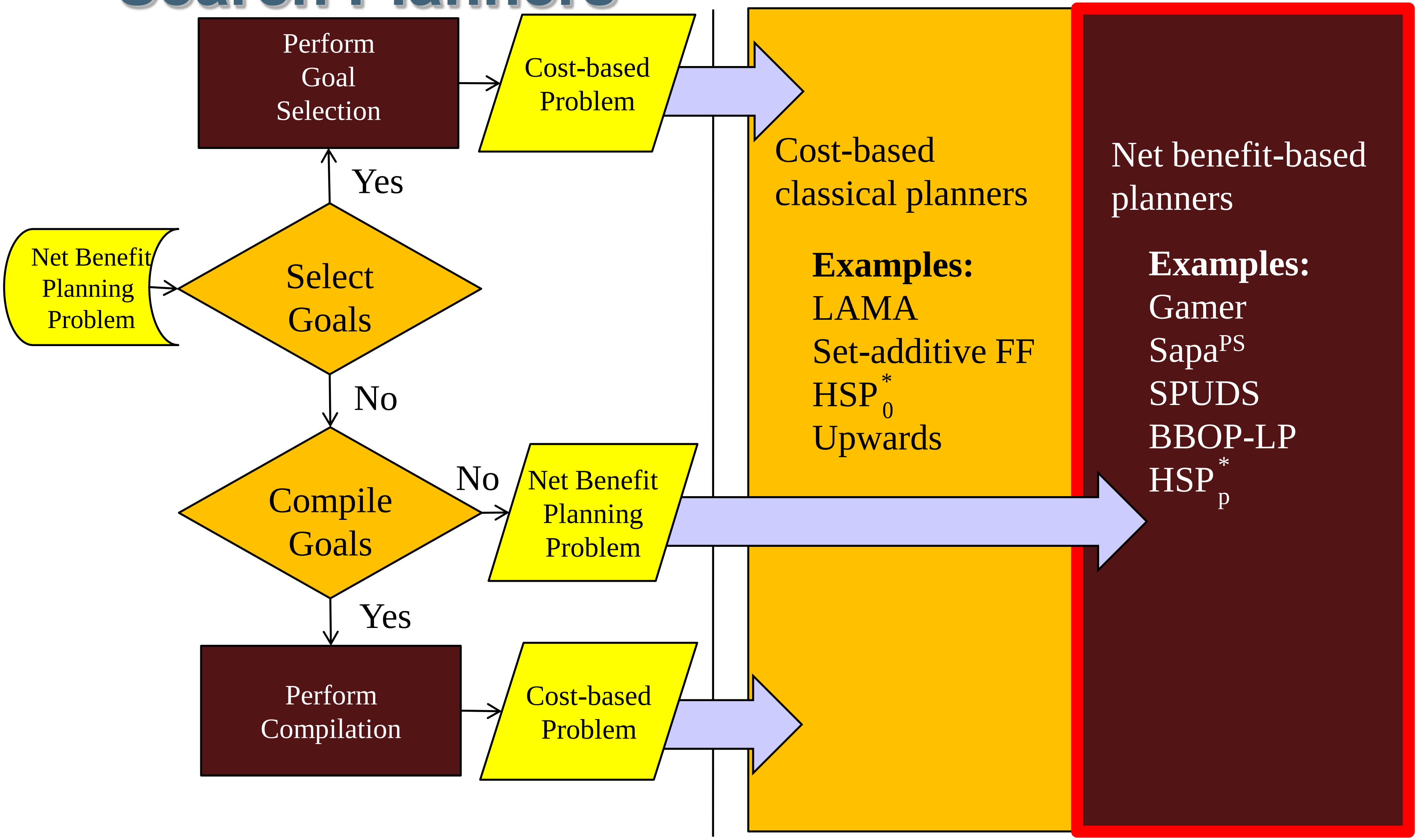
forgo-have(soil)	Pre: ~Have(soil) Cost = reward(Have(soil))	*claim-have(soil)*	Pre: Have(soil) Cost = 0
forgo-have(rock)	Pre: ~Have(rock) Cost = reward(Have(rock))	*claim-have(rock)*	Pre: Have(rock) Cost = 0
forgo-have(image)	Pre: ~Have(image) Cost = reward(Have(image))	*claim-have(image)*	Pre: Have(image) Cost = 0

Soft Goal Compilation

[Keyder & Geffner 2007, 2009; Benton et al., 2009; Russell & Holden 2010; Geffner & Bonet, 2006; Sanchez & Kambhampati, 2005]

- Compilation from soft goal *net benefit* to equivalent cost-based planning problem
  - Basic compilation, for every soft goal *g*:
    - Generate a hard goal *g’,* and actions *forgo* and *claim*
    - Reward(*g*) cost: *forgo;* takes ~g as a precondition and has the effect *g’*
    - 0 cost: *claim;* takes *g* as a precondition and gives the effect *g’*
    - Conversion to from max-to-min like MaxSAT method
  - More compilation tricks; generate a “done” space:
    - Create a hard goal “done” with an action “make-done” that gives “done”
    - Only allow *forgo*s and *claim*s to occur after *done* is true
    - Good idea for *satisficing* planners (otherwise you have to delete *g’* everytime you change the value of *g*)
    - Same idea as MDP
- For PDDL3 “simple preferences”
  - Similar compilation in YochanPS / YochanCost

How to Leverage Modern Heuristic Search Planners



Gamer

- Symbolic branch and bound search
  - Uses BDDs to represent sets of states
- Generates a minimization version of the problem
- Bi-directional perimeter search
  - First performs a regression search to construct a partial pattern database heuristic
  - Then performs a forward breadth-first symbolic search
- For cost-based planning can prune poor-valued states
- For Net Benefit: Cannot prune since reward on goals can cause non-monotonic changes

$Sapa^{PS}$: Anytime BFS Forward Search

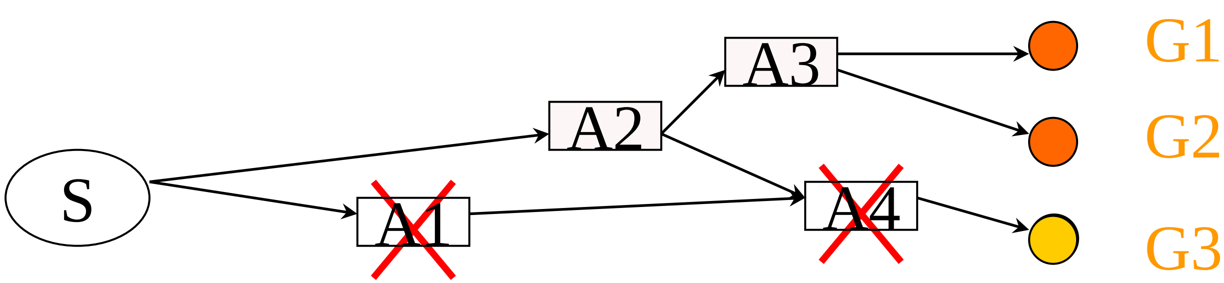
[van den Briel et al., 2004; Do & Kambhampati 2004; Benton, et al., 2009]

- Anytime PSP search (best-first branch and bound)
  - Return better solutions as they are found (any node can be solution)
- Variation of **A***: **f = g + h** (with negative edge cost)
  - **Edge cost** (S,a,S'): (Util(S') – Util(S)) – Cost(a)
  - **g**-value: net-benefit of (real) plan so far
  - **h**-value: (relaxed plan) estimate of benefit to go to achieve the best goal set
    - » Relaxed plan found for all goals
    - » Iterative goal removal, until net benefit does not increase
  - **Anytime:** returns plans with increasing **g**-values.
  - If we reach a node with **h** = 0, then we know we can stop searching (no better solutions can be found)
    - » Optimal if **h** is admissible (over-estimate)

$\underbrace{\textbf{SPUDS}}_{\text{Sapa}^{\text{Ps}}\text{ Utility DependencieS}}$: Heuristic

[Do et al., 2007]

- Extends Sapa^{PS} to handle Goal Utility Dependencies by solving an IP encoding of the relaxed plan
- Extracts a usual relaxed plan
  - Encode relaxed plan as an IP with special attention to cost and utility (reward) dependencies
  - Solve the IP to find the optimal set of goals, G, for the relaxed plan
    - » Remove non-optimal goals and actions not involved in achieving G



$\underbrace{\text{BBOP-LP}}_{\text{Branch and Bound Oversubscription Planning with Linear Programming}}$: Heuristic

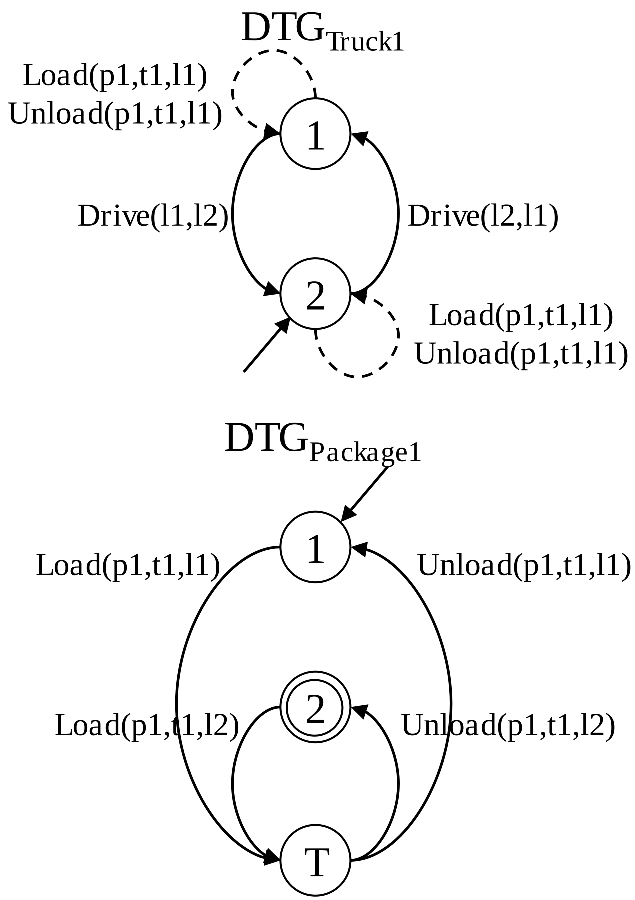
"Be-bop a' loop"

A Tribute to Georgia native Little Richard

[Benton et al., 2007]

- Uses a unique integer programming-based heuristic
  - Based on network flow model of planning problem
  - Maintains negative interactions (unlike planning graph heuristics)
  - Relaxes *ordering* of actions as against *delete effects*
  - Admissible

BBOP-LP : Heuristic



(Similar to orienteering planner)

BBOP-LP : Heuristic

- Further relaxation: Solves the linear program relaxation
- Over all better quality heuristic than SPUDS/Sapa^{PS}
  - Heuristic is slower than SPUDS
  - Can affect scalability when the degree of interactions between *fluents* is high

BBOP-LP **: Lookahead Search**

- "Lookahead" in the search space using a relaxed plan
  - Extract the relaxed plan using the LP solution as a guide
  - Prefer actions that also appear in the LP solution
- Generate sets using only actions in the relaxed plan
  - Finds new solutions (i.e., upper bound values) more quickly
  - Provides an *anytime optimal* behavior

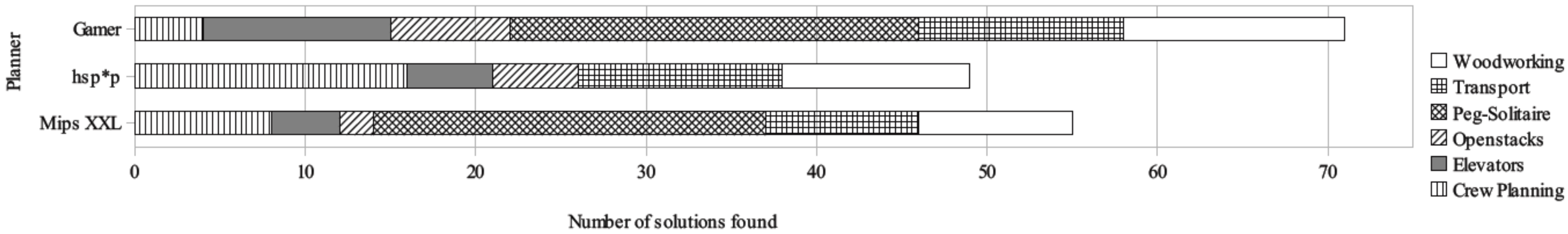
What wins?

- MDP
  - Optimal
- Integer Programming
  - Bounded-optimal
- MaxSAT
  - Bounded-optimal
- Heuristic Search
  - Optimal
  - Anytime Optimal
  - Satisficing

Scalability Improves

What wins?

Gamer won the IPC-2008 net benefit optimal planning track



From [Edelkamp & Kissmann 2009]

	Net-benefit optimal planners			Sequential optimal planners			
Domain	Gamer	HSP^*_P	Mips-XXL	Gamer	HSP^*_F	HSP^*_0	Mips-XXL
crewplanning(30)	4	16	8	-	8	**21**	8
elevators (30)	11	5	4	**19**	8	8	3
openstacks (30)	**7**	5	2	6	4	6	1
pegsol (30)	24	0	23	22	**26**	14	22
transport (30)	12	12	9	-	**15**	**15**	9
woodworking (30)	13	11	9	-	10	**14**	7
total	71	49	55		71	**78**	50

From [Keyder & Geffner 2009]

Compiled soft goals

What wins?

Domain	Net-benefit satisficing planners			Cost satisficing planners
	SGPlan	YochanPS	Mips-XXL	Lama
elevators (30)	0	0	8	**23**
openstacks (30)	2	0	2	**28**
pegsol (30)	0	5	23	**29**
rovers (20)	8	2	1	**17**
total	10	7	34	**97**

Compiled soft goals

From [Keyder & Geffner 2009]

References

[Bacchus & Grove 1995] F. Bacchus and A. Grove; Graphical Models for Preference and Utility; UAI-95, 1995

[Benton et al., 2007] J. Benton, M. van den Briel, and S. Kambhampati; A Hybrid Linear Programming and Relaxed Plan Heuristic for Partial Satisfaction Planning Problems; ICAPS-07, 2007

[Benton et al., 2009] J. Benton, M. Do, and S. Kambhampati; Anytime Heuristic Search for Partial Satisfaction Planning; AIJ, Volume 173, Numbers 5-6, April 2009

[Bryce & Kambhampati, 2007] D. Bryce and S. Kambhampati; How to Skin a Planning Graph for Fun and Profit: A Tutorial on Planning Graph-based Reachability Heuristics; AI Magazine, Spring 2007

[Do et al., 2007] M. Do, J. Benton, and S. Kambhampati; Planning with Goal Utility Dependencies; IJCAI-07, 2007

[Do & Kambhampati, 2002] M. Do and S. Kambhampati; Planning Graph-based Heuristics for Cost-Sensitive Temporal Planning; AIPS-02, 2002

[Do & Kambhampati, 2004] M. Do and S. Kambhampati; Partial Satisfaction (Over-subscription) Planning as Heuristic Search; Knowledge Based Computer Systems (KBCS-04), 2004

[Geffner & Bonet, 2006] H. Geffner & B. Bonet; Heuristics for Planning with Penalties and Rewards using Compiled Knowledge; KR-06, 2006

[Haslum, 2008] P. Haslum; Additive and Reversed Relaxed Reachability Heuristics Revisited; International Planning Competition 2008 booklet; 2008

[Edelkamp & Kissmann, 2009] S. Edelkamp and P. Kissmann; Optimal Symbolic Planning with Action Costs and Preferences; IJCAI-09, 2009

[Keyder & Geffner, 2009] E. Keyder and H. Geffner; Soft Goals Can Be Compiled Away; JAIR, Volume 36, 2009

[Ruml et al., 2005] W. Ruml, M. Do, and M.P.J. Fromherz; On-line Planning and Scheduling for High-speed Manufacturing; ICAPS-05, 2005

[Russell & Holden, 2010] R.A. Russell and S. Holden; Handling Goal Utility Dependencies in a Satisfiability Framework; ICAPS-10, 2010

[Sanchez & Kambhampati, 2005] R. Sanchez and S. Kambhampati; Planning Graph Heuristics for Selecting Objectives in Over-subscription Planning Problems; ICAPS-05, 2005

[Smith 2004] D. Smith; Choosing Objectives in Over-Subscription Planning; ICAPS-04, 2004

[Talamadupula et al. 2010] K. Talamadupula, J. Benton, S. Kambhampati, P. Schermerhorn and M. Scheutz; Planning for Human-Robot Teaming in Open Worlds; ACM Transactions on Intelligent Systems and Technology (TIST), 2010 (accepted for publication)

[van den Briel, et al. 2004] M. van den Briel, R. Sanchez, M. Do, and S. Kambhampati; Effective Approaches for Partial Satisfaction (Over-subscription) Planning; AAAI-04, 2004

Tutorial Outline

- Planning for net benefit
- Break
- Trajectory Constraints and Preferences
- Qualitative Preferences
- HTN Planning with Preferences
- Handling Partial / Unknown Preference Models

PDDL3 and Compilation Approaches

J. Benton[1], **Jorge Baier**[2], Subbarao Kambhampati[1]

[1]Dept. of Computer Science & Engg.,
Fulton School of Engineering
Arizona State University, Tempe Arizona

[2]Departmento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Santiago, Chile

AAAI-2010 Tutorial on Partial Satisfaction Planning
July 12, 2010

PSP has some expressivity limitations

PSP allows the specification of soft goals and actions have costs

PSP does not allow specifying (easily) preferences that:

- Events that occur during the *execution* of a plan.
  E.g. "It would be great to schedule a museum visit"
- Temporal relations between those events.
  E.g. "I want to eat and see a movie, but I prefer to eat first"
- *Hard goals* combined with soft goals.

From PSP to PDDL3

In this session of the tutorial I will:

- Give a brief overview of PDDL3, an extension to PSP
- Show existing techniques to planning with PDDL3

In this session...

- Trajectory Constraints in PDDL3
- IPC-5 Planning Competition
- HPLan-P: Compiling Away Temporally Extended Preferences
- MIPS-BDD and MIPS-XXL: Compiling Away TEPs
- Yochan$^{\mathcal{PS}}$: Compiling Away Precondition Preferences
- PDDL3 planning in any cost-sensitive planner

In this session...

- Trajectory Constraints in PDDL3
- IPC-5 Planning Competition
- HPLan-P: Compiling Away Temporally Extended Preferences
- MIPS-BDD and MIPS-XXL: Compiling Away TEPs
- Yochan$^{\mathcal{PS}}$: Compiling Away Precondition Preferences
- PDDL3 planning in any cost-sensitive planner

PDDL3

PDDL3 was developed by Gerevini, Haslum, Long, Saetti, & Dimopoulos (2009) for the 2006 International Planing Competition.

Based on PDDL2.1. Adds the following features:

- Soft and Hard *trajectory* constraints (in a subset of LTL).
- Conditional costs via precondition preferences.
- Quality of a plan is measured using a metric function.

PDDL overview

PDDL (Planning Domain Definition Language) is the *de facto* standard for describing planning instances.

A planning task is described by two files.

1. A **domain** file, describing actions and types.
2. A **problem** file, describing the initial state and the goal.

PDDL Domain File for Logistics in PDDL

```
(define (domain logistics-strips)
  (:requirements :strips)
  (:predicates
               (at ?obj - MOBILE ?loc - LOCATION)
               (in ?obj1 - OBJ ?obj2 - MOBILE))

    (:types TRUCK AIRPLANE OBJ - MOBILE LOCATION CITY) ; default object

...

(:action load_truck
  :parameters
   (?obj - OBJ ?truck - TRUCK ?loc - LOCATION)
  :precondition
   (and (at ?truck ?loc) (at ?obj ?loc))
  :effect
  (and (not (at ?obj ?loc)) (in ?obj ?truck)))


(:action load_airplane

 ;; details omitted

)
...
```

PDDL3 Constraints

PDDL3 constraints (soft and hard) are declared under `(:constraints ...)`

A PDDL3 soft-constraint is denoted by the keyword `preference`.

Important: In the PDDL3 jargon, a “preference” is just a formula that may or may not hold in a plan.

Soft goals (a type of soft constraint), may be declared in the `(:goal ...)` section of the problem definition.

Temporally Extended Constraints: Examples I

```
(:constraints  (and

;; Go to recharging station after holding a heavy object

 (preference  cautious
     (sometime-after (exists (?x - heavy-obj) (holding ?x))
                     (at recharging-station)))


;; Never pick up an explosive object

 (always (forall (?x - explosive) (not (holding ?x))))


;; each block should be picked up at most once:
 (forall (?b - block) (at-most-once (holding ?b)))
..)
```

Temporally Extended Constraints: Examples II

```
(:constraints

;; We prefer that every fragile package to be transported
  is insured

  (and (forall (?p - package)
       (preference P1 (always (implies (fragile ?p)
                                       (insured ?p)))))

;; Soft goals expressed as a preference in the goal section

(:goal (and (at package1 london)
       (preference (at package2 london))
...) )
```

Semantics: Preliminary definitions

- As before, a state is a collection of atoms (facts).
- $S \models \varphi$ denotes that φ is satisfied in S.
- A PDDL domain D describes the actions and object types.

Definition (Trajectory, Gerevini et al. (2009))

Given a domain D, a plan π and an initial state I, π generates the trajectory

$$(S_0, 0), (S_1, t_1), ..., (S_n, t_n)$$

iff $S_0 = I$ and each state-time pair (S_{i+1}, t_{i+1}) corresponds to the application of an action in π to (S_i, t_i). Furthermore all actions in π have been applied in the correct order.

Semantics of Temporally Extended Formulae

Let $\sigma = \langle (S_0, t_0), \ldots, (S_n, t_n) \rangle$

$\sigma \models (\texttt{always}\ \phi)$	iff	$\forall i : 0 \leq i \leq n\ S_i \models \phi$
$\sigma \models (\texttt{sometime}\ \phi)$	iff	$\exists i : 0 \leq i \leq n\ S_i \models \phi$
$\sigma \models (\texttt{at-end}\ \phi)$	iff	$S_n \models \phi$
$\sigma \models (\texttt{sometime-after}\ \phi\ \psi)$	iff	$\forall i : 0 \leq i \leq n$ if $S_i \models \phi$ then $\exists j : i \leq j \leq n\ S_j \models \psi$
$\sigma \models (\texttt{sometime-before}\ \phi\ \psi)$	iff	$\forall i : 0 \leq i \leq n$ if $S_i \models \phi$ then $\exists j : 0 \leq j < i\ S_j \models \psi$

Important Restriction: Temporal operators **cannot** be nested.

Explicit Time Constraints: Examples

```
;; if the energy of a rover is below 5, it should be at
;; the recharging location within 10 time units:

(:constraints
  (forall (?r - rover)
    (always-within 10 (< (energy ?r) 5)
                      (at ?r recharging-point)))
```

Semantics of Temporal Preferences

Let $\sigma = \langle (S_0, t_0), \ldots, (S_n, t_n) \rangle$

$$\sigma \models (\texttt{within}\ t\ \phi) \quad \text{iff} \quad \exists i : 0 \leq i \leq n\ S_i \models \phi \text{ and } t_i \leq t$$

$$\sigma \models (\texttt{within}\ t\, \phi\, \psi) \quad \text{iff} \quad \forall i : 0 \leq i \leq n \text{ if } S_i \models \phi \text{ then } \exists j : i \leq j \leq n\ S_j \models \psi \text{ and } t_j - t_i \leq t$$

Precondition Preferences

Precondition Preferences allow discriminating between actions:

```
;; pick an object with the small gripper

(:action pick-with-small-gripper
 :parameters (?obj - object ?loc - location)
 :precondition (and (at robby ?loc) (at ?obj ?loc)
                    (available small-gripper)
                    (preference small (not (large ?obj))))
 :effect (and (not (available small-gripper)) (holding ?obj)))

;; pick an object with the large gripper

(:action pick-with-large-gripper
 :parameters (?obj - object ?loc - location)
 :precondition (and (at robby ?loc) (at ?obj ?loc)
                    (available large-gripper)
                    (preference large (large ?obj)))
 :effect (and (not (available large-gripper)) (holding ?obj)))
```

Comparing two plans in PDDL3

Question: Is plan p_1 at least as preferred as plan p_2?

Answer: First evaluating a **metric function** over the plan.

Comparing two plans in PDDL3

Question: Is plan p_1 at least as preferred as plan p_2?

Answer: First evaluating a **metric function** over the plan.

```
(:constraints
  (and
   (preference break (sometime (at coffee-room)))
   (preference social (sometime (and (at coffee-room)
                                         (coffee-time))))
   (preference reviewing (reviewed paper1))) )

(:metric minimize (+ (* 5 (total-time))
                     (* 4 (is-violated social))
                     (* 2 (is-violated break))
                     (is-violated reviewing)) )
```

Minimizing or Maximizing: two valid options

Answer (cont'd): The answer depends on whether you maximize/minimize.
The metric:

```
(:metric minimize (+ (* 5 (total-time))
                     (* 4 (is-violated social))
                     (* 2 (is-violated break))
                     (is-violated reviewing)) )
```

Can be rewritten as:

```
(:metric maximize (+ (* -5 (total-time))
                     (* -4 (is-violated social))
                     (* -2 (is-violated break))
                     (- (is-violated reviewing))))
```

Semantics of `is-violated`

If σ is a trajectory generated by plan p, and p does not appear in a precondition:

$$(\texttt{is-violated}\ p) = \begin{cases} 1 & \text{if } \sigma \models \texttt{p} \\ 0 & \text{otherwise} \end{cases}$$

If p appears in a precondition

$$(\texttt{is-violated}\ p) = \text{“number of times } \texttt{p} \text{ is violated”}$$

PDDL3 metrics too expressive?

PDDL3 metrics allow expressing unnatural preferences.

Below, the more times you **violate** a preference the **better** the plan gets!

```
(:action pick-with-small-gripper
 :parameters (?obj - object ?loc - location)
 :precondition (and (at robby ?loc) (at ?obj ?loc)
                    (available small-gripper)
                    (preference small (not (large ?obj))))
 :effect ...


(:metric  maximize (is-violated small))
```

In this session...

- Trajectory Constraints in PDDL3
- IPC-5 Planning Competition
- HPLan-P: Compiling Away Temporally Extended Preferences
- MIPS-BDD and MIPS-XXL: Compiling Away TEPs
- Yochan$^{\mathcal{PS}}$: Compiling Away Precondition Preferences
- PDDL3 planning in any cost-sensitive planner

IPC-5

The 2006 International Planning Competition had 3 tracks:

- *Simple Preferences*: Soft-goals and precondition preferences.
- *Qualitative Preferences*: Simple Preferences + Temporally Extended Preferences.
- *Metric Preferences*: Qualitative + temporal preferences.

The winner of all 3 tracks was SGPLAN_5 (Hsu, Wah, Huang, & Chen, 2007). To our knowledge:

- it **ignores** the metric function
- it selects the preferences to achieve at the outset with an unpublished heuristic algorithm.

Planning for PDDL3 preferences?

Existing PDDL3 planners use *compilation* approaches.

Why: PDDL3 is too expressive and existing heuristics do not work immediately with these new elements.

Gain: By compiling away some of the new elements we can use/modify existing heuristics.

We will now review a compilation approach

In this session...

- Trajectory Constraints in PDDL3
- IPC-5 Planning Competition
- HPLan-P: Compiling Away Temporally Extended Preferences
- MIPS-BDD and MIPS-XXL: Compiling Away TEPs
- Yochan$^{\mathcal{PS}}$: Compiling Away Precondition Preferences
- PDDL3 planning in any cost-sensitive planner

HPlan-P's features (Baier, Bacchus, & McIlraith, 2009)

The planner entered the *Qualitative Preferences* track

- Handles discrete domains.
- Does not support durative actions.
  Output: a linear plan

HPlan-P's features (Baier et al., 2009)

The planner entered the *Qualitative Preferences* track

- Handles discrete domains.
- Does not support durative actions.
  Output: a linear plan

Supported PDDL3 features

- *Trajectory* **preferences** (TEPs) and **hard constraints** (THCs)
  We lift a PDDL3 restriction: Planner allows nesting of modalities
- **Precondition** and **goal** preferences

HPlan-P's features (Baier et al., 2009)

The planner entered the *Qualitative Preferences* track

- Handles discrete domains.
- Does not support durative actions.
  Output: a linear plan

Supported PDDL3 features

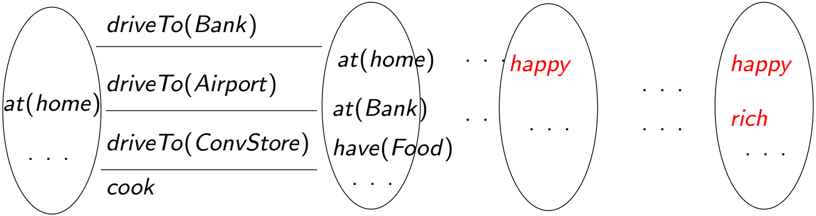
- *Trajectory* **preferences** (TEPs) and **hard constraints** (THCs)
  We lift a PDDL3 restriction: Planner allows nesting of modalities
- **Precondition** and **goal** preferences

Additional feature

- **Incremental**: Produces plans with improving metric value

Heuristic domain-independent planning

- Solve a relaxed planning problem.
- "relaxed = ignore negative effects"
- Expand a relaxed Graphplan planning graph. E.g.



- Obtain a heuristic estimate.

Compiling TPs into the domain

PDDL3
(TPs + THCs)

Compiling TPs into the domain

PDDL3
(TPs + THCs)

$\Rightarrow$

Generate **PNFA** for TPs and THCs

Compiling TPs into the domain

PDDL3
(TPs + THCs)

⇒

Generate **PNFA** for TPs and THCs

⇒

New domain with PNFAs encoded in it

We propose heuristic estimates on this new domain

Compiling TEPs into the domain

Original TEP

```
(forall (?x)
  (sometime-after (loaded ?x)
                  (delivered ?x)))
```

$\Downarrow$

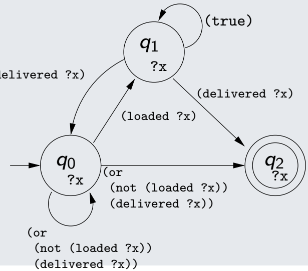
Compiling TEPs into the domain

Original TEP

```
(forall (?x)
  (sometime-after (loaded ?x)
                  (delivered ?x)))
```

⇓

PNFA for the TP



⇒

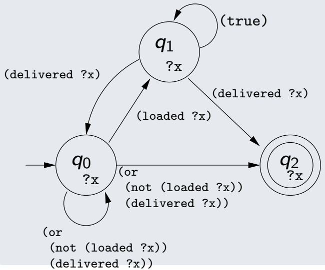
Compiling TEPs into the domain

Original TEP

```
(forall (?x)
  (sometime-after (loaded ?x)
                  (delivered ?x)))
```

⇓

PNFA for the TP



⇒

Final update rule

```
(forall (?x)
  (implies
      (and (aut-state q0 ?x)
           (loaded ?x))
      (add (aut-state q1 ?x))))
```

Heuristic Estimations

We always want to **satisfy** *our goal*

Heuristic Estimations

We always want to **satisfy** *our goal*

Goal distance (G)

A distance-to-the-goals function computed from the expanded *relaxed graph*. Based on a heuristic proposed by (Zhu & Givan, 2005).

Heuristic Estimations (cont.)

try to satisfy *preference goals that are* **highly valued**

Heuristic Estimations (cont.)

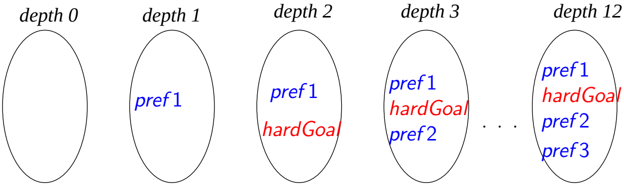
try to satisfy *preference goals that are* **highly valued**

don't want *our search to be "***obsessed***" with prefs that* **look too hard**

Heuristic Estimations (cont.)

try to satisfy *preference goals that are* **highly valued**

don't want *our search to be "***obsessed***" with prefs that* **look too hard**

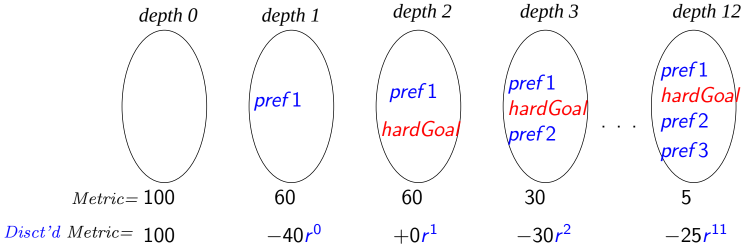


	depth 0	depth 1	depth 2	depth 3	depth 12
$Metric=$	100	60	60	30	5
Δ $Metric=$	100	-40	$+0$	-30	-25

Heuristic Estimations (cont.)

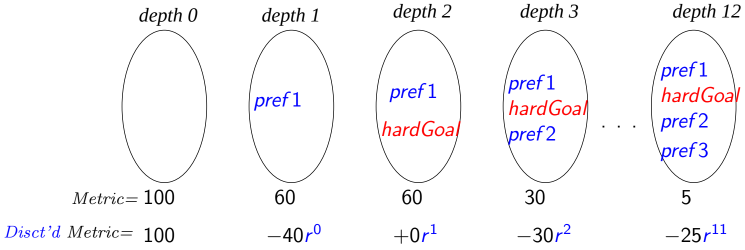
try to satisfy *preference goals that are* **highly valued**

don't want *our search to be "***obsessed***" with prefs that* **look too hard**



try to satisfy *preference goals that are* **highly valued**

don't want *our search to be "***obsessed***" with prefs that* **look too hard**



Discounted Metric ($D(r)$)

$D(r) = M(s) + \sum_{i=0}^{n-1}(M(s_{i+1}) - M(s_i))r^i$, where $s, s_0, \ldots, s_n$ are relaxed states. $r \leq 1$.

Heuristic Estimations (cont.)

try to satisfy *preference goals*

Heuristic Estimations (cont.)

try to satisfy *preference goals*

Preference distance (P)

A distance-to-the-preferences function computed from the expanded *relaxed graph*. Similar to G. Also based on (Zhu & Givan, 2005).

Heuristic Estimations (cont.)

if found plan with metric M, **don't extend** *plans that* **won't reach a value better than** M

Heuristic Estimations (cont.)

if found plan with metric M, **don't extend** *plans that* **won't reach a value better than** M

Optimistic Metric (O)

- *Best metric value* that the *partial plan can achieve* if it becomes a plan
- Computed assuming prefs. that have not been completely violated **will be satisfied**.
- Similar to the optimistic metric in (Bienvenu et al., 2006).

Best Relaxed Metric (B)

- An *estimation* of the best metric value that a partial plan can achieve if it becomes a plan
- Best metric value on the relaxed worlds

hplan-p's Algorithm

Do *best-first* search, where:

- The heuristic is a *prioritization* of the heuristic estimates. Examples:
  - G-$D(0.3)$-O
  - G-B-$D(0.3)$

  G is **always** first

hplan-p's Algorithm

Do *best-first* search, where:

- The heuristic is a *prioritization* of the heuristic estimates. Examples:
  - G-$D(0.3)$-O
  - G-B-$D(0.3)$

  G is **always** first
- If best plan found has metric value M, then prune states whose B value is worse than M.
- Output a plan when its metric is the best found so far.
- Execute until the search space is empty.

The result is a **heuristic**, **incremental** planner for TPs.

Implementation

- PDDL3 Preprocessor:
  - Parses PDDL3
  - Does the TP to automata conversion
  - Generates TLPlan files.
- Modified TLPlan:
  - Compute heuristic estimates using relaxed graphs
  - Handle efficiently the automata updates, and lots of other nice optimizations.

Experimental Evaluation

- We evaluated different strategies on the test domains of IPC-5 (TPP, trucks, openstacks, storage, rovers). 20 problems per domain.
- In particular, we evaluated 44 different strategies
  - G-O
  - G-B
  - G-O-P
  - G-P-O
  - G-B-P
  - G-P-B
  - G-O-M
  - G-M-O
  - G-B-$D(r)$, for $r \in R$.
  - G-$D(r)$-B, for $r \in R$.
  - G-O-$D(r)$, for $r \in R$.
  - G-$D(r)$-O, for $r \in R$.

$R = \{0, 0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1\}$

Summary of Results

- We ran all strategies on all 80 problems for 15 min.

Problem	**Found 1 Plan**	**Found 1+ Plans**	**(Not)Useful heuristics**	**Eff. of Pruning**
openstacks	18	18	**Good:** D-, -D, BP **Bad:** O, OM, MO	Essential
trucks	3	3	**Good:** DO, OD, BP **Bad:** OM, MO	Essential
storage	16	9	Similar Performance BD slightly better,	Important
rovers	11	10	Good DB, DO for small *r*	Not clear
TPP	20	20	**Very Good:** O, **Bad:** all the rest	Important
Overall	67	59	DO(r=0) !!	Very Important

- Worst overall: PO

More details on the usefulness of pruning

The effect of pruning is mixed:

- In the storage and TPP pruning has no effect in practice.
- In rovers O and B responsible for (only) 0.05% average improvement.
- In trucks, B and O are responsible for a 9% and 7% average improvement.
- In openstacks, B is responsible of 12% improvement, while O has no effect.

In this session...

- Trajectory Constraints in PDDL3
- IPC-5 Planning Competition
- HPLan-P: Compiling Away Temporally Extended Preferences
- MIPS-BDD and MIPS-XXL: Compiling Away TEPs
- Yochan$^{\mathcal{PS}}$: Compiling Away Precondition Preferences
- PDDL3 planning in any cost-sensitive planner

MIPS-BDD and MIPS-XXL

Two compilation-based approaches:

MIPS-BDD (Edelkamp, 2006):

- Compile away TEPs via Büchi Automata.
- Use a cost-optimal blind search. States represented as BDDs.

MIPS-XXL (Edelkamp, Jabbar, & Naizih, 2006):

- Compile away TEPs via Büchi Automata.
- Iteratively invoke a version of MIPS-XXL
- Similar to the approach by Feldmann, Brewka, & Wenzel (2006).

In this session...

- Trajectory Constraints in PDDL3
- IPC-5 Planning Competition
- HPLan-P: Compiling Away Temporally Extended Preferences
- MIPS-BDD and MIPS-XXL: Compiling Away TEPs
- Yochan$^{\mathcal{PS}}$: Compiling Away Precondition Preferences
- PDDL3 planning in any cost-sensitive planner

Yochan$^{\mathcal{PS}}$ (Benton *et al.* 2009)

Yochan$^{\mathcal{PS}}$ is another compilation-based approach.

Compilation simple-preferences-PDDL3 (soft-goals + precond. preferences) $\Rightarrow$ PSP problem.

1. $\mathcal{A}$:= actions in the planning task
2. For each action $a \in \mathcal{A}$ with the set $\mathcal{P}$ of formulae in precondition preferences
   i. $\mathcal{A} := (\mathcal{A} \setminus \{a\}) \cup \{a_1, a_2\}$
   ii. a_1 is like a but contains $\mathcal{P}$ as a precondition and cost 0
   iii. a_2 is just like a without preferences and cost c

Where

c = **sum of the costs associated to preferences in $\mathcal{P}$ in** `metric`

Example of *Yochan*$^{\mathcal{PS}}$'s compilation I

```
(:action drive
  :parameters
     (?t - truck ?from ?to - place)
  :precondition (and
                 (at ?t ?from) (connected ?from ?to)
                (preference p-drive (and
                   (ready-to-load goods1 ?from level0)
                   (ready-to-load goods2 ?from level0)
                   (ready-to-load goods3 ?from level0))))
  :effect ... )))
```

A plan metric assigns a weight to our preferences:

```
(:metric (+ (* 10 (is-violated p-drive) )
(* 5 (is-violated POA) )))
```

Example of $Yochan^{\mathcal{PS}}$'s compilation II

```
(:action drive-0
   :parameters
       (?t - truck ?from ?to - place)
   :precondition (and
                   (at ?t ?from) (connected ?from ?to)
                   (ready-to-load goods1 ?from level0)
                   (ready-to-load goods2 ?from level0)
                   (ready-to-load goods3 ?from level0)))
   :effect ...)

(:action drive-1
   :parameters
       (?t - truck ?from ?to - place)
   :cost 10
   :precondition (and (at ?t ?from) (connected ?from ?to))
:effect ...)
```

In this session...

- Trajectory Constraints in PDDL3
- IPC-5 Planning Competition
- HPLan-P: Compiling Away Temporally Extended Preferences
- MIPS-BDD and MIPS-XXL: Compiling Away TEPs
- Yochan$^{\mathcal{PS}}$: Compiling Away Precondition Preferences
- PDDL3 planning in any cost-sensitive planner

In this session...

- Trajectory Constraints in PDDL3
- IPC-5 Planning Competition
- HPLan-P: Compiling Away Temporally Extended Preferences
- MIPS-BDD and MIPS-XXL: Compiling Away TEPs
- Yochan$^{\mathcal{PS}}$: Compiling Away Precondition Preferences
- PDDL3 planning in any cost-sensitive planner

State-of-the-art Planners and PDDL3 Preferences

The compilations techniques we presented can be combined with those presented earlier.

PDDL3
(TEPs + THCs)

State-of-the-art Planners and PDDL3 Preferences

The compilations techniques we presented can be combined with those presented earlier.

PDDL3
(TEPs + THCs)

$\Rightarrow$

Problem with softgoals and conditional costs

State-of-the-art Planners and PDDL3 Preferences

The compilations techniques we presented can be combined with those presented earlier.

PDDL3 (TEPs + THCs)	$\Rightarrow$	Problem with softgoals and conditional costs	$\Rightarrow$	Problem with only hard goals

State-of-the-art Planners and PDDL3 Preferences

The compilations techniques we presented can be combined with those presented earlier.

PDDL3 (TEPs + THCs)	$\Rightarrow$	Problem with softgoals and conditional costs	$\Rightarrow$	Problem with only hard goals

Question: Is this a reasonable approach?
My answer: Not clear.

References I

Baier, J. A., Bacchus, F., & McIlraith, S. A. (2009). A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, *173*(5-6), 593–618.

Bienvenu, M., Fritz, C., & McIlraith, S. (2006). Planning with qualitative temporal preferences. In *Proceedings of the 10th International Conference on Knowledge Representation and Reasoning (KR)*, pp. 134–144, Lake District, England.

Edelkamp, S. (2006). Optimal symbolic PDDL3 planning with MIPS-BDD. In *5th International Planning Competition Booklet (IPC-2006)*, pp. 31–33, Lake District, England.

Edelkamp, S., Jabbar, S., & Naizih, M. (2006). Large-scale optimal PDDL3 planning with MIPS-XXL. In *5th International Planning Competition Booklet (IPC-2006)*, pp. 28–30, Lake District, England.

Feldmann, R., Brewka, G., & Wenzel, S. (2006). Planning with prioritized goals. In *Proceedings of the 10th International Conference on Knowledge Representation and Reasoning (KR)*, pp. 503–514, Lake District, England.

Gerevini, A., Haslum, P., Long, D., Saetti, A., & Dimopoulos, Y. (2009). Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, *173*(5-6), 619–668.

Hsu, C.-W., Wah, B., Huang, R., & Chen, Y. (2007). Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1924–1929, Hyderabad, India.

Zhu, L., & Givan, R. (2005). Simultaneous heuristic search for conjunctive subgoals. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, pp. 1235–1241, Pittsburgh, Pennsylvania, USA.

Qualitative Preference Languages

J. Benton[1], **Jorge Baier**[2], Subbarao Kambhampati[1]

[1]Dept. of Computer Science & Engg.,
Fulton School of Engineering
Arizona State University, Tempe Arizona

[2]Departmento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Santiago, Chile

AAAI-2010 Tutorial on Partial Satisfaction Planning
July 12, 2010

From Quantitative to Qualitative

Both PSP and PDDL3 are quantitative languages.

- Users have to assign **numeric** rewards to soft-goals/preferences.

Arguably it is easier for humans to express preferences in a qualitative way.

- *"I prefer plan where I drink red rather than white wine"*
- *"I'd rather go to the movies than not"*

Preference Aggregation: A challenge

Assuming I have the qualitative preferences:

- *"I prefer plan where I drink red rather than white wine"*
- *"I'd rather go to the movies than not"*

Which of the following plans is better?

- A plan where white wine is ordered and I go to the movies.
- A plan where red wine is ordered and I do not go to the movies.

Preference Aggregation is also a challenge!

Qualitative Preference Languages for Planning

There are a number of qualitative languages that have been used for planning :

- CP-nets (Boutilier, Brafman, Domshlak, Hoos, & Poole, 2004)
- Temporal Preference Framework (Delgrande, Schaub, & Tompits, 2007)
- $\mathcal{PP}$ (Son & Pontelli, 2006)
- $\mathcal{LPP}$ (Bienvenu, Fritz, & McIlraith, 2006)

Qualitative Preference Languages for Planning

There are a number of qualitative languages that have been used for planning :

- CP-nets (Boutilier et al., 2004)
- Temporal Preference Framework (Delgrande et al., 2007)
- $\mathcal{PP}$ (Son & Pontelli, 2006)
- $\mathcal{LPP}$ (Bienvenu et al., 2006)

We discuss two of them in more detail
See (Baier & McIlraith, 2008) for more details.

In this session...

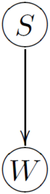
- Planning in TCP-nets Formalisms
  - TCP-net background.
  - Overview of pref-plan.
- Planning with $\mathcal{LPP}$
  - $\mathcal{LPP}$
  - Overview of pplan
- Concluding remarks

In this session...

- Planning in TCP-nets Formalisms
  - TCP-net background.
  - Overview of pref-plan.
- Planning with $\mathcal{LPP}$
  - $\mathcal{LPP}$
  - Overview of pplan
- Concluding remarks

CP-nets (Boutilier et al., 2004)

- CP-nets: compact graphical representations for preferences.
- A CP-net specifies a set of conditional preference statements.



$$\boxed{S_f \succ S_v}$$

S_f	$W_w \succ W_r$
S_v	$W_r \succ W_w$

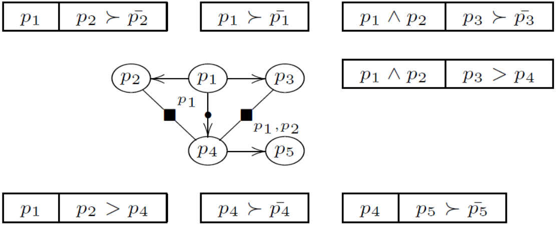
S_f: fish soup
S_v: veggie soup
W_w: white wine
W_r: red wine

"If I'm having fish, I prefer white wine (all other things being equal)"

- Clearly can be used to represent preferences over goal states.

TCP-nets (Brafman, Domshlak, & Shimony, 2006)

- TCP-nets are an extension of CP-nets.
- Allow representing importance between variables [1]



Since p_3 is more important than p_4 when $p_1 \wedge p_2$:

$$p_1 p_2 p_3 \overline{p_4} p_5 \succ p_1 p_2 \overline{p_3} p_4 p_5$$

[1] Diagram from (Brafman & Chernyavsky, 2005)

In this session...

- Planning in TCP-nets Formalisms
  - TCP-net background.
  - Overview of pref-plan.
- Planning with $\mathcal{LPP}$
  - $\mathcal{LPP}$
  - Overview of pplan
- Concluding remarks

PrefPlan (Brafman & Chernyavsky, 2005)

Idea: Try most preferred solutions first

Given a planning problem $\mathcal{P}$, a TCP-net $\mathcal{N}$, a natural n:

- Builds a n-bounded CSP representation of $\mathcal{P}$ (Do & Kambhampati, 2001)
- Solves the CSP with a specific *variable/domain ordering*:
  - Iteratively choose a variable that has no predecessors in the TCP-net.
  - The order of the remaining vars is arbitrary.
  - Choose the value for the variable according to the current assignment and the TCP-net.

PrefPlan's properties

PrefPlan is **sound**, **complete**, and **pareto bounded optimal**.

In this session...

- Planning in TCP-nets Formalisms
  - TCP-net background.
  - Overview of pref-plan.
- Planning with $\mathcal{LPP}$
  - $\mathcal{LPP}$
  - Overview of pplan
- Concluding remarks

Acknowledgements

I thank Meghyn Bienvenu, Christian Fritz and Sheila McIlraith for providing part of the material presented in this session.

A motivating example

The Dinner Example (Bienvenu et al., 2006)
It's dinner time and Claire is tired and hungry. Her goal is to be at home with her hunger sated. There are three possible ways for Claire to get food: cook at home, order take-out, or go to a restaurant.

Claire prefers:

- to eat pizza over spaghetti and spaghetti to crêpes
- takeout to cooking at home (if she has the necessary ingredients) to going out to a restaurant
- cooking to take-out to a restaurant

Preference Formulas

User preferences are represented by a single formula called an Aggregated Preference Formula.

Aggregated Preference Formulae (AgPF) are composed of:

- Basic Desire Formulae (BDF)
- Atomic Preference Formulae (APF)
- General Preference Formulae (GPF)

Basic Desire Formulae (BDF)

Basic Desire Formulae are temporally extended formulas

Similar to PDDL3 preference formulae but adds $occ(a)$ to state action occurrence.

A few example BDFs:

- $(\exists x).\, hasIngrnts(x) \wedge knowsHowToMake(x)$
- $final(kitchenClean)$
- $(\exists x).eventually(occ(cook(x)))$
- $always(\neg((\exists x).occ(eat(x)) \wedge chinese(x)))$

Atomic Preference Formulae (APF)

BDFs establish properties of situations.
APFs express preferences over those properties.

An APF is of the form:

$$\phi_0[v_0] \gg \phi_1[v_1] \gg ... \gg \phi_n[v_n]$$

where:

- the ϕ_i are BDFs representing a set of alternatives
- the v_i are values indicating the level of preference
- the v_i are strictly increasing elements of a totally ordered set $\mathcal{V}$ with bounds v_{min} and v_{max}

Atomic Preference Formulae, cont.

Example APFs:

$$eventually(occ(eat(pizza)))[best] \gg \\ eventually(occ(eat(pasta)))[reallygood] \gg \\ eventually(occ(eat(salad)))[bad]$$

$[best < reallygood < good < okay < bad < reallybad < worst]$

Atomic Preference Formulae, cont.

Example APFs:

$$eventually(occ(eat(pizza)))[best] \gg eventually(occ(eat(pasta)))[reallygood] \gg eventually(occ(eat(salad)))[bad]$$

$$\exists x \exists y.eventually(occ(orderRestaurant(x,y))[best] \gg \exists x \exists y.eventually(occ(orderTakeout(x,y)))[okay] \gg$$

$$[best < reallygood < good < okay < bad < reallybad < worst]$$

General Preference Formulae

BDFs establish properties of situations.
APFs express preferences over those properties.
GPFs provide syntax for combining preferences.

Types of GPFs:

- APFs
- Conditional: $\gamma : \Phi$, where γ is a BDF and Φ a GPF
- Conjunction: $\Phi_1 \& \Phi_2 \& ... \& \Phi_n$
- Disjunction: $\Phi_1 | \Phi_2 | ... | \Phi_n$

Sketching the Semantics of GPFs I

When evaluating $\Phi_1 \& \Phi_2 \& ... \& \Phi_n$ we evaluate each Φ_i and return the worse value.

$$P_1 = eventually(occ(eat(pizza)))[best] \gg \\ eventually(occ(eat(pasta)))[reallygood] \gg \\ eventually(occ(eat(salad)))[bad]$$

$$P_2 = \exists x \exists y.eventually(occ(orderRestaurant(x,y))[best] \gg \\ \exists x \exists y.eventually(occ(orderTakeout(x,y)))[okay] \gg$$

$[best < reallygood < good < okay < bad < reallybad < worst]$

$p_1 =$ "order takeout pasta" $\Rightarrow$ $w_{p_1}(P_1 \& P_2) = okay$

$p_2 =$ "eat pasta at the restaurant" $\Rightarrow$ $w_{p_2}(P_1 \& P_2) = reallygood$

Sketching the Semantics of GPFs I

When evaluating $\Phi_1|\Phi_2|...|\Phi_n$ in a plan, we return the **best** value.

$$P_1 = eventually(occ(eat(pizza)))[best] \gg eventually(occ(eat(pasta)))[reallygood] \gg eventually(occ(eat(salad)))[bad]$$

$$P_2 = \exists x \exists y.eventually(occ(orderRestaurant(x,y))[best] \gg \exists x \exists y.eventually(occ(orderTakeout(x,y)))[okay] \gg$$

$[best < reallygood < good < okay < bad < reallybad < worst]$

$p_1 =$ "order takeout pasta" $\Rightarrow$ $w_{p_1}(P_1|P_2) = reallygood$

$p_2 =$ "eat pasta at the restaurant" $\Rightarrow$ $w_{p_2}(P_1|P_2) = best$

Aggregated Preference Formulae (AgPF)

Aggregated preference formulae the most general class of preference formulae.

Types of AgPFs:

- GPFs
- **lex**$(\Psi_1, ..., \Psi_n)$: lexicographical preference
- **leximin**$(\Psi_1, ..., \Psi_n)$: sorted lexicographical order
- **sum**$(\Psi_1, ..., \Psi_n)$ (for numeric $\mathcal{V}$)

Lexicographical Order

Given:

- Plans p_1 and p_2
- Preference formula: $\mathbf{lex}(\Psi_1, ..., \Psi_n)$

Determine if p_1 is preferred to p_2 by lexicographically comparing

$$(w_{p_1}(\Psi_1), w_{p_1}(\Psi_2), ..., w_{p_1}(\Psi_n))$$

to

$$(w_{p_2}(\Psi_1), w_{p_2}(\Psi_2), ..., w_{p_2}(\Psi_n))$$

In this session...

- Planning in TCP-nets Formalisms
  - TCP-net background.
  - Overview of pref-plan.
- Planning with $\mathcal{LPP}$
  - $\mathcal{LPP}$
  - Overview of pplan
- Concluding remarks

pplan: a planner for $\mathcal{LPP}$ preferences

pplan is an **optimal** planner for $\mathcal{LPP}$ preferences

- Carries out an A* in the space of states.
- Given a partial plan, it uses *progression* (Bacchus & Kabanza, 1998) to evaluate preference formulae.
- Heuristic for s is a vector $(h_o(s), h_p(s))$

$$h_o(s) = \text{"optimistic weight for } s\text{"}$$

"Assumes preferences that still have a chance will be satisfied"

$$h_p(s) = \text{"pessimistic weight for } s\text{"}$$

"Assumes preferences that may be falsified will not be satisfied"

HPLAN-QP

HPLAN-QP (Baier & McIlraith, 2007) is an extension of **hplan-p** for the $\mathcal{LPP}$ language.

It uses **inadmissible** heuristics

Returns a plan faster than **pplan**(and solves more instances)

Non-optimal!

Concluding Remarks

We've seen two qualitative preference languages.

- Both allow representing relative **importance**.
- For TCP-nets plans may be **incomparable**.
- $\mathcal{LPP}$ allows trajectory constrains.

We've briefly described three planners.

- PrefPlan is bounded pareto optimal.
- **pplan** is optimal (unbounded).
- HPlan-QP is incremental (non-optimal).

References I

Bacchus, F., & Kabanza, F. (1998). Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, *22*(1-2), 5–27.

Baier, J. A., & McIlraith, S. A. (2007). On domain-independent heuristics for planning with qualitative preferences. In *7th Workshop on Nonmonotonic Reasoning, Action and Change (NRAC)*.

Baier, J. A., & McIlraith, S. A. (2008). Planning with preferences. *Artificial Intelligence Magazine*, *29*(4), 25–36.

Bienvenu, M., Fritz, C., & McIlraith, S. A. (2006). Planning with qualitative temporal preferences. In *Proceedings of the 10th International Conference on Knowledge Representation and Reasoning (KR)*, pp. 134–144.

Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., & Poole, D. (2004). CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, *21*, 135–191.

Brafman, R., & Chernyavsky, Y. (2005). Planning with goal preferences and constraints. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 182–191, Monterey, CA.

Brafman, R. I., Domshlak, C., & Shimony, S. E. (2006). On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, *25*, 389–424.

Delgrande, J. P., Schaub, T., & Tompits, H. (2007). A general framework for expressing preferences in causal reasoning and planning. *Journal of Logic and Computation*, *17*, 871–907.

Do, M. B., & Kambhampati, S. (2001). Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence*, *132*(2), 151–182.

Son, T. C., & Pontelli, E. (2006). Planning with preferences using logic programming. *Theory and Practice of Logic Programming*, *6*(5), 559–607.

Preferences and HTNs

J. Benton[1], **Jorge Baier**[2], Subbarao Kambhampati[1]

[1]Dept. of Computer Science & Engg.,
Fulton School of Engineering
Arizona State University, Tempe Arizona

[2]Departmento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Santiago, Chile

AAAI-2010 Tutorial on Partial Satisfaction Planning
July 12, 2010

In this session...

- Background: HTN planning
- HTN-specific preferences
- Summary

In this session...

- Background: HTN planning
- HTN-specific preferences
- Summary

Example: HTN Planning

Planning Task: Make my travel arrangements

An HTN specifies **how** the task is achieved:

$$\underbrace{\textit{Book my transportation}}_{\text{task}} \text{ and } \underbrace{\textit{book my accommodation}}_{\text{task}}$$

Example: HTN Planning

Planning Task: Make my travel arrangements

An HTN specifies **how** the task is achieved:

$$\underbrace{\textit{Book my transportation}}_{\text{task}} \text{ and } \underbrace{\textit{book my accommodation}}_{\text{task}}$$

To *book transportation*, either:

- go to a travel agency, find a flight, book the flight, pay,
- go online, find a flight, book and pay for the flight
- go online, find a car, book and pay for the car

To *book accommodation*:

- go online, find a hotel, book and pay for the hotel

HTNs: Instances and Plans

Definition (HTN Planning Problem)

An HTN instance is a 3-tuple $\mathcal{P} = (s_0, D, w_0)$ where:

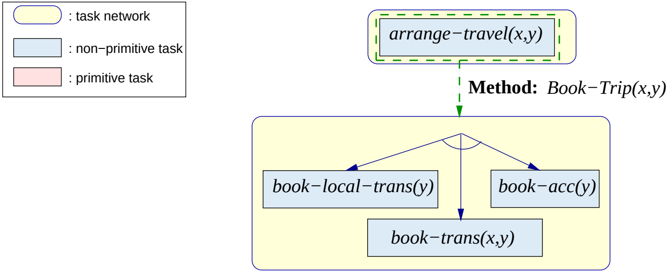
- s_0 is the initial state,
- D is the HTN (deterministic) planning domain.
- w_0 is a task network called the *initial task network*.

Definition (Plan)

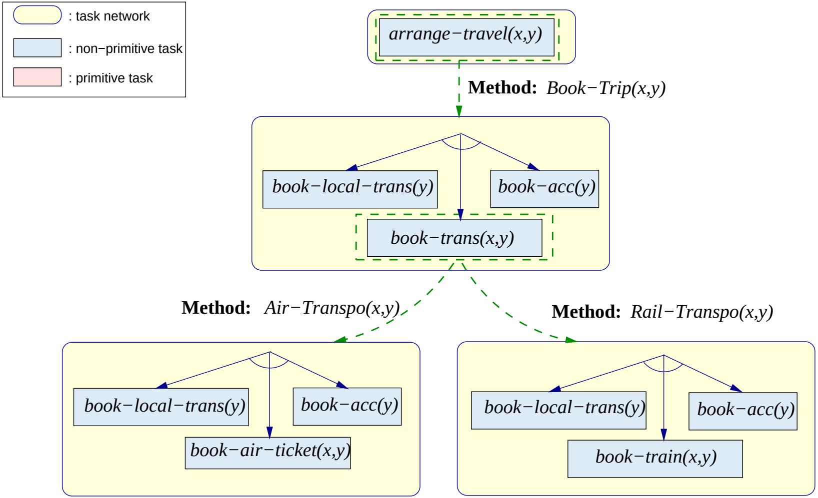
$\pi = o_1 o_2 \cdots o_k$ is a **plan** for HTN instance $\mathcal{P} = (s_0, D, w_0)$ if there is a **primitive decomposition**, w, of w_0 of which π is an **instance**.

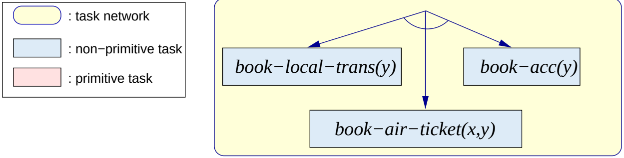
Example: Travel Arrangements in HTN

Example: Travel Arrangements in HTN

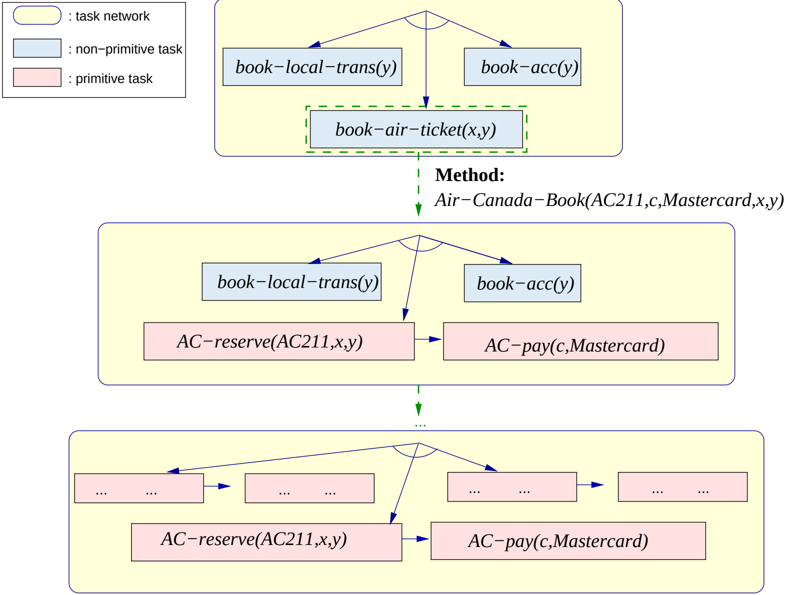


Example: Travel Arrangements in HTN




: task network
: non−primitive task
: primitive task
book−local−trans(y)
book−acc(y)
book−air−ticket(x,y)

...



: task network
: non−primitive task
: primitive task
book−local−trans(y)
book−acc(y)
book−air−ticket(x,y)
Method:
Air−Canada−Book(AC211,c,Mastercard,x,y)
book−local−trans(y)
book−acc(y)
AC−reserve(AC211,x,y)
AC−pay(c,Mastercard)
...
... ...
... ...
... ...
... ...
AC−reserve(AC211,x,y)
AC−pay(c,Mastercard)

HTN & HTN Preference-based Planning

HTN Planning

Given:

- Initial state, set of tasks, domain description

Objective:

- find *any* plan

HTN & HTN Preference-based Planning

HTN Planning

Given:

- Initial state, set of tasks, domain description

Objective:

- find *any* plan

HTN Preference-Based Planning (PBP)

Given:

- Initial state, set of tasks, domain description
- **preferences** that define the **plan's quality**

Objective:

- find a plan that **optimizes** quality

Examples of HTN User Preferences

Planning Task: Make my travel arrangements

An HTN **specifies a set of plans** for the task:

Book my transportation and *book my accommodation*

Examples of HTN User Preferences

Planning Task: Make my travel arrangements

An HTN **specifies a set of plans** for the task:

Book my transportation and *book my accommodation*

We **add preferences**. E.g.:

- I prefer to book my flight after my hotel reservation is confirmed.
- If my return flight departs before 9am, then I prefer to stay in a hotel located at the airport the night before departure.
- I prefer to stay at the conference hotel.
- I prefer to spend \$100/night or less on my hotel room.

In this session...

- Background: HTN planning
- HTN-specific preferences
- Summary

HTN-specific preferences

HTN-specific preferences:

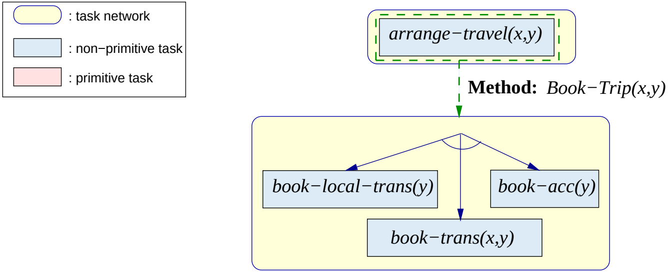
how the user prefers to **decompose** the HTN.

E.g.

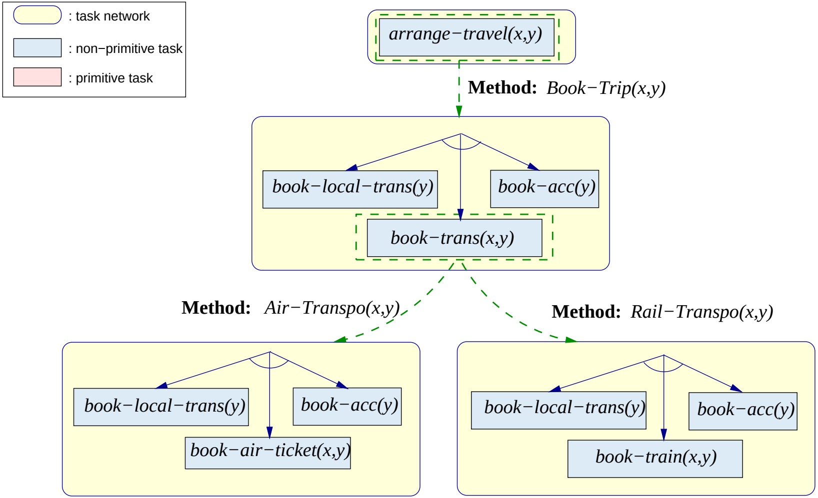
- *I prefer to pay with MasterCard for transportation and Visa for accommodation*
- *I prefer Rail transportation when travel distance is less than 200Km*

Example: Travel Arrangements in HTN

Example: Travel Arrangements in HTN



Example: Travel Arrangements in HTN



HTN Planners with preferences

HTN Planners with Preferences:

- SHOP2 (Nau et al., 2003)
- Advice for decomposing HTNs (Myers, 2000) (HTN-specific)
- HTNPlan (Sohrabi & McIlraith, 2008) (HTN-specific)
- HTNPlan-P (Sohrabi et al., 2009) (HTN-specific)
- SCUP (Lin, Kuter, & Sirin, 2008)

We are going to focus on approaches with HTN-specific preferences

Advice for HTNs (Myers, 2000) I

Role Advice

Template:

$\langle$Use/Don't Use$\rangle$ $\langle$*object*$\rangle$ in $\langle$*role*$\rangle$ for $\langle$*context* $-$ *activity*$\rangle$.

E.g.:

- Stay in 3-star ensuite hotels while vacationing in Scotland
- Layovers longer than 90 minutes are not desired for domestic flights.

Advice for HTNs (Myers, 2000) II

Method Advice

Template: $\langle$Use/Don't use$\rangle$ $\langle method - activity \rangle$ for $\langle context - activity \rangle$

E.g.:

- Find a package bike tour starting in Athens for the vacation in Greece
- Don't fly between cities less than 200 miles apart

The task (Myers, 2000)

Given:

- A planning problem specified as an (sort of) HTN.
- A set A of advices.

Task:
Find a plan that maximally satisfies a set of advices $A' \subseteq A$

Observation: Obviously impractical to try all $2^{|A|}$ subsets.

Two greedy approaches (Myers, 2000)

MILAV

At each decision point choose an option that violates the least number of advices.
Observation: Not even local minimum is guaranteed.

Local Search

Given a plan that satisfies the set A' of advices try to find a plan for $A' \cup \{a\}$, for some $a \in A$. Start again if successful.
Observation: Local minimum is guaranteed.

HTNPLan-P (Sohrabi et al., 2009): Preference Language

HTNPLan-P's preference language:

- Written in PDDL syntax.
- Preferences are **independent** of the HTN problem.

PDDL3 is **extended** with:

- **occ**(a): "primitive action a occurs"
- **initiate**(u): "initiate task/method u"
- **terminate**(u): "terminate task/method u"

Example preferences

1. If origin is close to destination, I prefer the train

```
(imply (close origin dest)
       (sometime (initiate Rail-Transpo)))
```

2. I prefer direct economy window-seated flight with a Star Alliance (SA) carrier

```
(sometime (occ (book-flight SA Eco Direct WindowSeat)))
```

3. I prefer not to pay with my MasterCard

```
(always (not (occ (pay MasterCard))))
```

4. I prefer booking accommodations after transportation

```
(sometime-after (terminate arrange-trans)
                (initiate arrange-acc))
```

A Preference-Based HTN Planner

Two-Step Approach:

1. **Preprocess** the original problem into PBP HTN problem with **final-state preferences only**.
2. **Plan** on preprocessed instance.

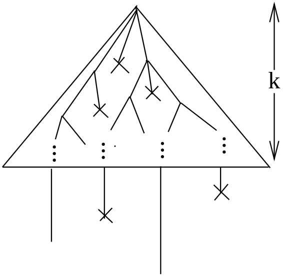
Highlights this HTN PBP algorithm:

- Returns a *sequence* of plans with increasing quality.
- Best-first search w/**inadmissible heuristics** for fast planning.
- Branch-and-bound pruning.

Heuristic Functions

Depth (D), Optimistic Metric (OM), Pessimistic Metric (PM), Look-Ahead Metric (LA)

- *Optimistic Metric (OM)*: is an admissible heuristic used for pruning.
- *Look-Ahead Metric (LA)*,
  1. Solves the current node up to a certain **depth**.
  2. Computes a single primitive decomposition for each of the resulting nodes.
  3. Returns the best metric value among all the fully decomposed nodes.



Sound Pruning and Optimality

Theorem (Sound Pruning)

[Informally] If the metric function is non-decreasing in the number of satisfied preferences then the OM metric never prunes a node from the search space that could lead to a plan that is better than the one we already found.

Theorem (Optimality)

If the algorithm provides sounds pruning, and it stops, the last plan returned (if any) is optimal.

In this session...

- Background: HTN planning
- HTN-specific preferences
- Summary

Concluding Remarks

- HTN is one of the most widely used planning formalisms in industry.
- Extensions and algorithms exist for incorporating preferences.
- Algorithms use state-of-the-art techniques.
- Interestingly however, many authors have shown how to translate (restricted) HTN's into PDDL (Lekavý & Návrat, 2007; Fritz et al., 2008; Alford et al., 2009).

References I

Alford, R., Kuter, U., & Nau, D. S. (2009). Translating htns to pddl: A small amount of domain knowledge can go a long way. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1629–1634.

Fritz, C., Baier, J. A., & McIlraith, S. A. (2008). ConGolog, sin Trans: Compiling ConGolog into basic action theories for planning and beyond. In *Proceedings of the 11th International Conference on Knowledge Representation and Reasoning (KR)*, pp. 600–610, Sydney, Australia.

Lekavý, M., & Návrat, P. (2007). Expressivity of STRIPS-like and HTN-like planning. In *Proceedings of Agent and Multi-Agent Systems: Technologies and Applications, First KES International Symposium (KES-AMSTA)*, pp. 121–130, Wroclaw, Poland.

Lin, N., Kuter, U., & Sirin, E. (2008). Web service composition with user preferences. In *Proceedings of the 5th European Semantic Web Conference (ESWC)*, pp. 629–643.

Myers, K. L. (2000). Planning with conflicting advice. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, pp. 355–362.

Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, *20*, 379–404.

Sohrabi, S., Baier, J., & McIlraith, S. A. (2009). Htn planning with preferences. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, Pasadena, California.

Sohrabi, S., & McIlraith, S. A. (2008). On planning with preferences in HTN. In *Fourth Multidisciplinary Workshop on Advances in Preference Handling (M-Pref)*, pp. 103–109.