Planning Graph Based Reachability Heuristics

Daniel Bryce & Subbarao Kambhampati
IJCAI’07 Tutorial 12
January 8, 2007

http://rakaposhi.eas.asu.edu/pg-tutorial/
dan.bryce@asu.edu
rao@asu.edu

Also See Our Tutorial Article in the Winter Issue of AI Magazine

Scalability of Planning

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

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

- Ways to improve Planner Scalability
  - Problem Formulation
  - Search Space
  - Reachability Heuristics
    - Domain (Formulation) Independent
    - Work for many search spaces
    - Flexible – work with most domain features
    - Overall complement other scalability techniques
    - Effective!!

Topics

- Classical Planning
- Cost Based Planning
- Partial Satisfaction Planning
- Non-Deterministic/Probabilistic Planning
- Resources (Continuous Quantities)
- Temporal Planning
- Wrap-up
Classical Planning

Rover Domain

```
(define (domain rovers_classical)
  (:requirements :strips :typing)
  (:types waypoint data)
  (:predicates
    (at ?x - waypoint)
    (avail ?d - data ?x - waypoint)
    (comm ?d - data)
    (have ?d - data))
  (:action drive
    :parameters (!x !y - waypoint)
    :precondition (at ?x)
    :effect (and (at !y) (not (at ?x))))
  (:action commun
    :parameters (?)data
    :precondition (have ?d)
    :effect (comm ?d))
  (:action sample
    :parameters (?)data ?x - waypoint
    :precondition (and (at ?x) (avail ?d ?x))
    :effect (have ?d))
)
```

```
(define (problem rovers_classical1)
  (:domain rovers_classical)
  (:objects
    soil image rock - data
    alpha beta gamma - waypoint)
  (:init (at alpha)
    (avail soil alpha)
    (avail rock beta)
    (avail image gamma))
  (:goal (and (comm soil)
              (comm image)
              (comm rock)))
)
```
Classical Planning

- Relaxed Reachability Analysis
- Types of Heuristics
  - Level-based
  - Relaxed Plans
- Mutexes
- Heuristic Search
  - Progression
  - Regression
  - Plan Space
- Exploiting Heuristics

Planning Graph and Search Tree

- Envelope of Progression Tree (Relaxed Progression)
  - Proposition lists: Union of states at kth level
  - Lowerbound reachability information
Level Based Heuristics

- The distance of a proposition is the index of the first proposition layer in which it appears
  - Proposition distance changes when we propagate cost functions – described later
- What is the distance of a Set of propositions??
  - Set-Level: Index of first proposition layer where all goal propositions appear
    - Admissible
    - Gets better with mutexes, otherwise same as max
  - Max: Maximum distance proposition
  - Sum: Summation of proposition distances

Example of Level Based Heuristics

```
set-level(s₁, G) = 3
max(s₁, G) = max(2, 3, 3) = 3
sum(s₁, G) = 2 + 3 + 3 = 8
```
Distance of a Set of Literals

\[ h(S) = \sum_{p \in S} \text{lev}(\{p\}) \]

- \( \text{lev}(p) \): index of the first level at which \( p \) comes into the planning graph
- \( \text{lev}(S) \): index of the first level where all props in \( S \) appear non-mutexed.
  - If there is no such level, then
    - If the graph is grown to level off, then \( \infty \)
    - Else \( k+1 \) (\( k \) is the current length of the graph)

How do Level-Based Heuristics Break?

The goal \( g \) is reached at level 2, but requires 101 actions to support it.
Relaxed Plan Heuristics

- When Level does not reflect distance well, we can find a relaxed plan.
- A relaxed plan is subgraph of the planning graph, where:
  - Every goal proposition is in the relaxed plan at the last level
  - Every proposition in the relaxed plan has a supporting action in the relaxed plan
  - Every action in the relaxed plan has its preconditions supported.
- Relaxed Plans are not admissible, but are generally effective.
- Finding the optimal relaxed plan is NP-hard, but finding a greedy one is easy. Later we will see how “greedy” can change.
  - Later, in over-subscription planning, we’ll see that the effort of finding an optimal relaxed plan is worthwhile.

Example of Relaxed Plan Heuristic

Support Goal Propositions Individually
Count Actions
$\text{RP}(s_i, G) = 8$
Identify Goal Propositions
Results

Optimizations in Heuristic Computation

- Taming Space/Time costs
  - Bi-level Planning Graph representation
  - Partial expansion of the PG (stop before level-off)
    - It is FINE to cut corners when using PG for heuristics (instead of search)!!
  - Branching factor can still be quite high
    - Use actions appearing in the PG (complete)
      - Select actions in lev(S) vs Levels-off (incomplete)
      - Consider action appearing in RP (incomplete)
Adjusting for Negative Interactions

- Until now we assumed actions only positively interact. What about negative interactions?
- Mutexes help us capture some negative interactions
  - Types
    - Actions: Interference/Competing Needs
    - Propositions: Inconsistent Support
  - Binary are the most common and practical
  - |A| + 2|P|-ary will allow us to solve the planning problem with a backtrack-free GraphPlan search
    - An action layer may have |A| actions and 2|P| noops
  - Serial Planning Graph assumes all non-noop actions are mutex

Binary Mutexes

- Set-Level(sI, {at(B), have(soil)}) = 2
- Max(sI, {at(B), have(soil)}) = 1
- have(soil) only supporter is mutex with at(B) only supporter
  --Inconsistent Support--
- have(soil) has a supporter not mutex with a supporter of at(B),
  --feasible together at level 2--
- sample needs at(α), drive negates at(α)
  --Interference--
Adjusting the Relaxed Plans

- Start with RP heuristic and adjust it to take subgoal interactions into account
  - Negative interactions in terms of “degree of interaction”
  - Positive interactions in terms of co-achievement links
    - Ignore negative interactions when accounting for positive interactions (and *vice versa*)
- It is NP-hard to find a plan (when there are mutexes), and even NP-hard to find an optimal relaxed plan.
  - It is easier to add a penalty to the heuristic for ignored mutexes

\[ H_{AdjSum2M}(S) = length(\text{RelaxedPlan}(S)) + \max_{p,q \in S} \delta(p,q) \]
where \( \delta(p,q) = lev\{p,q\} - \max\{lev(p), lev(q)\} \)
/*Degree of –ve Interaction */

### Anatomy of a State-space Regression planner

- **Problem Specification** (Initial and Goal State)
- **Regression Planner** (based on HSP-R)
- **Heuristic**
  - Actions in the Last Level
  - Extraction of Heuristics
  - Graphplan Graph
  - Extension Phase (based on STAN)
- **Plan**

**Problem:** Given a set of subgoals (regressed state) estimate how far they are from the initial state

[AAAI 2000; AIPS 2000; AIJ 2002; JAIR 2003]
Rover Example in Regression

\[ \text{Should be } \infty, s_4 \text{ is inconsistent, how do we improve the heuristic??} \]

\[ \text{Sum}(s_1, s_4) = 0 + 0 + 1 + 1 + 2 = 4 \]

\[ \text{Sum}(s_1, s_3) = 0 + 1 + 2 + 2 = 5 \]

AltAlt Performance

Level-based Adjusted RP

Logistics

Scheduling

Problem sets from IPC 2000
Then it was cruelly UnPOPped

The good times return with Re(vived)POP

In the beginning it was all POP.

January 18, 2007
IJCAI'07 Tutorial T12

Plan Space Search

POP Algorithm

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

Choice points
- Flaw selection (open condition? unsafe link? Non-backtrack choice)
- Flaw resolution/Plan Selection (how to select (rank) partial plan?)
Distance heuristics to estimate cost of partially ordered plans (and to select flaws)

If we ignore negative interactions, then the set of open conditions can be seen as a regression state

Mutexes used to detect indirect conflicts in partial plans

A step threatens a link if there is a mutex between the link condition and the steps’ effect or precondition

Post disjunctive precedences and use propagation to simplify

Regression and Plan Space
RePOP’s Performance

- RePOP implemented on top of UCPOP
  * Dramatically better than any other partial order planner before it
  * Competitive with Graphplan and AltAlt
  * VHPOP carried the torch at ICP 2002

<table>
<thead>
<tr>
<th>Problem</th>
<th>UCPOP</th>
<th>RePOP</th>
<th>Graphplan</th>
<th>AltAlt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gripper-8</td>
<td>-</td>
<td>1.01</td>
<td>66.82</td>
<td>.43</td>
</tr>
<tr>
<td>Gripper-10</td>
<td>-</td>
<td>2.72</td>
<td>47min</td>
<td>1.15</td>
</tr>
<tr>
<td>Gripper-20</td>
<td>-</td>
<td>81.86</td>
<td>-</td>
<td>15.42</td>
</tr>
<tr>
<td>Rocket-a</td>
<td>-</td>
<td>8.36</td>
<td>75.12</td>
<td>1.02</td>
</tr>
<tr>
<td>Rocket-b</td>
<td>-</td>
<td>8.17</td>
<td>77.48</td>
<td>1.29</td>
</tr>
<tr>
<td>Logistics-a</td>
<td>-</td>
<td>3.16</td>
<td>306.12</td>
<td>1.59</td>
</tr>
<tr>
<td>Logistics-b</td>
<td>-</td>
<td>2.31</td>
<td>262.64</td>
<td>1.18</td>
</tr>
<tr>
<td>Logistics-c</td>
<td>-</td>
<td>22.54</td>
<td>-</td>
<td>4.52</td>
</tr>
<tr>
<td>Logistics-d</td>
<td>-</td>
<td>91.53</td>
<td>-</td>
<td>20.62</td>
</tr>
<tr>
<td>Bw-large-a</td>
<td>45.78</td>
<td>(5.23)</td>
<td>14.67</td>
<td>4.12</td>
</tr>
<tr>
<td>Bw-large-b</td>
<td>-</td>
<td>(18.86)</td>
<td>122.56</td>
<td>14.14</td>
</tr>
<tr>
<td>Bw-large-c</td>
<td>-</td>
<td>(137.84)</td>
<td>-</td>
<td>116.34</td>
</tr>
</tbody>
</table>

Written in Lisp, runs on Linux, 500MHz, 250MB

You see, pop, it is possible to Re-use all the old POP work!

[IJCAI, 2001]

Exploiting Planning Graphs

- Restricting Action Choice
  - Use actions from:
    * Last level before level off (complete)
    * Last level before goals (incomplete)
    * First Level of Relaxed Plan (incomplete) – FF’s helpful actions
    * Only action sequences in the relaxed plan (incomplete) – YAHSP

- Reducing State Representation
  - Remove static propositions. A static proposition is only ever true or false in the last proposition layer.
Classical Planning Conclusions

- Many Heuristics
  - Set-Level, Max, Sum, Relaxed Plans
- Heuristics can be improved by adjustments
  - Mutexes
- Useful for many types of search
  - Progresssion, Regression, POCL

Cost-Based Planning
Cost-based Planning

- Propagating Cost Functions
- Cost-based Heuristics
  - Generalized Level-based heuristics
  - Relaxed Plan heuristics

Rover Cost Model
Cost Propagation

\begin{align*}
\text{avail}(\text{soil}, \alpha) & \quad 20 \\
\text{sample}(\text{soil}, \alpha) & \quad \text{drive}(\alpha, \beta) \\
\text{avail}(\text{rock}, \beta) & \quad 10 \\
\text{drive}(\alpha, \beta) & \quad \text{drive}(\alpha, \gamma) \\
\text{avail}(\text{image}, \gamma) & \quad 30 \\
\text{drive}(\alpha, \gamma) & \quad \text{at}(\alpha) \\
\text{at}(\alpha) & \quad 20 \\
\text{have}(\text{soil}) & \quad 10 \\
\text{at}(\beta) & \quad 30 \\
\text{at}(\gamma) & \quad 30 \\
\end{align*}

Cost Reduces because Of different supporter At a later level

1-lookahead

Cost Propagation (cont’d)
Terminating Cost Propagation

- Stop when:
  - goals are reached (no-lookahead)
  - costs stop changing ($\infty$-lookahead)
  - k levels after goals are reached (k-lookahead)

Guiding Relaxed Plans with Costs

Start Extract at last level (goal proposition is cheapest)
Cost-Based Planning Conclusions

- Cost-Functions:
  - Remove false assumption that level is correlated with cost
  - Improve planning with non-uniform cost actions
  - Are cheap to compute (constant overhead)

Partial Satisfaction (Over-Subscription) Planning
Partial Satisfaction Planning

- Selecting Goal Sets
  - Estimating goal benefit
- Anytime goal set selection
- Adjusting for negative interactions between goals

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

Partial Satisfaction (Oversubscription) Planning

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

Example: Rover Mission Planning (MER)

Need automated support for Over-subscription/Partial Satisfaction Planning
Adapting PG heuristics for PSP

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

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

---

Goal Set Selection In Rover Problem

<table>
<thead>
<tr>
<th>comm(soil)</th>
<th>comm(rock)</th>
<th>comm(image)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>110</td>
<td>65</td>
<td>45</td>
</tr>
<tr>
<td>70</td>
<td>60</td>
<td>20</td>
</tr>
<tr>
<td>130</td>
<td>100</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>comm(soil)</th>
<th>comm(rock)</th>
<th>comm(image)</th>
<th>Cost</th>
<th>Utility</th>
<th>Net Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>20</td>
<td></td>
<td>0</td>
<td>35</td>
<td>-15</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td></td>
<td>0</td>
<td>40</td>
<td>-15</td>
</tr>
<tr>
<td>65</td>
<td>50</td>
<td></td>
<td>0</td>
<td>65</td>
<td>15</td>
</tr>
<tr>
<td>25</td>
<td>70</td>
<td></td>
<td>0</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>60</td>
<td>80</td>
<td></td>
<td>0</td>
<td>60</td>
<td>20</td>
</tr>
<tr>
<td>65</td>
<td>110</td>
<td></td>
<td>0</td>
<td>65</td>
<td>45</td>
</tr>
<tr>
<td>90</td>
<td>130</td>
<td></td>
<td>0</td>
<td>90</td>
<td>40</td>
</tr>
</tbody>
</table>

- Found By RP
- Found By Cost Propagation
- Found By Biased RP
SAPAPPS (anytime goal selection)

- A* Progression search
  - g-value: net-benefit of plan so far
  - h-value: relaxed plan estimate of best goal set
    - Relaxed plan found for all goals
    - Iterative goal removal, until net benefit does not increase
  - Returns plans with increasing g-values.

Some Empirical Results for AltAltp

Exact algorithms based on MDPs don’t scale at all

[AAAI 2004]
Adjusting for Negative Interactions (AltWlt)

- **Problem:**
  - What if the apriori goal set is not achievable because of negative interactions?
  - What if greedy algorithm gets bad local optimum?

- **Solution:**
  - Do not consider mutex goals
  - Add penalty for goals whose relaxed plan has mutexes.
    - Use interaction factor to adjust cost, similar to adjusted sum heuristic
  - \[ \max_{g_1, g_2 \in G} \{ \text{lev}(g_1, g_2) - \max(\text{lev}(g_1), \text{lev}(g_2)) \} \]
  - Find Best Goal set for each goal

---

**Goal Utility Dependencies**

**BAD**
- One Shoe
  - Cost: 50
  - Utility: 0
  - High-res photo utility: 150
  - Soil sample utility: 100
  - Both goals: Additional 200

**GOOD**
- Two Shoes
  - Cost: 100
  - Utility: 500
  - High-res photo utility: 150
  - Low-res photo utility: 100
  - Both goals: Remove 80
Dependencies

goal interactions exist as two distinct types

cost dependencies

Goals share actions in the plan trajectory
Defined by the plan

utility dependencies

Goals may interact in the utility they give
Explicitly defined by user

Exists in classical planning, but
is a larger issue in PSP

All PSP Planners
AltWlt, Optiplan, Sapa$_{PS}$

No need to consider this in
classical planning

SPUDS

Our planner based on Sapa$_{PS}$

Defining Goal Dependencies

(has-left-shoe)
(has-right-shoe)
+500

(has-image high-res loc1)
150

(has-image low-res loc1)
100

(has-soil loc1)
100

(has-image high-res loc1)
(has-soil loc1)
+200

(has-image high-res loc1)
(has-image low-res loc1)
-80
**SapaPS**

Example of a relaxed plan (RP) in a rovers domain

<table>
<thead>
<tr>
<th>a1: Move(l1,l2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a2: Calibrate(camera)</td>
</tr>
<tr>
<td>a3: Sample(l2)</td>
</tr>
<tr>
<td>a4: Take_high_res(l2)</td>
</tr>
<tr>
<td>a5: Take_low_res(l2)</td>
</tr>
<tr>
<td>a6: Sample(l1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>g1: Sample(l1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>g2: Sample(l2)</td>
</tr>
<tr>
<td>g3: HighRes(l2)</td>
</tr>
<tr>
<td>g4: lowRes(l2)</td>
</tr>
</tbody>
</table>

**Anytime Forward Search**

- Propagate costs on relaxed planning graph
- Using the RP as a tree lets us remove supporting actions as we remove goals
- We remove goals whose relaxed cost outweighs utility

**Example of a relaxed plan (RP) in a rovers domain**

<table>
<thead>
<tr>
<th>g1: Sample(l1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>g2: Sample(l2)</td>
</tr>
<tr>
<td>g3: HighRes(l2)</td>
</tr>
<tr>
<td>g4: lowRes(l2)</td>
</tr>
</tbody>
</table>

In SapaPS goals are removed independently (and in pairs)
If cost > utility, the goal is removed.
...but now we have more complicated goal interactions.

**New challenge:**
- Goals no longer independent
- How do we select the best set of goals?

**Idea:**
- Use declarative programming

January 18, 2007  IJCAI’07 Tutorial T12
Heuristically Identifying Goals

We can encode all goal sets and actions

We encode the relaxed plan

Binary Variables

| goal utility dependencies | \( \forall G' \subseteq G, \ f_u(G') \neq 0 : X_{G'} \)
| goals | \( \forall g \in G : X_g \)
| actions | \( \forall a \in P : X_a \)

Constraints

- Select actions for goals: \( \forall a \in P, \forall g \in GS(a) : (1 - X_g) + X_a \geq 1 \)
- Select goals in dependency: \( \sum_{g \in G} (1 - X_g) + X_{G'} \geq 1 \)
- Select goals in dependency: \( \forall g \in G' : (1 - X_{G'}) + X_g \geq 1 \)

Objective function: \( \text{maximize} \ (\sum f_u(G') * X_{G'} - \sum X_a * c_a) \)

where \( f_u(G') \) gives the utility given by the goal set \( G' \), \( c_a \) cost of \( a \) and \( GS(a) \) gives all goals action \( a \) is involved in reaching
Heuristically Identifying Goals

We can now remove combinations of goals that give inoptimal relaxed solutions.

We are finding optimal goals given the (inadmissible) relaxed plan.

Results

ZenoTravel

January 18, 2007 IJCAI'07 Tutorial T12

January 18, 2007 IJCAI'07 Tutorial T12
Results

Satellite

Only in this instance does SapaPS do slightly better

SPUDS
SapaPS

problems

benefit

PSP Conclusions

- Goal Set Selection
  - Apriori for Regression Search
  - Anytime for Progression Search
  - Both types of search use greedy goal insertion/removal to optimize net-benefit of relaxed plans

- Orienteering Problem
  - Interactions between goals apparent in OP
  - Use solution to OP as heuristic
  - Planning Graphs help define OP
Non-Deterministic Planning

- Belief State Distance
- Multiple Planning Graphs
- Labelled Uncertainty Graph
- Implicit Belief states and the CFF heuristic
Conformant Rover Problem

\[
\begin{align*}
\text{(define (domain rovers-conformant)} \\
\text{(:requirements :strips :typing)} \\
\text{(:types waypoint data)} \\
\text{(:predicates} \\
\text{ (at ?x - waypoint)} \\
\text{ (avail ?d - data ?x - waypoint) } \\
\text{ (comm ?d - data) } \\
\text{ (have ?d - data) )} \\
\text{(:action drive} \\
\text{ :parameters (?x ?y - waypoint) } \\
\text{ :precondition (at ?x) } \\
\text{ :effect (and (at ?y) (not (at ?x))) )} \\
\text{(:action common} \\
\text{ :parameters (?d - data) } \\
\text{ :precondition (have ?d) } \\
\text{ :effect (comm ?d)) )} \\
\text{(:action sample} \\
\text{ :parameters (?d - data ?x - waypoint) } \\
\text{ :precondition (at ?x) } \\
\text{ :effect (when (avail ?d ?x) (have ?d))) )}
\end{align*}
\]

Search in Belief State Space
Belief State Distance

Compute Classical Planning Distance Measures
Assume can reach closest goal state
Aggregate State Distances

Max = 15, 20
Sum = 20, 20
Union = 17, 20

P1
P2
P3

Belief State Distance

Estimate Plans for each state pair
Capture Positive Interaction & Independence

Union = 17

[ICAPS 2004]
State Distance Aggregations [Bryce et al., 2005]

Figure 16: Ratio of heuristic estimates for distance between BSF and BSF to optimal plan length. RV = Rovers, L = Logistics, B = BT, BC = BTC, C = Cube Center, R = Ring.

Multiple Planning Graphs

Extract Relaxed Plans from each
Step-wise union relaxed plans

Build A planning Graph
For each State in the belief state

h = 7
Labelled Uncertainty Graph

- Action labels are the Conjunction (intersection) of their Precondition labels.
- Effect labels are the Disjunction (union) of supporter labels.
- Stop when goal is Labeled with every state.
- Subgoals and Supporters need not be used for Every state where they are reached (labeled).

Labelled Relaxed Plan

- Subgoals and Supporters need not be used for Every state where they are reached (labeled).
- Must Pick Enough Supporters to cover the (sub)goals.
Comparison of Planning Graph Types

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

State Agnostic Planning Graphs (SAG)

- LUG represents multiple explicit planning graphs
- SAG uses LUG to represent a planning graph for every state
- The SAG is built once per search episode and we can use it for relaxed plans for every search node, instead of building a LUG at every node
- Extract relaxed plans from SAG by ignoring planning graph components not labeled by states in our search node.

[JAIR, 2006]
Build a LUG for all states (union of all belief states)

- Ignore irrelevant labels
- Largest LUG == all LUGs

[AAAI, 2005]
Non-Deterministic Planning Conclusions

- Measure positive interaction and independence between states co-transitioning to the goal via overlap
  - Labeled planning graphs efficiently measure conformant plan distance
- Conformant planning heuristics work for conditional planning without modification

Stochastic Planning
Stochastic Rover Example

```
(define (domain rovers_stochastic)
  (:requirements :strips :typing)
  (:types waypoint data)
  (:predicates
    (at ?x - waypoint)
    (avail ?d - data)
    (comm ?d - data)
    (have ?d - data))
  (:action drive
    :parameters (?x ?y - waypoint)
    :precondition (at ?x)
    :effect (and (at ?y) (not (at ?x))))
  (:action commun
    :parameters (?d - data)
    :precondition (have ?d)
    :effect (probabilistic 0.8 (comm ?d)))
  (:action sample
    :parameters (?d - data ?x - waypoint)
    :precondition (at ?x)
    :effect (when (avail ?d ?x)
      (probabilistic 0.9 (have ?d))))
)
```

```
(define (domain rovers_stochastic1)
  (:domain rovers)
  (:objects
    soil image rock - data
    alpha beta gamma - waypoint)
  (:init (at alpha)
    (probabilistic 0.4 (avail soil alpha)
    0.5 (avail soil beta)
    0.1 (avail soil gamma))
  (:goal (comm soil) 0.5)
)
```

Search in Probabilistic Belief State Space
Handling Uncertain Actions

- Extending LUG to handle uncertain actions requires label extension that captures:
  - State uncertainty (as before)
  - Action outcome uncertainty
    - Problem: Each action at each level may have a different outcome. The number of uncertain events grows over time – meaning the number of joint outcomes of events grows exponentially with time
    - Solution: Not all outcomes are important. Sample some of them – keep number of joint outcomes constant.

Monte Carlo LUG (McLUG)

- Use Sequential Monte Carlo in the Relaxed Planning Space
  - Build several deterministic planning graphs by sampling states and action outcomes
  - Represent set of planning graphs using LUG techniques
    - Labels are sets of particles
    - Sample which Action outcomes get labeled with particles
    - Bias relaxed plan by picking actions labeled with most particles to prefer more probable support
McLUG for rover example

Monte Carlo LUG (McLUG)
Logistics Domain Results

Scalable, w/ reasonable quality

Grid Domain Results

Again, good scalability and quality!

Need More Particles for broad beliefs
Stochastic Planning Conclusions

- Number of joint action outcomes too large
  - Sampling outcomes to represent in labels is much faster than exact representation
- SMC gives us a good way to use multiple planning graph for heuristics, and the McLUG helps keep the representation small
- Numeric Propagation of probability can better capture interactions with correlation
  - Can extend to cost and resource propagation

Planning with Resources
Planning with Resources

- Propagating Resource Intervals
- Relaxed Plans
  - Handling resource subgoals

---

Rover with power Resource

(define (domain rovers-resource)
  (:requirements :strips :typing)
  (:types waypoint data)
  (:predicates (comm ?d - data)
    (have ?d - data)
    (at ?x - waypoint)
    (avail ?d - data ?x - waypoint))
  (:functions (power)
    (effort ?x ?y - waypoint)
    (effort ?d - data))
  (:action drive
    :parameters (?x ?y - waypoint)
    :precondition (and (at ?x) (> (power) (effort ?x ?y)))
    :effect (and (not (at ?x))
               (decrease (power) (effort ?x ?y))))
  (:action commun
    :parameters (?d - data)
    :precondition (and (have ?d) (> (power) 5))
    :effect (and (comm ?d) (decrease (power) 5)))
  (:action sample
    :parameters (?d - data ?x - waypoint)
    :precondition (and (at ?x) (avail ?d ?x)
                      (> (power) (effort ?d)))
    :effect (and (have ?d) (decrease (power) (effort ?d))))
  (:action recharge
    :parameters ()
    :precondition (and (at ?x) (avail ?d ?x) (< (power) 75))
    :effect (and (have ?d) (increase (power) 25)))
)

(define (problem rovers-resource1)
  (:domain rovers-resource)
  (:objects
    soil image rock - data
    alpha beta gamma - waypoint)
  (:init (at alpha)
    (avail soil alpha)
    (avail rock beta)
    (avail image gamma)
    (= (effort alpha beta) 10)
    (= (effort beta alpha) 5)
    (= (effort alpha gamma) 30)
    (= (effort gamma alpha) 5)
    (= (effort beta gamma) 15)
    (= (effort gamma beta) 10)
    (= (effort soil) 20)
    (= (effort rock) 25)
    (= (effort image) 5)
    (= (power) 25))
  (:goal (and (comm soil)
              (comm image)
              (comm rock)))
)
Resource Intervals

Resource Interval assumes independence among Consumers/Producers

UB: UB(noop) + recharge = 50 + 25 = 75
LB: LB(noop) + drive + sample + drive + sample + commun + drive + drive + drive + sample + commun + drive + drive + sample + commun + drive + drive = -5 + (-30) + (-20) + (-10) + (-25) + (-15) + (-5) = -115

resource intervals (cont’d)
Relaxed Plan Extraction with Resources

Start Extraction as before
Track “Maximum” Resource Requirements For actions chosen at each level
May Need more than One Supporter For a resource!!

Results

Figure 13: Runtime curves on Settlers instances for the planners favoring speed. Time is shown on a logarithmic scale, instance size scales from left to right.
Planning With Resources Conclusion

- Resource Intervals allow us to be optimistic about reachable values
  - Upper/Lower bounds can get large
- Relaxed Plans may require multiple supporters for subgoals
- Negative Interactions are much harder to capture
Temporal Planning

- Temporal Planning Graph
  - From Levels to Time Points
  - Delayed Effects
- Estimating Makespan
- Relaxed Plan Extraction
Rover with Durative Actions

Search through time-stamped states

- **Goal Satisfaction:**
  
  $S = (P, M, \Pi, Q, t) \Rightarrow G$ if $\forall <p, t> \in G$
  
  either:
  - $\exists <p, t> \in P, t < t_i$ and no event in $Q$
    deletes $p$.
  - $\exists e \in Q$ that adds $p_i$ at time $t_e < t_i$.

- **Action Application:**
  
  Action $A$ is applicable in $S$ if:
  - All instantaneous preconditions of $A$ are satisfied by $P$ and $M$.
  - $A$’s effects do not interfere with $\Pi$ and $Q$.
  - No event in $Q$ interferes with persistent preconditions of $A$.
  - $A$ does not lead to concurrent resource change

- **When $A$ is applied to $S$:**
  - $P$ is updated according to $A$’s instantaneous effects.
  - Persistent preconditions of $A$ are put in $\Pi$.
  - Delayed effects of $A$ are put in $Q$.

---

**Search:**
Pick a state $S$ from the queue.
If $S$ satisfies the goals, end
Else non-deterministically do one of
  - Advance the clock
    (by executing the earliest event in $Q$)
  - Apply one of the applicable actions to $S$
Temporal Planning

<table>
<thead>
<tr>
<th>drive(α, β)</th>
<th>drive(β, α)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drive(α, γ)</td>
<td>drive(γ, α)</td>
</tr>
<tr>
<td>commun(soil)</td>
<td>sample(rock, α)</td>
</tr>
<tr>
<td>sample(image, γ)</td>
<td>commun(image)</td>
</tr>
<tr>
<td>commun(image)</td>
<td>have(soil)</td>
</tr>
</tbody>
</table>

Record First time Point Action/Proposition is First Reachable

Assume Latest Start Time for actions in RP

SAPA at IPC-2002

Satellite (complex setting)

Satellite (complex setting)

Rover (time setting)

Rover (time setting)

[JAIR 2003]
Temporal Planning Conclusion

- Levels become Time Points
- Makespan and plan length/cost are different objectives
- Set-Level heuristic measures makespan
- Relaxed Plans measure makespan and plan cost

Overall Conclusions

- Relaxed Reachability Analysis
  - Concentrate strongly on positive interactions and independence by ignoring negative interaction
  - Estimates improve with more negative interactions
- Heuristics can estimate and aggregate costs of goals or find relaxed plans
- Propagate numeric information to adjust estimates
  - Cost, Resources, Probability, Time
- Solving hybrid problems is hard
  - Extra Approximations
  - Phased Relaxation
  - Adjustments/Penalties
Why do we love PG Heuristics?

- They work!
- They are “forgiving”
  - You don't like doing mutex? okay
  - You don't like growing the graph all the way? okay.
- Allow propagation of many types of information
  - Level, subgoal interaction, time, cost, world support, probability
- Support phased relaxation
  - E.g. Ignore mutexes and resources and bring them back later…
- Graph structure supports other synergistic uses
  - e.g. action selection
- Versatility…

Versatility of PG Heuristics

- **PG Variations**
  - Serial
  - Parallel
  - Temporal
  - Labelled

- **Propagation Methods**
  - Level
  - Mutex
  - Cost
  - Label

- **Planning Problems**
  - Classical
  - Resource/Temporal
  - Conformant

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