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 http://verde.eas.asu.edu
rao@asu.edu, http://rakaposhi.eas.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
- <Break>
- Non-Deterministic/Probabilistic Planning
- Resources (Continuous Quantities)
- Temporal Planning
- Wrap-up
Classical Planning

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

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 (?d - data)
    :precondition (have ?d)
    :effect (comm ?d))
  (:action sample
    :parameters (?d - 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)
  (:goal (and (comm image)
    (comm rock))))

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

Distance of a Set of Literals

- \( h(S) = \sum_{p \in S} lev(p) \) (Sum)
- \( h(S) = lev(S) \) (Set-Level)

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

Example of Level Based Heuristics

- \( set-level(s_I, G) = 3 \)
- \( \max(s_I, G) = \max(2, 3, 3) = 3 \)
- \( \text{sum}(s_I, G) = 2 + 3 + 3 = 8 \)

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

Count Actions
RP(sI, G) = 8

Support Goal Propositions Individually

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

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

- have(soil) only supporter is mutex with at(β) only supporter
- Inconsistent Support
- have(soil) has a supporter not mutex with a supporter of at(β)
- --feasible together at level 2--

---

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

---

Anatomy of a State-space Regression planner

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

---

<table>
<thead>
<tr>
<th>PROBLEM</th>
<th>Level</th>
<th>Sum</th>
<th>AdjSum2M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gripper-25</td>
<td>6/1.42</td>
<td>6/1.42</td>
<td>6/1.42</td>
</tr>
<tr>
<td>Gripper-30</td>
<td>8/1.57</td>
<td>8/1.57</td>
<td>8/1.57</td>
</tr>
<tr>
<td>Tower-7</td>
<td>12/1.32</td>
<td>12/1.32</td>
<td>12/1.32</td>
</tr>
<tr>
<td>Tower-9</td>
<td>18/1.54</td>
<td>18/1.54</td>
<td>18/1.54</td>
</tr>
<tr>
<td>BBlocks3</td>
<td>3/0.35</td>
<td>3/0.35</td>
<td>3/0.35</td>
</tr>
<tr>
<td>BBlocks2</td>
<td>3/0.35</td>
<td>3/0.35</td>
<td>3/0.35</td>
</tr>
<tr>
<td>Pellet4</td>
<td>3/0.52</td>
<td>3/0.52</td>
<td>3/0.52</td>
</tr>
<tr>
<td>Pellet3</td>
<td>3/0.52</td>
<td>3/0.52</td>
<td>3/0.52</td>
</tr>
<tr>
<td>Pellet2</td>
<td>3/0.52</td>
<td>3/0.52</td>
<td>3/0.52</td>
</tr>
<tr>
<td>AipGrid1</td>
<td>3/0.52</td>
<td>3/0.52</td>
<td>3/0.52</td>
</tr>
<tr>
<td>AipGrid2</td>
<td>3/0.52</td>
<td>3/0.52</td>
<td>3/0.52</td>
</tr>
</tbody>
</table>

HAdjSum2M(S) = length(RelaxedPlan(S)) + max $p,q \in S \delta(p,q)$

Where $\delta(p,q) = \text{lev}(p,q) - \max\{\text{lev}(p), \text{lev}(q)\}$

/*Degree of -ve interaction*/
Rover Example in Regression

\[
\text{comm(soil)} \quad \text{comm(rock)} \quad \text{comm(image)}
\]

\[
\text{comm(rock)} \quad \text{have(image)}
\]

\[
\text{avail(rock, } \beta \text{)} \quad \text{have(image)}
\]

\[
\text{sample(rock, } \beta \text{)} \quad \text{comm(rock)}
\]

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

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

Should be \( \infty \), \( s_4 \) is inconsistent, how do we improve the heuristic??

Plan Space Search

Then it was cruelly UnPOPped

The good times return with Re(vived)POP

AltAlt Performance

Logistics Domain (AIPS-00).

AltAlt Performance

Problems.

Time (Seconds)

STAN 3.0

HSP 2.0

HSP-r

AltAlt 1.0

Schedule Domain (AIPS-00)

Problems

Time (Seconds)

STAN 3.0

HSP 2.0

AltAlt 1.0

POP Algorithm

1. **Plan Selection**: Select a plan \( P \) from the search queue
2. **Flaw Selection**: Choose a flaw \( f \) (open condition 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
4. Return NULL if no resolution exist

1. Initial plan:

   \[
   S_0 \quad g_0 \quad S_{\infty}
   \]

2. Plan refinement (flaw selection and resolution):

   \[
   q_1 \quad S_1 \quad p \quad S_2 \quad g_1 \quad S_{\infty}
   \]

   \[
   \text{oc}_1 \quad S_2 \quad g_2 \quad \sim p \quad S_{\infty}
   \]

Choice points

- Flaw selection (open condition? unsafe link? Non-backtrack choice)
- Flaw resolution/Plan Selection (how to select (rank) partial plan?)
PG Heuristics for Partial Order Planning

- 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

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

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
  - Progression, Regression, POCL

Cost-Based Planning

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

Rover Cost Model
Cost Propagation

Terminating Cost Propagation

- Stop when:
  - goals are reached (no-lookahead)
  - costs stop changing (∞-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

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

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

**Challenges:**
- Need to propagate costs on the planning graph
- The exact set of goals are not clear
  - Interactions between goals
  - Obvious approach of considering all $2^n$ goal subsets is infeasible
- **Idea:** Select a subset of the top level goals upfront
- **Challenge:** Goal interactions
  - Approach: Estimate the net benefit of each goal in terms of its utility minus the cost of its relaxed plan
  - Bias the relaxed plan extraction to reuse the actions already chosen for other goals

**Goal Set Selection In Rover Problem**

<table>
<thead>
<tr>
<th>Comm(rock)</th>
<th>Comm(image)</th>
<th>Comm(soil)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50-25 = 25</td>
<td>60-40 = 20</td>
<td>20-35 = -15</td>
</tr>
<tr>
<td>110-65 = 45</td>
<td>70-60 = 20</td>
<td></td>
</tr>
<tr>
<td>130-100 = 30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Found By**
- Cost Propagation
- Biased RP

**SAPA$^P$S (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 AltAlt$^{P}$S**

- 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
- Two Shoes
  - Cost: 100
  - Utility: 500

### GOOD
- High-res photo utility: 150
- Soil sample utility: 100
- Both goals: Additional 200

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

**utility dependencies**
- Goals may interact in the utility they give
- Explicitly defined by user
- No need to consider this in classical planning

All PSP Planners
- AltWlt, Optiplan, SapaPSP

Our planner based on SapaPSP

---

Defining Goal Dependencies

| (have-left-shoe)          | 150 |
| (have-image high-res loc1)| +500|
| (have-right-shoe)         | 100 |
| (have-soil loc1)          | 100 |
| (have-image high-res loc1)| +200|
| (have-soil loc1)          | 200 |
| (have-image low-res loc1) | -80 |
**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

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

**Idea:**
Use declarative programming

---

**Heuristically Identifying Goals**

We encode the relaxed plan

**Binary Variables**

- goal utility dependencies
- goals
- actions

| $G \subseteq \bar{G}$, $f^u(G') \neq 0 : X_G$ |
| $G \subseteq \bar{G}$, $f^u(G') \neq 0 : X_G$ |
| $\forall g \in G$, $X_g$ |
| $\forall a \in P$, $X_a$ |

**Constraints**

- Select actions for goals
- Select goals in dependency
- Select goals in dependency

| $\forall a \in P$, $\forall g \in GS(a)$ : $(1 - X_g) + X_a \geq 1$ |
| $\sum_{g \in G'}(1 - X_g) + X_{G'} \geq 1$ |
| $\forall g \in G'$, $\sum_{g \in G'}(1 - X_g) + X_{G'} \geq 1$ |

**Objective function:**

| maximizes $(\sum f^u(G') \cdot X_{G'} - \sum X_a \cdot c_a)$ |

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

- \( a_1: \text{Move(l1,l2)} \)
- \( a_2: \text{Calibrate(camera)} \)
- \( a_3: \text{Sample(l2)} \)
- \( a_4: \text{Take\_high\_res(l2)} \)
- \( a_5: \text{Take\_low\_res(l2)} \)
- \( a_6: \text{Sample(l1)} \)

\( g_1: \text{Sample(l1)} \)
\( g_2: \text{Sample(l2)} \)
\( g_3: \text{HighRes(l2)} \)
\( g_4: \text{lowRes(l2)} \)

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

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

Results

**ZenoTravel**

- **SPUDS**
- **SapaPS**

Results

**Satellite**

- **SPUDS**
- **SapaPS**

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

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

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

January 18, 2007
IJCAI'07 Tutorial T12
61

Belief State Distance

Estimate Plans for each state pair
Capture Positive Interaction & Independence

Union = 17

January 18, 2007
IJCAI'07 Tutorial T12
62

State Distance Aggregations

[Bryce et al., 2005]

Just Right

Figure 16: Ratio of heuristic estimates for distance between BS<sub>3</sub> and BS<sub>2</sub> to optimal plan length. RV = Rovers, L = Logistics, B = BT, BC = BTC, C = Cube Center, R = Ring.
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 where they are reached (labeled).

Subgoals and Supporters need not be used for every state where they are reached (labeled).

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.
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)
  (have ?d - data))
 (action drive
  :parameters (?x ?y - waypoint)
  :precondition (at ?x)
  :effect (and (at ?y) (not (at ?x))))
 (action common
  :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))))
)

January 18, 2007 ICAT07 Tutorial T12

Search in Probabilistic Belief State Space

January 18, 2007 ICAT07 Tutorial T12

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

January 18, 2007 ICAT07 Tutorial T12
McLUG for rover example

---

Monte Carlo LUG (McLUG)

---

Logistics Domain Results

---

Grid Domain Results

---

Sample States for Initial layer -- avail(soil, ?) not sampled

Particles in action Label must sample action outcome

¾ of particles Support goal, need At least ½

½ of particles Support goal, Okay to stop

Support Preconditions for Particles the Action supports

Pick Persistence By default, Commun covers Other particles

Must Support Goal in All Particles

Scalable, w/ reasonable quality

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

- Propagating Resource Intervals
- Relaxed Plans
  - Handling resource subgoals

Rover with power Resource

```merlin
(define (domain rovers-resource)
  (types waypoint data)
  (predicates (can `m` - data) (have `m` - data) (at `m` - waypoint) (available `m` - data `m` - waypoint)
    (function (power) (effort `m` `y` - waypoint) (effort `m` - data))
  (action drive
    :parameters (`x` `y` - waypoint)
    :precondition (and (at `m`) (= (power) (effort `x` `y`)))
    :effect (and (not (at `m`)) (not (at `m`)) (decrease (power) (effort `x` `y`)))
  )
  (action common
    :parameters (`m` - data)
    :precondition (and (have `m`) (= (power) 5))
    :effect (and (common `m`) (decrease (power) 5))
  )
  (action sample
    :parameters (`m` - data `x` - waypoint)
    :precondition (and (at `m`) (available `m` `m`)
      (= (power) (effort `m`))
    :effect (and (and `m`) (decrease (power) (effort `m`)))
  )
  (action recharge
    :parameters (`m`)
    :precondition (and (at `m`) (available `m` `m`)
      (= (power) 75)
    :effect (and (and `m`) (increase (power) 25))
  )
)
```

```merlin
(define (problem rovers-resource1)
  (domain rovers-resource)
  (objects soil image rock - data alpha beta gamma - waypoint)
  (init (at alpha)
    (available soil alpha)
    (available rock beta)
    (available image gamma)
    (= (effort alpha beta) 10)
    (= (effort alpha gamma) 5)
    (= (effort beta gamma) 15)
    (= (effort gamma beta) 10)
    (= (effort soils 20)
      (= (effort rocks) 25)
      (= (effort image) 5)
      (= (power) 25)
      (goal (and (common soil)) (common image) (common rock)))
)
```
Resource Intervals

\[
\begin{align*}
\text{avail(soil, } \alpha) & \quad \text{sample(soil, } \alpha) \\
\text{avail(rock, } \beta) & \quad \text{drive(\alpha, } \gamma) \\
\text{avail(image, } \gamma) & \quad \text{recharge} \\
\end{align*}
\]

Resource Interval assumes independence among Consumers/Producers
UB: UB(noop) + recharge = 25 + 25 = 50
LB: LB(noop) + sample + drive = 25 + (-20) + (-10) = -5

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

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 Scheds 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

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

- **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 \( Q \).

---

### Temporal Planning

- **Record First time Point Action/Proposition is First Reachable**

- **Assume Latest Start Time for actions in \( RP \)**

---

### SAPA at IPC-2002

- **SAPA at IPC-2002**

- **Rover (time setting)**

- **Satellite (complex setting)**

---

**Search through time-stamped states**

- **Set of protected persistent conditions (could be binary or resource conds).**
- **Set of functions represent resource values.**
- **Event queue (contains resource as well as binary fluent events).**

**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 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

<table>
<thead>
<tr>
<th>PG Variations</th>
<th>Propagation Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>Level</td>
</tr>
<tr>
<td>Parallel</td>
<td>Mutex</td>
</tr>
<tr>
<td>Temporal</td>
<td>Cost</td>
</tr>
<tr>
<td>Labelled</td>
<td>Label</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Planning Problems</th>
<th>Planners</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>Regression</td>
</tr>
<tr>
<td>Resource/Temporal</td>
<td>Progression</td>
</tr>
<tr>
<td>Conformant</td>
<td>Partial Order</td>
</tr>
<tr>
<td></td>
<td>Partial Order</td>
</tr>
<tr>
<td></td>
<td>Graphplan-style</td>
</tr>
</tbody>
</table>