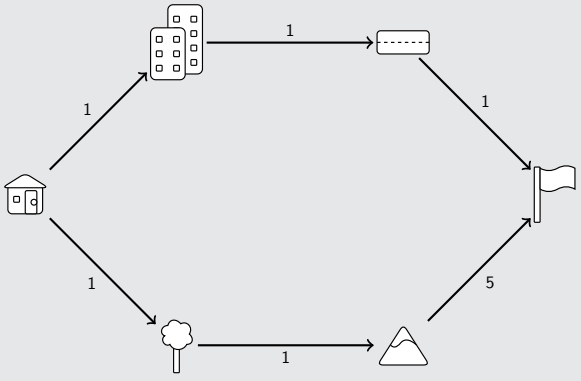
Probabilistic Replanning with Guarantees

Johannes Schmalz
Supervisor: Felipe Trevizan

Australian National University

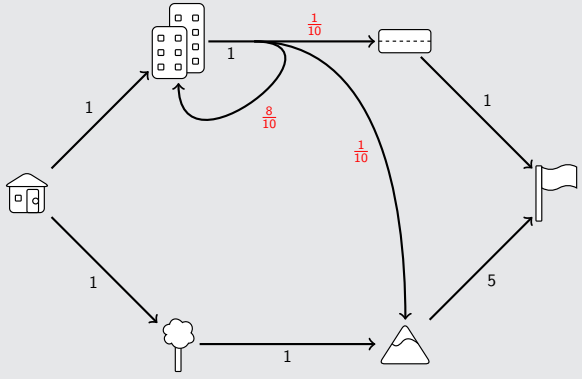
8th of April 2022

Example Problem (Biking to Uni.)



A solution is called a *plan*, and we want to minimise its cost.

Example Problem (Biking to Uni. but I can get lost)



Now a solution is called a *policy* ($\pi : S \rightarrow A$) and we want to minimise its **expected** cost.

Some of the existing algorithms

- Value Iteration – 1957
- RTDP – 1995
- LRTDP – 2003

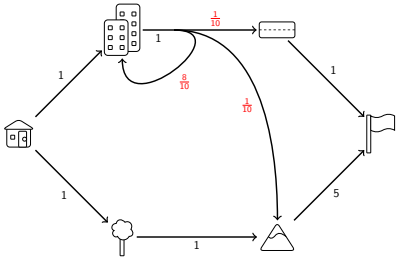
} **Optimal Planners**

- **FF-Replan** – winner of IPPC 2004
- **Robust-FF** – winner of IPPC 2008

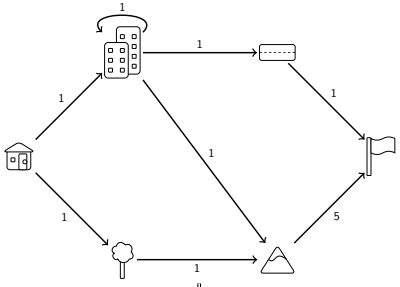
} **Replanners** – return some policy quickly and improve on it iteratively

IPPC = International Probabilistic Planning Competition

SSP



Det'ise



Det.

Solve

Policy

$\pi(\text{home}) = \text{bike to city}$
$\pi(\text{city}) = \text{bike to road}$
$\pi(\text{road}) = \text{bike to uni.}$

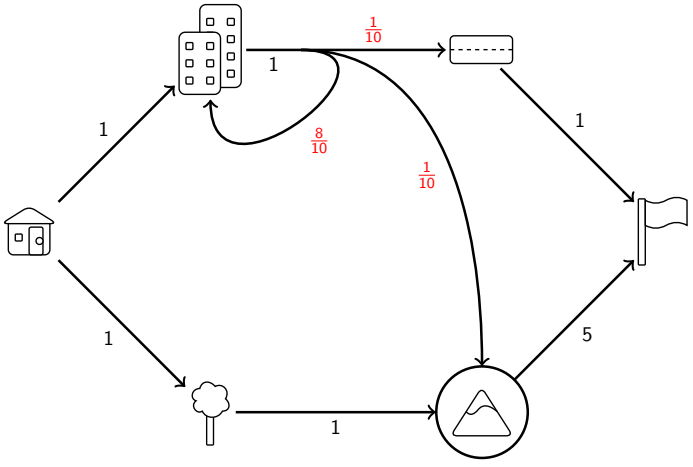
Cast

home $\xrightarrow{\text{bike to city}}$ city
city $\xrightarrow{\text{bike to road}}$ road
road $\xrightarrow{\text{bike to uni.}}$ uni.

Plan

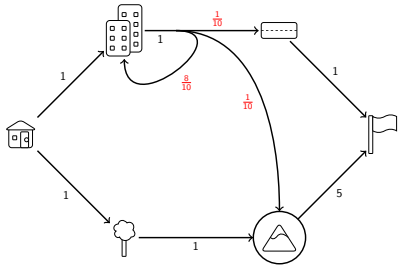
This policy is *open* – there is a state on which π is undefined!

$\pi(\text{home}) = \text{bike to city}$
$\pi(\text{city}) = \text{bike to road}$
$\pi(\text{road}) = \text{bike to uni.}$
$\pi(\text{mountain}) = ?$

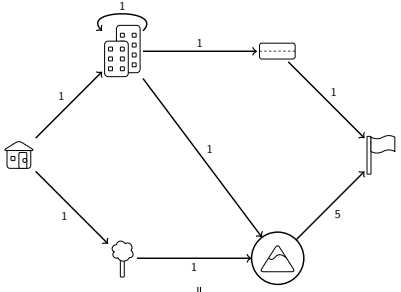


So, we need to replan from the open state!

SSP



Det'ise



Det.

Solve

Policy

$\pi(\text{home}) = \text{bike to city}$
$\pi(\text{city}) = \text{bike to road}$
$\pi(\text{road}) = \text{bike to uni.}$
$\pi(\text{mountain}) = \text{bike to uni.}$

Cast

mountain $\xrightarrow{\text{bike to uni.}}$ uni.

Plan

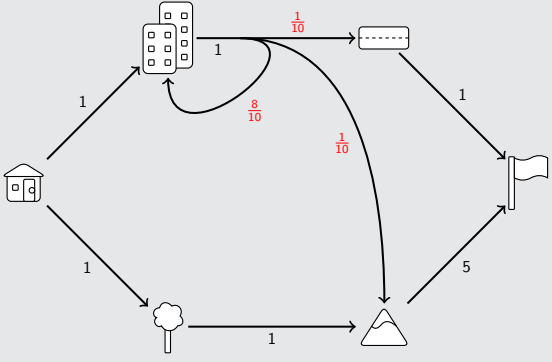
Policy is Closed

$\pi(\text{home}) = \text{bike to city}$
$\pi(\text{city}) = \text{bike to road}$
$\pi(\text{road}) = \text{bike to uni.}$
$\pi(\text{mountain}) = \text{bike to uni.}$

All states reachable by following π are defined.

Policy is **not** Optimal

- expected cost through city $=$
  $1 + \sum_{i=0}^{\infty} \left(\frac{8}{10} \cdot 1\right)^i + 1 + \frac{1}{2} + \frac{5}{2} = 10$
- expected cost through park $= 7$.



Probabilistic Replanners are very fast, but can not guarantee an optimal solution.

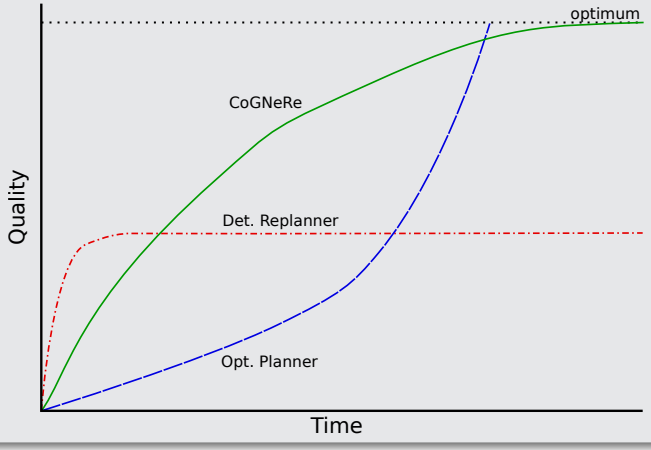
Can we construct a replanner with optimality guarantees?

Probabilistic Replanners are very fast, but can not guarantee an optimal solution.

Can we construct a replanner with optimality guarantees?

Spoiler alert: yes.

Bridging the Gap between Replanners and Optimal Planners



(Disclaimer: this graph does not show actual data!)

Master Problem (Gives Optimal Policy π^* for SSPs)

$$\begin{aligned}
\text{minimise} \quad & \sum_{f \in \text{plans} \cup \text{cycles}} \text{flow}(f) \cdot \text{cost}(f) & & \\
\text{subject to} \quad & \text{flow}(f) \geq 0 & \forall f \in \text{plans} \cup \text{cycles} \quad & \text{(C1)} \\
& \sum_{\phi \in \text{plans}} \text{flow}(\phi) = 1 & & \text{(C2)} \\
& \text{regrouping constraints} & & \text{(C3)}
\end{aligned}$$

Master Problem (Gives Optimal Policy π^* for SSPs)

$$\begin{aligned}
\text{minimise} \quad & \sum_{f \in \text{plans} \cup \text{cycles}} \text{flow}(f) \cdot \text{cost}(f) & & \\
\text{subject to} \quad & \text{flow}(f) \geq 0 & \forall f \in \text{plans} \cup \text{cycles} \quad & \text{(C1)} \\
& \sum_{\phi \in \text{plans}} \text{flow}(\phi) = 1 & & \text{(C2)} \\
& \text{regrouping constraints} & & \text{(C3)}
\end{aligned}$$

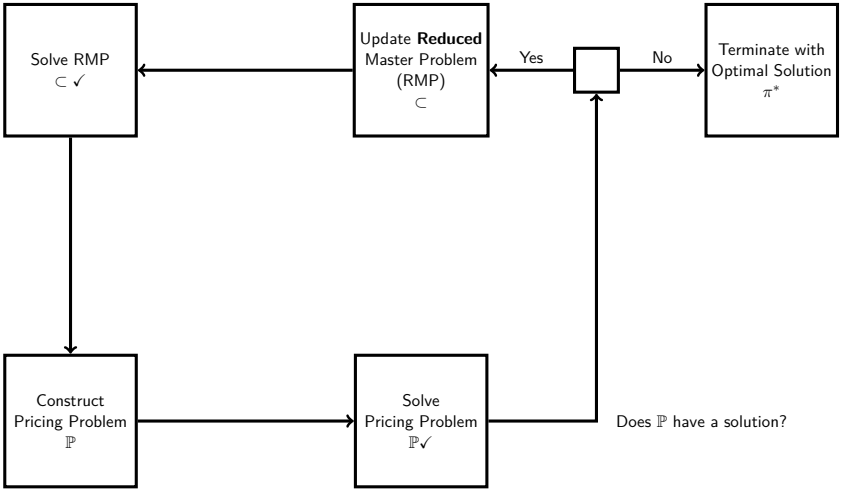
Problem: one variable $\text{flow}(f)$ for each plan and cycle

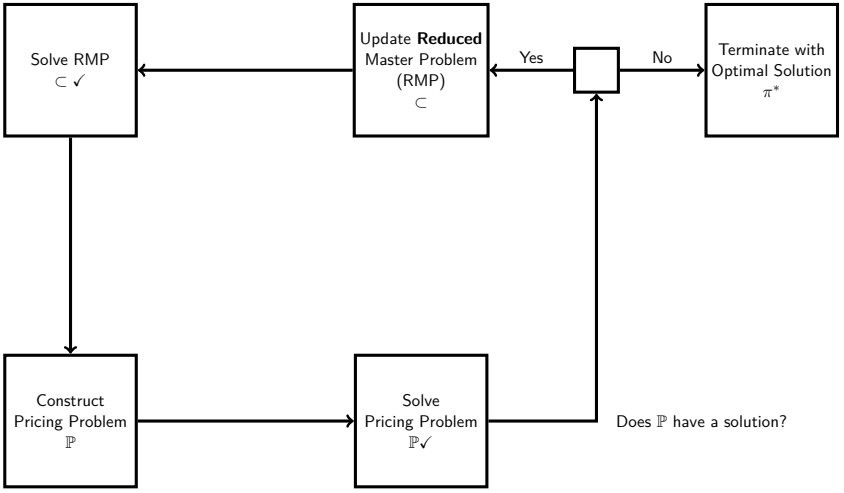
Master Problem (Gives Optimal Policy π^* for SSPs)

$$
\begin{aligned}
\text{minimise} \quad & \sum_{f \in \text{plans} \cup \text{cycles}} \text{flow}(f) \cdot \text{cost}(f) & & \\
\text{subject to} \quad & \text{flow}(f) \geq 0 & \forall f \in \text{plans} \cup \text{cycles} \quad & \text{(C1)} \\
& \sum_{\phi \in \text{plans}} \text{flow}(\phi) = 1 & & \text{(C2)} \\
& \text{regrouping constraints} & & \text{(C3)}
\end{aligned}
$$

Problem: one variable $\text{flow}(f)$ for each plan and cycle

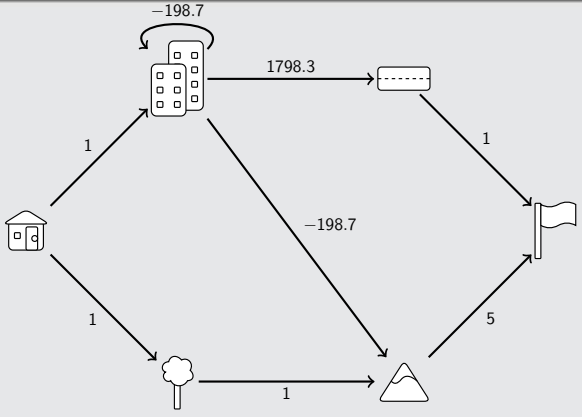
Solution: Column Generation


Solve RMP
$\subset \checkmark$
Update **Reduced** Master Problem (RMP)
$\subset$
Yes
No
Terminate with Optimal Solution
π^*
Construct Pricing Problem
$\mathbb{P}$
Solve Pricing Problem
$\mathbb{P}\checkmark$
Does $\mathbb{P}$ have a solution?



Note: we can extract a (potentially suboptimal) solution at any moment!

Pricing Problem for Biking to Uni but I can get lost (after first RMP)



Variant of deterministic planning!
Need to find

- plan with cost < 1800.3 or
- negative cycle

or show that neither exist.

How easy is it to solve this variant of the deterministic planning problem?

Bad News

- Negative costs
- Need to deal with negative cycles (for the rest of the talk, **ignore this**)
- Action costs depend on the state they're in — hard to encode for heuristics

Good News

- We don't need the cheapest plan/cycle, just something that's cheap enough
- Structure of the problem remains identical
- Not too many costs get updated each turn

State-of-the-art deterministic solver: A^* with good heuristics.

A Small Selection of Modern Heuristics

Heuristic	Pros	Cons
Op. Counting	Easy to adapt	Slow
Abstraction	Theoretically easy to adapt	Not the fastest
PDBs	Construction cost can be amortised with reuse, then very fast!	Negative costs are painful

TLDR: it is non-trivial to adapt strong heuristics to our problem.

A different approach:

Non-optimal searches

- Greedy Search
- Large Neighbourhood Search

Remember: column generation needs a sufficiently cheap solution, not the cheapest.

Large Neighbourhood Search (LNS)

1. Start with previous solution:

$$\text{home} \xrightarrow{\text{bike to city}} \text{city} \xrightarrow{\text{try to bike to road}} \text{mountain} \xrightarrow{\text{bike to uni.}} \text{uni.}$$

2. Destroy part of it:

$$\text{home} \underbrace{\xrightarrow{\text{bike to city}} \text{city} \xrightarrow{\text{try to bike to road}}} \text{mountain} \xrightarrow{\text{bike to uni.}} \text{uni.}$$

3. Repair it:

$$\text{home} \xrightarrow{\text{bike to park}} \text{park} \xrightarrow{\text{bike to mountain}} \text{mountain} \xrightarrow{\text{bike to uni.}} \text{uni.}$$

Iterate steps (2) and (3) until the solution is cheap enough.
Pros: can reuse information from previous pricing problem and (usually) very fast!

Large Neighbourhood Search (LNS)

1. Start with previous solution:

$$\text{home} \xrightarrow{\text{bike to city}} \text{city} \xrightarrow{\text{try to bike to road}} \text{mountain} \xrightarrow{\text{bike to uni.}} \text{uni.}$$

2. Destroy part of it:

$$\text{home} \underbrace{\xrightarrow{\text{bike to city}} \text{city} \xrightarrow{\text{try to bike to road}}} \text{mountain} \xrightarrow{\text{bike to uni.}} \text{uni.}$$

3. Repair it:

$$\text{home} \xrightarrow{\text{bike to park}} \text{park} \xrightarrow{\text{bike to mountain}} \text{mountain} \xrightarrow{\text{bike to uni.}} \text{uni.}$$

Iterate steps (2) and (3) until the solution is cheap enough.
Pros: can reuse information from previous pricing problem and (usually) very fast!

Currently CoGNeRe with LNS overtakes Robust-FF on some problems!

What we've seen

- There is a gap between replanners (fast but no guarantees) and optimal planners (optimal but not designed for any-time)
- CoGNeRe bridges this: it is a replanner with optimality guarantees
- The ongoing challenge is to make CoGNeRe as fast as possible
  - the bottleneck is solving the pricing problem;
  - LNS is promising!

Bonus facts about CoGNeRe

- Objective doesn't have to be expected cost, e.g. max. probability of reaching goal
- Column generation gives a running lower bound
- CoGNeRe can piggy-back off other replanners
- Finding an optimal policy is defined w.r.t. all trajectories through the state-space, an infinite set! but we can do this with the finite set of plans and cycles