AI Planning for Robotics and Human-Robot Interaction

Michael Cashmore — King's College London

Luca Iocchi — Sapienza University of Rome

Daniele Magazzeni — King's College London

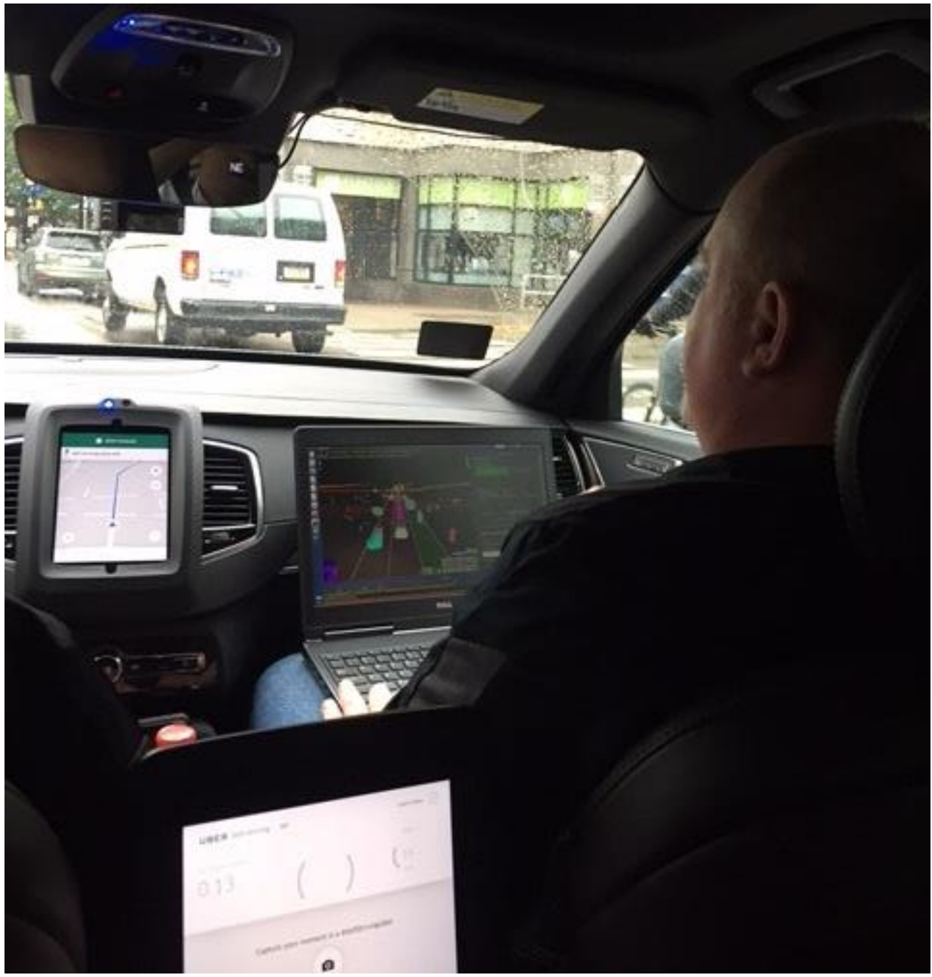
ICAPS 2017

19 June 2017

Pittsburgh –USA

Why Human-Robot Interaction is important…

Coming here this morning….



2 people for driving a car

AI is CREATING jobs!

Disclaimer 1

Planning and Robotics is a growing area!

ICAPS workshops PlanRob
ICAPS Special Track on Planning and Robotics
PlanRob workshop + tutorial at ICRA 2017
Dagstuhl workshop on Planning and Robotics

This tutorial covers only some aspects

PlanRob workshop tomorrow (full day)

Disclaimer 2

One can use several formalisms to model robotics domains.
And one can use several techniques for planning in these domains.

Having said that, this tutorial will focus on
Domain-Independent Planning through PDDLx

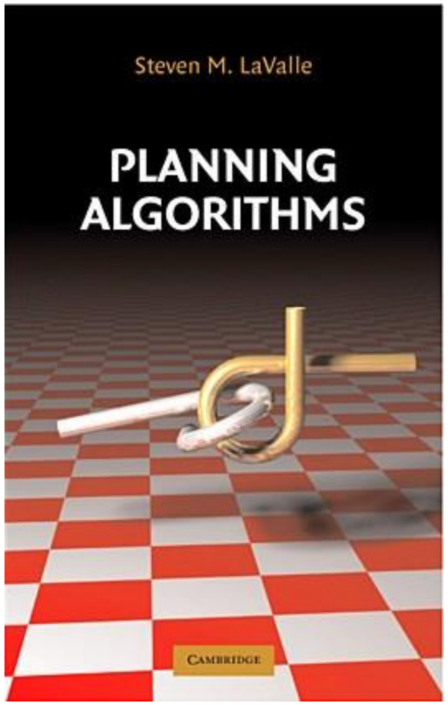
Disclaimer 3

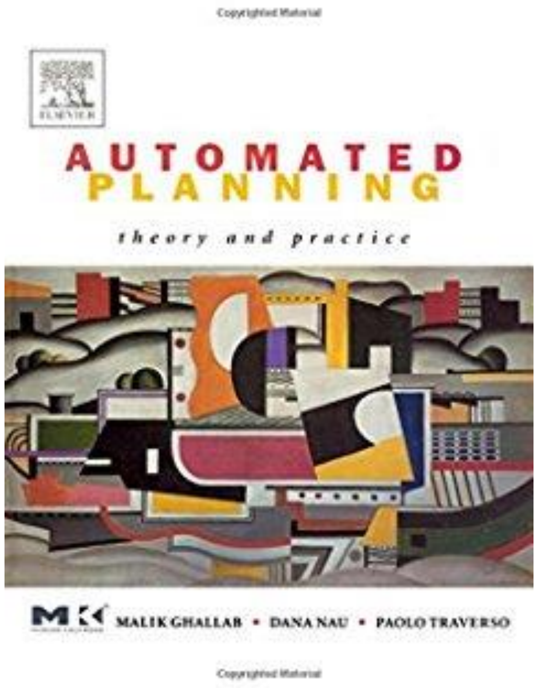
Planning is actually *plural*

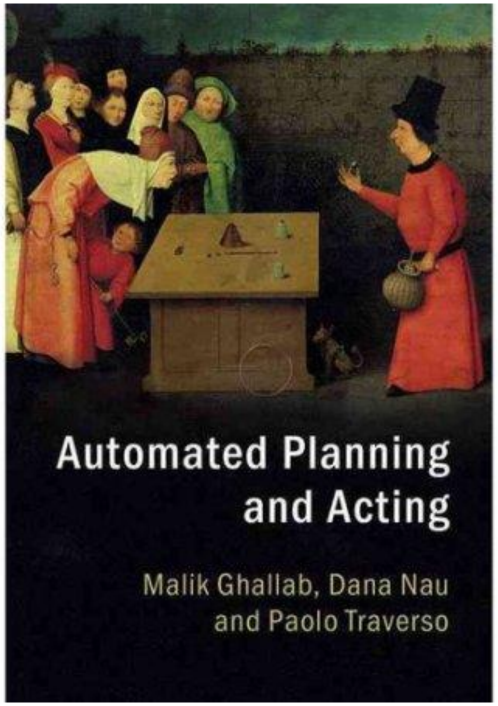
Thanks to Malik Ghallab!

planning includes many things

in this tutorial: “planning”=“task planning”






Automated Planning and Acting
Malik Ghallab, Dana Nau and Paolo Traverso

Disclaimer 4

This is a tutorial
and we agreed to make it an *accessible* one

Slides + Virtual Machine + Demo
available in the ROSPlan website

Outline

- **Why PDDL Planning for Robotics and HRI?**
- **ROSPlan I: Planning with ROS**

Coffee (10.30-11.00)

- **ROSPlan II: Planning with Opportunities**
- **Petri Net Plan Execution**
- **Open challenges**

Outline

- **Why PDDL Planning for Robotics and HRI?**

Where PDDL planning is NOT useful for Robotics?

- **Single/Repetitive Tasks (no PDDL for manipulation/grasping!)**
- **Safe Navigation (Sampling is much better!)**

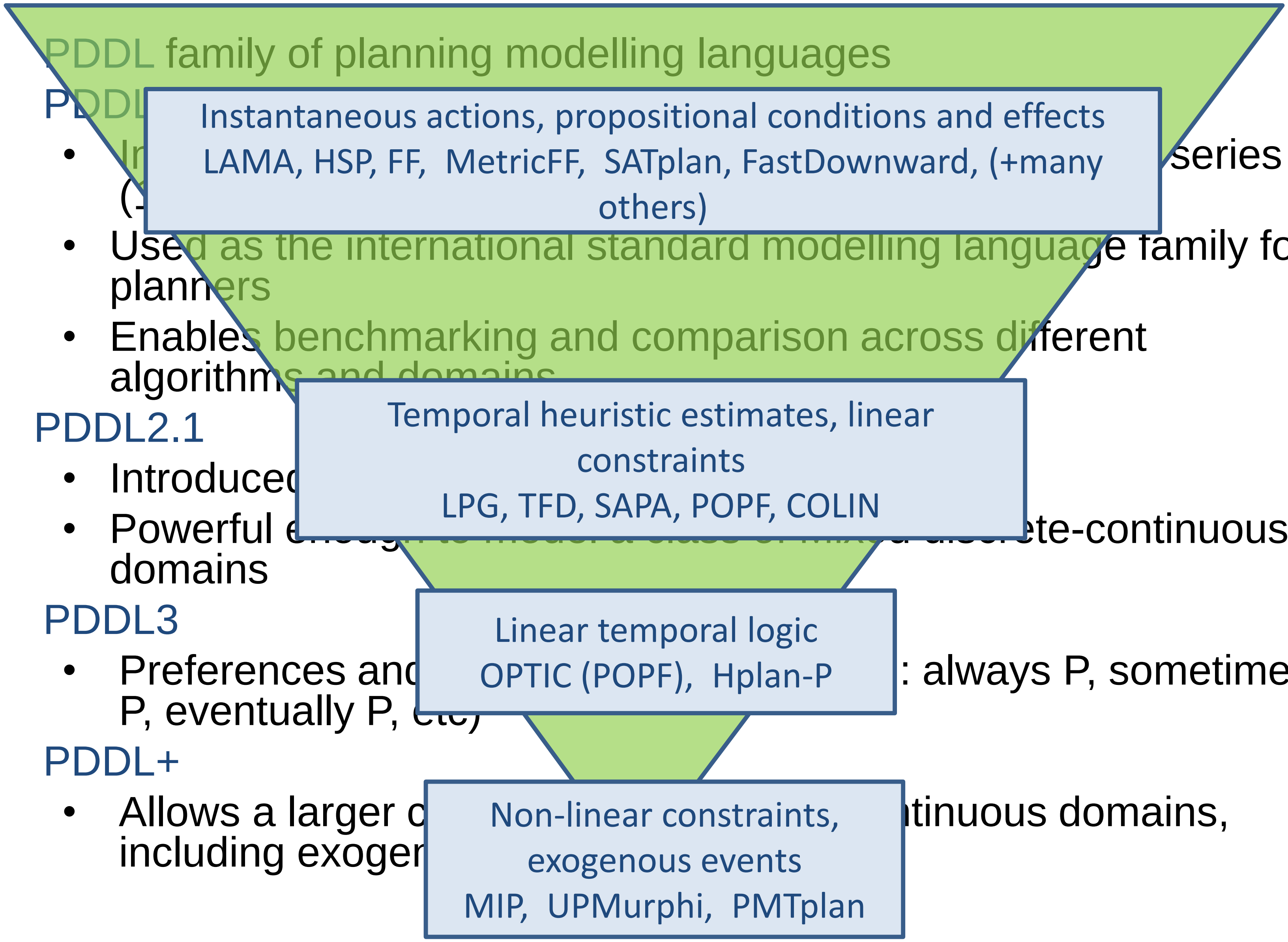
- **PDDL planning is really useful when there is room for optimisation at a task level**

Outline

- Why PDDL Planning for Robotics and HRI?
  - Expressive Planning
  - Opportunistic Planning
  - Strategic Planning
  - eXplainable Planning (XAIP)
  - Planning with Uncertainty

Expressive Planning

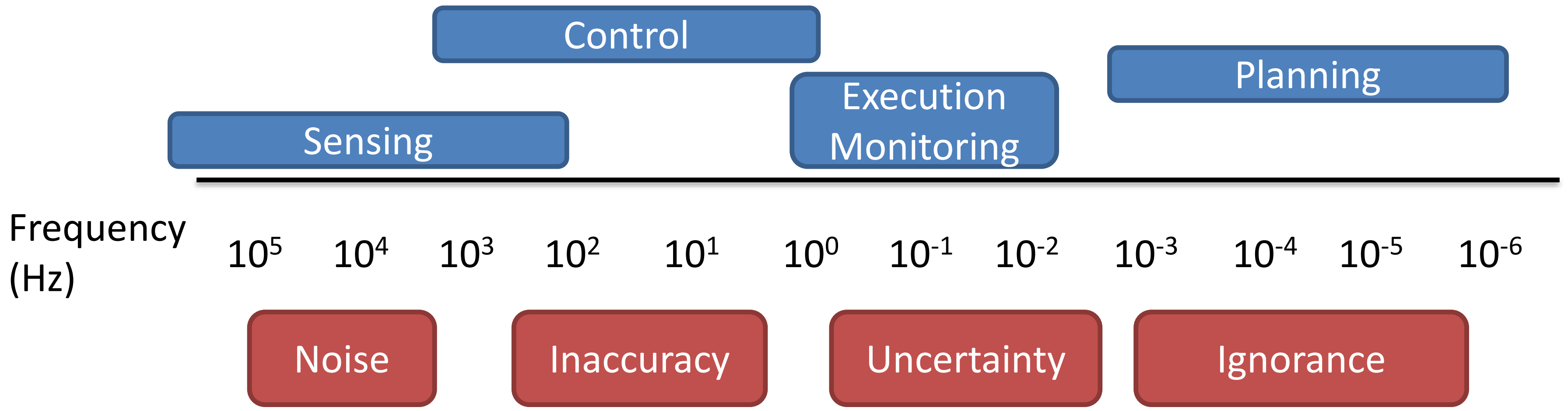
- PDDL family of planning modelling languages
- PDDL
  - In series (
  - Used as the international standard modelling language family for planners
  - Enables benchmarking and comparison across different algorithms and domains
- PDDL2.1
  - Introduced
  - Powerful enough to model a class of mixed discrete-continuous domains
- PDDL3
  - Preferences and : always P, sometimes P, eventually P, etc)
- PDDL+
  - Allows a larger c tinuous domains, including exogen


Instantaneous actions, propositional conditions and effects
LAMA, HSP, FF, MetricFF, SATplan, FastDownward, (+many others)
Temporal heuristic estimates, linear constraints
LPG, TFD, SAPA, POPF, COLIN
Linear temporal logic
OPTIC (POPF), Hplan-P
Non-linear constraints, exogenous events
MIP, UPMurphi, PMTplan

Planning and Control

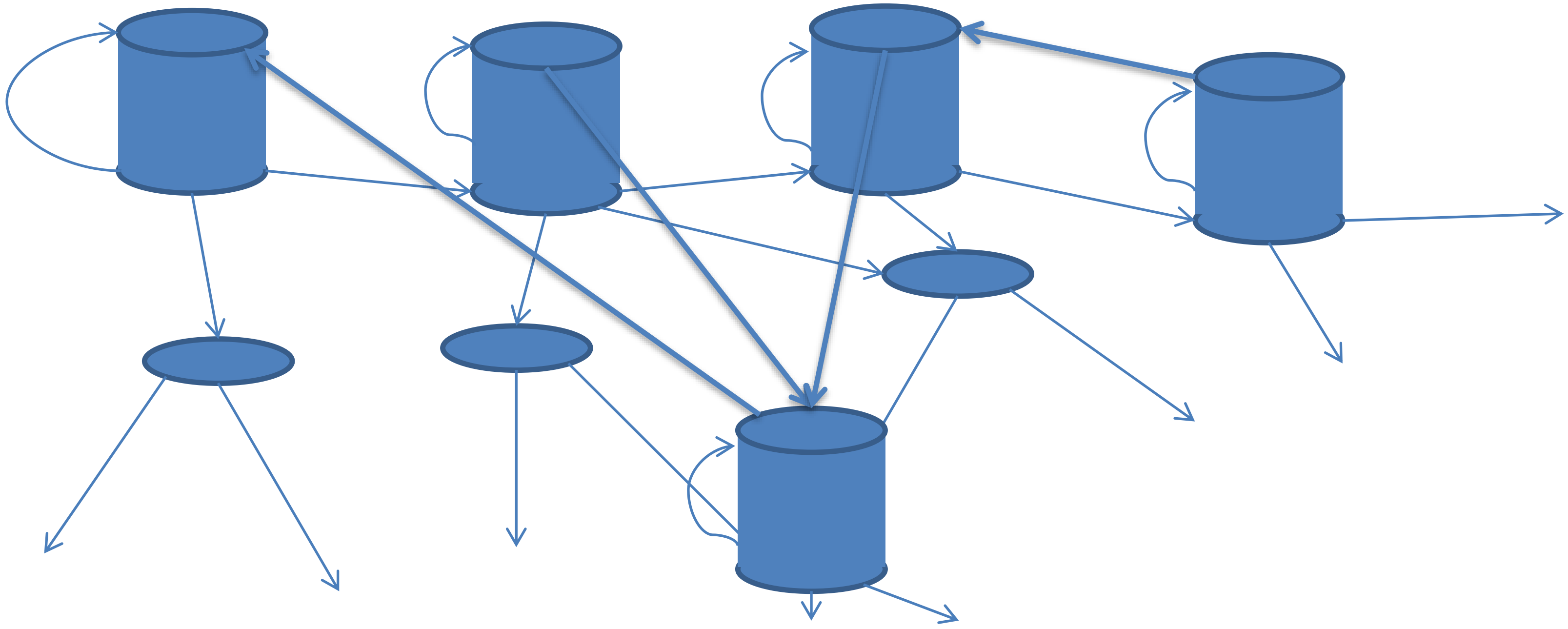
Planning is an AI technology that seeks to select and organise activities in order to achieve specific goals

Plan Dispatch: a controller is responsible for realising each plan action



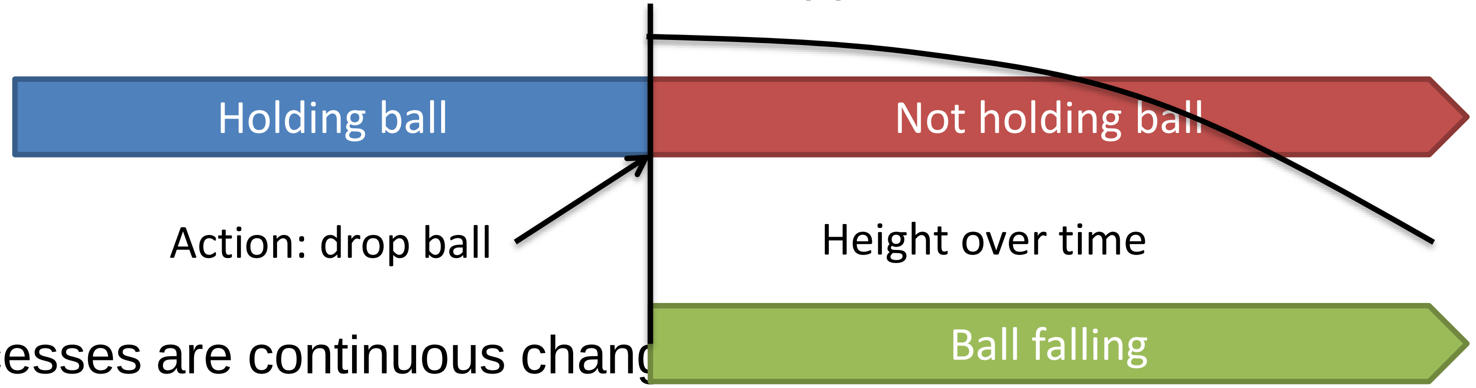
Planning with Time: An Additional Dimension

- Processes mean time spent in states matters



Planning in *Hybrid* Domains

- When actions or events are performed they cause instantaneous changes in the world
  - These are discrete changes to the world state
  - When an action or an event has happened it is over



- Processes are continuous chang
  - Once they start they generate continuous updates in the world state
  - A process will run over time, changing the world at every instant

PDDL+: Let it go

- First drop it…

```
(:action release
 :parameters (?b - ball)
 :precondition (and (holding ?b) (= (velocity ?b) 0))
 :effect (and (not (holding ?b))))
```

- Then watch it fall…

```
(:process fall
 :parameters (?b - ball)
 :precondition (and (not (holding ?b)) (>= (height ?b) 0)))
 :effect (and (increase (velocity ?b) (* #t (gravity)))
              (decrease (height ?b) (* #t (velocity ?b)))))
```

- And then?

PDDL+: See it bounce

- Bouncing…

```
(:event bounce
 :parameters (?b - ball)
 :precondition (and   (>= (velocity ?b) 0)
                    (<= (height ?b) 0))
 :effect (and (assign (height ?b) (* -1 (height ?b)))
                (assign (velocity ?b) (* -1 (velocity ?b)))))
```

- Now let’s plan to catch it…

```
(:action catch
 :parameters (?b - ball)
 :precondition (and (>= (height ?b) 5) (<= (height ?b) 5.01))
 :effect (and (holding ?b) (assign (velocity ?b) 0)))
```

A Valid Plan

- Let it bounce, then catch it...

```
0.1:   (release b1)
4.757: (catch b1)
```

- The validator VAL can be used to check plan validity.
  (https://github.com/KCL-Planning/VAL)

1.51421: **Event triggered!**
Triggered event (bounce b1)
Unactivated process (fall b1)
Updating **(height b1)** (-2.22045e-15) by 2.22045e-15 assignment.
Updating **(velocity b1)** (14.1421) by -14.1421 assignment.

1.51421: **Event triggered!**
Activated process (fall b1)

4.34264: Checking Happening... ...OK!
(height b1)$(t) = -5t^2 + 14.1421t + 2.22045e - 15$
(velocity b1)$(t) = 10t - 14.1421$
Updating **(height b1)** (2.22045e-15) by -2.44943e-15 for continuous update.
Updating **(velocity b1)** (-14.1421) by 14.1421 for continuous update.

4.34264: **Event triggered!**
Triggered event (bounce b1)
Unactivated process (fall b1)
Updating **(height b1)** (-2.44943e-15) by 2.44943e-15 assignment.
Updating **(velocity b1)** (14.1421) by -14.1421 assignment.

4.34264: **Event triggered!**
Activated process (fall b1)

4.757: Checking Happening... ...OK!
(height b1)$(t) = -5t^2 + 14.1421t + 2.44943e - 15$

Updating **(height b1)** (2.44943e-15) by 5.00146 for continuous update.
Updating **(velocity b1)** (-14.1421) by -9.99854 for continuous update.

4.757: Checking Happening... ...OK!
Adding (holding b1)
Updating **(velocity b1)** (-9.99854) by 0 assignment.

4.757: **Event triggered!**
Unactivated process (fall b1)

Plan executed successfully - checking goal

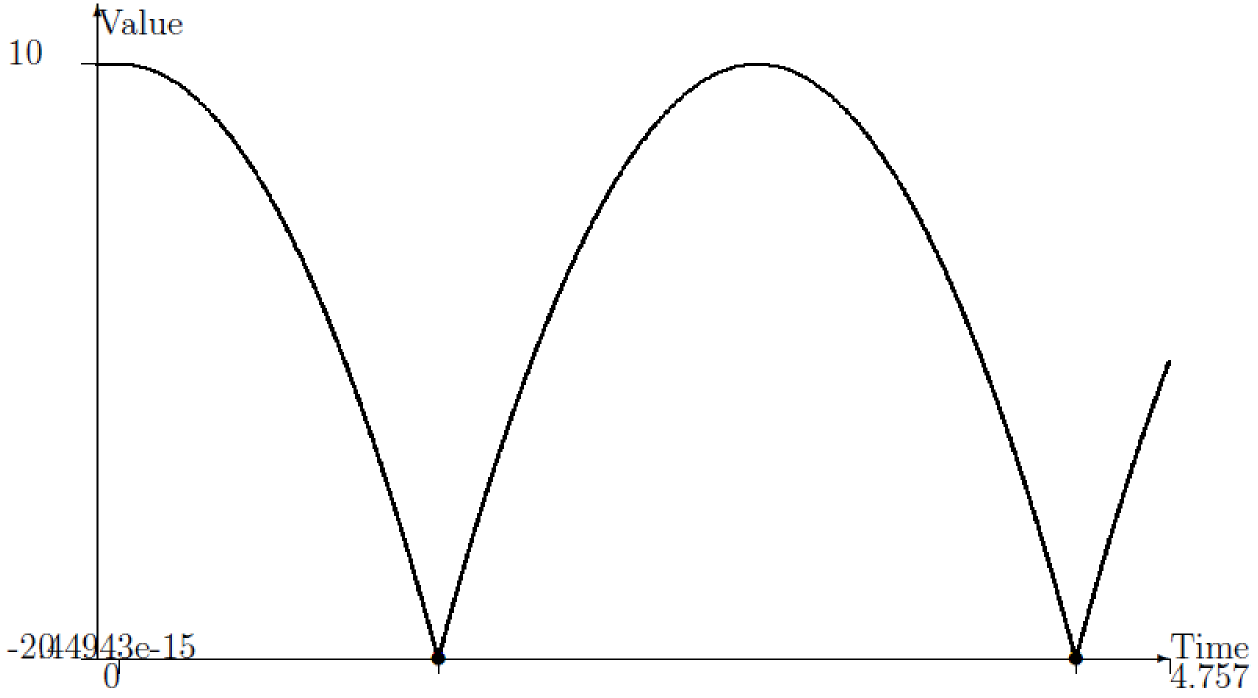


Figure 2.1: Graph of (height b1).

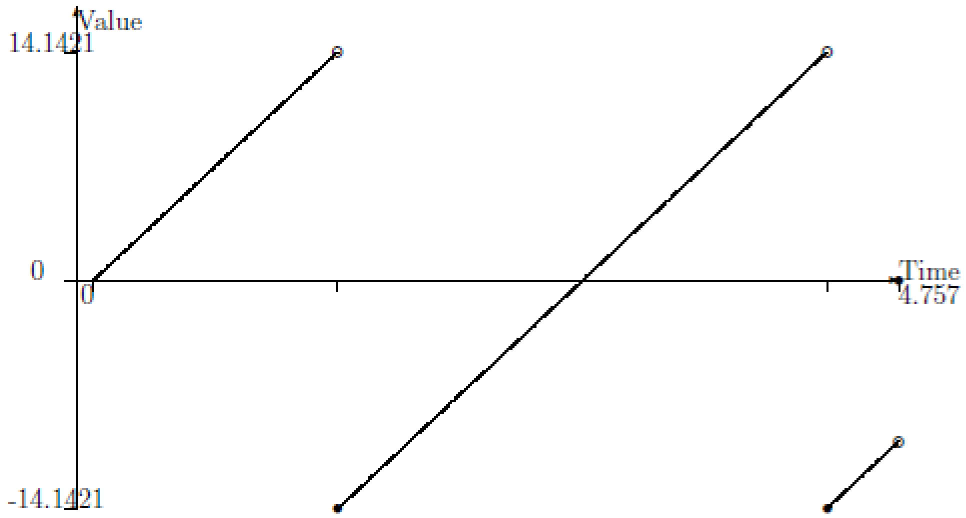


Figure 2.2: Graph of (velocity b1).

Some PDDL+ Planners

- **UPMurphi** (Della Penna et al.) [ICAPS'09]

  Based on Discretise and Validate

  (Baseline for adding new heuristics:
  multiple battery management [JAIR'12] or urban traffic control [AAAI'16])

- **DiNo** (Piotrowski et al.) [IJCAI'16]

  Extend UPMurphi with TRPG heuristic for hybrid domains

- **SMTPlan** (Cashmore et al.) [ICAPS'16]

  Based on SMT encoding of PDDL+ domains

- **ENHSP** (Scala et al.) [IJCAI'16]

  Expressive numeric heuristic planning

- **dReach/dReal** (Bryce et al.) [ICAPS-15]

  Combine SMT encoding with dReal solver

- **POPF** (Coles et al.) [ICAPS-10]

  Combine Forward Search and Linear Programming

One more PDDL+ example

Vertical Take-Off Domain

The aircraft takes off vertically and needs to reach a location where stable fixed-wind flight can be achieved.

The aircraft has fans/rotors which generate lift and which can be tilted by 90 degrees to achieve the right velocity both vertically and horizontally.

V-22 Osprey

Vertical Take-Off

```
(:action start_engines
:parameters ()
:precondition (and (not (ascending)) (not (crashed)) (= (altitude) 0) )
:effect (ascending))
```

```
(:process ascent
:parameters ()
:precondition (and (not (crashed)) (ascending) )
:effect (and (increase (altitude) (* #t (- (* (v_
                  (* (angle) 0.0174533) ) 2) ) ) (g)
                  (increase (distance) (* #t (* (v_f
                  (- 40500 (* (angle) (- 180 (angle
```

Timed Initial Fluents

```
(at 5.0 (= (wind_x) 1.3))
(at 5.0 (= (wind_y) 0.2))
(at 9.0 (= (wind_x) -0.5))
(at 9.0 (= (wind_y) 0.3))
.. …
```

```
(:durative-action increase_angle
:parameters ()
:duration (<= ?duration (- 90 (angle)) )
:condition (and (over all (ascending)) (over all (<= (angle) 90)) (over all (>= (angle) 0)) )
:effect (and (increase (angle) (* #t 1)) ))
```

```
(:event crash
:parameters ()
:precondition (and (< (altitude) 0))
:effect ((crashed))
)
```

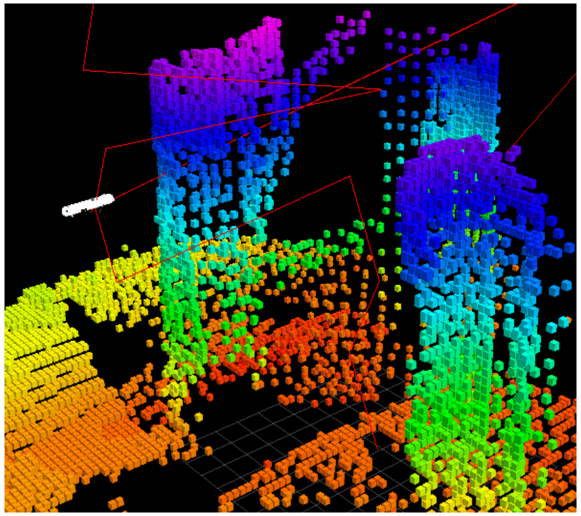
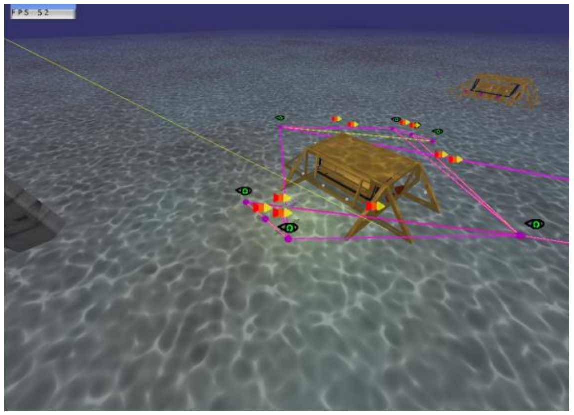
```
(:process wind
 :parameters ()
 :precondition (and (not (crashed)) (ascending) )
 :effect (and (increase (altitude) (* #t (wind_y) 1)
                   (increase (distance) (* #t (wind_x) 1)))
```

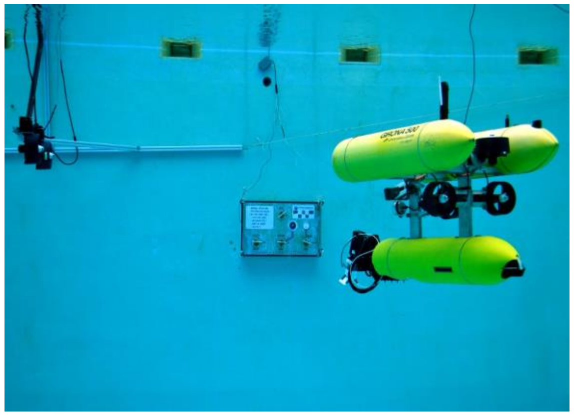
Outline

- **Why PDDL Planning for Robotics and HRI?**
  - **Expressive Planning**
  - **Opportunistic Planning**
  - **Strategic Planning**
  - **eXplainable Planning (XAIP)**
  - **Planning with Uncertainty**

Opportunistic Planning

- Very important in persistent autonomy
- Use case: PANDORA (EU funded project)

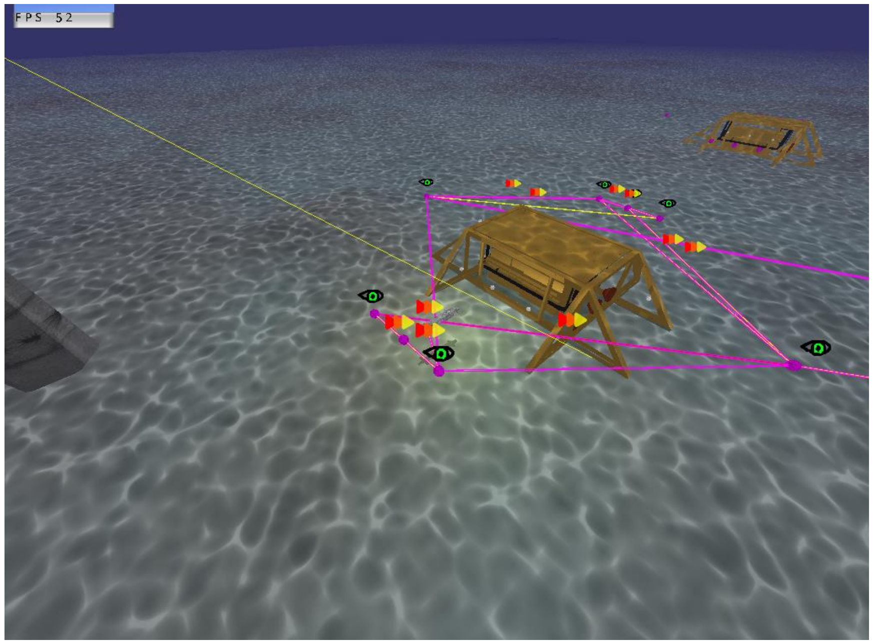





Persistent Autonomy (AUVs)

Inspection and maintenance of a seabed facility:

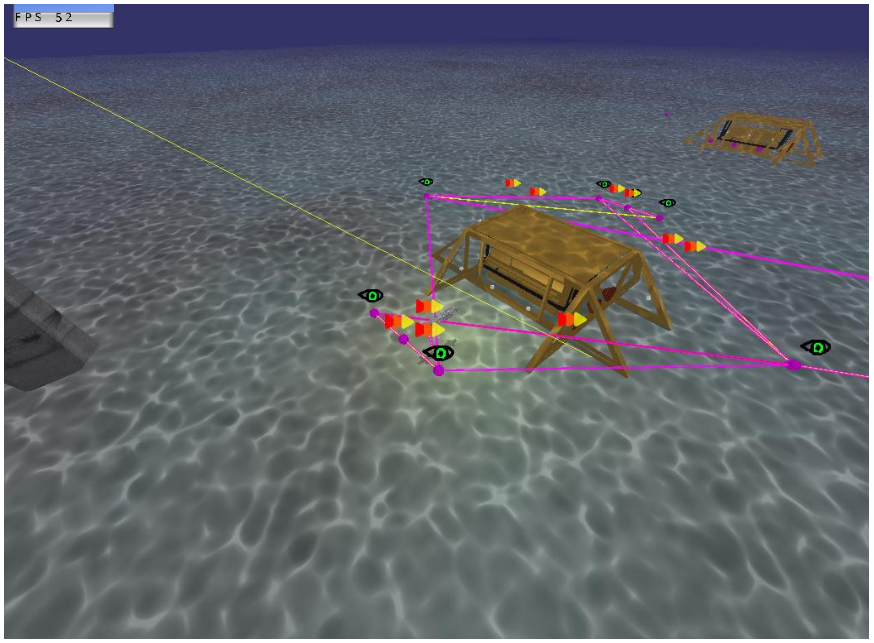
- without human intervention
- inspecting manifolds
- cleaning manifolds
- manipulation valves
- **opportunistic tasks**



Persistent Autonomy (AUVs)

Inspection and maintenance of a seabed facility:

- without human intervention
- inspecting manifolds
- cleaning manifolds
- manipulation valves
- **opportunistic tasks**

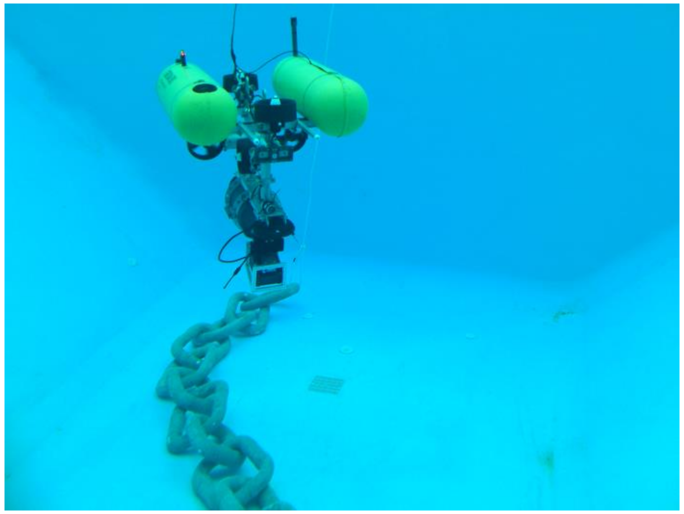


AUV mission, many tasks at scattered locations.

- long horizon plans
- large amount of uncertainty
- **discovery**

High utility, low-probability opportunities for new tasks.

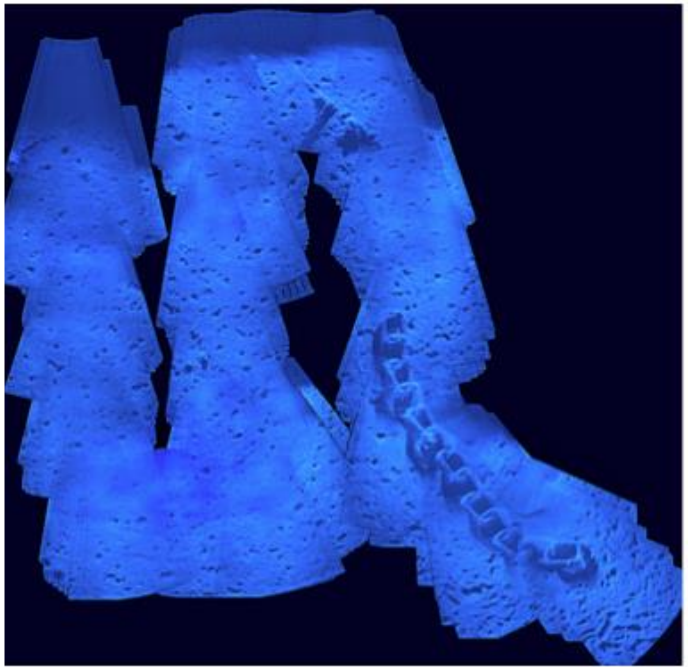
Persistent Autonomy (AUVs)



High Impact Low-Probability Events (HILPs)

- the probability distribution is unknown
- cannot be anticipated
- **our example is chain following**

If you see an unexpected chain, it's a good idea to investigate...



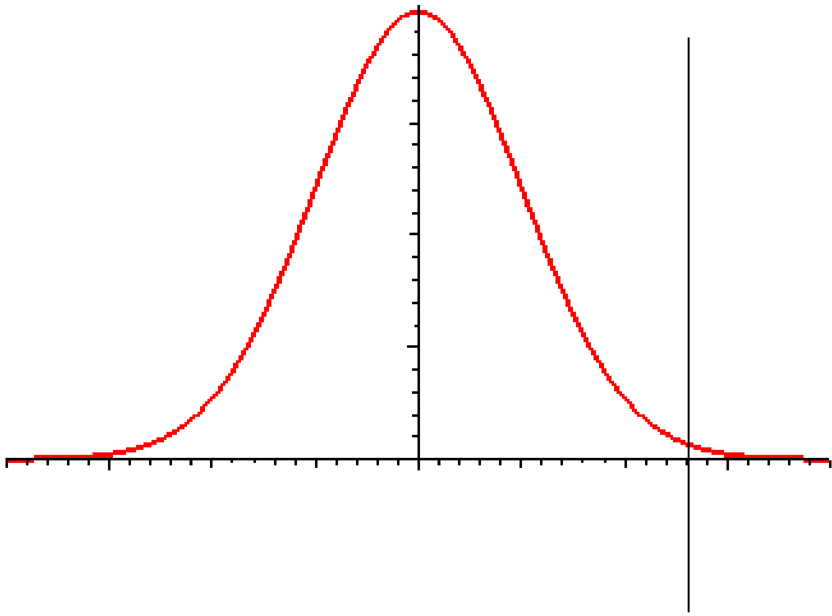
2011 Banff	5 of 10 lines parted.
2011 Volve	2 of 9 lines parted
2011 Gryphon Alpha	4 of 10 lines parted, vessel drifted a distance, riser broken
2010 Jubarte	3 lines parted between 2008 and 2010.
2009 Nan Hai Fa Xian	4 of 8 lines parted; vessel drifted a distance, riser broken
2009 Hai Yang Shi You	Entire yoke mooring column collapsed; vessel adrift, riser broken.
2006 Liuhua (N.H.S.L.)	7 of 10 lines parted; vessel drifted a distance, riser broken.
2002 Girassol buoy	3 (+2) of 9 lines parted, no damage to offloading lines (2 later)

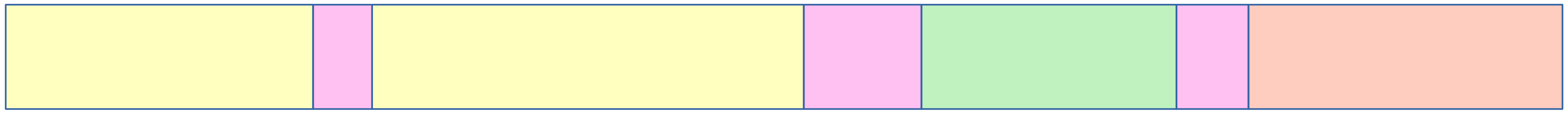
Opportunistic Planning

In PANDORA we plan and execute missions over long-term horizons (days or weeks)

Our planning strategy is based on the assumption that actions have durations normally distributed around the mean.

To build a robust plan we therefore use estimated durations for the actions that are longer than the mean.
(95th percentile of the normal distribution)



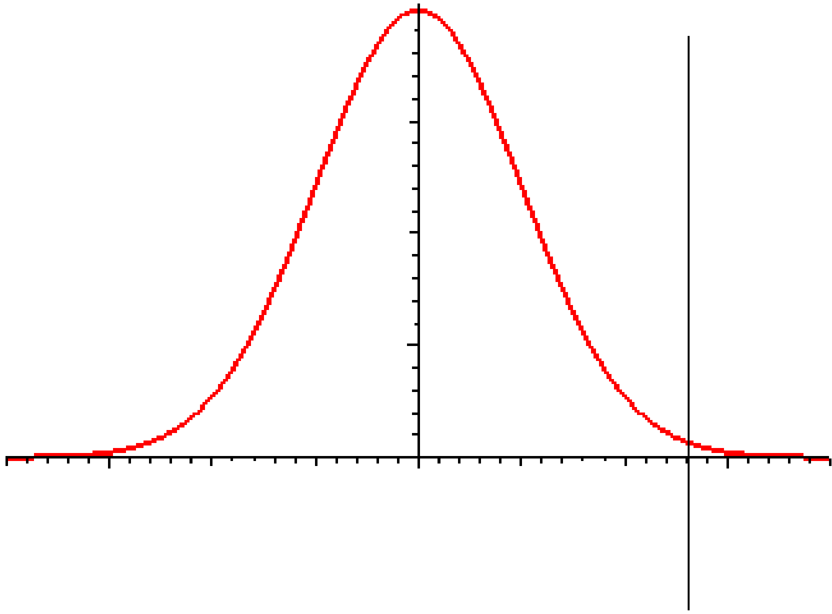


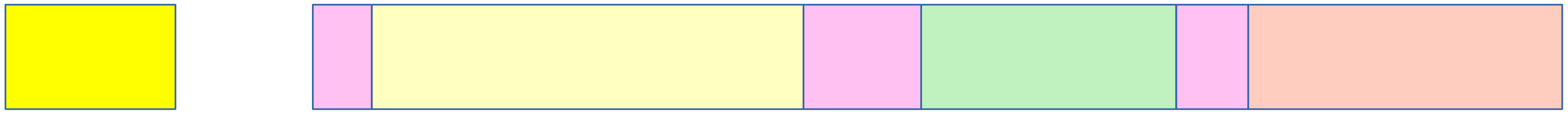
Opportunistic Planning

In PANDORA we plan and execute missions over long-term horizons (days or weeks)

Our planning strategy is based on the assumption that actions have durations normally distributed around the mean.

To build a robust plan we therefore use estimated durations for the actions that are longer than the mean.
(95th percentile of the normal distribution)



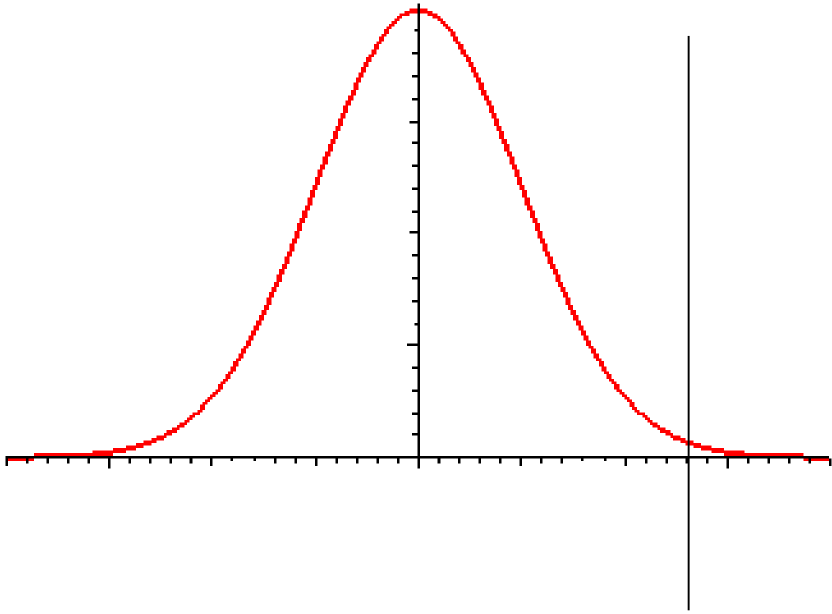


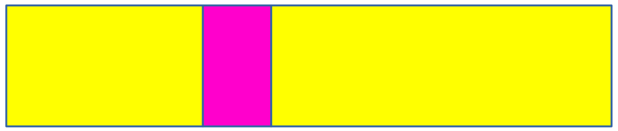
Opportunistic Planning

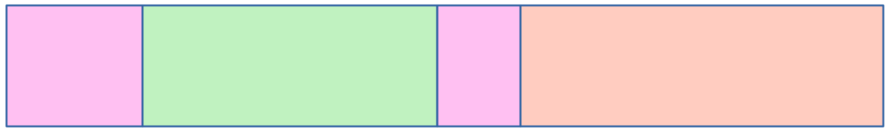
In PANDORA we plan and execute missions over long-term horizons (days or weeks)

Our planning strategy is based on the assumption that actions have durations normally distributed around the mean.

To build a robust plan we therefore use estimated durations for the actions that are longer than the mean.
(95th percentile of the normal distribution)







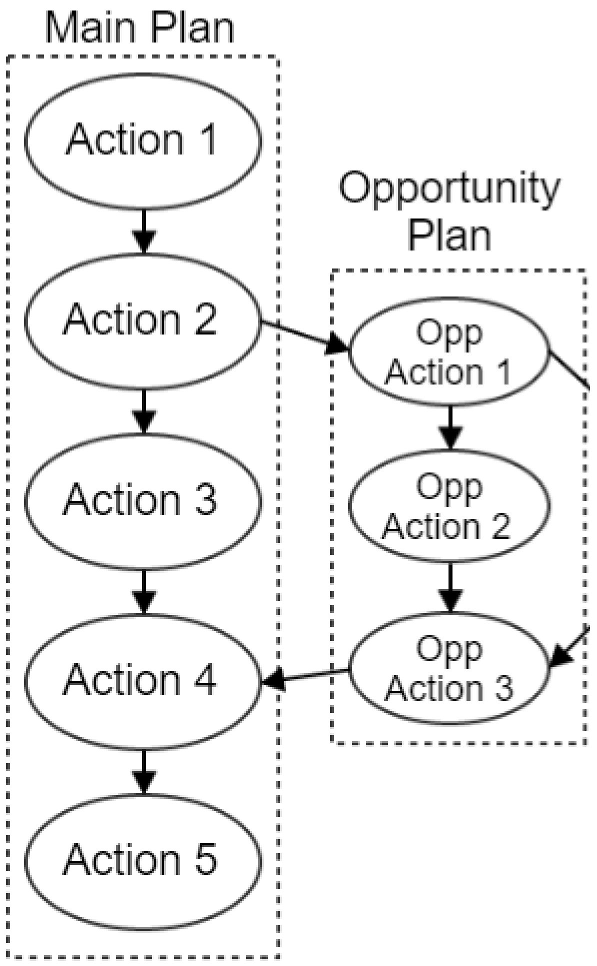
Opportunistic Planning

We use an execution stack (of goals & plans)

The current plan tail can be pushed onto the stack

New plans are generated for the opportunistic goals and the goal of returning to the tail of the current plan.

If the new plan fits inside the free time window, then it is immediately executed.



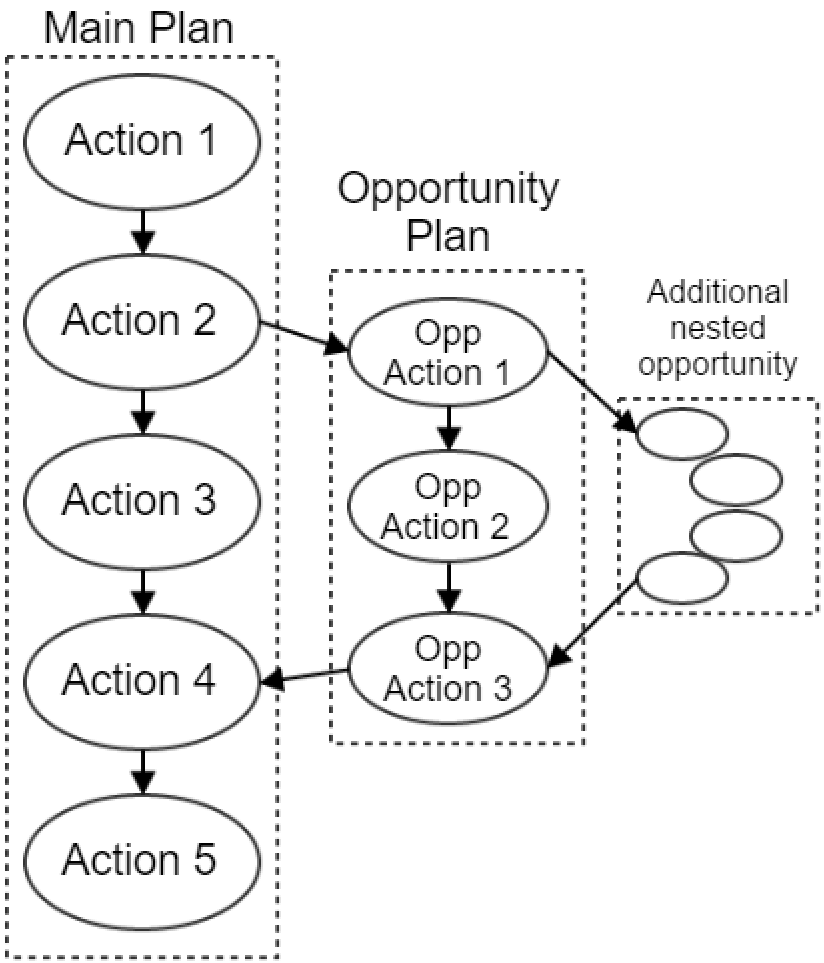
Opportunistic Planning

We use an execution stack (of goals & plans)

The current plan tail can be pushed onto the stack

New plans are generated for the opportunistic goals and the goal of returning to the tail of the current plan.

If the new plan fits inside the free time window, then it is immediately executed.



Why not just replan?

We compare the opportunistic approach against replanning the mission when an opportunity is discovered. When an opportunity is discovered a new initial state is generated.

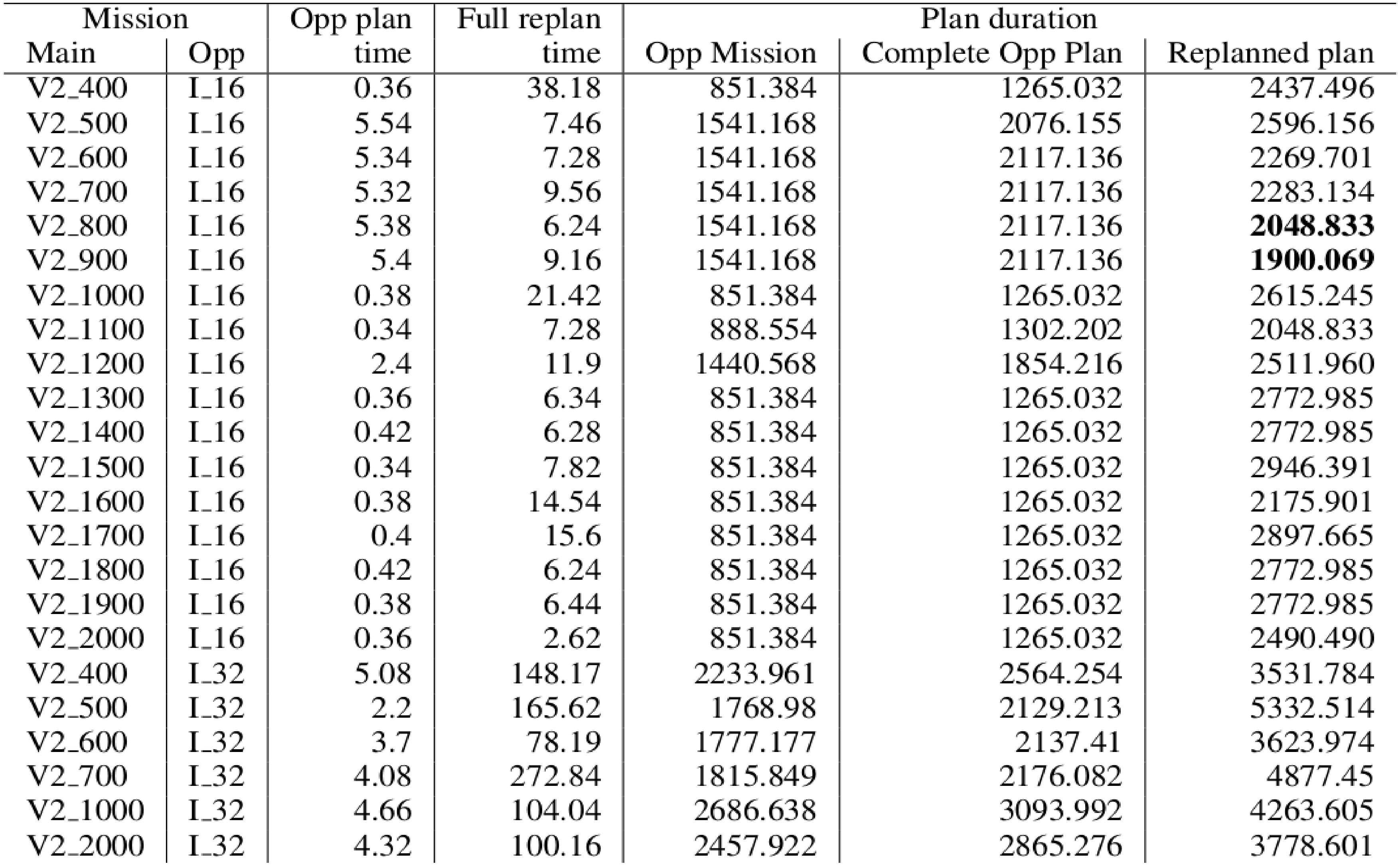
Replanning:

- the problem is more difficult to solve
- the planning time can be increased

+ the opportunity can be ordered later in the plan
+ the existing plan can be reordered to make more time for exploiting the opportunity
+ the resulting plan can be more efficient

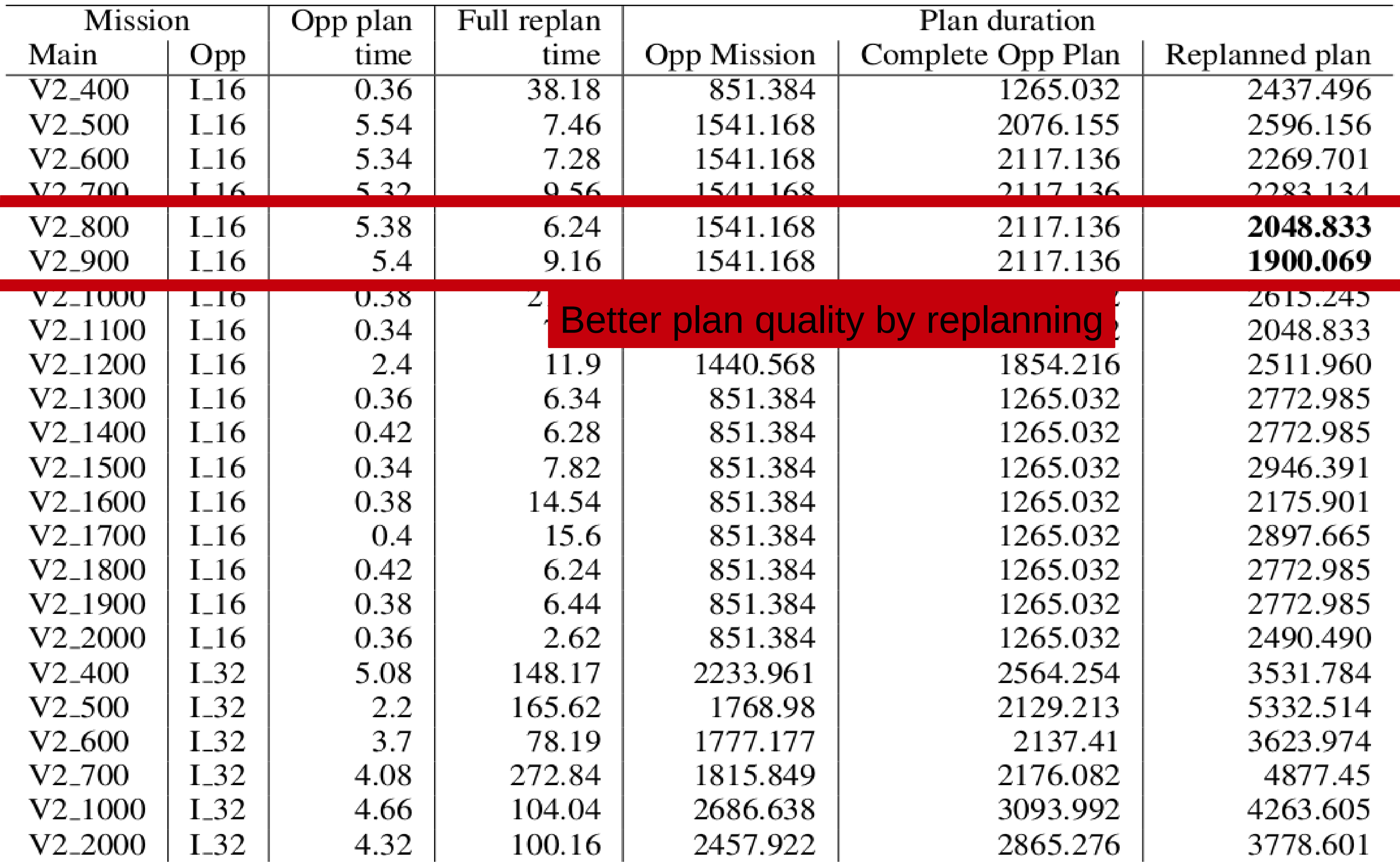
We examine situations where we have just discovered an opportunity:
10 second bound on planning for the opportunity alone
30 minute bound for replanning

Why not just replan?



Mission		Opp plan	Full replan	Plan duration		
Main	Opp	time	time	Opp Mission	Complete Opp Plan	Replanned plan
V2_400	I_16	0.36	38.18	851.384	1265.032	2437.496
V2_500	I_16	5.54	7.46	1541.168	2076.155	2596.156
V2_600	I_16	5.34	7.28	1541.168	2117.136	2269.701
V2_700	I_16	5.32	9.56	1541.168	2117.136	2283.134
V2_800	I_16	5.38	6.24	1541.168	2117.136	**2048.833**
V2_900	I_16	5.4	9.16	1541.168	2117.136	**1900.069**
V2_1000	I_16	0.38	21.42	851.384	1265.032	2615.245
V2_1100	I_16	0.34	7.28	888.554	1302.202	2048.833
V2_1200	I_16	2.4	11.9	1440.568	1854.216	2511.960
V2_1300	I_16	0.36	6.34	851.384	1265.032	2772.985
V2_1400	I_16	0.42	6.28	851.384	1265.032	2772.985
V2_1500	I_16	0.34	7.82	851.384	1265.032	2946.391
V2_1600	I_16	0.38	14.54	851.384	1265.032	2175.901
V2_1700	I_16	0.4	15.6	851.384	1265.032	2897.665
V2_1800	I_16	0.42	6.24	851.384	1265.032	2772.985
V2_1900	I_16	0.38	6.44	851.384	1265.032	2772.985
V2_2000	I_16	0.36	2.62	851.384	1265.032	2490.490
V2_400	I_32	5.08	148.17	2233.961	2564.254	3531.784
V2_500	I_32	2.2	165.62	1768.98	2129.213	5332.514
V2_600	I_32	3.7	78.19	1777.177	2137.41	3623.974
V2_700	I_32	4.08	272.84	1815.849	2176.082	4877.45
V2_1000	I_32	4.66	104.04	2686.638	3093.992	4263.605
V2_2000	I_32	4.32	100.16	2457.922	2865.276	3778.601

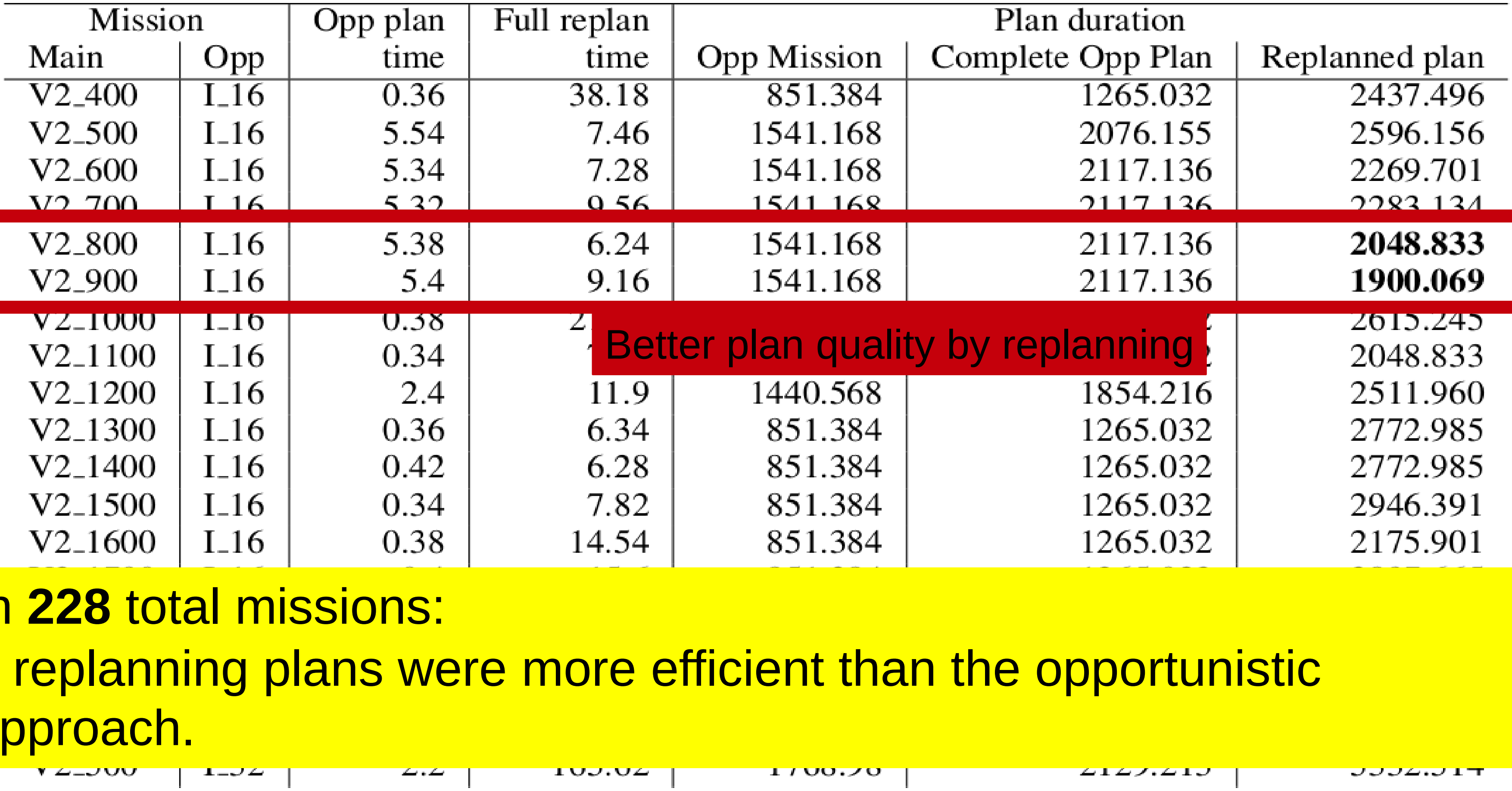
Why not just replan?



Mission		Opp plan	Full replan	Plan duration		
Main	Opp	time	time	Opp Mission	Complete Opp Plan	Replanned plan
V2_400	I_16	0.36	38.18	851.384	1265.032	2437.496
V2_500	I_16	5.54	7.46	1541.168	2076.155	2596.156
V2_600	I_16	5.34	7.28	1541.168	2117.136	2269.701
V2_700	I_16	5.32	9.56	1541.168	2117.136	2283.134
V2_800	I_16	5.38	6.24	1541.168	2117.136	**2048.833**
V2_900	I_16	5.4	9.16	1541.168	2117.136	**1900.069**
V2_1000	I_16	0.38	[illegible]	[illegible]	[illegible]	2615.245
V2_1100	I_16	0.34	[illegible]	[illegible]	[illegible]	2048.833
V2_1200	I_16	2.4	11.9	1440.568	1854.216	2511.960
V2_1300	I_16	0.36	6.34	851.384	1265.032	2772.985
V2_1400	I_16	0.42	6.28	851.384	1265.032	2772.985
V2_1500	I_16	0.34	7.82	851.384	1265.032	2946.391
V2_1600	I_16	0.38	14.54	851.384	1265.032	2175.901
V2_1700	I_16	0.4	15.6	851.384	1265.032	2897.665
V2_1800	I_16	0.42	6.24	851.384	1265.032	2772.985
V2_1900	I_16	0.38	6.44	851.384	1265.032	2772.985
V2_2000	I_16	0.36	2.62	851.384	1265.032	2490.490
V2_400	I_32	5.08	148.17	2233.961	2564.254	3531.784
V2_500	I_32	2.2	165.62	1768.98	2129.213	5332.514
V2_600	I_32	3.7	78.19	1777.177	2137.41	3623.974
V2_700	I_32	4.08	272.84	1815.849	2176.082	4877.45
V2_1000	I_32	4.66	104.04	2686.638	3093.992	4263.605
V2_2000	I_32	4.32	100.16	2457.922	2865.276	3778.601

Better plan quality by replanning

Why not just replan?



Mission		Opp plan	Full replan	Plan duration		
Main	Opp	time	time	Opp Mission	Complete Opp Plan	Replanned plan
V2_400	I_16	0.36	38.18	851.384	1265.032	2437.496
V2_500	I_16	5.54	7.46	1541.168	2076.155	2596.156
V2_600	I_16	5.34	7.28	1541.168	2117.136	2269.701
V2_700	I_16	5.32	9.56	1541.168	2117.136	2283.134
V2_800	I_16	5.38	6.24	1541.168	2117.136	**2048.833**
V2_900	I_16	5.4	9.16	1541.168	2117.136	**1900.069**
V2_1000	I_16	0.38	[illegible]	[illegible]	[illegible]	2615.245
V2_1100	I_16	0.34	[illegible]	[illegible]	[illegible]	2048.833
V2_1200	I_16	2.4	11.9	1440.568	1854.216	2511.960
V2_1300	I_16	0.36	6.34	851.384	1265.032	2772.985
V2_1400	I_16	0.42	6.28	851.384	1265.032	2772.985
V2_1500	I_16	0.34	7.82	851.384	1265.032	2946.391
V2_1600	I_16	0.38	14.54	851.384	1265.032	2175.901

Better plan quality by replanning

In **228** total missions:
5 replanning plans were more efficient than the opportunistic approach.

We examine situations where we have just discovered an opportunity:
10 second bound on planning for the opportunity alone
30 minute bound for replanning

Opportunistic Planning

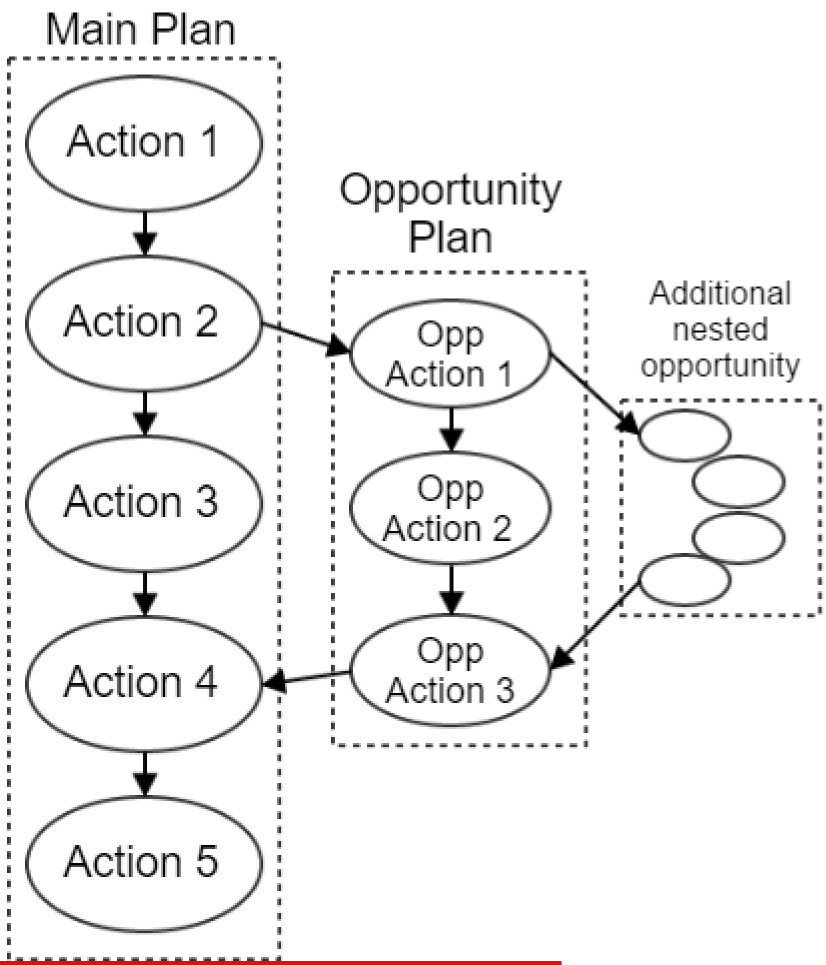
We use an execution stack (of goals & plans)

The current plan tail can be pushed onto the stack

New plans are generated for the opportunistic goals and the goal of returning to the tail of the current plan.

If the new plan fits inside the free time window, then it is immediately executed.

NOTE: Opportunities can also arise for supervisor requests!



More details on Friday morning
(Paper on Opportunistic Planning at the Journal Track)

Outline

- Why PDDL Planning for Robotics and HRI?
    - Expressive Planning
    - Opportunistic Planning
    - Strategic Planning
    - eXplainable Planning (XAIP)
    - Planning with Uncertainty

Strategic Planning for Persistent Autonomy

Planning over long horizons (days, weeks)

Missions with strict deadlines and time windows in which goals need to be accomplished.

Example in underwater robotics:
Seabed facilities need to be inspected at certain intervals.

Current planning systems struggle in generating complex plans over long horizons.

One possible solution:
Decompose into **Strategic/Tactical Layers**

Strategic/Tactical Planning

Cluster the *goals* into *tasks*

Strategic Layer: contains a high lever plan that achieves all tasks and manages the <u>resource </u>and <u>time constraints</u>.

Tactical Layer: contains a plan that solves a single task.

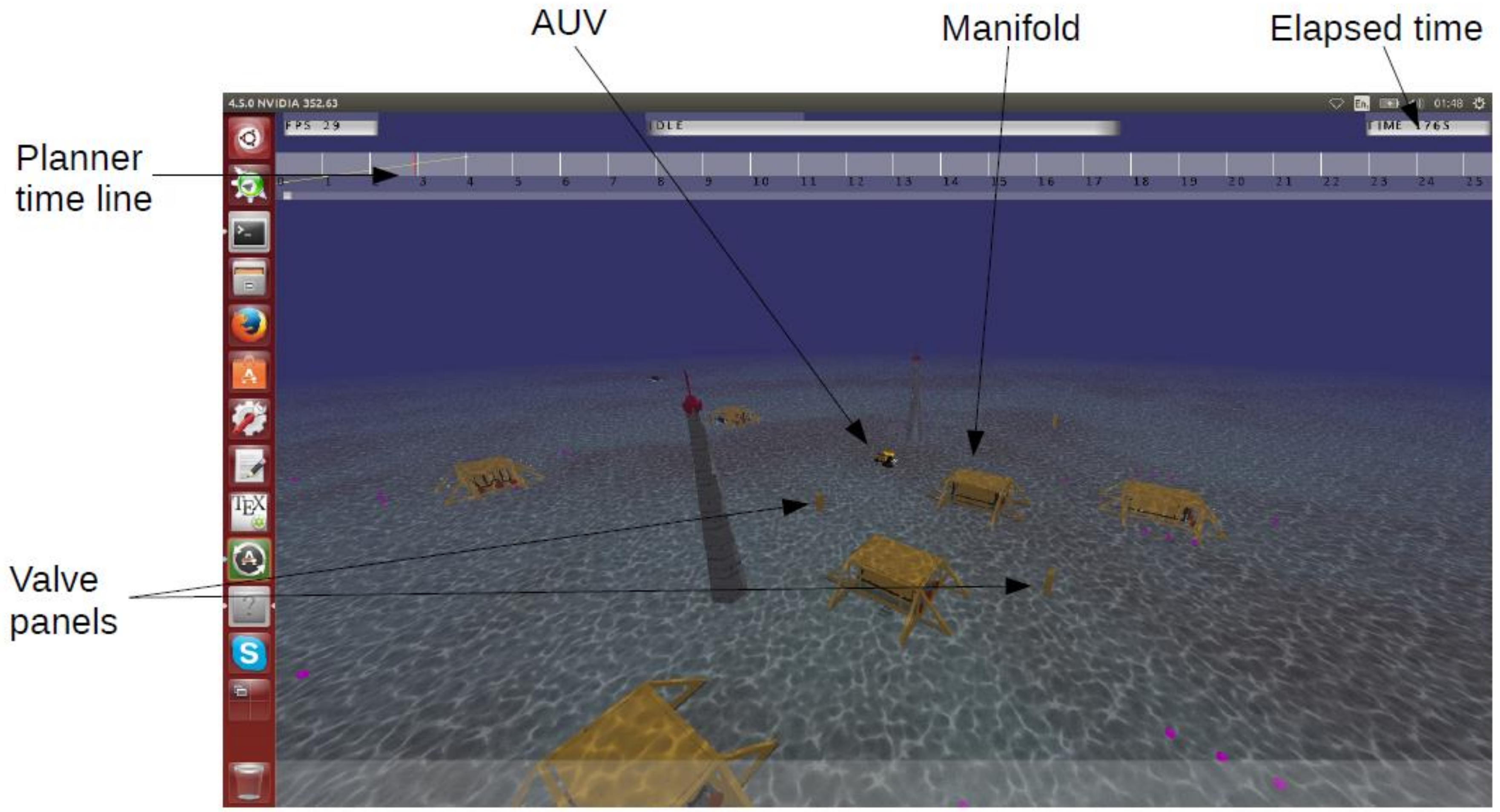
Example from underwater robotics.

Long term maintenance of seabed facility includes

-Inspecting the structures are regular intervals.
-Changing the configuration of the site by interacting with interfaces within specific time windows.
-Recharging the AUVs.

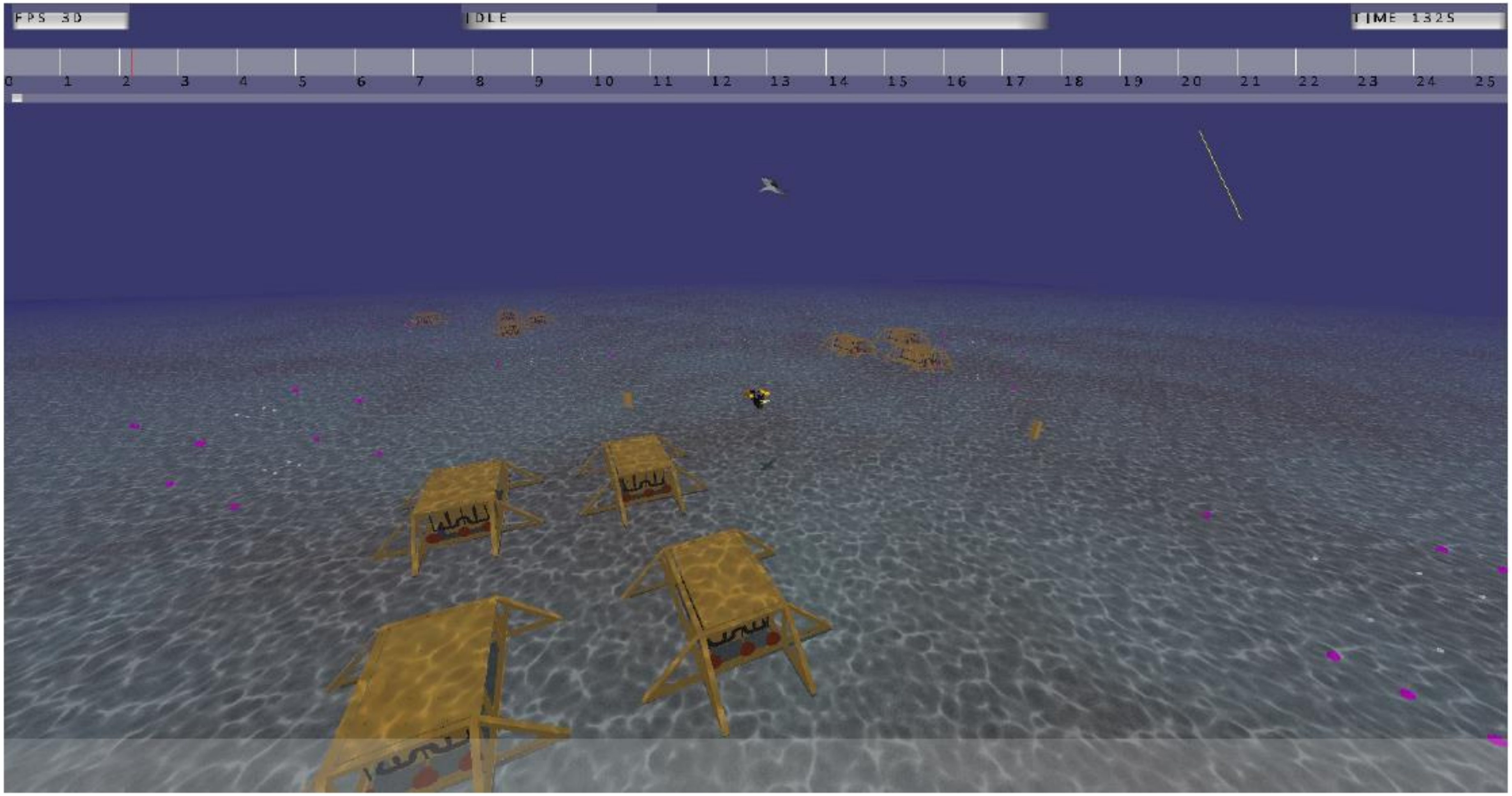
Additional challenges:
-Ever changing environment (currents, visibility)
-Wildlife

Strategic/Tactical Planning

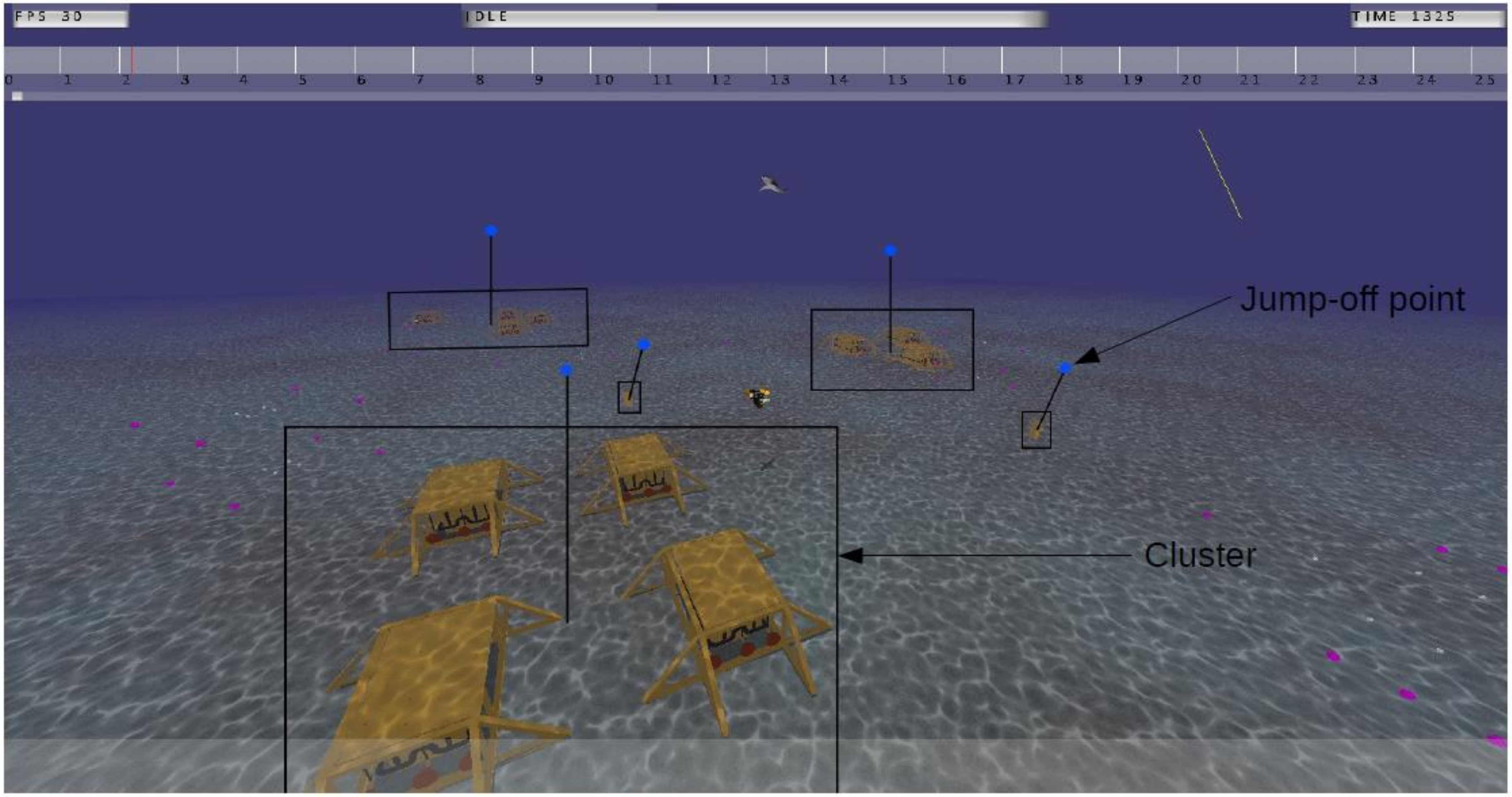


Strategic/Tactical Planning

Clustering



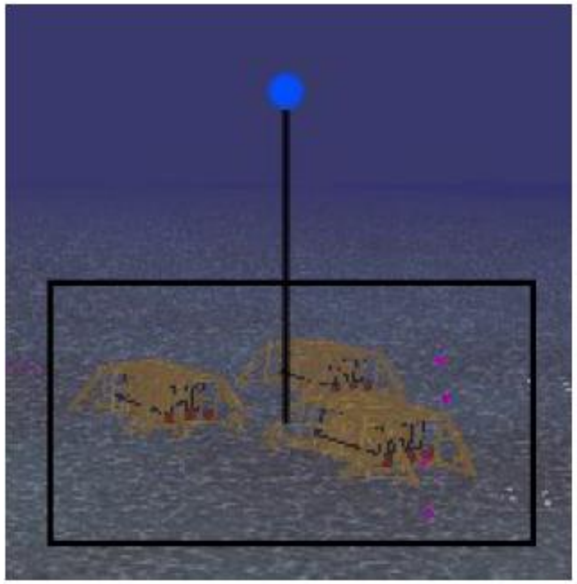
Strategic/Tactical Planning
Clustering



Strategic/Tactical Planning

Tactical Layer

For each Task the planner generates a plan and stores:
-duration
-resource constraints



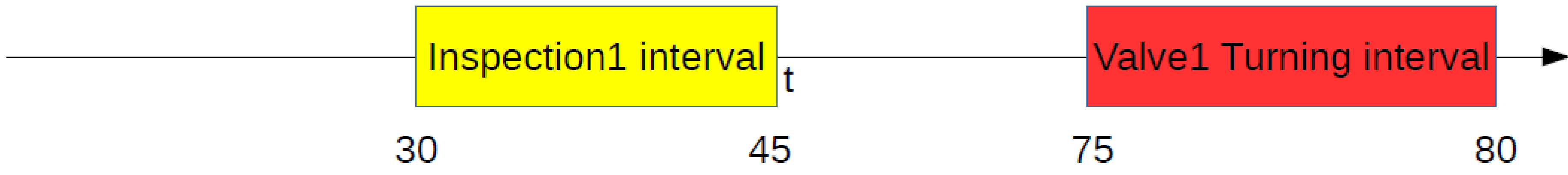
```
0.000: (correct_position auv0 wp_auv0) [3.000]
3.001: (do_hover_fast auv0 wp_auv0 strategic_location_7)
[11.403]
14.405: (correct_position auv0_strategic_location_78)
[3.000]
17.406: (observe_inspection_point auv0 strategic_location_7
inspection_point_2) [10.000]
27.407: (correct_position auv0 strategic_location_7)
[3.000]
45.083: (do_hover_controlled auv0 strategic_location_5
strategic_location_5) [4.000]
49.084: (observe_inspecetion_point auv0
strategic_location_5 inspection_point_4) [10.000]
...
```

Energy consumption = 10W
Duration = 86.43s

Strategic/Tactical Planning

Strategic Layer

On the strategic layer the planner constructs a plan that conforms to the time and resource constraints.

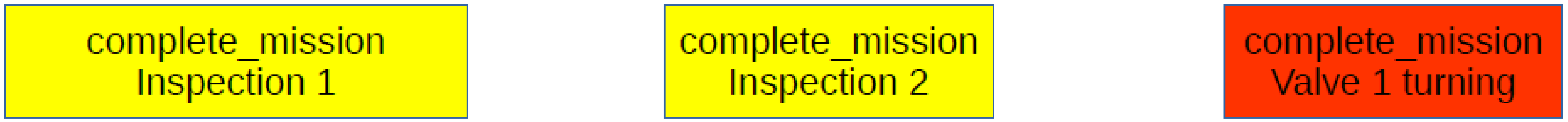


Strategic/Tactical Planning

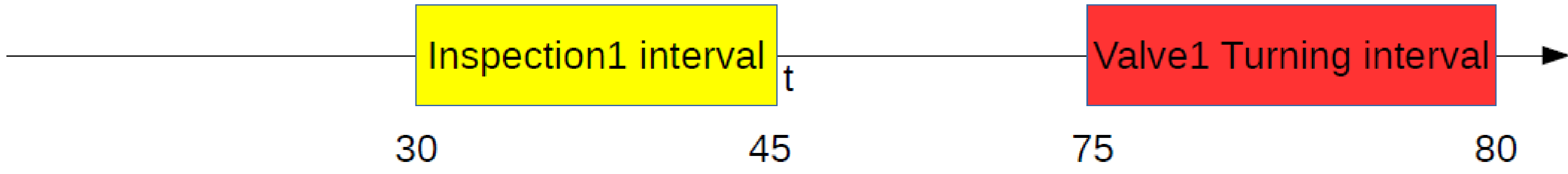
Strategic Layer

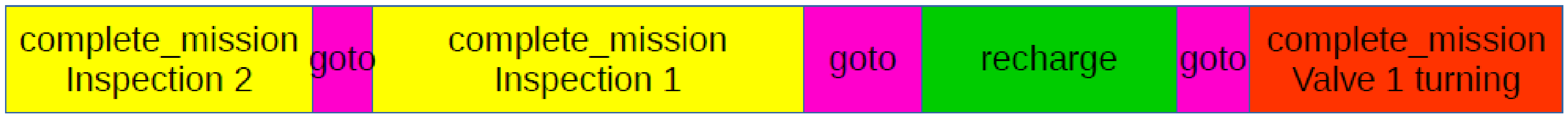
On the strategic layer the planner constructs a plan that conforms to the time and resource constraints.

All the tactical plans are collected.

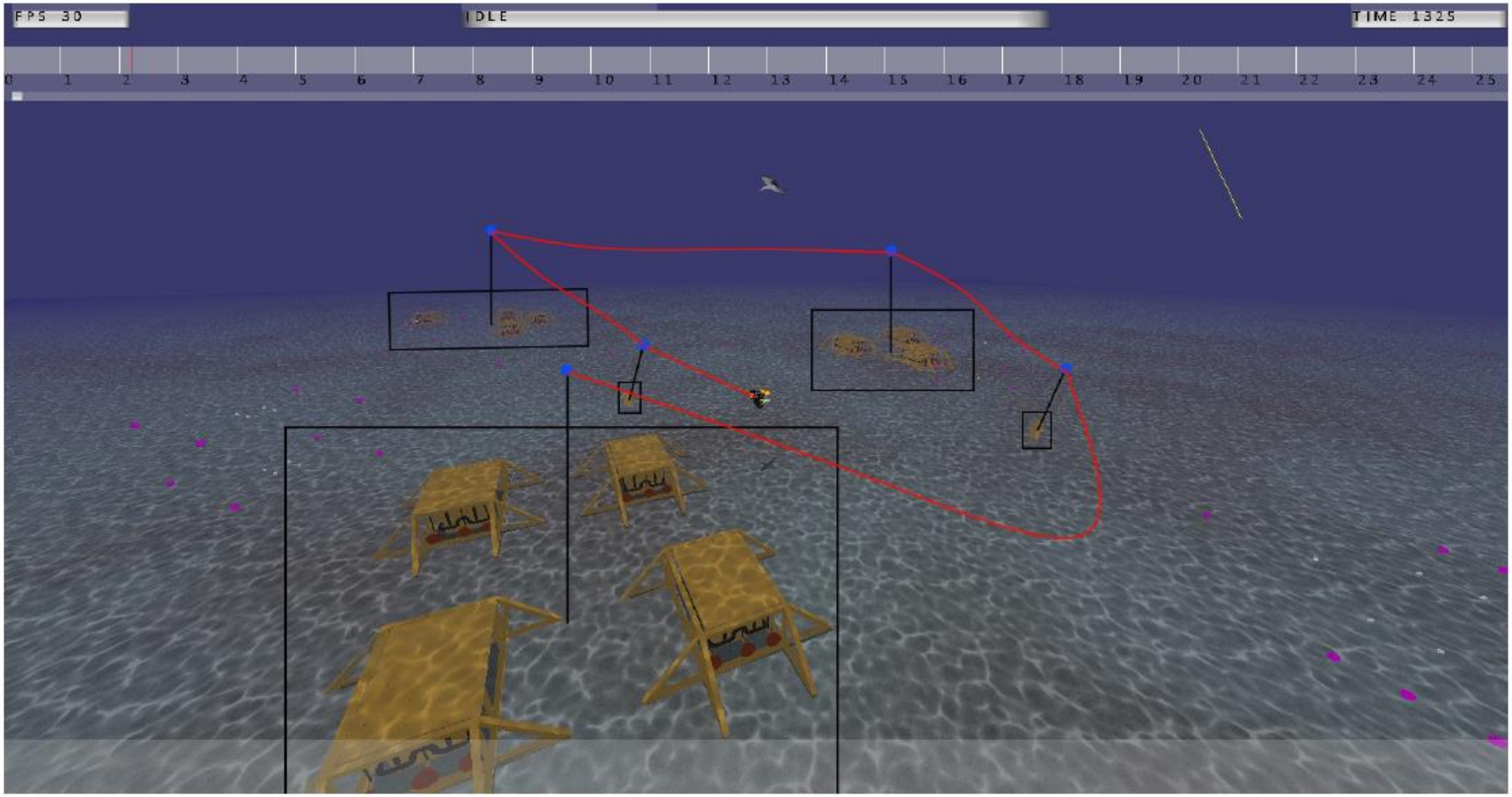

complete_mission Inspection 1
complete_mission Inspection 2
complete_mission Valve 1 turning

And the strategic plan is generated, not violating resource/time constraints




complete_mission Inspection 2
goto
complete_mission Inspection 1
goto
recharge
goto
complete_mission Valve 1 turning

Strategic/Tactical Planning



Outline

- Why PDDL Planning for Robotics and HRI?
  - Expressive Planning
  - Opportunistic Planning
  - Strategic Planning
  - eXplainable Planning (XAIP)
  - Planning with Uncertainty

eXplainable Planning (XAIP)

Planners can be trusted

Planners can allow an easy interaction with humans

Planners are transparent
(at least, the process by which the decisions are made are understood by their programmers)

To note: entirely trustworthy and theoretically well-understood algorithms can still yield decisions that are hard to explain.
Ex: Linear Programming ….

To note: XAI and the need to explain machine/deep learning remain of critical importance!
XAIP is important in domains where learning is not an option.

What eXplainable Planning is NOT !

XAIP is **not** explaining what is **obvious** !

Many planners select actions in their plan-construction process by minimising a heuristic distance to goal (relaxed plan)

Q: *Why did the planner do that* ?

A: *Because it got me closer to the goal* !

What eXplainable Planning is NOT !

XAIP is **not** explaining what is **obvious** !

Many planners select actions in their plan-construction process by minimising a heuristic distance to goal (relaxed plan)

Q: *Why did the planner do that ?*

~~A: *Because it got me closer to the goal* !~~

What eXplainable Planning is NOT !

XAIP is **not** explaining what is **obvious** !

Many planners select actions in their plan-construction process by minimising a heuristic distance to goal (relaxed plan)

Q: *Why did the planner do that ?*

~~A: *Because it got me closer to the goal* !~~

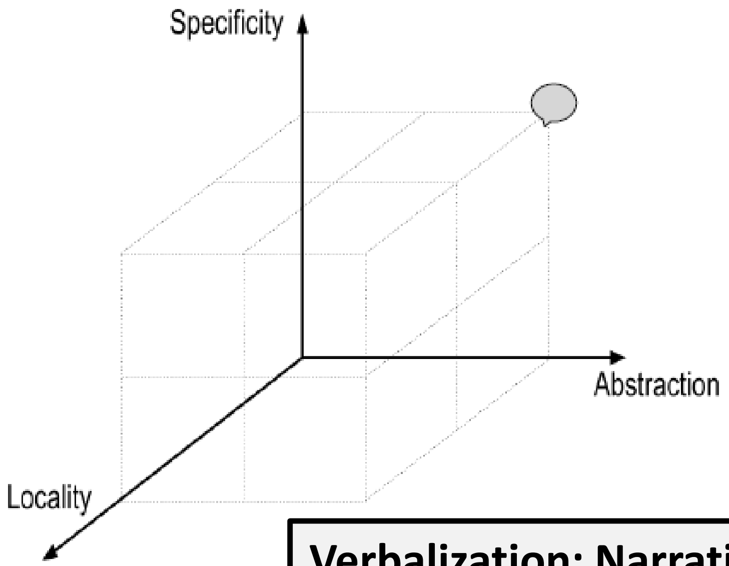
A request for an explanation is an attempt to uncover a piece of knowledge that the questioner believes must be available to the system and that the questioner does not have.

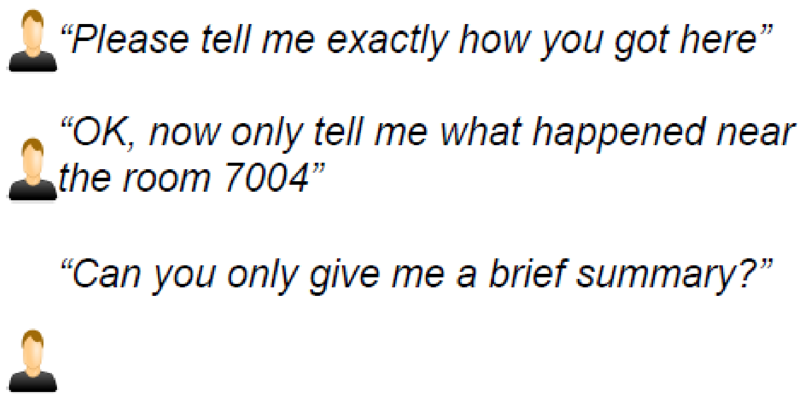
Towards XAIP

- Plan explanation
  - Translate PDDL in forms that humans can understand [Sohrabi et al. 2012]
  - Design interfaces that help this understanding [Bidot et al. 2012]
  - Describe causal/temporal relations for plan steps [Seegebarth et al. 2012]
  - Explaining observed behaviours [Sohrabi, Baier, McIlraith, 2011]
  - Understanding the past [Molineaux et al., 2012]
  - … ... …
- Plan Explicability
  - Focus on human's interpretation of plans [Seegebarth et al. 2012]
- Verbalization and *transparency* in autonomy
  - Generate narrations for autonomous robot navigations [Veloso et al. 2016]
- Explainable Agency [Langley et al. 2017]
- Model Reconciliation (Sreedharan et al.)
  - Identify/reconcile different human/robot models [Chakraborti et al 2017]

Transparency in Autonomy
(Manuela Veloso et al.)

Verbalization: the process by which an autonomous robots converts its own experience into language
Verbalization space: to capture different nature of explanations.
And to learn to correctly infer an explanation level in the verbalization space.
Specificity – Locality - Abstraction




"Please tell me exactly how you got here"
"OK, now only tell me what happened near the room 7004"
"Can you only give me a brief summary?"

Verbalization: Narration of Autonomous Mobile Robot Experience.
Rosenthal, Selvaraj, Veloso. IJCAI 2016.

Things to Be Explained
(*some*)

- Q1: Why did you do that?
- Q2: Why didn't you do *something else*? (that I would have done)
- Q3: Why is what you propose to do more efficient/safe/cheap than something else? (that I would have done)
- Q4: Why can't you do that ?
- Q5: Why do I need to replan at this point?
- Q6: Why do I not need to replan at this point?

Illustrative Example

`Rover Time` domain from IPC-4 (problem 3)

```
0.000: (navigate r1 wp3 wp0)   [5.0]
0.000: (navigate r0 wp1 wp0)   [5.0]
5.001: (calibrate r1 camera1 obj0 wp0)   [5.0]
5.001: (sample_rock r0 r0store wp0)   [8.0]
10.002: (take_image r1 wp0 obj0 camera1 col)   [7.0]
13.001: (navigate r0 wp0 wp1)   [5.0]
17.002: (navigate r1 wp0 wp3)   [5.0]
18.001: (comm_rock_data r0 general wp0 wp1 wp0)   [10.0]
22.003: (navigate r1 wp3 wp2)   [5.0]
27.003: (sample_soil r1 r1store wp2)   [10.0]
28.002: (comm_image_data r1 general obj0 col wp2 wp0) [15.0]
43.003: (comm_soil_data r1 general wp2 wp2 wp0)   [10.0]

[Duration = 53.003]
```

Q1: *why did you use Rover0 to take the rock sample at waypoint0 ?*

NA: *so that I can communicate_data from Rover0 later (at 18.001)*

Illustrative Example

`Rover Time` domain from IPC-4 (problem 3)

```
0.000: (navigate r1 wp3 wp0)  [5.0]
0.000: (navigate r0 wp1 wp0)  [5.0]
5.001: (calibrate r1 camera1 obj0 wp0)  [5.0]
5.001: (sample_rock r0 r0store wp0)  [8.0]
10.002: (take_image r1 wp0 obj0 camera1 col)  [7.0]
13.001: (navigate r0 wp0 wp1)  [5.0]
17.002: (navigate r1 wp0 wp3)  [5.0]
18.001: (comm_rock_data r0 general wp0 wp1 wp0)  [10.0]
22.003: (navigate r1 wp3 wp2)  [5.0]
27.003: (sample_soil r1 r1store wp2)  [10.0]
28.002: (comm_image_data r1 general obj0 col wp2 wp0) [15.0]
43.003: (comm_soil_data r1 general wp2 wp2 wp0)  [10.0]

[Duration = 53.003]
```

Q1: *why did you use Rover0 to take the rock sample at waypoint0 ?*

NA: *so that I can communicate_data from Rover0 later (at 18.001)*

Illustrative Example

`Rover Time` domain from IPC-4 (problem 3)

```
0.000: (navigate r1 wp3 wp0)   [5.0]
0.000: (navigate r0 wp1 wp0)   [5.0]
5.001: (calibrate r1 camera1 obj0 wp0)   [5.0]
5.001: (sample_rock r0 r0store wp0)   [8.0]
10.002: (take_image r1 wp0 obj0 camera1 col)   [7.0]
13.001: (navigate r0 wp0 wp1)   [5.0]
17.002: (navigate r1 wp0 wp3)   [5.0]
18.001: (comm_rock_data r0 general wp0 wp1 wp0)   [10.0]
22.003: (navigate r1 wp3 wp2)   [5.0]
27.003: (sample_soil r1 r1store wp2)   [10.0]
28.002: (comm_image_data r1 general obj0 col wp2 wp0) [15.0]
43.003: (comm_soil_data r1 general wp2 wp2 wp0)   [10.0]

[Duration = 53.003]
```

Q1: *why did you use Rover0 to take the rock sample at waypoint0* ?

why didn't Rover1 take the rock sample at waypoint0 ?

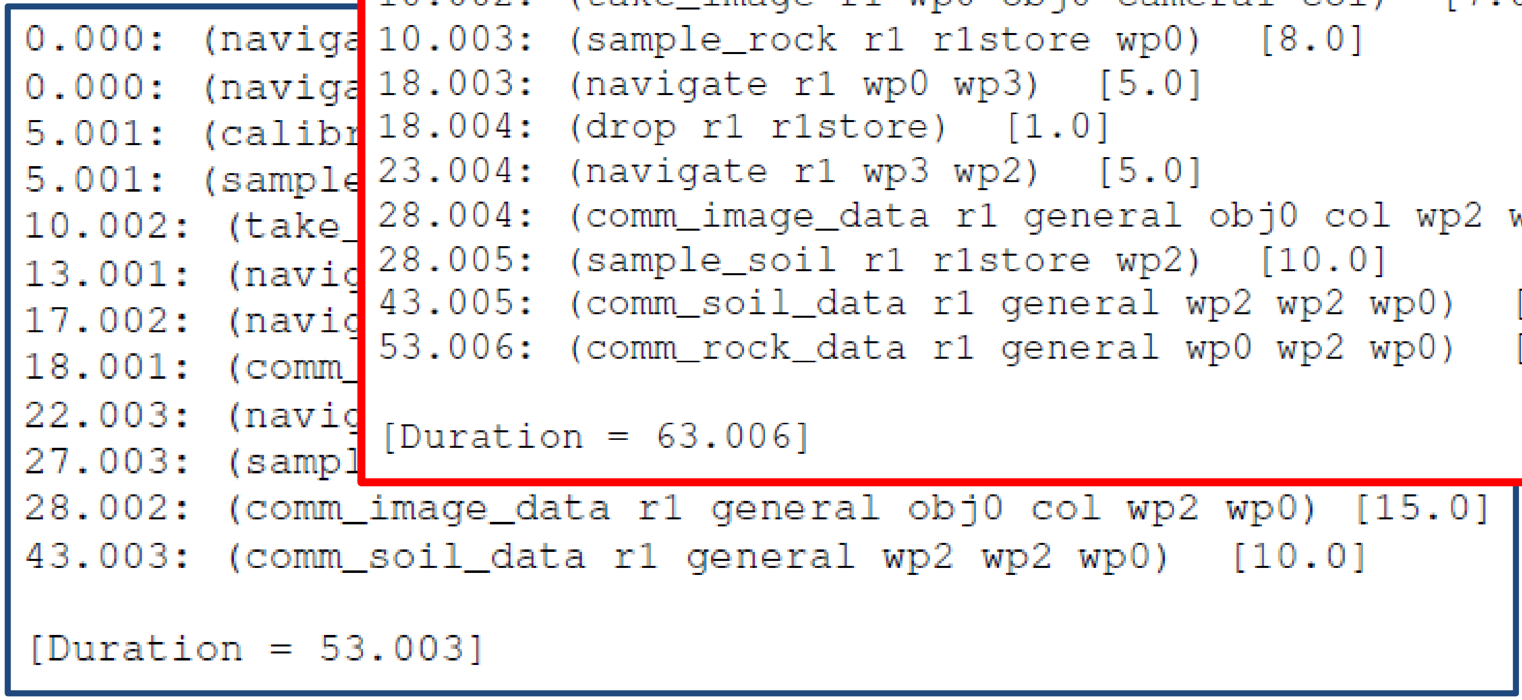
Illustrative Example

Q1: *why did you use Rover0 to take the rock sample at waypoint0 ?*

why didn’t Rover1 take the rock sample at waypoint0 ?

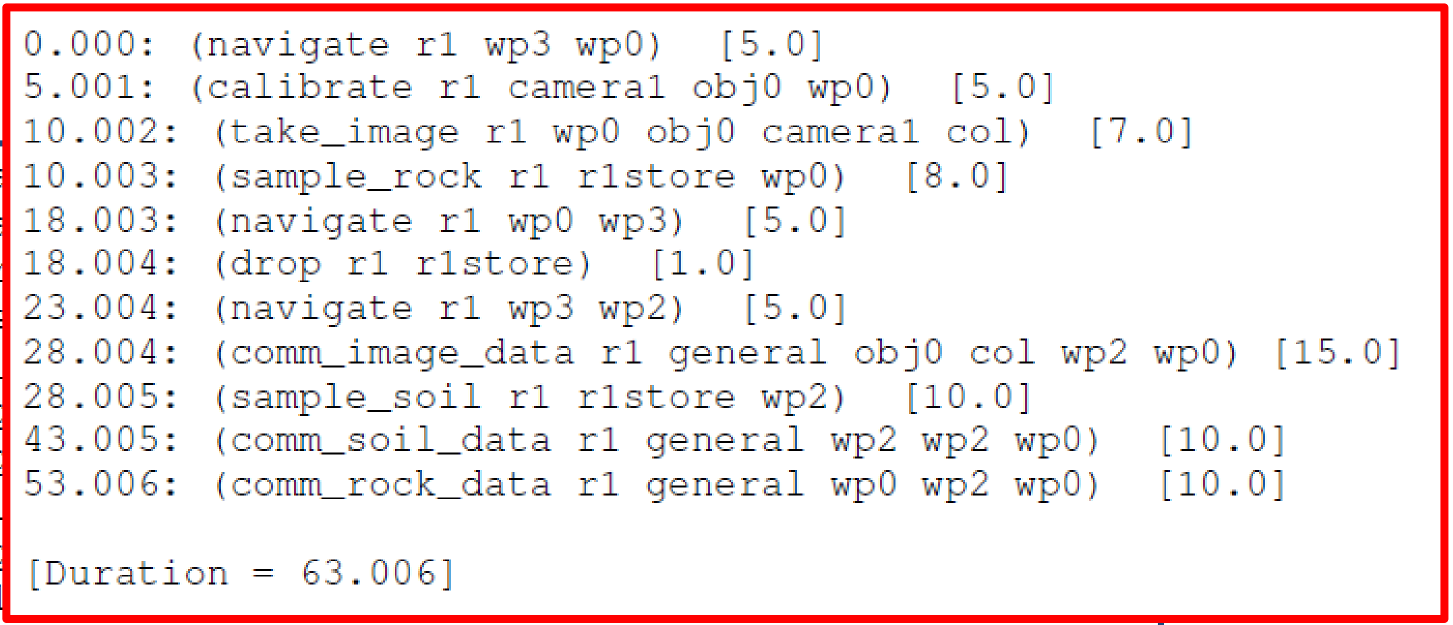
We remove the ground action instance for Rover0 and re-plan

A: *Because not using Rover0 for this action leads to a longer plan*



```
0.000:  (naviga
0.000:  (naviga
5.001:  (calibr
5.001:  (sample
10.002:  (take_
13.001:  (navig
17.002:  (navig
18.001:  (comm_
22.003:  (navig
27.003:  (sampl
28.002:  (comm_image_data r1 general obj0 col wp2 wp0)  [15.0]
43.003:  (comm_soil_data r1 general wp2 wp2 wp0)   [10.0]

[Duration = 53.003]
```



```
0.000:  (navigate r1 wp3 wp0)   [5.0]
5.001:  (calibrate r1 camera1 obj0 wp0)   [5.0]
10.002:  (take_image r1 wp0 obj0 camera1 col)   [7.0]
10.003:  (sample_rock r1 r1store wp0)   [8.0]
18.003:  (navigate r1 wp0 wp3)   [5.0]
18.004:  (drop r1 r1store)   [1.0]
23.004:  (navigate r1 wp3 wp2)   [5.0]
28.004:  (comm_image_data r1 general obj0 col wp2 wp0)  [15.0]
28.005:  (sample_soil r1 r1store wp2)   [10.0]
43.005:  (comm_soil_data r1 general wp2 wp2 wp0)   [10.0]
53.006:  (comm_rock_data r1 general wp0 wp2 wp0)   [10.0]

[Duration = 63.006]
```

Illustrative Example

Q1: *why did you use Rover0 to take the rock sample at waypoint0 ?*

why didn't Rover1 take the rock sample at waypoint0 ?

We remove the ground action instance for Rover0 and re-plan

A: *Because not using Rover0 for this action leads to a longer plan*

Q2: But why does Rover1 do everything in this plan?

```
0.000: (navigate r1 wp3 wp0)  [5.0]
5.001: (calibrate r1 camera1 obj0 wp0)  [5.0]
10.002: (take_image r1 wp0 obj0 camera1 col)  [7.0]
10.003: (sample_rock r1 r1store wp0)  [8.0]
18.003: (navigate r1 wp0 wp3)  [5.0]
18.004: (drop r1 r1store)  [1.0]
23.004: (navigate r1 wp3 wp2)  [5.0]
28.004: (comm_image_data r1 general obj0 col wp2 wp0) [15.0]
28.005: (sample_soil r1 r1store wp2)  [10.0]
43.005: (comm_soil_data r1 general wp2 wp2 wp0)  [10.0]
53.006: (comm_rock_data r1 general wp0 wp2 wp0)  [10.0]

[Duration = 63.006]
```

Illustrative Example

Q1: *why did you use Rover0 to take the rock sample at waypoint0 ?*

why didn't Rover1 take the rock sample at waypoint0 ?

We remove the ground action instance for Rover0 and re-plan

A: *Because not using Rover0 for this action leads to a longer plan*

Q2: But why does Rover1 do everything in this plan?

We require the plan to contain at least one action that has Rover0 as argument (add dummy effect to all actions using Rover0 and put into the goal)

```
0.000:  (na
5.001:  (ca
10.002:  (t
10.003:  (s
18.003:  (r
18.004:  (d
23.004:  (r
28.004:  (c
28.005:  (s
43.005:  (c
53.006:  (c

[Duration
```

```
0.000:  (navigate r0 wp1 wp0)   [5.0]
0.000:  (navigate r1 wp3 wp0)   [5.0]
5.001:  (calibrate r1 camera1 obj0 wp0)   [5.0]
10.002:  (take_image r1 wp0 obj0 camera1 col)   [7.0]
10.003:  (sample_rock r1 r1store wp0)   [8.0]
18.003:  (navigate r1 wp0 wp3)   [5.0]
18.004:  (drop r1 r1store)   [1.0]
23.004:  (navigate r1 wp3 wp2)   [5.0]
28.004:  (comm_image_data r1 general obj0 col wp2 wp0)   [15.0]
28.005:  (sample_soil r1 r1store wp2)   [10.0]
43.005:  (comm_soil_data r1 general wp2 wp2 wp0)   [10.0]
53.006:  (comm_rock_data r1 general wp0 wp2 wp0)   [10.0]
```

Illustrative Example

Q1: *why did you use Rover0 to take the rock sample at waypoint0 ?*

why didn't Rover1 take the rock sample at waypoint0 ?

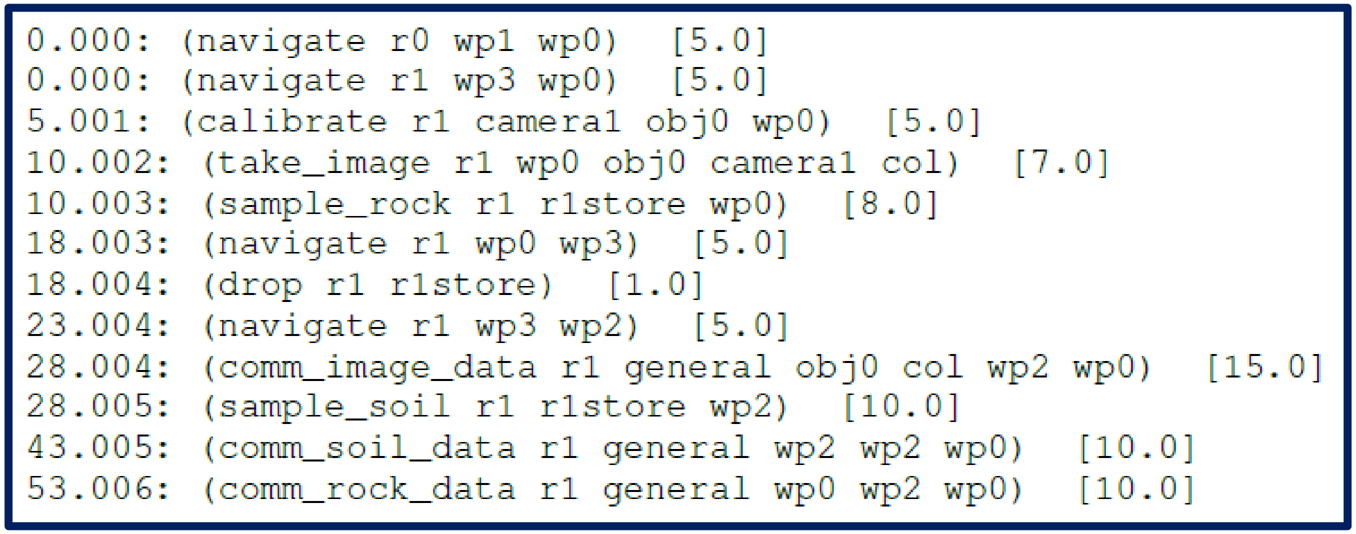
We remove the ground action instance for Rover0 and re-plan

A: *Because not using Rover0 for this action leads to a longer plan*

Q2: But why does Rover1 do everything in this plan?

We require the plan to contain at least one action that has Rover0 as argument (add dummy effect to all actions using Rover0 and put into the goal)

A: *There is no useful way to use Rover0 for improve this plan*



```
0.000: (navigate r0 wp1 wp0)   [5.0]
0.000: (navigate r1 wp3 wp0)   [5.0]
5.001: (calibrate r1 camera1 obj0 wp0)   [5.0]
10.002: (take_image r1 wp0 obj0 camera1 col)   [7.0]
10.003: (sample_rock r1 r1store wp0)   [8.0]
18.003: (navigate r1 wp0 wp3)   [5.0]
18.004: (drop r1 r1store)   [1.0]
23.004: (navigate r1 wp3 wp2)   [5.0]
28.004: (comm_image_data r1 general obj0 col wp2 wp0)   [15.0]
28.005: (sample_soil r1 r1store wp2)   [10.0]
43.005: (comm_soil_data r1 general wp2 wp2 wp0)   [10.0]
53.006: (comm_rock_data r1 general wp0 wp2 wp0)   [10.0]
```

eXplainable Planning
at execution time

- Q5: Why do I need to replan at this point?

In many real-world scenarios, it is not obvious that the plan being executed will fail. Often plain failures is discovered too late.

One possible approach is to use the “*Filter Violation*” (ROSPlan)

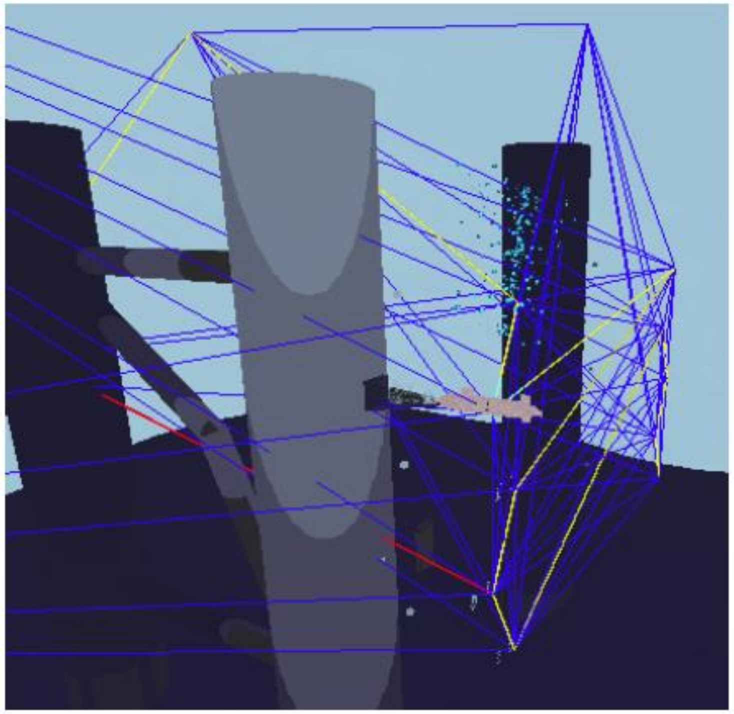
Once the plan is generated, ROSPlan creates a filter, by considering all the preconditions of the actions in the plan.

Ex: navigate (?from ?to - waypoint) has precondition (connected ?from ?to)
If the plan contains navigate (wp3 wp5),
then (connected wp3 wp5) is added to the filter.

Illustrative Example

AUV domain from (Cashmore et al, ICRA 2015)

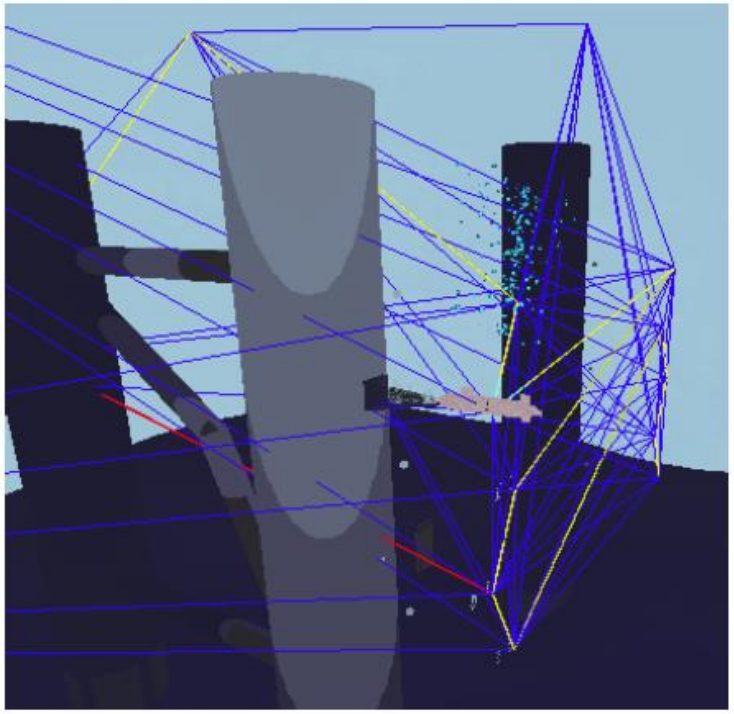
```
0.000: (observe auv wp1 ip3)  [10.000]
10.001: (correct_position auv wp1)  [10.000]
20.002: (do_hover auv wp1 wp2)  [71.696]
91.699: (observe auv wp2 ip4)  [10.000]
101.700: (correct_position auv wp2)  [10.000]
111.701: (do_hover auv wp2 wp23)  [16.710]
128.412: (observe auv wp23 ip5)  [10.000]
138.413: (correct_position auv wp23)  [10.000]
148.414: (observe auv wp23 ip1)  [10.000]
158.415: (correct_position auv wp23)  [10.000]
168.416: (do_hover auv wp23 wp22)  [16.710]
185.127: (do_hover auv wp22 wp26)  [30.201]
215.329: (observe auv wp26 ip7)  [10.000]
225.330: (correct_position auv wp26)  [10.000]
235.331: (do_hover auv wp26 wp21)  [23.177]
258.509: (observe auv wp21 ip2)  [10.000]
268.510: (correct_position auv wp21)  [10.000]
278.511: (do_hover auv wp21 wp27)  [21.255]
299.767: (observe auv wp27 ip8)  [10.000]
309.768: (correct_position auv wp27)  [10.000]
319.769: (observe auv wp27 ip6)  [10.000]
329.770: (correct_position auv wp27)  [10.000]
339.771: (do_hover auv wp27 wp17)  [23.597]
363.369: (do_hover auv wp17 wp25)  [21.413]
384.783: (do_hover auv wp25 wp32)  [16.710]
401.494: (do_hover auv wp32 wp36)  [21.451]
422.946: (observe auv wp36 ip9)  [10.000]
432.947: (correct_position auv wp36)  [10.000]
442.948: (observe auv wp36 ip15)  [10.000]
```



Illustrative Example

AUV domain from (Cashmore et al, ICRA 2015)

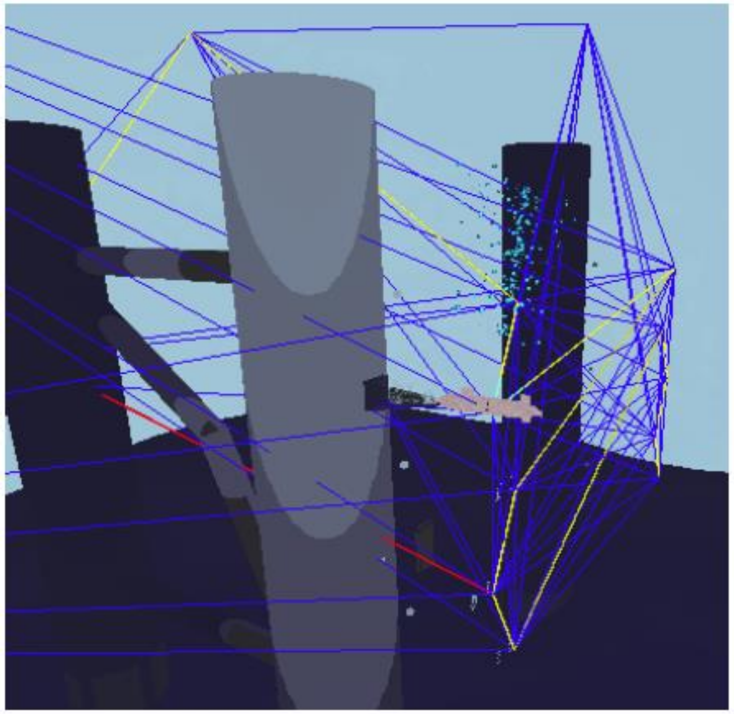
```
0.000: (observe auv wp1 ip3)  [10.000]
10.001: (correct_position auv wp1)  [10.000]
20.002: (do_hover auv wp1 wp2)  [71.696]
91.699: (observe auv wp2 ip4)  [10.000]
101.700: (correct_position auv wp2)  [10.000]
111.701: (do_hover auv wp2 wp23)  [16.710]
128.412: (observe auv wp23 ip5)  [10.000]
138.413: (correct_position auv wp23)  [10.000]
148.414: (observe auv wp23 ip1)  [10.000]
158.415: (correct_position auv wp23)  [10.000]
168.416: (do_hover auv wp23 wp22)  [16.710]
185.127: (do_hover auv wp22 wp26)  [30.201]
215.329: (observe auv wp26 ip7)  [10.000]
225.330: (correct_position auv wp26)  [10.000]
235.331: (do_hover auv wp26 wp21)  [23.177]
258.509: (observe auv wp21 ip2)  [10.000]
268.510: (correct_position auv wp21)  [10.000]
278.511: (do_hover auv wp21 wp27)  [21.255]
299.767: (observe auv wp27 ip8)  [10.000]
309.768: (correct_position auv wp27)  [10.000]
319.769: (observe auv wp27 ip6)  [10.000]
329.770: (correct_position auv wp27)  [10.000]
339.771: (do_hover auv wp27 wp17)  [23.597]
363.369: (do_hover auv wp17 wp25)  [21.413]
384.783: (do_hover auv wp25 wp32)  [16.710]
401.494: (do_hover auv wp32 wp36)  [21.451]
422.946: (observe auv wp36 ip9)  [10.000]
432.947: (correct_position auv wp36)  [10.000]
442.948: (observe auv wp36 ip15)  [10.000]
```



Illustrative Example

AUV domain from (Cashmore et al, ICRA 2015)

```
0.000: (observe auv wp1 ip3)  [10.000]
10.001: (correct_position auv wp1)  [10.000]
20.002: (do_hover auv wp1 wp2)  [71.696]
91.699: (observe auv wp2 ip4)  [10.000]
101.700: (correct_position auv wp2)  [10.000]
111.701: (do_hover auv wp2 wp23)  [16.710]
128.412: (observe auv wp23 ip5)  [10.000]
138.413: (correct_position auv wp23)  [10.000]
148.414: (observe auv wp23 ip1)  [10.000]
158.415: (correct_position auv wp23)  [10.000]
168.416: (do_hover auv wp23 wp22)  [16.710]
185.127: (do_hover auv wp22 wp26)  [30.201]
215.329: (observe auv wp26 ip7)  [10.000]
225.330: (correct_position auv wp26)  [10.000]
235.331: (do_hover auv wp26 wp21)  [23.177]
258.509: (observe auv wp21 ip2)  [10.000]
268.510: (correct_position auv wp21)  [10.000]
278.511: (do_hover auv wp21 wp27)  [21.255]
299.767: (observe auv wp27 ip8)  [10.000]
309.768: (correct_position auv wp27)  [10.000]
319.769: (observe auv wp27 ip6)  [10.000]
329.770: (correct_position auv wp27)  [10.000]
339.771: (do_hover auv wp27 wp17)  [23.597]
363.369: (do_hover auv wp17 wp25)  [21.413]
384.783: (do_hover auv wp25 wp32)  [16.710]
401.494: (do_hover auv wp32 wp36)  [21.451]
422.946: (observe auv wp36 ip9)  [10.000]
432.947: (correct_position auv wp36)  [10.000]
442.948: (observe auv wp36 ip15)  [10.000]
```



Outline

- Why PDDL Planning for Robotics?
  - Expressive Planning
  - Opportunistic Planning
  - Strategic Planning
  - eXplainable Planning (XAIP)
  - Planning with Uncertainty

Planning with Uncertainty

Uncertainty and lack of knowledge is a huge part of AI Planning for Robotics.

- Actions might fail or succeed.
- The effects of an action can be non-deterministic.
- The environment is dynamic and changing.
- Humans are unpredictable.
- The environment is often initially full of unknowns.

The domain model is *always* incomplete as well as inaccurate.

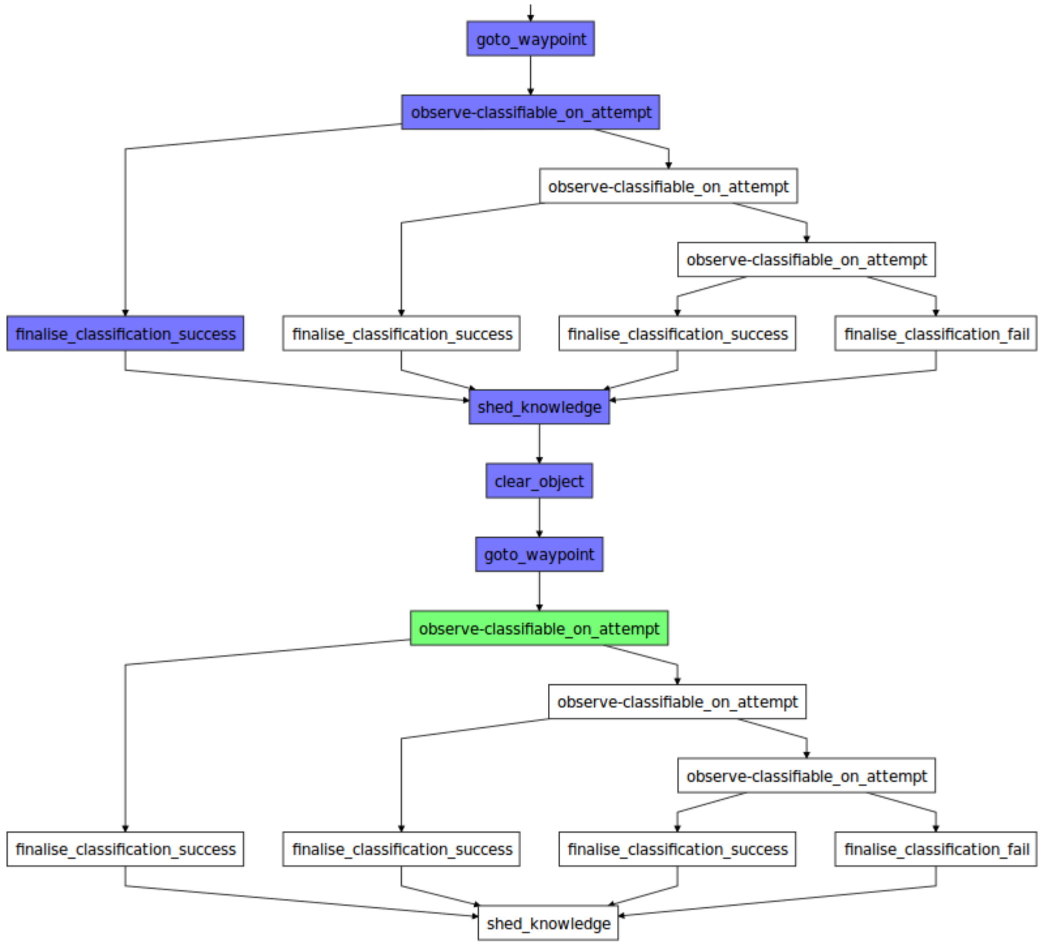
Uncertainty in AI Planning

Some uncertainty can be handled at planning time:

- Fully-Observable Non-deterministic planning.

- Partially-observable Markov decision Process.

- Conditional Planning with Contingent Planners. (e.g. ROSPlan with Contingent-FF)


goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge
clear_object
goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge

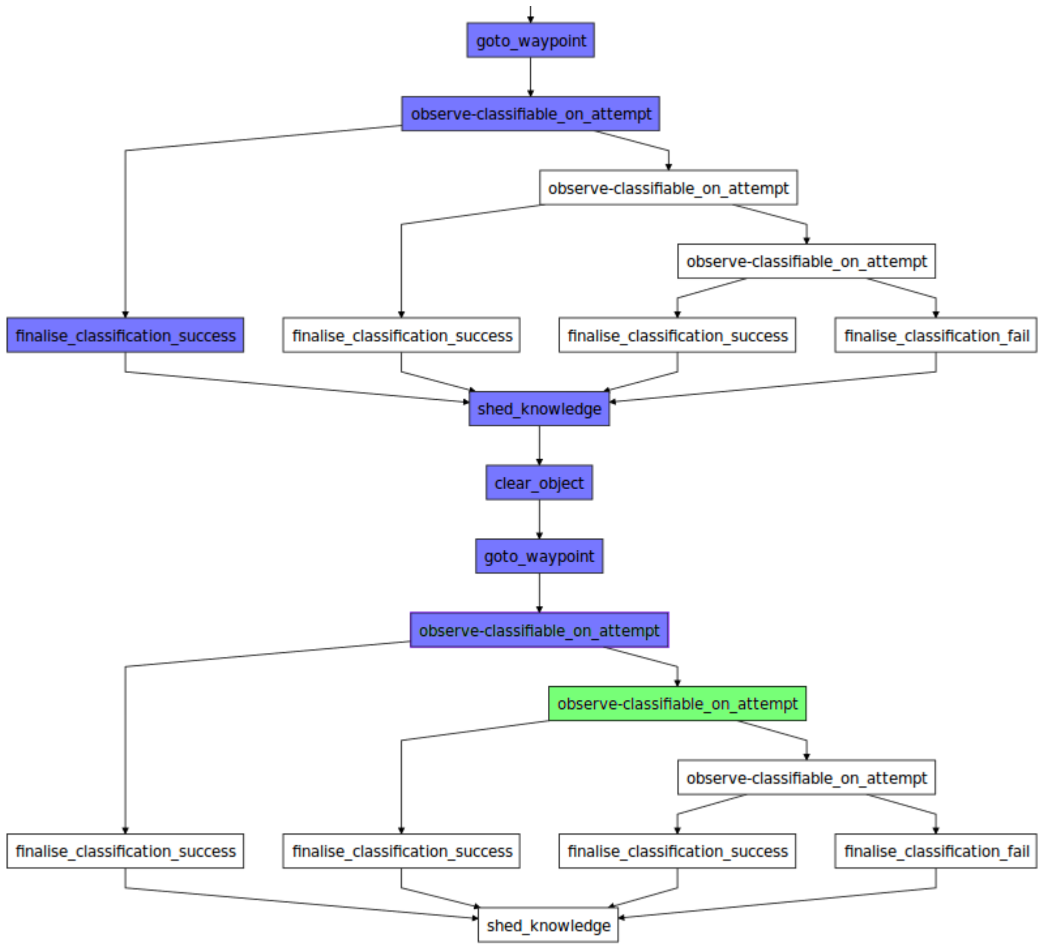
Uncertainty in AI Planning

Some uncertainty can be handled at planning time:

- Fully-Observable Non-deterministic planning.

- Partially-observable Markov decision Process.

- Conditional Planning with Contingent Planners. (e.g. ROSPlan with Contingent-FF)


goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge
clear_object
goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge

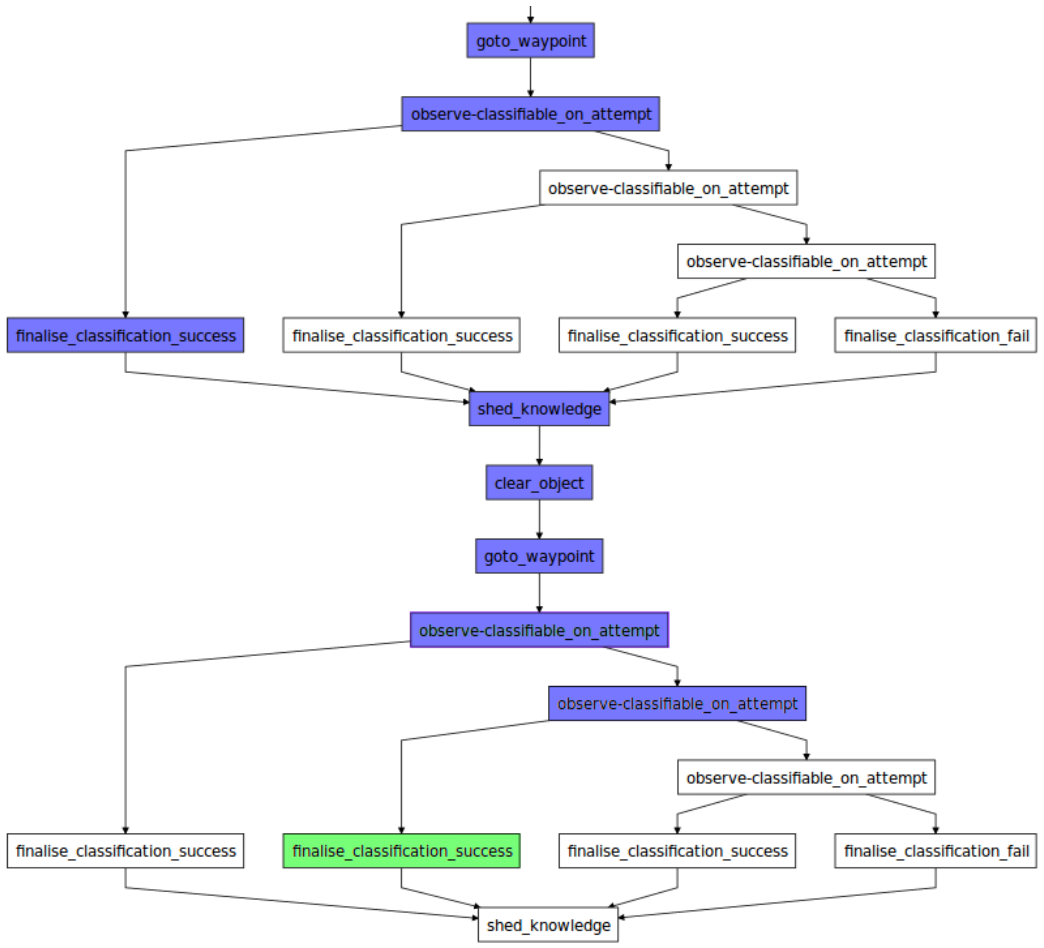
Uncertainty in AI Planning

Some uncertainty can be handled at planning time:

- Fully-Observable Non-deterministic planning.

- Partially-observable Markov decision Process.

- Conditional Planning with Contingent Planners. (e.g. ROSPlan with Contingent-FF)


goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge
clear_object
goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge

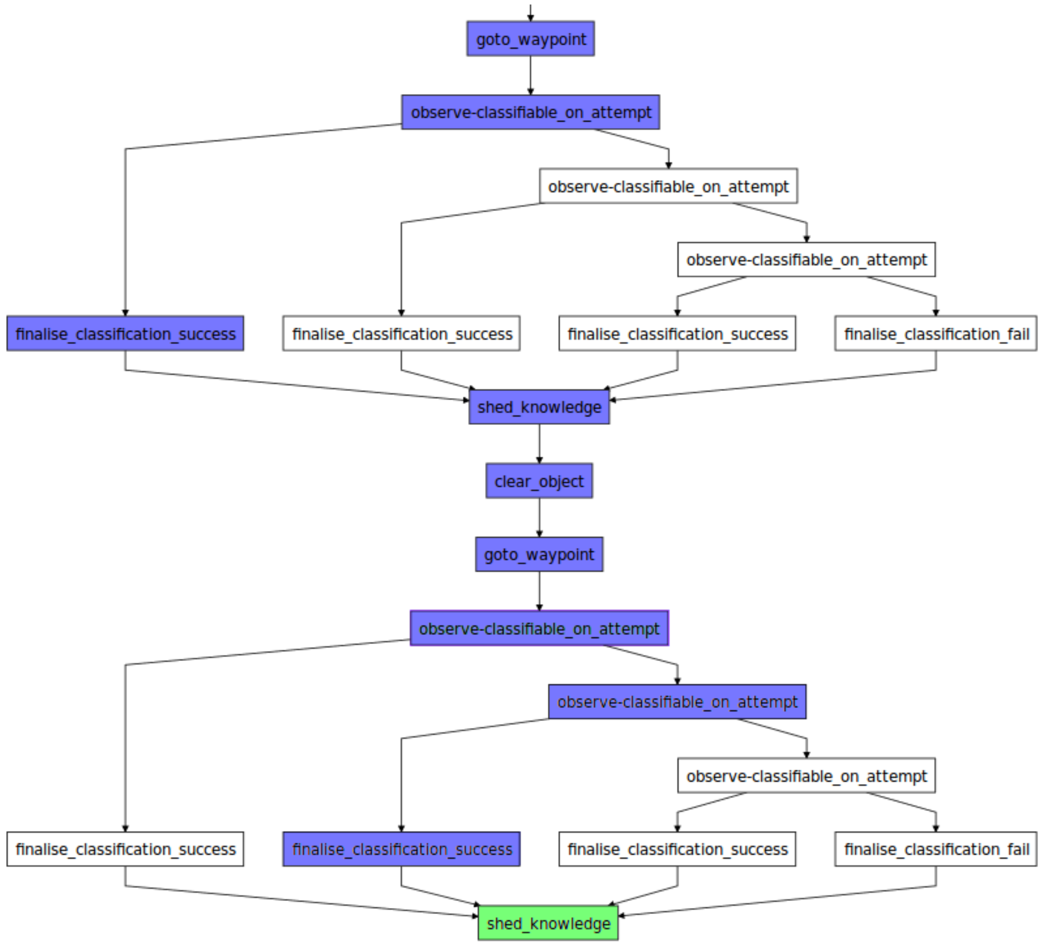
Uncertainty in AI Planning

Some uncertainty can be handled at planning time:

- Fully-Observable Non-deterministic planning.

- Partially-observable Markov decision Process.

- Conditional Planning with Contingent Planners. (e.g. ROSPlan with Contingent-FF)


goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge
clear_object
goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge

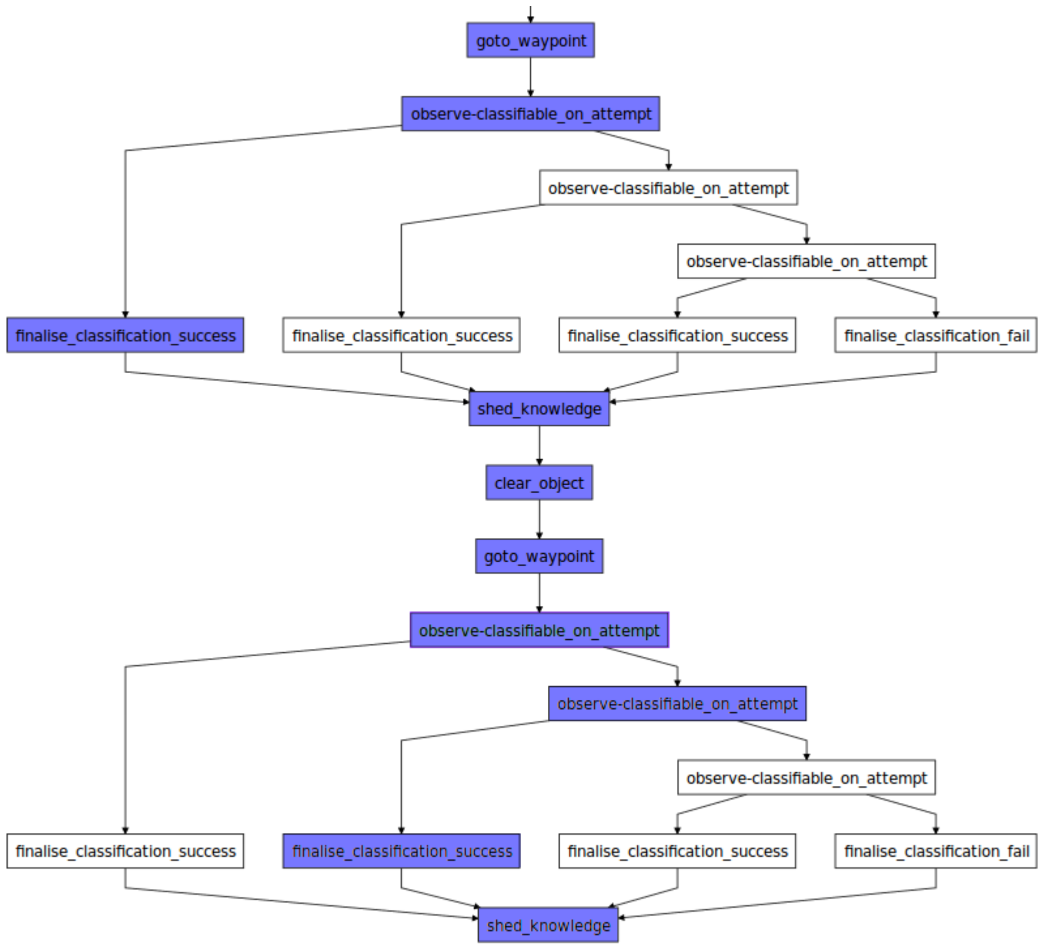
Uncertainty in AI Planning

Some uncertainty can be handled at planning time:

- Fully-Observable Non-deterministic planning.

- Partially-observable Markov decision Process.

- Conditional Planning with Contingent Planners. (e.g. ROSPlan with Contingent-FF)


goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge
clear_object
goto_waypoint
observe-classifiable_on_attempt
observe-classifiable_on_attempt
observe-classifiable_on_attempt
finalise_classification_success
finalise_classification_success
finalise_classification_success
finalise_classification_fail
shed_knowledge

ROSPlan: Planning in the Robot Operating System