Task and Motion Planning (TAMP)

Chris Paxton

Slides modified from: Caelan Reed Garrett
github.com/caelan/pddlstream

NVIDIA

1

Introduction

- "Robot, cook me a bowl of tomato soup"
- Must find out a sequence of grasps and motions that will result in a cooked can of soup
- In this case: grabs soup, pours it out into the red bowl, puts the red bowl on the stove, and then takes it off.
- Easy, right?

2

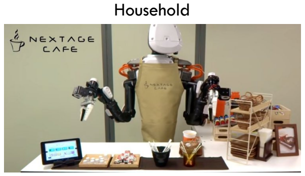
Planning for Autonomous Robots

3

- Robot must select both **high-level** actions & **low-level** controls
- **Application areas:** semi-structured and human environments

Household

Warehouse fulfilment



Food service

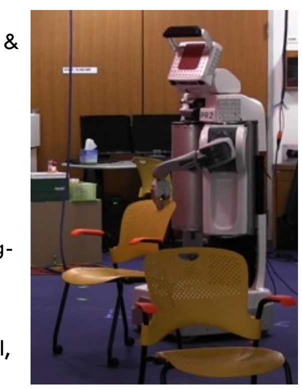
Construction

3

Task and Motion Planning (TAMP)

4

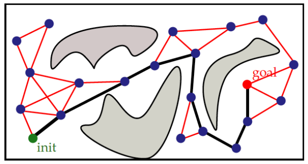
- Plan in a **factored, hybrid** space
  - **Discrete** and **continuous** variables & actions
- **Variables**
  - **Continuous:** robot configuration, object poses, door joint positions,
  - **Discrete:** is-on, is-in-hand, is-holding-water, is-cooked, …
- **Actions:** move, pick, place, push, pull, pour, detect, cook, …



4

(Probable) Roadmap

1. **Background**
   1. Task Planning
   2. Motion Planning
2. **Hybrid Planning**
   1. Prediscretized & Numeric Planning
   2. Multi-Modal Motion Planning
   3. Integrated TAMP
3. **STRIPStream**
4. Temporal TAMP
5. TAMP under Uncertainty



[Fig from Erion Plaku]

5

Cooking and Stacking

6

6

Preparing Coffee

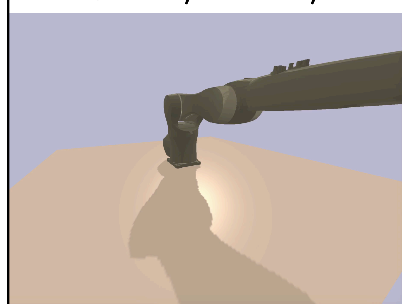
7

7

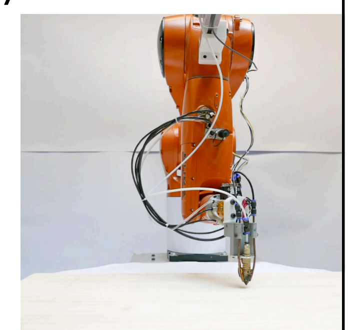
Automated Fabrication

8

- Plan sequence of **306** 3D printing extrusions (actions)
- Collision, kinematic, **stability** and **stiffness** constraints



[Huang, Garrett, & Mueller 2018]



8

Problem Class

- **Discrete-time**
  - Plans are finite sequences of controls
- **Deterministic** (for now)
  - Actions always produce the intended effect
  - Solutions are **plans** (instead of policies)
- **Observable** (for now)
  - Access to the full world state
- **Hybrid**
  - States & controls composed of **mixed discrete-continuous variables**

9

Classical (Task) Planning

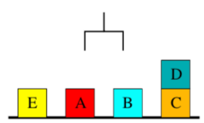
11

- Key focus: **discrete** problems with **many variables**
  - Often enormous, but **finite,** state-spaces
- Problems typically described using an **action language**
  - **Propositional Logic** (STRIPS) [Fikes 1971]
  - **Planning Domain Description Language** (PDDL) [Aeronautiques 1998]
- Develop **domain-independent** algorithms
  - Can apply to **any problem** expressible using PDDL
- Exploit **factored** and **sparse** structure to develop efficient algorithms

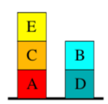
11

Classical Planning Representations

12



Initial State



Goal State

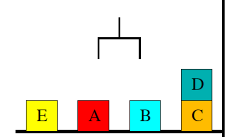
- Facts: $on(x, y)$, $onTable(x)$, $clear(x)$, $holding(x)$, $armEmpty()$.
- Initial state: $\{onTable(E), clear(E), \ldots, onTable(C), on(D, C), clear(D), armEmpty()\}$.
- Goal: $\{on(E, C), on(C, A), on(B, D)\}$.
- Actions: $stack(x, y)$, $unstack(x, y)$, $putdown(x)$, $pickup(x)$.
- $stack(x, y)$? $pre : \{holding(x), clear(y)\}$
  $add : \{on(x, y), armEmpty()\}$
  $del : \{holding(x), clear(y)\}$.

12

First-Order Action Languages

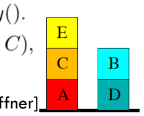
13

- **Predicate:** boolean function `(On ?b1 ?b2)=True/False`
- **Facts** (literals)**:** instantiated predicates `(On D C)=True`
- **State:** set of facts `{(On A B)=False, (On D C)=True, …}`
  - Equivalently, boolean state variables
  - **Closed-world** assumption: unspecified facts are **false**
- Example: **Blocksworld** domain



Initial State

Facts: $on(x, y)$, $onTable(x)$, $clear(x)$, $holding(x)$, $armEmpty()$.
Initial state: $\{onTable(E), clear(E), \ldots, onTable(C), on(D, C), clear(D), armEmpty()\}$.
Goal: $\{on(E, C), on(C, A), on(B, D)\}$. [Figs from Hector Geffner]
Actions: $stack(x, y)$, $unstack(x, y)$, $putdown(x)$, $pickup(x)$.



Goal State

13

(Lifted) Action Schema

14

- A tuple of free **parameters**
- A **precondition** formula tests applicability
- An **effect** formula modifies the state
- Logical **conjunctions** enable factoring
- Effects are **deltas**

```
(:action stack
  :parameters (?b1 ?b2)
  :precondition (and
    (Holding ?b1) (Clear ?b2))
  :effect (and
    (ArmEmpty)
    (On ?b1 ?b2) (Clear ?b1)
    (not (Holding ?b1))
    (not (Clear ?b2))))
```

```
(:action unstack
  :parameters (?b1 ?b2)
  :precondition (and
    (ArmEmpty) (On ?b1 ?b2)
    (Clear ?b1))
  :effect (and
    (Holding ?b1) (Clear ?b2)
    (not (Clear ?b1))
    (not (ArmEmpty))
    (not (On ?b1 ?b2))))
```

14

Planning Approaches

15

- **State-space** search: [Bonet 2001] [Hoffman 2001] [Helmert 2006]
  - **Progression** (forward) or regression (backward)
  - Best-first **heuristic search** algorithms
- **Partial-order** planning [Penberthy 1992]
  - Search directly over plans (**plan-space**)
- Planning as **Satisfiability** [Kautz 1999]
  - Compile to **fixed-horizon** SAT instance
  - SAT is **NP-Complete**
  - Planning is **PSPACE-Complete**
  - **Increase horizon if formula unsatisfiable**

15

Forward Best-First Search

16

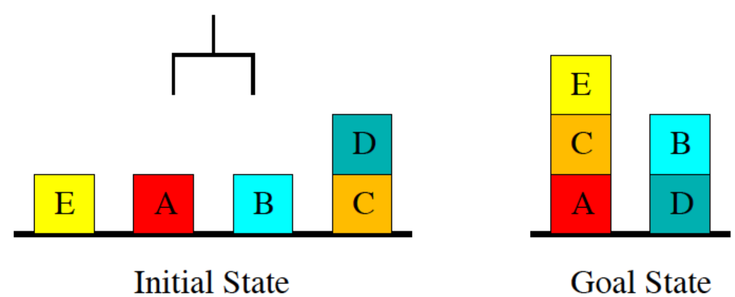
- For a state s
  - Path **cost**: $g(s)$
  - **Heuristic** estimate: $h(s)$
  - Open list **sorted** by priority $f(s)$
- **Weighted A***: $f(s) = g(s) + wh(s)$
  - Uniform cost search: $w = 0 \implies f(s) = g(s)$
  - A* search: $w = 1 \implies f(s) = g(s) + h(s)$
  - **Greedy** best-first search: $w = \infty \implies f(s) = h(s)$
- How do we estimate $h(s)$?
  - No obvious metric (no metric-space embedding)

16

Predict the Minimum Plan Length

17

- Can stack / unstack anywhere on the ground
- Hint: is an **even** number

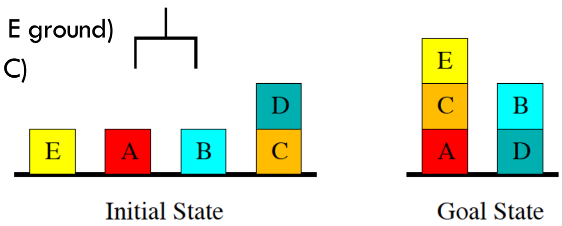


17

Predict the Minimum Plan Length

18

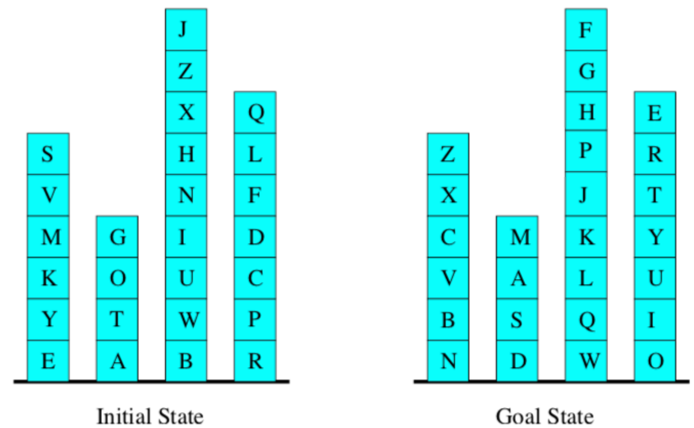
- **Solution** (length=6):
  - (unstack D C)
  - (stack D B)
  - (unstack C ground)
  - (stack C A)
  - (unstack E ground)
  - (stack E C)



18

Predict the Minimum Plan Length

19



19

Domain-Independent Heuristics

20

- Estimating $h(s)$ is **nontrivial**
- Can we do it in an a **domain-independent** manner?
- Solve a related, **approximate** planning problem
  - Primary focus for almost all of classical planning
- Suggestions for how to do this?
  - **Independently** plan for each goal
  - **Remove** some action preconditions [Helmert 2006]
  - Remove negative (**delete**) **effects** [Bonet 2001] [Hoffman 2001]
  - …

20

Delete-Relaxation Heuristics

21

- Remove all negative (**not**) effects
  - Solving optimally is **NP-Complete**
  - Can greedily find a short plan in polynomial time
- Basis for both **admissible** and **greedier,** non-admissible heuristics

(**:action** stack
  :parameters (?b1 ?b2)
  :precondition (**and**
    (Holding ?b1) (Clear ?b2))
  :effect (**and**
    (ArmEmpty)
    (On ?b1 ?b2) (Clear ?b1)
    ~~(**not** (Holding ?b1))~~
    ~~(**not** (Clear ?b2))~~))

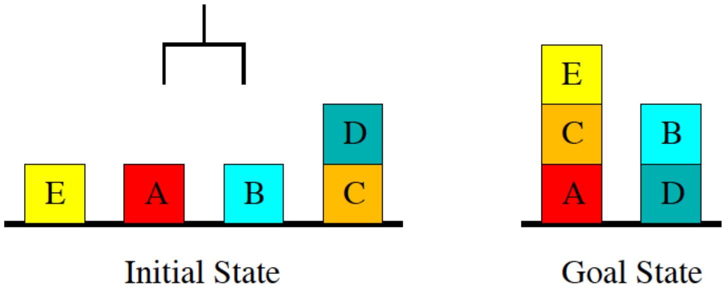
(**:action** unstack
  :parameters (?b1 ?b2)
  :precondition (**and**
    (ArmEmpty) (On ?b1 ?b2)
    (Clear ?b1))
  :effect (**and**
    (Holding ?b1) (Clear ?b2)
    ~~(**not** (Clear ?b1))~~
    ~~(**not** (ArmEmpty))~~
    ~~(**not** (On ?b1 ?b2))~~))

21

Predict the Minimum Delete-Relaxed Plan Length

22

- Can stack / unstack anywhere on the ground
- Hint: is **no greater** than 6

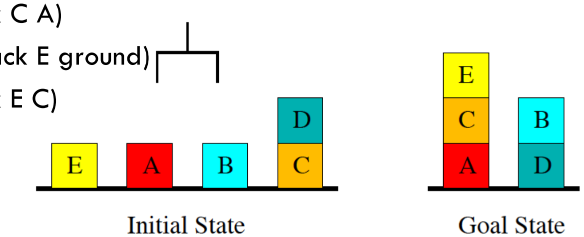


22

Predict the Minimum Delete-Relaxed Plan Length

23

- **Solution** (length=6):
  - (unstack D C)
  - (stack D B)
  - (unstack C ground)
  - (stack C A)
  - (unstack E ground)
  - (stack E C)

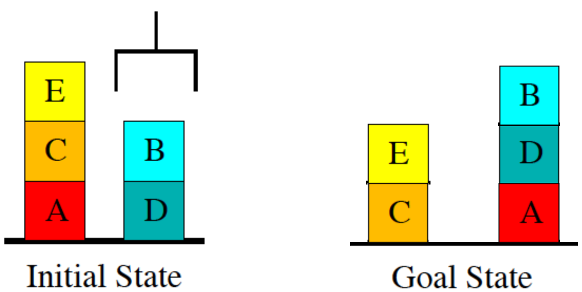


23

Predict the Minimum Plan Length

24

- Can stack / unstack anywhere on the ground
- Hint: is an **even** number

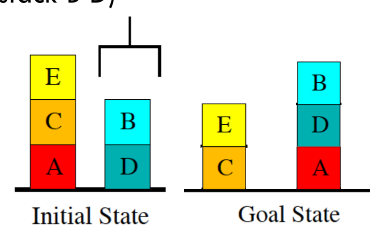


24

Predict the Minimum Plan Length

25

- **Solution** (length=12):
  - (unstack E C)
  - (stack E ground)
  - (unstack C A)
  - (stack C ground)
  - (unstack E ground)
  - (stack E C)
  - (unstack B D)
  - (stack B ground)
  - (unstack D ground)
  - (stack D A)
  - (unstack B ground)
  - (stack B D)

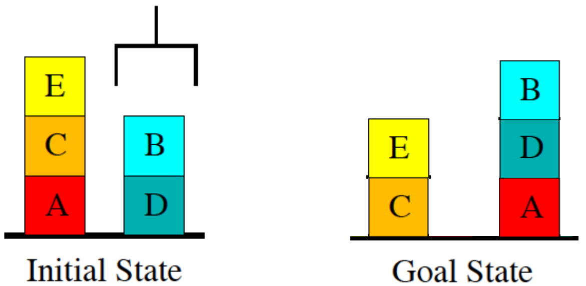


25

Predict the Minimum Delete-Relaxed Plan Length

26

- Can stack / unstack anywhere on the ground
- Hint: is **no greater** than 12

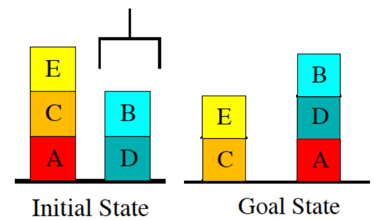


26

Predict the Minimum Delete-Relaxed Plan Length

27

- **Solution** (length=5):
  - (unstack E C)
  - (unstack C A)
  - (unstack B D)
  - (unstack D ground)
  - (stack D A)

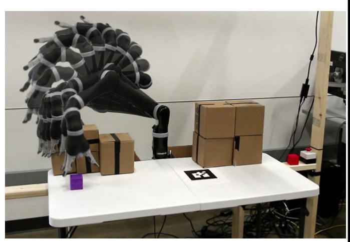


27

Review: Motion Planning

29

- Plan a **path** for a robot from an initial configuration to a goal configuration that **avoids obstacles**
  - Sequence of **<u>continuous</u>** configurations
  - Configurations often are **high-dimensional**
    - Example: 7 DOFs
- High-level approaches:
  - Geometric decomposition
  - **Sampling-based**
  - Optimization-based

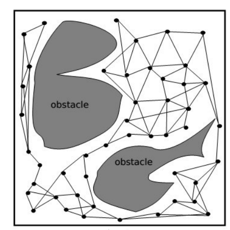


29

Sampling-Based Motion Planning

32

- **Discretize** configuration space by **sampling**
  - Sampling be deterministic or **random**
- **Implicitly** represent the collision-free configuration space using an blackbox **collision checker**
  - **Abstracts away** complex robot geometry
- Algorithms
  - Probabilistic Roadmap (PRM)
  - Rapidly-Exploring Random Tree (RRT)
  - Bidirectional RRT (BiRRT)

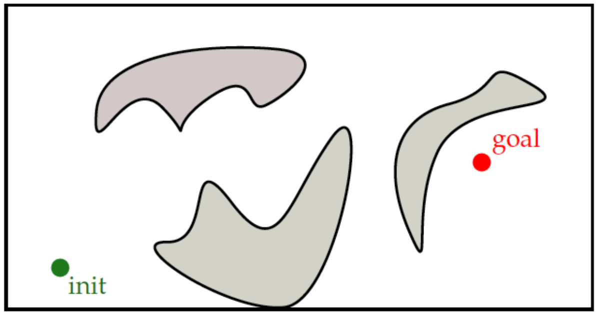


[Kavraki 1994][Kuffner 2000][LaValle 2006]

32

Probabilistic Roadmap (1/7)

33



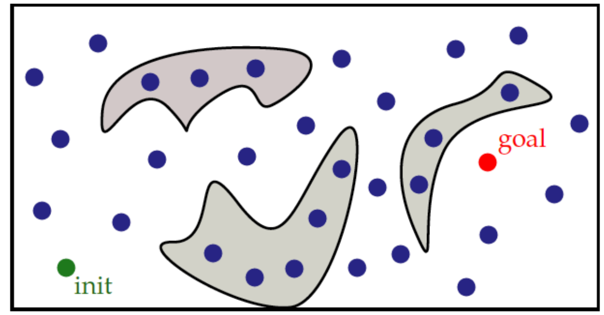
[Fig from Erion Plaku]

Find a path from init to goal that avoids the obstacles

33

Probabilistic Roadmap (2/7)

34



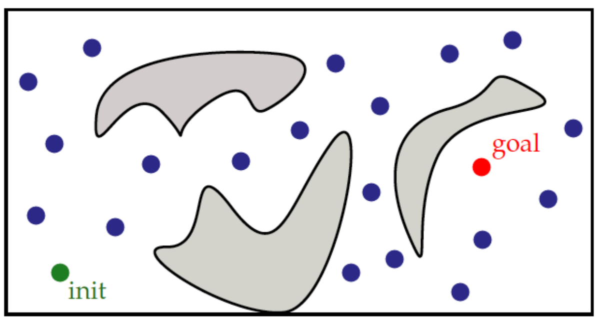
[Fig from Erion Plaku]

Sample a set of configurations

34

Probabilistic Roadmap (3/7)

35



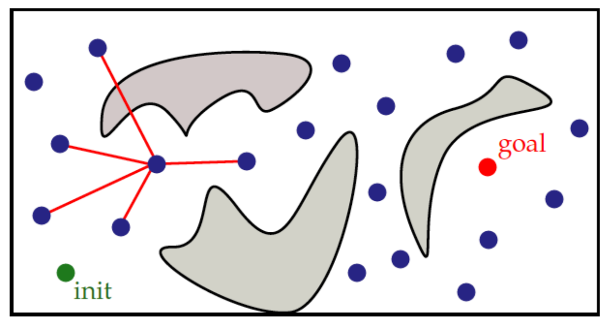
[Fig from Erion Plaku]

Remove configurations that collide with the obstacles

35

Probabilistic Roadmap (4/7)

36



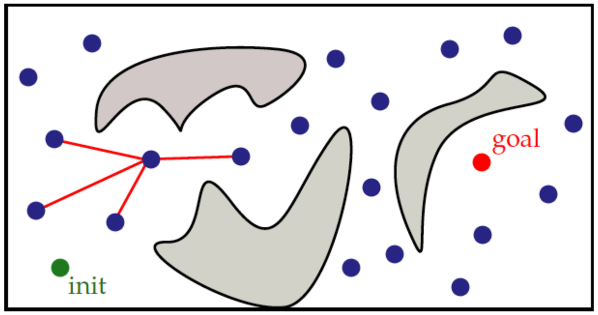
[Fig from Erion Plaku]

Connect nearby configurations

36

Probabilistic Roadmap (5/7)

37



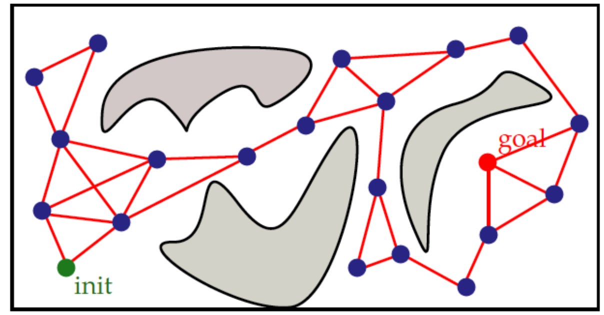
[Fig from Erion Plaku]

Prune connections that collide with the obstacles

37

Probabilistic Roadmap (6/7)

38



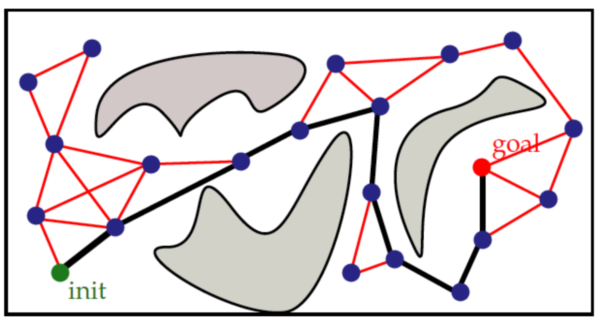
[Fig from Erion Plaku]

The resulting structure is a finite roadmap (graph)

38

Probabilistic Roadmap (7/7)

39



[Fig from Erion Plaku]

Search for the shortest-path on the roadmap

39

Task and Motion Planning (TAMP)

54

Shakey the Robot (1969)

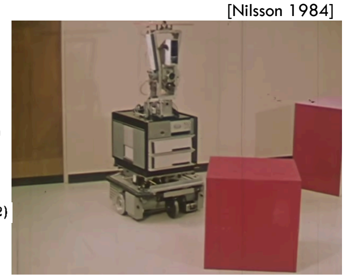
55

- **First autonomous mobile manipulator** (via pushing)
  - Visibility graph, A* search, and STRIPS!
- **Decoupled** task and motion planning
  - Task planning **then** motion planning

[Fikes 1971]
[Nilsson 1984]

```
type(robot robot)   type(o1 object)
name(robot shakey)  name(o1 box1)
at(robot 4.1 7.2)   at(o1 3.1 5.2)
theta(robot 90.1)   inroom(o1 r1)
                    shape(o1 wedge)
                    radius(o1 3.1)
```

GOTHRU(d,r1,r2)
- Precondition: INROOM(ROBOT,r1) ∧ CONNECTS(d,r1,r2)
- Delete List: INROOM(ROBOT,$)
- Add List: INROOM(ROBOT,r2)

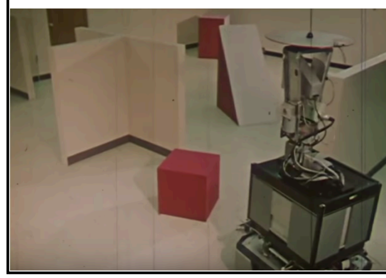


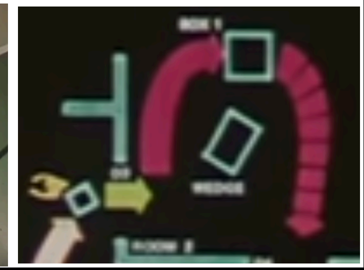
55

Obstacle Blocks Shakey's Path

56

- What if a movable block **prevented** Shakey from safely moving into the adjacent room?
- Shakey could **push** it out of the way or **go around** it
  - What's more efficient? How to push it? …



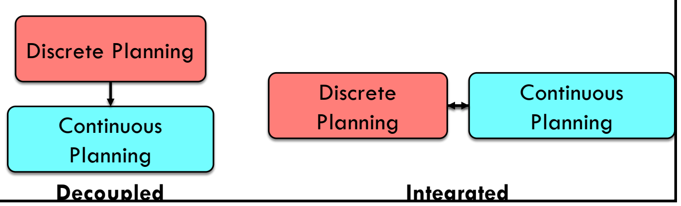


56

Decoupled vs Integrated TAMP

59

- **Decoupled:** discrete (task) planning **then** continuous (motion) planning
- Requires a strong **downward refinement** assumption
  - **Every** correct discrete plan can be **refined** into a correct continuous plan (from hierarchal planning)
- **Integrated:** **simultaneous** discrete & continuous planning



59

Geometric Constraints Affect Plan

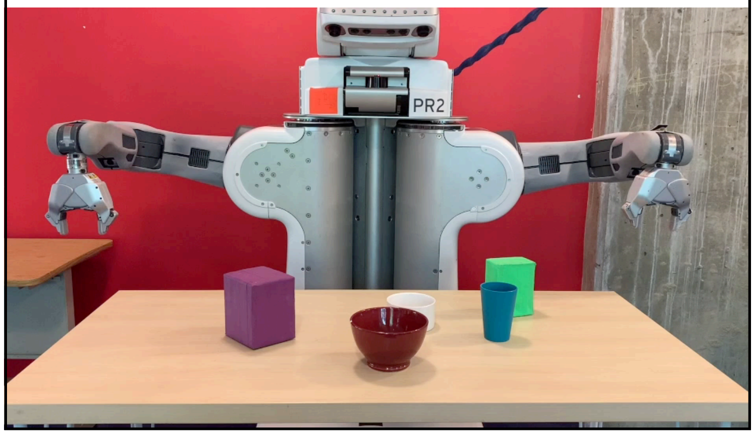
60

- **Inherits challenges** of both motion & classical planning
  - **High-dimensional, continuous** state-spaces
  - State-space **exponential** in number of variables
  - **Long horizons**
- Continuous constraints **limit high-level strategies**
  - Kinematics, reachability, joint limits, collisions grasp, visibility, stability, stiffness, torque limits, …

60

Pouring Among Obstacles

61



61

Preparing a Meal for Two

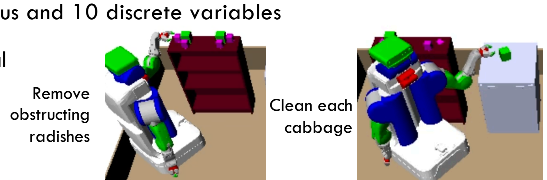
62

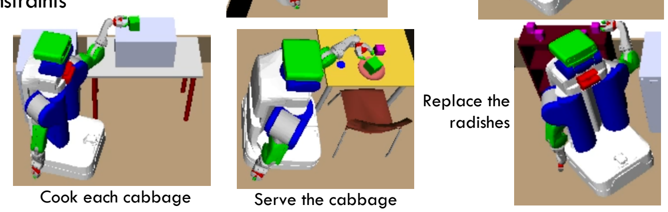
Breaking Down "Preparing a Meal"

63

- Clean 3 blue cups and clean/cook 2 green cabbages
- 64 continuous and 10 discrete variables

1. High-dimensional
2. Long horizon
3. Discrete state
4. Geometric constraints





63

Block in Left Cabinet & Doors Closed

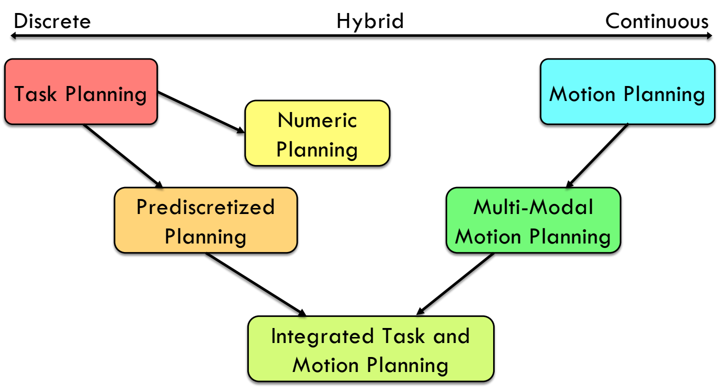
64

- Robot forced to **regrasp** the object
  - Change from a **top** grasp to a **side** grasp
- **Non-monotonic** problem
  - Plan must **undo goals** to solve
  - **Open** then **close** the cabinet door
- Physical constraints can be subtle!

64

Hybrid Planning Spectrum

66



66

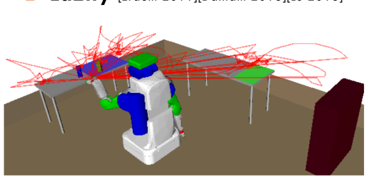
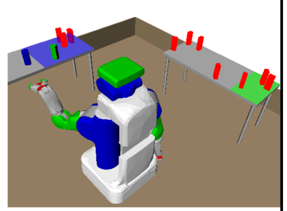
Prediscretized & Numeric Planning

67

Prediscretized Planning

68

- Assumes that a **finite set** of object placements, object grasps, and (sometimes) robot configurations are **given**
- Can **directly** perform discrete task planning
- Still need to evaluate **reachability**
  - Eagerly in **batch** [Lozano-Pérez 2014][Garrett 2017][Ferrer-Mestres 2017]
  - Eagerly during **search** [Dornhege 2009]
  - **Lazily** [Erdem 2011][Dantam 2018][Lo 2018]

68

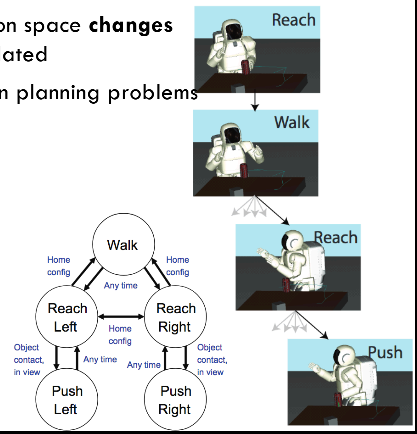
Multi-Modal Motion Planning

72

Multi-Modal Motion Planning

73

- Collision-free configuration space **changes** when objects are manipulated
- Use a **sequence** of motion planning problems each defined by a **mode**
- **Mode:** a set of motion constraints
  - Gripper is empty
  - Relative object pose remains **constant**



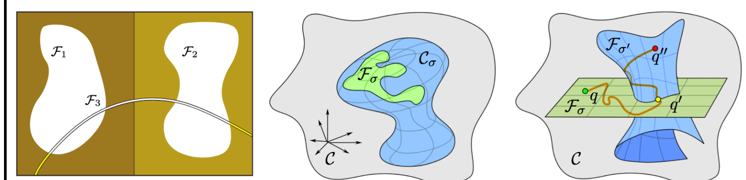
[Alami 1994][Siméon 2004][Hauser 2011]
[Barry 2013][Vega-Brown 2016]

73

Low-dimensional Intersections

74

- Need samples that **connect** adjacent modes
- Intersection of two modes is often **low-dimensional**
  - **Special-purpose** samplers are needed
- **Example:** transition from gripper **empty** to **holding**
- Configurations at the **intersection** obtained using **inverse kinematics** (IK)



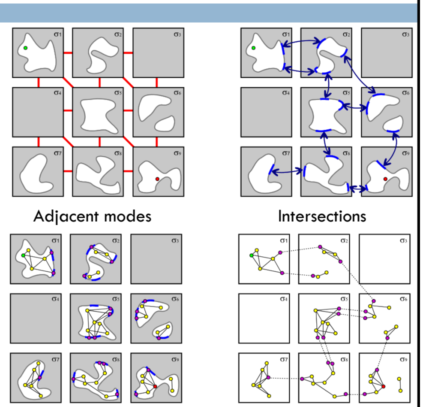
[Hauser 2011]

74

Sampling-Based Multi-Modal Planning

75

1. Sample from the set of **modes**
2. Sample at the **low-dimensional intersection** of adjacent modes
3. Sample a roadmap **within** each mode
4. Discrete search on the multi-modal **roadmap**



Adjacent modes

Intersections

Individual mode roadmaps

Combined Roadmap

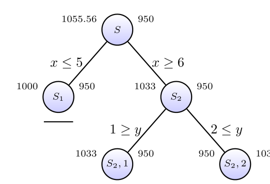
[Hauser 2011]

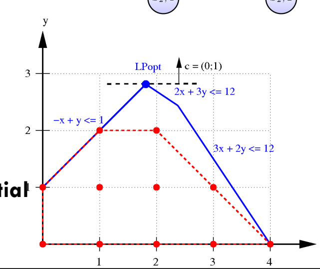
75

Mixed Integer Programming (MIP)

76

- Continuous and integer variables
- Convex constraints and costs
- **Branch-and-bound**
  - Split on integer variables
- **Integrality relaxation**
  - Lower bound on cost
  - Loose when **logical** operations
- Planning limitation
  - \# of variables may be **exponential** in problem size



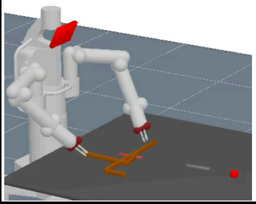


76

Optimization-Based Multi-Modal Motion Planning

77

- Discrete search over sequences of **mode switches**
  - Sequences have **varying length**
  - Each sequence induces a **non-convex constrained optimization problem**
- Sequences can be pruned using **lower bounds** obtained by **relaxing** some constraints [Lagriffoul 2014]



[Toussaint 2015]
[Toussaint 2018]

$$\begin{aligned}\min_{x,a_{1:K},s_{1:K}} \quad & \int_0^T f_{\text{path}}(\bar{x}(t))\ dt + f_{\text{goal}}(x(T)) \\ \text{s.t.} \quad & x(0)=x_0,\ h_{\text{goal}}(x(T))=0,\ g_{\text{goal}}(x(T))\le 0, \\ & \forall t\in[0,T]:\ h_{\text{path}}(\bar{x}(t),s_{k(t)})=0, \\ & \qquad g_{\text{path}}(\bar{x}(t),s_{k(t)})\le 0 \\ & \forall k\in\{1,..,K\}:\ h_{\text{switch}}(\hat{x}(t_k),a_k)=0, \\ & \qquad g_{\text{switch}}(\hat{x}(t_k),a_k)\le 0, \\ & \qquad s_k\in \text{succ}(s_{k\text{-}1},a_k)\ .\end{aligned}$$

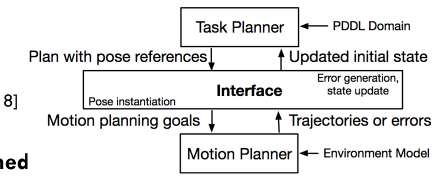
77

Integrated TAMP

78

- Geometric search **guided** by classical planning
  - Both heuristic and sampling guidance [Gravot 2005][Plaku 2010]
- Task and motion planning **interface**
  - Maintain **separate** discrete and continuous descriptions
  - Custom interface to communicate between the two
  - How are failures **diagnosed?**

[Erdem 2011][De Silva 2013]
[Srivastava 2014][Dantam 2018]



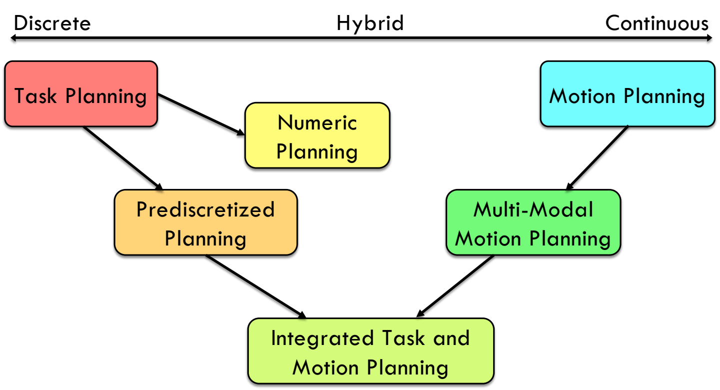
- Direct search in **combined**

[Kaelbling 2011] [Garrett 2018a] [Garrett 2018b]

78

Hybrid Planning Spectrum Revisited

81



81

One Approach: STRIPStream

83

- **No general-purpose, flexible framework** for planning in a variety of TAMP domains
- Extends **PDDL** to incorporate **sampling procedures**
  - Can model domains with **infinitely-many** actions
- Develop **domain-independent** algorithms that treat the samplers as **blackbox inputs**
- Algorithms solve a **sequence of finite** PDDL problems
  - Leverage existing **classical planners** as subroutines
- Algorithms are particularly **fast when downward refinement holds** while remaining **complete**

83

STRIPStream Language

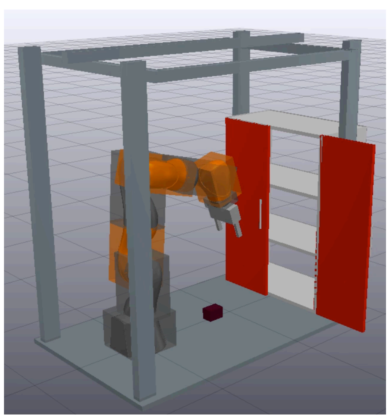
85

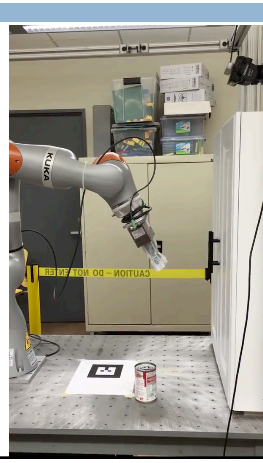
Benefits of Extending PDDL

- **Standardized** action description language
- Emphasis on describing and solving problems in a **domain-independent** way
- Large wealth of efficient, **existing algorithms** that exploit **factored** state & action structure
- Encodes the **difference** between two states using preconditions & effects
  - Most variables are **unchanged**
  - Actions can be described using **few parameters**

86

STRIPStream + Drake

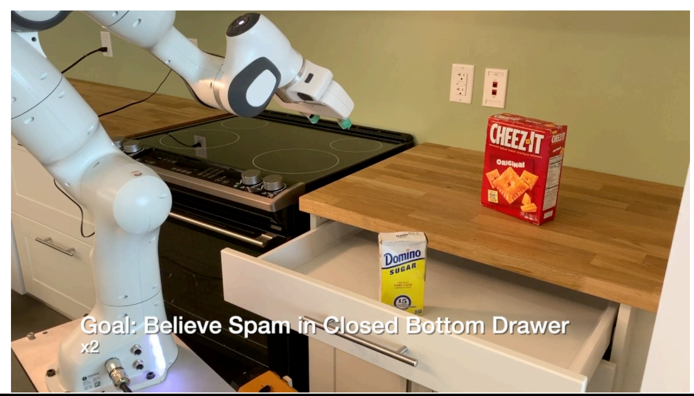




88

Solved Using the Same Algorithm

Framework not specific to a single robot or robotics at all!

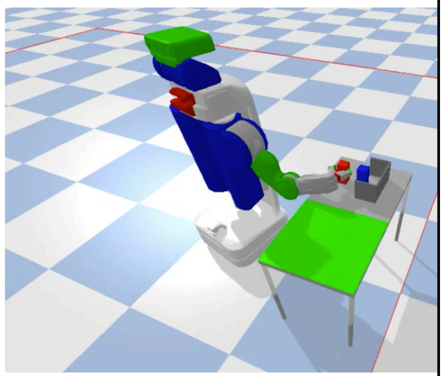


89

Motivating Pick & Place Example

91

- Single object **prevents** a goal object from being reachable
- Focus on a compact 2D version
- Formulation almost the same for 3D
- Algorithms agnostic to number of DOFs

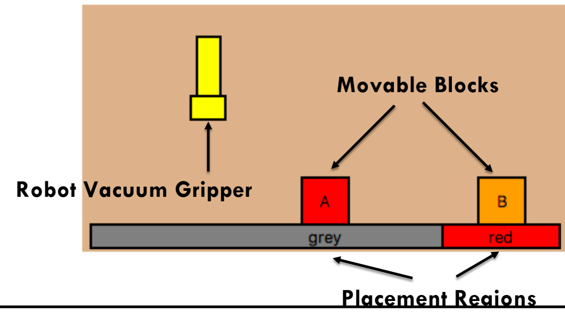


91

2D Pick-and-Place Example

92

- **Goal:** block **A** within the **red** region
- Robot and block poses are continuous (x, y) pairs
- Block **B** obstructs the placement of **A**

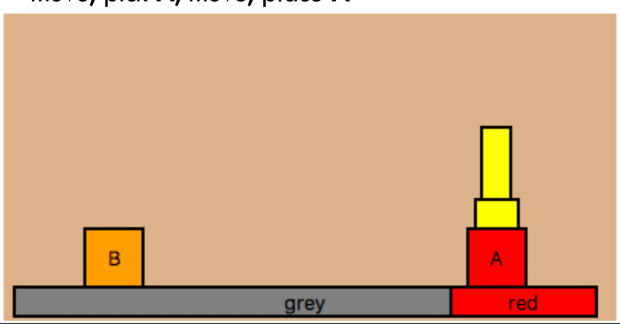


92

2D Pick-and-Place Solution

93

- One (of infinitely many) possible solutions
  - move, pick **B**, move, place **B**, move, pick **A**, move, place **A**



93

2D Pick-and-Place Initial & Goal

94

- Some constants are **numpy arrays**
- **Static initial facts** - value is **constant** over time
  - (Block, A), (Block, B), (Region, red), (Region, grey), (Conf, [-7.5 5.]), (Pose, A, [0. 0.]), (Pose, B, [7.5 0.]), (Grasp, A, [0. -2.5]), (Grasp, B, [0. -2.5])
- **Fluent initial facts** - value **changes** over time
  - (AtConf, [-7.5 5.]), (HandEmpty), (AtPose, A, [0. 0.]), (AtPose, B, [7.5 0.])
- **Goal formula:** `(exists (?p) (and (Contained A ?p red) (AtPose A ?p)))`

94

2D Pick-and-Place Actions

95

- Typical PDDL action description except that arguments are **high-dimensional** & **continuous**!
- To use the actions, must **prove** the following **static facts:**

```
(Motion ?q1 ?t ?q2), (Kin ?b ?p ?g ?q)
```

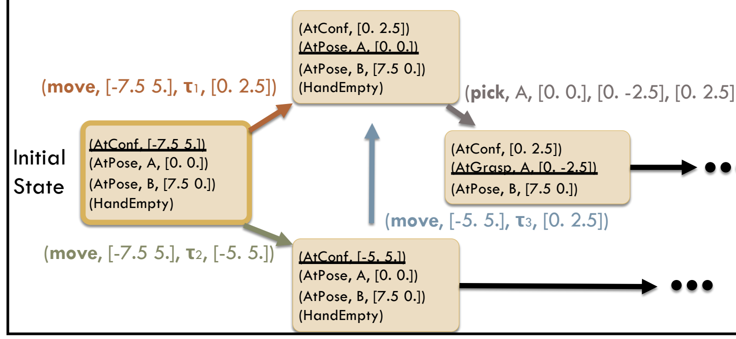
```
(:action move
  :parameters (?q1 ?t ?q2)
  :precondition (and (Motion ?q1 ?t ?q2) (AtConf ?q1))
  :effect (and (AtConf ?q2) (not (AtConf ?q1))))

(:action pick
  :parameters (?b ?p ?g ?q)
  :precondition (and (Kin ?b ?p ?g ?q)
                     (AtConf ?q) (AtPose ?b ?p) (HandEmpty))
  :effect (and (AtGrasp ?b ?g)
               (not (AtPose ?b ?p)) (not (HandEmpty))))
```

95

BFS in Discretized State-Space

96

- Suppose we were **given** the following additional static facts:
  - (Motion, [-7.5 5.], τ_1, [0. 2.5]), (Motion, [-7.5 5.], τ_2, [-5. 5.]), (Motion, [-5. 5.], τ_3, [0. 2.5]), (Kin, A, [0. 0.], [0. -2.5], [0. 2.5]), …



96

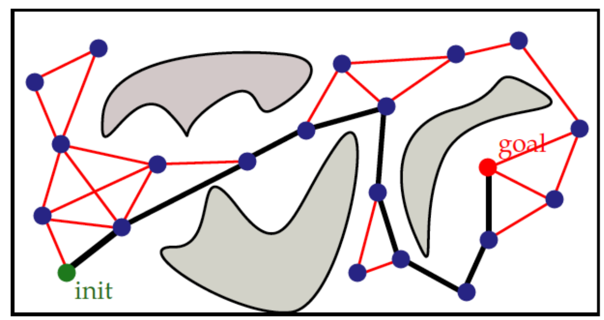
No a Priori Discretization

- **Values given at start:**
  - 1 initial configuration: (Conf, **[-7.5 5.]**)
  - 2 initial poses: (Pose, A, **[0. 0.]**), (Pose, B, **[7.5 0.]**)
  - 2 grasps: (Grasp, A, **[0. -2.5]**), (Grasp, B, **[0. -2.5]**)
- **Planner needs to find:**
  - 1 pose within a region: `(Contain A ?p red)`
  - 1 collision-free pose: `(CFree A ?p ? B ?p2)`
  - 4 grasping configurations: `(Kin ?b ?p ?g ?q)`
  - 4 robot trajectories: `(Motion ?q1 ?t ?q2)`

97

Intuition: Build a Roadmap

98



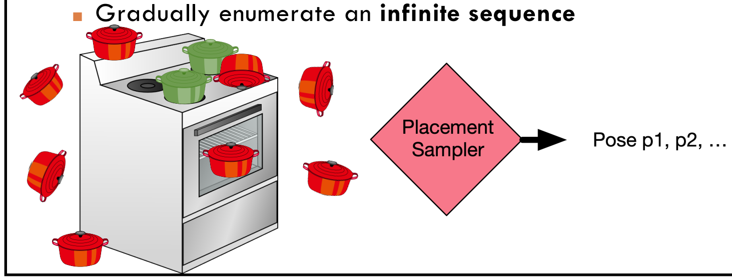
[Fig from Erion Plaku]

Search for the shortest-path on the roadmap

98

What Samplers Do We Need?

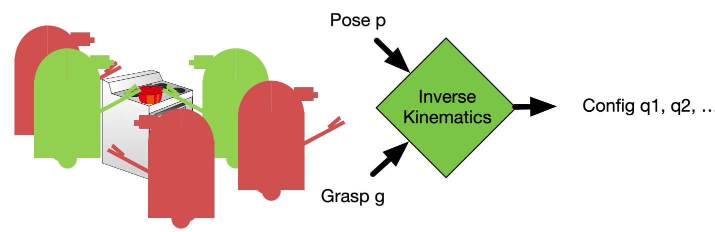
- **Low-dimensional** placement stability constraint (`Contain`)
  - *i.e.* 1D manifold embedded in 2D pose space
- Directly **sample values that satisfy the constraint**
- May need **arbitrarily many** samples
  - Gradually enumerate an **infinite sequence**



99

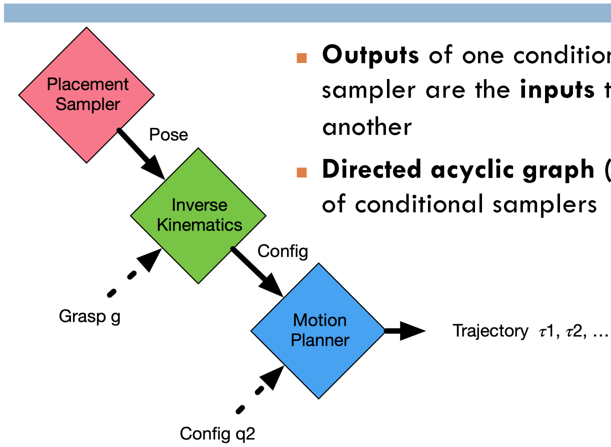
Intersection of Constraints

- **Kinematic constraint** (`Kin`) involves poses, grasps, and configurations
- **Conditional samplers** - samplers with inputs



100

Composing Conditional Samplers



- **Outputs** of one conditional sampler are the **inputs** to another
- **Directed acyclic graph (DAG)** of conditional samplers

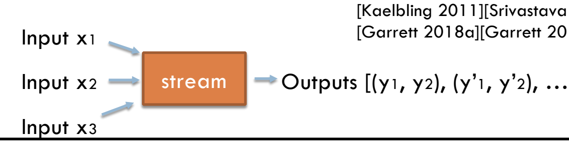
101

Stream: a function to a generator

- **Advantages**
  - Programmatic implementation
  - Compositional
  - Supports infinite sequences
- **Stream** - function from an **input object tuple** (x_1, x_2, x_3) to a (potentially infinite) sequence of **output object tuples** [(y_1, y_2), (y'_1, y'_2), …]

```
def stream(x1, x2, x3):
    i = 0
    while True:
        y1 = i*(x1 + x2)
        y2 = i*(x2 + x3)
        yield (y1, y2)
        i += 1
```

[Kaelbling 2011][Srivastava 2014]
[Garrett 2018a][Garrett 2018b]



102

Stream Certified Facts

103

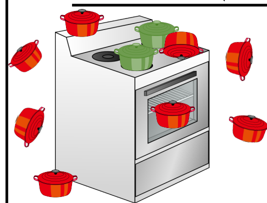
- Objects alone aren't helpful: **what do they represent?**
  - Communicate semantics using **predicates**!
- Augment stream specification with:
  - **Domain facts** - static facts declaring legal **inputs**
    - *e.g.* only configurations can be motion inputs
  - **Certified facts** - static facts that all **outputs** satisfy with their corresponding **inputs**
    - *e.g.* poses sampled from a region are within it

103

Sampling Contained Poses

104

```
(:stream sample-region
  :inputs (?b ?r)
  :domain (and (Block ?b) (Region ?r))
  :outputs (?p)
  :certified (and (Pose ?b ?p) (Contain ?b ?p ?r)))
```



```
def sample_region(b, r):
  x_min, x_max = REGIONS[r]
  w = BLOCKS[b].width
  while True:
      x = random.uniform(x_min + w/2,  x_
      p = np.array([x, 0.])
      yield (p,)
```

Block b → sample-region → Pose [(p), (p'), (p''), …]
Region r → sample-region

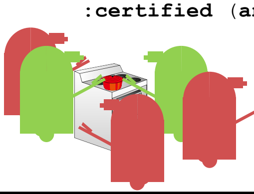
104

Sampling IK Solutions

105

- **Inverse kinematics** (IK) to produce robot grasping configuration
  - Trivial in 2D, non-trial in general (*e.g.* 7 DOF arm)

```
(:stream sample-ik
  :inputs (?b ?p ?g)
  :domain (and (Pose ?b ?p) (Grasp ?b ?g))
  :outputs (?q)
  :certified (and (Conf ?q) (Kin ?b ?p ?g ?q)))
```



Block b → sample-ik → Conf [(q'), (q'')]
Pose p → sample-ik
Grasp g → sample-ik

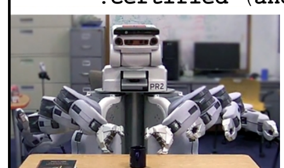
105

Calling a Motion Planner

106

- "Sample" (*e.g.* via a PRM) **multi-waypoint trajectories**
- Include **joint limits & fixed obstacle collisions**, but not movable object collisions

```
(:stream sample-motion
   :inputs (?q1 ?q2)
   :domain (and (Conf ?q1) (Conf ?q2))
   :outputs (?t)
   :certified (and (Traj ?t) (Motion ?q1 ?t ?q2)))
```



Conf q_1 → sample-motion → Trajectory [(t)]
Conf q_2 → sample-motion

106

2D Place Collisions

107

- Add parameters for the pose of each block - **bad!**
- Use a **derived predicate** for whether currently **unsafe**
  - Predicate defined by **logical formula** [Fox 2003] [Thiébaux 2005]
  - Enables lightweight **logical inference**
  - Decomposes collision checking into a logical **AND**

```
(:action place
  :parameters (?b ?p ?g ?q)
  :precondition (and ... (not (UnsafePose ?b ?p)))
  :effect (and ...)

(:derived (UnsafePose ?b1 ?p1)
  (exists (?b2 ?p2) (and (Pose ?b1 ?p1) (Pose ?b2 ?p2)
                         (not (= ?b1 ?b2)) (AtPose ?b2 ?p2)
                         (not (CFree ?b1 ?p1 ?b2 ?p2)))))
```

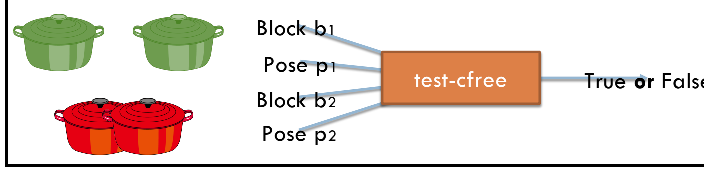
107

Check Block Collisions

108

- **Test stream:** stream without output objects
- Return True if **collision-free** placement (e.g. via querying a collision checker)

```
(:stream test-cfree
  :inputs (?b1 ?p1 ?b2 ?p2)
  :domain (and (Pose ?b1 ?p1) (Pose ?b2 ?p2))
  :outputs ()
  :certified (CFree ?b1 ?p1 ?b2 ?p2))
```

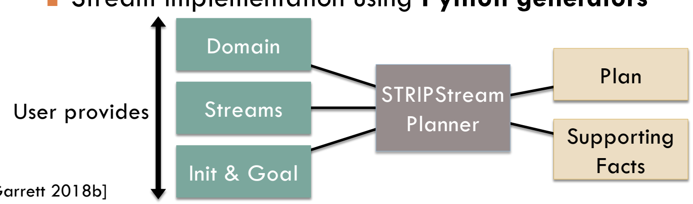


108

STRIPStream = STRIPS + Streams

109

- **Domain dynamics** *(domain.pddl): declares* actions
- **Stream properties** (*stream.pddl*)
  - Declares stream inputs, outputs, and certified facts
- **Problem** and **stream implementation** (*problem.py*)
  - Initial state, **Python constants,** & goal formula
  - Stream implementation using **Python generators**



[Garrett 2018b]

109

STRIPStream Algorithms

110

Two STRIPStream Algorithms

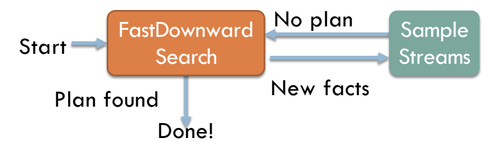
- **STRIPStream planners decide** which streams to use
- Algorithms alternate between **searching & sampling:**
  1. **Search** a finite PDDL problem for plan
  2. **Modify** the PDDL problem (depending on the plan)
- Search implemented using **off-the-shelf algorithms**
  - **Off-the-shelf AI planner** - FastDownward
    - Exploits factoring in its search heuristics (e.g. hFF)
    - http://www.fast-downward.org/
  - **Probabilistically complete** given *sufficient* samplers

[Garrett 2018a]
[Garrett 2018b]

111

Incremental Algorithm

- Incrementally construct all possible initial facts
- Periodically check if a solution exists
- Repeat:
  1. **Compose** and **evaluate** a finite number of streams to unveil more facts in the initial state
  2. **Search** the current PDDL problem for plan
  3. **Terminate** when a plan is found



[Garrett 2018a]
[Garrett 2018b]

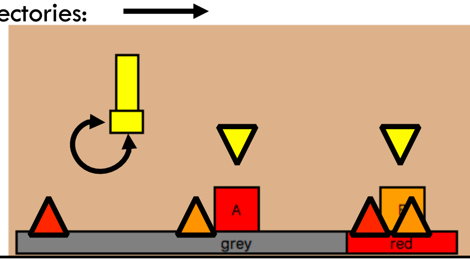
112

Incremental: Sampling Iteration 1

113

Iteration 1 - 14 stream evaluations

- **Sampled:**
  - 2 new robot configurations: 
  - 4 new block poses: 
  - 2 new trajectories: 

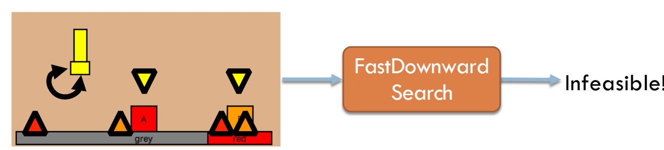


113

Incremental: Search Iteration 1

114

- Pass current discretization to FastDownward
- If **infeasible,** the current set of samples is insufficient



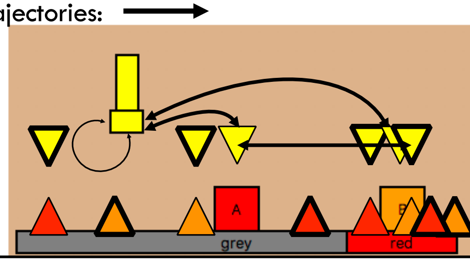
114

Incremental: Sampling Iteration 2

115

Iteration 2 - 54 stream evaluations

- **Sampled:**
  - 4 new robot configurations:
  - 4 new block poses:
  - 10 new trajectories:

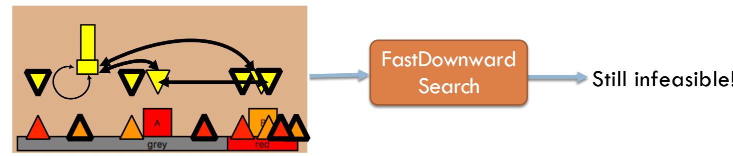


115

Incremental: Search Iteration 2

116

- Pass current discretization to FastDownward
- If **infeasible,** the current set of samples is insufficient



116

Incremental Example: Iterations 3-4

117

Iteration 3 - 118 stream evaluations
Iteration 4 - 182 stream evaluations

Solution:
1) **move** [-7.5 5.] [[-7.5 5.], [-7.5 5.], [7.5 5.], [7.5 2.5]] [7.5 2.5]
2) **pick B** [7.5 0.] [0. -2.5] [7.5 2.5]
3) **move** [7.5 2.5] [[7.5 2.5], [7.5 5.], [10.97 5.], [10.97 2.5]] [10.97 2.5]
4) **place B** [10.97 0.] [0. -2.5] [10.97 2.5]
5) **move** [10.97 2.5] [[10.97 2.5], [10.97 5.], [0. 5.], [0. 2.5]] [0. 2.5]
6) **pick A** [0. 0.] [0. -2.5] [0. 2.5]
7) **move** [0. 2.5] [[0. 2.5], [0. 5.], [7.65 5.], [7.65 2.5]] [7.65 2.5]
8) **place A** [7.65 0.] [0. -2.5] [7.65 2.5]

- **Drawback** - many unnecessary samples produced
  - **Computationally expensive** to generate
  - Induces **large discrete-planning problems**

117

Optimistic Stream Outputs

118

- Many TAMP streams are exceptionally **expensive**
  - Inverse kinematics, motion planning, collision checking
- **Only** query streams that are **identified** as useful
  - Plan with **optimistic hypothetical** outputs [Srivastava 2014]
- Inductively create **unique placeholder** output objects for each stream instance (has **#** as its prefix)

Optimistic evaluations:

1. **s-region:**(b0, red)->(**#p0**)
2. **s-ik:**(b0, [0. 0.], [0. -2.5])->(**#q0**),
3. **s-ik:**(b0, **#p0**, [0. -2.5]) ->(**#q2**)

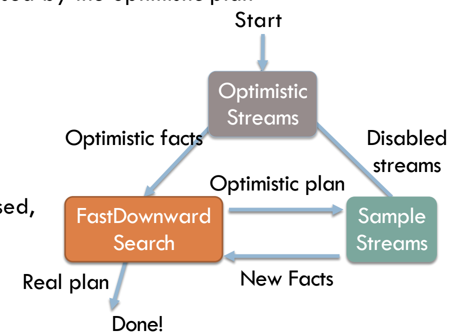
[Garrett 2018a]
[Garrett 2018b]

118

Focused Algorithm

119

- **Lazily** plan using optimistic outputs **before** real outputs
- **Recover** set of streams used by the optimistic plan
- Repeat:
  1. Construct active **optimistic** objects
  2. **Search** with **real & optimistic** objects
  3. If **only real objects** used, **return plan**
  4. **Sample** used streams
  5. **Disable** used streams



[Garrett 2018a][Garrett 2018b]

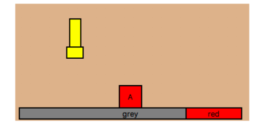
119

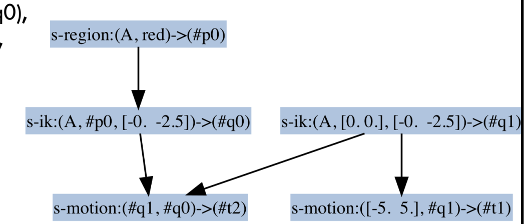
Focused Example 1

120

Optimistic Plan:
move([-5. 5.], #t0, #q0), **pick**(A, [0. 0.], [-0. -2.5], #q0),
move(#q0, #t2, #q1), **place**(A, #p0, [-0. -2.5], #q1)

Constraints:
(kin, A, #q0, #p0, [-0. -2.5]),
(kin, A, #q1, [0. 0.], [-0. -2.5]),
(motion, [-5. 5.], #t1, #q1),
(motion, #q1, #t2, #q0),
(contain, A, #p0, red),





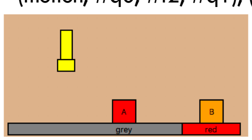
120

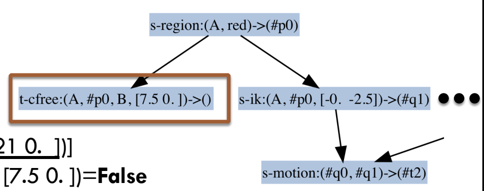
Focused Example 2: Iteration 1

121

Optimistic Plan:
move([-5. 5.], #t0, #q0), **pick**(A, [0. 0.], [-0. -2.5], #q0), **move**(#q0, #t2, #q1), **place**(A, #p0, [-0. -2.5], #q1)

Constraints:
(cfree, A, #p0, B, [7.5 0.]), (contain, A, #p0, red),
(kin, A, #q0, [0. 0.], [-0. -2.5]), (kin, A, #q1, #p0, [-0. -2.5]),
(motion, #q0, #t2, #q1), (motion, [-5. 5.], #t0, #q0)





Stream evaluations:
1. **s-region:**(A, red)->[([8.21 0.])]
2. **t-cfree:**(A, [8.21 0.], B, [7.5 0.])=**False**

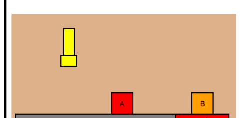
These stream instances are **removed** from subsequent searches

121

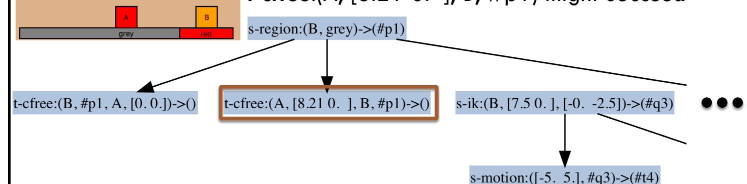
Focused Example: Iteration 2

122

Optimistic Plan:
move([-5. 5.], #t4, #q2), **pick**(B, [7.5 0.], [-0. -2.5], #q2),
move(#q2, #t9, #q3), **place**(B, #p1, [-0. -2.5], #q3),
move(#q3, #t6, #q0), **pick**(A, [0. 0.], [-0. -2.5], #q0),
move(#q0, #t8, #q4), **place**(A, [8.21 0.], [-0. -2.5], #q4)

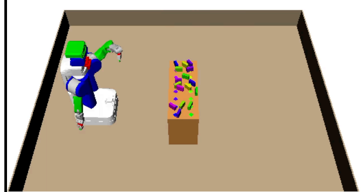


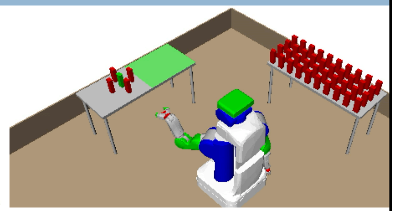
t-cfree:(A, [8.21 0.], B, [7.5 0.]) previously **failed**
t-cfree:(A, [8.21 0.], B, #p1) might **succeed**



122

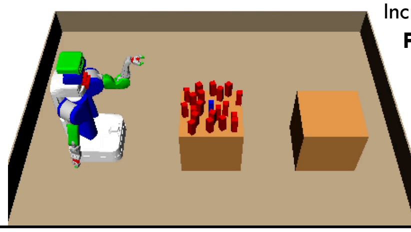
Focused Outperforms Incremental





Incremental ~20 s
Focused ~10 s

Incremental N/A
Focused ~25s

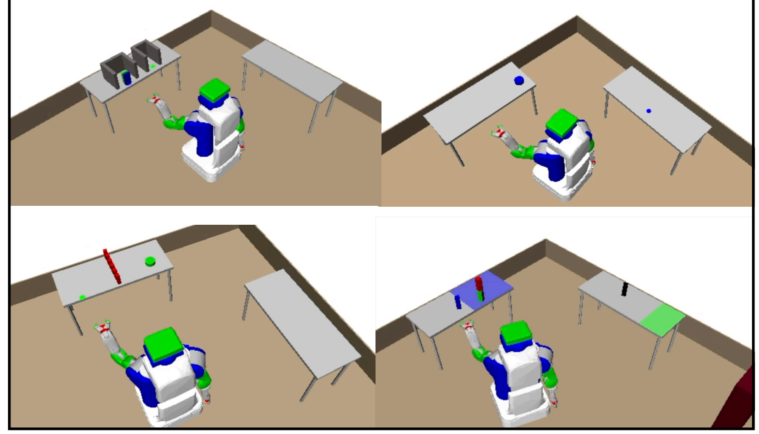


Incremental N/A
Focused ~20s

[Garrett 2018a]

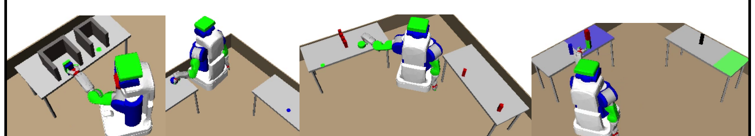
124

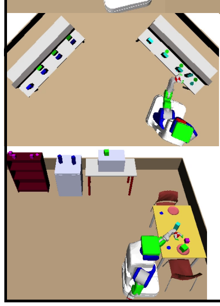
Diverse Experiments



126

Diverse Experiments





	Incr.		*Incr. - H*		*Focus*		*Focus - H*	
Problem	**%**	**t**	**%**	**t**	**%**	**t**	**%**	**t**
Regrasp	98	1	100	2	98	1	95	1
Push	100	11	100	13	100	13	100	9
Wall	95	10	98	13	100	6	100	8
Stacking	100	9	100	9	100	2	100	3
Nonmon.	25	21	98	15	0	-	88	43
Dinner	0	-	100	27	0	-	98	22

Success percentage (%), Average runtime in sec. (t)

127

Cost-Minimizing Planning

128

- Actions costs specified as **nonnegative functions**

```
(:action move                                  (:function (Length ?t)
  :parameters (?q1 ?t ?q2)                         (Traj ?t))
  :precondition (and (Motion ?q1 ?t ?q2)
                     (AtConf ?q1))
  :effect (and (AtConf ?q2)
               (not (AtConf ?q1)))
               (increase (total-cost) (Length ?t))))
```

- Function specification similar to derived predicates

```
def Length(t):
    return sum(np.linalg.norm(q2 - q1)
               for q1, q2 in zip(t[:-1], t[1:]))
```

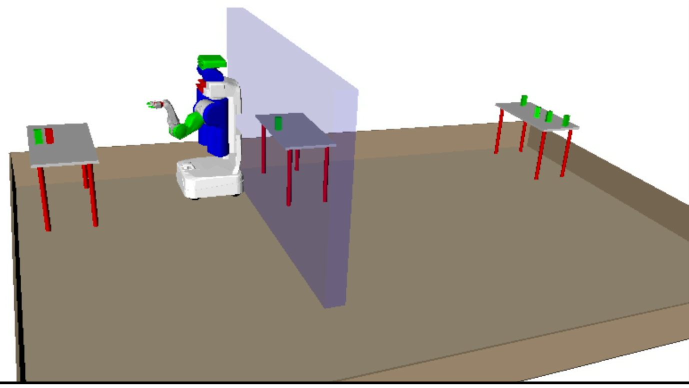
- **Asymptotically optimal** algorithms

128

Goal: Hold Any Green Block

129

- **Lower bounds on costs** improve focused performance



129

TAMP Under Uncertainty

141

A Real World Example

143

In the real world…

Three big problems:

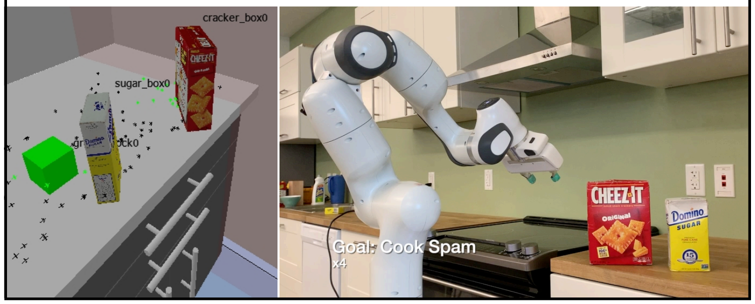
1) Partially observability: we don't have full knowledge of the world; objects not visible.
2) Stochastic actions: objects or the robot will not appear where we expect when grasped, placed, or detected. This means we must be able to re-plan efficiently.
3) Problem complexity: plans are long, with many free parameters. It's impossible to explore all possible results of our actions.

144

POMDP: Partially-Observable State

145

- Update a **belief** (probability distribution) over states
- Plan in the **space of beliefs** (belief space planning)
  - Intentionally take observation actions



145

MDP: Stochastic Action Effects

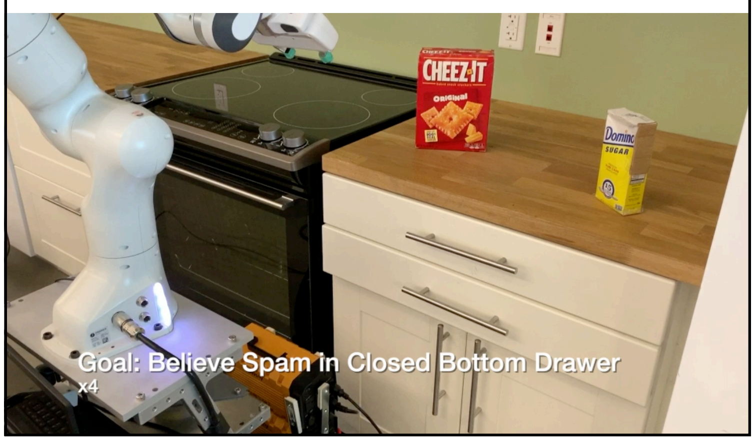
146

- Approximate as cost-sensitive **deterministic** problem
- **Policy** computed online via **replanning**

146

Geometric & Probabilistic Constraints

147



147

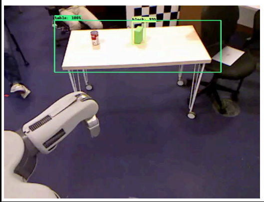
Three Insights

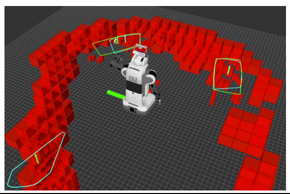
1) **Particle-based belief representation:** allows us to represent belief over where objects might be, given a prior based on the environment. Determinize this belief before planning.
2) **Online replanning:** we constrain new plans to be decreasing in length, and re-use elements of the plan structure. This prevents issues with (1).
3) **Deferred evaluation:** only evaluate *streams*, i.e. find continuous values, for actions that occur before the next time we will need to replan.

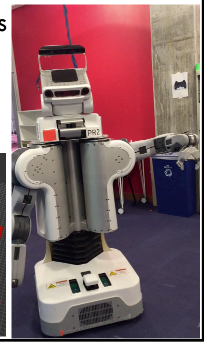
148

Belief-Space TAMP System

- Convolutional Neural Network (**CNN**) Object Detector
- Point cloud **plane estimation** to identify surfaces
- Point cloud **pose estimation** for objects
- **Occupancy grid** for non-manipulable
- Plan, execute, & observe in **real time**







152

Takeaways

- **Task and Motion Planning (TAMP):** hybrid planning where <u>continuous constraints affect discrete decisions</u>
- **Sampling** is powerful for exploring continuous spaces
- **STRIPStream:** planning language that supports **sampling procedures** as blackbox streams
  - **Domain-independent** algorithms
  - **Lazy/optimistic** planning intelligently queries only a small number of samplers (focused algorithm)
- Ongoing work involving **cost-sensitive, multi-agent, probabilistic & partially observable** TAMP

153

Other Directions

154

Robustness and Reactivity

Task-Level Disturbances

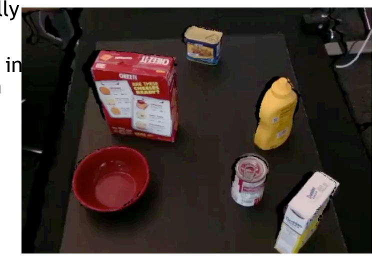
Action-Level Disturbances

Sensors can be wrong, actions can have unpredictable effects, and external factors can interfere with execution.

155

Goal-Based Imitation Learning

- Goal-based imitation allows us to map from human demonstrations to dramatically different environments.
- Take a human demonstration in one environment, use motion planning to determine what their intentions were.



Huang, D., Chao, Y., Paxton, C., Deng, X., Li, F., Naebles, J., Garg, A., Fox, D. Motion Reasoning for Goal-Based Imitation Learning. ICRA 2020

156

Goal-Based Imitation Learning

- Goal-based imitation allows us to map from human demonstrations to dramatically different environments.
- Take a human demonstration in one environment, use motion planning to determine what their intentions were.
- Use the inferred logical goal and execute robustly in the new environment.

Huang, D., Chao, Y., Paxton, C., Deng, X., Li, F., Naebles, J., Garg, A., Fox, D. Motion Reasoning for Goal-Based Imitation Learning. ICRA 2020

157

Deep Planning Domain Learning

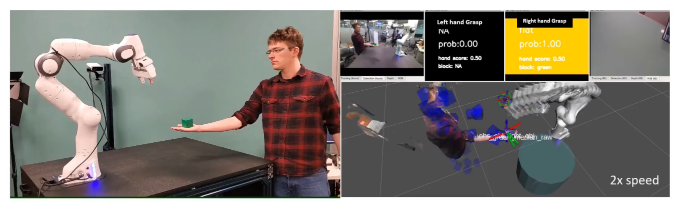
- Learn symbols for high-level task planning, as well as low-level policies for reactive execution.
- Advantage: highly reactive learned hierarchical architecture that can generate plans for held-out tasks.

158

Collaborating with Humans

Human Grasp Classification for Reactive Human-to-Robot Handovers

Wei Yang[*1], Chris Paxton[*1], Maya Cakmak[1,2], and Dieter Fox[1,2]



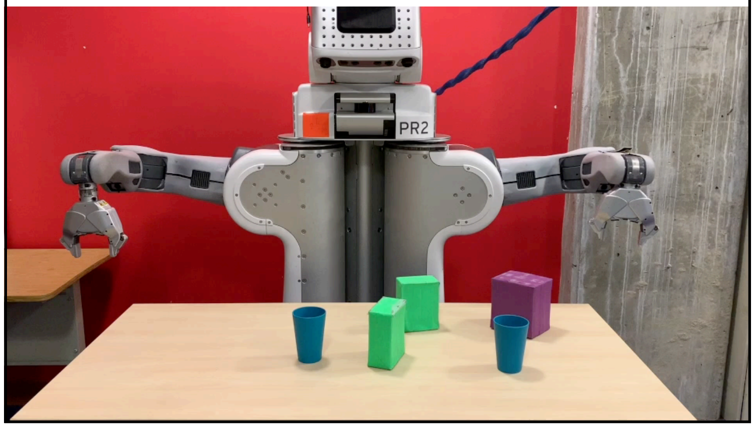
NVIDIA. UNIVERSITY of WASHINGTON

[*] Equal contribution | [1] NVIDIA, USA | [2] University of Washington, USA

159

Questions? (and Outtakes!)

160



160

References

161

161

Task Planning

163

- **[Fikes 1971]** Fikes, R.E. and Nilsson, N.J., 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, *2*(3-4), pp.189-208.
- **[Nilsson 1984]** Nilsson, N.J., 1984. *Shakey the robot*. SRI INTERNATIONAL MENLO PARK CA.
- **[Penberthy 1992]** Penberthy, J.S. and Weld, D.S., 1992. UCPOP: A Sound, Complete, Partial Order Planner for ADL. *Kr*, *92*, pp.103-114.
- **[Aeronautiques 1998]** Aeronautiques, C., Howe, A., Knoblock, C., McDermott, I.D., Ram, A., Veloso, M., Weld, D., SRI, D.W., Barrett, A., Christianson, D. and Friedman, M., 1998. PDDL| The Planning Domain Definition Language.
- **[Kautz 1999]** Kautz, H. and Selman, B., 1999, June. Unifying SAT-based and graph-based planning. In *IJCAI* (Vol. 99, pp. 318-325).
- **[Bonet 2001]** Bonet, B. and Geffner, H., 2001. Planning as heuristic search. *Artificial Intelligence*, *129*(1-2), pp.5-33.
- **[Hoffman 2001]** Hoffmann, J. and Nebel, B., 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, *14*, pp.253-302.
- **[Ghallab 2004]** Ghallab, M., Nau, D. and Traverso, P., 2004. *Automated Planning: theory and practice*. Elsevier.
- **[Thiébaux 2005]** Thiébaux, S., Hoffmann, J. and Nebel, B., 2005. In defense of PDDL axioms. *Artificial Intelligence*, *168*(1-2), pp.38-69.
- **[Helmert 2006]** Helmert, M., 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, *26*, pp.191-246.

163

Motion Planning

164

- **[Lozano-Pérez 1979]** Lozano-Pérez, T. and Wesley, M.A., 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, *22*(10), pp.560-570.
- **[Kavraki 1994]** Kavraki, L., Svestka, P. and Overmars, M.H., 1994. Probabilistic roadmaps for path planning in high-dimensional configuration spaces (Vol. 1994).
- **[Bohlin 2000]** Bohlin, R. and Kavraki, L.E., 2000, April. Path planning using lazy PRM. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065) (Vol. 1, pp. 521-528). IEEE.
- **[Kuffner 2000]** Kuffner Jr, J.J. and LaValle, S.M., 2000, April. RRT-connect: An efficient approach to single-query path planning. In *ICRA* (Vol. 2).
- **[Kuffner 2001]** LaValle, S.M. and Kuffner Jr, J.J., 2001. Randomized kinodynamic planning. *The International Journal of Robotics Research*, *20*(5), pp.378-400.
- **[LaValle 2006]** LaValle, S.M., 2006. *Planning algorithms*. Cambridge university press.
- **[Ratliff 2009]** Ratliff, N., Zucker, M., Bagnell, J.A. and Srinivasa, S., 2009. CHOMP: Gradient optimization techniques for efficient motion planning.
- **[Schulman 2013]** Schulman, J., Ho, J., Lee, A.X., Awwal, I., Bradlow, H. and Abbeel, P., 2013, June. Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization. In *Robotics: science and systems* (Vol. 9, No. 1, pp. 1-10).
- **[Dellin 2016]** Dellin, C.M. and Srinivasa, S.S., 2016, March. A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*.

164

Prediscretized Planning

165

- [**Dornhege 2009**] Dornhege, C., Eyerich, P., Keller, T., Trüg, S., Brenner, M. and Nebel, B., 2009, October. Semantic attachments for domain-independent planning systems. In *Nineteenth International Conference on Automated Planning and Scheduling.*
- [**Erdem 2011**] Erdem, E., Haspalamutgil, K., Palaz, C., Patoglu, V. and Uras, T., 2011, May. Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *2011 IEEE International Conference on Robotics and Automation* (pp. 4575-4581). IEEE.
- [**Lagriffoul 2014**] Lagriffoul, F., Dimitrov, D., Bidot, J., Saffiotti, A. and Karlsson, L., 2014. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, *33*(14), pp.1726-1747.
- [**Lozano-Pérez 2014**] Lozano-Pérez, T. and Kaelbling, L.P., 2014, September. A constraint-based method for solving sequential manipulation planning problems. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3684-3691). IEEE.
- [**Garrett 2017**] Garrett, C.R., Lozano-Perez, T. and Kaelbling, L.P., 2017. FFRob: Leveraging symbolic planning for efficient task and motion planning. *The International Journal of Robotics Research*, *37*(1), pp.104-136.
- [**Ferrer-Mestres 2017**] Ferrer-Mestres, J., Frances, G. and Geffner, H., 2017. Combined task and motion planning as classical AI planning. *arXiv preprint arXiv:1706.06927*.
- [**Dantam 2018**] Dantam, N.T., Kingston, Z.K., Chaudhuri, S. and Kavraki, L.E., 2018. An incremental constraint-based framework for task and motion planning. *The International Journal of Robotics Research*, *37*(10), pp.1134-1151.
- [**Lo 2018**] Lo, S.Y., Zhang, S. and Stone, P., 2018, July. PETLON: Planning Efficiently for Task-Level-Optimal Navigation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (pp. 220-228). International Foundation for Autonomous Agents and Multiagent Systems.
- [**Huang 2018**] Huang, Y., Garrett, C.R. and Mueller, C.T., 2018. Automated sequence and motion planning for robotic spatial extrusion of 3D trusses. *Construction Robotics*, *2*(1-4), pp.15-39.

165

Numeric Planning

166

- [**Fox 2003**] Fox, M. and Long, D., 2003. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*, *20*, pp.61-124.
- [**Hoffmann 2003**] Hoffmann, J., 2003. The Metric-FF Planning System: Translating``Ignoring Delete Lists''to Numeric State Variables. *Journal of artificial intelligence research*, *20*, pp.291-341.
- [**Eyerich 2009**] Eyerich, P., Mattmüller, R. and Röger, G., 2009, October. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Nineteenth International Conference on Automated Planning and Scheduling*.
- [**Deits 2015**] Deits, R. and Tedrake, R., 2015, May. Efficient mixed-integer planning for UAVs in cluttered environments. In *2015 IEEE international conference on robotics and automation (ICRA)* (pp. 42-49). IEEE.
- [**Shoukry 2016**] Shoukry, Y., Nuzzo, P., Saha, I., Sangiovanni-Vincentelli, A.L., Seshia, S.A., Pappas, G.J. and Tabuada, P., 2016, December. Scalable lazy SMT-based motion planning. In *2016 IEEE 55th Conference on Decision and Control (CDC)* (pp. 6683-6688). IEEE.
- [**Fernandez-Gonzalez 2018**] Fernandez-Gonzalez, E., Williams, B. and Karpas, E., 2018. ScottyActivity: Mixed Discrete-Continuous Planning with Convex Optimization. *Journal of Artificial Intelligence Research*, *62*, pp.579-664.

166

Multi-Modal Motion Planning

167

- [**Alami 1994**] Alami, R., Laumond, J.P. and Siméon, T., 1994. Two manipulation planning algorithms. In *WAFR Proceedings of the workshop on Algorithmic foundations of robotics* (pp. 109-125). AK Peters, Ltd. Natick, MA, USA.
- [**Siméon 2004**] Siméon, T., Laumond, J.P., Cortés, J. and Sahbani, A., 2004. Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, *23*(7-8), pp.729-746.
- [**Hauser 2011**] Hauser, K. and Ng-Thow-Hing, V., 2011. Randomized multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research*, *30*(6), pp.678-698.
- [**Barry 2013**] Barry, J., Kaelbling, L.P. and Lozano-Pérez, T., 2013, May. A hierarchical approach to manipulation with diverse actions. In *2013 IEEE International Conference on Robotics and Automation* (pp. 1799-1806). IEEE.
- [**Toussaint 2015**] Toussaint, M., 2015, June. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.
- [**Vega-Brown 2016**] Vega-Brown, W. and Roy, N., 2016, December. Asymptotically optimal planning under piecewise-analytic constraints. In *Workshop on the Algorithmic Foundations of Robotics*.
- [**Toussaint 2018**] Toussaint, M., Allen, K., Smith, K.A. and Tenenbaum, J.B., 2018. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning. In *Robotics: Science and Systems*.

167

Task and Motion Planning

168

- [**Gravot 2005**] Gravot, F., Cambon, S. and Alami, R., 2005. aSyMov: a planner that deals with intricate symbolic and geometric problems. In *Robotics Research. The Eleventh International Symposium* (pp. 100-110). Springer, Berlin, Heidelberg.
- [**Plaku 2010**] Plaku, E. and Hager, G.D., 2010, May. Sampling-based motion and symbolic action planning with geometric and differential constraints. In *2010 IEEE International Conference on Robotics and Automation* (pp. 5002-5008). IEEE.
- [**Kaelbling 2011**] Kaelbling, L. P. and Lozano-Pérez, T. Hierarchical task and motion planning in the now. *2011 IEEE International Conference on Robotics and Automation*, Shanghai, 2011, pp. 1470-1477.
- [**De Silva 2013**] De Silva, L., Pandey, A.K., Gharbi, M. and Alami, R., 2013. Towards combining HTN planning and geometric task planning. *arXiv preprint arXiv:1307.1482.*
- [**Srivastava 2014**] Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S. and Abbeel, P., 2014, May. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)* (pp. 639-646). IEEE.
- [**Garrett 2018a**] Garrett, C.R., Lozano-Pérez, T. and Kaelbling, L.P., 2018. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research*, *37*(13-14), pp.1796-1825.
- [**Garrett 2018b**] Garrett, C.R., Lozano-Pérez, T. and Kaelbling, L.P., 2018. STRIPStream: Integrating Symbolic Planners and Blackbox Samplers. *arXiv preprint arXiv:1802.08705.*

168

Probabilistic & Partially-Observable

169

- [**Kaelbling 1998**] Kaelbling, L.P., Littman, M.L. and Cassandra, A.R., 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, *101*(1-2), pp.99-134.
- [**Kocsis 2006**] Kocsis, L. and Szepesvári, C., 2006, September. Bandit based monte-carlo planning. In *European conference on machine learning* (pp. 282-293). Springer, Berlin, Heidelberg.
- [**Yoon 2007**] Yoon, S.W., Fern, A. and Givan, R., 2007, September. FF-Replan: A Baseline for Probabilistic Planning. In *ICAPS* (Vol. 7, pp. 352-359).
- [**Silver 2010**] Silver, D. and Veness, J., 2010. Monte-Carlo planning in large POMDPs. In *Advances in neural information processing systems* (pp. 2164-2172).
- [**Platt 2010**] Platt Jr, R., Tedrake, R., Kaelbling, L. and Lozano-Perez, T., 2010. Belief space planning assuming maximum likelihood observations.
- [**Kaelbling 2013**] Kaelbling, L.P. and Lozano-Pérez, T., 2013. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, *32*(9-10), pp.1194-1227.
- [**Hadfield-Menell 2015**] Hadfield-Menell, D., Groshev, E., Chitnis, R. and Abbeel, P., 2015, September. Modular task and motion planning in belief space. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4991-4998). IEEE.

169