

Plan and Goal Recognition

SICSA DVF Master Class

Felipe Meneguzzi

Pontifical Catholic University of Rio Grande do Sul, Brazil
felipe.meneguzzi@pucrs.br

Aberdeen, October, 2017

Table of Contents

- 1 Introduction
 - Motivation and Intuition
 - Formalism
- 2 Goal Recognition as reasoning over Heuristics
 - Motivation and Background
 - Estimating Goal Completion with Landmarks
 - Results
- 3 Online Goal Recognition as Reasoning over Landmarks
- 4 Goal Recognition in Incomplete Domains
 - Motivation and Background
 - Landmarks in Incomplete Domains
 - Recognizing Goals in Incomplete Domains
 - Results
- 5 Real World Applications
 - Optimality Monitoring and Plan Abandonment
 - Plan Recognition using Video Data
- 6 Summary and Future Directions

Introduction

Motivation and Intuition

- Recognizing plans and goals of others is a critical ability for intelligent interaction:
 - important for humans/agents working in the same environment
 - increasingly important as we build more intelligent systems
- Overall area of Plan, Activity and Intent Recognition
 - Activity recognition: recognizing meaningful activities from low-level sensor data
 - Plan/Intent/Goal recognition: recognizing intentional higher-level sequences of activities

- **Goal Recognition** is the task of recognizing agents' goal that explains a sequence of observations of its actions;
 - Related to plan recognition, i.e. recognizing a *top-level* action
 - A specific form of the problem of abduction
- Approaches to goal and plan recognition divided into roughly two types:
 - Plan-library based (*classical* plan recognition)
 - Domain-theory based (plan recognition as planning, or PRAP)

An example of Activity Recognition



An example of Activity Recognition



An example of Activity Recognition














An example of Activity Recognition



breaking egg

An Example of Goal/Plan Recognition

from Miquel Ramirez's thesis

	A	B	C	D	E
0		 ₁			 ₅
1	 ₂				
2	 ₂		 ₄	 ₁	
3				 ₆	 ₇
4	 ₃		 ₃		

Wooden pieces p_1, p_2, \dots, p_n

Pieces have shapes and colors












Bins b_1, b_2, \dots, b_n

The possible **goals** the trainer expected to pursue:

- ① Store all triangles in b_1
- ② Store all spheres in b_2
- ③ Store all cubes in b_3
- ④ Store red objects in b_2
- ⑤ Store green objects in b_3
- ⑥ Store blue objects in b_1

An Example of Goal/Plan Recognition

from Miquel Ramirez's thesis

	A	B	C	D	E
0		 ₁			 ₅
1	 ₂				
2	 ₂		 ₄	 ₁	
3				 ₆	 ₇
4	 ₃		 ₃		

Wooden pieces p_1, p_2, \dots, p_n

Pieces have shapes and colors

Bins b_1, b_2, \dots, b_n

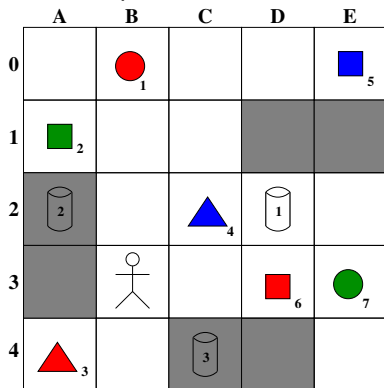
One possible *plan* for the trainer to achieve task #1

(store all triangles in b_1):

- ① Walk from B3 into A4
- ② Pick p_3 up
- ③ Walk from A4 into B3
- ④ Walk from B3 into C2
- ⑤ Pick p_4 up
- ⑥ Throw p_3 into b_1
- ⑦ Throw p_4 into b_1

An Example of Goal/Plan Recognition

from Miquel Ramirez's thesis



If sensors miss 70% of *walk* actions and half *pick* and *drop* actions, we may only see:

- ① Pick p_3 up
- ② Walk from A4 into B3












Wooden pieces p_1, p_2, \dots, p_n

Pieces have shapes and colors

Bins b_1, b_2, \dots, b_n

An Example of Goal/Plan Recognition

from Miquel Ramirez's thesis

	A	B	C	D	E
0		 ₁			 ₅
1	 ₂				
2	 ₂		 ₄	 ₁	
3				 ₆	 ₇
4	 ₃		 ₃		

Wooden pieces p_1, p_2, \dots, p_n

Pieces have shapes and colors

Bins b_1, b_2, \dots, b_n

If sensors miss 70% of *walk* actions and half *pick* and *drop* actions, we may only see:

- ① Pick p_3 up
- ② Walk from A4 into B3

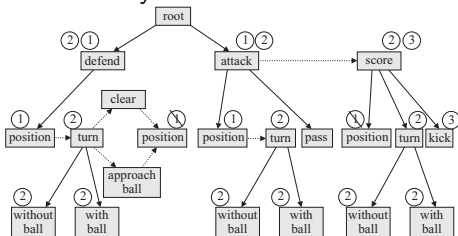
Here, we could deduce either task #1 or #4 (store all red objects in b_2), as other tasks are less *likely*.

Introduction

Formalism

Flavors of Recognition Formalism

Plan Library



Domain Theory (PRAP)

```

(define (domain grid)
  (:requirements :strips :typing)
  (:types place shape key)
  (:predicates
    (conn ?x ?y — place)
    (key—shape ?k — key ?s — shape)
    (lock—shape ?x — place ?s — shape)
    (at ?r — key ?x — place )
    (at—robot ?x — place)
    (locked ?x — place)
    (carrying ?k — key)
    (open ?x — place)
  )

  (:action unlock
    :parameters (?curpos ?lockpos — place ?key — key ?shape — shape)
    :precondition (and (conn ?curpos ?lockpos) (key—shape ?key ?shape)
      (lock—shape ?lockpos ?shape) (at—robot ?curpos)
      (locked ?lockpos) (carrying ?key))
    :effect (and (open ?lockpos) (not (locked ?lockpos))))

  (:action move
    :parameters (?curpos ?nextpos — place)
    :precondition (and (at—robot ?curpos) (conn ?curpos ?nextpos) (open ?nextpos))
    :effect (and (at—robot ?nextpos) (not (at—robot ?curpos))))

  (:action pickup
    :parameters (?curpos — place ?key — key)
    :precondition (and (at—robot ?curpos) (at ?key ?curpos))
    :effect (and (carrying ?key)
      (not (at ?key ?curpos))))
  )
  )
  
```

Definition (**Planning**)

A planning instance is represented by a triple $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, in which:

- $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is the **domain definition**, and consists of a finite set of **facts** Σ and a finite set of **actions** \mathcal{A} (action costs typically 1);
- $\mathcal{I} \subseteq \Sigma$ and $G \subseteq \Sigma$ represent the **planning problem**, in which $\mathcal{I} \subseteq \Sigma$ is the **initial state**, and $G \subseteq \Sigma$ is the **goal state**.

- Actions $a \in \mathcal{A}$ are tuples $a = \langle name, pre(a), eff(a), cost(a) \rangle$
- Facts Σ can be modeled in a variety of ways:
 - As a logic language (restricted FOL):
states are truth assignments
 - As a set of variables \mathcal{V} with finite domains:
states are variable assignments

Roboschool Domain in PDDL

```
(define (domain roboschool)
  (:requirements :typing
                 :negative-preconditions)

  (:types room piece bin - object)

  (:predicates (at ?room - room)
               (in ?object ?place - object)
               (has ?piece - piece)
               (connected ?r1 ?r2 - room)
               (color ?piece ?color - object)
               (shape ?piece ?shape - object)
               (blocked ?room - room)
               (encumbered))

  (:action pick
    :parameters (?p - piece ?r - room)
    :precondition (and (at ?r) (in ?p ?r)
                       (not (encumbered)))
    :effect (and
             (has ?p)
             (not (in ?p ?r))
             (encumbered))
  )

  (:action throw
    :parameters (?p - piece
                 ?r1 ?r2 - room ?b - bin)
    :precondition (and (at ?r1) (in ?b ?r2)
                       (has ?p) (connected ?r1 ?r2))
    :effect (and
             (in ?p ?b)
             (not (has ?p))
             (not (encumbered))
            )
  )

  (:action walk
    :parameters (?r1 ?r2 - room)
    :precondition (and (at ?r1)
                       (connected ?r1 ?r2)
                       (not (blocked ?r2)))
    :effect (and
             (at ?r2)
             (not (at ?r1))
            )
  )
)
```

Roboschool Problem in PDDL

```
(define (problem pb2)
  (:domain roboschool)
  (:objects p1 p2 p3 p4 p5 p6 p7 — piece
    b1 b2 b3 — bin
    roomA_0 roomA_1 roomA_2 roomA_3 roomA_4 — room
    roomB_0 roomB_1 roomB_2 roomB_3 roomB_4 — room
    roomC_0 roomC_1 roomC_2 roomC_3 roomC_4 — room
    roomD_0 roomD_1 roomD_2 roomD_3 roomD_4 — room
    roomE_0 roomE_1 roomE_2 roomE_3 roomE_4 — room)

  (:init
    (connected roomA_0 roomB_0) (connected roomB_0 roomA_0)
    (connected roomA_0 roomA_1) (connected roomA_1 roomA_0)
    (connected roomA_1 roomB_1) (connected roomB_1 roomA_1)
    (connected roomA_1 roomA_2) (connected roomA_2 roomA_1)
    (connected roomA_2 roomB_2) (connected roomB_2 roomA_2)
    (connected roomA_2 roomA_3) (connected roomA_3 roomA_2)
    (connected roomA_3 roomB_3) (connected roomB_3 roomA_3)
    (connected roomA_3 roomA_4) (connected roomA_4 roomA_3)
    (connected roomA_4 roomB_4) (connected roomB_4 roomA_4)
    (connected roomB_0 roomC_0) (connected roomC_0 roomB_0)
    (connected roomB_0 roomB_1) (connected roomB_1 roomB_0)
    (connected roomB_1 roomC_1) (connected roomC_1 roomB_1)
    (connected roomB_1 roomB_2) (connected roomB_2 roomB_1)
    (connected roomB_2 roomC_2) (connected roomC_2 roomB_2)
    (connected roomB_2 roomB_3) (connected roomB_3 roomB_2)
    (connected roomB_3 roomC_3) (connected roomC_3 roomB_3)
    (connected roomB_3 roomB_4) (connected roomB_4 roomB_3)
    (connected roomB_4 roomC_4) (connected roomC_4 roomB_4)
    (...))
    (blocked roomA_2) (blocked roomA_3) (blocked roomD_1)
    (blocked roomE_1) (blocked roomC_4) (blocked roomD_4)

    (at roomB_3)
    (in b1 roomD_2) (in b2 roomA_2) (in b3 roomC_4)
    (in p1 roomB_0) (color p1 red) (shape p1 circle)
    (in p2 roomA_1) (color p2 green) (shape p2 square)
    (in p3 roomA_4) (color p3 red) (shape p3 triangle)
    (in p4 roomC_2) (color p4 blue) (shape p4 triangle)
    (in p5 roomE_0) (color p5 blue) (shape p5 square)
    (in p6 roomD_3) (color p6 red) (shape p6 square)
    (in p7 roomE_3) (color p7 green) (shape p7 circle)

    ; All spheres in b2
    (:goal (and (in p1 b2) (in p7 b2)))
  )
)
```

Solving Planning Problems

Actions $a \in \mathcal{A}$ are tuples $a = \langle name, pre(a), eff(a), cost(a) \rangle$. They induce a state transition system such that:

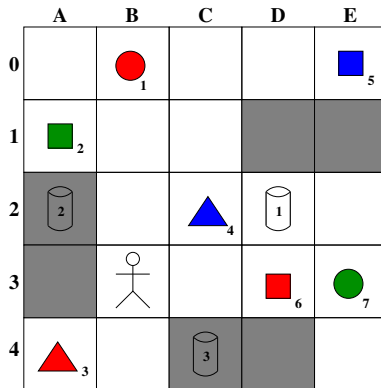
- $eff(a)$ constitute a set of negative (delete) and positive (add) effects:
 $eff^+(a), eff^-(a)$
- an action a is applicable to a state s iff $s \models pre(a)$
- applying a to s ($\gamma(s, a)$) yields a state $s' = (s/eff^-(a)) \cup eff^+(a)$,
i.e. it is such that $s' \models eff(a)$

The solution to a planning problem $\Pi = \langle \Xi, \mathcal{I}, G \rangle$ is a plan $\pi = \langle a_1, \dots, a_n \rangle$ such that:

- $\mathcal{I} \models pre(a_1)$
- $\gamma(\gamma(\dots \gamma(\mathcal{I}, a_1), \dots), a_n) \models G$
- The cost of π is $\sum_{i=1}^n cost(a_i)$

Roboschool PDDL Plan

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
; cost = 15 (unit cost)
```



Goals in PDDL

```
; All triangles in b1
(:goal (and (in p3 b1) (in p4 b1)))

; All spheres in b2
(:goal (and (in p1 b2) (in p7 b2)))

; All cubes in b3
(:goal (and (in p2 b3) (in p5 b3)
            (in p6 b3)))

(walk roomb.3 roomb.4)
(walk roomb.4 rooma.4)
(pick p3 rooma.4)
(walk rooma.4 roomb.4)
(walk roomb.4 roomb.3)
(walk roomb.3 roomb.2)
(walk roomb.2 roomc.2)
(throw p3 roomc.2 roomd.2 b1)
(pick p4 roomc.2)
(throw p4 roomc.2 roomd.2 b1)
; cost = 10 (unit cost)

(walk roomb.3 roomc.3)
(walk roomc.3 roomd.3)
(walk roomd.3 roomc.3)
(pick p7 roomc.3)
(walk roomc.3 roomd.3)
(walk roomd.3 roomc.3)
(walk roomc.3 roomb.3)
(walk roomb.3 roomb.2)
(throw p7 roomb.2 rooma.2 b2)
(walk roomb.2 roomb.1)
(walk roomb.1 roomb.0)
(pick p1 roomb.0)
(walk roomb.0 rooma.0)
(walk rooma.0 rooma.1)
(throw p1 rooma.1 rooma.2 b2)
; cost = 15 (unit cost)

(walk roomb.3 roomb.2)
(walk roomb.2 roomb.1)
(walk roomb.1 rooma.1)
(pick p2 rooma.1)
(walk rooma.1 roomb.1)
(walk roomb.1 roomc.1)
(walk roomc.1 roomc.2)
(walk roomc.2 roomc.3)
(throw p2 roomc.3 roomc.4 b3)
(walk roomc.3 roomc.2)
(walk roomc.2 roomc.1)
(walk roomc.1 roomc.0)
(walk roomc.0 roomd.0)
(walk roomd.0 roomc.0)
(pick p5 roomc.0)
(walk roomc.0 roomd.0)
(walk roomd.0 roomc.0)
(walk roomc.0 roomc.1)
(walk roomc.1 roomc.2)
(walk roomc.2 roomc.3)
(throw p5 roomc.3 roomc.4 b3)
(walk roomc.3 roomd.3)
(pick p6 roomd.3)
(walk roomd.3 roomc.3)
(throw p6 roomc.3 roomc.4 b3)
; cost = 25 (unit cost)
```

Goal Recognition Problem

Definition (**Goal Recognition Problem**)

A goal recognition problem is a tuple $P_G = \langle \Xi, \mathcal{I}, \mathcal{G}, \mathbf{O} \rangle$, where:

- $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is the domain definition (facts and actions) ;
 - $\mathcal{I} \subseteq \Sigma$ is the initial state;
 - \mathcal{G} s.t. $\forall G \in \mathcal{G}, G \subseteq \Sigma$ is a set of candidate goals (with an assumed hidden goal G); and
 - \mathbf{O} is a sequence $\langle o_1, \dots, o_n \rangle$ of observations, where $o_i \in \mathcal{A}$
-
- The solution for a goal recognition problem is the hidden goal $G \in \mathcal{G}$ that is most consistent with observation sequence \mathbf{O} .
 - Caveat: we may have other representations for the observations
 - This is what I will refer to as PRAP

Plan Recognition Problem

Definition (Plan Recognition Problem)

A plan recognition problem is a tuple $P_P = \langle \Xi, \mathcal{M}, \mathcal{I}, \mathbf{O} \rangle$, where:

- $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is the domain definition (facts and actions) ;
- \mathcal{M} is a plan-library containing task-decomposing methods of the form $\langle t, \langle t_1, \dots t_n \rangle \rangle$ (implicitly defining a set \mathcal{T} of task symbols); and
- $\mathcal{I} \subseteq \Sigma$ is the initial state;
- \mathbf{O} is a sequence $\langle o_1, \dots o_n \rangle$ of observations, where $o_i \in \mathcal{A}$

Here, \mathcal{T} comprises non-primitive tasks and primitive tasks corresponding to the names of actions $a \in \mathcal{A}$.

- The solution to a plan recognition goal is the top level task $t_l \in \mathcal{T}$ that generates a decomposition consistent with the observations in \mathcal{O}
- Requires a lot more domain knowledge (i.e. the plan-library)
- This is what I will refer to as classical plan recognition

Observations

- Observations often refer to *observed action symbols*
 - This observation sequence can be either **partial** or **full**
 - Observation sequence can be **noisy**
 - Observations can also be of **states/fluents** rather than actions

Original Plan

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
; cost = 15 (unit cost)
```


Observations

- Observations often refer to *observed action symbols*
 - This observation sequence can be either **partial** or **full**
 - Observation sequence can be **noisy**
 - Observations can also be of **states/fluents** rather than actions

Original Plan

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
; cost = 15 (unit cost)
```

Full Observation

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
```

Observations

- Observations often refer to *observed action symbols*
 - This observation sequence can be either **partial** or **full**
 - Observation sequence can be **noisy**
 - Observations can also be of **states/fluents** rather than actions

Original Plan

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
; cost = 15 (unit cost)
```

Partial Observation

```
(walk roomb_3 roomc_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
```

Observations

- Observations often refer to *observed action symbols*
 - This observation sequence can be either **partial** or **full**
 - Observation sequence can be **noisy**
 - Observations can also be of **states/fluents** rather than actions

Original Plan

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
; cost = 15 (unit cost)
```

Noisy Observation

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_2 rooma_2)
(walk rooma_2 rooma_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
```

Observations

- Observations often refer to *observed action symbols*
 - This observation sequence can be either **partial** or **full**
 - Observation sequence can be **noisy**
 - Observations can also be of **states/fluents** rather than actions

Original Plan

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
; cost = 15 (unit cost)
```

Noisy Partial Observation

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(walk roomb_2 roomb_1)
(walk roomb_2 rooma_2)
(walk rooma_2 rooma_1)
(walk roomb_1 roomb_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
```

Observations

- Observations often refer to *observed action symbols*
 - This observation sequence can be either **partial** or **full**
 - Observation sequence can be **noisy**
 - Observations can also be of **states/fluents** rather than actions

Original Plan

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
; cost = 15 (unit cost)
```

State Observation

```
(at roomB_3) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomc_3) (at roomB_4) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomd_3) (encumbered) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roome_3) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roome_3) (has p7) (encumbered) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomd_3) (encumbered) (has p7) (encumbered) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomc_3) (has p7) (encumbered) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomb_3) (has p7) (encumbered) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomb_2) (has p7) (encumbered) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomb_1) (in p7 b2) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomb_0) (in p7 b2) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomb_0) (has p1) (encumbered) (in p7 b2) (in p2 roomA_1) (in p3 roomA_4)
(at rooma_0) (has p1) (encumbered) (in p7 b2) (in p2 roomA_1) (in p3 roomA_4)
(at rooma_1) (has p1) (encumbered) (in p7 b2) (in p2 roomA_1) (in p3 roomA_4)
(at rooma_1) (in p1 b2) (in p7 b2) (in p2 roomA_1) (in p3 roomA_4)
```

Observations

- Observations often refer to *observed action symbols*
 - This observation sequence can be either **partial** or **full**
 - Observation sequence can be **noisy**
 - Observations can also be of **states/fluents** rather than actions

Original Plan

```
(walk roomb_3 roomc_3)
(walk roomc_3 roomd_3)
(walk roomd_3 roome_3)
(pick p7 roome_3)
(walk roome_3 roomd_3)
(walk roomd_3 roomc_3)
(walk roomc_3 roomb_3)
(walk roomb_3 roomb_2)
(throw p7 roomb_2 rooma_2 b2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomb_0)
(pick p1 roomb_0)
(walk roomb_0 rooma_0)
(walk rooma_0 rooma_1)
(throw p1 rooma_1 rooma_2 b2)
; cost = 15 (unit cost)
```

Noisy State Observation

```
(at roomB_3) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomc_3) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roomd_3) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roome_3) (in p1 roomB_0) (in p2 roomA_1) (in p3 roomA_4)
(at roome_3) (has p7) (encumbered) (in p1 roomB_0) (in p2 r
(at roomd_3) (has p7) (encumbered) (in p1 roomB_0) (in p2 r
(at roomc_3) (has p7) (encumbered) (in p1 roomB_0) (in p2 r
(at roomb_3) (has p7) (encumbered) (in p1 roomB_0) (in p2 r
(at roomb_2) (has p7) (encumbered) (in p1 roomB_0) (in p2 r
(at roomb_1) (in p7 b2) (in p1 roomB_0) (in p2 roomA_1) (in
(at roomb_0) (in p7 b2) (in p1 roomB_0) (in p2 roomA_1) (in
(at roomb_0) (has p1) (encumbered) (in p7 b2) (in p2 roomA
(at rooma_0) (has p1) (encumbered) (in p7 b2) (in p2 roomA
(at rooma_1) (has p1) (encumbered) (in p7 b2) (in p2 roomA
(at rooma_1) (in p1 b2) (in p7 b2) (in p2 roomA_1) (in p3 r
```

Table of Contents

- 1 Introduction
 - Motivation and Intuition
 - Formalism
- 2 Goal Recognition as reasoning over Heuristics
 - Motivation and Background
 - Estimating Goal Completion with Landmarks
 - Results
- 3 Online Goal Recognition as Reasoning over Landmarks
- 4 Goal Recognition in Incomplete Domains
 - Motivation and Background
 - Landmarks in Incomplete Domains
 - Recognizing Goals in Incomplete Domains
 - Results
- 5 Real World Applications
 - Optimality Monitoring and Plan Abandonment
 - Plan Recognition using Video Data
- 6 Summary and Future Directions

Goal Recognition as reasoning over Heuristics

Motivation and Background

- Plan Recognition as Planning (PRAP) requires much less domain knowledge
No need for a plan library
- Existing approaches are accurate, but very inefficient
- Key Insight: Landmark-based heuristics are efficient and informative
 - landmarks provide evidence of **expected sequence** of observations
 - can be computed once before recognition time

Previous Work: Ramirez and Geffner (2009 and 2010)

- First approaches to goal recognition: Plan Recognition as Planning (PRAP)
- Probabilistic model aims to compute $P(G \mid O)$
- Following Bayes Rule $P(G \mid O) = \alpha P(O \mid G)P(G)$
- Given $P(G)$ as a prior, key bottleneck is computing $P(O \mid G)$
 - In their work $P(O \mid G)$ is computed in terms of a cost difference $c(G, O) - c(G, \bar{O})$
 - Computational cost is **two planner calls per goal hypothesis**
 - For online recognition: two planner calls per goal hypothesis **per observation**
- Some conclusions challenged for path planning domains (Masters and Sardina 2017)

Previous Work: E-Martín et al. (2015)

- Improvement over Ramirez and Geffner's approach:
 - Relaxes assumption that $c(G \mid O) < c(G, \bar{O})$
 - Based on pre-computation of cost estimates using a planning graph
 - Often orders of magnitude faster than Ramirez
- Performance degrades substantially with low observability
- Does not cope with noisy observations

- Formalizes PRAP with explicit assumptions and notions of
 - noisy observations
 - incomplete observations
 - action costs
- Solves the goal recognition problem using a combination of:
 - transformation of the recognition problem into a new P' planning problem
 - approximation of $P(G \mid O)$ by generating “diverse plans” for P'
- Very accurate recognition for some domains
- No runtime efficiency evaluation (likely to be slow)

- In this work, we use a **planning domain definition** to represent agent behavior and environment properties;
- Previous approaches involve multiple calls to a modified planner.
- Our main contribution is twofold:
 - We **obviate the need to execute a planner multiple times** for recognizing goals; and
 - We develop novel goal recognition heuristics that **use planning landmarks**.
- We show that our approaches are **more accurate** and **orders of magnitude faster** than Ramírez and Geffner's approach.

From STRIPS Problem P to state model S(P)

Most modern planning algorithms convert a planning problem $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, with $\Xi = \langle \Sigma, \mathcal{A} \rangle$ into one of state space search, where:

- the states $s \in S$ are **collections of atoms** from Σ
- the initial state $s_0 = I$
- the goal states S_g are such that $\forall s \in S_g s \models G$
- the applicable actions for a state s in $A(s)$ are actions from $a \in \mathcal{A}$ such that $s \models \text{pre}(a)$
- the successor state $s' = (s / \text{eff}^-(a)) \cup \text{eff}^+(a)$
- all action costs are assumed to be 1

How do we solve these problems?

Heuristic Search Planning

- Explicitly **searches** graph associated with model $S(P)$ with **heuristic** $h(s)$ that estimates cost from s to goal
- **Key idea:** Heuristic h extracted automatically from problem Π
- This is the mainstream approach in classical planning (and other forms of planning as well), enabling the solution of problems over **huge spaces**

Heuristic Graph Search

```
1: function GRAPHSEARCH(problem  $p$ , strategy  $s$ )
2:    $closed \leftarrow \{\}$  ▷ We now keep a list of explored states
3:    $frontier.ADD(newNode(p.initial))$ 
4:   loop
5:     if  $frontier$  is empty then
6:       return fail
7:      $n \leftarrow s.REMOVECHOICE(frontier)$ 
8:      $closed.ADD(n.state)$  ▷ Where we keep states we already visited
9:     if  $P.GOALTEST(n.state)$  then
10:      return  $GETPATH(n)$ 
11:     for all  $a \in p.ACTIONS(n)$  do
12:        $n' \leftarrow a.RESULT(n.state)$ 
13:       if  $n'.state \notin closed$  then
14:          $frontier.ADD(n')$  ▷ And only explore states we haven't visited
```


- A heuristic function **estimates** the true cost of reaching a goal G
- Only requirement for an heuristic $h(n)$ to **work** with A^* is that $h(G) = 0$
 - Though this is **not** necessarily **optimal**

Heuristic Functions

7	2	4
5		6
8	3	1

Start State

- Heuristics

1	2	3
4	5	6
7	8	

Goal State

Heuristic Functions

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

- Heuristics
 - The number of misplaced tiles – **Hamming distance** ($h_1 = 7$)
 - The sum of distances of the tiles from their goal positions – **Manhattan distance** ($h_2 = 16$)
- Are they admissible?

Creating Heuristics

- Ignore specific preconditions of actions (need some domain knowledge)
- Consider the sliding blocks from search

```
(:action slide
  :parameters (?t — tile ?s1 ?s2 — slot)
  :precondition (and (on ?t ?s1) (blank ?s2)
                    (adjacent ?s1 ?s2) )
  :effect (and (on ?t ?s2) (blank ?s1)
              (not (on ?t ?s1)) (not (blank ?s2)) )
)
```

- If we remove (blank s2), we are left with Manhattan distance

- h_2 is always greater than h_1 .
 - It therefore dominates h_1 (it is a tighter bound on the actual cost)
 - And is more efficient
- How can we invent good heuristics?
- Typically, by **relaxing** the problem.

Heuristics for Classical Planning

- Key development in planning in the 90's, is automatic extraction of **heuristic functions** to guide search for plans
- The general idea was known: heuristics often **explained** as **optimal** cost functions of **relaxed** (simplified) problems (Minsky 61; Pearl 83)
- Most common relaxation in planning, P^+ , obtained by dropping **delete-lists** from ops in P . If $c^*(P)$ is optimal cost of P , then

$$h^+(P) \stackrel{\text{def}}{=} c^*(P^+)$$

- Heuristic h^+ **intractable** but easy to **approximate**; i.e.
 - computing **optimal** plan for P^+ is **intractable**, but
 - computing a non-optimal plan for P^+ (**relaxed plan**) easy
- State-of-the-art heuristics as in FF or LAMA still rely on P^+

Additive Heuristic

- For all **atoms** p :

$$h(p; s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } p \in s, \text{ else} \\ \min_{a \in O} [\text{cost}(a) + h(\text{Pre}(a); s)] \end{cases}$$

- For **sets** of atoms C , assume **independence**:

$$h(C; s) \stackrel{\text{def}}{=} \sum_{r \in C} h(r; s)$$

- Resulting **heuristic function** $h_{\text{add}}(s)$:

$$h_{\text{add}}(s) \stackrel{\text{def}}{=} h(\text{Goal}; s)$$

- Heuristic not admissible, but informative and fast

Max Heuristic

- For all **atoms** p :

$$h(p; s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } p \in s, \text{ else} \\ \min_{a \in O} [1 + h(\text{Pre}(a); s)] \end{cases}$$

- For **sets** of atoms C , assume **independence**:

$$h(C; s) \stackrel{\text{def}}{=} \max_{r \in C} h(r; s)$$

- Resulting **heuristic function** $h_{add}(s)$:

$$h_{max}(s) \stackrel{\text{def}}{=} h(\text{Goal}; s)$$

- Heuristic admissible, but not very informative

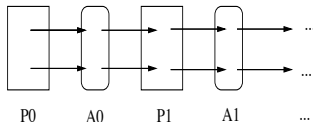
Max Heuristic and (Relaxed) Planning Graph

- Build reachability graph $P_0, A_0, P_1, A_1, \dots$

$$P_0 = \{p \in s\}$$

$$A_i = \{a \in O \mid \text{Pre}(a) \subseteq P_i\}$$

$$P_{i+1} = P_i \cup \{p \in \text{Add}(a) \mid a \in A_i\}$$



- Graph implicitly **represents** max heuristic

$$h_{\max}(s) = \min i \text{ such that } G \subseteq P_i$$

Relaxed Planning Graph

- A variation of the Planning Graph from Graphplan
 - Alternating fact and action levels
 - Ignores negative effects from actions, no mutex relations
- Fact level F_0 contains the facts that are true in the initial state,
- Action level A_0 contains actions whose preconditions are reached from F_0 ,
- F_1 contains F_0 plus the add effects of the actions in A_0 , and so on.

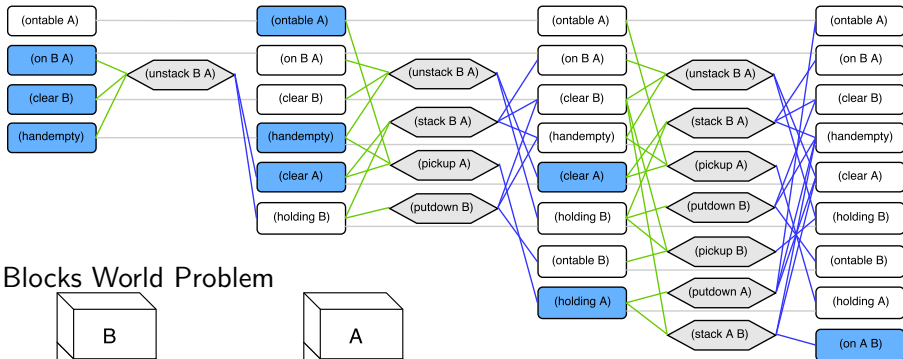
RPG Construction

```
1: function BUILDFULLRPG( $\Xi$ ,  $\mathcal{I}$ ,  $G$ )
2:    $i \leftarrow 0$ 
3:   RPG.FACTLEVEL0  $\leftarrow \mathcal{I}$ 
4:   while  $G \not\subseteq \text{RPG.FACTLEVEL}_i$  do
5:     RPG.ACTIONLEVEL $i$   $\leftarrow \{a \in \mathcal{A} \mid \text{pre}(a) \in \text{RPG.FACTLEVEL}_i\}$ 
6:     RPG.FACTLEVEL $i+1$   $\leftarrow \text{RPG.FACTLEVEL}_i \cup \text{eff}(a)^+, \forall a \in$   

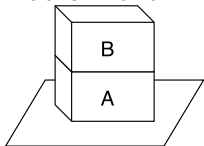
    RPG.ACTIONLEVEL $i$ 
7:     if  $\text{RPG.FACTLEVEL}_{i+1} \equiv \text{RPG.FACTLEVEL}_i$  then
8:       return  $G$  UNREACHABLE  $\triangleright$  The algorithm fails if at some point  

before reaching the facts of the goal no new fact level is added in the graph.
9:      $i \leftarrow i + 1$ 
10:  return RPG
```

Blocks World: RPG

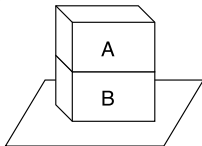


Blocks World Problem



- Initial State:

(ontable A)
(on B A)
(clear B)
(handempty)



- Goal State:

(on A B)

Goal Recognition as reasoning over Heuristics

Estimating Goal Completion with Landmarks

Background: Planning and Landmarks

Definition (**Planning**)

A planning instance is represented by a triple $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, in which:

- $\Xi = \langle \Sigma, \mathcal{A} \rangle$ is the **domain definition**, and consists of a finite set of **facts** Σ and a finite set of **actions** \mathcal{A} (action costs = 1);
- \mathcal{I} and G represent the **planning problem**, in which $\mathcal{I} \subseteq \Sigma$ is the **initial state**, and $G \subseteq \Sigma$ is the **goal state**.

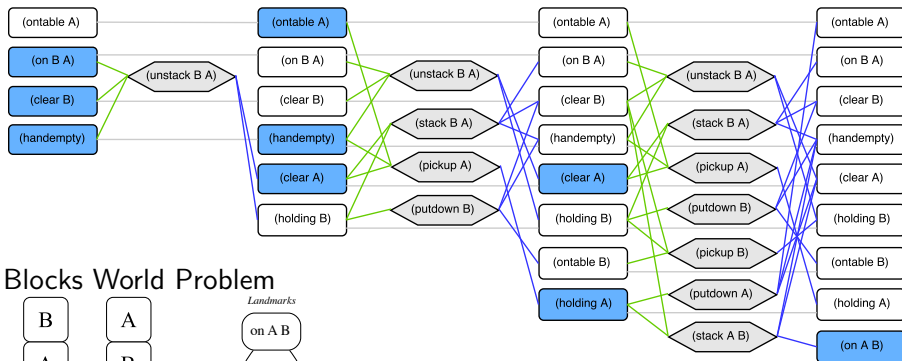
Definition (**Landmarks**)

Given a planning instance $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, a **fact** (or **action**) L is a landmark in Π iff L must be **satisfied** (or **executed**) at some point along all valid plans that achieve G from \mathcal{I} .

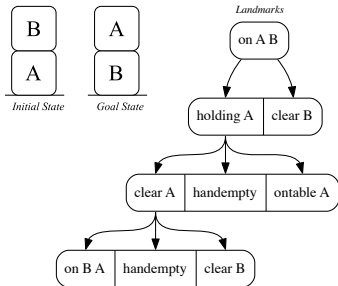
Computing Landmarks

- Deciding whether a fact is a landmark is PSPACE-complete
Involves deciding whether a plan exists with out actions that achieve it
- Multiple types of landmarks/orderings
- Multiple ways of computing landmarks:
 - Complete set of landmarks (very expensive)
 - Various methods to compute incomplete set of landmarks
(e.g. delete relaxation, backchaining on RPG)
 - Disjunctive landmarks
- In our work we use an algorithm from Hoffman *et al.* (2004) to compute an incomplete set of landmarks with a partial order relation

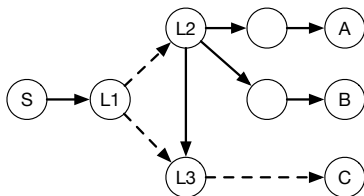
Computing Landmarks



Blocks World Problem



Computing Achieved Landmarks



- Our heuristics require identifying which fact landmarks have been achieved during the observed plan execution for every candidate goal $G \in \mathcal{G}$;
- For every candidate goal $G \in \mathcal{G}$:
 - Extract *ordered* landmarks for G ;
 - Use achieved landmarks of G in preconditions and effects of every observed action $o \in O$;
 - Under partial observability, we deal with missing actions by inferring that predecessors of observed landmarks must have been achieved;

Landmark-Based Goal Completion Heuristic

- Goal Completion h_{gc} aggregates the percentage of completion of each sub-goal into an overall percentage of completion for all facts of a candidate goal;

$$h_{gc}(G, \mathcal{AL}_G, \mathcal{L}_G) = \left(\frac{\sum_{g \in G} \frac{|\mathcal{AL}_g \in \mathcal{AL}_G|}{|\mathcal{L}_g \in \mathcal{L}_G|}}{|G|} \right) \quad (1)$$

where:

- \mathcal{AL}_G achieved landmarks for goals in G
- \mathcal{L}_G all landmarks for goals in G

Landmark-Based Goal Completion Heuristic: Algorithm

- Our approach allows the use of a threshold θ , giving us **flexibility to avoid eliminating candidate goals** whose the percentage of goal completion are close to the highest completion value;

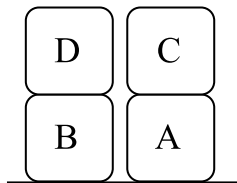
Algorithm 2 Recognize goals/plans using the heuristic h_{gc} .

Input: Ξ planning domain definition, \mathcal{I} initial state, \mathcal{G} set of candidate goals, O observations, and θ threshold.

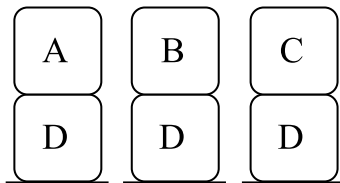
Output: Recognized goal(s).

```
1: function RECOGNIZE( $\Xi, \mathcal{I}, \mathcal{G}, O, \theta$ )
2:    $\mathcal{L}_{\mathcal{G}} \leftarrow \text{EXTRACTLANDMARKS}(\Xi, \mathcal{I}, \mathcal{G})$ 
3:    $\Lambda_{\mathcal{G}} \leftarrow \text{COMPUTEACHIEVEDLANDMARKS}(\mathcal{I}, \mathcal{G}, O, \mathcal{L}_{\mathcal{G}})$ 
4:    $maxh \leftarrow \max_{G' \in \mathcal{G}} h_{gc}(G', \Lambda_{\mathcal{G}}(G'), \mathcal{L}_{\mathcal{G}}(G'))$ 
5:   return all  $G$  s.t  $G \in \mathcal{G}$  and
        $h_{gc}(G, \Lambda_{\mathcal{G}}(G), \mathcal{L}_{\mathcal{G}}(G)) \geq (maxh - \theta)$ 
6: end function
```

Example (1 of 3)



Initial State



Set of Candidate Goals

- Observations:
 - (unstack D B); and
 - (unstack C A).
- The real goal is: (and (ontable D) (on C D) (clear C))

Example (2 of 3)

Achieved Landmarks in Observations:

- (and (ontable D) (clear A) (on A D)):

(ontable d) → [(clear d), (on d b), (handempty)], [(holding d)],
[(ontable d)]

(clear a) → [(on c a), (clear c), (handempty)], [(clear a)],

(on a d) → [(on c a), (clear c), (handempty)],
[(clear a), (ontable a), (handempty)],
[(holding a), (clear d)], [(on a d)],

- (and (ontable D) (clear B) (on B D)):

(ontable d) → [(clear d), (on d b), (handempty)], [(holding d)],
[(ontable d)]

(clear b) → [(clear d), (on d b), (handempty)], [(clear b)]

(on b d) → [(ontable b), (handempty)], [(clear d), (holding b)], [(on b d)]

- (and (ontable D) (clear C) (on C D)):

(ontable d) → [(clear d), (on d b) (handempty)], [(holding d)], [ontable d]

(clear c) → [(clear c)]

(on c d) → [(on c a), (clear c), (handempty)], [(holding c), (clear d)],
[on c d]

Example (3 of 3) - h_{gc}

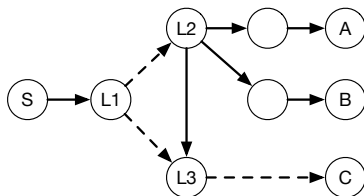
Landmark-Based Goal Completion Heuristic

- (and (ontable D) (clear A) (on A D)):
 - Goal Completion: $\frac{\frac{2}{3} + \frac{2}{2} + \frac{2}{4}}{3} = 0.7222$
- (and (ontable D) (clear B) (on B D)):
 - Goal Completion: $\frac{\frac{2}{3} + \frac{2}{2} + \frac{1}{3}}{3} = 0.6666$
- (and (ontable D) (clear C) (on C D)):
 - Goal Completion: $\frac{\frac{2}{3} + \frac{1}{1} + \frac{2}{3}}{3} = 0.7733$ (**highest estimated value**)

Landmark-Based Uniqueness Heuristic (1 of 2)

- Our second heuristic computes **landmark uniqueness**:
inverse frequency of a landmark within landmarks for candidate goals:

$$L_{Uniq}(L, \mathcal{L}_G) = \left(\frac{1}{\sum_{L \in \mathcal{L}_G} |\{L | L \in \mathcal{L}\}|} \right) \quad (2)$$



$$L_{Uniq}(L2) = 1/2$$

$$L_{Uniq}(L1) = 1/3$$

$$L_{Uniq}(L3) = 1$$

Landmark-Based Uniqueness Heuristic (2 of 2)

- Our second heuristic, called h_{uniq} , estimates the goal completion of a candidate goal G by calculating the ratio between the sum of the uniqueness value of the achieved landmarks of G and the sum of the uniqueness value of all landmarks of G ;

$$h_{uniq}(G, \mathcal{AL}_G, \mathcal{L}_G, \Upsilon_{uv}) = \left(\frac{\sum_{\mathcal{A}_L \in \mathcal{AL}_G} \Upsilon_{uv}(\mathcal{A}_L)}{\sum_{L \in \mathcal{L}_G} \Upsilon_{uv}(L)} \right) \quad (3)$$

where:

- Υ_{uv} is a table of uniqueness values
- \mathcal{AL}_G achieved landmarks for goals in G
- \mathcal{L}_G all landmarks for goals in G

Landmark-Based Uniqueness Heuristic: Algorithm

- Our second heuristic is called h_{uniq} ;

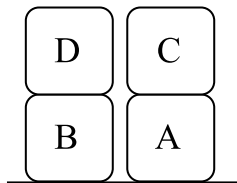
Algorithm 3 Recognize goals/plans using the heuristic h_{uniq} .

Input: Ξ planning domain definition, \mathcal{I} initial state, \mathcal{G} set of candidate goals, O observations, and θ threshold.

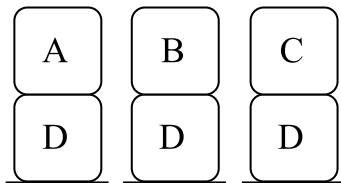
Output: Recognized goal(s).

```
1: function RECOGNIZE( $\Xi, \mathcal{I}, \mathcal{G}, O, \theta$ )
2:    $\mathcal{L}_{\mathcal{G}} \leftarrow \text{EXTRACTLANDMARKS}(\Xi, \mathcal{I}, \mathcal{G})$ 
3:    $\Lambda_{\mathcal{G}} \leftarrow \text{COMPUTEACHIEVEDLANDMARKS}(\mathcal{I}, \mathcal{G}, O, \mathcal{L}_{\mathcal{G}})$ 
4:    $\Upsilon_{uv} \leftarrow \langle \rangle \quad \triangleright \text{Map of landmarks to their uniqueness value.}$ 
5:   for each fact landmark  $L$  in  $\mathcal{L}_{\mathcal{G}}$  do
6:      $\Upsilon_{uv}(L) \leftarrow L_{Uniq}(L, \mathcal{L}_{\mathcal{G}})$ 
7:   end for
8:    $maxh \leftarrow \max_{G' \in \mathcal{G}} h_{uniq}(G', \Lambda_{\mathcal{G}}(G'), \mathcal{L}_{\mathcal{G}}(G'), \Upsilon_{uv})$ 
9:   return all  $G$  s.t  $G \in \mathcal{G}$  and
       $h_{uniq}(G, \Lambda_{\mathcal{G}}(G), \mathcal{L}_{\mathcal{G}}(G), \Upsilon_{uv}) \geq (maxh - \theta)$ 
10: end function
```

Example (1 of 2)



Initial State



Set of Candidate Goals

- Observations:
 - (unstack D B); and
 - (unstack C A).
- The real goal is: (and (ontable D) (on C D) (clear C))

Example (2 of 2) - h_{uniq}

Landmark-Based Uniqueness Heuristic

- (and (ontable D) (clear A) (on A D)), $Total_{Uniq} = 5.5$:
 - [(clear A)] = 1, [(clear A) (ontable A) (handempty)] = 1,
[(on C A) (clear C) (handempty)] = 0.5, [(holding D)] = 0.3333,
[(clear D) (on D B) (handempty)] = 0.3333
 - $h_{uniq} = 3.1666 / 5.5 = 0.5757$
- (and (ontable D) (clear B) (on B D)), $Total_{Uniq} = 5$:
 - [(clear B)] = 1, [(ontable B) (handempty)] = 1,
[(on D B) (clear D) (handempty)] = 0.3333, [(holding D)] = 0.3333
 - $h_{uniq} = 2.6666 / 5 = 0.5333$
- (and (ontable D) (clear C) (on C D)), $Total_{Uniq} = 4.5$:
 - [(clear C)] = 1, [(on C A) (clear C) (handempty)] = 0.5,
[(clear D) (holding C)] = 1, [(holding D)] = 0.3333
[(clear D) (on D B) (handempty)] = 0.3333
 - $h_{uniq} = 3.1666 / 4.5 = 0.71$

Recognized (and (ontable D) (clear C) (on C D)) **with:**
 $h_{uniq} = 0.71$

Goal Recognition as reasoning over Heuristics

Results

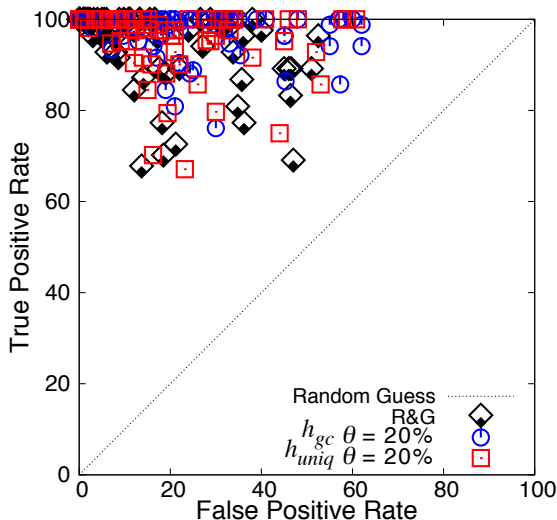
Experiments and Evaluation

- We evaluate our heuristics over datasets with 15 planning domains (6 of these domains from original Ramírez and Geffner paper):
 - BLOCKS-WORLD, CAMPUS, DEPOTS, DRIVER-LOG, DOCK-WORKER-ROBOTS, EASY-IPC-GRID, FERRY, INTRUSION-DETECTION, KITCHEN, LOGISTICS, MICONIC, ROVERS, SATELLITE, SOKOBAN, AND ZENO-TRAVEL;
- These datasets contain hundreds of goal recognition problems, varying the observability (10%, 30%, 50%, 70%, and 100%);
- We compared our heuristics against the original approach of Ramírez and Geffner (Plan Recognition as Planning. IJCAI, 2009), which is their fastest and most accurate approach;

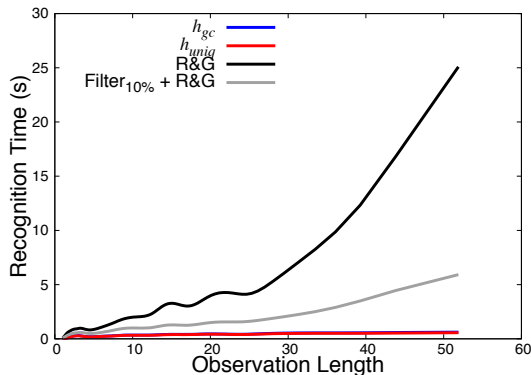
Experiments and Evaluation - ROC Space (1 of 2)

- Results of our heuristics use threshold $\theta = 20\%$;
- We compare Ramírez and Geffner's approach over ROC space, which shows the trade-off between TPR and FPR;
- We aggregate multiple domains and plot these goal recognition results in ROC space.

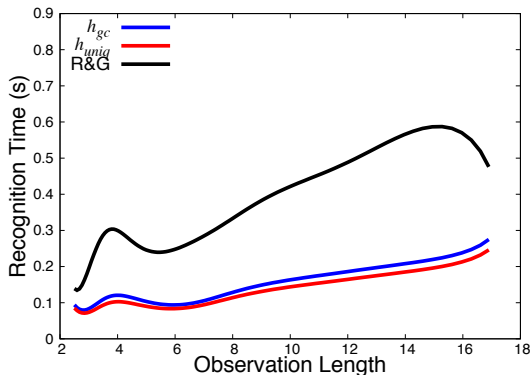
Experiments and Evaluation - ROC Space (2 of 2)



Experiments and Evaluation - Recognition Time



Experiments and Evaluation - Recognition Time with Noise



- **Contribution so far:**

- Use planning landmarks for goal recognition;
- Obviate the need to run a planner during goal recognition, resulting in much faster and highly accurate recognition; and
- Robust dataset to evaluate goal recognition algorithms

- **Limitations:**

- Sensitive to the presence of landmarks; and
- Low accuracy with very few observations, *i.e.*, 10% of observability;

Table of Contents

- 1 Introduction
 - Motivation and Intuition
 - Formalism
- 2 Goal Recognition as reasoning over Heuristics
 - Motivation and Background
 - Estimating Goal Completion with Landmarks
 - Results
- 3 Online Goal Recognition as Reasoning over Landmarks**
- 4 Goal Recognition in Incomplete Domains
 - Motivation and Background
 - Landmarks in Incomplete Domains
 - Recognizing Goals in Incomplete Domains
 - Results
- 5 Real World Applications
 - Optimality Monitoring and Plan Abandonment
 - Plan Recognition using Video Data
- 6 Summary and Future Directions

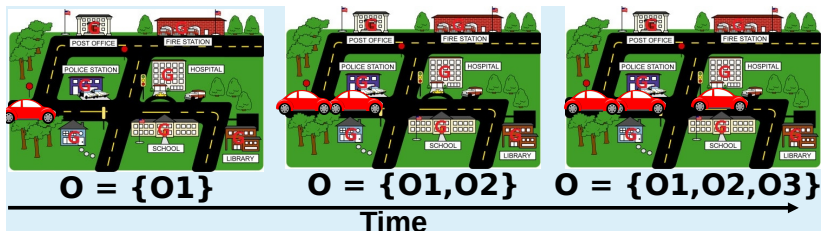
Motivation for Efficient Online Goal Recognition

Most goal recognition approaches using domain models have three key limitations:

- ① assumption of a discrete state-space in a PDDL-like formalism
 - not viable for use with path planning scenarios
- ② assume all access to all observations at once
 - approaches do not consider the time to recognition
- ③ need to call a planner multiple times per goal to rank hypotheses
 - PRAP is computationally expensive, impractical for long plans

Online vs. Offline Plan Recognition

- Offline plan recognition:
 - All observations received at once;
 - Observations may be incomplete or noisy;
 - One-shot recognition;
- Online plan recognition:
 - Observations received incrementally;
 - Observations may be incomplete or noisy;
 - Objective is to recognize goal as soon as possible, without the full observation sequence



Our approach:

- is efficient for online goal recognition;
- works in both discrete and continuous domains;
- uses goal-mirroring to minimize planner calls;
- reasons about landmarks to minimize the number of goal hypotheses;
- returns reliable goal ranking as soon as possible

Previous Work: Goal Mirroring

- Focuses on efficient online recognition
- Uses a planner to generate plan hypotheses using $(|O| + 1)|G|$ planner calls
 - Computes optimal plans for each goal
 - For every incoming observation compute:
 - prefix – concatenation of observations
 - suffix – plan from last observation to every goal
 - Rank plan hypotheses by comparing to ideal plan

Previous Work: Goal Recognition using Heuristics

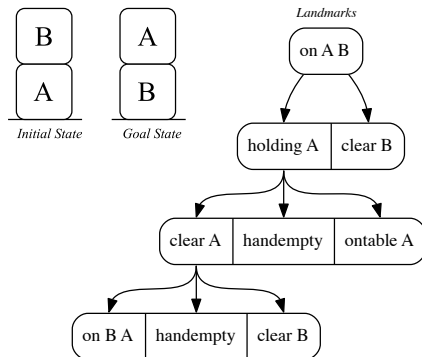
- Focuses on efficient recognition with **no planner calls**
- Uses heuristic estimation of states resulting from observation
- Ranks goals as a weighed sum based on the number of **landmarks** visited by the observations
- Key characteristics:
 - Very fast (linear on the number of goals and observations)
 - Less accurate in domains with few landmarks and/or observability

The underlying representation adapts from the Transition Normal Form (TNF) (from Pommerening and Helmert), specifically

- Domain model originally dealt with operators that modify the truth-value of *facts* or *fluents*
- We adopt a domain model $M = \langle \mathcal{V}, \mathcal{O} \rangle$:
 - \mathcal{V} is a set of variables with a, possibly infinite, domain
 - operators in \mathcal{O} change the value of all variables in \mathcal{V} from s into s'
 - $cost(o)$ is usually 1, but when dealing with continuous variables it is the euclidean distance of such values between s and s'

Landmarks in Discrete Domains

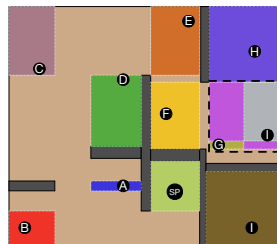
- Facts that must be true in all valid plans
- Root node is the goal condition
- Leaves are facts of initial state
- Connected boxes – facts that must be true together



Landmarks in Continuous Domains

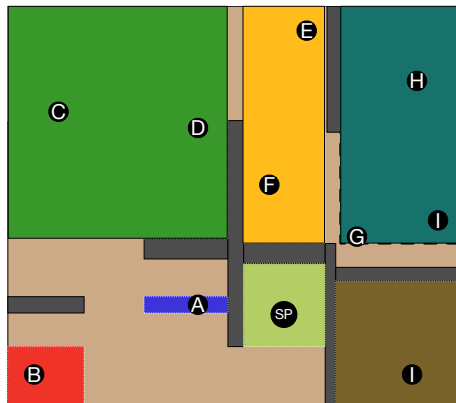
We need a notion of landmark in continuous domains

- Redefine landmarks as areas surrounding goals
 - Goals – Black dots
 - Surrounding Rectangles – continuous landmark areas
- To reach a goal the observed motion must intersect (go through) the corresponding landmark area.
- In this work, landmark areas roughly correspond to rooms partitioned as rectangular Voronoi diagrams
 - Other notions of numeric landmarks may apply (e.g. Scala et al. IJCAI 2017)



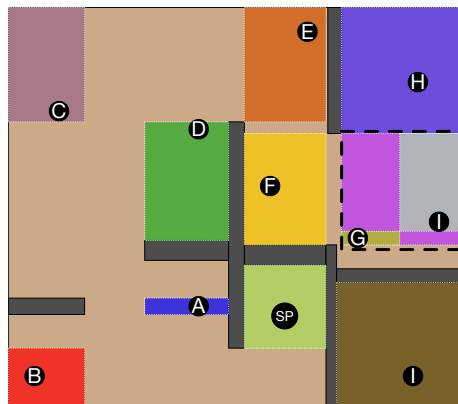
Extracting Landmarks in Continuous Domains

- 1 Partition landmarks using wall aligned squares.



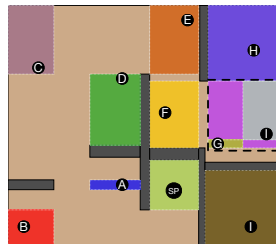
Extracting Landmarks in Continuous Domains

- 1 Partition landmarks using wall aligned squares.
- 2 Recursively divide squares with multiple goals using the mid-point in x, y between two randomly selected goals.



Online Recognition with Landmarks

- Generate the ordered set of achieved landmarks
- Maintain the group of goals eliminated due to landmarks
- For every observation:
 - Check if it “achieved” a landmark
 - If observations backtrack, re-instate goals
- Rank goals using the landmark completion heuristic h_{gc}



Goal Mirroring with Landmarks

Combines landmark reasoning with goal mirroring

- Compute landmarks and optimal plans i_k for all goals
- For every observation:
Compute plan prefix m^- , and for every goal
 - Either prune goals that have **passed** the last landmark; or
 - Compute plan suffix m_k^+ (from last observation) using planner
 - Compute **cost ratio** between prefix+suffix and optimal plan:

$$P(O \mid g) = \frac{\text{cost}(i_k)}{\text{cost}(m^-) + \text{cost}(m_k^+)}$$

- Rank unpruned goals based on a **normalized cost ratio**

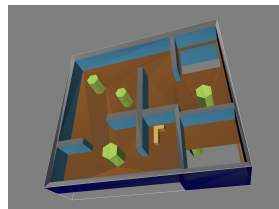
Computes $P(g_k \mid O)$ using a normalizing factor

$$\eta = 1 / \sum_{g_k \in G} \text{rank}(g_k)$$

- Approximates $P(g \mid O) = \eta \sum_{g_k \in G} P(O \mid g_k) P(g_k)$ for all goals, assuming $P(g_k) = 0$ for pruned goals

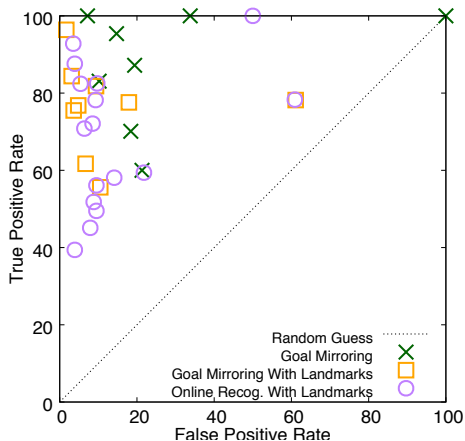
Continuous Evaluation

- Cubicles environment and robot (OMPL)
- 11 points spread evenly over the environment
- 220 problems

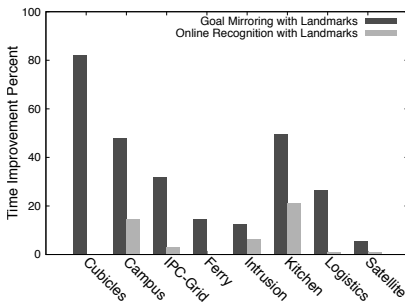
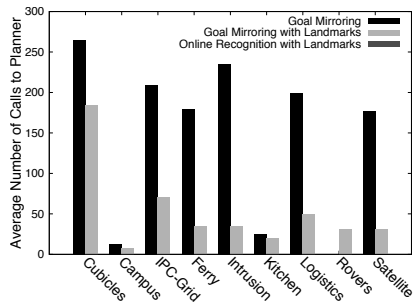


- Dataset expanded from Ramirez and Geffner's original work
- Domains extracted from the IPC competition
BLOCKS-WORLD, CAMPUS, DEPOTS, DRIVER-LOG, DOCK-WORKER-ROBOTS,
EASY-IPC-GRID, FERRY, INTRUSION-DETECTION, KITCHEN, LOGISTICS,
MICONIC, ROVERS, SATELLITE, SOKOBAN, AND ZENO-TRAVEL;
- Hundreds of goal recognition problems

Performance Results



Efficiency Results



- **Contribution so far:**

- Extended the idea of landmarks for continuous domains; and
- Developed online algorithms able to recognize plans in discrete and continuous domains;
- **Very** efficient in both discrete and continuous domains.

- **Limitations:**

- Naive notion of spatial landmarks;
- Much better performance on discrete domains.

Table of Contents

- 1 Introduction
 - Motivation and Intuition
 - Formalism
- 2 Goal Recognition as reasoning over Heuristics
 - Motivation and Background
 - Estimating Goal Completion with Landmarks
 - Results
- 3 Online Goal Recognition as Reasoning over Landmarks
- 4 Goal Recognition in Incomplete Domains
 - Motivation and Background
 - Landmarks in Incomplete Domains
 - Recognizing Goals in Incomplete Domains
 - Results
- 5 Real World Applications
 - Optimality Monitoring and Plan Abandonment
 - Plan Recognition using Video Data
- 6 Summary and Future Directions

Goal Recognition in Incomplete Domains

Motivation and Background

Motivation

- So far, all goal recognition techniques assume that:
 - domain knowledge is available;
 - a domain engineer can build a **complete** and **correct** model of such domain knowledge
 - observations from sensor data agree with the model
- However, most real world domains have two sources uncertainty:
 - ambiguity in how actions performed by agents are realized; and
 - ambiguity from how imperfect sensor data reports features of the world.
- We aim to overcome both limitations by accepting **incomplete** domains

- In this work, we develop a goal recognition approach that copes with **incomplete** planning **domain models**
- Our main contributions are:
 - a new goal recognition formalization in incomplete domains
 - the notion of *potential landmarks* to deal with such incompleteness
 - *on-the-fly* landmark extraction techniques to deal with *overlooked landmarks*
 - an **efficient** goal recognition technique for incomplete domains
- We evaluate the techniques empirically in an extensive new dataset that:
 - includes hundreds of non-trivial problems
 - includes various levels of domain model incompleteness

Incomplete Planning Domain

Definition (Incomplete Planning Domain)

An incomplete domain models is a tuple $\tilde{\mathcal{D}} = \langle \mathcal{R}, \tilde{\mathcal{O}} \rangle$, where:

- \mathcal{R} is the logic language
- $\tilde{\mathcal{O}}$ is a set of incomplete operators

$\tilde{op} = \langle pre(\tilde{op}), \tilde{pre}(\tilde{op}), eff^+(\tilde{op}), eff^-(\tilde{op}), \tilde{eff}^+(\tilde{op}), \tilde{eff}^-(\tilde{op}) \rangle$,
where:

- $pre(\tilde{op})$ and $eff(\tilde{op})$ have the same semantics as in the STRIPS domain models;
- possible preconditions $\tilde{pre}(\tilde{op}) \subseteq \mathcal{R}$; and
- possible add and delete effects $\tilde{eff}^+(\tilde{op}) \subseteq \mathcal{R}$ and $\tilde{eff}^-(\tilde{op}) \subseteq \mathcal{R}$.

- $\tilde{\mathcal{D}}$ has a *completion set* $\langle \langle \tilde{\mathcal{D}} \rangle \rangle$ comprising all derivable domain models.
- $2^{\sum_{\tilde{op} \in \tilde{\mathcal{O}}} (|\tilde{pre}(\tilde{op})| + |\tilde{eff}^+(\tilde{op})| + |\tilde{eff}^-(\tilde{op})|)}$ such models
- single ground-truth model \mathcal{D}^* that drives observed states.

Incomplete domain example

- $\mathcal{F} = [p, q, r, g];$
- $\tilde{\mathcal{A}} = [\tilde{a}, \tilde{b}, \tilde{c}]$, where:
 - $pre(\tilde{a}) = [p, q], \widetilde{pre}(\tilde{a}) = [r], \widetilde{eff}^+(\tilde{a}) = [r], \widetilde{eff}^-(\tilde{a}) = [p]$
 - $pre(\tilde{b}) = [p], \widetilde{eff}^+(\tilde{b}) = [r], \widetilde{eff}^-(\tilde{b}) = [p], \widetilde{eff}^-(\tilde{b}) = [q]$
 - $pre(\tilde{c}) = [r], \widetilde{pre}(\tilde{c}) = [q], \widetilde{eff}^+(\tilde{c}) = [g]$
- $\mathcal{I} = [p, q];$ and
- $G = [g].$

Solution:

- Here, $[\tilde{a}, \tilde{b}, \tilde{c}]$ is a valid plan that achieves the goal stated $[g]$ from the initial state $[p, q]$.
- corresponds to *optimistic* state sequence:
 $s_0 = [p, q], s_1 = [p, q, r], s_2 = [q, r], s_3 = [q, r, g].$
- This example has 2^5 completions
(2 possible preconditions and 3 possible effects)

Goal Recognition in Incomplete Domain

Definition (**Goal Recognition Problem**)

A goal recognition problem with an incomplete domain model is a quadruple $\tilde{T} = \langle \tilde{\mathcal{D}}, Z, \mathcal{I}, \mathcal{G}, O \rangle$, where:

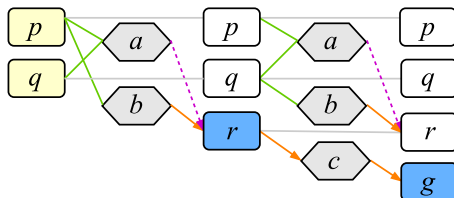
- $\tilde{\mathcal{D}} = \langle \mathcal{R}, \tilde{\mathcal{O}} \rangle$ is an incomplete domain model;
- Z is a set of typed objects in the environment, \mathcal{F} is a set instantiated facts from Z , and $\tilde{\mathcal{A}}$ is a set of instantiated incomplete actions from $\tilde{\mathcal{O}}$ with objects from Z ;
- $\mathcal{I} \in \mathcal{F}$ an initial state;
- \mathcal{G} is a set of possible goals, including a hidden goal G ; and
- $O = \langle o_1, o_2, \dots, o_n \rangle$ is an observation sequence.

Each observation $o_i \in \tilde{\mathcal{A}}$. O corresponds to the sequence of actions (i.e., a plan) that achieves the correct hidden goal G in a complete domain in $\langle \langle \tilde{\mathcal{D}} \rangle \rangle$.

Goal Recognition in Incomplete Domains

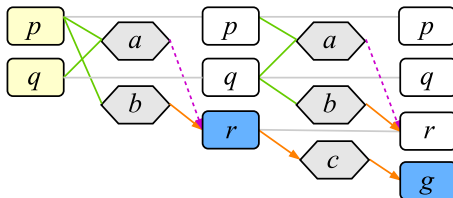
Landmarks in Incomplete Domains

Optimistic RPG and Landmarks



- Previous approaches to landmark-based plan recognition extract landmarks using a *Relaxed Planning Graph* (RPG)
- In order to cope with uncertainty, we build an *optimistic* RPG
 - ignores all *possible* preconditions and delete effects (definite and possible)
 - includes all *possible add* effects to the graph
- Extracted landmarks are now either *definite* or *possible*

Definite and Possible Landmarks



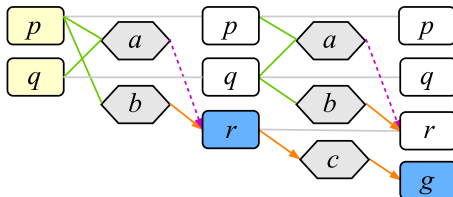
Definition (**Definite Landmark**)

A *definite* landmark $L_{Definite}$ is a fact (landmark) extracted from a known add effect ($eff^+(a)$) of an achiever a (action) in the ORPG.

Definition (**Possible Landmark**)

A *possible* landmark $L_{Possible}$ is a fact (landmark) extracted from a possible add effect ($\widetilde{eff}^+(a)$) of an achiever a (action) in the ORPG.

Definite and Possible Landmarks



- Landmarks: $[p, q, r, g]$
 - Definite Landmarks: $[r, g]$
 - Possible Landmarks: $[p, q]$

Goal Recognition in Incomplete Domains

Recognizing Goals in Incomplete Domains

Computing Landmarks on the Fly

- Most *efficient* landmark extraction algorithms (e.g. based on Relaxed Planning Graphs) are *incomplete*
Since extracting all landmarks is as difficult as planning itself
- Given the observations, we can focus landmark extraction to identify *overlooked* landmarks
 - Rebuild the ORPG without an observation
 - Check for satisfiability

Heuristic for Incomplete Domains

- Combining these three types of landmarks, we define the $h_{\widetilde{GR}}(G)$ heuristic
 - definite* landmarks: $\mathcal{AL}_G, \mathcal{L}_G$;
 - possible* landmarks: $\widetilde{\mathcal{AL}}_G, \widetilde{\mathcal{L}}_G$; and
 - overlooked* landmarks $\mathcal{ANL}_G, \mathcal{NL}_G$
- $h_{\widetilde{GR}}(G)$ computes the ratio of achieved landmarks over computed ones

$$h_{\widetilde{GR}}(G) = \left(\frac{\mathcal{AL}_G + \widetilde{\mathcal{AL}}_G + \mathcal{ANL}_G}{\mathcal{L}_G + \widetilde{\mathcal{L}}_G + \mathcal{NL}_G} \right)$$

Goal Recognition in Incomplete Domains

```

1: function RECOGNIZE( $\tilde{\mathcal{D}}, Z, \mathcal{I}, \mathcal{G}, O$ )
2:    $\mathcal{H}_{\mathcal{G}} \leftarrow \langle \rangle$  ▷ Map goals to heuristic estimated values.
3:   for each goal  $G$  in  $\mathcal{G}$  do
4:      $\mathcal{NL}_G \leftarrow \langle \rangle$  ▷ Set of overlooked landmarks for  $G$ .
5:      $\mathcal{L}_G, \tilde{\mathcal{L}}_G \leftarrow \text{EXTRACTLANDMARKS}(\tilde{\mathcal{D}}, Z, \mathcal{I}, G)$ 
6:      $\mathcal{AL}_G, \widetilde{\mathcal{AL}}_G, \mathcal{ANL}_G \leftarrow \langle \rangle$  ▷ Achieved landmarks.
7:     for each observed action  $o$  in  $O$  do
8:        $OF_o \leftarrow \text{all facts in } \text{pre}(o) \cup \text{eff}^+(o) \cup \widetilde{\text{eff}}^+(o)$  ▷ Observed facts in the
       incomplete model of action  $o$ .
9:        $\mathcal{AL}_G \leftarrow \text{all landmarks } l \text{ in } \mathcal{L}_G \text{ s.t. } l \in OF_o$ 
10:       $\widetilde{\mathcal{AL}}_G \leftarrow \text{all landmarks } \tilde{l} \text{ in } \tilde{\mathcal{L}}_G \text{ s.t. } \tilde{l} \in OF_o$ 
11:      for each fact  $f$  in  $OF_o$  s.t.  $f \notin (\mathcal{L}_G \cup \tilde{\mathcal{L}}_G)$  do
12:        if ISLANDMARK( $f, \mathcal{I}, G$ ) then
13:           $\mathcal{NL}_G \leftarrow \mathcal{NL}_G \cup f$ 
14:           $\mathcal{ANL}_G \leftarrow \mathcal{ANL}_G \cup f$ 
15:       $\mathcal{H}_G := \mathcal{H}_G \cup \langle G, \left( \frac{\mathcal{AL}_G + \widetilde{\mathcal{AL}}_G + \mathcal{ANL}_G}{\mathcal{L}_G + \tilde{\mathcal{L}}_G + \mathcal{NL}_G} \right) \rangle$  ▷ Estimate value for  $G$  using the Use
        $h_{\widetilde{GR}}$  heuristic.
16:   return all  $G$  s.t.  $\langle G, v \rangle \in \mathcal{H}_{\mathcal{G}}$  and
        $v \geq (\max_{v_i} \langle G', v_i \rangle \in \mathcal{H}_{\mathcal{G}})$ 

```

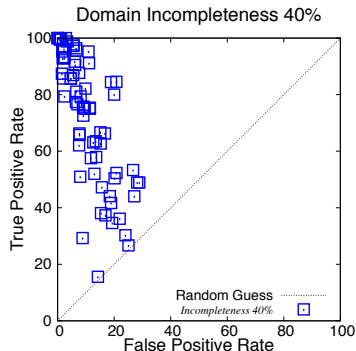
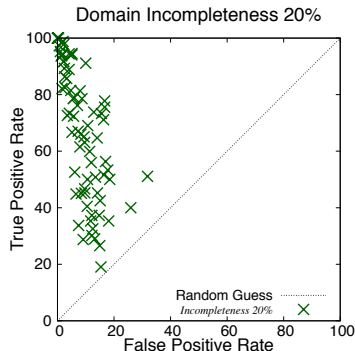
Goal Recognition in Incomplete Domains

Results

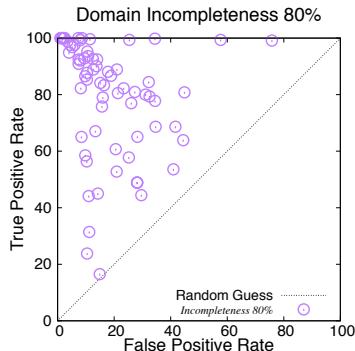
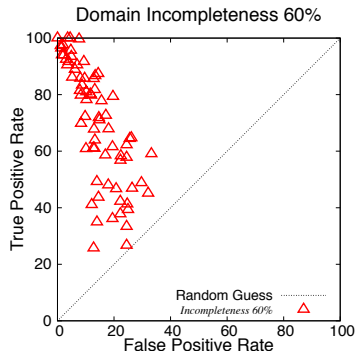
Experiments

- Our evaluation uses a modified version of our IPC dataset
- We automatically generated incomplete versions of these domains:
 - ① randomly move a percentage of known preconditions and effects into possible lists of preconditions and effects;
 - ② randomly add possible preconditions from delete effects that are not preconditions of a corresponding operator; and
 - ③ randomly add into possible lists (of preconditions, add effects, or delete effects) predicates whose parameters fit into the operator signatures.
- Results show accuracy and runtime

Accuracy for Low Incompleteness



Accuracy for High Incompleteness



Recognition Runtime

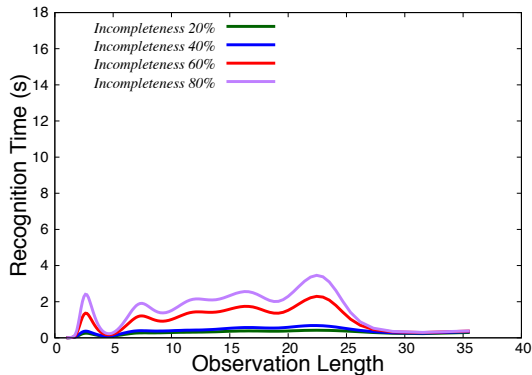


Table of Contents

- 1 Introduction
 - Motivation and Intuition
 - Formalism
- 2 Goal Recognition as reasoning over Heuristics
 - Motivation and Background
 - Estimating Goal Completion with Landmarks
 - Results
- 3 Online Goal Recognition as Reasoning over Landmarks
- 4 Goal Recognition in Incomplete Domains
 - Motivation and Background
 - Landmarks in Incomplete Domains
 - Recognizing Goals in Incomplete Domains
 - Results
- 5 Real World Applications
 - Optimality Monitoring and Plan Abandonment
 - Plan Recognition using Video Data
- 6 Summary and Future Directions

Real World Applications

Optimality Monitoring and Plan Abandonment

Optimality Monitoring

- Agents often **deviate** from the optimal plan for multiple reasons
 - they have concurrent/multiple goals; or
 - they are not perfect optimizers;
- We would like to be able to detect which actions in a plan do not advance (are non-optimal) towards a monitored goal;
- Our contribution is twofold:
 - We formalize this problem using **planning domain definition**; and
 - We combine two planning techniques to solve this problem: **landmarks** and **domain-independent heuristics**.
- We evaluate our approach using several planning domains, and show that our approach yields **high accuracy** at **low computational cost**.

Plan Optimality Monitoring Problem

Definition (Plan Optimality Monitoring Problem)

- Domain definition (Facts and Actions) $\Xi = \langle \Sigma, \mathcal{A} \rangle$;
 - Initial state \mathcal{I} ;
 - A monitored goal G ; and
 - An observation sequence $O = \langle o_1, o_2, \dots, o_n \rangle$, representing a full observable plan execution;
-
- The solution to a plan optimality monitoring problem is the set of observations (**non-optimal actions**) that do not advance an optimal plan that the agent may be following.

Roboschool Example

Optimal Plan

```
(walk roomb_3 roomb_4)
(walk roomb_4 rooma_4)
(pick p3 rooma_4)
(walk rooma_4 roomb_4)
(walk roomb_4 roomb_3)
(walk roomb_3 roomb_2)
(walk roomb_2 roomc_2)
(throw p3 roomc_2 roomd_2 b1)
(pick p4 roomc_2)
(throw p4 roomc_2 roomd_2 b1)
```

Suboptimal Plan

```
(walk roomb_3 roomb_4)
(walk roomb_4 rooma_4)
(pick p3 rooma_4)
(walk rooma_4 roomb_4)
(walk roomb_4 roomb_3)
(walk roomb_3 roomb_2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomc_1)
(walk roomc_1 roomc_2)
(throw p3 roomc_2 roomd_2 b1)
(pick p4 roomc_2)
(throw p4 roomc_2 roomd_2 b1)
```

Roboschool Example

Optimal Plan

```
(walk roomb_3 roomb_4)
(walk roomb_4 rooma_4)
(pick p3 rooma_4)
(walk rooma_4 roomb_4)
(walk roomb_4 roomb_3)
(walk roomb_3 roomb_2)
(walk roomb_2 roomc_2)
(throw p3 roomc_2 roomd_2 b1)
(pick p4 roomc_2)
(throw p4 roomc_2 roomd_2 b1)
```

Suboptimal Plan

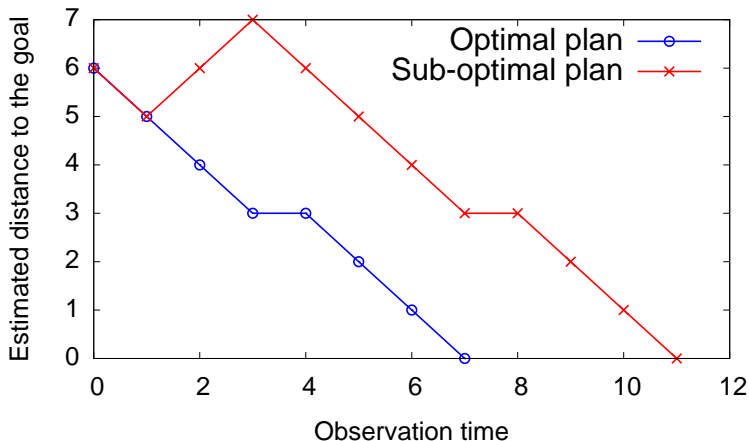
```
(walk roomb_3 roomb_4)
(walk roomb_4 rooma_4)
(pick p3 rooma_4)
(walk rooma_4 roomb_4)
(walk roomb_4 roomb_3)
(walk roomb_3 roomb_2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomc_1)
(walk roomc_1 roomc_2)
(throw p3 roomc_2 roomd_2 b1)
(pick p4 roomc_2)
(throw p4 roomc_2 roomd_2 b1)
```

Plan Optimality Monitoring Approach

- Our approach **combines planning techniques**, *i.e.*, landmarks and domain-independent heuristics.
- We use **landmarks** to obtain information about **what cannot be avoided** to achieve a monitored goal G ; and
- We use **heuristics** to analyze possible **plan execution deviation**.

Analyzing Plan Execution Deviation

- If an observation o_i results a state s_i , we consider a **deviation from a plan** to occur if $h(s_{i-1}) < h(s_i)$.



Predicting Non-regressive Actions via Landmarks

- To predict which actions could be executed in the next observation, we **analyze the closest landmarks by estimating the distance** (using h_{max}) from the current state to the extracted landmarks \mathcal{L} , namely:
 - For every fact landmark $l \in \mathcal{L}$ in which the estimated **distance is 0**, we select those actions $a \in \mathcal{A}$ such that $l \in pre(a)$; and
 - For every fact landmark $l \in \mathcal{L}$ in which the estimated **distance is 1**, we select those actions $a \in \mathcal{A}$ such that $pre(a) \in \text{current state}$ and $l \in eff(a)^+$;
- These predicted actions **may reduce the distance** to the **monitored goal** and **next landmarks**.

Finding Non-regressive actions

Require: $\Xi = \langle \Sigma, \mathcal{A} \rangle$ *planning domain*, δ *current state*, and \mathcal{L} *ordered fact landmarks*

```
1: function NONREGRESSIVEACTIONS( $\Xi, \delta, \mathcal{L}$ )
2:    $\eta_{PActions} \leftarrow \langle \rangle$ 
3:   for each fact landmark  $l$  in  $\mathcal{L}$  do
4:      $Al \leftarrow \langle \rangle$ 
5:     if  $h_{max}(l) = 0$  then  $\triangleright h_{max}(l)$  estimates  $l$  from  $\delta$ .
6:        $Al \leftarrow$  all  $a$  in  $\mathcal{A}$  s.t.  $l \in pre(a)$ 
7:     else if  $h_{max}(l) = 1$  then
8:        $Al \leftarrow$  all  $a \in \mathcal{A}$  s.t.  $pre(a) \in \delta \wedge l \in eff(a)^+$ 
9:      $\eta_{PActions} := \eta_{PActions} \cup Al$ 
10:  return  $\eta_{PActions}$   $\triangleright$  set of possible upcoming actions
```

Detecting Sub-Optimal Steps

- To detect sub-optimal steps (actions) in observation sequence O for a monitored goal G , we combine the techniques we developed and filter with the following condition:
 - **An observed action $o \in O$ is considered sub-optimal if:**
 $o \notin \text{set of predicted actions AND } (h(s_{i-1}) < h(s_i)).$

Detecting Sub-Optimal Steps (Monitor Plan Optimality)

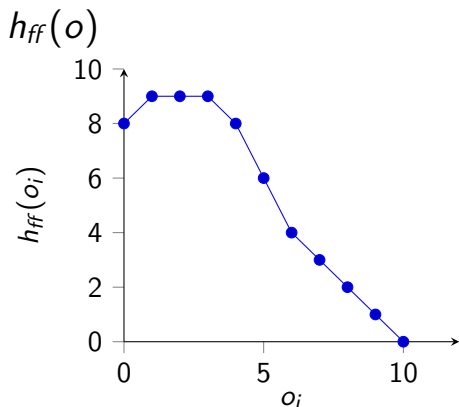
Require: $\Xi = \langle \Sigma, \mathcal{A}, \rangle$ *planning domain*, \mathcal{I} *initial state*, G *monitored goal*, and O *observed actions*.

```
1: function MONITORPLANOPTIMALITY( $\Xi, \mathcal{I}, G, O$ )
2:    $A_{SubOptimal} \leftarrow \langle \rangle$   $\triangleright$  Non-contributing actions towards goal  $G$ .
3:    $L \leftarrow \text{EXTRACTLANDMARKS}(\mathcal{I}, G)$ 
4:    $\delta \leftarrow \mathcal{I}$   $\triangleright$   $\delta$  is the current state.
5:    $\eta_{PActions} \leftarrow \text{NONREGRESSIVEACTIONS}(\Xi, \delta, L)$ 
6:    $D_G \leftarrow \text{ESTIMATEGOALDISTANCE}(\delta, G)$   $\triangleright$  Any domain-independent heuristic
7:   for each observed action  $o$  in  $O$  do
8:      $\delta \leftarrow \delta.\text{APPLY}(o)$ 
9:      $D'_G \leftarrow \text{ESTIMATEGOALDISTANCE}(\delta, G)$ 
10:    if  $o \notin \eta_{PActions} \wedge (D'_G > D_G)$  then
11:       $A_{SubOptimal} \leftarrow A_{SubOptimal} \cup o$ 
12:     $\eta_{PActions} \leftarrow \text{NONREGRESSIVEACTIONS}(\Xi, \delta, L)$ 
13:     $D_G \leftarrow D'_G$ 
14:  return  $A_{SubOptimal}$ 
```

Example 1: Optimal Plan

Observations

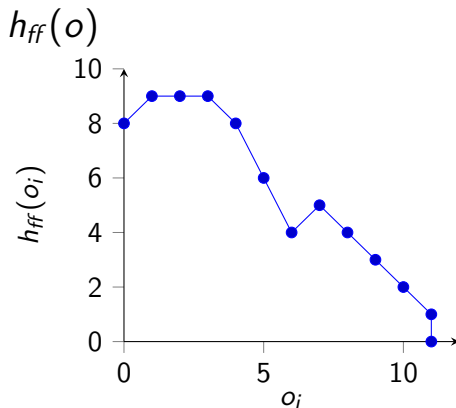
```
(walk roomb_3 roomb_4)
(walk roomb_4 rooma_4)
(pick p3 rooma_4)
(walk rooma_4 roomb_4)
(walk roomb_4 roomb_3)
(walk roomb_3 roomb_2)
(walk roomb_2 roomc_2)
(throw p3 roomc_2 roomd_2 b1)
(pick p4 roomc_2)
(throw p4 roomc_2 roomd_2 b1)
```



Example 2: Suboptimal Plan

Observations

(walk roomb_3 roomb_4)
(walk roomb_4 rooma_4)
(pick p3 rooma_4)
(walk rooma_4 roomb_4)
(walk roomb_4 roomb_3)
(walk roomb_3 roomb_2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomc_1)
(walk roomc_1 roomc_2)
(throw p3 roomc_2 roomd_2 b1)
(pick p4 roomc_2)
(throw p4 roomc_2 roomd_2 b1)



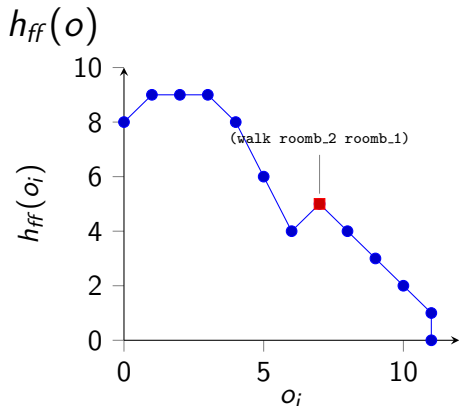
Example 2: Suboptimal Plan

Observations

(walk roomb_3 roomb_4)
(walk roomb_4 rooma_4)
(pick p3 rooma_4)
(walk rooma_4 roomb_4)
(walk roomb_4 roomb_3)
(walk roomb_3 roomb_2)
(walk roomb_2 roomb_1)
(walk roomb_1 roomc_1)
(walk roomc_1 roomc_2)
(throw p3 roomc_2 roomd_2 b1)
(pick p4 roomc_2)
(throw p4 roomc_2 roomd_2 b1)

Suboptimal actions:

(walk roomb_2 roomb_1)



Step 7: (walk roomb_2 roomb_1)

Expected Landmarks: None

$$h_{ff}(o_7) = 5 \quad h_{ff}(o_6) = 4$$

Experiments and Evaluation (1 of 2)

- We evaluate our approach over 10 planning domains;
 - Precision: percentage of correctly detected sub-optimal steps;
 - Recall: percentage of true sub-optimal steps, actually detected.
- We use 6 domain-independent heuristics:
 - h_{adjsum} , $h_{adjsum2}$, $h_{adjsum2M}$, h_{combo} , h_{ff} , and h_{sum} ;
- To extract landmarks and their ordering, we use an algorithm developed by Hoffman *et al.* (Ordered Landmarks in Planning. JAIR, 2004);
- We manually generate the dataset from medium and large planning problems, containing both optimal and sub-optimal plan execution.

Experiments and Evaluation (2 of 2)

Domain	$ O $	$ \mathcal{L} $	Heuristic	Time	Precision / Recall / F1
BLOCKS-WORLD	15.2	20.1	$h_{adjsum2} / h_{ff}$	0.19 / 0.21	100% / 74.2% / 85.2%
DRIVER-LOG	20.1	53.6	$h_{adjsum2M}$	1.33	100% / 100% / 100%
DEPOTS	16.7	64.7	$h_{adjsum2} / h_{ff}$	1.22 / 1.43	81.2% / 100% / 89.6%
EASY-IPC-GRID	14.1	48.5	$h_{adjsum2} / h_{ff}$	0.77 / 0.86	100% / 100% / 100%
FERRY	13.8	18.1	h_{adjsum} / h_{sum}	0.23 / 0.19	88.8% / 78.5% / 83.1%
LOGISTICS	20.8	24.0	$h_{adjsum2} / h_{ff}$	0.35 / 0.55	100% / 91.3% / 95.4%
MICONIC	18.1	19.4	h_{adjsum} / h_{sum}	0.29 / 0.21	100% / 86.9% / 93.1%
SATELLITE	25.7	60.8	$h_{adjsum2M}$	9.58	88.8% / 53.3% / 66.6%
SOKOBAN	24.0	76.5	h_{combo}	4.28	90.9% / 83.3% / 86.9%
ZENO-TRAVEL	12.2	38.7	$h_{adjsum2} / h_{ff}$	0.86 / 0.99	100% / 92.8% / 96.2%

Table: Plan Optimality Monitoring experimental (best) results.

- **Contribution:**

- Formalized plan optimality monitoring problem as planning;
- Developed an approach based on landmarks and heuristics;
- We show that our approach has high accuracy in almost all domains (besides SATELLITE).

- **Limitations:**

- We do not yet deal with partial observability;

Real World Applications

Plan Recognition using Video Data

Plan Recognition using Video Data

- **Plan recognition**

- Task of recognizing the plan (i.e., the sequence of actions) the observed agent is following in order to achieve his intention (Sadri, 2012)

- **Activity recognition**

- The task of recognizing the independent set of actions that generates an interpretation to the movement that is being performed (Poppe, 2010)
- Such task is particularly challenging in the real physical world
- Much research effort focuses on activity and plan recognition as separate challenges;
- We develop a hybrid approach that comprises both activity and plan recognition;
- The approach infers, from a set of candidate plans, which plan a human subject is pursuing based exclusively on fixed-camera video.

Poppe, R. A survey on vision-based human action recognition. *Image and Vision Computing* 28(6), pp. 976–990, 2010.

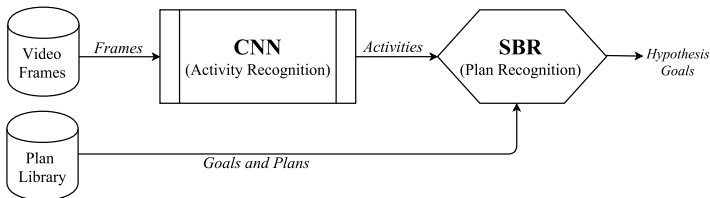
Sadri, Fariba. *Intention Recognition in Agents for Ambient Intelligence: Logic-Based Approaches*.

Ambient Intelligence and Smart Environments, pp. 197-236, 2012.

A Hybrid Architecture for Activity and Plan Recognition

- **Conceptually divided in two main parts**

- CNN-based activity recognition (CNN)
- CNN-backed symbolic plan recognition (SBR)

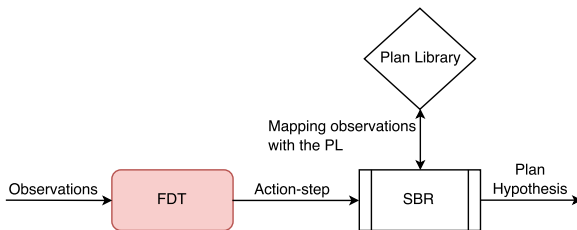


- **Convolutional Neural Network**

- Architecture: GoogLeNet
- 22-layer deep network based on the Inception module
- Input images: 224x224 (3 channels: RGB)
- Output classes: 9 (activities)

• Symbolic Behavior Recognition (SBR)

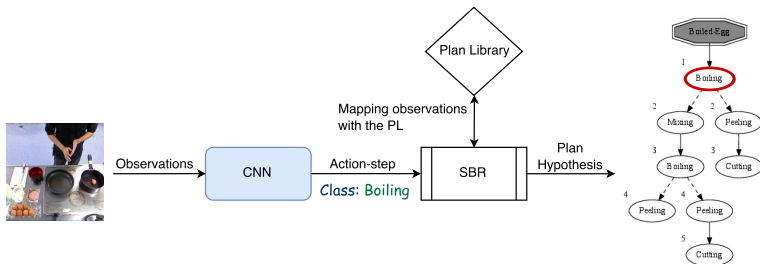
- A plan recognition approach that takes as input a plan library and a sequence of observations
- Feature decision tree (FDT) maps observable features to plan-steps in a plan library
- SBR returns set of hypotheses plans such that each hypothesis represents a plan that achieves a top-level goal in a plan library



CNN-backed Symbolic Plan Recognition

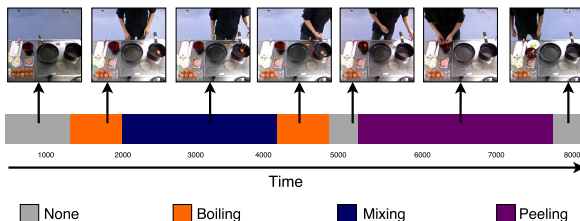
• Our Symbolic Behavior Recognition

- We modify the SBR and replace the FDT with the CNN-backed Activity Recognition
- The CNN-backed Activity Recognition maps frames directly into nodes (activities) in the plan library used by SBR to compute sequential consistency of plan steps



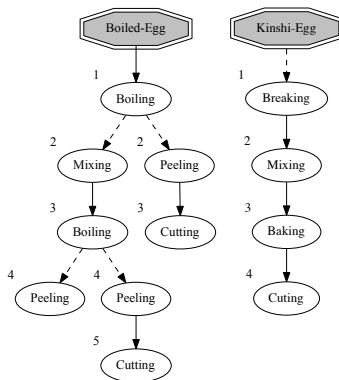
Experiments: Dataset

- ICPR 2012 Kitchen Scene Context based Gesture Recognition dataset (KSCGR)
- **5 recipes for cooking eggs in Japan**
 - Ham and Eggs, Omelet, Scrambled-Egg, Boiled-Egg and Kinshi-Tamago
 - Each recipe is performed by 7 subjects (5 in training set, 2 in testing set)
- **9 cooking activities composes the dataset**
 - Breaking, mixing, baking, turning, cutting, boiling, seasoning, peeling, and none



Experiments: Plan Library Modeling

- We model a plan library containing knowledge of the agent's possible goals and plans based on the KSCGR dataset
- We consider that a sequence of cooking gestures is analogous to a sequence of a plan in the plan library



Activity Recognition results

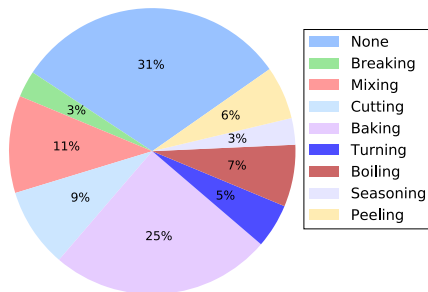
Precision, Recall, F-measure and Accuracy scores for each activity

Activity	Precision	Recall	F-measure	Accuracy
<i>None</i>	0.65	0.97	0.78	0.64
<i>Breaking</i>	0.44	0.41	0.42	0.27
<i>Mixing</i>	0.67	0.34	0.45	0.29
<i>Baking</i>	0.74	0.88	0.80	0.67
<i>Turning</i>	0.77	0.38	0.51	0.34
<i>Cutting</i>	0.87	0.63	0.73	0.58
<i>Boiling</i>	0.61	0.34	0.43	0.28
<i>Seasoning</i>	0.89	0.37	0.52	0.35
<i>Peeling</i>	0.72	0.10	0.18	0.09

Activity Recognition results

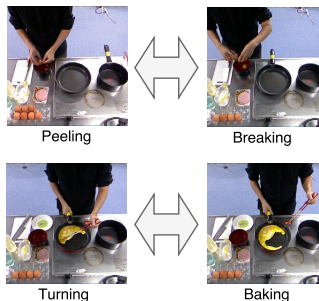
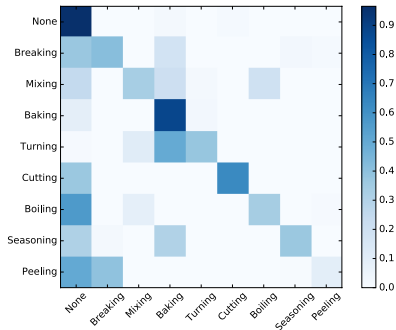
Accuracy scores for each activity and the distribution of frames in KSCGR dataset

Activity	Frames	Accuracy
<i>None</i>	31%	0.64
<i>Breaking</i>	3%	0.27
<i>Mixing</i>	11%	0.29
<i>Baking</i>	25%	0.67
<i>Turning</i>	5%	0.34
<i>Cutting</i>	9%	0.58
<i>Boiling</i>	7%	0.28
<i>Seasoning</i>	3%	0.35
<i>Peeling</i>	6%	0.09



Activity Recognition results

Confusion matrix



- Close position in the scene
- Similar movements

Plan Recognition results

- We evaluate the whole pipeline using the number of hypotheses inferred by the plan recognizer
- **Score** weights correct predictions by the number of hypotheses

$$Score = \frac{c}{\#recipesFromSBR}$$

- c : 1 if the correct recipe was inferred, 0 otherwise
- $\#recipesFromSBR$: Number of recipes yielded by the recognizer

Plan Recognition results

#	True Recipe	Predicted Recipes	Score
10	Boiled-Egg	Scramble-Egg, Omelette, Ham-Egg	0.00
	Ham-Egg	Scramble-Egg, Omelette	0.00
	Kinshi-Egg	Kinshi-Egg	1.00
	Omelette	Scramble-Egg, Omelette	0.50
	Scramble-Egg	Ham-Egg	0.00
11	Boiled-Egg	Kinshi-Egg, Omelette, Ham-Egg	0.00
	Ham-Egg	Scramble-Egg	0.00
	Kinshi-Egg	Scramble-Egg, Omelette, Ham-Egg	0.00
	Omelette	Kinshi-Egg, Scramble-Egg, Omelette, Ham-Egg	0.25
	Scramble-Egg	Kinshi-Egg	0.00
Average:			0.18

Contributions and Future Work

- We developed a hybrid architecture for activity and plan recognition
- Pipeline includes:
 - A CNN for activity recognition that feeds directly into:
 - a modified (SBR) approach that uses the CNN to index activities in the plan library
- Approach limited by the plan library in the plan recognizer
- Next steps:
 - Employ other deep learning architectures such as Long-Short Term Memory networks (LSTM) and 3D CNNs
 - Use a more flexible approach for plan recognition, such as PRAP
 - Explore object recognition to provide additional clues of the activity that is being performed

Demo video: <https://youtu.be/BoiLjU1vg3E>

Table of Contents

- 1 Introduction
 - Motivation and Intuition
 - Formalism
- 2 Goal Recognition as reasoning over Heuristics
 - Motivation and Background
 - Estimating Goal Completion with Landmarks
 - Results
- 3 Online Goal Recognition as Reasoning over Landmarks
- 4 Goal Recognition in Incomplete Domains
 - Motivation and Background
 - Landmarks in Incomplete Domains
 - Recognizing Goals in Incomplete Domains
 - Results
- 5 Real World Applications
 - Optimality Monitoring and Plan Abandonment
 - Plan Recognition using Video Data
- 6 Summary and Future Directions

FRAGA PEREIRA, Ramon; MENEGUZZI, Felipe. **Landmark-based Plan Recognition**. ECAI, 2016.

PEREIRA, Ramon F.; OREN, Nir; and MENEGUZZI, Felipe. **Landmark-Based Heuristics for Goal Recognition**. AAAI, 2017.

PEREIRA, Ramon F.; OREN, Nir; and MENEGUZZI, Felipe. **Monitoring Plan Optimality using Landmarks and Domain-Independent Heuristics**. PAIR Workshop@AAAI, 2017.

GRANADA, Roger L.; PEREIRA, Ramon F.; MONTEIRO, Juarez; BARROS, Rodrigo; RUIZ, Duncan; and MENEGUZZI, Felipe. **Hybrid Activity and Plan Recognition for Video Streams**. PAIR Workshop@AAAI, 2017.

PEREIRA, Ramon F.; OREN, Nir; and MENEGUZZI, Felipe. **Detecting Commitment Abandonment by Monitoring Plan Execution**. AAMAS, 2017.

MONTEIRO, Juarez; GRANADA, Roger; BARROS, Rodrigo and MENEGUZZI, Felipe. **Deep Neural Networks for Kitchen Activity Recognition**. IJCNN, 2017.

VERED, Mor; PEREIRA, Ramon F.; MAGNAGUAGNO, Maurício C.; KAMINKA, Gal; and MENEGUZZI, Felipe. **Online Goal Recognition Combining Landmarks and Planning**. GRW@IJCAI, 2017.

- We progressively relaxed many assumptions about plan recognition:
 - Domain knowledge
 - Quality of observations
 - Exclusively discrete domains
 - Precise domain knowledge
- We illustrated applications of these techniques:
 - Real world video-data
 - Multiagent systems

- Plan Recognition with Domain Theories
 - Use different landmark extraction algorithms;
 - Extend landmark-based heuristics to temporal and non-uniform-cost domains
 - Experiment with more advanced notions of numeric landmarks (e.g. Scala et al.)
- Applications of Plan Recognition
 - Use object recognition techniques (deep learning) to generate fact observations in video
 - Couple the above with plan recognition in domain theories
 - Do plan recognition in latent space

Thanks and Acknowledgement

People involved in this research

- Ramon Fraga Pereira (PhD Student)
- Mor Vered (PhD Student, Bar Ilan University)
- Maurício Magnaguagno (PhD Student)
- Juarez Monteiro (MSc Student)
- Roger Granada (Postdoc)
- Gal Kaminka (Bar Ilan University)
- Nir Oren (University of Aberdeen)
- Rodrigo Barros (PUCRS colleague)
- Duncan Ruiz (PUCRS colleague)

Institutions

- The Scottish Informatics and Computer Science Alliance (SICSA)
Distinguished Visiting Fellowship (DVE)
- Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) – PQ Fellowship

Thank you!