Occupation Measures: how OR can help Planning under Uncertainty

Felipe Trevizan

THE AUSTRALIAN NATIONAL UNIVERSITY

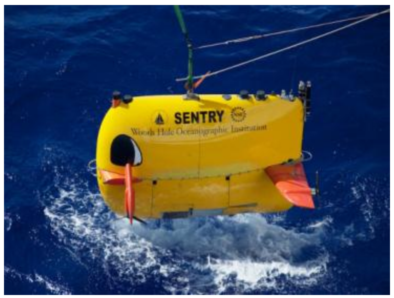
PlanSOpt – 25/Jun/2018

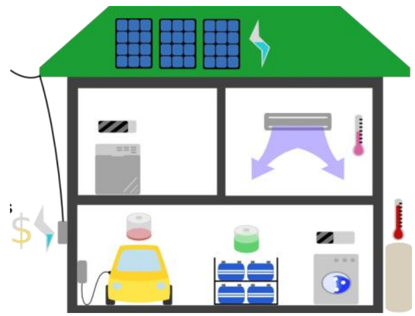
Funded by AFOSR

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Motivation

Planning under Uncertainty is **ubiquitous:**



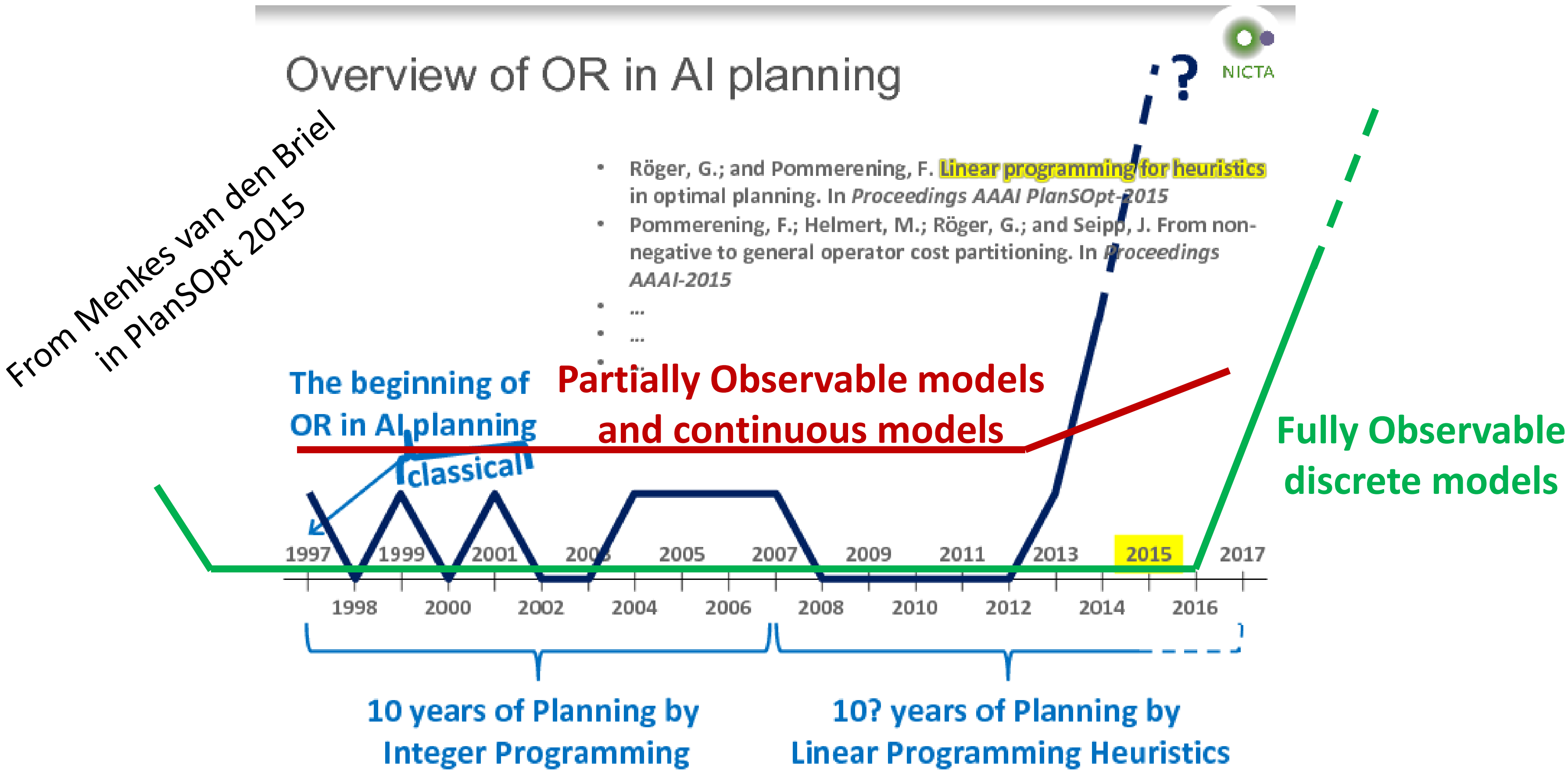




Planning agents need to handle:

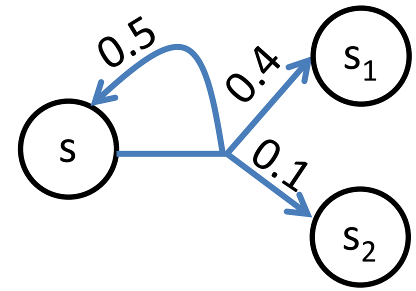
- unsafe or even hostile environments
- uncertain, costly, limited resources
- dynamic events, uncertain action outcomes
- maximizes utility with acceptable risk

OR in AI planning

Unfortunately the collaboration between OR and AI for planning under uncertainty has been very limited:



Challenges from Planning under Uncertainty



- Actions have **stochastic outcomes**:
- The solution is a **policy** not a plan:
  - Accounts for the uncertainty in the environment
  - Minimizes the expected cost to the goal
  - Maps states to actions or to prob. dist. of actions
- **Chance constraints:**
  - failure probability: $\mathbf{Pr}(\boldsymbol{failure}) \leq \boldsymbol{\theta}$
  - expected resource constraints: $\mathbf{E}(\sum_{t=0}^{\infty} \boldsymbol{fuel(t)}) \leq \boldsymbol{\theta}$
  - logic constraints: **Pr(F(transmit data)) ≥ 0.5**
    - » Translation: *with probability at least 0.5, transmit the data before finishing the mission*

Outline

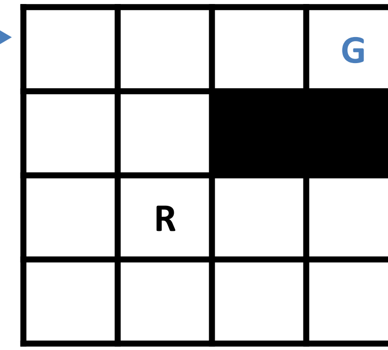
- **Background**
  - Stochastic Shortest Path Problems (SSPs)
  - Constrained SSPs
- Heuristic Search in the Occupation Measure Space
- Heuristics based on Occupation Measures
- Beyond the Resource Constraints

Stochastic Shortest Path Problems

An **SSP** the tuple [S,s_0,s_G,A,P,C]:

- set of states S
- initial state s_0
- goal state s_G
- set of actions A
- transition probability P(s'|s,a)
- cost function C(s,a):

Robot's location



{North, South, East, West}
x
{slow, normal, fast}

Action	Pr. North	Pr. Stay	C(s,a) Time Cost
move-north-slow	0.99	0.01	4
move-north-normal	0.95	0.05	2
move-north-fast	0.90	0.10	1

Optimal Solution for SSPs

- A solution to an SSP is a **policy** π
  - π(s) = action to be applied at state **s**

- The **optimal solution** is a policy π^* that minimizes the **expected cost** of reaching $s_{\mathbf{G}}$ from s_0

Primal LP for SSPs

- Variables
  - v_s: **expected cost** to reach the goal s_G from s

$$\max_{v} \sum_{s \in S} v_s$$

$$\text{s.t.} \quad v_s \geq 0$$

$$v_{s_G} = 0 \qquad \forall s \in S$$

Expected cost of reaching s_G after applying action a in s

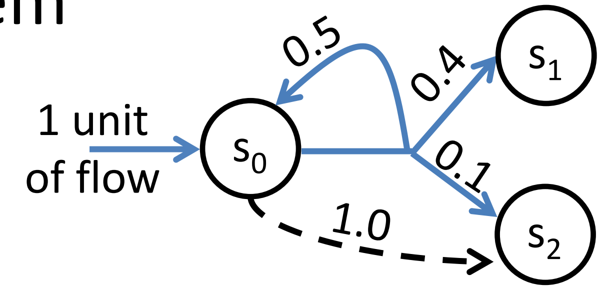
$$v_s \leq \boxed{C(s,a) + \sum_{s' \in S} P(s'|s,a) v_{s'}} \quad \forall s \in S \setminus \{s_G\}, a \in A(s)$$

- This LP is equivalent to Value Iteration ($V(s) = v_s$)
- An optimal policy:

$$\arg\min_{a \in A(s)} C(s,a) + \sum_{s' \in S} P(s'|s,a) v^*_{s'}$$

Dual LP for SSPs

- Defined in the space of **occupation measures**
  - $x_{s,a}$: expected number of times action **a** is applied in state **s**
- Intuition: **"probabilistic" flow** problem



$$\min_{x} \sum_{\substack{s \in S \\ a \in A(s)}} x_{s,a} C(s,a)$$

Expected cost of the solution

$$\text{s.t.} \quad x_{s,a} \geq 0$$

$$\underbrace{\sum_{a \in A(s)} x_{s,a}}_{\text{outflow}} - \underbrace{\sum_{\substack{s' \in S \\ a \in A(s')}} x_{s',a} P(s|s',a)}_{\text{inflow}} = \begin{cases} 1 & s = s_0 \\ 0 & \forall s \in S \setminus \{s_G, s_0\} \end{cases}$$

Flow conservation

$$\sum_{\substack{s' \in S \\ a \in A(s')}} x_{s',a} P(s_G|s',a) = 1$$

Sink

- Expected value of a function F: S x A → $\mathbb{R}$ is $\sum_{\substack{s \in S \\ a \in A(s)}} x_{s,a} F(s,a)$

Adding Cost Constraints

- A **Constrained SSP** (C-SSP) is a SSP with **multiple cost functions**:
  - C_0: cost function to be **minimized**
  - $C_1 \ldots C_k$: cost functions with **expectation upper bounded** by $u_1 \ldots u_k$

Action	$C_0(s,a)$ Time Cost	$C_1(s,a)$ Fuel Cost
move-north-slow	4	2
move-north-normal	2	4
move-north-fast	1	10

u_1 Fuel Cost
9

Optimal Solution to C-SSP

- A **solution** to an C-SSP is a **stochastic policy** π
  - π(s,a) = probability of applying action **a** in state **s**
- The **optimal solution** is a stochastic policy π^* that
  - minimizes the **expected cost** C_0 to reach s_G from s_0
  - subject to the **expected cost** $C_i \leq u_i$ for all i
- Same dual LP as before with extra constraint:

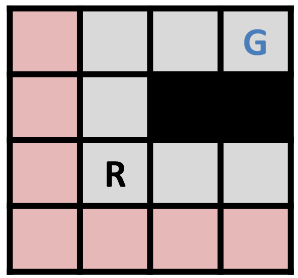
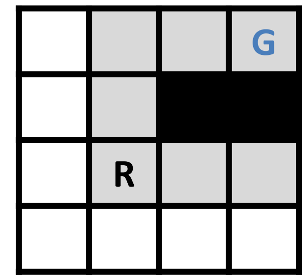
$$\sum_{\substack{s \in S \\ a \in A(s)}} x_{s,a} C_i(s,a) \leq u_i$$

- Optimal policy: $\pi^*(s,a) = \dfrac{x^*_{s,a}}{\sum_{a \in A(s)} x^*_{s,a}}$

Outline

- Background
- **Heuristic Search in the Occupation Measure Space**
  - i-dual
  - i-dual and A*
  - i-dual and Column Generation
- Heuristics based on Occupation Measures
- Beyond the Resource Constraints

Solving SSPs and C-SSPs

	Blind Search	Heuristic Search
	• Explore all states OR  	• Explore **promising** states AI 
Standard SSPs	• Value iteration • Policy iteration • **Linear Programming** 60's	• **Dynamic Programming** (RTDP, LRTDP, etc) • LAO* and extensions 90's
Constrained SSPs	• **Linear Programming** in the dual space 70's	• **Heuristic search in the dual space:** • **i-dual (2016)** • **i^2-dual (2017)**

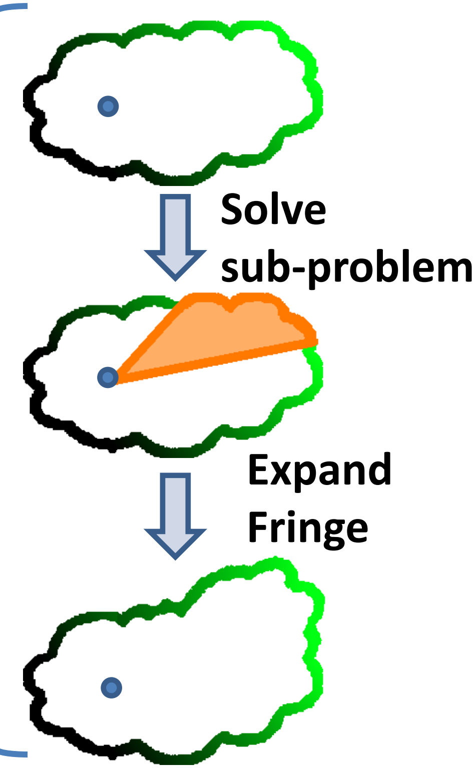
Challenge of the Dual Space

- C-SSPs are trivially encoded in the dual space ($x_{s,a}$), but:
  - **No domain-independent heuristic** (lower bound) **is known** for $x_{s,a}$ to guide the heuristic search
  - Moreover, deriving such heuristic is a hard problem because

    $x_{s,a} > 0$ **if and only if** $\pi^*(s,a) > 0$
- **i-dual** addresses this challenge by using heuristics of the primal space, i.e., **cost heuristics:**
  - $H_i(s)$: **lower bound** on expected cost C_i to reach s_G from s
  - each H_i is obtained using standard AI planning techniques
- H_0 **guides** the search towards promising regions
- H_i (for $i > 0$) does **early pruning** of infeasible solutions

i-dual: LP solved

- At each iteration of i-dual:
  - Ŝ: subset of S explored so far
  - F: fringe of the search



$$\min_{x} \sum_{\substack{s \in \hat{S} \\ a \in A(s)}} x_{s,a} C(s,a) + \boxed{\sum_{s \in F} \text{inflow}_s H_0(s)}$$

lower bound on expected cost from the fringe

$$\text{s.t.} \quad x_{s,a} \geq 0$$

$$\sum_{a \in A(s)} x_{s,a} - \sum_{\substack{s' \in \hat{S} \setminus F \\ a \in A(s')}} x_{s',a} P(s|s',a) = \begin{cases} 1 & s = s_0 \\ 0 & \forall s \in \hat{S} \setminus (F \cup \{s_G, s_0\}) \end{cases}$$

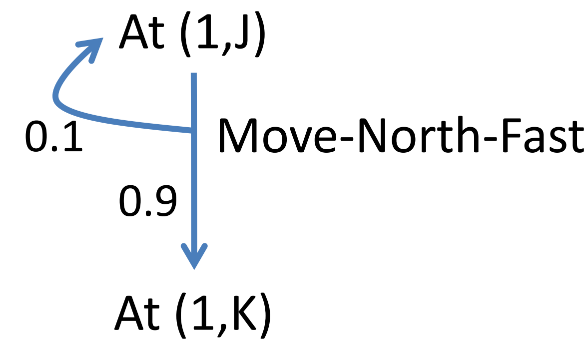
$$\sum_{s \in F \cup s_G} \sum_{\substack{s' \in \hat{S} \setminus F \\ a \in A(s')}} x_{s',a} P(s|s',a) = 1$$

Sink: goal U Fringes

$$\sum_{\substack{s \in \hat{S} \\ a \in A(s)}} x_{s,a} C_i(s,a) + \sum_{s \in F} \text{inflow}_s H_i(s) \leq u_i$$

i-dual and A*

In our running example:



- Can we use A*?
  - **No** because of stochastic actions
- Can we use AO*?
  - **No** because of loopy actions
- Can we use LAO*?
  - **No** because of **constraints** and **stochastic policies**
- Can we use i-dual? **Yes!**

f(n) = g(n) + h(n)

$E[\text{fuel}|\pi,s_0] \leq 9$

i-dual can be seen as a **generalization of A*** where the best f(n) is computed using an LP

$$\min_{x} \overbrace{\sum_{\substack{s\in\hat{S}\\ a\in A(s)}} x_{s,a} C(s,a)}^{g(n)} + \overbrace{\sum_{s\in F} \text{inflow}_s H_0(s)}^{h(n)}$$

i-dual guarantees

Similarly to A*, we showed that i-dual is:

Lower bound

Always returns a feasible solution

Heuristics for **secondary** costs (H_i) \ Heuristic for **main** cost (H_0)	Admissible	**Non**-admissible
Admissible	**Optimal**	**Sound** and **complete** but **sub-optimal**
Non-admissible	**Sound** and **incomplete**	

Might fail to find a feasible solution even if one exists

i-dual and Column Generation

- i-dual is an instance of column generation:
  - a column for i-dual is an occupation measure $x_{s,a}$
  - at each iteration of i-dual, **a set of columns is added** to the current LP.
  - the columns are chosen based on the heuristic H_0
- This expansion procedure is inherited from A*

Column Generation & Reduced Cost

- **Idea**: to solve an LP representing only a subset of its variables (columns)
  – Initially, we have one variable
  – Solve the current LP

  Set of available columns

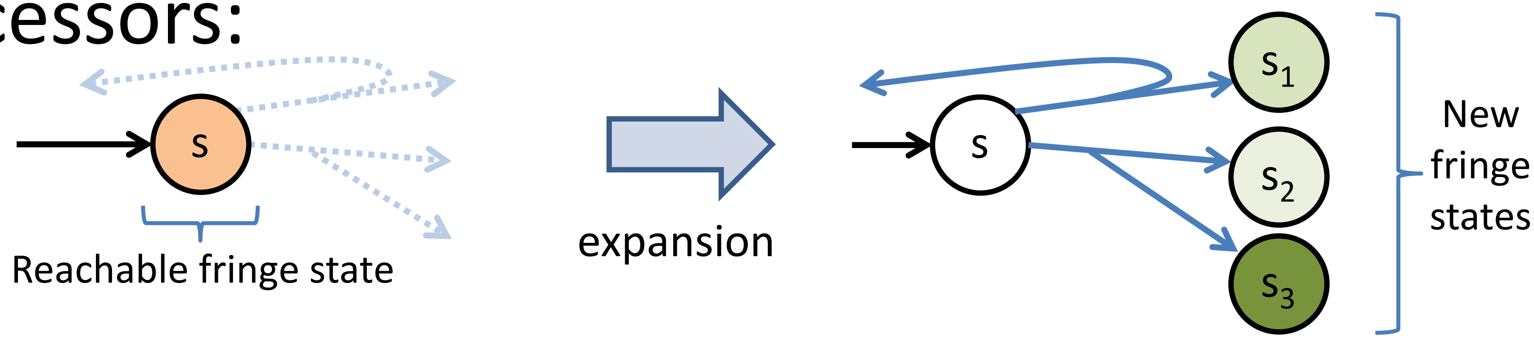
  – Add a promising column $z \in Z$ to the LP and repeat
  – Stop when there is no more promising columns
- A column $z \in Z$ is promising if:

We are minimizing cost

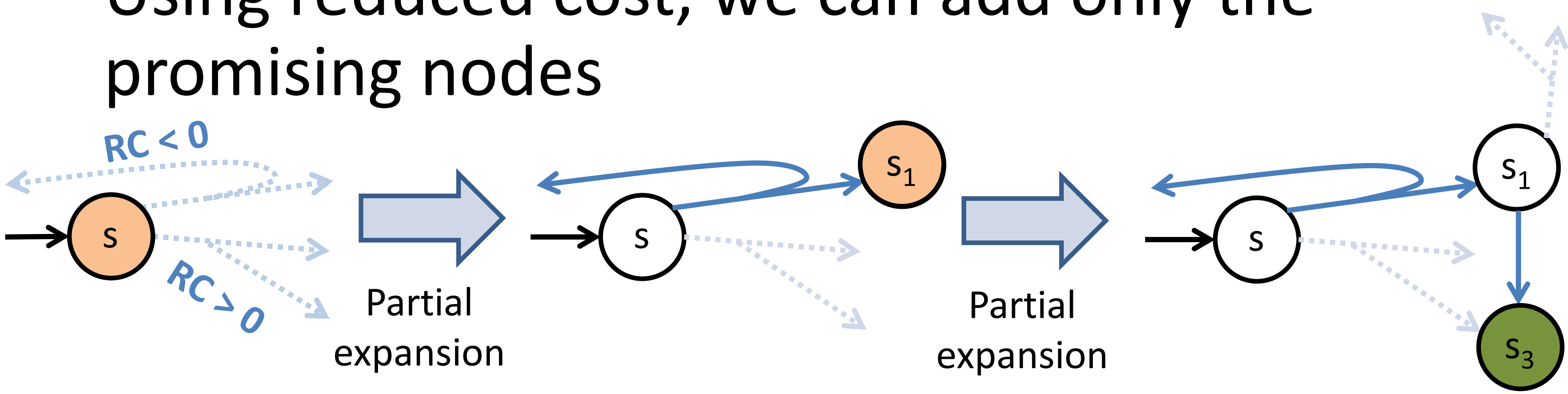
$$\text{Reduced-Cost}(z) = \underbrace{w(z)}_{\text{Coefficient of z in the obj. func.}} - \underbrace{\mu^t}_{\text{Opt. dual solution for the current LP}} z \mathbf{< 0}$$

I-dual with Partial Node Expansion

- As in A*, i-dual expands a node by adding all its successors:



- Using reduced cost, we can add only the promising nodes



- Advantage of this approach: potentially much smaller LPs on each iteration

Reduced Cost for i-dual

- A **column** (s,a) for i-dual represents an **occupation measure** $x_{s,a}$ and its reduced cost is:

$$RC(s,a) = C_0(s,a) - \mu^t z_{s,a} \boxed{+ \Sigma_{s' \text{ is unseen}} P(s'|s,a)H_0(s') - x_{s,a}H_0(s)}$$

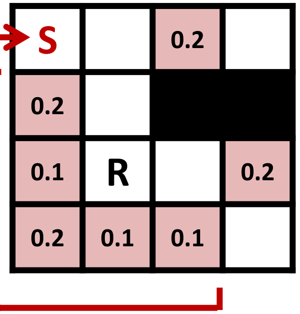
- $z_{s,a}$ encompass
  - **cost constraints**: function of $C_i(s,a)$
  - **flow preservation constraints**: function of $P(s'|s,a)$
- States s' s.t. $P(s'|s,a) > 0$ might not be in the current LP!
  - Thus, there is no value of μ associated to the flow preservation constraint for s'
  - In this case, we approximate the reduced cost

CG-dual

- At each iteration:
  - solve the current LP
  - if there is **no negative reduced** cost column available:
    - **done if** all the injected flow reaches the goal
    - otherwise, **partially expand** the fringe according to H_0
  - otherwise, add **k columns** with negative reduced cost
- We call this new algorithm **CG-dual** and it has the same guarantees as **i-dual**

Experiments: Search and Rescue

- Extension of the navigation problem:
  - one known survivor
  - presence of survivors at several locations is unknown
  - **goal**: rescue one survivor
  - **main cost**: time to rescue
  - **cost constraint**: fuel



CPU-time in seconds

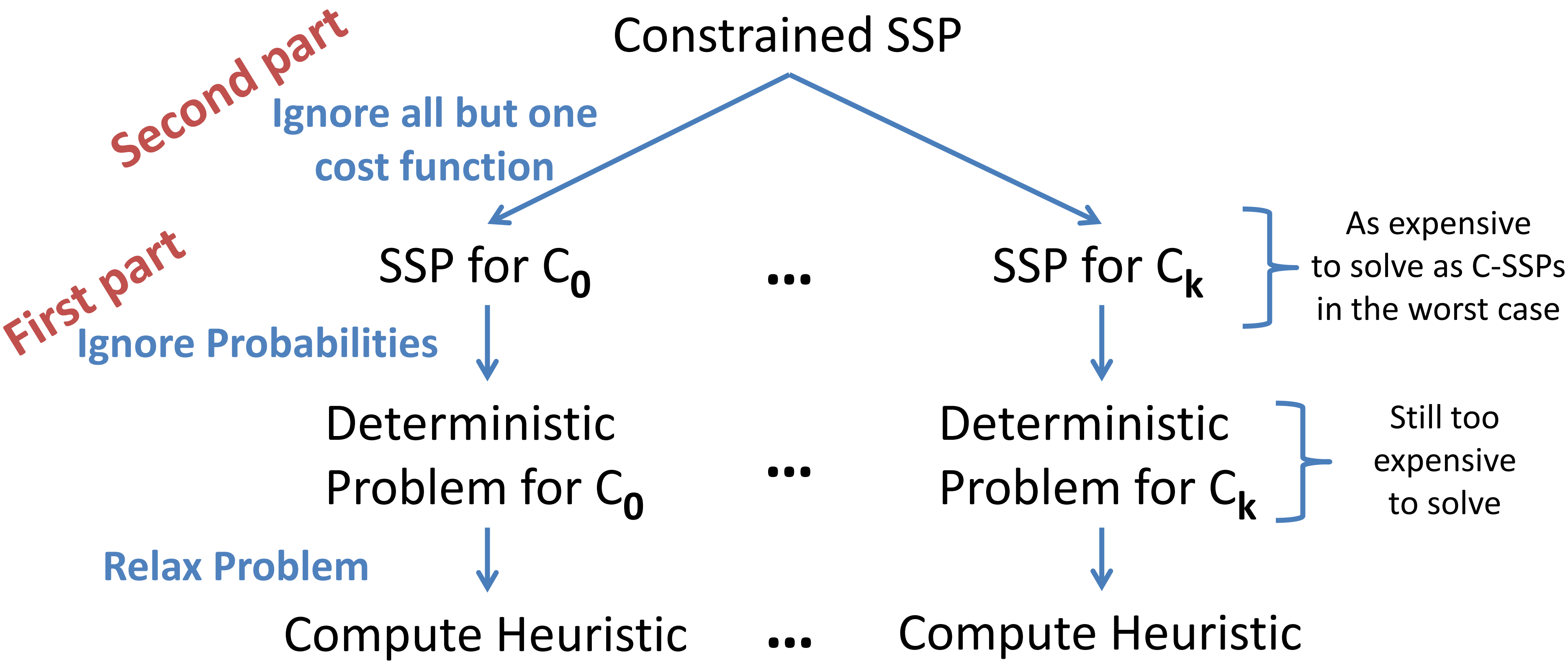
Survival density = 0.5	Dist to S	dual LP	Admissible Heuristics ($h^{lm\text{-}cuts}$,h^{max}) i-dual	Admissible Heuristics ($h^{lm\text{-}cuts}$,h^{max}) CG k=100	Inadmissible Heuristics (h^{add},h^{add}) i-dual	Inadmissible Heuristics (h^{add},h^{add}) CG k=100
4x4 grid	1	598.4	0.05	0.05	0.04	0.04
	2	540.6	**0.16**	0.22	**0.08**	0.14
	3	546.5	**4.26**	5.95	**2.48**	3.66
	4	622.6	95.02	**58.46**	67.11	**40.46**
5x5 grid	1	timed out (1800)	0.07	0.07	0.05	0.05
	2		**0.48**	0.71	**0.17**	0.31
	3		**15.61**	16.25	**7.75**	8.64
	4		794.07	**451.38**	604.13	**283.18**

Outline

- Background
- Heuristic Search in the Occupation Measure Space
- **Heuristics based on Occupation Measures**
  - Projection-based heuristics for SSPs
  - Operator counting for SSPs
  - Constrained SSPs heuristics
  - Combining Search and Heuristic Computation
- Beyond the Resource Constraints

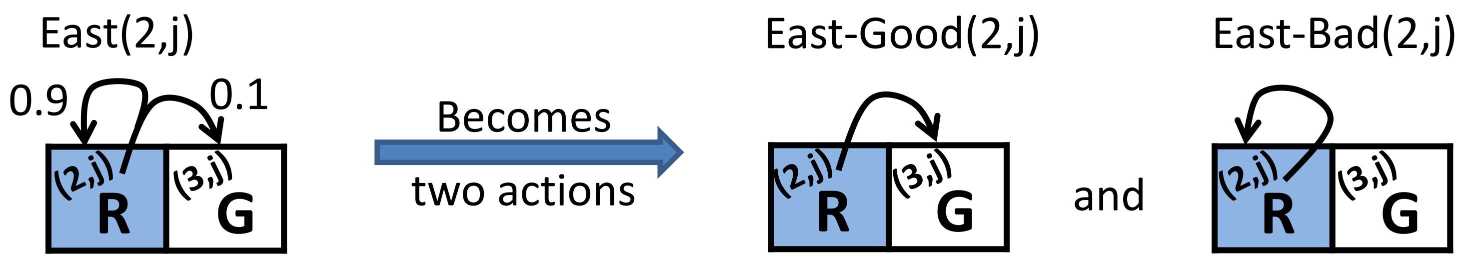
Motivation

- For i-dual to perform well, we need good **heuristics (lower bounds) for Constrained SSPs**
- The approach so far:



Heuristics for SSPs

- Until now, all heuristics for SSPs are based on **determinization**:
  1. Relax the problem into a deterministic problem:



  2. Use any heuristic from deterministic planning in the relaxed problem
- All-outcomes determinization:
  – preserves admissibility
  – but ignores the bad side-effects of actions

Background: Factored SSPs

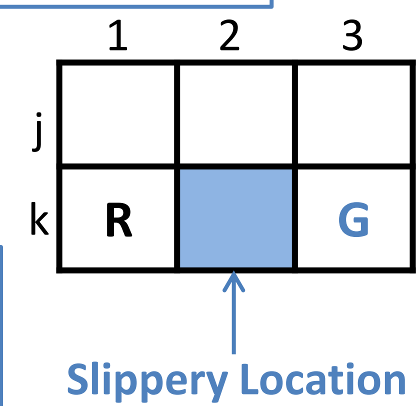
A Probabilistic SAS+ problem is the tuple $[V, s_0, s_*, A, C]$:

- set of **variables** $V = \{v_1, \ldots, v_n\}$ → $V = \{\text{At-X}, \text{At-Y}\}$
  - domain of each variable is D_v → $D_{\textbf{At-X}} = \{1, 2, 3\}$, $D_{\textbf{At-Y}} = \{j, k\}$
- initial state s_0 → At-X = 1, At-Y = j
- goal formula s_* → At-X = 3, At-Y = j
- set of actions A

East(2,j):

Precondition: At-X = 2 and At-Y = j

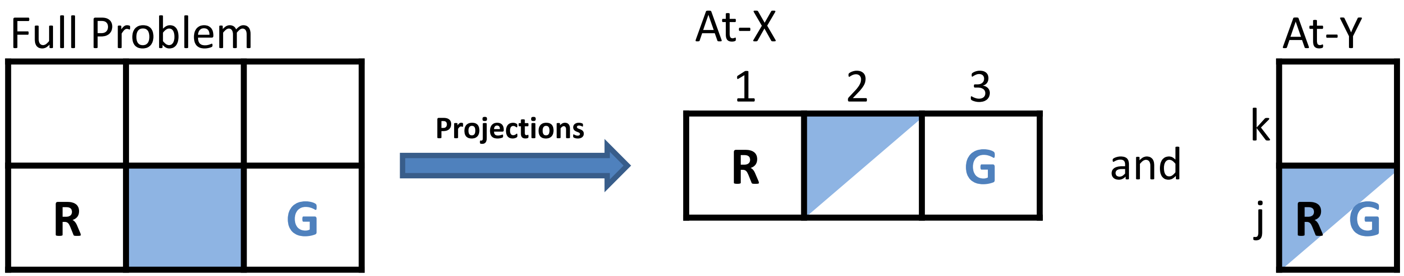
Effect: $\left.\begin{array}{l} 0.1\text{: At-X} \leftarrow 3 \\ 0.9\text{: nothing} \end{array}\right\}$ **Probability distribution of effects**



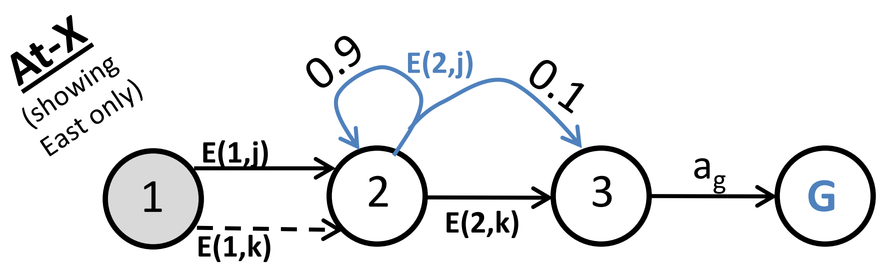
- C(a) cost of action a

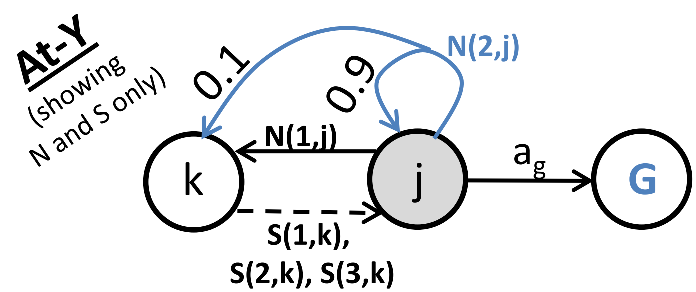
Projections

- A projection onto a variable:



- Formally: represent each projection as an SSP





Optimal solution:

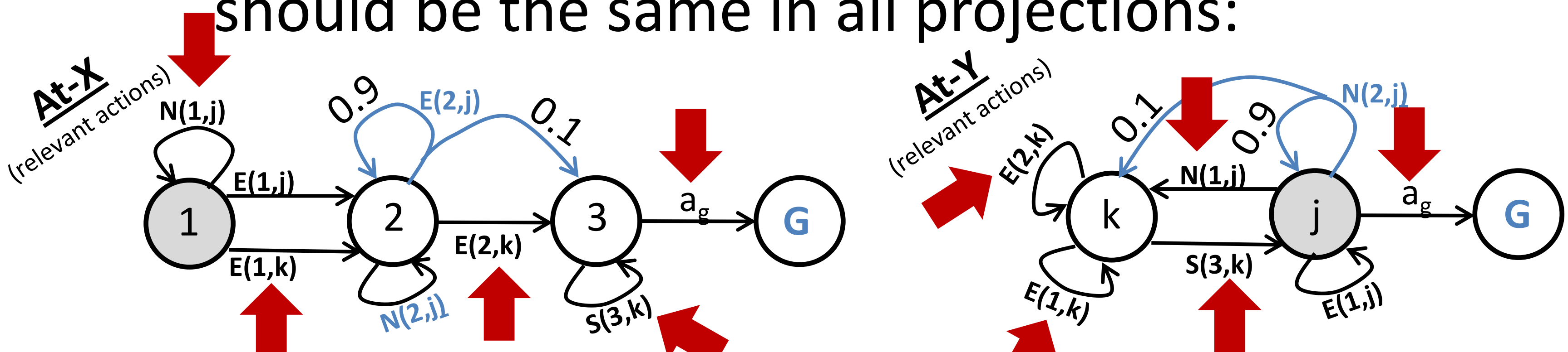
- East(1,j), East(2,k), a_g
- Expected cost: 2

Optimal solution:

- a_g
- Expected cost: 0

Tying Projections Together

- Two ways of using projections:
  - Using **cross-product**: original problem
  - Using them **independently**: no improvements
- <u>Idea</u>: **weakly** tie projections together
  - The expected number of times an action is executed should be the same in all projections:



	Independently	Optimal Solution
At-X	E(1,j) E(2,k) a_g	N(1,j) E(1,k) E(2,k) S(3,k) a_g
At-Y	a_g	
E[Cost]	Infeasible! (2)	**4**

H-POM

The Projection Occupation Measure Heuristic:

- Variables:
  - $x^{v}_{d,a}$: occupation measure for the projection onto variable v
- $\mathrm{h}^{\mathbf{pom}}(s)$ =

$$\min_{x} \sum_{a \in \mathrm{A}} x^{v}_{d,a} \mathrm{C}(a)$$

$$\forall v \in \mathrm{V}, d \in \mathrm{D}_v, a \in \mathrm{A}$$

$$\text{s.t.} \quad \boxed{\begin{aligned} & x^{v}_{d,a} \geq 0 \\ & \sum_{a \in \mathrm{A}(s)} x^{v}_{d,a} - \sum_{\substack{d' \in \mathrm{D}_v \\ a \in \mathrm{A}}} x^{v}_{d',a} \mathrm{P}(d|d',a) = \begin{cases} 1 & \text{value of } v \text{ in } s \text{ is } d \\ -1 & d = \text{art. goal} \\ 0 & \text{otherwise} \end{cases} \end{aligned}}$$

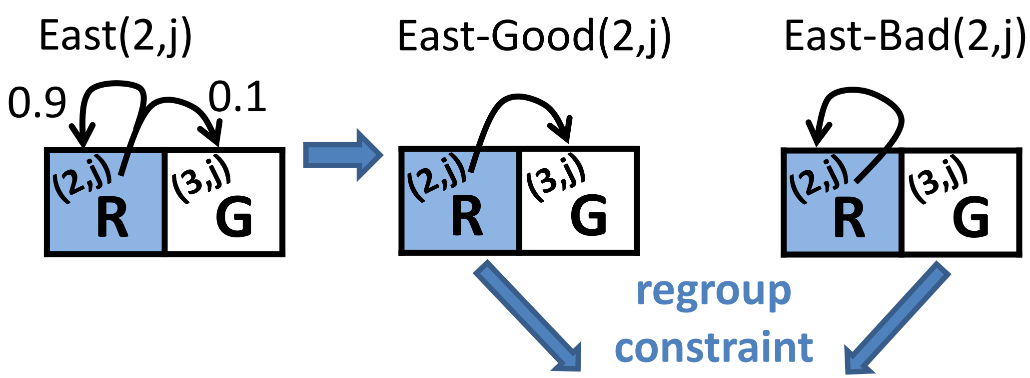
Dual LP for SSPs applied to projection onto v

$$\left. \sum_{d_i \in \mathrm{D}_{v_i}} x^{v_i}_{d,a} = \sum_{d' \in \mathrm{D}_{v_j}} x^{v_j}_{d',a} \quad \forall v_i, v_j, a \in \mathrm{A} \right\}$$

Tying constraint

Determinization is not dead

- We can add similar constraints for determizations that tie together the deterministic effects:



$$\frac{\#\ \text{East-Good}(2,j)}{\#\ \text{East-Bad}(2,j)} = \frac{0.9}{0.1}$$

- These constraints can be added to LP-based heuristics for deterministic planning
  - For instance, operator counting [Pommerening et al., 2014]

H-ROC

The Regrouped Operator Counting Heuristic:

- Variables:
  - $Y_{a,e}$: number of times effect *e* of action *a* is applied in the **all-outcomes determinization**
- h$^{\text{roc}}$(*s*) =

$$\min_{\mathrm{Y}} \sum_{a \in \mathrm{A}} \mathrm{Y}_{a,e} \mathrm{C}(a)$$

$$\text{s.t.} \quad \mathrm{Y}_{a,e} \geq 0 \quad \forall a \in \mathrm{A}, e \in \mathrm{eff}(a)$$

$$\underbrace{\sum_{(a,e) \text{ produces } d} \mathrm{Y}_{a,e} - \sum_{(a,e) \text{ consumes } d} \mathrm{Y}_{a,e} = \begin{cases} -1 & \text{if } d \in s, d' \in s_*, d \neq d' \\ 0 & \text{if } d \in s, d \in s_* \\ 1 & \text{if } d \neq s, d \in s_* \end{cases}}_{\text{Net-change constraints}}$$

$$\underbrace{\mathrm{P}(a,e)\mathrm{Y}_{a,e'} = \mathrm{P}(a,e')\mathrm{Y}_{a,e} \qquad \forall a \in \mathrm{A}, (e,e') \subseteq \mathrm{eff}(a)}_{\text{Regroup constraints}}$$

Experiments

Coverage: # of times (out of 30) the same problem is optimally solved with a different random seed. Max. cputime: 30mins. **Best** coverage in smallest time in **bold**

Blocks World: Put-on-block and Pick-up **can fail**. Towers can be moved but fails with probability 0.9

Parc Printer: Print *n* sheets using a modular printer. Some modules get **jammed** with probability 0.1

Domain	Instance	LRTDP					iLAO				
		h^{max}	h^{lmc}	h^{net}	h^{roc}	h^{pom}	h^{max}	h^{lmc}	h^{net}	h^{roc}	h^{pom}
Blocks World	8	3	0	26	30	30	2	30	30	**30**	30
	8	28	0	30	30	30	30	30	30	**30**	30
	8	2	0	12	30	29	2	30	30	**30**	30
	10	0	0	0	30	18	0	0	1	**30**	30
	10	0	0	0	30	0	0	0	0	**30**	30
	12	0	0	0	0	0	0	0	0	**30**	5
Parc Printer	F,4,2	30	30	30	**30**	30	4	30	30	30	30
	F,4,3	30	30	30	**30**	30	0	30	30	30	30
	F,5,2	0	30	0	**30**	0	2	16	0	30	0
	F,5,3	0	30	0	**30**	0	0	0	0	30	0
	T,4,2	0	0	0	1	0	1	30	30	**30**	0
	T,4,3	0	0	0	0	0	0	30	30	**30**	0
	T,5,1	0	0	0	0	0	0	0	0	**30**	0

Best performing solver for each problem uses h^{roc}

h^{roc} vs h^{pom}: performance

h^{roc} is faster than h^{pom} because:

- **h^{roc}** solves a **smaller LP:**
  - h^{roc} : $Y_{a,e}$ defined for all action *a* and effect *e* of *a*
  - h^{pom} : $x^{v}_{d,a}$ defined for all values *d* of all state variables *v* and all actions *a*
  - h^{pom} also has more constraints than h^{roc}
- h^{roc} returns the **same lower bound** as h^{pom}

Adding Cost Constraints

- Recap of **Constrained SSPs** (C-SSPs):
  - SSP with **multiple cost functions**:
    - C_0: cost function to be **minimized**
    - $C_1 \ldots C_n$: cost functions with **expectation upper bounded** by $u_1 \ldots u_n$

- Heuristics for C-SSPs so far:

Problem:

R	G

Moving with speed control

Action	$C_1(a)$ Time	$C_2(a)$ Fuel
East-Slow	10	1
East-Normal	3	3
East-Fast	1	10

Constraints:
- Expected Time ≤ 4
- Expected Fuel ≤ 4

Conflicting recommendation

  - Treat each cost independently: $\begin{cases} H_1(s) = 1 & \text{(East-Fast)} \\ H_2(s) = 1 & \text{(East-Slow)} \end{cases}$

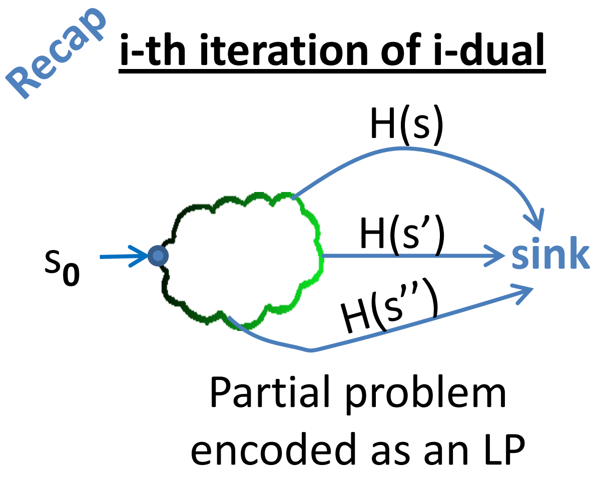
Constrained versions of h^{pom} and h^{roc}

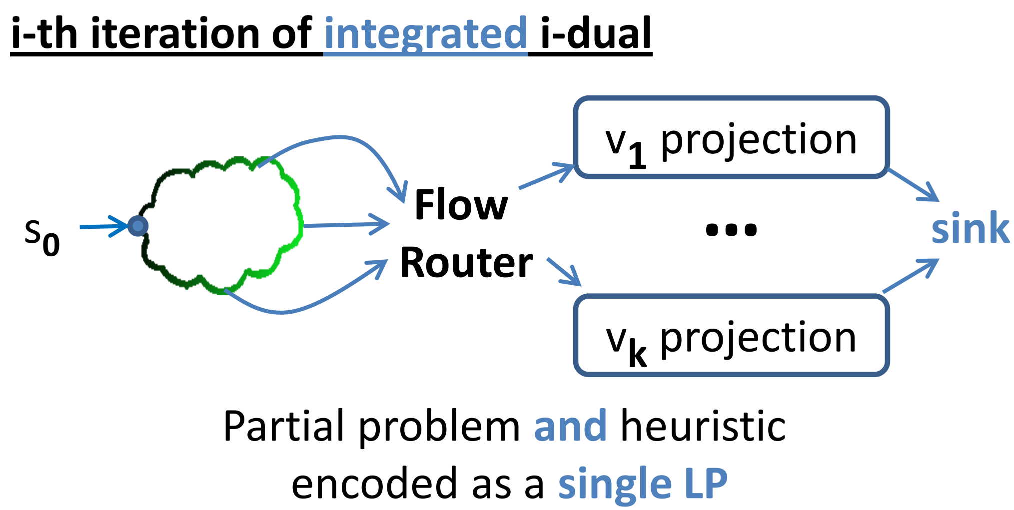
- Since both h^{pom} and h^{roc}:
  - are defined using **LPs**, and
  - **count** the number of times **actions** are executed
- Then we can directly add the cost constraints:
  - h^{pom}: $\sum_{d \in \mathsf{D}_v, a \in \mathsf{A}} x^v_{d,a} C_i(a) \leq u_i$
  - h^{roc}: $\sum_{(a,e) \in \mathsf{A}} Y_{a,e} C_i(a) \leq u_i$
- The result are the heuristics $h^{c\text{-}pom}$ and $h^{c\text{-}roc}$:
  - admissible for C-SSPs
  - take probabilities in consideration
  - take cost constraints in consideration

Combining Search and Heuristic Computation

- $h^{c\text{-}pom}$ can be **integrated** with i-dual:

Recap





- This allows to compute **at the same time**
  - the expected cost of a partial solution π
  - $h^{c\text{-}pom}$ for all states in fringe of π

  neither drive each other, they work in unison

- We call this algorithm **i^2-dual**

Experiments: Constrained SSPs

Coverage: # of times (out of 30) the same problem is optimally solved with a different random seed. Max. cputime: 30mins. **Best** coverage in smallest time in **bold**

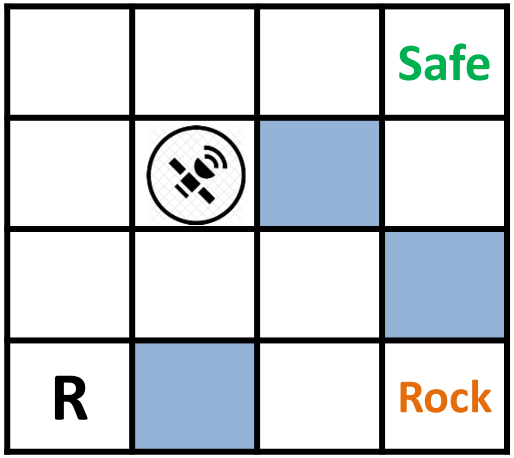
		i-dual						i^2-dual
		h^{max}	$h^{lmc\text{-}m}$	h^{roc}	$h^{c\text{-}roc}$	h^{pom}	$h^{c\text{-}pom}$	
Parc Printer Same as in SSP case. **Constraint** on the expected # of paper jams **and** expected usage of reliable module	$0, 1$	30	30	30	30	25	28	**30**
	$0, \infty$	30	**30**	30	30	30	30	**30**
	$0.1, 1$	0	0	0	30	0	27	**30**
	$0.1, \infty$	0	0	0	30	0	30	**30**
	$0.2, 1$	0	0	0	0	0	0	**30**
	$0.2, \infty$	0	0	0	0	0	0	**30**
Search and Rescue Grid navigation to rescue one survivor. There are potentially multiple survivors. **Constraint** on expected fuel consumption.	$4, 0.50, 3$	**30**	**30**	**30**	**30**	30	30	**30**
	$4, 0.50, 4$	29	**30**	30	30	29	30	**30**
	$4, 0.75, 3$	26	**30**	29	29	28	28	**30**
	$4, 0.75, 4$	0	4	1	1	1	1	**7**
	$5, 0.50, 3$	30	**30**	30	30	30	30	**30**
	$5, 0.50, 4$	5	9	9	9	9	9	**14**
	$5, 0.75, 3$	19	**28**	23	23	20	21	**28**
	$5, 0.75, 4$	0	2	2	2	1	1	**6**

i^2-dual out performs all combos of planner and heuristic

Outline

- Background
- Heuristic Search in the Occupation Measure Space
- Heuristics based on Occupation Measures
- **Beyond the Resource Constraints**

Beyond Resource Constraints



Goal: Analyse rock then go to the safe location

Cost Constraints: On energy (cost constraint)

Linear Temporal Logic (LTL) Constraints:

- **G**(rock has evidence of life → **F** transmit data)
  - » Translation: *every time a rock has evidence of life, transmit the data before finishing the mission*

Probabilistic LTL Constraints:

- Pr[**F**(transmit data)] ≥ 0.5
  - » Translation: *with probability at least 0.5 transmit the data before finishing the mission*
- Pr[**G**(sand storm → $\mathbf{F}^{\leq 3}$(at safe location **Until** ¬(sand storm))] ≥ 0.9
  - » Translation: *with probability at least 0.9, every time a sandstorm happens, in at most 3 time steps, the robot must be in the safe location and it remains there until the sand storm is over*

C-SSPs with PLTL Constraints

- Solution to C-SSPs + PLTL constraints are **finite-memory** stochastic policies

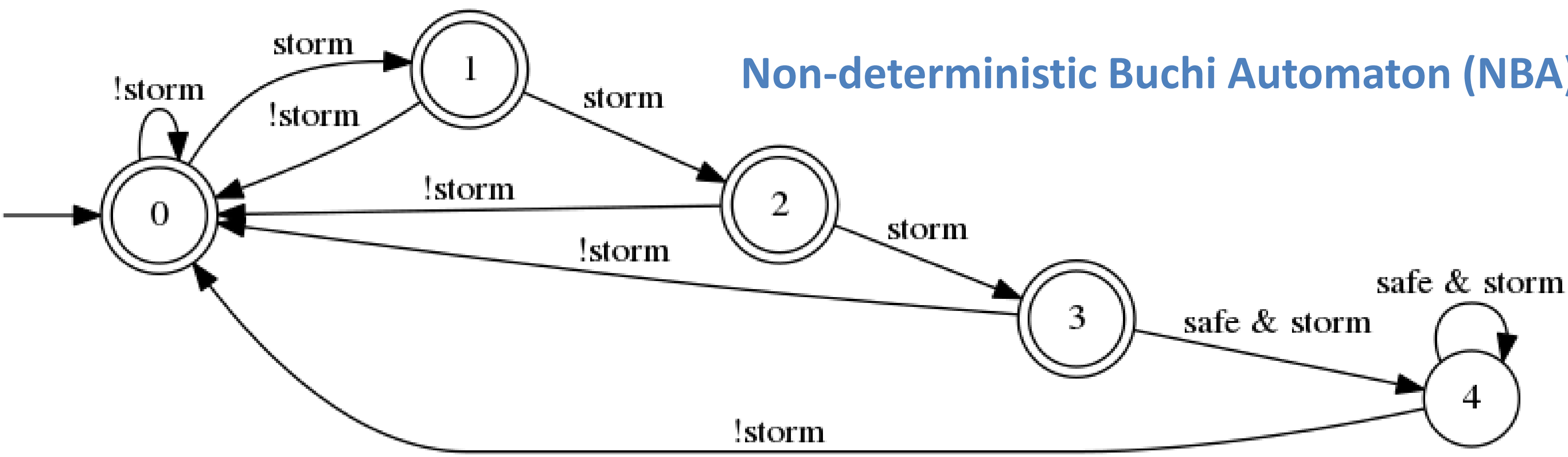
  → The policy needs to be aware of the *status* of the formulas

- Example:

  G(sand storm → **F**$^{\leq 3}$(at safe location **Until ¬**(sand storm))

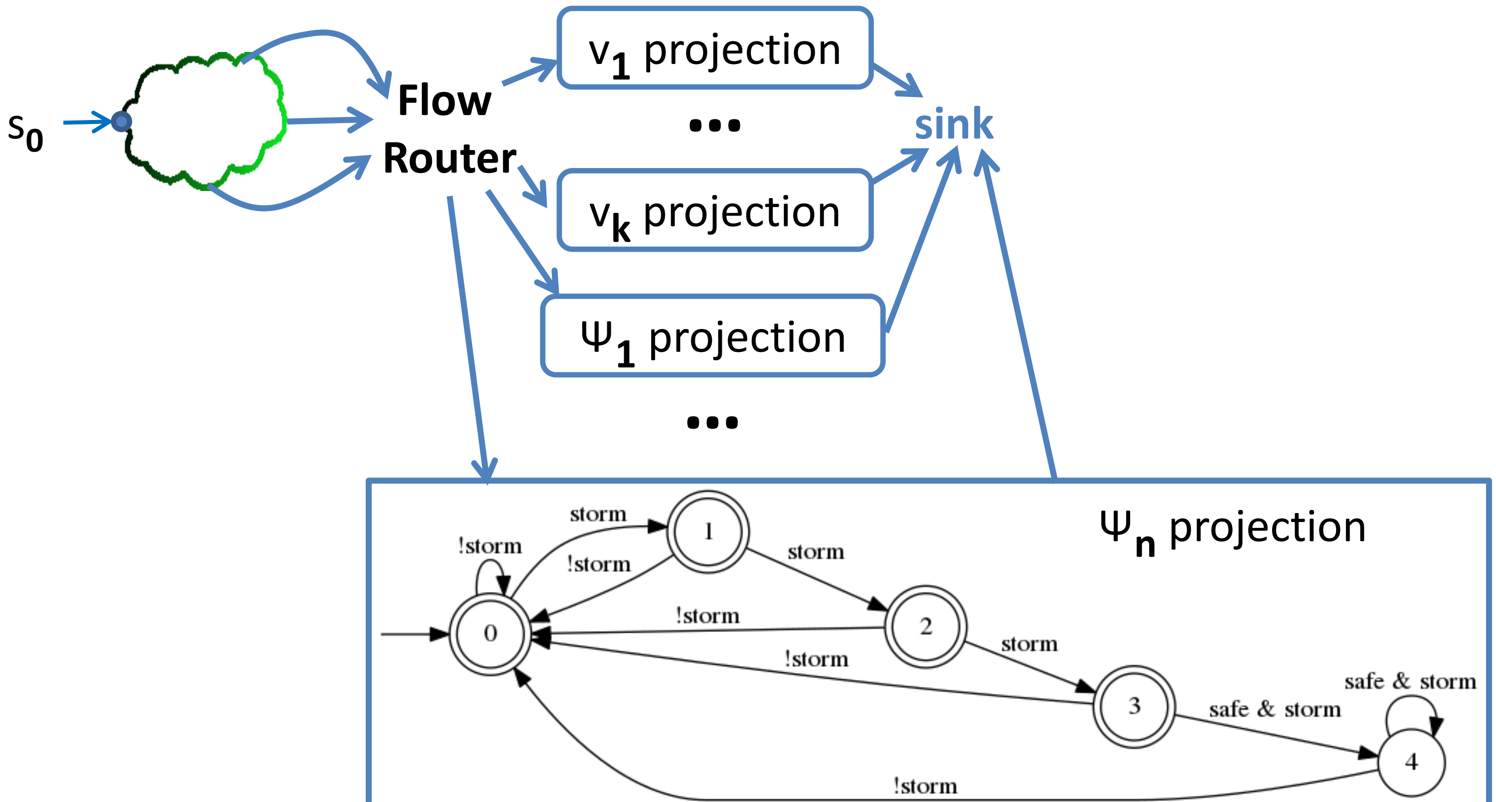
  » Translation: *every time a sandstorm happens, in at most 3 time steps, the robot must be in the safe location and it remains there until the sand storm is over*

Non-deterministic Buchi Automaton (NBA)

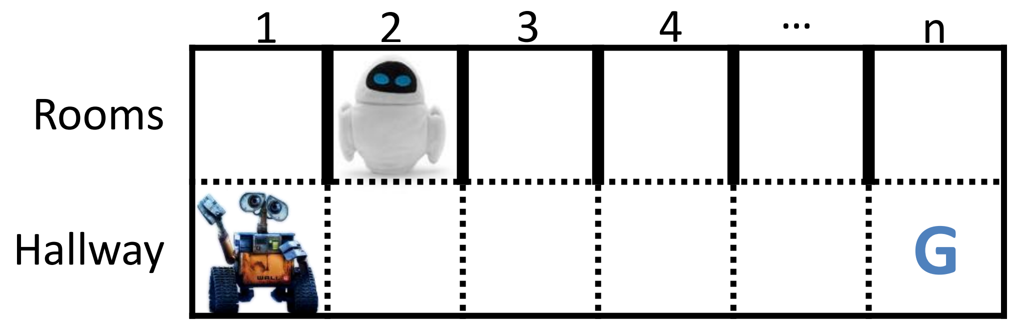


PLTL-dual

- Our approach:
  - Embed the formula tracking into the state space
  - Extend i^2-dual with extra projections for the formulas



Experiment: Wall-e and Eve



- **Goal**: Wall-e at **G**
- **Constraints**:
  1. Wall-e and Eve must eventually be together (P ≥ 0.5)
  2. Eve must be in a room until they are together (P ≥ 0.8)
  3. Once together, they eventually stay together (P = 1)
  4. Eve must visit the rooms 1, 2, and 3 (P = 1)
  5. Wall-e never visits a room twice (P ≥ 0.8)

	n =	**4**	**5**	**6**	**7**
PLTL-dual	no PLTL heuristic	15.9	83.4	472.8	---
PLTL-dual	NBA proj. heur.	9.2	**52.7**	280.6	---
PLTL-dual	NBA proj. heur. (100)	9.1	52.8	**142.1**	**572.7**
	PRISM	**8.5**	68.1	---	---

Summary

- Occupation measure space:
  - represents problems as a **probabilistic flow networks** where each $x_{s,a}$ is the expected number of times action a is executed in state s
  - is equivalent to the **stochastic policy space**
- Occupation measures allow us to
  - derive the first domain-independent **heuristics that take probabilities into account** and also constraints
  - **efficiently solve** problems with
    - **Cost constraints**
    - **PLTL constraints**

Some Open Questions

- Bounds for occupation measures:
  - When can we easily find a lower bound for $x_{s,a}$?
  - Can we efficiently compute an upper bound for $x_{s,a}$?
- Specialization of occupation measures for SSPs:
  - Is it possible to efficiently compute deterministic policies for SSPs in the dual space?
- How much more expressive can we make the constraints in the dual space?

Work done in collaboration with

From Australian National University (ANU) & Data61 (formerly NICTA):

Sylvie Thiébaux, Patrik Haslum, Peter Baumgartner

From MIT:

Brian Williams, Pedro Santana

Thank you!

Questions?