Computing Superior Counter-Examples for Conformant Planning:

Xiaodi Zhang, Alban Grastien, Enrico Scala

Australian National University

Australian National University, February 8, 2020

Conformant Planning — Motivation

Problem:

- Conformant planning = find a plan that leads to a given goal
- Uncertainty in the initial state and no observability
- No uncertainty on the action effect (deterministic conformant planning)

Motivation:

- Useful for robots with little processing capability and in dangerous environments
- Target language from probabilistic conformant planning and epistemic planning
- The ideas will apply for more sophisticated problems

Conformant Planning — Example

Dispose (simplified):

- Three items 1 to 3, four locations A to D
- Initial location of each item unknown
- Goal: drop all items in another location T
- Actions:
  - Go-to: moves the robot
  - Pick-up: grabs the item if it is where the robot is
  - Drop: drops the item if the robot is holding it
- One solution:
  - go-to A, pick-up 1, pick-up 2, pick-up 3
  - go-to B, pick-up 1, pick-up 2, pick-up 3
  - go-to C, pick-up 1, pick-up 2, pick-up 3
  - go-to D, pick-up 1, pick-up 2, pick-up 3
  - go-to T, drop 1, drop 2, drop 3

Conformant Planner: gCPCES

Assuming the problem is "easy" if the set of initial states is small

- $\mathcal{B} := \{ \}$
- **repeat**
  - $\pi :=$ *compute-plan*$(\mathcal{B})$
  - **if** no π
    - **return** unsolvable
  - $q :=$ *compute-counter-example*(π)
  - **if** no q
    - **return** π
  - $\mathcal{B} := \mathcal{B} \cup \{q\}$

Illustration on Dispose :

counter-ex.	1	2	3	4
Init loc of item 1	A	B	C	D
Init loc of item 2	A	B	D	C
Init loc of item 3	A	B	C	D

Illustration on Dispose :)

counter-ex.	1	2	3	4
Init loc of item 1	A	B	C	D
Init loc of item 2	A	B	D	C
Init loc of item 3	A	B	C	D

What happens in practice :(

counter-ex.	1	2	3	4	5	6	7	8	9	10
Init loc of item 1	A	B	C	D	A	A	A	A	A	A
Init loc of item 2	A	A	A	A	B	C	D	A	A	A
Init loc of item 3	A	A	A	A	A	A	A	B	C	D

Our Goal

We want to minimise the number of counter-examples that are generated by gCPCES

1. Fewer iterations
   → faster (?) gCPCES
2. Smaller set of counter-examples
   → better "explanation"
3. More diverse counter-examples
   → less "biased" plans when using non-admissible heuristics

Question:

- How do we know that q' is a better counter-example than q?

Superiority: Intuition

- Let $\mathcal{B}_1 \subset \mathcal{B}_2 \subset \ldots$ be the sequence of samples built by gCPCES
- Then: $\Pi(\mathcal{P}[\mathcal{B}_1]) \supset \Pi(\mathcal{P}[\mathcal{B}_2]) \supset \cdots \supseteq \Pi(\mathcal{P})$
- gCPCES terminates when $\Pi(\mathcal{P}[\mathcal{B}]) = \Pi(\mathcal{P})$ (sometimes before)

$\rightarrow$ To accelerate convergence, we want to minimise $\Pi(\mathcal{P}[\mathcal{B}_i])$ at each i

Superiority: Intuition (continued)

Properties we are looking for: if q' is superior to q (given $\mathcal{B}$)

1. $\Pi(\mathcal{B} \cup \{q\}) \supseteq \Pi(\mathcal{B} \cup \{q'\})$ $\leftarrow$ so q' is better now

2. for all subset $\mathcal{B}'$ of initial states:
   $\Pi(\mathcal{B} \cup \{q\} \cup \mathcal{B}') \supseteq \Pi(\mathcal{B} \cup \{q'\} \cup \mathcal{B}')$ $\leftarrow$ so q' will be better

I.e., q' is always better than q

Known Notions: Tags

(Palacios & Geffner, 2009; Albore, Palacios, & Geffner, 2010)

- A plan is valid iff
  - all its actions' preconditions are satisfied when they are applied
  - and the goal is satisfied at the end

  $\rightarrow$ validity condition

- The context of a validity condition φ is the list of all variables that φ depends on (including through other actions)
  Example in dispose:
  - Context of disposed(i) $=$ { disposed(i), holding(i), location(i) }

- A tag t is a possible initial assignment of the variables in the context of a validity condition

Known Notions: Tags (continued)

- An initial state q exhibits a number of tags: $Tags(q)$
- It is possible to associate each tag t with a set of plans $\Pi(t)$ such that:
- The set of valid plans of problem $\mathcal{P}$ is:

$$\Pi(\mathcal{P}) = \bigcap_{t \in Tags(q),\ q \in I} \Pi(t)$$

Tag-Based Superiority (definition)

Remember:

$$Tags(\mathcal{B}) \subseteq Tags(\mathcal{B}') \Rightarrow \Pi(\mathcal{P}[\mathcal{B}]) \supseteq \Pi(\mathcal{P}[\mathcal{B}'])$$

- Let $\mathcal{B} \subseteq I$ be a sample
- Let q and q' be two counter-examples
- q' is superior to q (given $\mathcal{B}$) if:

$$Tags(\mathcal{B} \cup \{q\}) \subset Tags(\mathcal{B} \cup \{q'\})$$

Computing Superior Counter-Examples

Let q be the current counter-example and $\mathcal{B}$ the sample
Let $C_1, \dots, C_k$ be the contexts
Let $t_{i,1}, \dots, t_{i,p}$ be the tags of C_i in $\mathcal{B}$
Let t_i be the tag of q for C_i
Let j be such that $t_i \notin \{t_{i,1}, \dots, t_{i,p}\}$ is a new tag iff $i \leq j$

Then

$$\mathit{Initial\_State} \wedge \bigwedge_{i \in \{1,\dots,j\}} t_i \wedge \neg \left(\bigwedge_{i \in \{j+1,\dots,k\}} \bigvee_{\ell \in \{1,\dots,p\}} t_{i,\ell} \right)$$

is satisfiable iff there is a counter-example superior to q

Experiments (setup)

Planners:

- gCPCES (using `z3` and `ff`)
- new CPCES: Superb (using `z3` and `ff`)
- T1, a planner based on Conformant FF that performs very well when the contexts include only one unknown variable

Experiments (expectations)

Definitions: a problem instance is

- vertical if all contexts include exactly one variable initially unknown ("width" = 1)
- horizontal if all contexts are identical

We expect (">" means "faster"):

- Vertical & horizontal: trivial problems
- Vertical & non-horizontal: T1> SUPERB> gCPCES
- Non-vertical & horizontal: gCPCES= SUPERB> T1
- Non-vertical & non-horizontal: SUPERB> gCPCES> T1

Experiments (benchmarks)

(crudely)

- Vertical & non-horizontal: DISPOSE, COINS, BOMB, UTS
- Non-vertical & horizontal: BLOCKWORLD, RAOSKEY, EMPTYGRID, WALLGRID, DISPOSE-ONE, LOOKANDGRAB
- Non-vertical & non-horizontal: (new domain!) MAWALLGRID

Experiments (results)

	Coverage			Plan Quality			Planning Time		
Domain	C	S	T1	C	S	T1	C	S	T1
LookAndGrab(18)	**18**	**18**	15	42	42	34	22	36	117
BlockWorld(3)	**3**	**3**	2	13	13	13	0.7	0.8	0.2
UTS(15)	**13**	**13**	11	36	36	41	3	4	0.2
RaosKeyS(2)	**2**	**2**	1	16	16	21	0.6	1.2	0.5
Dispose-One(10)	**5**	**5**	4	62	68	79	30	67	377
wallgrid(18)	**18**	**18**	4	18	18	18	0.7	0.9	0.1
emptygrid(4)	**4**	**4**	**4**	18	18	18	0.6	1.3	0.1
Bomb(9)	7	**9**	**9**	106	106	101	96	4	0.1
Coins(9)	8	8	**9**	88	86	149	3	3	0.6
dispose(11)	4	6	**8**	184	184	212	580	259	6

Experiments (results, continued)

MAWALLGRID

	Planning Time		Iterations		Sampling Time		
Pro	C	S	C	S	C	S	T1Time
4_4_2	1.43	**1.17**	10	**7**	0.41	0.42	*0.1*
4_4_3	20.02	**10.34**	19	**11**	0.86	1.09	*0.3*
6_6_2	4.29	4.25	13	**12**	0.7	1.14	*0.1*
6_6_3	1037.74	**904.75**	14	14	1.08	1.74	*4.9*
8_8_2	124.14	**77.75**	29	**25**	2.74	3.31	*TO*
8_8_3	TO	TO	NA	NA	NA	NA	*TO*
10_10_2	**874.49**	1876.62	**40**	50	4.11	9.75	*TO*
10_10_3	TO	TO	NA	NA	NA	NA	*TO*
11_11_2	2287.07	**1606.3**	43	**38**	6.09	9.3	*TO*
11_11_3	TO	TO	NA	NA	NA	NA	*TO*

Conclusion

- We identify that some counter-examples are more informative than others in the context of gCPCES
- We show one characterisation of this relation ("superiority")
- We show how to compute maximally-superior counter-examples
- We show experimentally the benefits of this approach

More broadly:

- We combine a technique that is oblivious of the structure (gCPCES) with a technique that leverages on the structure (superiority)
- Can we characterise informativeness more precisely?
- Can we import this type of solution in other problems?