Solving Multiagent Planning Problems with Concurrent Conditional Effects

Daniel Furelos-Blanco[1] and Anders Jonsson[2]

[1]Imperial College London

[2]Universitat Pompeu Fabra

January 18, 2019

Motivation

What is concurrent multiagent planning?

- Agents collaborate to solve a problem.
- Collaboration = concurrent/joint actions executed simultaneously by multiple agents.

What is the challenge?

- The number of joint actions is worst-case exponential in the number of agents.
- Few planners are designed to handle concurrency.

Build planner that supports different kinds of concurrency efficiently.

Proposed approach

Solve multiagent planning problems that involve concurrency by translating them into classical planning.

Concurrency expressed using **concurrency constraints** which model when

1. two actions **must** occur in parallel, or
2. two actions **cannot** occur in parallel.

Concurrent Multiagent Planning - Definition

- A **classical planning** problem is defined as

$$\Pi = \langle F, A, I, G \rangle$$

where

  - F is a set of fluents,
  - A is a set of atomic actions,
  - $I \subseteq F$ is an initial state, and $G \subseteq F$ is a goal condition.

- A **multiagent planning** problem (MAP) is a tuple

$$\Pi = \left\langle N, F, \left\{A^i\right\}_{i=1}^{n}, I, G \right\rangle$$

where $N = \{1, \ldots, n\}$ is the agent set, and A^i is the action set of agent $i \in N$.

Concurrent Multiagent Planning - Joint Actions

- Each action is a **joint/concurrent action**: a combination of atomic actions simultaneously performed.
- Given a concurrent action $a = \left(a^1, \ldots, a^k\right)$, its precondition and effects are defined as

$$\mathsf{pre}(a) = \bigcup_{j=1}^{k} \mathsf{pre}(a^j),\ \mathsf{eff}(s, a) = \bigcup_{j=1}^{k} \mathsf{eff}(s, a^j)$$

- Constraints are imposed on atomic actions to ensure joint actions are well-defined.

Concurrent Multiagent Planning - Concurrency Constraints

- Formulation in [Boutilier and Brafman, 2001] (later extended in [Kovacs, 2012]) uses actions as **fluents**:
  - **Positive concurrency:** action a^1 has a^2 as precondition.
  - **Negative concurrency:** action a^1 has $\neg a^2$ as precondition.

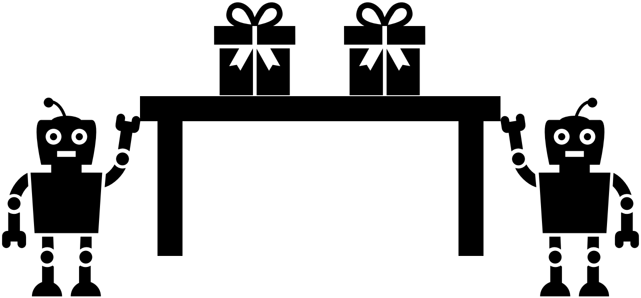
- **Effects** of an action a^1 can be **conditioned** to the simultaneous execution of another action a^2.

- Each agent contributes **at most once** to the joint action.

Concurrent Multiagent Planning - Example

TableMover [Boutilier and Brafman, 2001]:

- Two agents must move blocks between rooms.
- Put blocks on a table, carry the table *together* to another room, and tip the table to make the blocks fall down.



Concurrent Multiagent Planning - Example

TableMover [Boutilier and Brafman, 2001]:

- Two agents must move blocks between rooms.
- Put blocks on a table, carry the table *together* to another room, and tip the table to make the blocks fall down.

Concurrent Multiagent Planning - Example

```
(:action lift-side
 :agent ?a - agent
 :parameters (?s - side)
 :precondition (and
   (at-side ?a ?s)
   (down ?s)
   (handempty ?a)
   (forall
    (?a2 - agent ?s2 - side)
    (not(lower-side ?a2 ?s2))
   )
 )
 :effect (and (not (down ?s))
  (up ?s)
  (lifting ?a ?s)
  (not (handempty ?a ?s))
  ...
```

```
...
(forall
 (?b - block ?r - room ?s2 -
     side)
 (when
  (and (inroom Table ?r)
   (on-table ?b)
   (down ?s2)
   (forall (?a2 - agent)
    (not (lift-side ?a2 ?s2))
   )
  )
  (and (on-floor ?b)
   (inroom ?b ?r)
   (not (on-table ?b))
  )
 )
)))
```

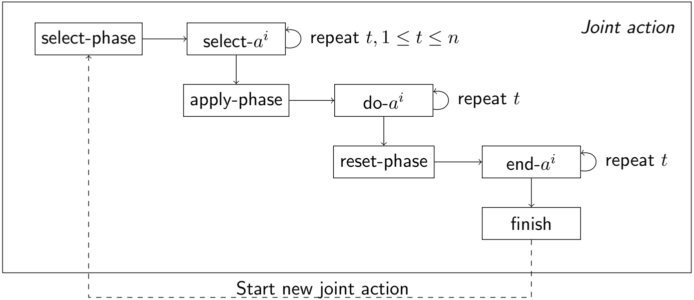
Compilation from Multiagent to Classical Planning (I)

- Transform a MAP $\Pi = \left\langle N, F, \left\{A^i\right\}_{i \in N}, I, G \right\rangle$ into a classical planning problem $\Pi' = \langle F', A', I', G' \rangle$.

- Sound and complete transformation:
  - Add fluents and actions to simulate joint actions while respecting concurrency constraints.

- Divide simulation of a joint action in three different phases:
  1. **Action selection:** check preconditions of constituent atomic actions.
  2. **Action application:** apply effects of constituent atomic actions.
  3. **Resetting:** reset auxiliary fluents.

Compilation from Multiagent to Classical Planning (II)



The resulting number of actions is **polynomial**, not exponential:

$$|A'| = 3 \sum_{i \in N} |A^i| + 4.$$

Compilation from Multiagent to Classical Planning (III)

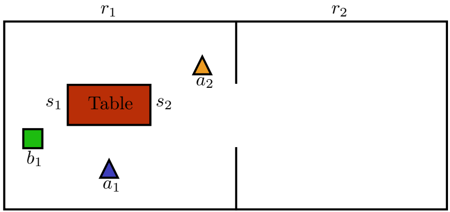
Extension: joint actions with bounded size C.

- At most C agents can act at a time.
- Purpose: reduce branching factor.
- The number of actions is still polynomial:

$$\left|A'\right| = (2C + 1) \sum_{i \in N} \left|A^i\right| + 4.$$

Compilation from Multiagent to Classical Planning (IV)

Example



Multiagent plan

1 (to-table a1 r1 s2)(pickup-floor a2 b1 r1)
2 (putdown-table a2 b1 r1)
3 (to-table a2 r1 s1)
4 (lift-side a1 s2)(lift-side a2 s1)
5 (move-table a1 r1 r2 s2)(move-table a2 r1 r2 s1)
6 (lower-side a1 s2)

Classical plan (1st joint action)

1 (select-phase)
2 (select-to-table a1 r1 s2)
3 (select-pickup-floor a2 b1 r1)
4 (apply-phase)
5 (do-pickup-floor a2 b1 r1)
6 (do-to-table a1 r1 s2)
7 (reset-phase)
8 (end-to-table a1 r1 s2)
9 (end-pickup-floor a2 b1 r1)
10 (finish)

Experiments

Tests on domains that require concurrency:

- TableMover [Boutilier and Brafman, 2001].
- Maze [Crosby et al., 2014].
- BoxPushing [Brafman and Zoran, 2014].
- Workshop.

Test three **variants** of the compilation + Fast-Downward:

- Unbounded (∞).
- Joint action size $\leq$ 2 ($C = 2$).
- Joint action size $\leq$ 4 ($C = 4$).

Experiments - Required Concurrency Domains (I)

- MAZE - Move between two cells in a grid using:
  - Doors: traversed only by one agent at a time.
  - Bridges: can be traversed by multiple agents at once.
  - Boat: used by two or more agents at once (same direction).
- BOXPUSHING - Push boxes between two locations in a grid.
  - A small box requires 1 agent to push.
  - A medium box requires 2 agents to push.
  - A large box requires 3 agents to push.

Experiments - Required Concurrency Domains (II)

- TABLEMOVER - Move blocks between rooms using a table.
  - The table must be moved simultaneously.
  - The blocks on the table fall if only one side is lifted.
- WORKSHOP - Inventory pallets in a high-security facility.
  - Open door = press switch + turn key.
  - Inventory a pallet = lift pallet + examine pallet.

Experiments - Planners for comparison (I)

Compare our approach with CJR [Crosby et al., 2014] and SB [Shekhar and Brafman, 2018]:

- Compilations to classical planning.
- Concurrency constraints in the form of affordances on subsets of objects.
- Main limitation:
    - Concurrency constraints are not as expressive $\rightarrow$ *Conditional effects on simultaneous actions are not supported.*

Experiments - Planners for comparison (II)

CJR [Crosby et al., 2014]

- Effects are applied immediately for atomic actions → *Some joint actions cannot be simulated.*

SB [Shekhar and Brafman, 2018]

- Adds mechanisms to avoid CJR problem (deferred effects).
- Concurrency constraints can only be defined if an object is shared → WORKSHOP domain not supported.
- Effects cannot be conditioned to the execution of an arbitrary action.

Experiments - Results (I)

Domain	N	Coverage					Time (s.)					Makespan					# Grounded actions ($\times 10^3$)				
		2	4	∞	CJR	SB	2	4	∞	CJR	SB	2	4	∞	CJR	SB	2	4	∞	CJR	SB
Maze	20	**13**	8	6	11	9	361.5	444.2	**145.6**	195.1	216.1	47.2	22.0	**11.7**	77.3	67.7	41.7	69.3	**27.9**	156.8	108.2
$a = 10$	10	**8**	6	5	7	6	250.2	575.6	**170.4**	228.4	323.1	48.3	25.0	**12.2**	79.6	69.8	39.9	67.4	**26.1**	119.3	102.1
$a = 15$	10	**5**	2	1	4	3	**539.5**	-	-	-	-	**45.4**	-	-	-	-	43.9	71.8	**30.0**	194.3	115.1
BoxPushing	20	9	15	16	-	**18**	**5.2**	36.4	143.3	-	305.8	**11.2**	11.3	12.9	-	20.5	3.5	5.7	2.5	-	**2.0**
$a = 2$	10	9	9	9	-	**10**	**5.2**	7.6	6.0	-	158.9	**11.2**	11.9	11.3	-	18.4	1.8	3.2	**1.1**	-	1.2
$a = 4$	10	0	6	7	-	**8**	-	**79.7**	319.9	-	489.5	-	**10.5**	15	-	23.1	5.2	8.2	3.8	-	**2.9**
TableMover	24	**15**	12	**15**	-	-	**263.4**	336.7	341.1	-	-	**58.7**	59.0	61.5	-	-	7.4	13.1	**4.6**	-	-
$a = 2$	12	10	10	**11**	-	-	**103.9**	226.6	214.7	-	-	63.5	**62.0**	64.5	-	-	3.4	6.1	**2.0**	-	-
$a = 4$	12	**5**	2	4	-	-	**582.4**	-	-	-	-	**49.0**	-	-	-	-	11.5	20.1	**7.2**	-	-
Workshop	20	**15**	13	13	-	-	134.3	301.4	**52.5**	-	-	35.7	37.0	**32.5**	-	-	18.0	31.0	**11.5**	-	-
$a = 4$	10	**8**	**8**	**8**	-	-	42.8	263.3	**37.1**	-	-	**37.3**	43.9	**37.3**	-	-	7.7	13.6	**4.8**	-	-
$a = 8$	10	**7**	5	5	-	-	238.8	362.3	**77.1**	-	-	33.9	26.0	**24.8**	-	-	28.2	48.3	**18.1**	-	-

- Unbounded compilation (∞) has the highest coverage.
- Compilation $C = 2$ is usually fast but cannot solve problems involving > 2 agents.
- Our approach can solve a wider range of problems.

Experiments - Results (II)

#Agents	# Grounded actions		Time (s.)	
	Naive	∞	Naive	∞
2	48	100	0.089	0.226
4	992	260	0.494	0.226
6	31248	484	53.864	0.354
8	-	772	-	0.535
10	-	1124	-	0.758
50	-	21604	-	41.979
100	-	83204	-	289.887

- Compare our approach to "naive" compilation in the MAZE domain.
- Instances = 3x3 grid, k agents have the same starting and goal locations, single path to the goal (bridges + boats).

Our approach scales much better!

Conclusions

- Sound and complete method for compiling MAPs into classical planning problems.
- The number of resulting actions is polynomial in the description of the MAP.
- Handles concurrency constraints including conditional effects.
- Solves problems out of reach of previous approaches.

Questions

- Contact:
  - d.furelos-blanco18@imperial.ac.uk
  - anders.jonsson@upf.edu
- Software: `https://github.com/aig-upf/universal-pddl-parser-multiagent`

Boutilier, C. and Brafman, R. I. (2001).
Partial-Order Planning with Concurrent Interacting Actions.
J. Artif. Intell. Res. (JAIR), 14:105–136.

Brafman, R. I. and Zoran, U. (2014).
Distributed Heuristic Forward Search with Interacting Actions.
In *Proceedings of the 2nd ICAPS Distributed and Multi-Agent Planning workshop (ICAPS DMAP-2014)*.

Crosby, M., Jonsson, A., and Rovatsos, M. (2014).
A Single-Agent Approach to Multiagent Planning.
In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 237–242.

Kovacs, D. L. (2012).
A Multi-Agent Extension of PDDL3.1.
In *Proceedings of the 3rd Workshop on the International Planning Competition (IPC)*, pages 19–27.

Shekhar, S. and Brafman, R. I. (2018).
Representing and planning with interacting actions and privacy.
In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018.*, pages 232–240.