Solving Concurrent Multiagent Planning using Classical Planning

Daniel Furelos-Blanco and Anders Jonsson

Universitat Pompeu Fabra

June 26, 2018

Motivation

Lot of progress in multiagent planning since CoDMAP-15.

- **Limitation:** benchmark domains can be solved using sequential plans.

Many applications **require** agents to act in parallel, like RoboCup Soccer or RoboCup Rescue.

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 **concurrent multiagent planning** problem (CMAP) 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

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

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

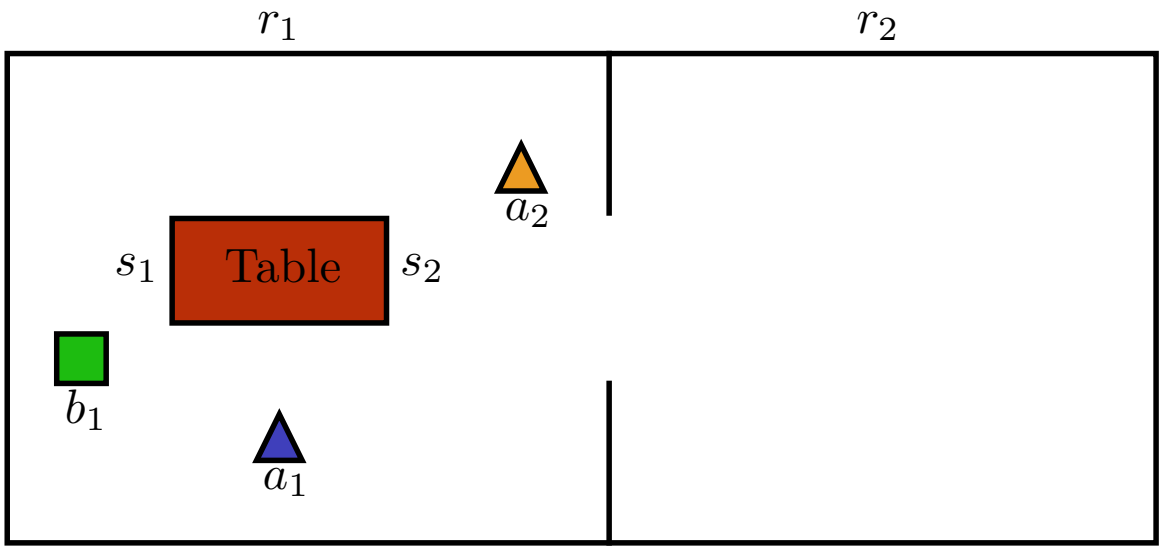
Concurrent Multiagent Planning - Concurrency Constraints

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

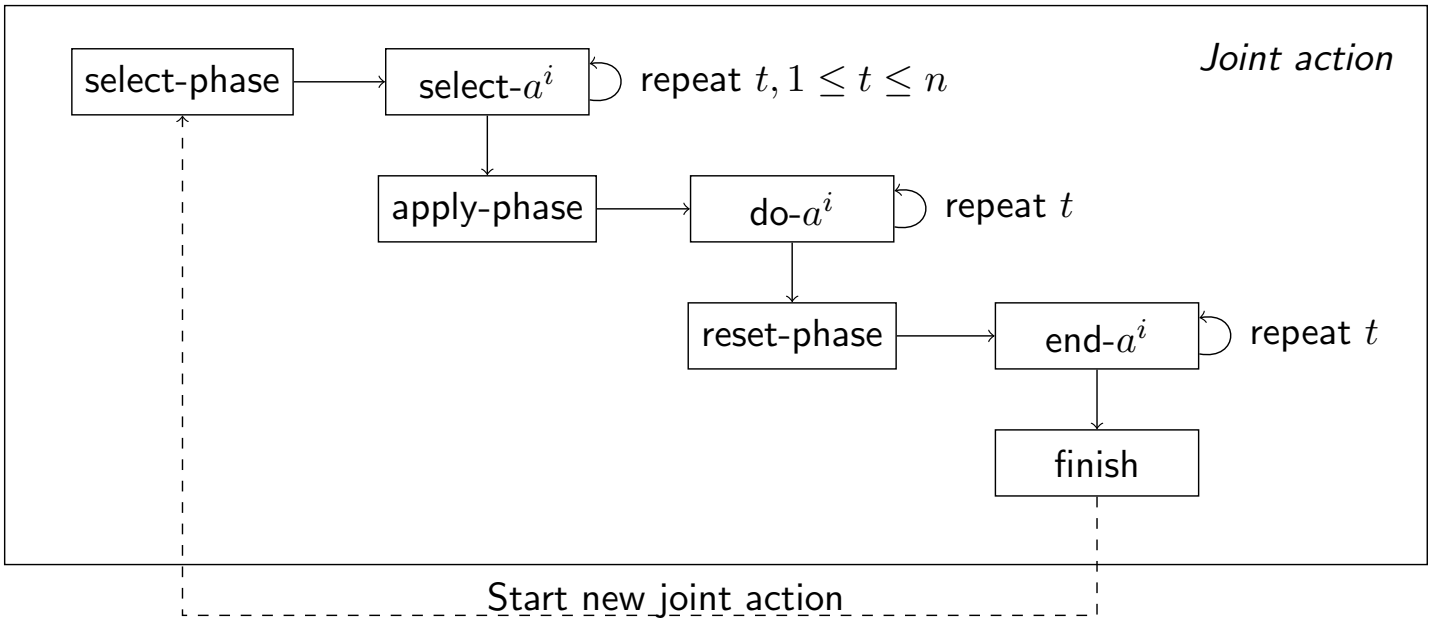
```
(:action lift-side
 :agent ?a - agent
 :parameters (?s - side)
 :precondition (and
   (at-side ?a ?s)
   (down ?s)
   (handempty ?a)
   (forall
    (?a2 - agent ?s2 - side)
    (not(lower-side ?a2 ?s2))
   )
 )
 :effect (and (not (down ?s))
  (up ?s)
  (lifting ?a ?s)
  (not (handempty ?a ?s))
 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

```
...
(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 CMAP $\Pi = \langle N, F, \{A^i\}_{i=1}^n, I, G \rangle$ into a classical planning problem $\Pi' = \langle F', A', I', G' \rangle$.
- Sound and complete transformation:
  - Adds new fluents and actions that allow to select and apply 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:

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

Compilation from Multiagent to Classical Planning (III)

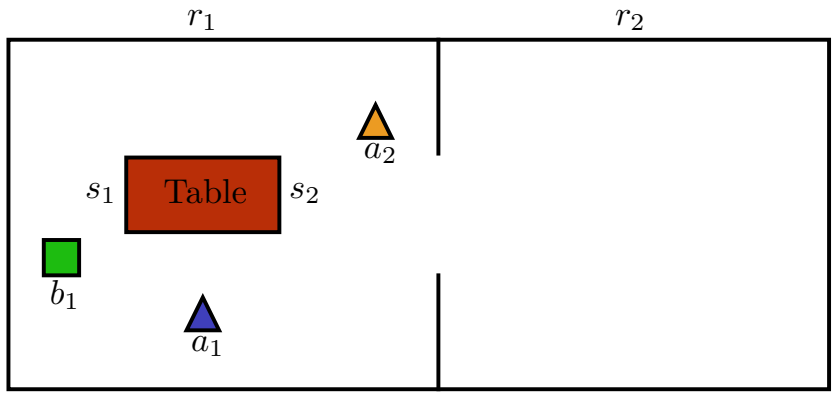
Extension: joint actions with bounded size C.

- 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 two sets of **domains**:

1. CoDMAP-15 domains (do not require concurrency).
2. 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 ≤ 2 ($C = 2$).
- Joint action size ≤ 4 ($C = 4$).

Experiments - CoDMAP-15 domains (I)

- Used to show that our algorithm is complete. However, they **do not require concurrency** (can be sequentially solved).
  - Use Fast-Downward (FD) for comparison.
- They **do not specify negative concurrency constraints** (incompatible actions).
  - Add explicit negative concurrency constraints to avoid invalid plans $\rightarrow$ Complexity to find a plan increases!

Experiments - CoDMAP-15 domains (II)

Domain	#	Coverage				Time (s.)				Plan length				# Actions			
		2	4	∞	FD	2	4	∞	FD	2	4	∞	FD	2	4	∞	FD
Blocksworld	20	7	2	4	**20**	759.5	-	-	**0.2**	**32.1**	-	-	32.8	6848	12323	4110	**1270**
Depot	20	13	10	9	**17**	202.9	246.4	223.9	**58.3**	30.6	15.7	**14.9**	44.0	10100	18176	6061	**2007**
Driverlog	20	18	17	18	**20**	67.3	58.8	73.7	**26.1**	21.1	**20.5**	25.2	35.6	38416	69145	23051	**7386**
Elevators08	20	9	8	10	**20**	13.8	12.5	9.5	**0.2**	31.0	**30.3**	36.4	65.1	10779	19399	6469	**2155**
Logistics00	20	**20**	**20**	**20**	**20**	1.9	2.9	212.1	**0.0**	30.3	**28.0**	30.1	50.2	1781	3202	1070	**318**
Rovers	20	**20**	**20**	19	**20**	45.2	75.2	20.9	**0.1**	46.5	47.4	**42.9**	56.8	18314	32962	10990	**2609**
Satellites	20	19	17	19	**20**	82.8	128.0	32.2	**1.0**	**32.6**	35.5	34.2	55.9	45106	81188	27065	**8122**
Sokoban	20	0	0	0	**18**	-	-	-	**32.3**	-	-	-	**54.1**	3319	5970	1993	**663**
Taxi	20	**20**	**20**	**20**	**20**	1.3	2.6	0.7	**0.0**	14.8	**14.7**	**14.7**	18.7	544	975	328	**108**
Wireless	20	2	2	2	**4**	-	-	-	-	-	-	-	-	15644	28156	9388	**3128**
Woodworking08	20	14	8	4	**20**	290.0	256.0	-	**0.9**	22.4	**11.4**	-	46.1	17406	31327	10445	**3447**
Zenotravel	20	16	16	18	**20**	87.2	125.6	164.8	**1.5**	**23.6**	24.1	34.2	46.9	67586	121652	40553	**13502**
Total	240	158	140	143	**219**												

- Lower coverage and slower than FD.
- Solutions are always shorter (FD does not compress plans).

Experiments - Required Concurrency Domains (I)

- 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.
- 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).

Experiments - Required Concurrency Domains (II)

- 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.
- Workshop - Inventory pallets in a high-security facility.
  - Open door = press switch + turn key.
  - Inventory a pallet = lift pallet + examine pallet.

Experiments - Required Concurrency Domains (III)

Compare our approach with CJR [Crosby et al., 2014]:

- Compilation to classical planning.
- Concurrency constraints in the form of affordances on subsets of objects.
- Limitations:
  - Concurrency constraints are not as expressive $\rightarrow$ *Conditional effects on simultaneous actions are not supported.*
  - Effects are applied immediately for atomic actions $\rightarrow$ *Some joint actions cannot be simulated.*

Experiments - Required Concurrency Domains (IV)

Domain	#	Coverage				Time (s.)				Plan length				# Actions			
		2	4	∞	CJR	2	4	∞	CJR	2	4	∞	CJR	2	4	∞	CJR
MAZE	20	**13**	8	6	11	351.9	435.2	**144.4**	192.8	47.2	22.0	**11.7**	77.3	41723	69368	**27900**	156886
$a = 10$	10	**8**	6	5	7	243.6	564.8	**169.1**	225.5	48.3	25.0	**12.2**	79.6	39909	67417	**26155**	119374
$a = 15$	10	**5**	2	1	4	**525.2**	-	-	-	**45.4**	-	-	-	43989	71807	**30080**	194397
TABLEMOVER	24	**15**	12	**15**	-	**263.3**	336.5	341.0	-	**58.7**	59.0	61.5	-	7487	13127	**4667**	-
$a = 2$	12	10	10	**11**	-	**103.8**	226.4	214.6	-	63.5	**62.0**	64.5	-	3450	6154	**2098**	-
$a = 4$	12	**5**	2	4	-	**558.2**	-	-	-	**49.0**	-	-	-	11524	20100	**7236**	-
WORKSHOP	20	**15**	13	13	6	132.3	298.6	**51.8**	629.0	35.7	37.0	**32.5**	63.5	18002	31000	11502	**5425**
$a = 4$	10	**8**	**8**	**8**	5	42.1	261.6	**36.6**	587.3	**37.3**	43.9	**37.3**	65.8	7772	13621	4847	**2351**
$a = 8$	10	**7**	5	5	1	235.5	357.8	**76.0**	-	33.9	26.0	**24.8**	-	28231	48378	18157	**8499**
BOXPUSHING	69	39	56	**59**	-	**26.8**	79.9	63.9	-	**9.4**	11.0	10.2	-	3075	5360	**1932**	-
$a = 2$	21	**21**	**21**	**21**	-	**14.1**	15.6	16.2	-	10.5	10.6	**10.3**	-	2099	3775	**1261**	-
$a = 3$	48	18	35	**38**	-	**41.5**	118.5	90.2	-	**8.2**	11.2	10.2	-	3502	6054	**2226**	-
$l = 0$	21	**18**	17	**18**	-	**41.5**	64.7	53.3	-	**8.2**	8.3	8.3	-	3373	5887	**2116**	-
$l > 0$	27	-	18	**20**	-	-	169.2	**123.5**	-	-	13.9	**12.0**	-	3602	6184	**2312**	-

- Unbounded compilation (∞) has the highest coverage.
- Compilation $C = 2$ is usually fast but cannot solve problems involving > 2 agents.

Conclusions

- Sound and complete method for compiling CMAPs into classical planning problems.
- The number of resulting actions is polynomial in the description of the CMAP.
- Competitive performance in CoDMAP-15 domains and domains requiring concurrency.

Questions

- Contact:
  - daniel.furelos@upf.edu
  - 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.