Forward Search Temporal Planning with Simultaneous Events

Daniel Furelos-Blanco [1] Anders Jonsson [1]
Héctor Palacios [2] Sergio Jiménez [3]

[1]Universitat Pompeu Fabra

[2]Nuance Communications

[3]Universitat Politècnica de València

June 25, 2018

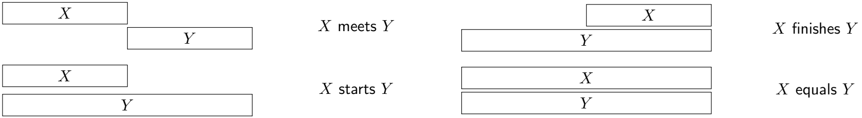
Motivation (I)

Many situations in the real-world involve **simultaneous events** (e.g. relay races).

Current temporal planning algorithms do not support this kind of situations.

Motivation (II)

Allen's Interval Algebra [Jiménez et al., 2015] a domain with required simultaneous events.



PDDL 2.1 induces **temporal gaps** [Rintanen, 2015]:

- State-of-the-art planners using PDDL do not solve problems with simultaneous events.
- Potentially, more decision points.

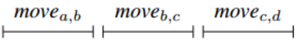


Figure 1: Gaps in Action Schedule

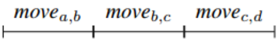


Figure 2: Action Schedule without Gaps

Proposed Approach

Solve temporal planning problems involving simultaneous events using classical planning

- Previous approaches already used classical planning to solve temporal problems [Long and Fox, 2003, Coles et al., 2009, Cooper et al., 2013, Jiménez et al., 2015].
- Our approach:
  1. Compile temporal problem into classical problem.
  2. Solve problem using classical planner maintaining STNs (Simple Temporal Networks) to check temporal consistency.

Classical Planning

A classical planning **problem** is defined as

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

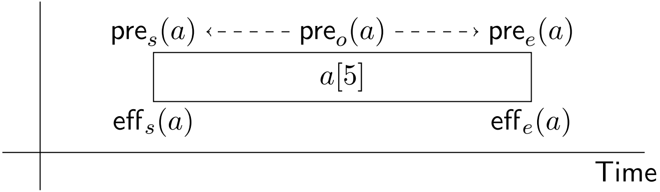
where

- 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 **plan** for P is an action sequence $\pi = \langle a_1, \ldots, a_n \rangle$.

Temporal Planning - Definition

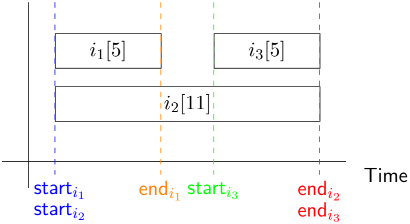
- A temporal planning **problem** is a tuple $P = \langle F, A, I, G \rangle$.
- Actions have the following structure:



- Temporal **plan** $=$ list of (time, action) pairs.
- The quality of a temporal plan is given by its **makespan**.

Temporal Planning - Events

- An action a can be defined in terms of two discrete events: start_a and end_a.
- Two events are **simultaneous** if they occur exactly at the same time.



- Given an individual event e, no effect of e can be mentioned by another event simultaneous with e [Fox and Long, 2003].

Simple Temporal Networks (STNs)

STNs [Dechter et al., 1991] are used to represent temporal constraints on time variables using a directed graph:

- Nodes $=$ time variables τ_i.
- Edges (τ_i, τ_j) with label $c =$ constraints $\tau_j - \tau_i \leq c$.

Possible outcomes:

- If the STN contains negative cycles, scheduling fails.
- Else, τ_i can take values from $[-d_{i0}, d_{0i}]$ where:
  - $d_{ij} =$ shortest distance from τ_i to τ_j.
  - $\tau_0 = 0$ is the reference variable.

Compilation from Temporal to Classical Planning (I)

- STP = **S**imultaneous **T**emporal **P**lanner.
- Extension of the TP planner [Jiménez et al., 2015] to handle simultaneous events:
  1. Add STNs to Fast Downward (FD):
     - STN: checks temporal constraints.
     - FD: manages preconditions and effects.
  2. Impose a bound K on the number of active temporal actions.
  3. Described for problems with static durations and no duration dependent effects.

Compilation from Temporal to Classical Planning (II)

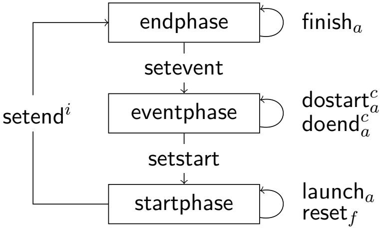
Compilations to classical must ensure that [Coles et al., 2009]:

1. Temporal actions end before reaching the goal.
2. Contexts (pre_o) are not violated.
3. Temporal constraints are preserved.

Compilation from Temporal to Classical Planning (III)

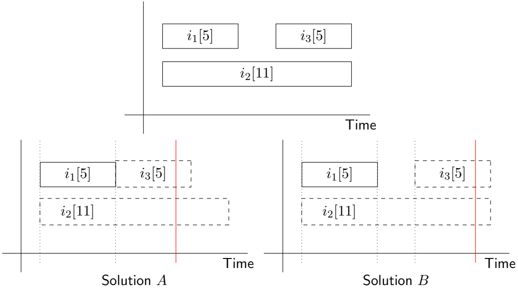
The compilation divides each joint event in 3 phases:

1. End phase: active actions are scheduled to end.
2. Event phase: simultaneous events take place.
3. Start phase: check that pre_o of active actions are satisfied.



Compilation from Temporal to Classical Planning (IV)

- C: cyclic counter $(0, \ldots, C, 0, \ldots)$, counts the number of end phases.
- Motivation: avoid ignoring states that are
  1. propositionally identical, and
  2. temporally different.



Compilation from Temporal to Classical Planning (V)

Modifications applied to Fast Downward [Helmert, 2006]:

- Each search node contains an STN.
- When a successor node is generated:
  1. The STN of its predecessor is copied.
  2. A new edge (τ_i, τ_j) is added to the STN.
  3. Shortest paths are recomputed.

Compilation from Temporal to Classical Planning (VI)

Introduce temporal constraints every time we generate events:

1. For a concurrent event $\{e_1, \ldots, e_k\}$, add constraints

$$\tau_{e_j} \leq \tau_{e_{j+1}}, \tau_k \leq \tau_1$$

to ensure they occur at the same time.

2. For each active action a' that started before and has to end after the concurrent event, add

$$\tau_{e_j} + u \leq \tau_{a'} + d(a').$$

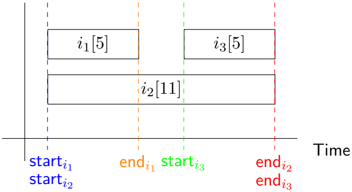
3. For two consecutive concurrent events $\{e_1, \ldots, e_k\}$ and $\{e'_1, \ldots, e'_m\}$, add constraint

$$\tau_{e_k} + u \leq \tau_{e'_1}.$$

$u =$ slack unit of time

Simple Temporal Networks (STNs) - Example (I)

Target scheduling



1. start$_{i_1}$, start$_{i_2}$
2. end$_{i_1}$
3. start$_{i_3}$
4. end$_{i_2}$, end$_{i_3}$

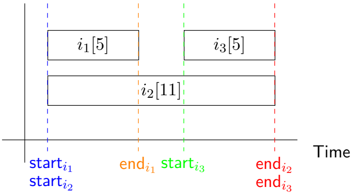
STN constraints

$$\tau_{i_1} < \tau_{i_1} + d(i_1),$$
$$\tau_{i_2} < \tau_{i_1} + d(i_1),$$
$$\tau_{i_1} + d(i_1) < \tau_{i_3},$$
$$\tau_{i_3} < \tau_{i_3} + d(i_3),$$
$$\tau_{i_3} < \tau_{i_2} + d(i_2),$$

$$\tau_{i_1} \leq \tau_{i_2},$$
$$\tau_{i_2} \leq \tau_{i_1},$$
$$\tau_{i_3} + d(i_3) \leq \tau_{i_2} + d(i_2),$$
$$\tau_{i_2} + d(i_2) \leq \tau_{i_3} + d(i_3).$$

Simple Temporal Networks (STNs) - Example (II)

Target scheduling



1. start$_{i_1}$, start$_{i_2}$
2. end$_{i_1}$
3. start$_{i_3}$
4. end$_{i_2}$, end$_{i_3}$

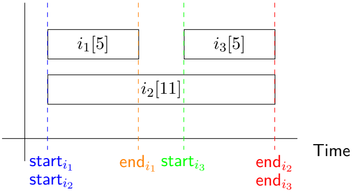
Reformulated STN constraints

~~$\tau_{i_1} - \tau_{i_1} \leq 5 - u,$~~
~~$\tau_{i_2} - \tau_{i_1} \leq 5 - u,$~~
$\tau_{i_1} - \tau_{i_3} \leq -5 - u,$
~~$\tau_{i_3} - \tau_{i_3} \leq 5 - u,$~~
~~$\tau_{i_3} - \tau_{i_2} \leq 11 - u,$~~

$$\tau_{i_1} - \tau_{i_2} \leq 0,$$
$$\tau_{i_2} - \tau_{i_1} \leq 0,$$
$$\tau_{i_3} - \tau_{i_2} \leq 6,$$
$$\tau_{i_2} - \tau_{i_3} \leq -6.$$

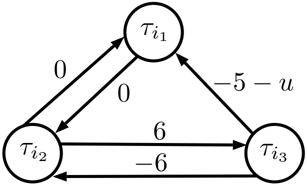
Simple Temporal Networks (STNs) - Example (III)

Target scheduling



1. $\text{start}_{i_1}, \text{start}_{i_2}$
2. end_{i_1}
3. start_{i_3}
4. $\text{end}_{i_2}, \text{end}_{i_3}$

Resulting STN



$$\tau_{i_1} = 0,$$
$$\tau_{i_2} \in [-d_{21}, d_{12}] = [0,0] \rightarrow \tau_{i_2} = 0,$$
$$\tau_{i_3} \in [-d_{31}, d_{13}] = [6,6] \rightarrow \tau_{i_3} = 6.$$

Experiments - Coverage and IPC quality (I)

	TPSHE	TP(3)	TP(4)	STP(3)	STP(4)	POPF2	YAHSP3-MT	ITSAT
AIA[25]	3/3	7.5/9	8.5/10	19.51/24	**23.5**/**25**	10/10	3/3	3/3
CUSHING[20]	0/0	4.07/**20**	4.93/**20**	3.31/14	2.28/5	**20**/**20**	0/0	0/0
DRIVERLOG[20]	**14.78**/**15**	1.08/4	0.91/3	0/0	0/0	0/0	2.31/4	1/1
DLS[20]	9.37/11	7.7/9	8.06/9	3.9/4	3.49/4	7/7	0/0	**16.18**/**19**
FLOORTILE[20]	0/0	0/0	0/0	0/0	0/0	0/0	4.93/5	**19.7**/**20**
MAPANALYSER[20]	**17.38**/**20**	12.34/**20**	12.02/19	10.09/16	7.69/12	0/0	1/1	0/0
MATCHCELLAR[20]	15.72/**20**	15.71/**20**	15.71/**20**	15.71/**20**	15.71/**20**	**20**/**20**	0/0	18.91/19
PARKING[20]	6.73/**20**	5.67/17	5.33/16	1.93/6	1.93/6	12/13	**16.84**/**20**	0.96/6
RTAM[20]	**16**/**16**	2.73/6	2.79/6	0/0	0/0	0/0	0/0	0/0
SATELLITE[20]	**16.63**/18	5.04/13	4.67/12	0/0	0/0	2.92/3	13.82/**20**	1.68/7
STORAGE[20]	4.92/**9**	0/0	0/0	0/0	0/0	0/0	3.91/**9**	**9**/**9**
TMS[20]	0.06/9	0/0	0/0	0/0	0/0	0/0	0/0	**16**/**16**
TURN&OPEN[20]	**15.53**/**19**	5.03/10	5.19/10	0/0	0/0	7.31/8	0/0	5.88/6
Total	**120.12**/**160**	66.87/128	68.11/125	54.45/84	54.61/72	79.22/81	45.8/62	92.3/106

Experiments - Coverage and IPC quality (II)

- STP is top performer at AIA (only domain with simultaneous events).
- Bad performance in domains with simpler forms of concurrency but combinatorially challenging.
- Higher values of K usually improve performance.

Conclusions

- Method that returns sound temporal plans if used in a forward-search planner maintaining STNs.
- Good performance in domain requiring simultaneous events.
- Not competitive in combinatorially challenging domains requiring simpler forms of concurrency.

Future work: Analyze problems before solving them $\rightarrow$ Choose an appropriate solver.

Questions

- Contact:
  - daniel.furelos@upf.edu
  - anders.jonsson@upf.edu
  - hector.palaciosverdes@nuance.com
  - serjice@dsic.upv.es
- Software and domains:
  `https://github.com/aig-upf/temporal-planning`

Coles, A., Fox, M., Halsey, K., Long, D., and Smith, A. (2009).
Managing concurrency in temporal planning using planner-scheduler interaction.
Artif. Intell., 173(1):1–44.

Cooper, M. C., Maris, F., and Régnier, P. (2013).
Managing Temporal Cycles in Planning Problems Requiring Concurrency.
Computational Intelligence, 29(1):111–128.

Dechter, R., Meiri, I., and Pearl, J. (1991).
Temporal Constraint Networks.
Artif. Intell., 49(1-3):61–95.

Fox, M. and Long, D. (2003).
PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains.
J. Artif. Intell. Res. (JAIR), 20:61–124.

Helmert, M. (2006).
The Fast Downward Planning System.
J. Artif. Intell. Res. (JAIR), 26:191–246.

Jiménez, S., Jonsson, A., and Palacios, H. (2015).
Temporal Planning With Required Concurrency Using Classical Planning.
In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, pages 129–137.

Long, D. and Fox, M. (2003).
Exploiting a Graphplan Framework in Temporal Planning.
In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003), June 9-13, 2003, Trento, Italy*, pages 52–61.

Rintanen, J. (2015).
Models of Action Concurrency in Temporal Planning.
In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1659–1665.