A Temporal Approach to Multiagent Planning with Concurrent Actions

A Temporal Approach to Multiagent Planning with Concurrent Actions

Matthew Crosby
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK
m.crosby@ed.ac.uk

Abstract

The ability of temporal planners to find concurrent plans can potentially be exploited in multiagent planning with concurrent actions. However, in recent years temporal planning has not been very prevalent in the multiagent planning literature. This paper introduces a simple multiagent planning domain (with concurrent interacting actions) and shows how it can be efficiently translated to temporal planning. The performance of a temporal planner (POPF2) is then evaluated and shown to be a useful benchmark for future multiagent approaches.

Introduction

While there has been some recent work involving concurrent actions in multiagent planning, the area has not been strongly represented since the turn of the century (Boutilier and Brafman 2001; Brenner 2003). The prevalence of temporal planning in the multiagent planning community has similarly declined. This paper attempts to redress this balance looking at a temporal planning approach to centralised, cooperative, multiagent planning for domains with concurrent actions.

This paper introduces a simple base problem (Vehicles) for modelling multiagent concurrent actions, which can be easily extended to include non-concurrent multiagent interactions. This domain can be represented in PDDL with the addition of concurrency constraints, (similar to (Boutilier and Brafman 2001) except that constraints are attached to objects instead of actions). It is then shown how such problems can be translated into temporal planning without the need for conditional effects and just two actions per concurrent constraint regardless of the number of agents, number of other actions, or type of constraint. POPF2 (Fox and Long 2003), a well-known temporal planner is then tested on the translated problems to determine the applicability of this method in practice.

The Vehicles Domain

In multiagent planning, there are two possible methods by which agents' actions can interact concurrently (which each have a non-concurrent counterpart):

1. Concurrent Coordination: Coordinating simultaneous actions, e.g. when carrying a heavy object together.

2. Concurrent Interference: Accessing a limited resource, e.g. attempting to pass through a narrow doorway.

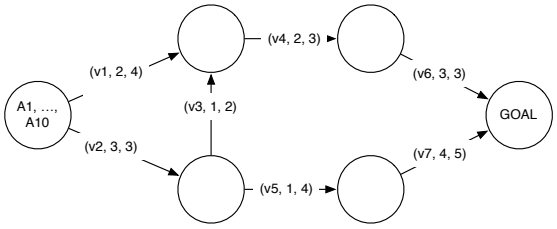

Figure 1: Example Vehicles Problem (solved in 96s POPF2). Ten agents starting on the left must travel to the rightmost location.

The Vehicles domain attempts to model both of these with different types of vehicles. In the domain, there are a number of locations connected by vehicles. Each vehicle has a related concurrency constraint that specifies the minimum and maximum number of agents required to operate the vehicle. For example, a unicycle would have `(unicycle, 1, 1)` while a car may have `(car, 1, 5)`, a tandem `(tandem, 2, 2)` and a boat `(boat, 3, 8)`. These constraints can be added to the problem file of a PDDL specification, one for each object that has a concurrency constraint (the exact details omitted for space reasons). An example problem is shown in Figure 1. In this problem there are ten agents starting at the far left location that must all navigate to the far right location.

Translating to Temporal Planning

Only the specific translation for the Vehicles domain will be shown here with the full algorithm to appear in a full paper. The translation requires that the problem contains an `agent` type with each action containing a single agent parameter and that each action only contains at most one object with a concurrency constraint. In the full version of the translation agents can be split into subtypes and actions containing more than one concurrent object can be dealt with.

For each action, the translation creates two new temporal actions, action–start and action–join (shown for the Vehicles domain in Figure 2). For each object-type `?v`, with a concurrency constraint three new functions must be added to the domain, `(using ?v)`, `(min ?v)`, and `(max ?v)`. The first of these is updated by the corresponding actions to show how many agents are currently using the constrained resource. The latter two are used to ensure that the

```
*(:durative-action move-start
:param (?a ?v ?x ?y) :dur (= ?duration 1)
:con (and (at start (at ?a ?x))
(at start (free ?a))
(at start (vehicle ?v ?x ?y))
(at start (= (using ?t) 0))
(at end (>= (using ?t) (min ?t)))
(at end (<= (using ?t) (max ?t))))
:eff (and (at start (increase (using ?t) 1))
(at start (not (free ?a)))
(at start (not (at ?a ?x)))
(at end (at ?a ?y))
(at end (free ?a))
(at end (assign (using ?t) 0))))
*(:durative-action move-join
:param (?a ?v ?x ?y) :dur (= ?duration 1)
:con (and (at start (at ?a ?x))
(at start (free ?a))
(at start (vehicle ?v ?x ?y))
(at start (> (using ?t) 0)))
:eff (and (at start (increase (using ?t) 1))
(at start (not (free ?a)))
(at start (not (at ?a ?x)))
(at end (at ?a ?y))
(at end (free ?a))))
```

Figure 2: Durative Actions for Moving using a Vehicle (with abbreviations for space reasons).

	Coord (v, 2, 2)		Int (v, 1, 1)		Mix (v, 1, 5)	
Agents	Time	Dur.	Time	Dur.	Time	Dur.
10	0.00	5	0.00	10	0.00	2
20	0.02	10	0.03	20	0.01	4
30	0.08	15	0.08	30	0.02	6
40	0.18	20	0.19	40	0.05	8
50	0.36	25	0.36	50	0.08	10
60	0.62	30	0.62	60	0.15	12
70	1.01	35	0.99	70	0.21	14
80	1.49	40	1.44	80	0.32	16
90	2.13	45	2.08	90	0.45	18
100	2.93	50	2.88	100	0.62	20

Table 1: Results showing how POPF2 performs as the number of agents increases over domains showing coordination, interference and a mix of vehicles.

amount of agents that simultaneously use a constrained resource is between the minimum and maximum specified for that resource. As an example, the problem file for the instance shown in Figure 1 contains the function initialisations `(using v1 0)`, `(min v1 2)`, and `(max v1 4)`.

Evaluation

This section discusses the performance of POPF2 (Coles et al. 2010) on the temporal Vehicles domain as the number of agents, and number and type of vehicles are varied. The domains are run with the metric set to minimize total time and the tables display the time taken to find a plan (in seconds) and the total duration of the joint plan.

Table 1 shows how POPF2 performed as the number of agents increased (up to 100). The problem instance used for these experiments was the simplest possible domain with just two locations and one vehicle. For example, a solution for the ten agent coordination domain involves five separate pairs of two agents using the vehicle. The vehicles in the mix domain were of the form $(v, 1, 5)$ for Table 1 and chosen randomly for Table 2. The results show a surprising similarity between the Coordination and Interference problems, suggesting that the approach deals equally well with the two interference effects. Interestingly, the Mix problems scaled much better with the number of agents suggesting that coordination is not a problem for this approach.

	Coord (v, 4, 6)		Int (v, 1, 3)		Mix (v, 1-4, 1-6)	
Vehic.	Time	Dur.	Time	Dur.	Time	Dur.
1	0.00	2	0.00	5	0.00	4
2	0.01	3	0.01	6	0.83	6
3	0.00	2	0.54	4	4.60	3
4	5.55	2	0.12	4	0.12	4
5	6.15	2	0.13	4	0.19	4
6	6.52	2	0.15	4	8.45	4
7	306	3	5.8	5	95.59	5

Table 2: Results showing POPF2's performance as the number of vehicles increases. The last entry's problem is that from Figure 1.

Table 2 shows POPF2's performance as the domain size increases with the final entry being the problem shown in Figure 1. The number of locations in the domain increases with the amount of vehicles so that there is only ever (at most) one vehicle between any two locations. These results show that the coordination domain, while leading to shorter overall plans, did not scale as well as the problem size increased. The results for the mixed domain are a little more chaotic as the vehicle capacities were chosen randomly leading to some problems being intrinsically harder than others.

Conclusion

It has been shown that temporal planning can be used to solve simple multiagent problem instances with concurrent actions and that multiagent concurrent actions can be translated to two temporal actions regardless of the size of domain or type of concurrency constraint used. An extended paper will explain the original multiagent problem specification and the translation method in full detail and compare multiagent approaches to the benchmark set by POPF2. Future work will try to further integrate temporal planning techniques into multiagent planning.

References

Boutilier, C., and Brafman, R. 2001. Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research* 14:105–136.

Brenner, M. 2003. Multiagent planning with partially ordered temporal plans. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 1513–1514.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10)*.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20.