Multirobot/Multiagent Reading Group Notes
9/23/05
Discussion led by Mary Koes

1 Papers

[1] Jeffrey S. Cox and Edmund H. Durfee. An efficient algorithm for multiagent plan coordination. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 828–835, New York, NY, USA, 2005. ACM Press.

[2] Jeffrey S. Cox, Edmund H. Durfee, and Thomas Bartold. A distributed framework for solving the multiagent plan coordination problem. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 821–827, New York, NY, USA, 2005. ACM Press.

2 Summary

The single agent partial order planning problem has been around for ages (think STRIPS). Weld-1994 has a good description of all this. The planning problem for a single agent is to create a consistent Partial Order Causal Link (POCL) plan [using as few steps as possible].

Single Agent Problem Generate consistent POCL plan	**Multiagent problem** Given a bunch of agents' POCL plans, generate consistent multiagent parallel POCL plan
$P = \langle O, S, \prec_T, \prec_C \rangle$	$P = \langle A, O, S, \prec_T, \prec_C, \#, =, X \rangle$
	A is set of agents
O is set of operators; each operator described by preconditions and postconditions	O is set of operators; each operator described by preconditions, *inconditions*, and postconditions
S is set of plan steps and are instances of O (may be many to steps in S from one operator in O)	same (except operators now have inconditions)
$\prec_T$ is set of ordering constraints: $\langle s_i, s_j \rangle$ means step s_i must be before s_j	same
$\prec_C$ is set of causal links (Tate, 1977): $\langle s_i, s_j, c \rangle$ means step s_i produces c for s_j	same
implicitly includes all steps	$\#$ is set of non-concurrency relations $\langle s_i, s_j \rangle \in \#$: s_i and s_j don't happen at same time
implicitly empty	$=$ is set of concurrency relations $\langle s_i, s_j \rangle \in =$: s_i and s_j happen at same time
	X represents assignments: $\langle s, a \rangle$: agent a is assigned to step s

What does it mean for a plan to be consistent?

There are no possible linearizations that are invalid. How do we know there are no possible invalid linearizations? The plan doesn't have any flaws. What does that mean?

Flaws

Open precondition flaws occur if there's a step with precondition c but no guarantee that this precondition is met (no causal link producing c for this step). To fix this flaw, if there's already a step in the plan that produces c and there is no temporal ordering constraint that says it has to be done later than the step that uses c, we can add a causal link. If there's no step that currently produces c but the operators support such a step, we fix this flaw by adding another step.

Causal link threat flaws occur if step 1 produces c for step 2 (and step 2 needs c) but there's nothing to prevent step 3 from being executed after c is produced but before c is used. The simple fix for this is to add a constraint that step 3 isn't executed between steps 1 and 2 (either it has to be *demoted* to before step 1 or *promoted* to after step 2).

Parallel step threat flaws occur only in the multiagent parallel planning problem if two steps have conflicting postconditions or inconditions (post exclusion principle) and are not constrained to occur at different times. These flaws are resolved by constraining the steps to occur at separate times (adding the pair to #).

Plan merge flaws occur if there is a possible ordering where two steps could be executed but one step would meet all the necessary conditions which creates an inefficiency. These flaws don't have to be fixed but can be resolved according to the process in both papers.

What else is in the first paper, *An Efficient Algorithm for Multiagent Plan Coordination*?

Since they have these plan merge flaws, the first consistent plan may not be optimal so they use branch and bound to find the optimal solution. Somebody else (Yang) already solved the single agent problem (with a few differences) in 1997 so they compare their approach to his approach. They conclude that, for loosely coupled systems, their approach is great and even for tightly coupled systems, their branch and bound algorithm is still better than Yang's dynamic programming algorithm.

What else is in the second paper, *A Distributed Framework for Solving the Multiagent Plan Coordination Problem*?

They have a method for modeling the problem as a constraint optimization problem and apply ADOPT (distributed constraint optimization algorithm by Jay Modi) to the problem. There are all sorts of challenges is modeling the problem so that it can be used by ADOPT. Then the compare two different flaw distribution strategies (who has control over which variables) and find that one is much better than the other, but only works in loosely coupled problems which is why that's what they're focusing on. They also show that if you're willing to make small sacrafices on optimality guarantees, performance improves signicantly.

3 Discussion

- What are the strengths of using a POCL problem formulation? What sorts of problems can it handle?

- Is this work useful to the robotics community? Are there issues that are ignored that should be addressed in robot work (instead of agent work)?

- Could work in the robotics community have been applied to this problem?

- What are some advantages of using the distributed algorithm rather than the centralized algorithm and vice versa?

- How dependent is system performance on problem modeling? Is this a limitation of constraint optimization problem formulations in general? How can we compare different approaches if problem formulation makes a big difference?