Continual Collaborative Planning for Mixed-Initiative Action and Interaction

(Short Paper)

Michael Brenner
Institute for Computer Science
Albert-Ludwigs-University
Freiburg, Germany
brenner@informatik.uni-freiburg.de

ABSTRACT

Multiagent environments are often highly dynamic and only partially observable which makes deliberative action planning computationally hard. In many such environments, however, agents can take a more proactive approach and suspend planning for partial plan execution, especially for active information gathering and interaction with others. This paper presents a new algorithm for Continual Collaborative Planning (CCP) that enables agents to deliberately interleave planning, acting, perception and communication. Our implementation of CCP has been evaluated with MAPSIM, a tool that automatically generates multiagent simulations from formal multiagent planning (MAP) domains. For different such simulations, we show how CCP leads to collaborative planning and acting and, despite minimal linguistic capabilities, to fairly natural dialogues between agents.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Algorithms, Design, Languages

Keywords

Multiagent planning; BDI, speech act theory; Distributed problem solving; Reactive vs deliberative behaviour

1. INTRODUCTION

Multiagent environments are often highly dynamic and only partially observable which makes deliberative action planning computationally hard. In many such environments, however, agents can take a more *proactive* approach and suspend planning for partial plan execution, especially for active information gathering and coordinating with others. By employing such a Distributed Continual Planning (DCP) [5] approach agents can reduce their uncertainty about the present and constrain possible contingencies in the future effectively. This paper introduces a novel DCP algorithm called Continual Collaborative Planning (CCP) that enables agents to reason about and affect their respective (mutual) beliefs, perceptions and goals. CCP models how agents actively gather new information (by active sensing or by communication) and how collaboration arises from *goal revision* and goal adoption *during* the continual planning process.

For evaluating CCP (and other DCP algorithms), we have developed MAPSIM, a multiagent simulation environment in which agents can plan, act and interact. Crucially, MAPSIM creates the simulation *automatically* by analyzing a formal planning domain. Since no domain-specific *programming* is needed MAPSIM can be used to quickly evaluate DCP on a wide range of domains and problems. Since CCP switches between planning, execution and communication as necessary it is particularly suited for domains where communicative and physical actions must be mixed. Additionally, the continual update of the agents' goals during CCP leads to fairly natural mixed-initiative *dialogues* (as exemplified by the MAPSIM output shown in Fig. 1).

(1) Anne: "Please bring me the coffee, R2D2."
(2) R2D2: "Okay."
(3) R2D2: "Where is the coffee, Anne?"
(4) Anne: "The coffee is in the kitchen."
(5) R2D2: "Thanks, Anne."
(6) R2D2: "Please open the kitchen door, Anne."
(7) Anne: "Okay."
(8) Anne opens the kitchen door.
(9) R2D2: "Thanks for opening the kitchen door, Anne."
(10) R2D2 moves to the kitchen.
(11) R2D2 takes the coffee.
(12) R2D2 moves to the living room.
(13) R2D2 brings Anne the coffee.
(14) Anne: "Thanks for bringing me the coffee, R2D2."

Figure 1: Collaborative problem solving and dialogue between artificial agents in MAPSIM (*Household* domain).

2. MULTIAGENT PLANNING FORMALISM

Planning in dynamic MA environments means reasoning about the environment, about (mutual) beliefs, perceptual capabilities and the possible physical and communicative actions of oneself and of others. All of these elements can be modeled in the multiagent planning language MAPL that we

Cite as: Continual Collaborative Planning for Mixed-Initiative Action and Interaction (Short Paper), Michael Brenner, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 1371-1374.

have developed. In this section, we introduce MAPL informally and discuss its suitability for DCP; formal definitions can be found in a supplemental technical report [3].

MAPL is a multiagent variant of PDDL, the de facto standard language for classical planning [7]. One important extension in MAPL is the use of multi-valued state variables (MVSVs) instead of propositions. For example, a state variable *color(ball)* would have exactly one of its possible *domain* values *red, yellow,* or *blue* compared to the three semantically unrelated propositions *(color ball red), (color ball yellow), (color ball blue)*, all or none of which could be true in a given STRIPS state. MVSVs have been used successfully in classical planning in recent years [10], but they also provide distinctive benefits when used for multiagent planning. In particular, we can use MVSVs to model *knowledge* and *ignorance* of agents: If no value is known for a state variable it is *unknown* (contrast this with the closed world assumption of classical planning where what is not known to be true is *false*). This concept can also extended to beliefs about other agents' beliefs and mutual beliefs which are modeled by so-called **belief state variables**.

MAPL **actions** are similar to those of PDDL. In MAPL, every action has a **controlling agent** who will execute the action and, in particular, controls *when* this will happen. Agents are fully autonomous when executing actions, i. e. there is no external synchronization or scheduling component. As a consequence an action will only be executed if, in addition to its preconditions being satisfied, the controlling agent *knows* that they hold. Implicitly, all MAPL actions are extended with such **knowledge preconditions**. Similarly, there are implicit **commitment preconditions**, intuitively describing the fact that another agent will only execute actions if he has agreed to do so.

A MAPL domain can define three different ways to affect the beliefs of agents (necessary, e. g.in order to satisfy knowledge preconditions): Sensing, copresence (joint sensing) and communication. All three are MAPL actions that have knowledge effects. **Sensor models** describe the circumstances in which the current value of a state variable can be perceived. **Copresence models** are multiagent sensor models that induce mutual belief about the perceived state variable. Informally, agents are copresent when they are in a common situation where they do not only perceive the same things but also each other. Individual and joint sensing are important for multiagent systems because they help avoiding *communication*: An agent does not need to ask for what he senses himself, and he does not need to verbalize what he knows to be perceived by the other agents as well. Communicative acts come in two forms: as **declarative statements**, i. e. actions that (similarly to sensory actions) can change the belief state of another agent in specific circumstances, or as **requests** which correspond to *questions* and *commands*. Requests do not have to be modeled explicitly in the MAPL domain, but automatically generated during CCP; this is discussed in the next section.

MAPL **goals** correspond to PDDL goal formulae. However, MAPL has two additional goal-like constructs: **Temporary subgoals** (TSGs) are mandatory, but not necessarily permanent goals, i. e. they must be satisfied by the plan at some point, but may be violated in the final state. **Assertions**, on the other hand, describe *optional* "landmarks", i. e. TSGs that may helpful in achieving specific effects in later phases of the continual planning processes, which cannot be fully planned for yet because of missing information [4, 3]. For example, the MAPL domain used for Fig. 1 contains an assertion which, informally speaking, states that in order to get something one must first know where it is. Assertions are related to HTN methods but do not need an explicitly given decomposition hierarchy.

MAPL plans differ from PDDL plans in being only *partially ordered*. This is inevitable since we assume that there is no central executive which could guarantee a totally ordered execution. We use the term **asynchronous plans** since MAPL plans also allow for *concurrent* occurrence of actions. An asynchronous plan that guarantees that the implied knowledge preconditions will be satisfied during execution (e. g. by explicitly naming the perceptions to be made and speech acts to be used) is called **self-synchronizing plan** because it "explains" how the agents can coordinate their behavior during execution.

3. CONTINUAL COLLABORATIVE PLANNING

Continual Collaborative Planning (CCP) agents switch between planning, partial plan execution, monitoring, plan adaptation and communication. Alg. 1 gives a high-level description of the CCP algorithm. CCP is specified as a Distributed Algorithm, i. e.the current state of the algorithm not only depends on what the agent has been doing, but also on the messages received from others.

Algorithm 1 CCP AGENT(S, G)

$P = \emptyset$
Received no message:
  if S satisfies G do
    return "goal reached"
  else
    $P =$ MONITORINGANDREPLANNING(S, G, P)
  if $P = \emptyset$ then
    return "cannot achieve goal G"
  else
    $(S, P) =$ EXECUTIONANDSTATEESTIMATION(S, P)
Received (tell-val vx) **from** agent a:
  add $v \doteq x$ to S
Received request(e) **from** agent a:
  $sg =$ TRANSLATEREQUESTTOGOAL(e)
  $P =$ MONITORINGANDREPLANNING$(S, G \cup sg, \emptyset)$
  if $P = \emptyset$ then
    send "cannot execute request e" **to** a
  else
    add sg to G as temporary subgoal

We will first discuss the base case where no messages have been sent or received yet by the CCP agent. Roughly speaking, the agent alternates between (re-)planning and acting in this case. The two phases are detailed in Algs. 2 and 3. Alg. 2 shows how a new planning phase is triggered: The agent *monitors* whether his current plan has become invalid due to unexpected events or changes in his goals. If this is the case, the agent adapts its plan by replanning those parts that are no longer executable. In order to exploit the power of state-of-the-art planning systems, Alg. 2 uses an unspecified classical planner PLANNER; cf. the next section for details about the planner used in our implementation.

As soon as a CCP agent has found (or repaired) a valid plan it enters the execution phase, described in Alg. 3. First,

Algorithm 2 MonitoringAndReplanning(S, G, P)

```
if res(S,P) ⊉ G
   RemoveObsoleteSuffixGraph(P)
   P' = Planner(A, res(S,P), G)
   P = concat(P, P')
return P
```

an action on the first level of the plan, i. e. one whose preconditions are satisfied in the current state, is chosen nondeterministically. If the action is controlled by the CCP agent himself, it is executed. If not, the planning agent tries to determine whether the action was executed by its controlling agent. In both cases, the CCP agent will try to update its knowledge about the world state based on the expected effects and the actual perceptions made (Fuse function).

Algorithm 3 ExecutionAndStateEstimation(S, P)

```
e = choose a first-level event from P
if e ='negotiate_plan with agent a'
   r = SelectBestRequest(P, a)
   send request(r) to a
else if agt(e) = self then
   execute(e)
S' = app(S, e)
exp = ExpectedPerceptions(S', A^s)
perc = GetSensorData()
if perc ⊇ exp or exp = ∅ then
   remove e from P
S = Fuse(S', perc)
return (S,P)
```

The most important case for collaboration is the one where the chosen action is *negotiate_plan*. This means that the CCP agent is in a situation where he can communicate with another agent that he intends to collaborate with, i. e. his plan includes at least one action controlled by the other that the latter has not yet committed to. In this case, the CCP agent will send a *request* to the corresponding agent. However, if a plan contains several actions by another agent, i. e. a whole subplan, it is often best not to request execution of the actions individually, but to simply ask for the the final action in the subplan or its end result. In other situations it may even be reasonable to request the achievement of subplans that include more than one agent. CCP does not stipulate a specific implementation of SelectBestRequest. Several variants will be discussed in Section "Analysis".

When an agent receives a request, Alg. 1 enters into a new phase. First the request is translated into a goal formula [2] and tested for achievability. There are several reasons for this: What matters to the other agent is usually not the exact action, but its *result*, i. e. the achievement of a goal or precondition for a subsequent action by the requesting agent. Additionally, when interaction happens in natural language, e. g. in HRI, requests use *referring expressions* (e. g. "clean the dirty plates") which are easier to model as goal constraints than as actions [2]. Accepted requests are adopted as *temporary subgoals (TSGs)*. This means that they must only be achieved temporarily and do not have to hold any more when the agent's main goal is achieved.

The adoption of requests as TSGs is a crucial element of CCP that, to the best of our knowledge, has not been described in other (Distributed) Continual Planning approaches: In addition to repeatedly revising their beliefs about the world, CCP agents also perform continual *goal revision*. In the simplest case, this leads to short information-seeking *subdialogues*, as in lines 3–5 of Fig. 1. But newly adopted TSGs also explain why agents engage in jointly solving more complex *subproblems* that mix physical and communicative actions (as in lines 6–9 of the same example).

```
(1)  Anne: request R2D2 'give R2D2 coffee Anne'.
(2)  R2D2: accept_request 'give R2D2 coffee Anne'.
(3)  R2D2: request Anne 'tell_val Anne R2D2 pos(coffee)'.
(4)  Anne: execute 'tell_val Anne R2D2 pos(coffee)'.
(5)  R2D2: ack_achieved 'tell_val Anne R2D2 pos(coffee)'.
(6)  ...
```

Figure 2: The MAPSIM run of Fig. 1 without NL verbalization.

4. MAPSIM

Continual Planning approaches can only be tested in environments where agents can actually execute, monitor and revise their plans. *Distributed* Continual Planning agents additionally must be able to interact. To this end we have developed MAPSIM, a software environment that automatically generates multiagent simulations from MAPL domains. In other words, MAPSIM interprets the planning domain as an *executable model* of the environment. Thus, MAPSIM allows designers of DCP algorithms to evaluate their approaches on various domains with minimal effort.

MAPSIM parsed and analyzes a MAPL domain description and turns it into perception, action, and communication models for CCP agents. During the simulation, MAPSIM maintains and updates the global world state and it uses the sensor models to compute individual and joint perceptions of agents. The agents interact with the simulation by sending *commands* in the form of plain MAPL actions. The simulator then executes the action, i. e. it checks the preconditions and applies effects as specified in the MAPL domain. If the controlling agent of a command is not identical to the agent who sent it to the simulator this is interpreted as a *request* which, of course, is not directly executed but passed on to the corresponding agent. MAPSIM also accepts some specific commands for acknowledging subgoal acceptance and subgoal achievement.

Agents do not need to know anything about how their actions are executed. Thus, they can implement arbitrary deliberative or reactive methods to determine their behavior and their reactions to requests. We believe that this can make MAPSIM a valuable evaluation tool even when the DCP algorithms investigated differ significantly from CCP.

MAPSIM, as well as the CCP agents described in this paper, are implemented in Python. The generic planner currently used for CCP is a slightly modified version of Axioms-FF [13]. To enable the use of a classical PDDL planner like FF, the MAPL domain is *compiled* to PDDL (details of the compilation process will be given in a future publication). Due to using a state-of-the-art planning system as a subsolver, MAPSIM can generate multiagent plans very quickly. For example, during the MAPSIM run of Fig. 1 CCP called the Planner function 13 times with a total planning time of less than half a second on a 1.6 GHz AMD Athlon.

CCP was designed to be applicable in environments where

artificial agents must collaborate with *humans*. In order to investigate the necessary natural-language interactions MAPSIM includes a verbalization module called the *reporter* agent. The reporter observes all events in the simulation and verbalizes them using a simple template engine that extracts most nouns, verbs and adjectives from the MAPL domain *automatically* (but can easily be extended with domain-specific verbalization patterns). Fig. 1 is a direct, unaltered output of the reporter. Fig. 2 shows the beginning of the same run with reporting turned off.

5. RELATED WORK

This work integrates ideas from several subfields of AI, in particular Classical and Distributed Planning, Multiagent Systems, Epistemic Logic, and Dialogue Systems. We can only discuss some prototypical related work here.

Continual Planning is often implemented by simply switching repeatedly between planning and execution. Previous work that more tightly integrates planning, monitoring, execution and information gathering includes [6, 11, 8]. Our work introduces the concept of assertions and extends CP to the multiagent case, as proposed by desJardins and colleagues [5]. Most work within this field relies on the use of *hierarchical* action and plan representations. In contrast, CCP could be said to decompose planning problems *over time* by postponing and "outsourcing" subproblems to other agents. The two approaches are not mutually exclusive and CCP could be adapted easily to hierarchical planning.

The explicit inclusion of beliefs and mutual beliefs in our planning approach follows BDI models of multiagent planning, e. g. the SharedPlans model of Grosz and Kraus [9] that describes the role of (mutual) beliefs as necessary conditions for planful MA behaviour. By explictly modeling perception and copresence, our approach complements such approaches to MA plans, since it can explain how knowledge conditions for joint behaviour can be achieved during the CCP process.

Our CCP approach is also close in spirit to frameworks for collaborative *dialogue* [12, 1]. Due to our use of a general-purpose planning method, these approaches can deal with more elaborate linguistic phenomena. However, CCP provides a natural explanation for the *causal* reasoning occuring during *situated* dialogue planning, because the generated dialogues directly depend on the agent's knowledge about the current situation and its physical actions in the world.

6. CONCLUSION AND FUTURE WORK

We have presented a novel algorithm for Distributed Continual Planning called Continual Collaborative Planning (CCP). It extends the idea of planning for active information gathering to planning for communication, and shows how collaborative behaviour emerges from negotiation about the joint adoption of temporary subgoals. The approach has been evaluated in MAPSIM, a new software tool that automatically generates MA simulations from formal multiagent planning domains.

An important practical benefit of CCP is its firm grounding in classical AI planning research. Not only did this allow us to exploit the power of a state-of-the-art planner in our implementation, it also facilitates comparison with and adaption of classical planning techniques to multiagent planning. Our simulation tool MAPSIM allows for the simple evaluation of MAP approaches on a wide number of planning domains. Indeed, MAPSIM substantially facilitates the *design* of such domains, because they can be immediately "run" and tested. We are currently adapting existing PDDL domains for MAPSIM and will use them in an empirical study of CCP on a wide range of such benchmarks.

In CCP communication is regarded as a special case of standard action planning, i. e. communication arises whenever it is causally relevant for achieving the goals of an agent. In linguistic terms, communication planning happens on the *pragmatic* level, i. e. a communicative action describes how to achieve communicative intentions. As a result, the Collaborative Continual Planning algorithm presented is able to integrate action and communication planning seamlessly. This opens up many possibilities for future research on *dialogue* as a form of Collaborative Continual Planning.

7. ACKNOWLEDGMENTS

This work has been supported by the EU in the Integrated Project "CoSy" (FP6-004250).

8. REFERENCES

[1] N. Blaylock, J. Allen, and G. Ferguson. Managing communicative intentions with collaborative problem solving. In *Current and New Directions in Dialogue*. Kluwer, 2003.

[2] M. Brenner. Situation-aware interpretation, planning and execution of user commands by autonomous robots. In *Proceedings of IEEE RO-MAN 2007*, 2007.

[3] M. Brenner. The multiagent planning language MAPL. Technical report, Albert-Ludwigs-Universität, Institut für Informatik, Freiburg, Germany, 2008.

[4] M. Brenner and B. Nebel. Continual planning and acting in dynamic multiagent environments. In *Proc. PCAR-06*, 2006.

[5] M. DesJardins, E. Durfee, J. C. Ortiz, and M. Wolverton. A survey of research in distributed, continual planning. *The AI Magazine*, 1999.

[6] O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson. An approach to planning with incomplete information. In *Proc. KR-92*, 1992.

[7] M. Fox and D. Long. PDDL 2.1: an extension to PDDL for expressing temporal planning domains. *JAIR*, 2003.

[8] K. Golden. Leap before you look: Information gathering in the PUCCINI planner. In *Proc. AIPS-98*, 1998.

[9] B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86, 1996.

[10] M. Helmert. The Fast Downward planning system. *JAIR*, 26:191–246, 2006.

[11] C. A. Knoblock. Planning, executing, sensing, and replanning for information gathering. In *Proc. IJCAI-95*, 1995.

[12] K. E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 1998.

[13] S. Thiebaux, J. Hoffmann, and B. Nebel. In defense of axioms in PDDL. In *Proc. IJCAI*, 2003.

[14] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: a review. *Autonomous Agents and Multi-Agent Systems*, 3(2), 2000.