Multi-agent Planning with Confidentiality

(Extended Abstract)

Jiefei Ma, Alessandra Russo, Krysia Broda, Emil Lupu
Department of Computing, Imperial College London
{jm103,ar3,kb,ecl1}@doc.ic.ac.uk

1. INTRODUCTION

Distributed planning has seen a wide range of techniques aiming to address problems such as distributed plan formation and distributed plan execution. Common to these techniques is the notion of *Multi-Agent Planning* (MAP) as a combination of *planning* and *coordination* by de Weerdt et al. [2]. Among the three types of MAP problems summarised by Durfee [3], *distributed planning for distributed plans* is considered to be the most challenging. The key issue to tackle is the *plan merging*, which concerns about resolving potential conflicts (*negative interactions*) between individual plans (sequences of actions) of the agents. A more complex version of the problem that has many real world examples, is that where agents need to maintain confidential information such as private knowledge, and/or do not wish to expose their individual plans. Such confidentiality constraints may make the central/direct analysis of action interactions impossible. This paper focuses on the task of *distributed planning for distributed plans with confidentiality*.

Abductive Reasoning (AR) is a special type of reasoning technique that can generate hypotheses to explain given observations. When combined with *Event Calculus* [6] (EC), it can be used for planning [8] where actions and goals are viewed as hypotheses and observations respectively. This approach is particularly suitable for non-classical planning problems involving, for instance, explicit time (e.g. actions with duration), concurrent actions or extended goals [5]. The inherent logical semantic of the framework allows properties of MAP such as the soundness to be proven. A distributed version of AR, called DARE [7], can be used as a unified framework for distributed planning. Its direct use would see plan construction, goal refinement and allocation handled by the *Global Abductive Phase*, and negative interactions analysis and resolution by the *Global Consistency Phase*. During the planning process, the plan (actions as hypotheses) would be collected and passed around the agents. However, by doing so confidentiality may not be preserved.

This paper shows how DARE can be extended to allow agents to collaboratively construct consistent sub-plans for a given global goal without exposing them. The idea is that during the local plan construction, each agent collects the minimal pre-conditions of its partial plan as constraints and propagates them, so that other agents can construct their partial plans whilst maintaining the constraints. Thus, the partial plans can be kept confidential by each agent locally, whilst still guaranteeing their consistency. Although the pre-conditions of private plans are shared during the planning process, their "owners" are not disclosed.

2. DISTRIBUTED PLANNING FRAMEWORK

Our formalism of the MAP problem is logic based. EC is used to model the domain: (1) $holds(F, T)$ represents that a *fluent* F is true at time T. A fluent is some domain property whose value may change over time as consequence of a performed action; (2) $happens(E, T)$ represents that an action E (to be) performed at T; (3) $static(F)$ represents that a property F holds unconditionally over time; (4) $prec(T1, T2)$ represents that time $T1$ precedes $T2$; (5) $initially(F)$ represents that the fluent F holds at the beginning; (6) action effect conditions for an action E at time T over a fluent F are represented as clauses of the form $[initiates/terminates](E, F, T) \leftarrow L_1, \ldots, L_n$ where each L is a (positive or *negative-as-failure* [1]) literal with predicates in (1)-(4); (7) The domain independent EC axioms (Ψ) capture the relationship between actions and fluents over time:

$$holds(F, T) \leftarrow initially(F), \textbf{not } clipped(F, 0, T).$$
$$holds(F, T) \leftarrow happens(E, T1), initiates(E, F, T1),$$
$$prec(T1, T), \textbf{not } clipped(F, T1, T).$$
$$clipped(F, T1, T2) \leftarrow happens(E, T), prec(T1, T),$$
$$prec(T, T2), terminates(E, F, T).$$

In our planning problem, it is assumed that each agent has a set Γ of action effect conditions and a set Λ of local knowledge expressed in terms of *initially* and *static* literals. Action conditions could be modelled and handled as integrity constraints, but in this paper they are not considered. Γ is confidential to each agent. The set of actions and the set of fluents appearing in the heads of clauses in Γ are denoted as $actions(\Gamma)$ and $achievable(\Gamma)$ respectively, and they represent the agent's capability. Each agent may *advertise* a subset of $achievable(\Gamma)$ as the sub-goals it would accept during the collaborative planning process. Λ may be shared between agents. The planning problem can be modelled as a *distributed planning framework* $\Omega = \langle \Sigma, \Gamma, \Lambda \rangle$, where (a) Σ is a set of agent identifiers, (b) $\Gamma = \bigwedge_{i \in \Sigma} \Gamma_i$ and $actions(\Gamma_i) \cap actions(\Gamma_j) = \emptyset$ for any $i \neq j \in \Sigma$, (c) $\Lambda = \bigwedge_{i \in \Sigma} \Lambda_i$ and Λ is consistent.

During the collaborative planning process, each agent i maintains its *local partial plan* $\Delta_i = \langle id, \delta_i, \tau_i \rangle$, where id is the plan ID, δ_i is a set of actions of $happens(\alpha, t)$ and $\alpha \in$

$actions(\Gamma_i)$, and τ_i is a set of temporal orders $prec(t1, t2)$. To maintain the confidentiality of the plan, δ_i should not be exposed to others during or after the planning process. Given a distributed planning framework $\Omega = \langle \Sigma, \Gamma, \Lambda \rangle$, and a list of $holds(\beta, t)$ or **not** $holds(\beta, t)$ as the goal (state) $\mathcal{G}$, a *global total plan* is $\Delta = \langle \delta, \tau \rangle$, where (a) $\delta = \bigwedge_{i \in \Sigma} \delta_i$ and $\tau = \bigcup_{i \in \Sigma} \tau_i$; (b) $\Psi \wedge \Gamma \wedge \Lambda \wedge \delta \wedge \tau \models \mathcal{G}$ under the Stable Model [4] semantics; (c) $\Psi \wedge \Gamma \wedge \Lambda \wedge \delta \wedge \tau$ is consistent.

3. DISTRIBUTED ABDUCTIVE PLANNING

Given a distributed planning framework and a goal, our distributed abductive algorithm computes a global total plan (GP) as a set of local partial plans (LPs), which is consistent and distributed among the agents.

The planning process is a sequence of states $\langle \mathcal{G}, GPS \rangle$, where $\mathcal{G}$ is a goal, and GPS is a *global plan state*. A GPS is a tuple $\langle \mathcal{C}, \mathcal{T}, \mathcal{N} \rangle$, where (a) $\mathcal{C}$, called a *cluster*, is a set of pairs of $agent\_id$ and LP_id. The cluster records where the global plan "segments" are distributed; (b) $\mathcal{T}$ contains a set of all temporal orders collected and agreed by all the agents' LPs; (c) $\mathcal{N}$ is set of all the negative constraints (i.e. **not** $clipped(f, t1, t2)$ and **not** $holds(f, t)$ literals) collected so far by the agents. The information stored in a GPS is not confidential as it is shared between the agents. Furthermore, $\mathcal{T}$ and $\mathcal{N}$ together represent the "integrity constraints" of the total global plan (GP) – the LPs can be merged according to $\mathcal{T}$ and can be expanded while satisfying $\mathcal{N}$.

A successful execution of the algorithm begins with $GPS = \langle \emptyset, \emptyset, \emptyset \rangle$ and $\mathcal{G}$ as the top level goal, and finishes with $\mathcal{G} = \emptyset$ and the final GPS from which the GP can be extracted – the merging of LPs marked by $\mathcal{C}$ according to $\mathcal{T}$. The algorithm interleaves three derivations: (1) *global abductive derivation* (GAD), (2) *global consistency derivation* (GCD) and *local consistency derivation* (LCD). Upon receiving a goal, an agent starts with GAD. During GAD, the agent can reduce the goal to a set of new sub-goals by adding suitable private actions to the LP. The agent then either continues to solve the new sub-goals itself, or asks other agents to solve them by sending the sub-goals with the latest GPS. Temporal orders are collected and checked. Whenever the LP of the agent is expanded (with a new action), or a negative constraint of the LP is collected (i.e. into the GPS), a GCD will be initiated, which forces every agent in the cluster in turn to check all the constraints collected so far (may be from other agents) using the LCD. During the LCD, an agent may impose additional temporal ordering of the actions in its LP to satisfy the constraints (and hence resolve potential conflicts between its plan and those of other agents). The GCD restarts if the GPS changes after the LCDs; otherwise, the updated GPS is returned to the initiator of the GCD so it can resume the GAD.

Consider a simple example where three agents A,B,C are required to draw a cat and a canary on a dry board. Each agent's knowledge (Γ and Λ) is:

A	$initiates(draw, hasCanary, T) \leftarrow holds(dry, T)$. $terminates(draw, dry, T)$.
B	$initiates(spray, hasCat, T) \leftarrow holds(dry, T)$. $terminates(spray, dry, T)$.
C	$initially(dry)$. $initiates(blow, dry, T)$.

During the collaboration for the goal $\{holds(hasCanary, t), holds(hasCat, t)\}$, A first produced the LP of $\{happens(draw, t1)\}$ for the first sub-goal, and collected temporal and negative constraints such as $prec(t1, t)$ and **not** $clipped(dry, 0, t1)$.

B then tried to abduced $happens(spray, t2)$ for the second sub-goal such that $prec(t2, t)$. However, due to the constraints collected by A, $prec(t1, t2)$ was also collected and the new sub-goal $holds(dry, t2)$ was added. This then caused C to help by abducing $happens(blow, t3)$ such that $prec(t1, t3)$ and $prec(t3, t2)$. Hence, the final GP was: A draws at $t1$, C blows at $t3$, B sprays at $t2$ where $0 < t1 < t3 < t2 < t$. But the LP of each agent was not disclosed to others.

The GP computed is a virtual plan whose segments are distributed. The execution of GP can be as follows: (1) the time points appearing in $\mathcal{T}$ (shared by all agents) are sorted chronologically; (2) the time points are broadcasted in order; (3) the agent executes its actions at the time their associated time points are received.

4. CONCLUSION AND FUTURE WORK

We outlined a distributed planning framework and an algorithm that allows agents to compute confidential and consistent collaborative plans for a common goal. The distributed plans are *confidential* because their actions are known solely to their owners; they are guaranteed to be *consistent* because constraints are propagated and checked during plan constructions; and they are *collaborative* because omitting any of them the goal won't be achievable any more. These properties also make the approach suitable for multiagent hierarchical planning, where agents' local plans are not shared but must be consistent.

Although based on DARE, this work is still preliminary. In a future presentation, we will give a detailed description of the algorithm, present theorems and proofs for the soundness and completeness, investigate the complexity and give testing results.

5. ACKNOWLEDGEMENTS

6. REFERENCES

[1] K. Clark. Negation as failure. *Logic and Data Bases*, pages 293–322, 1978.

[2] M. de Weerdt, A. ter Mors, and C. Witteveen. Multi-agent planning: An introduction to planning and coordination. In *Handouts of the European Agent Summer School*, pages 1–32, 2005.

[3] E. H. Durfee. Distributed problem solving and planning. pages 118–149, 2001.

[4] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of the 5th Int. Conf. on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press.

[5] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufman, San Francisco, CA, 2004.

[6] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Gen. Comput.*, 4(1):67–95, 1986.

[7] J. Ma, A. Russo, K. Broda, and K. Clark. DARE: a system for distributed abductive reasoning. *AAMAS*, 16(3):271–297, 2008.

[8] M. Shanahan. An abductive event calculus planner. *Journal of Logic Programming*, 44(1-3):207–240, 2000.