Plan coordination

– extended abstract –

Mathijs de Weerdt
Delft University of Technology
PO Box 5031, 2600 GA Delft
The Netherlands
M.M.deWeerdt@cs.tudelft.nl

Abstract

Autonomous agents usually plan their actions. Sometimes agents can benefit from cooperation, and sometimes they cannot even do without resources from another. We can either coordinate the plans of agents after they have constructed their plans (plan merging), or we can *plan* the coordination of agents. *Plan merging* can only be used for agents that are able to (first) create a valid plan on their own.

We distinguish two ways to plan coordination. With *service-based* plan coordination agents *offer* resources they can produce for others. With *task-based* coordination agents are allowed to place *requests* for resources thay need. These forms of coordination can be integrated in the plans, and be realized by adapting existing planning algorithms at a few points. All methods use a model of the state of the world and of actions where a proposition is extended to describe all properties of an available entity relevant for planning. Such a proposition is called a *resource*, and can be exchanged among plans more conveniently than propositions as used by, e.g., STRIPS. *Actions* consume all resources that are given as a precondition, and produce the resources defined by the post-condition.

Introduction

When autonomous agents coordinate their actions, their combined potential increases significantly. An agent that plans its *actions* usually is far more efficient than a reactive agent. Similarly, we expect that agents that plan their *interactions* are more efficient as well. We would like to have a method to plan the interactions among agents and combine this with the use existing single-agent planning methods. Moreover, in this approach, (i) agents should be able to decide themselves *when* they cooperate and *which information* they share, (ii) agents should be able to *request* services *from* other agents and include the results in their plans, and (iii) agents should be able to *offer* services *to* other agents and, upon a request, add these to their plans. Such a method is called a *multi-agent planning* method (Georgeff 1984; Mali & Kambhampati 1999).

An interaction between two plans occurs when the effect of one agent's action is used by the other. Usually, the effect of an actions is described by a set of propositions. We propose to use a proposition for each physical object that the agents can exchange. This proposition denotes that this object is *available* and describes all attributes of such an object. We call such a complex proposition a *resource*. First, we give a more formal definition of a resource and we define actions as processes that consume and produce these resources. Then, we describe two ways to plan the coordination of agents: *service-based coordination* where an agent can *offer* to produce certain resources, and a method where agents publish *requests* for specific resources, i.e., *task-based coordination*. A special form of task-based coordination is *plan merging*. In this approach, agents first construct their plans and then request replacements for resources in their plans to be able to remove some actions.

Resources, actions and plans

The *Action Resource Framework*, abbreviated ARF, is based upon work by Tonino et al. (Tonino *et al.* 2002; de Weerdt *et al.* 2003). The ARF distinguishes two basic notions in planning: *resource (facts)* and *actions*. Goals and plans are derived notions that are defined using resources and actions.

A *resource* is an object that is relevant to an agent with respect to the planning problem at hand. Such a resource is either a physical object such as a truck or a block, or an abstract conceptual notion such as the right to do something. To describe the connection between a resource and the (current) state of the world it is in, we use the notion of a *resource fact*. A resource fact specifies the state of a given resource. Syntactically, a resource fact is denoted by a *predicate name* together with a complete specification of all its *attributes*. The predicate name serves to indicate the *type* of resource mentioned in the fact (e.g., a carrier cycle or a taxi). To uniquely identify resource facts, a special attribute *identity* is used to distinguish it from other resources having the same type and possibly the same values of their attributes. Because of the special nature of this identifier, we denote a resource of type t with identifier i and attributes $a_1, \ldots, a_n$ of sorts $s_1, \ldots . s_n$ respectively, as $t_i(a_1 : s_1, \ldots, a_n : s_n)$

When the values of all attributes of a resource fact are ground, i.e., they are constant, we call this a *ground resource fact*. However, attributes may also be variables or functions. In this case, a resource fact describes a *set* of ground resource facts (instances) of the same resource type.

Goals can be efficiently specified by such general resources. Usually, a set of goals G is specified by a set of resources $G = \{g_1, \ldots, g_n\}$. We say that a set of goals G is *satisfied* by a given set of resources R, abbreviated by

$R \models G$, if there exists a ground substitution θ such that $G\theta \subseteq R$, i.e., there is a set of ground instances of the goals that is provided by the resources in R. Two resources r_1 and r_2 are called *compatible*, denoted by $r_1 \equiv r_2$, when they are equal except for the value of their identity attribute.

The state of the world (as far as it is relevant to a planner) is modeled by the set of resource facts that are true at a certain time point. Possible transitions from one state to another are described by *actions*. An action is a basic process that consumes and produces resources. An action o has a set of input resources $in(o)$ that it consumes, and a set of output resources $out(o)$ that it produces. Furthermore, an action may contain a specification of some variables occurring in the set of output resources as *parameters* $param(o)$. To ensure that output resources are uniquely defined, these resources may only contain variables that already occur in the input resources or in the set of the parameters.

An action o can be applied to a set of (ground) resources R if a ground substitution θ exists such that $in(o)\theta \subseteq R$. Application of this action to R results in consuming the set $in(o)\theta$ of input resources while producing the set $out(o)\theta$. The result of o applied to R (under θ) therefore is a resource transformation: starting with R, the set $R \setminus in(o)\theta \cup out(o)\theta$ is produced.

We define plans over a set of actions O as structured objects composed of actions in O. First of all, we have to specify how actions are interrelated. To this end, we use the notion of a *dependency function*. Plans are composed of partially ordered actions. A dependency function d specifies an immediate dependency of input resources of an action on output resources of another action, so d can only specify a valid dependency if (i) the resources involved are compatible and (ii) d generates a partial order between the actions.

The first requirement is met if there exists a substitution θ such that for two resources r and r', $d(r) = r'$ implies $r\theta \equiv r'\theta$, that is θ is a *unifier* for every pair of resources $(r, d(r))$. In particular, we are looking at a *most general unifier* (mgu) θ with this property.

The second condition requires that there are no loops in the dependency relation between actions generated by d: we say that o directly depends on o', abbreviated as $o' \ll_d o$, if resources $r \in in(o)$ and $r' \in out(o')$ exist such that $d(r) = r'$. Let $<_d = \ll^+$ be the transitive closure of $\ll_d$. Then the second condition simply requires $<_d$ to be a (strict) partial order on O.

A *plan* is a triple (O, d, θ) where O is the set of actions in the plan, d is a partial order on the actions in the plan, and θ is a mgu, such that the requirements above are met.

Given a plan $P = (O, d, \theta)$, the set of input resources of a plan, denoted by $In(P)$, is the set of resources $\{r\theta \mid d(r) = \bot\}$ not depending on other resources in the plan. The set of output resources denoted by $Out(P)$ is the set $\{r\theta \mid d^{-1}(r) = \bot\}$ of resources that are not consumed by actions in the plan.

A *problem* consists of a description of the initial state, the goal state, and the set of actions that may be used in a plan. The goal and the initial state can both be described by sets of resources. Resources that are not used as a goal or as an input of another actions are called *free resources*. These free resources of an agent play an important role in plan coordination methods, because they may be used by other agents.

Plan coordination methods

Service-based plan coordination

Service-based coordination is based on the idea that an agent can use the capabilities of other agents. To implement service-based coordination we need to add the following to an existing planning algorithm.

- The problem should be modeled in terms of resources.
- Agents need to determine what services they are able to offer. This can be simply all resources they produce in their plan and do not need themselves (free resources), or it can be a fixed set of resources they are always able to produce. Furthermore, an agent may also try to determine resources they are able to produce using an additional planning algorithm.
- We need a structure to distribute the offers of services efficiently, for example by using an organized blackboard (Corkill 1991) or an auctioneer (Wellman *et al.* 2001).
- Agents need to be able to use such offers in their plans. In the planning process, choices are made to reduce the set of potential solutions (plans), according to the theory of refinement planning (Kambhampati 1997). The most common of these choices is to add an action to a partial plan. In a multi-agent system, agents can also include the effect of services, i.e., one or more resources, offered by other agents. Such an effect of a service can be included in the plan of an agent as a special action.

An alternative to exchanging services is to let an agent inquire whether other agents can provide the resources it needs, without knowing which agent is able to provide these. This is called *task-based* coordination.

Task-based plan coordination

Another solution to planning problems in a multi-agent environment is when agents are able to request missing resources from others without knowing beforehand which other agents are able to provide them these resources.

In this case, the contract net protocol (Davis & Smith 1983) can be used to coordinate the communication between requesters (managers) and providers (contractors). This protocol uses three types of messages.

- TASK(I_1, G_1) is sent by the *requester* agent to announce a task to get from a state I_1 to a state G_1. The agent that sends this message is called a *provider* (for this specific contract). Usually, such an announcement is sent to many other agents.
- BID(I_2, P_2, G_2, c) contains information about in what way an agent (a_2) can deal with (part of) the task: an alternative initial state I_2, an alternative goal G_2, for which holds that $G_2 \cap G_1 \neq \emptyset$, and a plan P_2 to attain this goal from this initial state. This message can also include the

costs the requester has to pay if it awards this agent a_2 the task (I_2, G_2). In multi-agent systems where agents do not trust each other, the plan P_2 may be omitted.

- AWARD(I_3, P_3, G_3) is sent only to an agent from which the requester previously has received a BID(I_2, P_2, G_2, c). This means that the requester is prepared to pay the costs c for a subplan P_3 of P_2.

When a requester has received enough bids (I_2, P_2, G_2) to attain its goals, it needs to find a selection of these bids, and a series of operators such that the combination is an adequate plan. One possible way to do this, is by adding all received bids as actions to the set of possible actions, and using a refinement planning algorithm. For each bid where a (part of its) plan is used, the agent should receive a corresponding REWARD message.

Once a requester has awarded one or more agents with parts of the task, these results can be incorporated in the plan of the requester with the found plan operators. For the providers, the award (I_3, P_3, G_3) is added to their initial resource, their plan (using the addition operator), and their goals, respectively.

Plan merging

An instance of task-based coordination is plan-merging (de Weerdt *et al.* 2003). To facilitate the exchange of resources, we assume one of the agents, or a trusted third party, acts as the auctioneer. The auctioneer starts with announcing a minimal required potential cost reduction. All agents deposit requests that have at least such a potential cost reduction with this auctioneer. Each request corresponds with the removal of an action from an agent's plan. The auctioneer deals with the request with the highest potential cost reduction first. Right before the auction is started, the requesting agent (a_i) is asked for the specific set of resources that has to be replaced by resources of other agents. This set is not already included in the initial request, because agents may give resources they could use themselves to other agents (that had requests of a higher priority).

To put up an auction for a request of an agent a_i, the set of requested resources is sent to each agent, except to a_i. The agents return all their free resources for which there is a compatible one in the request set $RequestSet$, and include the price of each of their offered resources. When all bids (collected in R') are collected by the auctioneer, it selects for each requested resource the cheapest bid. If for each resource in $RequestSet$ a replacement can be found, the auctioneer tells the requesting agent a_i that it may discard the corresponding action(s). The replacing resources R'' are marked as goals for the providing agents, and become additional 'initial' resources for agent a_i. If not all resources can be replaced, the request is retried after completion of all other requests. This process is repeated until none of the auctions has been successful.

In (de Weerdt *et al.* 2000; 2003), we have already published a variant of this algorithm. In these original versions, all requests are collected once, in the beginning, and no threshold is used. This has mainly consequences for the order, and the frequency of the requests that could not be fulfilled in their first round. In the version presented in this thesis, failed requests have a retry at each step, not only at the end.

The plan-merging algorithm is an *any-time* algorithm (Dean & Boddy 1988), because it can be stopped at any moment. If the algorithm is stopped, it will still return an improved set of agent plans, because this algorithm used a greedy policy, i.e., dealing with the requests with the largest potential cost reduction first.

Discussion

A conventional model for cooperative multi-agent systems assumes that each agent makes its own plans and then (partly) shares them with other agents to detect helpful or harmful interactions (de Weerdt *et al.* 2003; Foulser, Li, & Yang 1992; Rosenschein 1982; Stuart 1985), for example by applying additional restrictions to the construction of plans (Yang, Nau, & Hendler 1992). These methods are called *multi-agent plan merging* methods. In general, however, it is not always possible for each agent to first construct its plan and then to coordinate. Therefore, we study the *interleaving* of planning and coordination.

(Generalized) Partial Global Planning (PGP) (Decker & Lesser 1992; Durfee & Lesser 1987; Durfee 1999) is a technique to build such systems where agents communicate parts of their local plans to build plans that are partially global. These partial global plans specify the relations among actions, and can be used by agents to adapt their local plans to other agents' actions. This approach provides a framework to exchange crucial information in specific domains to prevent conflicts and potentially exploit positive interactions.

Most solutions to multi-agent planning problems, such as PGP, consist in fact of two parts: on the one hand *planning* methods such that each agent can find a plan for itself, and on the other hand *coordination* methods for the plans of the agents. The purpose of this research is to develop a distributed algorithm that *plans the coordination*.

By coordinating plans, agents can become more efficient. For our proposed plan merging algorithm, including more agents in the coordination process leads to more efficient plans. We expect even better results using the coordinated planning algorithm. Both algorithms use a resource oriented view on the world, and can be combined with most existing planning techniques.

However, we have not discussed how to deal with agents that cannot or do not fulfill their contracts. Nor have we proposed a good way to reward agents that offer services and share resources. These issues are future work. Inaddition, we need to look at a more dynamic version of the proposed algorithm where planning, replanning and execution are integrated. Even then, this approach cannot be used in open multi-agent environments (e.g., the internet) before a way is devised to deal with different ontologies (i.e., what are the resource in this domain), and a standard for agent communication and negotiation is chosen.

References

Corkill, D. D. 1991. Blackboard systems. *AI Expert* 6(9):40–47.

Davis, R., and Smith, R. 1983. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence* 20(1):63–109.

de Weerdt, M. M.; Bos, A.; Tonino, J.; and Witteveen, C. 2000. Fusion of plans in a framework with constraints. In *Proceedings of the ISCS Conference on Intelligent Systems and Applications (ISA-00)*, 393–399.

de Weerdt, M. M.; Bos, A.; Tonino, J.; and Witteveen, C. 2003. A resource logic for multi-agent plan merging. *Annals of Mathematics and Artificial Intelligence, special issue on Computational Logic in Multi-Agent Systems* 37(1–2):93–130.

Dean, T., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, 49–54. St. Paul, MN: AAAI Press.

Decker, K. S., and Lesser, V. R. 1992. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems* 1(2):319–346.

Durfee, E. H., and Lesser, V. R. 1987. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, 875–883. San Mateo, CA: Morgan Kaufmann Publishers.

Durfee, E. H. 1999. Distributed problem solving and planning. In Weiß, G., ed., *A Modern Approach to Distributed Artificial Intelligence*. San Francisco, CA: The MIT Press. chapter 3.

Foulser, D.; Li, M.; and Yang, Q. 1992. Theory and algorithms for plan merging. *Artificial Intelligence Journal* 57(2–3):143–182.

Georgeff, M. P. 1984. A theory of action for multiagent planning. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, 121–125. Menlo Park, CA: AAAI Press.

Kambhampati, S. 1997. Refinement planning as a unifying framework for plan synthesis. *AI Magazine* 18(2):67–97.

Mali, A., and Kambhampati, S. 1999. Distributed planning. In *The Encyclopaedia of Distributed Computing*. Dordrecht, The Netherlands: Kluwer Academic Publishers. To appear.

Rosenschein, J. S. 1982. Synchronization of multi-agent plans. In *Proceedings of the Second National Conference on Artificial Intelligence (AAAI-82)*, 115–119. Menlo Park, CA: AAAI Press.

Stuart, C. J. 1985. An implementation of a multi-agent plan synchronizer. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, 1031–1033. San Mateo, CA: Morgan Kaufmann Publishers.

Tonino, J.; Bos, A.; de Weerdt, M. M.; and Witteveen, C. 2002. Plan coordination by revision in collective agent-based systems. *Artificial Intelligence* 142(2):121–145.

Wellman, M. P.; Walsh, W. E.; Wurman, P. R.; and MacKie-Mason, J. K. 2001. Auction protocols for decentralized scheduling. *Games and Economic Behavior* 35(1–2):271–303.

Yang, Q.; Nau, D. S.; and Hendler, J. 1992. Merging separately generated plans with restricted interactions. *Computational Intelligence* 8(4):648–676.