Plan coordination

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 autonomously construct their plans and just benefit from cooperation, and sometimes they cannot even do without using resources from another. We distinguish two ways to facilitate cooperation by using each others resources. With *service-based* plan coordination agents *offer* resources they can produce for others. With *request-based* coordination agents are allowed to place *requests* for resources they need. These forms of coordination can be integrated in the plans, and be realized by adapting existing planning algorithms. All methods use a special model of the relevant state of the world and of possible actions. This model consists of predicates that are restricted to describe the properties of an available entity relevant for planning. Such a *resource fact* can be exchanged among plans more conveniently than sets of ordinary propositions as used by, e.g., STRIPS to describe a resource. *Actions* consume all resource facts that are given as a precondition, and produce the resource facts defined by the post-condition.

One form of request-based coordination is when the plans of agents are coordinated after they have constructed their plans. This method, called *plan merging* is used for agents that are able to (first) create a valid plan on their own.

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 even more efficient. We would like to have a method to plan the interactions among agents and combine this with the use of existing single-agent (refinement) planning methods (Kambhampati 1997). 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).

Many planning and coordination problems can be modeled as a multi-agent planning problem. For example, the coordination of transport organizations, armies, production processes, etc., to name just a few. Currently, quite a body of research is focused on such problems, but "most of this research is still in its early stages, and many research challenges remain" (DesJardins *et al.* 2000). In this paper we focus on several autonomously planning agents that can interact through the exchange of resources without needing to provide more information if they do not want to.

An interaction between two plans occurs if 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 special predicate for each physical object that the agents can exchange. This predicate denotes that this object is *available* and describes all attributes of such an object. We call such a predicate a *resource fact*. First, we give a more formal definition of a resource fact and we define actions as processes that consume and produce these resource facts. Then, we describe two ways to plan the coordination of agents: *service-based coordination* where an agent can *offer* to produce certain resource facts, and a method where agents publish *requests* for specific resource facts, i.e., *request-based coordination*. Finally, we present a special form of request-based coordination, called *plan merging*, where agents first construct their plans and then request replacements for resource facts in their plans to be able to remove some actions.

Resource facts, actions and plans

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

A *resource fact* describes an object that is relevant to an agent with respect to the planning problem at hand. Such a resource fact represents either a physical object such as a truck or a block, or an abstract conceptual notion such as the right to do something. While using ordinary (classical) propositions (STRIPS) we need a *set* of propositions to describe all properties of such an object, in the ARF, these properties are be combined in *one* resource fact. 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 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

resource facts having the same type and possibly the same values of their attributes. Because of the special nature of this identifier, we denote a resource fact of type t with identifier i and attributes $a_1, \ldots, a_n$ of sorts $s_1, \ldots . s_n$ respectively, as $t_i(a_1 : s_1, \ldots, a_n : s_n)$. For example, we describe a truck at a location A, at 10 o'clock, and with left capacity 8 by $truck_i(A : loc,\ 10 : time,\ 8 : cap)$.

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 resource facts. Usually, a set of goals G is specified by a set of resource facts $G = \{g_1, \ldots, g_n\}$. We say that a set of goals G is *satisfied* by a given set of resource facts 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 resource facts in R. Two resource facts 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 resource facts. An action o has a set of input resource facts $in(o)$ that it consumes, and a set of output resource facts $out(o)$ that it produces. Furthermore, an action may contain a specification of some variables occurring in the set of output resource facts as *parameters* $param(o)$. To ensure that output resource facts are uniquely defined, these resource facts may only contain variables that already occur in the input resource facts or in the set of the parameters.

An action o can be applied to a set of (ground) resource facts 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 resource facts while producing the set $out(o)\theta$. The result of o applied to R (under θ) therefore is a resource fact 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 resource facts of an action on output resource facts of another action, so d can only specify a valid dependency if (i) the resource facts 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 resource facts r and r', $d(r) = r'$ implies $r\theta \equiv r'\theta$, that is θ is a *unifier* for every pair of resource facts $(r, d(r))$. 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 resource facts $r \in in(o)$ 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 dependency function, and θ is a mgu, such that the requirements above are met.

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

A *planning 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 resource facts. Resource facts that are not used as a goal or as an input of another actions are called *free resource facts*. These free resource facts of an agent play an important role in plan coordination methods, because they may be used by other agents. In a *multi-agent planning problem* each agent has its own planning problem.

Plan coordination methods

We distinguish two approaches to coordinate the plans of agents. During the construction of its plan, an agent may discover that it has a difficulty reaching a (sub-)goal, or that it cannot attain a certain (sub-)goal at all. In this case, an agent can request other agents whether they are able to provide the resource fact(s) required for this (sub-)goal. We call this approach *request-based plan coordination*. First, however, we discuss an approach where agents help each other by *offering* services (resource facts).

Service-based plan coordination

Service-based plan coordination is based on the idea that each agents publishes its capabilities that it would like to share with others. To implement service-based plan coordination we need to add the following to an existing planning algorithm.

- The problem should be modeled in terms of resource facts, i.e., for each "shareable object" all information should be specified in one resource fact.
- Agents need to determine what services they are able to offer. This can be simply all resource facts they produce in their plan and do not need themselves (free resource facts), or it can be a fixed set of resource facts they are always able to produce. Furthermore, an agent may also try to determine resource facts they are able to produce using an additional planning algorithm. Naturally, the offered services may change over time.
- 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, for example as a special action that has no input resource facts, and the offer as an output resource fact.

Task reduction methods (Clement & Durfee 1999; Erol, Hendler, & Nau 1994) can be combined with our action resource fact formalism to abstract from the result of a task offered by another agent (service) as follows. In addition to an agent's own potential task reductions, a globally accessible blackboard can contain task reduction rules from other agents. In an existing task reduction algorithm, the selection of a rule from the blackboard usually has a lower priority than the selection of one of the agent's own rules, depending on the costs. When a rule from another agent is selected, a contract is made to exchange this service, and further refinement is done by the other agent.

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

Request-based plan coordination

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

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 request for s sub-problem 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 requested sub-problem: 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 the goals for its sub-problem, 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 need to be incorporated in the plan of the requester. For each award (I_3, P_3, G_3) the resource facts I_3 are marked as special goals, and the resource facts in G_3 can be used as special initial resource facts. For the providers, given an award (I_3, P_3, G_3), the resource facts in I_3 are marked as special initial resource facts, P_3 is added to their plan, and the resource facts in G_3 are marked as special goals.

Plan merging

An instance of request-based coordination where all agents have constructed their plans is called *plan-merging* (de Weerdt *et al.* 2003). To facilitate the exchange of resource facts, we assume one of the agents, or a trusted third party, acts as the auctioneer. All agents deposit requests 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 resource facts that has to be replaced by resource facts of other agents. (This set is not already included in the initial request, because agents may give resource facts 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 resource facts is sent to each agent, except to a_i. The agents return all their free resource facts for which there is a compatible one in the request set $RequestSet$, and include the price of each of their offered resource facts. When all bids (collected in R') are collected by the auctioneer, it selects for each requested resource fact the cheapest bid. If for each resource fact 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 resource facts R'' are marked as goals for the providing agents, and become additional 'initial' resource facts for agent a_i. If not all resource facts 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.

The plan-merging algorithm is an *any-time* algorithm, because it can be stopped at any moment. If the algorithm is stopped, it still returns 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. The time complexity of this algorithm is $O(n^2)$ where n is the number of actions of the plans of all agents involved in the plan merging. When we include constraints in the problem specification, the time complexity becomes $O(n^3)$ for domains where the number of input and output resource facts is restricted. The formalism presented in this paper, the plan merging algorithm, and a both formal and practical analysis of this algorithm are included in my thesis (de Weerdt 2003, under construction). The extension to a more general coordination framework as described in the previous section is not included in this thesis, and is currently still being developed.

Discussion and related work

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), 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 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, either cooperatively create plans for all agents, called cooperative distributed planning (DesJardins *et al.* 2000), such as PGP, or focus on task allocation before planning and conflict resolution after planning. The purpose of our research is to develop a distributed algorithm that creates plans for self-interested agents including coordinated actions (negotiated distributed planning).

By coordinating plans by plan merging, agents can become more efficient. We expect even better results using the coordinated planning algorithm to plan the exchanges of resources during the construction of the plan. Both algorithms use a resource fact oriented view on the world, and can be combined with most existing planning techniques. Such a general approach to coordinating plans of multiple agents can be used to solve many practical coordination problems, such as hospital scheduling, coordinating the transportation of goods or people, and managing the planning of joint forces on a mission of the UN.

To be able to use the proposed methods on integrating planning and coordination in these situation, still much work need to be done. Firstly, we need an adequate way to reward agents that offer services and share resource facts. Secondly, we need to know how to deal with agents that cannot or do not fulfill their contracts. Furthermore, we should test the developed algorithms in more realistic environments and improve them with (maybe even domain-dependent) heuristics. In addition, we need to look at a more dynamic (continual) version of the proposed algorithm where planning, replanning and execution are integrated. Finally, 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 fact in this domain), and a standard for agent communication and negotiation is chosen.

References

Clement, B. J., and Durfee, E. H. 1999. Theory for coordinating concurrent hierarchical planning agents using summary information. In *Proc. of the Sixteenth National Conf. on Art. Intel. (AAAI-99)*, 495–502.

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. *Art. Intel.* 20(1):63–109.

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

de Weerdt, M. M. 2003. *Coordinated Planning in Multi-Agent Systems (under construction)*. Ph.D. Dissertation, Delft Technical University, Delft, The Netherlands.

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

DesJardins, M. E.; Durfee, E. H.; Ortiz, C. L.; and Wolverton, M. J. 2000. A survey of research in distributed, continual planning. *AI Magazine* 4:13–22.

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

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proc. of the Twelfth National Conf. on Art. Intel. (AAAI-94)*, volume 2, 1123–1128. Seattle, Washington, USA: AAAI Press/MIT Press.

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

Georgeff, M. P. 1984. A theory of action for multiagent planning. In *Proc. of the Fourth National Conf. on Art. Intel. (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*. Kluwer Academic Publishers. To appear.

Rosenschein, J. S. 1982. Synchronization of multi-agent plans. In *Proc. of the Second National Conf. on Art. Intel. (AAAI-82)*, 115–119. Menlo Park, CA: AAAI Press.

Tonino, J.; Bos, A.; de Weerdt, M. M.; and Witteveen, C. 2002. Plan coordination by revision in collective agent-based systems. *Art. Intel.* 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 Intel.* 8(4):648–676.