Multi-agent Planning
An introduction to planning and coordination

Mathijs de Weerdt, Adriaan ter Mors, and Cees Witteveen
Dept. of Software Technology, Delft University of Technology
E-MAIL: {M.M.DEWEERDT, A.W.TERMORS, WITT}@EWI.TUDELFT.NL

1 Introduction

Many day-to-day situations involve decision making: for example, a taxi company has some transportation tasks to be carried out, a large firm has to distribute a lot of complicated tasks among its subdivisions or subcontractors, and an air-traffic controller has to assign time slots to planes that are landing or taking off. Intelligent *agents* can aid in this decision-making process. Agents are often classified into two categories according to the techniques they employ in their decision making: *reactive* agents (cf. (Ferber and Drogoul, 1992)) base their next decision solely on their current sensory input; *planning* agents, on the other hand, take into account anticipated future developments — for instance as a result of their own actions — to decide on the most favourable course of action.

When an agent should plan and when it should be reactive depends on the particular situation it finds itself in. Consider the example where an agent has to plan a route from one place to another. A reactive agent might use a compass to plot its course, whereas a planning agent would consult a map. Clearly, the planning agent will come up with the shortest route in most cases, as it won't be confounded by uncrossable rivers, one-way streets, and labyrinthine city layouts. On the other hand, there are also situations where a reactive agent can at least be equally effective, for instance if there are no maps to consult, for instance in a domain of (Mars) exploration rovers. Nevertheless, the ability to plan ahead is invaluable in many domains, so in this paper we will focus on *planning agents*.

The general structure of a planning problem is easy to explain: (the relevant part of) the world is in a certain state, but managers or directors would like it to be in another state. The (abstract) problem of how one should get from the current state of the world through a sequence of actions to the desired goal state is a *planning problem*. Ideally, to solve such planning problems, we would like to have a general planning-problem solver. However, such an algorithm solving all planning problems can be proven to be non-existing.[1] We therefore start to concentrate on a simplification of the general planning problem called 'the *classical planning problem*'. Although not all realistic problems can be modeled as a classical planning problem, they can help to solve more

[1] That is, the general planning problem is undecidable.

complex problems. In this paper we first give an overview of planning techniques for this classical planning problem and techniques for extensions of this problem. Please, skip this first section if you are already familiar with AI planning techiques. After this introduction to AI planning, we briefly discuss coordination in a multi-agent context, and we will introduce a way to organize current work on multi-agent planning by defining several phases in the multi-agent planning process (in Section 3). Finally, we will describe some multi-agent planning techniques in more detail, and we conclude with an outlook on open issues in this field.

2 AI Planning

In this section we give an overview of AI planning techniques. It will not come as a surprise that AI planning techniques are techniques to search for a plan: *forward planning* is a planning technique building a plan starting from the initial state, *backward planning* starts from the goal states, and *least-commitment planning* by constructs plans by adding actions in a non-sequential order. Hereafter we discuss some advanced heuristics to guide this search, which is quite useful since in general the (classical) planning problem is very hard (see Theorem 8). Finally, we discuss some extensions of the classical planning problem, such as dealing with uncertainty, time, and limited resources (fuel, capacity, money, etc.).

2.1 The classical planning problem

The *classical planning problem* can be defined as follows (Weld, 1999):

> Given
>
> - a description of the known part of the initial state of the world (in a formal language, usually propositional logic) denoted by I,
> - a description of the goal (i.e., a set of goal states), denoted by G, and
> - a description of the possible (atomic) actions that can be performed, modeled as state transformation functions,
>
> determine a plan, i.e., a sequence of actions that transforms each of the states fitting the initial configuration of the world into one of the goal states.

The formal language that was used for STRIPS (Fikes and Nilsson, 1971) is common to most classical planning frameworks. This language is also used in Definition 6 to formally define the classical planning problem.

Example 1. *Suppose that initially (i.e., in all states of the world that match the description I), there is a taxi at a location A, represented by a binary state variable $taxi(A)$, and a passenger at a location B, represented by $passgr(B)$. In each of the states described by G the passenger should be at a location C, denoted by $passgr(C)$. Furthermore, suppose that there are three actions that can transform (some part of) the state of the world.*

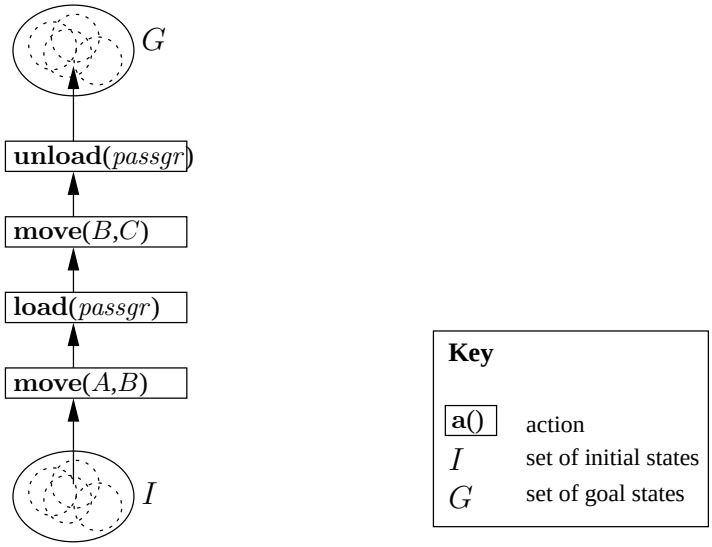

Figure 1: A sequence of actions (plan) leading from each of the initial states in I to one of the goal states in G.

1. *The taxi can move from one location to another:* $\mathbf{move}(x, y)$ *with* $x, y \in \{A, B, C\}$. *This action requires that a priori* $taxi(x)$ *holds, and ensures that in the resulting state* $\neg taxi(x)$ *and* $taxi(y)$ *hold.*

2. *The passenger can get in the taxi:* $\mathbf{load}(p)$. *This action requires a priori* $taxi(x)$ *and* $passgr(y)$ *and* $x = y$, *and in the resulting state both* $\neg passgr(y)$ *and* $passgr(taxi)$ *should hold.*

3. *The passenger can get out of the taxi:* $\mathbf{unload}()$. *This action requires* $taxi(x)$ *and* $passgr(taxi)$, *and results in* $\neg passgr(taxi)$ *and* $passgr(x)$.

A plan that represents a solution to this problem is shown in Figure 1.

Remark 2. *Note that it is not possible nor desirable to completely describe the state of the world and everything that changes. We use the assumption that "all that is not explicitly changed by an action remains unchanged" (Janlert, 1987) to deal with this* frame problem *(Raphael, 1971). Consequently, we describe (frame) a state of the world only by the relevant literals. Such a set of literals is called a* state specification. *Under this assumption, the difference between 'state' and 'state specification' is irrelevant. So often 'state' is used while 'state specification' is meant.*

In STRIPS, the STanford Research Institute Problem Solver (Fikes and Nilsson, 1971), states are described using binary state variables, called *propositions*. Actions or operators are specified by *(i)* conditions on propositions, called *preconditions (ii)* propositions that are changed to '$true$' in the new state, called *add effects*, and *(iii)*

propositions that are changed to '$false$', called *delete effects*. Furthermore, goals can be described by conditions on propositions.

The propositional STRIPS planning language has been formalized, and the complexity of the problems that can be specified in this language has been analyzed by (Lifschitz, 1987). The following formal treatment is based on a further analysis of STRIPS planning instances performed by (Nebel, 2000).

To formally specify a propositional planning formalism, we need the following concepts and notations. Given the set of all propositional atoms Σ, a literal over Σ is an element s of Σ or its negation $\neg s$. The set of all literals over Σ including $\bot$ (bottom, to denote '$false$') and $\top$ (top, to denote '$true$') is denoted by $\hat{\Sigma}$. For a set of literals $L \subseteq \hat{\Sigma}$ we define $\neg L$ to be the set of literals $\{\neg l \mid l \in L\}$, where $\neg l \equiv l'$ if $l = \neg l'$ and $l' \in \Sigma$. The set of all formulas is denoted by $PROP$, and defined as follows: if $p \in \Sigma$ then $p \in PROP$, and if $A, B \in PROP$ then $\neg A$, $(A \vee B)$, $(A \wedge B)$, $(A \rightarrow B)$, and $(A \leftrightarrow B) \in PROP$.

Definition 3. *(Nebel, 2000) An* operator $o \in 2^{\Sigma} \times 2^{\hat{\Sigma}}$ *is defined by a precondition* pre *and its effect* $post$*, denoted by* $\langle pre, post \rangle$*, where* $pre \subseteq \Sigma \subseteq \hat{\Sigma}$ *is a set of propositional atoms, and* $post \subseteq \hat{\Sigma}$ *is a set of literals.*

We use the notation $pre(o)$ and $post(o)$ to denote the precondition and the effect of an operator o, respectively. The propositional atoms in the precondition of an operator must all be true in the state $s \in 2^{\hat{\Sigma}}$ to which the operator is applied. Let O be the set of all operators in a domain.

Definition 4. *(Nebel, 2000) The* application $App : 2^{\hat{\Sigma}} \times O \rightarrow 2^{\hat{\Sigma}}$ of an operator or action o *to a state (specification)* s *is defined as*[2]

$$App(s, o) = \begin{cases} (s - \neg\, post(o)) \cup post(o) & \text{if } s \models pre(o) \text{ and } s \not\models \bot \text{ and} \\ & post(o) \not\models \bot \\ \{\bot\} & \text{otherwise} \end{cases}$$

That is, the negations of the literals in the effect clause will be removed, and the literals themselves will be added to the state s.

Using Definition 4, we can define the result $Res(s, \Delta)$ of applying a sequence of operators Δ to a state s.

Definition 5. *The result* $Res : 2^{\hat{\Sigma}} \times O^* \rightarrow 2^{\hat{\Sigma}}$ *of* applying a sequence of operators $\Delta = \langle o_1, \ldots, o_n \rangle$ *to a state (specification, see Remark 2)* s *is recursively defined by*

$$Res\,(s, \langle\rangle) = s$$
$$Res\,(s, \langle o_1, o_2, \ldots, o_n \rangle) = Res\,(App(s, o_1), \langle o_2, \ldots, o_n \rangle)$$

Finally, Nebel defines a planning problem in propositional STRIPS as follows.

Definition 6. *(Nebel, 2000) A* planning problem *in the propositional STRIPS formalism is a four tuple* $\Pi = (\Sigma, O, I, G)$ *where*

[2] We use the standard propositional entailment ($\models$) here.

- Σ *is a countably infinite set of propositional atoms, called facts or fluents,*
- $O \subseteq 2^{\Sigma} \times 2^{\hat{\Sigma}}$ *is the set of all possible operators to describe state changes in this domain,*
- $I \subseteq \hat{\Sigma}$ *is the initial state, and*
- $G \subseteq \Sigma$ *is the goal specification, i.e., the set of propositions that is to be satisfied.*

Definition 7. *A sequence of operators* $\Delta = \langle o_1, \ldots, o_n \rangle \in O^*$ *is called a* solution *or a* plan *for a planning instance* $\Pi = (\Sigma, O, I, G)$ *iff* $Res(I, \Delta) \models G$ *and* $Res(I, \Delta) \not\models \bot$.[3]

The propositional STRIPS formalism (denoted by $\mathcal{S}$) requires complete state specifications, unconditional effects, and propositional atoms as the formulas in the precondition list. This formalism can be extended in various ways: *(i)* state specifications may be incomplete ($\mathcal{S}_{\mathcal{I}}$), *(ii)* effects can be conditional ($\mathcal{S}_{\mathcal{C}}$), *(iii)* formulas in preconditions (and effect conditions) can be literals $\subseteq \hat{\Sigma}$ ($\mathcal{S}_{\mathcal{L}}$), and *(iv)* the formulas in preconditions (and effect conditions) can be arbitrary boolean formulas $\subseteq PROP$ ($\mathcal{S}_{\mathcal{B}}$).

Theorem 8. *(Nebel, 2000) Complexity of planning. The problem of deciding whether a plan* exists *for a given instance (i.e., the* plan existence problem*) is PSPACE-complete for specifications in the propositional STRIPS formalism and for all combinations of the four extensions of this language ($\mathcal{S}_{\mathcal{I}}$, $\mathcal{S}_{\mathcal{C}}$, $\mathcal{S}_{\mathcal{L}}$, and $\mathcal{S}_{\mathcal{B}}$) described above.*

This result is from (Nebel, 2000), who in turn relies on a proof by (Bylander, 1994). The idea behind this proof is as follows. Plan existence is in PSPACE because *(i)* a state is described by n propositions, and thus there are at most 2^n states. Since any plan can be conceived as a sequence of states, *(ii)* the maximal plan length needed to transform an initial state into a goals state is at most 2^n. Therefore, *(iii)* the existence of a plan of length p transforming a state s_1 into a state s_2 can be decided using an algorithm that uses $O\left(log(p)\right)$, hence polynomial, space: for any intermediate state s_3 decide plan existence both from s_1 to s_3, and from s_3 to s_2, recursively. Since each state requires $O(n)$-space and the depth of the recursion of this algorithm is at most $O\left(log(p)\right) = O(log(2^n)) = O(n)$, the algorithm clearly runs in polynomial space.

The hardness of plan existence in Bylander's proof is based on the fact that each PSPACE problem can be translated into a planning problem.

Nebel also analyzes which instances of the different extensions of the planning problem can be translated into each other, and which cannot. Figure 2 shows a graph of these specialization relations between combinations of the problem extensions, indicated by combinations of the subscripts $\mathcal{I}$, $\mathcal{C}$, $\mathcal{L}$, and $\mathcal{B}$. An arrow from an extension$\mathcal{S}_{\mathcal{L}}$ (allow literals) to another extension $\mathcal{S}_{\mathcal{B}}$ (allow arbitrary boolean formulas) means that any planning problem specified in $\mathcal{S}_{\mathcal{L}}$ can be translated into a problem in $\mathcal{S}_{\mathcal{B}}$ in polynomial time, while the plan size increases at most linearly in the size of the input.

Apart from a specialization relationship of all extensions of the STRIPS formalism, Nebel's most interesting results are *(i)* that incomplete state specifications and literals in preconditions can be compiled to the basic STRIPS formalism where the plan size is

[3] We use the word 'iff' as a short hand for 'if and only if'.

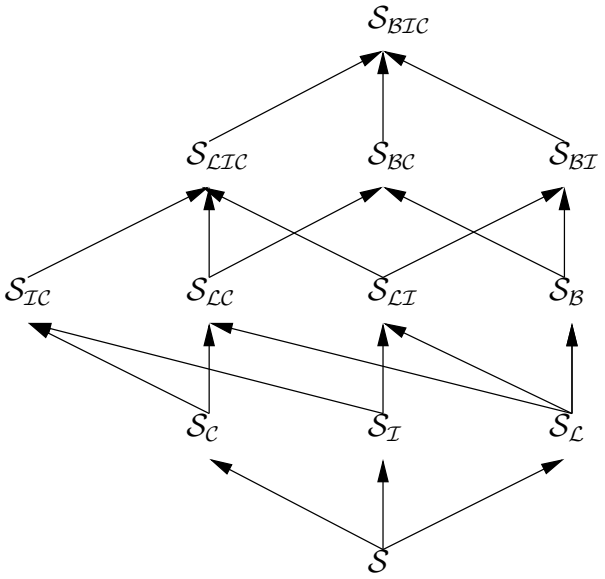

Figure 2: The specialization relationships of planning formalisms based on syntactic restrictions (Nebel, 2000).

preserved, and *(ii)* that incomplete state specifications and literals in preconditions and effect conditions can be compiled to the basic STRIPS formalism with conditional effects preserving plan size exactly, and *(iii)* that there are no other compilation schemes preserving plan size linearly except those implied by the specialization relationship and results *(i)* and *(ii)*.

There exist many different approaches to solve the classical planning problem that make use of the STRIPS formalism and its extensions. Later on, we will discuss some of these approaches in a multi-agent setting. To prepare for this discussion, in the next section, we briefly discuss the most relevant approaches and we describe them as variants of a general method based on the refinement of the set of all possible plans. This generalization has been made by (Kambhampati, 1997).

2.2 Refinement planning

The search for a sequence of actions (a plan) to get from an initial state to a goal state can be seen as a refinement of the set of all possible sequences (Kambhampati, 1997). We can describe most existing (classical) planning algorithms using this unifying view on planning. The unifying concept, representing a set of *candidate* sequences, is a so-called *partial plan*. This partial plan is used to describe a set of partial solutions. Planning algorithms are given by defining how a partial plan is modified such that in the end all (partial) solutions represented by the partial plan are complete, feasible solutions.

Definition 9. *(Kambhampati, 1997) The syntax of a* partial plan*, usually denoted by P_i, for a planning problem $\Pi = (\Sigma, O, I, G)$ is a directed acyclic graph (V, E, IPC, PTC), where V is a set of the nodes representing the application of actions from O, $E \subseteq V \times V$ is the set of directed edges representing the required precedences of these actions, and IPC and PTC are two kinds of auxiliary constraints:*

- *$IPC \subseteq V \times V \times PROP$ is a set of* interval preservation constraints *specifying that a formula should hold during a certain interval (between two actions). Currently, the only used instantiations of such constraints are so-called causal links.*[4] *A causal link $o_i \xrightarrow{p} o_j$ specifies that the precondition p of action $o_j$should be an effect of o_i that may not be undone between o_i and o_j.*
- *$PTC \subseteq V \times PROP$ is a set of* point truth constraints *specifying that a formula should hold at a certain point before an action. In current planners these constraints are only used to specify open preconditions of actions.*

The semantics of a partial plan is the set of action sequences that *(i)* contain at least the actions represented by the nodes V, and *(ii)* fulfill the precedence constraints implied by the directed edges E. Furthermore, these action sequences should satisfy *(iii)* all *interval preservation constraints* IPC, and *(iv)* the *point truth constraints* PTC. These action sequences are called *candidate* plans or sequences of a partial plan P_i, denoted by $candidates(P_i)$.

The IPC and PTC are especially useful if plans are not constructed bottom-up or top-down. When actions are added in no particular order, somehow it must be stored why a particular action is added at all (e.g., to meet the precondition of a subsequent action), and, more importantly, it has to be prevented that another action undoes its required result. This kind of information can be stored using the interval preservation constraints. Point truth constraints specify which preconditions still have to be satisfied. They can also be used to indicate at (or before) which point in a plan they need to be fulfilled.

For some algorithms, intermediate results cannot be represented by one partial plan. Therefore, we think in terms of sets of partial plans. We define the candidate plans of a set of partial plans PS as $candidates(PS) = \bigcup_{P_i \in P} candidates(P_i)$. Each subset of candidates is represented by a partial plan $P_i \in PS$, called a *component. Minimal candidates* are those candidates that contain only actions that are included in a partial plan (i.e., that are represented in V).

Given a partial plan, we need to define when a plan (sequence) is one of the candidates represented by this partial plan. Such a sequence is called a *safe linearization*.

Definition 10. *(Kambhampati, 1997) A sequence $\Delta = \langle o_1, \ldots, o_n \rangle \in O^*$ is called a* safe linearization *of a partial plan $P_i = (V, E, IPC, PTC)$, if*

- *there exists a bijective function $f : V \rightarrow \Delta$, such that any $v \in V$ represents the application of operator $f(v) \in \Delta$ ($v \equiv f(v)$),*

[4] Causal links are used by least-commitment planners (Weld, 1994). These planners are briefly discussed at the end of this section.

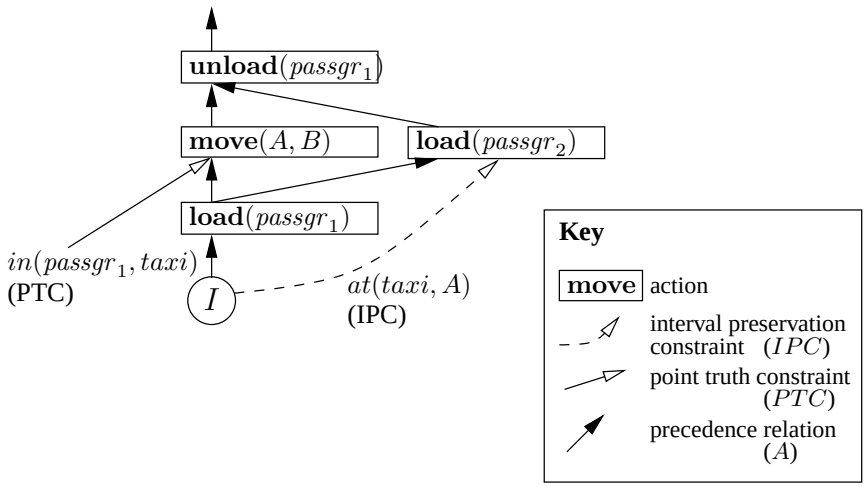

Figure 3: A partial plan describing all candidates that contain at least the four given actions, in the order given by the precedence relation and matching the interval preservation constraints and the point truth constraints.

- *for any $v, w \in V$, $v \neq w$, if there is a path from v to w in (V, E) then $f(v) < f(w)$ in Δ,*
- *for any $(v, w, \varphi) \in IPC$, if $o_i = f(v) < f(w) = o_j$ and if $Res\,(I, \langle o_1, \ldots, o_i \rangle) \models \varphi$ then $Res\,(I, \langle o_1, \ldots, o_{j-1} \rangle) \models \varphi$, and*
- *for any $(v, \varphi) \in PTC$, $Res\,(I, \langle o_1, \ldots, f(v) \rangle) \models \varphi$.*

Example 11. *A taxi has to take two passengers $passgr_1$ and $passgr_2$ from location A to location B. Initially, both the taxi and the passengers are at location A. The partial plan that describes the set of candidate plans after some additional refinements (i.e., the addition of two* **load** *actions, a* **move***action, and an* **unload** *action) is depicted in Figure 3. After some more refinements, each of the candidates should be a correct sequence of actions from the initial state to one of the desired goal states (i.e., where $at(passgr_1, B)$ and $at(passgr_2, B)$ hold).*

A planner can reduce the set of candidate plans represented by a set of partial plans, by *(i)* adding actions, *(ii)* adding precedences on these actions, and *(iii)* adding auxiliary constraints to one or more of the partial plans. A technique to find suitable refinements is called a *refinement strategy*, denoted by $\mathcal{R}$.

Proposition 12. *(Kambhampati, 1997) Refinement. A refinement strategy $\mathcal{R}$ refines (i.e., reduces) the set of candidate plans, represented by a set of partial plans PS: so, if $PS' = \mathcal{R}(PS)$ then $candidates(PS') \subseteq candidates(PS)$.*

To evaluate the effectiveness of such strategies for refining the set of candidate plans, we look at specific properties. For example, it is important to know whether a strategy preserves all possible solutions (i.e., is complete).

Definition 13. *A refinement strategy $\mathcal{R}$ to find a set of $solutions$ is called*

1. progressive *iff* $candidates(\mathcal{R}(PS)) \subset candidates(PS)$ *for any set of partial plans P,*
2. complete *iff* $solutions \cap candidates(\mathcal{R}(PS)) = solutions \cap candidates(PS)$, *and*
3. systematic *iff the set* $\{candidates(P_i) \mid P_i \in \mathcal{R}(PS)\}$ *is a partition of* $candidates(\mathcal{R}(PS))$.

Although planning algorithms are implemented in many different ways, they can be rewritten in an alternative, uniform way, based on the theory of refining a set of potential solutions (Kambhampati, 1997). The structure they then have in common can be found in Algorithm 14. This algorithm describes how a solution $result$ can be obtained from a set of partial plans PS. Each time the function REFINE is executed, and none of the minimal candidates is a solution, a refinement strategy is selected and applied to one of partial plans P. Then repeatedly an element of the resulting set PS is selected, and this function REFINE is called recursively using this element. Once a solution has been found, this process stops, and the result is returned.

Algorithm 14. ***(***REFINE (P, Π) ***)***

Input: *A partial plan P and a problem Π.*

Output: *A minimal candidate of P that is a solution to Π or 'fail'.*

begin

1. **if** *a minimal candidate c of P is a solution to Π* **then**
   1.1. **return** c
2. **else**
   2.1. $result :=$ *fail*
   2.2. *select a refinement strategy $\mathcal{R}$*
   2.3. $PS := \mathcal{R}(P)$
   2.4. **while** $PS \neq \emptyset$ *and* $result =$ *fail* **do**
      2.4.1. *non-deterministically select an element P_i of PS*
      2.4.2. $PS := PS \setminus \{P_i\}$
      2.4.3. $result :=$ REFINE(P_i, Π)
   2.5. **return** $result$

end

Example 15. *The planning algorithm Fast Forward (FF) (Hoffmann and Nebel, 2001) starts with the initial state and an empty sequence of actions (plan). Repeatedly, the sequence is extended with actions, always adding to the end of the sequence. For each of the possible extensions of the sequence (first with one action, then with two actions,*

etc.), a heuristic value is calculated. The first of the possible extensions that leads to a state with a lower heuristic value than the current state is chosen.

The heuristic uses the relaxation that no subsequent action has negative effects. Under this assumption, a so-called relaxed plan can be constructed where all actions with satisfied preconditions are added in parallel. The relaxed plan is constructed until the goal state is reached. The heuristic value is the cost of all actions in the relaxed plan that are needed to reach the goal state.

Although FF does not use the presented refinement framework, it is in fact a form of refinement planning with the following refinement strategy $\mathcal{R}_{\text{FF}}$ (which can be used in step 2.3 in Algorithm 14). Given one partial plan that represents a set of possible solutions, a new partial plan is constructed by extending this partial plan at the end. The extension is selected using the heuristic described above. See also Algorithm 16. Note that this particular refinement strategy uses only a singleton *set of partial plans to represent of the set of candidates.*

Algorithm 16. ***($\mathcal{R}_{\mathbf{FF}}(PS)$)***

Input: *A singleton set of partial plans PS.*

Output: *A singleton set of partial plans that lead to a state with a lower heuristic cost.*

begin

1. *$h :=$ the heuristic costs of the current end state*
2. *$\Delta :=$ the action sequence of a minimal candidate of PS*
3. *$h' := \infty$*
4. **while** *$h' \geq h$* **do**
   - 4.1. *breadth-first select a sequence of actions Δ' to extend Δ*
   - 4.2. *$h' :=$ the heuristic value of the end state of $\Delta + \Delta'$*
5. **return** *the partial plan that represents $\Delta + \Delta'$*

end

Refinement strategies such as $\mathcal{R}_{\text{FF}}$ can be roughly divided into three categories.

1. *Progression* or *forward planning methods* construct (partial) plans bottom-up: an action to be executed is selected and added to the end of the partial plan. The action-selection mechanism varies, but usually a heuristic is used to determine a next action to execute. Some methods do not use backtracking: once an action is selected, it will not be removed from the partial plan. Such planners are not complete, but can often find correct plans much faster. Examples of such planners are FF (Hoffmann and Nebel, 2001), HSP (Bonet and Geffner, 2002), and Prodigy (Veloso et al., 1995).

2. *Regression refinement methods*, also called backward planning methods, construct (partial) plans top-down: starting from the description of the set of end

states, they determine an action to reach such a state from a state s that is presumably 'closer' to the initial state (i.e., a shorter sequence of actions is needed to reach this state s). The STRIPS planner (Fikes and Nilsson, 1971) is one of the first planners that use this technique. Many others use some of the ideas from this technique, for example as a heuristic for progression refinement, as in GRT (Refanidis and Vlahavas, 2001).

An advantage of both progression and regression refinement is that these methods can describe a partial plan by a sequence of actions and a description of the state reached last, and that they can search in the *state space* to find what action to add next, and do not need a (more complex) *plan space* representation. Given such a state-space representation one can easily see whether a plan is valid by comparing the final state of the plan to the requirements of the goal state, or by comparing the first state of a plan to the initial state, respectively.

3. The third category, *least-commitment planning* (Weld, 1994), really needs the complicated plan space representation of the partial plans as used in the refinement framework, because this type of planning refines plans not only by extending the prefix or the postfix of the plan, but also adds constraints on the possible solutions in many ways. This category is sometimes also called *partial order planning*, because during the construction of a plan the order of the actions in the plan is partial, while the order of the actions in a partial plan based on forward planning is usually complete. The planners Noah (Sacerdoti, 1975) and POP (Weld, 1994) fit into this category.

Some approaches use a slightly different variant of a partial plan, called a *disjunctive* partial plan. The nodes in the disjunctive partial plan may consist of several actions. The semantics of such a node is that exactly at that point, one of those actions is to be executed. For example Graphplan (Blum and Furst, 1997) uses a form of disjunctive planning.

Example 17. *The Graphplan algorithm consists of two phases. First a planning graph is constructed that represents all possible solutions that reach the goal state in a minimum amount of planning steps. Then a solution is extracted from this planning graph.*

The planning graph is a directed, layered graph. An example of such a graph is shown in Figure 4. In this graph two types of layers are interleaved: proposition layers and action layers. A proposition layer consists of nodes that each represent an atom or the negation of an atom. The first layer is a proposition layer representing the initial state. An action layer contains a node for each possible action. The nodes are connected by three types of arcs. Precondition arcs connect the action to the atoms of its precondition of a previous layer. Add arcs connect an action to the nodes of the next layer, representing an atom that is a direct consequence of the postcondition of the action, and delete arcs connect actions to the nodes representing atoms that are disabled by the action. Propositions are always reproduced in the next layer by so-called "no-op" actions. Except when another action is chosen that has a delete effect for this proposition, this leads to a conflict.

Such mutual exclusions (mutexes) are represented explicitly in Graphplan. These mutexes indicate for each action layer which actions cannot be fulfilled at the same time

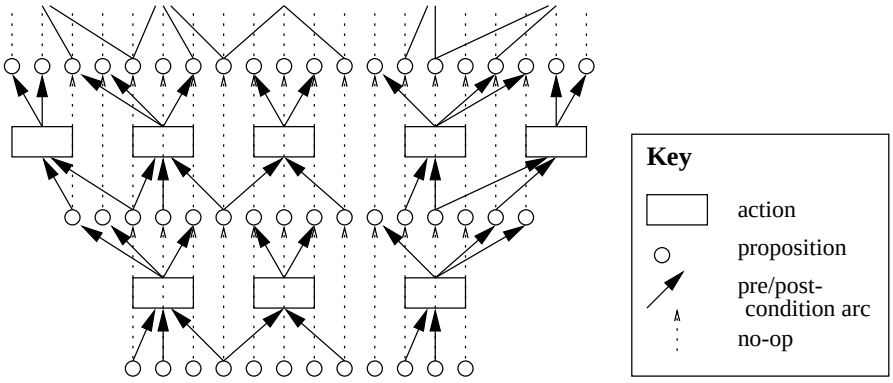

Figure 4: Graphplan uses a plan graph consisting of proposition layers and action layers.

(in the same layer). To find a correct plan, Graphplan builds the graph while searching for a proposition layer that implies the goal state. Backwards from this goal state the mutual exclusion relations are verified. If it is impossible to satisfy all mutexes, the plan graph is extended with another layer and the process is repeated.

2.3 Extended planning problems

Over the last few years many extensions of the classical planning problem have been studied: dealing with time (Do and Kambhampati, 2001; Penberthy and Weld, 1994; Smith and Weld, 1999), costs (or utility maximization) (Haddawy and Hanks, 1998), limited resources (Koehler, 1998; Wolfman and Weld, 2001), and planning under uncertainty (Boutilier et al., 1999). Of these extensions, planning under uncertainty is maybe the most relevant when multiple agents are acting in the same environment. Such domains introduce several types of uncertainty.

Firstly, actions can have probabilistic effects: for example, upon moving to another location by train, we know that we have 85 percent chance of actually reaching our destination in time, 10 percent of getting stuck somewhere, and 5 percent chance of getting involved in an accident. When the outcomes of actions can be (partially) observed, a plan can be constructed including sensing actions and conditional branches. This problem is called *contingent planning*. This leads to a second type of uncertainty.

The sensing actions may fail as well or may not be able to observe the world completely. The unobservable, partially observable, and fully observable domain versions of this problem are EXPTIME-complete, EXPSPACE-complete, and NEXPTIME-complete, respectively (Bernstein et al., 2000; Haslum and Jonsson, 1999). The additional complexity over propositional STRIPS plan existence (Theorem 8) comes from the uncertain results of actions. Not only is the length of a plan exponential, now each plan can have an exponential number of resulting end states.

Thirdly, planning in a domain where we lack even a probability distribution over

the possible outcomes of a non-deterministic operator, is called *non-deterministic planning* or *conformant planning*. So, conformant planning is the problem of finding, in a nondeterministic domain, a sequence of actions which will achieve the goal for all possible contingencies. The complexity of a conformant planning problem is EXPSPACE-complete, i.e., strictly higher than that of classical planning (Haslum and Jonsson, 1999).

Theorem 18. *(Haslum and Jonsson, 1999) Complexity of conformant planning. Deciding the existence of a conformant plan (sequence) for a problem* Π *with an unobservable propositional domain and actions is EXPSPACE-complete.*[5]

In a multi-agent system, agents often perform actions unexpectedly and independently of each other. When each agent is planning on its own without communicating and coordinating with the other agents, each agent has to solve some sort of non-deterministic planning problem. Theorem 18 and the complexity analysis of contingent planning show that non-determinism in planning is EXPTIME-hard. So we can conclude that such an individual approach to multi-agent planning is EXPTIME-hard.

Corollary 19. *An individual approach to planning in multi-agent systems using either conformant or contingent planning is EXPTIME-hard.*

However, by communicating parts of plans of the agents, we can reduce or even in some domains remove the uncertain effects of actions. Such approaches to planning in multi-agent systems are discussed in the remainder of this paper.

3 Coordinated planning

One may wonder why we need to study multi-agent planning problems and techniques as a separate topic. Is not the multi-agent case covered by the general discussions of planning? The answer is *no*, because in real-life problems, we deal with multiple agents having their own goals, and it is often impractical or undesirable to create the plan for all agents centrally. These agents may be people or companies simply demanding to plan their actions themselves, or refusing to make all information necessary for planning available to someone else. Consequently, such agents want to be able make their own plans independently of what the other agents are planning to do. This itself is not a compelling reason to differentiate between planning and multi-agent planning, but in many cases dependencies between the tasks of the agents make independent planning impossible. That is, if the agents do not take into account the dependencies between their plans, then they might come into conflict when they try to execute their plans. To resolve their dependencies, agents must *coordinate* their efforts. In the (multi-agent) literature, several definitions for coordination are given. A concise and clear definition is from (Malone and Crowston, 1991):

[5]The class EXPSPACE is the class of decision problems that can be solved using an amount of space bounded by $2^{p(n)}$, where p is a polynomial and n is the input length. It is known that the class EXPTIME$\subseteq$EXPSPACE contains problems that are intractable (Garey and Johnson, 1979, Theorem 7.16 in), even if P=NP.

Coordination *is the act of managing interdependencies between activities.*

Another, more planning-agent specific definition is due to (Jennings, 1996):

> *Participation in any social situation should be both simultaneously constraining, in that agents must make a contribution to it, and yet enriching, in that participation provides resources and opportunities which would otherwise be unavailable.* Coordination*, the process by which an agent reasons about its local actions and the (anticipated) actions of other agents to try to ensure the community acts in a coherent manner, is the key to achieving this objective.*

Clearly then, the multi-agent planning problem has both a planning and a coordination component. We therefore define the multi-agent planning problem as follows:

Definition 20. *The* multi-agent planning problem *is the following problem: Given a description of the initial state, a set of global goals, a set of (at least two) agents, and for each agent a set of its capabilities and its private goals, find a plan for each agent that achieves its private goals, such that these plans together are coordinated and the global goals are met as well.*

Summarizing, the following statement perfectly captures our concept of multi-agent planning:

Multi-agent planning = planning + coordination

In the remainder of this section we will first discuss the nature of the coordination problem, and then discuss the various approaches that combine the planning and the coordination part.

3.1 Coordination in multi-agent systems

To characterize multi-agent coordination problems, we identify several key characteristics. The first is the nature of the dependencies that necessitate the coordination. One may wonder whether there are a number of common problems that underlie any coordination situation. Having identified the nature of the coordination problem, we then have to choose the mechanism to solve the coordination problem. Again, we might ask whether there exist general coordination mechanisms that can be applied to a variety of situations. We are also interested in determining which coordination mechanism is the best for a given situation, assuming we can choose from a number of mechanisms. The third aspect is the applicability and usability of a given coordination mechanism. Any multi-agent system designer must make certain assumptions — for instance, about the number of agents in the system environment and the rate of change in the environment — about the environment in which the agents will operate. Hence, we need to know the factors determining the applicability of the coordination mechanism.

(Jennings, 1996) and (Nwana et al., 1996) state five reasons that might necessitate coordination in multi-agent systems. The first two are very general in nature and do not directly refer to any characteristic of an *interaction situation*, the other points do refer to characteristics of the interaction situation.

1. *Prevent anarchy or chaos:* As an example they state that in British Telecom, no one in the company is aware of the activities of all 130000 employees. Consequently, agents have only local views, which is bound to lead to conflicts without coordination.

2. *Efficiency:* Even if agents can work independently, sometimes working together means being able to solve problems faster.

3. *Meeting global constraints:* If there are global budget limits, then agents must make agreements so that they do not inadvertently exceed the budget.

4. *Distributed information, expertise or resources:* Often, a task cannot be performed (or performed efficiently) by a single agent alone. If the required capabilities are distributed among the agents, coordination is necessary.

5. *Dependencies between the agents' actions:* For instance, different tasks may need the same resources or there may exist a precedence relation between tasks.

Of course, the above categories are not orthogonal; the first four are all, in fact, derivable from the fifth.

In their discussion of their well-known multi-agent planning framework, the GPGP-planning framework, (Decker and Lesser, 1994) summarize the need for coordination from a different point of view. Whereas Jennings and Nwana state reasons for coordination due to the characteristics of the interaction situation, Decker and Lesser state reasons for coordination based on the point of view of an individual agent. Decker and Lesser use the definition of coordination given by Malone and Crowston. That is, that coordination is the act of managing interdependencies between activities. Using this definition as a context, Decker and Lesser say that there is a coordination *problem* if one of the following conditions is met:

- The agent has a choice of actions and that choice affects performance.
- The order in which activities are carried out affects performance.
- The time at which actions are executed affects performance.

(Malone and Crowston, 1991, 1993) summarize the need for coordination in their coordination definition — to manage interdependencies between activities. The focus of their research is finding general coordination *mechanisms* that can be applied in coordination situations of different research disciplines. Their research is characterized by the following questions they raise:

> *Are there fundamental coordination processes that occur in all coordinated systems? . . . How far can we get by analyzing very general coordination processes and when will we find that most of the important factors are specific to coordinating a particular kind of task? For example, are there general heuristics for coordination that are analogous to the general problem-solving heuristics studied in cognitive science and artificial intelligence?*

They identify the following coordination processes that occur in many domains:

- coordinated goal selection and decomposition,
- coordinated resource allocation, a special case of which is task assignment,
- coordinated sequencing (one activity after the other) and synchronizing (activities at the same time).

Of course, the above list of processes represent classes of coordination mechanisms; there is not one task assignment protocol or one goal decomposition algorithm.

(Durfee, 2001) identifies factors influencing the usability of specific coordination mechanisms. That is, he identifies the factors that determine how far the applicability of specific coordination mechanisms reach — and he concludes that the applicability of every coordination mechanism has its limits:

> *. . . various coordination strategies for computational agents have emerged over the years. It does not seem possible, however, to devise a coordination strategy that works well under all circumstances; if such a strategy existed, human societies would substitute it for the myriad constructs employed today such as corporations, governments, markets, teams, committees, professional societies, and mailing groups. Whatever strategy we adopt, certain situations can stress it to the breaking point.*

Durfee identifies three varying properties (dimensions) of an interaction situation: The *agent population*, the *task environment* and the *solution properties*. For each of these dimensions, he identifies the three most obvious properties that impact the usability of a coordination strategy.

Agent population :

- *Quantity:* The number of agents.
- *Heterogeneity:* For example, agents can have different capabilities, internal architectures and communication languages.
- *Complexity:* Complexity refers to how hard it is to predict what a *versatile* agent will do, i.e., how *predictable* the agents are.

Task environment :

- *Degree of interaction:* Some issues concern large groups of agents, such as a resource that regulates exclusive access to a blackboard datastore, other issues concern small groups agents, such as collision avoidance.
- *Dynamics:* The rate at which the environment changes, typically due to events outside the influence of the agents.
- *Distributivity:* For instance, tasks can originate centrally or distributively.

Solution properties :

- *Quality:* We can measure the quality of a solution by judging how well it coordinates agent interactions (e.g. near optimal or merely acceptable) or how efficient it is in utilizing agent resources.
- *Robustness:* To what extent do changes in the environment invalidate the plans or goals of agents?
- *Overhead limitations:* Communication bandwidth may be limited.

Scaling up along combinations of these dimensions poses even greater challenges. For instance, the delays associated with propagating information in a highly distributed setting compound the difficulties that arise in a dynamic environment.

3.2 Approaches to multi-agent planning

Multi-agent planning techniques cover quite a range of solutions to different parts of the problem. In this section we structure existing work using the phases in the process of solving a multi-agent planning problem. In general, the following phases can be distinguished (generalizing the main steps in task sharing by (Durfee, 1999)).

1. Refine the global goals or tasks until subtasks remain that can be assigned to individual agents (global task refinement).
2. Allocate this set of subtasks to the agents (task allocation).
3. Define rules or constraints for the individual agents to prevent them to produce conflicting plans (coordination before planning).
4. For each agent: make a plan to reach its goals (individual planning).
5. Coordinate the individual plans of the agents (coordination after planning).
6. Execute the plans and synthesize the results of the subtasks (plan execution).

Not always all phases of this general multi-agent planning process need to be included. For example, if there are no common or global goals, there is no need for phase 1 and 2, and possible conflicts can be dealt with on forehand (in phase 3) or afterwards (in phase 5). Also, some approaches combine different phases. For example, agents can already coordinate their plans while constructing their plans (combination of phase 4 and 5), or postpone coordination until the execution phase (combination of phase 5 and 6), as, e.g., robots may do when they unexpectedly encounter each other while following their planned routes.

For each of the phases that can be distinguished in a multi-agent planning process, we describe some of the currently most well-known approaches that can be used to deal with the issues arising in such a phase.

Global task refinement

In the first phase, the global tasks or goals are refined such that each remaining task can be done by a single agent. Apart from single-agent planning techniques such as HTN (Erol et al., 1994), or non-linear planning (Penberthy and Weld, 1992; Sacerdoti, 1975), special purpose techniques have been developed to create a global multi-agent plan. Such so-called centralized multi-agent planning approaches in fact use the classical planning framework to construct and execute multi-agent plans (Katz and Rosenschein, 1989; Pednault, 1987).

Task allocation

The centralized multi-agent planning methods mentioned before usually also take care of the assignment of tasks to agents (phase 2). There are, however, many other methods to establish such a task assignment in a more distributed way, giving the agents a higher degree of autonomy and privacy, e.g., via complex task allocation protocols (Shehory and Kraus, 1998) or auctions and market simulations. An auction is a way to make sure that a task is assigned to the agent that attaches the highest value (called private value) to it (Walsh et al., 2000; Wellman et al., 2001). A (Vickrey, 1961) auction is an example of an auction protocol that is quite often used. In a Vickrey auction each agent can make one (closed) bid, and the task is assigned to the highest bidder for the price of the second-highest bidder. This auction protocol has the nice property that bidding agents are stimulated to bid their true private value (i.e., exactly what they think it's worth to them). Market simulations and economics can also be used to distribute large quantities of resources among agents (Walsh and Wellman, 1999; Wellman, 1993; Wellman et al., 1998). For example, in (Clearwater, 1996) it is shown how costs and money are turned into a coordination device. These methods are not only used for task assignment (phase 2), but can also be used for coordinating agents after plan construction (phase 5). In the context of value-oriented environments, game-theoretical approaches (where agents reason about the cost of their decision making (or communication) become more important. See, for example, work by Sandholm, supported by results from a multiple dispatch center vehicle routing problem (Sandholm and Lesser, 1997). An overview of value-oriented methods to coordinate agents is given in (Fischer et al., 1998). Especially Markov decision processes give an interesting opportunity to deal with a partially observable world as well (Pynadath and Tambe, 2002).

Coordination before planning

In phase 3 the agents are coordinated before they even start creating their plans. This can be done, for example, by introducing so-called social laws. A social law is a generally accepted convention that each agent has to follow. Such laws restrict the agents in their behavior. They can be used to reduce communication costs and planning and coordination time. In fact, the work of (Yang et al., 1992) and (Foulser et al., 1992) about finding restrictions that make the plan merging process easier, as discussed in the previous section, is a special case of this type of coordination. Typical examples of social laws in the real world are traffic rules: Because everyone drives on the right side

of the road (well, almost everyone), virtually no coordination with oncoming cars is required. Generally, solutions found using social laws are not optimal, but they may be found relatively fast. How social laws can be created in the design phase of a multi-agent system is studied by (Shoham and Tennenholtz, 1995). (Briggs, 1996) proposed more flexible laws, where agents first try to plan using the strictest laws, but when a solution cannot be found agents are allowed to relax these laws somewhat.

Another way to coordinate agents is to figure out the exact interdependencies between their tasks beforehand. Prerequisite constraints can be dealt with centrally using existing planning technology (such as partial order planning (Weld, 1994, among others)) by viewing these tasks as single-agent tasks. More recently, an approach has been proposed to deal with interferences (such as shared resources) between the goals of one agent (Thangarajah et al., 2003).

Coordination before planning can also be used to coordinate competitive agents that insist on their planning autonomy. Here, the problem is that we have a set of interrelated (sub)goals that have to be reached by a set of planning agents, that do not want to be interfered during their planning activity. That is, each of the agents requires full planning autonomy, but at the same time we have to be sure that whatever (sub) plan they will construct to solve their part of the problem, these sub plans can be coordinated seamlessly without requiring replanning. Planning problems like these often occur in multi-modal transportation problems: several parties have to ensure that packages are transported from their source locations to their destinations. The planning agents are prepared to carry out their part of the job if it can be guaranteed that they will not be interfered by the activities of other agents.

It is clear that most of those planning problems cannot be decomposed into independent subproblems without changing the original planning problem. In (ter Mors et al., 2004) a preplanning coordination method is described that adds a minimal set of additional constraints to the subgoals to be performed in order to ensure a coordinated solution by independent planning.

Individual planning

The fourth phase consists of individual planning for each of the agents. In principle, any planning technique can be used here, and different agents may even use other techniques. There are a couple of approaches that integrate planning (phase 4) and the coordination of plans (phase 3 and 5). In the Partial Global Planning (PGP) framework (Durfee and Lesser, 1987), and its extension, Generalized PGP (Decker and Lesser, 1992, 1994), each agent has a partial conception of the plans of other agents using a specialized plan representation. In this method, coordination is achieved as follows. If an agent A informs another agent B of a part of its own plan, B merges this information into its own partial global plan. Agent B can then try to improve the global plan by, for example, eliminating redundancy it observes. Such an improved plan is shown to other agents, who might accept, reject, or modify it. This process is assumed to run concurrently with the execution of the (first part of the) local plan. PGP has first been applied to the distributed vehicle monitoring test bed, but, later on, an improved version has also been shown to work on a hospital patient scheduling problem. Here (Decker and Li, 2000) used a framework for Task Analysis, Environment Modeling, and Simulation

(TAEMS) to model such a multi-agent environment. An overview of the PGP related approaches is given by (Lesser et al., 1998). (Clement and Barrett, 2003) improved upon this PGP framework by separating the planning algorithm from coordinating the actions, using a more modular approach called shared activities (SHAC).

Coordination after planning

A large body of research focused on what to do after plans have been constructed separately (phase 5). These plan merging methods aim at the construction of a joint plan for a set of agents given the individual (sub) plans of each of the participating agents. (Georgeff, 1983, 1988) was one of the first to actually propose a plan-synchronization process starting with individual plans. He defined a so-called process model to formalize the actions open to an agent. Parts of such a process model are the correctness conditions, which are defined on the state of the world and must be valid before execution of the plan may succeed. Two agents can help each other by changing the state of the world in such a way that the correctness conditions of the other agent become satisfied. Of course, changing the state of the world may help one agent, but it may also interfere with another agent's correctness conditions (Georgeff, 1984).

(Stuart, 1985) uses a propositional temporal logic to specify constraints on plans, such that it is guaranteed that only feasible states of the environment can be reached. These constraints are given to a theorem prover to generate sequences of communication actions (in fact, these implement semaphores) that guarantee that no event will fail. To both improve efficiency and resolve conflicts, one can introduce restrictions on individual plans (in phase 3) to ensure efficient merging. This line of action is proposed by (Yang et al., 1992) and (Foulser et al., 1992), and can also be used to merge alternative plans to reach the same goal.

Another approach to merging a set of plans into a global plan deals with problems arisen from both conflicts and redundant actions by using the search method A* and a smart cost-based heuristic: (Ephrati and Rosenschein, 1993) showed that, by dividing the work of constructing sub plans over several agents, one can reduce the overall complexity of the merging algorithm (Ephrati and Rosenschein, 1994).

In other works on plan merging, (Ephrati et al., 1995a; Rosenschein, 1995) propose a distributed polynomial-time algorithm to improve social welfare (i.e., the sum of the benefits of all agents). Through a process of group constraint aggregation, agents incrementally construct an improved global plan by voting about joint actions. They even propose algorithms to deal with insincere agents and to interleave planning, coordination, and execution (Ephrati and Rosenschein, 1995).

An approach that considers both conflicts and positive relations is proposed by (von Martial, 1989, 1990, 1992). He presents plans hierarchically, and the top level needs to be exchanged among the agents to determine such relations. If possible, relations are solved or exploited at this top level. If not, a refinement of the plans is made, and the process is repeated. For each specific type of plan relationship, a different solution is presented. Relations between the plans of autonomous agents are categorized. The main aspects are positive/negative relations, (non) consumable resources, requests, and favor relationships.

Recently, (Tsamardinos et al., 2000) succeeded in developing a plan merging algorithm that deals with both durative actions and time. They construct a conditional simple temporal network to specify (temporal) conflicts between plans. Based on this specification, a set of constraints is derived that can be solved by a constraint solver. The solution specifies the required temporal relations between actions in the merged plan. One of the problems with the plan merging approaches described above is that one agent may become dependent on another, while this was in the beginning not the case at all.

Finally, (Cox and Durfee, 2003) describe how to maintain the autonomy, while still being able to use results from other agents to improve the efficiency. Basically, their idea is to add these dependencies conditionally to the plan: if the other agent succeeds, this more efficient branch of the plan can be executed; otherwise the normal course of action can still be followed.

Plan execution

We consider the sixth phase to be of a slightly different order and a bit off-topic. (This in fact includes a vast body of work such as on reactive agents and behavior models.)

4 Multi-agent planning systems

In this section, we discuss three approaches to planning and coordination from the multi-agent literature in more detail — one approach in which coordination occurs prior to planning (phase 3), one in which planning and coordination are interleaved (phase 4), and one in which agents first make their own plans, and then perform *plan merging* (phase 5).

4.1 Coordination through filtering

(Ephrati et al., 1995b) distinguish two approaches to multi-agent coordination. *Explicit coordination* involves agents reasoning about their interactions and negotiations. A problem with explicit coordination is that it can be extremely time-consuming, which can be impractical in dynamic domains. With the second approach, *implicit coordination*, agents follows 'local rules of behaviour' that ensure that agents can operate without having to worry about interference from other agents. Social laws are an example of implicit coordination.

Ephrati's paper deals with implicit coordination in the form of *multi-agent filtering*. Multi-agent filtering is an extension of single-agent filtering, which was a strategy designed for agents in dynamic environments. An agent using a single-agent filtering strategy has set a set of goals. Due to changes in the environment, opportunities arise to take alternative or additional action. A filtering strategy filters out those *options* that are incompatible with the agent's current goal. A multi-agent filtering strategy bypasses options that are incompatible with the goals of *other* agents. Ephrati et al. ask themselves the question whether rational agents should employ a filtering strategy,

since bypassing options for the sake of other agents effectively reduces the number of possible actions for the filtering agent.

Filtering strategies can be augmented with an override mechanism. If an option interferes with its own or other agents' goals, but it looks particularly promising, then the option can be taken into deliberation, the same as non-conflicting options. The override mechanisms is based on a threshold value. A *bold* agent will not consider many new options (he will have high threshold), a *cautious* agent will use a low threshold value.

The notion of *interference* in this paper differs from the 'standard' notion of interference that e.g. (Ephrati and Rosenschein, 1993) use, where for instance the preconditions for one action must not be undone by the post-conditions of another action. Instead, the authors identify a single type of conflict situation in their example domain, the multi-agent tileworld domain. In the tileworld domain, there is a map (grid) that is littered with tiles. Objects (holes, tiles, obstacles, etc.) appear and disappear dynamically and agents receive payment for filling holes with tiles. Conflict is defined as two agents trying to fill the same hole.

For this domain, several filtering strategies are presented. One filtering strategy, *Static geographic filtering*, is based on the location of the agents and the location of the holes. Agents are assigned non-overlapping portions of the map and they filter out any options of fillings holes that are not in their region. Using the filtering strategy *Intention posting*, agents must post on a blackboard their intention to fill a hole. Agents bypass options of filling holes that another agent intends to fill.

To answer the question of whether employing a filtering strategy is rational, experiments where run where a number of agents, from one to all, employ a filtering strategy (static geographic) and all others do not. Using filtering proved the dominant strategy, because whatever the number of agent using filtering, the agents using filtering had, on average, the highest utility. Ephrati et al. do not, however, give any evidence that this conclusion can be generalized to other domains. Indeed, even their claim that filtering is the dominant strategy in the tileworld domain is insufficiently supported, since they do not precisely specify the strategy used by agents *not* using a filtering strategy.

4.2 Generalized Partial Global Planning

The Partial Global Planning (PGP) (Durfee, 1991) framework is perhaps one of the most influential approaches in distributed artificial intelligence. In the PGP framework, agents cooperate because no agent has complete information. PGP assumes a cooperative distributed problem solving environment (CDPS, these days referred to as simply DPS, which contrasts with multi-agent systems (MAS), where agents are self-interested), where agents are willing to help each other without any compensation. For CDPS environments, Durfee and Lesser identify four categories of coordination techniques, all of which are encompassed in PGP.

1. *Contracting:* The distributed problem solving process is viewed as one big process and many potential solvers, such as in parallel computing. The goal of coordination is to utilize the problem solvers to the utmost. In case agents can execute their tasks independently, then contracting (which redistributes tasks) always has positive utility.

2. *Result-sharing:* Result-sharing concentrates on domains where tasks are inherently distributed, but the problems arising for one agent may be related to problems arising for other agents.

3. *Organizing:* Organizational knowledge about e.g. agents' roles and responsibilities can help agents decide *what* information to communicate to *which* agents.

4. *Planning:* Traditional planning in distributed artificial intelligence has focused on avoiding resource conflicts. If agents can act independently, the focus is on cooperation.

PGP is geared towards a particular kind of multi-agent domain, that of distributed sensor networks. In their paper, a distributed network of acoustic sensors monitoring vehicle movement is used as the running example. The goal of this network is to provide a consistent view of vehicle movements. In order to do so, agents must interpret their sensor data. Because there is too much data (in particular, too much noise), exhaustively analyzing the sensor inputs is impractical. Fortunately, interrelationships between the data of other agents mean that by sharing information, it is possible to interpret the data accurately and timely. More specifically, coordination techniques for distributed sensor networks must allow agents to:

- instruct (or propose) other agents to collect specific data (for instance monitor a certain road),
- determine which information to send to whom and when,
- cooperate in other ways. For instance, an agent who has no data of himself to interpret can be put to work analyzing data of other agents.

Dynamic sensor networks are highly dynamic environments and different PGP coordination techniques are used in different circumstances. If, however, we pretend for a moment that the environment is static, then the following four coordination techniques are used subsequently:

1. *Local planning:* An agent makes very rough characterizations about all of the possible interpretations of its dataset, and makes tentative plans for all these interpretations. A plan represents future actions at two levels of abstraction. At the high level of abstraction, it outlines the major steps; at the low level of abstraction, it details the actions for the next major step.

2. *Communication with other agents:* To know what information to send to whom and when, PGP employs two types of organizations. The *task-level* organization defines the roles and responsibilities of agents with regard to performing tasks. The *meta-level* organization defines authority roles, that is, some agents may give orders to other agents.

3. *Initializing a Partial Global Plan:* To integrate the plans from other agents with his own plan, an agent will try to relate the goals of one plan with the goals of another agent. Goals can be related in various ways. An example of a goal relation in the vehicle monitoring domain is if different agents each monitor a different stretch of track, such that all stretches belong to the same road.

4. *Modifying PGPs:* If agents have received or constructed a partial global plan, they can try to improve it using techniques such as task re-distribution and task re-ordering. By sending such a (modified) partial global plan to other agents, the agent effectively proposes this plan to other agents. Other agents may accept the roles the first agent has outlined for them in the PGP, they may refuse, or they may send a counterproposal in the form of a modified partial global plan.

In Generalized Partial Global Planning (GPGP) (Decker and Lesser, 1994), PGP is extended by defining a task-oriented framework (described in (Decker and Lesser, 1993)) which allows coordination mechanisms to be 'inserted' into the framework. In this way, the set of coordination techniques used in the PGP framework for distributed vehicle monitoring is merely one configuration of a framework that defines a 'family' of coordination algorithms that are not tied to a single domain.

In the TAEMS (Task Analysis, Environment Modeling, and Simulation) framework, tasks can be composed of subtasks, forming a hierarchy with one root node called the *task group*. Multiple task groups can exist at the same time. The other relationships between tasks are *enables* (a task can only be started if the enabling task has completed), *facilitates* (that is, a positive relation) and *hinders* (a negative relation). For evaluating performance of execution of a (set of) task(s), two characteristics are relevant, the *elapsed time* and the *quality of task execution*, which is meant as a performance measure that can be instantiated for specific domains. An agent has *belief*, which means that the agent believes the part of the global task structure that he can see. Agents can commit themselves to performing tasks for other agents.

Each agent has a local scheduler that schedules the *computational resources* of agent, i.e., the scheduler determines which tasks an agent will execute and when. The purpose of the coordination mechanisms is to ensure that the local scheduler is provided with the best possible input that allows construction of a high utility schedule. In other words, coordination mechanisms are used to enable an agent to make a good plan.

4.3 Plan merging using a resource formalism

Starting point in (de Weerdt et al., 2003) is that there are two agents that have plans that can be executed without taking into account the plan of the other agent. In other words, the agents can perform their plans without need for (further) coordination. However, by cooperating the agents can find more efficient plans.

A *resource logic* is used to represent plans and actions. The goal situation is to have a set of resources of a particular *resource type*. The initial situation is also a set of resources. Agents can perform actions, called *skills*, that consume resources and produce resources. A plan can be represented as a directed acyclic graph where the vertices are resources and skills and the arcs connect resources with skills, to indicate either a consumption or production relationship (for example, see Figure 5 where a skill s_1 consumes one resource of type *a* and produces two resources of respectively type *b* and *c*).

A plan may produce more resources than required if (i) the skills produce resources that are not used by subsequent skills in the plan and those resources are not required for the goal, or if (ii) such resources are present in the initial situation. These unused

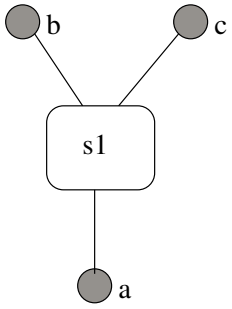

Figure 5: A plan: skill s_1 consumes a resource of type 'a' and produces resources of types 'b' and 'c'.

resources can be considered *side effects* of a plan. Other agents may be able to use these unused resources. Although every agent already has all the resources he needs for his plan, by buying resources from other agents, he does not need to produce them himself. More precisely, if an agent can buy all the useful (i.e., not unused) resources that a particular skill produces, then that skill can be removed from the plan. The removal of a skill in turn frees up the resources that were previously consumed by the deleted skill, which opens up possibilities for further trading of resources.

This work does not go into the details of the negotiation between agents, because if agents cooperate (i.e., successfully trade resources), utility can only go up. To the selling agent, the resources are worthless if he does not sell them and, assuming rational agents, the buying agent will only buy resources if this leads to a reduction in cost.

5 Future work

Many of the accomplishments of multi-agent research, though fine within their intended settings, fail to impress when applied 'real-world' problems. That is, the simplifying assumptions that underlie these approaches often prove too restrictive for practical use. A number of these simplifying assumptions include (DesJardins et al., 2000):

1. **The environment is fairly stable.** In the early days of AI planning research, the world was assumed to be static, meaning that changes to the state of the world are always the result of actions performed by the agent. Of course, researchers have long since recognized the importance of allowing unexpected changes to occur in the environment, yet many research still relies on the assumption of a (semi-) static environment.

2. **The world is deterministic.** We assume that we know the result of each action. Unfortunately, especially in a multi-agent environment, this is not the case, because for example another agent may have changed the world after the precondition of an action has been established. However, under the assumption that

all agents' actions are coordinated, this deterministic assumption is quite acceptable.

3. **There is a fair degree of coherence.** Either the agents are designed to work together or they are rational and have an incentive to do so. In other words, agents will try to maximize their expected utility (Zlotkin and Rosenschein, 1996).

4. **Knowledge about the world is correct** and consistent among all agents. In other words, the (relevant part of the) world is completely observable.

5. **A feasible goal state exists** in which all global goals are achieved, and all private goals are also met (at least to some degree, such as in "make a lot of money").

6. **Learning is not required.** In other words, (past) events do not affect the agents other than a change of the current state.

7. **Communication is reliable and (almost) free.** All messages come across safely, and the agents share a common ontology and utility units. Furthermore, there is no significant cost associated with communication actions. Most of these assumptions are commonly used, and, fortunately, they are acceptable in many application domains.

To relax the first assumption, concerning changes in the environment, Desjardins et al. propose to introduce a new multi-agent planning paradigm: *Distributed Continual Planning* (DCP). A traditional approach to handling uncertainty is to plan for all the *contingencies* that might arise. If we plan for both the case that the contingency arises and for the case that it does not, then obtain a conditional plan that branches for each of the possible contigencies. It is not hard to see that if the number of possible contigencies becomes very large, then the conditional plan will become unmanageably large. Another approach is therefore to have a single plan and *monitor* its execution. As soon as a deviation from the plan is detected, some *plan repair* must be done to ensure that the goals will still be reached.

Distributed Continual Planning aims to go beyond replanning on failure, or contingency planning, by viewing planning as an ongoing, dynamic process in which planning and execution are interleaved. Continual planning should not only react to changes that threaten the execution of the plan, but also look for opportunities to improve the plan. This might mean for instance that if the goal of a plan has become obsolete yet the plan is still feasible, then the agent should abandon the current plan and search for a better use of its resources. Desjardins et al. state that an agent should engage in continual planning when (i) aspects of the world changes dynamically beyond control of the agent, when (ii) aspects of the world are revealed incrementally, when (iii) time pressure requires an agent to start the execution of his plan before it is complete, or when (iv) the agent's objectives can evolve over time.

References

Allen, J. F., Hendler, J., and Tate, A., editors (1990). *Readings in Planning*. Morgan Kaufmann Publishers, San Mateo, CA.

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2000). The complexity of decentralized control of markov decision processes. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 32–37, San Mateo, CA. Morgan Kaufmann Publishers.

Blum, A. L. and Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300.

Bond, A. H. and Gasser, L., editors (1988). *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA.

Bonet, B. and Geffner, H. (2002). Planning as heuristic search. *Artificial Intelligence*, 129:5–33. Special issue on Heuristic Search.

Boutilier, C., Dean, T. L., and Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of AI Research*, 11:1–94.

Briggs, W. (1996). *Modularity and Communication in Multi-Agent Planning*. PhD thesis, University of Texas at Arlington.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204.

Clearwater, S. (1996). *Market-Base Control – a paradigm for distributed resource allocation*. World Scientific Publishing Co.

Clement, B. J. and Barrett, A. C. (2003). Continual coordination through shared activities. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-03)*.

Cox, J. S. and Durfee, E. H. (2003). Exploiting synergy while managing agent autonomy. In *AAMAS '03 Workshop on Autonomy, Delegation and Control*.

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.

Decker, K. and Lesser, V. R. (1993). Quantitative modeling of complex computational task environmnents. In *Proceedings of the 12th international workshop on distributed artificial intelligence*, pages 67–82, Hidden Valley, Pennsylvania.

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.

Decker, K. S. and Lesser, V. R. (1994). Designing a family of coordination algorithms. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence (DAI-94)*, pages 65–84.

Decker, K. S. and Li, J. (2000). Coordinating mutually exclusive resources using gpgp. *Autonomous Agents and Multi-Agent Systems*, 3(2):113–157.

DesJardins, M. E., Durfee, E. H., Ortiz, C. L., and Wolverton, M. J. (2000). A survey of research in distributed, continual planning. *AI Magazine*, 20(4):13–22.

Do, M. and Kambhampati, S. (2001). Sapa: A domain-independent heuristic metric temporal planner. In *Proceedings of the Sixth European Conference on Planning (ECP-01)*, pages 109–120.

Durfee, E. H. (1991). Organizations, plans, and schedules: An interdisciplinary perspective on coordinating AI agents. *Journal of Intelligent Systems*. Special Issue on the Social Context of Intelligent Systems.

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

Durfee, E. H. (2001). Scaling up agent coordination strategies. *Computer*.

Durfee, E. H. and Lesser, V. R. (1987). Planning coordinated actions in dynamic domains. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, pages 18.1–18.10.

Ephrati, E., Pollack, M., and Rosenschein, J. S. (1995a). A tractable heuristic that maximizes global utility through local plan combination. In Lesser, V., editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 94–101, San Francisco, CA. AAAI Press, distributed by The MIT Press.

Ephrati, E., Pollack, M. E., and Ur, S. (1995b). Deriving multi-agent coordination through filtering strategies. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 679–687, San Mateo, CA. Morgan Kaufmann Publishers.

Ephrati, E. and Rosenschein, J. S. (1993). Multi-agent planning as the process of merging distributed sub-plans. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (DAI-93)*, pages 115–129.

Ephrati, E. and Rosenschein, J. S. (1994). Divide and conquer in multi–agent planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 375–380, Menlo Park, CA. AAAI Press.

Ephrati, E. and Rosenschein, J. S. (1995). A framework for the interleaving of execution and planning for dynamic tasks by multiple agents. In Castelfranchi, C. and Müller, J., editors, *From Reaction to Cognition — Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-93), LNAI Volume 957*, pages 139–156, Berlin. Springer Verlag.

Erol, K., Hendler, J., and Nau, D. S. (1994). HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1123–1128, Seattle, Washington, USA. AAAI Press/MIT Press.

Ferber, J. and Drogoul, A. (1992). Using reactive multi-agent systems in simulation and problem solving. In Avouris, N. M. and Gasser, L., editors, *Distributed Artificial Intelligence: Theory and Praxis*, volume 5 of *Euro Courses: Computer and Information Science*, pages 53–80. Kluwer Academic, The Netherlands.

Fikes, R. E. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5(2):189–208.

Fischer, K., Ruß, C., and Vierke, G. (1998). Decision theory and coordination in multiagent systems. Technical Report RR-98-02, DFKI GmbH: German Research Center for Artificial Intelligence.

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

Garey, M. and Johnson, D. (1979). *Computers and intractability – a guide to the theory of NP-completeness*. W.H. Freeman and company, New York, NY.

Georgeff, M. P. (1983). Communication and interaction in multi-agent planning. In *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, pages 125–129, Menlo Park, CA. AAAI Press. Also published in Bond and Gasser (1988), pages 200–204.

Georgeff, M. P. (1984). A theory of action for multiagent planning. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pages 121–125, Menlo Park, CA. AAAI Press. Also published in Bond and Gasser (1988), pages 205–209.

Georgeff, M. P. (1988). Communication and interaction in multi-agent planning. In Bond, A. and Gasser, L., editors, *Readings in Distributed Artificial Intelligence*, pages 200–204. Morgan Kaufmann Publishers, San Mateo, CA. Also published as Georgeff (1983).

Haddawy, P. and Hanks, S. (1998). Utility models for goal-directed, decision-theoretic planners. *Computational Intelligence*, 14(3):392–429.

Haslum, P. and Jonsson, P. (1999). Some results on the complexity of planning with incomplete information. In *Proceedings of the Fifth European Conference on Planning (ECP-99)*, volume 1809 of *Lecture Notes on Artificial Intelligence*, pages 308–318, Berlin. Springer Verlag.

Hoffmann, J. and Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of AI Research*, 14:253–302.

Janlert, L.-E. (1987). Modeling change—the frame problem. In Pylyshyn, Z., editor, *The Robot's Dilemma: The Frame Problem in Artificial Intelligence*, pages 1–40. Ablex Publishing Co., Norwood, NJ.

Jennings, N. R. (1996). Coordination techniques for artificial intelligence. In O'Hare, G. and Jennings, N., editors, *Foundations of Distributed Artificial Intelligence*, pages 187–210. John Wiley & Sons, New York, NY.

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

Katz, M. J. and Rosenschein, J. S. (1989). Plans for multiple agents. In Gasser, L. and Huhns, M. N., editors, *Distributed Artificial Intelligence*, volume 2 of *Research Notes in Artificial Intelligence*, pages 197–228. Pitman Publishing and Morgan Kaufmann Publishers, London, UK.

Koehler, J. (1998). Planning under resource constraints. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence (ECAI-98)*, pages 489–493. John Wiley & Sons.

Lesser, V., Decker, K., Carver, N., Garvey, A., Neimen, D., Prassad, M., and Wagner, T. (1998). Evolution of the gpgp domain independent coordination framework. Technical Report UMASS CS TR 1998-005, University of Massachusetts.

Lifschitz, V. (1987). On the semantics of STRIPS. In *Reasoning About Actions and Plans — Proceedings of the 1986 Workshop*, pages 1–9, San Mateo, CA. Morgan Kaufmann Publishers.

Malone, T. W. and Crowston, K. (1991). Toward an interdisciplinary study of coordination. Center for Coordination Science, MIT.

Malone, T. W. and Crowston, K. (1993). The interdisciplinary study of coordination. *ACM computing surveys*.

Nebel, B. (2000). On the compilability and expressive power of propositional planning formalisms. *Journal of AI Research*, 12:271–315.

Nwana, H. S., Lee, L., and Jennings, N. R. (1996). Coordination in software agent systems. *BT Technology Journal*, 14(4):79–88.

Pednault, E. P. (1987). Formulating multi-agent dynamic-world problems in the classical planning framework. In Georgeff, M. P. and Lansky, A. L., editors, *Reasoning About Actions and Plans — Proceedings of the 1986 Workshop*, pages 47–82, San Mateo, CA. Morgan Kaufmann Publishers. Also published in Allen et al. (1990).

Penberthy, J. and Weld, D. S. (1992). UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning (KR&R-92)*, pages 103–114. Morgan Kaufmann Publishers.

Penberthy, J. and Weld, D. S. (1994). Temporal planning with continuous change. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1010–1015, Menlo Park, CA. AAAI Press.

Pynadath, D. and Tambe, M. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of AI Research*, 16:389–423.

Raphael, B. (1971). The frame problem in problem-solving systems. In Findler, N. and Meltzer, B., editors, *Proceedings of the Advanced Study Institute on Artificial Intelligence and Heuristic Programming*, pages 159–169, Edinburgh, UK. Edinburgh University Press.

Refanidis, I. and Vlahavas, I. (2001). The GRT planner: Backward heuristic construction in forward state-space planning. *Journal of AI Research*, 15:115–161.

Rosenschein, J. S. (1995). Multiagent planning as a social process: Voting, privacy, and manipulation. In Lesser, V. R., editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, page 431, San Francisco, CA. AAAI Press, distributed by The MIT Press.

Sacerdoti, E. D. (1975). The nonlinear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, pages 206–214, San Mateo, CA. Morgan Kaufmann Publishers.

Sandholm, T. W. and Lesser, V. R. (1997). Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137.

Shehory, O. and Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200.

Shoham, Y. and Tennenholtz, M. (1995). On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1–2):231–252.

Smith, D. E. and Weld, D. S. (1999). Temporal planning with mutual exclusion reasoning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, San Mateo, CA. Morgan Kaufmann Publishers.

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)*, pages 1031–1033, San Mateo, CA. Morgan Kaufmann Publishers. Also published in Bond and Gasser (1988), pages 216–219.

ter Mors, A., Valk, J., and Witteveen, C. (2004). Coordinating autonomous planners. In *Proceedings of the international conference on artificial intelligence*, pages 795–801. CSREA Press.

Thangarajah, J., Padhgam, L., and Winikoff, M. (2003). Detecting and avoiding interference between goals in intelligent agents. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*.

Tsamardinos, I., Pollack, M. E., and Horty, J. F. (2000). Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS-00)*, pages 264–272, Menlo Park, CA. AAAI Press.

Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E., and Blythe, J. (1995). Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):81–120.

Vickrey, W. (1961). Computer speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37.

von Martial, F. (1989). Multiagent plan relationships. In *Proceedings of the Ninth International Workshop on Distributed Artificial Intelligence (DAI-89)*, pages 59–72.

von Martial, F. (1990). Coordination of plans in multiagent worlds by taking advantage of the favor relation. In Huhns, M., editor, *Proceedings of the Tenth International Workshop on Distributed Artificial Intelligence (DAI-90)*, number ACT-AI-355-90 in MCC Technical Report, Austin, TX.

von Martial, F. (1992). *Coordinating Plans of Autonomous Agents*, volume 610 of *Lecture Notes on Artificial Intelligence*. Springer Verlag, Berlin.

Walsh, W. E. and Wellman, M. P. (1999). A market protocol for decentralized task allocation and scheduling with hierarchical dependencies. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS-98)*, pages 325–332. An extended version of this paper is also available.

Walsh, W. E., Wellman, M. P., and Ygge, F. (2000). Combinatorial auctions for supply chain formation. In *Second ACM Conference on Electronic Commerce*, pages 260–269. ACM Press.

Weld, D. S. (1994). An introduction to least-commitment planning. *AI Magazine*, 15(4):27–61.

Weld, D. S. (1999). Recent advances in AI planning. *AI Magazine*, 20(2):93–123.

Wellman, M. P. (1993). A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23.

Wellman, M. P., Walsh, W. E., Wurman, P., and MacKie-Mason, J. (1998). Auction protocols for decentralized scheduling. In *Proceedings of the Eighteenth International Conference on Distributed Computing Systems*.

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.

Wolfman, S. A. and Weld, D. S. (2001). Combining linear programming and satisfiability solving for resource planning. *Knowledge Engineering Review*, 16(1):85–99.

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

Zlotkin, G. and Rosenschein, J. S. (1996). Mechanisms for automated negotiation in state oriented domains. *Journal of Artificial Intelligence Research*, 5:163–238.