ELSEVIER

Artificial Intelligence 127 (2001) 199–220

Artificial Intelligence

www.elsevier.com/locate/artint

Evaluating new options in the context of existing plans

John F. Horty [a,*], Martha E. Pollack [b]

[a] *Philosophy Department and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA*

[b] *Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA*

Received 6 September 1999; received in revised form 13 June 2000

Abstract

This paper contributes to the foundations of a theory of rational choice for artificial agents in dynamic environments. Our work is developed within a theoretical framework, originally due to Bratman, that models resource-bounded agents as operating against the background of some current set of intentions, which helps to frame their subsequent reasoning. In contrast to the standard theory of rational choice, where options are evaluated in isolation, we therefore provide an analysis of situations in which the options presented to an agent are evaluated against a background context provided by the agent's current plans—commitments to future activities, which may themselves be only partially specified. The interactions between the new options and the background context can complicate the task of evaluating the option, rendering it either more or less desirable in context than it would have been in isolation. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Agents; Plan management; Planning; Rationality

1. Introduction

This paper contributes to the foundations of a theory of rational choice for artificial agents in dynamic environments. As usually formulated in the economic and philosophical literature [11,23], the theory of rational choice assumes that an agent evaluates alternative actions by reference to a probability distribution over their possible outcomes together with a utility function defined on those outcomes, which are both taken as part of the background

* Corresponding author.
E-mail addresses: horty@umiacs.umd.edu (J.F. Horty), pollackm@eecs.umich.edu (M.E. Pollack).

0004-3702/01/$ – see front matter © 2001 Elsevier Science B.V. All rights reserved.
PII: S0004-3702(01)00060-1

setting and assumed to be immediately apparent to the agent. In the simplest case, the agent combines probability and utility into a notion of expected utility defined over actions, and then chooses some action whose expected utility is maximal. The theory developed here differs in two important ways from this standard approach.

First, while the standard theory of rational choice takes the utility of an outcome as part of the background setting, we note that the overall desirability of an option presented to an agent is often not immediately apparent; and we are explicitly concerned with the mechanism through which it might be discovered by the agent. We focus, in particular, on the case in which the option presented to an agent has a known benefit, but requires some effort—the execution of a plan—for its achievement. In order to evaluate the overall desirability of the option, the agent thus has to arrive at an assessment of the cost involved in achieving it.

Second, we insist that the task of evaluating an option should be computationally realizable; and in particular, our work here is developed within a theoretical framework first articulated in [3], and then further elaborated in [4,19], that models resource-bounded agents as operating always against the background of some current set of intentions, or plans, which helps to frame their subsequent reasoning. In contrast to the standard theory of rational choice, where actions are evaluated in isolation, we therefore develop a model in which the options presented to an agent are evaluated against a background context provided by the agent's current plans—commitments to future activities, which, at any given point, may themselves be only partially specified. The interactions between the new option and the background context can complicate the task of evaluating the option, rendering it either more or less desirable in context than it would have been in isolation.

As an example, suppose an agent is already committed to going to the airport tomorrow afternoon to catch a plane, but has not yet decided whether to get there by taxi or by taking the airport shuttle van. Given this background context, the agent might then have to evaluate the newly presented option of attending a lunch time meeting tomorrow. If the meeting is to be held on campus, and is likely to run late, a decision to attend may rule out the possibility of taking the van. Assuming that the van costs less than the taxi, the new option would then be less desirable in context than it would have been in isolation; the benefit of attending the meeting must be at least great enough to compensate for the difference in cost between taxi and van to make the new option worthwhile. On the other hand, suppose the meeting is to be held at an airport hotel. In this case, the background context reduces the cost associated with the new option, increasing its overall desirability, since the agent is already committed to going to the airport: the agent might rationally choose to attend the meeting in this context, since it is going to the airport anyway, even if this option is not one that the agent would have decided to pursue in isolation.

Although this situation might appear to be trivial, there are two reasons for trying to understand the reasoning processes involved. The first is as an exercise in cognitive science: the ability to manage one's activities in a dynamic environment—which crucially involves the evaluation of new options—is an important component of human cognitive capacity, and thus worth exploring as part of the effort to understand the human mind. The second is as a prerequisite to designing better computational tools. It is generally agreed that

intelligent autonomous agents would be extremely useful in applications ranging from military to industrial to educational. But for many of these applications, autonomous agents would need to be able to perform the kind of option evaluation illustrated by our example.

The work presented here begins the task of providing a theoretical and computational analysis of the reasoning involved in these situations, where a new option must be evaluated within the context of a background plan, or set of plans, to which the agent is already committed. While we make no commitments as to how such plans are generated, we do restrict our attention in the present paper to plans that are primitive (not hierarchical) and complete, and in which all actions have deterministic outcomes. In this simple setting, the only ways in which one plan can influence the cost of another is by allowing or blocking the possibility that separate steps might be merged into one. (In the airport story, for instance, when the meeting is held at the airport, the step of getting to the meeting can be merged with the step of getting to the airport, which is already part of the agent's background plan.) Although our restriction to this special case prevents us from considering many of the more interesting ways in which plans might interact, even this very simple setting is sufficiently rich to allow us to illustrate the shape of our theory, and we defer a detailed treatment of more complicated plan interactions to subsequent work.

2. Basic concepts

We represent primitive plans using a standard formalism [15,18,27], in which a plan consist of a set of steps, temporal constraints on those steps, and causal links, which record dependency relations among steps. As usual, we assume a set of action types, each with associated preconditions and effects; for clarity, we limit our attention only to propositional preconditions and effects. The plan steps are instances of the action types. Much of the planning literature tends to concentrate on qualitative temporal constraints, which specify only the relative order of steps (but see [1,7] for some notable exceptions). In our approach, we allow also for quantitative temporal constraints, which associate steps with actual time points. To this end, we model time as a totally ordered set of moments $\{m_0, m_1, \ldots\}$, with $m_i < m_j$ just in case $i < j$, and we assume here that each plan step occupies a single moment of time.

Definition 1 (*Primitive plan*). A *primitive plan* $\mathcal{P}$ is a triple of the form $\langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$, with these components defined as follows: $\mathcal{S}$ is a set of steps of the form S_i, each associated with a time indicator t_i; $\mathcal{O}$ is a set of ordering constraints, of the form $t_i = t_j$, $t_i < t_j$, $t_i = m_k$, or $t_i < m_k$, where t_i and t_j are time indicators associated with steps belonging to $\mathcal{S}$ and m_k is a moment; $\mathcal{L}$ is a set of causal links of the form $\langle S_i, Q, S_j \rangle$, where Q is an effect of the step S_i and a precondition of the step S_j.

We assume a function *type* associating each step S_i with $type(S_i)$, its action type. We require $\mathcal{O}$ to contain a temporal constraint of the form $t_i < t_j$ whenever there is a link $\langle S_i, Q, S_j \rangle$ in $\mathcal{L}$. And we suppose that an entailment relation $\vdash$ is defined on the temporal constraint language, allowing us to draw out implicit consequences (for example,

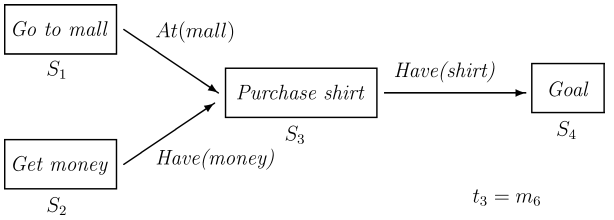

Fig. 1. A sample plan: buying a shirt.

$\{t_j = m, t_i < t_j\} \vdash t_i < m$), and providing us, also, with a notion of consistency for a set of temporal constraints.

To illustrate, let us consider the plan, depicted in Fig. 1, of buying a shirt at the mall precisely at the moment m_6. In the current framework, this plan would be represented as $\mathcal{P}_1 = \langle \mathcal{S}_1, \mathcal{O}_1, \mathcal{L}_1 \rangle$, where $\mathcal{S}_1 = \{S_1, S_2, S_3, S_4\}$, $\mathcal{O}_1 = \{t_1 < t_3, t_2 < t_3, t_3 < t_4, t_3 = m_6\}$, and $\mathcal{L}_1 = \{\langle S_1, At(mall), S_3\rangle, \langle S_2, Have(money), S_3\rangle, \langle S_3, Have(shirt), S_4\rangle\}$. The types, effects, and preconditions of the various steps from $\mathcal{S}_1$ are as depicted. Thus, S_1 represents an action of going to the mall, with the proposition *At*(*mall*) as its effect; S_2 is the action of getting money (which we suppose can be performed either before going to the mall, or at the mall, using an cash machine), with *Have*(*money*) as its effect.[1] The step S_3 represents the action of actually buying the shirt; it requires *At*(*mall*) and *Have*(*money*) as preconditions and generates *Have*(*shirt*) as an effect. Finally, S_4 is a dummy step representing the achievement of the goal, taking *Have*(*shirt*) as its precondition and generating no effects. The plan includes four ordering constraints. The first three are qualitative constraints derived from the causal links and are implicit in the plan graph. The fourth is a quantitative constraint specifying that S_3 must be performed precisely at m_6—the moment, perhaps, that the shirt goes on sale. This constraint is explicitly shown in the figure.[2]

We will say that a plan is scheduled when each of its steps has been assigned a specific moment of execution. In this paper, we adopt the simplifying assumption that different actions cannot be performed at the same time. We thus prohibit schedules with concurrent actions, although, importantly, two steps of the same action type can be *merged*—assigned to the same moment of execution.

[1] In this example, we are treating *Have*(*money*) simply as a proposition, which can be true or false, not as representing access to a consumable resource. Modeling consumable resources in an interesting problem, but one that is orthogonal to the problem explored here; we believe our theory could integrate many alternative models of resources that might be independently developed.

[2] Most standard plan representations also include an "initial state", specifying the propositions that are true or false at the beginning of the plan. Since our research falls within the broad area of dynamic planning, where propositions can become true at various points in time, we omit the notion of a single initial state as unrealistic. In general, of course, plan steps must be linked to the truth of their preconditions in a dynamic plan, but rather than spelling this matter out in detail, we make due with the following assumption governing the plans displayed here: the preconditions of those steps in a plan that have no establishing actions are true at the time the steps are performed. Note that this allows a more flexible representation than the standard treatment, since not all "initial conditions" must be true at once, and indeed, some may not even become true until after execution of the plan has begun.

Definition 2 (*Schedule, scheduled and schedulable plans*). A *schedule* for a plan $\mathcal{P} = \langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$ is a set of constraints $\overline{\mathcal{O}}$ such that:

(1) there is a constraint of the form $t_i = m$ in $\overline{\mathcal{O}}$ for each S_i in $\mathcal{S}$;
(2) $\mathcal{O} \cup \overline{\mathcal{O}}$ is consistent; and
(3) $type(S_i) = type(S_j)$ whenever $\mathcal{O} \cup \overline{\mathcal{O}} \vdash t_i = t_j$.

The plan $\mathcal{P}$ is said to be *scheduled* whenever there exists a set of constraints $\overline{\mathcal{O}} \subseteq \mathcal{O}$ such that $\overline{\mathcal{O}}$ is a schedule for the plan; $\mathcal{P}$ is said to be *schedulable* whenever there exists a schedule for it.

As an example, the constraint set $\overline{\mathcal{O}} = \{t_1 = m_3, t_2 = m_4, t_3 = m_6, t_4 = m_7\}$ is a schedule for the plan $\mathcal{P}_1$ for getting a shirt at the mall, showing that this plan is schedulable. Of course, a plan whose ordering constraints are themselves inconsistent cannot be scheduled, but even a plan whose ordering constraints are consistent may nevertheless fail to be schedulable, since its only consistent linearizations may be those in which type distinct steps are assigned to the same moment. Schedulability is thus a stronger requirement than mere consistency of temporal constraints.

We focus in this paper on plans that are complete, in the sense that no further planning is needed in order to guarantee the preconditions of their various steps, although additional scheduling may still be required.

Definition 3 (*Complete plans*). Let $\mathcal{P} = \langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$ be a plan. A precondition A of a step S_i from $\mathcal{S}$ is *established* whenever there is some link $\langle S_j, A, S_i \rangle$ in $\mathcal{L}$. A link $\langle S_j, A, S_i \rangle$ from $\mathcal{L}$ is *threatened* whenever there is both an action S_k in $\mathcal{S}$ with effect $\neg A$ and a schedule $\overline{\mathcal{O}}$ for $\mathcal{P}$ such that $\mathcal{O} \cup \overline{\mathcal{O}} \vdash t_j < t_k < t_i$. The plan $\mathcal{P}$ is *complete* just in case each precondition of each step from $\mathcal{S}$ is established and no link from $\mathcal{L}$ is threatened.

This definition of plan completeness is equivalent to the standard notion from the literature, except that it replaces the idea of temporal consistency with the stronger notion of schedulability.

In order to assess the desirability of a new option against a background context, we need to be able to reason about the plans that are formed when two others are combined, as follows.

Definition 4 (*Union of plans*). Given plans $\mathcal{P} = \langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$ and $\mathcal{P}' = \langle \mathcal{S}', \mathcal{O}', \mathcal{L}' \rangle$, the *union* of the two plans is $\mathcal{P} \cup \mathcal{P}' = \langle \mathcal{S} \cup \mathcal{S}', \mathcal{O} \cup \mathcal{O}', \mathcal{L} \cup \mathcal{L}' \rangle$.

Note that the union of two independently schedulable plans might not be schedulable, since their temporal constraint sets may not even be jointly consistent; also, the union of two complete plans might not be complete, since steps in one may threaten links in the other. If the union of two complete plans can be made complete and schedulable simply through the addition of ordering constraints, we say that the plans are strongly compatible.

Definition 5 (*Strong compatibility*). Let $\mathcal{P} = \langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$ and $\mathcal{P}' = \langle \mathcal{S}', \mathcal{O}', \mathcal{L}' \rangle$ be complete plans. Then $\mathcal{P}$ and $\mathcal{P}'$ are *strongly compatible* just in case there is a temporal constraint set $\mathcal{O}''$ such that $\langle \mathcal{S} \cup \mathcal{S}', \mathcal{O} \cup \mathcal{O}' \cup \mathcal{O}'', \mathcal{L} \cup \mathcal{L}' \rangle$ is complete and schedulable.

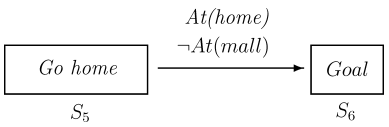

Fig. 2. A strongly compatible plan.

As an example, consider the plan of going home, depicted in Fig. 2, and represented as $\mathcal{P}_2 = \langle \mathcal{S}_2, \mathcal{O}_2, \mathcal{L}_2 \rangle$, where $\mathcal{S}_2 = \{S_5, S_6\}$, $\mathcal{O}_2 = \{t_5 < t_6\}$, and $\mathcal{L}_2 = \{\langle S_5, At(\textit{home}), S_6 \rangle\}$; the step S_5, representing the action of going home, has both $At(\textit{home})$ and, of course, $\neg At(\textit{mall})$ as effects. Here, it is clear that $\mathcal{P}_1 \cup \mathcal{P}_2$, though schedulable, is not complete, since the action S_5 from $\mathcal{P}_2$ of going home threatens the link $\langle S_2, At(\textit{mall}), S_3 \rangle$ from $\mathcal{P}_1$. Still, it is easy to see that the two plans $\mathcal{P}_1$ and $\mathcal{P}_2$ are strongly compatible. This is shown by the constraint set $\mathcal{O}'' = \{t_3 < t_5\}$, since $\langle \mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{O}'', \mathcal{L}_1 \cup \mathcal{L}_2 \rangle$ is complete and schedulable. The intuitive force of this additional constraint, of course, is to guarantee that the agent goes home only after it purchases the shirt.

The notion of strong compatibility defined here is, in fact, a very strong notion, but it is not the strongest available. A stronger notion is that of perfect compatibility, where two complete plans $\mathcal{P}$ and $\mathcal{P}'$ are defined as *perfectly compatible* just in case their union $\mathcal{P} \cup \mathcal{P}'$ is itself complete and schedulable. As we have seen, the example plans $\mathcal{P}_1$ and $\mathcal{P}_2$, though strongly compatible, are not perfectly compatible. Another compatibility notion—perhaps more realistic and useful in the long run—is that of weak compatibility, which classifies two plans as compatible whenever their union can be made complete and schedulable through the addition, not only of additional constraints, but also, perhaps, of additional steps and links; formally, then, the plans $\mathcal{P} = \langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$ and $\mathcal{P}' = \langle \mathcal{S}', \mathcal{O}', \mathcal{L}' \rangle$ can be defined as *weakly compatible* whenever there are sets $\mathcal{S}''$, $\mathcal{O}''$, and $\mathcal{L}''$ such that

$$\langle \mathcal{S} \cup \mathcal{S}' \cup \mathcal{S}'', \mathcal{O} \cup \mathcal{O}' \cup \mathcal{O}'', \mathcal{L} \cup \mathcal{L}' \cup \mathcal{L}'' \rangle$$

is complete and schedulable. As an example, imagine an agent's plan to arrive at its office at 10:00 a.m. in order to meet with a colleague there at 1:00 p.m., where arriving at the office establishes the effect $In(\textit{office})$ that is necessary as a precondition for the latter action. This plan would not be strongly compatible with the plan to get lunch at a restaurant at noon, since departing for the restaurant would interrupt the link through which arriving at the office establishes the precondition for the later action of meeting the colleague. But the two plans are weakly compatible, since this precondition could be reestablished, and completeness guaranteed, by supplementing the union of these plans also with a further action of returning to the office after lunch, along with appropriate temporal constraints and causal links.

Each of the three compatibility notions considered here—perfect, strong, and weak—focuses only on ways in which plans might be supplemented in order to guarantee completeness and schedulability. Other compatibility notions might involve actually modifying, not just supplementing, one of the two plans: perhaps joint execution would be possible if certain inessential steps were replaced by others. In fact, there are a variety of notions of plan compatibility that might be worth exploring. Nevertheless, we limit our attention in the present paper to the notion of strong compatibility, which we take as a reasonable starting point.

Turning now to the semantics of our representation language, we take complete and scheduled plans to be the points in the semantic space, and we associate more abstract plans with sets of these. We begin by adapting the notion of refinement [12] from the plan generation literature.

Definition 6 (*Refinement*; $\sqsubseteq$). Let $\mathcal{P} = \langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$ and $\mathcal{P}' = \langle \mathcal{S}', \mathcal{O}', \mathcal{L}' \rangle$ be plans. Then $\mathcal{P}'$ is a *refinement* of $\mathcal{P}$ ($\mathcal{P} \sqsubseteq \mathcal{P}'$) just in case $\mathcal{S} \subseteq \mathcal{S}'$ and $\mathcal{O} \subseteq \mathcal{O}'$ and $\mathcal{L} \subseteq \mathcal{L}'$.

Letting Π represent the set of all complete and scheduled plans, we define the semantic interpretation of a plan as follows.

Definition 7 (*Interpretation*; $v[\mathcal{P}]$). The *interpretation* of a plan $\mathcal{P}$ is the set of its complete and scheduled refinements: $v[\mathcal{P}] = \{\mathcal{P}' \colon \mathcal{P} \sqsubseteq \mathcal{P}'\} \cap \Pi$.

The idea, of course, is that a plan is to be interpreted as the set of ways in which it might be carried out, and so it is natural to define a plan as consistent whenever there is some way in which it can be carried out.

Definition 8 (*Plan consistency*). A plan $\mathcal{P}$ is *consistent* just in case $v[\mathcal{P}] \neq \emptyset$.

Note that a complete plan is consistent just in case it is schedulable, and that an incomplete plan is consistent just in case it has a complete and schedulable refinement.

3. Evaluation of options

For the purposes of this paper, we define an *option* as a plan that is presented to an agent for acceptance or rejection. This terminology may seem peculiar, since it is often natural to think of an option as something more along the lines of a goal state—having a new shirt, say. Nevertheless, even when the value of an option lies entirely in the achievement of some goal state, sensible reasoning demands that goal states and their means of achievement—in this case, a trip to the mall—must be evaluated together. [3]

We suppose that an agent evaluates each new option $\mathcal{P}$ against the background of a context $\mathcal{C}$, some plan to which it is already committed, the process of evaluation then proceeds as follows. First, the agent determines whether $\mathcal{P}$ is compatible with $\mathcal{C}$—where again, for the purposes of this paper, we will assume that the concept of compatibility can be usefully approximated through our notion of strong compatibility—and if not, $\mathcal{P}$ is rejected. This first step reflects a central idea from the theory of resource-bounded reasoning developed in [3,4]: that one function of an agent's current commitments is to act as a "filter of admissibility", impeding the consideration of options that are incompatible with those commitments. If $\mathcal{P}$ is found to be compatible with $\mathcal{C}$, the agent should accept the

[3] Of course, goal states themselves could also be evaluated, by reference to the set of potential plans that achieve them.

new option just in case the benefit of this option outweighs its cost in the current context.[4] Again, we simplify by supposing that the benefit of the option $\mathcal{P}$—represented here as $\beta(\mathcal{P})$—is independent of context. In the most natural case, this benefit will derive from the goal state towards which the plan is directed. All that remains to be specified, then, is the cost of the new option $\mathcal{P}$ in the context $\mathcal{C}$.

Note that the current policy of simply rejecting incompatible options is a highly simplified implementation of the general idea of using background contexts to filter new options. More realistically, an agent faced with an incompatible option $\mathcal{P}$ could explore either local revisions to the plan that might guarantee compatibility, or else alternative plans for achieving the goal that $\mathcal{P}$ aims at; and if the goal is valuable, the agent might also consider modifications of its background context. The presumption, however, will always be in favor of maintaining plans to which the agent has already committed—that is, of minimizing modifications to the background context. The full development of this idea would require a theory of localizing plan conflict and performing localized plan repair. This is a challenging problem, which we intend to address in future research.

3.1. Cost in isolation

We begin by defining the cost of a plan in isolation. We take as given a function *Cost* mapping action types into real numbers representing their costs; this function extends to the steps of a plan in the natural way: $Cost(S_i) = Cost(type(S_i))$.

Next, we introduce an auxiliary notion of point cost, defined only for complete, scheduled plans—the points in the semantic space. Where $\mathcal{P} = \langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$ is such a plan, we partition the plan steps into sets of actions forced (by the temporal constraints) to occur at the same moment. For each $S_i \in \mathcal{S}$, we take $[S_i] = \{S_j\colon\ \mathcal{O} \vdash t_i = t_j\}$. We then let $\mathcal{P}^*$ represent the partition of S induced by this equivalence relation: $\mathcal{P}^* = \{[S_i]\colon\ S_i \in \mathcal{S}\}$. It follows from our definition of a schedule that steps in the same equivalence class will necessarily represent actions of the same type; these type-identical steps performed at the same moment are to be thought of as collapsing into a single merged step. We therefore define the point cost of the plan itself as the sum of the costs assigned to the merged steps it contains:

$$\textit{Point-cost}\,(\mathcal{P}) = \sum_{[S_i] \in \mathcal{P}^*} Cost(S_i).$$

Given this auxiliary notion, it is now natural to define the cost of an arbitrary consistent plan as the point cost of the least expensive way in which it might be carried out, the least expensive point in its semantic interpretation.

Definition 9 (*Cost of a plan*; $\kappa(\mathcal{P})$). Where $\mathcal{P}$ is a consistent plan, the *cost* of $\mathcal{P}$ is the point cost of its least expensive complete and scheduled refinement: $\kappa(\mathcal{P}) = \min\{\textit{Point-cost}\,(\mathcal{P}')\colon\ \mathcal{P}' \in \upsilon[\mathcal{P}]\}$.

[4] Modeling the costs and benefits of actions and plans is the subject of the field of decision analysis [22], which has developed procedures for eliciting and codifying cost/benefits models. These models could be used to define the costs and benefits that are taken as primitive in this paper.

It is easy to see that $\kappa(\mathcal{P}) = \textit{Point-cost}(\mathcal{P})$ whenever $\mathcal{P}$ is itself a complete and scheduled plan, and that $\kappa(\mathcal{P}_\emptyset) = 0$ for the null plan $\mathcal{P}_\emptyset = \langle\emptyset, \emptyset, \emptyset\rangle$.

3.2. Cost in context

Having defined the cost of a plan in isolation, we now turn to our central task of defining the cost of a new option $\mathcal{P}$ in the context of a background plan $\mathcal{C}$. Our treatment of this concept is simple: we take the cost of the new option in context to be its *marginal* cost—the cost of carrying out $\mathcal{P}$ along with $\mathcal{C}$, less the cost of carrying out $\mathcal{C}$ alone.

Definition 10 (*Cost of a plan in a context*; $\kappa(\mathcal{P}/\mathcal{C})$). Where the plans $\mathcal{C}$ and $\mathcal{P}$ are strongly compatible, the *cost of* $\mathcal{P}$ *in the context* $\mathcal{C}$ is $\kappa(\mathcal{P}/\mathcal{C}) = \kappa(\mathcal{P} \cup \mathcal{C}) - \kappa(\mathcal{C})$.

It follows immediately from this definition that the cost of a plan in the null context is identical to its cost in isolation: $\kappa(\mathcal{P}/\mathcal{P}_\emptyset) = \kappa(\mathcal{P})$. It is also worth noting that the cost of a plan in any context that already includes that plan as a component is zero: $\kappa(\mathcal{P}/\mathcal{P} \cup \mathcal{C}) = 0$.

This definition can be illustrated with a case in which the cost of a new option is actually affected by the background context. Suppose the agent's background context is simply the plan to buy a shirt at the mall, represented by $\mathcal{P}_1$ from Fig. 1, and imagine that the agent is presented with the new option of going to the mall for some swim goggles, depicted as $\mathcal{P}_3$ in Fig. 3. Formally, we have $\mathcal{P}_3 = \langle\mathcal{S}_3, \mathcal{O}_3, \mathcal{L}_3\rangle$, where $\mathcal{S}_3 = \{S_7, S_8, S_9, S_{10}\}$, $\mathcal{O}_3 = \{t_7 < t_9, t_8 < t_9, t_9 < t_{10}\}$, and $\mathcal{L}_3 = \{\langle S_7, \textit{At}(\textit{mall}), S_9\rangle, \langle S_8, \textit{Have}(\textit{money}), S_9\rangle, \langle S_9, \textit{Have}(\textit{goggles}), S_{10}\rangle\}$. Here, the steps S_7 and S_8 again represent actions of going to the mall and getting money, steps sharing the respective types of S_1 and S_2 from the background plan $\mathcal{P}_1$; the step S_9 represents the action of purchasing the goggles, and S_{10} is again a dummy step representing goal achievement. Let us suppose that these various steps carry the following costs: each of S_2, S_3, S_8, and S_9 carries a cost of 1, since both getting money and making a purchase are easy to do; each of S_1 and S_7 carries a cost of 10, since any trip to the mall is abhorrent; and S_4 and S_{10}, as dummy steps, both carry a cost of 0.

Given this information, it is clear that $\kappa(\mathcal{P}_1) = 12$—the cost of the agent's background plan is 12. Now, turning to the new option, suppose that the agent would like to have swim goggles, but that it is not terribly important: $\beta(\mathcal{P}_3) = 2$. It is clear also that $\kappa(\mathcal{P}_3) = 12$, so that, considered in isolation, this new option would not be worth pursuing

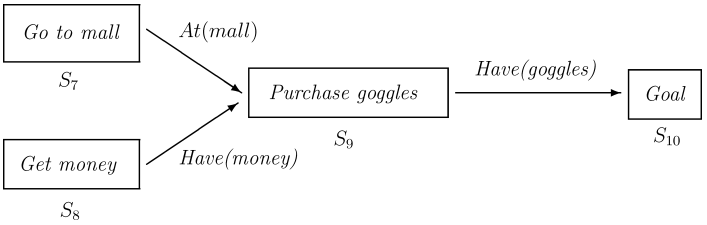

Fig. 3. Another plan: buying goggles.

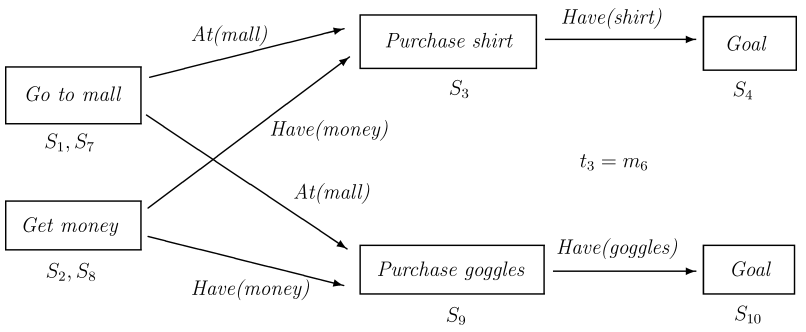

Fig. 4. Merged plans.

($\beta(\mathcal{P}_3) < \kappa(\mathcal{P}_3)$). On the other hand, it is easy to see that $\kappa(\mathcal{P}_3 \cup \mathcal{P}_1) = 13$, since the least expensive execution of the joint plan, in which both the steps S_1 and S_7 as well as the steps S_2 and S_8 are merged, carries a cost of 13; see Fig. 4. Therefore, we have $\kappa(\mathcal{P}_3/\mathcal{P}_1) = \kappa(\mathcal{P}_3 \cup \mathcal{P}_1) - \kappa(\mathcal{P}_1) = 1$. Even though the new option would not have been worth pursuing in isolation, it is worth pursuing in context, since its benefit is greater than its cost in context.

As this example shows, the cost of a plan in context may be less than its cost in isolation, but it is also possible for the cost in context to be greater. Our earlier taxi/van story already illustrates this possibility, but it is worth noting that it also arises even in the more restricted framework of complete, primitive plans. In this setting, a plan in context will have a higher cost than it has in isolation if the background plan contains steps that might be merged if it were performed in isolation, but the new option blocks that merge possibility. For instance, suppose the agent already intends to purchase a shirt at the mall, and intends also to buy a suitcase at the luggage store, also located in the mall. Although the agent would prefer to take care of both tasks on the same trip, it is not yet committed to this. (Suppose that each task is worthwhile even if done separately.) Now suppose that the agent is considering an option of seeing a movie at the mall theater. Given its other commitments and the constraints of the store hours, the agent will not be able to combine all three events—the shirt purchase, the suitcase purchase, and the movie—into a single trip. If the movie is combined with the shirt purchase, for example, the agent will need to make another trip to the mall to purchase the suitcase, and similarly if the movie is combined with the suitcase purchase. Attending the movie, in this context, has an extra cost, since it means an additional trip to the mall to carry out the already intended plans. Just as in the taxi/van story, the present example illustrates a case in which a new option is more expensive in context than in isolation, here because its adoption would rule out the most efficient executions of the background plan.

3.3. Cost estimates

Although the notion of cost as the least expensive method of execution is defined for any consistent plan, we do not necessarily assume that the agent knows the true cost either

of its background plan or of any new options under consideration. Instead, the agent may only compute an estimate of the cost of its plans.

Definition 11 (*Cost estimate for a plan*). Where $\mathcal{P}$ is a consistent plan, a *cost estimate* for $\mathcal{P}$ is an interval of the form $\varepsilon = [\varepsilon^-, \varepsilon^+]$, where ε^- and ε^+ are nonnegative real numbers such that $\varepsilon^- \leqslant \kappa(\mathcal{P}) \leqslant \varepsilon^+$.

Cost estimates, so defined, accurately bound the actual cost of a plan, and are thus related to the interval measures of plan cost used in the decision-theoretic plan generation literature [8,10,28].

We now show that, under certain coherence conditions, a cost estimate for a plan in context can be derived from a cost estimate for the context together with a cost estimate for the plan and context combined, using notions from interval arithmetic [17]. Assume that $\mathcal{P}$ and $\mathcal{C}$ are strongly compatible plans, and that $\varepsilon_{\mathcal{C}} = [\varepsilon^-_{\mathcal{C}}, \varepsilon^+_{\mathcal{C}}]$ and $\varepsilon_{\mathcal{P}\cup\mathcal{C}} = [\varepsilon^-_{\mathcal{P}\cup\mathcal{C}}, \varepsilon^+_{\mathcal{P}\cup\mathcal{C}}]$ are cost estimates for the plans $\mathcal{C}$ and $\mathcal{P}\cup\mathcal{C}$ respectively. We know from the definition of a cost estimate that $\varepsilon^-_{\mathcal{C}} \leqslant \varepsilon^+_{\mathcal{C}}$ and $\varepsilon^-_{\mathcal{P}\cup\mathcal{C}} \leqslant \varepsilon^+_{\mathcal{P}\cup\mathcal{C}}$, but the definition tells us nothing about the relations among the intervals $\varepsilon_{\mathcal{C}}$ and $\varepsilon_{\mathcal{P}\cup\mathcal{C}}$ themselves. Nevertheless, we can conclude that $\varepsilon^-_{\mathcal{C}} \leqslant \varepsilon^-_{\mathcal{P}\cup\mathcal{C}}$, since the least expensive execution of the compound plan $\mathcal{P}\cup\mathcal{C}$ cannot be less costly than the least expensive execution of $\mathcal{C}$, one of its components; and similarly, we can conclude that $\varepsilon^+_{\mathcal{C}} \leqslant \varepsilon^+_{\mathcal{P}\cup\mathcal{C}}$, since the most expensive execution of $\mathcal{P}\cup\mathcal{C}$ cannot be less costly than the most expensive execution of $\mathcal{C}$. We characterize the pair of estimates $\varepsilon_{\mathcal{C}}$ and $\varepsilon_{\mathcal{P}\cup\mathcal{C}}$ as *jointly coherent* just in case these two conditions hold: $\varepsilon^-_{\mathcal{C}} \leqslant \varepsilon^-_{\mathcal{P}\cup\mathcal{C}}$ and $\varepsilon^+_{\mathcal{C}} \leqslant \varepsilon^+_{\mathcal{P}\cup\mathcal{C}}$.

As long as $\varepsilon_{\mathcal{C}}$ and $\varepsilon_{\mathcal{P}\cup\mathcal{C}}$ are jointly coherent we can derive a cost estimate $\varepsilon_{\mathcal{P}/\mathcal{C}} = [\varepsilon^-_{\mathcal{P}/\mathcal{C}}, \varepsilon^+_{\mathcal{P}/\mathcal{C}}]$ for the plan $\mathcal{P}$ in the context $\mathcal{C}$ by defining $\varepsilon^-_{\mathcal{P}/\mathcal{C}}$ and $\varepsilon^+_{\mathcal{P}/\mathcal{C}}$—the minimum and maximum possible costs that might be incurred in executing the compound plan $\mathcal{P}\cup\mathcal{C}$ in place of $\mathcal{C}$ alone—in the following way. Given joint coherence, the end points of the intervals $\varepsilon_{\mathcal{C}}$ and $\varepsilon_{\mathcal{P}\cup\mathcal{C}}$ can stand in only two possible ordering relations:

(1) $\varepsilon^-_{\mathcal{C}} \leqslant \varepsilon^+_{\mathcal{C}} \leqslant \varepsilon^-_{\mathcal{P}\cup\mathcal{C}} \leqslant \varepsilon^+_{\mathcal{P}\cup\mathcal{C}}$,

(2) $\varepsilon^-_{\mathcal{C}} \leqslant \varepsilon^-_{\mathcal{P}\cup\mathcal{C}} \leqslant \varepsilon^+_{\mathcal{C}} \leqslant \varepsilon^+_{\mathcal{P}\cup\mathcal{C}}$.

Case (1) is illustrated on the left-hand side of Fig. 5, and case (2) on the right-hand side. In either case, it is clear that $\varepsilon^+_{\mathcal{P}/\mathcal{C}}$ should be defined as $\varepsilon^+_{\mathcal{P}\cup\mathcal{C}} - \varepsilon^-_{\mathcal{C}}$, the maximum possible distance between points in $\varepsilon_{\mathcal{P}\cup\mathcal{C}}$ and $\varepsilon_{\mathcal{C}}$; this is shown by the solid lines in the figure. In

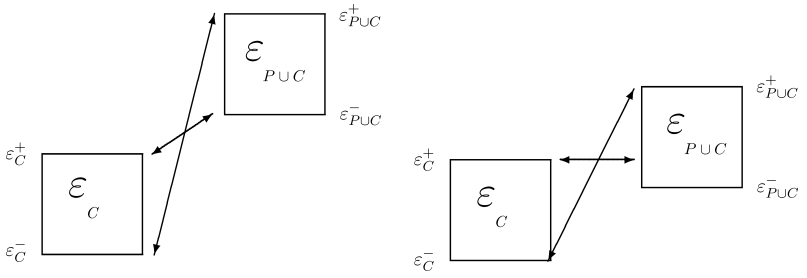

Fig. 5. Jointly coherent cost estimates.

case (1), we know that $\varepsilon^-_{\mathcal{P}/\mathcal{C}}$ should likewise defined as $\varepsilon^-_{\mathcal{P}\cup\mathcal{C}} - \varepsilon^+_{\mathcal{C}}$, the minimum possible distance. In case (2), it is reasonable to take $\varepsilon^-_{\mathcal{P}/\mathcal{C}}$ as 0, since we know, even when the low estimate for executing $\mathcal{P}\cup\mathcal{C}$ is less than the high estimate for executing $\mathcal{C}$, that the true cost of executing $\mathcal{P}\cup\mathcal{C}$ can be no less than the true cost of executing $\mathcal{C}$. The minimum possible distances are shown by the dotted lines in the figure. Combining cases (1) and (2), we can therefore take $\varepsilon^-_{\mathcal{P}/\mathcal{C}}$ as $\max[0, \varepsilon^-_{\mathcal{P}\cup\mathcal{C}} - \varepsilon^+_{\mathcal{C}}]$, leading to the following general definition.

Definition 12 (*Cost estimate for a plan in context*). Where the plans $\mathcal{P}$ and $\mathcal{C}$ are strongly compatible, let $\varepsilon_{\mathcal{C}} = [\varepsilon^-_{\mathcal{C}}, \varepsilon^+_{\mathcal{C}}]$ and $\varepsilon_{\mathcal{P}\cup\mathcal{C}} = [\varepsilon^-_{\mathcal{P}\cup\mathcal{C}}, \varepsilon^+_{\mathcal{P}\cup\mathcal{C}}]$ be a pair of jointly coherent cost estimates for the plans $\mathcal{C}$ and $\mathcal{P}\cup\mathcal{C}$. Then the *cost estimate for the plan* $\mathcal{P}$ *in the context* $\mathcal{C}$ is the interval $\varepsilon_{\mathcal{P}/\mathcal{C}} = [\varepsilon^-_{\mathcal{P}/\mathcal{C}}, \varepsilon^+_{\mathcal{P}/\mathcal{C}}]$, where $\varepsilon^-_{\mathcal{P}/\mathcal{C}} = \max[0, \varepsilon^-_{\mathcal{P}\cup\mathcal{C}} - \varepsilon^+_{\mathcal{C}}]$ and $\varepsilon^+_{\mathcal{P}/\mathcal{C}} = \varepsilon^+_{\mathcal{P}\cup\mathcal{C}} - \varepsilon^-_{\mathcal{C}}$.

It follows immediately from this definition that $\kappa(\mathcal{P}/\mathcal{C})$, the true cost of $\mathcal{P}$ in the context $\mathcal{C}$, lies within the derived interval $\varepsilon_{\mathcal{P}/\mathcal{C}}$; and it is also easy to see that the derived interval $\varepsilon_{\mathcal{P}/\mathcal{C}}$ narrows monotonically as the intervals $\varepsilon_{\mathcal{C}}$ and $\varepsilon_{\mathcal{P}\cup\mathcal{C}}$ are narrowed.

The derived interval estimate of cost in context is useful because, in many cases, it allows an agent to accept or reject an option without calculating its true cost, as illustrated in Fig. 6. Suppose, for example, that an agent with background plan $\mathcal{C}$ is considering the new option $\mathcal{P}$ with benefit $\beta(\mathcal{P})$; and imagine that the agent has assigned estimated costs $\varepsilon_{\mathcal{C}}$ and $\varepsilon_{\mathcal{P}\cup\mathcal{C}}$ to the plans $\mathcal{C}$ and $\mathcal{P}\cup\mathcal{C}$, from which it derives the estimate $\varepsilon_{\mathcal{P}/\mathcal{C}} = [\varepsilon^-_{\mathcal{P}/\mathcal{C}}, \varepsilon^+_{\mathcal{P}/\mathcal{C}}]$ for the cost of $\mathcal{P}$ in the context $\mathcal{C}$. Then if $\beta(\mathcal{P}) > \varepsilon^+_{\mathcal{P}/\mathcal{C}}$, the agent is justified in adopting the new option, since the cost in context of the option is necessarily less than its benefit; and likewise, the agent is justified in rejecting the option if $\beta(\mathcal{P}) < \varepsilon^-_{\mathcal{P}/\mathcal{C}}$, since its cost in context is necessarily greater than its benefit. If $\varepsilon^-_{\mathcal{P}/\mathcal{C}} \leqslant \beta(\mathcal{P}) \leqslant \varepsilon^+_{\mathcal{P}/\mathcal{C}}$, there are two subcases to consider. First, if it happens that $\varepsilon^-_{\mathcal{P}/\mathcal{C}} = \varepsilon^+_{\mathcal{P}/\mathcal{C}}$, then, since we know that $\kappa(\mathcal{P}/\mathcal{C})$ lies within the interval $\varepsilon_{\mathcal{P}/\mathcal{C}}$, it follows that $\beta(\mathcal{P}) = \kappa(\mathcal{P}/\mathcal{C})$, and so the agent is justified either in accepting or rejecting the option. If $\varepsilon^-_{\mathcal{P}/\mathcal{C}} < \varepsilon^+_{\mathcal{P}/\mathcal{C}}$, on the other hand, the agent's interval estimates do not provide enough information to determine whether the option should be adopted or rejected. In this last case, and only this case, the agent is forced to refine its estimates further before making a rational decision, narrowing its cost estimates for $\mathcal{C}$ and $\mathcal{P}\cup\mathcal{C}$, and thereby also narrowing its derived estimate for $\mathcal{P}$ in the context of $\mathcal{C}$.

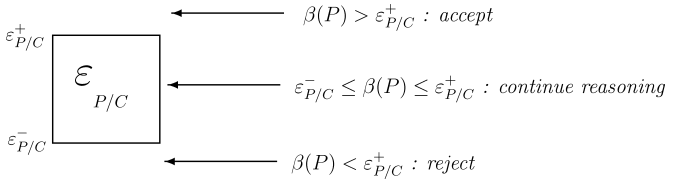

Fig. 6. Using cost estimates.

4. Reasoning procedures

We now present some algorithms for the reasoning processes described above. As explained earlier, the first step of the process involves determining whether $\mathcal{P}$ is, in fact, strongly compatible with the context $\mathcal{C}$. In work reported elsewhere [26], we have recently developed a new algorithm to compute consistency for plans that have quantitative temporal constraints and steps with extended duration, as well as observation actions and conditional branches. The algorithm uses a CSP-based approach to find alternative sets of constraints that guarantee consistency of the input plans.

In the present paper, we focus on the reasoning that occurs after a set of conditions for ensuring compatibility has been computed. That is, we develop algorithms by which an agent can compute a cost estimate for the plan $\mathcal{P}$ in the context $\mathcal{C}$. An algorithm for this purpose was developed in [29], using a dynamic programming approach to find an optimally merged plan, one with minimal cost. This cost is precisely $\kappa(\mathcal{P} \cup \mathcal{C})$, which could then be combined with an exact value for $\kappa(\mathcal{C})$ to yield an exact value for $\kappa(\mathcal{P}/\mathcal{C})$. The option $\mathcal{P}$ could then be accepted or rejected depending on the relation between $\kappa(\mathcal{P}/\mathcal{C})$ and $\beta(\mathcal{P})$.

We do not rely on the algorithm developed in [29], however, for two reasons. First, it applies only to classical plans, which lack metric temporal constraints. And second, it is not in general necessary to compute the exact value of $\kappa(\mathcal{P}/\mathcal{C})$ in order to evaluate the new option; instead, as suggested earlier, an agent may be able to accept or reject a new option simply on the basis of an interval estimate of its cost in context. The remainder of this section develops an algorithm to implement this idea: evaluating a new option by estimating its cost in context, and then progressively refining the estimate where necessary. Such an approach may prove to be efficient if, as we suspect, it can frequently terminate in realistic cases without the need to compute an exact cost in context. In addition, the algorithm presented here displays anytime performance, producing cost estimates of monotonically increasing accuracy. Thus, if the agent "runs out of time" in evaluating an option, and is forced to a decision before shrinking the cost range sufficiently, the agent can at least make an informed decision; it can determine, for example, how much it stands to lose. Finally, by reasoning about partially scheduled plans and estimated costs, the algorithm matches our intuitions about deliberation in dynamic environments.

4.1. Stepsets

Given our current restriction to complete plans, the only factor influencing plan cost is step merging. To support reasoning about possible step merges, we therefore introduce a data structure called a stepset, which represents possible ways of merging steps in a plan (or in the union of two plans). A stepset clusters (some) steps in a plan that share the same time of execution: where $\mathcal{P} = \langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$ is a consistent plan, a *stepset* for the plan $\mathcal{P}$ is a partition $M = \{[S_1], \ldots, [S_n]\}$ of $\mathcal{S}$ subject to the restriction that $type(S_i) = type(S_j)$ for any steps S_i and S_j belonging to the same equivalence class $[S_k]$ in M. Intuitively, a stepset for a plan $\mathcal{P}$ represents the set of schedules for $\mathcal{P}$ that merge all and only plan steps in the same stepset equivalence class.

Because type-equivalence is a necessary but not sufficient condition for placing two steps in the same equivalence class, a given plan may have several different stepsets. Consider an extremely simple plan $\mathcal{P}$ containing only two steps, S_1 and S_2, both of which are of the same type. The plan $\mathcal{P}$ will have one stepset in which its two steps are clustered, and another in which they are not:

(1) $M_1 = \{[S_1, S_2]\}$,
(2) $M_2 = \{[S_1], [S_2]\}$.

Intuitively, the former corresponds to all schedules for $\mathcal{P}$ in which its two steps are merged, while the latter corresponds to those in which they are not. Note that the temporal constraints in $\mathcal{P}$ may prevent the merging of its two steps.

More precisely, let be $\mathcal{P} = \langle \mathcal{S}, \mathcal{O}, \mathcal{L} \rangle$ be a plan and let $M = \{[S_1], \ldots, [S_n]\}$ be a stepset based on $\mathcal{P}$. We define the *stepset constraints* associated with M to be the constraint set containing $t_i = t_j$ whenever $[S_i] = [S_j]$, and $t_i \neq t_j$ whenever $[S_i] \neq [S_j]$, and we will write $\mathit{Const}(M)$ to denote the stepset constraints for stepset M. A stepset M is then defined as *feasible* just in case the plan $\langle \mathcal{S}, \mathcal{O} \cup \mathit{Const}(M), \mathcal{L} \rangle$ is complete and schedulable, and *infeasible* otherwise.

Stepsets represent only decisions about which steps in a plan are to be merged, neglecting any other information about the order of steps or the exact times of their performance. On the other hand, stepsets do capture all the information that is necessary for computing plan cost, because, under the current assumptions, cost depends only on step merging. Given a stepset M for a plan $\mathcal{P}$, we can introduce a notion of *stepset cost* determined by M—written $\mathit{SSCost}(M)$—as follows:

$$\mathit{SSCost}(M) = \sum_{[S_i] \in M} \mathit{Cost}(S_i).$$

The stepset cost determined by M then coincides with the cost of any scheduled refinement of the plan $\mathcal{P}$ whose schedule is consistent with $\mathit{Const}(M)$.

The stepsets based on a plan $\mathcal{P}$ can be organized into a lattice, as follows. The top element of the lattice is the minimally merged stepset $\mathit{Minmerge}(\mathcal{P})$, defined as the partition $\{[S_1], \ldots, [S_n]\}$ in which each equivalence class $[S_i]$ is identified with the unit set $\{S_i\}$. The bottom element of the lattice is the maximally merged stepset $\mathit{Maxmerge}(\mathcal{P})$, defined as the partition $\{[S_1], \ldots, [S_n]\}$, in which each equivalence class $[S_i]$ is identified with the set $\{S_j\colon \mathit{type}(S_j) = \mathit{type}(S_i)\}$ containing all steps sharing the type of S_i. Clearly, the minimally merged stepset has maximal cost, and the maximally merged stepset has minimal cost.

We define one stepset as being below another in the lattice if it results from increased merging. More exactly, where M and M' are elements of the lattice, we define $M \leqslant M'$ just in case: for each $[S_i]$ in M' there is an $[S_j]$ in M such that $[S_i] \subseteq [S_j]$. We can then define the *down successors* of a stepset M as those stepsets that are below M in the lattice and contain exactly one fewer member; and we can define the *up successors* of M as those stepsets that are above M in the lattice and contain exactly one more member. The down successors of M are those stepsets that can be obtained from M by merging two of its members, and the up successors of M are those stepsets from which M can be obtained through the merge of two members.

These various stepset concepts are implemented as the following functions: SSCOST(M) calculates the stepset cost of a stepset M; MINMERGE($\mathcal{P}$) and MAXMERGE($\mathcal{P}$) form the minimally and maximally merged stepsets of the plan $\mathcal{P}$; DOWN-SUCCESSORS(M) and UP-SUCCESSORS(M) return the down successors and up successors of the stepset M in the relevant lattice; and FEASIBLE($M, \mathcal{P}$) determines whether M is a feasible stepset for plan $\mathcal{P}$. All but the last of these functions are trivial. Feasibility requires checking whether $\langle \mathcal{S}, \mathcal{O} \cup \mathit{Const}(M), \mathcal{L} \rangle$ is complete and schedulable. We do this by casting the problem as a CSP, in which the constrained variables are the steps in $\mathcal{S}$, and their domains are moments of execution: a solution to the CSP problem then consists of an assignment of a time point to each step. Two sets of constraints must be observed: the temporal constraints in $\mathcal{O} \cup \mathit{Const}(M)$, and the threat-avoidance constraints that derive from $\mathcal{L}$. Although solving a CSP is computationally intractable in the worst case, there are a number of powerful techniques that are known to work very well in practice, and in fact, have recently been applied to large planning and scheduling problems [5,6,13,14].

4.2. The algorithm

We now present our algorithm, depicted in Fig. 7, for evaluating an option $\mathcal{P}$ in the context $\mathcal{C}$, under the assumption that the two plans are strongly compatible. The algorithm

```
procedure EVALUATE-OPTION(P, C) return Accept or Reject
    INITIALIZE(P, C)
    loop
        if COHERENT(ε⁺_C, ε⁻_C, ε⁺_{P∪C}, ε⁻_{P∪C}) then
            ε⁻_{P/C} ← max[0, ε⁻_{P∪C} − ε⁺_C]
            ε⁺_{P/C} ← ε⁺_{P∪C} − ε⁻_C
            if β[P] > ε⁺_{P/C} then
                return Accept
            end if
            if β[P] < ε⁻_{P/C} then
                return Reject
            end if
            if ε⁻_{P/C} = ε⁺_{P/C} then
                return Accept or Reject
            end if
        end if
        Call either REFINE(C) or REFINE(P ∪ C)
    end loop
```

Fig. 7. EVALUATE-OPTION($\mathcal{P}, \mathcal{C}$).

works with two stepset lattices, based on the plans $\mathcal{C}$ and $\mathcal{P} \cup \mathcal{C}$. In a fashion somewhat reminiscent of the candidate-elimination algorithm [16], our algorithms maintains, for each lattice, an upper frontier containing the highest nodes in the lattice not yet known to be infeasible, and similarly a lower frontier. It then systematically attempts to establish the feasibility or infeasibility of the nodes in the frontiers, refining the cost estimates for the plans $\mathcal{C}$ and $\mathcal{P} \cup \mathcal{C}$, and using them to update the derived cost estimate for $\mathcal{P}$ in the context $\mathcal{C}$. After each refinement, the derived estimate of cost in context is compared with $\beta(\mathcal{P})$, the benefit of $\mathcal{P}$. If $\beta(\mathcal{P})$ is outside the range of the current estimate, the algorithm then terminates with a recommendation to either accept or reject $\mathcal{P}$.

We have left the decision about which estimate to refine ($\varepsilon_{\mathcal{C}}$ or $\varepsilon_{\mathcal{P}\cup\mathcal{C}}$) nondeterministic in the algorithm, thereby decoupling the control heuristics from the algorithm itself. A similar decoupling has proven to be useful in the analysis of many AI algorithms. This tactic has two advantages. First, it simplifies the base algorithm; see [27] for an example of this applied to classical plan generation. Second, it allows for separate, focused investigation of heuristics for efficient control, something we are now exploring in our ongoing work. For example, recent work in decision-theoretic plan generation [8] has analyzed the use of interval-based utility estimates in finding optimal plans, and there is potential for the transfer of this analysis to our framework. See also our discussion in Section 4.3.

The procedure REFINE($\mathcal{Q}$)—where $\mathcal{Q}$ ranges over $\mathcal{C}$ and $\mathcal{P} \cup \mathcal{C}$—is itself straightforward, simply calling either REFINE-DOWN($\mathcal{Q}$) or REFINE-UP($\mathcal{Q}$), again, nondeterministically choosing between them.

The two refinement procedures are similar, and we first describe REFINE-DOWN($\mathcal{Q}$), shown in Fig. 8. Essentially, what it does is select a node M from the upper frontier of $\mathcal{Q}$, and check its feasibility. The nodes are selected in decreasing order of cost. Thus, if M is

```
procedure REFINE-DOWN(Q)
    if Actual_Q^- = F then
        Select a maximal cost element M from Upper_Q
        Delete M from Upper_Q
        Add DOWN-SUCCESSORS(M) to Upper_Q
        if FEASIBLE(M, Q) then
            Actual_Q^+ ← T
            ε_Q^+ ← SSCOST(M)
        else
            if Actual_Q^+ = F then
                ε_Q^+ ← Argmax_{U ∈ Upper_Q} SSCOST(U)
            end if
        end if
    end if
```

Fig. 8. REFINE-DOWN($\mathcal{Q}$).

found to be feasible, its cost is less than that of any previously examined feasible node; hence the current upper bound on the cost estimate, $\varepsilon^+_{\mathcal{Q}}$, can be set to the cost of M. What if M is found to be infeasible? Both $\mathcal{C}$ and $\mathcal{P} \cup \mathcal{C}$ are consistent; the latter is a consequence of the strong compatibility of $\mathcal{P}$ and $\mathcal{C}$. Thus, regardless of whether $\mathcal{Q}$ is $\mathcal{C}$ or $\mathcal{P} \cup \mathcal{C}$, its stepset lattice is guaranteed to contain at least one feasible node. If a feasible node has not yet been found—that is, if $Actual^+_{\mathcal{Q}} = F$—we know that some feasible element exists whose stepset cost is less than that of M, and so $\varepsilon^+_{\mathcal{Q}}$ can be set to reflect the highest stepset cost of the remaining elements. Again, this can only lower the value of $\varepsilon^+_{\mathcal{Q}}$. However, if a feasible point has already been found, we have no guarantee of finding another, and so determining that M is infeasible does not lead to updating the value of $\varepsilon^+_{\mathcal{Q}}$.

After examining M, the procedure removes it from the upper frontier, and replaces it with its down-successors, so that they can be considered in subsequent iterations.

The procedure REFINE-UP($\mathcal{Q}$), presented in Fig. 9, is nearly a dual to REFINE-DOWN($\mathcal{Q}$): it refines the lower bound $\varepsilon^-_{\mathcal{Q}}$ of the cost estimate for the plan $\mathcal{Q}$, working from the lower frontier. It differs, however, in its behavior when a feasible node is found. The reason for this is that the first feasible node found represents $\kappa(\mathcal{Q})$, the true cost of $\mathcal{Q}$: because the procedure selects nodes in increasing order of cost, the first feasible node it finds necessarily corresponds to the least expensive schedule for $\mathcal{Q}$, which is the true cost of $\mathcal{Q}$. Therefore, immediately upon finding a feasible node, REFINE-UP($\mathcal{Q}$) sets the upper bound $\varepsilon^+_{\mathcal{Q}}$ to $\varepsilon^-_{\mathcal{Q}}$, narrowing the estimate to a point.

One final comment about the algorithms concerns the initial lines of both REFINE-UP($\mathcal{Q}$) and REFINE-DOWN($\mathcal{Q}$), which require that $Actual^-_{\mathcal{Q}} = F$. This is simply a bookkeeping condition: because REFINE-UP($\mathcal{Q}$) sets both $\varepsilon^-_{\mathcal{Q}}$ and $\varepsilon^+_{\mathcal{Q}}$ to the exact cost of $\mathcal{Q}$ at the same time as it sets $Actual^-_{\mathcal{Q}} = T$, this condition blocks additional calls to either procedure from producing any further effect.

```
procedure REFINE-UP(Q)
    if Actual⁻_Q = F then
        Select a minimal cost element M from Lower_Q
        Delete M from Lower_Q
        Add UP-SUCCESSORS(M) to Lower_Q
        if FEASIBLE(M, Q) then
            Actual⁻_Q ← T
            ε⁻_Q ← SSCOST(M)
            ε⁺_Q ← ε⁻_Q
        else
            ε⁻_Q ← Argmin_{U ∈ Lower_Q} SSCOST(U)
        end if
    end if
```

Fig. 9. REFINE-UP($\mathcal{Q}$).

4.3. Implementation and analysis

We implemented our algorithms in Allegro Common Lisp. The amount of time required for each iteration is dominated by the step that checks whether a stepset is feasible; the remaining tasks—generating successor stepsets, computing stepset costs, and comparing the cost estimate to the plan benefit—are all quite fast. To perform feasibility checking, we used an algorithm we developed to compute compatibility of plans with conditional branches and rich temporal constraints, as mentioned above [20]. The performance of our feasibility-checking algorithm depends significantly on a number of factors, including the size of the plans, the number of branches they contain, the amount of potential conflict between them, and the tightness of their temporal constraints. Because the algorithm relies on CSP techniques, its worst-case performance is NP-complete, but there are well-known methods for achieving in-practice acceptable behavior, and we are currently developing others [25].

Here we note simply that we have so far used only brute-force constraint propagation techniques in compatibility checking, and are currently investigating techniques for significantly more efficient processing.

The performance of our overall algorithm is clearly a function of the number of nodes in the stepset lattice that must be generated and processed by the EVALUATE-OPTION routine. We can compute the size of the largest possible stepset lattice for a given plan as follows. First, note that the number of ways to partition m objects into n buckets leaving no bucket empty, is

$$S(m,n) = \frac{1}{n!} \cdot \sum_{k=0}^{n} \left[(-1)^k \cdot C(n, n-k) \cdot (n-k)^m \right],$$

and that, because a set of m elements can be partitioned into i buckets, where i varies from 1 to m, the total number of distinct ways of partitioning a set of m elements is given by the following formula [9]:

$$N(m) = \sum_{n=1}^{m} S(m,n).$$

Suppose that the plan $\mathcal{P}$ includes z distinct types of actions, represented as $T_1, T_2, \ldots, T_z$; and define $num(\mathcal{P}, T_i)$ to be the number of actions from $\mathcal{P}$ belonging to the type of T_i. Clearly, the number of ways of partitioning the actions in $\mathcal{P}$ of type T_i is $N[num(\mathcal{P}, T_i)]$. The partitions of the different actions types are independent, and so the overall size of the stepset lattice for the plan $\mathcal{P}$ is simply

$$L(\mathcal{P}) = \prod_{i=1}^{z} N\left[num(\mathcal{P}, T_i)\right],$$

the product of the numbers of partitions for each of the various action types in the plan.

In the worst case, then, the size of the stepset lattice for a plan is exponential in the size of the plan, and so the time required by EVALUATE-OPTION to search through the entire stepset lattices generated by $\mathcal{C}$ and $\mathcal{P} \cup \mathcal{C}$ is likewise exponential. However, there are several reasons to believe that worst case performance will rarely occur. First, we suspect that in

realistic cases—plans like those confronted by real agents in the real world—it will seldom be necessary to search through the entire stepset lattices in order to determine whether a new option should be accepted or rejected. More often, the system should determine fairly quickly that the value of the plan, $\beta[\mathcal{P}]$, lies outside the cost range $[\varepsilon^-_{\mathcal{P}/\mathcal{C}}, \varepsilon^+_{\mathcal{P}/\mathcal{C}}]$.

Second, the algorithm as described searches through the stepset lattices blindly, but a number of promising heuristics suggest themselves. For example, if the value of the new option $v[\mathcal{P}]$ is very close to $\varepsilon^+_{\mathcal{P}/\mathcal{C}}$—much closer than to $\varepsilon^-_{\mathcal{P}/\mathcal{C}}$—then it is natural to attempt to reduce $\varepsilon^+_{\mathcal{P}/\mathcal{C}}$, hoping to push this value lower than $v[\mathcal{P}]$, so that the reasoning can terminate with a decision to accept the new option. Because $\varepsilon^+_{\mathcal{P}/\mathcal{C}}$ is defined as $\varepsilon^+_{\mathcal{P}\cup\mathcal{C}} - \varepsilon^-_{\mathcal{C}}$, this suggests calling either REFINE-DOWN$(\mathcal{P}\cup\mathcal{C})$ with the hope of lowering the value of $\varepsilon^+_{\mathcal{P}\cup\mathcal{C}}$ or calling REFINE-UP$(\mathcal{C})$ with the hope of raising the cost of $\varepsilon^-_{\mathcal{C}}$. A dual argument applies when $v[\mathcal{P}]$ is very close to $\varepsilon^-_{\mathcal{P}/\mathcal{C}}$. Specifically, it is natural to raise the value of $\varepsilon^-_{\mathcal{P}/\mathcal{C}}$ with the hope of pushing this value above that of $v[\mathcal{P}]$, guaranteeing a quick decision to reject. Since $\varepsilon^-_{\mathcal{P}/\mathcal{C}}$ is defined as $\max[0, \varepsilon^-_{\mathcal{P}\cup\mathcal{C}} - \varepsilon^+_{\mathcal{C}}]$, this suggests attempting either to raise $\varepsilon^-_{\mathcal{P}\cup\mathcal{C}}$ through a call to REFINE-UP$(\mathcal{P}\cup\mathcal{C})$ or attempting to lower $\varepsilon^+_{\mathcal{C}}$ through a call to REFINE-DOWN$(\mathcal{C})$. Heuristics such as these cannot be accurately tested on generic cases, but must be explored in the context of specific realistic domains.

Finally, the worst case, in which the size of the stepset lattice is exponential in the number of steps, occurs only with highly artificial plans, in which virtually all of the plan steps share a very small number of types. The size of the associated stepset lattices rapidly diminishes as plan steps become more evenly distributed among types; in the limiting case in which each action in the plan has a unique type, the stepset lattice contains only a single point. We illustrate this point in Fig. 10, showing the number of nodes in the stepset lattice for plans of various lengths, under the assumption that the plan steps are divided evenly into one, two, and four distinct types. Note that the y-axis of this graph is logarithmic—necessary because of the very rapid growth when all steps have the same type—so that, although the trend lines have similar slope, the top line (representing the case where all steps have the same time) actually indicates much more rapid growth than the middle line

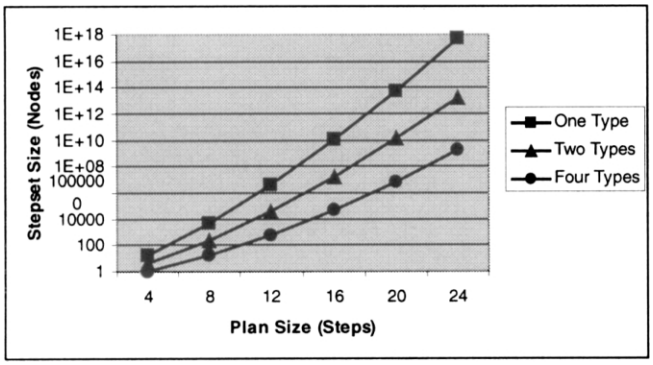

Fig. 10. Growth in size of a stepset lattice.

(representing the case where the half the steps are one action type and half are another), and similarly the middle line indicates much more rapid growth than the bottom line.

5. Conclusion

In this paper, we have developed the foundations of a theory of rational choice that takes seriously the view that agents make decisions in the context of their existing plans. The overall picture we have is one in which an agent maintains a set of commitments, some of which, at any given time, will be only partially specified. A new option for action must then be evaluated taking into account interactions between that option and the background context. Recognizing that agents have computational resource limits, we have developed an account that does not require the agent always to compute the exact cost of an option in context; instead, we have shown how estimates of option cost can support rational choice.

The work done to date provides an initial development of the theory. Both formal and computational extensions are required to increase its robustness. As an instance of the former, consider our current use of particular executions as points in the semantic space. It seems reasonable to generalize that approach and interpret a plan as specifying a set of allowed futures—intuitively, those futures consistent with an execution of the plan. An interpretation along these lines might be developed within the general logical framework of branching time [21,24]. With regard to computational issues, it seems clear that significant work remains to develop heuristics to improve the efficiency of our algorithms.

In addition, we have so far provided details only for a very restricted case of the general problem, and it is clear that the theory must be extended along a number of dimensions. Some such extensions are relatively straightforward. For example, we have discussed only cost savings that result from merging of type-identical steps, but sometimes it is also possible to merge steps of different types that both achieve the particular conditions required in the current context. Also, we have discussed only plans that involve a single, albeit possibly merged, action at a time, but have not here considered plans with true concurrency: two or more steps of different types performed at the same moment. To achieve such concurrency, we could make use of mutex relations as in the GraphPlan literature [2].

Other generalizations will require more significant extensions to the theory as we have developed it in the paper. Most notably, we have dealt here only with complete, primitive plans, all of whose actions are instantaneous. In related work, we have been directly concerned with plans that include rich temporal constraints such as deadlines and durations [26], and we believe that the techniques developed there can be adapted to the problem of step merging in a more realistic temporal framework. Handling incomplete plans, however, requires extending the algorithms for cost computation, because the cost of an incomplete plan depends not just on possible ways of scheduling it, but also on possible ways of completing it. Similarly, handling hierarchical plans requires reasoning about possible decompositions. In both of these cases, methods for plan generation must be interwoven with the cost computation process.

Finally, we have focused our attention in this paper on new options that are at least strongly compatible with the background context. Although this is a reasonable starting

point, it is clearly important to generalize the theory so it can handle other situations. We are currently examining weaker notions of compatibility—for instance, situations in which the union of the new option and the background context results in threats that cannot be resolved with just ordering and linking constraints, but instead require the introduction of new plan steps. In this case, the computation of plan cost must take into account the cost of the threat-repair steps. A further generalization, hinted at earlier, would cover situations of true incompatibility between a background context and a new option, so that adoption of the new option would require real modification, not just augmentation, of the background context.

Acknowledgements

This research was supported by the National Science Foundation grants IRI-9619562 and IRI-9619573 and by the Air Force Office of Scientific Research contract F49620-98-1-0436.

References

[1] F. Bacchus, F. Kabanza, Planning for temporally extended goals, in: Proc. AAAI-96, Portland, OR, 1996.
[2] A. Blum, M. Furst, Fast planning through planning graph analysis, Artificial Intelligence 90 (1997) 281–300.
[3] M. Bratman, Intention, Plans, and Practical Reason, Harvard University Press, Cambridge, MA, 1987.
[4] M. Bratman, D. Israel, M. Pollack, Plans and resource-bounded practical reasoning, Comput. Intelligence 4 (1988) 349–355.
[5] A. Cesta, A. Oddi, S.F. Smith, Profile-based algorithms to solve multiple capacitated metric scheduling problems, in: Proc. 4th International Conference on Artificial Intelligence Planning Systems (AIPS-98), Pittsburgh, PA, 1998, pp. 214–223.
[6] C. Cheng, S.F. Smith, Generating feasible schedules under complex metric constraints, in: Proc. AAAI-94, Seattle, WA, 1994.
[7] M. Ghallab, H. Laruelle, Representation and control in IxTeT, a temporal planner, in: Proc. 2nd International Conference on Artificial Intelligence Planning Systems (AIPS-94), Chicago, IL, 1994, pp. 61–67.
[8] R. Goodwin, R. Simmons, Search control of plan generation in decision-theoretic planners, in: Proc. 4th International Conference on Artificial Intelligence Planning Systems (AIPS-98), Pittsburgh, PA, 1998, pp. 94–101.
[9] R. Grimaldi, Discrete and Combinatorial Mathematics, Addison-Wesley, Reading, MA, 1994.
[10] P. Haddawy, A. Doan, R. Goodwin, Efficient decision-theoretic planning: Techniques and empirical analysis, in: Proc. 11th Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, 1995.
[11] R. Jeffrey, The Logic of Decision, McGraw-Hill, New York, 1965.
[12] S. Kambhampati, C.A. Knoblock, Q. Yang, Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning, Artificial Intelligence 76 (1–2) (1995) 167–238.
[13] H. Kautz, B. Selman, Pushing the envelope: Planning, propositional logic, and stochastic search, in: Proc. AAAI-96, Portland, OR, 1996.
[14] H. Kautz, B. Selman, The role of domain-specific knowledge in the planning as satisfiability framework, in: Proc. 4th International Conference on Artificial Intelligence Planning Systems (AIPS-98), Pittsburgh, PA, 1998, pp. 181–189.
[15] D. McAllester, D. Rosenblitt, Systematic nonlinear planning, in: Proc. AAAI-91, Anaheim, CA, 1991, pp. 634–639.
[16] T.M. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.

[17] R. Moore, Interval Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1966.
[18] J.S. Penberthy, D. Weld, UCPOP: A sound, complete, partial order planner for ADL, in: Proc. 3rd International Conference on Knowledge Representation and Reasoning (KR-92), Cambridge, MA, 1992, pp. 103–114.
[19] M. Pollack, The uses of plans, Artificial Intelligence 57 (1992) 43–68.
[20] M.E. Pollack, I. Tsamardinos, J.F. Horty, Merging plans with quantitative temporal branches, temporally extended actions, and conditional branches, Technical Report TR99-05, Dept. of Computer Science, Univ. of Pittsburgh, Pittsburgh, PA, 1999.
[21] A. Prior, Past, Present, and Future, Oxford University Press, Oxford, 1967.
[22] H. Raiffa, Decision Analysis: Introductory Lectures on Choices under Uncertainty, McGraw Hill, New York, 1968.
[23] L. Savage, The Foundations of Statistics, Wiley, New York, 1954; Second revised edition published by Dover Publications, 1972.
[24] R. Thomason, Indeterminist time and truth-value gaps, Theoria 36 (1970) 264–281.
[25] I. Tsamardinos, Efficient algorithms for merging temporal and contingent plans, PhD Proposal, Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA, 2000.
[26] I. Tsamardinos, M. Pollack, J. Horty, Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches, in: Proc. 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000), Breckenridge, CO, AAAI Press, Menlo Park, CA, 2000.
[27] D.S. Weld, An introduction to least commitment planning, AI Magazine 15 (4) (1994) 27–61.
[28] M. Williamson, S. Hanks, Optimal planning with a goal-directed utility model, in: Proc. 2nd International Conference on Artificial Intelligence Planning Systems (AIPS-94), Chicago, IL, 1994, pp. 176–181.
[29] Q. Yang, Intelligent Planning: A Decomposition and Abstraction Based Approach, Springer, New York, 1997.