

Emergent Planning with Philosophical Agents

William Turkett, Jr. and John R. Rose

Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29208
{turkett, rose}@cse.sc.edu

Abstract

Future mission plans for the space program call for complex autonomous systems that must be able to perform with high levels of reliability. Features of proposed missions include wide-scale distribution of agents, partial observability, and interaction between large numbers of active platforms. These domain features prove to be difficult for traditional planning algorithms. We are developing a planning architecture, based on the integration of continual and negotiated distributed planning techniques, that is capable of supporting reliable execution within such domains. Agents within our architecture perform utility-based localized planning. Alternative plans are evaluated with respect to an agent's philosophy, an ordered layering of principles implemented through utility functions. Loose collaborations between agents are formed dynamically during execution. Through agent interactions, an implicit global plan is developed for the mission. This plan is never fully realized or known by the agents, but is rather an emergent property of localized interactions. We illustrate these ideas through a comparison of our proposed architecture to CASPER, a continual planning approach from NASA JPL, in the context of a typical mission scenario.

Introduction

Future plans for NASA missions call for technology that extends beyond the state-of-the-art in computing. One area of particular interest is control of highly autonomous systems. Autonomous systems have already been employed by NASA in the Remote Agent experiment and are being tested for use in upcoming missions. Potential missions in which autonomous systems are applicable share several characteristics that must be handled effectively by a planning control system. These characteristics stem from the fact that missions will require the execution of multiple parallel tasks, all performed in an unknown and dynamic environment and with minimal interaction from mission-control.

Planning architectures for these types of systems must be able to operate robustly and coherently in environments that present notoriously difficult challenges to the planning process. Problems that planners will face include an environment producing exception cases and unforeseen circumstances, varying constraints on task execution,

incomplete knowledge of the state of other systems, and dynamic interactions between multiple active systems.

Traditional planning algorithms based on generating an entire plan before execution prove computationally complex in domains as large as those that will be seen in future missions. As well, these approaches tend to generate plans that can be quickly invalidated due to changes in the environment. Chien *et al.* at JPL have modified traditional approaches by incorporating continual planning techniques and have realized the benefits of rapid adaptation to changes and reduced reliance on predictive models. Their approach, however, relies on generating an explicit overall plan and provides little explicit support for the generation of collaborative plans (Chien *et al.* 1999).

We are developing a planning architecture that supports the level of autonomy required by future NASA missions through the integration of continual and negotiated distributed planning. Agents within our architecture will perform localized planning, relying on philosophical principles to guide action selection and evaluate plan alternatives. In our architecture, a global plan emerges from the interaction of localized planners. This plan is never explicitly modeled but instead is implicit in the coordination of local planners. By allowing each agent to embody a unique philosophical framework, our architecture supports agents that can exhibit a wide range of tendencies towards how actions are performed and resources are used.

We begin with a review of other planning architectures and of the techniques that are used in our architecture. This is followed by a description of the key components and contributions of our architecture. Finally, we conclude with an example scenario illustrating the execution of our architecture in a surface exploration mission.

Background

Planned NASA missions of the future will require increasingly autonomous computing devices active in all parts of the mission. To be successful, a planning architecture that can operate effectively in dynamic environments and gracefully handle unexpected events is required. Researchers at the NASA JPL AI Group have developed CASPER, a planning system based on continuous planning algorithms. Plans developed in

CASPER are generated in a hierarchical manner and are developed at various levels of abstraction, dependent on the length of the planning period. A current plan is updated continuously to represent changes in the world state from which the goals of the system are then updated. Benefits of using a system such as CASPER include quick response to unplanned events that change the environment and reduced reliance on prediction and modeling methods because the plan, continually updated, is always close to the actual state of the world (Chien *et al.* 1999).

Recent work by Tara Estlin, also at NASA JPL, has evaluated CASPER as a planning system for a set of exploration rovers. Rovers are homogeneous in their abilities in this system, but are capable of localized planning with CASPER to determine how to achieve assigned mission goals. A centralized, non-CASPER planner also exists in the system and is responsible for generation of a batch plan for all rovers and distribution of mission goals to individual rovers (Estlin *et al.* 1999).

Research in Negotiated Distributed Planning (NDP) has primarily focused on two areas: development of negotiation protocols and development of theories that allow for the construction of social concepts from individual mental concepts. Cohen and Levesque's theory of joint intentions is developed to explain how group plans are distinct from a pure aggregate of the individual plans of members of a group. A joint intention is defined as a group commitment to perform actions to achieve a group goal with explicit rules on how a commitment can be formed and broken by the group members (Cohen, Levesque, and Smith 1997). Grosz and Kraus introduce the notion of intention-that, a concept of committing to enter a certain state rather than committing to perform an action. This level of abstraction allows sharing of partially developed plans between agents and provides planners with a large degree of freedom in selecting the actions they will perform. Collaboration between agents begins when an agent moves from desiring to enter a world-state to having an intention that a group of agents cause this state to come about. Once an intention-that is formed by a collaborative group, it influences the intentions and commitments that members of the group will adopt, spawns further planning towards achieving the committed state, and causes agents to perform acts in support of the collaboration, such as monitoring group activity or performing actions that will be useful to other members of the group (Grosz and Kraus 1999).

When a group of agents decides to work together, a means of reaching agreement on how to coordinate tasks is required. To this end, a variety of negotiation mechanisms have been developed. These methods allow agents to come to a joint agreement, while also allowing agents to express their locally optimal preferences. Negotiation mechanisms include determining maximum-utility or minimal-work approaches, allowing agents to vote, and following market protocols (desJardins *et al.* 1999).

A feature of our architecture significantly different from that of CASPER is that an explicit global plan never needs to be completely generated. With our approach, only the

portions of the plan currently under consideration are expounded and interpreted. Collaborative planning in our architecture builds on the basic concepts introduced in NDP theory but is less strict in the mental attitudes required to develop and execute collaborative plans. Additionally, we use a novel technique of implementing philosophical principles through decision networks for expressing preferences during negotiation.

Architectural Design

A review of domains in which we are interested illuminates several mission characteristics that influence our design of an agent control mechanism. Three main features that characterize future mission domains are the size of the environment, the rate of change of the environment, and the uncertainty present in the environment.

The systems employed in future missions and the environments in which these systems will be situated will be much larger than those of today. For many missions, there may exist hundreds or thousands of distinct agents, each of which can perform multiple tasks and has unique preferences for how tasks should be performed. Any given task may also have several possible outcomes. The types of missions being considered will take place over large distances and will likely have limited bandwidth available for communication. The environments of future missions will also evolve fairly rapidly, similar to rates of change seen in everyday human activity. Finally, agents in future mission environments may have difficulty in accurately sensing the environment and will be limited to gathering incomplete information.

Given these assumptions about the planning domain, there exist several natural consequences. The planning state space for the proposed problem is too large to compute a policy and too dynamic to centrally compute good aggregate plans offline. There is a high probability of agents encountering situations in the environment that are unable to be modeled beforehand. Since tasks may have multiple outcomes, the utility of performing an action must be based on the likely outcomes of that action. The ability to predict the time required to complete tasks is also diminished in these types of environments, preventing a centralized planner from generating accurate global orderings. The distribution of agents over a wide area hinders centralized information gathering and global interactions. The reasoning approach taken by agents needs to be capable of handling uncertain and conflicting information and agents should be able to adapt at a rates similar to the rate of change of the environment.

Since actions in these domains may have multiple outcomes, any planning algorithm that is used needs to support probabilistic planning. Additionally, since there are potential situations agents can enter that designers cannot predict, it is impossible to perform exhaustive pre-mission contingency planning. Posing the planning problem as a Markov Decision Problem (MDP), one can

develop a policy over domain states, denoting optimal actions to be performed in each state. With the size of the domains, number of agents, and number of actions that will be required in future missions, the number of states that have to be modeled in a MDP grow too large to solve in reasonable times.

This analysis suggests the use of localized and continual planning for the types of domains we are interested in. Centralized planning would require large amounts of resources to encapsulate the planning search space and would require interactions between centralized planners and task agents across wide areas. Localized planning, on the other hand, distributes the planning space to individual agents and supports localized (proximity-based) and abstract (partial-plan) interactions. Localized planning also supports planning in parallel for independent tasks and can reduce the overall time required for planning. Finally, localized planning helps to limit the effects of inaccurate sensing to a small set of interacting agents.

Continuous planning provides the ability for agents to adapt rapidly to the environment. With continuous planning, an agent's current world model is continuously updated and the current plan in execution is evaluated with respect to this updated model. By adopting continuous planning over alternating periods of planning and execution, agents are freed from having to perform potentially outdated planned actions before reaching the next planning period. Constant updates of an agent's belief set also provide the agent with the most current information when planning. By having information as current as possible, an agent's valuations of actions are more informed. Continual planning allows agents to adapt to changing objectives during execution, facilitating the acceptance of changes to mission goals and requirements (desJardins *et al.* 1999).

Based on this reasoning, we are developing a planning architecture based on localized and continuous task planning and execution by individual agents. Each agent in our architecture attempts to act rationally, planning with dynamic decision networks and selecting actions that are most likely to achieve a goal or successfully perform a task. By using decision networks, which allow for an effective factoring of a Markov Decision Problem, and localized planning, which isolates agents to those actions that are locally important, we reduce the difficulty of the planning problem. Agents will have parallel planning and execution threads to enable continual planning. An agent evaluates locally developed plans and competing actions during plan generation in accordance with the agent's philosophy, where a philosophy implies a utility function implemented through decision networks (Rose and Huhns 2001, Rose *et al.* 2002a).

In order to support planning across an agent society, agents are capable of sharing high-level information about their tasks with other active agents. From individual plans, agents may find that it is useful to enlist the aid of other agents in performing a task, either because the agent cannot achieve the task on its own or because another agent may perform that task better. This leads to the

formation of small coalitions of agents that generate collaborative plans to achieve tasks. Plan evaluation with respect to philosophies can be used to resolve conflicts between competing plans presented by multiple agents. Conflicts are resolved by selecting the plan that has the highest utility - the plan that has the highest probability of success, is most in line with the goals of the mission, and is most aligned with the philosophy of the deciding agent. From the collaborative plans produced by groups of agents and the local filtering of competing plans through utility valuation, an implicit overall global plan *emerges* from localized interactions. Our architecture moves away from traditional contingency planning and its associated complexity, while providing support for the generation of plans that are capable of adapting to changes in the environment, allowing for planning with incomplete information, and supporting integration of planning and execution.

Distributed Planning

There are two standard methods of performing distributed planning with agents: dividing up the task of generating an overall plan among specialized planning agents or localized planning by all agents in the environment. One method of distinguishing between these two approaches is to determine whether the goal is to develop an overarching plan by working together or whether agents are optimizing their own actions and influencing other agents to account for their preferences. In our planning architecture, agents will take the second approach - localized plan generation, optimization, and execution.

While this choice makes agents more complex by requiring that each agent be able to execute a planning algorithm, it also provides several benefits. Localized planning provides a level of fault-tolerance to the system. If a specialized agent were used for scheduling and another for conflict resolution, the loss of such agents would hinder or halt the planning process. By having localized planning, the effect of losing a single agent on the overall planning process is mitigated. Additionally, localized planning agents can gather information through sensors and short-range communication, while a set of specialized planner agents would continuously have to receive sensory information from the other agents executing in the environment to build a correct model of the world when planning. By using distributed planning, the planning space can be conceptually decomposed into smaller planning spaces which are much easier to search and can be evaluated in parallel. A consequence, however, of our localized approach is that there is no expectation that the emergent plan that is developed will necessarily be globally optimal, as the agents will not have global information.

At the individual agent level, dynamic decision networks (DDNs) are the means of implementing utility based planning (Russell and Norvig 1995; Rose *et al.* 2002b). DDNs have several properties that make them suitable for use by localized planning agents. First, DDNs

support agents with limited sensing capability by representing beliefs with probabilistic states and outcomes. DDNs also provide the ability to determine the utility of decisions over multiple time steps, allowing for n-step horizon planning. DDNs support goal changes through updates of utility values and simplified re-planning through resetting of current belief states. If the method in which a task is performed does not highly affect the utility of achieving the task, using DDNs can facilitate domain modeling by supporting the separation of decision-making and task implementation. As an example, the DDN can suggest that a “move towards goal” action has the highest utility, while the actual move mechanism can be performed via low-level domain specific functions.

A dynamic decision network can also handle planning with temporal constraints by defining utility functions over the times that actions will be executed. These utility functions can be used to represent explicit time constraints, such as “Action X must be completed before 9:00”, as well as relative constraints between actions. Determining the utility of temporally variable actions stems directly from time constraint utility functions, as variable rates for performing a task can be probabilistically extrapolated to time estimates (Rose *et al.* 2002b).

The effectiveness of collaborative planning depends on when planning takes place and when enough information has been gathered to generate an effective collaborative plan. In our architecture, this will be handled through updates to utility functions as plans are adopted. When an agent plans certain actions, the utility for other actions related to the chosen action may increase or decrease as necessary. By modeling search for collaborative partners as an increasing utility action when an agent plans for a collaborative task, the process of collaborative planning can be triggered at the points where the agents have enough information to effectively plan.

Agents in our architecture will plan at detailed levels over short-term planning horizons and maintain abstract plans for long-term horizons. This approach prevents agents from becoming nearsighted from one-step planning as well as prevents agents from planning for events too far in the future that have a high probability of being modified or not taking place.

We also propose a set of agents that maintain abstract views of mission goals and the mission state. These agents serve two main purposes: to interact with agents outside of the active system and to perform high-level scheduling and goal distribution. An agent of this type would be responsible for accepting new goals and requirements from outside sources, such as ground-based mission control, and reporting back mission successes or failures. As this agent has knowledge of the priorities of the mission from its outside interactions and an abstract view of the system’s state from knowledge of successes and failures, it can be utilized to provide task agents with the long-term goal sets and goal priorities over which they should plan. High-level task agent can be modeled similarly to the standard task agents in our system – Given goals that one can not achieve on its own from outside sources, a high level task

agent should form collaborations with localized task agents to achieve such goals and should maintain an abstract view of whether or not those goals are being achieved. This abstract view can be provided by task agents via success or failure notification, without having to provide specific information of what actions are being taken or why a failure occurred.

Coalition Planning

Within a real-world system where agents have distinct capabilities, there are often goals to be achieved that require a combination of tasks to be performed by different agents. One approach to solving this problem is to assign agents to teams off-line so that at runtime agents can work together with their team to achieve collaborative goals. In a changing environment, however, flexible and dynamic collaborations between agents are more useful. Dynamic collaborations allow agents to form coalitions on the fly to solve problems. This allows an agent to be able to evaluate whom it wishes to work with based on its current plan and perceived current state of the mission. The ability to dynamically join and exit collaborations prevents agents from being committed to helping others perform actions that do not fit with the agent’s philosophy and beliefs about the world. This approach also facilitates the integration of new agents into an environment. Instead of being assigned statically to a team, a new resource can collaborate as needed with the initial set of executing agents.

There are three main stages to collaborative planning: coalition formation, coalition plan development, and commitment management. These phases are shown in Figure 1 within the context of the general agent collaboration structure. The coalition formation process is spawned when an agent is unable to perform a required task by itself. Potential reasons for being unable to perform a task include that the agent may not have the required capabilities or may not have access to the necessary resources. The formation process consists of three phases: discovery, proposal, and reply. The discovery phase consists of the requesting agent searching for an appropriate task agent to perform the required task. Discovery can be implemented through a number of mechanisms, including broadcast messages, use of a directory service, or direct knowledge of agent capabilities. Once a task agent has been found, the requesting agent sends a proposal to the task agent that includes an abstract plan for achieving the proposed task. The task agent reviews the proposed task plan and evaluates the schedulability and utility of adopting the requested task. If the task agent believes executing the task is beneficial, an accept reply is given and the coalition plan development stage is entered. Otherwise, the agent can reject the proposal in the reply phase, causing the requesting agent to search for other task agents or to re-examine its own plan.

If two agents decide to collaborate, they then must develop a collaborative plan. The plan proposal stage plays a large part in this negotiation process. A task agent

should not accept a task unless it is beneficial to the agent as defined by the agent's philosophy. Thus, when an agent agrees to perform a collaborative task, it agrees to the basic plan structure and constraints provided by the requesting agent. Upon accepting a task, an agent may perform a period of plan evaluation with knowledge of the additional task to determine plan optimizations. Revisions to the collaborative plan are forwarded to the task-requesting agent if the revisions alter the original commitment. If the requesting agent accepts the proposed revisions (by evaluating them with his own metrics), the collaborative plan is finalized. A worst-case scenario, where revisions are not accepted, should still allow for collaboration as outlined in the original proposal and commitment.

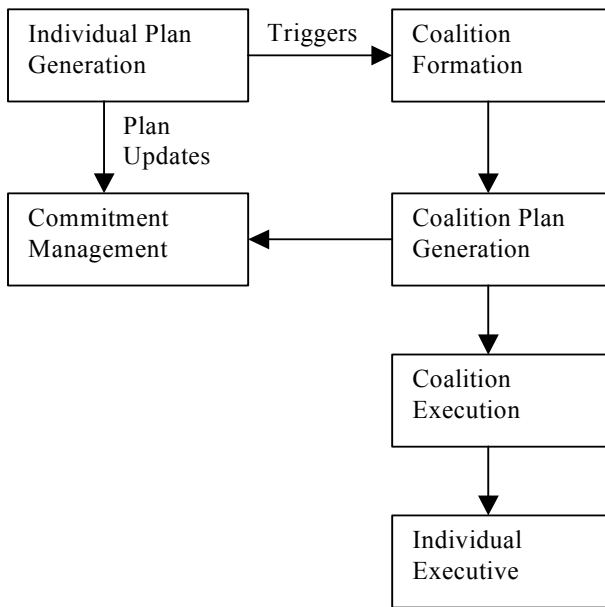


Figure 1. Coalition Planning Process

When an agreement is made on a plan, agents create commitments to each other to achieve their respective parts of collaborative tasks. Respecting this commitment is one of the keys to the success of distributed planning. Once a collaborative plan is finalized, agents store information about the commitment they have made, including the action to be performed, any constraints on the action, and agents that are dependent on the performance of the action. Agents use commitment information to support task notification, determining who needs to be notified when a collaborative task is completed or the plan for a collaborative task is updated. These plan updates can include updates for when a task has been rescheduled, when a task is taking longer than expected, or when an agent believes a task will fail. Commitments are also used when evaluating alternative plans. If a commitment is to be broken in an alternative plan, a negative utility is included in the utility valuation for breaking the

commitment. This negative utility supports environments where social and mission directed activities are of high importance. It also helps to prevent a wave of broken commitments if there are high levels of task and agent interdependence and deters agents from being myopic and constantly switching plans in attempts to reach local optima.

In collaborative planning, there are no agents that are specifically designated as leaders. However, there are two instances where leadership-type roles are taken. The first is high-level task agents. These agents maintain knowledge of the mission goals and have a broad view of what should happen during each planning period (as an example, they may know that the high-level goals for today are to perform imaging of the lander area and to drill for samples). These agents should also receive notification of goal successes and failures to be relayed back to mission control and to be evaluated to determine the appropriate goals for the next planning period. This role, however, is intended to be primarily passive and is not involved in directly managing how individual tasks are performed or how agents interact. The second leadership type role is seen in collaborations. In collaborations, the agent requesting assistance can be viewed as a form of leader. In achieving his goals, he has the responsibility of finding an agent or group of agents that can assist in executing a task. A task agent, when it joins a collaboration, is constrained to work within the proposed bounds provided by the requesting agent. When multiple agents are working together, the requesting agent's constraints are the primary guiding force for focusing individual efforts.

By having distributed agents that collaborate to achieve goals, additional communication costs will be incurred under our architecture. There are four main types of communication that are required. The simplest is task signaling, the process of notifying other agents when a task is completed. Task signaling should be of minimal cost as it is akin to sensing task completion in other architectures and could potentially be implemented through physical sensors. The coalition formation and plan development processes do, however, pose additional overhead. Coalition formation involves communication between a requesting agent and a number of potential task agents. The actual amount of communication depends on the dependency of tasks in the mission, the number of agents that can fulfill a task, and the number of agents that must be contacted before an agreement is made. Thus, for highly independent tasks, there is minimal additional overhead. However, in systems where there are highly interdependent tasks and a large number of active agents, the communication effects are no longer negligible. It is our hope that the costs of coalition formation and coalition plan development will be small relative to the long-term benefits of developing collaborations. The communication requirements for plan development are dependent on the compatibility between the requesting agent and task agent plans. In the best case, there will be minimal overhead if the task agent is content with the proposed plan and schedules it for execution as-is. Otherwise, there may be

several rounds of revisions before agreement is reached and collaboration begins.

The highest computational costs may arise from the process of updating commitments. If commitments are kept, task signaling will satisfy most commitment requirements. However, if commitments are broken or updated, all agents dependent on the commitment (and transitively, agents dependent on those agents) will need to be notified and potentially re-plan.

The four types of communication described above are the basis of the communication requirements needed for our architecture. Other forms of communication may be useful, such as intermittent updates of task progress or high-level sharing of plans and sensory information, but are not specifically required.

Competitive Planning

As agents build localized plans, there is the possibility that conflicts will arise between plans and between agents. Our architecture supports resolution of such conflicts through the concept of competitive planning. In competitive planning, the utility of different plans is evaluated with respect to an agent's philosophy, which incorporates principles related to the state of the overall mission, individual beliefs, and potential plan effectiveness.

Plan evaluation within an agent is required for an agent to generate and execute what it believes to be a locally optimal plan. There are often several possible methods in which a task can be performed and each method has different costs and different probabilities of success. These factors are taken into account to select the set of actions that have the highest current utility. Plan evaluation is also useful when agents are collaborating. As previously explained, agents can use plan evaluation to determine whether or not collaboration is useful and if there are alternative approaches that can be taken. Finally, there will often be times when multiple agents require the use of the same resource or request services from the same agent. This is the case in which the concept "competitive planning" is most appropriate within our architecture. When an agent receives several different requests, it evaluates each request relative to other requests as well as with respect to its own plans. This agent, in the most general case, selects the plan that has both the highest utility to the mission as well as fits within the agent's future plans. The agent does not necessarily have to create a win/lose situation, but could offer a win/win schedule where two competing agents both receive services or are both allowed to use a resource, but under marginally different constraints than the ones they requested.

Emergent Planning

In our architecture, planning is distributed among agents that are acting in the environment, with each agent generating localized plans and attempting to accommodate the local plans of other agents as needed. The goal of each agent is not to build components of an overall plan to

achieve mission goals, but instead to generate small and dynamic plans that fit individual needs within the mission. From local interactions, a global plan emerges as the aggregate of individual plans. This global plan is, however, never explicitly defined and known to all of the agents. Figure 2 depicts how a plan develops through the local interactions of agents. The process of competitive planning helps lead the agents towards the emergence of a coherent and coordinated aggregate plan. By using competitive planning, agents implicitly act as a filter, preventing the execution of plans that are least likely to help achieve mission and individual goals.

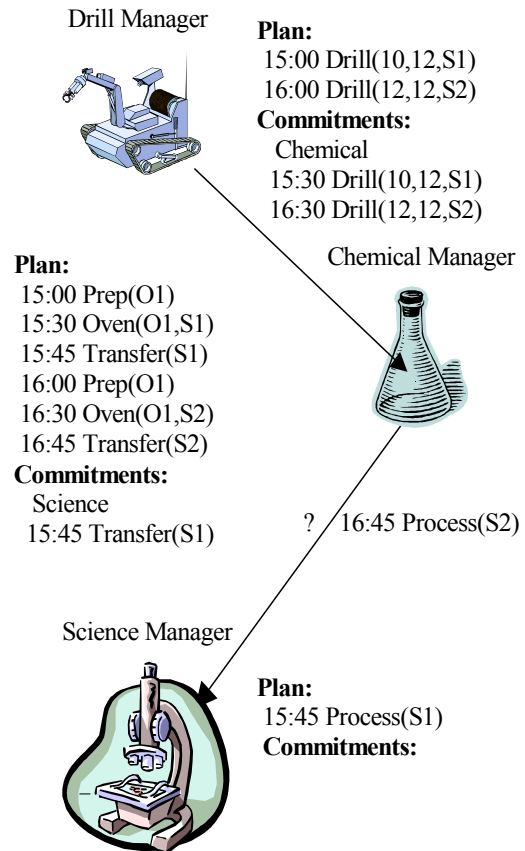


Figure 2. Emergent Global Planning Through Local Interactions

The benefits of an emergent global plan are numerous. The traditional model in which an overall plan is generated, divided into tasks designated for independent agents, and then executed performs poorly in environments of interest. The complexity of developing an overall plan is too large for environments as complicated as those that will be seen in NASA missions. Global re-planning in the face of changes in the environment is also similarly complex. An alternative to re-planning, enumeration of all possible states before planning to handle contingencies, is

very complex in large environments and is impossible for environments where humans as designers do not have complete knowledge of all possible states. Local plans can be updated easily when changes occur in the environment. The task of updating plans can also be fairly well localized to the subset of agents that sense changes in the environment and their immediate collaborators.

Philosophical Principles

The final key component of our architecture is that each active agent follows a set of philosophical principles that guide the agent in its choice of actions. These principles are the means by which plan evaluation can be performed by the agents. The philosophical principles, along with the current state of the world and agent, guide the decisions required by any form of competitive planning in the environment, including whether or not to join in a collaborative group, how to schedule resources to multiple requesting agents, determining which actions to take towards one's own goals, and negotiating to develop a collaborative plan.

The philosophy of an agent is concerned with primarily two areas: attitudes towards other agents, and attitudes concerning mission and individual goals. An agent's attitude towards other agents corresponds to the level at which the agent is interested in helping others to satisfy their localized needs. On the other hand, an agent's goal-based philosophy corresponds to the level at which an agent is interested in achieving its own goals versus those of the mission. Using these two centers, it is possible to develop agents that range from being self-interested and isolated to extremely benevolent and social.

Within each agent, the agent's philosophy is defined by a layered set of philosophical principles. Figure 3 represents a set of socially directed principles that could be used in developing an agent. The layering of these principles represents the relative importance of each principle to the agent, with the most important principles layered highest. This layering can be mapped to a means of computing the relative utility of actions that the agent is considering to perform, and thus allows for relative comparisons of plans (Rose and Huhns 2001; Rose *et al.* 2002a).

Every agent acting within the environment will have the same underlying data-structure for representing their philosophy, but agents are not required to be homogeneous in the principles and ranking of principles that they adopt. It is our intention that agents with a wide variety of philosophies will exist at the same time. Philosophies will be tuned to the roles that the agents are playing within the domain. As an example, resource agents may be conservative or lenient depending on the availability of resources, while exploration agents may be risk-adverse, risk-neutral, or risk seeking, dependent on the goals of the mission and the independent agents.

Although different agents can have different philosophies, the basis philosophy of architecture is biased towards success of the mission and being socially pro-

active. This philosophy tends to support actions that are directed towards successful achievement of mission goals and providing support to the other agents in the mission. Successful application of this type of philosophy allows for agents that maintain their promises and commitments to others, help others agents willingly, and are capable of repaying kind acts. When required, however, agents have the capability of falling back on selection of actions that satisfy immediate personal needs (Rose and Huhns 2001).

1. An agent shall not harm the mission through its actions or inactions.
2. An agent shall not harm participants in the mission.
3. An agent shall not harm itself.
4. An agent shall make rational progress towards mission goals.
5. An agent shall follow established social conventions.
6. An agent shall make rational progress towards its own goals.
7. An agent shall operate efficiently.

Figure 3. Socially Directed Philosophical Principles

Decision Theory

To use the philosophical principles outlined previously as a means of evaluating plans, our architecture views the principles an agent adopts as a set of preferences for the agent. From this point, the preferences are mapped to utilities via utility functions. The decision network formalism can then be used to combine probabilistic knowledge of the domain and the preferences expressed by the utility functions. The evaluation and ranking of competing actions can then be explicitly calculated. By implementing philosophies in terms of decision networks, we use a trusted method to choose actions that are consistent with an agent's goals and beliefs. Additionally, decision networks are applicable to estimating the probability of success in execution of a plan. Together, this allows our agents to choose the actions that are most in-line with the values and goals of the agent and the mission, as well as those that are most likely to succeed in achieving such goals. The use of decision networks allows an agent to use the maximum expected utility principle,

and from this, allows agents to act rationally. Algorithms for decision networks are also relatively computationally efficient, reducing the infrastructure and time constraints required for planning agents. Agent Encapsulated Decision Networks (AEDNs) have been recently introduced and provide us with a means of embedding each agent with the required decision theoretic algorithms (Bloemeke 1998; Valtorta, Kim, and Vomlel 2002).

Decision theory also supports value of information (VOI) calculations. VOI is extremely important in applications where agents are required to plan with incomplete information. VOI can determine when an agent has sufficient information concerning its current plans to announce and execute plan fragments. This prevents agents from wasting valuable time and resources during planning gathering information that will not influence the plans that are being constructed. VOI also provides a secondary means of evaluating plans. Plans to achieve tasks that provide the most useful feedback and information into the system should be rated relatively higher than those with low VOI scores (Russell and Norvig 1995; Valtorta and Huhns 2001).

An Example Scenario

One of the key domains in which effective planning and scheduling among multiple agents is required for coherence is the exploration of remote areas to collect interesting scientific data. Chien *et al.* (Chien *et al.* 1999) have developed several test scenarios for the CASPER planning system that simulate these types of missions. Several planned missions (Techsat-21, Deep Space 4, and 3 Corner SAT) have scenarios that are similar in the actions that are being performed and the level of difficulty required for planning. One of these scenarios is based on the current plans for Deep Space 4 and its surface study of the comet Tempel 1. To evaluate the makeup of Tempel 1, NASA plans to maneuver and anchor a lander onto the comet surface. The lander is then tasked with goals focused in three areas – drilling, scientific analysis, and imaging. Drilling consists of mining at three different depths at each of three different drill locations to extract samples. One set of samples is to be returned to earth, while the others will be analyzed on site. Scientific analysis goals consist primarily of gathering gas chromatography/mass spectroscopy data from each sample, as well as examination with an infrared/spectrometer microscope and performing gamma-ray spectroscopy experiments. There are currently plans to provide the lander with two chemical ovens, providing redundancy if an oven fails and the ability to interleave experiments. Imaging of the comet surface will be managed by several cameras that are onboard the lander. Data gathered from analyses will be stored onboard in a limited set of memory buffers and can be up-linked to an orbiter as necessary.

The CASPER system has been shown to be able to adapt to major failures within this scenario including exogenous

state failures, such as the loss of a chemical oven; exogenous resource conflicts, such as over-allocated data buffers; and activity updates, such as late-finishing tasks (Chien *et al.* 1999). We have extended this scenario to incorporate an additional task of science processing. Science processing, being performed on the upcoming Techsat-21 mission (Chien *et al.* 2002), allows agents to dynamically add goals during mission execution to handle interesting and opportunistic science data.

Although this type of scenario is relatively limited (compared to long-term mission plans) in terms of the number of executing systems and activities that can take place, it has several features which lead us to believe that localized planning would be effective as an agent control mechanism. This scenario also provides a basis for comparison between centralized, continuous planners such as CASPER and our proposed planning architecture. Through this scenario, we explain how our architecture should be able to handle the same types of problems as CASPER, as well as demonstrate areas where our architecture can show improved performance.

In CASPER, a global plan is generated by a central planner with complete knowledge of the state of the lander. Under our planning architecture, distributed and independent agents are responsible for developing localized plans for achieving mission goals. Using our architecture, assume that there are seven agents that co-exist within the lander platform and are responsible for handling the major tasks of the mission as follows:

- Drill Manager – Responsible for performing drilling activities
- Chemical Manager – Responsible for chemical analysis of drilled samples using two redundant chemical ovens
- Science Manager – Responsible for on-board data processing for significant events and interesting data
- Imaging Manager – Responsible for camera operations and surface imaging
- Power Manager – Responsible for allocating power to parts of the lander, recharging, and moving
- Data Manager – Responsible for buffering scientific data and uplink to an orbiter
- High Level Task Manager – Responsible for abstract plan management and input from ground control

Plan evaluation by agents at three different levels - localized, collaborative, and competitive - allows for a coherent plan to develop out of the interactions of these agents even when they face failures and faults in the system.

As previously mentioned, CASPER is capable of handling three major types of failures in this mission (Chien *et al.* 1999). The first type of failure considered is when a resource becomes unusable. An example of this type of failure is an inoperative chemical oven, a resource that is normally used for analysis of drilled samples. In our architecture, this failure would first be discovered via sensory monitors of the chemical agent. These sensory inputs update the current beliefs of the agent's dynamic decision network, and if the inoperable oven is in the

current plan, cause the agent to re-plan. By incorporating the failure into the agent's beliefs, future decisions to use the failed oven should now have low utility, and the high utility plans selected for execution would be those in which future experiments are scheduled on the operational oven. Depending on the state of the chemical ovens and how far ahead plans have been developed, it is possible to isolate this failure to only the chemical agent. However, if new tasks need to be introduced, such as a repair task or a prep period for the second oven, task achievement delays would need to be propagated to the agents that are expecting the results of an analysis process.

A second type of failure that can occur in this scenario is when a resource is over-allocated. In the CASPER example, this type of failure occurs when a scheduled scientific experiment generates new data that overflows the data buffer resources. Under our architecture, the storage of scientific data requires the interaction of two agents, a data producer (such as the chemical agent or science agent) and a data consumer (the data manager agent). The producer's goal, analyze a sample, leads to a maximum utility plan similar to [prepare to analyze a sample, perform the analysis, store results]. Since the producer cannot store results on its own, collaboration is required with the data manager agent. By agreeing to collaborate and accept a data storage task, the data manager re-evaluates its current plan. In light of the agent's commitment to accept new data, the utility of performing an upload of current data to free memory buffers is high over the next several time periods. This failure can be isolated to the data manager agent.

The third type of failure successfully handled by CASPER is a task that takes longer than expected to complete. An example of such a failure is a drilling task that is slowed by rock denser than predicted. The failure to finish a task within a scheduled (and committed to) timeframe produces a temporal conflict over potential actions within an agent, triggering re-planning to resolve the conflict. The alternatives the agent faces are to continue its expected task, notifying other agents of a new estimated completion time, or to switch to the alternative task that the agent had originally been planning to perform in the next time period. Either of these alternatives may be chosen dependent on their respective utilities.

This case, however, is probably the most complicated in our scenario, as the utility of each alternative may be based on several factors: how far along the agent is on the initial task, the mission-level priority of the alternative tasks that can be pursued, dependencies between the agent's local tasks, and dependencies between collaborating agents for task execution. This example also demonstrates the potential for a task-failure to lead to a chain-reaction failure in the system. A cascade of commitments between agents could be broken if a highly interdependent task is completed late or not completed at all. This cascade should be mitigated by the inclusions of negative utilities as explained previously under coalition planning.

Besides being able to handle the same types of failures as other continuous planners, our proposed architecture has

several other features that make it attractive for application in this scenario. One such feature is that planning can be performed in parallel for independent tasks since each agent is capable of independent processing. This has the potential of significantly reducing the amount of time required for planning and allowing more time to be focused on execution and monitoring. Our architecture also supports localized sensing and knowledge management. The knowledge required by each agent is restricted primarily to its own domain. As an example, a drill manager only needs to focus on drilling activities. If a chemical oven fails, the drill manager may never be made aware of such a problem. The chemical manager in charge of the oven would need to reschedule activities on the operational oven and alert with a revised schedule only those agents with whom it needs to alter commitments. Compared to a centralized planner that would need to continuously receive sensory updates from all parts of the lander, this localization of knowledge can also reduce the communication bandwidth required for sensing.

This scenario demonstrates at a basic level how a coherent overall plan can emerge from the interaction of localized planners. There is no central authority managing how the plan is built, nor are the agents all working together to generate one large plan before execution. Instead, collaborations are developed between agents in order to achieve goals dynamically during execution. Conflicts are resolved by evaluating conflicting plans according to local philosophies that are primarily biased towards the good of the society and achievement of mission goals. Plans are evaluated relative to the state of the mission and the state of the world during execution, allowing agents to quickly adapt to changes in the environment.

Most of the descriptions of planned missions available in the literature are single platform based or rely on a small set of platforms controlled by a single planning mechanism. In the future, however, a typical mission is likely to require large numbers of independent platforms working together to achieve mission goals. This type of scenario is hinted at in Tara Estlin's work on CASPER based Mars rovers (Estlin *et al.* 1999) and can easily be envisioned for any type of wide-scale mission such as surface exploration or terra forming. In these missions, there will be orders of magnitude more agents active in the environment and each of these agents will have different roles to play in the mission. Not only will there be high levels of interdependencies between separate tasks, but there will also be tasks, such as moving large building materials, that require the simultaneous execution of a group of agents. The large number of agents and scale of these missions will lead to agents being located large distances apart, effecting how communication between agents is performed and which agents are available for interaction. The potential to enter into uncertain states or base decisions on inaccurate sensor readings is also magnified in these types of situations, especially when missions are very exploratory. These types of missions provide a strong argument for using localized planning for

coordination and are the scenarios towards which our architecture is most applicable.

Conclusion

The planning architecture we are developing provides for mission robustness, effective planning, and successful goal achievement, even when planning in dynamic and uncertain environments (domains that are traditionally difficult for planners). Our architecture is novel in that it utilizes decision-theoretic philosophical principles embedded in agents as a means of plan evaluation. Agents use these principles to evaluate plans at several different levels – within an agent for evaluating local plans, within a collaborative group for evaluating joint plans, and between agents for evaluating competing plans. The use of decision theoretic methods also supports planning under uncertainty. As a result of plan evaluation and the interleaving of planning and execution, our architecture allows for the development of an implicit global plan that emerges naturally from the localized interactions of agents. Our architecture also explicitly supports dynamic collaborative planning as an extension of localized planning.

Agents that are developed around our planning architecture will be able to adapt quickly to changes in the environment, including taking advantage of serendipitous opportunities. The high costs of explicit global plan generation and the potential for executing outdated plans are reduced as planning is divided among distributed agents and performed in parallel with execution. By generating plans at local levels, agents can share partial plan information and evaluate what information they need to continue planning, allowing for effective plans even when agents can only gather incomplete information from the environment. Localized interactions reduce problems stemming from limited bandwidth and the distribution of agents across large areas. Finally, by distributing planning to the local level, effects stemming from agent faults can be minimized relative to architectures that depend on specialized planning agents.

References

Bloemeke, M. 1998. Agent Encapsulated Bayesian Networks. Ph.D. Dissertation, Department of Computer Science, University of South Carolina.

Chien, S.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 1999. Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling for Autonomous Spacecraft. In *Proceedings of IJCAI99 Workshop on Scheduling and Planning Meet Real-Time Monitoring in a Dynamic and Uncertain World*. San Francisco, CA: Morgan Kaufman.

Chien, S.; Sherwood, R. Rabideau, G. *et al.* 2002. The Techsat-21 Autonomous Space Science Agent. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems*. 570-577. New York, New York: ACM Press.

Cohen, P., Levesque, H., and Smith, I. 1997. On Team Formation. In *Contemporary Action Theory, Volume 2: Social Action*, eds. Holmstrom-Hintikka, G. and Tuomela, R. 87-114.

desJardins, M.; Durfee, E.; Ortiz, Jr., C.; and Wolverton., M. 1999. A Survey of Research in Distributed, Continual Planning. *AI Magazine* 20(4):13-22.

Estlin, T.; Rabideau, G.; Mutz, D.; and Chien, S. 1999. Using Continuous Planning Techniques to Coordinate Multiple Rovers. In *Proceedings of IJCAI99 Workshop on Scheduling and Planning meet Real-time Monitoring in a Dynamic and Uncertain World*. San Francisco, CA: Morgan Kaufman.

Grosz, B. and Kraus, S. 1999. The Evolution of SharedPlans. In *Foundations and Theories of Rational Agencies*, eds. Rao, A. and Wooldridge, M. 227-262.

Rose, J. 2002. Dynamic Decision Support for Command, Control, and Communication in the Context of Tactical Defense: Office of Naval Research Grant Number N00014-97-1-0806 Final Report. Department of Computer Science and Engineering, University of South Carolina. <http://www.cse.sc.edu/research/dag/docs/FinalOnrReport.pdf>

Rose, J. and Huhns, M. 2001. Philosophical Agents. *IEEE Internet Computing* 5(3):104-106.

Rose, J.; Huhns, M.; Sinha Roy, S; and Turkett, Jr., W. An Agent Architecture for Long-Term Robustness. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems*. 1149-1156. New York, New York: ACM Press.

Russell, S. and Norvig, P. 1995. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, New Jersey: Prentice Hall.

Valtorta, M. and Huhns, M. 2001. Probability and Agents. *IEEE Internet Computing* 6(6):77-79.

Valtorta, M.; Kim, Y.; and Vomlel, J. 2002. Soft Evidential Update for Multiagent Systems. *International Journal of Approximate Reasoning* 29(1): 71-106.