A Collaborative Model for Astronaut-Rover Exploration Teams

Maxime Ransan                                         Ella M. Atkins
University of Maryland Space Systems Laboratory
Neutral Buoyancy Research Facility, College Park, MD 20742
{mransan@umd.edu, atkins@eng.umd.edu}

Abstract
Astronauts and robots must collaborate effectively to safely and efficiently explore harsh extraterrestrial environments. This paper describes a basic planning tool to distribute tasks among a hybrid human-astronaut team, then examines likely scenarios in which an astronaut either deviates from the plan (e.g., to opportunistically explore) or requires assistance from a rover (e.g., when in distress or when a task can benefit from a human-robot execution team). Through adjustment of the search-space branching factor, the planner is able to act in a reactive capacity to redirect activities when a task is not completed as planned or when the astronaut requires assistance. Preliminary results are presented to illustrate system performance during nominal and off-nominal operational scenarios.

Introduction
In future exploration missions astronauts will work with robots to accomplish ambitious scientific and engineering tasks (Fong and Nourbakhsh 2005). Planetary surface rovers must be capable of autonomous navigation and manipulation, and they also must appropriately interact with their human companions. A variety of mission scenarios will involve navigation to a set of locations where tasks such as site inspection, sampling, or construction must be completed. Cost and efficiency factors may favor specialist rovers that team to accomplish a broad range of tasks. With such a team, rovers must be assigned tasks based on their capabilities as well as expected traversal cost between worksites.

Rather than directing activities from a remote control station, astronauts will move in the same environment as their robot companions. Each astronaut will be assigned an initial set of tasks to accomplish, some of which may be completed alone and others that will require the assistance of one or more rovers. Rovers may support astronauts by providing tools, storing samples, or relaying sensor data to a mission control station. Any rover tasked with directly supporting an astronaut activity will follow the lead of the astronaut, understanding their high-level role but requiring real-time characterization of astronaut motions to effectively operate at the same worksite. Even for missions in whom rovers and astronauts operate at distinct worksites, rovers may be constrained to execute only nearby tasks so that they can quickly reach the astronaut to provide support when requested. Rovers may be equipped with life support supplies as well as tools and sensors, providing improved safety for the astronauts as well as increased operational efficiency.

Activity planning and execution for a rover team has previously been investigated with systems such as CLEaR (Closed-Loop Execution and Recovery) (Estlin et al 2005a/b). CLEaR relies on the ASPEN planner, which incorporates an iterative repair algorithm to facilitate dynamic response. This system has been successfully demonstrated but has not yet been extended to reason about direct collaboration with humans. Other researchers (e.g., (Fong Thorpe and Baur 2003)) have specifically focused on the interaction between humans and robots, but primarily with the humans as “supervisors” or “reasoning experts” rather than as “mobile agents” operating in the same environment as the rovers.

This paper focuses on activity planning and execution for a collaborative astronaut-robot team. Figure 1 illustrates a mission in which multiple tasks must be completed by a team of two astronauts and three rovers.

![Figure 1: Astronaut-Rover Team Task Execution.](image-url)
Figure 1 flags represent tasks to be performed at each location (e.g., dark gray for rock sampling, striped for construction, light grey for image capture). The rovers are also marked with squares representing tools they can use to accomplish specific tasks (e.g., a grasp/scoop end effector to retrieve a rock sample). As illustrated, rover2 is capable of performing construction and image capture tasks but cannot execute the rock extraction task shown at location 1. Astronauts are presumed to always be capable of undertaking a task, but in certain cases task completion by an astronaut requires the assistance of a rover.

Provided all marked worksite targets are accurately mapped and task completion requirements are known in advance, the task allocation/planning problem is straightforward, and task schedules can be optimized based on traversal distances and task execution capability and cost constraints. However, extraterrestrial worksites are likely not well-mapped, at least initially, and static plans cannot capitalize on the acute sensing and real-time decision-making capabilities of astronaut team members. In reality, as an SEVA (surface extra-vehicular activity) mission progresses, astronauts can be expected to frequently alter the “default” plan to explore new targets of opportunity or to adapt when unanticipated situations arise. The rovers, of course, can also plan for contingencies and search for opportunities, but such behaviors would likely be quite limited in scope thus more predictable.

Astronaut safety is of paramount importance in all SEVA activities. Drawing analogy to the “buddy system” used by hikers and divers, the Figure 1 scenario calls for each astronaut to be accompanied by a rover that need not directly participate in the astronaut’s task, but that must not stray further from the astronaut than some maximum distance. Unless assistance is requested or a cooperative task is scheduled, the “buddy” rover is free to complete its own tasks and periodically verifies the astronaut is safe through sensors (e.g., visual) or direct communication. In Figure 1, rover1 is the “buddy” that must remain within the safety region of astronaut1. This rover cannot reach location1 to accomplish the rock sampling task; so rover3 must be assigned instead of rover1. Should the astronaut require assistance, rover1 can quickly provide life support or other backup equipment without extended traversal.1

Robust fully-cooperative mission planning and execution will require substantial research in areas ranging from dynamic, multi-agent task allocation to cooperative human-robot motion planning. Below, we describe basic planning/execution software to build and adapt cooperative exploration plans. For this paper, we present preliminary simulation results that illustrate the initial planning process and the ability of the planner to adapt to user-injected emergencies and deviations from the commanded plans. We also describe future work to incorporate a rover and “laboratory astronaut” to better investigate the challenges imposed by two “real” agents that each possess a unique set of preferences and capabilities.

### System Overview

To minimize implementation overhead, a centralized planning and execution system has been implemented, as illustrated in Figure 2. The planner directs the activities of each “agent” based on its goals and feedback from the agents (e.g., locations, available energy, capabilities). The planner also responds to directives issued by the astronaut team members. Such directives range from “I need help” to “I’m doing another task” to “Tell a rover to execute this new task” (defined by the astronaut). With the centralized planning structure, communication between astronauts and rovers is limited to data associated with the accomplishment of common tasks, with all other messages relayed through the planning/execution system.

Our application is composed of three interacting components:

- **Planning Agent**: The centralized planning module takes the current state of the world and computes a plan (a set of high-level actions) so that a maximum number of tasks can be accomplished by the team of rovers and astronauts. The planner can be configured to react quickly or consume more time to optimize its plan.

- **Execution Agent**: The execution module controls the flow of information between the planner and the distributed set of rovers and astronauts, dispatching actions in real-time in accordance with the nominal policy and any dynamic updates. The execution agent also gathers all the state information from the rovers/astronauts to enable detection of off-nominal events. This agent is responsible for managing the reaction to such occurrences, either by adjusting the policy (e.g., task execution timings) or by notifying the planner that a new/modifed plan must be created. For instance, when an astronaut requires help, the execution module will receive the information, detect the event, notify the rovers, and call for the planner to compute an emergency plan as quickly as possible.

- **Mobile Agents**: Each rover and astronaut is simulated by a unique process that communicates with the execution module via the network. Rovers are assumed to understand high-level directives (task descriptions) and either to correctly execute them or return an error message indicating the task was not successfully accomplished. Although only simulation results are reported in this paper, the software is configured such that the simple simulated agents can be replaced with “real” robotic and astronaut agents, ultimately enabling the planner to direct an integrated simulation-robot-human team in future tests.

### Activity Planning for Rover-Astronaut Team

The cooperative exploration scenario described above implies numerous constraints and significant planning complexity given a non-trivial task set. Each rover or astronaut has a limited set of time/energy resources...
implying it may not be possible to accomplish all tasks even without off-nominal circumstances. Unanticipated events can also occur during execution, and quick responses must be computed to ensure safety. Finally, optimization is also a crucial issue, given the importance of efficiency (e.g., no “infinite A/C power supply) and the need to achieve the maximum possible set of tasks.

Numerous planning techniques have been developed to find optimal solutions, but all must be simplified to guarantee completion within time constraints. Our search-space is analogous to the well-known traveling salesman problem (Johnson and McGeoch 1997) with exponential complexity. We have implemented a reconfigurable planner that trades off optimality and speed based on available planning time. Typically, an optimal solution will be requested for offline planning (e.g., during a Martian night), while contingency plans (e.g., to rescue an astronaut) will emphasize timeliness over optimality. Below, a general set of planner states and actions are described for the cooperative exploration problem. Next, details of the search strategy are presented.

Problem Representation

Based on anticipated planetary surface missions, a set of state features and operators are defined to characterize the task-level goals, agents, and associated resource requirements/availability. Three state variable types were defined and uniquely indexed (i): tasks (\textit{task}_i), rovers (\textit{rover}_i), and astronauts (\textit{astronaut}_i).

Each task (\textit{task}_i) is defined by the attributes listed below in Table 1. Locations are discrete instantiated symbols defined by x and y coordinates (e.g., \textit{loc1 is situated at (45.2, 17.2)}). The tasks represent goals that the rovers and astronauts must complete, after which \textit{done} is set to \textit{true}. Tasks are defined as a certain \textit{type}, constraining their assignment to only those mobile agents with the proper equipment. Preconditions (previous actions) are also provided for situations in which execution of a certain task requires the agent to have already executed another set of tasks. Ultimately we will incorporate a suite of functions to compute values for quantitative attributes (energy and duration), expected to be unique for each different rover and each astronaut. In the current implementation, each task has a constant (approximate) energy cost, representing the level of effort (resource expenditure) expected to accomplish this task. For the rover, this cost would represent power and fuel required, while for the astronaut, it would represent physical exertion level and associated oxygen use. Constant expected task duration (execution time) values are also specified for each task in the current implementation, although durations are also expected to vary by agent and will benefit from function-based computation in the future.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>pointer to a location object ((x,y))</td>
</tr>
<tr>
<td>type</td>
<td>{ ROCK_SAMPLING, IMAGE_CAPTURE, IR_ANALYSIS, DRILLING }</td>
</tr>
<tr>
<td>previous actions</td>
<td>list of pointer to previous tasks</td>
</tr>
<tr>
<td>done</td>
<td>{true or false}</td>
</tr>
<tr>
<td>energy_</td>
<td>\in\mathbb{R}, expected amount of energy required to accomplish the task.</td>
</tr>
<tr>
<td>duration_</td>
<td>\in\mathbb{R}, expected time needed for the task to be executed.</td>
</tr>
</tbody>
</table>

Although the above definition is somewhat rigid, expansion to a function-driven cost model is in progress. Feedback from field operation can also be incorporated within these functions to dynamically tune estimates based on real-world conditions.
Attributes for the rover state variable are listed in Table 2. Similar to the task state variable, each rover has an (x, y) location and a set of task types for which it is equipped. For the planning problems presented in this paper, therovers are assumed to be fully able to navigate between task locations and to reliably accomplish the tasks they are equipped to handle. Rovers can also help an astronaut in an emergency (e.g., by providing oxygen or power when the astronaut runs low) or when the astronaut requires a specific tool onboard the rover. Although these cases will later be clearly delineated by priority, currently both are modeled in the same fashion: the rover becomes the “buddy” companion of the astronaut. The status variable indicates the current state of the rover, including nominal and exceptional situations. IDLE indicates the rover is waiting for its next task, while NOMINAL shows the rover is following its planned set of traversal and task execution actions. DISABLE is set when the rover is unable to continue its operations, BUDDY/RESCUER the rover is supporting an astronaut, in which case the astronaut directly communicates with rover without planner involvement. Rovers have a limited amount of onboard energy, a value diminished by each traversal and task execution.

Astronauts are capable of accomplishing many of the same tasks the rovers can complete, thus they are included as “mobile agent” resources within the planner. However, they also may elect not to execute the planned activities assigned to them. One of the reasons to have humans in the field is to capitalize on their superior sensing and reasoning abilities. While astronauts are trained to accept instructions, they also will not prefer to be strictly “pawns” of a computerized planning agent. In practice, the astronaut might decide to select a different task based on his preference or observations. In this work, we assume the astronaut will inform the planner of such deviations. Although a difficult task requiring uncertainty models, we are currently working to develop a behavior-based protocol for astronaut activities that is sufficient to anticipate situations in which the plan is not being followed. It is important to detect such behavior because the computed plan relies on the astronaut to complete assigned actions.

Table lists the attributes of the astronaut state variable. As with the rover, the status variable indicates the current activity execution state, including a BUDDY/RESCUER status set when an astronaut is assisting another “lead” astronaut with a task or during an emergency. The astronaut state is augmented with two unique flags: need-help and supported. The first (need-help) is set when assistance is requested, prompting the planner to redirect a rover (or other astronaut) as an assistant. The second (supported) is set once the assistance arrives, after which need-help is reset to prevent further assignment of assisting agents. In an emergency, for example, any rover or astronaut that has life support equipment can be tasked as an assistant. For non-emergency task execution, a rover can be requested to provide a specific tool through the same protocol.

### Table 3: Astronaut State Parameters.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>pointer to a location</td>
</tr>
<tr>
<td>set of types</td>
<td>example: { ROCK_SAMPLING &amp; IMAGE_CAPTURE }</td>
</tr>
<tr>
<td>status</td>
<td>{ IDLE or NOMINAL or DISABLE or BUDDY/RESCUER }</td>
</tr>
<tr>
<td>need help</td>
<td>[true / false]</td>
</tr>
<tr>
<td>supported</td>
<td>[true / false]</td>
</tr>
<tr>
<td>energy</td>
<td>∈ℜ, current level of energy</td>
</tr>
</tbody>
</table>

Rovers and astronauts can perform three different actions that can change the status of the world:

- **Navigate between 2 locations.**
  
  Goto(rover3, loc1, loc2)

- **Perform a task**
  
  Execute(astronaut2, task3)

- **Rescue an astronaut.**
  
  Rescue(rover5, astronaut2)

We are currently working on a cooperate action that would enable two or more agents to jointly perform a task. For example, two astronauts may be required to repair a solar panel due to its size or complexity. To enable such actions, a consistent list of preconditions must be true in the current state. Those preconditions include obvious statements such as co-located collaborators as well as less obvious conditions involving security, etc.

### Planning Algorithm

The planning module implemented for this work is based on a modified forward search algorithm. Many techniques have been developed to build plans (Ghallab, Nau and Traverso 2004). Techniques such as plan space planning are not applicable because the output plan should be optimal (when possible) and the computation of cost requires state knowledge. Forward search techniques have recently gained interest as new heuristics have been created. Planners such as STL-planner (Bacchus and Kabanza 2000) have been proven to give very good results and effectively influence decisions.

For this work, we adapted a uniform cost search algorithm to the planning task. Task execution costs are
currently constants that are independent of the specific agent executing the task, thus cost is based strictly on the time required to complete traversal actions. Node expansion is performed using a closest point policy. In other words, when traversing between locations, close locations are preferred to distant task locations. This problem can be related to the Multi Traveling Salesman Problem but has significantly more constraints. In our situation the salesmen are not allowed to go everywhere and cities are ordered, which makes traditional heuristics impossible to apply. Nevertheless, our implementation contains methods and techniques that enable us to control search speed. First, control rules are applied when adding new actions to the plan. Those rules prune irrelevant new actions which can significantly reduce branching factor over an exhaustive search.

An additional cost-based branch control strategy has also been. As stated previously, managing computational time is a critical issue for safety versus plan quality. Controlling the width of the search graph also enables control of optimality versus planning time. The selection among candidate actions for each node is made with the distance-based traversal cost function. For instance, when the branching factor is two, only the two best (closest) applicable tasks for that agent will be selected. This timeliness feature compromises optimality; however, since fewer partial plans are explored during the search. This trade off is inevitable and considered acceptable for our application given the obvious need to rapidly replan, particularly when an astronaut requests emergency assistance.

This simple forward search process cannot plan parallel actions, therefore to compute a schedule for the rovers, each instantiated action includes two parameters that define its start and end times. Each time an action is added to a partial plan, its start time is found by examining action constraints and the placement of preceding actions in the plan. Because tasks are potentially ordered, the rovers and even the astronauts may have to wait to perform an action. Hard precedence constraints are most likely when construction work is involved, since interconnected components often require appropriate assembly ordering.

Ideally, output plans will satisfy all specified goals. However, achieving the full set of tasks may be impossible for two reasons: (1) no agent has sufficient energy to move or to perform a task, or (2) no agent has the capability (toolset) to accomplish the remaining tasks. This scenario can either occur when the full set of mobile agents is missing capabilities or when an agent with a unique capability begins rescuing/assisting an astronaut prior to completion of their task plan. In order to deal with this issue, partial plans are stored each time backtracking is initiated. When the goal can’t be reached and all the nodes have been expanded, the best partial plan is selected. The plan returned is then the one that can achieve the maximum number of tasks with the lowest [traversal] cost.

Managing Unexpected Events

Once a plan has been computed, the task schedule is passed to the execution module. Actions are sent to the human and rover agents as they become applicable. Note that, in our “simple” centralized system, the cooperation between agents is strictly managed by the execution module, a “feature” we hope to extend in future work. For example, when two tasks are ordered and performed by two distinct agents, the command to start the second task will only be sent after confirmation has been received that the first task was completed.

Rovers and astronauts send real-time updates of their location, traversal speed, and task completion status. Astronauts additionally are expected to communicate any requests for assistance or deviations from tasks they were assigned by the planner.

We are working toward more automated detection of unanticipated events, particularly to relieve the astronaut of the burden to explicitly communicate every decision they make with potential impact to the planner. Regardless of the mechanism, when an anomalous event is detected, an alert plus stop command is sent to the agents and a new plan is computed based on the information. In such situations, replanning time may be significantly constrained. First, it is somewhat absurd to expect astronauts to remain idle while a lengthy replanning process is completed. Second, if an astronaut is injured, a response is required as quickly as possible. For replanning, therefore, the branching factor is significantly reduced (in the limit to 1) to trade optimality for efficiency.

Simulation Results

To illustrate planner performance, a series of randomly-generated tasks were generated for various team sizes and compositions. Figure 3 shows a simple scenario with two rovers and two astronauts. The initial plan is computed with a branching factor of two and succeeds in accomplishing all the tasks. At time t=21.9s astronaut A0 requires help and a new plan is computed. Rover R0 is the chosen to rescue him and all remaining R0 actions are split among astronaut A1 and rover R1. Even if the planner succeeds in finding a solution that achieves the full task set, note that the agents’ energy level at the end of the second plan is much smaller than it would have been with the initial plan. This reflects the branching factor tradeoff. The planner set branching factor to one during the replanning phase, enabling it to complete its new plan within milliseconds. For problem of a larger size, the plan computation can be very expensive, especially when the branching factor increases. For more than 27 locations and tasks, the computational time is too large to be acceptable for replanning when the branching factor is greater than one (given a maximum waiting time constraint of 30 sec).

Figure 4 illustrates this issue. The three graphs show the time needed to find a plan with three different branching
More time is required to find a plan when branching factor is increased but the solution is more likely to be optimal. Note that the graph with branching factor one does not have an exponential shape. By selecting only the best action during the search and not allowing backtracking, search time is approximately linear with respect to the total number of actions. Indeed, the total number of actions is a function of problem size (number of locations and tasks). Using a branching factor of one can thus be useful to plan for very large problems.

Low branching factor plans ensure reactive response but are not optimal. We are working to augment the planner with heuristics that dynamically modify the branching factor during the planning process depending on the available set of possible actions.

Figure 5 shows the nominal plan computed for the team of two rovers and two astronauts performing 14 tasks. In this simple example, each astronaut or rover is capable of achieving all tasks, and no ordering constraints are imposed. The initial plan is computed with a
branching factor of two and succeeds in accomplishing all tasks. At time t=21.9s, astronaut0 requires help and a new plan is computed. Rover0 diverts to assist, as shown in Figure 6, and the remaining tasks are split among astronaut1 and rover1. In Figure 6, the level of gray with indicates astronaut/rover energy level as the plan is executed; darkening indicates a reduction in energy. As illustrated in Figure 4, the initial plan required appreciably less energy than the plan that accommodates astronaut assistance, but the replanning effort (with default branching factor of one) only required a few milliseconds of computation time.

Conclusion and Future Work

This paper has introduced a planning and execution framework applicable to astronaut-rover teams operating in the same physical environment. Emphasis is placed on identification and management of the unique properties of the astronauts, enabling the planner to react when either the astronaut elects not to complete an assigned task or requires assistance. Although our work is in a relatively early stage, we believe a planner will be most successful that integrally plans tasks for both rovers and astronauts. Although complexity is always an issue for centralized planning systems, we have worked to mitigate this complexity through control of the search-space size, enabling the planner to robustly and efficiently respond to off-nominal events.

We are currently implementing and testing this system with “real” rovers and humans. Such tests are crucial to understand both low-level automation and human factors aspects of cooperation. We also plan hybrid tests in which simulated and embodied physical human/robotic agents “cooperatively” execute missions, enabling test scenarios to scale beyond limits imposed by hardware cost and development time constraints. The simulation-based scenarios completed to-date indicate the planning algorithms and centralized architecture are applicable provided global communication links are available and reliable. We will continue to build and work to understand the unique properties of astronauts as collaborative agents, both through task performance metrics and evaluations by human test subjects.

References


