GROUND-BASED AUTOMATED SCHEDULING FOR THE MARS 2020 ROVER

Virtual Conference 19–23 October 2020

A. Yelamanchili[1], J. Agrawal[1], S. Chien[1], J. Biehl[1], A. Connell[1], U. Guduri[1], J. Hazelrig[1], I. Ip[1], K. Maxwell[1], K. Steadman[1], S. Towey[1]

[1]Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Dr Pasadena, CA 91109, United States, E-mail: firstname.lastname@jpl.nasa.gov

ABSTRACT

The Mars 2020 Rover Mission will be using an automated ground-based scheduling system called Copilot to schedule the rover's activities at landing. Using automated scheduling technology will allow for plans to be generated more quickly. Because automated scheduling tools have not been widely used for prior rover missions, developing users' trust in the system is crucial. An explainable scheduling tool called Crosscheck has been developed to visualize the creation of a schedule, and to explain why activities failed to schedule given their constraints. This will allow science planners to change activity constraints to allow failed activities to successfully schedule, achieving their science goals.

1 INTRODUCTION

The Mars 2020 Perseverance Rover is scheduled to land on Mars on February 18, 2021 [1]. Once on Mars, the rover's activities will be scheduled by a ground-based scheduling system [2] known as Copilot. A scheduler onboard the rover [3] has been developed for later in the mission. Both the ground and onboard schedulers use the same core scheduling algorithm [2,3,4,5].

Automated scheduling systems have not been widely used in prior rover missions, with the exception of a system developed for the Mars Exploration Rover Mission [6]. In most operations scenarios, science planners manually generate a schedule consisting of activities, and ensure the schedule satisfies all given constraints. This manual process results in a significant amount of time devoted to creating a schedule. Automating the generation of a schedule of activities based on their input constraints aims to speed up this process.

It is also imperative that users have trust in the automated system. To build this trust, as well as to provide feedback on how to change constraints to achieve a better schedule, an explainable scheduling tool called Crosscheck was developed. Crosscheck provides a visual representation of the schedule being constructed and what constraints were considered. For activities that failed to schedule, additional analysis is done to determine why they failed to schedule. This information is also provided to users in the interface. This information allows them to change constraints on activities that would lead to them being successfully scheduled.

The remainder of this paper describes the scheduling problem, as well as how the scheduler auto-generates certain required activities based on other activities in the schedule. It also describes the algorithms Crosscheck uses to determine why activities failed to schedule. Finally, it describes how science planners will generate a schedule, how they use Crosscheck to understand the schedule, and how to resolve any activities that failed to schedule.

2 SCHEDULING PROBLEM

2.1 Activity Scheduling and Constraints

The scheduler is a non-backtracking scheduler that schedules in priority first order and never removes or moves an activity after it is placed during a single run of the scheduler. The priorities are determined using a squeaky wheel approach [2]. Activities are given constraints such as duration, resources claimed, dependencies on other activities, state requirements, and start time windows. Activities can also change states at their start and end times, and use resources that are globally constrained, such as data volume, energy, and peak power. Additional details about the core scheduling algorithm can be found in [3].

2.2 Scheduling Wake/Sleep Activities

The M2020 rover's power source is a Multi-Mission Radioisotope Thermoelectric Generator (MMRTG) [7]. While the MMRTG constantly generates energy, the CPU's awake and "idle" state (i.e. no other activities) consumes more energy than the MMRTG provides. Therefore, the rover's energy, or battery state of charge (SOC) only increases when it is asleep. The rover, however, must be awake to execute certain activities.

The scheduler is responsible for creating and scheduling wakeup and shutdown activities as necessary when scheduling activities. Fig. 1 shows an example of activities with their required wakeup and shutdown activities, as well as the corresponding awake/asleep periods.

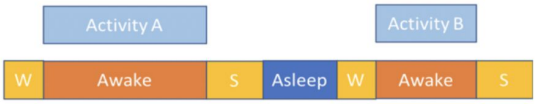

Figure 1. Activities with wakeups and shutdown scheduled for them, as well as the awake/asleep periods

There are additional constraints on awake and asleep periods to prevent the rover from waking up and shutting down too frequently - a minimum sleep time constraint, and a minimum awake time constraint. These constraints are also considered when determining where to place the wakeup and shutdown activities.

If an activity requires the rover to be awake but there are no suitable times for wake/sleep activities to be scheduled for it, the activity will fail to schedule. Additional details on the wake/sleep scheduling algorithm can be found in [4].

2.3 Scheduling Heating Activities

In addition to some activities requiring the rover to be awake, certain activities require specific areas of the rover to be sufficiently heated before the activity can commence. They also require the areas to maintain a proper temperature throughout the duration of the activity.

The scheduler is responsible for creating and scheduling preheat and maintenance heating activities as activities necessitate. Fig. 2 shows an example of an activity with its required preheat and maintenance activities.

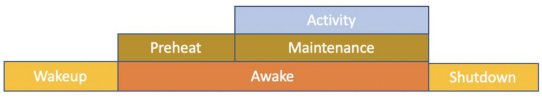

Figure 2. Activity with its required heating activities

If an activity's required preheat and maintenance activities are unable to be scheduled, the activity will fail to schedule.

3 ACTIVITY FAILURE ANALYSIS

Crosscheck is an explainable scheduling tool that has been developed to give users information on how the schedule made by Copilot was constructed, as well as to give information about why activities failed to be included in the schedule. This information will allow science planners to understand how they can modify the input activities and their constraints to obtain their desired schedule that does not violate any constraints.

There are two main factors that contribute to where an activity is scheduled and whether it is successfully added to the schedule at all. These factors are the scheduling step the activity is scheduled at, determined by its priority, and its constraints. For activities that fail to schedule, additional analysis is done to determine two pieces of information that can aid in the resolution of the failure: the earliest scheduling step at which the activity would have failed to schedule, called its first failure step, and the constraints that were violated at this scheduling step.

3.1 Determining First Failure Step

Identifying the first failure step gives insight into what activities were scheduled prior to the failed activity that led to it being unable to schedule. Changing constraints on activities scheduled after the first failure step would not allow the activity to schedule, however, changing constraints on the activities scheduled prior to the first failure step may. The activity scheduled directly prior to the first failure step is considered to be in direct conflict with the failed activity. For the activity to possibly schedule, science planners must do one of the three things:

- Modify how the conflicting activity is scheduled by directly changing its constraints
- Modify how the conflicting activity is scheduled by changing constraints on previously scheduled activities
- Modify how the failed activity is scheduled relative to the conflicting activity is scheduled by changing its constraints

It is also possible an activity may fail on its own with no other activities in the plan, in which case there are no directly conflicting activities. In the case of a state requirement that cannot be met by anything in the current plan, science planners may need to add an activity to the plan that would allow the rover to be in the required state.

3.2 Determining Unsatisfiable Constraints

Once the earliest failure step of the activity is determined, we determine the constraints that were unsatisfiable at that step, leading to the activity not being able to be scheduled. The scheduling algorithm consists of two primary phases, and activities can fail to schedule in either of those phases for a variety of reasons.

In the first phase, valid intervals for a subset of the activity's constraints are computed. A valid interval for a constraint is a continuous time interval in which that constraint is satisfiable for the duration of the activity. The valid intervals for each constraint are intersected together to create final valid intervals that are passed to the second phase of scheduling.

In the second phase of scheduling, wake/sleep activities, heating activities, and plan wide constraints are considered to determine the activity's final start time.

Crosscheck determines which of the phases the activity fails to schedule during, and which constraints it would violate that are considered during that phase.

3.1 Failures due to Valid Intervals

First, Crosscheck checks if there were any final valid intervals after each constraint's valid intervals were intersected together. If the final valid intervals are empty, there was some combination of constraints that were incompatible with each other. Crosscheck seeks to find the minimal set of constraints that are incompatible with each other.

For each constraint, we first check if any have no valid intervals on their own. If any do, these constraints are output as the cause of the failure. If each constraint has at least one valid interval, the valid intervals from each pair of constraints are intersected together. If there are any pairs of constraints that have no valid intervals after intersection, these sets of constraints are output as the cause of the failure. If each pairwise set of constraints has valid intervals, this process continues with each set of three constraints, and so on, until we find some combination of constraints that do not have valid intersections between their valid intervals.

In order to allow the failed activity to schedule, the valid intervals for the constraints identified in the above step need to be changed by the user somehow. This can be done by either directly changing constraints on the failed activity itself, or changing constraints on previously scheduled activities, that would allow the valid intervals for the failed activity's constraints to change.

3.2 Failures due to Plan-wide Constraints and Sleep/Heat Scheduling

Activities may fail to schedule during the second phase of scheduling for one of the following reasons:

- An activity's required preheat would be outside of the plan horizon
- The peak power used by an activity and its required sleep/heat activities would violate the maximum peak power constraint
- The energy used by an activity and its required sleep/heat activities would violate the minimum state of charge constraint
- Scheduling an activity's required sleep activities would violate the minimum asleep constraint or the minimum awake constraint
- An activity's required heating activities would be scheduled outside of the operability window for the required heaters

Multiple time points may be considered for scheduling the activity's start time during this phase. Each time point may fail for one of the above reasons, so there may be multiple different reasons for the failure. Crosscheck will indicate each unique one.

4 CROSSCHECK VISUALIZATION

Crosscheck enables users to visualize the state of the schedule at each scheduling step, as well as the information on why each failed activity failed to schedule. The knowledge of why activities failed to schedule will let users understand how they may modify constraints to enable those activities to successfully schedule.

Five timelines are visible at each scheduling step:

- Output plan - the activities that have been successfully scheduled at or prior to this scheduling step
- Yet to be scheduled - the activities that have a lower scheduling priority and will be scheduling in later scheduling steps
- Failed to schedule - the activities that were unsuccessfully scheduled at or prior to this scheduling step
- Energy profile - the energy usage over time for activities in the output plan at this scheduling step

- Peak power profile - the peak power used at each time for activities in the output plan at this scheduling step

Fig. 3 shows an example of the last scheduling step of a plan visualized in Crosscheck. The final grounded schedule is visible in the top timeline, along with all of the activities that failed to schedule in the third timeline.

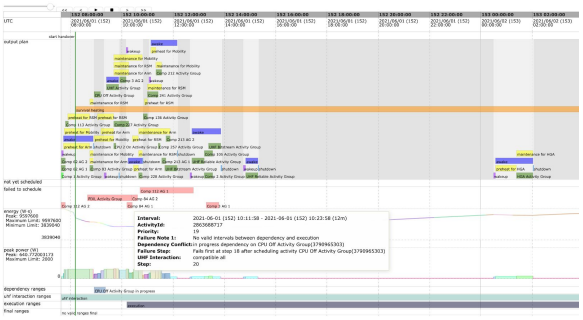

Figure 3. Crosscheck visualization of a plan

4.1 Activity Information

For each activity, users are able to change the view to see the state of the plan at the scheduling step in which that activity was attempted to be scheduled. For activities in the output plan or failed activities timeline, users can see the valid intervals calculated for each of that activity's constraints, as well as the final intersected valid interval. For activities in the failed to schedule timeline, users can see the two pieces of information identified during the failure analysis - the first scheduling step at which the activity would have failed to schedule, and the constraints that were unsatisfiable.

4.2 Resource Usage

For the energy and peak power timelines, additional information is available on what activities contributed to the resource usage and how much they contributed. At any time in either of these timelines, users can inspect the resource users.

For the energy timeline, this will show a list of the energy used by each activity from the start of the plan up to that time, in decreasing order of how much energy they used. This allows users to see which activities are causing the most strain on the resource. If an activity fails to schedule due to lack of energy available, they may want to change constraints on an activity that is causing the strain on the energy to allow it to schedule at a different time in the plan. Fig. 4 shows an example of this information.

Energy Users before 2021-06-01 (152) 08:50:48

OBA Name	Energy Use (W·s)	OBA Start Time
preheat for Mobility	836222.793762207	2021-06-01 (152) 07:44:52
preheat for Arm	582892.9940414429	2021-06-01 (152) 07:43:20
preheat for RSM	68705	2021-06-01 (152) 07:53:08
survival heating	41852.19652843475	2021-06-01 (152) 08:11:10
awake	11221.056000000022	2021-06-01 (152) 07:43:20
Comp 62 AG 2	4200	2021-06-01 (152) 07:37:50
Comp 62 AG 1	1800	2021-06-01 (152) 07:37:50
Comp 112 AG 2	1200	2021-06-01 (152) 07:43:20
wakeup	1008.5999999999984	2021-06-01 (152) 07:38:20
Comp 1 Activity Group	600	2021-06-01 (152) 07:37:50

Figure 4. Information about energy usage prior to a certain time

Similar information is available for the peak power timeline. Inspecting a time point on this timeline will show a list of each activity drawing power at that time, and what their peak power draw is, in decreasing order of power usage. Fig. 5 shows an example of this information.

Peak Power Users at 2021-06-01 (152) 09:50:05

OBA Name	Peak Power Use (W)	OBA Start Time
preheat for Mobility	253.0999984741211	2021-06-01 (152) 09:16:17
preheat for Arm	200.89999771118164	2021-06-01 (152) 09:33:05
awake	112.772	2021-06-01 (152) 09:16:17
preheat for RSM	35	2021-06-01 (152) 09:23:24
UHF Activity Group	10	2021-06-01 (152) 09:23:28
CPU 2 On Activity Group	10	2021-06-01 (152) 09:41:09

Figure 5. Information about peak power draw at a certain time

5 CREATING A SCHEDULE AND USING CROSSCHECK

Science planners use the Component-based Campaign Planning, Implementation and Tactical tool (COCPIT), shown in Fig. 6, to add activities and constraints to a plan.

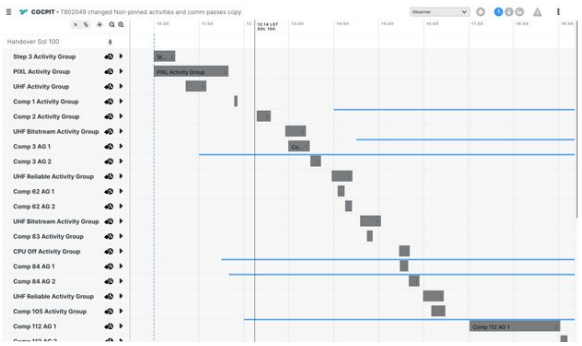

Figure 6. COCPIT is the interface used to create a schedule

Activities are defined in an activity dictionary. Some activity constraints are defined in the dictionary, such as resources used, and state requirements and effects. In COCPIT, users can add execution time constraints and dependency constraints to activities.

Once the user has determined all of the activities and constraints for the plan, they call Copilot to produce a grounded schedule. COCPIT then displays all of the activity start times, heating activities, wake/sleep activities. From a grounded schedule, lower-level command sequences are generated and uplinked to the rover on a daily basis.

COCPIT will also indicate which activities failed to schedule. Users are given a link to the Crosscheck visualization of the schedule to understand how they may change constraints to allow failed activities to schedule. Examples of this are given in the following subsections.

5.1 Failure due to Valid Intervals

Fig. 7 shows an example of an activity failing to schedule due to there being no intersections between the valid intervals of the activity's execution time and UHF interaction constraints. The UHF interaction constraint dictates what sorts of communication passes an activity can be in parallel with.

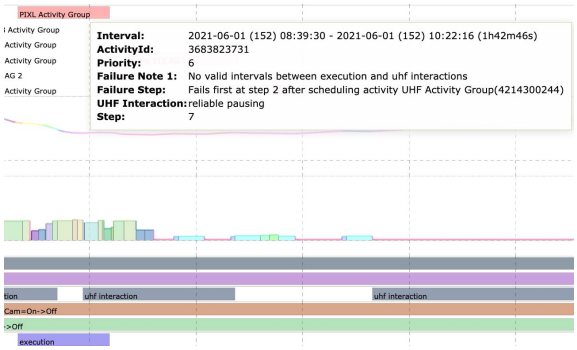

Figure 7. An activity failed to schedule due to its incompatible execution time constraint and UHF interaction constraint

The first failure step is identified as two. The activity that was scheduled prior to the first failure step is identified as a UHF activity. Thus the user must either change where this UHF activity is scheduled, change the UHF interaction constraint on the failed activity, or change the execution constraint on the failed activity. The UHF activity time and UHF interaction constraint are not feasible for science planners to change. This would lead them to change the execution time constraint on the activity.

Fig. 8 shows another example of an activity failing to schedule due to the intersections of its constraints' valid intervals.

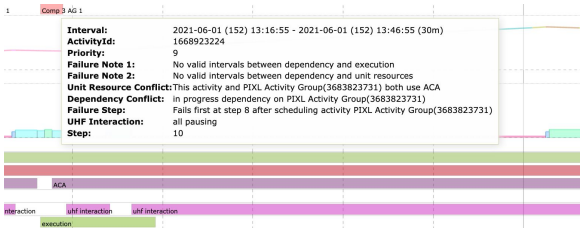

Figure 8. An activity failed to schedule due to incompatibilities between its execution and dependency constraints, as well as its dependency and unit resource constraints

The execution valid intervals do not intersect with the dependency valid intervals and the dependency valid intervals do not intersect with the unit resource valid intervals. The first failure step shows us the conflicting activity. Users are also given the information about unit resources the failed activity and the conflicting activity have in common, and any dependency constraints the activity has.

In this example, the dependency requires the failed activity and the conflicting activity to be in parallel, but they both require the same resources. A science planner would likely change or remove the dependency constraint on the failed activity. Likewise, the science planner would likely change the failed activity's execution constraint, by widening it to give the activity more possible times to schedule at, or moving it to a more appropriate point in time.

5.2 Failure during sleep/heat scheduling

Fig. 9 shows an example of an activity failing to schedule due to its final valid intervals being outside of the operability window of the activity's required heaters. Science planners would remedy this to change the activity's constraints, likely its execution constraints, to be within the operability window of the activity's heaters.

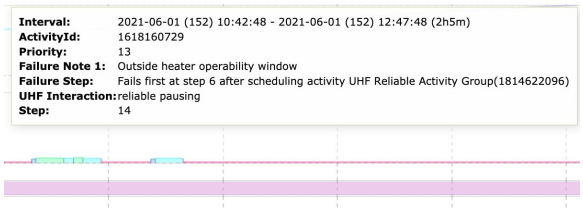

Figure 9. An activity failed to schedule due to its final valid intervals being outside of the heater operability window

Fig. 10 shows an example of an activity failing to schedule due to peak power draw being too high during its final valid intervals. Users can inspect the peak power timeline at the time they would expect the activity to schedule at to determine what other activities are drawing the most power at that time. They may choose to change constraints on the failed activity to allow it to schedule at a time when peak power draw is lower, or they may choose to change constraints on previously scheduled activities occurring at the same time, to allow for more power availability at this time.

Figure 10. An Activity failed to schedule due to the peak power draw being too high

6 CONCLUSION

We have described the ground-based automated scheduling system, Copilot, that will be used for the Mars 2020 Rover, as well as the explainable scheduling tool, Crosscheck, developed for use alongside Copilot. Copilot will allow for science planners to spend less time creating a schedule of their desired activities that do not violate any constraints. Crosscheck will aid in this by allowing the science planners to understand how the schedule was created, why activities failed to schedule given their constraints, and how they may alter activities and their constraints to achieve the desired schedule.

7 RELATED WORK

Bresina et al. [6] created an automated scheduling system for the Mars Exploration Rover Mission called MAPGEN. MAPGEN was a constraint-posting planner, and it explicitly indicated temporal flexibility, allowing users to drag activities until they reached their earliest or latest allowed start times. Unlike Crosscheck, it does not explicitly indicate why the planner could not achieve the desired result.

Ramaswamy et al. [8] created a tool to visualize various execution runs of a given input plan to the Mars 2020 scheduler. This tool indicates how often an activity failed to execute over all execution runs, as well as how often activities executed in parallel or switched order temporally. Crosscheck focuses on visualizing a single schedule, and gives information about why an activity failed to schedule.

The Rosetta Orbiter mission [9] used automated scheduling to schedule science activities. A static visualization of each scheduling iteration was available to users, as well as the valid intervals for each activity. The constraints that caused the failure as well as the earliest scheduling step an activity would have failed at is not given as it is in Crosscheck, rather, users have to infer the conflicting constraints by inspecting the valid intervals visually.

Acknowledgement

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. © 2020 California Institute of Technology. Government sponsorship acknowledged.

References

[1] Jet Propulsion Laboratory (2020). Mars 2020 Perseverance Rover. Online at https://mars.nasa.gov/mars2020/ (as of September 3, 2020).

[2] Chi, W., Agrawal, J., Chien, S., Fosse, E. & Guduri, U. (2019). Optimizing parameters for uncertain execution and rescheduling robustness. *International Conference on Automated Planning and Scheduling (ICAPS 2019)*.

[3] Rabideau G. & Benowitz E. (2017). Prototyping an Onboard Scheduler for the Mars 2020 Mission. *IWPSS*. 142-150.

[4] Chi, W., Chien, S. & Agrawal, J. (2020). Scheduling with complex consumptive resources for a planetary rover. *International Conference on Automated Planning and Scheduling (ICAPS 2020)*.

[5] Agrawal, J., Chi, W., Chien, S., Rabideau, G., Kuhn, S. & Gaines, D. (2019). Enabling limited resource-bounded disjunction in scheduling. *11th International Workshop on Planning and Scheduling for Space (IWPSS 2019)*. 7–15.

[6] Bresina, J. L., Jonsson, A. K.. Morris, P. H. & Rajan, K. (2005). Activity planning for the mars exploration rovers. *International Conference on Automated Planning and Scheduling (ICAPS 2005)*. 40–49.

[7] Jet Propulsion Laboratory (2020). Electrical power - nasa mars. Online at https://mars.nasa.gov/mars2020/spacecraft/rover/electrical-power/ (as of September 3, 2020).

[8] Alper Ramaswamy, B. B., Agrawal, J., Chi, W., Kim Castet, S. Y., Davidoff, S. & Chien, S. (2019). Supporting automation in spacecraft activity planning with simulation and visualization. *AIAA Scitech 2019 Forum*. 2348.

[9] Chien, S., Rabideau, G., Tran, D., Troesch, M., Doubleday, J., Nespoli, F., Ayucar, M. P., Sitje, M. C., Vallat, C., Geiger, B., Altobelli, N., Fernandez, M., Vallejo, F., Andres, R., & Kueppers, M. (2015). Activity-based scheduling of science campaigns for the rosetta orbiter. *International Joint Conference on Artificial Intelligence (IJCAI 2015)*.