Automated Scheduling of personnel to Staff Operations for the Mars Science Laboratory

Russell Knight, Andrew Mishkin, and Alicia Allbaugh

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109-8099

Leveraging previous work on scheduling personnel for space mission operations, we have adapted ASPEN (Activity Scheduling and Planning Environment) [1] to the domain of scheduling personnel for operations of the Mars Science Laboratory. Automated scheduling of personnel is not new. We compare our representations to a sampling of employee scheduling systems available with respect to desired features. We described the constraints required by MSL personnel schedulers and how each is handled by the scheduling algorithm.

I. Introduction

Unlike most deep-space robotic missions, MSL operators must interact with their spacecraft on a near-daily basis. Command sequences are sent to the Curiosity rover each Martian morning, after which the rover operates independently of the ground until the next morning. The rover transmits "decisional" telemetry each Martian afternoon, which is used by the operations team to seed the next tactical uplink planning cycle.

During each tactical planning cycle, the operations team must assess the health of the vehicle, update activity plans governing the rover's actions for the next Martian day, validate that those plans are within the vehicle's resources, generate and validate the command sequences that implement the activity plan, and uplink the command sequences to the rover.

In addition to the tactical planning cycle that produces the commands that are transmitted to the rover, there is a "supra-tactical" process that looks ahead 2-5 Martian days, managing the dependencies among rover activities and the complexity of plans being fed to the tactical process. Personnel are regularly rotated between the supra-tactical and tactical processes to ensure a tight linkage between the products of the supra-tactical process and the needs of the tactical process.

Factors impacting staffing of personnel performing MSL operations include:

- **Must command rover every available opportunity, unless precluded by safety concerns**. The effectiveness of the mission is largely ties to the number of command cycles successfully completed. Failure to generate a set of commands to uplink due to personnel staffing issues is unacceptable.
- **Long shifts**. Many roles have shifts 10-11 hours in duration, sometimes longer in cases when multiple Martian days are planned during a single workshift on Fridays.
- **Intensive workshifts**. Tactical and supra-tactical processes are workload intensive, with limited breaks and tight deadlines throughout the processes.
- **Need for team continuity from day to day**. Although all roles on the team document their rationale and products each shift, additional continuity from one planning day to the next is provided by staggering the days when personnel roll into and out of the operational schedule. This approach ensures that there will always be a source of context for personnel starting their personal work-week.
- **Fatigue management**. Command errors could potentially put the rover on Mars at risk. Among other means of preventing command errors, MSL manages team-member fatigue by limiting the number of consecutive days working tactical cycles.
- **Multiple operations roles within MSL teams and across MSL teams**. Some personnel on the mission have been certified for multiple roles. These personnel provide more depth to the team and flexibility in staffing roles, at the expense of increasing the complexity of the staffing problem.
- **Part-time personnel**. As the mission continues, the number of personnel who support both MSL and other JPL projects increases. MSL staffing plans must therefore accommodate additional personnel availability constraints where possible.

From this, we see that scheduling personnel for science and operations planning of the the Mars Science Laboratory (MSL) is challenging. The large number of roles to be filled, the unique nature of each day's planning requirements, and the large number of constraints make the scheduling problem quite daunting. There already exist several personnel scheduling systems that handle most of these types of constraints, but one of the most daunting requirements is that not all constraints can be satisfied, thus schedules need to represent a graceful retraction of constraints. We describe the MSL scheduling constraints and our scheduling algorithm. We compare our system to existing personnel scheduling systems. Finally, we describe planned improvements that applying our system to MSL personnel scheduling has brought to light.

II. MSL personnel scheduling constraints

The constraints that we must meet fall into a few basic categories: skills, "off-days", day of the week constraints, shift-weighting (by day), and duty-cycle constraints.

A. Skills or roles

Skills indicate what function or role the person is qualified to be scheduled for. A sample of roles for MSL include:

- OCPlan – On-call communications planner
- SPL – strategic planning lead
- TUL – tactical uplink lead
- SUL – strategic uplink lead
- SP – science planner
- RP – rover planner. Note that there are up to 3 RP roles, RPA, RPB, and RPC, that are filled depending on the day's requirement for staffing.
- SIE – sequence integration engineer
- ECAM – engineering camera payload uplink lead
- SSSAL – SSS (surface sampling system) activity lead
- TC – IPE (integrated planning and execution) team chief
- SSIE – Strategic SIE

Optional shadow roles are often filled for each of the SPL, TUL, SUL, SP, SSP, RP, SRP, SIE, ECAM, SSSAL and SSIE roles.

Thus, for any given day, any of these roles might need to be scheduled, although each day's requirement depends on what is planned for that day in general. The system needs to be flexible enough to allow for plans to be developed and modified from beginning to end, both in terms of the time covered by the plan and in terms of the process of developing the schedule.

An example schedule can be seen in Figure 1. (Names of individuals are blurred for privacy).

B. Off-days

Off-days are specific days that a person cannot be scheduled. The types of off-days include:

- LEAV – planned leave
- PART-LV – partial day leave
- M/P-LEAV – parental leave
- JURY – jury duty
- NO-MSL – non-MSL work conflict
- N/A – MSL work conflict (the individual is performing work on roles that are not currently scheduled)

Note that the granularity for off-days is finer than one might expect. This is due to the soft nature of our constraints and to the fact that people with the skills to operate this robotic explorer are rare… sometimes you don't get the vacation you planned. This fine granularity helps personnel schedulers determine the best (or least worst) option to choose when filling a role.

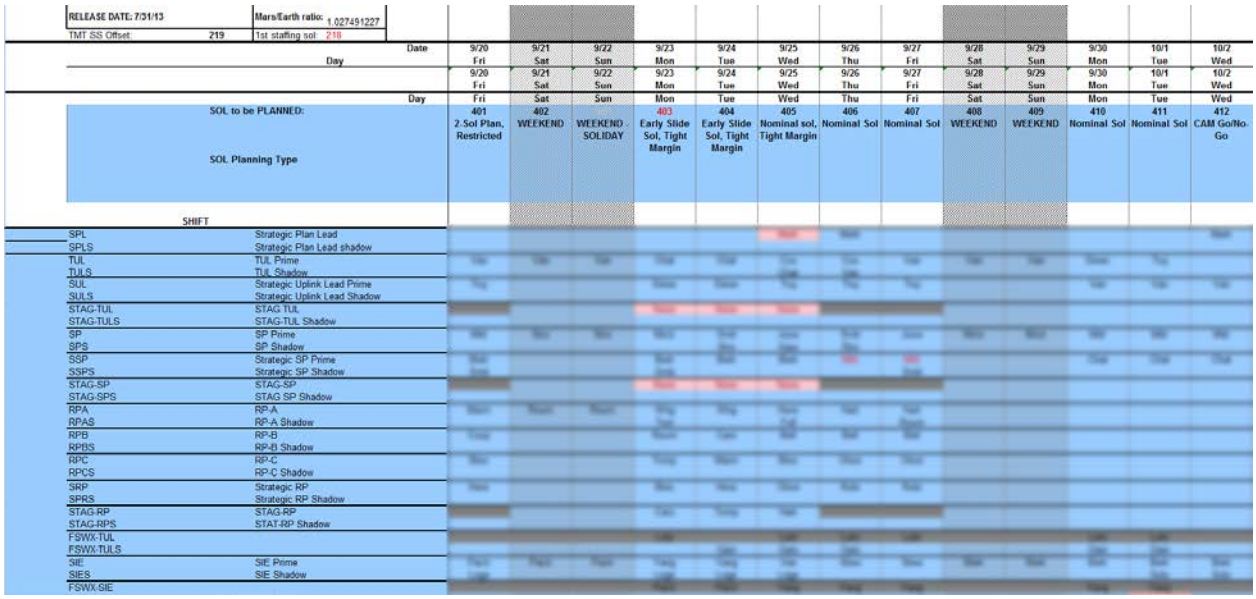

Figure 1 MSL personnel schedule

C. Day of week

Day of the week constraints document which days of the week in general a person can be scheduled, e.g. MWF for someone who only works on Mondays, Wednesdays, and Fridays. These are standard constraints found in all of the personnel planning systems we evaluated.

D. Shift-weighting

Shift-weighting is used to weight more demanding days more heavily when counting shifts. For MSL, Friday counts as two shifts because it is considered twice as hard as you need to plan for the entire weekend's worth of operations. In the future, some Thursdays will likely count as at least two shifts as they preceed a "Regular Day Off" Friday. Many staff members at JPL work a 9/80 schedule, meaning that every other Friday is a day off (often called the Regular Day Off). Should MSL allow their personnel to go to this schedule, then the plans produces on some Thursdays will need to cover threed days.

E. Duty cycle

Duty cycle constraints limit the total number of shifts someone can be scheduled over a given period. We think of this as a balancing constraint, where we are trying to avoid burning out personnel. There are three basic types of periods: shifts, days, and weeks. A *shifts* period is the total number of shifts, including weighting. Thus, for MSL, one week is 6 shifts. A *days* period is the total number of days. A *weeks* period is the total number of weeks. Weeks are only counted at Monday boundaries.

F. Graceful constraint descope

For all of these constraints, we desire the ability to descope to less restrictive constraints in a prescribed order should we fail to find a suitable schedule. Thus, for any individual, a collection of similar constraints are possible, each labeled with a descope level. Of course, hard constraints (without descope annotation) can be associated as well. For example, we might prefer John to work only 3 days per week with a descope level of 5 (e.g., "`(5)3:1w`"). But, John might be available 5 days per week, which we give a higher descope level to, e.g. 10 (e.g., "`(10)5:1w`). By building up our preferred options for each individual, we can characterize thresholds that we wish to maintain and the priority for each. This allows the automated scheduler to know our intent, and also allows us to evaluate various schedules with respect to each other.

III. Scheduling Algorithm

Scheduling starts with the lowest descope level and then attempts to find a schedule using squeaky-wheel optimization (SWO) [3]. Should SWO fail, then the descope level is increased, until either only hard constraints

remain or a schedule is discovered. In general, the schedule with the lowest descope level with the most assigned tasks is returned (with the most assigned tasks being the primary criterion).

The actual scheduling problem is the assignment of individuals to tasks. Each task represents a role to be filled on a certain day. Each individual consists of an identifier (a short name) and the collections of constraints for the individual.

A. Squeaky wheel optimization

For our application, each task (role/day pair) is given a scheduler priority. During each iteration of squeaky-wheel optimization, the task list is sorted. During each attempted scheduling of a task, if the scheduler fails to schedule a task (assign an individual to it) then the scheduler increases the scheduling priority of the task. Note that *scheduler priority* does not indicate how important a task is, merely how often other tasks conflict with it.

Each scheduling cycle, the scheduler takes each task in order and scans down the available individuals in order and assigns the task to the first individual possible that causes no constraint violations. Thus the fundamental underlying scheduler is a greedy scheduler. SWO allows a myopic greedy scheduler to take task interactions into account without explicitly modeling the interactions. This allows us to provide a rich set of constraints without having to provide an unwieldy and costly automated scheduler.

B. Constraint Descoping

The scheduler starts with all constraints enabled. Then, every iteration, the scheduler calls SWO to produce a schedule. Schedule quality is the number tasks that have been fulfilled. If all tasks have been fulfilled, then we stop and return the schedule. But, should we have remaining tasks, we descope to a new constraint level. This implies removing all constraints that have a descope level equal to or less than the current level, and the call SWO to check for feasibility. The number of scheduled tasks is our baseline descope level. Then, the scheduler introduces each level that had been descoped, one at a time. If any scheduling is successful (has at least the baseline number of successfully schedule tasks), then we adopt it as a new baseline schedule. In this way, we allow for a more graceful degradation of constraints than simply removing the constraints wholesale from the scheduler.

C. Architecture

The overall architecture of the system is simple. The schedule and constraints are part of an Excel spreadsheet. A macro saves the appropriate worksheet out as a comma separated value (CSV) file and invokes the scheduler. The scheduler performs the necessary scheduling and then saves its results out as another CSV file. The macro then loads the newly created CSV file and copies the appropriate values into the worksheet.

This simple architecture allows the system to run under Windows or Mac, which is a requirement for our environment.

IV. Personnel scheduler comparison

Automated scheduling of personnel is not new. A general comparison of the three schedulers we chose to compare can be found in [1]. We compare our scheduler to three of the top employee scheduling systems available: ShiftPlanning, Time Forge Scheduling [4], and NimbleSchedule [4]. In each, we see features that we need, but still all lack crucial features. Each of these scheduling systems handle many constraints and include an automated scheduler.

A. ShiftPlanning

Constraints handled by ShiftPlanning include:

- Skill/role
- Day of week
- Off-days (Vacation and holidays only)
- Duty cycle (hours per week only)

Constraints not handled by ShiftPlanning include

- Duty cycle (all other)
- Shift weighting
- Graceful descoping

B. Time Forge Scheduling

Constraints handled by Time Forge Scheduling include:

- Skill/role
- Day of week
- Off-days (can be labeled with any text)
- Duty cycle (hours and shifts per week)

Constraints not handled by Time Forge Scheduling include:

- Duty cycle (all other)
- Shift weighting
- Graceful descoping

C. NimbleSchedule

NimbleSchedule is a web-based scheduling tool and is by far the easiest system to set up. Constraints handled by NimbleSchedule include:

- Skill/role (role is equivalent to a department)
- Day of week
- Off-days (no labels)

Constraints not handled by Time Forge Scheduling include:

- Duty cycle
- Shift weighting
- Graceful descoping

V. Results and future directions

The speed of the scheduler is adequate given the size of schedules usually demanded of it, with complete schedules being delivered in seconds.

The quality of the schedules is good in the context of optimizing the existing set of constraints, but after use we learned of several new constraints that need to be integrated into it if we expect to maximize automated scheduling of personnel. These new features include:

- Minimim run length – a minimum number of consecutive days to schedule an individual, thus getting them up to speed and maintaining it without constantly switching to other individuals for the same role.
- Maximum “new start” count – limit the number of new individuals that are coming onto the team at any given time. This ensures that a certain level of human context is maintained between shifts.
- Encorporate off-day to role allowances – certain types of off-days can be used to fulfill certain roles. The scheduler should allow this.

Given the nature of our scheduler (a greedy scheduler wrapped in SWO and constraint descoping), we believe that each of these is fairly straightforward to attain. We look forward to reporting our results after future deliveries of the scheduler.

Acknowledgments

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[1] J. Carlsen. “TimeForge Scheduling.” March 2014. http://employee-scheduling-software-review.toptenreviews.com/timeforge-scheduling-review.html

[2] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran. "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling", International Conference on Space Operations (SpaceOps 2000). Toulouse, France. June 2000.

[3] D. Joslin and D. Clements. “Squeaky Wheel Optimization”, Journal of Artificial Intelligence Research 10 (1999) 353-373.

[4] Labor Management. “Timeforge: Management is NOW simple.” April 2014. www.timeforge.com/site/

[5] R. Sharma. “Make your Employee Schedules Minble – Review of NimbleSchedule.” January, 2012. http://www.getapp.com/blog/schedules-nimble-review-nimbleschedule-software/

[6] ShiftPlanning.com. “ShiftPlanning: the Workforce Management Champion.” April 2014. http://www.shiftplanning.com/compare/