

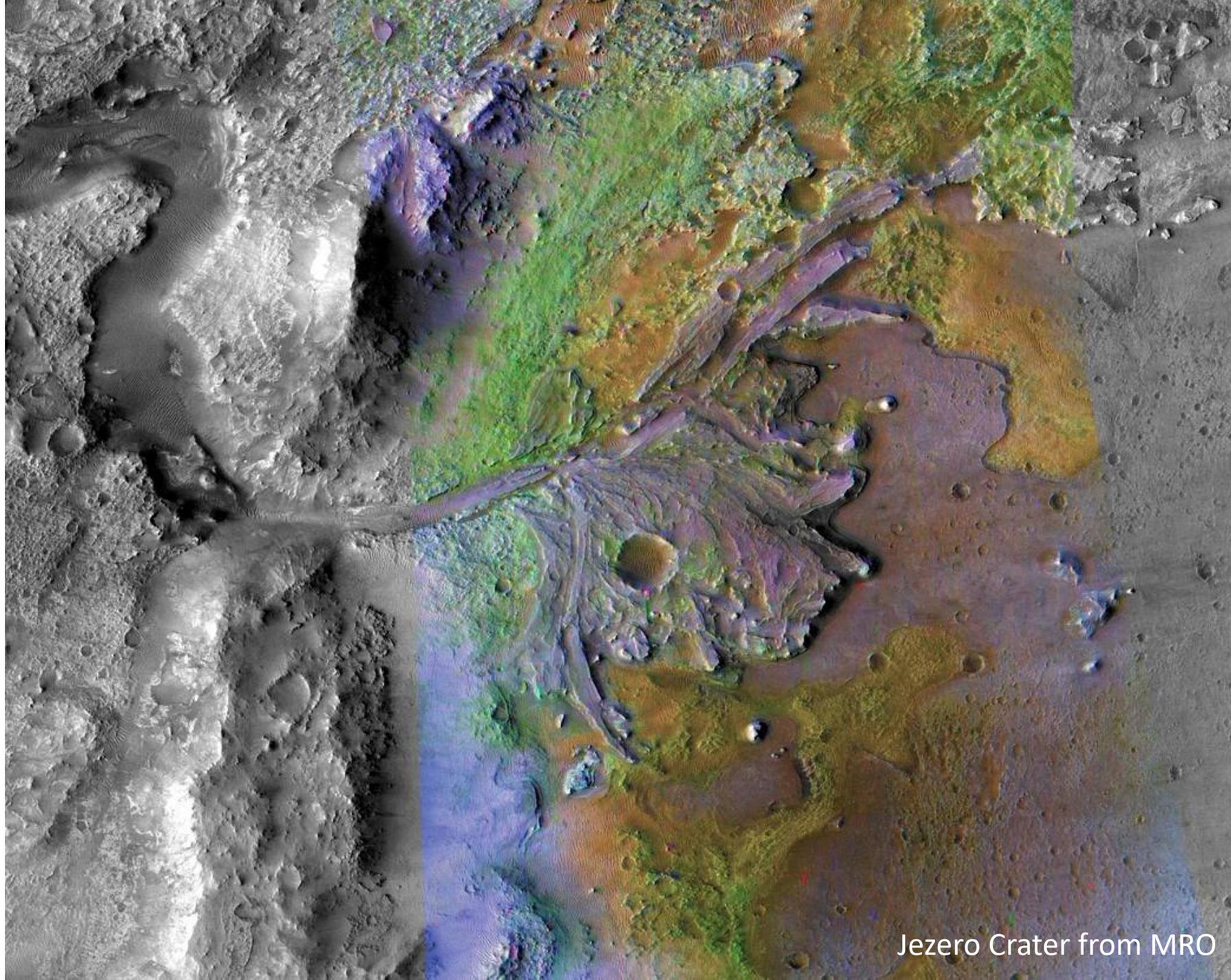


Jet Propulsion Laboratory
California Institute of Technology

Optimizing Parameters for Uncertain Execution and Rescheduling Robustness

Wayne Chi, Jagriti Agrawal, Steve Chien, Elyse Fosse, Usha Guduri

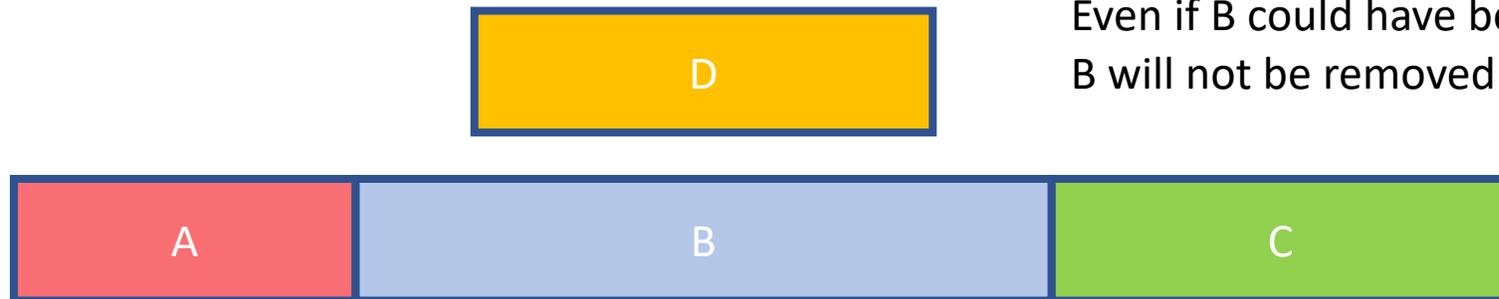
Artificial Intelligence Group
Jet Propulsion Laboratory
California Institute of Technology



Jezero Crater from MRO

Mars 2020 Onboard Scheduler

- M2020 Rover mission is developing an onboard scheduler to utilize unexpected additional resources (time, energy, data volume) from prior onboard execution.
- The Mars 2020 Onboard Scheduler is a (Rabideau and Benowitz 2017)
 - single-shot, non-backtracking scheduler that
 - schedules in *priority first order* and
 - never removes or moves an activity after it is placed during a single scheduler run.
 - Activities are not preempted
 - It does not search except for
 - Valid intervals calculations
 - sleep and preheat scheduling.



Activity D is pinned to this time. If Activity D has resource conflicts with B it will not be scheduled. Even if B could have been scheduled somewhere else, B will not be removed after it has been scheduled.

Challenges with Priority Setting

- The *order (activity priority)* in which activities are considered for scheduling will greatly affect how many activities can be scheduled and the efficiency of the of the plan.
- Finding a priority set for multiple conditions of finding a schedule is difficult. While taking into account execution uncertainty is even more difficult than that.

Different sets of activities could have already executed in the schedule

Activity D finishes early and C and B can be scheduled earlier.

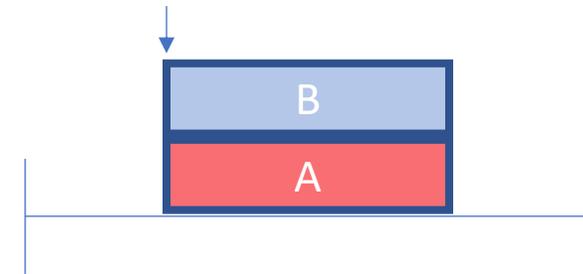
Activity D finishes late and C can no longer fit in the schedule.



Preferred Time

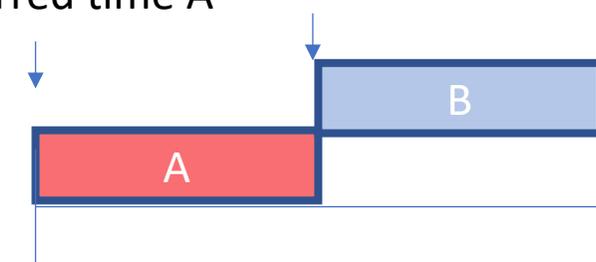
- Activities can be given a preferred time.
 - Scheduler will try to place the activity as close to the its preferred time as possible
 - Defaults to earliest valid time.
- Preferred time can drastically affect the effectiveness of the schedule.
 - Can affect whether or not activities will be able to be scheduled.
 - Can affect how long certain setup activities will take.
 - E.g. Preheats take longer earlier in the day when it is colder.
- Other parameters may affect the effectiveness of the schedule, but are not considered for this paper.

Preferred time



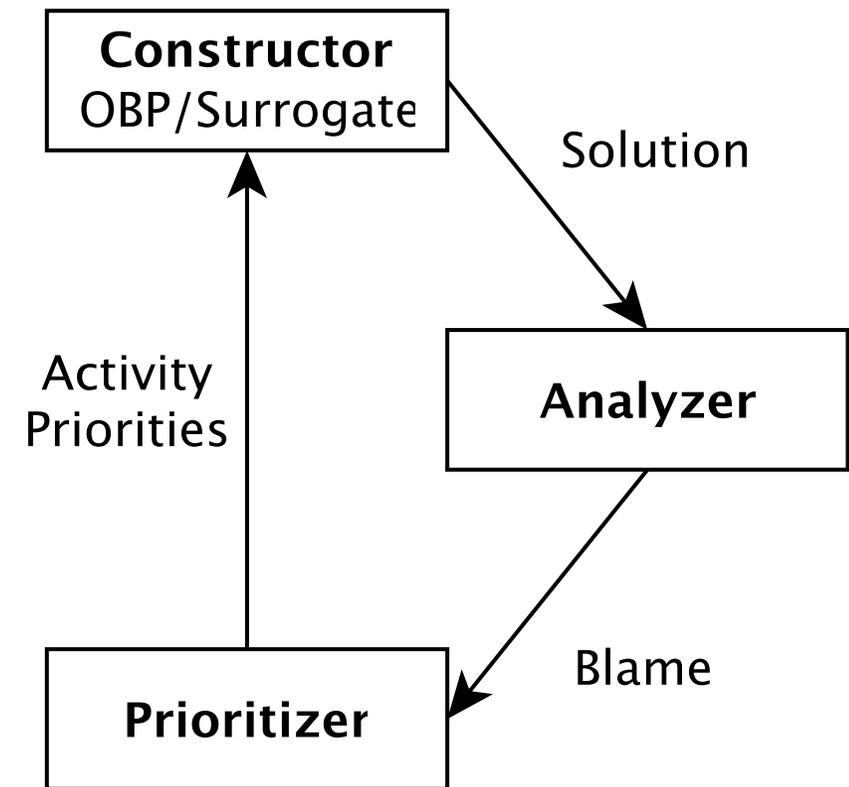
Preferred time A

Preferred time B



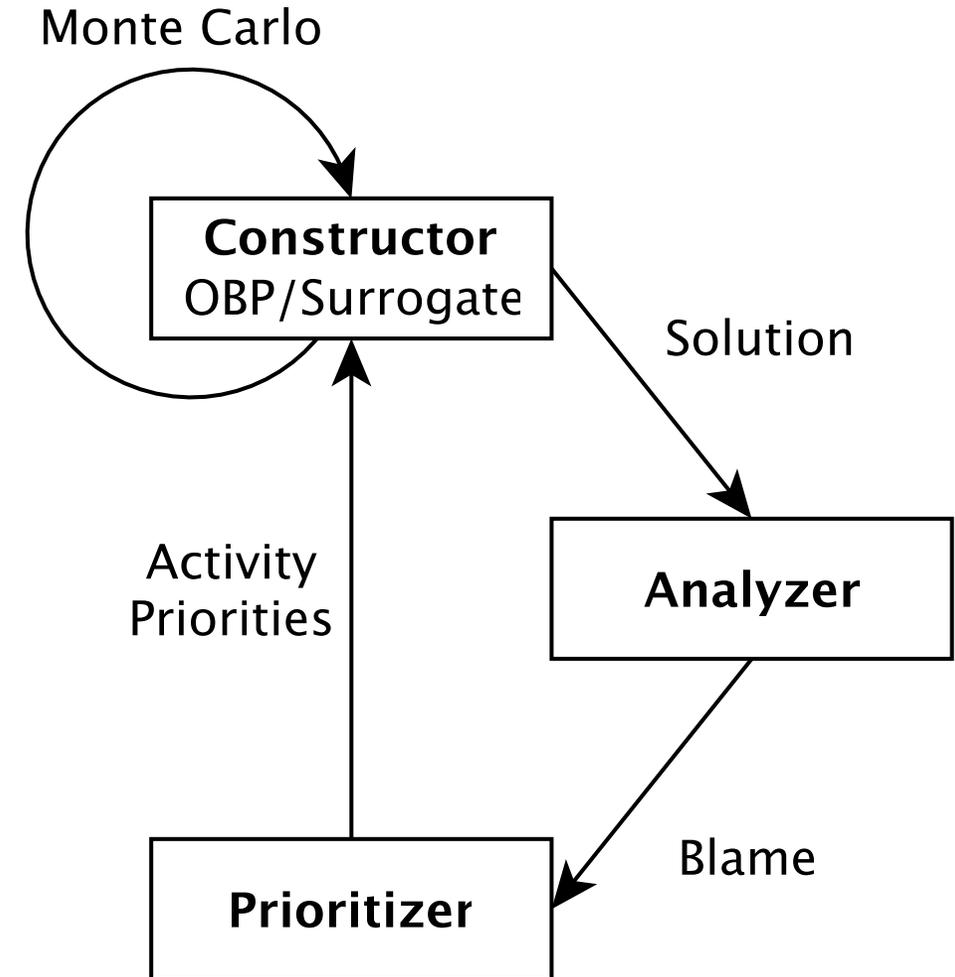
Squeaky Wheel

- The Squeaky Wheel Optimization Algorithm consists of three components – the **Constructor**, **Analyzer**, and **Prioritizer** – that repeat until an acceptable solution is found or time runs out [Joslin & Clements, 1998].
- The **Constructor** generates a schedule (lightweight, fast) as the **solution**.
 - In our case, the constructor is the Onboard Scheduler (Surrogate).
 - Its inputs are the requested activities, dependencies, resource and time constraints, and *activity priorities*.
- The **Analyzer** takes the solution, determines the problem areas, and assigns **blame** to those areas.
- The **Prioritizer** takes the blamed elements (activities) and assigns them new **parameters** so that the Constructor may generate a better solution.

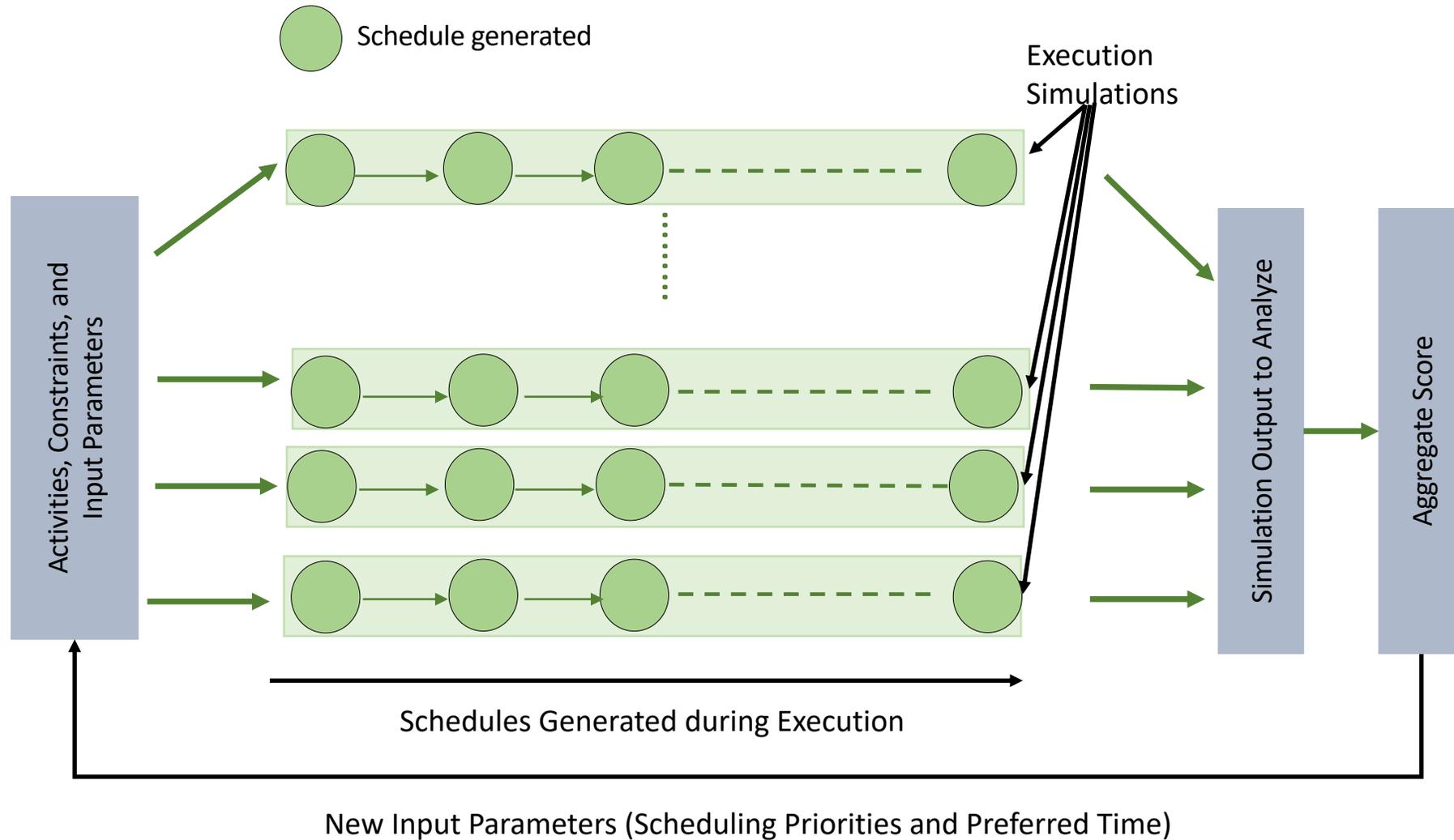


Parameter Search

- A single output schedule from the Constructor is insufficient when dealing with execution uncertainty.
- In **Parameter Search**, the **Constructor** is ran through a Monte Carlo of (lightweight) simulations.
 - Activity final execution durations are varied based on a probabilistic model of plan execution.
 - All plan executions are then passed as the Solution to the Analyzer.
- The **Analyzer** assigns blame to every unscheduled activity.
- We present multiple **Prioritizer** variants
- Repeat until we find a priority set that satisfies some measure of “goodness” OR a certain step bound (time limit) is reached.
 - “Goodness” is evaluated through a scoring function described in Empirical Results



Constructor

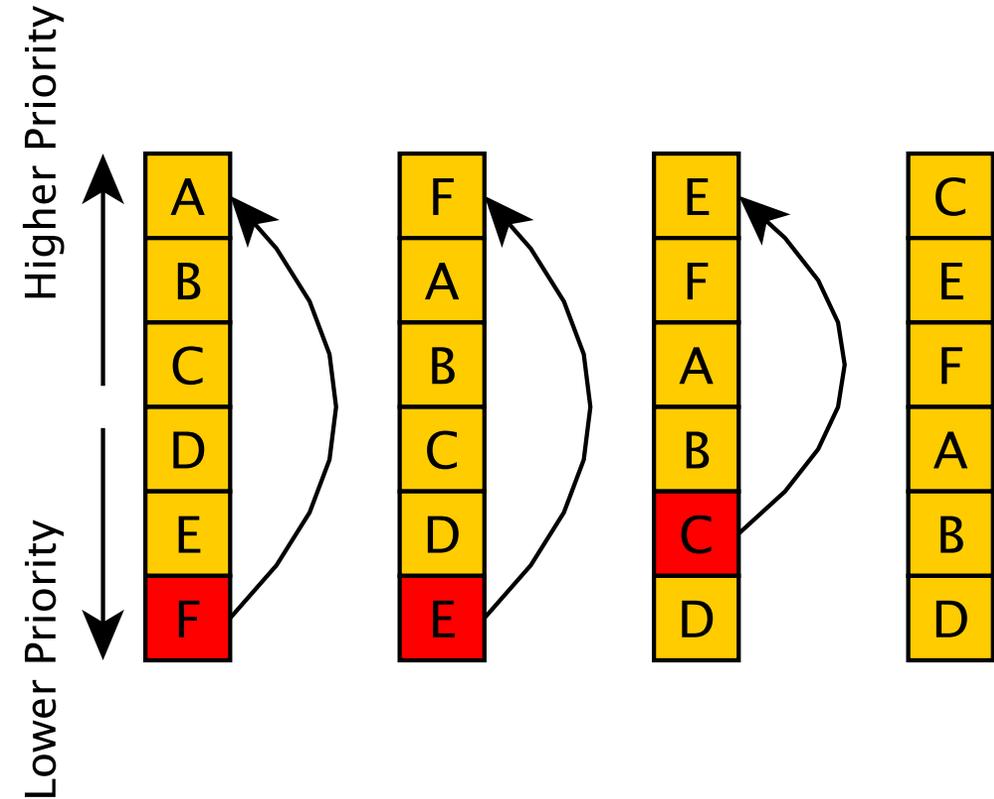


Analyzer (Scoring)

- Primary Metric: Number of activities scheduled
- Secondary metrics
 - Handover SOC – SOC leftover at the end of the plan
 - Cumulative distance from each activity's preferred time (lower is better)
 - Secondary metrics can be swapped or combined in a non-strict hierarchy

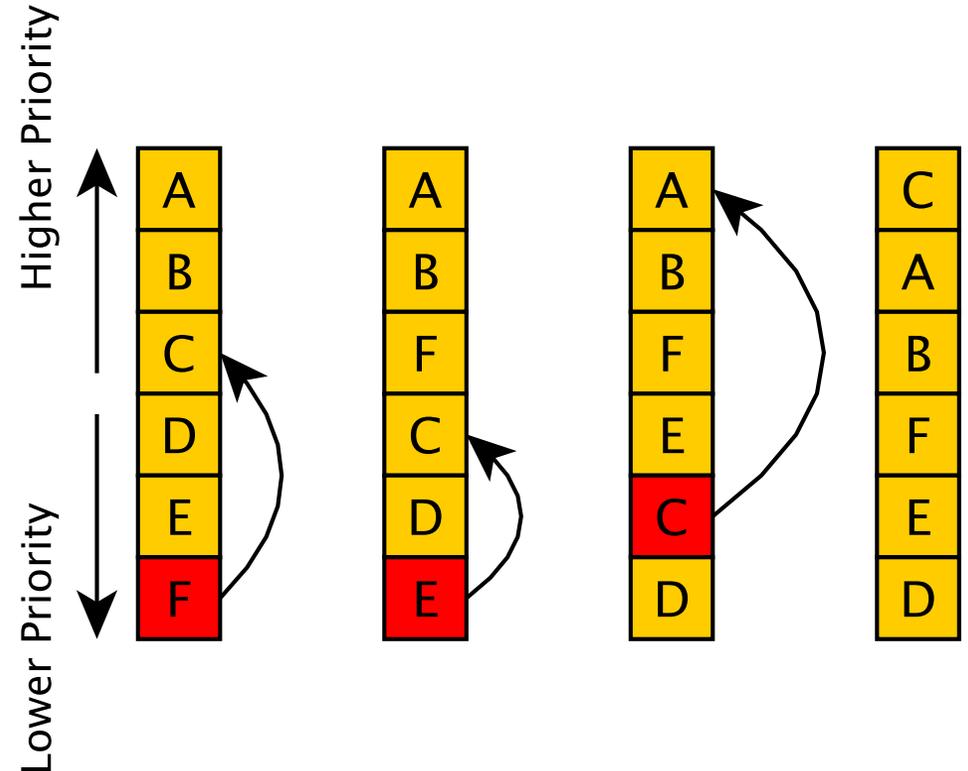
Max Step Reprioritization

- For each activity that is not scheduled, assign it the highest priority.
- Since this is a fixed operation, it is possible for Max Step Reprioritization to encounter a cycle for a valid input plan.
 - When a cycle is encountered, randomly restart.
- Can be too coarse in its search and promote activities more than necessary.



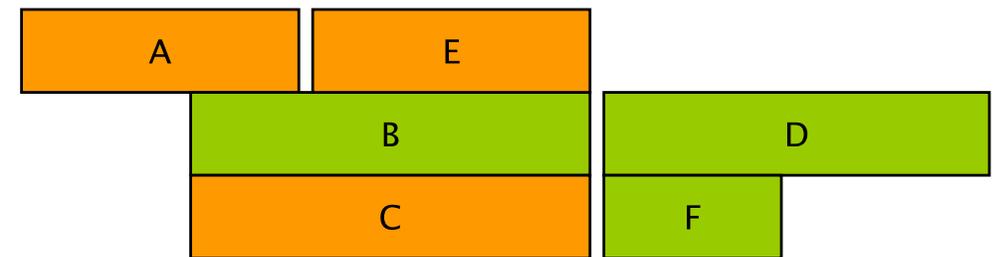
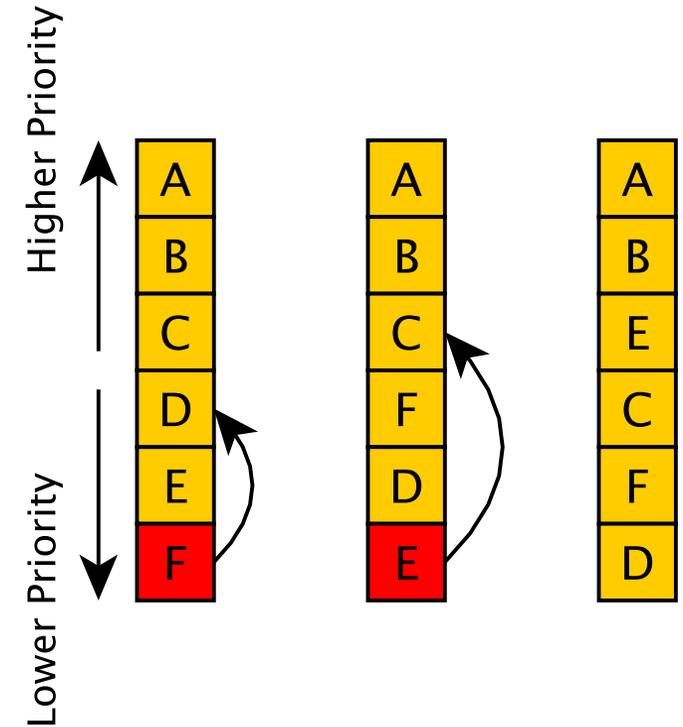
Stochastic Step Reprioritization

- For each activity that is not scheduled, increase its relative priority by a random $x \in \{1 \dots \text{len}(\text{activities})\}$.
- The randomness allows us to escape cycles, plateaus, and local maximas.
 - It also emulates search without actually having to search.
- Randomness doesn't guarantee the solution will be found.
- Runtime is not insignificant.



Intersect Reprioritization

- For each activity that is not scheduled, increase its priority above those that share resource bits and have intersecting execution time windows.
- Time is one of the most constraining resources. Therefore, only promote above activities where time is a conflicting resource.
 - Energy is much harder to impact by changing priorities.



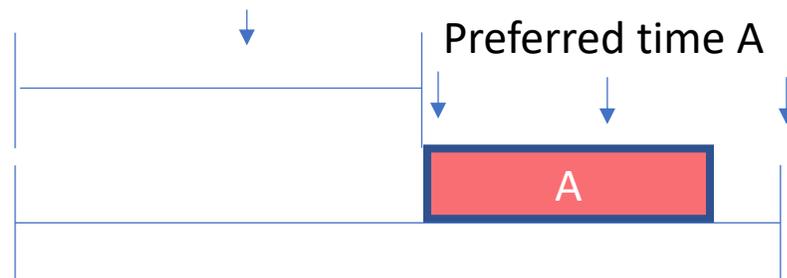
Intermediate Schedules

- Other methods focused on the final executed schedule, but the *intermediate schedules* may provide additional information.
- If an activity was executed, but not scheduled in any intermediate schedule, then its priority is increased by (*weight * number of failed schedules*)
- If an activity failed to execute then it's priority is increased by (*weight * number of failed executions*)

Preferred Time Manipulation

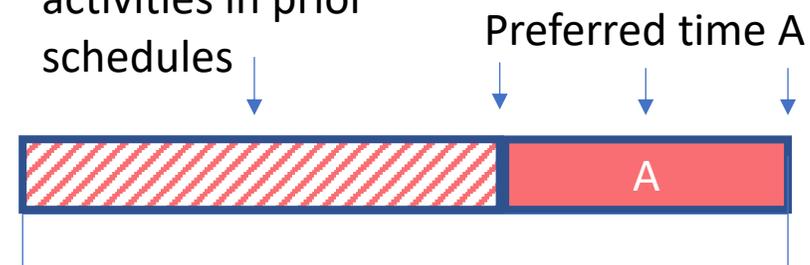
- 1) *Start Time Windows* - Preferred times of an activity is shifted such that the activity's expected placement does not intersect with the execution time windows of activities that share resource bits.
- 2) *Past Start Time* – Shift activities away from regions where activities sharing resource bits were successfully scheduled in previous Monte Carlos.
- Stochastically shift to Earliest Start, Midpoint, Latest Start, or random time within execution time window if (1) and (2) are unable to provide a place to schedule the activity.

Start Time Windows of
resource bit sharing activities



Start Time Windows

Resource bit sharing
activities in prior
schedules



Past Start Time

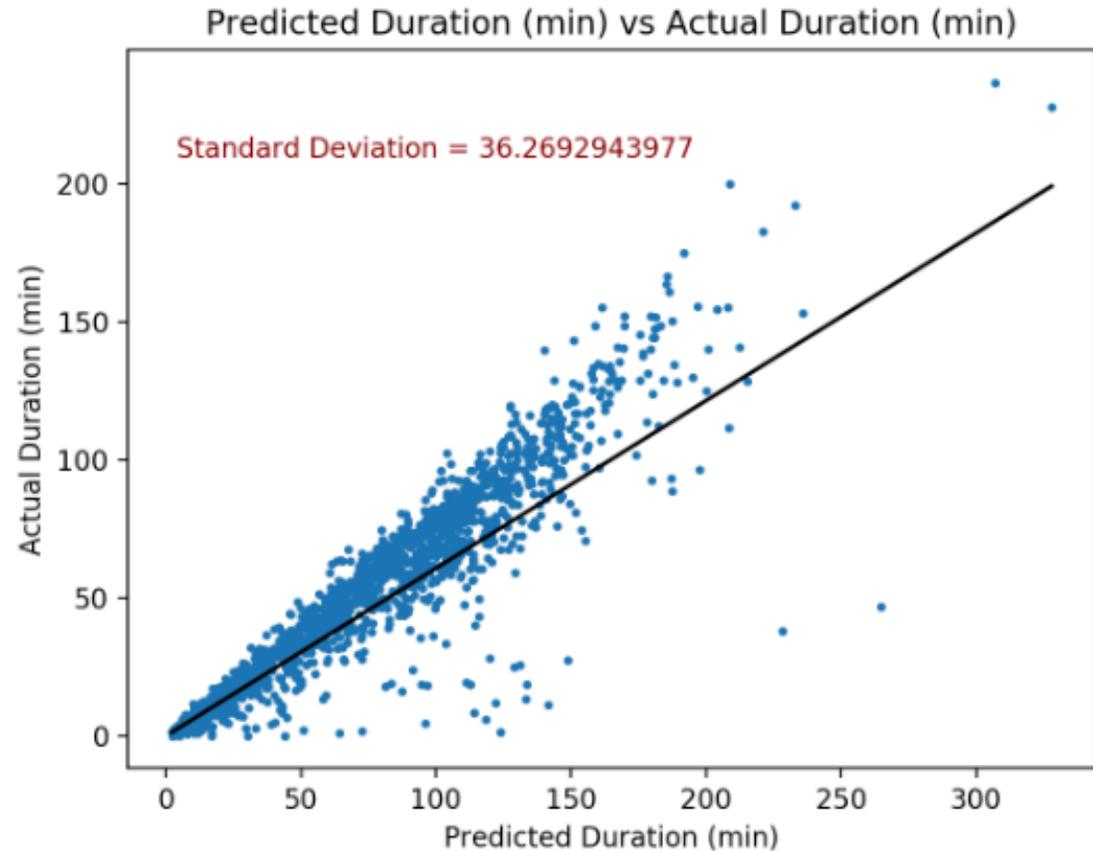
Portfolio

- Stochastically choose one of the previous heuristics at each step of the search.
- Currently a uniform distribution, but a better distribution could be learned.

Empirical Evaluation

Model to Vary Activity Durations

- Use predicted and actual durations from MSL Submaster Data
- Scale actual durations values by dividing by corresponding conservative durations
- Use linear regression on scaled values to derive mean and standard deviation
 - Assume ratio of predicted to actual execution times is normally distributed
- Value on regression line for conservative duration is mean
- Activities complete on average 32 % early



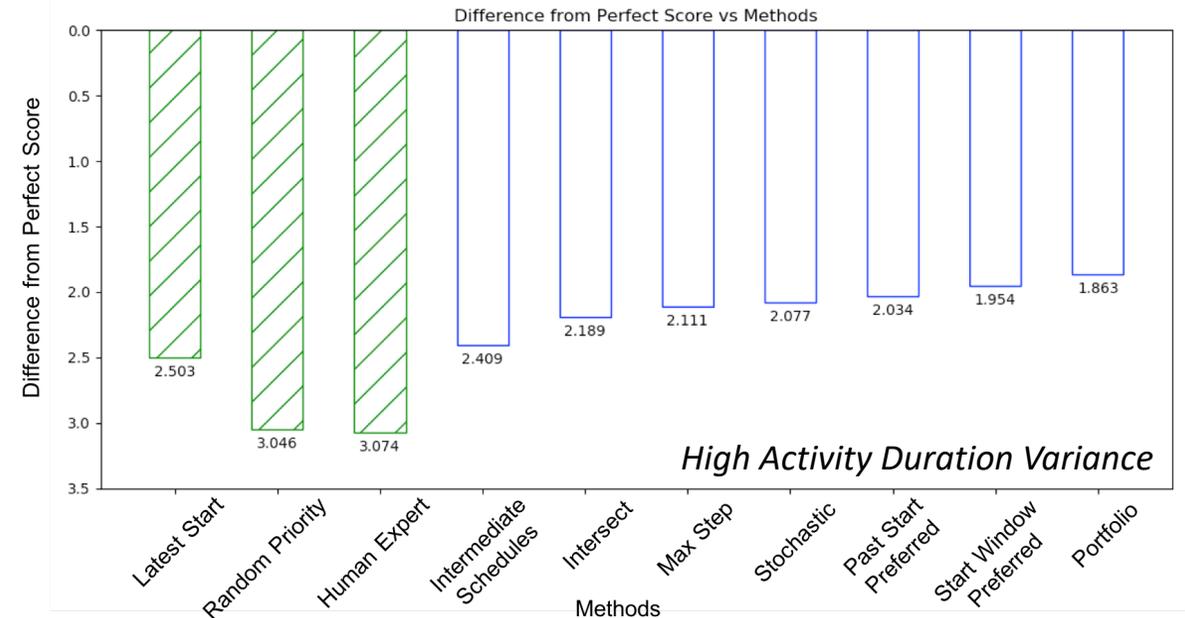
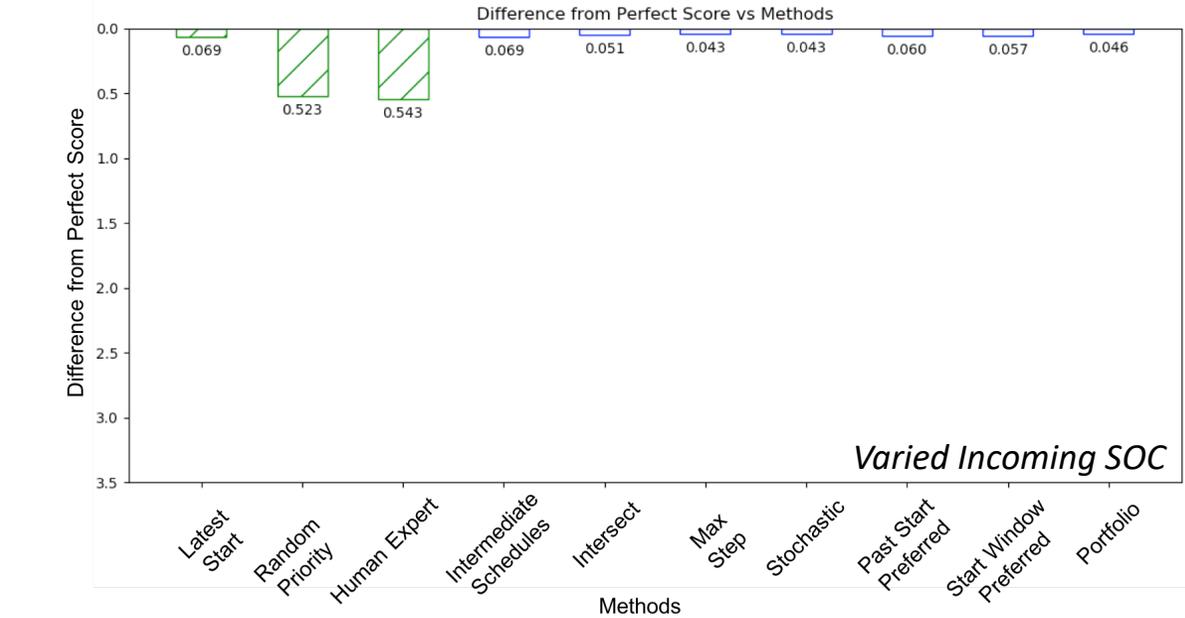
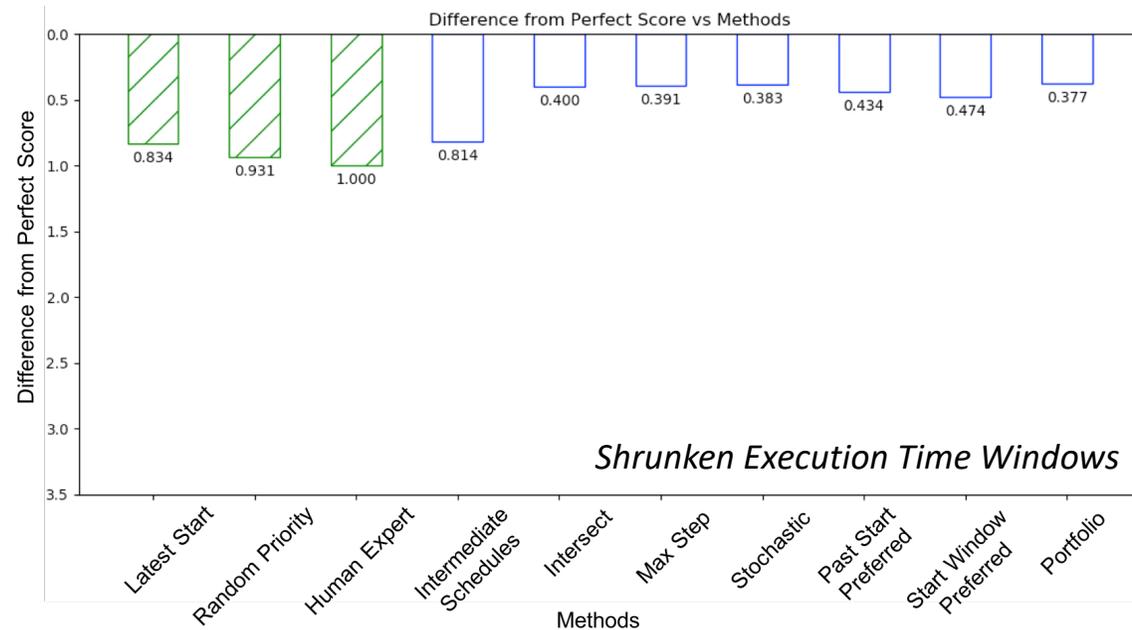
Sol Type Variations

- **Sol Types**- currently best available data on expected M2020 rover operations
 - Not always completely serial, contain execution, dependency constraints
 - 8 different sol type variations
- Average number of activities scheduled
 - Higher (lower value) is better

Activity Duration Variance	Shrink Execution Time Window	Varied Incoming SOC
0% activities finish late (mean 70%)	90% overall size (-5% each side)	100% maximum SOC
10% activities finish late (mean 70%)	80% overall size (-10% each side)	90% maximum SOC
20% activities finish late (mean 70%)	70% overall size (-15% each side)	80% maximum SOC
30% activities finish late (mean 70%)	60% overall size (-20% each side)	70% maximum SOC
40% activities finish late (mean 70%)	50% overall size (-25% each side)	60% maximum SOC

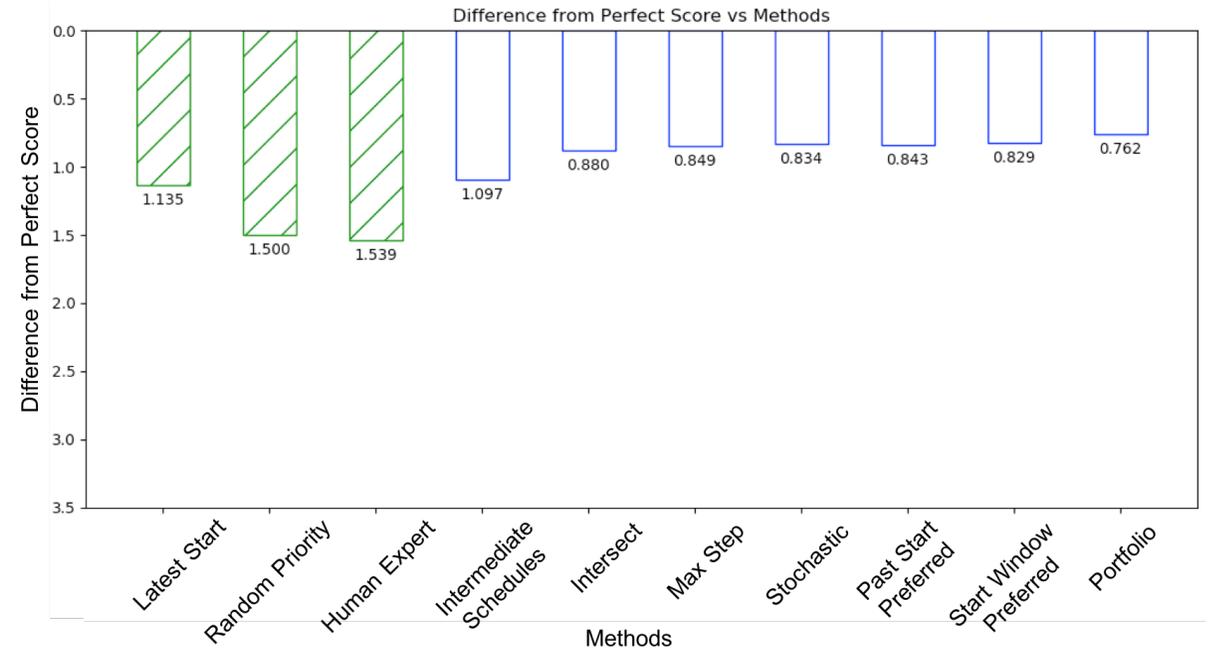
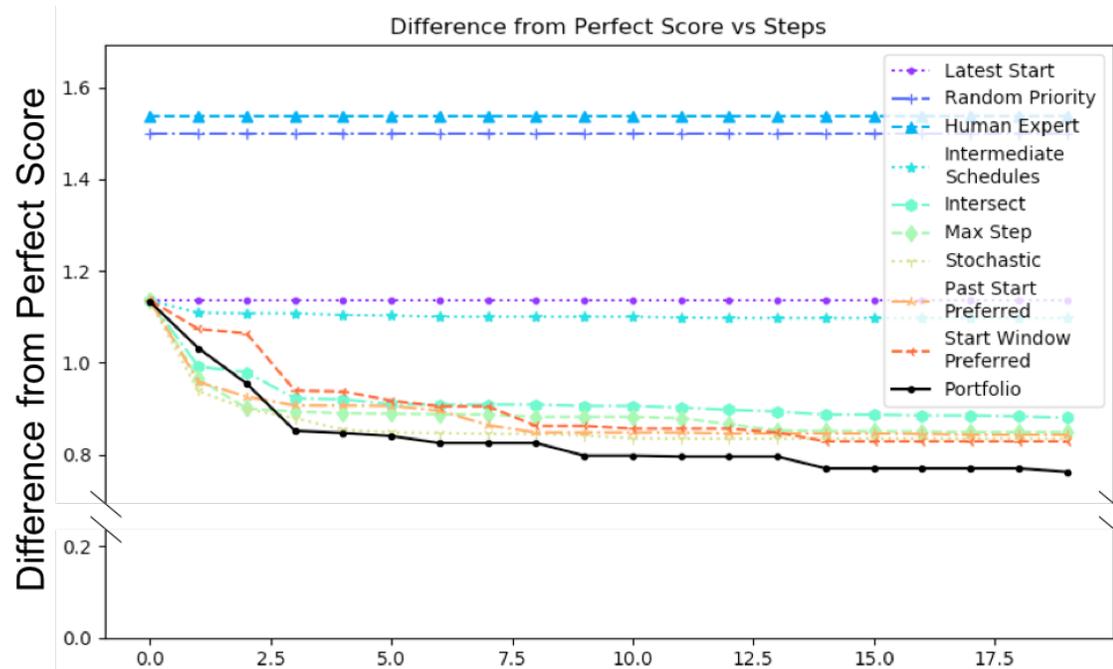
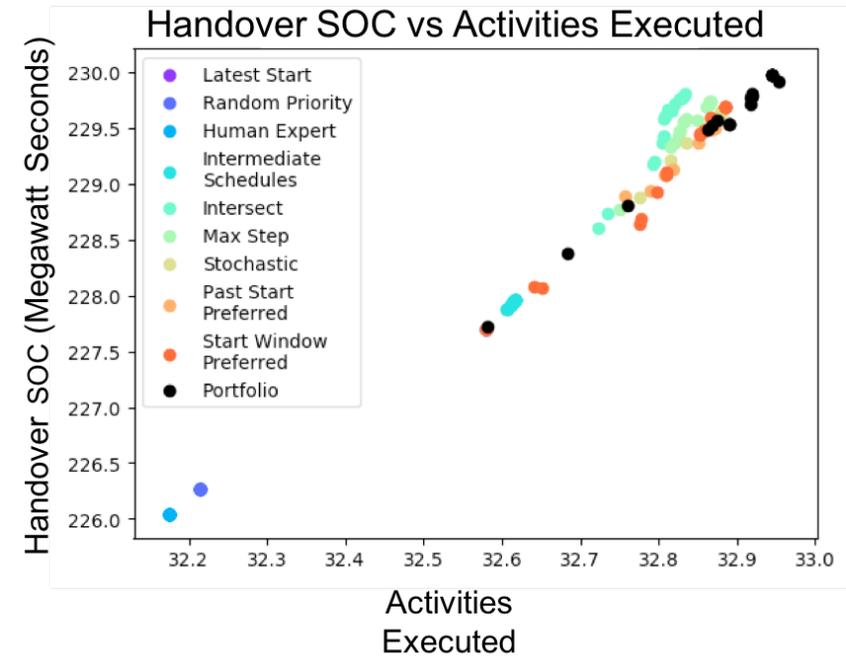
Results

- All Parameter Search methods consistently beat static methods
- Different methods perform better under different constraints
 - When activity duration variance is high, preferred time performs best
 - When ET windows are shrunk, priority is more important



Results

- Portfolio performs better than all other methods overall.
 - Converges faster
 - Increase handover SOC
 - Results are statistically significant ($p < 0.01$)



Future Work

- Learning techniques to improve portfolio weighting
 - Current approach is too naive
- Better analysis of activities to avoid undershooting or overshooting
 - Vaquero, T. et al., Temporal Brittleness Analysis of Task Networks for Planetary Rovers. *In Internal Conference on Automated Planning and Scheduling (ICAPS 2019)*, Berkeley, CA, USA, July 2019.
- Decreasing overall Monte Carlo runtime
 - Will allow for more time to search the parameter space

Conclusions

- Parameter setting for M2020 OBP is challenging search problem
- Offline (ground) priority setting with Monte Carlo over a probabilistic model of duration can be formulated as SWO (analogue) problem
- We have proposed and evaluated several approaches of Parameter Search to solve the activity parameter assignment problem.
- Parameter Search outperforms all static algorithms for activity parameter assignment.
 - A portfolio of Parameter Search methods allows for robustness to multiple types of plans