Developer Tools for Evaluating Multi-Objective Algorithms

Mark E. Giuliano and Mark D. Johnston

Space Telescope Science Institute
3700 San Martin Drive
Baltimore, MD 21218
Giuliano@stsci.edu

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109
mark.d.johnston@jpl.nasa.gov

Abstract

Multi-objective algorithms for scheduling offer many advantages over the more conventional single objective approach. By keeping user objectives separate instead of combined, more information is available to the end user to make trade-offs between competing objectives. Unlike single objective algorithms, which produce a single solution, multi-objective algorithms produce a set of solutions, called a Pareto surface, where no solution is strictly dominated by another solution for all objectives. From the end-user perspective a Pareto-surface provides a tool for reasoning about trade-offs between competing objectives. From the perspective of a software developer multi-objective algorithms provide an additional challenge. How can you tell if one multi-objective algorithm is better than another? This paper presents formal and visual tools for evaluating multi-objective algorithms and shows how the developer process of selecting an algorithm parallels the end-user process of selecting a solution for execution out of the Pareto-Surface.

Introduction

Effective scheduling of space based astronomy missions requires the ability to make trade-offs between competing mission objectives. A typical mission includes many objectives such as increasing time on target, minimizing use of consumables, minimizing the use of critical mechanisms, preferring the highest priority science first, etc. These objectives are often competing in that improving one objective means worsening another. Traditional scheduling optimization techniques are generally based on a single objective that combines all criteria into a single value, often by weighting the values of the individual objectives. However, this necessarily loses information about the individual components of the objective, and pre-determines the tradeoff among them. Multi-objective scheduling techniques allow the retention of separate objective components and thus for explicit visibility into trade-offs.

Previous work (Giuliano, Johnston 2010) describes visualization tools for end-users of a multi-objective scheduling system. The tools allow multiple distributed users to explore a Pareto surface trade-off space in order to converge on a single solution for execution. This paper builds on the previous results and presents tools from the perspective of a software developer building a multi-objective scheduling system.

Algorithms for solving multi-objective scheduling problems have been developed that are effective in building a uniformly sampled approximation of the Pareto surface (Kukkonen, Lampinen 2005). These algorithms typically have many variants to select from, ranging from settings of control parameters to alternate decompositions of a search problem. A developer obviously wants to select the best possible algorithm for delivery to operations. Single objective algorithms produce a single value as output allowing algorithms to be compared directly. In contrast, each multi-objective algorithm variant outputs a Pareto surface. A multi-objective algorithm is better than another if it produces a better Pareto-surface. The challenge for developers is to compare Pareto-surfaces. This paper presents a set of tools which allow developers to compare Pareto-surfaces and demonstrates the tools using two space based scheduling domains.

The remainder of the paper reads as follows. First, the system architecture of the multi-objective tools presented is described. Second, two multi-objective scheduling domains are briefly described. Third, the problem of evaluating algorithms is described. Fourth, formal tools for comparing Pareto-surfaces are described, Fifth, visualization tools for comparing Pareto-surfaces are described. Finally, the conclusion describes recursive relationships found in multi-objective scheduling.

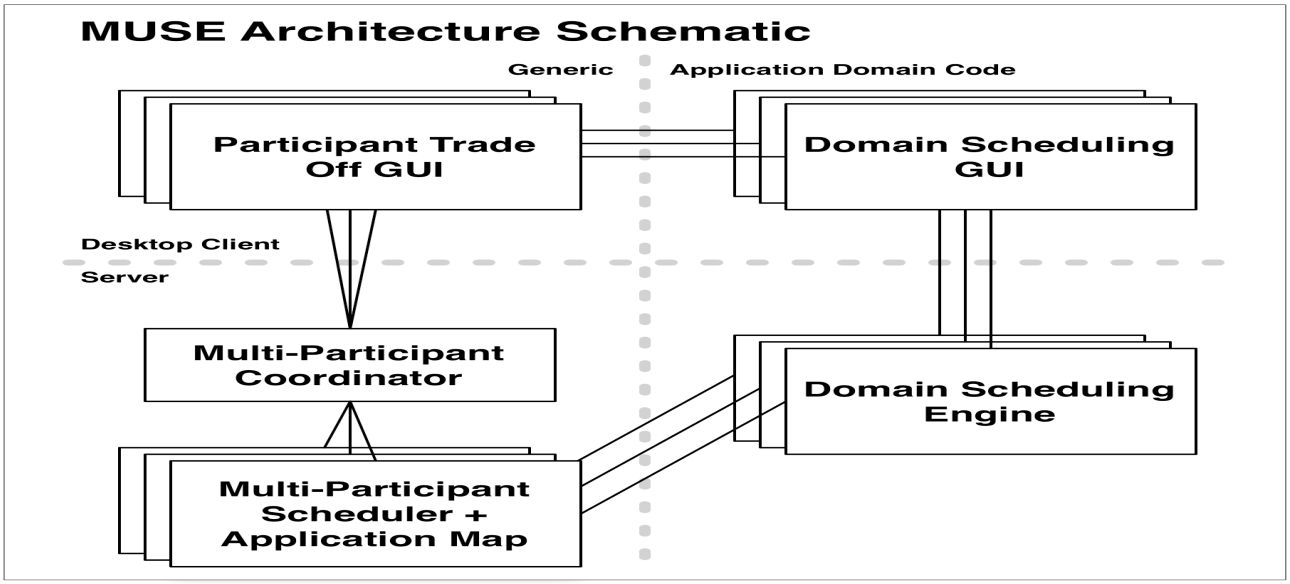

Figure 1: Multiple User Scheduling Environment (MUSE) system architecture

System Architecture

The tools described below are part of the Multi-User Scheduling Environment (MUSE) overall architecture, as illustrated in Figure 1 (Johnston and Giuliano 2009, Johnston Giuliano 2010). MUSE provides a generic environment for integrating existing tools (where they exist), providing persistent storage for various types of schedule data, and supporting both online and offline collaboration (in consideration of distributed users working across multiple time zones). MUSE incorporates server components (Fig. 1 lower half) as well as components that are resident on the user's workstation. MUSE distinguishes generic components (left) from those that may be highly domain specific (right). The architecture is designed so that domain specific components can be run as separate processes or can be compiled into the same image as the generic code.

On the server side, the Multi-Participant Coordinator acts as a central "clearing house" for schedule data, participant's selections, and scheduling runs. It provides an interface that communicates with the individual participants, providing up to date schedules, schedule status, and other participant selections of objective value ranges. The Multi-Objective Scheduler provides the evolutionary algorithm optimizer that evolves a population of candidate schedules towards the Pareto-optimal surface. The Application Map provides a transformation between decision variable values and domain-specific scheduling decisions as represented and evaluated in the Domain Scheduling Engine components. The Multi-Objective Scheduler supports parallel evaluations of schedules, which can frequently help speed the generation of a Pareto surface for participants. The Domain Scheduling Engine is the application-specific scheduling software that MUSE uses to evaluate candidate schedules. This evaluation utilizes the decision variable values, and can potentially perform internal conflict resolution or optimization steps on its own before returning a set of objective function values to the Multi-Objective Scheduler.

Application Domains

Data from two space based planning and scheduling applications are used to demonstrate Pareto-surface comparison techniques. First, we present data from an application of MUSE to scheduling the James Webb Space Telescope (Giuliano Johnston 2008). In this domain there are three objectives to:

- Minimize gaps in the schedule;
- Minimize momentum build-up in telescope reaction wheels used to move the telescope;
- Minimize observations which would be dropped as they missed their last opportunity to schedule.

The JWST scheduling scenario involves planning a 22 day scheduling bin. The observation data for these scheduler runs was obtained using the JWST Science Operations Design Reference Mission. Second, we present data from an application of MUSE to scheduling the Hubble Space Telescope (HST). For this domain we utilize the following scheduling objectives:

- Minimizing the number of observations that are not scheduled in the first year of the plan.
- Minimizing the number of observations that can be scheduled in South Atlantic Anomaly (SAA) orbits but are scheduled elsewhere. About half of HST's orbits cross a zone of radiation in the south Atlantic where no observing can take place. The idea is to find observations whose target occultation hide the SAA crossing.

- Minimize the number of orbits that are dropped from scheduling as they cannot be scheduled without resource conflicts.

The HST scheduling scenario involves long range planning for a cycles worth of data (~7000 observations), over the course of a year and a half planning session. The data and scenario for these schedule runs was obtained using a snap shot of the latest operational HST observing cycle.

For the JWST application two types of algorithm variants are considered. The first varies the number of generations versus the population size of each generation used in the MUSE evolutionary algorithm. Given a fixed number of evolutionary evaluations is it better to search with a large number of generations but a small population size (deep-but-narrow), or to have a search with a large population size but a small number of generations (broad-but-narrow)? The second algorithm variant considers two different decompositions of the search. A JWST search assigns time and roll to an observation. One algorithm variant assigns both decisions *all at once*. The other algorithm variant first assigns time and then performs a *delayed* roll search. The HST experimental data varies the number of generations versus the population size. Both the JWST and HST applications utilize the SPIKE planning and scheduling system (Johnston, Miller 1994) to implement the MUSE domain scheduling engine.

Evaluating Scheduling Algorithms

When evaluating a scheduling algorithm for inclusion in an operational system a developer must consider many factors including:

- The performance of the algorithm in terms of space and time.
- The ability of the algorithm to integrate with other systems both human and computer.
- The transparency of the algorithm. Is it easy to understand why the algorithm made a particular scheduling decision.
- The maintainability of the underlying code base.
- The quality of the solutions produced by the algorithm over different problem instances.

The factors listed above apply to both single and multi-objective algorithms. However, single and multi-objective algorithms differ in how they measure the quality of solutions. A single objective algorithm produces a single solution for any problem instance that maximizes an objective function that combines multiple criteria. The quality of different single-objective algorithms for a problem instance can be directly compared using the single objective function. In contrast, a multi-objective algorithm produces a Pareto-surface of solutions where no solution is dominated by another solution for all criteria.

The remainder of the paper concentrates on the problem of comparing Pareto-surfaces. First, formalisms for comparing surfaces are defined. Next, we present visualization tools that allow users to compare surfaces created by different algorithms. While presenting the graphical tools we present examples showing how the formalisms do and do not match our intuitions from the visual displays.

Note that the process of selecting a scheduling algorithm can itself be viewed a multi-objective optimization problem. There are multiple features that can be used to evaluate an algorithm. Although it would be fortunate, it is not likely that any one algorithm will dominate in terms of all the features. The job of the developer is to pick the algorithm with the best combination of features.

Formalisms for Evaluating Pareto-Surfaces

This section examines the use of mathematical formalisms for evaluating Pareto-surfaces. (Zitzler et al. 2003) contrasts *unary* Pareto surface evaluation functions that measure the quality of a single Pareto surface with *binary* evaluation functions that compare the quality of two Pareto surfaces. They show that no unary function or combination of unary functions can tell whether or not a Pareto surface P_1 is strictly better than a Pareto surface P_2. That is there is no function F such that $F(P_1) > F(P_2)$ if and only if surface P_1 strictly dominates surface P_2. In contrast binary evaluation functions can be designed to tell if one surface is better than another surface. For example, using the comparison operator from [Giuliano and Johnston 2008], first construct the combined Pareto frontier of the two surfaces $\text{Combined}(P_1,P_2)$. If $\text{Intersect}(P_1,\text{Combined}(P_1,P_2)) == P_1$ and $\text{Intersect}(P_2,\text{Combined}(P_1,P_2)) == \text{null}$ then surface P_1 dominates surface P_2.

If there is no strict domination between surfaces, then we cannot determine if one surface is better than another. The purpose of a multi-objective algorithm is to produce a surface of Pareto optimal solutions so that a user can explore the space and ultimately select one of the solutions for execution. If there is no strict domination between a pair of surfaces, then the combined surface will contain elements from both input sets and the end user may prefer an element from either set. Without additional knowledge no formulation can unequivocally distinguish between pairs of non-dominating surfaces. In this way the selection of a multi-objective algorithm is like the selection of a solution out of a Pareto optimal frontier. Since there are multiple objectives there is no formal way to make the selection if there is no dominating solution.

Despite these pessimistic truths analysts and system developers who code and set parameters for multi-objective algorithms need tools for distinguishing between algorithms beyond the notion of strict dominance. In this paper we explore some formulations that measure the quality of a Pareto surface and show how the formulations do and do not match the intuitions obtained from plots of schedule values. We start with two binary evaluation functions. The E-indicator [Zitzler, et al. 2003] gives the factor by which one Pareto surface is worse than another

with respect to all objectives. In other words the value of $E(P_1,P_2)$ is the minimum factor e such that for any solution in P_2 there exists a solution in P_1 that is not worse by a factor of e in all objectives. The E-indicator can be used to detect surface dominance as follows. If $E(P_1,P_2) < 1$ and $E(P_2,P_1) > 1$ then P_1 dominates P_2. If $E(P_1,P_2)$ is smaller than $E(P_2,P_1)$ then the indicator implies that P_1 is preferable to P_2. The second binary indicator is based on the naïve comparison operator in [Giuliano and Johnston 2008]. This comparison is called the F-indicator and is the fraction of P_1 that occurs in the combined surface. Define $F(P_1,P_2) = \text{length(Intersect(P1,C))/Length(C)}$ where C is the combined Pareto surface of P_1 and P_2, Intersect returns the set intersection of two Pareto-surfaces, and Length gives the number of elements in a surface.. By definition $F(P_1,P_2) + F(P_2,P_1) = 1$. If $F(P_1,P_2) > F(P_2,P_1)$ then according to this metric P_1 is preferable to P_2. This corresponds to the intuition that the algorithm that provides a larger fraction of the combined surface is preferable to an algorithm that produces a smaller fraction of the combined surface..

Visualization Tools

A traditional method to displaying a Pareto-surface is to use a series of X-Y trade-off plots. X-Y plots can be used to compare algorithms simply by plotting solutions from a different algorithm with different colours. Although X-Y plots are intuitive to understand, they have several problems. First, the number of plots grows geometrically as the number of objectives increases. Second, it is hard to connect a point in one plot with the corresponding points in other plots. An alternate view of the data is a parallel coordinate plot where each solution is represented by a single line that runs down the plot vertically. Each criteria is represented by a row where values are plotted horizontally on a normalized scale. For example, Figure 2 illustrates both X-Y plots and a parallel coordinate plot for a pair of JWST schedule runs that vary the number of generations versus the size of each generation. Figure 3 gives a parallel coordinate plot comparing a pair of JWST schedule runs that utilize different decompositions of the search space.

Schedule Run	E metric	F Metric
Deep-but Narrow	1.61	0.58
Broad-but-Shallow	3.45	0.42
Delayed	1.58	0.60
All-at-Once	1.98	0.40

Table 1: Shows binary performance metrics for two separate JWST algorithm evaluations.

The E and F metrics for the data in figures 2 and 3 are shown in table 1. Both metrics prefer the Deep-but-narrow run over the broad-but-shallow run and the Delayed run over the All-at-once run. These values correspond to our intuitions based on examining the plots. However, neither metric captures obvious features of the graphs. For example, the deep-but-narrow search performs best for the

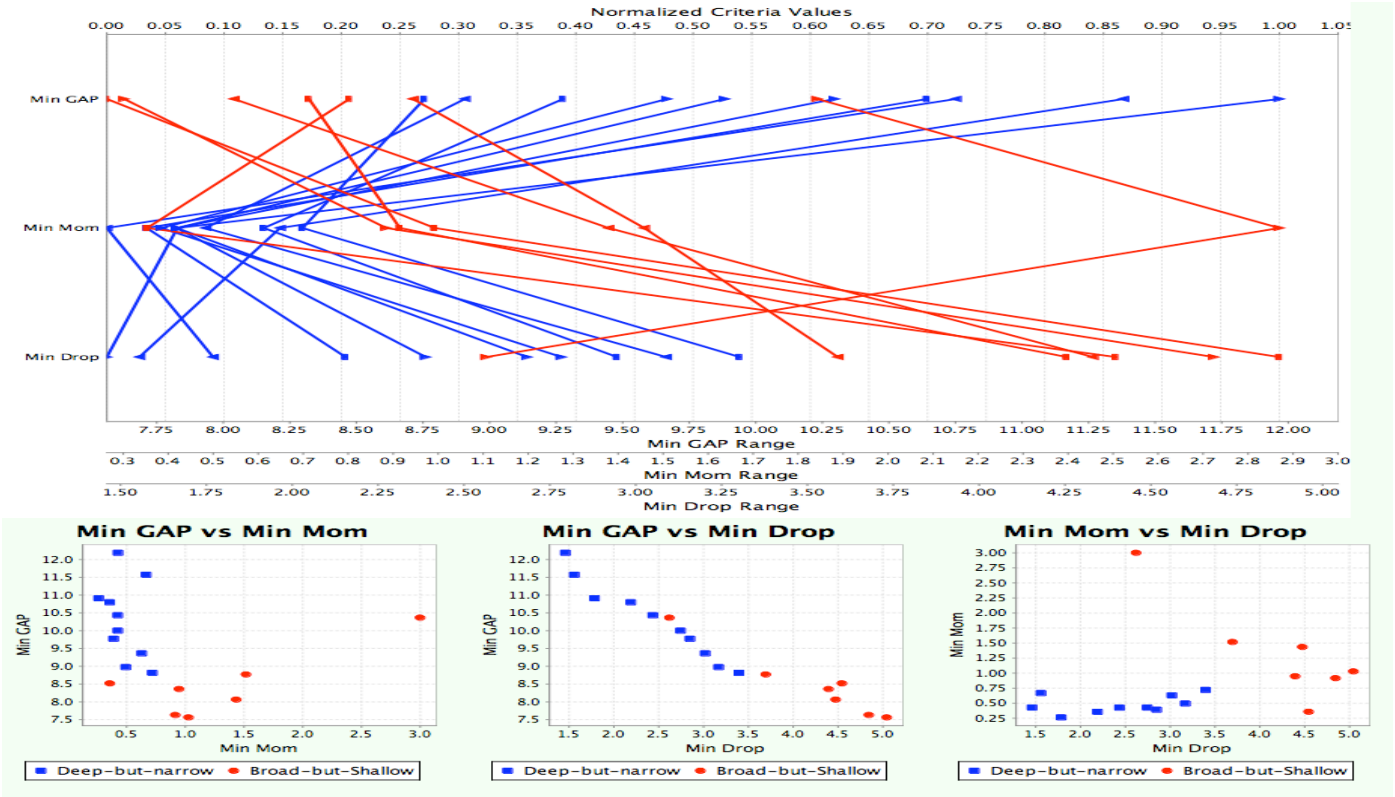

Figure 2: Muse Snapshot showing X-Y trade-off plots and parallel coordinate plot comparing two algorithm variants for JWST scheduling. Blue points represent deep-but-narrow solutions and red broad-but-shallow solutions.

min-dropped and minimum momentum criteria while the broad-but-shallow search performed best in terms of gaps. Likewise in Figure 3 the delayed search performs much better in terms of the momentum criteria. These types of visual observation are critical to our understanding of the algorithms but are not apparent in the metrics. While metrics can detect domination between surfaces and give a high level comparison of surfaces visual tools provide needed insight into how surfaces compare to each other. To this end we developed additional graphical tools for comparing surfaces.

Coordinate plots solve the problems with X-Y plots in that all the data for a single point is grouped together and the number of plots grows linearly with the number of criteria. However, coordinate plots can become unreadable with a large number of points on a Pareto-surface. For example, Figure 4 shows a plot comparing a narrow-but-deep search (in blue) with a broad-but-shallow search (in red) for an HST application of MUSE. With so many points on the surface it becomes hard to understand the data. To this end we investigated techniques for displaying interesting subsets of the Pareto-surface. The GDE3 algorithm as used in MUSE uses a *crowding distance* measure to reduce the size of a generation to the population size if the number of non-dominated solutions in a generation is larger than the population size. The idea is to prefer solutions with greater diversity in terms of the criteria. We re-used the crowding metric to sort the Pareto-surface and to display the most interesting fraction of the surface. Figure 5 displays the top 25% of solutions for the same experiments as in Figure 4. The difference between Figures 4 and 5 is striking. With the full surface, as displayed in Figure 4, the narrow-but-deep search dominates the display contributing 76 percent of the solutions. Displaying the 25% most diverse solutions the broad-but-shallow search contributes the same number of solutions as the narrow-but-deep search. By examining the most interesting subset of a surface the tool provides a visualization of the relative diversity of solutions produced by each algorithm variant.

A key lesson is that no single visualization of a Pareto-Surface is always the best so a GUI needs to provide multiple views. With this goal the MUSE interface provides several additional features that allow a user to dynamically explore a Pareto-surface. First, MUSE provides a tabular view of the data that supports dynamic sorting. Second, we provide a plot for each criteria that graphs the criteria values in order. Third, the user can select a solution in one plot and have the point highlighted in all of the plots. Each of the plots is linked so selecting a solution or region in one plot highlights the corresponding solution or region in other plots. Finally the tool provides the option of scaling each graph to only those points which are displayed versus keeping a constant scaling.

Conclusions and Future Work

A series of tools were described that allow developers to evaluate the quality of solutions produced by multi-objective algorithms. Formal binary evaluation functions were defined that can detect strict domination between pairs of surfaces. However these formalisms were shown to be of limited utility as strict domination of surfaces is rare, and the formalisms do not manifest structural features that can easily be seen in graphical displays of surfaces. A set of dynamic tools were presented that allow developers to compare and contrast surfaces using different views of the data.

Several recursive relationships were presented in the paper. The selection of an algorithm is itself is a multi-objective problem as a developer has multiple criteria with which to judge an algorithm including performance, usability, ease of maintenance, and the quality of solutions selected. Likewise the process of comparing the quality Pareto-surfaces is like the process an end user will use to select a

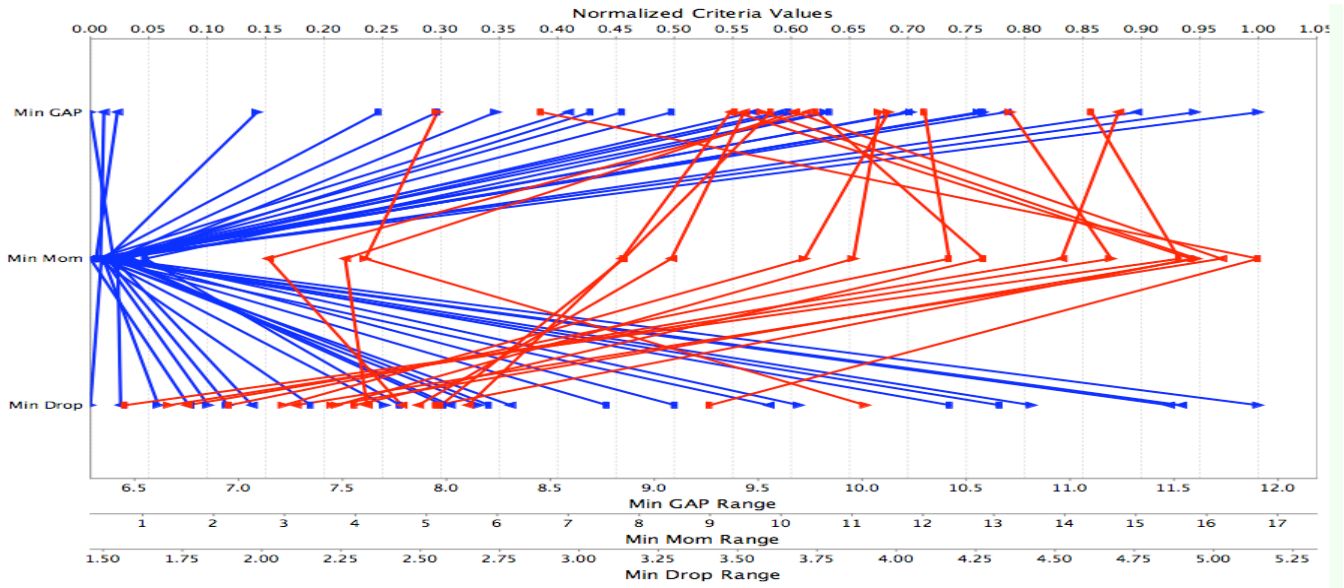

Figure 3. A MUSE snapshot showing a parallel coordinate plot comparing two JWST algorithm variants for an all-at-once search (red) and a delayed search (blue).

solution out a Pareto-surface. If there is strict domination the choice is easy, otherwise graphical tools provide the best way to pick a solution.

The SPIKE team is currently in the process of refactoring the HST SPIKE engine to be used in JWST (Giuliano et al 2011). As part of this effort a multi-objective component has been added to the SPIKE core and can be used in both JWST and HST. As these tools are integrated into operations visualization tools such as those described above will be key for both developers and end users.

Acknowledgement

The research described in this paper was carried out at the Space Telescope Science Institute, and at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration, under the NASA Applied Information Systems Research Program grant number NNX07AV67G.

References

Giuliano, M.E., and Johnston, M.D. 2008. Multi-Objective Evolutionary Algorithms for Scheduling the James Webb Space Telescope. In *International Conference on Automated Planning and Scheduling (ICAPS)*, Sydney, Australia.

Giuliano, M.E., and Johnston, M.D. 2010. Visualization Tools For Multi-Objective Algorithms. Demonstration: *International Conference on Automated Planning and Scheduling (ICAPS)*, Toronto, Canada.

Giuliano, M.E., Hawkins R, Rager R., 2011. A Status update on JWST Long Range Planning Developments. In *IWPSS Darmstadt, Germany*

Johnston, M.D. and Giuliano, M.E, 2009. MUSE: The Multi-User Scheduling Environment for Multi-Objective Scheduling of Space Science Missions. In *IJCAI Workshop on Space Applications of AI*, Pasadena, CA.

Johnston, M.D. and Giuliano, M.E., 2010. Multi-User Multi-Objective Scheduling for Space Science Missions. In *SpaceOps 2010*, Huntsville, AL.

Johnston, M.D. and Miller, G.E. 1994. Spike: Intelligent Scheduling of Hubble Space Telescope Observations. In *Intelligent Scheduling*, M. ZWEBEN AND M. FOX Eds. Morgan Kaufmann, San Mateo, 391-422.

Kukkonen, S. and Lampinen, J. 2005. GDE3: The Third Evolution Step of Generalized Differential Evolution. In *The 2005 Congress on Evolutionary Computation*, 443.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M. and Grunert Da Fonseca, V. 2003. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE transactions on evolutionary computation 7*, 117-132.

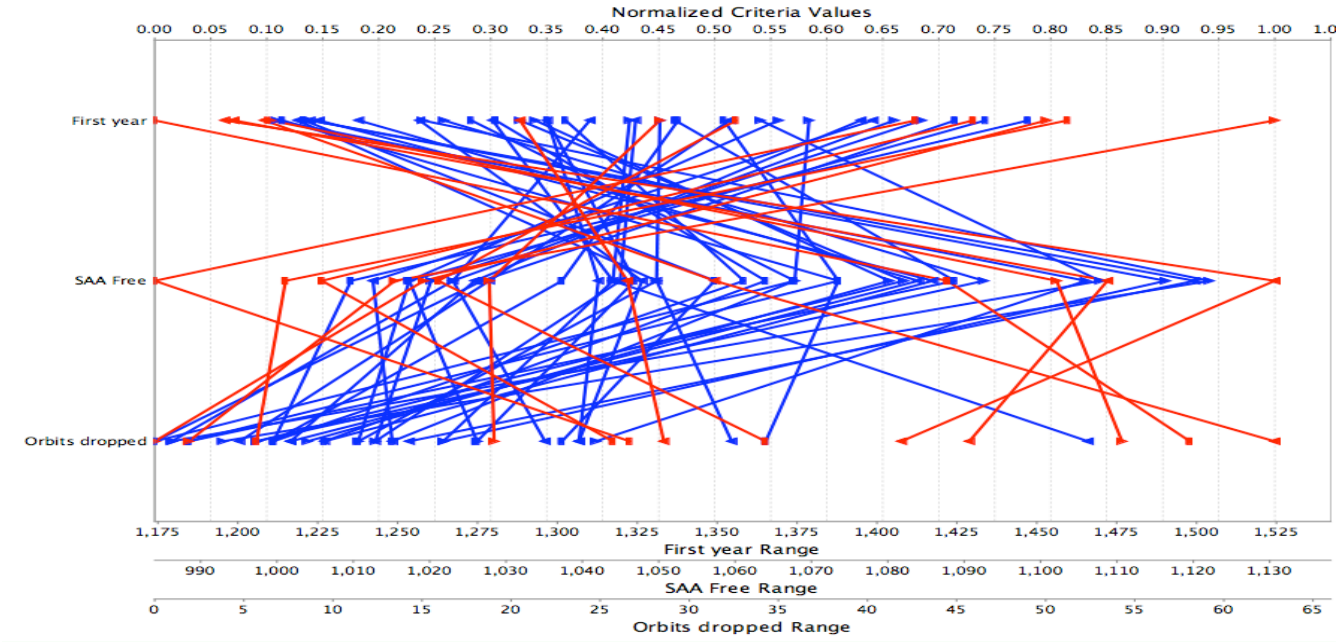

Figure 4: Muse Snapshot showing X-Y trade-off plots and parallel coordinate plot comparing two algorithm variants for HST scheduling. Blue points represent deep-but-narrow solutions and red broad-but-shallow solutions. This plot shows all the data. Note that the blue deep but narrow search dominates the display.

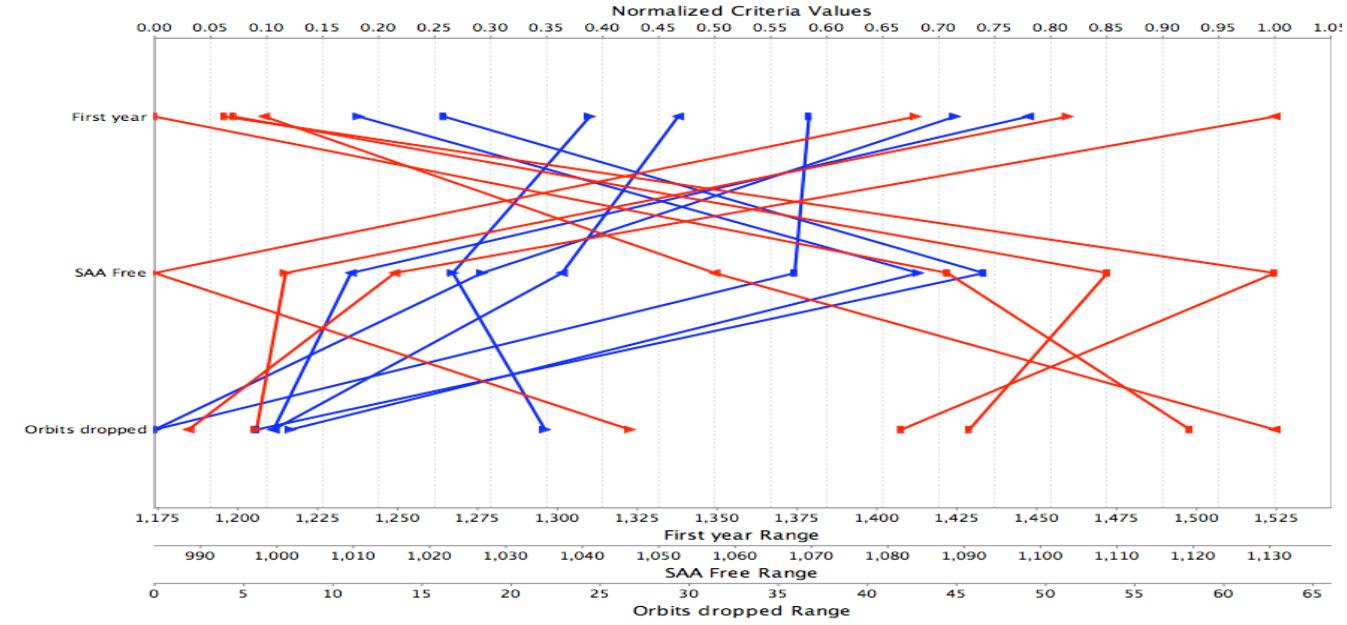

Figure 5. Muse Snapshot showing the 25% most interesting solutions from Figure 4. Note that the parity between the red broad but shallow search and the blue deep but narrow search in this display.