Trade Space Exploration of a Wing Design Problem Using Visual Steering and Multi-Dimensional Data Visualization

Timothy W. Simpson,[*] Daniel E. Carlsen,[†] and Christopher D. Congdon[‡]
The Pennsylvania State University, University Park, PA 16802, USA

Gary Stump[§] and Michael A. Yukish[¶]
The Applied Research Laboratory, State College, PA 16804, USA

Trade space exploration is a promising decision-making paradigm that provides a visual and intuitive means for formulating, adjusting, and ultimately solving multi-objective design optimization problems. This is achieved by combining multi-dimensional data visualization techniques with visual steering commands to allow designers to "steer" the optimization process while searching for the best, or Pareto optimal, designs. In this paper, we investigate the impact of constraint handling on the trade space exploration process. Specifically we consider three different constraint handling methods: (1) no constraint handling, (2) manual constraint handling, and (3) automatic constraint handling, and assess their impact on the efficiency and effectiveness of the visual steering commands used to explore the trade space. We find that the performance of the constraint handling method is highly correlated with the visual steering command that is being used and is consistent with the user's *a priori* knowledge about the constraints, which is reflected in how constraints are handled in each method. The implications of these findings on the trade space exploration process are also discussed in conjunction with future work.

I. Introduction

ENGINEERING designers frequently use optimization-based tools and approaches to help them make decisions when designing complex systems such as automobiles, aircraft, and spacecraft. Designing complex systems such as these requires tradeoffs between multiple conflicting and competing objectives, and trade space exploration is a promising alternative to optimization-based approaches for solving these types of problems. Trade space exploration provides a visual and more intuitive means for formulating, adjusting, and ultimately solving multi-objective design optimization problems and is an embodiment of the Design by Shopping paradigm advocated nearly a decade ago by Balling[1] based on his work with a group of urban planners. In short, the goal for Balling and his team was to develop a zoning plan for a large city while satisfying multiple objectives (12 in total) such as minimum traffic congestion and maximum green space.[2] In their first iteration, they obtained a relative ranking of the importance of these objectives and then optimized the plan for the city. Upon completion, they presented the results to the urban planners, which included a graphical description of the proposal city layout. The urban planners felt that the plan presented did not reflect their true desires and adjusted their preference, and the process was repeated for several iterations. The team modified their approach from optimizing to a single design to presenting a set of best designs, namely, the set of Pareto points. They created graphical tools to help the urban planners explore the Pareto front, and they used this information to obtain what they considered to be the best design.

This is one example of many that reinforces the notion that designers want to be able to "shop" for the best design, to gain intuition about trades, to see what is feasible and what is not, and to learn about their alternatives before making a decision. Balling[1] has noted that the traditional optimization-based design process of "1) formulate the design problem, 2) obtain/develop analysis models, and 3) execute an optimization algorithm" often leaves

[*] Professor of Mechanical and Industrial Engineering and Engineering Design, 314D Leonhard Building, Senior Member AIAA. Corresponding Author. Phone/fax: (814) 863-7136/4745. Email: tws8@psu.edu.
[†] Graduate Research Assistant, Mechanical & Nuclear Engineering, Applied Sciences Building.
[‡] Graduate Research Assistant, Industrial & Manufacturing Engineering, Applied Sciences Building, Student Member AIAA.
[§] Research Assistant, Product & Process Design Department, Applied Sciences Building.
[¶] Head, Product & Process Design Department, Applied Sciences Building, Member AIAA.

designers unsatisfied with their results because the problem is usually improperly formulated: "the objectives and constraints used in optimization were not what the owners and stakeholders really wanted…in many cases, people don't know what they really want until they see some designs". Similarly, Shanteau[3] observed that when people are dissatisfied with the results of a rational decision making process, they often change their ratings to achieve their desired result. Meanwhile, Wilson and Schooler[4] found that people do worse at some decision tasks when asked to analyze the reasons for their preferences or evaluate all the attributes of their choices.

In this paper, we present results from ongoing research that is investigating multi-dimensional visualization tools and visual steering commands to support the trade space exploration process.[5] In particular, we investigate the impact of constraint handling on the trade space exploration process. We consider three different constraint handling methods – no constraint handling, manual constraint handling, and automatic constraint handling – and assess their impact on the efficiency and effectiveness of the visual steering commands used to explore the trade space. Related work and our previous work are summarized next, and the wing design problem used in this study is described in Section III. The experimental set-up and results of our study are presented in Section IV, and closing remarks and future work are discussed in Section V along with the implications of the results.

II. Related and Previous Work

The notion of trade space exploration is most akin to work in *computational steering* whereby the user interacts with a simulation during the optimization process to help "steer" the search process toward what looks like an optimal solution. The user, or designer in our case, observes some sort of a visual representation of the optimization process and then uses intuition, heuristics, and/or some other methods to adjust the design space to move toward something that may not have been intuitive at the beginning of the simulation. For instance, Kesavadas and Sudhir[6] created large-scale manufacturing simulations by allowing users to make quick changes "on-the-fly" and continue with the simulation. Wright, et al.[7] used computational steering to design the geometry and select the grade of glass for a furnace. Visual Design Steering[8,9] allows users to stop and redirect the optimization process to improve the solution; however, their visualization capabilities are currently limited to 2-D and 3-D representations of constraints and objectives. Messac and Chen[10] proposed an interactive visualization method wherein the progress of the optimization is visualized – but not steered – throughout the process. Michalek and Papalambros[11] allow designers to "dynamically change the optimization representation on-the-fly by adding, deleting, and modifying objectives, constraints, and structural units" but their methodology is specific to the architectural layout problems that they solve. Madar, et al.[12] are investigating the effects of human interaction on a particular optimization algorithm, namely, particle swarm optimization. By using their visual, cognitive, and strategic abilities, users can improve the performance of the computer search algorithm by combining expert knowledge with computational power. Scott, et al.[13] investigated the effects of integrating humans into the optimization process and also found that "combining the human's superior intelligence with the computer's superior computational speed can result in better solutions than neither could produce alone". Additional advantages of such approaches include learning about the problem as well as the interrelationships between objectives and having the ability to guide the solution process in a desired direction and possibly even changing one's mind while learning.[14] Solutions generated through human interaction are better understood by the user than solutions returned by an optimization algorithm. Moreover, the computational costs can be significantly reduced since only solutions of interest to the decision-maker are generated.[13]

As part of our work in trade space exploration, collaborations between researchers at Penn State University and the Applied Research Laboratory (ARL) have led to the development of the ARL Trade Space Visualizer (ATSV).[15,16] The ATSV is a Java-based application that is capable of visualizing multi-dimensional trade spaces using glyph, 1-D and 2-D histogram, 2-D scatter, scatter matrix, and parallel coordinate plots, linked views,[17] and brushing.[18] Figure 1 shows several examples of the multi-dimensional data visualization capabilities in ATSV. We note that the 3D glyph plot (top left) can display up to seven dimensions by assigning variables to the x-axis, y-axis, z-axis, size, color, orientation, transparency of the glyph icons .

The design variable (input) and performance (output) data for different alternatives can either be generated off-line and then read into ATSV for visualization and manipulation or it can be generated dynamically "on-the-fly" by linking a simulation model directly with ATSV using its Exploration Engine capability.[5] If the simulation model is too computationally expensive to be executed in real-time, then low-fidelity metamodels can be constructed and used as approximations for quickly searching the trade space.[19] Once this link to the simulation model is in place, ATSV provides a suite of controls to help designers navigate and explore the trade space, including visual steering commands to (1) randomly sample the design space, (2) search near a point of interest, (3) search in a direction of preference, or (4) search for the Pareto front. A summary of each type of sampler follows; we refer the reader to Ref. [5] for more details.

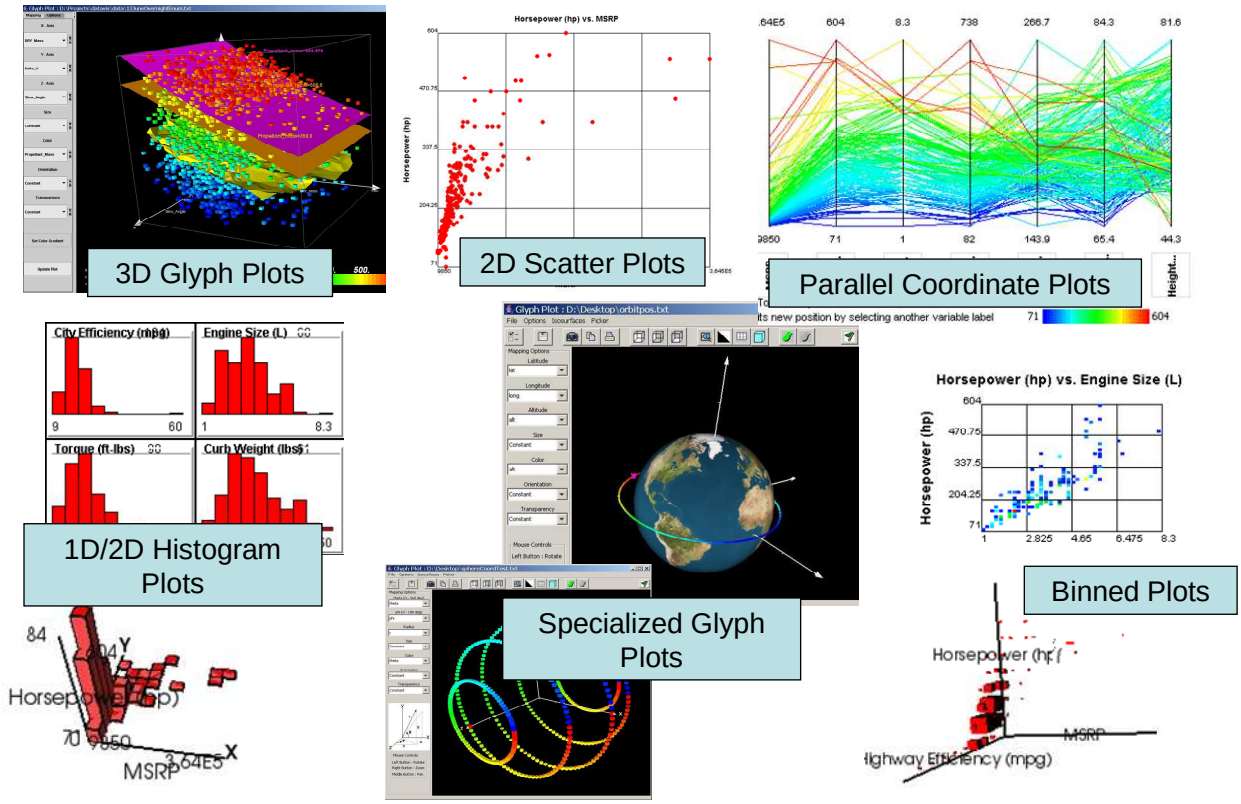
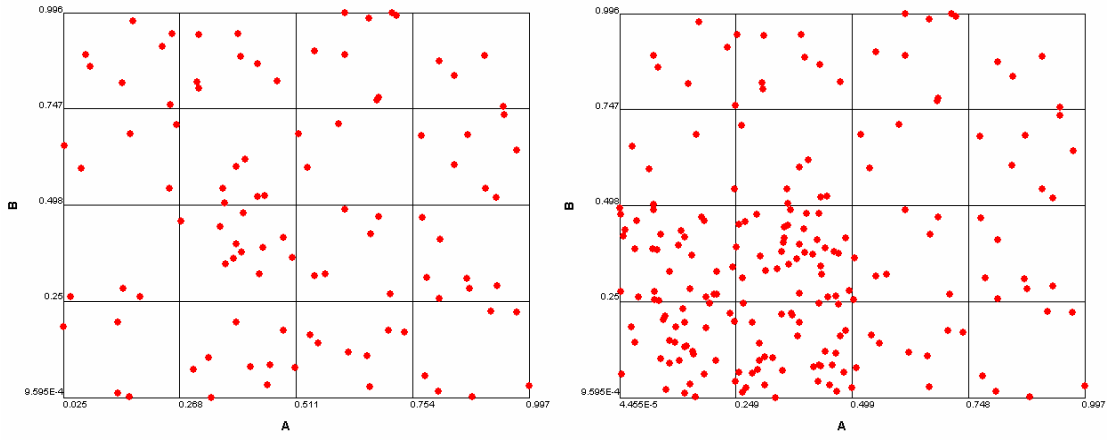


Figure 1. Multi-dimensional data visualization capabilities in ATSV.

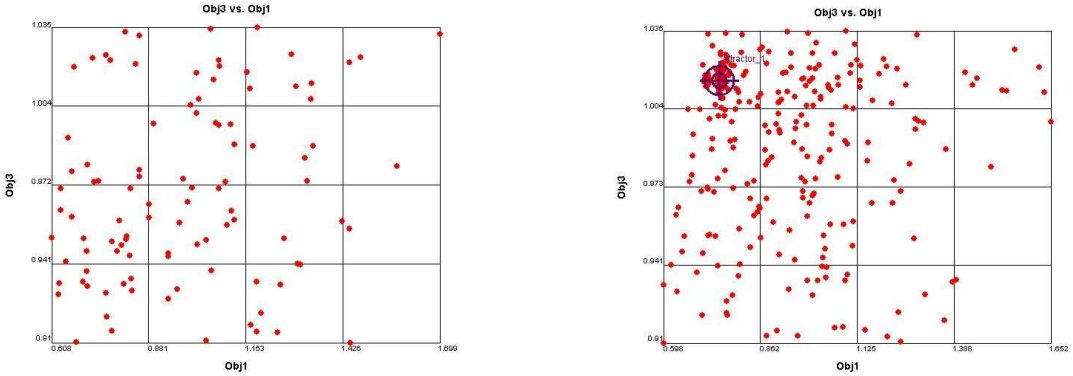
1) *Design space samplers* are used to populate the trade space and are typically invoked if there is no initial data available. The user can sample the design space manually using slider bar controls for each input dimension or randomly. When sampling randomly, the user specifies the number of samples to be generated and the bounds of the multi-dimensional hypercube of *X*. Monte Carlo sampling then randomly samples the inputs – drawing from a uniform, normal, or triangular distribution – and executes the simulation model, storing the corresponding output in the database. The bounds of the design variables can be reduced at any point to bias the samples in a given region if desired. An example is shown in Figure 2 where the region of interest is for the two design variables (A, B) $\leq$ 0.5.



(a) 100 initial samples (b) 100 new samplers in a reduced region of interest

Figure 2. Example of design space sampler.

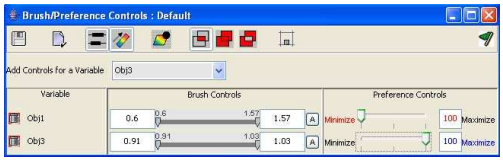
2) *Point samplers*, also referred to as *attractors*, are used to generate new sample points near a user-specified location in the trade space. The attractor is specified in the ATSV interface with a graphical icon that identifies an *n*-dimensional point in the trade space, and then new sample points are generated near the attractor – or as close as they can get to it. Unbeknownst to the user, the attractor generates new points using the Differential Evolution (DE) algorithm,[20] which assess the fitness of each new sample based on the normalized Euclidean distance to the attractor. As the population evolves in DE, the samples get closer and closer to the attractor. An example is shown in Figure 3 where the user specifies an attractor to fill in a "gap" in the trade space (see Figure 3(a)). The new samples cluster tightly around Attractor_1 as seen in Figure 3(b).



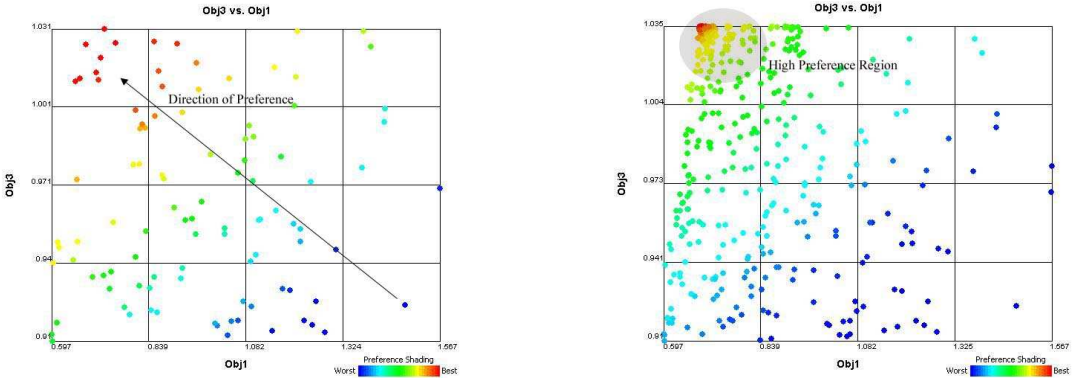
(a) 100 initial samples (b) New samples generated near attractor

Figure 3. Example of point sampler using an attractor.

3) *Preference samplers* allow users to populate the trade space in regions that perform well with respect to a user-defined preference function. New sample points are generated again by using the DE algorithm, but the fitness of each sample is now defined by the user's preference instead of the Euclidean distance. An example of the preference sampler is shown in Figure 4. Using ATSV's brushing and preference controls, the user specifies a desire to minimize Obj1 and maximize Obj3 with equal weighting (see Figure 4(a)). We currently employ a linear weighted sum of the user's stated preferences, but other preference functions could be implemented just as easily. Figure 4(b) shows the initial samples shaded based on preference, and Figure 4(c) shows the new samples, where the concentration of points increases in the direction of preference, namely, the upper left hand corner of the plot.



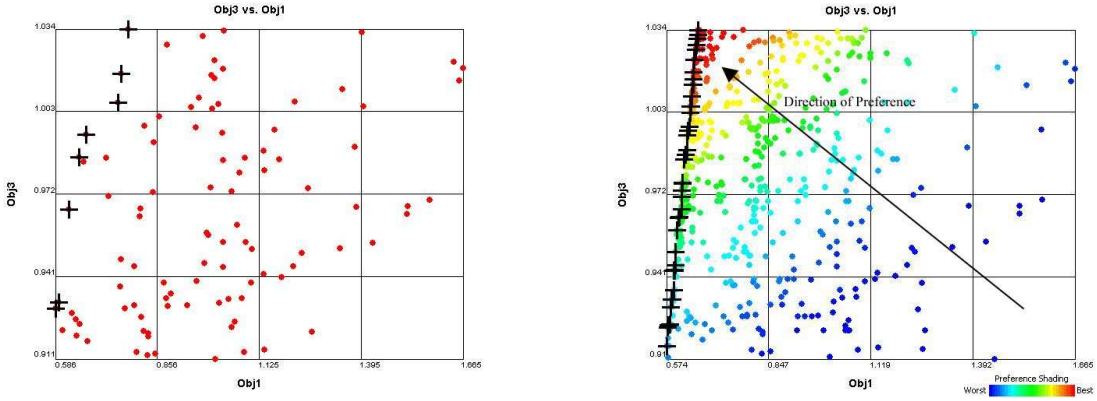
(a) Brush/Preference control settings used to indicate a user's preferences



(b) Initial samples shaded based on preference (c) New samples generated in the direction of preference

Figure 4. Example of Brush/Preference controls and preference sampler.

4) *Pareto samplers* are used to bias the sampling of new designs in search of the Pareto front once the user has defined his/her preferences on the objectives. The Pareto sampler uses the Pareto Differential Evolution algorithm developed by Madavan,[21] which differs from DE in terms of how selection is performed. In particular, a non-dominated sorting procedure is implemented at the end of a generation to select the best *NP* individuals from the pool of parents and children – DE only competes children against their own parent vectors.[20] An example of this sampler is shown in Figure 5. Using the same preference as before (i.e., minimize Obj1 and maximize Obj3 with equal weighting), Figure 5(a) shows the Pareto points in the initial samples while Figure 5(b) shows the Pareto front after executing 7 generations of the DE with a population size of 25 points. The points are also shaded to indicate the region of high (red) and low (blue) preference along the Pareto front.



(a) Initial samples (Pareto points denoted by +) (b) New samples generated along Pareto front

Figure 5. Example of Pareto sampler.

In their current implementation, none of these visual steering commands explicitly considers constraints when exploring the trade space unless they are hard-coded into the underlying simulation model used for analysis. This is because in our approach to trade space exploration, we advocate exploring the entire trade space first, feasible and infeasible, and then "brushing out" (i.e., filtering) infeasible designs using the Brush/Preference controls in ATSV.[15] This approach works well since many of the actual constraint limits are imposed subjectively by the designers based on their experience; however, in tightly coupled or highly constrained systems, there may only be a narrow band of feasible designs, and exploring the entire trade space may yield many solutions that are truly infeasible. Therefore, having the capability to enforce constraints during the exploration process may increase its efficiency and effectiveness, i.e., allow designers to find better designs with fewer function evaluations. Consequently, the objective in this study is to investigate the impact of constraint handling on the trade space exploration process when using these visual steering commands. The wing design problem used for this study is described next. The constraint handling methods and experimental set-up are described in Section IV along with the results.

III. Wing Design Problem

The wing design problem was developed by Simpson and Meckesheimer[22] and involves sizing the wing of an aircraft to minimize its cost subject to constraints on range, buffet altitude, and takeoff field length. Six design variables (inputs) are used to size the wing:

1) Semi-span, x_1
2) Aspect ratio, x_2
3) Sweep angle of quarter chord, x_3
4) Taper ratio, x_4
5) Sparbox root chord, YCoff, x_5
6) Fan diameter, x_6

The aspect ratio, taper ratio, and semi-span affect the overall wing area and geometry as shown in Figure 6, while the sweep angle defines the leading edge sweep of the wing at the quarter chord. The sparbox root chord, YCoff, defines the width of the sparbox at the wing centerline as shown in the figure. Finally, the fan

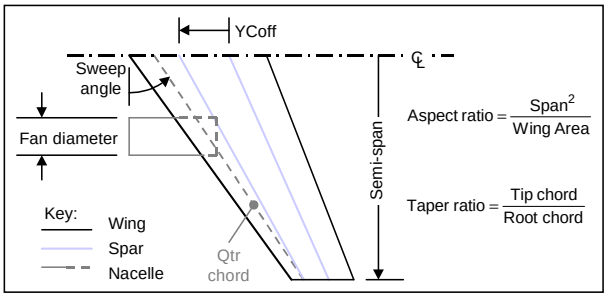


Figure 6. Definition of wing design variables.

diameter is used to scale the diameter of the nacelle mounted underneath the wing. All design variables are scaled to vary between 0 and 1 based on their lower and upper bounds, respectively.

Analysis for the wing design problem is achieved using second-order response surface models that relate the six design variables to the four performance variables (responses) of interest, namely, Cost, Range, Buffet Altitude, and Takeoff Field Length (TOFL).[22] Sample data to construct these response surfaces were obtained from 243-point orthogonal array, of which only 200 points provided feasible designs. The response surface models are constructed using ordinary least squares regression in the JMP® software, and with the exception of Buffet Altitude, they are full second-order models that include first-order, second-order, and two-factor interaction terms for all six design variables. The response surface model for Buffet Altitude does not include Fan Diameter, x_6, since this variable does not have a significant effect; hence, this response surface model is a full second-order model of only the remaining five design variables. The resulting response surface equations for Cost, Range, TOFL, and Buffet Altitude are included in the Appendix. We note that these responses have been normalized to range between 0 and 1 based on the minimum and maximum observed values, respectively.

The original optimization problem includes only a single objective, Cost, which is to be minimized, with constraints on the remaining performance variables.[22] We modified the optimization problem for this study to make it multi-objective by minimizing Cost as well as maximizing Range as follows:

$$\begin{array}{ll} \textit{Minimize:} & \text{Cost} \\ \textit{Maximize:} & \text{Range} \\ \textit{Subject to:} & \text{Range} \geq 0.589 \\ & \text{Buffet altitude} \geq 0.603 \\ & \text{TOFL} \leq 0.377 \end{array} \quad (1)$$

The wing design problem is especially well suited for this study. Preliminary experiments with expert users found that the method for constraint display significantly impacted a user's search process – those that saw fewer constraints in the graphical display felt that they had greater freedom to explore the trade space and ultimately found better solutions given the tightly constrained nature of the problem.[22] The next section describes the experimental set-up for this study, including the different constraint handling methods that are compared, along with the results.

IV. Experimental Study

The objective in this study is to investigate the impact of constraint handling on the trade space exploration process. This is done within the context of the wing design problem and ATSV; however, the procedures and results are applicable to problems of a similar nature and any trade space exploration process, regardless of the software.

A. Experimental Set-Up

For this study, we evaluate three methods for handling constraints when using the visual steering commands within ATSV: (1) no constraint handling, (2) manual constraint handling, and (3) automatic constraint handling. For the no constraint handling case, the user ignores the predefined constraints specified for the wing design problem, simulating the situation in which the only *a priori* knowledge about the problem is the desire to minimize Cost and maximize Range. Constraints are only applied at the end of the trade space exploration process in order to select the best designs that satisfy Eq. (1). In the manual constraint handling case, the user treats the constraints as objectives when exploring the design space by specifying directions of preference that will ensure the constraints are satisfied. In this case, the user specifies not only to minimize Cost and maximize Range when exploring the trade space but also to maximize Buffet Altitude and minimize Take-off Field Length. By maximizing Buffet Altitude (and Range) and minimizing Take-off Field Length, the exploration process should gravitate toward regions in the trade space that will be feasible after the constraint limits are imposed (via the brush controls in ATSV).

In the automated constraint handling case, the user specifies preferences and constraints prior to implementing any of the visual steering commands such that the constraints are explicitly (i.e., automatically) handled during the exploration process. This is achieved by integrating the constraint domination work of Deb et al.,[23] into the DE and Pareto DE algorithms used in ATSV. In particular, the domination of two solutions *i* and *j* is modified to be:

A solution *i* constraint-dominates solution *j* if any of the following conditions are true:
1) *i* is feasible and *j* is infeasible: *i* constraint-dominates *j*
2) *i* and j are feasible: if *i* Pareto dominates *j*, then *i* constraint-dominates *j*
3) *i* and *j* are infeasible: if *i* dominates *j* in the constraint-violation space, then *i* constraint-dominates *j*

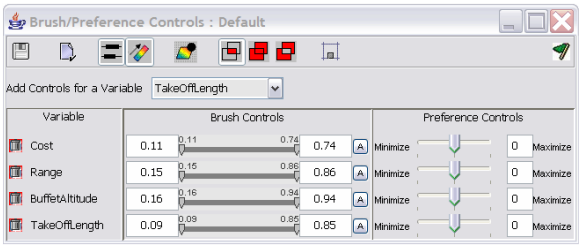
Any ties for dominance are broken by the crowding diversity metric implemented by Deb et al.[23] and used by Madavan.[21] Constraint limits are obtained directly from the user-specified settings in the brush controls in ATSV (see Figure 4(a) for an example). Constraint dominance is implemented within all three DE-based samplers – point, preference, and Pareto samplers – in the automated constraint handling case with the goal of increasing the number of feasible points obtained when using visual steering commands in ATSV.

To compare the performance of these three constraint handling methods, we implement four sampler trials for each constraint handling method based on the visual steering commands available in ATSV:

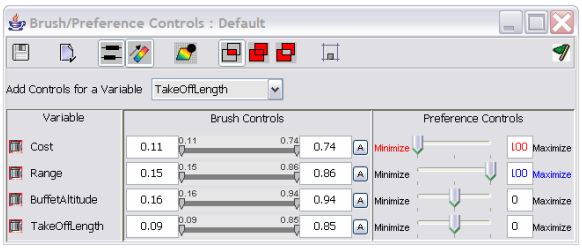
1) Basic (design) sampler – provides a "baseline" for comparison by randomly searching the design space.
2) Point (attractor) sampler – placed at various locations within the trade space to generate new design points.
3) Preference sampler – used to drive the exploration process in the direction of preference.
4) Pareto sampler – used to generate points along the Pareto front.

Each sampler trial is allocated roughly 10,000 function evaluations, i.e., the user is limited to generating 10,000 points with a given sampler. The 10,000 point limit was used so that enough points were generated to enable comparisons between the sets of sampler trials across all three constraint handling methods, yet did not allow for an exhaustive search in which all of the resulting Pareto fronts would be found. Each sampler trial was performed three times (i.e., each trial had three versions denoted v1, v2, v3) to ensure that the results obtained from the sampler were reliable and not due to the inherent randomness in the DE algorithm.

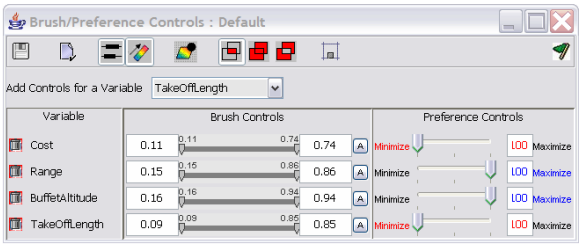
The implementation details for each sampler trial are summarized in Tables A.1 - A.3, respectively, for the no constraint handling, manual constraint handling, and automated constraint handling methods. Figure 7(a) shows the Brush/Preference controls for the wing design problem prior to any user-defined settings. The settings for the Pareto and preference sampler trials are shown in Figure 7(b) for the no constraint handling case. The Brush/Preference controls for the preference and Pareto samplers for the manual constraint handling case are shown in Figure 7(c). At the end of each trial the settings shown in Figure 7(d) were applied so that only feasible Pareto points were obtained. For the automatic constraint handling case, the Brush/Preference controls were also set as shown in Figure 7(d). Since the basic sampler randomly searches the design space, it is not affected by the Brush/Preference controls regardless of how constraints are handled; the user simply provides the settings in Figure 7(d) to obtain the feasible Pareto points after 10,000 points are generated. Likewise the point (attractor) sampler is specified directly in the trade space and is only impacted by the Brush/Preference controls in the automated constraint handling case when the underlying DE is modified to include constraint-dominance during the exploration process.



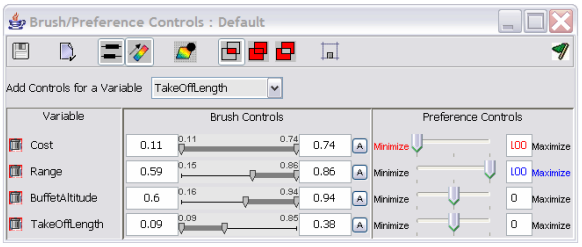
(a) No constraints specified



(b) Settings for preference and Pareto samplers without constraint handling



(c) Settings for manual constraint handling trials



(d) Settings for automatic constraint handling trials and to obtain final feasible Pareto points in all other trials

Figure 7. Brush/Preference controls for all trials.

We compare the 36 sets of results (= 3 constraint handling methods x 4 trials x 3 versions of each trial) both graphically and quantitatively. The graphical analysis entails overlaying the feasible Pareto fronts from each version of each trial in different colors for each constraint handling method to visualize the differences in the resulting Pareto fronts. The quantitative analysis involves computing the following metrics for each set of results:

1) Percentage of feasible points in the final set
2) Number of Pareto points in the final set (Note: points must be feasible in order to be considered Pareto)
3) Percentage of the non-dominated points in the combined Pareto front (Note: the combined Pareto front is obtained by combining the Pareto points from the solution sets from a set of four trials and then removing dominated points).

The first metric assesses the efficiency of the exploration process: a small percentage of feasible points indicates that many function evaluations were "wasted" generating infeasible designs while a high percentage of feasible points indicates a more efficient use of function evaluations. The second metric measures the number of best points that were found, i.e., feasible non-dominated points that satisfy Eq. (1). This metric provides a means of assessing the relative effectiveness of the exploration process for a given trial. Finally, the third metric compares the effectiveness of a given constraint handling method across all four trials by indicating which approach yielded the most non-dominated points after 10,000 function evaluations.

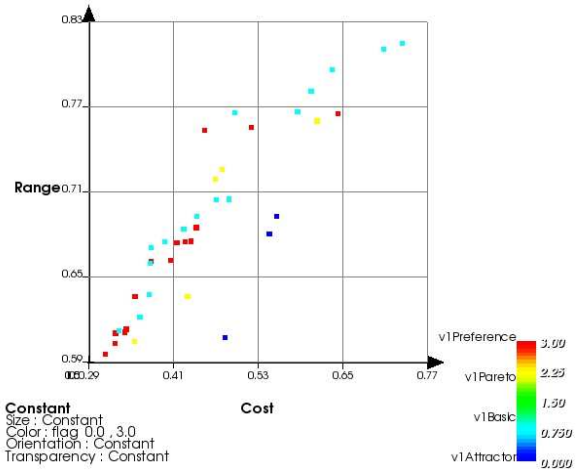
B. Experimental Results

The resulting feasible Pareto fronts for the no constraint handling, manual constraint handling, and automatic constraint handling methods are shown graphically in Figure 8, Figure 9, and Figure 10, respectively. The plots within each figure are sorted by version (i.e., v1, v2, v3) of the sampler trial and color-coded to show how the resulting Pareto fronts compare to one another. In all of these plots, the light blue points are the feasible Pareto points obtained from the basic sampler, the dark blue points are the feasible Pareto points obtained from the point (attractor) sampler, the yellow points are the feasible Pareto points obtained from the Pareto sampler, and the red points are the feasible Pareto points obtained from the preference sampler. The combined Pareto front is shown on the right-hand side of each figure, where the non-dominated solutions are denoted by '+' and the same color coding is used to indicate from which sampler trial the non-dominated point were obtained.

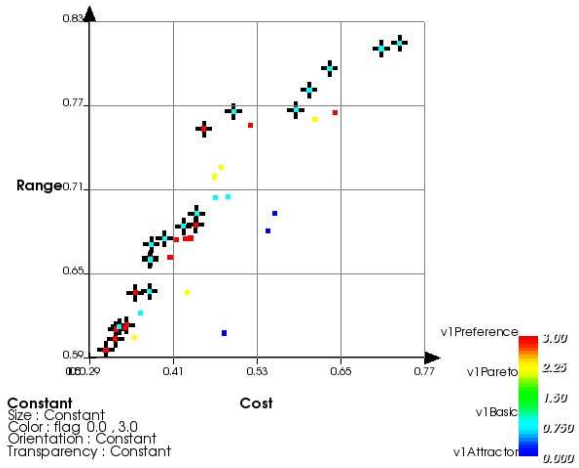
The differences between the three constraint handling methods are readily apparent based on visual comparison of the plots in each of these figures. First, by comparing left-hand plots (a), (c), and (e) in each figure, we note similarities among the resulting Pareto fronts from each version (v1, v2, v3) of each trial that uses a different sampler. This indicates that the results obtained are reliable and achieved as a result of the methods used, not due to the inherent randomness in the underlying DE algorithm or its creation of initial seeds.

Next, we note a strong correlation between constraint handling method and trial in terms of which sampler is the most effective and most ineffective. In Figure 8, the basic sampler trials yield the most Pareto points in all three versions (see Figure 8(a), Figure 8(c), and Figure 8(e)) of the no constraint handling case, and they contribute the majority of the Pareto points to the combined Pareto fronts as seen in Figure 8(b), Figure 8(d), and Figure 8(f). The attractor performs the worst in the no constraint handling case, but it is the most effective sampler in the manual constraint handling cases shown in Figure 9. The attractor sampler contributes the majority of the Pareto points to the combined Pareto fronts shown in Figure 9(b), Figure 9(d), and Figure 9(f) while the preference sampler yields the worst points in the manual constraint handling case. Meanwhile, in the automatic constraint handling cases in Figure 10, the preference and Pareto samplers are the most effective. As seen in Figure 10(b), Figure 10(d), and Figure 10(f), the preference sampler tends to contribute the points in the low Cost region to the combined Pareto front while the Pareto sampler favors designs the high Range region. Finally, we note that the basic sampler performs the worst in the automatic constraint handling case, yielding points that are dominated by nearly all of the other samplers.

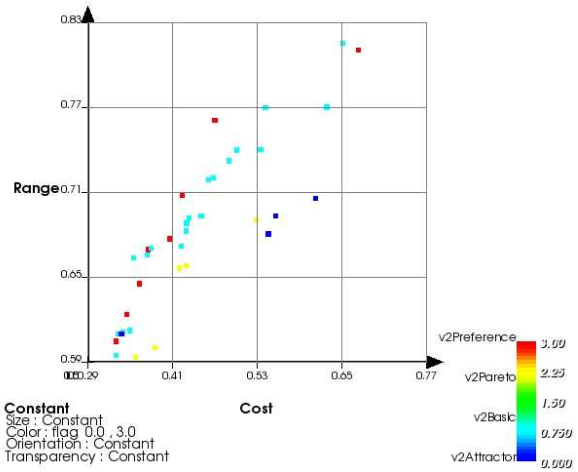
The quantitative assessment of each trial's performance confirms these qualitative findings. Table 1 summarizes the results for each version of each sampler trial for the three constraint handling methods using the aforementioned metrics, namely, percentage of feasible points, number of feasible Pareto points, and percentage of non-dominated points in the combined Pareto front. In the no constraint handling case, the basic sampler consistently yields the highest percentage of feasible points, finds the most Pareto points, and contributes 61-89% of the points (70% on average) to the combined Pareto front. The preference sampler contributes the next highest percentage of non-dominated points, while the attractor and Pareto samplers offer less than 2% of the non-dominated points in the combined front; however, on average 1% or less of the points from the preference, Pareto, and attractor samplers are feasible – conversely, 99% of the function evaluations used in these trials yield infeasible points, which is a highly inefficient way of exploring the trade space. In the manual constraint case, the attractor trials outperform the other samplers, returning an average of nearly 30% feasible points and contributing more than 87% of the non-dominated points to the combined Pareto front, on average. The Pareto sampler does slightly better than in the no constraint handling case, and the basic sampler trials yield comparable results to the Pareto sampler. The preference sampler performs the worst by far, offering zero non-dominated points to the combined Pareto front in all three versions.



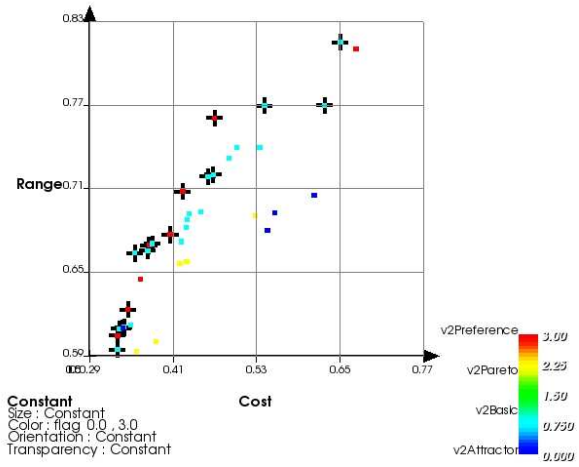
(a) No constraint handling trials (v1)



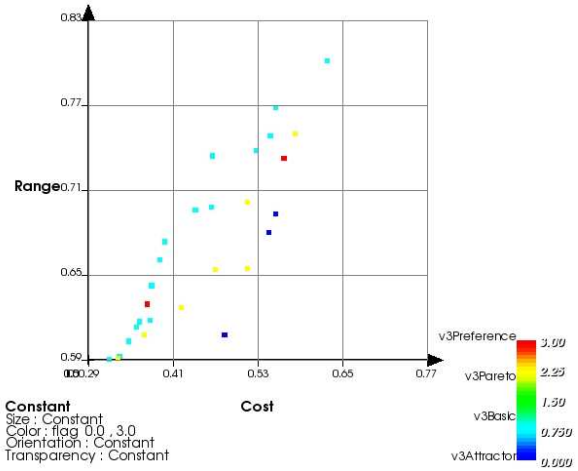
(b) Combined Pareto front v1 (Pareto points denoted by +)



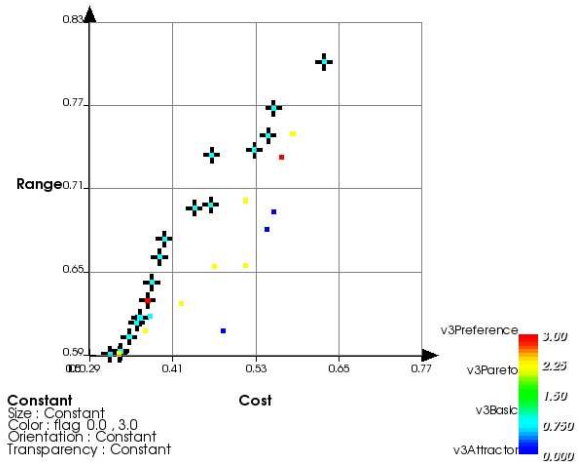
(c) No constraint handling trials (v2)



(d) Combined Pareto front v2 (Pareto points denoted by +)

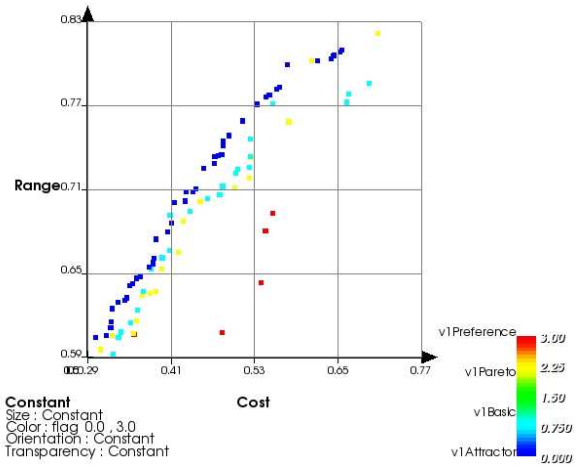


(e) No constraint handling trials (v3)

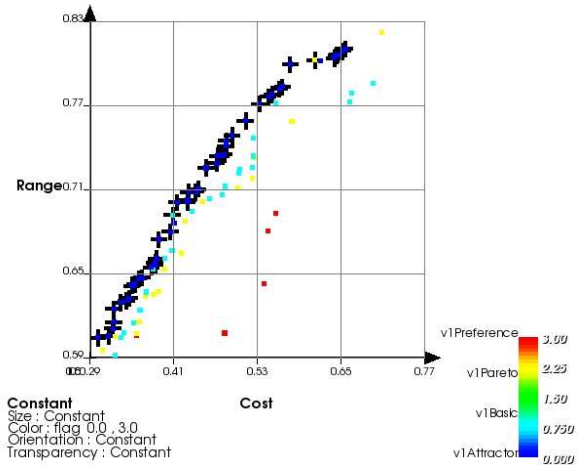


(f) Combined Pareto front v3 (Pareto points denoted by +)

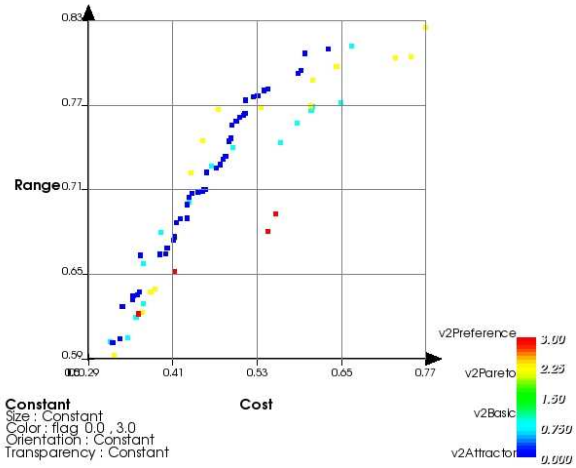
Figure 8. Individual and combined Pareto fronts obtained with no constraint handling.



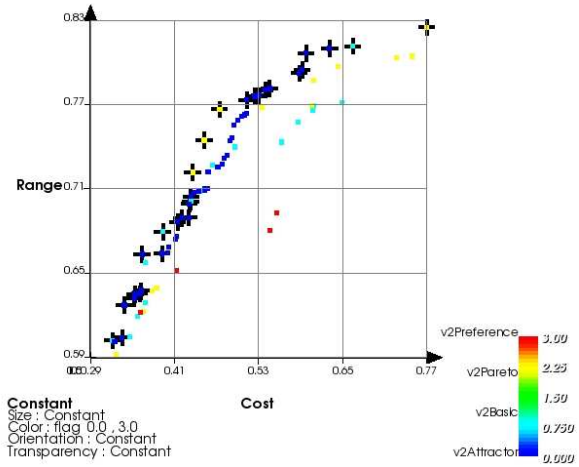
(a) Manual constraint handling trials (v1)



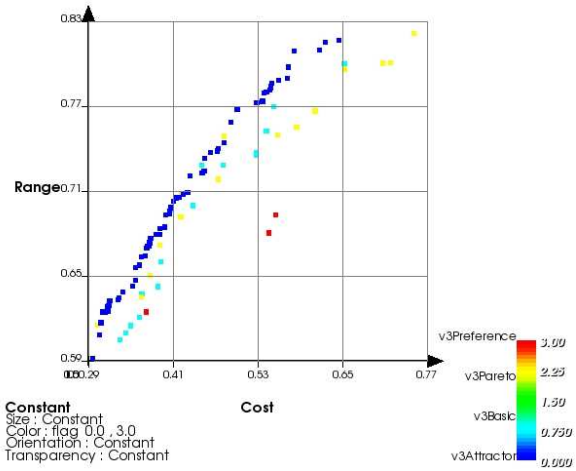
(b) Combined Pareto front v1 (Pareto points denoted by +)



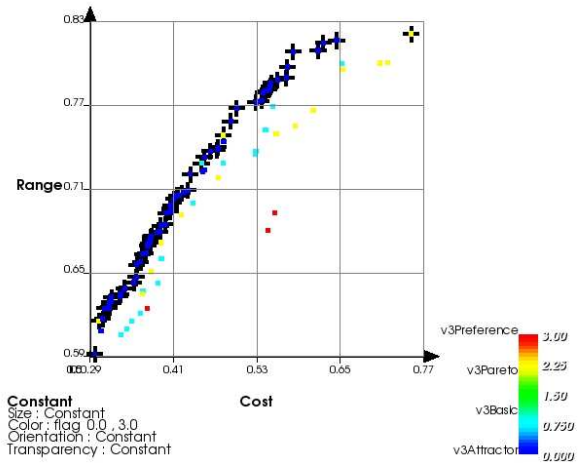
(c) Manual constraint handling trials (v2)



(d) Combined Pareto front v2 (Pareto points denoted by +)

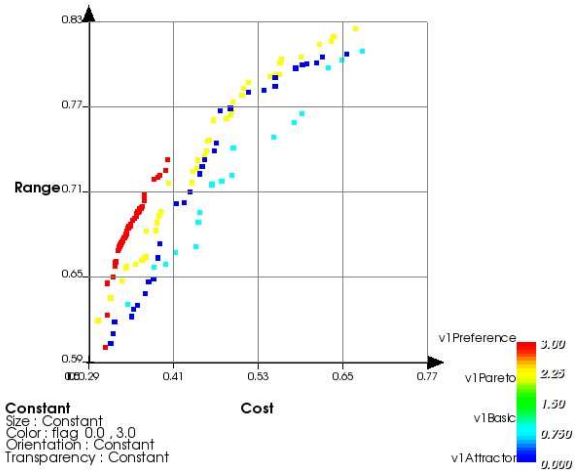


(e) Manual constraint handling trials (v3)

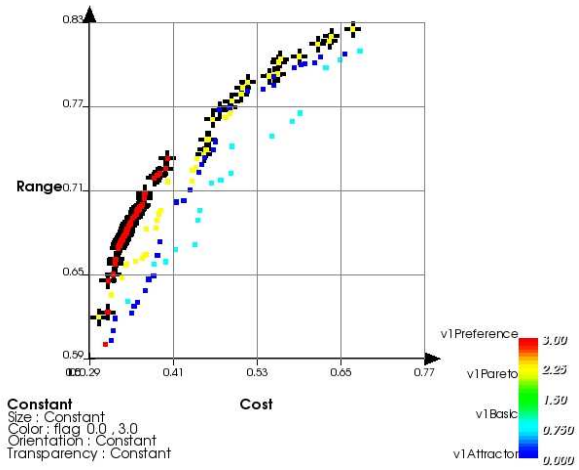


(f) Combined Pareto front v3 (Pareto points denoted by +)

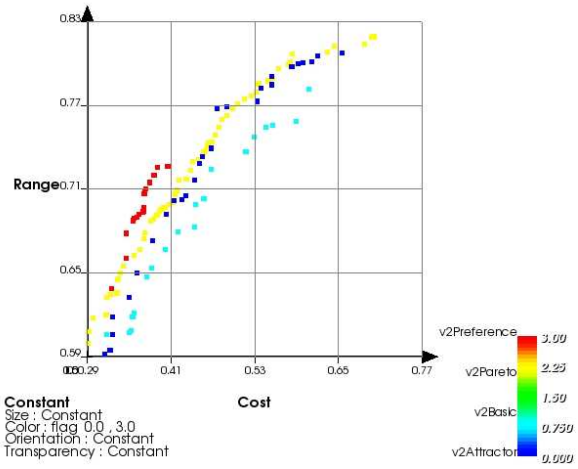
Figure 9. Individual and combined Pareto fronts from manual constraint handling trials.



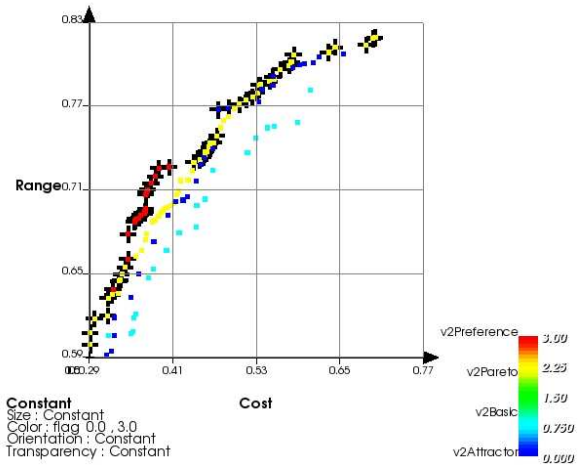
(a) Automatic constraint handling trials (v1)



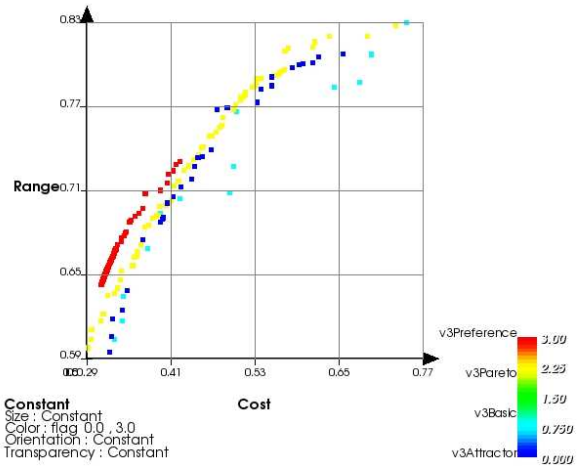
(b) Combined Pareto front v1 (Pareto points denoted by +)



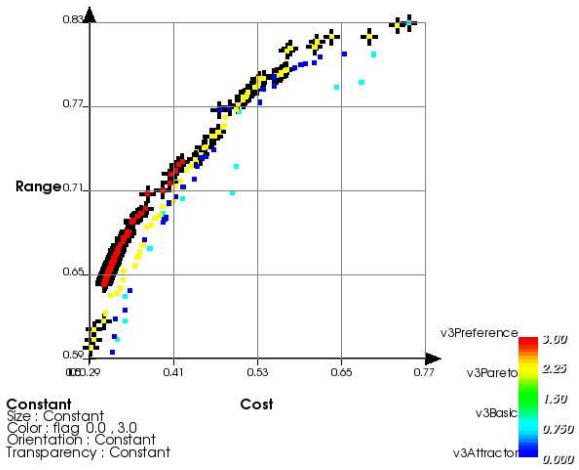
(c) Automatic constraint handling trials (v2)



(d) Combined Pareto front v2 (Pareto points denoted by +)



(e) Automatic constraint handling trials (v3)



(f) Combined Pareto front v3 (Pareto points denoted by +)

Figure 10. Individual and combined Pareto fronts from automatic constraint handling trials.

Table 1. Performance of each version of each sampler trial for each constraint handling method.

Metric	*No Constraint Handling*				*Manual Constraint Handling*				*Automatic Constraint Handling*			
	Basic	Attractor	Pareto	Preference	Basic	**Attractor**	Pareto	Preference	Basic	Attractor	**Pareto**	**Preference**
v1												
% Feasible	**6.22**	0.67	0.27	1.17	5.77	**28.42**	7.71	0.57	5.59	9.96	**37.46**	6.20
# Pareto Points	16	3	5	15	23	47	17	5	17	31	44	56
% Combined Pareto Front	**61.90**	0.00	0.00	38.10	2.13	**95.74**	2.13	0.00	0.00	2.53	27.85	**69.62**
Combined Front # Points	21				47				79			
v2												
% Feasible	**6.24**	0.68	0.15	1.33	5.97	**28.16**	6.16	1.63	6.31	9.35	**39.97**	4.34
# Pareto Points	20	4	5	8	16	44	16	4	19	27	63	20
% Combined Pareto Front	**61.11**	5.56	0.00	33.33	13.33	**73.33**	13.33	0.00	0.00	6.56	**60.66**	32.79
Combined Front # Points	18				30				61			
v3												
% Feasible	**5.63**	0.67	0.14	0.42	5.93	**33.38**	7.52	0.80	6.27	10.03	**43.02**	13.97
# Pareto Points	17	3	7	4	16	68	16	3	13	29	65	205
% Combined Pareto Front	**88.89**	0.00	5.56	5.56	1.47	**92.65**	5.88	0.00	0.41	1.22	14.69	**83.67**
Combined Front # Points	18				68				245			
Average of v1, v2, v3												
% Feasible	**6.03**	0.67	0.19	0.97	5.89	**29.99**	7.13	1	6.06	9.78	**40.15**	8.17
# Pareto Points[a]	18	3	6	9	18	53	16	4	16	29	57	94
% Combined Pareto Front	**70.63**	1.85	1.85	25.66	5.64	**87.24**	7.11	0	0.14	3.44	34.4	**62.03**
Combined Front # Points[a]	19				48				128			

[a] The average # of Pareto points and # points in the combined Pareto front are rounded to the nearest integer value.

In the automatic constraint handling case, the Pareto sampler tends to find the highest percentage of feasible points (40% on average). It contributes nearly 61% of the non-dominated points to the combined Pareto front in one of the trials (v2), but the preference sampler contributes the most non-dominated points to the combined Pareto front on average (62%). Surprisingly, the preference sampler tends to have a rather low percentage of feasible points (8% on average) while contributing the most non-dominated points to the combined Pareto front in two out of the three trials (v1 and v3). The attractor trials yield more feasible points on average (10%) but contribute less than 4% of the non-dominated points to the combined Pareto front. The basic sampler performs the worst in the automatic constraint handling case as noted earlier (see Figure 10). Finally, we note that the automatic constraint handling method leads to the highest number of non-dominated points in the combined Pareto front on average across trials.

To understand why the attractor, preference, and Pareto samplers perform so poorly in the no constraint handling case, consider Figure 11, which shows the Pareto fronts for a set of points (not from an actual trial). The infeasible points are shown in gray, and the feasible points are shown in red – these red points are the points that remain after applying the Brush/Preference control settings shown in Figure 7(d). Before these controls are set to reveal the feasible region, the user would be inclined to place attractors along the infeasible region's Pareto front since it dominates what is the resultant feasible region. Due to the user's bias to explore in this infeasible region, the only points generated in the feasible region come from random points generated by the DE algorithm; hence, the low percentage of feasible points. The same logic applies to the Pareto and preference samplers, which would bias their sampling along the infeasible Pareto front when no constraint information is present, which also explains the low percentage of feasible points. Since the basic sampler is sampling uniformly across the entire trade space, it outperforms the other samplers by chance.

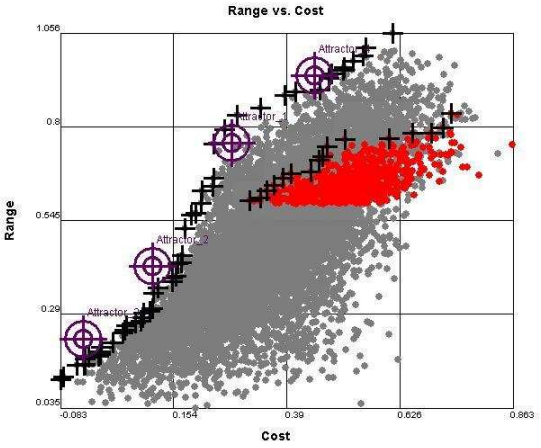
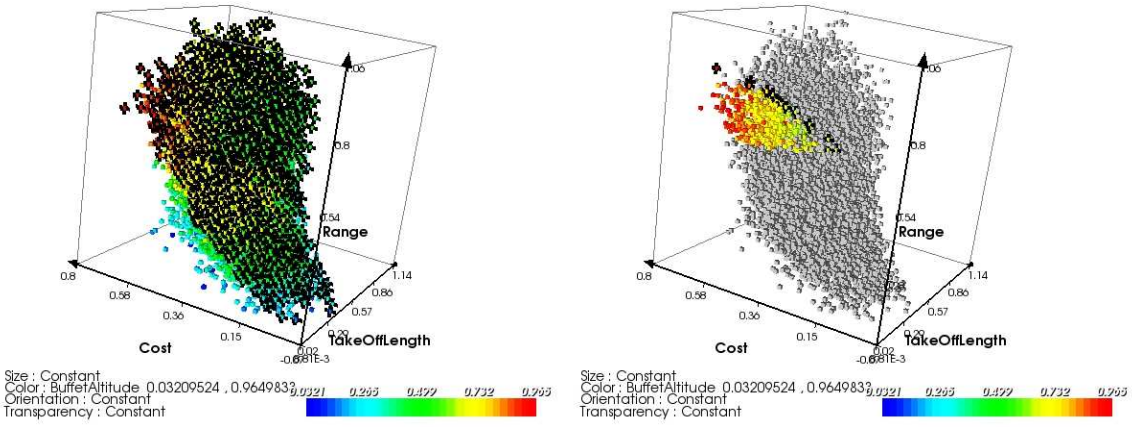


Figure 11. Infeasible (gray) and feasible (red) Pareto fronts (+ denotes Pareto points).

In the manual constraint handling case, the basic sampler is unable to compete – it generates about the same percentage of feasible points and the same number of Pareto points as before, but the attractors excel in this case due to their placement in the trade space, i.e., they can be placed to minimize Cost and maximize Range while being just beyond the constraint limits for Range, TOFL, and Buffet Altitude. The Pareto and preference samplers, on the other hand, suffer from the manner in which the constraints were handled manually, i.e., they "overshot" the desired region when minimizing TOFL and maximizing Buffet Altitude were added as separate objectives (instead of specifying them as constraints) to bias the sampling towards feasible designs. Figure 12(a) shows the corresponding Pareto front for the Brush/Preference control settings for the manual constraint handling case (see Figure 7(c)). The Pareto and preference samplers tried to generate points along this new four-dimensional front even though the desired Pareto front is located on the edge of the small colored region in Figure 12(b). As a result, many evaluations were wasted exploring the infeasible space as indicated by the multitude of gray points in Figure 12(b). This figure also provides an accurate representation of the size of the feasible region compared to the entire trade space.



(a) Pareto front of entire trade space (b) Pareto front of feasible space (gray points are infeasbile)

Figure 12. Representation of Pareto front for manual constraint handling trials.

In the automatic constraint handling case, the Pareto and preference samplers appear to have performed well for different reasons. The Pareto sampler generated a much higher percentage of feasible points once it took the constraints into account – as the underlying DE population evolved, it found more and more feasible points based on

the constraint-dominance concept that we implemented. The same goes for the preference sampler; however, the preference sampler concentrated its search in one direction, namely, the direction of preference, whereas the Pareto sampler tried to find the entire Pareto front. As the search populations evolved, the preference sample outperformed the Pareto sampler since it was essentially searching along a line instead of along a two-dimensional front, leading to a higher percentage of non-dominated points, on average, for the preference sampler in the automatic constraint handling case. This was seen earlier in Figure 10, which shows how the points from the preference sampler are confined to a small area along the front. Meanwhile, the basic sampler continued to randomly explore the design space and performed about the same as before, verifying its consistency in how it was intended to operate. The attractor samplers performed better than in the no constraint handling case, but they were not as effective in this case as they were in the manual constraint handling case since the user specified locations only in the Range-Cost trade space (see Table A.3) not the entire four-dimensional trade space (see Table A.2).

V. Closing Remarks and Future Work

Trade space exploration is a powerful alternative to optimization-based approaches for formulating, adjusting, and ultimately solving multi-objective design problems. It provides a visual and more intuitive means for designers to explore the trade space by integrating multi-dimensional data visualization and visual steering commands. As part of our ongoing investigations into the trade space exploration process, we examine the impact of constraint handling on the exploration process using a wing design example in this paper. In particular, we examined three constraint handling methods – no constraint handling, manual constraint handling, and automatic constraint handling – and evaluated their impact on the efficiency and effectiveness of the four samplers available in the ATSV, namely, the basic, point (attractor), preference, and Pareto samplers. Graphical and quantitative analysis determined that the performance of each sampler trial was highly correlated with the constraint handling method: the basic sampler performed best for the no constraint handling case, the point (attractor) samplers performed best in the manual constraint handling case, and the preference and Pareto samplers performed best in the automatic constraint handling case with the Pareto sampler being more efficient (based on the percentage of feasible points) and the preference sampler being more effective (based on the percentage of non-dominated points in the combined Pareto front).

While initially surprising, the results are consistent with the user's *a priori* knowledge about the constraints, which is reflected in how constraints are handled in each method. The implications of these findings can be summarized as follows. If users are uncertain about the constraint limits, then the basic sampler should be used to randomly search the trade space – this is equivalent to the no constraint handling case wherein the basic sampler offers consistent performance in terms of efficiency and effectiveness. Meanwhile, if constraint limits are known and the user wants to explore directly along these boundaries to explore specific tradeoffs, then attractors should be used since they can be placed at specific locations in the trade space. This is equivalent to the manual constraint handling case, wherein users may have some *a priori* knowledge about the constraints, but they want the flexibility to "push the boundaries" when looking for the best designs. Finally, if constraint limits are fairly certain or if the problem is highly constrained with a small region of feasibility, then the Pareto and preference samplers are the best choice. This is equivalent to the automatic constraint handling case, and users should select the preference sampler for maximum effectiveness when their preferences (i.e., important weighting for each objective) are known *a priori* and with a high degree of certainty. If the importance weights are not know with a high degree of certainty, then the Pareto sampler should be used due to maximize efficiency while also finding good designs.

This study is part of ongoing work to provide empirical evidence and quantify the benefits of putting designers back "in-the-loop" during the design optimization process. Immediate extensions to this particular study include comparing the resulting Pareto fronts in each case against a "reference set" to determine how many of the non-dominated points are in the actual Pareto front and using ε-dominance[24] or s-Pareto[25] filtering techniques to obtain a better representation of the number of different Pareto points found in each case. Finally, similar tests should be conducted on problems of varying size and complexity, ranging from unconstrained to highly constrained in nature, to ensure that the results are indeed applicable to a wide range of design problems.

Appendix

The following equations are the second-order response surface models for the wing design problem.[22]

$$\begin{aligned}\text{Cost} = {} & 0.2854 - 0.005*x_6 + 0.3109*x_5 - 0.0122*x_3 - 0.2095*x_4 - 0.4836*x_2 + 0.4431*x_1 + 0.1037*x_6*x_6 - 0.0592*x_5*x_5 \\ & - 0.0204*x_3*x_3 + 0.1057*x_4*x_4 + 0.2494*x_2*x_2 + 0.0218*x_1*x_1 + 0.0581*x_6*x_5 + 0.0025*x_6*x_3 + 0.0034*x_6*x_4 \\ & + 0.0502*x_6*x_2 - 0.0326*x_6*x_1 + 0.1254*x_5*x_3 - 0.1362*x_5*x_4 + 0.1664*x_5*x_2 - 0.4223*x_5*x_1 + 0.1039*x_3*x_4 \\ & - 0.0155*x_3*x_2 - 0.0735*x_3*x_1 - 0.1281*x_4*x_2 + 0.2183*x_4*x_1 - 0.2109*x_2*x_1\end{aligned} \quad \text{(A.1)}$$

$$\text{Range} = 0.3576 - 0.0329*x_6 + 0.1978*x_5 + 0.0149*x_3 - 0.0389*x_4 - 0.4652*x_2 + 0.4453*x_1 + 0.0149*x_6*x_6 - 0.051*x_5*x_5 + 0.0075*x_3*x_3 - 0.0229*x_4*x_4 + 0.0987*x_2*x_2 - 0.0188*x_1*x_1 - 0.0524*x_6*x_5 - 0.0272*x_6*x_3 + 0.0281*x_6*x_4 - 0.0147*x_6*x_2 + 0.0083*x_6*x_1 + 0.1018*x_5*x_3 + 0.0563*x_5*x_4 - 0.0349*x_5*x_2 + 0.064*x_5*x_1 + 0.0073*x_3*x_4 + 0.0176*x_3*x_2 + 0.0341*x_3*x_1 + 0.1063*x_4*x_2 - 0.0374*x_4*x_1 + 0.0143*x_2*x_1 \quad (A.2)$$

$$\text{TOFL} = 0.2884 - 0.2896*x_6 + 0.3376*x_5 + 0.0088*x_3 - 0.0478*x_4 - 0.1448*x_2 + 0.1239*x_1 + 0.0714*x_6*x_6 - 0.029*x_5*x_5 + 0.0148*x_3*x_3 + 0.0068*x_4*x_4 + 0.2251*x_2*x_2 + 0.1654*x_1*x_1 - 0.12*x_6*x_5 - 0.0475*x_6*x_3 + 0.0426*x_6*x_4 - 0.0486*x_6*x_2 - 0.1058*x_6*x_1 + 0.1712*x_5*x_3 + 0.0071*x_5*x_4 - 0.0887*x_5*x_2 + 0.0759*x_5*x_1 + 0.0028*x_3*x_4 - 0.0056*x_3*x_2 + 0.064*x_3*x_1 + 0.0063*x_4*x_2 + 0.0456*x_4*x_1 - 0.2902*x_2*x_1 \quad (A.3)$$

$$\text{Buffet altitude} = 0.617 - 0.1221*x_5 - 0.0485*x_3 + 0.0141*x_4 - 0.4507*x_2 + 0.6968*x_1 + 0.0248*x_5*x_5 + 0.0277*x_3*x_3 + 0.011*x_4*x_4 - 0.0873*x_2*x_2 - 0.295*x_1*x_1 - 0.061*x_5*x_3 - 0.0789*x_5*x_4 + 0.0546*x_5*x_2 - 0.1674*x_5*x_1 - 0.0008*x_3*x_4 + 0.0422*x_3*x_2 - 0.0371*x_3*x_1 + 0.017*x_4*x_2 - 0.0507*x_4*x_1 + 0.2845*x_2*x_1 \quad (A.4)$$

The following tables summarize the implementation details for each sampler trial for each constraint handling method.

Table A.1. Description of sampler trials with no constraint handling.

Basic (Total Points: 10,000)	**Pareto (Total Points: 10,080)**
- Basic sampler: 10,000 runs - Brush and preference controls (to obtain feasible and Pareto) data: - Minimize (-100) Cost - Maximize (100) Range ≥ 0.589 - BuffetAltitude ≥ 0.603 - TOFL ≤ 0.377	- Generation size set at 60 - Brush and preference controls: - Maximize (100) Range - Minimize (-100) Cost - Pareto sampler: 10,000 runs - Brush and preference controls (to obtain feasible and Pareto) data: - Minimize (-100) Cost - Maximize (100) Range ≥ 0.589 - BuffetAltitude ≥ 0.603 - TOFL ≤ 0.377
Preference (Total Points: 10,080)	**Attractor (Total Points: 10,203)**
- Generation size set at 60 - Brush and preference controls: - Maximize (100) Range - Minimize (-100) Cost - Preference sampler: Looped until 10,000 or more runs are reached - Brush and preference controls (to obtain feasible and Pareto) data: - Minimize (-100) Cost - Maximize (100) Range ≥ 0.589 - BuffetAltitude ≥ 0.603 - TOFL ≤ 0.377	- Basic Sampler: 25 runs - Generation size changed to 60 - Point Attractor: # Runs set to 600 - [Cost = 0.165, Range = 0.875] - [Cost = 0.087, Range = 0.643] - [Cost = 0.100, Range = 0.435] - [Cost = 0.165, Range = 0.524] - [Cost = 0.326, Range = 0.919] - [Cost = 0.228, Range = 0.732] - [Cost = 0.029, Range = 0.242] - [Cost = 0.040, Range = 0.362] - [Cost = 0.263, Range = 0.943] - [Cost = 0.383, Range = 0.995] - [Cost = 0.170, Range = 0.654] - [Cost = 0.047, Range = 0.446] - [Cost = 0.162, Range = 0.824] - [Cost = 0.188, Range = 0.747] - [Cost = 0.213, Range = 0.803] - [Cost = 0.143, Range = 0.530] - [Cost = 0.014, Range = 0.294] - Brush and preference controls (to obtain feasible and Pareto) data: - Minimize (-100) Cost - Maximize (100) Range ≥ 0.589 - BuffetAltitude ≥ 0.603 - TOFL ≤ 0.377

Table A.2. Description of manual constraint handling sampler trials.

Basic (Total Points: 10,000)	Pareto (Total Points: 10,080)

- Basic sampler: 10,000 runs
- Brush and preference controls (to obtain feasible and Pareto) data:
  - Minimize (-100) Cost
  - Maximize (100) Range ≥ 0.589
  - BuffetAltitude ≥ 0.603
  - TOFL ≤ 0.377
	- Generation size changed to 60
	- Brush and preference controls:
	  - Maximize Range
	  - Minimize Cost
	  - Minimize TOFL
	  - Maximize BuffetAltitude
	- Pareto sampler: 10,000 runs
	- Brush and preference controls (to obtain feasible and Pareto) data:
	  - Minimize (-100) Cost
	  - Maximize (100) Range ≥ 0.589
	  - BuffetAltitude ≥ 0.603
	  - TOFL ≤ 0.377

Preference (Total Points: 10,446)	**Attractor (Total Points: 10,335)**

- Generation size changed to 60
- Brush and preference controls:
  - Maximize (100) Range
  - Minimize (-100) Cost
  - Minimize (-100) TOFL
  - Maximize (100) BuffetAltitude
- Preference sampler: Looped until 10,000 or more runs are reached
- Brush and preference controls (to obtain feasible and Pareto) data:
  - Minimize (-100) Cost
  - Maximize (100) Range ≥ 0.589
  - BuffetAltitude ≥ 0.603
  - TOFL ≤ 0.377
	- Basic Sampler: 25 runs
	- Generation size changed to 60
	- Point Attractor: # Runs set to 600
	  - [Cost = 0.469, Range = 0.607, TOFL = 0.352, BuffetAltitude = 0.797]
	  - [Cost = 0.592, Range = 0.749, TOFL = 0.317, BuffetAltitude = 0.779]
	  - [Cost = 0.481, Range = 0.725, TOFL = 0.351, BuffetAltitude = 0.720]
	  - [Cost = 0.539, Range = 0.800, TOFL = 0.351, BuffetAltitude = 0.720]
	  - [Cost = 0.373, Range = 0.665, TOFL = 0.351, BuffetAltitude = 0.720]
	  - [Cost = 0.320, Range = 0.662, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.656, Range = 0.813, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.644, Range = 0.836, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.495, Range = 0.773, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.407, Range = 0.708, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.323, Range = 0.637, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.648, Range = 0.823, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.483, Range = 0.774, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.425, Range = 0.729, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.672, Range = 0.847, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.598, Range = 0.823, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.635, Range = 0.844, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.358, Range = 0.690, TOFL = 0.370, BuffetAltitude = 0.720]
	  - [Cost = 0.672, Range = 0.839, TOFL = 0.370, BuffetAltitude = 0.720]
	- Brush and preference controls (to obtain feasible and Pareto) data:
	  - Minimize (-100) Cost
	  - Maximize (100) Range ≥ 0.589
	  - BuffetAltitude ≥ 0.603
	  - TOFL ≤ 0.377

Table A.3. Description of automatic constraint handling sampler trials.

Basic (Total Points: 10,000)	Pareto (Total Points: 10,080)
- Basic sampler: 10,000 runs - Brush and preference controls (to obtain feasible and Pareto) data: - Minimize (-100) Cost - Maximize (100) Range ≥ 0.589 - BuffetAltitude ≥ 0.603 - TOFL ≤ 0.377	- Generation size changed to 60 - Brush and preference controls: - Maximize (100) Range ≥ 0.589 - Minimize (-100) Cost - BuffetAltitude ≥ 0.603 - TOFL ≤ 0.377 - Pareto sampler: 10,000 runs
Preference (Total Points: 10,080)	**Attractor (Total Points: 10,000)**
- Generation size changed to 60 - Brush and preference controls: - Maximize (100) Range ≥ 0.589 - Minimize (-100) Cost - BuffetAltitude ≥ 0.603 - TOFL ≤ 0.377 - Preference sampler: Looped until 10,000 or more runs are reached	- Basic Sampler: 25 runs - Generation size changed to 60 - Brush and preference controls: - Range ≥ 0.589 - BuffetAltitude ≥ 0.603 - TOFL ≤ 0.377 - Point Attractor: # Runs set to 600 - [Cost = 0.204, Range = 0.569] - [Cost = 0.442, Range = 0.721] - [Cost = 0.345, Range = 0.527] - [Cost = 0.488, Range = 0.700] - [Cost = 0.357, Range = 0.629] - [Cost = 0.492, Range = 0.713] - [Cost = 0.425, Range = 0.664] - [Cost = 0.510, Range = 0.757] - [Cost = 0.591, Range = 0.798] - [Cost = 0.349, Range = 0.675] - [Cost = 0.409, Range = 0.715] - [Cost = 0.564, Range = 0.809] - [Cost = 0.500, Range = 0.796] - [Cost = 0.600, Range = 0.800] - [Cost = 0.363, Range = 0.658] - [Cost = 0.455, Range = 0.761] - [Cost = 0.396, Range = 0.709] - Brush and preference controls (to obtain feasible and Pareto) data: - Minimize (-100) Cost - Maximize (100) Range ≥ 0.589 - BuffetAltitude ≥ 0.603 - TOFL ≤ 0.377

Acknowledgments

This work is supported by the National Science Foundation under NSF Grant No. CMMI-0620948. Any opinions, findings, and conclusions or recommendations presented in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1]Balling, R., "Design by Shopping: A New Paradigm?" *Proceedings of the Third World Congress of Structural and Multidisciplinary Optimization (WCSMO-3)*, Buffalo, NY, University at Buffalo, 1999, pp. 295-297.

[2]Balling, R. J., Taber, J. T., Brown, M. R. and Day, K., "Multiobjective Urban Planning Using Genetic Algorithm," *Journal of Urban Planning and Development*, Vol. 125, No. 2, 1999, pp. 86-99.

[3]Shanteau, J., "Competence in Experts: The Role of Task Characteristics," *Organizational Behavior and Human Decisions*, Vol. 53, No. 2, 1992, pp. 252-266.

[4]Wilson, T. D. and Schooler, J. W., "Think Too Much: Introspection can Reduce the Quality of Preferences and Decisions," *Journal of Personality and Social Psychology*, Vol. 60, No. 2, 1991, pp. 181-192.

[5]Stump, G., Lego, S., Yukish, M., Simpson, T. W. and Donndelinger, J. A., "Visual Steering Commands for Trade Space Exploration: User-Guided Sampling with Example", *ASME Design Engineering Technical Conferences - Design Automation Conference*, Las Vegas, NV, ASME, 2007, DETC2007/DAC-34684.

[6]Kesavadas, T. and Sudhir, A., "Computational Steering in Simulation of Manufacturing Systems", *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, IEEE, 2000, pp. 2654-2658.

[7]Wright, H., Brodlie, K. and David, T., "Navigating High-Dimensional Spaces to Support Design Steering", *Proceedings of IEEE Visualization 2000*, Salt Lake City, UT, IEEE Computer Society Press, 2000, pp. 291-296.

[8]Winer, E. H. and Bloebaum, C. L., "Visual Design Steering for Optimization Solution Improvement," *Structural Optimization*, Vol. 22, No. 3, 2001, pp. 219-229.

[9]Winer, E. H. and Bloebaum, C. L., "Development of Visual Design Steering as an Aid in Large-Scale Multidisciplinary Design Optimization. Part I: Method Development," *Structural and Multidisciplinary Optimization*, Vol. 23, No. 6, 2002, pp. 412-424.

[10]Messac, A. and Chen, X., "Visualizing the Optimization Process in Real-Time Using Physical Programming," *Engineering Optimization*, Vol. 32, No. 6, 2000, pp. 721-747.

[11]Michalek, J. and Papalambros, P. Y., "Interactive Design Optimization of Architectural Layouts," *Engineering Optimization*, Vol. 34, No. 5, 2002, pp. 485-501.

[12]Janos Madar, J. A. a. F. S., "Interactive Particle Swarm Optimization", *International Conference on Intelligent Systems Design and Applications*, IEEE, 2005.

[13]Stacey D. Scott, N. L., Gunnar W. Klau, 2001, Investigating Human-Computer Optimization, CHI. Minneapolis, MN.

[14]Kaisa Miettinen, M. M. M., "Synchronous approach in interactive multiobjective optimization," *European Journal of Operational Research*, Vol. 170, 2006, pp. 909-922.

[15]Stump, G., Yukish, M. and Simpson, T. W., "The ARL Trade Space Visualizer: An Engineering Decision-Making Tool", *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, AIAA, 2004, AIAA-2004-4568.

[16]Stump, G., Yukish, M., Simpson, T. W., Harris, E. N. and O'Hara, J. J., "Trade Space Exploration of Satellite Datasets Using a Design by Shopping Paradigm", *IEEE Aerospace Conference*, Big Sky, MT, IEEE, 2004, IEEE-1039-04.

[17]Buja, A., McDonald, J. A., Michalak, J. and Stuetzle, W., "Interactive Data Visualization Using Focusing and Linking", *Proceedings of the IEEE Conference on Visualization '91*, San Diego, CA, IEEE Computer Society Press, 1991, pp. 156-163.

[18]Becker, R. A. and Cleveland, W. S., "Brushing Scatterplots," *Technometrics*, Vol. 29, No. 1, 1987, pp. 127-142.

[19]Wang, G. G. and Shan, S., "Review of Metamodeling Techniques in Support of Engineering Design Optimization," *ASME Journal of Mechanical Design*, Vol. 129, No. 4, 2007, pp. 370-380.

[20]Price, K., Storn, R. and Lampinen, J., *Differential Evolution - A Practical Approach to Global Optimization*, Berlin, Springer, 2005.

[21]Madavan, N. K., "Multiobjective Optimization Using a Pareto Differential Evolution Approach", *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, Honolulu, HI, IEEE, 2002, pp. 1145-1150.

[22]Simpson, T. W. and Meckesheimer, M., "Evaluation of a Graphical Design Interface for Design Space Visualization", *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, Palm Springs, CA, AIAA, 2004, AIAA-2004-1683.

[23]Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182-197.

[24]Laumanns, M., Thiele, L., Deb, K. and Zitzler, E., "Archiving with Guaranteed Convergence and Diversity in Multi-objective Optimization", *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, 2002, pp. 439-447.

[25]Mattson, C. A. and Messac, A., "Concept Selection Using s-Pareto Frontiers," *AIAA Journal*, Vol. 41, No. 6, 2003, pp. 1190-1204.