Title

Parasol: an open source, interactive parallel coordinates library for multi-objective decision making

William J. Raseman†* (william.raseman@colorado.edu), Joshuah Jacobson‡ (josh.jacobson@colorado.edu), Joseph R. Kasprzyk† (joseph.kasprzyk@colorado.edu)*

† Department of Civil, Environmental, and Architectural Engineering, University of Colorado Boulder, Boulder, Colorado 80309, United States

‡ Department of Applied Mathematics, University of Colorado Boulder, Boulder, Colorado 80309, United States

Highlights

- We introduce Parasol, an open source visualization library
- Parallel coordinates (PC) are well-suited for environmental decision making
- Parasol provides building blocks for constructing PC-based web apps
- Web apps are easily shared and promote interactive data visualization

Abstract

This paper introduces Parasol—an open source, interactive visualization library to support the development of web applications for multi-objective decision making. Multi-objective optimization is a popular way to explore competing objectives in environmental management problems. Interactive visualizations allow stakeholders to explore and gain insights about the large, high-dimensional datasets produced by multi-objective optimization. Among visualization methods, parallel coordinates are well-suited for this task. However, current software and open source libraries have limited support for these plots. The Parasol library described in this work provides developers with the building blocks to create sharable, interactive parallel coordinates web applications. Moreover, by incorporating state of the art clutter reduction techniques—such as clustering, linking, brushing, marking, and bundling—Parasol improves upon traditional parallel coordinates visualizations. We demonstrate the benefit of such features through simple examples and by exploring a real-world water resources problem commonly used in multi-objective optimization literature.

Keywords

visualization; parallel coordinates; decision making; optimization; web applications

Software availability

- Name of software: Parasol
- Description: an interactive visualization library to support the development of web applications for multi-objective decision making.
- Developer: J. Jacobson (josh.jacobson@colorado.edu) with contributions by W. Raseman and J. Kasprzyk
- Source Languages: JavaScript, HTML, and CSS
- Supported Browsers: Chrome, Firefox, and Opera
- License: MIT
- Availability: https://github.com/ParasolJS/parasol-es
- Cost: Free

1. Introduction

Multi-objective optimization methods generate a suite of diverse solutions to environmental problems with conflicting objectives. These techniques produce Pareto optimal solutions to environmental management problems—meaning that for each solution, an improvement in any objective would decrease performance in another (Pareto, 1964). Such techniques are classified as *a posteriori* approaches because decision maker preferences are incorporated only after the optimization has searched for solutions (Coello Coello et al., 2007; Cohon and Marks, 1975). In contrast, *a priori* approaches incorporate decision maker preferences before optimization and aggregate multi-objective problems to single objective problem (Castelletti et al., 2010), resulting in a single "best" solution. Such aggregated methods have been criticized because they tend to penalize and reward objectives in ways that are difficult to predict (Franssen, 2005; Kasprzyk et al., 2015) and because they reinforce "cognitive myopia" in decision making (Brill et al., 1990). By using *a posteriori* approaches, decision makers can gain new insights about the problem as they explore solutions and consider new objectives (Kasprzyk et al., 2009). For these reasons and due to recent advances in multi-objective optimization, these methods have become increasingly popular for solving complex environmental management problems, particularly for water resources (Maier et al., 2014; Reed et al., 2013), watershed management (Bekele and Nicklow, 2005), and water distribution (Ostfeld et al., 2008; Prasad and Park, 2004). However, *a posteriori* approaches are criticized because they produce large, high-dimensional datasets which can overwhelm and confuse decision makers (Coello Coello et al., 2007; Haimes, 2015; Zeleny, 2005).

To address this problem, interactive visualization tools have been developed to aid in the discovery of environmental management solutions generated by multi-objective optimization (e.g., Kollat and Reed (2007a) and Hadka et al. (2015)). These tools generally apply methods from information visualization—often summarized as overview first, zoom and filter, and details on demand (Shneiderman, 2003)—to explore Pareto optimal solutions using multiple linked plots. Such methods allow decision makers to sift through thousands of solutions with relative ease. Moreover, this interactive, linked visualization approach can help inform the optimization problem itself. For instance, Woodruff et al. (2013) demonstrate integrating these methods with visual analytics (Keim et al., 2008) offers useful insights for improving the problem formulation. The primary issue with this visualization approach is that many plotting types do not scale well for multi-objective problems. Due to its ability to represent high-dimensional data, parallel coordinates (PC) plots have become increasingly popular for interactive, multi-objective optimization visualizations [e.g., (Rosenberg, 2015; Smith et al., 2018)].

Parallel coordinates (PC) is a visualization technique typically used for exploratory analysis of multivariate data and high-dimensional geometry (Inselberg, 2009). Using PC, N-dimensional data is represented by N equally spaced, parallel axes. Each data point in this N-dimensional space is represented by a so-called *polyline* that intersects each axis according to its value for that dimension. Despite the ability of PC to represent high-dimensional data, PC visualization software is still in its infancy. As a result, these plots tend to be simplistic, static, and difficult to share and access. Visualizing PC plots using web applications would alleviate these issues since these applications are easy to share, lend themselves to interactivity, and offer a familiar web browser experience for users (Walker and Chapra, 2014). Such features are essential for environmental decision making projects which involve diverse set of stakeholders, analysts, and decision makers. However, current methods available for developing such applications would require considerable time and money. To address this challenge and promote best practices for PC visualizations, we have created a new, open source library.

In this paper, we introduce Parasol, a JavaScript library for developing parallel coordinates visualizations to enhance environmental decision making. Parasol provides developers with a toolbox for creating their own custom, interactive PC visualizations. This toolbox, known as the application programming interface (API), includes state of the art visualization techniques that allow users to better interact with PC and reduce visual clutter. Parasol is built on D3—a popular visualization library for web development (Bostock et al., 2011)—which offers developers complete control over the form and function of their applications. The goals of this paper are to motivate the use of parallel coordinates for environmental multi-objective decision making, illustrate how Parasol can improve people's access to and the quality of PC visualizations, and more broadly, highlight the usefulness of embedding interactive visualizations into academic literature. To do so, we begin by reviewing the best practices for parallel coordinates described in the visualization literature (Section 2. Parallel coordinates). Next, we provide an overview of Parasol, describe capabilities of the API, and walk through examples that highlight key elements of the API (Section 3. Parasol). To demonstrate the accessibility of these applications, we have embedded URLs in this paper for each example we discuss. We encourage the reader to navigate to and explore these applications in addition to reading the text. Next, we illustrate the development and use of a Parasol application for a multi-objective water resources problem, known as the Lower Rio Grande River (LRGV) case study (Section 4. Multi-objective decision making with Parasol). Last, we discuss further applications for Parasol and future directions (Section 5. Conclusions).

2. Parallel coordinates

Parallel coordinates (PC) is commonly used for exploratory analysis of multivariate data and high-dimensional geometry (Inselberg, 2009). These plots scale well for high-dimensional datasets but PC is often criticized for issues related to overplotting, crossover, order of axes (Fua et al., 1999; Zhou et al., 2008). Parasol implements best practices from recent PC literature to alleviate these issues.

Overplotting, also known as visual clutter, occurs when overlapping polylines obscure patterns of the data. Next, the problem of crossover (i.e., line-tracing) arises when multiple polylines intersect an axis at the same value, making it impossible to be certain which line is which (Heinrich and Weiskopf, 2013). Another criticism of PC is that the ordering of axes "implicitly defines which patterns emerge between adjacent axes" (Heinrich and Weiskopf, 2013). This is important for determining correlations between variables. The ordering of axes issue is commonly alleviated by making the axes interactively reorderable so that users can dynamically explore various pairwise comparisons. Both overplotting and crossover can be mitigated using clutter reduction strategies (Figure 1).

Clutter reduction methods include brushing, density, clustering (using either color or geometry), bundling, highlighting, marking and linking. *Brushing* allows users to dynamically filter plotted data, reducing the total number of polylines significantly (compare Figures 1a and 1b). In addition to filtering, other brushing operations include deleting and labeling data (Becker and Cleveland, 1987). Moreover, altering the transparency of polylines can illustrate high- and low-density regions of data (Figure 1c). Allowing the user to specify transparency dynamically, can enhance the utility of such *density*-based

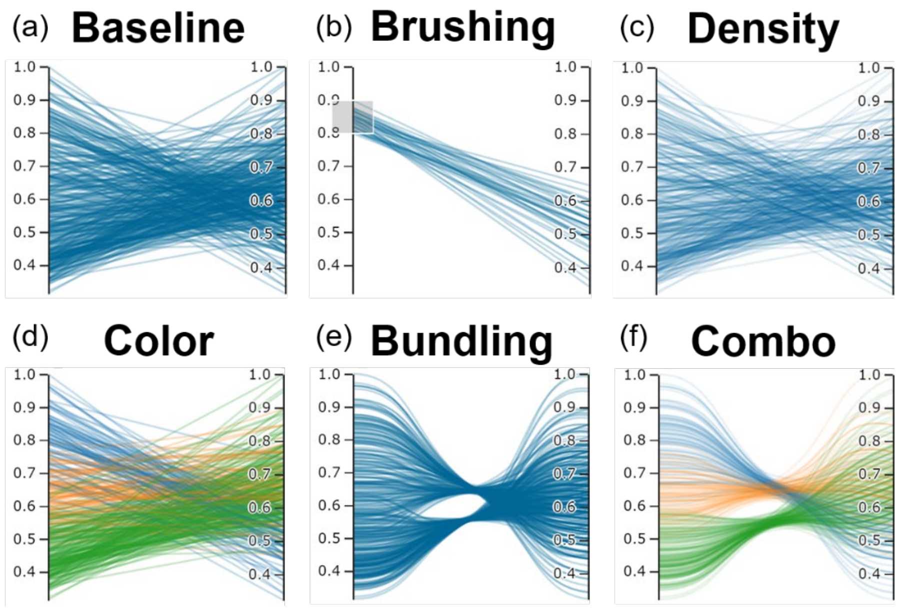

Figure 1. Clutter reduction strategies: a) example of overplotting, b) interactive brushes allow users to subset the data, c) transparency reveals density of the data, d) clustering encoded using color (bottom left), e) clustering encoded geometrically using *curve bundling* (bottom center), and f) an example of combining clutter reduction strategies—cluster encoding with color and curve bundling and adjusting polyline transparency.

clutter reduction methods. Regarding *clustering*, there are several approaches that have been developed for PC, each intended to reveal structure within the underlying data. Clusters can be visually encoded using color (Figure 1d) or geometrically using *bundling* (Figure 1e) (Johansson et al., 2005; Palmas et al., 2014; Zhou et al., 2008). Bundling is a technique that provides visual separation between clusters and is typically implemented with Bézier curves (Figure 1e) to reduce crossover issues, known as *curve bundling*. In a fourteen participant evaluation, Luo et al. (2008) found *curve bundling* to be equally effective as linear polylines for understanding correlations among variables and displaying cluster information. Furthermore, many of these clutter reduction methods can be employed simultaneously in a complementary manner (Figure 1e).

Lastly, clutter can be reduced by *linking* multiple plots together or even connecting them to other plots or data tables. This feature is central to the Parasol library. Linking PC to interactive data tables helps users focus on individual solutions using *marking* and *highlighting*, provides details on demand, and dramatically reduces crossover problems.

3. Parasol

3.1 Library overview

Parasol is an open source, interactive visualization library for developing PC web applications for environmental decision making. We chose a web-based approach for Parasol because web applications are easily shared across diverse groups (Walker and Chapra, 2014), such as the stakeholders, decision makers, and analysts involved in the multi-objective decision making process. Parasol is distinct from most PC software because it is a library rather than a tool, meaning that developers have the freedom to create customized visualizations for their datasets. This library enables users to link multiple PC plots and interactive data tables together, making it ideal for exploratory data analysis for multivariate datasets. Furthermore, Parasol incorporates state of the art clutter reduction techniques to improve user understanding of large datasets.

At its core, the Parasol library is built on D3 (data-driven documents) (Bostock et al., 2011), a popular library for web-based visualization, and three other libraries: Parcoords—a D3-based PC library, SlickGrid—a fast, interactive data table library, and ML—for machine learning. A full list of dependencies can be found in Table 2. We decided to build Parasol around D3 because it is a visualization library that provides developers with enormous control over the aesthetics and function of their visualizations. This control makes Parasol-based visualizations highly customizable and allows developers to couple Parasol visualizations with other plotting types. Parasol streamlines the integration of D3 and the other libraries to lower the barrier for developers creating linked parallel coordinates visualizations.

Table 2. Dependencies for the Parasol library include, D3, Parcoords, Lodash, ML, FileSaver, and SlickGrid.

Dependency	Purpose	npm Package
D3	Interactive visualization library, serves as the foundation for Parcoords.	d3
Parcoords	Interactive parallel coordinates library.	parcoord-es
SlickGrid	Interactive data table that can be linked to parallel coordinates plots.	slickgrid-es6
ML	Machine learning library for implementing *k*-means clustering.	ml-kmeans
FileSaver	Library for exporting data and plots across different browsers.	file-saver
Lodash	Utility function library for JavaScript. Specifically, the functions *intersection*, *union*, and *difference* are used for selections logic.	lodash-es

Among the dependencies, Parasol is most similar to Parcoords, and therefore, we find it important to distinguish between features that we have introduced in Parasol and those that Parasol inherits from Parcoords. In Table 3, we provide a side-by-side comparison of important features for both libraries. This is not an exhaustive list by any means. For a full list of features, please refer to the API for each library. With respect to clutter reduction techniques, Parasol contributes clustering methods to dynamically keep or remove data from plots and the ability to easily link plots and tables together. Moreover, we collaborated with Parcoords developers to add *marking* (i.e., selecting individual solutions) to the Parcoords API because we felt it was an important feature to include in both libraries. Next, we incorporated a simple approach from multi-criteria decision analysis called the weighted sum method into the Parasol API which reflects that Parasol has been tailored to decision making applications. The weighted sum method (i.e., *weighting*) allows users to apply weights to different variables according to their preference for each variable. Based on user-defined weights, each data point is scored from zero to one, with one being the most preferred. Lastly, Parasol has functions that allow user to export data and PC plots.

Table 3. Comparison of features between Parcoords and Parasol parallel coordinates libraries. *Denotes a feature of Parcoords contributed by the authors of this manuscript.

Feature Category	Feature	Parcoords	Parasol
Clutter reduction techniques	Brushing	✓	✓
	Transparency	✓	✓
	Bundling	✓	✓
	Clustering		✓
	Marking	✓*	✓
	Keep/remove data		✓
	Linking		✓
Alleviate order of axes issue	Reorderable axes	✓	✓
Multi-criteria decision analysis	Weighting		✓
Export	Export data		✓

3.2 API

In this section, we discuss the core elements of the Parasol API and walkthrough example code for Parasol-based web applications. To understand how to create a web application, we begin by providing background about web development practices. Web applications are primarily created using web technologies, such as Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript.

Together these technologies integrate with one another to create what is known as the document object model (DOM) to represent a web page (Bostock et al., 2011). HTML code dictates the structure of the webpage, the styling is described in CSS, and JavaScript provides the computational engine and interactivity. Creating a webpage requires a familiarity with web technologies and an understanding of how they communicate with one another.

Developers do not need extensive web development experience to create Parasol-based web applications. The Parasol examples, tutorials, and documentation provides novice developers with the information they need to create an array of simple applications. These developers will likely find example apps similar to those they wish to create and edit the code to better suit their needs. In contrast, experienced web developers will have the freedom to create highly custom and varied applications by leveraging the full capacity of web technologies and other open source visualization libraries. These users will likely go far beyond the examples we provide, finding new and innovative applications for the Parasol library. In other words, we created Parasol and its documentation to scale well for all levels of web development experience. To prove this, we will now demonstrate how to create a simple Parasol web application.

```
<body>
  <div id="plot0" class="parcoords" style="height:200px; width:800px;"></div>
  <div id="plot1" class="parcoords" style="height:200px; width:800px;"></div>
  <div id="grid" class="slickgrid-container" style="height:500px; width:100%;"></div>
</body>

<script>
// create function to create visualize PC and data table
function visualize(data) {
  var layout = {  // specify the axes included in each PC plot
      0: ['power (hp)', 'weight (lb)', '0-60 mph (s)', 'year'],  // plot #1 axes
      1: ['economy (mpg)', 'cylinders', 'displacement (cc)'] // plot #2 axes
    }

  var ps = Parasol(data)('.parcoords')  // create Parasol object
            .setAxesLayout(layout)  // specify axes
            .attachGrid({container: '#grid'})  // attach data table
            .linked()  // link PC plots and data table together
            // add additional Parasol API below
}

var data = d3.csv('data/cars.csv')  // read in data
data.then(visualize)  // pass data into visualize() function
</script>
```

Figure 2. Example code for how to create a Parasol object that links together two parallel coordinates plots and a data table. In the HTML body, space is allocated for the plots and data table. In the HTML script, JavaScript is used to read in and visualize the data using the Parasol API.

To create an application, the developer must first create an HTML document, which will contain HTML and JavaScript code (Figure 2) and references to CSS files for aesthetics. Within this HTML document, the developer must designate space for any PC plot or data table they would like to include in a web page using an HTML <div> elements tag. By creating a <div> element in HTML, the developer can divide a web page into sections. Assigning a unique *id* to each element offers developers the ability to manipulate elements individually. By grouping elements into *classes*, it is possible to manipulate the styling and function of any element within that class simultaneously. For example, in Figure 2, we created three <div> elements, for two PC plots and one SlickGrid data table. In Parasol, we use the class convention of "parcoords" for PC plots and "slickgrid-container" for data tables. Following this convention will preserve the styling and function of the Parasol library.

After allocating space for plots and tables, the developer can bring them to life by creating a Parasol object. To do so, the developer must write JavaScript that reads in a dataset and passes it into a function that visualizes the data. In our example (Figure 2), we read in a dataset on the attributes of cars—a popular dataset for multivariate visualizations—using methods from D3. Then, we pass that data into a user-defined function called *visualize()*. Within *visualize()*, we first specify on which plots the cars variables are rendered by defining an object we call *layout* which contains variables names from the dataset. Next, we create a Parasol object called *ps* that initializes the PC plots of class "parcoords". Once we create *ps*, we can add features from the API by *chaining* commands together. For example, by chaining *.setAxesLayout(), .attachGrid(),* and *.linked()* to *ps*, we can set the axes structure for the plots, initialize the interactive data table, and connect the plots and table together, respectively.

Now that we have described how to implement the API, we will step through three example web applications that provide an overview of the API and illustrate the versatility of the library. The first web application (Figure 3) shows the cars data using two linked PC plots for which the polylines are colored according to their fuel economy. By default, the user can dynamically filter the data on a PC plot by clicking and dragging their mouse along an axis, creating what are known as *brushes*. *Brushes, marks,* and *highlights* (described below) are referred to collectively as *selections* in the Parasol API. Brushes can be resized or deleted completely by clicking anywhere on the brushed axis outside the brushed extents. Since the PC plots are linked, brushing in one plot will impact the data that appears in the other. However, brushes merely filter the data temporarily; if the brushes are removed the data will reappear on the plot.

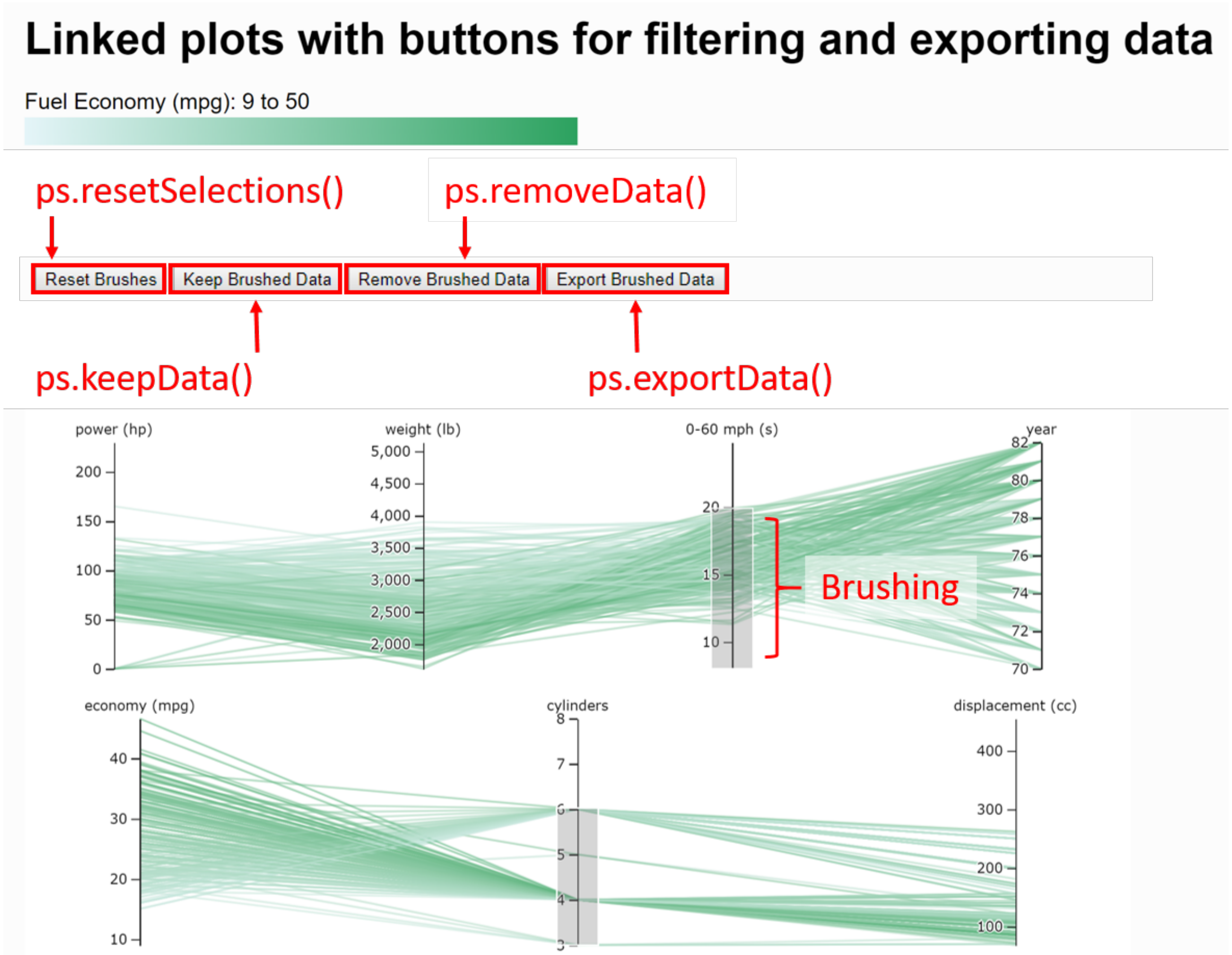

Figure 3. Parasol-based web application that demonstrates brushing across linked parallel coordinates plots and how buttons can be used to modify a Parasol object, *ps,* using the Parasol API. Parasol API: 1) *ps.resetSelections()* clears any brushes on either plot; 2) *ps.keepData()* keeps any data within the brush extents from *ps* and removes the rest; 3) *ps.removeData()* removes only the data within the brush extents; and 4) *ps.exportData()* exports only the data within the brush extents. URL: https://parasoljs.github.io/demo/paper-example-1.html

To remove data from the Parasol object permanently, we have developed the *keepData()* and *removeData()* methods for the API. *ps.keepData()* keeps all data within the brushed extents but removes all other data from the Parasol object, *ps*, whereas *ps.removeData()* removes the data within the brushed extents. As shown in Figures 3 and 4, methods from the Parasol API, like *keepData()*, can be embedded into interactive buttons. The other buttons shown in Figure 3 allow users to reset brushes across plots and export the brushed data to a comma-separated value file.

```
<body>
  <button id="keep_brushed">Keep Brushed Data</button>
  <button id="remove_brushed">Remove Brushed Data</button>
<body>

<script>
// create Parasol object
var data = d3.csv('data/cars.csv')  // read in data
data.then(visualize)  // pass data into visualize() function
function visualize(data) {
  var ps = Parasol(data)('.parcoords')  // create Parasol object
}
// activate buttons
d3.select('#keep_brushed').on('click', function() {
  ps.keepData('brushed')  // keep brushed data on button click
});
d3.select('#remove_brushed').on('click', function() {
  ps.removeData('brushed')  // removed brushed on button click
});
</script>
```

Figure 4. HTML and JavaScript code demonstrating how the Parasol API can be embedded in interactive elements of a web page. In this example, *ps.keepData()* and *ps.removeData()* are activated when their respective buttons are clicked by users and their effects are applied to brushed data.

The next two web applications (Figures 5 and 6) visualize the same dataset as the first and each apply clustering to reduce visual clutter. As discussed previously, clustering is used to reveal structure within the data by identifying which data are most similar. In the Parasol API, *cluster()* performs the clustering analysis using *k*-means clustering—an algorithm often used for clustering in PC literature. Using this approach, to specify a *k* of three, the developer would write *ps.cluster(k=3)*. However, choosing this value *k* is not always clear as there is no consensus on a single best approach for how *k* should be chosen (James et al., 2013). One method is to choose *k* based on the number of clusters based on how effectively each additional cluster reduces the within cluster sum of squared deviations from each observation and its centroid. If the best *k* is not clear from such an analysis, the developer can make it possible to change *k* dynamically. For example, in Figure 5, we have created an example in which *k* can be altered using an interactive slider. In contrast, the number of clusters can also be hard-coded by the developer, like in Figure 6 where *k* is set equal to four. The clustering method in Parasol can also be used to specify which variables are included in the clustering calculation. Figure 5 shows an example in which the user can interactively specify which variables are included in the clustering calculation.

After each data point is assigned to a cluster, the clusters can be encoded geometrically, using color, or both with the Parasol API. By default, the *ps.cluster()* method will assign color to the polylines according to each cluster. If the developer would like to use color for another purpose, clusters can be represented geometrically using bundling instead. As mentioned previously, bundling is generally combined with the use of Bézier curves called *curve bundling* (Luo et al., 2008). Curve bundling is controlled by two parameters: bundling strength—*bundlingStrength()*—and curve smoothness—*smoothness()*. To bundle based on clusters, the clustering variable would be input to *bundleDimension()*. Since there is currently no automated procedure for determine the best values of these curve bundling parameters (Luo et al., 2008), they can be tuned by the user as they see fit. This approach is demonstrated in Figure 6.

In addition to clustering, these figures demonstrate *highlighting* (Figure 5) and *marking* (Figure 6)—features that are most useful when linking plots and tables using the Parasol API. To highlight a data point, the user can simply hover the computer mouse over the row of interest on the data table. Once the user moves the mouse outside that row, the highlight will vanish. On the other hand, if users want a more permanent way to select individual polylines, they can *mark* them using the checkbox on the data table. Marked data will remain marked unless the box is unchecked or unless the *ps.resetSelection()* API is used to clear the selected data. Both highlighting and marking are highly effective for clutter reduction and alleviating crossover issues.

Lastly, we have incorporated the weighted sum method from multi-criteria decision analysis (MCDA) literature into the API so that users can assign preference to different variables and calculate an aggregate score for each data point. In Figure 6, weights can be assigned based on user input and implemented in Parasol using the *weightedSum()* method. These weights can be determined based on MCDA weighting schemes (Zanakis et al., 1998), such as Analytic Hierarchy Process (Saaty, 2008), or specified by the user. To accommodate differing scaling practices across approaches, *weightedSum()* allows the user to specify whether the variables are normalized from zero to one. After the weighted sum is calculated, the score is also normalized.

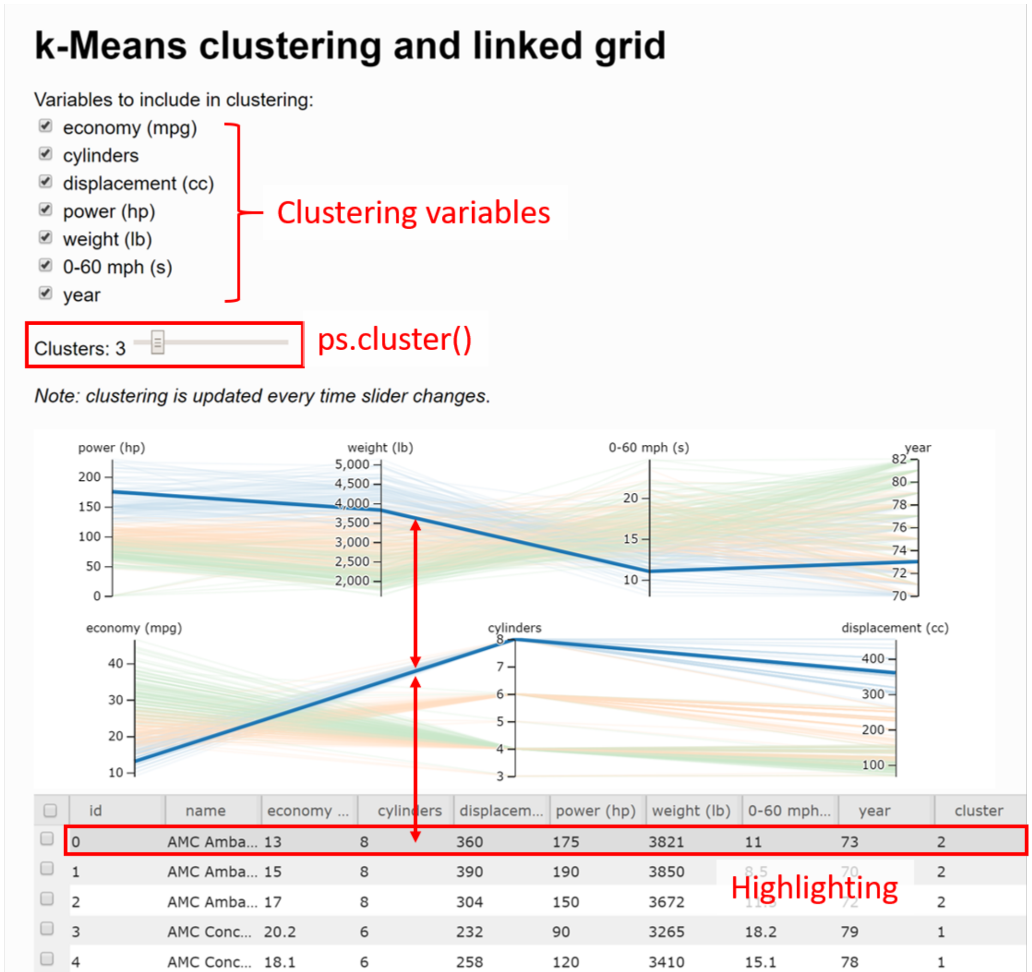

Figure 5. Parasol-based web application that colors parallel coordinates plot polylines based on *k*-means clustering. Using HTML sliders and checkboxes, the user can alter arguments to the clustering method, *ps.cluster()*. The web application also demonstrates how linking plots and tables allows the user to *highlight* individual data points by hovering their mouse over a row on the data table. URL: https://parasoljs.github.io/demo/paper-example-2.html

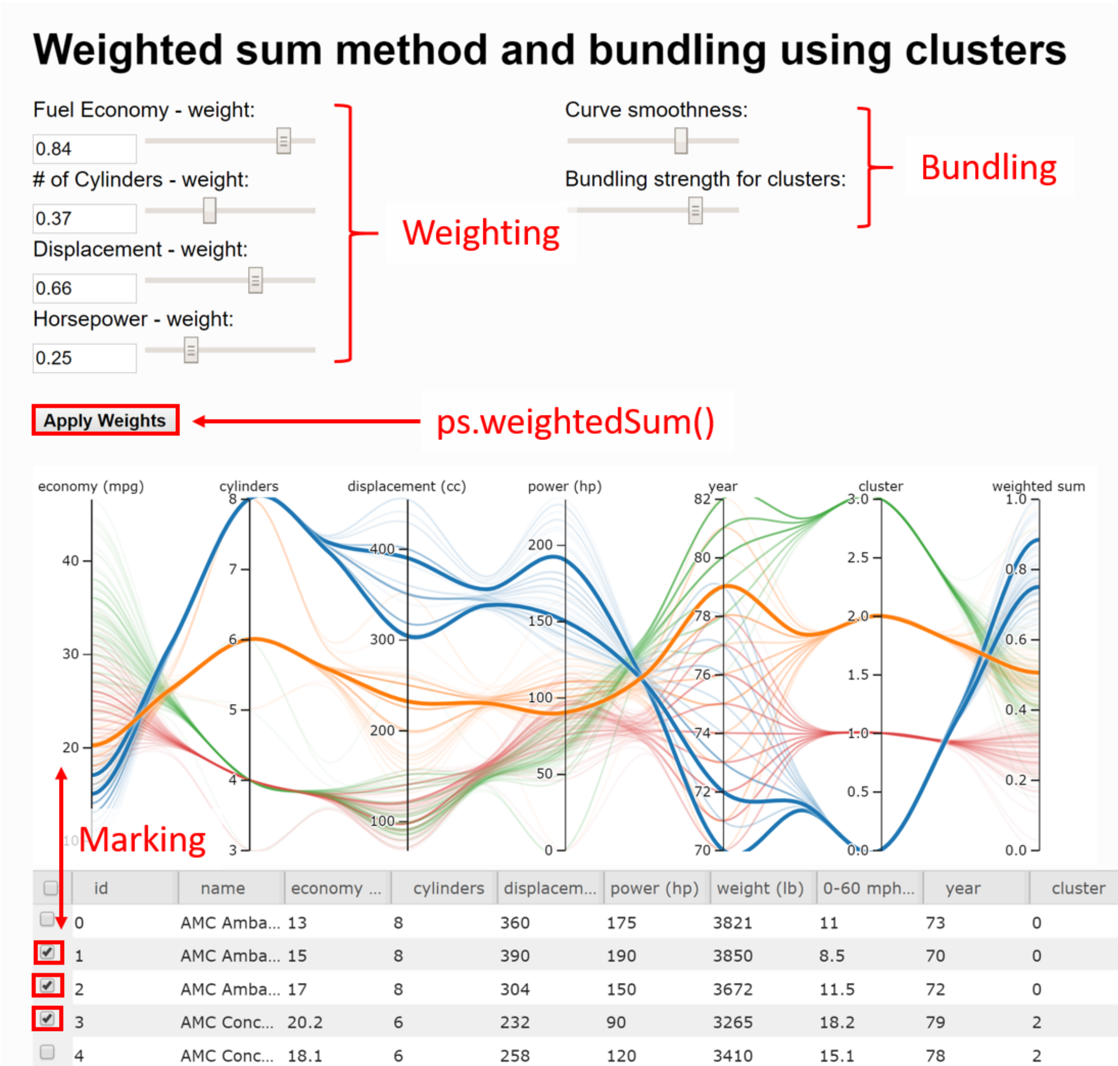

Figure 6. Parasol-based web application that allows users to specify weights to different metrics to calculate an aggregate score for each data point (i.e., car). This is achieved using the *ps.weightedSum()* function of the Parasol API. This example also shows how linking a data table with the plot allows users to *mark* solutions of interest and how *curve bundling* can be used to reduce visual clutter. URL: https://parasoljs.github.io/demo/paper-example-3.html

4. Multi-objective decision making with Parasol

In the previous section, we described the functionality of Parasol using a multivariate dataset about cars. Although Parasol is suited for many forms of multivariate analysis, we created it specifically for *a posteriori* multi-objective decision making. As discussed previously, such *a posteriori* methods search for Pareto optimal solutions (i.e., management alternatives), which can inform decision makers about tradeoffs between the objectives they care about. Instead of aggregating objectives based on *a priori* preferences to find a single solution, *a posteriori* methods use an exploratory approach to make decisions. Using this discovery-based method of multi-objective decision making, the decision maker can gain insights about the problem that may diverge from their *a priori* preferences.

In this section, we demonstrate the utility of Parasol for performing such an analysis by investigating the Pareto optimal solutions from the Lower Rio Grande Valley (LRGV) water resources management case study (Characklis et al., 2006; Kirsch et al., 2009). This case study has been widely used in the literature as representative of a real-world management problem. In this paper, we use a dataset which results from the "constrained" multi-objective formulation described in Clarkin et al. (2018) that is based on the problem formulations in Kasprzyk et al. (2012, 2009).

4.1 Lower Rio Grande Valley (LRGV) case study

In the LRGV case study, a hypothetical municipality attempts to manage their water supply efficiently in the face of uncertain supply and demand due to population growth, agricultural demand, and transboundary water issues between the United States and Mexico. This municipality has three instruments with which it develops its water supply planning portfolio: permanent rights, spot market leases, and adaptive options contracts. It is assumed that the city and all other water users in the region get their supply from a single reservoir source. By buying permanent rights, the city can acquire a percentage of reservoir inflows. The city can also purchase water using two market-based instruments known as "transfers": spot market leases and adaptive options contracts. Spot market leases can be acquired in any month of the year but have a variable price. Adaptive options contracts can be purchased early in the year to guarantee a fixed price for purchasing water at a specified time later in the year. These market-based instruments enable the city to diversify its water supply portfolio, rather than relying solely on permanent rights.

Supply portfolios are evaluated based on their performance for two types of simulation: 1) a Monte Carlo approach representative of historical conditions and 2) a drought scenario characterized by low flow and high demand (Kasprzyk et al., 2009). Both simulations are run on a monthly timestep. For the Monte Carlo approach, each portfolio is evaluated based on its performance over a 10-year period across 1,000 Monte Carlo simulations of supply, demand, and market prices from historical data. Portfolio metrics are calculated using expected values and other statistical measures from the distribution of Monte Carlo simulations. In contrast, the drought scenario is single-year, deterministic simulation; therefore, performance metrics for drought do not need to be summarized using statistical measures.

For the multi-objective optimization problem formulation in this paper, there are nine objectives (i.e., performance metrics) which are controlled by eight decision variables and subject to four constraints (equations 1-6):

$$F(x) = (f_{cost}, f_{num.\ leases}, f_{cost\ var.}, f_{dropped}, f_{dr.\ trans.\ cost}, f_{rel.}, f_{crit.\ rel.}, f_{dr.\ vuln.}, f_{surplus}) \quad (1)$$

$$x = (N_R, N_{O,low}, N_{O,high}, \xi, \alpha_{Jan-Apr}, \beta_{Jan-Apr}, \alpha_{May-Dec}, \beta_{May-Dec}) \quad (2)$$

$$Subject\ to: \quad c_{rel.}: \ f_{rel.} \geq 0.98 \quad (3)$$

$$c_{crit.\ rel.}: \ f_{crit.\ rel.} \geq 0.99 \quad (4)$$

$$c_{cost\ var.}: \ f_{cost\ var.} \leq 1.2 \quad (5)$$

$$c_{dr.\ vuln.}: \ f_{dr.\ vuln.} = 0 \quad (6)$$

where $F(x)$ is a vector of the objectives, x is a vector of decisions and c_i is a constraint on objective i.

These nine objectives are categorized into three groups: efficiency, risk indicator, and market use. The five efficiency objectives include the cost, surplus, cost variability, dropped transfers, and drought transfer cost. The three risk indicator objectives include the reliability, critical reliability, drought vulnerability. The ninth objective is the 10-year expected surplus water, an indirect measure of environmental impacts of water supply management. By lowering surplus, the municipality can divert water to nonurban uses such as ecological flows. We provide additional details about these objectives in Table 4. The performance of the portfolios is subject to four constraints on reliability, critical reliability, cost variability, and drought vulnerability (equations 3-6). Each supply portfolio is comprised of eight decisions that dictate the timing and magnitude of water purchases and the instrument by which the water is purchased. These decisions are fixed in time for each simulation but are formulated so the acquisition of water by the city via market-based instruments is flexible to changing conditions.

Unlike market-based instruments, permanent rights can only be bought at the beginning of the simulation; therefore, the municipality's rights, N_R, are constant throughout the simulation. Permanent rights are purchased volumetrically (in acre-ft), but water is allocated as a percentage of the total inflow to the reservoir for each month after accounting for losses like evaporation. Allocating water proportional to inflow means the city generally does not receive its full volume. On average 0.725 acre-ft is allocated for every 1 acre-ft purchased for this system (Characklis et al., 2006; Kasprzyk et al., 2009).

Table 4. Objectives for Lower Rio Grande Valley water resources problem

Objective type	Objective	Symbol	Description
Efficiency	Cost	f_{cost}	Minimize cost of rights, options, and leases over 10 years
Market use	Number of leases	$f_{num.\ leases}$	Minimize number of spot leases over 10 years: a proxy for transaction costs for acquiring leases
Efficiency	Cost variability	$f_{cost\ var.}$	Minimize cost variability for the year with the highest variability over 10 year planning horizon
Efficiency	Dropped transfers	$f_{dropped}$	Minimize the number of leases and exercised options that expired after nonuse over 10 years
Efficiency	Drought transfer cost	$f_{dr.\ trans.\ cost}$	Minimize cost of options and leases during the drought scenario
Risk indicator	Reliability	$f_{rel.}$	Maximize the probability of avoiding failure (i.e., expected supply is less than expected demand in a given month). Based on the worst year of the 10-year simulation
Risk indicator	Critical reliability	$f_{crit.\ rel.}$	Maximize the probability of avoiding *critical* failure (i.e., expected supply is less than 60% of expected demand in a given month) over 10 years
Risk indicator	Drought vulnerability	$f_{dr.\ vuln.}$	Minimize the volume of the most severe supply failure during the drought scenario
Efficiency	Surplus water	$f_{surplus}$	Minimize average surplus water at the end each year to support nonurban uses (e.g., ecological flows) over 10 years

The choice of options contract, N_O, determines the maximum volume of water the city can purchase in the options exercise month (i.e., May). Each simulation year, whether the city can purchase high- ($N_{O,high}$) or low-volume options ($N_{O,low}$) is dependent on their ratio of current supply at the start of the year to its permanent rights, ξ. These three decisions, $N_{O,high}$, $N_{O,low}$, and ξ, dictate the type of options contracts available to the municipality each year. If the city purchases transfers during the options exercise month, they will buy options unless spot leases are less expensive. In all other months, they can only buy spot leases. How much and when the city purchases water on the market is dependent on two anticipatory thresholds, α and β.

With regard to market-based supply, the city's choice of α and β determine "when" and "how much" water they will purchase, respectively. Specifically, they must buy water on the market when the ratio of the city's current supply to expected demand is less than α in that month. The amount of water they purchase through leases or options must increase that ratio to β. In this problem formulation, the values of α and β are time-dependent; one set of thresholds is used from January-April ($\alpha_{Jan\text{-}Apr}$ and $\beta_{Jan\text{-}Apr}$) and another is used from May-December ($\alpha_{May\text{-}Dec}$ and $\beta_{May\text{-}Dec}$). The beginning of the year (January-

April) is characterized by lower flows and demand than the May-December period on average (Characklis et al., 2006).

4.2 LRGV web application and analysis

Based on the problem formulation described above, Clarkin et al. (2018) generated a Pareto optimal set of water supply portfolios using the Borg Multi-Objective Evolutionary Algorithm (Hadka and Reed, 2013). Using this dataset, we created a Parasol-based web application to perform an exploratory analysis of these Pareto optimal portfolios. In this section, we describe: 1) the structure of the web application and the API that was used to create the example and 2) an example analysis of the the data. To understand the interactive experience more fully, we recommend that the reader opens the example using their web browser to perform their own mock analysis.

4.2.1 Creating the web application

The LRGV application is composed of two PC plots and an interactive data table. For this application, we chose to visualize the objectives and decisions for the portfolios in separate but linked parallel coordinates plots. Kollat and Reed (2007a) suggest that linking the objective and decision space in the manner provides a more holistic at the performance and design of the Pareto optimal solutions. Like the first example Parasol application above (Figure 4), assigning different variables in the dataset to different plots is achieved by using *ps.setAxesLayout()*. Using this method, we assign the objectives to the top plot and the decisions to the bottom. Adding the interactive data table using *ps.attachGrid()* provides the user with details on demand for individual solutions of interest, and linking the plots and table together using *ps.linked()* enables the user to explore the relationship between the objectives and decisions.

By default, the extents of the PC plots are set to the maximum and minimum value of the data for each axis. In this case, we would like to alter these extents manually to improve the visual comparison between similar decision variables. For instance, α and β have the same units and their ranges are nearly identical. By setting the extents for each $\alpha_{Jan\text{-}Apr}$, $\beta_{Jan\text{-}Apr}$, $\alpha_{May\text{-}Dec}$, and $\beta_{May\text{-}Dec}$ based on their joint maximum and minimum values, it is easier to examine the relationships between these variables (Figure 7a). The same is true with the options variables $N_{O,high}$ and $N_{O,low}$. Therefore, *ps.scale()* was used to alter the extents the α, β, and N_O axes. By doing so, it becomes clear that the $N_{O,high}$ and $N_{O,low}$ are nearly equal to one another for all the Pareto optimal portfolios. This might suggest that the problem formulation could be simplified by combining these decisions into a single number of options variables, N_O.

To help users of the application identify similar water supply portfolios, we implemented *k*-means clustering with *ps.cluster()*. We decided to encode these clusters using color in this case. We chose $k = 3$ for the number of clusters by performing an external analysis of the within cluster sum of squared deviations from each observation and its centroid for several values of *k*. Because the *k*-means clustering algorithm does not guarantee the best global solution, the search is inherently random; therefore, the clusters may vary between runs of the algorithm (James et al., 2013). In this web application, the user has the option to search for clusters based on the objectives alone, decisions alone, or both the objectives and decisions together. A user that clusters based on the objectives is looking for similar *performing* portfolios, while one that clusters on decisions might be more interested in similar portfolio *design*. In our example, we will consider performance and design in the clustering. Based on this procedure, we arrive at the clusters shown in Figure 7 and perform an exploratory analysis of the solutions.

4.2.2 Exploratory analysis

Before we begin the analysis, it is important to note that each solution in this dataset is Pareto optimal and meets the constraints defined in the problem formulation. Therefore, all solutions should be acceptable to the decision maker. The goal of this exploratory analysis is to gain insights about the problem to inform the decision maker about what solutions they prefer most. To begin our example analysis, we will examine the tradeoffs between the Pareto optimal solutions (i.e., portfolios).

Horizontal lines between two objectives axes suggests that the objectives are highly correlated. In other words, there is little to no conflict between these objectives among the Pareto optimal solutions. For instance, this is the case with surplus supply and cost (Figure 7c). Portfolios with high cost related to rights, leases, and options also have high surplus, which may lead to low ecological flows. Tradeoffs, or negative correlations, are represented by crossing lines. This behavior is demonstrated between the number of leases and surplus water (Figure 7d). This suggests that for solutions that there is a conflict between leases and surplus water for our Pareto optimal solutions: decreased market activity leads to increases in surplus supply.

If this PC were static, the user would only be able to examine pairwise relationships between variables. However, Parasol-based PC plots can be made dynamically reorderable axes using the *reorderable()* method. With reordering enabled, the user can simply click on the axis label and drag axes around to analyze relationships between any variable on that plot (Figure 7e). For example, by moving the dropped transfers axis next to the leases axis, we notice an interesting relationship between these variables. Most portfolios—those in the blue and green clusters—exhibit a tradeoff between leases and

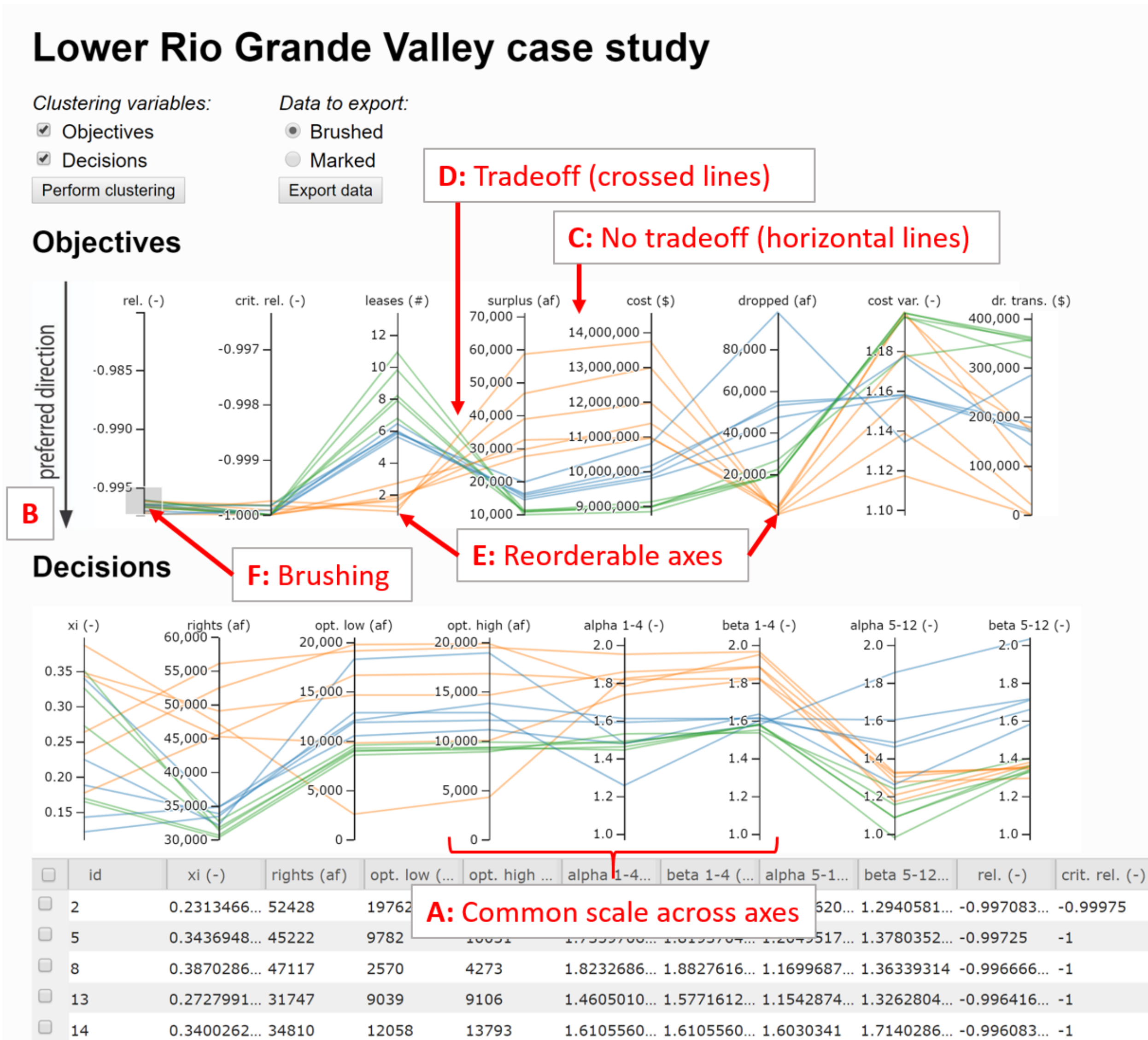

Figure 7. The Lower Rio Grande Valley Parasol-based web application. A) Using *ps.scale()*, the extents of parallel coordinate axes can be altered. B) The objectives are oriented so there is a common preferred direction across all objectives—negative values indicate that the objective was maximized during the optimization. C) Horizontal lines represent that there is no tradeoff between variables, while D) crossing lines represent tradeoffs. E) To examine additional pairwise relationships, the user can dynamically reorder parallel coordinates axes. F) Filtering solutions using brushes reduces the number of plotted solutions and can be exported using ps.*exportData()*. URL: https://parasoljs.github.io/demo/lrgv.html

From a risk perspective, the clusters have similar performance with respect to reliability and critical reliability and all solutions are constrained to have zero drought vulnerability. In fact, even if we filter out solutions with reliability less than 99.5% with brushing, we still have multiple portfolios from each cluster (Figure 7f). Assuming a risk-averse perspective, we can select these "high reliability" solutions to examine the clusters further. Since there is little difference between clusters with respect to risk, the differences must lie in efficiency and market activity objectives and the decisions that make up the portfolios. Let us examine each cluster individually.

The orange cluster is characterized by high surplus water and cost and low dropped transfers and number of leases. Drought transfer costs and cost variability tend to be low but have considerable variability. In fact, the orange cluster contains portfolios with the lowest and the highest cost variability among these "high reliability" solutions. The decisions that make up this cluster are distinct from the others in a few ways. The orange cluster has most permanent rights by far, with some portfolios purchasing nearly the maximum allowable volume of 60,000 acre-ft. These portfolios also have strikingly similar α and β values, with high values during January-April and relatively low values during May-December. The decisions related to options, on the other hand, are quite mixed. There is a negative correlation between the number of options and the options threshold, ξ, for these solutions, which is a behavior unique to this cluster. In contrast, the green cluster represents the opposite end of the spectrum compared to the orange cluster with respect to both performance and decision making.

The green cluster has high market activity—represented by many leases, drought transfers costs, and cost variability. However, these seemingly volatile portfolios do have the best performance regarding surplus water and cost. This cluster also has some of the lowest dropped transfers performance, second to the orange cluster. Additionally, the portfolios in the green cluster have remarkably similar objective and decisions values except for the number of leases which appears to be controlled by varying ξ. It appears that incremental improvements in other objectives have a dramatic effect on the number of leases required.

In many respects, the blue cluster can be described as a compromise between the orange and green clusters. It has moderate performance in the number of leases, drought transfers, and cost variability compared to the other clusters. The values of surplus water and cost for blue portfolios are nearly as low as the green. What differentiates this cluster from the rest is the high number of dropped transfers. As a reminder, dropped transfers are volumes of water that were purchased on the market but expired before they could be used by the city. The decisions that characterize this cluster are low permanent rights and relatively constant α and β values over time. The portfolios in the other clusters tend to have higher $\alpha_{Jan\text{-}Apr}$ and $\beta_{Jan\text{-}Apr}$ and $\alpha_{May\text{-}Dec}$ and $\beta_{May\text{-}Dec}$ values than. These decisions represent higher market activity during the low flow and demand period at the beginning of the year. Market activity for the blue cluster solution is relatively independent of time, with the exception of a few solutions that actually increase market activity during the latter part of the year. These portfolios are also the ones with the highest dropped transfer values.

In summary, each cluster represents a group of similar solutions with respect to both performance and design. These clusters reveal structure in the data and provide visual separation between different types of solutions (Luo et al., 2008) for decision makers. For instance, in the LRGV case study the orange cluster relies most heavily on permanent rights and has low market activity. It contains the highest cost and surplus portfolios but has low cost variability, drought transfer costs, and number of leases. The green cluster portfolios take the opposite approach, with high market activity few rights. The blue cluster has moderate performance across objectives, in general, but has the highest volume of dropped transfers. Each of these clusters represents characteristics that might align with different stakeholder preferences. For instance, if a user has no preference about dropped transfers, then they would likely want to consider the portfolios within the blue cluster. If this is the case, they could use brushing to examine exclusively portfolios from the blue cluster. Then, using highlighting on the interactive data table, the user can inspect individual solutions in detail and mark solutions of interest.

At any point during this analysis, the user can export data of interest. In the LRGV Parasol application, we demonstrate the use of *exportData()* for exporting brushed and marked data to a comma-separated values (CSV) file. This method can also be used to export any selected data—either brushed or marked data—and or to export all plotted data.

5. Conclusions

This paper presented Parasol, an interactive parallel coordinates library to support multi-objective decision making in environmental management. This library was created to fill the need for high quality, accessible parallel coordinates visualizations for *a posteriori* decision making. Developed using the JavaScript programming language, Parasol builds upon D3, Parcoords, SlickGrid, and ML. Parcoords provides the foundation for the PC visualizations, SlickGrid offers fast and dynamic data tables, ML support machine learning techniques, and D3 provides general purpose visualization functions like web page and data manipulation. By integrating and expanding upon these libraries, the Parasol API provides developers with the building blocks to create web applications for interactive, linked PC plots and data tables. Using simple examples and real-world environmental management problems, we showed that Parasol applications enable users to efficiently explore high-dimensional datasets and with best practice parallel coordinates features.

We envision that Parasol applications will be used by decision making practitioners and researchers in environmental management and beyond. We expect most developers will create Parasol-based tools composed of exclusively of parallel coordinates and data tables, similar to those we have described in this paper. However, we built Parasol on D3 to provide developers with the freedom to create linked visualizations that accommodate a range of plotting types. For example, parallel coordinates plots linked to interactive maps have been shown to facilitate the understanding of multivariate spatial data (Opach and Rød, 2014). Such tools could be developed using Parasol in conjunction with D3 or other visualization libraries.

More broadly, it is our vision that the multi-objective decision making community will embrace the use of interactive plots for publications, rather than relying solely on static visualizations. Such interactive visualizations would allow the reader to experience the process of *a posteriori* decision making firsthand. We have illustrated this vision in this paper by including hyperlinks to Parasol visualizations in addition to traditional, static plots. Eventually, we imagine a future in which authors could embed interactive visualizations directly into the body of publications. As the dissemination of research continues to shift from a print-centric paradigm towards a more modern, digital approach, such functionality may not be far off. Until that time, we see external, web-based visualizations—like those made with Parasol—as one way to bridge that gap.

Acknowledgements

This work was supported by the U.S. Environmental Protection Agency “National Priorities: Systems-Based Strategies to Improve the Nation's Ability to Plan and Respond to Water Scarcity and Drought Due to Climate Change”, Grant No. R835865 and the Discovery Learning Apprenticeship Program at the University of Colorado Boulder. The contents of this manuscript are solely the responsibility of the grantee and do not necessarily represent the official views of either funding organization. Figures 2 and 3 were created using Carbon which is published and sponsored by Dawn Labs.

References

Becker, R.A., Cleveland, W.S., 1987. Brushing Scatterplots. Technometrics 29, 127–142. https://doi.org/10.1080/00401706.1987.10488204

Bekele, E.G., Nicklow, J.W., 2005. Multiobjective management of ecosystem services by integrative watershed modeling and evolutionary algorithms. Water Resour. Res. 41. https://doi.org/10.1029/2005WR004090

Bostock, M., Ogievetsky, V., Heer, J., 2011. D^3 Data-Driven Documents. IEEE Trans. Vis. Comput. Graph. 17, 2301–2309. https://doi.org/10.1109/TVCG.2011.185

Brill, E.D., Flach, J.M., Hopkins, L.D., Ranjithan, S., 1990. MGA: a decision support system for complex, incompletely defined problems. IEEE Trans. Syst. Man Cybern. 20, 745–757. https://doi.org/10.1109/21.105076

Castelletti, A., Lotov, A.V., Soncini-Sessa, R., 2010. Visualization-based multi-objective improvement of environmental decision-making using linearization of response surfaces. Environ. Model. Softw. 25, 1552–1564. https://doi.org/10.1016/j.envsoft.2010.05.011

Characklis, G.W., Kirsch, B.R., Ramsey, J., Dillard, K.E., Kelley, C.T., 2006. Developing portfolios of water supply transfers. Water Resour. Res. 42.

Clarkin, T., Raseman, W., Kasprzyk, J., Herman, J.D., 2018. Diagnostic Assessment of Preference Constraints for Simulation Optimization in Water Resources. J. Water Resour. Plan. Manag. 144, 04018036. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000940

Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A., 2007. Evolutionary algorithms for solving multi-objective problems. Springer.

Cohon, J.L., Marks, D.H., 1975. A review and evaluation of multiobjective programing techniques. Water Resour. Res. 11, 208–220. https://doi.org/10.1029/WR011i002p00208

Franssen, M., 2005. Arrow's theorem, multi-criteria decision problems and multi-attribute preferences in engineering design. Res. Eng. Des. 16, 42–56.

Fua, Y.-H., Ward, M.O., Rundensteiner, E.A., 1999. Hierarchical parallel coordinates for exploration of large datasets, in: Proceedings of the Conference on Visualization'99: Celebrating Ten Years. IEEE Computer Society Press, pp. 43–50.

Hadka, D., Herman, J., Reed, P., Keller, K., 2015. An open source framework for many-objective robust decision making. Environ. Model. Softw. 74, 114–129. https://doi.org/10.1016/j.envsoft.2015.07.014

Hadka, D., Reed, P., 2013. Borg: An auto-adaptive many-objective evolutionary computing framework. Evol. Comput. 21, 231–259.

Haimes, Y.Y., 2015. Risk modeling, assessment, and management. John Wiley & Sons.

Heinrich, J., Weiskopf, D., 2013. State of the Art of Parallel Coordinates., in: Eurographics (STARs). pp. 95–116.

Inselberg, A., 2009. Parallel Coordinates: Visual Multidimensional Geometry and Its Applications. Springer-Verlag, New York.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An introduction to statistical learning. Springer.

Johansson, J., Ljung, P., Jern, M., Cooper, M., 2005. Revealing structure within clustered parallel coordinates displays, in: IEEE Symposium on Information Visualization, 2005. INFOVIS 2005. Presented at the IEEE Symposium on Information Visualization, 2005. INFOVIS 2005., pp. 125–132. https://doi.org/10.1109/INFVIS.2005.1532138

Kasprzyk, J.R., Reed, P.M., Characklis, G.W., Kirsch, B.R., 2012. Many-objective de Novo water supply portfolio planning under deep uncertainty. Environ. Model. Softw., Emulation techniques for the reduction and sensitivity analysis of complex environmental models 34, 87–104. https://doi.org/10.1016/j.envsoft.2011.04.003

Kasprzyk, J.R., Reed, P.M., Hadka, D.M., 2015. Battling arrow’s paradox to discover robust water management alternatives. J. Water Resour. Plan. Manag. 142, 04015053.

Kasprzyk, J.R., Reed, P.M., Kirsch, B.R., Characklis, G.W., 2009. Managing population and drought risks using many-objective water portfolio planning under uncertainty. Water Resour. Res. 45, W12401. https://doi.org/10.1029/2009WR008121

Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., Melançon, G., 2008. Visual analytics: Definition, process, and challenges, in: Information Visualization. Springer, pp. 154–175.

Kirsch, B.R., Characklis, G.W., Dillard, K.E.M., Kelley, C.T., 2009. More efficient optimization of long-term water supply portfolios. Water Resour. Res. 45. https://doi.org/10.1029/2008WR007018

Kollat, J.B., Reed, P., 2007. A framework for Visually Interactive Decision-making and Design using Evolutionary Multi-objective Optimization (VIDEO). Environ. Model. Softw. 22, 1691–1704. https://doi.org/10.1016/j.envsoft.2007.02.001

Luo, Y., Weiskopf, D., Kirkpatrick, A.E., 2008. Cluster Visualization in Parallel Coordinates Using Curve Bundles. IEEE Trans. Vis. Comput. Graph. 12.

Maier, H.R., Kapelan, Z., Kasprzyk, J., Kollat, J., Matott, L.S., Cunha, M.C., Dandy, G.C., Gibbs, M.S., Keedwell, E., Marchi, A., Ostfeld, A., Savic, D., Solomatine, D.P., Vrugt, J.A., Zecchin, A.C., Minsker, B.S., Barbour, E.J., Kuczera, G., Pasha, F., Castelletti, A., Giuliani, M., Reed, P.M., 2014. Evolutionary algorithms and other metaheuristics in water resources: Current status, research challenges and future directions. Environ. Model. Softw. 62, 271–299. https://doi.org/10.1016/j.envsoft.2014.09.013

Opach, T., Rød, J.K., 2014. Do choropleth maps linked with parallel coordinates facilitate an understanding of multivariate spatial characteristics? Cartogr. Geogr. Inf. Sci. 41, 413–429. https://doi.org/10.1080/15230406.2014.953585

Ostfeld, A., Uber, J.G., Salomons, E., Berry, J.W., Hart, W.E., Phillips, C.A., Watson, J.-P., Dorini, G., Jonkergouw, P., Kapelan, Z., others, 2008. The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. J. Water Resour. Plan. Manag. 134, 556–568.

Palmas, G., Bachynskyi, M., Oulasvirta, A., Seidel, H.P., Weinkauf, T., 2014. An Edge-Bundling Layout for Interactive Parallel Coordinates, in: 2014 IEEE Pacific Visualization Symposium. Presented at the 2014 IEEE Pacific Visualization Symposium, pp. 57–64. https://doi.org/10.1109/PacificVis.2014.40

Pareto, V., 1964. Cours d’économie politique. Librairie Droz.

Prasad, T.D., Park, N.-S., 2004. Multiobjective genetic algorithms for design of water distribution networks. J. Water Resour. Plan. Manag. 130, 73–82.

Reed, P.M., Hadka, D., Herman, J.D., Kasprzyk, J.R., Kollat, J.B., 2013. Evolutionary multiobjective optimization in water resources: The past, present, and future. Adv. Water Resour., 35th Year Anniversary Issue 51, 438–456. https://doi.org/10.1016/j.advwatres.2012.01.005

Rosenberg, D.E., 2015. Blended near-optimal alternative generation, visualization, and interaction for water resources decision making. Water Resour. Res. 51, 2047–2063. https://doi.org/10.1002/2013WR014667

Saaty, T.L., 2008. Decision making with the analytic hierarchy process. Int. J. Serv. Sci. 1, 83–98.

Shneiderman, B., 2003. The eyes have it: A task by data type taxonomy for information visualizations, in: The Craft of Information Visualization. Elsevier, pp. 364–371.

Smith, R., Kasprzyk, J., Basdeka, L., 2018. Experimenting with Water Supply Planning Objectives Using the Eldorado Utility Planning Model Multireservoir Testbed. J. Water Resour. Plan. Manag. 144, 04018046. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000962

Walker, J.D., Chapra, S.C., 2014. A client-side web application for interactive environmental simulation modeling. Environ. Model. Softw. 55, 49–60. https://doi.org/10.1016/j.envsoft.2014.01.023

Woodruff, M.J., Reed, P.M., Simpson, T.W., 2013. Many objective visual analytics: rethinking the design of complex engineered systems. Struct. Multidiscip. Optim. 48, 201–219. https://doi.org/10.1007/s00158-013-0891-z

Zanakis, S.H., Solomon, A., Wishart, N., Dublish, S., 1998. Multi-attribute decision making: A simulation comparison of select methods. Eur. J. Oper. Res. 107, 507–529. https://doi.org/10.1016/S0377-2217(97)00147-1

Zeleny, M., 2005. The Evolution of Optimality: De Novo Programming, in: Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science. Presented at the International Conference on Evolutionary Multi-Criterion Optimization, Springer, Berlin, Heidelberg, pp. 1–13. https://doi.org/10.1007/978-3-540-31880-4_1

Zhou, H., Yuan, X., Qu, H., Cui, W., Chen, B., 2008. Visual Clustering in Parallel Coordinates. Comput. Graph. Forum 27, 1047–1054. https://doi.org/10.1111/j.1467-8659.2008.01241.x