CITYPLOT: VISUALIZATION OF HIGH-DIMENSIONAL DESIGN SPACES WITH MULTIPLE CRITERIA

Nathaniel Knerr
Cornell University, Aerospace and Mechanical Engineering
405 Rhodes Hall, Cornell University
Ithaca, New York 14853
e-mail nsk52@cornell.edu
ASME Membership (if applicable) Student #100812424

Daniel Selva
Cornell University, Aerospace and Mechanical Engineering
212 Upson Hall, Cornell University
Ithaca, New York 14853
e-mail ds925@cornell.edu
ASME Membership (if applicable): Member

ABSTRACT

In the early-phase design of complex systems, a model of design performance is coupled with visualizations of competing designs and used to aid human decision makers in finding and understanding an optimal design. This consists of understanding the tradeoffs among multiple criteria of a “good” design and the features of good designs. Current visualization techniques are limited when visualizing many performance criteria and/or do not explicitly relate the mapping between the design space and the objective space. We present a new technique called Cityplot, which can visualize a sample of an arbitrary (continuous or combinatorial) design space and the corresponding single or multi-dimensional objective space simultaneously. Essentially a superposition of a dimensionally reduced representation of the design space simultaneously.
decisions and bar plots representing the multiple criteria of the objective space, Cityplot can provide explicit information on the relationships between the design decisions and the design criteria. Cityplot can present decision settings in different parts of the space and reveal information on the decision → criteria mapping, such as sensitivity, smoothness and key decisions that result in particular criteria values. By focusing the Cityplot on the Pareto-frontier from the criteria, Cityplot can reveal trade-offs and Pareto-optimal design families without prior assumptions on the structure of either. The method is demonstrated on two toy problems and two real engineered systems, namely the NASA Earth Observing System and a Guidance, Navigation and Control system.

Keywords: Visualization, trade-offs, Preference Incorporation, Multi-Objective Optimization, Multi-Criteria Decision Making, Multi-Attribute Design, Design, Systems Architecting, Dimension Reduction.

1. INTRODUCTION

Decision-based approaches to design as advocated by Hazelrigg [1], Mistree [2] and others have become an increasingly popular form of system design. In the decision-based design paradigm, the decision maker formulates a model that maps design decisions to design criteria. Optimizing the model then finds the preferred design [3,4]. For some examples of formulating such models see [5].

In practice, there is rarely a single quantity of interest so many authors have advocated for considering multiple criteria [6,7]. By not articulating an explicit preference between the design criteria before the optimization stage, a designer is able to consider the possible optimal designs and choose the one best suited to his or her overall goals, and is more informed of what is or is not possible within the design space [6,7].

Visualization tools are used in this a posteriori articulation of preferences (i.e. design by shopping [8]) to help the decision maker understand the tradeoffs between
design criteria, relationships between design decisions, and sensitivities of design criteria to design decisions.

1.1. Common Methods, Interactive Methods and Recent Tradespace Visualizations

The most common method of visualizing the values of criteria of a set of designs is to plot all the criteria values with each criterion as an axis in a scatterplot. However, visualizing the tradeoffs of more than 3-5 criteria quickly becomes intractable. More significantly, it gives no information on what decision changes drive performance changes.

A popular method to visualize more than 3-5 criteria is to instead plot the criteria/objective in a parallel axis plot. This allows plotting an unlimited number of criteria. However, these plots quickly become difficult to read and it is not obvious how to order the axes, which can alter which criteria appear to trade and can unintentionally alter the perception of the objective space. Again, there is no indication relating the design space changes to objective space changes. [6,9–11]

Interactive tools have been devised to provide information about both the design and the objective spaces using multiple views. Cloud Visualization [12], the ARL Trade Space Visualizer [10], RAVE [9] and commercial software such as DiscoveryDV [11] allow for user interaction between elements of standard plots and allow for linking multiple types of views to aid user understanding of the tradeoffs in the space. Kanukolanu et al. use boxes in a scatterplot to represent robust solutions of a simulation [13]. Zhang et al. use boxes of differing size to represent aggregate statistical quantities [14]. Both [13,14] allow for user interaction in choosing what to plot. These methods place interaction and collating information via multiple standard types of plots as the
centerpiece of their approach. This allows for amassing data and aids in user understanding, but requires work to make effective static images representing the decision → criteria map.

Visualizations have been developed specifically for tradeoff analysis. Unal et al. developed an index based on crossings of parallel axis plots to visualize tradeoffs between criteria [15]. Chiu et al. developed the Hyper-Radial Visualization method, which parameterizes multiple criteria and plotting the results on two axis to try and replicate the standard picture of Pareto optimality in 2 dimensions [16]. The Hyperspace Pareto Frontier uses a winding path to form a lossless parameterization of every point in the high dimensional space; however, neighborhoods in the original space may not be reflected by neighborhoods in the visualization [17]. These methods do not take into account the decisions and focus only on finding ways to represent the tradeoffs in 2 dimensions.

It should be noted that a technique known as Cityscapes has a similar name to our technique (Cityplot), but refers to a completely different technique, with more resemblance to a mesh plot built out of bar graphs than any of the techniques described here [18]. Regardless, it often appears in the 90s visualization literature [19,20] and can be a source of confusion.

1.2. Virtual Reality and Dimension Reduction Methods

In the 90s the Information Retrieval and Artificial Intelligence communities looked to virtual reality to organize the early internet and produced the Vineta Virtual Reality for Document Retrieval systems [20] and Benediktine spaces [21]. These techniques build a 3d world to visualize the space of documents and seek to replicate
natural structures. Figures are designed for users to navigate interactively [19,20]. These approaches are valuable as they create a space where one can see relationships between abstract objects, although we believe such methods should strive to accurately represent the abstract space over attempting to replicate real-world structures. One major weakness in these techniques is a need to project high-dimensional objects to low dimensional space. This should be less problematic in a lower-dimensional problem such as systems architecting (~30 decisions) than classifying documents (~30,000 words).

Dimension reduction techniques have been used in visualization to represent high-dimensional objects a low-dimensional space that can be drawn. The Geometric Analysis for Interactive Aid (GAIA) technique in the Preference Ranking Organization METHOD for Enrichment Evaluations (PROMETHEE) method considers pairwise preferences between criteria and then classifies the preference of decisions with respect to the criteria with PCA [22]. Unlike PROMETHEE, we do not seek to represent preference, but rather the mapping from decisions to criteria values. Richardson and Winer pioneered use self-organizing maps (SOM) to visualize a single objective value [23]. Shimoyaa et al. then used self-organizing maps to present the results of a multi-criteria optimization algorithm by presenting each criterion and corresponding robustness measure as a separate SOM [24].

1.3. On Distance Functions
A key requirement of our method will be that a user-supplied distance function is available. Similarity measures and distance functions are commonly used for applications ranging from machine learning to spell checking. Section 14.3 of [25] provides a brief overview of some typical ways to describe object similarities in machine learning and the
relationship to kernel methods [25]. A discussion of building empirical similarity matrices (sufficient for our method) by asking humans to rank pairs of designs by similarity or doing pairwise comparisons is available in [26]. Of interest for discrete design problems is the edit distance of graphs. Among the early papers to give the idea of a graph edit distance is [27] which finds edit distance for a large number of operations when the graphs are attributed by a graph grammar.

Distance functions have been considered in the engineering design literature. McAdams and Wood created a similarity metric for design-by-analogy by considering weighted customer needs and relating these to functional flows. The similarity is then the inner product of the related functional flows weighted by customer needs [28]. These ideas are extended and reapplied to find similar functional requirements over multiple products in [29].

1.4. Paper Overview

The goals of Cityplot are to: (1) assist in seeing how design decisions affect design criteria; (2) allow for understanding the tradeoffs between conflicting criteria; (3) be capable of being applied to both continuous and combinatorial design spaces in high dimension.

Beyond presenting Cityplot, the primary contribution of this paper is to demonstrate:

(1) it is possible to find distance functions in many practical situations that have natural interpretations such as changed decisions; (2) one can visualize many criteria and find structure about tradeoffs and relationships to decisions which achieve those tradeoffs; (3)
modifications in the distance function or set to visualize affect the resulting plot and what
the results mean; (4) Cityplot can be used to gain insight on real-world problems.

2. NOMENCLATURE/SETTING
We define a multi-criteria design problem as an optimization problem of the form:

$$\max_{x \in X} f(x)$$

(1)

where \( f(x) = \tilde{y} \in Y \subseteq \mathbb{R}^d \) is the multi-objective value function of \( d \) criteria, \( Y \)
(the objective space) is the range of \( f \), each component of \( \tilde{y} \in Y \) is a design criterion,
\( \tilde{x} \in X \) is a design, and \( X \) is the space of all feasible designs (the design space), with each
component of \( \tilde{x} \) representing a design decision. For simplicity, we will always depict
our problems as maximization so that a tall building in Cityplot represents a criterion in
which a design performs well. It is possible to take a minimization objective (such as
cost) and turn it into maximization by simply multiplying by -1 (hence, we maximize
negative cost but simply call it “cost”).

We assume (cf. section 3) there is a distance function \( m: X \times X \rightarrow \mathbb{R} \) (where \( \times \)
means Cartesian product) that describes how dissimilar two designs are and obeys the
usual definition of a metric: a nonnegative, symmetric function that satisfies the triangle
inequality and is zero if and only if the designs are identical.
A Cityplot is a graphical representation created by using dimension reduction and bar graphs (cf. section 4) to approximately represent the relationships between designs $\vec{x}$ with respect to the distance function $m$ and the objective function $\vec{f}$ (cf. Fig. 1).

We assume we are able to obtain a sample of known designs and corresponding criteria values $S = \{(\vec{x}_i, y_i)_{i=1}^N\}$ for an interesting subset of the design-objective space. For the purposes of drawing the Cityplot on the generated sample, it is possible for the objective functions to be fully black box.

3. PROBLEM TYPES AND DISTANCE FUNCTIONS

The Cityplot is designed to handle both continuous and combinatorial design spaces that appear often in the early-phase design of complex engineered systems. This section describes five simple, typical problems in early-phase design and some distance functions that can be used on each [30]. Section 7 has actual problem instances for each type.

3.1. Down-Selection

In a down-selection problem, the space of feasible designs can be represented as $X = \{0,1\}^n$. That is, there are $n$ yes/no decisions represented by 1 or 0 respectively. Examples of down selection problems in systems architecting are choosing a subset of functions that the system must perform, or a set of components that the system will comprise [30]. In such a case, we propose simply using the Hamming distance (the
number of changed characters in a fixed length string to transform one string to another) as the distance function:

$$m(\bar{x}_1, \bar{x}_2) = \sum_{i=1}^{n} |(\bar{x}_1)_i - (\bar{x}_2)_i|$$  \hspace{1cm} (2)

This distance has the intuition of the number of decisions that must be changed to make the two designs the same. By using 0 and 1 as elements of $\bar{x}$ this is equivalent to the usual $L_1$ or $L_2$ distance.

### 3.2. Assignment

In an assignment problem, a design consists of assigning $n$ objects (e.g. workers) to $b$ bins (e.g., machines). An object can be assigned to any number of bins and a bin can have zero or multiple objects. The assigning problem appears in system architecture when mapping functions to components, among other applications [30]. A solution to an assigning problem can be represented as a matrix $A \in \{0,1\}^{n\times b}$ indicating if an object is assigned to a corresponding bin (that is, $A_{ij} = 1$ iff object $i$ is assigned to bin $j$). We thus use the following distance function:

$$m(\bar{x}_1, \bar{x}_2) = \sum_{i} \sum_{j} |(A^{\bar{x}_1})_{ij} - (A^{\bar{x}_2})_{ij}|$$  \hspace{1cm} (3)
where the superscript $A_{\vec{x}_1}$ means “the assignment matrix for design $\vec{x}_1$.” This distance is consistent with treating assignment as down-selecting where we down-select on (object, bin) pairs and using the reasoning in section 3.1.

### 3.3. Partitioning

In a partitioning problem, each design, $\vec{x}$, is a partitioning of $n$ objects—a grouping of $n$ objects into discrete subsets such that each object is in exactly one subset. Partitioning is distinct from and more constraining than assigning in the sense that the bin numbers are irrelevant and all objects are each assigned to exactly one bin. Partitioning problems appear in system architecture when choosing a system decomposition, among other applications. Since it is more intuitive to imagine moving objects across bins, we propose using the transfer distance, which is the number of objects that must be moved to make one partition identical to another. Known as the transfer distance, an $O(s^3)$ matching algorithm, where $s$ is the sum of the number of subsets in each partition, exists to calculate the distance efficiently [31]. This algorithm observes that the number of decisions is fixed and hence:

$$m(\vec{x}_1, \vec{x}_2) = n - \max \left( \sum_{i} |S_{\vec{x}_1}^{x_i} \cap S_{\vec{x}_2}^{x_i}| \right)$$

(4)

where $\Theta$ is a mapping that pairs subsets of $\vec{x}_1$ with subsets of $\vec{x}_2$ ($S_{\vec{x}_1}^{x_i}$ and denoted $S_{\Theta(i)}^{\vec{x}_2}$ respectively) and $|\cdot|$ means set cardinality. When $(\vec{x}_1, \vec{x}_2)$ are fixed, it is simply
necessary to calculate all intersections of subsets and then pick the optimal matching, which can be done with the Hungarian Algorithm [31,32].

3.4. Decision-Option

In decision-option problems, better known as catalog design, a design consists of a set of components, each of which is taken from options of a catalog. If we give the elements of the catalog for each decision indices \([1, c_i] \subset N\) then the decision option is given by \(X = [1, c_1] \times [1, c_2] \times \ldots \times [1, c_n]\) where we have \(n\) decisions to make. Decision option problems usually arise when choosing components of different types or from competing vendors to fulfill defined roles in a system [30]. Again, we use the Hamming distance

\[
m(x_1, x_2) = \sum_{i=1}^{n} I((x_1)_i \neq (x_2)_i)
\]

(5)

where \(I(E) = \begin{cases} 1 & \text{if } E \text{ is true} \\ 0 & \text{if } E \text{ is false} \end{cases}\) is the indicator function.

3.5. Continuous Design Spaces

For our purposes, we will assume the design decisions have been normalized and thus are non-dimensional and similar scale. Hence, we take the classic Euclidean distance as a measure of dissimilarity. This is frequently the case where the operators that define performance are assumed to lie on the usual \(L_2\) Hilbert space of functions. The Euclidean distance is also fast, simple and has strong mathematical properties that make it easy to compute and optimize.
3.6. Distance Themes and Summary

The distances we provide for down-selection, assignment and partitioning problems are summarized in Table 1. These distances share the common theme that chosen “default” distances are edit distances—the number of decisions that must be changed to transform one design into another. In continuous design problems, there is not an immediately natural analog for “number of changes” because there is not a discrete “next” level for a given design to take. Instead of counting decision number of changed decisions, the distance will give a radius of a ball that describes sets of decision changes of a similar scale. The Euclidean ball is then the most familiar such ball. If only one decision is changed at a time when creating new designs, the Euclidean distance between each pair in the sequence will recover the size of each change.

When combining distance functions to form composite distances (e.g. as in section 7.4), it is likely that some decisions will be much more significant than other decisions. These decisions should be given higher weight in the distance function. If the value of a given decision disallows values for other decisions, the former decision should also be more significant than the decisions it disallows.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Domain</th>
<th>“Default” Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down-select</td>
<td>({0,1}^n)</td>
<td>(\sum_{i=1}^{N}</td>
</tr>
<tr>
<td>Assignment</td>
<td>({0,1}^{n\times b})</td>
<td>(\sum_{i} \sum_{j}</td>
</tr>
<tr>
<td>Partitioning</td>
<td>Partitions of a set of size (n)</td>
<td>(n - m_{\text{max}} \left( \sum_{i}</td>
</tr>
<tr>
<td>Decision-Option</td>
<td>(\prod_{i=1}^{n}(1, c_i))</td>
<td>(\sum_{i} 1((x_1^i) \neq (x_2^i)))</td>
</tr>
<tr>
<td>Continuous</td>
<td>(X \subset \mathbb{R}^n)</td>
<td>(\sqrt{\sum_{i=1}^{n} ((x_1^i) - (x_2^i))^2})</td>
</tr>
</tbody>
</table>

Table 1: Summary of Problems Presented

4. CITYPLOT DRAWING/TECHNIQUE

Drawing the visualization consists of the four steps below. An example plot is Fig. 1

1. Perform Multidimensional Scaling (abbr. MDS, see section 4.2) on the decisions of a chosen set of designs, usually the Pareto frontier, to ‘flatten’ it to \(\mathbb{R}^2\) (city locations)
2. For each design, plot the criteria values as bar graphs in the 3\(^{rd}\) dimension (skyscrapers)
3. Draw lines to connect close designs and give an indication of their distances (roads). Add a legend to give numeric interpretation to roads (e.g., red means distance =1 in design space \(X\)).
4. Use the mouse interrupts to show text information about a design of interest when clicked. This is left out of static images as it only shows one design at a time.
Fig. 1: Example plot. The data cursor window (text box) displays the design and the criteria values of the selected design (black square). The legend labels road distances.

Reading the plot is intuitive due to an analogy with buildings, cities, and roads. Cities represent designs. Roads connect cities and the color of a road (which is labelled in the legend) gives the distance in the original design space. The height of skyscrapers in each city give the values of each criteria for that design (each skyscraper is a separate criteria for a given design): the taller the building, the better the value of that criterion. The color of a skyscraper represents which criterion the given skyscraper represents; the colors are labelled elsewhere.

When reading a Cityplot, the following facts are important to remember: (1) Multidimensional Scaling considers all distances regardless of whether a road is drawn; roads are purely for visual interpretation and help combat distortion resulting from the dimension reduction. (2) The x and y axis of the Cityplot result from the MDS output and are meaningless—the absolute position of a city in the plot is irrelevant; instead, the relative positions (Euclidean distances between designs) in the XY plane are important and approximate the distances in the original design space.
4.1. Skyscraper Normalization and Visibility

The value of each objective is normalized separately so that the drawing space can be better utilized to show the changes in performance. This is similar to normalizing axes in a parallel axis plot; the normalization gives more visual range to show changes and makes changes in different criteria more visually comparable. The normalization used was:

$$B_i^j = \frac{f(x_i^j) - \min f(x_i^j)}{\max f(x_i^j) - \min f(x_i^j)}$$

(6)

Where $j$ is an index over the designs to visualize and $i$ is the $i^{th}$ objective. The $z$ axis of the Cityplot is the value of each $B_i^j$.

3d rotation features can be used to get a good view of the design space as represented in the Cityplot. Usually, the best angle is isometric to limit the occultation of points by buildings and allow for viewing the most data; the exact optimal settings vary with the exact layout of each Cityplot. Zoom and pan tools provide a way to look at regions of interest.

4.2. City Placement with Multidimensional Scaling

Classical multidimensional scaling (MDS) attempts to map distances between elements in a dataset to a lower dimensional space such that the inter-point distances are well represented in a 2-d plane with the Euclidean distance. The Euclidean distance is preferred in the plane as it is the most natural to the human eye [26].

Classical MDS consists of the following minimization:
where $\vec{x}_i$ is the position of the i\textsuperscript{th} design in design space and $|\vec{x}_i - \vec{x}_j|$ is the Euclidean distance between the designs i and j in reduced space [26]. Intuitively, this objective states that we seek to minimize the average difference between the distances in the 2d onscreen representation and the original distances in the design space. Unfortunately, it is impossible to guarantee the method will always reduce the dimension cleanly, but one purpose of this paper is to show that this can still be useful in many cases [26].

As pointed out in [25], classical MDS is equivalent to PCA when the set to visualize is centered about $\vec{x} = \vec{0}$ and the distance metric $m$ is Euclidean. However, MDS is capable of representing a wider range of distances, including non-Euclidean distance metrics and can perform nonlinear mappings.

Alternative MDS weighting schemes are available [26]. Sammon mapping normalizes the distances as per:

$$\arg\min_{\vec{z}_i} \sum_{i<j} \frac{1}{m(\vec{x}_i, \vec{x}_j)} \left| |\vec{z}_i - \vec{z}_j| - m(\vec{x}_i, \vec{x}_j) \right|^2$$

(8)

This normalization has the effect of amplifying small distances in the design space and is useful when differences in scale in distance values became too large to be cleanly represented or would result in colocation of designs [33].
For this paper, MATLAB’s `cmdscale` (classical MDS) [34] and `mdscale` (non-classical MDS, including Sammon mapping) [35] were used. MDS is simple, has a long, well-understood history and standard implementations in a variety of languages, including MATLAB [34,35] Python [36] and R [37].

5. APPLICATION METHODOLOGY AND CONSIDERATIONS

The proposed methodology to apply this technique is as follows:

1. Choose a distance function $m$ in the design space
2. Choose a set of designs to visualize
3. Draw the Cityplot (see section 4)
4. Examine the Cityplot (see section 6)

   The method is interactive and iterative: after step 4, users can use the data cursor to get information about specific designs and take representatives of regions of the Cityplot. The Cityplot can be rotated or rescaled to give a better view. Queries can be answered by defining appropriate functions and redrawing the plot with these functions as additional criteria. It may also emerge that the distance function is flawed or the plotted set is uninteresting, in which case the user may revisit steps 1 and 2 and redraw the plot.

   To elaborate on step 1, the distance function is encoding what it means for two designs to be alike or different. While the most natural distance functions will emerge from the nature of the problem itself, we provide some default distance functions for different types of problems in section 3 and Table 1. The presented distances should be sufficient in many cases.

   The selection of the set to visualize is tightly related to the original multi-criteria design formulation we wish to examine. Visualizing the entire design space will usually
not be feasible nor desirable due to the very large number of designs that would need to be evaluated, organized and visualized. For this reason, we recommend first isolating the Pareto frontier—the set of designs such that all other options are worse in one or more criteria. Formally, if we use the optimization problem from section 2, then the Pareto frontier is:

\[
P = \{ \bar{x} \in X | \forall \bar{w} \in X - \{ \bar{x} \} \exists \bar{t} \text{ st } \bar{f}(\bar{w}) \leq \bar{f}(\bar{x}) \}
\]  

(9)

In multi-criteria decision-making, the Pareto frontier is the set of all “rational” options. Picking a design amongst the frontier represents trading between criteria, which requires expressing subjective preferences. This set (or an approximation of it) can be found via meta-heuristic algorithms as in [5,6] or classical multi-criteria decision making techniques such as those described in [38]. Unless otherwise noted, all Cityplots in this paper visualize the Pareto set of their respective problems for reasons discussed in section 6.2.

Alternatively, it is possible to take a random subset of the design space; this will yield approximate global information about the design space and objective functions (see section 6.2).

Other problem-dependent sets of interest can also be visualized, but are outside the scope of this paper.

6. FEATURES TO EXTRACT FROM CITYPLOT

By nature, the Cityplot plots similar decisions with the corresponding criteria and provides a natural way to organize the criteria values to be queried by a user. This means that with the data cursor, a user can find decisions that tend to lead to criteria values in
any circumstance. However, the strength of Cityplot lies in its ability to extract global information.

6.1. Smoothness, Design Families and Tradeoffs

The Cityplot can inspect how sensitive designs are to small changes in design features. We define \( (\mathcal{f}, X) \) to be smooth if the distance between any two designs, 

\[
m(x_1, x_2) \text{ is small implies } |f(x_1)_i - f(x_2)_i| \text{ is also small for each criterion } i.
\]

Intuitively, if the distance function \( m \) represents changed decisions then the criteria values usually will not change dramatically as decisions are changed. This definition does not require the decision or objective spaces to be continuous or discrete.

If the space is smooth with respect to the distance function then smoothness should be reflected in the images as a slow transition in the values of the criteria across the image when the Cityplot is drawn. This also makes it easier to find the decisions that drive criteria values as it bounds the changes one would expect in criteria around a design that can be extracted from the data cursor. Knowing whether a given design space is smooth is useful. For example, smoothness indicates robustness to changes in the design. As an example of such transitions, see Fig. 4 or Fig. 8.

Smoothness is not a requirement to apply Cityplot; in fact, Cityplot can be used to assess the smoothness in a region of the design space. If the drawn Cityplot has sharp transitions in the drawn criteria, it demonstrates that the problem is sensitive to “small” changes in designs. It is possible to be smooth over most of the domain and still have unsmooth regions. As example of non-smoothness, see the discussion of Fig. 6.
When plotting the Cityplot, clusters can become visible as closely located cities in the xy plane and group very similar designs in the set of plotted designs. These communities represent design families, which are sets of similar designs. Such families emerge naturally from the application of MDS to the visualized set. Because criteria are a function of design variables, when the function is smooth, designs within the family will have similar criteria values. In such cases, a single design can be used instead of many designs to reduce cognitive load. For examples of such families, see Fig. 3, Fig. 4 or Fig. 8.

As the Cityplot includes skyscrapers representing the objective function values, tradeoffs and objective value relationships between design families and individual designs become quickly apparent by inspection of skyscrapers in the Cityplot. Regions changing color due to the heights of given skyscrapers will represent tradeoffs between objectives.

6.2. Interactions of Distance and Set Choices
The features just discussed are dependent on the choice of set. Clearly, the distance function is also important as it defines the notions of smoothness and design families.

Particular attention should be paid when using the Pareto frontier as the set to visualize. Because inclusion in the set is dependent on the objective function values, clusters represent design families among the optimal designs and objective function values are included in the creation of design families indirectly. Additionally, tradeoffs indicated in the Cityplot will be only amongst the Pareto optimal designs, which coincide with the usual interpretation of tradeoffs being defined between designs in the Pareto-
optimal set. Notice that it is possible for the Pareto frontier to be smooth while the rest of the design space is not (because $P \subseteq X$).

Alternatively, when choosing designs at random, clustering should be ignored, especially when the sample size is very small compared with the size of the design space. However, tradeoffs represent relationships of the criteria in the more general design space. Choosing a random subset allows for a visual query (and possible disproof) of the smoothness of the general space and gives a designer insight into overall behaviors of the objective functions.

7. EXAMPLES

7.1. Toy Example 1: Effect of distance function choice

To demonstrate how the distance function affects the Cityplot and the interpretation, we begin with a down-selection problem in 10 decisions. The objective function produces a vector of three components as follows:

$$\overrightarrow{f(x)} = \left[ \sum_{i=1}^{10} (1 - x_i) \times 10^{-\frac{(i-1)}{3}} \right]$$

$$\sum_{i=1}^{10} (1 - x_i) \times 10^{-\frac{(10-1)}{3}}$$

That is, the function value is simply the weighted sum of decisions of $\overrightarrow{x}$ with weights exponentially distributed from 1 to $10^{-3}$ for the first objective. The second objective simply reverses the order of the sum. This weighting scheme is simple, provides a wide range of effects of decisions and is consistent with a ‘power law’ belief.
in the influence of factors [39]. The last objective simply counts the number of 0’s. We perform analysis by exhaustively enumerating the design space. In the case shown in Fig. 2, we follow section 3.1 and use the Hamming distance as the distance between designs.

![Hamming Distance Demonstration](image)

**Fig. 2: Hamming Distance Demonstration.** Blue skyscrapers are 1\textsuperscript{st} criteria, red are 2\textsuperscript{nd}, green are 3\textsuperscript{rd}. Roads indicate distance (see legend).

To demonstrate the significance of the distance function, we define an alternative distance function:

\[
m'(\overline{x}_1, \overline{x}_2) = \sum_{i=1}^{10} (\overline{(x_i)}_1 - \overline{(x_i)}_2) \ast 10^{-\frac{(i-1)}{3}}
\]

(11)

This distance function closely resembles the form of the 1\textsuperscript{st} criteria but not the 2\textsuperscript{nd} or 3\textsuperscript{rd} criteria. It might emerge from a scenario in which each decision is a decision in a multiscale problem and each decision exists on a different scale from the others. Fig. 3 is the result.
Fig. 3: Exponentially Weighted Distance Function. Blue skyscrapers are 1st criteria, red are 2nd, Green are 3rd. Road colors indicate distance (see legend).

Figure 3 appears to group the designs into communities based on the 1st (blue) objective because the distance function directly corresponds to the calculation of the 1st objective. This results in more variation within the communities in the other objectives. The clustering is a result of the vast difference in weights of the distance function among the first couple of design decisions (components of \( \vec{x} \)) when compared to the last couple of design decisions.

In Fig. 2, the tradeoffs can be seen amongst criteria. The right side of the image has designs maximizing the 1st and 2nd criteria at the cost of the 3rd; the left has designs maximizing the 3rd objective at the cost of the other 2. The designs at the top of the image are minimizing the 2nd objective and the designs at the bottom are minimizing the 1st objective. Looking closely, it is possible to see some sharp transitions in the region with low 3rd objective as the addition of a nonzero element picks a position that causes a roughly ½ the total possible value of the 1st or 2nd objective.

Neither Fig. 2 nor Fig. 3 are necessarily “wrong.” As noted in section 5, the distance function should represent the wider problem context. The former represents a
case where decisions are equally important for determining design similarity and the latter represents a case where some decisions strongly dominate others.

7.2. Toy Example 2: Continuous Function

To demonstrate the applicability of the method to continuous spaces and spaces with many criteria, we design a simple example of a continuous objective function, perform simple optimization and show the usefulness of the Cityplot. The objective function is defined as:

\[
\begin{align*}
&\left( \sum_{i=1}^{\dim(\mathbb{R})} \left( \sum_{j=1}^{i} (j + 0.5) \left( \frac{\mathbf{r}_i}{j} - 1 \right) \right)^2 \\
&\quad - \sum_{i=2}^{\dim(\mathbb{R})-1} 100 (\mathbf{r}_{i+1} - \mathbf{r}_i)^2 + (\mathbf{r}_i - 1)^2 \\
&\quad 100 \| \mathbf{r} + [3, 3, 3, 3] \|_2^4 \\
&\quad 100 \| \mathbf{r} - [1, 1, 0, -1, -1] \|_2^4 \\
&\quad \sum_{i=1}^{\dim(\mathbb{R})} (2.5i - 2.5) |\mathbf{r}_i| 
\end{align*}
\]

(12)

The first component is commonly known as the Perm function [40], the second component is the Dixon-Price function [41] and the third component is the Rosenbrock function [41]. The next two functions are the usual Euclidean norm raised to the 4th power with a minimum at the -3 vector and the [1, 1, 0, -1, -1] vectors respectively. The last function is simply a weighted L_1 norm on the components of \( \mathbf{r} \). To ensure the existence of optima, the solutions are restricted such that \( \mathbf{x} \in \mathbb{R}^5 = [-5, 5] \times \ldots \times [-5, 5] \).
MATLAB gamultiobj is used as an easy method to approximate the Pareto frontier. As there are 6 criteria, the Pareto front is a 5-dimensional space. We use the Euclidean distance as our distance as in section 3.4. Fig. 4 is the resulting Cityplot for 150 Pareto architectures:

![Cityplot Image]

Fig. 4: Continuous Toy Problem Cityplot. Skyscraper colors: blue 1st criteria, red 2nd, green 3rd, black 4th, cyan 5th, yellow 6th. Road Colors are distances (see legend). (top) taller building correspond to optimized criteria values (bottom) flipped heights so taller building correspond to sacrificed criteria values.

Figure 4 indicates that the space of Pareto optimal designs can be thought of as a multiple families of architectures. The top-left family consists of designs, which are usually around $\bar{x} = [4.5, -3, -3, -1, 1]$ with the latter two components being the most variable and ranging from -3 to about 0.25. This family accepts a penalty on the 3rd
objective and a slight penalty to the 6th criteria to optimize the other objectives, especially the 4th. The left-bottom family usually looks like \( \bar{x} = [-4, -3, -2, -\frac{1}{2}, 0] \) and although there is significant spread in the actual decision values, the first component is large and negative. The family accepts a penalty on the 1st criteria and occasionally the 6th criteria to optimize primarily the 2nd and 4th criteria with some designs also performing well on the 5th and 6th criteria. Continuing to move counterclockwise there is a pair of families which are usually around \([-2.5, -2.8, -3.1, 4.6, -2.0]\) and \([-2.5, -2.3, -3, 4.6, 0]\).

Relative to other designs families, these families balance the criteria values. From here there is a large family comprised of the many designs from \([-3, -3, -3, 4.7, 4.8]\) to \([2.5, 3, 3, 1.62, 2.76]\). The large number of designs in this family indicates that there are many ways to adjust these decisions and remain Pareto optimal. Generally, these designs tend to sacrifice in the 6th and 2nd criteria in favor of the 1st and 3rd criteria. There is considerable diversity in how the 4th and 5th criteria are traded with other criteria within this family. On a very close inspection, the family can be thought of as two sub-families with a number of intermediary designs between the two. One family usually keeps the first three decisions negative; the other family has all the decisions positive. The former sub-family usually performs better on 4th criteria whereas the latter prefers the 5th criteria.

7.3. Real-World Example 1: Earth Observing System

In 1990, the US congress authorized the $17 billion Earth Observing System (EOS). Five years later, it had dropped to below $8 billion and had been de-scoped
significantly. In [5] a retrospective case study is performed on the program. In the packaging phase, 16 instruments were placed onto spacecraft for launch. This was treated as a partitioning problem with the instruments being partitioned and each subset representing a spacecraft. The transfer distance is used as the distance function.

In this problem, the Pareto set is approximated with a genetic algorithm and it finds only three designs. When the Pareto frontier is small, it is beneficial to consider more designs to gain a better understanding of the near-optimal solution space. Hence, we include not only the Pareto front, but also the next two Pareto frontiers found after removing the previous Pareto frontiers (this is known as the set of points of Pareto rank 3 or less). Taking all these frontiers together produces a set of 27 designs of roughly optimal objective values to visualize.

The Cityplot for the EOS dataset is seen in Fig. 5. As is seen in the figure, transitions tend to be smooth with a couple exceptions in costs, but the cost transitions much faster than the science score, indicating that there are many of cheaper-but-not-much-less-effective ways to package the instruments. Designs in the far left of the image tend to be mid-high science and lower cost. Very low cost designs are at the bottom and differ strongly from the other architectures. The highest-science high-cost architectures are at the top. Operational risks tend to follow cost and launch risks.
Due to the sparsity of the Pareto front, an interesting question is: how do the criteria vary outside the Pareto front? To answer this question, we create a random subset of designs and run them through the analysis of [5] to extract criteria values. The resulting Cityplot is shown in Fig. 6.

As can be seen in the legend of Fig. 6, the partition space is connected with relatively few changes with designs. It comes then as no surprise that the space has some sensitive criteria. Looking at the very bottom or very left red edges of Fig. 6 has dramatic changes in the 1st criterion (roughly 50% of the range of the criterion). This contrasts with
the smoothness of the approximate Pareto frontier in Fig. 5. Unlike in Fig. 5, objectives
do not always trade consistently in the randomly selected set—there are many ways to
partitioning the instruments that create a suboptimal design.

7.4. Real-World Example 2: Guidance, Navigation and Control System

Dominguez-Garcia et al. [42] conducted a study of how the components of
guidance, navigation and control (GNC) systems are connected in historical NASA
spacecraft. A follow-up study looking at the potential to develop a family of GNC
systems was for this paper. In the follow-up model, a design consists of not only how
many computers and sensors are included but also which subsystems are selected and
how the sensors are assigned to computers. Complete designs are then evaluated in terms
of weight and reliability (weight is a proxy for cost in space systems so the negative
weight will be maximized).

This problem involves multiple related decisions in multiple simple problem
classes (down-selection, decision-option and assignment) from section 3. Regardless, it
still admits a natural distance function created by composing distances of constituent sub-
problems. The number of sensors/computers dictates how many can be taken, and the
components dictate the connectivity pattern leading to the decision hierarchy in Fig. 7.

![Decision Hierarchy for the GNC problem.]

Following Fig. 7, we design the following custom distance function:
where \( M_i = 9 \) and \( M_m = 54 \) is the maximum distance achievable by just changing the low-level and mid-level or below decisions respectively. Superscripts \( C_i \) and \( S_j \) correspond to the \( i \)th computer and \( j \)th sensor respectively, \( I \) is the indicator function and \( C_i \land S_j \) means “computer \( i \) and sensor \( j \) are present and connected.” The distance function can be seen to follow the hierarchy of Fig. 7 with the higher levels in the hierarchy demanding more weight to be completely distinguishable from the lower levels.

The Sammon map was used aid in visually distinguishing elements of the large cluster on the left.

An outside sensitivity analysis reveals that the key driving factor for the various criteria is \( \min(\#S, \#C) \), which is consistent with reliability theory. This is captured very nicely in Fig. 8 as a clustering behavior as the Pareto frontier distinguishes the number of sensors/computers and the distance function then correctly places these as distinct families with jumps in the log reliability.

There is a very large number of designs which use as many computers and sensors as possible. This is because when there are more sensors or computers there is more
flexibility in the other choices. Looking at the substructure of the family, the distinguishing decision is the computer and sensor selections.

The criteria transition smoothly within and across designs families. This is most evident in the performance of different families when moving to the right of each Fig.. There are also decisions, such as number of connections or selection of computer/sensors that cause slight changes in weight and reliability, which roughly corresponds to the y-axis.

**Fig. 8:** GNC when connections expensive. Blue is weight, Red is log-reliability. Road colors indicate distance (see legend).

### 8. LIMITATIONS AND FUTURE WORK

Multidimensional scaling was chosen in this application because it closely resembles what would be desired when trying to represent a high-dimensional graph in 2d space and has well-studied history and developed tools. However, it does not guarantee an accurate approximation of the space [26]. A natural follow-up question is: when do other dimension reduction or machine learning techniques more accurately represent the design space or Pareto front?

When plotting a very large number of designs (300+) the plots can become unacceptably crowded. Thus, one possibility would be to group designs into
neighborhoods that can be approximated with a single design to reveal the similar designs within. It might also be possible to automatically detect design families and group the families into representative designs.

We do not foresee the future of Cityplot as a single standalone solution for all design analysis. A simple but important area of development would be integrating Cityplot into a larger optimization and visualization framework (such as RAVE [9]) and use Cityplot to give global information and select sets of designs for further analysis.

9. CONCLUSIONS

We have presented a visualization technique that combines design similarity-based dimensionality reduction and multi-criteria analysis to aid in systems engineering and design decision-making. In this technique, designs are compared by a similarity measure/distance function and the space of Pareto optimal designs is reduced to a 2-dimensional plane with the values of decision criteria plotted on the same chart to allow for examining how the criteria values change with changing decisions.

Because all criteria are plotted together for multiple designs, it is possible to examine the tradeoffs between decision criteria while maintaining a notion of what designs and decisions make which tradeoffs. The tradeoffs can be seen as a transition in the criteria values as one moves among the space of Pareto-optimal designs. This also makes it possible to visually see sensitivity and smoothness of the Pareto-surface or the general space. The Cityplot is also able to visually extract Pareto-optimal design families where these families are defined by design space similarity.
We have demonstrated the significance of picking a suitable distance function and suggested some simple distance functions for common problems. We have also shown that changing the plotted set can give slightly different information to a designer. Finally, we have demonstrated the applicability of the technique to real world problems and studies.

The ability of Cityplot to visually gather all this information easily is of import for design-by-shopping paradigms. This allows for a human understanding of the tradeoffs in engineering design and decision making as well as how the tradeoffs relate to designs and design decisions, which can help fix decisions early in the design process or suggest key features of a final architecture. Furthermore, the visualization itself places no constraints on the functions to model and hence is widely applicable to many engineering and decision-making scenarios.

The code used to draw the plots is available online at: https://bitbucket.org/Nathan-Knerr/cityplot-matlab.
# NOMENCLATURE

\( f \) \hspace{1cm} \text{Multi-objective value function of} \ d \ \text{objectives}\\
\( Y \) \hspace{1cm} \text{Objective space: the set of all values of objectives possible from} \ f(x)\\
\( x \) \hspace{1cm} \text{A design instance}\\
\( X \) \hspace{1cm} \text{Design space: the set of all feasible designs}\\
\( m \) \hspace{1cm} \text{A distance function (metric). Describes how dissimilar two designs are}\\
\( B_i \) \hspace{1cm} \text{Normalized objective value for the} \ i \ \text{objective}\\
\( z_i \) \hspace{1cm} \text{The position of a design in the reduced (2D) space out. Output of MDS.}\\
\( P \) \hspace{1cm} \text{The Pareto frontier—the set of designs which are not outperformed by another design.}\\
\( S \) \hspace{1cm} \text{A subset from a portioning of a set of objects}\\
\( A^x \) \hspace{1cm} \text{The assignment matrix with represents design} \ x \ \text{in an assignment problem}
REFERENCES


Figure Captions List

Fig. 1  Example plot. The data cursor window (text box) displays the design and the criteria values of the selected design (black square). The legend labels road distances.

Fig. 2  Hamming Distance Demonstration. Blue skyscrapers are 1st criteria, red are 2nd, green are 3rd. Roads colors indicate distance (see legend).

Fig. 3  Exponentially Weighted Distance Function. Blue skyscrapers are 1st criteria, red are 2nd, Green are 3rd. Road colors indicate distance (see legend).

Fig. 4  Continuous Toy Problem Cityplot. Skyscraper colors: blue 1st criteria, red 2nd, green 3rd, black 4th, cyan 5th, yellow 6th. Road Colors are distances (see legend). (top) taller building correspond to optimized criteria values (bottom) flipped heights so taller building correspond to sacrificed criteria values.

Fig. 5  Cityplot of EOS Partitioning Problem. Skyscraper colors: Blue is science return, Red is cost. Green is operational risks, Black is launch risks. Road colors indicate distance (see legend).

Fig. 6  Cityplot of a random selection of designs in the EOS partitioning
problem. Skyscraper colors: Blue is science return, Red is cost, Green is operational risks, Black is launch risks. Road colors indicate distance (see legend).

Fig. 7  Decision Hierarchy for the GNC problem.

Fig. 8  GNC when connections expensive. Blue is weight, Red is log-reliability. Road colors indicate distance (see legend).
Table Captions List

Table 1  Summary of Problems Presented
<table>
<thead>
<tr>
<th>Problem</th>
<th>Domain</th>
<th>“Default” Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down-select</td>
<td>${0,1}^n$</td>
<td>$\sum_{i=1}^n (|x_i|_1 - (|x_i|_1)^d)$</td>
</tr>
<tr>
<td>Assignment</td>
<td>${0,1}^{m+n}$</td>
<td>$\sum_{i,j} | (A^i)<em>{ij} - (A^j)</em>{ij} |$</td>
</tr>
<tr>
<td>Partitioning</td>
<td>Partitions of a set of size $n$</td>
<td>$n - \max_{\sigma} \left( \sum_{i} | S^n_i \cap S^n_{\sigma(i)} | \right)$</td>
</tr>
<tr>
<td>Decision-Option</td>
<td>$\Pi^n_1 {1, c_i}$</td>
<td>$\Sigma_i f((x)<em>{\sigma i} = (x)</em>{\sigma i})$</td>
</tr>
<tr>
<td>Continuous</td>
<td>$X \subset \mathbb{R}^n$</td>
<td>$\sqrt{\sum_{i=1}^{n} (x_i - (\bar{x}_i))^2}$</td>
</tr>
</tbody>
</table>
Journal of Mechanical Design