Identifying Representative Traffic Management Initiatives

Alexander Estes
Applied Mathematics & Statistics, and Scientific Computation
University of Maryland-College Park
College Park, MD, United States of America
aestes@math. umd. edu

David Lovell
Department of Civil & Environmental Engineering and
Institute for Systems Research
University of Maryland-College Park
College Park, MD, United States of America
lovell@umd. edu

***Abstract*—The Federal Aviation Administration uses traffic management initiatives to prevent excessive congestion of airspace resources. We present a method that would aid in the planning of traffic management initiatives by identifying a representative set of initiatives that have been run in the past. In a more general unsupervised learning context, this method could be used to identify a small set of data points that are representative of the entire data set.**

Keywords-air traffic management; traffic management initiatives; unsupervised learning; clustering; representative data

I. INTRODUCTION

Our goal is to provide a method that will increase the availability of data support for planning Traffic Management Initiatives (TMIs), which are used to prevent congestion of high-demand airspace resources. In order to implement a TMI, a decision-maker usually must choose values for several parameters. These may include the time interval that the TMI will be in effect, the set of flights that are controlled by the TMI, and the number of flights that are allowed to access the restricted resource. The decision-maker may wish to inform their decision by consulting a list of previously-implemented TMIs. However, the list could be very large and varied, making manual examination difficult. We would like to take such a list and produce a much smaller set of TMIs that summarize the entire list. This short list would be more easily examined for the purpose of TMI decision-making. In order to aid interpretation, it is important that the TMIs in the list are TMIs from our original dataset. It is also important that a decision-maker has some control over the number of TMIs that are displayed, as well as some feedback as to how precisely the chosen TMIs represent the original data. We provide a method that meets these criteria.

We would like to implement this method as part of a decision-support tool under development in a NASA-funded research project. This tool would start by soliciting from the user a reference day. If real-time operations are being planned, the user would use the current day as the reference day, or the user would choose some historical day if the intent is to conduct post-operations analysis. The tool would then find a (perhaps large) set of other historical days that were similar to the reference day from the perspective of traffic and weather characteristics. The methods described here would then take the set of TMI actions that were taken on the similar days and would construct a short but representative 'menu' of possible TMI actions that could be presented to the operator for further consideration. The menu could also display supporting historical information, such as the system-wide performance that each representative TMI action produced.

In general, we may consider the problem of taking a set of data and choosing a much smaller number of observations from that set to be 'representative' observations. This is primarily for the purpose of data exploration. If a user is presented these representatives, perhaps with extra information attached to each representative, the user should be able to get a good sense of the general shape or variety of the data. As far as the authors are aware, there is no existing published work that treats this exact problem. However, clustering algorithms have been used for this purpose in the past.

In our applications, it is important that the resulting representative data be strictly a subset of the original data. Some cluster algorithms characterize each cluster with a representative that is a member of the original dataset. These methods are the most analogous to our method, and results from these methods can be compared directly. Other cluster algorithms produce cluster representatives that are not members of the original dataset or do not produce cluster representatives at all. It is more difficult to see the relevance of these algorithms to the problem of finding representatives.

II. LITERATURE REVIEW

A. Representatives in Air Traffic Flow Management

There have been instances in the air traffic management literature in which authors attempted to create small instances that are representative of some larger dataset. In [1], a set of

Research supported by NASA grant no. NNX14AJ79A

ground delay programs were clustered using the *k*-means algorithm. The cluster centroids were treated as representatives and used in simulations. The *k*-means algorithm has been used in a similar fashion to produce vectors of hourly airports capacities with the intent of using these vectors as scenarios in simulations or in optimization models [2], and has also been used to find 'typical days' in terms of weather conditions [3]. There are some disadvantages to the *k*-means algorithm that our method attempts to remedy. For example, *k*-means is only applicable when Euclidean distance is used, and it produces centroids that are probably not in the original dataset.

B. Clustering

The authors are not aware of any literature that addresses the general problem of trying to find representative data points in a data set. The nearest methodological analogue is clustering, which has a vast and well-developed literature. The goal of clustering is to partition the data set so that similar points are placed in the same cluster while dissimilar points are placed in different clusters. Some methods produce a cluster representative for each cluster. The cluster representative, which may or may not be an element of the original dataset, is the most representative point of the cluster by some criteria. The set of cluster representatives could potentially be used as a list of representative data points. Some clustering algorithms produce no such representatives. The production of cluster representatives is not generally the goal of a clustering algorithm. Conversely, we do not find representatives with the intent of assigning each data point to a representative. Instead, the goal of finding representatives is to find the set of data points that would be most useful if all other data points were removed from consideration. We will not attempt to define what 'most useful' means, as this is inherently subjective and depends greatly on the application.

While a thorough review of the clustering literature is well beyond the scope of this paper, many good references exist. For example, see [4] or Chapter 8 of [5]. We will present two methods for finding representatives. The first of these methods involves forming a graph and solving a graph optimization problem. In this respect, it is similar to spectral clustering methods and other graph-based clustering methods. See [6] for a tutorial on spectral clustering, or [7] for a more general survey on graph-based methods. The optimization problem arising in our second method is equivalent to a method for clustering proposed in [8]. However, the results are applied in a different manner, and we propose a different solution method.

C. Minimum Dominating Sets and the k-center problem

In some cases, we will frame the problem of finding representatives as a *minimum dominating set problem*. Let there be a graph *G* with vertices *V*. A subset *S* of the vertices *V* is a dominating set if for any vertex *u* of *V* then either *u* is an element of *S* or *u* is adjacent to an element of *S*. A minimum dominating set is a dominating set that is of minimum size. There is a well-developed body of literature on minimum dominating sets from a graph theory and computational complexity perspective. Reference [9] gives a thorough treatment of this topic, as well as a long list of references. Alternatively, [10] provides an extensive bibliography. Some heuristics have been proposed and studied for the minimum dominating set problem ([11], [12], [13]). As far as we know, the minimum dominating set problem has not previously been used for clustering, nor has it been used for any problem closely resembling the problem of finding representatives.

Under other circumstances, we will instead propose the closely related *k*-center facility location problem. In this problem, the goal is to place *k* facilities to minimize the maximum distance that any customer would have to travel in order to reach the nearest facility. There are exact methods for the *k*-center facility location problem ([14], [15], [16], [17], [18]) as well as approximation algorithms and heuristics ([19], [20], [16], [21], [22], [23], [8], [24]). As mentioned above, [8] also proposed a clustering algorithm that approximately solves the *k*-center facility problem. We provide a different motivation for this problem, and we also provide a different manner to interpret the results.

III. METHODS

A. Minimum Dominating Set (MDS) Method

1) Similarity Graph

We begin by constructing a similarity graph from our data. The set of vertices contains one node for each data observation, and there is an edge between two observations if they are 'similar' in some sense chosen by the user. Some clustering methods such as spectral clustering also require a similarity graph, so there are some commonly-used methods for judging similarity. For example, [6] mention the following two methods:

- ε-neighborhood. We assume that we have a measure of distance $d(u,v)$ for observations *u* and *v*. Then, we say that *u* and *v* are similar if $d(u,v) < \varepsilon$ for some chosen positive real number ε. Common measures of distance include Euclidean distance or taxi-cab distance, possibly with some normalization or rescaling.
- *k*-nearest neighbors. Again, we assume we have a measure of distance between each pair of observations. For each point *u*, we find the *k* points that are closest to *u*. These are the *k*-nearest neighbors of *u*. Note that is possible for *v* to be a *k*-nearest neighbor of *u*, yet *u* not to be a *k*-nearest neighbor of *v*. This inspires two different similarity rules. In the first variation, we say two observations are similar if one is a *k*-nearest neighbor of the other, while in the second variation we say that two observations are similar if they are both *k*-nearest neighbors of each other.

In many datasets, the features may have completely different units. For example, one feature of a TMI is the time that the TMI was initiated and another feature is the number of flights that are allowed to access some resources in a given time period. For this reason, Euclidean distance or Taxicab distance is difficult to interpret, and we recommend the following similarity method:

- feature-wise ε-neighborhood. For each feature *f*, choose a positive, real number ε_f. Let u_f and v_f be the values of feature *f* from observations *u* and *v*, respectively. We say *u* and *v* are similar if and only if $|u_f - v_f| \le \varepsilon_f$ for all *f*.

The feature-wise ε-neighborhood is a special case of ε-neighborhood where the distance is given by rescaling features and then taking the infinity norm. We mention it separately because this distance measure is more difficult to interpret than the similarity relationship described here.

2) Minimum Dominating Sets

We would like our representative data points to include the variety present throughout the entire data set. For this reason, we require that every observed data point be similar to one of our representatives. There are many choices of representatives that would satisfy this requirement. For example, we could choose the entire data set to be our choice of representatives. In some sense, this would be the choice that best represents the original data. Of course, this choice is also completely unhelpful for a human researcher who wants to understand the data better. The smaller the number of representatives, the easier it will be for the human to examine the representatives. For this reason, we will search for the minimum set of representatives that satisfies the domination requirement. Let *G* be the similarity graph and let *V* be its vertices. Our problem is to find a subset *R* of *V* such that:

1. For any vertex *v* of *V*, then either *v* is an element of *R* or *V* is adjacent to an element of *R*.
2. For any subset *T* of *V* that satisfies (1.), then $|R| \le |T|$.

This is exactly the minimum dominating set problem. A solution to this problem is a minimum dominating set (MDS). The number of vertices in an MDS of a graph is known as the domination number. An MDS is not necessarily unique - any graph may have multiple minimum dominating sets. In the absence of any secondary discriminating criteria, we will accept any such set.

3) Solution of the MDS problem

The decision problem of whether a dominating set of size *k* exists in a general graph is known to be NP-Complete ([25]). Therefore, the MDS problem is NP-Hard and polynomial-time algorithms are unlikely to exist. Let *G* be a graph, let *V* be its vertices, and let *N*(*v*) be the neighborhood of the vertex *v* in *G*. We may find an exact solution for the MDS using the following integer program:

$$\min \sum_{v \in V} x_v$$

$$\text{such that} \quad x_v + \sum_{u \in N(v)} x_u \ge 1 \quad \forall v \in V$$

$$x_v \in \{0,1\} \quad \forall v \in V$$

The binary variable x_v takes a value of one if the vertex *v* is included in the set of representatives. The objective minimizes the size of the set, and the constraint enforces that each vertex is adjacent to a representative vertex. As we would expect in an NP-Hard problem, the solution to the LP relaxation may have fractional elements. This may render the exact solution of this IP infeasible for large problem instances. Alternatively, we could use one of several previously proposed heuristics discussed in the literature review section. These generally produce a dominating set, which is not necessarily minimal.

B. The k-Center method

Consider an alternative setting, where we have a distance score for each pair of points. We could form an ε-neighborhood similarity graph, as previously mentioned, but perhaps we are not sure what value we should choose for ε. Instead, we could choose a desired maximum number of representatives *k*. This is particularly appealing from a human factors point of view because an application interface might have limited space to display results and a human operator presented with an excessive number of representatives will become overwhelmed. This presents a natural constraint on the size of the representative set.

We could choose a value of ε and form an ε-neighborhood similarity graph, then use the MDS method to produce a set of representatives. Consider what happens as we change the value of ε. When ε is equal to 0 then the similarity graph is completely disconnected and the only MDS is the set of all data points. As ε increases, we add edges to the similarity graph and the size of a MDS decreases, until the similarity becomes completely connected and a MDS consists of a single point. The choice of ε allows us to balance the size of the set of representatives with how precisely the representatives describe the data.

The requirement that we have at most *k* representatives provides a desired level of conciseness. Presumably, we would like to find the most precise set of representatives that still satisfies the conciseness requirement. Since increasing ε will decrease precision and improve conciseness, we should choose ε to be the smallest value such that the MDS method produces at most *k* representatives.

As it turns out, the problem of finding the smallest ε such that the ε-neighborhood similarity graph has a dominating set of size *k* is equivalent to an existing problem known as the *k*-center facility location problem. This relationship was observed in [18]. In the *k*-center facility location problem, we have a set of points and we choose *k* points to be facilities. Our goal is to choose the facilities such that we minimize the maximum distance from any point to its nearest facility.

This provides an alternative interpretation of this method. We could measure how well a given point is represented by finding the minimum distance from this point to any representative. The maximum of all these distances would then be a measure of the quality of our representatives in general. We are then trying to find the set of representatives that minimizes this maximum distance. This problem was proposed as a method of finding cluster centers in [8].

1) Solution of the k-center problem

The *k*-center problem is an NP-Hard problem ([26]), so we do not expect that polynomial-time algorithms for solving the *k*-center problem exist. There are a number of heuristics, approximation algorithms, and exact algorithms mentioned in the literature review. We implemented the exact algorithm described in [15]. The solution times for this algorithm seem to be sufficiently fast for the purposes of our application, as the involved datasets are relatively small. However, heuristics would likely be necessary for large datasets.

2) Defining distance

Similarly to the MDS method, there is some question of how to define distance when some features are incomparable. We suggest the following method for numerical data. Let *X* be the set of observations. For each pair of observations *u* and *v*, and for each feature *f*, calculate the difference $|u_f - v_f|$. Sort the pairwise differences, and let $p_f(u,v)$ be the percentile of the pair (u,v) with respect to the difference $|u_f - v_f|$. That is,

$$p_f(u,v) = \frac{\left|\left\{\{x,y\} : x,y \in X, |x_f - y_f| \le |u_f - v_f|\right\}\right|}{\left|\{x,y\} : x,y \in X\right|}$$

Then, define the distance $d(u,v)$ by:

$$d(u,v) = \max_f p_f(u,v)$$

The distance $d(u,v)$ finds the feature in which *u* and *v* differ the most when compared to other pairs of observations. The distance then reports the percentile-rank of $d(u,v)$ in terms of that feature difference. The disadvantage of this method is that if we wish to find the distance between any pair of points, it requires finding all pairs of differences and sorting them. This produces a time complexity of $O(N^2 \log N)$, where *N* is the number of data points. In large datasets, this will not be computationally tractable. We can instead calculate the estimated percentile rank of each pair based on a sample of our data. Let *Y* be a set of data points sampled from *X*, either by bootstrapping or by choosing a random subset. Then, we can calculate the distances similarly as:

$$p_f(u,v) = \frac{\left|\left\{\{x,y\} : x,y \in Y, |x_f - y_f| \le |u_f - v_f|\right\}\right|}{\left|\{x,y\} : x,y \in Y\right|}$$

with the same distance metric as before, but applied to the estimated percentile ranks. This measure of distance is not affected by linear rescaling of the data. It is also robust to outliers, as an extreme observation in some feature will not greatly affect the percentiles of the other observations in that feature. These methods also allow us to easily interpret the solution to our *k*-representatives problem. Let ε^* be the distance in the solution. For a feature *f*, let δ_f be the ε^*-quantile of the feature *f*. Then we know that for each observation *x*, there is a representative *r* such that $|x_f - r_f| \le \delta_f$ for all features *f*.

3) Similarity scores

Consider alternatively a case where we have a similarity score $s(u,v)$ for each pair of points. We can define a similar *k*-center problem that we can solve with the same method. Let the ε-plus-neighborhood similarity graph be the graph such that *u* and *v* are adjacent if their similarity is greater than ε. We might wish to find the maximum ε such that the ε-plus-neighborhood similarity graph has a domination number of at most *k*. We can solve this problem using the same method we have previously described. Simply let *g* be any invertible, strictly decreasing function, and define the distance letting $d(u,v)$ equal $g(s(u,v))$. Then, we may proceed as previously described, and obtain an optimal distance value ε. We may then convert the optimal value ε back into a similarity value by finding $g^{-1}(\varepsilon)$.

C. Prevalence

The representatives that we produce are not necessarily all of equal importance. For some representatives, there may be many similar observations in our data set, while other representatives may have few similar data points. We define a measure, which we call *prevalence*, to reflect this.

In the MDS method, let *G* be the similarity graph. In the *k*-representatives method, take *G* to be the ε-neighborhood similarity graph produced by the optimal value of ε. Let *R* be the set of representatives. We may define the set *S*(*r*) to be the set of vertices adjacent to *r* in *G*. Since *R* is a dominating set, then each vertex must be included in *S*(*r*) for some vertex *r* in *R*. However, this is not necessarily a clustering, as some of the

sets may intersect. We define the prevalence of a representative r to be the value $|S(r)|$. This the number of observations that are similar to the representative r. Representatives whose prevalence is high can be interpreted as representing common cases, while representatives of low prevalence can be interpreted as representing unusual or anomalous cases. Including both possibilities is extremely important in the decision support context: the most common historical occurrences are not necessarily the best. Coupled with some exogenous performance data, a representative of low prevalence may be extremely important for the decision-maker.

IV. Properties of the MDS and *k*-center representative methods

A. Data coverages

The defining properties of the MDS and *k*-center representative methods are their coverage guarantees. The MDS method guarantees that each data point will be similar to a representative according to a chosen measure of similarity, and that we provide a smallest set of representatives that satisfies this condition. The *k*-center method guarantees that each data point will be within a distance of ε from a representative and that there is no choice of k representatives that would provide a smaller distance. The downside of this method is that NP-hard problems must be solved in order to provide these guarantees. It may not practical to solve these problems exactly on larger datasets. In order to gain computational efficiency, we must turn to heuristics. However, some coverage guarantees will still exist. For the MDS method, we may use heuristics that provide a small dominating set, in which case we may say that each data point is similar to a representative, although we cannot guarantee that a smaller such set does not exist. For the *k*-center method, we still will be able to provide an ε such that each data point is within ε of a representative. However, we would not be able to claim that we have attained the smallest possible distance ε.

B. Local Density

A key difference between our proposed methods and some existing centroid-based clustering algorithms is that the density of points within a small region of the data does not greatly affect the representatives that are produced. To make this notion clear, consider a data set X. Consider some data point x in X, and consider what would happen if we add data points $x_1, x_2, \ldots, x_n$ that take the same feature values as x but that are treated as separate observations. This addition would not affect the representatives produced by either the MDS method or the *k*-center method. However, in methods such as *k*-means or mean shift, the addition of the new data points will draw the centroids closer to the point x. Thus, we expect that the representatives of our methods will be spread evenly across the regions where points occur, regardless of how many points occur locally in those regions, while many other clustering methods will tend to place more centroids in higher-density regions. In some applications, it may be desirable to have more representatives from the high density regions. We believe that in our application, there is more benefit from having a more general coverage of the data at the cost of less representation in the higher-density areas. However, practitioners should use their best judgment as to whether or not this would be appropriate in their application.

C. Data Membership

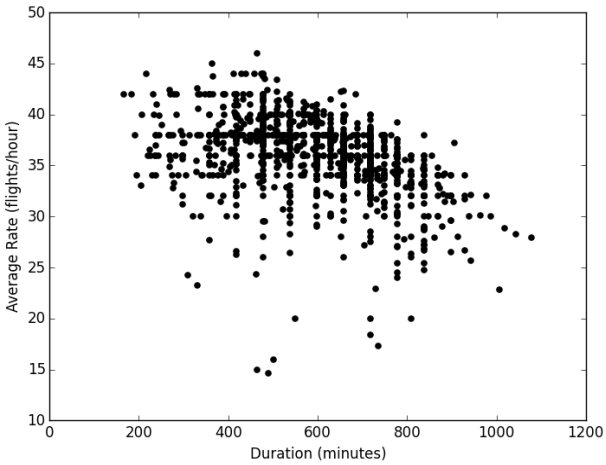

Figure 1: Ground Delay Program Features at EWR

Our method will always produce representatives that are members of the original dataset. This is true of some existing clustering methods such as affinity propagation, but other methods such as *k*-means or mean shift often produce cluster representatives that are not in the original dataset. Other existing clustering methods including spectral clustering and hierarchical clustering do not even produce representatives. This property is especially useful when some features are numerical, but are restricted in the values they can take. For example, some features may take only integer values. Our method will always produce valid data points in these cases.

D. Flexibility in modeling

A practitioner has a choice of similarity in the MDS method or a choice of distance in the *k*-center method. Some existing clustering methods also allow this, including many hierarchical clustering methods, spectral clustering, and affinity propagation. Other clustering algorithms such as *k*-means and mean shift do not. In our application, Euclidean distance is very difficult to interpret because the features have disparate units. For this reason, it is important to be able to use other measures of distance.

V. Example: Ground Delay Programs at EWR

We will compare our method to existing clustering algorithms that produce representatives. Our dataset contains a

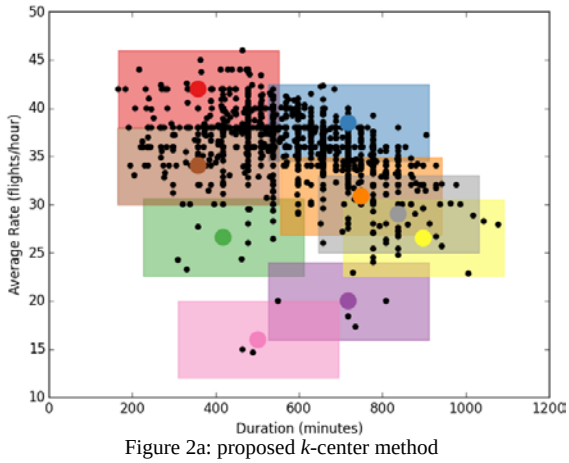

Figure 2a: proposed *k*-center method

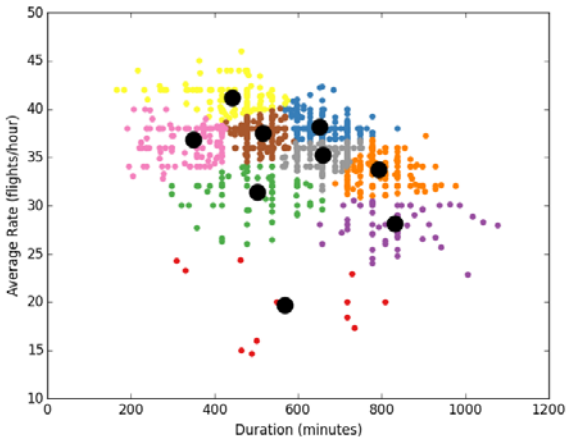

Figure 2b: *k*-means

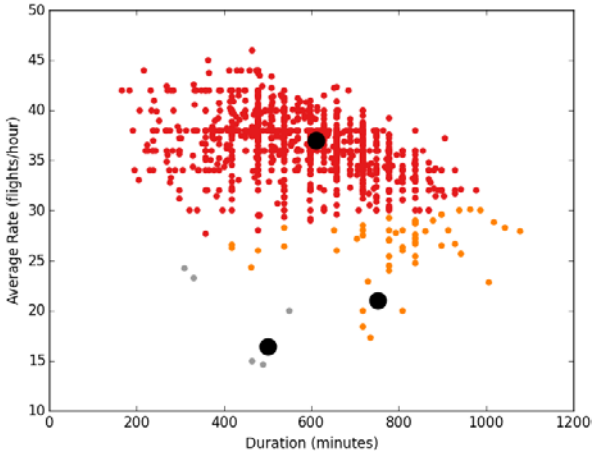

Figure 2c: mean shift

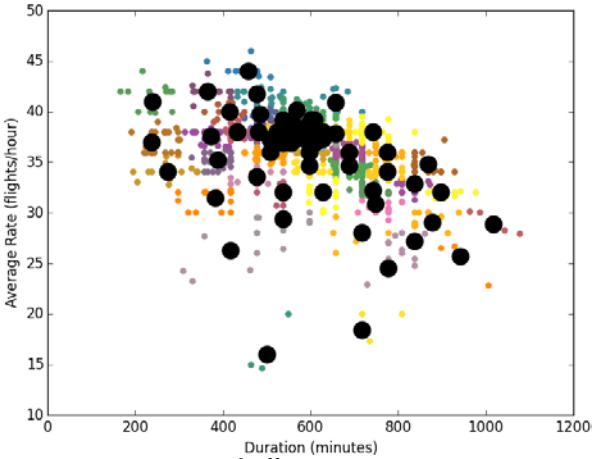

Figure 2d: affinity propagation

Figure 2: comparison of methods for producing representatives

set of ground delay programs (GDPs) from the FAA advisory database. A GDP restricts the rate at which flights are allowed to arrive at an airport by delaying flights pre-departure, thus reducing congestion at the arrival airport. In order to easily visually examine our dataset, we will restrict our attention to two parameters. The first is the average number of flights allowed to arrive per hour at the destination airport, which we will refer to as the average rate. The second feature is the number of minutes during which arrivals to the airport will be restricted. The dataset we will use consists of every GDP that appears in the FAA advisory database from January 1st, 2007 to December 31st, 2014. The FAA may revise GDPs after they are issued. We do not include these revisions in our dataset, so our features consist of the first parameters associated with each GDP. There are a total of 1302 GDPs in the dataset.

A plot of the dataset is shown in figure 1 on the previous page. Based on visual examination, the data set seems to consist of a single cluster. This cluster consists of a high-density core, surrounded by lower density regions. We implemented an exact *k*-center method ([15]) in Python and applied it to this data. We also ran the *k*-means, mean shift, and affinity propagation clustering algorithms, using the implementations in the Scikit-learn package for Python ([27]). These results are shown in figure 2, shown above.

The distance measure for the *k*-center method was calculated according to the percentile-rank method described in section 3. 2. 2. The affinity propagation method requires a similarity score $s(x, y)$ for each pair of points x and y in the dataset X. We set $s(x, y)$ to be $1 - d(x, y)$, where $d(x, y)$ is the distance according to the percentile-rank method. The affinity propagation method also requires an input for each point called the preference. Common choices for the preference include the minimum or median of the similarity scores. We tried both options. When the preference was set to the minimum value of the similarity score then all data points were placed in a single cluster, so we instead show the results with the preference set to the median similarity score. We chose k equal to 9 for the *k*-center method and the *k*-means algorithm, as we feel this would be a reasonable number of

representatives to present to a decision-maker in our application. All other parameters in all methods were left to the default settings. In all cases, we rescaled each feature as a *z*–score before applying clustering. This scaling was then reversed to produce the resulting figures.

For the clustering algorithms, the cluster representatives are shown as large black points and the non-representative cluster members are shown as smaller points whose color corresponds to their cluster membership. The representatives produced by our proposed *k*-center method are shown as large colored points. The colored rectangle surrounding a representative gives the region of points that are within the optimal coverage distance from the representatives.

Neither the mean shift algorithm nor affinity propagation produce results that are well-suited for our application. The mean shift algorithm provides only three representatives. The majority of the high-density region is represented by a single point, which does not give many options for a controller attempting to make TMI decisions. The affinity propagation clustering method produces an overwhelming number of representatives. This would be difficult for a practitioner to consider. Potentially, we could choose a different value for the preference that could produce a more amenable set of representatives, but it is not clear what this choice of preference would be.

The *k*-center and *k*-means algorithms produce a more reasonable number of representatives, which seem to be spread reasonably across the areas where data occur. There are two main differences between the representatives produced by the *k*-means clustering as and those in our proposed *k*-center method. First, the *k*-center method produces representatives that are in the original dataset, while *k*-means does not. The representatives *k*-means produces may not be valid GDP parameter choices, which makes it harder to use these to make TMI decisions. The second difference is that the *k*-center method has more coverage of the lower density regions and less coverage of the higher density. This is not necessarily an advantage in all applications. We believe that in the case of our application, this will provide TMI decision-makers with a wider variety of options to consider and a more thorough knowledge of what actions have been taken in the past. We hope that this will allow them to make better decisions.

VI. Conclusions and Further Work

We discussed the problem of finding a set of representatives from a set of data points, and we proposed two related methods for this problem. The MDS method was developed for this problem, while the *k*-center method had previously been proposed for clustering. The representatives produced by these methods are always members of the original dataset. Along with the representatives, the methods produce coverage guarantees. If the dataset is small enough that we may use exact methods, then the coverage guarantee is also optimal in some sense. On larger datasets, we still produce coverage guarantees, but they are only approximately optimal. The MDS method is resistant to outliers. While the *k*-center method is not as resistant to outliers, it may be used to detect and remove outliers. Compared to *k*-means clustering, the *k*-center method will tend to provide better coverage of the low-density regions of data while providing sparser coverage of the high-density regions.

We also proposed a method of determining whether two points are similar and some methods of assigning a distance to two points in a dataset. These methods provide more interpretable results than existing methods in the case when units of separate features are drastically different. We suggest a measure of prevalence of a representative to aid in interpretation of the results produced by our methods.

The *k*-center method may be solved exactly on small datasets, and existing heuristics are scalable to moderately large datasets. However, existing heuristics for the *k*-center facility location problem tend to assume that the distance between any pair of points may be computed. This would not be tractable for larger datasets. Thus, we would need to develop new heuristics for the *k*-center method if we wish to apply it to extremely large datasets. Similarly, the heuristics for the MDS problem have been studied for graphs that are relatively small compared to the size of many large datasets. It may also be necessary to develop new heuristics to solve the MDS problem on these larger datasets.

While we understand the qualitative differences between the results of our methods as compared with other methods, there is more that we could do to validate how well our methods perform. In the case of our TMI application, we could solicit feedback from subject matter experts about which sets of representatives would be more useful.

It is clear how our method may be applied to a set of TMIs that are all of the same type. If we are given a list that only contains GDPs, we may apply this method immediately. However, there are other types of TMIs that are used, such as ground stops and airspace flow programs. Each TMI may also be revised, extended or cancelled. More study would be necessary to provide a method that would be applicable to lists of TMIs that include multiple types of TMIs or that include TMIs that are modified over time.

There is also more work that we could do after the representatives have been generated in order to make them more useful to a TMI decision-maker. For example, we could construct an estimate of the historical performance of a representative TMI based on the TMIs that are with the optimal *k*-center distance. We hope to implement these features in a decision-support tool that would make it easier to examine historical TMIs. This tool would then be used to aid the planning of TMIs as well as to aid in the review of a TMI after it has been implemented.

References

[1] L. Delgado, X. Prats and B. Sridhar, "Cruise speed reduction for ground delay programs: A case study for San Francisco International Airport arrivals," *Transportation Research Part C: Emerging Technologies,* vol. 36, pp. 83-96, 2013.

[2] P. -c. B. Liu, M. Hansen and A. Mukherjee, "Scenario-based air traffic flow management: From theory to practice," *Transportation Research Part B,* vol. 42, no. 7, pp. 685-702, 2008.

[3] S. Grabbe, B. Sridhar and A. Mukherjee, "Similar days in the NAS: an airport perspective," *AIAA Aviation Technology, Integration, and Operations Conference,* 2013.

[4] A. K. Jain, N. M. Murty and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR),* vol. 31, no. 3, pp. 264-323, 1999.

[5] P. -N. Tan, M. Steinbach and V. Kumar, Introduction to data mining, Pearson Addison Wesley Boston, 2006.

[6] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing,* vol. 17, no. 4, pp. 3953-416, 2007.

[7] S. E. Schaeffer, "Graph clustering," *Computer Science Review,* vol. 1, no. 1, pp. 27-64, 2007.

[8] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science,* vol. 38, pp. 293-306, 1985.

[9] T. W. Haynes, S. Hedetniemi and P. Slater, Fundamentals of domination in graphs, CRC Press, 1998.

[10] S. T. Hedetniemi and R. C. Laskar, "Bibliography on domination in graphs and some basic definitions of domination parameters," *Annals of Discrete Mathematics,* vol. 48, pp. 257-277, 1991.

[11] C. K. Ho, Y. P. Singh and H. T. Ewe, "An enhanced ant colony optimization metaheuristic for the minimum dominating set problem," *Applied Artificial Intelligence,* vol. 20, no. 10, pp. 881-903, 2006.

[12] L. A. Sanchis, "Experimental analysis of heuristic algorithms for the dominating set problem," *Algorithmica,* vol. 33, no. 1, pp. 3-18, 2002.

[13] A. K. Parekh, "Analysis of a greedy heuristic for finding small dominating sets in graphs," *Information processing letters,* vol. 39, no. 5, pp. 237-240, 1991.

[14] D. Chen and R. Chen, "New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems," *Computers & Operations Research,* vol. 36, no. 5, pp. 1646-1655, 2009.

[15] S. Elloumi, M. Labbe and Y. Pochet, "A new formulation and resolution method for the p-center problem," *INFORMS Journal on Computing,* vol. 16, no. 1, pp. 84-94, 2004.

[16] C. Caruso, A. Colorni and L. Aloi, "Dominant, an algorithm for the p-center problem," *European Journal of Operational Research,* vol. 149, no. 1, pp. 53-64, 2003.

[17] T. Ilhan, F. A. Özsoy and M. C. Pinar, "An efficient exact algorithm for the vertex p-center problem and computational experiments for different set covering subproblems," Bilkent University, Department of Industrial Engineering, 2002.

[18] E. Minieka, "The m-center problem," *Siam Review,* vol. 12, no. 1, pp. 138--139, 1970.

[19] T. Davidovic, D. Ramljak, M. Selmic and D. Teodorovic, "Bee colony optimization for the p-center problem," *Computers & Operations Research,* vol. 38, no. 10, pp. 1367-1376, 2011.

[20] B. Robic and J. Mihelic, "Solving the k-center problem efficiently with a dominating set algorithm," *CIT. Journal of computing and information technology,* vol. 13, no. 3, pp. 225-234, 2005.

[21] R. Hassin, A. Levin and D. Morad, "Lexicographic local search and the p-center problem," *European Journal of Operational Research,* vol. 151, no. 2, pp. 265-279, 2003.

[22] N. Mladenovic, M. Labbe and P. Hansen, "Solving the p-Center problem with Tabu Search and Variable Neighborhood Search," *Networks,* vol. 42, no. 1, pp. 48-64, 2003.

[23] J. Mihelic and B. Robic, "Approximation Algorithms for the k-center Problem: An Experimental Evaluation," in *Operations Research Proceedings 2002*, Berlin, Springer, 2002, pp. 371-376.

[24] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of operations research,* vol. 10, no. 2, pp. 180-184, 1985.

[25] D. Johnson, Computers and Intractability-A Guide to the Theory of NP-Completeness, San Fransisco, CA: Freeman, 1979.

[26] D. S. Hochbaum, "When are NP-hard location problems easy?," *Annals of Operations Research,* vol. 1, no. 3, pp. 201-214, 1984.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.