Clustering of Power System Data and Its Use in Load Pocket Identification

Katherine M. Rogers
University of Illinois
krogers6@illinois.edu

Thomas J. Overbye
University of Illinois
overbye@illinois.edu

Abstract

When lines in a power system are constrained, the sensitivity of the power flows on these lines to generator output provides information about how the constraints divide the system and about the ability of sets of generators to increase revenue without increasing dispatch. Clustering is used to identify generators into groups with the potential for market advantage. In this paper, we discuss the implementation of several different clustering methods for identifying load pockets with potential market advantage.

1. Introduction

Constrained lines in a power system can divide the system in the sense that generation may become non-substitutable and “load pockets” may form [1]. Because of the transmission constraints, the power to supply the load in the load pocket must come from a generator within the load pocket. Thus, generators in the load pocket have the opportunity to increase prices without changing dispatch and can increase revenue [2]. Suppliers with this ability can enjoy some isolation from market competition [3]. We will refer to this ability as load pocket potential (LPP).

In this paper, we build from the analysis in [4], which presents a two-stage algorithm for identifying generators with LPP. We seek suitable clustering methods for use in the algorithm’s first stage. The clustering tools must successfully find LPP groups, and the analysis must be completed quickly, even for large systems.

This paper begins with a brief background on market power, and in Section 3 the LPP algorithm of [4] is summarized. A metric is established to quantify the LPP of clusters which are found. In the subsequent sections, several clustering algorithms to identify LPP groups are described. Results are discussed for several systems.

2. Market power potential background

The standard definition of market power is the ability of a market participant to profitably maintain prices above a competitive level for a significant period of time [5]. There are many relevant considerations when assessing the competitiveness of an electricity market [6]. Several factors which are not captured by traditional measures are described in [7] and include price-responsiveness of competitors and consumers.

A single market can at times exhibit very little market power whereas at other times it can exhibit considerable amounts [7]. The transition of a competitive system to a system with market power occurs when demand rises to a level which is so high that producers can no longer compete with each other to supply the load. In the electric industry, this effect is enhanced by the limited production capacities of producers, congestion in the transmission system, and the lack of sufficient electricity storage [7]. A participant does not need to have a large market share to have market power; it may be the case that no other firm is able to replace the supply of one firm because of system constraints.

The potential of generators in load pockets to experience market advantage is explained in [4], and the definitions are based on [9]. We wish to identify suppliers with the ability to increase revenue by raising prices while not affecting the revenues of the remaining participants. Analysis can be done to identify situations where the following conditions hold: (1) An increase in nodal price at certain generators (2) results in an increase in revenue at the same generators while (3) not impacting the revenue at other generators. Condition (3) can be represented by changes in revenue which are equal to zero at all buses not exercising market power. We enforce condition (3) by ensuring that dispatch and price are unchanged for all other buses, so changes in revenue for those buses must equal zero. Dispatch is unchanged at all buses because price perturbations we consider are in the null space of the dispatch sensitivity matrix. In particular, the algorithm seeks price perturbation vectors where price is unchanged outside of the LPP group. Thus, generators outside of the LPP group should be unaffected. The sets of generators which satisfy the above conditions have LPP by acting together.
3. Overview of load pocket identification algorithm

The sensitivity-based two-stage algorithm involved in this work is presented in [4]. Constrained-line sensitivity information is used to highlight certain vectors that show LPP concentrated in the fewest entries. For completeness, we summarize the algorithm in this section before proceeding to discuss clustering algorithms.

The relation between generator dispatch, $\Delta g$, and price perturbations, $\Delta y$, near an operating point is represented by the linear sensitivity matrix, $M$:

$$\Delta g = M\Delta y$$  \hspace{1cm} (1)

Further discussion of $M$ can be found in [8], [3], [9], and [10]. The matrix $M$ is singular; there exists a price perturbation vector $\Delta y$, such that there is no change in dispatch,

$$0 = M\Delta y$$  \hspace{1cm} (2)

where vector $\Delta y$ is in the null space of $M$, $\text{Nul}(M)$. LPP can be identified by finding price perturbation vectors that satisfy (2). The dimension of $\text{Nul}(M)$ increases with the number of transmission constraints. The basis for $\text{Nul}(M)$ for a system with $(m-1)$ line constraints will have up to $m$ linearly independent vectors. A treatment of this property is found in [1].

Suppose $B$ is an orthonormal basis for $\text{Nul}(M)$. Any linear combination $x$ of the columns of $B$, given by a price perturbation vector $\Delta y$, is also in $\text{Nul}(M)$ [11], a fact which is stated as follows:

$$\Delta y = Bx \in \text{Nul}(M)$$  \hspace{1cm} (3)

There are infinite vectors in $\text{Nul}(M)$, but only a certain form is of interest. The desired form of $\Delta y$ consists of a few large entries for generators in a LPP group and mostly zero or near-zero entries otherwise. The problem is thus how to determine $x$ to give $\Delta y$ of the desired form. The basis $B$ of $\text{Nul}(M)$ is the same as the basis for the augmented transmission constraint sensitivities $S$; the proof is given in [4].

3.1. Clustering stage (stage 1)

The augmented sensitivity matrix for a particular set of constraints is obtained from the linear programming optimal power flow (LP OPF) tableau, and its basis $B$ is found. Methods to find a basis include Gaussian elimination, the Modified Gram-Schmidt (MGS) process, QR factorization processes, and Singular Value Decomposition (SVD). These operations are discussed in [12]. We use SVD, which is of order $O(m^2)$ [12], where $m$ is small because the number of constraints is small.

The rows of $B$ are clustered to obtain groups of generators. Each row represents a generator and the column values are based on the binding constraints. Clustering reflects the fact that generators with similar impact on the constrained lines may be able to jointly raise their prices without affecting dispatch. To see this, consider a generator in a small area which is connected by only one line to the rest of the system, and suppose the line is constrained; such a generator is in a load pocket. Clustering extends this notion to identify groups that are not as obvious. Euclidean distance indicates similarity between objects,

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$  \hspace{1cm} (4)

and is used by most algorithms in this work. The algorithms can easily be modified to use other metrics. Correlation between two vectors, $\cos(x,y) = x \cdot y / (\|x\|\|y\|)$  \hspace{1cm} (5)

may also be used. In both (4) and (5), $x$ and $y$ designate two row vectors of $B$.

We denote the output from this stage as a $\text{ClusterVector}$, a vector with length equal to the number of generators and entries equal to the number of the cluster to which each generator belongs.

3.2. Price perturbation vector stage (stage 2)

A set of clusters in a $\text{ClusterVector}$ is called a clustering. Once a particular clustering of generators is obtained, the second stage of the algorithm finds price perturbation vectors of the desired form for each cluster. For each cluster $i$, we partition $B$ into a matrix $B_i$ consisting of the rows of $B$ which are in $i$, and a matrix $B_{-i}$ consisting of all other rows. Subscript $-i$ denotes generators not in group $i$:

\[
\begin{bmatrix}
B_i \\
B_{-i}
\end{bmatrix}
\begin{bmatrix}
x_i \\
x_{-i}
\end{bmatrix}
= 
\begin{bmatrix}
\Delta y_i \\
\Delta y_{-i}
\end{bmatrix}
\]  \hspace{1cm} (6)

The elements in $\Delta y_i$ should be much larger than those of $\Delta y_{-i}$, and this is done by design of an appropriate choice of $x$. Ideally, the elements of $\Delta y_{-i}$ are zero. The problem to find such an $x$ may be written as follows,

\[
\begin{array}{ll}
\text{Max} & \Delta y_i^T \Delta y_i - \Delta y_{-i}^T \Delta y_{-i} \\
\text{st} & \|x\| = 1
\end{array}
\]  \hspace{1cm} (7)

where (6) is substituted into (7). The maximum is obtained by choosing $x$ to be the eigenvector corresponding to the largest eigenvalue $\lambda_{\text{max}}$ of $B_i^TB_i$, $B_{-i}^TB_{-i}$ for each cluster $i$. The proof follows from the
definition of eigenvalues and eigenvectors and is given in [4]. Then, when generators adjust their prices according to $\Delta y = Bx$, they can earn revenue without affecting dispatch.

3.3. LPP group evaluation metric

Results of the load pocket identification problem (LPP) can be described in terms of the price perturbation vectors. It is said that load pocket potential exists for a given scenario if we can find a clustering of generators, where for at least one cluster $i$, the objective function given by (7) evaluated at the optimal value of $x$ produces well-separated price perturbation vectors. The entries in a price perturbation vector are well-separated if entries $\Delta y_i$ are larger than the entries $\Delta y_i$. Generators in such a cluster are said to have LPP. To measure this quality of the clusters, we define a metric denoted $f_{LPP}$,

$$f_{LPP} = \frac{\Delta y_i^T \Delta y_i}{N_i} - \frac{\Delta y_i^T \Delta y_i}{N_i},$$

(8)

where $N_i$ denotes the number of generators in the LPP group $i$ and $N_i$ denotes all other generators.

The $f_{LPP}$ takes values between -1 and 1. For LPP identification, we are interested in the values which are nearest to 1. A value of $f_{LPP}=1$ represents the case where all generators in the group have price perturbation entries of 1 while generators outside the group have entries of 0. Conversely, $f_{LPP}$ near 0 indicates that the price perturbation vector is not well-separated, meaning that generators outside of the LPP group have non-zero price perturbations, some of which may be large.

We use the $f_{LPP}$ to evaluate cluster method effectiveness in Section 5. Each clustering method produces a clustering, and for each cluster, a price perturbation vector $\Delta y$ is found, and (8) is evaluated. The $f_{LPP}$ provides information about the extent to which an algorithm has identified load pockets, so it is important for our application, but it should be noted that an assortment of more general metrics to quantify clustering quality are also available [13].

All price perturbation vectors found from this algorithm result in no change in dispatch [4]. Generators outside of the LPP cluster are assumed not to change their prices, which is why we search for vectors with entries in $\Delta y$ which are zero. However, they are not always exactly zero. Experimentally, we have found that even when these entries are not exactly zero, we can often approximate them as zero by implementing the price perturbations only for the generators in the LPP cluster; the result still produces no change in dispatch. Thus, rather than requiring $f_{LPP}$ exactly equal to 1 for LPP to exist, values slightly less than 1 may also be acceptable. In the examples in Section 5, results are shown for $f_{LPP} > 0.9$. Presumably, there is some threshold where, if entries in $\Delta y$ become too large, they will no longer be negligible in this manner. However, more research is needed to identify this threshold by examining the range over which price perturbation vectors of the desired form exist.

4. Algorithms for clustering

Several algorithms for generator clustering are presented here, and results for LPP identification are presented later in Section 5. These algorithms may be easily applied to other problems. One application of clustering in power systems is to identify coherent generators to create dynamic equivalents [14]. Related applications include the classification of contingencies [15][16], the clustering of load signatures for non-intrusive load monitoring [17], and the classification of power system faults [18][19].

4.1. Hierarchical clustering

The first clustering method we examine is hierarchical clustering. Hierarchical clustering may begin with all items as individual clusters and successively merge the clusters, or it may begin with all items in one cluster which it successively divides. The former type is referred to as agglomerative clustering, and the latter is divisive. Agglomerative is the more common type.

Agglomerative clustering algorithms are “bottom-up” in the sense that row of $B$ (generator), starts out in its own cluster. At each layer in the hierarchy, clusters merge together. At the top and final layer, there is only one cluster to which all generators belong. Thus, we may slice the hierarchy at desired levels to obtain particular clusterings. Each level is defined by a distance threshold, where all items which are closer than this threshold to other items have been clustered.

There are different types of agglomerative clustering, differentiated by the way clusters are combined. We discuss single-linkage cluster analysis (SLCA). Then, we discuss minimal spanning trees and their relationship to SLCA. In SLCA, the distance between two clusters is the minimum distance between its elements. A known issue with SLCA is the chaining effect: due to the way clusters are formed, it is possible for two items in the same cluster to be further apart than two items in different clusters.

The main appeal of hierarchical methods is that they are flexible and non-parametric, meaning that they can be applied without having to choose
parameters. Thus, hierarchical clustering can be applied when little or nothing is known about the relationships in the input data, so it can be used for an instructive first stage of analysis, which may stimulate further analysis.

4.2. Nearest-neighbor searches

Hierarchical clustering provides more information than a single clustering, but it can also take longer. A naive implementation can have computational complexity of $O(n^3)$. Generally, however, hierarchical clustering has complexity $O(n^3)$ by applying some minor improvements to the $O(n^3)$ implementation. $O(n \log n)$ can be achieved using careful implementations and special data structures [20]. The need to compute distances between all points is a main factor in the computational cost. Thus, nearest-neighbor searching is an important sub-problem that must be carefully handled. Much effort has been devoted to making these searches as fast as possible and reducing the number of searches needed.

We implement and utilize a data structure called a kd-tree to partition the items in the k-dimensional coordinate space. Kd-trees are introduced in [21] and are efficiently used in [22] to find the $m$ nearest-neighbors of a given query point. There are several variants on kd-trees and the associated algorithms, so we describe our implementation.

A kd-tree is a binary search tree in multiple dimensions. An kd-tree for points (2,3), (5,4), (9,6), (4,7), (8,1), and (7,2) is shown in Figure 1:

![Figure 1. Example kd-tree](image)

Each level of the kd-tree corresponds to one of the k-dimensions. In Figure 1, there are two dimensions. The first level splits the first dimension, the second level splits the second dimension, and the third level splits the first dimension again. The tree is formed recursively by cycling through the axes of the data and splitting the data at the median point for each axis. Since the median value is used to split the data, the tree is balanced by design. The kd-tree can be built in $O(n \log n)$ time. Its construction is outlined in Algorithm 1.

**Algorithm 1. Recursive kd-tree construction**

Kd-trees support several types of queries [21]. We are most interested in efficient nearest-neighbor searches, in [22]. One important consideration is when the query point itself is a point in the kd-tree. If the query point is in the tree, using a true nearest-neighbor search would simply find and return the query node, since the distance between the query node and the node found in the tree will be zero. However, we actually are interested in returning the nearest neighbor to the query point which is not the query point. Another case of interest is when we require $m$ nearest-neighbors instead of one. The single-nearest-neighbor search is a special case of this more general situation, given in Algorithm 2.

**Algorithm 2. Recursive m-nearest-neighbor search using kd-trees**
The special consideration when the query point is in the tree can be accounted for by searching for \( m = 2 \) nearest-neighbors; then, the nearest-neighbor will be the query point and the second nearest-neighbor will be the answer.

4.3. Minimal spanning trees for clustering

Consider a set of nodes and a set of branches connecting the nodes. For LPP identification, each node corresponds to a row of \( B \), and branches are added in order to connect the nodes by the least total distance. The minimal spanning tree (MST) is the subset of the branches of a graph which has the minimum total distance while providing a route between every pair of nodes [23]. We discuss the use of minimal spanning trees for SLCA. The relationship between minimal spanning trees and SLCA is proven in [24] and is also recognized in [25] and [26].

Fast algorithms for constructing minimal spanning trees in coordinate spaces are discussed in [23]. We implement an approximate minimal spanning tree algorithm which uses kd-trees to perform efficient nearest-neighbor searches.

To discuss the minimal spanning tree algorithm and its usefulness in SLCA, we must first introduce the concepts of fragments and nearest-neighbor chains. A fragment is a part of the final MST. A nearest-neighbor chain is one approach for forming fragments, and it builds the fragment towards increasing density. Sections of \( k \)-dimensional space with points closer together have a higher density.

A more formal definition of nearest-neighbor chains is now given [20]. A nearest-neighbor chain is a set of points whose directed edges are such that

\[
i, j = \text{NN}(i), k = \text{NN}(j), \ldots, q = \text{NN}(p), p = \text{NN}(q)
\]

where \( \text{NN}(i) \) indicates the nearest-neighbor of point \( i \). If \( q = \text{NN}(p) \) and \( p = \text{NN}(q) \), \( p \) and \( q \) are said to be reciprocal nearest-neighbors (RNNs). The final two points in a NN-chain form a RNN pair, and the beginning point \( i \) is any arbitrary point. Distances between points in a NN-chain are monotonically non-increasing. That is, NN-chains are grown towards increasing density. To see why this must be true, suppose we are forming a NN-chain which so far has \( i, j = \text{NN}(i) \). Suppose \( d_{ij} \) is the distance between \( i \) and \( j \). The nearest-neighbor to \( j \) cannot be further away than \( d_{ij} \), because then \( j \) would be the nearest-neighbor of \( i \), and this would make \( i \) and \( j \) RNNs and would indicate the end of the NN-chain.

Consider the set of points shown in Figure 2 as a simple example:

![Figure 2. 1-D example, set of points](image)

This is the simplest type of example to visualize since it is one-dimensional. A nearest-neighbor chain from Point A consists of A->B->C->D, and a nearest-neighbor chain from Point H contains H->G->F->E. These two fragments are shown in Figure 3.

![Figure 3. 1-D example, two fragments](image)

Then, the final MST is formed by linking together the existing nearest-neighbor chains. The two closest points on the two fragments are D and E, so connecting D and E forms the final MST shown in Figure 4.

![Figure 4. 1-D example, MST](image)

The SLCA hierarchy for the MST of Figure 4 is shown by the dendogram in Figure 5.

![Figure 5. 1-D example, dendogram](image)

Algorithm 6 in [20] is called the multiple-fragment algorithm and is based on the minimal spanning tree algorithm (MST) and approximate minimal spanning tree algorithm (AMST) of [23]. The algorithm builds NN-chains, then merges each fragment with its nearest-neighbor fragment until all are connected. If there are \( L \) fragments, there are \( L-1 \) links to be added to obtain the MST. The AMST is introduced with the perspective that in statistical and clustering applications, the exact MST is not required. In clustering situations, the distances between items in a cluster tends to be smaller than the distances of items in different clusters. The AMST algorithm does not guarantee that the link it finds to connect two fragments will be the smallest, but if the link is not the smallest, its length is “nearly the same” as the smallest one, so the spanning tree is “very nearly minimal.”

As the algorithm proceeds, the obtained fragments are essentially clusters, so the links used to connect these fragments may be unnecessary. In general, it is practical to stop after clusters have attained some desired size. Our implementation is described by Algorithm 4.
The Quality Threshold (QT) and K-Means clustering algorithms are also used in [4]. The QT algorithm was developed to cluster genes [27]. The developers desired the ability to form an unknown number of potentially large clusters with cluster diameters beneath a certain threshold. The computation of the distance between every point to every other point is required, so its computational order is \(O(n^2)\) [28]. A threshold and optionally a maximum cluster size must be specified. Points are first placed into temporary clusters with points nearer than the threshold. After each iteration, the largest temporary cluster becomes a true cluster. The process continues until all points belong to a true cluster.

### 4.5. K-Means clustering

The K-Means clustering technique is one of the oldest and most widely used algorithms [29] and is simple and fast in practice [30]. Initial points are chosen as cluster centers. Each point is assigned to its nearest cluster center, and the centers are recomputed. This process continues until points stop changing clusters.

The K-means algorithm requires the user to specify the number of clusters, \(k\). If there are \((m-1)\) constrained lines in the system, we default to the choice of \(k=m\) clusters. Choosing \(k=m\) works very well in most cases. If there is one constrained line, expecting two groups to form is a good initial guess. Another drawback of K-Means clustering is that different initial clusters may lead to different results. Currently, for each column, we find the row with the greatest absolute-value element, and choose this to be an initial cluster. If \(k\) is greater than the number of columns, we choose more points until \(k\) points are added.

### 4.6. Coupling Index (CI) clustering

The Coupling Index (CI) algorithm is originally presented in [31] for use in the identification of reactive support groups. The coupling index (CI) is given in \((5)\). The CI has values between -1 and 1. When the CI has an absolute value of 1, the angle between the vectors is zero, and there is complete correlation, either positive or negative. When the CI is zero, the vectors are orthogonal or independent. Algorithm 4 forms clusters based on correlations of generator impact on constrained line flows. More refinement to this algorithm is possible.
If $A_i$ and $A_j$ match then begin
$A_j$ is a strong cluster
End
End

Algorithm 4. CI clustering algorithm

5. Examples and results

Results of the clustering algorithms for systems of different sizes are presented. For each scenario, the $f_{LPP}$ (8) is calculated for each cluster in the particular clustering. Values of $f_{LPP} > 0.9$, indicating LPP, are reported in the tables. Each algorithm is a row in the tables, specified by name. For QT, the threshold is specified. For K-Means and SLCA, the number of clusters is specified. Then, we discuss the results in context of the different clustering algorithms in Section 5.5.

5.1. Seven-bus case

A seven-bus system is shown in Figure 6. The system has binding constraints on lines (2,6) and (5,7). Results are shown in Table 1.

### Table 1. Seven-bus clustering results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>No. Total Clusters</th>
<th>No. of Clusters with $f_{LPP}&gt;0.9$</th>
<th>$f_{LPP}$ of Clusters &gt; 0.9</th>
<th>Gens in Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>QT, 0.25</td>
<td>3</td>
<td>3</td>
<td>0.995</td>
<td>6</td>
</tr>
<tr>
<td>Qt, 0.1</td>
<td>4</td>
<td>2</td>
<td>0.995</td>
<td>6, 4, 2, 1</td>
</tr>
<tr>
<td>K-Means, 3</td>
<td>3</td>
<td>3</td>
<td>0.995</td>
<td>6, 7</td>
</tr>
<tr>
<td>K-Means, 4</td>
<td>4</td>
<td>2</td>
<td>0.995</td>
<td>6, 7, 4, 1</td>
</tr>
<tr>
<td>CI</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1, 12, 4</td>
</tr>
<tr>
<td>SLCA, 3</td>
<td>3</td>
<td>3</td>
<td>0.995</td>
<td>6, 7</td>
</tr>
</tbody>
</table>

Three clusters with $f_{LPP}>0.9$ are found. However, K-Means with four clusters and QT with a 0.1 threshold both miss the (4,2,1) cluster.

5.2. IEEE 118-bus case

The IEEE 118-bus system [32] is shown in Figure 7 below. In this scenario, lines (5,8), (23,32), and (65,68) are binding. Results are given in Table 2.

### Table 2. 118-bus clustering results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>No. Total Clusters</th>
<th>No. of Clusters with $f_{LPP}&gt;0.9$</th>
<th>$f_{LPP}$ of Clusters &gt; 0.9</th>
<th>Gens in Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>QT, 0.25</td>
<td>6</td>
<td>4</td>
<td>0.930</td>
<td>10</td>
</tr>
<tr>
<td>Qt, 0.1</td>
<td>6</td>
<td>4</td>
<td>0.930, 10</td>
<td>87,80,69,111, 103,100,89</td>
</tr>
<tr>
<td>K-Means, 4</td>
<td>5</td>
<td>4</td>
<td>0.930, 10</td>
<td>87,80,69,111, 103,100,89</td>
</tr>
<tr>
<td>SLCA, 4</td>
<td>5</td>
<td>4</td>
<td>0.930, 10</td>
<td>87,80,69,111, 103,100,89</td>
</tr>
</tbody>
</table>

Four clusters with $f_{LPP}>0.9$ are found. However, K-Means with four clusters and SLCA do not report the largest of the four clusters.

Figure 6. Seven-bus example system

Figure 7. 118-Bus system, constrained lines (5,8), (23,32), and (65,68)
5.3. Eastern case, area A

We next consider a large model of the Eastern Interconnect. There are a total of 43,311 buses in the system. We focus our study on a particular region’s footprint. The area of interest has 6,453 buses and 638 generators, and we consider six binding constraints. Results are shown in Table 3. Most algorithms find four LPP groups. Exceptions are discussed in Section 5.5.

Table 3. Area A clustering results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>No. Total Clusters</th>
<th>No. of Clusters with f_{LPP}&gt;0.9</th>
<th>f_{LPP} of Clusters</th>
<th>Gens in Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>QT, 0.25</td>
<td>6 5</td>
<td>0.999 37550</td>
<td>0.989 5067</td>
<td></td>
</tr>
<tr>
<td>QT, 0.1</td>
<td>10 4</td>
<td>0.999 37550</td>
<td>0.989 5067</td>
<td></td>
</tr>
<tr>
<td>K-Means, 6</td>
<td>6 4</td>
<td>0.999 37550</td>
<td>0.998 32</td>
<td></td>
</tr>
<tr>
<td>K-Means, 7</td>
<td>7 4</td>
<td>0.999 37550</td>
<td>0.998 32</td>
<td></td>
</tr>
<tr>
<td>CI</td>
<td>6 1</td>
<td>1 32,32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLCA, 7</td>
<td>7 3</td>
<td>0.998 37550</td>
<td>0.998 32</td>
<td></td>
</tr>
</tbody>
</table>

5.4. Eastern case, area B

A list of binding constraints is available on the MISO website [33]. For each day, a file records the binding constraints in the system every five minutes. A constraint which was biding 119 times in for an arbitrary day is given in Table 4.

Table 4. Example MISO Binding Constraints

<table>
<thead>
<tr>
<th>Constraint Name</th>
<th>Branch Name</th>
<th>Contingency Description</th>
<th>No. Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>'AMI13148_OTTO'</td>
<td>WATP_IP-1516</td>
<td>'HENNEPIN - HALLOCK 138 (1512)'</td>
<td>119</td>
</tr>
</tbody>
</table>

Figure 8 shows a few buses in the area of interest. The contingency is represented by opening the line between “4RICHLND JCT” and “4TLV STL ETP.”

The thick, dark line is assumed to be binding. Results are given in Table 5. Two well-separated clusters are found. The small cluster contains two generators at the same bus.

Table 5. Area B clustering results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>No. Total Clusters</th>
<th>No. of Clusters with f_{LPP}&gt;0.9</th>
<th>f_{LPP} of Clusters</th>
<th>Gens in f_{LPP}&gt;0.9 Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>QT, 0.25</td>
<td>2 1</td>
<td>0.999 349107, 349107</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QT, 0.1</td>
<td>4 1</td>
<td>0.999 349107, 349107</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-Means, 2</td>
<td>2 1</td>
<td>0.999 349107, 349107</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-Means, 3</td>
<td>3 1</td>
<td>0.999 349107, 349107</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI</td>
<td>2 0</td>
<td>0.999 349107, 349107</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.5. Discussion of clustering results

The results for LPP identification are strikingly similar, regardless of the choice of clustering method. However, there are some differences. First, the results of the CI algorithm seem to be less discerning when applied to a large case. For Area A, the CI only finds one cluster with f_{LPP}>0.9, and in Area B it finds none. The CI algorithm also takes the longest to execute. For the large examples, the K-Means and SLCA slightly faster than QT (approximately 1 second vs. 2 seconds for Area B), but CI takes about 10 times as long. The CI algorithm should be modified to be practical for large systems.

Results from QT, K-Means, and SLCA agree well. While QT and K-Means find different sets of clusters based on algorithm inputs, the results show that the clusters with f_{LPP}>0.9 are mostly the same. In the Area A case, the only difference between QT and K-Means is that QT with a 0.25 threshold finds one additional cluster with f_{LPP}>0.9, but its value is lower (0.927, compared to 0.999,0.989,0.998, and 1). For the Area B case, QT and K-Means produce the same results. Thus, the impact of these parameter choices is minor for LPP identification.

One of the benefits of the SLCA algorithm for LPP applications is to allow the user flexibility and interaction to select a particular clustering. Instead of only one clustering result, we obtain several. Viewing
Clustering is a useful analysis tool for determining the potential of generators in load pockets to experience market advantage. The use of various clustering tools for sensitivity-based LPP analysis is examined. Offering all of them as choices to the user is important. The QT, K-Means, and SLCA algorithms are able to quickly find LPP groups in large systems. The same LPP groups are identified with large systems. The same LPP groups are identified in Section 5. It is beneficial to run several algorithms using different algorithms; exceptions are discussed in this paper. We present the results of five random clusterings of each system are shown in Table 6, except for the 7-bus system. As the larger study systems are examined, randomly-selected clusters are less likely to select “true” clusters. The random-clustering results are poor for both large systems, as expected; out of five clusterings, there are no clusters with \( f_{LPP} > 0.9 \).

### Table 6. \( f_{LPP} \) for Random Cluster Assignments

<table>
<thead>
<tr>
<th>Case</th>
<th>No. of Clusters with ( f_{LPP} &gt; 0.9 )</th>
<th>Max ( f_{LPP} ) of any Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area A</td>
<td>118</td>
<td>0.653</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.956</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.653</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>0.986</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.312</td>
</tr>
<tr>
<td></td>
<td>233</td>
<td>0.842</td>
</tr>
<tr>
<td></td>
<td>195</td>
<td>0.482</td>
</tr>
</tbody>
</table>

It is important to note that these techniques are general and applicable to a wide range of data mining and data analysis projects. A particular application for LPP analysis is presented in this paper, but the framework is in place to allow these special data structures and algorithmic tools to be used for other types of analysis. We are continuing to pursue a wide range of data mining techniques and to develop insightful power system applications with these tools.

### 7. References


