Regarding Cluster Overlaps

For any classification mechanism, overlap between clusters such that one cannot confidently identify a point as belonging to one cluster as compared to another cluster indicates insufficient feature collection. The key problem can be thought of in the context of a typical Venn diagram with overlapping circles, which cluster do the points in the overlapping region belong to? Without additional information, such as an additional segregating dimension (e.g., color), it is unknown which of the two clusters the point belongs to.

The problem occurs both in real-life situations, e.g., Iris classification, where the statement “not linearly separable” is used, as well as with random, synthetically created clusters, especially when too many clusters are squeezed into too small of dimensional space. With real life data, we seldom know the overlap. With synthetic clusters, the overlap can be computed.

I have made up my own formula for scoring cluster overlaps below. I chose the following (which is to say it may not be the best way, but it was ‘a’ way).

1. Each cluster had an assigned center-point, around which its points were randomly assigned (in each dimension). Thus, the distance between the two cluster centers is the Euclidian distance of the two center points, let’s call it ‘dist’.
2. K-Means computations are based on the sum square errors (SSE). The square-root of the per-point average SSE for each cluster was taken, let’s call them rSSE1 and rSSE2.
3. An overlap occurs when rSSE1 + rSSE2 > dist, which is so say, 1 < (rSSE1 + rSSE2) / dist. I arbitrarily choose to use a scoring mechanism of log( (rSSE1 + rSSE2) / dist) when an overlap occurs.
4. For a collection of clusters, overlap score is computed as:

1 + ∑ over all clusters i,j where i < j

+ 0 – if no overlap

+ log( (rSSE1 + rSSE2) / dist) otherwise

As an example, K-clusters in D-dimensions with K < D, usually have a value of 1 because there is plenty of separation in the random initialization of the cluster centers. However, when K >> D, e.g., 23-clusters in 3-dimensions, we frequently get a computation in the range of 2-3. I did not run the mean or standard deviation, although it may be computed from the data provided.

My initial hypothesis was to be able to make a statement about the overlap with respect to the number of iterations to converge. The only “correlation” is that it usually takes substantially more iterations of the K-Means algorithm when such overlaps occur, because of the “tug-of-war” that occurs over the points and the slow movement of the centroids, especially as the number of points in a cluster increase. However, there really was not enough room in the paper to or available time to chase this down. Which also means that perhaps a different computation for this overlap would be better. Two somewhat ‘obvious’ possibilities are 1) simple linear or 2) sqrt(). However, early testing, anecdotal evidence only, is that simple linear resulted in values that were too high, thus yielding iterations / overlap to be less than one, which doesn’t make good sense. I did not try the sqrt() option, but that too might cause issues as the sqrt of a fractional value actually grows, which might yield similar issues. A third possibility is to take the average score (i.e., sum (from #4) / #clusters).