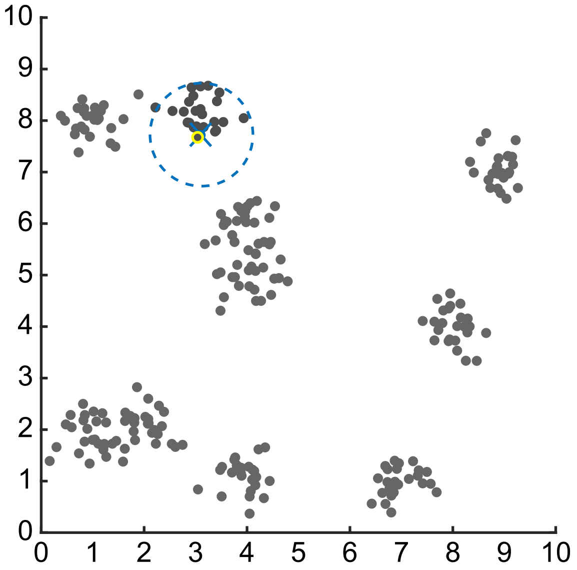
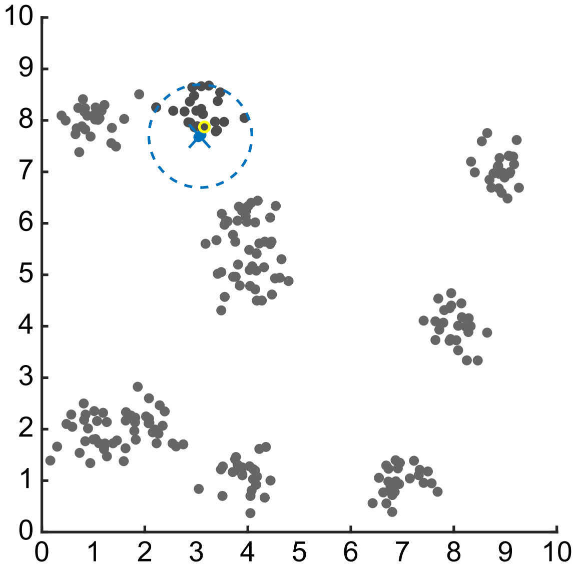
Step 1: Crawl through clusters



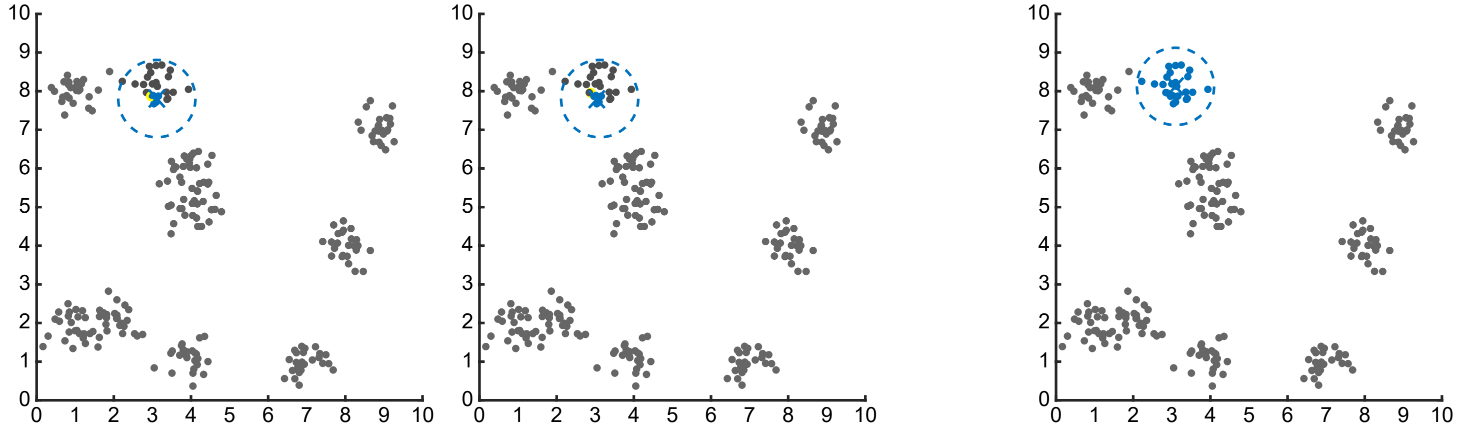
Begining at your first point, establish cluster 1 as only this single point (blue), and use its position as the centroid for the cluster (blue cross).
Then find the closet point (yellow), and determine, if it is within the threshold (dashed circle)



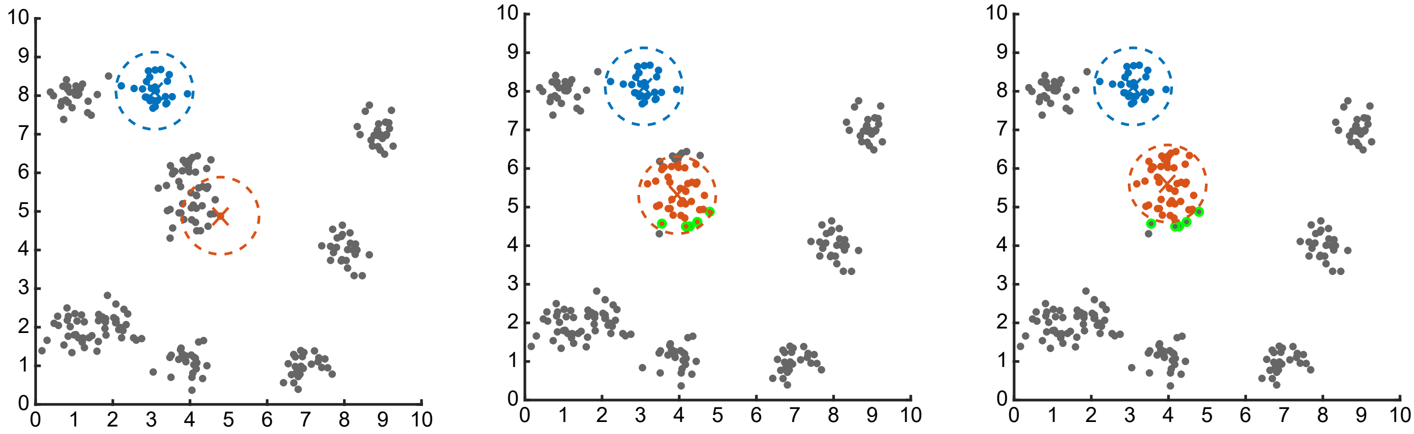
If the next point is within the threshold, then we include it into our cluster. And update the centroid position.

We then search to see if the next point is within the threshold distance to the the cluster

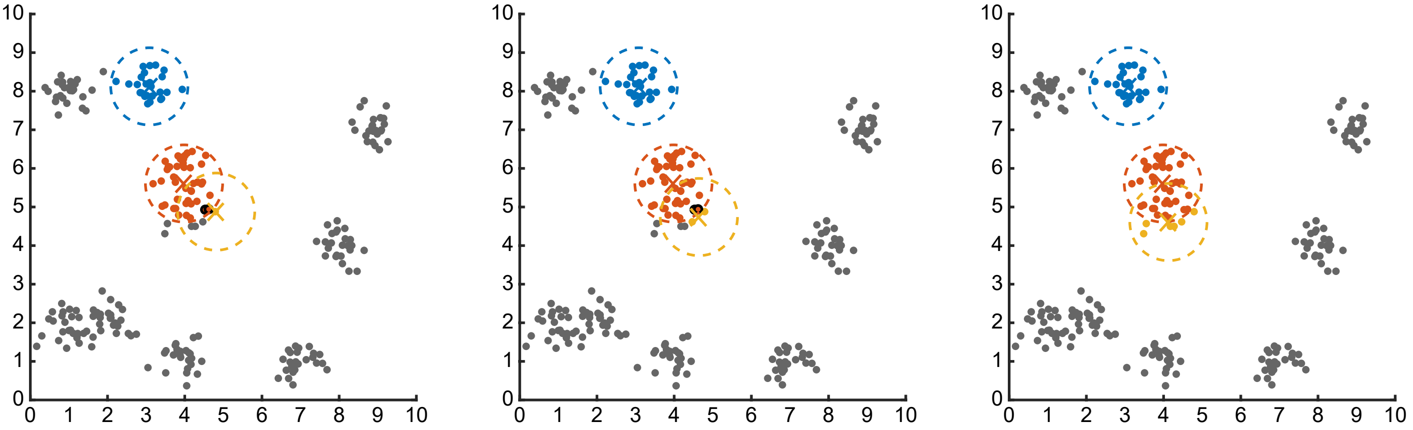
We keep searching through until there are no more new points within the threshold distance to the centroid

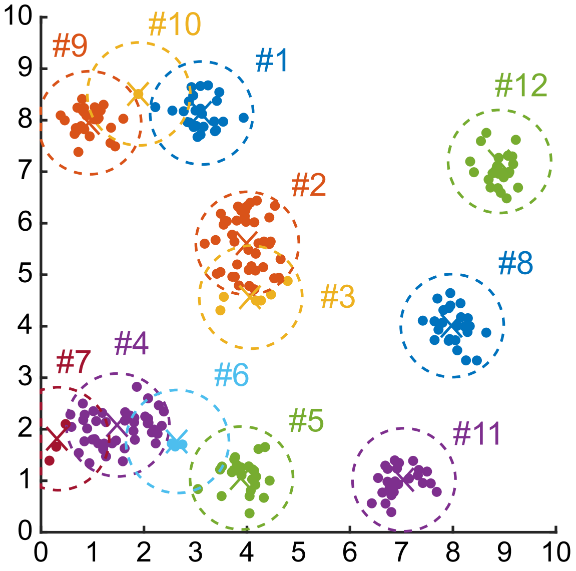


However, sometimes it is possible for points to leave the cluster if the distance from the updated centroid position is larger than the threshold (green)



Additionally, for the first sweep, if closest point is within the threshold but already belongs to a different cluster they are ignored (black outline)

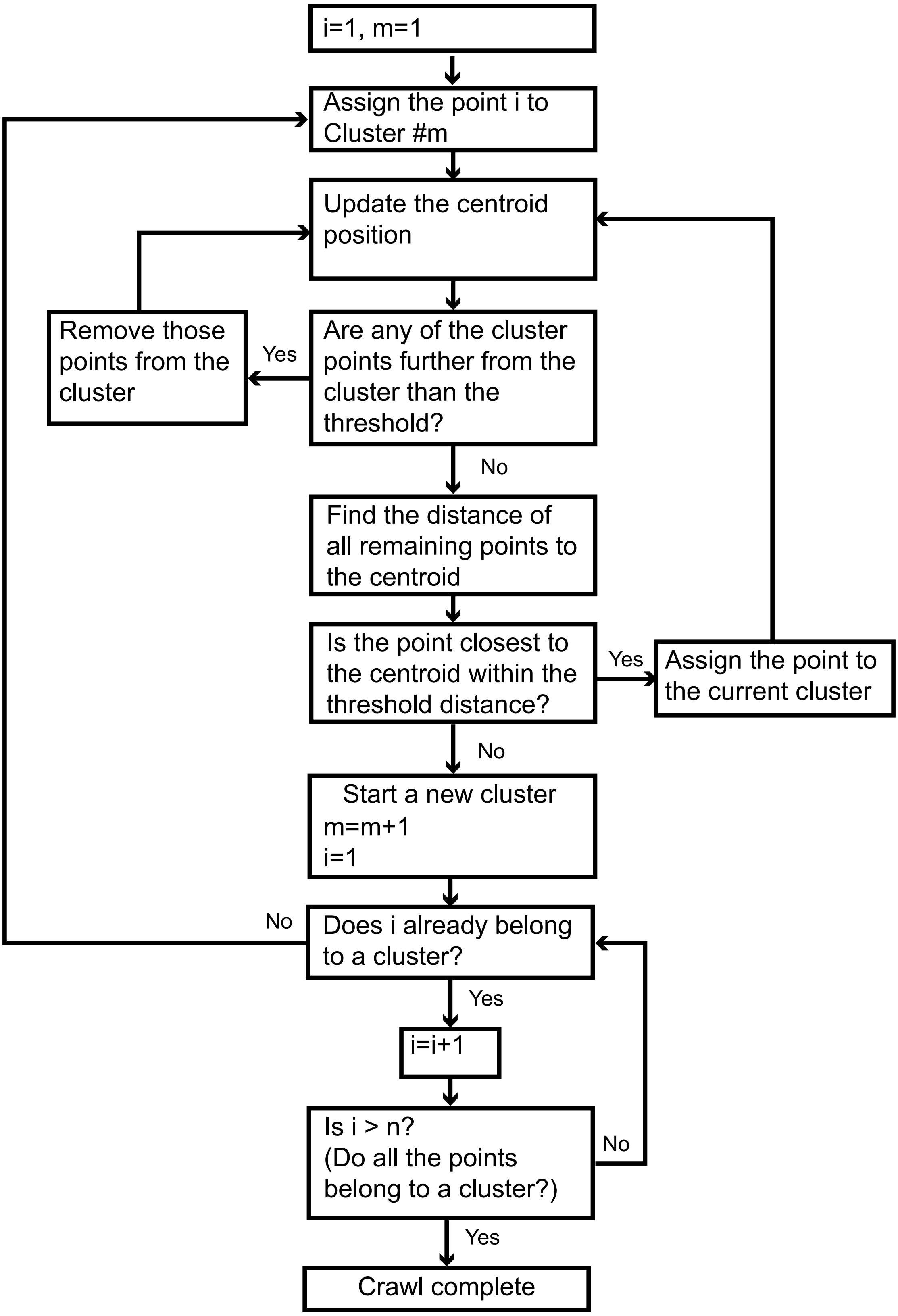




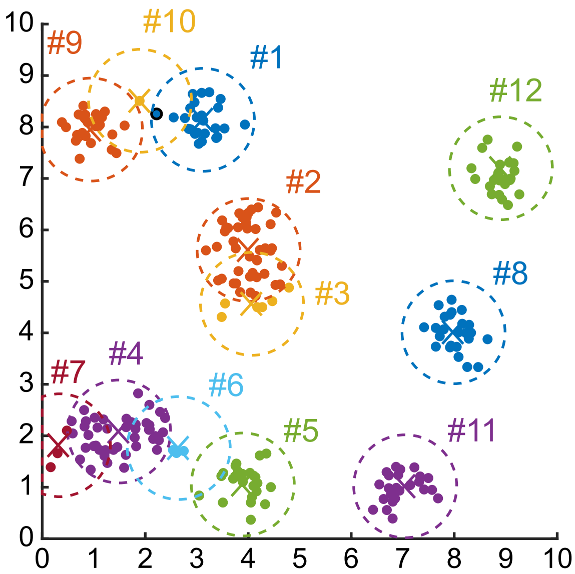
This generates our first pass of the crawler. However, there is a primacy bias which we will need to correct for. (That is the clusters that are generated first will include points that are closer a later generated cluster)

Flowchart for the Crawler

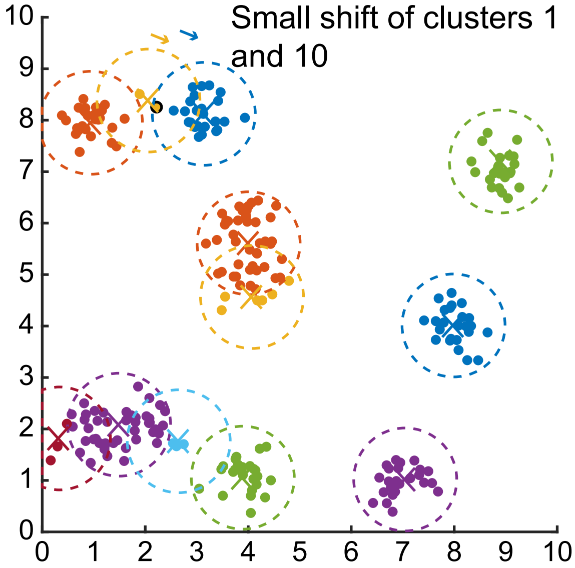
For points i=1 to n


i=1, m=1
Assign the point i to Cluster #m
Update the centroid position
Remove those points from the cluster
Yes
Are any of the cluster points further from the cluster than the threshold?
No
Find the distance of all remaining points to the centroid
Is the point closest to the centroid within the threshold distance?
Yes
Assign the point to the current cluster
No
Start a new cluster
m=m+1
i=1
No
Does i already belong to a cluster?
Yes
i=i+1
Is i > n?
(Do all the points belong to a cluster?)
No
Yes
Crawl complete

Step 2: Adjust Clusters

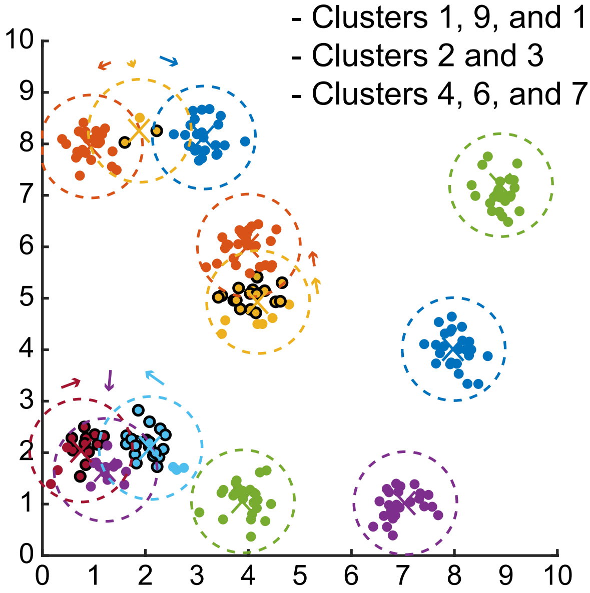


So now we go through each point and assign each point to the cluster with the closest centroid (e.g. black outline).Then we update the centroid positions after each adjustment.
NB For a fast implementation of the program. Instead of restarting the loop after each adjustment. I keep checking the remaining points. The adjustment is only considered complete when the final loop has no adjustments.

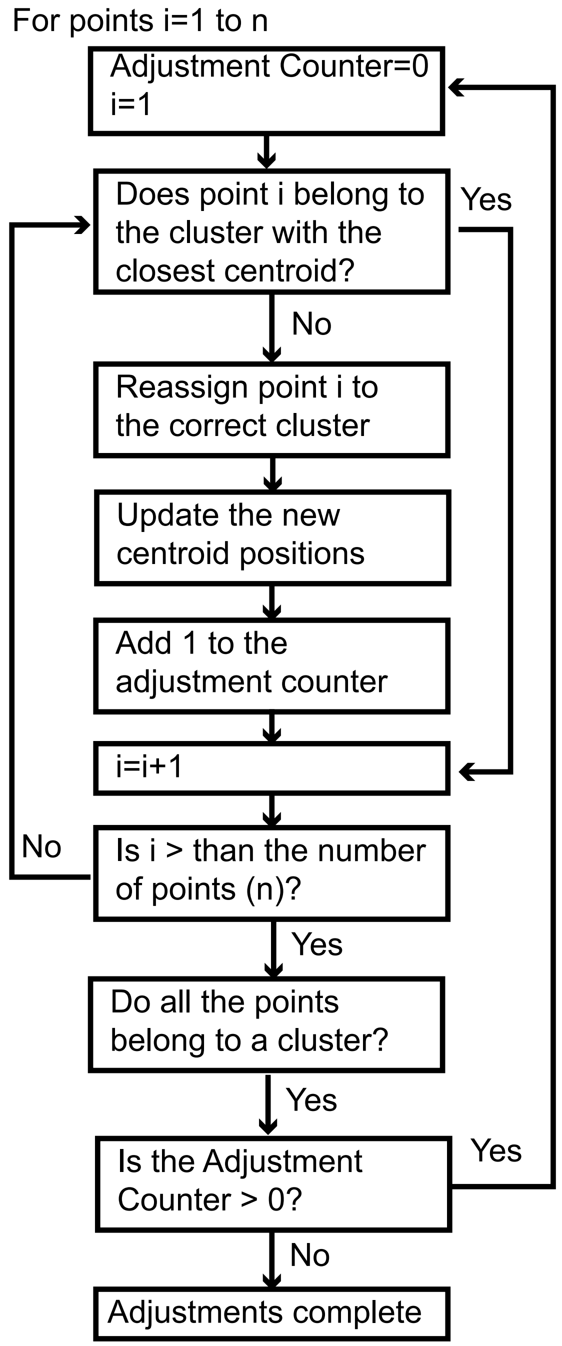


Small shift of clusters 1 and 10

Small shifts between
- Clusters 1, 9, and 10
- Clusters 2 and 3
- Clusters 4, 6, and 7



Flowchart for Adjusting Clusters



For points i=1 to n

Adjustment Counter=0
i=1

Does point i belong to the cluster with the closest centroid?

Yes

No

Reassign point i to the correct cluster

Update the new centroid positions

Add 1 to the adjustment counter

i=i+1

No

Is i > than the number of points (n)?

Yes

Do all the points belong to a cluster?

Yes

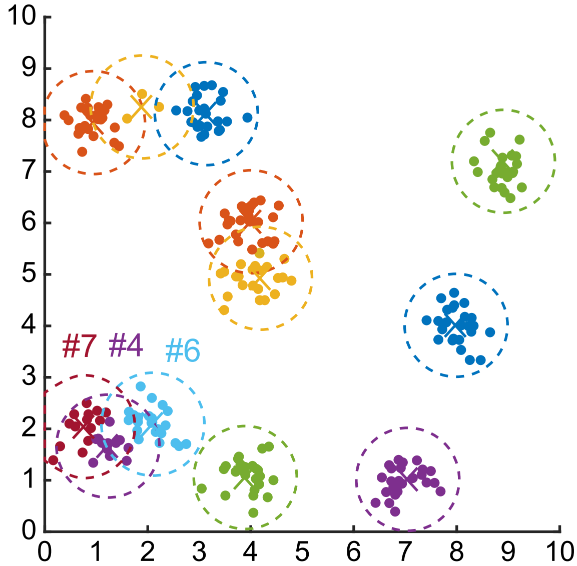
Is the Adjustment Counter > 0?

Yes

No

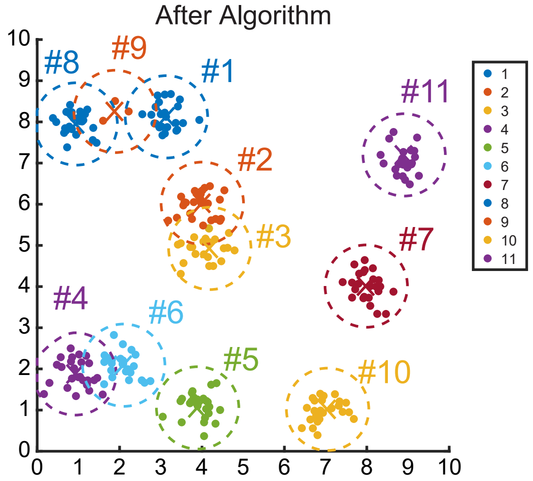
Adjustments complete

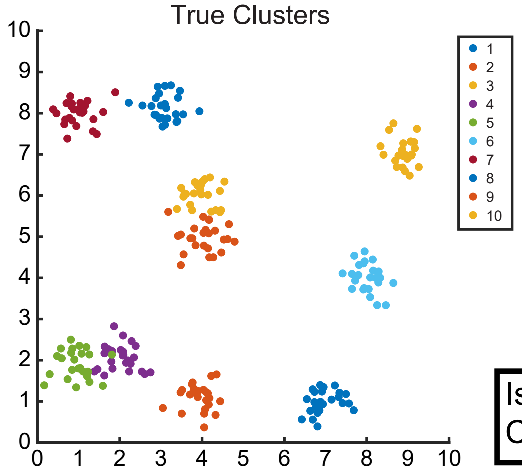
Step 3: Merge clusters



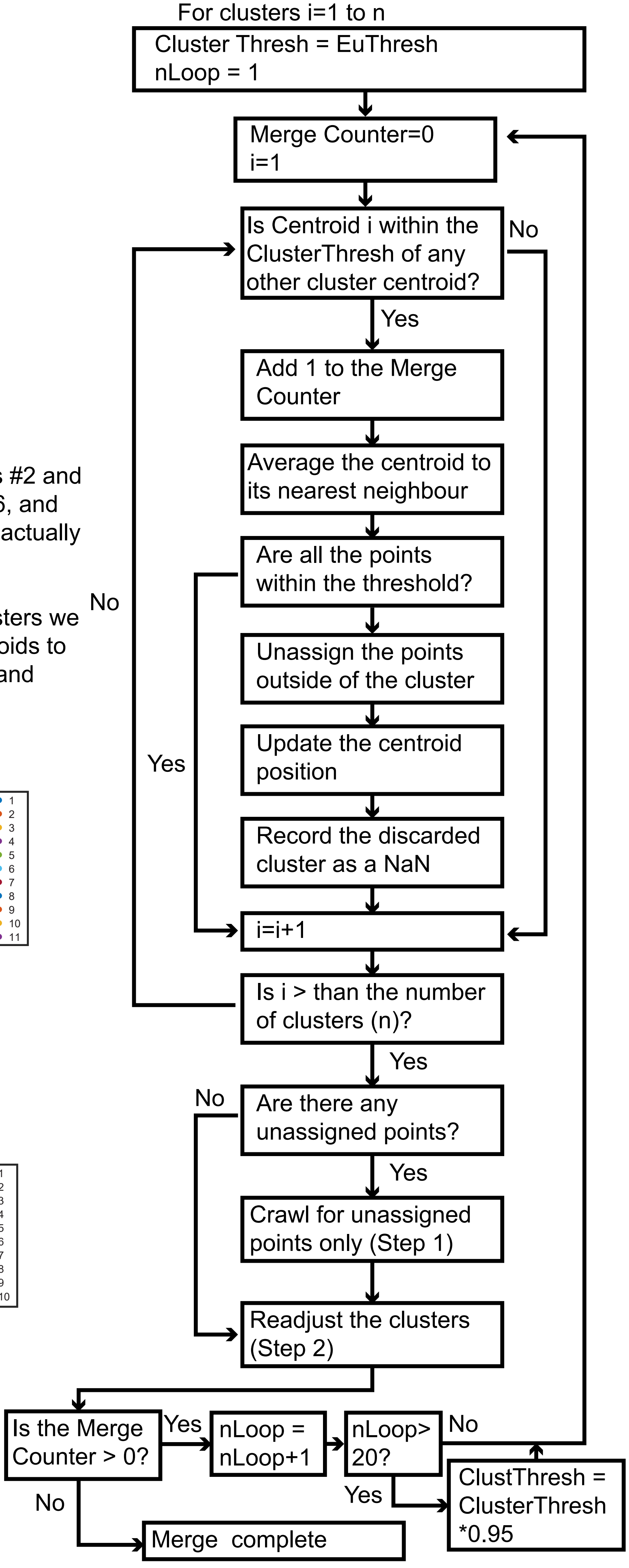
After the adjustment it looks like Clusters #2 and #3 are resolved. However clusters #4, #6, and #7 (esp #4 and #7) look like they should actually be the same cluster.

So now to evaluate the closeness of clusters we measure the distance between the centroids to evaluate this. Then we redo the crawler and make adjusstments if necessary.


After Algorithm


True Clusters

Flowchart for Merging Clusters


For clusters i=1 to n

Cluster Thresh = EuThresh
nLoop = 1

Merge Counter=0
i=1

Is Centroid i within the ClusterThresh of any other cluster centroid?

No

Yes

Add 1 to the Merge Counter

Average the centroid to its nearest neighbour

Are all the points within the threshold?

No

Unassign the points outside of the cluster

Yes

Update the centroid position

Record the discarded cluster as a NaN

i=i+1

Is i > than the number of clusters (n)?

Yes

No

Are there any unassigned points?

Yes

Crawl for unassigned points only (Step 1)

Readjust the clusters (Step 2)

Is the Merge Counter > 0?

Yes

nLoop = nLoop+1

nLoop> 20?

No

Yes

ClustThresh = ClusterThresh *0.95

No

Merge complete