```
function [final_cluster_matrix delta] = Genetic_Cluster_Algorithm(dataset, N)

  %delta is the minimum number of matching bases for inclusion in a given cluster
  delta_max = N; %the maximum possible number of matches
  delta = 1; %the initial value of delta, which iterates up to N

  M = size(dataset,1); %the number of rows in the dataset / total population size

  %these two matrices are used to store the clusters, entry (i,j) is 1 if row j is included in
the cluster for row i
  current_cluster_matrix = zeros(M,M);
  prior_cluster_matrix = zeros(M,M);

  max_change = 0; %initial housekeeping value

  %the initial loop is outside the main loop to make the code easier to read

  %iterates over all rows, testing each for inclusion in the cluster for a given row
  for current_row = 1 : M

    current_vector = dataset(current_row,:);

    for test_row = 1 : M

      %precludes a row from matching with itself
      if(test_row != current_row)

        test_vector = dataset(test_row,:);
        count = cmp_base_vectors(current_vector, test_vector, N);

        %if true, then the minimum number of matching bases is met
        if(count >= delta)

          current_cluster_matrix(current_row,test_row) = 1;

        endif

      endif

    endfor

  endfor

  %the main loop that iterates over values of delta
  for delta = 2 : delta_max

    prior_cluster_matrix = current_cluster_matrix; %housekeeping that lets the algorithm
```

```
work
   current_cluster_matrix = zeros(M,M);

   %iterates over all rows, testing each for inclusion in the cluster for a given row
   for current_row = 1 : M

     current_vector = dataset(current_row,:);

     for test_row = 1 : M

       %precludes a row from matching with itself
       if(test_row != current_row)

         test_vector = dataset(test_row,:);
         count = cmp_base_vectors(current_vector, test_vector, N);

         %if true, then the minimum number of matching bases is met
         if(count >= delta)

           current_cluster_matrix(current_row,test_row) = 1;

         endif

       endif

     endfor

   endfor

   %this counts the number of changed entries in the cluster matrix
   change = sum(abs(prior_cluster_matrix(:) .- current_cluster_matrix(:)))

   %if true, the rate of change per iteration is greatest, and the clusters are final
   if(change > max_change)

     max_change = change;
     final_cluster_matrix = prior_cluster_matrix;
     final_delta = delta;

   endif

 endfor

endfunction
```