APPENDIX A: Heuristic clustering

Our heuristic assumes that each module has features A,B,C, etc and that each pair of modules has a distance $\sqrt{(A1-A2)^2+(B1-B2)^2+(C1-C2)^2+\ldots}$ Using this information, our algorithm recursively divides the space as follows:

- Pick any module at random, which we call "start".
- Find the point furthest from "start", which we call "east".
- Next, it finds the module furthest from "east", which we call "west".
- A line drawn from "east" to "west" is (approximately) then dimension of greatest variance in the data. This is analogous to *component[1]* in PCA. However, given *N* points, this analysis requires only *2N* calculations and not the polynomial time analysis of PCA.
- If this line has length δ, then if it is laid down on the x-axis of a 2-d plot , it will run from "west" (at 0,0) to "east" (at 0,δ).
- All modules in the code fall at some point *(x,y)* in this 2-d plot. To compute *(x,y),* we note that have distances α,β from east and west. Hence:
  1) $\beta^2 = (x-0)^2 + (y-\delta)^2$
  2) $\alpha^2 = x^2 + y^2$
- These can be re-arranged to:

$y^2 = \alpha^2 - x^2$

  3)
  4) $x = \sqrt{\alpha^2 - y^2}$

- Expanding equation (1) and substituting in equation (3) yields:
  5) $\beta^2 = x^2 + \alpha^2 - x^2 - 2\delta y + \delta^2$
- If rearranged to isolate *y*, then Equation (5) becomes
  6) $y = \dfrac{\delta^2 + \alpha^2 - \beta^2}{2\delta}$

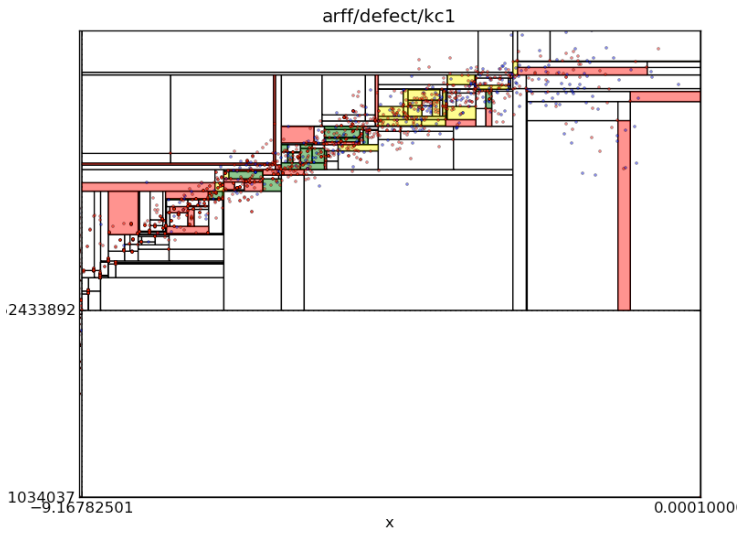
  so the point (x,y) can be computed by solving Equations (6), then (4).
- Once computed, we normalize the *(x,y)* values to run from zero to one:
  - All x-values are divided by δ
  - All y-values are divided by the max y value ever seen in Equation (6)
- Since this co-ordinate system is based on the dimension of greatest variance, often "east" is some strange outlier and all the data is bunched around "west". Hence, if most of the data falls into one half of the x or y axis, we
  - Flip the zero,one positions on that axis
  - Apply a log transform to that data (to spread out the data on that axis).
- Finally, we split each axis at its median point to divides the data into four "tiles" , then call this algorithm on each tile (stopping when there are too few modules in a sub-tile; currently, *tooFew=4*).
- The output of this process is an "index tree" to a set of tiles. This index tree is set of split and sub-split and sub-sub-splits that formed the tiles. Note that this tree can be used to quickly find the nearest tile to a new module.

Once this algorithm terminates, we can then apply grid clustering[6] to cluster the tiles. Let each tile have height and width *h,w,* contain *M* modules and have neighbors *N*. We say that "similar" tiles have densities *(M/(h*w))* within 10%. Starting with number of clusters = "*c*" =1 and with all tiles marked as "unclustered", then, we sort the tiles by density in descending order. The first *"a"* items in that sort with similar densities are labeled "active". Clustering proceeds as follows:

1) Add the first unclustered tile to cluster *c* and mark it "clustered"
2) Each time a tile is added to a cluster, consider also adding the neighbors of that addition (if their density is similar to their added neighbor). If an active tile is added, then *a= a-1.*
3) Once there are no more similar neighbors to add, then if there are more active tiles to process (i.e. *a>0)*, then
    a. Start a new cluster; i.e. *c = c+ 1*
    b. Goto 1

This divides the modules into *c* clusters. After the clusters exist, we prune the index tree. If all the sub-tiles of a node in the tree fall into one cluster, then the tree stops at that node.

Finally, we can apply our learners to each cluster generated in the above manner. If two neighboring regions have very different performance, then we would flag that region as requiring more investigation. For example, in the following, ref and green dots denote faulty and non-faulty modules . In this plot, the west-east dimension is the x-axis and all the squares are generated by the above process. Red, yellow, and green tiles show regions where our predictors are working poorly, ok, very good. Any green region next to a red indicates a region where the conclusions are unstable and require more investigation.



[6] Schikuta, E.; , "Grid-clustering: an efficient hierarchical clustering method for very large data sets," *Pattern Recognition, 1996., Proceedings of the 13th International Conference on* , vol.2, no., pp.101-105 vol.2, 25-29 Aug 1996