151228620/151248620: OBJECT-ORIENTED PROGRAMMING II
2021-2022 SPRING SEMESTERS
PROJECTS
Due: June 8, 2022, Wednesday, 17:00

Remarks:

- Module code should be cleverly commented.
- For the project, you should write a report explaining the design and implementation issues as well as anything a person may need to know while testing your program.
- You are expected to make a good design and exploit object-oriented programming principles (abstraction, inheritance, exception handling, etc.) as often as possible.
- You are expected to make a good design and exploit built-in data structures (list, dictionary, and set) and libraries (numpy, pandas, and matplot) as often as possible.
- For projects, you need to send the project workspace, including your .py files. In addition, you must submit your document. You must organize all this material in a single .zip or .rar archive, name the file as "yourID_Name_ProjectID.zip/rar", and send by e-mail.
- Grades will be given according to
    - 50% Class design and availability of object-oriented programming principles,
    - 15% code quality (commented and well formatted),
    - 15% documentation, (Use Doxygen)
    - 20% functionality (implemented and correctly running functionalities).

In this assignment, you will design an interface to cluster a data. An example interface is given in Figure 1. **It is important to note that the given interface is an example, and you have to design your interface.** In your backhand codes, you have to use the scikit-learn library (Please visit the website: https://scikit-learn.org/stable/). You do not need to download and install the scikit-learn library if you use WinPython distribution, which is used in our course. You can easily check that the library is available in your system by typing import sklearn in interactive mode.

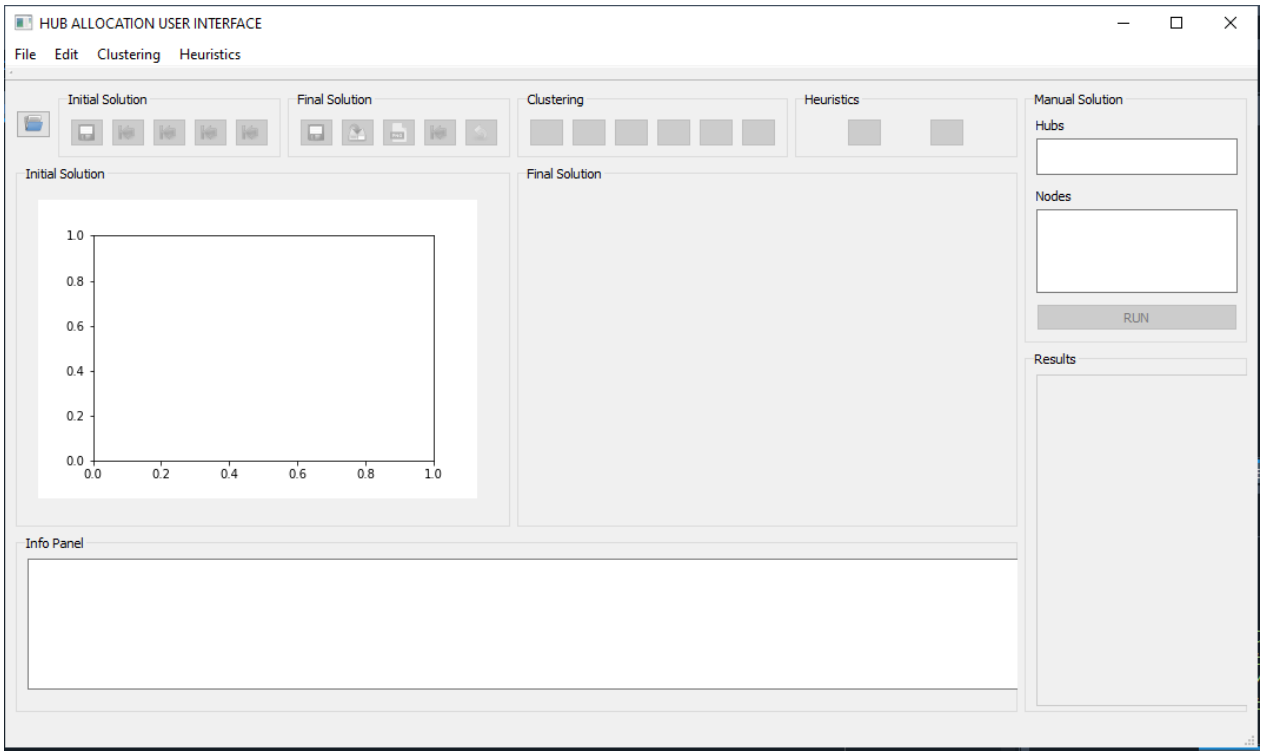
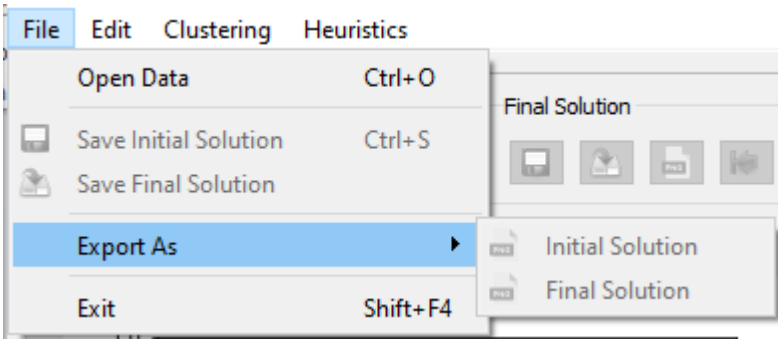


Figure 1. An example interface.

Your design must meet the following criteria:

- As seen from the figure, the interface must include two parts: One for initial solution and the other for final solution.
- The functionalities must be collected under four menus: File, Edit, Clustering, and Heuristic. Also, each functionality must have a button. You can use toolbar or group boxes for that purpose.
- Each functionality must have an appropriate shortcut, status bar tip, and icon.
- Do not use default object names that are given in Qt Designer. Each object in the interface must have an appropriate and unique name.

- **<u>File Menu</u>**



In the file menu, there are six functionalities: Open Data, Save Initial Solution, Save Final Solution, Export As for both Initial and Final Solution, and Exit. In the beginning, all functionalities except Open Data and Exit must be disabled. When the user selects Open Data functionality in the File menu or clicks the corresponding button, a dialog window must be opened. The user can only open the files with .txt extension. After the data is opened, it must be shown in Initial Solution part with black color (see Figure 2), and functionalities and also corresponding buttons for Clustering menu must be enabled. Save Initial Solution and Export As Initial Solution must be enabled after an operation is performed from Clustering. Save Final Solution and Export As Final Solution must be enabled after an operation is performed from Heuristics. Save Initial Solution and Save Final Solution functionalities must open a dialog window, and the user can enter the name and current folder of the solution. Then, these functionalities save the solution with .txt extension. Lastly, Export as functionalities must save the solution with .jpg extension.

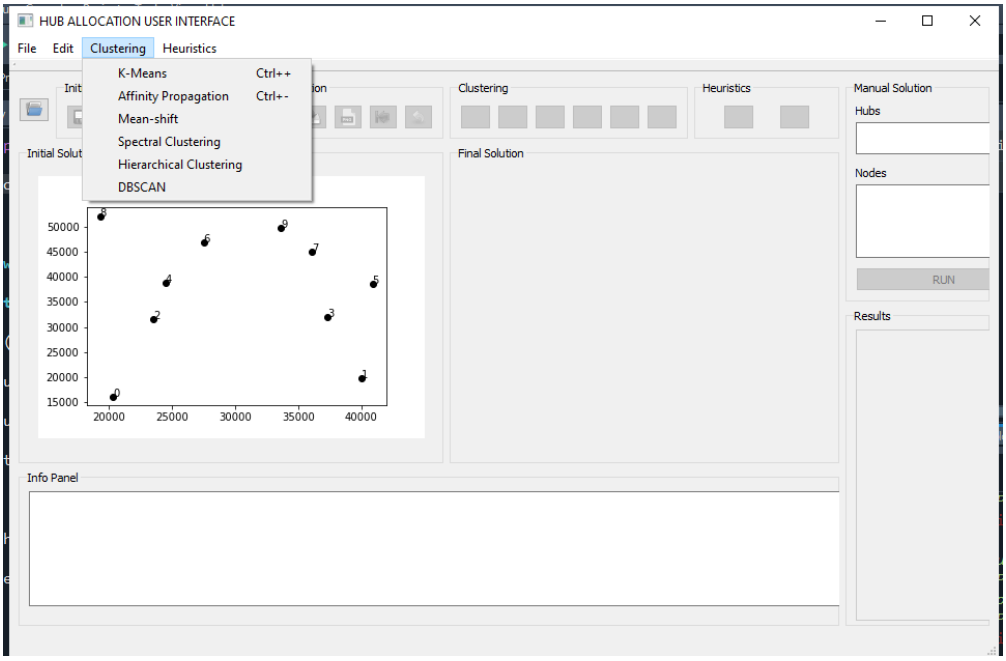
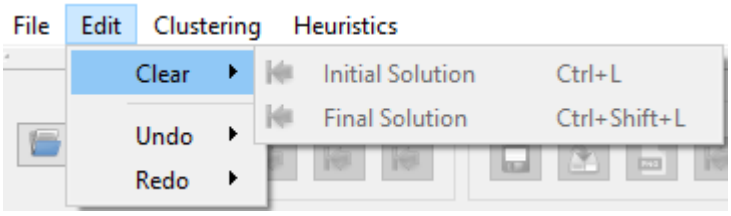


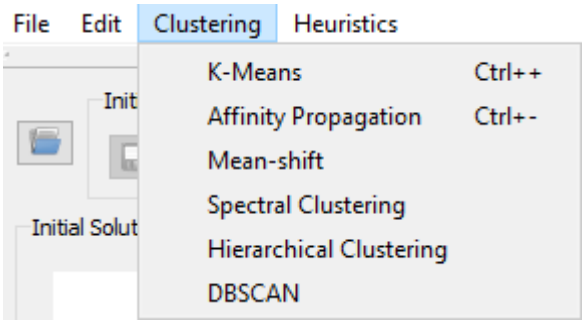
Figure 2. An example interface after data is opened

- **Edit Menu**



In the edit menu, there are three functionalities: Clear, Undo, and Redo for both Initial and Final Solution. In the beginning, all functionalities must be disabled. When the user opens a data file Clear Initial Solution functionality must be enabled. If the user selects Clear Initial Solution functionality in the Edit menu or clicks the corresponding button, Clear Final Solution functionality must also be invoked. Undo Initial Solution and Redo Initial Solution must be enabled after an operation is performed from Clustering menu. Undo Final Solution and Redo Final Solution must be enabled after an operation is performed from Heuristic menu. You have to implement a polymorphic class hierarchy for undo and redo functionalities.

Clustering Menu



In the clustering menu, there are six functionalities: K-Means, Affinity Propagation, Mean-shift, Spectral Clustering, Hierarchical Clustering, and DBSCAN. To implement these functionalities, please visit the website: https://scikit-learn.org/stable/modules/clustering.html#clustering In this part, when the user selects any functionality in the Clustering menu or clicks the corresponding button, a window must be opened for the parameters of the method. Please consider all critical parameters for the method. An example window for K-Means Clustering is shown in Figure 3. After user enters the parameters, clustering method is applied and the result is shown in Initial Solution part. Also, display crucial information in both Info Panel and Results.

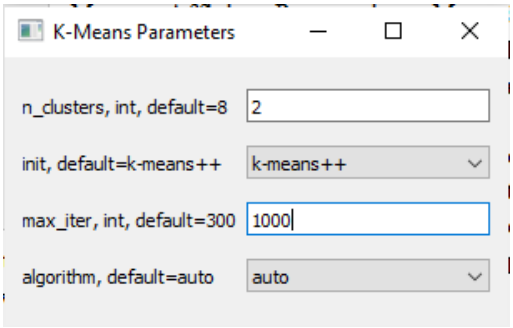


Figure 3. An example window for K-Means Parameters

In this point, first find cluster items (Hint: use clustering.labels_). Then, divide the points into clusters. An example is given in Figure 4. Here, we have 3 clusters so that clustering labels vary between 0 and 2. Lastly, print the clusters as shown in the figure. In the next stage, find cluster centers using the cluster items. Some methods such as K-Means return the cluster centers in clustering.cluster_centers_. However, some methods such DBSCAN does not return cluster centers and you have to calculate clusters, which are mean of points that belong to that cluster (shown with

red crosses). After you determine the cluster centers find the closest point in the cluster, which is defined as cluster center node (shown in red points). Then, calculate objective function. To do that, first determine distance of farthest point in the cluster to the cluster center node and find all possible pair combinations for cluster center nodes. Then, for each pair calculate the following objective function:

$$obj_{ij} = d_{ihi} + 0{,}75 * d_{hihj} + d_{jhj}$$

where d_{ihi} distance of farthest point in the cluster i, d_{hihj} diatnce between hub i and hub j and d_{jhj} distance of farthest point in the cluster j. Also, consider 2*max(d_{ihi}). Lastly, get the maximum of pair objectives as an objective function result.

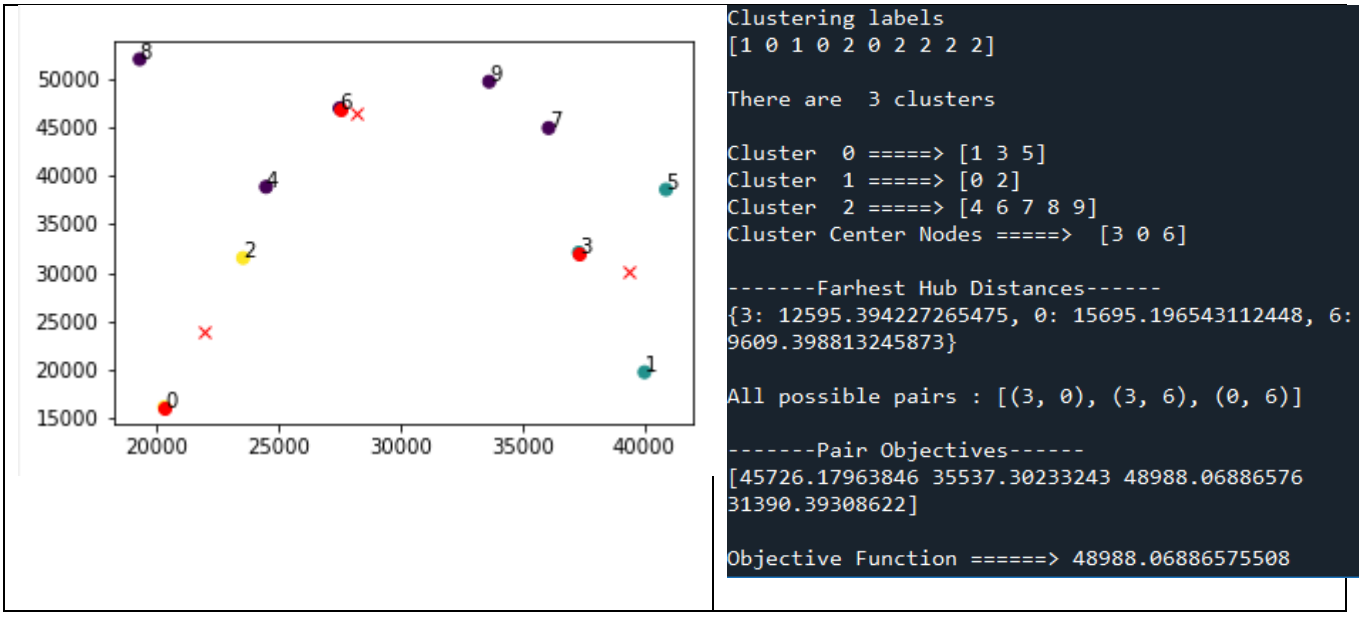
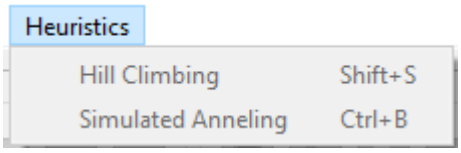


Figure 3. An example result for K-Means with K=3

Heuristics Menu



In the heuristics menu, there are two functionalities: Hill Climbing and Simulated Annealing. To implement these functionalities, please visit the websites: https://machinelearningmastery.com/stochastic-hill-climbing-in-python-from-scratch/ and https://machinelearningmastery.com/simulated-annealing-from-scratch-in-python/. These methods receive the initial solution as best, and generate new solution with three operators given below and repeat the same steps mentioned above.

RelocateHub: Change the location of the hub within a (randomly chosen) cluster to a different (randomly chosen) node in the cluster. This transition applies to clusters with at least one non-hub node.

ReallocateNode: Change the allocation of a (randomly chosen) non-hub node to a different (randomly chosen) cluster. In the special case where the cluster that is chosen for reallocation consists of just a single hub-node, we do not allow this hub to be allocated to another cluster.

SwapNodes: Swap the allocations of two (randomly chosen) non-hub nodes from different clusters. In other words, each node is allocated to the hub of the other.