**Parallel implementation of K-Means**

Final project

Course 10324, Parallel and Distributed Computation

2017 Summer Semester

# Problem Definition

Given a set of points in **n**-dimensional space. Implement simplified K-Means algorithm to find K clusters with best quality measure.

1. Set K = 2
2. Choose first K points as a cluster centers
3. Group points around the given cluster centers - for each point define center that is most close to the point
4. Recalculate the cluster centers – average of all points in the cluster
5. Check the termination condition – no points move to other clusters or maximum iteration LIMIT was performed.
6. Repeat from 3 till the termination condition fulfills.
7. Evaluate the Quality of the clusters found. The Quality is equal to an average of diameters of the cluster divided by distance to other clusters. For example, in case of k = 3 the quality is equal

**q = (d1/D12 + d1/D13 + d2/D21 + d2/D23 + d3/D31 + d3/D32) / 6**,

where di is a diameter of cluster **i** and Dij is a distance between centers of cluster **i** and cluster **j**.

1. Stop if Quality is less than Predefined Quality
2. Increment the value of K by 1, finish if K is bigger than predefined maximum value **MAX**
3. Return to step 2

Input data and Output Result of the project

You will be supplied with the following data, taken from

* N - Number of points
* MAX - maximum number of clusters to find
* LIMIT – the maximum number of iterations for K-MEAN algorithm.
* QM – quality measure to stop
* Coordinates of all points

**Input File format**

The input data is based on Sales Transaction Dataset[[1]](#footnote-1).

The first line of the file contains **N** - the number of given products, dimension **n** and **MAX** - the maximum number of clusters to find, maximum number of iterations **LIMIT** and **QM** - quality measure to reach

Following lines contain **product ID** and normalized number of sales of this product for 52 weeks,

one line per product.

For example:

**250 52 30 2000 7.3**

**P1 0.44 0.5 0.39 0.28 0.56 0.5 0.61 1 0.17 0.61 0.44 0.61 0.72 0.33 0.33 0.33 0.61 0.33 0 0.5 0.11 0.44 0.22 0.5 0.11 0.33 0.22 0.39 0.11 0.44 0.22 0.39 0.5 0.17 0.11 0.61 0.39 0.33 0.5 0.78 0.22 0.44 0.06 0.22 0.28 0.39 0.5 0 0.22 0.17 0.11 0.39**

**P2 0.7 0.6 0.3 0.2 0.7 0.1 0.6 0.3 0.3 0.3 0.2 0.2 0.6 0.2 0 0.6 0.2 0.7 0.7 0.9 0.4 0.7 0.2 0.4 0.5 0.3 0.5 0.8 0.5 0.5 0.3 0.1 0.3 0.2 0.3 1 0.5 0.2 0.7 0.3 0.2 0.5 0.2 0.4 0.5 0.1 0.1 0.4 0.5 0.1 0.6 0**

**…**

**P250 0.36 0.73 0.45 0.55 0.64 0.45 0.36 0.91 0.82 0.27 1 0.55 0.09 0.36 0.82 0.45 0.36 0.73 0.64 0.36 0.36 0.91 0.73 0.45 0.64 0.45 1 0.18 0 0.91 0.73 0.55 0.36 0.45 0.36 0.55 0.27 0.82 0.82 0.55 0 0.18 0.27 1 0.18 0.18 0.36 0.45 1 0.45 0.45 0.36**

**Output File format**

The output file contains information on the found clusters with the best quality measure

For example:

**Number of clusters with the best measure**

**K = 3 QM = 6.5**

**Centers of the clusters:**

**C1 0.56 0.72 0.33 0.49 0.67 0.87 0.62 0.64 0.69 0.64 0.77 0.54 1 0.97 0.74 0.79 0.69 0.72 0.74 0.31 0.54 0.38 0.87 0.26 0.49 0.51 0.23 0.26 0.31 0.36 0.46 0.51 0.62 0.67 0.46 0.44 0.92 0.56 0.74 0.72 0.46 0.46 0.62 0.56 0.64 0.46 0.44 0.64 0.23 0.67 0.46 0**

**C2 0.24 0.31 0.48 0.45 0.03 0.48 0.79 0.34 0.17 0.45 0.9 0.76 0.55 0.69 0.45 0.59 0.41 0.45 1 0.24 0.52 0.38 0.59 0.41 0.52 0 0 0.28 0.31 0.45 0.07 0.17 0.52 0.24 0.66 0.45 0.24 0.1 0.28 0.45 0.62 0.34 0.45 0.28 0.72 0.48 0.34 0.55 0.38 0.34 0.21 0.03**

**C3 0.61 0.61 0.33 0.28 0.33 0.22 0.33 0.44 0.39 0.61 0.72 0.22 0.67 0.28 0.83 0.28 0.44 0.5 0.33 0.67 0.72 0.94 0.22 0.39 0.5 0.33 0.33 0.33 0.11 0.28 0.39 0.39 0.39 0.22 0.17 0.17 0.44 0.22 0.39 0.28 0.39 0.11 0.67 0.11 0.33 0 0.56 0.33 0.39 0.44 0.39 1**

# Requirements

* Implement the K-MEANS algorithm explained in the class (see above). Use first K points as initial positions of the centers of the clusters. In case that in some iteration there will be no points in cluster – keep its center for the next iteration.
* The input file initially is known for one machine only. The output file has to be written to the file on the same machine.
* The computation time of the parallel program must be faster than the one for sequential solution.
* Be ready to demonstrate your solution running on at least three computers.
* **No code sharing between students is allowed.** Each part of code, if any, which was incorporated to your project must be referenced according to the academic rules.
* Be able to explain each line of the project code, including those that was reused from any source.
* The set contains at least **100** but not more than **5000** products. The maximum number of clusters **MAX** will be less than **100**.

# Grade Policy:

* **60 points** for the effective **proper** parallel implementation of the problem with two components: ***MPI+OpenMP*** or ***OpenMP+ CUDA*** or ***MPI+CUDA***. The project that produce wrong results will not be accepted
* **10 points** for implementation in ***MPI+OpenMP+CUDA*** configuration.
* **10 points** for the documentation of your solution – clear explanation what and how the problem was parallelized, what is a rational of choosing the specific architecture, complexity evaluation.
* **10 points** for the code quality – modularity, generality, self-explanatory, organization.
* **10 points** for the Load Balancing.

***Additional Bonus for the project grade***

**5 points** for implementation under LINUX OS

**5 points** for implementation with OpenCL

**5 points** for implementation of sophisticated variation of the K-MEANS algorithm(have to be approved by lecturer).

**5 points** for your own proposal (have to be approved by lecturer).

# הפרויקט יוגדר כמטלת הקורס. הגשת התוכנה והתיעוד רק דרך מערכת Moodle.

# יישום והגשת הפרויקט ביחידים בלבד.

בהצלחה

1. Sales Transactions Dataset Weekly Data Set   
   <https://archive.ics.uci.edu/ml/datasets/Sales_Transactions_Dataset_Weekly>, last seen on 1/09/2107 [↑](#footnote-ref-1)