International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:11, No:10, 2017

A Parallel Implementation of k-Means in MATLAB

Dimitris Varsamis, Christos Talagkozis, Alkiviadis Tsimpiris, Paris Mastorocostas

***Abstract*—**The aim of this work is the parallel implementation of k-means in MATLAB, in order to reduce the execution time. Specifically, a new function in MATLAB for serial k-means algorithm is developed, which meets all the requirements for the conversion to a function in MATLAB with parallel computations. Additionally, two different variants for the definition of initial values are presented. In the sequel, the parallel approach is presented. Finally, the performance tests for the computation times respect to the numbers of features and classes are illustrated.

***Keywords*—**K-means algorithm, clustering, parallel computations, MATLAB.

I. Introduction

K-MEANS is a popular algorithm that solves clustering problems [1]. This algorithm is simple and uses unsupervised learning to converge to solution. K-means needs only a certain number of clusters (k). A data set can be classified to this certain number of clusters by the algorithm procedure. K-means is a small repetitive part of a solution applied to many problems. Specifically, k-means is used in Feature Selection [2], [3], in Subsets Selections problems [4] and generally in any problem that requires clustering.

The execution of k-means algorithms requires large computational cost [5], [6], that leads researchers to implement k-means using parallel techniques. MATLAB is a software tool that supports, conveniently, numerical computations and parallel techniques [7]-[9]. Additionally, MATLAB supports parallel computations either in cluster of computers or in multicore CPUs [10].

In Section II, the serial implementation of k-means in MATLAB is presented. This implementation consists of two variants of k-means, corresponding to two ways of initialization. In Section III, a parallel implementation of algorithms from Section II were made, using MATLAB software tools and commands. Finally, in Section IV the implemented algorithms are tested for performance. The tests include both serial and parallel implementations of k-means from Sections II and III and additionally the k-means build-in implementation of MATLAB.

D. Varsamis is with the Department of Informatics Engineering, Technological Educational Institute of Central Macedonia, 62124 Serres, Greece (e-mail: dvarsam@teiser.gr).

C. Talagkozis is with the postgraduate program in Applied Informatics of Department of Informatics Engineering, Technological Educational Institute of Central Macedonia, 62124 Serres, Greece.

A. Tsimpiris is with the Department of Informatics Engineering, Technological Educational Institute of Central Macedonia, 62124 Serres, Greece (e-mail: alkisser@gmail.com).

P. Mastorocostas is with the Department of Computer Systems Engineering, Piraeus University of Applied Sciences, 12244 Egaleo - Athens, Greece (e-mail: mast@puas.gr).

II. Serial k-Means Implementation

A clustering algorithm was implemented using the MATLAB software tool. This algorithm was made from the beginning, based on the rules and steps of k-means implementation methods. The main requirement in development of serial k-means algorithm in MATLAB function was the availability to convert in MATLAB function using parallel techniques. The following MATLAB functions implement k-means in two variants. These variants correspond to two ways of initialization. The implemented functions take as input a set of data, where the lines are the patterns (n) and the columns are the features (m) of these patterns. In addition, they get the number of classes (k) that must be separated (nClusters). Finally, these functions take as a parameter the number of repetitions (tolerance) that will repeat the whole algorithm in order to avoid unfortunate bad initializations. It is important to note that due to the initial random conditions, it is likely to carry out a large number of steps to converge to the minimum sum of the distances (BCSS).

A. Random Centroids

This implementation of k-means in MATLAB picks random patterns and sets them as the initials centroids. This implementation of k-means is called Random Centroids hence forward **RC**.

B. Random Assignments

This implementation of k-means in MATLAB assigns every pattern to any class randomly and then calculates the centroids based on the assignments. This implementation of k-means is called Random Assignments hence forward **RA**.

III. Parallel k-Means Implementation

One of the major disadvantages of k-means is the complexity of $O(ndk + 1)$, where k is the number of classes and d is the number of dimensions. This disadvantage causes problems when the algorithm is executed in very large sets of data. This problem has a large computational cost, hence the long execution time. For this reason, an attempt was made to parallelize the serial implementations.

The most obvious way to parallelize the algorithm is to simultaneous calculate all repetitions within tolerance [11], [12]. Since the solution of the function is the best solution for all iterations of the k-means algorithm, it is possible to execute all of them in parallel and, after finishing, making the selection of the best solution that minimizes the sum of the distance of the standards from the center of the groups to which they belong. Both the first version with the random assignments of the centroid, and the second implementation with the random

assignments of the models in the groups can be parallelized. The parallelization is available because the serial MATLAB functions RC and RA are implemented using the principles of parallel programming in MATLAB [7], [8] with appropriate loops, variables, statements, indexing, matrices etc. Then, the MATLAB functions RC and RA are easy parallelized to parallel MATLAB functions Random Centroids Parallel hence forward **RCP** and Random Assignments Parallel hence forward **RAP**, respectively.

IV. Performance Tests

The performance tests are implemented in an efficient computing system with the following characteristics:

- CPU Intel Xeon E5640 64x 2.67GHz (multicore)
- RAM 16GB

Additionally, for the accuracy of the performance tests, the execution time of the tests are calculated with the formula is given by

$$Time = \frac{t_1 + t_2 + t_3 + \ldots + t_{12} - t_{Max} - t_{Min}}{10}$$

where $t_i (i = 1, ..12)$ is the execution time of each run with the same data and parameters.

A. Parameters of Performance Tests

The parameters of the k-means algorithms (RC, RA and build-in function of MATLAB `kmeans()`) are the following:

- n, the number of patterns
- f, the number of features
- t, tolerance (default value in `k-means()` is 100)
- c, number of classes

Two different performance tests are implemented. The first one runs with respect to the number of classes (c), while the second one runs respect to the number of features (f). In particular, the values of parameters are

- $n = 100.000$
- $f = 1, 2, 3, 4, 5$
- $t = 100$
- $c = 2, 4, 5$

The corresponding data are created using the build-in function of MATLAB `rand()`. The results are shown in table I

All MATLAB functions (k-means, RC, RA, RCP and RAP) converge in the same value of measurement: between-cluster sum of squares (BCSS) in tests that presented in Table I.

B. Performance Tests with Respect to the Number of Classes (c)

In the following figures the execution times for

$$c = 2, \quad c = 4, \quad c = 5$$

of `kmeans` (MATLAB build-in function), RC, RA, RCP, RAP (MATLAB user defined functions) are presented. In Figs. 1-5 the number of features is constant and equal to 1, 2, 3, 4 and 5, respectively.

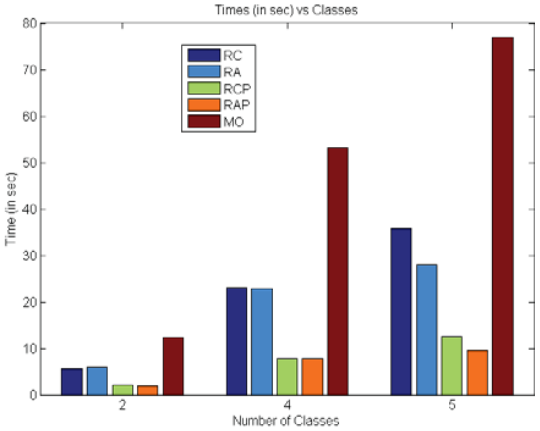

Fig. 1 Execution time of functions for $f = 1$, and $c = 2$, $c = 4$, $c = 5$

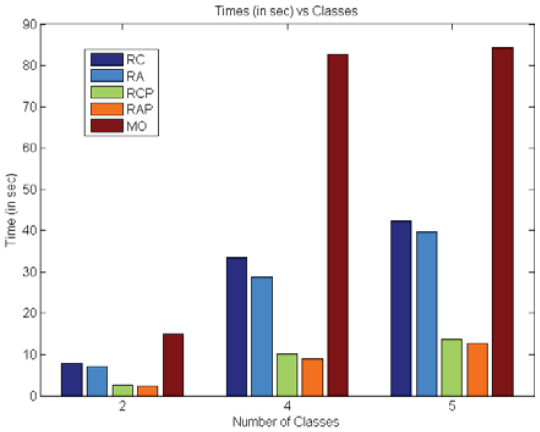

Fig. 2 Execution time of functions for $f = 2$, and $c = 2$, $c = 4$, $c = 5$

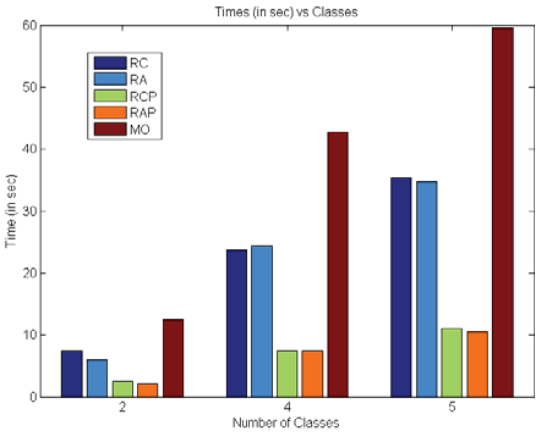

Fig. 3 Execution time of functions for $f = 3$, and $c = 2$, $c = 4$, $c = 5$

C. Performance Tests with Respect to the Number of Features (f)

In following figures the execution times for

$$f = 1, \quad f = 2, \quad f = 3, \quad f = 4, \quad f = 5$$

of `kmeans` (MATLAB build-in function), RC, RA, RPC, RAP (MATLAB user defined functions) are presented. In Figs. 6-8

TABLE I
COMPARISON OF SERIAL AND PARALLEL EXECUTION

c	T	f	n	RC	RA	RCP	RAP	k-means()
2	100	1	100000	5.616309	5.887572	2.103547	2.049393	12.26237
2	100	2	100000	7.789931	7.042285	2.653606	2.392734	15.10898
2	100	3	100000	7.391408	6.029671	2.562592	2.13248	12.54087
2	100	4	100000	11.30147	9.200706	3.841012	2.979793	15.92419
2	100	5	100000	11.97893	10.21499	3.743564	3.232818	16.67794
4	100	1	100000	23.00679	22.89751	7.888749	7.882546	53.15672
4	100	2	100000	33.42405	28.7702	10.24733	8.936011	82.71037
4	100	3	100000	23.72343	24.34376	7.398941	7.444965	42.70766
4	100	4	100000	31.41331	31.81684	9.215425	9.399122	46.43585
4	100	5	100000	34.88539	34.91674	10.06608	9.869597	50.23536
5	100	1	100000	35.83778	27.98714	12.46852	9.626066	77.0053
5	100	2	100000	42.45209	39.70101	13.6033	12.71917	84.22828
5	100	3	100000	35.37246	34.70817	11.07625	10.58004	59.57364
5	100	4	100000	35.4924	36.54768	10.89426	10.89837	57.80143
5	100	5	100000	45.63234	46.8906	13.55203	13.79513	64.13661

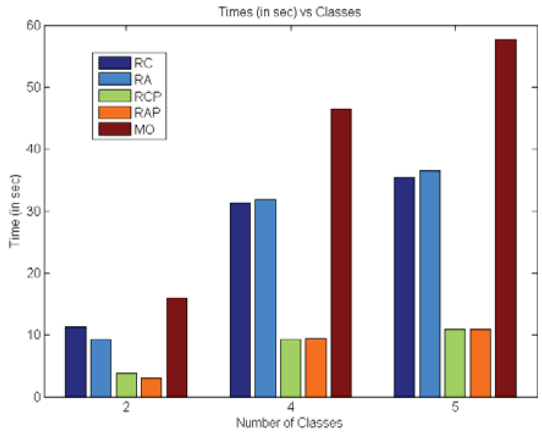

Fig. 4 Execution time of functions for $f = 4$, and $c = 2$, $c = 4$, $c = 5$

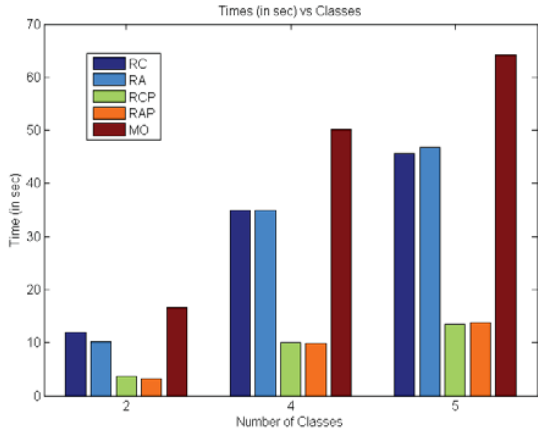

Fig. 5 Execution time of functions for $f = 5$, and $c = 2$, $c = 4$, $c = 5$

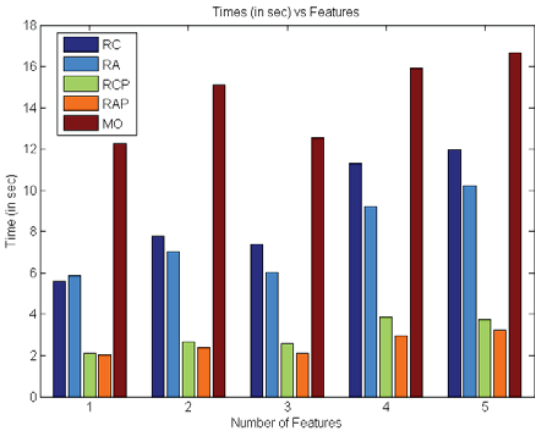

Fig. 6 Execution time of functions for $c = 2$, and $f = 1$, $f = 2$, $f = 3$, $f = 4$, $f = 5$

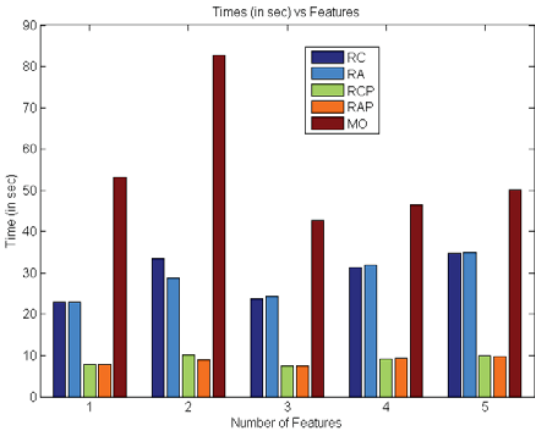

Fig. 7 Execution time of functions for $c = 2$, and $f = 1$, $f = 2$, $f = 3$, $f = 4$, $f = 5$

the number of classes is constant and equal to 2, 4 and 5 respectively.

V. CONCLUSIONS

From the aforementioned analysis it becomes evident that the parallel implementations of k-means lead to ameliorated

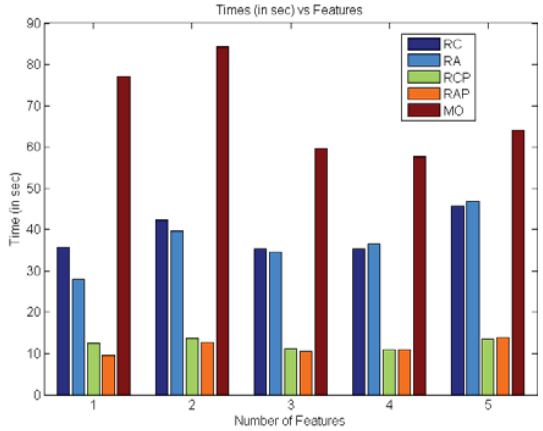

Fig. 8 Execution time of functions for $c = 2$, and $f = 1$, $f = 2$, $f = 3$, $f = 4$, $f = 5$

performance. The parallel implementations converge to the same solution as all the serial ones, in reduced execution times. This is happening with respect to both the number of classes and the number of features. Additionally, it is noticed that the parallel algorithms have a lower increase rate as the number of classes or the number of features increase. Comparing the execution times among serial implementations, it is clear that the implemented serial k-means converge faster to the solution than the build-in function of the MATLAB software tool. It needs to be noticed that both serial (RC, RA, build-in MATLAB `k-means()`) and parallel (RCP, RAP) implementations run on the same computing system, with the same resources. The parallel algorithms take advantage of the resources of the computing system, namely the cores of CPU.

Acknowledgment

This work was supported by the postgraduate program in Applied Informatics of Department of Informatics Engineering, Technological Educational Institute of Central Macedonia-Serres.

References

[1] P. Sneath and R. R. Sokal, *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. San Francisco: W.H. Freeman, 1973.

[2] A. Tsimpiris and D. Kugiumtzis, "Feature selection for classification of oscillating time series," *Expert Systems*, vol. 29, no. 5, pp. 456–477, 2012.

[3] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

[4] G. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 1994, pp. 121–129.

[5] D. Arthur and S. Vassilvitskii, "On the worst case complexity of the k-means method," Stanford InfoLab, Technical Report 2005-34, 2005.

[6] ——, "How slow is the k-means method?" in *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, ser. SCG '06, New York, NY, USA, 2006, pp. 144–153.

[7] P. Luszczek, "Parallel programming in matlab," *International Journal of High Performance Computing Applications*, vol. 23, no. 3, pp. 277–283, 2009.

[8] G. Sharma and J. Martin, "Matlab : A language for parallel computing," *International Journal of Parallel Programming*, vol. 37, pp. 3–36, 2009.

[9] D. N. Varsamis, P. A. Mastorocostas, A. K. Papakonstantinou, and N. P. Karampetakis, "A parallel searching algorithm for the insetting procedure in matlab parallel toolbox," in *Federated Conference on Computer Science and Information Systems (FedCSIS), 2012*. IEEE, 2012, pp. 587–593.

[10] C. Moler, "Parallel matlab: Multiple processors and multiple cores," *The MathWorks News & Notes*, 2007.

[11] C. Lin and L. Snyder, *Principles of Parallel Programming*. Boston, USA: Addison-Wesley, 2008.

[12] D. Varsamis, C. Talagkozis, P. Mastorocostas, E. Outsios, and N. Karampetakis, "The performance of the matlab parallel computing toolbox in specific problems," in *Advanced Information Science and Applications Volume I, 18th Int. Conf. on Circuits, Systems, Communications and Computers (CSCC 2014), July 17-21, 2014, Santorini Island, Greece*, vol. 1, 2014, pp. 145–150.