A Reconfigurable System Learning for Data Classification Using Parallel Processing

E. M. Moreira[1], **C. D. Maciel**[2], **E. M. Moreira**[3], **and F. S. Zanoni**[2]
[1]Federal Institute of Education, Science and Technology of São Paulo,
São João da Boa Vista, São Paulo, Brazil
[2]Signal Processing Laboratory, Department of Electrical Engineering / University of São Paulo,
São Carlos, São Paulo, Brazil
[3]Systems Engineering and Information Technology Institute / Federal University of Itajubá,
Itajubá, Minas Gerais, Brazil

Abstract— *This paper presents a System Learning with a task scheduler, which makes possible the utilization of several classification and validation methods, allowing the distribution of tasks between the module systems. This architecture is structured of such way that the classifications obtained through a specific technique can be reutilized in parallel by the same algorithm or by other techniques, producing new classifications through the refinement of the results achieved and expanding the use in databases with different characteristics. The results demonstrated that through a learning system, the complexity of the analysis of great databases is minimized, allowing to verify basis with different structures and to increase the methods applied in the analysis of each structure. It favors the comparison between the methodologies and provides more reliable results.*

Keywords: Machine Learning; System Learning; Clustering; Data Partitioning.

1. Introduction

The procedure called *data clustering* constitutes in a complex technique, which has being studied in several areas aiming to find patterns in data, considering different theories and methodologies [1]. This technique is focused on the development of partitioning procedures of data sets in order to join similar objects, generating subgroups [2]. In this partitioning procedure, there is a significant relationship between arguments, metrics of similarities and structure of data. Considering the specific characteristics of each database, it is not trivial to reach a satisfactory result in the partitioning procedure in many cases. For this reason, the manipulation or configuration of these elements is a fundamental pre-requisite to obtain suitable results. In this way, aiming the elaboration of subgroups through the evaluation of important properties to the total group, it is necessary a previous knowledge regarding the context in which the major group is involved in each analysis; its specific insertion areas; and its general relevance, independently of any subdivisions. Furthermore, it is interesting to identify elements that, in fact, have important characteristics to the large group, in such way these subdivision can contribute to the understanding of the inherent properties to the same subgroups as well as the original total group. Usually, these studies require great statistic evaluation and significant technical analysis, including, in several times, a contribution of subjective studies from specialists of the subject evaluated in each work.

In turn, the information of interest are contained in large database, wherein the data manipulation involves specific tasks with higher computational overhead. During the data manipulation, the results obtained by a classification methodology are frequently sensitive to a specific data base. Several works are focused on distincts techniques of classification which are dependents of parameters related to the data originated from the respective study [3], [4], [5]. Considering the peculiarities of each base and the impact referent to a small alteration in the method employed regarding the final partitioning, it is important to make a rigorous evaluation of the classifications obtained through these partitioning tasks.

In this context, it is interesting to consider the possibility of creating a refinement method which uses the characteristics of several other methods, since there is no guarantees that a particular method will clustered the data correctly, according to the understanding of the specialists that are contributing with the subjective task of interpretation of the data. In fact, various papers present this difficult about the classification procedure [6], [3], [7].

As function of this fact, works that associated and/or compare different classification methodologies become essential procedures to the choice of the best data treatment, depending of the several factors inherent to each case [8], [6].

Considering the difficulties of creation of a unique methodology to a satisfactory classification of distinct data bases and the difficult of interpretation of the implications related to the large volumes of data in each base, the classification efforts can, frequently, to be correlated to a significant increase in the inherent computational cost [9].

In this way, the present work proposes the development

of a parallel learning system that has the objective to administrate classification methods, partitioning refinement methods as well as the validation methodology of results to be applied in the data bases.

This article is structured in the following way: the next section presents the structure of the proposed system with the respective implemented methodologies; subsequently, it is presented the section focused on the experiments and results obtained. In the last section, the final considerations and proposal to the continuation of this work are presented.

2. Materials and Methods

2.1 System Learning Proposal

The System Learning described in this document presents four modules: `Initialization Module`, `Validation Module`, `Refinement Module`, and `Special Scheduling Module`. In each module can be used several algorithms to meet their goal.

Figure 1 shows the design of this system learning, allowing you to see how these modules interact. The activities of each module will be detailed in the following subsections.

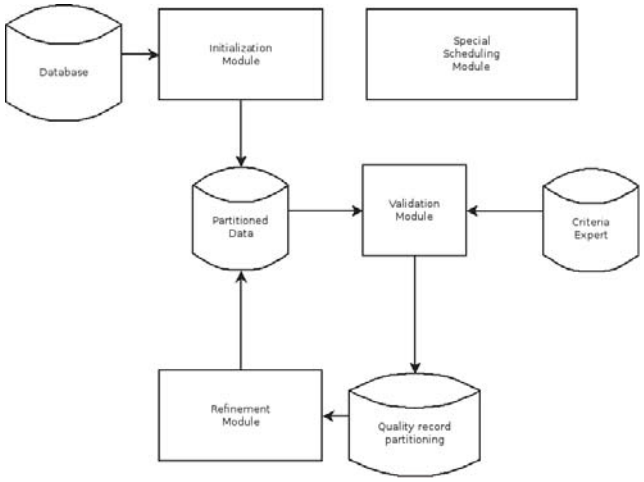

Fig. 1: Figure shows the structure of the system and the form in which the modules are related. The data set is firstly treated by the `Initialization Module` generating the first set of partitioned data. The partitioned data sets are assessed by the `Validation Module`, which employs different criteria and expert rules to record the quality of the partition. Parallely to this process, `Refinement Module` generates more partitions to be evaluated from the partitioned data set by `Initialization Module` and Refinement own module. `Special Scheduling Module` is responsible for the administration of the activities

2.1.1 Initialization Module

The goal of the `Initialization Module` is to prepare the original data to be classified. This module create an initial partitioning and organize the data in a pattern understood by the *Validation* and *Refinement* modules. In the `Initialization Module` were implemented four algorithms to generate initial partitions: *Binary Tree*, *Random*, *Quasi-random*, and *Fiedler Method*, that is a spectral method.

Noteworthy is the method of *Binary Tree* which, due to the analysis that is performed on the data during its execution, creates a partition with great potential to be the end result or next to it. Methods *Random* and *Quasi-random* are algorithms based on random mechanisms which, although simple, may achieve good results due to parallel processing and integration between modules for validation and refinement.

All classification produced in `Initialization Module` or `Refinement Module` receives a value that represents the quality of the created partition. To this value was given the name of **Quality Attribute**. The way of this attribute is calculated can vary with the type of data or according to the methods used.

The *Binary Tree* Method utilizes the *Binary Tree* data structure for construction of partitions. Each partition is represented by a tree that stores alike objects. The algorithm starts with the construction of a similarity matrix from the attributes of the original data file. When analyzing a set of data, or attributes, such as dimensions of a multi-dimensional space, the description of each figure corresponds to a point in that space. Therefore, the distance between pairs of objects can be used as a degree of similarity between them. In implementing this method, it was used the Euclidean distance between the geometric data [10].

The *Binary Tree* method performance basically the construction of a dendogram [11], that are convenient ways of depicting pairwise dissimilarity between objects. The construction of the *Binary Tree* is done in order *bottom/up*. Initially, it has n distinct sets or partitions, where n is the cardinality of the original data set. From the similarities matrix, two closest objects are identified. These objects will be merged into a single tree and thus become the left and right subtrees of the new tree. Thus, the number of partitions decreases by one. The matrix of similarities is redone by excluding the rows and columns referring to the objects, and by inserting a new line whose reference point in space is the average distance between the two objects chosen for the other partitions. This method is known as *Average Link* [12].

Importantly, the definition of the new coordinate location of the partition can be accomplished in several ways, for example, the use of the coordinate of the midpoint between the objects belonging to the partition or by using the coordinate of the object, which belongs to the partition that is more close relative to the other partitions, a method known as *Single Link* [13]. These changes were implemented and the proposed system run in parallel by providing,

to Validation Module, distinct partitions for analysis. The procedure for choosing the partitions closer repeats until it reaches the desired number of partitions.

The *Random* method is very simple. From an initial seed, there is a random choice of the objects that will compose each partition. Obviously, this method does not start of any knowledge of the data and therefore produces clusters whose **Quality Attribute** is inferior. The biggest advantage of this method, besides its simplicity, is the speed with which the partitions are created from new seeds of random numbers.

Turn, the semi-random method is a variation of the *Random* method. Despite using random choices, the objects are not chosen within the entire universe of the original data. A pre-assessment is conducted to the choices that are made of objects into subsets with the highest similarity among its components. The starter sets are constructed from a matrix of similarities. Initially, two objects are chosen randomly for each subset. The distance between these objects will be used as reference in the choice of other objects. The remaining objects are also randomly chosen, however, are inserted into the set consisting of the smallest distance between objects in the set and itself. Every time an object is inserted in the set, the reference center of the set is recalculated to be used in the next comparisons.

In 1973 and 1975, M. Fiedler published papers on the properties of the Laplacian matrix of the eigensystems [14], [15]. His studies began with the contribution of the paper by Anderson and Morely on the eigenvalues of the Laplacian matrix [16]. In 1990, Pothen, Simon, and Liou published a paper Applying Fiedler's ideas to the field of clustering [17], being the beginning of the theory for spectral graph partitioning. The implemented version of this method partitions the dataset into two subsets, based on positive and negative values of the eigenvector matrix Laplacian. The Laplacian matrix was obtained from the matrix of similarities, so it is a weighted matrix corresponding to a complete graph. In order to achieve the desired number of partitions, each subset was created recursively subdivided by applying the same method again.

After calculating the **Quality Attribute** of a partitioning, the Validation Module records that information in the 'Register Quality Partitioning'. At the criterion of the specialist, we can set minimum quality values for a partitioning to be handled by the Refinement Module. However, validation of an entire partition from the Initialization Module must necessarily pass through the Refinement Module.

2.1.2 Validation Module

Once a classification has been created by Initialization Module, or enhanced by the Refinement Module, it is necessary to check the quality of the partition. Usually, this task requires a specialist who will use the results of partitioning. However, based on information from these professionals, it is possible to create a rule base (Criteria Expert) that automates this process or at least deletes the partitions that are below a minimum standard of quality. For example, instead of using all attributes related to an object of the base for calculating the Euclidean distance, specific information about the data could be used to optimize the calculation of similarity. Thus, the specialist will assess only those partitions that have achieved a good value on **Quality Attribute**.

For the tests performed with the data in this article, the **Quality Attribute** used was the sum of the mean metric distances among all pairs of objects in the same partition. Obviously, other statistical parameters can be used according to the rules laid out on the basis of 'Criteria Expert'.

After calculating the **Quality Attribute** of a partitioning, the Validation Module records that information in the 'Register Quality Partitioning'. At the discretion of the specialist, minimum quality values can be set for a partitioning to be handled by the Refinement Module. However, validation of an entire partition from the Initialization Module must necessarily pass through the Refinement Module.

2.1.3 Refinement Module

Refining is to improve the quality of a partitioning previously realized. This is done by exchanging objects between partitions. This exchange helps to increase the value of **Quality Attribute**.

Once the exchange is performed and a new classification obtained, the Validation Module verifies the impact of this exchange. When the exchange process is successful, the partitioning obtained is available as a result achieved and as a started point to a new refinement. On the other hand, if the **Quality Attribute** value indicates a quality deterioration of original classification, it is discarded.

The integration of Validation Module and Refinement Module provides application of the evolutionary method that is the base of the genetic algorithms. The evolutionary paradigm has been used in research of the machine learning.

According Zhang [18], the steps performed by an evolutionary algorithm have two basic stages: **initialization**, which from a default configuration or a random configuration generates the first population, and **generation**, that is based on three functions: fitness, crossover and mutation.

In this work, the population of individuals is represented by a classification obtained and its **Quality Attribute**. The main genetic operators are crossover and mutation. The crossover operation is implemented through specific techniques developed in the Refinement Module, described below. The mutation operation is not used at this work, since changes always occur in the same individual. Fitness function is used to measure the quality of an individual and, as already discussed, this is done by the Validation

Module, through the calculation of **Quality Attribute**.

The main problem of the genetic algorithm is its computational overhead, especially due to the fitness function, which has to be repeatedly assessed using the evolutionary process. This signals the importance of the application of this technique on a cluster of computers [19].

As the essence of Refinement Module are the mechanisms that promote the exchange of objects, several mechanisms can be used for this purpose, an example used as base is the classic method *k-medoid* [20], because it is based on the most centrally located object in a cluster, it is less sensitive to outliers in comparison with classic method *K-means* [21].

For the initial composition of this system, three methods was created, *Radius Coverage*, *Query by Range*, and algorithm variation of the Kernighan and Lin [22] for partition graph.

By means of partitions defined in the Initialization Module, or previous results of partitioning, this method computes radius of coverage from of centroid of each partition, depending of method used, that is, the radius is distance between the center point and the most distance object that belongs to partition. In this sense, it can identify spheres involving objects of each partition, however, it is possible that objects of different partitions are reached by the radius of other partitions. These objects are in the areas of intersection between two or more partitions. From the identification of candidate partitions, ie, partitions which their coverage radius reach the object, which is chosen the partition where distance between the object and its *centroid* is the smallest. The object is displaced from its original partition to the partition chosen. A variation of this method is the possibility of increasing the radius of coverage to identify other objects in the intersection set. This increase variations have been implemented. The rate of increase is directly related to the value of **Quality Attribute**, i.e., dependent on the data of the problem.

The Coverage Radius Method performs a query on all data, identifying nearest *centroid*. If the *centroid* found is not the same, the method transfers the data to the partition of this *centroid*. After method verify all data, the system accesses the Validation Module for a new **Quality Attribute**. An interesting feature of this method is that, depending on the data set, there is a convergence of found partitions, since as they happen refinements, *Radius Coverage* tend to decrease, helping the Special Scheduling Module in the criterion stop.

Importantly, these methods were applied in sets ever built by the algorithms implemented in the Initialization Module. Thus, the choice of candidate objects to exchange the set was made from range queries in relation to the cluster center (centroid or medoid). Also, experiments were performed by increasing the radius of coverage with the aim of expanding the sets of intersection.

The Kernighan and Lin [22] proposed one of the earliest methods for graph partitioning, and more recent local improvement methods are often variations on their method. This method works primarily evaluating the gain in reducing the cost cuts between partitions during an exchange between pairs of vertices of different partitions. The algorithm developed in this study randomly selects two partitions and chooses an object that is farthest from the centroid of the first partition and does the same on the second partition neighbor, selecting the most distant object of its respective centroid. After selection, the algorithm calculates the gain to be obtained from the **Quality Attribute** if this change is made. If there is no gain, a new object is chosen from the second partition, ie, the second most distant object of its centroid, and so on. When two objects are chosen, a new object from the first partition is selected and the procedure repeats. When there are more gains the algorithm stops. It is important to note that the algorithm stops before replacing all the objects (which would be to simply rename the two partitions), as by changing all the objects, there would be no gain.

At the end of the algorithm, a new partitioning is obtained and therefore is ready for a new analysis by the Validation Module.

2.2 Special Scheduling Module

As presented, the architecture of this system learning enables its implementation on a parallel computer or a distributed architecture, for example, a cluster of computers.

The Special Scheduling Module features, in this context, two main functions: scheduling jobs produced by the other modules and identifying the time to stop the system learning.

The Initialization Module maintains a continuously running process (*trigger process*). Basically, this process is responsible for communication with the Special Scheduling Module. Its function is to wait for the signal of availability (when there is any ready processor to perform the task) and create a new task (the execution of an algorithm for a specific method, which has not yet been used, or the execution of a method already used with new values for its configuration attributes).

In Special Scheduling Module there is a *Scheduler Process*. This process maintains a multilevel priority queues system. Tasks are grouped into three classes of priority: tasks of the Validation Module, tasks of the Refinement Module and tasks of the Initialization Module. Figure 2 presents the structure of multilevel queues. The queue refers to the processes of Validation Module has the highest priority, i.e. no process to other queues that are waiting will be choose if there is any process in this queue. This is necessary because, since there are classifications already made, the validation should be performed with priority over the de-

velopment of new partitioning. The queue relating to the `Initialization Module` has the lowest priority, i.e., its processes can only be chosen if the other queues are empty. It is important to note, however, that in the initial operation of the system learning, all queues are empty and therefore the tasks of generating initial classifications will be executed, because there will be no ratings to be validated or refined, with their queues being empty.

The `Validation Module` has a unique type of process (*Quality Evaluator*), which is responsible for assessing the quality of a partitioning obtained. Every time a process of `Initialization Module` or `Refinement Module` finishes its execution, there is a new partition available to evaluation. In this context, a new process *Quality Evaluator* is created to analyzed the classification obtained. This process is thus inserted into their priority queue, waiting for the availability of a processor to select the scheduler.

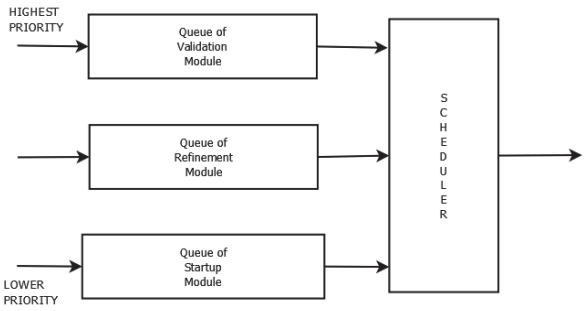

Fig. 2: Scheduling with Multilevels Queues - `Special Scheduling Module` defines priorities in the administration of the activities of the system. `Validation Module` has higher priority system that `Refinement Module` and this has a higher priority than the `Initialization Module`

An important issue to be considered concerns the criteria that the `Special Scheduling Module` uses to stop scheduling new tasks. When the system verifies that the partitioning presented by `Refinement Module` has been produced by a previous method (in `Initialization Module` or another refinement by the same method), there is no need to submit it for processing because it would result in data replication, the unless it is used another refinement method.

Another stop criteria used verifies the analysis of historical values for the **Quality Attribute** using the same method of refinement. When the values surrounding a reference value, indicating the proximity of a local minimum, `Special Scheduling Module` interrupts the scheduling of new tasks. It is noted that approaches to identify the local minimum can vary depending on the used database.

Obviously there are databases which partitions produced in the `Refinement Module` is not repeated because of combinatorial explosion caused by the amount of information. Furthermore, the permutations obtained from each refinement can significantly change the **Quality Attribute** and, in this case, it is not possible to identify the approach of a local minimum. In these situations, it is possible to define as stop criterion a maximum number of refinements from a partition obtained by one method of the `Initialization Module`.

3. Experiments Results

In order to obtain comparison, it was elaborated the table 1 using Iris data set, which presents the values of the **Quality Attribute** initially obtained in the methods already implemented in the system proposed, in agreement with description in the section 2.1.1. This table presents also the **Quality Attribute**, in agreement with UCI repository [23]. This value is **2,82201**, which corresponds to the original partitioning of Fisher [24]. Considering the method of *Binary Tree*, it was obtained a **Quality Attribute** that is still better, **2,7228**. This result, which was considered significantly better than the original one, demonstrates the existence of superpositions between the classes.

After the generation of initial groups, these selections were utilized by algorithms of the `Refinement Module`. In agreement with previous discussion in section 2.1.3, the choice of the candidate objects to develop the exchange of partitioning was made, following the two methods described.

Table 1 presents the atualizated values of the **Quality Attribute** after the treatment of the **Refinement Module** with method of *Radius Coverage* and the method of *Query by Range*.

Data Set *Iris* - Quality Attribute			
Repository	*Binary Tree*	*Radius Coverage*	*Query by Range*
	2,69466	**2.70427**	**2.72368**
	Quasi-random	*Radius Coverage*	*Query by Range*
2,82201	**2.71932**	**2,70231**	**2.85162**
	Random	*Radius Coverage*	*Query by Range*
	5,68444	**2.71379**	**4.18003**

Table 1: Determination of the **Quality Attribute** realized by the `Validation Module`, which was applied in the partitionings generated by the methodologies of the initializing module in Iris data set

In the execution of the *Random* and Semi-Random methods, the results presented in the Table 1 corresponds to the better value obtained by the `Initializing Module` in **3.000.000**, considering the partitionings obtained presently.

Table 2 constitute abstracts of some results of the proposed system. The first column has the method utilized to the generation of the first partition as well as the index of correction encountered. The second column corresponds to the methods of refinement containing, to each one, two informations. The first one is the quantity of iterations that was required until to reach some stop criterium; the second

one corresponds to the correct index of the better partition obtained in the evaluation.

Data Set *Iris*					
Initialization Module		Refinement Module			
Method	Index Correct	*Radius Coverage* Iteraction	Index Correct	*Query by Range* Iteraction	Index Correct
Binary Tree	91,33%	1	88,66%	2	93,33%
Random	54,66%	3	88%	6	55,33%
Quasi-random	90,66%	1	91,33%	2	92%

Table 2: Analysis of data set *Iris* after partitionings generated by the **Initialization Module** and treatment realized by the **Refinement module**

Corresponding to the information presented by the Table 2, it is possible highlight the refinement reached by the *Radius Coverage* Method in the partitioning generated by the *Random* Method, in which the selection founded is proximal of the best partitioning obtained by the others methodologies. Considering the method of *Query by Range*, the **Refinement Module** originated tasks to the **Special Scheduling Module**, starting from the own radius of approach of the original selections and increasing the radius step by step in **0,02**, in the distance of Euclides approach.

In relation to the stop criteria, the behavior was different in each one of the methods evaluated. In the method of *Query by Range*, the interruption occurred when a new selection did not decreased the value of quality assignment. In its time, in the case of the *Radius Coverage*, the interruptions, in all bases, happened when the result of the refinement produced a group already in previous partitions.

It is important to notice that in the method of *Query by Range*, the test to the *Radius Coverage* increase was of **15%** of the original radius of the class.

All the modules implemented were simulated sequentially e parallely to the performance analysis of the system. This experience was performed in a cluster of computers with the following structure:

- Four computers with following specifications:
  - Intel(R) processor Core(TM)2 Quad CPU 2,66GHz;
  - Cache L2 8Mb;
  - 1066 MHz Frontal bus;
  - 2 GBytes RAM.
- The network has the following features:
  - 100 Mbits/s Ethernet;
  - *Switch*: 3Com Baseline Switch 2948-SFP Plus 48-Port Gigabit (3CBL SG48).

The implementations of the programs were developed in the C++ language with the version 1.5.4 of library OpenMPI. The operational system was Linux Fedora 13 with kernel version 2.6.33.5-112.fc13.

It is clear that because of the parallel system characteristic and structure of the scheduler, it can possible to make various refinements on the data already partitioned while there are other tasks carrying out the first partition in others data sets.

To evaluate the performance of parallel system methods, it was calculated the Speedup, which measures the reduction factor of runtime on P processors and finally the efficiency that is determined by the ratio between the speedup obtained and the number of processors used [25].

To strengthen the contribution presented in this article, further illustrating the behavior of the implemented system, the **Special Scheduling Module** worked with 25 Databases for classification, at the same time. The databases were taken from the UCI repository (Iris, Zoo, Glass, Balance and Wine) and replicated five times, totaling 25 databases in parallel. Were carried on the following methods for each instance: Binary Tree, Random, Quasi-random, Radius Coverage, and Query by Range.

Table 3 shows the system behavior manipulating **6335** attributes.

Table 3: Performance Learning System in the cluster of computers

Clusters	Time	**SpeedUp**	**Efficiency**
1	32m13.849s	-	-
2	16m14.887s	1,98	99,18%
4	9m7.396s	3,53	88,32%

To standardize the data originated of distinct bases, a data modelling system was elaborated, which will favor the uniform scalability of the system. In this sense, the implementations of other methods, that were added to the system, should handling the data as described in the model created, requiring converting the original data to adjust and insert them in the system database.

4. Conclusion

The main feature of the system learning that was presented in this paper was its flexibility in the composition of algorithms, which must be employed in data classification. Moreover, through the configuration files it is possible to easily change the data sets.

The examples in this paper were chosen in order to demonstrate the accessibility involving the practical employment of the present methodology, that is, in using this system learning. With few changes, new methods may be included in each module of the system. Considering the results presented, the increase of processors permits increasing the data sets to be analyzed simultaneously.

In the results presented, this system is efficient because it searching good partition, even when the `Initialization Module` get a bad partition. The use

of `Refinement Module` can correct a bad choice of initialization seeds. This example is showed in `Refinement Module` when it works with the results presented by Randon Method in `Initialization Module`.

In addition, a graphical user interface is being developed to permit in the configuration of this system by the users. In this way, the perspectives of application of this proposed in several areas of knowledgment are very auspicious, which can achieve great methodological advancements in various protocols utilized by research groups of several areas.

References

[1] L. Xutao, Y. Yunming, L. J. Mark, and K. N. Michael, "On cluster tree for nested and multi-density data clustering," *Pattern Recognition*, pp. 3130–3143, 2010.

[2] M. Meila, "Comparing clusterings - an information based distance," *Journal of Multivariate Analysis*, no. 98, pp. 873–895, 2007.

[3] Y. Liu, X. Wu, and Y. Shen, "Automatic clustering using genetic algorithms," *Applied Mathematics and Computation*, 2011.

[4] B. Zahraie and A. Roozbahani, "Sst clustering for winter precipitation prediction in southeast of iran: Comparison between modified k-means and genetic algorithm-based clustering methods," *Expert Systems with Applications*, 2011.

[5] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert Systems with Applications*, 2013.

[6] R. K. Blashfield, "Mixture model tests of cluster analysis: Accuracy of four agglomerative hierarchical methods," 1976.

[7] F. Cao, J. Liang, and L. Bai, "A new initialization method for categorical data clustering," *Expert Systems with Applications*, no. 36, pp. 10 223–10 228, 2009.

[8] G. W. Milligan, "An examination of the effect of six types of error perturbation on fifteen clustering algorithms," *PSYCHOMETRIKA*, 1980.

[9] W. Yan, U. Brahmakshatriya, Y. Xue, M. Gilder, and B. Wise, "p-pic: Parallel power iteration clustering for big data," *J. Parallel Distrib. Comput.*, 2012.

[10] E. Eirola, G. Doquire, M. Verleysen, and A. Lendasse, "Distance estimation in numerical data sets with missing values," *Information Sciences: an International Journal*, vol. 240, pp. 115–128, 2013.

[11] C. Iyigun and R. T. S. U. of New Jersey New Brunswick. Graduate School New Brunswick, *Probabilistic Distance Clustering*. Rutgers The State University of New Jersey - New Brunswick, 2008. [Online]. Available: http://books.google.com.br/books?id=GRWIu1jc884C

[12] O. I. R. Handl, J. Knowles, J. Knowles, M. Dorigo, and U. L. D. Bruxelles, "Ant-based clustering: A comparative study of its relative performance with respect to k-means, average link and 1d-som," 2003.

[13] L. Hubert, "Approximate evaluation techniques for the single-link and complete-link hierarchical clustering procedures," *Journal of the American Statistical Association*, vol. 69, no. 347, pp. 698–704, 1974.

[14] M. Fiedler, "Algebric connectivity of graphs," *Czechoslovak Mathematical Journal*, no. 23, pp. 298–305, 1973.

[15] ——, "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," *Czechoslovak Mathematical Journal*, no. 25, pp. 619–633, 1975.

[16] W. N. Anderson and T. D. Morley, "Eigenvalues of the laplacian of a graph," 1968.

[17] A. Pothen, H. D. Simon, and K. P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, pp. 430–452, 1990.

[18] J. Zhang, Z. hui Zhan, Y. Lin, N. Chen, Y. jiao Gong, J. hui Zhong, H. S. Chung, Y. Li, Y. hui Shi, and Xian, "Evolutionary computation meets machine learning: A survey," *IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE*, pp. 68–75, 2011.

[19] P. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 2, pp. 121–144, march 2010.

[20] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2008.01.039

[21] *Data Mining and Analysis: Foundations and Algorithms*. Cambridge University Press, 2010.

[22] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 2, no. 49, 1970.

[23] A. Frank and A. Asuncion, "Uci machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[24] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[25] I. FOSTER, "Designing and building parallel programs: Concepts and tools for parallel software engineering. mathematics and computer science division - argonne national laboratory," pp. http://www–unix.mcs.anl.gov/dbpp/text/node4.html, 1995.