




Article

Gene Expression Analysis through Parallel Non-Negative Matrix Factorization

Angelica Alejandra Serrano-Rubio , Guillermo B. Morales-Luna  and Amilcar Meneses-Viveros 

Department of Computer Science, CINVESTAV-IPN, Av. IPN 2508, Gustavo A. Madero, San Pedro Zacatenco, Mexico City 07360, Mexico; gmorales@cs.cinvestav.mx (G.B.M.-L.); ameneses@cs.cinvestav.mx (A.M.-V.)

* Correspondence: aserrano@computation.cs.cinvestav.mx

Abstract: Genetic expression analysis is a principal tool to explain the behavior of genes in an organism when exposed to different experimental conditions. In the state of art, many clustering algorithms have been proposed. It is overwhelming the amount of biological data whose high-dimensional structure exceeds mostly current computational architectures. The computational time and memory consumption optimization actually become decisive factors in choosing clustering algorithms. We propose a clustering algorithm based on Non-negative Matrix Factorization and K-means to reduce data dimensionality but whilst preserving the biological context and prioritizing gene selection, and it is implemented within parallel GPU-based environments through the CUDA library. A well-known dataset is used in our tests and the quality of the results is measured through the Rand and Accuracy Index. The results show an increase in the acceleration of $6.22\times$ compared to the sequential version. The algorithm is competitive in the biological datasets analysis and it is invariant with respect to the classes number and the size of the gene expression matrix.

Keywords: gene expression analysis; clustering algorithm; non-negative matrix factorization; high-performance computing



Citation: Serrano-Rubio, A.A.; Morales-Luna, G.B.; Meneses-Viveros, A. Gene Expression Analysis through Parallel Non-Negative Matrix Factorization. *Computation* **2021**, *9*, 106. <https://doi.org/10.3390/computation9100106>

Academic Editors: Florentino Fdez-Riverola, Miguel Rocha, Roberto Casado Vara, Mohd Saberi Mohamad

Received: 23 August 2021
Accepted: 11 September 2021
Published: 30 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The gene expression analysis has been widely used to determine the mechanism of certain diseases in their early stages, and in determining gene expression profiles [1–3]. Data Mining (DM) techniques focus on extracting knowledge from large databases. Clustering algorithms within gene expression databases have been widely exploited in unsupervised learning techniques, whose objective is to segment data to obtain gene *clusters* with correlated behavior [4]. These types of algorithms describe gene functionality, regulation, and involvement in cellular processes [5]. The main objective of this analysis is to create clusters such that the intragroup variance is minimal and the extragroup variance is maximum. The algorithm defines the variance through a similarity metric and verifies whether a pair of objects are closely related. The clustering analysis can be achieved based on genes and experimental samples. The grouping in genes describes sets of genes with a level of expression correlated in all experimental conditions; on the other hand, grouping in experimental conditions presents a correlated expression profile in all genes [6]. The group can be partial or complete; the partial clustering tends to be more flexible since all the elements should not necessarily be grouped, while in the complete group, all the elements must be assigned to a specific group. In the gene expression analysis, the partial grouping allows excluding genes that could represent a source of noise derived from the biological experiment. The cluster can be classified in a hard or soft clustering [7]. The hard clustering assigns a gene to the group with the dominant degree of belonging, while the smooth pool allows the assignment of a gene to more than one group. There is no clustering algorithm that presents the best performance for any type of problem [8–10].

There is a wide range of approaches. The standard hard-partitioned clustering algorithms, such as K-means (KM) [11], are simple and easy to implement, but they have

drawbacks when clustering biological due to high dimensional data parsing [12]. In [13], a joint K-means-based Co-Clustering algorithm (KCC) is proposed for sparse and high-dimensionality data. KCC uses higher-order statistics in order to the assignment of objects to each group, as well as for the estimation of centroids in the iterative process. The fuzzy C-Means Algorithm (FCM) [14] is one of the most used diffuse grouping methods for the gene expression analysis due to it carrying out a smooth grouping in which each object has a value of belonging to each group. The Self-Organized Map (SOM) [15] was introduced as an algorithm based on single-layer neural network methods, which generate an intuitively attractive map of a high-dimensional dataset in a two-dimensional space. Although SOM performs better than KM and FCM when analyzing noisy data sets, merging different objects in a cluster can make the algorithm ineffective at finding clustering limits and balanced solutions [16]. However, it is difficult to come up with appropriate input nodes and may result in a non-convergence problem of the algorithm. Furthermore, SOM is sensitive to choice of number of nodes such as KM [17].

Algorithms based on Hierarchical Clustering (HC) [18] group genes are naturally co-expressed through a graphical representation. However, a small disturbance in the data or a poor decision performed in the initial steps may greatly modify the final results. The authors of [15,16,19] stated that HC lacks robustness because in each iteration the groups are mixed locally by the distance between peers as an alternative of a global criterion. Another important limitation of HC is its high computational complexity [20]. The Self-Organized Tree Algorithm (SOTA) [21,22] was proposed as an alternative to obtaining robustness in the analysis of data prone to various sources of noise. SOTA combines the advantages of both SOM and HC [16]. It allows a visual representation of the clusters and their structure and is also more flexible than the other clustering methods. Agglomerative Nesting (AGNES) has been proposed in [23]. AGNES is a bottom-up approach. Compute all pair-wise pattern likeness coefficients and merge two interconnected clusters into one. The AGNES rigidity is vital for its success since it allows to obtain a solution in a relatively short period of time [24]. On the other hand, the Divisive Analysis algorithm (DIANA) [25] performs a hierarchical division due to starting with the entire population and ending once all the groups are separated following a top-down approach. This method has the same deficiency as AGNES; that is, it will never be able to repair what was performed in a previous step [26]. However, top-down clustering is more complex and efficient; also, it is more accurate, because a flat clustering is needed as a subroutine and for a fixed number of top levels. Grouping algorithms based on density approaches have been proposed in [27–29], whose objective is to identify high-density regions that are surrounded by low-density areas. Each of the density-identified regions is called a Cluster.

Robust algorithms based on factoring methods have been proposed [30] to reduce the data dimensionality and to apply conventional clustering algorithms. The most common algorithms to reduce dimensionality are the Roust Principal Components Analysis (PCA) [31], Singular Values Decomposition (SVD) [32], Independent Components Analysis (ICA) [33], and Non-negative Matrix Factorization (NMF) [34]. Even though PCA captures the variance of the dataset from a small group of genes, its application is limited to a linearity assumption [35,36]. ICA identifies additional substructures by projection vectors that represent independent features; however, its application implies complex cancellations between positive and negative values, entailing that gene inhibition is hard to interpret [37]. Moreover, the factors and coefficients must be restricted to a non-negative environment because a gene can be either present with a variable intensity or absent. NMF is useful since it allows to describe biological data as the product of two positive matrices with a semantic consistent with the biological context [38–40]. Despite the advantages offered by NMF, most of the implementations [41–44] have become obsolete due to the constant data increase defined by the curse of dimensionality [45].

The authors of [46–50] demonstrated that Clustering algorithms based on NMF, as well as software tools developed for this purpose, are efficient. However, these implementations have a high computational complexity and they are computationally demanding

on processing units (CPUs), so their applicability is limited in many circumstances [51,52]. In this way, current research should focus on improving computational performance by creating efficient algorithms from the already effective ones. To overcome this problem, High-Performance Computing (HPC) mechanisms based on general-purpose Graphics Processing Units (GPUs) are starting to be used within this area [53–55], as they can considerably reduce the execution time required in a CPU and allow more intensive investigations of biological systems.

The objective of this work is to propose a parallel clustering algorithm in a GPU for a gene expression analysis, such that it reduces the necessary computation time to obtain a result, without losing biological importance. The algorithm approach should focus on matrix factoring methods. In this way, the algorithms used in the proposal are NMF and KM. NMF is expected to reduce the dimensionality of biological data and then be able to apply KM, taking advantage of its high performance in low-dimensional data. The hypothesis of this work is based on the idea that the parallelization of NMF through High Performance Computing (HPC) mechanisms will help to reduce the required calculation time and expand its use in the high dimensional data to obtain a computational result according to the biological context of the study. To confirm or refute our hypothesis, we define NMF implementation as the most intensive computational task. The parallelization of this task on a GPU is proposed. We add the Consensus Clustering algorithm (CC) [56] to evaluate the stability of NMF and refine the selection of centroids in the implementation of KM. Our proposal is compared with the proposal described in [57]. The results show that our parallel implementation achieves a speed-up of $6.22\times$ versus our sequential version. Moreover, our proposal is invariant to the classes number and the size of the gene expression matrix.

The document is organized as follows: Section 2 describes the problem statement. Section 3 contains the proposed clustering algorithm. The sequential NMF analysis is given in Section 4 and the obtained maximum theoretical acceleration is exposed. In Section 5, the parallelization is explained. Section 7 contains the experimental evaluation of the implementation compared to other proposals in the state of art. Section 8 includes a discussion of the work, some implications and some limitations. Finally, in Section 9, we describe the main implications of the results from a biological point of view.

2. Problem Statement

Let $e_p \in \mathbb{R}^m$ be a pattern or vector of features defined by a single observation in m -dimensional space (in the gene expression analysis, each gene can be seen as a pattern). Thus, a feature or attribute is the individual component of a pattern. A cluster is defined as a set of patterns whose features are described as similar based on a similarity metric.

The problem of clustering is formally defined below.

Let $E = \{e_1, e_2, \dots, e_n\}$ be a set of patterns, where $e_p \in E$, $p = 1, \dots, n$, is a pattern in the m -th feature dimension space and n is the number of patterns in E ; therefore, the clustering problem is defined as the partitioning of E into k clusters $C = \{c_1, \dots, c_k\}$, $c_i \subseteq E$, $i = 1, \dots, k$, $k \leq n$, such that the following conditions are satisfied:

- Each pattern must be assigned to a cluster; that is, $\bigcup_{\kappa=1}^k c_{\kappa} = E$.
- Each cluster has at least one pattern assigned; that is, $c_{\kappa} \neq \{\emptyset\}$, $\kappa \in \{1, \dots, k\}$.
- Each pattern is assigned to one and only one cluster, such that $c_i \cap c_j = \{\emptyset\}$, $i, j \in \{1, \dots, k\}$, $i \neq j$. This property is called hard clustering.

The clustering problem in gene expression data, in most cases, involves the analysis of high dimensional data. Therefore, the local feature relevance problem [58] and the curse of dimensionality must be considered during its resolution [59].

3. Materials and Methods

In gene expression analysis, matrix factorization (MF) focuses on reducing the dimensionality of the biological data in order to figure out pathways of genetic regulation in a specific genome [60,61] through linear combinations.

Factorization Problem. Given matrix $V \in \mathbb{R}^{n,m}$, express it as $V = W^T H$ with $W \in \mathbb{R}^{k,n}$ and $H \in \mathbb{R}^{k,m}$.

This problem may be modified in the following form:

Approximated Factorization Problem. Given matrix $V \in \mathbb{R}^{n,m}$, construct $V' = W^T H$ with $W \in \mathbb{R}^{k,n}$ and $H \in \mathbb{R}^{k,m}$, such that $V \approx V'$.

Naturally, the approximation “ \approx ” should be estimated with respect to a given metric in the space $\mathbb{R}^{n,m}$.

If $k(n + m) < n \cdot m$, then a gain in storage can be obtained, but most importantly, the dimension of the columns in V ; namely, n , is reduced to the dimension k of the factor matrices columns. In this sense, a *dimensionality reduction* is obtained.

With such factorization, each entry v_{ij} of V can be expressed as a linear combination $v_{ij} \approx \sum_{k=1}^k w_{ik} h_{kj}$.

3.1. NMF for the Analysis of Gene Expression Data

Let $V \in \mathbb{R}^{n,m}$ be a genetic expression matrix, namely, its rows correspond to *genes* and its columns to *conditions* with respective cardinalities n and m . Each entry v_{ij} indicates the level of expression of the i -th gene under the j -th experimental condition. The i -th row is the *expression profile* of the i -th gene, and the j -th column is the j -th *experiment profile*.

The **Non-negative Matrix Factorization (NMF) Problem** is the above stated **Approximated Factorization Problem** with the requirement that W and H have non-negative entries, i. e., $W \geq 0$ and $H \geq 0$, and, moreover:

$$k \ll \frac{mn}{m+n} < \min(n, m). \quad (1)$$

Condition (1) aims at data reduction [62].

If the matrix Euclidean norm

$$A \mapsto \|A\|_2 = \sqrt{\sum_{ij} |a_{ij}|^2}$$

is considered as the metric to estimate approximations, then the NMF Problem is posed equivalently as:

Euclidean approximation. Given matrix $V \in \mathbb{R}^{n,m}$, minimize $\|V - W^T H\|_2$ with $W \in \mathbb{R}^{k,n}$ and $H \in \mathbb{R}^{k,m}$, such that $W \geq 0$ and $H \geq 0$.

The solution (W, H) can be iteratively updated through the rule of Lee and Seung [63]:

$$\forall ij \in \llbracket 1, k \rrbracket \times \llbracket 1, n \rrbracket : \quad w_{ij} \leftarrow w_{ij} \cdot \frac{\sum_{\mu=1}^m v_{i\mu} h_{j\mu}}{\sum_{\kappa=1}^k w_{\kappa j} \left(\sum_{\mu=1}^m h_{\kappa\mu} h_{i\mu} \right)} \quad (2)$$

$$\forall ij \in \llbracket 1, k \rrbracket \times \llbracket 1, m \rrbracket : \quad h_{ij} \leftarrow h_{ij} \cdot \frac{\sum_{v=1}^n w_{iv} v_{vj}}{\sum_{\kappa=1}^k \left(\sum_{v=1}^n w_{iv} w_{\kappa v} \right) h_{\kappa j}} \quad (3)$$

There is no unique solution for the NMF problem, but some computationally efficient algorithms for minimizing the difference between V and $W^T H$ for different error functions have been proposed [64].

It might as well be considered [65], that the *squared Frobenius norm*:

$$A \mapsto \|A\|_F^2 = \text{Tr}(A \cdot A^T).$$

Frobenius approximation. Given matrix $V \in \mathbb{R}^{n,m}$, minimize $\|V - W^T H\|_F^2$ with $W \in \mathbb{R}^{k,n}$ and $H \in \mathbb{R}^{k,m}$, such that $W \geq 0$ and $H \geq 0$.

The solution (W, H) can be iteratively updated through the rules [66,67]:

$$\forall ij \in \llbracket 1, k \rrbracket \times \llbracket 1, n \rrbracket : \quad w_{ij} \leftarrow w_{ij} \frac{1}{\sum_{\mu=1}^m h_{i\mu}} \cdot \frac{\sum_{\mu=1}^m h_{i\mu} v_{j\mu}}{\sum_{\kappa=1}^k h_{\kappa i} w_{\kappa j}} \quad (4)$$

$$\forall ij \in \llbracket 1, k \rrbracket \times \llbracket 1, m \rrbracket : \quad h_{ij} \leftarrow h_{ij} \frac{1}{\sum_{v=1}^n w_{iv}} \cdot \frac{\sum_{v=1}^n w_{iv} v_{vj}}{\sum_{\kappa=1}^k w_{\kappa i} h_{\kappa j}} \quad (5)$$

The interpretation of NMF within biological context is the clustering of genes or conditions into k clusters whose expression level is similar [68]. The i -th column of the factor matrix W describes the degree of membership of the i -th gene within the k -th group. The j -th row of H indicates the degree of influence that the k -th group of genes has on the m -th condition. The subset of rows of H with high values in the same column of W represents a *centroid*. The set of genes and conditions with relatively high values in the same column of W and the corresponding rows of H is called *biccluster* [69]. Finally, whether some entries in a row of H have coefficients greater than zero, the conditions are influenced by local characters in the data. A detailed exposition of NMF appears in [70].

3.2. Hybrid Clustering Algorithm

CC is an algorithm that emerged as an important contribution of classical clustering and refers to the situation in which T different partitions were obtained $P = \{p^1, p^2, \dots, p^T\}$ for a dataset E , where $p^t, t = 1, \dots, T$ consists of a clusters set $C^t = \{c_1^t, c_2^t, \dots, c_k^t\}$ that complies with the restrictions imposed in Section 2; we wanted to find a consensus clustering p^* , such that:

$$\min_{p^*} J = \frac{1}{T} \sum_{t=1}^T d(p^t, p^*). \quad (6)$$

We defined the distance between two partitions p^1, p^2 as $d(p^1, p^2) = \sum_{i,j=1}^n d_{ij}(p^1, p^2)$, such that:

$$d_{ij}(p^1, p^2) = \begin{cases} 1 & (i, j) \in C_{\kappa, \kappa=1, \dots, k}^1 \text{ and } (i, j) \notin C_{\kappa, \kappa=1, \dots, k}^2 \\ 1 & (i, j) \in C_{\kappa, \kappa=1, \dots, k}^2 \text{ and } (i, j) \notin C_{\kappa, \kappa=1, \dots, k}^1 \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

where $(i, j) \in C_{\kappa, \kappa=1, \dots, k}^1$ means that i and j belong to the same cluster in partition p^1 and $(i, j) \notin C_{\kappa, \kappa=1, \dots, k}^1$ represents that i and j belong to different clusters in partition p^1 .

A consensus matrix M can be constructed, such that each input is defined as:

$$m_{ij}^t = \begin{cases} 1 & (i, j) \in C_{\kappa, \kappa=1, \dots, k}^t \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

Then, we could easily say that $d_{ij}(p^1, p^2) = [m_{ij}^1 - m_{ij}^2]^2$, since $|m_{ij}^1 - m_{ij}^2| = 0$ or 1 . Consequently, the Equation (9) can be rewritten as:

$$\min_{p^*} J = \frac{1}{T} \sum_{t=1}^T \sum_{i,j=1}^n [m_{ij}^t - m_{ij}^*]^2. \quad (9)$$

Let $u_{ij} = m_{ij}^*$ denote the solution to the optimization problem of CC (see Equation (9)). U is the consensus matrix. Let the consensus (average) association between i and j be $m'_{ij} = \frac{1}{T} \sum_{t=1}^T m_{ij}^t$. Define the average squared difference from the consensus association $M' : \Delta M^2 = \frac{1}{T} \sum_{t=1}^T \sum_{i,j=1}^n [m_{ij}^t - m'_{ij}]^2$. Clearly, the smaller ΔM^2 , the closer to each other the partitions are. This quantity was a constant. We had:

$$J = \frac{1}{T} \sum_{t=1}^T \sum_{i,j=1}^n (m_{ij}^t - m'_{ij} + m'_{ij} - u_{ij})^2 \quad (10)$$

$$= \Delta M^2 + \sum_{i,j=1}^n (m'_{ij} - u_{ij})^2. \quad (11)$$

Therefore, consensus clustering took the form of the following optimization problem:

$$\min_U \sum_{i,j=1}^n (m'_{ij} - u_{ij})^2 = \|M' - U\|_F^2. \quad (12)$$

where the matrix norm is the Frobenius norm. Therefore, consensus clustering was equivalent to clustering the consensus association. Then, it was easy to show that:

$$U = W^T H. \quad (13)$$

To solve Equation (13), NMF was used (see Section 3.1). The result would be the high dimensional biological dataset to a low dimensional dataset (W) and the candidate set of centroids (H). CC was applied directly to the H matrix in order to refine the centroids in a given t partition (p^t).

In order to solve the clustering problem in the low dimensional dataset, we implemented KM, taking into account the set of elements in E mapped to a k -dimensional space (rows in W) and the set of k centroids described by the columns in H . KM groups elements through a set of numerical properties in such a way that the elements within a cluster are more similar than the elements in different clusters. This is known as the default Sum of Squared Error (SSE) or total cohesion if a cosine-based metric of similarity is used. Therefore, it was necessary to provide a criterion to measure the similarity of the elements. Generally, KM uses Euclidean distance [71] as a similarity metric. However, it is not effective for analyzing high-dimensional datasets because, as the dimensionality of the features increases, the elements tend to move evenly away from each other; that is, the relationship between the closest and furthest elements tends to be 1 in all cases.

In [72], it is shown that the Manhattan distance has better performance for high-dimensional datasets. The use of fractional norms has been proposed as similarity metrics in high-dimensional data, which exhibit the property of increasing the contrast between the most distant and closest elements. This can be useful in some contexts, but it should be considered that they are not distance metrics because they violate the triangle inequality. In the end, the type of metric to use should depend on the features of the data and the type of analysis that is being carried out [73–75].

Finally, *Silhouette coefficient* [76] was used in order to quantitatively evaluate the clustering result. A maximum value indicated the stability of the results through the evaluation of intragroup and extragroup variance. The optimal value of k corresponded to the maximum silhouette value that reflected a better partition. It is worth mentioning that the estimation of the parameter k was considered an NP-complete problem [77].

Given a value of k and based on CC, we proceeded with Algorithm 1. The core of this procedure was the execution of NMF until a convergence criterion was fulfilled. We obtained two matrices W and H . The κ -th entry of the i -th row in the W array represented the degree of membership of the i -th gen to the κ -th cluster. On the other hand, the κ -th column of H represented the choice of the centroid. A set of k clusters was obtained to be analyzed (line 5 of Algorithm 1) to generate a consensus matrix $U \in \mathbb{R}^{m,m}$. The probability $u_{ij} \in [0, 1]$ corresponded to the event in which two conditions fell in the same cluster in two partitions i, j such that $i \neq j$ (Algorithm 2). Then, to obtain the class label of each condition ($Q \in \mathbb{R}^m$), KM was run on the matrix W , which contained the biological data mapped to a low dimensional space ($m \approx 12$ features in the worst case) and taking account the matrix C that represented the centroids obtained by NMF. Finally, the Silhouette coefficient was calculated for each cluster to determine the resulting quality. The result was given as a vector $S \in \mathbb{R}^k$. The implementation of the algorithms KM and SILHOUETTE followed [76].

Algorithm 1 Proposal of hybrid clustering algorithm based on NMF, KM, and CC.

Require: $V \in (\mathbb{R}^+)^{n,m}$ ($n, m \geq 2$): gene expression matrix; k : number of clusters; T : maximum number of partitions for NMF execution as part of CC; ϵ : convergence threshold.

Ensure: $Q \in \mathbb{R}^m$: class label vector of each gene; $S \in \mathbb{R}^k$: Silhouette coefficients of clusters.

```

1: procedure HYBCLUST( $V, k, \epsilon, T$ )
2:    $U = 0$  ▷  $U \in \mathbb{R}^{m,m}$  Initialization of consensus matrix.
3:   for  $t = 1 : T$  do
4:      $\{W, H\} = \text{NMF}(V, k, \epsilon)$  ▷ See Algorithm 4.
5:      $U+ = \text{CONSENSUSMATRIX}(H, k, T)$  ▷ See Algorithm 2.
6:   end for
7:    $Q = \text{KM}(W, U)$  ▷  $Q \in \mathbb{R}^m$  Class label vector.
8:    $S = \text{SILHOUETTE}(Q, V)$  ▷  $S \in \mathbb{R}^k$  Silhouette coefficients vector.
9:   return  $\{Q, S\}$  ▷ Return the class label and silhouette coefficients vectors.
10: end procedure

```

Algorithm 2 Consensus matrix update for the i -th iteration.

Require: $H \in (\mathbb{R}^+)^{k,m}$: right factor in NMF; k : number of clusters; T : maximum number of partitions for NMF executions as part of CC.

Ensure: $C \in \mathbb{R}^{m,m}$: updated consensus matrix.

```

1: procedure CONSENSUSMATRIX( $H, k, T$ )
2:    $temp = 0$ 
3:    $C = 0$  ▷  $C \in \mathbb{R}^{m,m}$  Initialization of consensus matrix.
4:   for  $l = 1 : m$  do
5:     for  $j = l + 1 : m$  do
6:        $c_{lj} = \frac{1}{T}$  ▷ Compute the probability.
7:       for  $\kappa = 1 : k$  do
8:         if  $h_{\kappa l} < 0$  or  $h_{\kappa j} < 0$  then ▷ Verifies whether all entries are greater or equal to 0.
9:            $c_{lj} = 0$  ▷ No belongs to the same group.
10:          break
11:        end if
12:      end for
13:    end for
14:  end for
15:  return  $C$  ▷ Return the consensus matrix.
16: end procedure

```

Algorithm 1 was executed for different values of k as described in Algorithm 3 in order to obtain their optimal value. For each of the many executions, NMF carried out a large number of costly matrix operations. Therefore, line 4 in Algorithm 1 was the critical step.

Algorithm 3 Gene expression analysis.

Require: $V \in (\mathbb{R}^+)^{n,m}$ ($n, m \geq 2$): gene expression matrix; K_{min} : lower bound to test k ; K_{max} : upper limit to test k ; T : maximum number of iterations for NMF executions as part of CC; ϵ : convergence threshold.

Ensure: Return 0 if an exception occurs, return 1 otherwise.

```

1: procedure GENEEXPRESSIONANALYSIS( $V, K_{min}, K_{max}, \epsilon, T$ )
2:   if  $K_{min} > \min(m, n)$  or  $K_{max} > \min(m, n)$  then
3:     return 0 ▷ Fails if  $K$  is too large (Equation (1)).
4:   end if
5:    $Q_{best}, S_{best} = \text{HYBCLUST}(V, K_{min}, \epsilon, T)$ 
6:    $k_{best} = K_{min}$ 
7:   for  $k = K_{min} + 1 : K_{max}$  do
8:      $\{Q, S\} = \text{HYBCLUST}(V, k, \epsilon, T)$  ▷ See Algorithm 1.
9:     if  $S$  is better than  $S_{best}$  then ▷ Update values according to the best value of the
10:       $Q_{best} = Q$ 
11:       $S_{best} = S$ 
12:       $k_{best} = k$ 
13:     end if
14:   end for
15:   return 1
16: end procedure

```

Algorithm 4 is, properly, an NMF algorithm. Given three parameters, $V \in (\mathbb{R}^+)^{n,m}$ ($n, m \geq 2$), representing a gene expression dataset, k such that $2 \leq k \leq \min(n, m)$ represented the current number of clusters, and the convergence threshold ϵ produced two matrices, $W \in (\mathbb{R}^+)^{k,n}$ and $H \in (\mathbb{R}^+)^{k,m}$, containing non-negative values such that $\|V - W^T H\|_F^2 \leq \epsilon$. The random initialization of W and H (line 2 of Algorithm 4) indicates the start of the algorithm; next, the algorithm calculates the inner products $\sum_{k=1}^k w_{ik} h_{kj}$ in line 3 of Algorithm 4. In NMF, the factor dimension (k) determines the quality of results as long as it determines the centroids of each cluster. The optimum value depends on the biological data type. If k tends to be too small, groups will be pretty generic to provide information, whilst, if k tends to be too large, although bounded by (1), the result will be pretty detailed and hard to interpret. CC is an algorithm to estimate an optimum value for k by consensus of the results obtained from the multiple executions of a clustering algorithm [56].

Finally, it enters an iterative process (lines 6–23 of Algorithm 4). W and H were updated based on (4) and (5). The number of iterations performed by Algorithm 4 would vary depending on the input data, on the initial values, as well as on the used convergence method. The iterative process stopped when the convergence criterion was met.

Algorithm 4 Non-negative matrix factorization (NMF)

Require: $V \in (\mathbb{R}^+)^{n,m}$ ($n, m \geq 2$) such that all rows and columns of V contain at least a value greater than zero, $2 \leq K \leq \min(n, m)$ and ϵ : a convergence threshold.

Ensure: $W \in (\mathbb{R}^+)^{k,n}$ and $H \in (\mathbb{R}^+)^{k,m}$ such that $\|V - W^T H\|_F^2 \leq \epsilon$.

```

1: procedure NMF( $V, K, \epsilon$ )
2:   INITIALIZE( $W, H$ )                                ▷ Random values non-negatives.
3:    $X \leftarrow W^T H$ 
4:    $prevIsDiv = 0$ 
5:    $isDiv = \text{ISCONVERGENCE}(V, X)$                     ▷ See Algorithm 5.
6:   while  $\|prevIsDiv - isDiv\|_F^2 \geq \epsilon$  do
7:      $prevIsDiv = isDiv$ 
8:     for  $i = 1 : n$  do                                ▷ Update  $W$ .
9:       for  $k = 1 : K$  do
10:         $\eta_{ik} = \frac{w_{ik}}{\sum_{\kappa=1}^m (h_{k\kappa})}$ 
11:         $w_{ik} = \eta_{ik} \sum_{\kappa=1}^m h_{k\kappa} \frac{v_{i\kappa}}{x_{i\kappa}}$ 
12:      end for
13:    end for
14:     $X \leftarrow W^T H$                                 ▷ Update  $X$ .
15:    for  $k = 1 : K$  do                                ▷ Update  $H$ .
16:      for  $j = 1 : m$  do
17:         $\eta_{kj} = \frac{h_{kj}}{\sum_{\kappa=1}^n (w_{\kappa k})}$ 
18:         $h_{kj} = \eta_{kj} \sum_{\kappa=1}^n w_{\kappa k} \frac{v_{\kappa j}}{x_{\kappa j}}$ 
19:      end for
20:    end for
21:     $X \leftarrow W^T H$                                 ▷ Update  $X$ .
22:     $isDiv = \text{ISCONVERGENCE}(V, X)$ 
23:  end while
24:  return  $\{W, H\}$ 
25: end procedure

```

Algorithm 5 Compute NMF convergence

Require: $V \in (\mathbb{R}^+)^{n,m}$ ($n, m \geq 2$). All rows and columns of V contain at least a value greater than zero and $X \in (\mathbb{R}^+)^{n,m}$ ($n, m \geq 2$).

Ensure: $X \in (\mathbb{R}^+)^{n,m}$ ($n, m \geq 2$).

```

1: procedure ISCONVERGENCE( $V, X$ )
2:    $isDiv = 0$ 
3:   for  $i = 1 : n$  do
4:     for  $j = 1 : m$  do
5:        $isDiv += \left(\frac{v_{ij}}{x_{ij}}\right) - \log\left(\frac{v_{ij}}{x_{ij}}\right) - 1$ 
6:     end for
7:   end for
8:   return  $isDiv$ 
9: end procedure

```

4. Sequential Analysis of NMF Algorithm

The performance analysis of the sequential algorithm was helpful to confirm information about the segment of code that could potentially be parallelized. We implemented the

sequential NMF in C language according to Algorithm 4. The experiments were run on an Intel Xeon E5-1603 v3 processor with four cores and 32 Gb of RAM. An ext4 filesystem and a SATA HDD were used. These features could change the speed-up of the read and write operations. The operating system was Ubuntu 18.04 and the compiler was GCC 9.2. Two dataset groups in silico (generated through a computational simulation.) were generated. *Group A* was used to analyze the computational time when the matrix size increased, whilst *Group B* was used to determine the most expensive operation in terms of computational time. The experiments allowed to detect the code section potentially parallelizable.

Group A. It consists of three matrices $V_1 \in \mathbb{R}^{1794,27}$, $V_2 \in \mathbb{R}^{15000,193}$, and $V_3 \in \mathbb{R}^{10180,856}$, whose contents were partitioned into five, seven and nine clusters, respectively. In all cases, the error threshold was $\epsilon = 10^{-12}$ and $K \in [2, 10]$. In order to make a statistically significant analysis, the whole process was run 32 times per each chosen value of K , as well as for each matrix. This dataset was optimal since the clusters contained contiguous elements and were pairwise disjointed. Figure 1 shows the computational time used by NMF. As the size of the matrix increased, the computation time tended to be infeasible; hence, sequential NMF was unfeasible in biological data processing due to a high dimensionality.

Group B was compound by eleven matrices whose contents were partitioned into three clusters. Each matrix was perturbed by a noise with a Gaussian distribution, and rows and columns were randomly switched, aiming to obtain the worst case instances. The computational time required to read data, process NMF, and write solutions, is shown in Figure 2. In the first three cases, the percentage of reading and writing was more significant than the processing time. As the size of the matrix increased, the portion of the processing time began to converge at approximately 77%. This percentage represented the limit of the algorithm's processing time when parallelized.

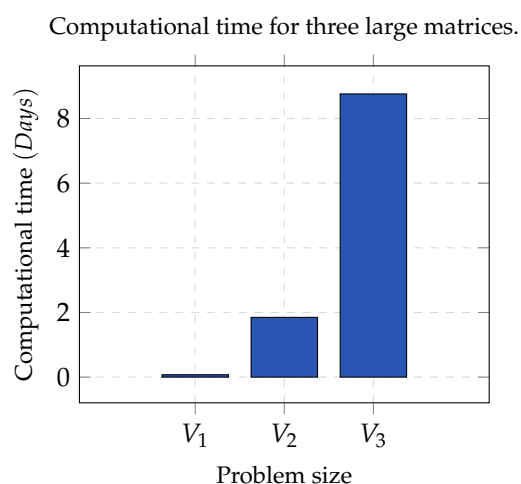


Figure 1. Analysis of the computational time while matrix size increased. Three constructed in silico datasets were proposed, represented by large matrices ($V_1 \in \mathbb{R}^{1794,27}$, $V_2 \in \mathbb{R}^{15000,193}$, and $V_3 \in \mathbb{R}^{10180,856}$). For the three experiments, it was fixed $\epsilon = 10^{-12}$ and $k \in [2, 10]$. We executed the algorithm 32 times in order to obtain a statistically significant analysis.

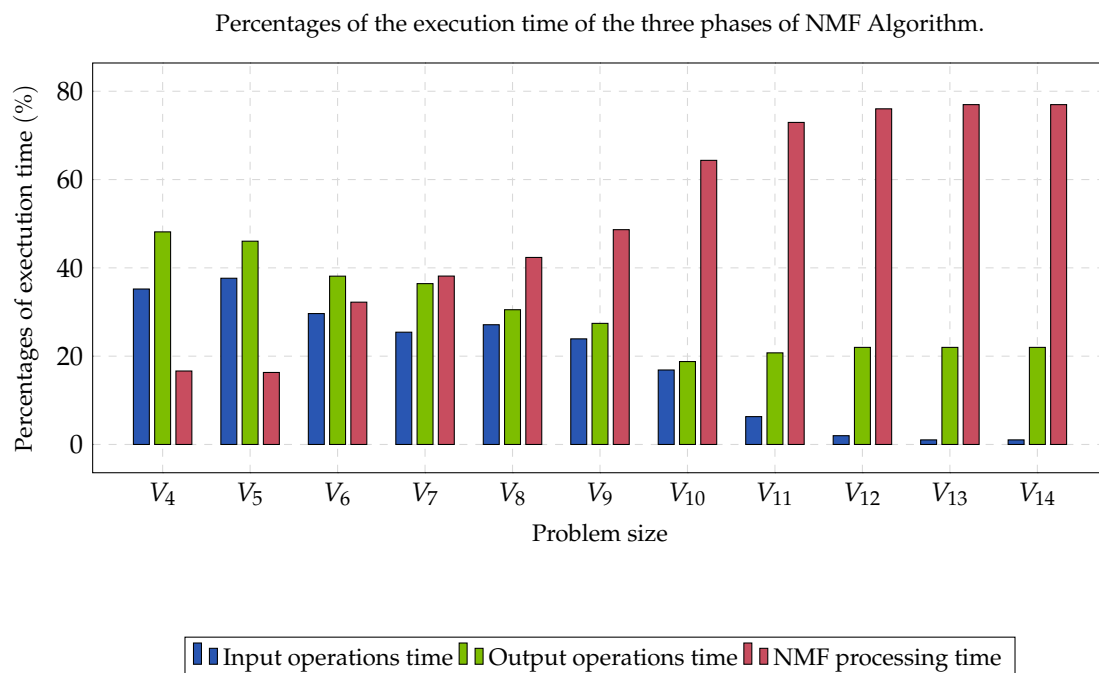


Figure 2. Percentages of the execution time of the three phases of the algorithm (reading, processing, and writing). Eleven matrices were tested ($V_4 \in \mathbb{R}^{10,200}$, $V_5 \in \mathbb{R}^{50,200}$, $V_6 \in \mathbb{R}^{100,200}$, $V_7 \in \mathbb{R}^{150,200}$, $V_8 \in \mathbb{R}^{200,200}$, $V_9 \in \mathbb{R}^{250,200}$, $V_{10} \in \mathbb{R}^{300,250}$, $V_{11} \in \mathbb{R}^{350,200}$, $V_{12} \in \mathbb{R}^{400,200}$, $V_{13} \in \mathbb{R}^{450,200}$, $V_{14} \in \mathbb{R}^{500,200}$), $\epsilon = 10^{-12}$, $k \in [2, 10]$, and $T = 1000$: the maximum number of executions to obtain a statistically significant analysis.

5. Strategies for Parallelizing NMF Algorithm

We discussed in detail the parallelization of NMF (see Algorithm 6) in a GPU through the CUDA library in line with NMF+HC [57,78,79]. However, as we mentioned in Section 1, a small disturbance in the data or a poor decision determined in the initial steps of the algorithm can profoundly modify the final results. The solution strategy is to implement parallelism at the data level [80], namely, to divide the data set into b blocks of size r , such that each processing unit on the GPU solves the problem by applying the same sequence of operations to a specific subset of data. Ideally, this simultaneous execution of operations resulted in a global acceleration of the computation.

The proposed algorithm started with INITIALIZEGPU, the initialization of CUDA and CUBLAS [81]. The amount of global memory was returned or the execution was finished if a failure occurred (line two of Algorithm 6). Next, SETUPGPU checked the matrix dimensions and estimated the parameter r and the convenient size of W and H . Memory allocation for W and H was performed, and the matrices were initialized with non-negative random values (line five of Algorithm 6). Then, V , H , and W were sent from the host to the device. Finally, iteration was performed by updating W and H (lines 9–13 of the Algorithm 6). Given the nature of these functions, they were executed directly in the device. When the algorithm met the convergence criterion, the whole procedure stopped.

Algorithm 7 describes the algorithm to update H according to relation (5). Products of submatrices were involved (line three of the Algorithm 7) and the function CUBLAS_R_GEMM predefined in the CUBLAS library was employed. For line four of Algorithm 7, we implemented a *kernel*: ELEMENTWISEDIV. The *kernel* REDUCE (line seven of the Algorithm 7) computed the reduction in column vectors. The last step of the algorithm was conducted by the *kernel* ELEMENTWISEMULTDIV.

The algorithm operated in the same way to update W . To optimize the memory consumption, we directly modified H and W to avoid allocating too many memory regions.

Algorithm 6 Parallel NMF

Require: $V \in \mathbb{R}_+^{n,m}$ ($n, m \geq 2$); $W \in \mathbb{R}_+^{n,k}$. $H \in \mathbb{R}_+^{k,m}$. k : factorization rank; T is the maximum number of iterations that NMF will be executed as part of CC; ϵ as a convergence threshold.

Ensure: $W \in (\mathbb{R}^+)^{k,n}$ and $H \in (\mathbb{R}^+)^{k,m}$ contains non-negative values such that $V \approx WH$ and $\epsilon \leq \|V - W^T H\|_F^2$.

```

1: procedure NMFGPU( $V, k, interMax, \epsilon$ )
2:    $mem\_size = \text{INITIALIZEGPU}(gpu\_device, K)$ 
3:    $r = \text{SETUPGPU}(mem\_size, K)$ 
4:    $\{W, H\} = \text{ALLOCATEMEMORYHOST}(n, r)$ 
5:    $\text{INITIALIZE}(W, H)$  ▷ Random values non-negative.
6:    $\text{COPYTODEVICE}(V, H, W)$ 
7:    $prevIsDiv = 0$ 
8:    $isDiv = 1$ 
9:   while  $\|prevIsDiv - isDiv\|_F^2 \geq \epsilon$  do ▷ Verify the convergence criterion.
10:     $prevIsDiv = isDiv$ 
11:     $\text{UPDATEW}()$  ▷ See Algorithm 6.
12:     $\text{UPDATEH}()$ 
13:     $isDiv = \text{ISCONVERGENCEGPU}(V, X)$ 
14:  end while
15:  return  $\{W, H\}$ 
16: end procedure

```

Algorithm 7 Parallel NMF

Require: $V_i^* \in \mathbb{R}^{n,r}$ as submatrix of V ; $H_i^* \in \mathbb{R}^{k,r}$ as submatrix of H ; $W \in (\mathbb{R}^+)^{k,n}$ contains non-negative values.

Ensure: $H \in (\mathbb{R}^+)^{k,m}$ contains non-negative update values.

```

1: procedure UPDATEH()
2:   for  $i = 1 : b$  do
3:      $X = \text{CUBLAS\_R\_GEMM}(W^T, H_i^*)$  ▷ Matrix product to obtain  $X \in \mathbb{R}^{n,r}$ .
4:      $X = \text{ELEMENTWISEDIV}(V_i^*, X)$ 
5:      $U = \text{CUBLAS\_R\_GEMM}(W^T, X)$  ▷ Matrix product.
6:      $R = \text{REDUCE}(H_i^*)$  ▷ Reduce columns of  $H_i^*$ .
7:      $H_i^* = \text{ELEMENTWISEMULTDIV}(H_i^*, U, R)$ 
8:   end for
9: end procedure

```

6. Experiment

Two datasets were used to evaluate the performance of the proposed algorithm. *Group C* was used to evaluate the performance of the algorithm when the size of the matrix increased ($V_{15} \in \mathbb{R}^{57,44}$, $V_{16} \in \mathbb{R}^{219,46}$, $V_{17} \in \mathbb{R}^{250,200}$, $V_{18} \in \mathbb{R}^{3500,50}$, $V_{19} \in \mathbb{R}^{5000,38}$), as well as obtaining the maximum acceleration when the algorithm was executed sequentially (CPU) and in parallel (GPU). *Group D* was used to evaluate the efficiency of the algorithm with three datasets described in [82,83]. Table 1 summarizes the information for each data set in *Group D*. The dataset was sufficiently representative since both the number of classes as well as the number of entries varied significantly [84].

Table 1. Test datasets.

Dataset Name	Tissue	No. Genes	No. Samples	No. Entries	No. Classes
Armstrong-2002-v2	Brain	72	2194	157,968	3
Yeoh-2002-v2	Bone	248	2526	62,649	6
Su-2001	Multi	174	1571	273,354	10

The experiment was planned in two stages. The first one focused on finding the maximum speed-up that we could obtain; the second one was used to measure the precision of the clustering analysis. In order to obtain the maximum speed-up ($S(p)$) that we could achieve when executing the proposed algorithm on a GPU, a comparison was determined between the sequential time ($T(1)$) against parallel time ($T(p)$), such that $S(p) = \frac{T(1)}{T(p)}$. On the other hand, we evaluated the clusters set (C) obtained through the Rand Index (RI) and Accuracy (Acc). For both experiments, $T = 1000$ and $\epsilon = 10^{-12}$ was set. In addition, we established the range of k between $[2, 10]$. Specifically, for KM, we used the Manhattan distance metric. In order to obtain statistically significant results, we ran the algorithm 32 times for each dataset. In this sense, the reported result represented the arithmetic mean of the 32 results obtained. Finally, we used a processor Xeon E5-1603 v3 with four cores, with Ubuntu v.18.04 as the operating system and CUDA v.10.2. Consider that an ext4 filesystem and a SATA HDD were used. Moreover, we used a NVIDIA Quadro RTX 4000 GPU with 2304 cores and 8 GB of DDR6 RAM (Turing microarchitecture).

7. Results

The algorithm's speed-up was measured by its computational time. We compared the computational time obtained in a sequential version, parallel proposal (NMF+HC) described in [57], and our parallel proposal NMF+KM. Figure 3 describes the time required for each implementation. In both algorithms, NMF+KM and NMF+HC time stability was observed. In the first two cases, the NMF+HC performance was better; in the last cases, the NMF+KM execution time was better. Regarding the speed-up $S(p)$, NMF+HC obtained a speed-up of 5.96x in the best cases, while NMF+KM obtained a maximum speed-up of 6.22x.

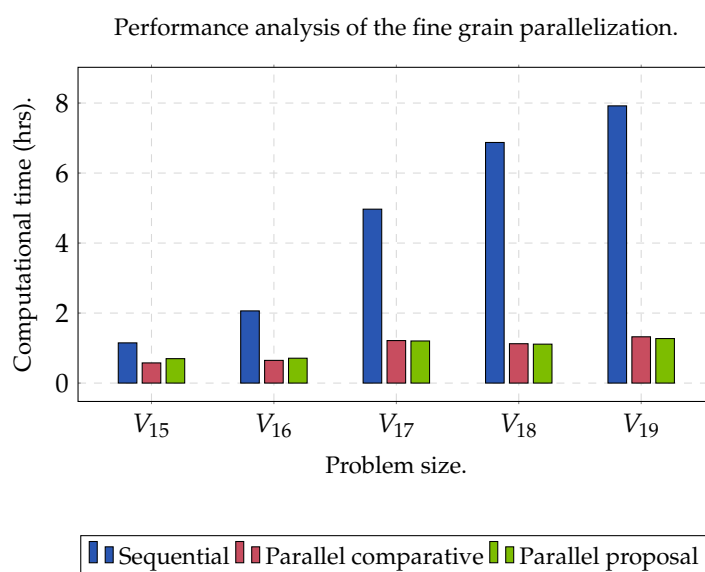


Figure 3. Performance for the number of executions by the NMF Algorithm with fine-grained parallelization. For this implementation, a set of matrices built in silico was used ($V_{15} \in \mathbb{R}^{57,44}$, $V_{16} \in \mathbb{R}^{219,46}$, $V_{17} \in \mathbb{R}^{250,200}$, $V_{18} \in \mathbb{R}^{3500,50}$, $V_{19} \in \mathbb{R}^{5000,38}$). For the experiments, we defined $T = 1000$ and $\epsilon = 10^{-12}$ was set. In addition, we established the range of K between $[2, 10]$. In all cases, the matrix was squared. We used a processor Xeon E5-1603 v3 with four cores, an operating system based on Ubuntu v.18.04, and CUDA v.10.2. Consider that an ext4 filesystem and a SATA HDD were used. Moreover, we used an NVIDIA Quadro RTX 4000 GPU with 2304 cores and 8 GB of DDR6 RAM (Turing microarchitecture).

In order to evaluate the efficiency of NMF+KM, we used the *Group D* dataset. We used Accuracy [85] and the Rand Index (RI) [86] as metrics to measure the quality of the clusters evaluated, respectively, in $(0, 1)$ and $(-1, 1)$. In both cases, the values furthest to

the right indicated a perfect match. Given the non-deterministic nature of NMF and KM. Table 2 describes the average of the Accuracy and RI obtained by NMF+KM and NMF+HC, respectively, and the standard deviation.

Table 2. Accuracy (Acc) and Adjusted Rand Index (RI) with NMF-KM and NMF-HC.

Dataset Name	NMF-KM		NMF-HC	
	Acc	RI	Acc	RI
Armstrong-2002-v2	0.8578 \pm 0.0183	0.7747 \pm 0.0296	0.8102 \pm 0.0177	0.7442 \pm 0.0180
Yeoh-2002-v2	0.7459 \pm 0.0354	0.5746 \pm 0.0445	0.6926 \pm 0.0284	0.5382 \pm 0.0305
Su-2001	0.7109 \pm 0.0223	0.6626 \pm 0.0306	0.6477 \pm 0.0446	0.6214 \pm 0.0209

8. Discussion

A wide range of clustering algorithms was proposed which showed to have a good performance within the biological area. However, these implementations had a high computational complexity and were computationally demanding on the CPU, so their applicability is limited in many circumstances. A parallel clustering algorithm was proposed based on matrix factorization. The results obtained showed that it was possible to obtain a speed-up of at least $6.22\times$ during the execution of the proposed algorithm in a GPU; thus, achieving to reduce the execution time. This analysis supported the theory that it is possible to increase the efficiency of current algorithms, which are already effective, through high-performance computing techniques. Likewise, it can be affirmed that any improvement conducted in NMF would have a very important impact on the overall performance of any proposal that uses this type of matrix factorization.

The reported speed-up was obtained through the comparison of the performance rate between the sequential version and the parallel version. In Figure 2, it was observed that the processing time of NMF increased considerably as the size of the gene expression matrix increased. The above caused the processing time to begin to be a preponderant factor in the execution of this type of algorithms ($\approx 77\%$). The NMF selection, as the most intensive computational task, was suitable due to us being able to explore the fine-grain parallelism. This type of parallelism corresponds to matrix products and other algebraic operations. Finally, the parallelization in a GPU was justified with the existence of highly parallel codes and the almost absence of synchronization operations that may derive from blockages and barriers, which are very expensive in the GPU (thousands of threads competing for a padlock or waiting for a barrier)

On the other hand, as we could see in Figure 2, the computational time for the input and output operations required at least 33% of the total computational time for this experiment. We can conclude that the type of HDD and the type of filesystem that was being used influenced the read and write operations. In this sense, if a file system based on Lustre [87] was used as a parallel file system and, in turn, SAS HDD was used, instead of SATA HDD, this computation time would have reduced considerably due to us having improved the performance of input and output operations. The most important part, then, is to focus on the data processing performance.

Regarding clustering algorithms based on factoring methods, PCA, SVD, ICA, and NMF were exposed for a gene expression analysis. For the purpose of this study, NMF was selected because it is capable of decomposing the matrix into two sub-matrices, which represent a compressed version of the set of original data without losing the original features. In this way, NMF allows reducing the dimensionality of the data while preserving the biological context, and KM presents good performance when applied to low dimensional datasets. As shown in Table 2, the clustering NMF+KM performed quite well on the three datasets described in Table 1. The dimensionality reduction improved the clustering performance NMF+HC in all cases. However, it is appreciated that, as the number of classes increased, the results began to be unsatisfactory. This implied that irrelevant or misleading genes may not be taken into account during the dimensionality reduction phase.

Another important aspect to mention is the similarity metrics used by KM. The selection of the best metric will depend on the features of the data. Furthermore, based on the proposal described in [56], we managed to establish a methodology that, in addition to providing an estimate of the parameter k , was capable of inducing the initialization of the centroids through NMF factorization. Finally, it should be considered that there are a variety of ways of measuring multivariable differences or distances between elements, which provide various possibilities for analysis. The use of them, as well as the features of the clustering algorithms, or different mathematical rules to assign the elements to different clusters, depends on the case study and the prior knowledge of biological data. In addition, and because the use of the cluster analysis already implies a lack of knowledge or incomplete knowledge of the data grouping, the researcher must be aware of the need to use several methods, none of them unquestionable, to contrast the results.

8.1. Implications

Results confirmed that we could combine simple algorithms in order to propose a solution to the gene expression analysis. If we wanted to obtain more reliable conclusions about this, we would have to investigate more. We could think that clustering algorithms are limited to solve the gene expression analysis, without taking into account limitations imposed by the computational architectures and computational performance. Then, we could conclude that scientist's expectations are obtaining results. Scientists have proposed clustering algorithms, but they did so to improve computational performance. Nowadays, computational time and memory consumption optimization could become the decisive factor in choosing the clustering algorithm. The aforementioned showed that when proposing a clustering algorithm, more attention should be given to improving the computational yield of the algorithm, as well as allowing the user with little or null knowledge in computing, to perform a genetic expression analysis in less time, and focus their attention on more practical aspects.

Although previous research has focused on clustering analysis through sequential algorithms [45,49,82,83], the results described in Table 2 showed that the cluster's quality was competitive compared to the results described in the state of art [83,88,89]. Therefore, we can confirm the main idea in [53], which mentioned that it is possible to improve computational performance to reduce the execution time of the clustering algorithms. It was evident that much has been conducted to propose clustering algorithms capable of obtaining results according to the biological context, but little to improve their computational performance. In our opinion, the current vision of Computer Science should consider the improvement of the computational performance of the algorithms described in the state of the art through high-performance computing. Furthermore, the research carried out may allow us to increase the objectivity for the extraction of genes, which can be used for analyzing the gene set involved in a biological process. We will use the results obtained to make a Genetic Ontology analysis(GO). This is the perspective of our research.

8.2. Limitations

The experience of the scientist in the gene expression analysis limits the interpretation and quality of results. To obtain better results, scientists ought to have experience and knowledge about the dataset features as well as of the biological context in order to choose the clustering algorithm which suits perfectly to solve the problem. Nowadays, a significant challenge is having a clustering algorithm general enough in order to analyze any biological datasets, invariant to the initial parameters, and with an optimal computational performance representing a significant challenge. We created the dataset in silico under controlled conditions, so we came to know better. Similarly, the limitations imposed by the high dimensionality of biological data made it difficult to obtain more specialized knowledge. In this work, we mitigated this situation through the parallelization of the algorithm. The development of methods that would allow the scientist to use them in an almost transparent way is still required, without posing an additional concern.

The acceleration obtained from the parallel algorithm was limited by the features of the available computational architecture. The computational performance of the proposed algorithm will depend directly on the number of available processing units, the type of operating system, the type of filesystem available, the type of hard disk, and the features of the GPU available. The experiment in this work was carried out on a Quadro-type GPU; however, the computational performance can improve if a Tesla, Pascal (P100), or volta (V100)-type GPU is used. In this sense, it must be considered that the algorithm was parallelized considering a specific computational architecture. An improvement in the architecture, for example, a distributed architecture, will mean that the algorithm has to be parallelized to continue making efficient use of the computational resources.

The parallelization of the proposed algorithm basically focused on NMF since it was identified as the most computationally intensive task speaking. In this work, we focused on fine-grained parallelization; however, adding medium-grain or coarse-grained parallelization should improve the overall performance of the algorithm.

9. Conclusions

NMF has been established as a valuable tool to generate low-range approximations of large biological datasets [46,90,91]. NMF has recently been applied in the resolution of grouping problems. In this context, we presented a clustering algorithm based on NMF and KM. The objective of NMF is to reduce the dimensionality, while KM focus is on the clustering problem resolution. Given the features of both matricial and algebraic operations that make up NMF, it was selected as the most intensively computational task and, therefore, we proposed its parallelization in a GPU. The parallelization showed a speed-up of $6.22\times$ with respect to the sequential implementation. Thus, the main contribution of this work was the implementation of an efficient NMF. It was expected that this implementation would help to obtain faster results, making use of current technologies, as well as that the parallel implementation can be used in solving complex problems. There is no, nor does there exist, a clustering algorithm generic enough to solve the clustering problem in a general way. It should be considered that the validity of the results depends on how well the algorithm is adapted to the biological experiment under study. For example, if the goal is to group genes with related functions, then existing functional annotations can be used to validate how well the objective was achieved. As future work, the GO [92] will be exposed to analyze the performance of the proposed algorithm by providing a coherent and structured nomenclature for biological concepts. In addition, the implementation of CC and NMF in a distributed environment is proposed.

Author Contributions: Conceptualization A.A.S.-R., G.B.M.-L., and A.M.-V.; methodology, A.A.S.-R., G.B.M.-L., and A.M.-V.; formal analysis, A.A.S.-R.; supervision, G.B.M.-L. and A.M.-V.; validation, G.B.M.-L. and A.M.-V.; writing—original draft preparation, A.A.S.-R.; writing—review and editing, G.B.M.-L. and A.M.-V.; funding acquisition, G.B.M.-L. and A.M.-V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank the financial support given by the Mexican National Council of Science and Technology (CONACyT), as well as the Center for Research and Advanced Studies of National Polytechnic Institute (CINVESTAV-IPN), for the encouragement and facilities provided to accomplish this publication. The authors wish to acknowledge the contribution of reviewers to improve the quality of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DM	Data Mining
KM	K-means algorithm
HC	Hierarchical clustering algorithm
PCA	Principal components analysis
SVD	Singular value decomposition
ICA	Independent component analysis
KCC	K-means based co-clustering
DIANA	Divisive analysis algorithm
SOM	Self-organized maps algorithm
FCM	Fuzzy c-means algorithm
SOTA	Self-organized tree algorithm
AGNES	Agglomerative nesting algorithm
CPU	Central processing unit
HPC	High-performance computing
NMF	Non-negative matrices factorization
CC	Consensus clustering algorithm
GPU	Graphics processing unit
MF	Matrix factorization
SSE	Sum of Squared Error
RI	Rand Index
HDD	Hard drive disk
SATA	Serial advanced technology attachment
SAS	Serial attached SCSI
GO	Gene ontology analysis

References

- Smieszek, S.P.; Przychodzen, B.P.; Polymeropoulos, M.H. Amantadine disrupts lysosomal gene expression: A hypothesis for COVID19 treatment. *Int. J. Antimicrob. Agents* **2020**, *55*, 106004.
- Manne, B.K.; Denorme, F.; Middleton, E.A.; Portier, I.; Rowley, J.W.; Stubben, C.; Campbell, R.A. Platelet gene expression and function in patients with COVID-19. *Blood* **2020**, *136*, 1317–1329.
- Ouyang, Y.; Yin, J.; Wang, W.; Shi, H.; Shi, Y.; Xu, B.; Qiao, L.; Feng, Y.; Pang, L.; Wei, F.; et al. Downregulated gene expression spectrum and immune responses changed during the disease progression in patients with COVID-19. *Clin. Infect. Dis.* **2020**, *71*, 2052–2060.
- Saxena, A.; Prasad, M.; Gupta, A.; Bharill, N.; Patel, O.P.; Tiwari, A.; Lin, C.T. A review of clustering techniques and developments. *Neurocomputing* **2017**, *267*, 664–681.
- Zou, Q.; Lin, G.; Jiang, X.; Liu, X.; Zeng, X. Sequence clustering in bioinformatics: An empirical study. *Brief. Bioinform.* **2020**, *21*, 1–10.
- Almugren, N.; Alshamlan, H. A survey on hybrid feature selection methods in microarray gene expression data for cancer classification. *IEEE Access* **2019**, *7*, 78533–78548.
- Dana, R.D.; Dikananda, A.R.; Sudrajat, D.; Wanto, A.; Fasya, F. Measurement of health service performance through machine learning using clustering techniques. In *Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2019; Volume 1360, p. 012017.
- Rodriguez, M.Z.; Comin, C.H.; Casanova, D.; Bruno, O.M.; Amancio, D.R.; Costa, L.D.F.; Rodrigues, F.A. Clustering algorithms: A comparative approach. *PLoS ONE* **2019**, *14*, e0210236.
- Rahman, M.A.; Islam, M.Z. A hybrid clustering technique combining a novel genetic algorithm with K-Means. *Knowl.-Based Syst.* **2014**, *71*, 345–365.
- Pirim, H.; Eksioğlu, B.; Perkins, A.D.; Yuceer, C. Clustering of high throughput gene expression data. *Comput. Oper. Res.* **2012**, *39*, 3046–3061.
- Jothi, R.; Mohanty, S.K.; Ojha, A. DK-means: A deterministic k-means clustering algorithm for gene expression analysis. *Pattern Anal. Appl.* **2019**, *22*, 649–667.
- Zhao, M.; Tang, Y.; Kim, H.; Hasegawa, K. Machine learning with k-means dimensional reduction for predicting survival outcomes in patients with breast cancer. *Cancer Inform.* **2018**, *17*, 1176935118810215.
- Hussain, S.F.; Haris, M. A k-means based co-clustering (kCC) algorithm for sparse, high dimensional data. *Expert Syst. Appl.* **2019**, *118*, 20–34.
- Dubey, A.K.; Gupta, U.; Jain, S. Comparative study of K-means and fuzzy C-means algorithms on the breast cancer data. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2018**, *8*, 18–29.

15. Nan, F.; Li, Y.; Jia, X.; Dong, L.; Chen, Y. Application of improved som network in gene data cluster analysis. *Measurement* **2019**, *145*, 370–378.
16. Oyelade, J.; Isewon, I.; Oladipupo, F.; Aromolaran, O.; Uwoghien, E.; Ameh, F.; Adebiyi, E. Clustering algorithms: Their application to gene expression data. *Bioinform. Biol. Insights* **2016**, *10*, BBI-S38316. <https://doi.org/10.4137/BBI.S38316>.
17. Jhalia, V.; Swarnkar, T. A Critical Review on the Application of Artificial Neural Network in Bioinformatics. *Data Anal. Bioinform. Mach. Learn. Perspect.* **2021**, 51–76. <https://doi.org/10.1002/9781119785620.ch3>.
18. Cohen-Addad, V.; Kanade, V.; Mallmann-Trenn, F.; Mathieu, C. Hierarchical clustering: Objective functions and algorithms. *J. ACM (JACM)* **2019**, *66*, 1–42.
19. Gupta, M.K.; Chandra, P. A comprehensive survey of data mining. *Int. J. Inf. Technol.* **2020**, *12*, 1243–1257.
20. Petegrosso, R.; Li, Z.; Kuang, R. Machine learning and statistical methods for clustering single-cell RNA-sequencing data. *Briefings Bioinform.* **2020**, *21*, 1209–1223.
21. Babichev, S.; Skvor, J. Technique of gene expression profiles extraction based on the complex use of clustering and classification methods. *Diagnostics* **2020**, *10*, 584.
22. Babichev, S.; Lytvynenko, V.; Skvor, J.; Fiser, J. Model of the objective clustering inductive technology of gene expression profiles based on SOTA and DBSCAN clustering algorithms. In *Conference on Computer Science and Information Technologies*; Springer: Cham, Switzerland, 2017; pp. 21–39.
23. Fyad, H.; Barigou, F.; Bouamrane, K. An Experimental Study on Microarray Expression Data from Plants under Salt Stress by using Clustering Methods. *Int. J. Interact. Multimed. Artif. Intell.* **2020**, *6*, 38–47.
24. Liu, F.; Zhou, Z.; Cai, M.; Wen, Y.; Zhang, J. AGNEP: An Agglomerative Nesting Clustering Algorithm for Phenotypic Dimension Reduction in Joint Analysis of Multiple Phenotypes. *Front. Genet.* **2021**, *12*, 648831.
25. Bulut, H.; Onan, A.; Korukoglu, S. An improved ant-based algorithm based on heaps merging and fuzzy c-means for clustering cancer gene expression data. *Sādhanā* **2020**, *45*, 1–17.
26. Roux, M. A comparative study of divisive and agglomerative hierarchical clustering algorithms. *J. Classif.* **2018**, *35*, 345–366.
27. Salman, A.D. Density Based Spatial Clustering for Noisy Gene Expression Data. *Turk. J. Comput. Math. Educ. (TURCOMAT)* **2021**, *12*, 5391–5402.
28. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD* **1996**, *96*, 226–231.
29. Aggarwal, C.C. A survey of stream clustering algorithms. In *Data Clustering*; Chapman and Hall/CRC: London, UK, 2018; pp. 231–258.
30. Gobin, E.; Bagwell, K.; Wagner, J.; Mysona, D.; Sandirasegarane, S.; Smith, N.; She, J.X. A pan-cancer perspective of matrix metalloproteases (MMP) gene expression profile and their diagnostic/prognostic potential. *BMC Cancer* **2019**, *19*, 581.
31. Todorov, H.; Fournier, D.; Gerber, S. Principal components analysis: Theory and application to gene expression data analysis. *Genom. Comput. Biol.* **2018**, *4*, e100041.
32. Liu, J.X.; Kong, X.Z.; Zheng, C.H.; Shang, J.L.; Zhang, W. Sparse singular value decomposition-based feature extraction for identifying differentially expressed genes. In *Proceedings of the 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Shenzhen, China, 15–18 December 2016; pp. 1822–1827.
33. Liebermeister, W. Linear modes of gene expression determined by independent component analysis. *Bioinformatics* **2002**, *18*, 51–60.
34. Zhu, X.; Ching, T.; Pan, X.; Weissman, S.M.; Garmire, L. Detecting heterogeneity in single-cell RNA-Seq data by non-negative matrix factorization. *PeerJ* **2017**, *5*, e2888.
35. Lopez, R.; Nazaret, A.; Langevin, M.; Samaran, J.; Regier, J.; Jordan, M.I.; Yosef, N. A joint model of unpaired data from scRNA-seq and spatial transcriptomics for imputing missing gene expression measurements. *arXiv* **2019**, arXiv:1905.02269.
36. Swain, S.; Banerjee, A.; Bandyopadhyay, M.; Satapathy, S.C. Dimensionality Reduction and Classification in Hyperspectral Images Using Deep Learning. In *Machine Learning Approaches for Urban Computing*; Springer: Singapore, 2021.
37. Frigyesi, A.; Höglund, M. Non-negative matrix factorization for the analysis of complex gene expression data: Identification of clinically relevant tumor subtypes. *Cancer Inform.* **2008**, *6*, CIN-S606. <https://doi.org/10.4137/CIN.S606>.
38. Lee, D.D.; Seung, H.S. Learning the parts of objects by non-negative matrix factorization. *Nature* **1999**, *401*, 788–791.
39. Kim, P.M.; Tidor, B. Subsystem identification through dimensionality reduction of large-scale gene expression data. *Genome Res.* **2003**, *13*, 1706–1718.
40. Carmona-Saez, P.; Pascual-Marqui, R.D.; Tirado, F.; Carazo, J.M.; Pascual-Montano, A. Biclustering of gene expression data by non-smooth non-negative matrix factorization. *BMC Bioinform.* **2006**, *7*, 78.
41. Boccarelli, A.; Coluccia, M. Breast Cancer’s Microarray Data: Pattern Discovery Using Nonnegative Matrix Factorizations. In *Machine Learning, Optimization, and Big Data*, Proceedings of the Second International Workshop on Machine Learning, Optimization, and Big Data, MOD 2016, Volterra, Italy, 26–29 August 2016; Springer: Cham, Switzerland, 2016; Volume 10122, p. 281.
42. Song, M.; Peng, Y.; Jiang, T.; Li, J.; Zhang, S. Accelerated image factorization based on improved NMF algorithm. *J. Real-Time Image Process.* **2018**, *15*, 93–105.

43. Battenberg, E.; Wessel, D. Accelerating Non-Negative Matrix Factorization for Audio Source Separation on Multi-Core and Many-Core Architectures. In Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009), Kobe, Japan, 26–30 October 2009; pp. 501–506.
44. Liu, F.; Shan, Z.; Chen, Y. Parallel Nonnegative Matrix Factorization with Manifold Regularization. *J. Electr. Comput. Eng.* **2018**, *2018*, 6270816.
45. Rafique, O.; Mir, A.H. A topological approach for cancer subtyping from gene expression data. *J. Biomed. Inform.* **2020**, *102*, 103357.
46. Hao, Y.J.; Hou, M.X.; Gao, Y.L.; Liu, J.X.; Kong, X.Z. Application of a deep matrix factorization model on integrated gene expression data. *Curr. Bioinform.* **2020**, *15*, 359–367.
47. Jiang, X.; Zhang, H.; Zhang, Z.; Quan, X. Flexible non-negative matrix factorization to unravel disease-related genes. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2018**, *16*, 1948–1957.
48. Yu, N.; Gao, Y.L.; Liu, J.X.; Wang, J.; Shang, J. Robust hypergraph regularized non-negative matrix factorization for sample clustering and feature selection in multi-view gene expression data. *Hum. Genom.* **2019**, *13*, 1–10.
49. Casalino, G.; Coluccia, M.; Pati, M.L.; Pannunzio, A.; Vacca, A.; Scilimati, A.; Perrone, M.G. Intelligent microarray data analysis through non-negative matrix factorization to study human multiple myeloma cell lines. *Appl. Sci.* **2019**, *9*, 5552.
50. Boccarelli, A.; Esposito, F.; Coluccia, M.; Frassanito, M.A.; Vacca, A.; Del Buono, N. Improving knowledge on the activation of bone marrow fibroblasts in MGUS and MM disease through the automatic extraction of genes via a nonnegative matrix factorization approach on gene expression profiles. *J. Transl. Med.* **2018**, *16*, 1–16.
51. Sinha, S.; Hazarika, A.; Hazarika, G.C. A Review on GPU Accelerated Bioinformatics Tool. *J. Sci.* **2020**, *3*, 5–20.
52. Shajii, A.; Numanagic, I.; Baghdadi, R.; Berger, B.; Amarasinghe, S. Seq: A high-performance language for bioinformatics. *Proc. ACM Program. Lang.* **2019**, *3*, 1–29.
53. Ocaña, K.; Galheigo, M.; Osthoff, C.; Gadelha, L.; Gomes, A.T.A.; De Oliveira, D.; Vasconcelos, A.T. Towards a science gateway for bioinformatics: Experiences in the Brazilian system of high performance computing. In Proceedings of the 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Larnaca, Cyprus, 14–17 May 2019; pp. 638–647.
54. Aydin, Z. Performance Analysis of Machine Learning and Bioinformatics Applications on High Performance Computing Systems. *Acad. Platf. J. Eng. Sci.* **2020**, *8*, 1–14.
55. Schmidt, B.; Hildebrandt, A. Next-generation sequencing: Big data meets high performance computing. *Drug Discov. Today* **2017**, *22*, 712–717.
56. Unulu, R.; Xanthopoulos, P. Estimating the number of clusters in a dataset via consensus clustering. *Expert Syst. Appl.* **2019**, *125*, 33–39.
57. Mejía-Roa, E.; Tabas-Madrid, D.; Setoain, J.; García, C.; Tirado, F.; Pascual-Montano, A. NMF-mGPU: Non-negative matrix factorization on multi-GPU systems. *BMC Bioinform.* **2015**, *16*, 1–12.
58. Kriegel, H.P.; Kroger, P.; Zimek, A. Subspace clustering. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2012**, *2*, 351–364.
59. Adachi, S. Rigid geometry solves “curse of dimensionality” effects in clustering methods: An application to omics data. *PLoS ONE* **2017**, *12*, e0179180.
60. Chalise, P.; Fridley, B.L. Integrative clustering of multi-level ‘omic data based on non-negative matrix factorization algorithm. *PLoS ONE* **2017**, *12*, e0176278.
61. He, Y.; Chhetri, S.B.; Arvanitis, M.; Srinivasan, K.; Aguet, F.; Ardlie, K.G.; Battle, A. sn-sPMF: Matrix factorization informs tissue-specific genetic regulation of gene expression. *Genome Biol.* **2020**, *21*, 1–25.
62. Du, K.; Swamy, M. *Neural Networks and Statistical Learning*; Springer: London, UK, 2019; pp. 427–428.
63. Lee, D.D.; Seung, H.S. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, Proceedings of the 13th International Conference on Neural Information Processing Systems, Denver, CO, USA, 1 January 2000; pp. 535–541.
64. Laurberg, H.; Christensen, M.G.; Plumbley, M.D.; Hansen, L.K.; Jensen, S.H. Theorems on positive data: On the uniqueness of NMF. *Comput. Intell. Neurosci.* **2008**, *2008*, 764206.
65. Gratton, S. On the condition number of linear least squares problems in a weighted Frobenius norm. *BIT Numer. Math.* **1996**, *36*, 523–530.
66. Hien, L.T.K.; Gillis, N. Algorithms for nonnegative matrix factorization with the Kullback–Leibler divergence. *arXiv* **2020**, arXiv:2010.01935.
67. Gillis, N. Algorithms for Nonnegative Matrix Factorization with the Kullback–Leibler Divergence. *J. Sci. Comput.* **2021**, *87*, 1–32.
68. Zeng, Z.; Vo, A.H.; Mao, C.; Clare, S.E.; Khan, S.A.; Luo, Y. Cancer classification and pathway discovery using non-negative matrix factorization. *J. Biomed. Inform.* **2019**, *96*, 103247.
69. Mounir, M.; Hamdy, M.; Khalifa, M.E. Bicluster Coherency Measures for Gene Expression Data. *Egypt. Comput. Sci. J.* **2019**, *43*, 15–25.
70. Blum, A.; Hopcroft, J.; Kannan, R. *Foundations of Data Science*; Cambridge University Press: Cambridge, UK, 2017.
71. Bouhmala, N. How good is the euclidean distance metric for the clustering problem. In Proceedings of the 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), Kumamoto, Japan, 10–14 July 2016; pp. 312–315.

72. Aggarwal, C.C.; Hinneburg, A.; Keim, D.A. On the surprising behavior of distance metrics in high dimensional space. In Proceedings of the 8th International Conference on Database Theory, London, UK, 4–6 January 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 420–434.
73. Arora, J.; Khatter, K.; Tushir, M. Fuzzy c-means clustering strategies: A review of distance measures. *Softw. Eng.* **2019**, 153–162.
74. Thant, A.A.; Aye, S.M.; Mandalay, M. Euclidean, Manhattan and Minkowski Distance Methods For Clustering Algorithms. *Int. J. Sci. Res. Sci. Eng. Technol.* **2020**, 7, doi:10.32628/IJSRSET2073118.
75. Zhu, X.; Li, Y.; Wang, J.; Zheng, T.; Fu, J. Automatic Recommendation of a Distance Measure for Clustering Algorithms. *ACM Trans. Knowl. Discov. Data (TKDD)* **2020**, 15, 1–22.
76. Yuan, C.; Yang, H. Research on K-value selection method of K-means clustering algorithm. *J* **2019**, 2, 226–235.
77. Brucker, P. On the complexity of clustering problems. In *Optimization and Operations Research*; Springer: Berlin/Heidelberg, Germany, 1978; pp. 45–54.
78. Lopez-Fernandez, A.; Rodriguez-Baena, D.; Gomez-Vela, F.; Divina, F.; Garcia-Torres, M. A multi-GPU biclustering algorithm for binary datasets. *J. Parallel Distrib. Comput.* **2021**, 147, 209–219.
79. Taylor-Weiner, A.; Aguet, F.; Haradhvala, N.J.; Gosai, S.; Anand, S.; Kim, J.; Ardlie, K.; Van Allen, E.M.; Getz, G. Scaling computational genomics to millions of individuals with GPUs. *Genome Biol.* **2019**, 20, 1–5.
80. Minakova, S.; Tang, E.; Stefanov, T. Combining task- and data-level parallelism for high-throughput CNN inference on embedded CPUs-GPUs MPSoCs. In Proceedings of the 20th International Conference on Embedded Computer Systems, SAMOS 2020, Samos, Greece, 5–9 July 2020; Springer: Cham, Switzerland, 2020; pp. 18–35.
81. Wang, X.; Liu, T.; Trinh-Hoang, M.; Pesavento, M. GPU-accelerated parallel optimization for sparse regularization. In Proceedings of the 2020 IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM), Hangzhou, China, 8–11 June 2020; pp. 1–5.
82. Mirzal, A. SVD based Gene Selection Algorithm. In Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013), Kuala Lumpur, Malaysia, 16–18 December 2013; Springer: Singapore, 2014; pp. 223–230.
83. Bhowmick, S.S.; Saha, I.; Rato, L.; Bhattacharjee, D. Integrated Classifier: A Tool for Microarray Analysis. In Proceedings of the International Conference on Computational Intelligence, Communications, and Business Analytics, Kolkata, India, 24–25 March 2017; Springer: Singapore, 2017; pp. 30–43.
84. De Souto, M.C.; Costa, I.G.; De Araujo, D.S.; Ludermir, T.B.; Schliep, A. Clustering cancer gene expression data: A comparative study. *BMC Bioinform.* **2008**, 9, 1–14.
85. Nazeer, K.A.; Sebastian, M.P. Improving the Accuracy and Efficiency of the k-means Clustering Algorithm. In Proceedings of the World Congress on Engineering, London, UK, 1–3 July 2009; Association of Engineers: London, UK, 2009; Volume 1, pp. 1–3.
86. Krieger, A.M.; Green, P.E. A generalized Rand-index method for consensus clustering of separate partitions of the same data base. *J. Classif.* **1999**, 16, 63–89.
87. Rybintsev, V.O. Optimizing the parameters of the Lustre-file-system-based HPC system for reverse time migration. *J. Supercomput.* **2020**, 76, 536–548.
88. Mirzal, A. Nonparametric orthogonal NMF and its application in cancer clustering. In Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013), Kuala Lumpur, Malaysia, 16–18 December 2013; Springer: Singapore, 2014; pp. 177–184.
89. Yu, Z.; Luo, P.; You, J.; Wong, H.S.; Leung, H.; Wu, S.; Han, G. Incremental semi-supervised clustering ensemble for high dimensional data clustering. *IEEE Trans. Knowl. Data Eng.* **2015**, 28, 701–714.
90. Chen, H.; Gao, M.; Zhang, Y.; Liang, W.; Zou, X. Attention-based multi-NMF deep neural network with multimodality data for breast cancer prognosis model. *BioMed Res. Int.* **2019**, 2019, 9523719.
91. Sharma, G.; Colantuoni, C.; Goff, L.A.; Fertig, E.J.; Stein-O'Brien, G. projectR: An R/Bioconductor package for transfer learning via PCA, NMF, correlation and clustering. *Bioinformatics* **2020**, 36, 3592–3593.
92. Gene Ontology Consortium. The gene ontology resource: 20 years and still GOing strong. *Nucleic Acids Res.* **2019**, 47, D330–D338.