Chapter 2

Big Data Supervised Pairwise Ortholog Detection in Yeasts

Deborah Galpert Cañizares, Sara del Río García,
Francisco Herrera, Evys Ancede Gallardo,
Agostinho Antunes and Guillermin Agüero-Chapin

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.70479

Abstract

Ortholog are genes in different species, evolving from a common ancestor. Ortholog detection is essential to study phylogenies and to predict the function of unknown genes. The scalability of gene (or protein) pairwise comparisons and that of the classification process constitutes a challenge due to the ever-increasing amount of sequenced genomes. Ortholog detection algorithms, just based on sequence similarity, tend to fail in classification, specifically, in *Saccharomycete* yeasts with rampant paralogies and gene losses. In this book chapter, a new classification approach has been proposed based on the combination of pairwise similarity measures in a decision system that consider the extreme imbalance between ortholog and non-ortholog pairs. Some new gene pair similarity measures are defined based on protein physicochemical profiles, gene pair membership to conserved regions in related genomes, and protein lengths. The efficiency and scalability of the calculation of these measures are analyzed to propose its implementation for big data. In conclusion, evaluated supervised algorithms that manage big and imbalanced data showed high effectiveness in *Saccharomycete* yeast genomes.

Keywords: ortholog detection, similarity measures, big data supervised classification, scalability

1. Introduction

Orthologs are genes in different species evolving from a common ancestor while paralogous genes are homologous sequences that evolved in a duplication event derived from a unique sequence [1]. The distinction between orthologs and paralogs is crucial since ortholog genes are considered evolution markers and they help to infer the function of unknown proteins.

The orthology relationship is expressed in terms of pairs with one-to-one relations, or of groups with one-to-many or many-to-many relationships. Thus, the scalability of both gene/protein pairwise comparisons and the classification process constitute a challenge due to the ever-increasing amount of sequenced genomes. The unsupervised ortholog classification problem is presented in [2] starting from the pairwise sequence comparison. Many algorithms built a similarity graph from these comparisons such as the Reciprocal Best Hit (RBH) heuristic [3], Inparanoid [4], OrthoMCL [5], the Comprehensive, Automated Project for the Identification of Orthologs from Complete Genome Data (OMA) [6], the well-known Reciprocal Smallest Distance (RSD) algorithm [7], among others. In this unsupervised learning approach there are also tree-based and hybrid algorithms [8]. On the other hand, [9], a supervised approach form pairwise classification approach is presented where there is no evidence of the imbalance management of the scarce ortholog pairs (minority class) among the majority of non-ortholog pairs. Besides, this approach does not couple with the generalization problem of the built model to external genome pair is not involved in the learning process.

An important issue to tackle with ortholog detection (OD) algorithms is the underlying information of gene/protein features used to classify the sequences. Algorithms just based on sequence similarity tend to fail in classification because OD is negatively affected by genetic and evolutionary events like duplications and gene losses, or changes between genome segments such as duplications, deletions, horizontal gene transfers, fusions, fissions, inversions, and transpositions. Although the high potentiality of the OD methods referenced either in the "omics"[1] tools site or the site of the Quest for Orthologs[2] Consortium, some of their limitations are reported in literature:

- They tend to include paralogs in the orthogroups increasing false positives [8]. In the presence of lots of gene losses caused by whole genome duplications (WGD) in *Saccharomycete* yeasts, a big amount of paralogs can be included in orthogroups, or in orthologous pairs [10].
- They can reduce specificity due to the presence of short sequences or those that is evolved in a convergent way. This kind of affection may arise when genes inherited by horizontal transfers brought about failure in the inference of near phylogenetic relationships between species that are distantly related and have recently exchanged a gene [11].
- They can fail in ortholog detection mainly in the twilight zone (less than the 30% of sequence identity), that is, they can raise false negatives in the presence of distantly related divergent sequences. They can also fail in front of mosaics of proteins [12] since orthogroups not considering hybrid proteins may contain proteins without a common ancestor.

With the aim of reducing these limitations and improving efficacy in ortholog detection, there is a tendency in algorithms to include diverse gene information in addition to the alignment-based features (AB). Some ortholog detection algorithms include information about: (i) synteny [13–15],

[1] http://omictools.com/
[2] http://questfororthologs.org/orthology_databases

(ii) global rearrangements [16, 17], (iii) protein interactions [9], (iv) the architecture of protein mosaics [18], and (v) evolution distances [6, 7]. So far, these attempts have not been enough for increasing efficacy in classification [19]. The combination of AB with other features like the physicochemical profiles, the length of the sequences, and the synteny may be positive for pairwise ortholog detection as well as the learning process from curated classifications in benchmark datasets may allow for a more effective approach.

In order to evaluate OD algorithms, Salichos and Rokas [10] constructed a *Saccharomycete* yeast benchmark dataset having orthogroups deprived of paralogs. Such orthogroups contain "curated or true" orthologs from yeast species that underwent pre- and post-whole genome duplications (WGD) [10]. Actually, when Multiparanoid [20], OrthoMCL, and extended versions of RBH and the RSD were evaluated using this benchmark dataset, they included paralogs in the orthogroups [10]. This fact assures that OD is a bioinformatics field in constant need of new effective algorithms mainly for yeasts.

In terms of efficiency, OD algorithms based on sequence alignments may take from weeks to years of CPU time to compute orthologs of sequenced species, due to the quadratic scaling capacity of these comparisons. That is why in the Quest for Orthologs Consortium meeting [19], efficiency and management of big data were target questions, and some efforts in these senses were presented. They stated that the increased demand of computation for sequence analysis has not been achieved by means of the increasing computation capacity, but it can be achieved with new approaches or algorithm implementations.

In this chapter, a new classification approach has been proposed based on the combination of pairwise protein features in a supervised decision system that consider the extreme imbalance between ortholog and non-ortholog pairs on the benchmark dataset proposed by [10]. Some new gene pair similarity measures considered as pairwise protein features are defined based on protein physicochemical profiles, their membership to conserved regions in related genomes, and their lengths. The efficiency and scalability of the calculation of these measures are analyzed to propose its implementation for big data. In conclusion, evaluated supervised algorithms that manage big and imbalanced data in scalable machine learning libraries as Mahout [21] and MLlib [22] overdid RBH, RSD, and the Comprehensive, Automated Project for the Identification of Orthologs from Complete Genome Data (OMA) [6] in *Saccharomycete* yeast genomes.

2. Methods

2.1. Yeast genomes in experiments

Yeast genomes selected for the experiments are well-studied eukaryote species that shared a common ancestor with human millions of years ago. *Saccharomyces cerevisiae* belongs to *Saccharomycete yeasts* and *Schizosaccharomyces pombe* is a unicellular organism from *Archiascomycete fungus*. These species bring about experimental models for many essential processes of eukaryotes since most of their genes have been predicted as homologs to multicellular eukaryotes ones. Specifically, *S. pombe* genes are more similar to mammals'

genes than *S. cerevisiae* ones [23]. However, *S. pombe* is a distant relative of *S. cerevisiae*, accordingly, ortholog detection between them is a difficult task due to the divergence of sequences [13]. On the other hand, other close *Saccharomycete* species as *Kluyveromyces* (*Kluyveromyces lactis* and *Kluyveromyces waltii*) and *Candida glabrata* have been included in one of the ortholog detection benchmarking datasets [10] since some of them underwent a whole genome duplication (WGD) process causing a rampant paralogy and a lot of gene losses. Precisely, the genome pair of *C. glabrata* and *S. cerevisiae* that are post-WGD species constitutes a target study case for OD algorithms. For the studies reported below we have identified each genome pair as follows:

- *S. cerevisiae – K. lactis*: ScerKlac
- *K. lactis – K. waltii*: KlacKwal
- *C. glabrata – K. lactis*: CglaKlac
- *S. cerevisiae – C. glabrata*: ScerCgla
- *S. cerevisiae – S. pombe*: ScerSpombe.

2.2. Similarity measures as gene pair features in ortholog detection

Five normalized pairwise sequence similarities (global and local alignment scores, protein length, synteny and physicochemical profile) were previously specified in [24]. **Table 1**

Measure	Description of the calculation
S_1 – local and global alignment	The sequence alignment measure averages positives local and global protein alignment scores from the Smith Waterman [32] and the Needleman-Wunsch [33] algorithms calculated with a specified scoring matrix, and "gap open" (GOP) and "gap extended" (GEP) parameters. The scores are normalized by using the maximum score of all (x_i, y_j) protein pairs
S_2 – protein length	Measure S_2 represents the normalized difference for continuous values [34] of the protein sequence lengths
S_3 – membership to locally collinear blocks	This measure is calculated from the distance between pairs of sequences considering their membership to Locally Collinear Blocks (LCBs) which represent truly homologous regions obtained with the Mauve software [35]. The normalized difference is selected for the comparison of the total number of codons in each block
S_4 – physicochemical profile	This measure is based on the spectral representation of sequences from the global protein pairwise alignment by using the linear predictive coding [34]. First, each amino acid that lies in a matching region without "gaps" between two aligned sequences is replaced by its contact energy [36]. The moving average for each spectrum, that is, the average of this physicochemical feature in the predefined window size W is then calculated. Next, the Pearson correlation coefficient with the corresponding significance level between the two spectral representations in a matching region is obtained. Finally, the significant similarities of the regions without "gaps" are aggregated considering the length of each region. From our previous studies presented in [37, 38], three features for the physicochemical profile with W values of 3, 5 and 7 have been considered

Table 1. Protein pair features.

summarizes the calculations for protein pairs (x_i, y_j) from the two proteome representations $P_1=\{x_1, x_2, \ldots, x_n\}$ and $P_2=\{y_1, y_2, \ldots, y_m\}$ with n and m sequences, respectively.

2.3. From parallel to big data calculation of similarity measures

Time complexity analysis of the sequential algorithms for the calculation of pairwise similarity measures in terms of N gene/protein pairs has brought about the need of a parallel version with the scalability target. The protein physicochemical profile turns to be the one with the highest quadratic cost $O(N \times (m + n)^2)$ since it operates over the aligned sequences with m and n as the maximum sequence lengths for two genomes, respectively. The parallel proposal is sketched in **Figure 1** with a fork control flow to distribute the calculation over different processors and a join control flow to terminate the parallel executions.

For the scalability analysis of the parallel physicochemical profile calculation, we estimated the T_S sequential execution time in expression (1) by using constants.

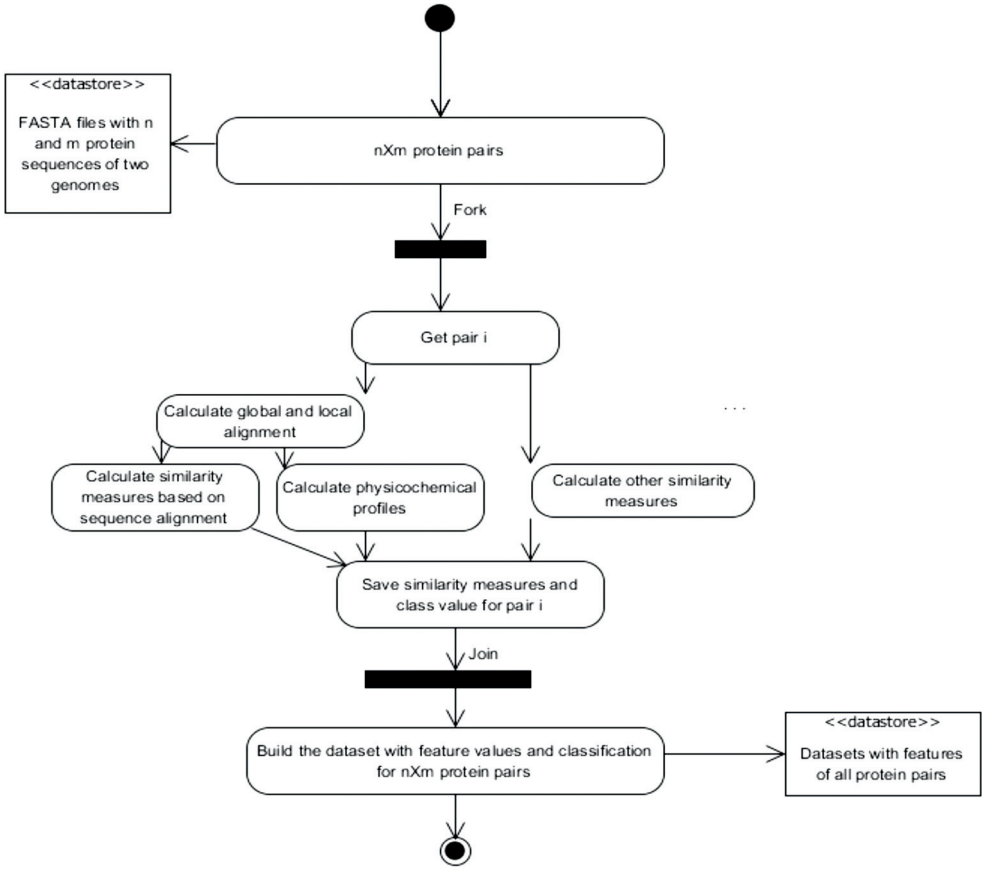

Figure 1. Parallel calculation of protein pair similarity measures.

$$T_S = N \times \left(n \times m + (m+n)^2\right) \times t_c \quad (1)$$

The constant t_c indicates the required time for the arithmetic operations. We assume t_c as optimum value, $t_c = 1$ [25]. Consequently, the parallel execution time T_P is defined in terms of the calculation time and the communication time among processors. In the parallel version, calculation time depends on the distribution of N cycles among P processors. With this distribution: (i) all the processors execute $\lfloor N/P \rfloor$ iterations or (ii) $p < P$ processors execute $\lfloor N/P \rfloor + 1$ iterations. Thus, when we consider the worst case, the calculation time T_{Calc} can be estimated as in expression (2).

$$T_{Calc} = (\lfloor N/P \rfloor + 1) \times \left(n \times m + (m+n)^2\right) \times t_c \quad (2)$$

The communication time among processors is directly related to the information accessed in the sequential section of the algorithm. In each iteration, each processor P needs to receive sequence information. Hence, communication time can be defined as in expression (3).

$$T_{Com} = (\lfloor N/P \rfloor + 1) \times (t_s + t_w \times P \times (m+n)) \quad (3)$$

where t_s represents the time required to establish the communication and to prepare the sending information and t_w is the required time to send a numeric value. These values can be considered as optimum ones, that is, $t_s = 0$ and $t_w = 1$ [25]. In this way, we combine expressions (2) and (3) to obtain an estimation of the parallel execution time for the physicochemical profile as in expression (4).

$$T_P = (\lfloor N/P \rfloor + 1) \times (n \times m + (m+n) \times (m+n+P)) \quad (4)$$

Time complexity of the parallel version of the algorithm can be estimated from the parallel execution time T_P. Expression $(\lfloor N/P \rfloor + 1)$ can be taken as N/P.

It holds that $m \times n \leq (m+n) \times (m+n+P)$. Therefore, time complexity is $O\left(\frac{N \times (n+m)^2}{P} + N \times (m+n)\right)$, that is lower than $O(N \times (m+n)2)$ as demonstrated in expression (5).

$$\frac{\frac{N \times (n+m)^2}{P} + N \times (m+n)}{N \times (n+m)^2} = \frac{1}{P} + \frac{1}{m+n} \leq 1 \quad (5)$$

By using expressions T_S and T_P, we can analyze the speed-up and the efficiency performance for different samples obtained from N protein pairs and a range of 1–100 processors. **Figure 2A** and **B** shows the speed-up and efficiency values, respectively, in a genome comparison dataset of *S. cerevisiae* and *S. pombe*. The 100% sample represents 5006 sequence pairs; the 75% sample represents 3754 pairs, the 50% represents 2503 pairs, and the 25% represents 1251 pairs. An increased number of processors lead to a saturation in the speed-up. Specifically, when the problem size increase, thus the speed-up and the efficiency values increase for a single number of processors, although both values tend to decrease with an increasing number of processors. The efficiency tends to keep constant when we simultaneously increase both the problem size and the number of processors, consequently, contributing to the scalable conception of the algorithm [26].

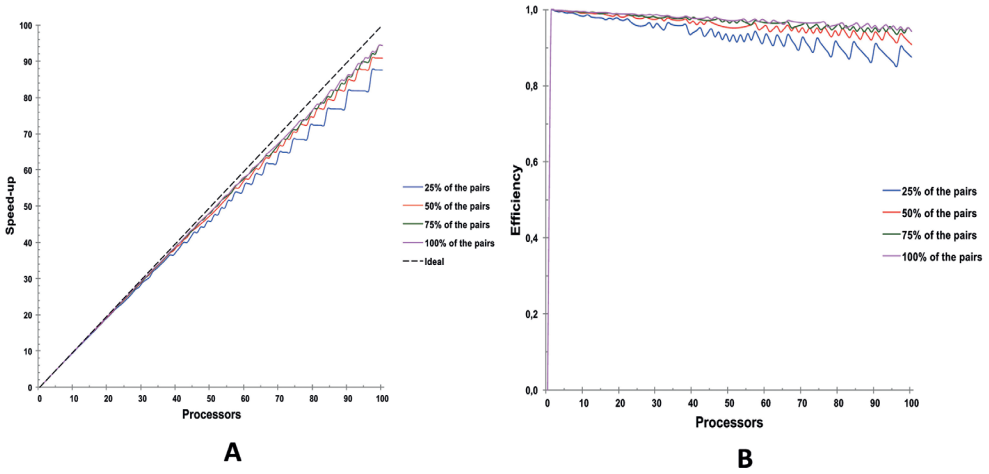

Figure 2. Estimated performance of the speed-up (A) and efficiency (B) in the parallel calculation of the physicochemical profile of proteins for the genome comparison dataset of *S. cerevisiae* and *S. pombe*.

With the sequential and parallel MATLAB (2010) implementations, we calculated real execution time for one, two, three and four processors in a personal computer Intel® Core™ i3 CPU, 2.53 GHz, RAM DDR3 de 4.0 Gb, 64Bits Windows7. With 1921 pairs of *S. cerevisiae* and *S. pombe* representing 100% of the sample, **Figure 3A** and **B** shows the real and estimated values of the speed-up and efficiency, respectively. We can observe a similar performance of estimated and real values for the total and 50% of the sample. The best speed-up was obtained for the execution in four processors, but the efficiency is best with two processors.

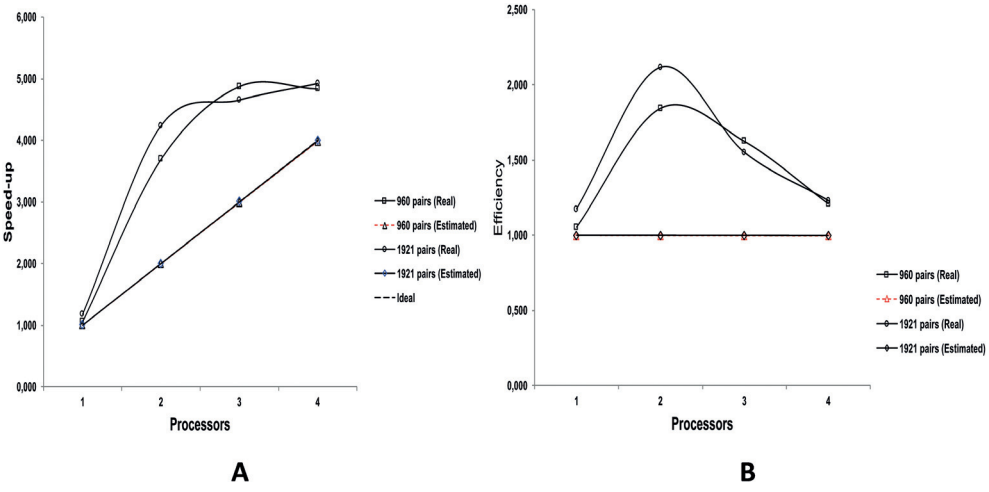

Figure 3. Real and estimated values for speed-up (A) and efficiency (B) in a *S. cerevisiae* and *S. pombe* sample.

Theoretically, the speed-up value is bounded by the number of processors and the efficiency value is in the interval [0, 1]. However, the real speed-up obtained is higher than the number of processors in each execution and the efficiency is higher than one. This event is known as super-linear speed-up. It is a consequence of the fact that each processor consume less time than $\frac{T_S}{P}$. It can be possible since the parallel version may execute less work than the sequential version or may take advantage of resources such as the cache memory [25].

In a further scalability analysis with 133,666,445 pairs of sequences from different yeast genomes, the increased number of processors from 1 to 100 saturates the speed-up. **Table 2** shows the genome data used for this scalability analysis. Datasets are conformed by accumulating gene pairs from different genome pairs with different maximum sequence lengths. **Figure 4A** illustrates the effects of the gene pair accumulation in the speed-up of the parallel physicochemical profile similarity measure calculation while **Figure 4B** depicted the effect in

Datasets	Total of gene pairs	Maximum length for the first genome *m*	Maximum length for the second genome *n*
ScerSpombe	16,324,500	4910	4924
ScerSpombe + ScerKlac	47,546,047	4910	4924
ScerSpombe + ScerKlac + ScerCgla	78,111,162	4910	4924
ScerSpombe + ScerKlac + ScerCgla + CglaKlac	105,891,467	4910	4924
ScerSpombe + ScerKlac + ScerCgla + CglaKlac + KlacKwal	133,666,445	4915	4924

Table 2. Genome data used in the scalability analysis.

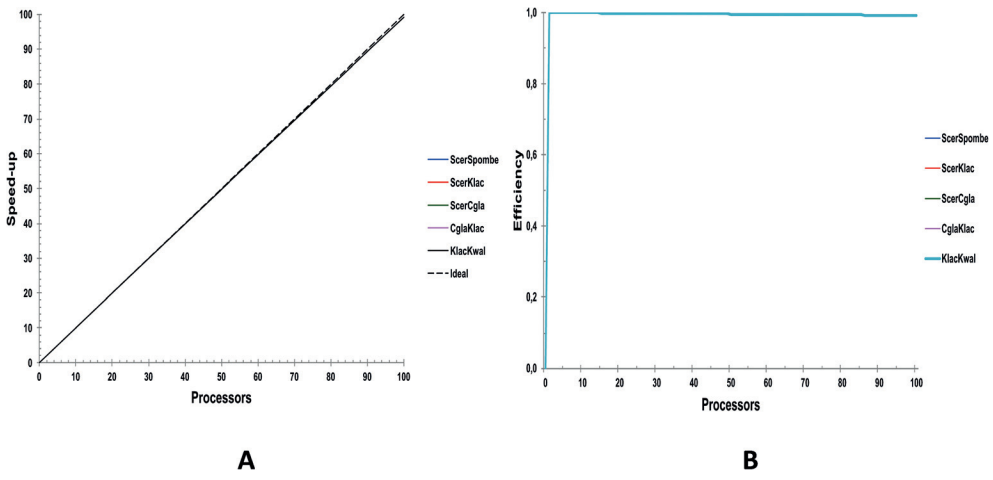

Figure 4. Estimated speed-up (A) efficiency (B) and when the input data and processors increase.

the efficiency of this calculation. When the problem size increases, the speed-up and the efficiency increase for a specific number of processors, although both values tend to decrease when the number of processors increases. Efficiency tends to be constant when both the problem size and the number of processors increase. This is an essential issue to achieve scalability, specifically, the horizontal one.

In sum, the parallel algorithm time complexity considering P processors is lower than the cost of the sequential version and the scalability may be achieved with the parallel proposal. Nevertheless, considering implementation hazards of the parallel models, scalability can be improved when these models are executed in a cloud within a big data framework as MapReduce [27, 28]. This framework guarantees reliability, availability, and a good execution performance in a cloud with a distributed file system.

In a MapReduce design, each mapper process should build a subset of the resulting dataset with calculated features of all protein pairs of its partition. **Figure 5** shows three steps such as Initial, Map and Final. The Initial phase divides the protein pair set into independent Hadoop distributed file system (HDFS) blocks; it also duplicates and transfers them to other nodes of the cloud. Then, in the Map step, each mapper calculates the protein features of all the pairs in its partition. Finally, in the Final step, the files created by each mapper would be concatenated to build the final dataset file. In this procedure the Reduce step is omitted since the mappers output are directly combined following the operation reported in [29].

Algorithm 1 shows el pseudo-code of the Map function in the MapReduce process for the protein feature calculations. Step 2 calculates the features for a pair of proteins and Step 5 save the previously calculated data.

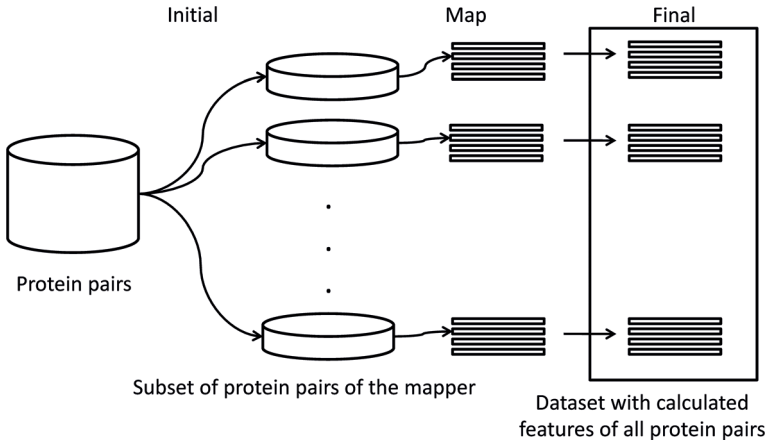

Figure 5. MapReduce design of protein pair feature calculations.

```
MAP (key, value)

Input: <key; value> % key represents the location in bytes, and value, the
contents of a protein pair.

Output: <key'; value'> % key' indicates the identification of the pair, and
value', the calculated features.

1  pair ← PAIR_REPRESENTATION(value)
% features variable contains the values of the feature for the pair

2  features ← CALCULATE_FEATURES(pair)

3  lkey ← lkey:set(pair:get_id)

4  lvalue ← lvalue:set(features)

5  EMIT (lkey, lvalue)
```

Algorithm 1. Map phase for feature calculation.

2.4. Big data supervised classification with imbalance management

Given a set $A=\{S_r\ (x_i,\ y_j)\}$ of gene pair features as discrete or continuous values of r gene pair similarity measure functions, previously specified, we represent a pairwise ortholog detection decision system $DS = (U, A \cup \{d\})$, where $U = \{(x_i, y_j)\}, \forall x_i \in G_1, \forall y_j \in G_2$ is the universe of the gene pairs, and $d \notin A$ is the binary decision feature obtained from a curated classification. This decision feature defines the low ratio of orthologs to the total number of possible gene pairs. The big data supervised classification divides DS into train and test sets to build a learning/training model and to classify the instances by means of a big data supervised algorithm managing the imbalance between classes. The built model can be generalized to related pairs of genomes/yeast species not used in the learning/training process following the generalization concept in [30]. Thus, the test set is used to compare both supervised and classical unsupervised algorithms. **Algorithm 2** shows the steps required in the training phase as well as **Algorithm 3** shows the classification phase. The general testing/evaluation scheme for supervised and unsupervised algorithms is sketched in **Figure 6** based on the use of a testing external set as reported in [31].

```
Training phase

Input: DS=(U, A∪{d}), A={Sr(xi, yj)}, U={(xi, yj)}, ∀xi∈G1, ∀yj∈G2

Output: Model, Test_set

1  Train_set, Test_set =
       Selection_of_training_and_testing_sets (DS)

2  Model = Model_building_with_imbalance_management  %Learning/Training step
```

Algorithm 2. Training phase for big data supervised classification with imbalance management.

```
Classification phase

Input: Model, Test_set
```

```
Output: Ortholog_pairs_of_the_testing_set

1 Ortholog_pairs_of_the_testing_set = Pairwise_classification (Model,
Test_set)
```

Algorithm 3. Classification phase for big data supervised ortholog detection with imbalance management.

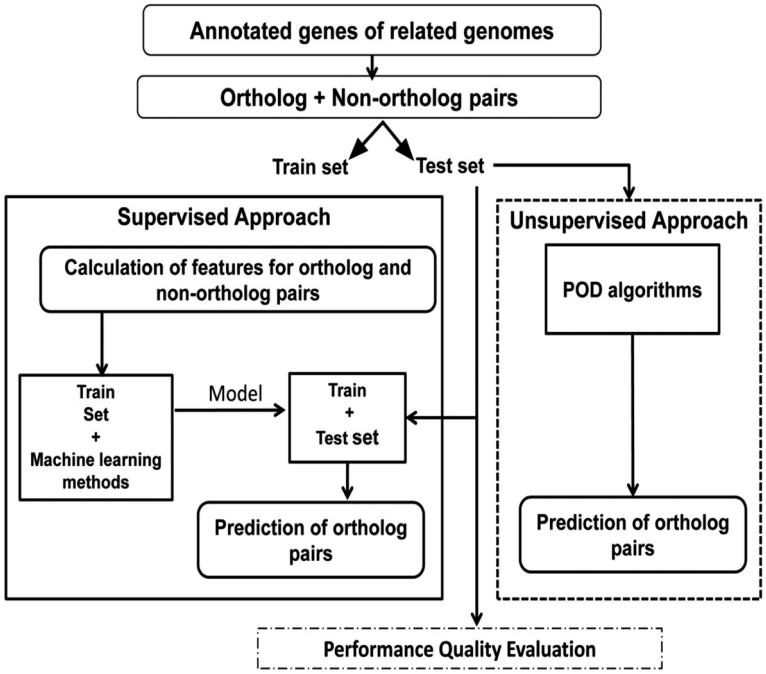

Figure 6. Workflow of the evaluation of supervised versus unsupervised pairwise ortholog detection algorithms.

The general scheme receives annotated genomes (proteomes) from related yeast species since some protein features as the membership to conserved regions require certain evolution closeness. For the evaluation of pairwise ortholog detection algorithms, we can compare the supervised solutions and the unsupervised reference algorithms such as RBH, RSD, and OMA. Firstly, protein pairs are separated into train and test sets and after that pairwise similarity measures for the pairs of both sets are calculated. Test sets are used for the assessment of the unsupervised algorithms while the training set is used to build the supervised models which will be further tested on the previously mentioned test set.

The classification performance will be assessed by evaluation metrics for imbalanced datasets. The Geometric Mean (*G-Mean*) is defined in [32] as:

$$G - Mean = \sqrt{sensitivity * specificity} \tag{6}$$

where sensitivity and specificity are traditionally calculated considering the true positives (*TP*), false negatives (*FN*), false positives (*FP*), and true negatives (*TN*).

The area under the curve (*AUC*) [33] is estimated from plotting of true positive rate (*TPR*) versus false positive rate (*FPR*) values. We approximate the *AUC* by averaging (*TPR*) and (*FPR*) values through the following equation:

$$AUC = \frac{1 + TPR - FPR}{2} \tag{7}$$

where $TPR = \frac{TP}{TP+FN}$ represents the percentage of orthologs (*TP*) correctly classified and $FPR = \frac{FP}{FP+TN}$, the percentage of non-orthologs (*TN*) misclassified. *G-Mean* measure maximizes the accuracy of the two classes (orthologs and non-orthologs) by considering the misclassification costs and *AUC* values in the estimation of the sensitivity and specificity, getting at a balance between them [34].

2.5. Ortholog detection experiments in related yeasts

2.5.1. Settings

For the evaluation of pairwise ortholog detection algorithms in related yeast genomes we carried out (i) three exploratory experiments for an external validation of supervised classification models and (ii) an experiment to evaluate a model built with a pair of yeasts into two external pairs. In Experiment 1, we evaluated the algorithms inside a genome pair by partitioning the complete set of protein pairs at random (75% for training and 25% for testing). Specifically, we divided the *S. cerevisiae* – *K. lactis* dataset into 16,986,996 pairs for training and 5,662,332 pairs for testing. In Experiment 2, we evaluated the model in the first experiment into 8,095,907 pairs of *S. cerevisiae* and *S. pombe* genomes. In Experiment 3, we tested in *S. cerevisiae* and *S. pombe* as in the second experiment but with models obtained by training with the complete *S. cerevisiae* – *K. lactis* dataset (22,649,328 pairs). In Experiment 4, we used the model in the third experiment and tested it in two different pairs: 29,887,416 pairs of *S. cerevisiae* and *C. glabrata* and 20,318,472 pairs of *C. glabrata* – *K. lactis*.

The details for the data used in the experiments are specified in **Table 3**. For each genome pair we built four datasets (BLOSUM50, BLOSUM62_1, BLOSUM 62_2, and PAM250) from combinations of alignment parameter settings shown in **Table 4**. The gene pair features included in the datasets are the average of local and global alignment similarity measures, the length of sequences, the gene membership to conserved regions (synteny), and the physicochemical

Genome pair	Number of features	Instances per class (majority; minority)	Imbalance ratio (*IR*)	Excluded genes
S. cerevisiae – *K. lactis*	6	(22,646,914; 2414)	9381.489	89 out of 5861 *S. cerevisiae* proteins 37 out of 5215 *C. glabrata* proteins 1403 out of 5327 *K. lactis* proteins
S. cerevisiae – *C. glabrata*	6	(29,884,575; 2841)	10519.034	
C. glabrata – *K. lactis*	6	(20,317,232; 1240)	16384.865	
S. cerevisiae – *S. pombe*	6	(8,090,950; 4957)	1632.227	

Table 3. Genome data used in the experiments.

Dataset	Substitution matrix	Gap open	Gap extended
BLOSUM50	BLOSUM50	15	8
BLOSUM62_1	BLOSUM62	8	7
BLOSUM62_2	BLOSUM62	12	6
PAM250	PAM250	10	8

Table 4. Datasets built with different alignment parameter values.

profiles within 3, 5, and 7 window sizes. We are aggregating global and local alignment similarities to combine structural and functional similarities of proteins.

The *S. cerevisiae* – *S. pombe* dataset represents the union of Inparanoid7.0 and GeneDB ortholog classifications as is described in [35]. On the other hand, the *S. cerevisiae* – *K. lactis* and *S. cerevisiae* – *C. glabrata* datasets contain all ortholog pairs in the gold groups reported in [10]. At the time of building the subset with all possible pairs, we just excluded 89 genes from *S. cerevisiae*, 37 from *C. glabrata*, and 1403 from *K. lactis* because the genome physical location data in the YGOB database [36], required for the LCB feature calculation, was not found.

The amount of pairs in each dataset prevented traditional machine learning methods from training and testing, so big data implementations were selected from scalable MapReduce Mahout [21] and Spark MLlib [37] libraries. The selected algorithms for the evaluation are listed in **Table 5**. Explicitly, we selected the random oversampling (ROS) and the cost-sensitive approaches for imbalance management. Besides, the big data framework details are specified in **Table 6**. On the other hand, the unsupervised algorithms parameter values used for the evaluation are specified in **Table 7**.

The infrastructure used to perform the big data experiments consists of 20 nodes connected via a 40 GB/s Infiniband network. Each node has two Intel Xeon E5-2620 microprocessors (at 2 GHz, 15 MB cache) and 64 GB of main memory working under Linux CentOS 6.5. The head node of the cluster has two Intel Xeon E5-2620 microprocessors (at 2.00 GHz, 15 MB cache) and 32 GB of main memory.

2.5.2. Results

Supervised classifiers that manage big data were compared from two different points of view:

i. The efficacy and the execution time when the training process is performed with one partition of the genome pair dataset or with the complete dataset.

ii. The efficacy and the execution time in the classification process performed on datasets with potential fails for ortholog detection:

- The one of related species (distant relatives), *S. cerevisiae* and *S. pombe*, false negatives can increase in the presence of divergent sequences.
- Other two *Saccharomycete* yeast datasets which complexity is a possible increased number of false positives due to lots of paralog losses produced by the WGD.

Algorithm	Parameter values
RF-BD[1]	Number of trees: 100 Random selected features per node: 3[2] Number of maps: 20
RF-BDCS	Number of trees: 100 Random selected features per node: 3 Number of maps: 20 $C(+\mid -)=IR$ $C(-\mid +)=1$
ROS (100%) + RF-BD	RS[3] = 100%
ROS (130%) + RF-BD	RS = 130%
SVM-BD	Regulation parameter: 1.0, 0.5, and 0.0 Number of iterations: 100 (by default) StepSize: 1.0 (by default) miniBatchFraction: 1.0 (percent of the dataset evaluated in each iteration 100%)
ROS (100%) + SVM-BD	RS = 100%
ROS (130%) + SVM-BD	RS = 130%

[1]BD means big data.
[2]int.($\log_2 N + 1$), where N is the number of features of the dataset.
[3]RS represents resampling size.

Table 5. Supervised algorithms and parameter values selected for the experiments.

Big data framework	Application	Algorithms
Hadoop 2.0.0 (Cloudera CDH4.7.1) with the head node configured as name-node and job-tracker, and the rest, as data-nodes and task-trackers	– MapReduce ROS implementation – A cost-sensitive approach for Random Forest MapReduce algorithm (RF-BD) – MapReduce RF implementation (Mahout Library)	RF-BDCS ROS (100%) + RF-BD ROS (130%) + RF-BD
Apache Spark 1.0.0 with the head node configured as master and name-node, and the rest, as workers and data-nodes	– Apache Spark Support Vector Machines (MLLib)	ROS (100%) + SVM-BD ROS (130%) + SVM-BD

Table 6. Big data framework, applications and algorithms.

Supervised and unsupervised classifiers are compared in terms of the efficacy in complex *Saccharomycete* yeast datasets.

Best mean results of *G-Mean*, *AUC*, *TPR*, and *TNR* (true negative rate) in Experiments 1, 2, and 3 are shown in **Tables 8–10**, respectively, with the most outstanding values in bold face. The best performance of *G-Mean*, *AUC* and the balance between *TPR* and *TNR* mean values were obtained with the cost-sensitive Big Data Random Forest algorithm (RF-BDCS) in Experiment 1 (testing in a partition of *S. cerevisiae* – *K. lactis*) and with support vector machine-BigData (SVM-BD) (regParam: 0.5) in Experiments 2 and 3 (testing in *S. cerevisiae* – *S. pombe*). There are no significant changes in classifiers with the best results (those that manage data imbalance)

Algorithm	Parameter values	Implementation
RBH	Soft filter and Smith Waterman alignment E-value = 1e−06	BLASTp program[1] Matlab script
RSD	E-value thresholds: 1e−05, 1e−10 and 1e−20 Divergence thresholds α: 0.8, 0.5 and 0.2.	BLASTp program[1] Python script[2]
OMA	Default parameter values	OMA stand-alone[3]

[1]Available in http://www.ncbi.nlm.nih.gov/BLAST/.
[2]Available in https://pypi.python.org/pypi/reciprocal_smallest_distance/1.1.4/.
[3]Available in http://omabrowser.org/standalone/OMA.0.99z.3.tgz.

Table 7. Unsupervised algorithms and parameter values in the experiments.

Algorithm	*G-Mean*	*AUC*	*TPR*	*TNR*
ROS (RS: 100%) + RF-BD	0.9807	0.9809	0.9621	1.0000
ROS (RS: 130%) + RF-BD	0.9812	0.9813	0.9629	0.9997
RF-BDCS	**0.9889**	**0.9889**	**0.9788**	**0.9997**
ROS (RS: 100%) + SVM-BD (regParam: 1.0)	0.9477	0.9477	0.9369	0.9586
ROS (RS: 100%) + SVM-BD (regParam: 0.5)	0.8791	0.8845	0.93038	0.96259
ROS (RS: 130%) + SVM-BD (regParam: 0.5)	0.8528	0.8629	0.93893	0.95476

Table 8. Best mean *G-Mean*, *AUC*, *TPR* and *TNR* results in Experiment 1 in *S. cerevisiae* – *K. lactis*. Highlighted *TPR* and *TNR* with the best stability in these measures.

Algorithm	*G-Mean*	*AUC*	*TPR*	*TNR*
RF-BDCS	0.7103	0.7542	0.5570	0.9997
ROS (RS: 100%) + SVM-BD (regParam: 1.0)	0.8543	0.8641	0.7432	0.9923
ROS (RS: 100%) + SVM-BD (regParam: 0.5)	**0.8795**	**0.8848**	**0.7900**	**0.9796**
ROS (RS: 100%) + SVM-BD (regParam: 0.0)	0.4312	0.5846	0.9781	0.1911

Table 9. Best mean *G-Mean*, *AUC*, *TPR* and *TNR* results in Experiment 2 in *S. cerevisiae* – *S. pombe*. Highlighted *TPR* and *TNR* with the best stability in these measures.

Algorithm	*G-Mean*	*AUC*	*TPR*	*TNR*
RF-BDCS	0.6745	0.7294	0.4590	0.9998
ROS (RS: 100%) + SVM-BD (regParam: 1.0)	0.8533	0.8632	0.7332	0.9933
ROS (RS: 100%) + SVM-BD (regParam: 0.5)	**0.8791**	**0.8845**	**0.7895**	**0.9795**
ROS (RS: 130%) + SVM-BD (regParam: 1.0)	0.7956	0.8164	0.6348	0.9979
ROS (RS: 130%) + SVM-BD (regParam: 0.5)	0.8528	0.8629	0.7331	0.9926

Table 10. Best mean *G-Mean*, *AUC*, *TPR* and *TNR* results in Experiment 3 in *S. cerevisiae* – *S. pombe*. Highlighted *TPR* and *TNR* with the best stability in these measures.

when we use the four alignment parameter value combinations in **Table 4**. Similarly, good efficacy is obtained when we trained with a partition of *S. cerevisiae – K. lactis* and with the corresponding complete training set.

The SVM algorithm combined with ROS showed the best time results in Experiment 1 (SVM-BD (regParam: 0.5) executed in 14 minutes and 19 seconds) and (SVM-BD (regParam: 0.0) executed in 14 minutes and 21 seconds). These algorithms also showed good classification quality, mainly, in the balance between *TPR* and *TNR*. In Experiments 2 and 3 SVM variants combined with ROS (RS:100) run very rapidly, in particular, SVM-BD (regParam: 0.5) executed in 14 minutes and 29 seconds in Experiment 2 and in 16 minutes and 53 seconds in Experiment 3.

Analyzing measure values of Experiment 4 we noticed that the performance of supervised classifiers is almost the same as in the previous experiments in terms of the stability in the four different alignment parameter value datasets. Best mean results of *G-Mean* and *AUC*, *TPR* and *TNR* for supervised and unsupervised classifiers are shown in **Table 11**. The underlined values highlight the most effective method in the experiment while the bold face values identify the best performing supervised and unsupervised algorithms. Supervised Random Forest classifiers managing data imbalance overdid the unsupervised ones, specifically, when the cost-sensitive version is applied. Among the unsupervised classifiers, RSD reaches the highest *G-Mean* (0.9374) using the recommended parameter values (*E-value* = 1e−05 and $\alpha = 0.8$) in [38]), higher values were also obtained for *AUC* and *TPR*. On the contrary, OMA was the best among the unsupervised algorithms in *S. cerevisiae – S. pombe* datasets.

The results of the Friedman test [39] for the *AUC* measure with the four datasets of *S. cerevisiae – C. glabrata*, *C. glabrata – K. lactis*, and *S. cerevisiae – S. pombe* and the supervised classifiers that manage imbalance are shown in **Table 12** (first column). The test yielded significant differences among classifiers, being RF-BDCS the one with the high mean rank followed by the classifiers based on Random Forest with ROS preprocessing and the SVM classifier with ROS (RS:100) and 0.5 regulation. In the Friedman test for *G-Mean* (**Table 12** second column) there are also significant

Dataset/algorithm	***S. cerevisiae – C. glabrata***				***C. glabrata – K. lactis***			
	AUC	***G-Mean***	***TPR***	***TNR***	***AUC***	***G-Mean***	***TPR***	***TNR***
RBH	0.8196	0.7995	0.6392	0.9999	0.8052	0.8242	0.6484	0.9999
RSD 0.2 1e−20	0.9238	0.9206	0.8476	0.9999	0.9047	0.9092	0.8185	0.9999
RSD 0.5 1e−10	0.9340	0.9316	0.8680	0.9999	0.9267	0.9294	0.8589	0.9999
RSD 0.8 1e−05	**0.9382**	**0.9362**	**0.8765**	**0.9999**	**0.9354**	**0.9374**	**0.8750**	**0.9999**
OMA	0.9287	0.9259	0.8574	0.9999	0.9125	0.9163	0.8328	0.9999
ROS (RS: 100%) + RF-BD	0.9901	0.9900	0.9805	0.9997	0.9806	0.9811	0.9615	0.9997
ROS (RS: 130%) + RF-BD	0.9901	0.9901	0.9806	0.9997	0.9807	0.9807	0.9617	0.9997
RF-BDCS	**0.9934**	**0.9934**	**0.9876**	**0.9993**	**0.9847**	**0.9862**	**0.9702**	**0.9992**

Table 11. Best mean *G-Mean, AUC, TPR* and *TNR* results in Experiment 4 in *S. cerevisiae – C. glabrata* and *C. glabrata – K. lactis.* Highlighted *TPR* and *TNR* with the best stability in these measures.

Algorithm	*AUC* mean rank	*G-Mean* mean rank	Execution time mean rank
ROS (RS: 100%) + RF-BD	5.96	5.71	8.00
ROS (RS: 130%) + RF-BD	5.96	5.79	9.00
RF-BDCS	7.67	7.67	7.00
ROS (RS: 100%) + SVM-BD (regParam: 1.0)	2.62	6.12	2.33
ROS (RS: 100%) + SVM-BD (regParam: 0.5)	5.50	5.79	2.25
ROS (RS: 100%) + SVM-BD (regParam: 0.0)	5.12	2.00	2.58
ROS (RS: 130%) + SVM-BD (regParam: 1.0)	2.92	4.46	4.33
ROS (RS: 130%) + SVM-BD (regParam: 0.5)	4.79	5.62	4.50
ROS (RS: 130%) + SVM-BD (regParam: 0.0)	4.46	1.83	5.00
Test statistics			
N	12.000	12.000	12.000
Chi-square	31.359	47.850	80.333
df	8.000	8.000	8.000
Asymp. Sig.	0.000	0.000	0.000

Table 12. Results of the Friedman tests comparing supervised classifiers in terms of *AUC*, *G-Mean* and execution time in the four datasets of *S. cerevisiae – C. glabrata*, *C. glabrata – K. lactis* and *S. cerevisiae – S. pombe*.

differences with higher mean rank values for RF-BDCS and ROS (RS: 100%) + SVM-BD (regParam: 1.0). When we applied the same test to compare execution time (**Table 12** third column) there are significant differences among the algorithms. In particular, SVM classifiers combined with ROS exhibited the best mean ranks.

2.5.3. Discussion

As a general result, experiments showed that supervised classifiers changed only slightly with the selection of different alignment parameters, maybe because of the appropriate combination of scoring matrices and gap penalties in relation to the sequence diversity between the two yeast genomes. The four scoring matrices assessed have been previously recommended to detect homologs in a wide range of amino acid identities. Moreover, gap penalty settings were not low enough to affect the sensitivity of the alignment [40].

The ROS pre-processing method for big data made SVM effective for pairwise ortholog detection and improved the performance of Random Forest for big data even more with a higher value for the resampling size parameter of 130% [41]. Conversely, the experiments showed that the variation in this parameter value from 100 to 130% did not significantly influence on the performance of the SVM big data classifier with different regulation values.

The cost-sensitive classifier RF-BDCS showed the best performance in *S. cerevisiae – C. glabrata*, *C. glabrata – K. lactis* and *S. cerevisiae – K. lactis* because probably it improved the training from the minority class. The best tree split was selected following the misclassification costs

assigned to the instances; such costs were also considered to associate certain class to a leaf [29]. This cost treatment does not imply changes in the sample distribution, and avoids possible overfitting that is commonly found in ROS solutions due to the presenceof duplicated instances. The setting of the cost values (($C(+|-)=IR$) and $C(-|+)=1$)) can also influence on the success of the algorithm.

In the case of SVM for big data classifier, the fixed regularization parameter defines the trade-off between the goal of minimizing the training error and minimizing the model complexity to avoid overfitting. The higher is its value, the simpler the model, however, a better performance in classification may be achieved by setting an intermediate regulation value, or one close to cero [37]. Specifically, the ROS (RS: 100%) + SVM-BD (regParam: 0.5) classifier exhibited the best *AUC* and *G-Mean* values in *S. cerevisiae – S. pombe*, and the best balance between *TPR* and *TNR* in the rest of the datasets.

All methodologies including the proposed supervised big data approach generally declined their performance for ortholog detection in *S. cerevisiae – S. pombe* datasets, probably because of *S. pombe* is a distant relative of *S. cerevisiae* [23]. The supervised classifiers performance was also negatively affected by differences in data distribution between the train and test sets [42]. On the contrary, ROS (RS: 100%) + SVM-BD (regParam: 0.5) remained stable in *S. cerevisiae – C. glabrata*, *C. glabrata – K. lactis* and *S. cerevisiae – S. pombe* datasets when considering the balance between TP_{Rate} and TN_{Rate}. Best results obtained in *S. cerevisiae – C. glabrata* are outstanding where algorithms are vulnerable to produce false positives since both genomes underwent a WGD and a subsequent differential loss of gene duplicates [10].

The initial assumption of RBH, RSD and OMA that the sequences of orthologous genes/proteins are more similar to each other than they are to any other genes from the compared organisms may produce classification errors [12]. The reduced quality shown by RBH, RSD and OMA, mainly in the case of RBH, could be caused by this assumption despite that BLAST parameters can be tuned as has been recommended in [43]. In particular, RBH infer orthology relationships simply based on reciprocal BLAST best hits. In contrast, the RSD procedure is less likely than RBH to be misled by existing close paralogs since it relies on both global sequence alignment and maximum likelihood estimation of evolutionary distances finding many putative orthologs missed by RBH. The OMA algorithm also displays advantages over RBH by using evolutionary distances instead of alignment scores. It allows the inclusion of one-to-many and many-to-many orthologs considering the uncertainty in distance estimations and detects potential differential gene losses.

On the other hand, the success of big data supervised classifiers managing imbalance over RSD and OMA may be explained by feature combinations together with the learning from curated classifications. The assembling of alignment measures with the comparison of sequence lengths, the membership of genes to conserved regions (synteny) and the physicochemical profiles of amino acids improved the detection of homology and certainly the supervised classification results on the test sets, even if both species underwent WGD. Specifically, the aggregation of global and local alignment scores allows us to combine protein structural and functional relationships between sequence pairs, respectively. The periodicity of the physicochemical properties of amino acids can detect similarity among protein pairs with

sequences having functional similarities despite their low amino acid sequence identities (<35%). These sequences may affect ortholog detection in *S. cerevisiae* – *S. pombe* which are moderately related and their orthologs may be diverged. The synteny information lets us to consider that genes belonging to the same conserved segment in genomes of different species will probably be orthologs. Besides, the length of sequences as the relative positions of amino acids within the same protein in different species and in duplicated regions within the same species may also contribute to the enhanced supervised classification results.

3. Conclusions

The combination of alignment measures with other protein pair features such as the sequence lengths, the gene membership to conserved regions, and the physicochemical profiles has complemented the homology detection in the proposed supervised approach for pairwise ortholog detection. Such combined features alongside curated orthologs pairs extracted from a curated dataset have led to an effective and efficient ortholog classification method in a big data scenario with the treatment of the low ratio of orthologs to the total possible gene pairs between two genomes.

The supervised classifiers that manage imbalance outdid the popular unsupervised (RBH, RSD, and OMA) algorithms even when the supervised model was extended to yeast datasets containing "traps" for ortholog detection algorithms. In future research, the introduction of new gene pair features might improve the effectiveness and efficiency of the supervised algorithms.

The scalability analysis of the proposed gene pair feature calculation highlighted the advances that can be achieved in the scalability issue when we parallelize the calculation, and later on, when we implement a big data model for the same calculation.

Acknowledgements

GACh was funded by a Postdoc fellowship (SFRH/BPD/92978/2013) granted by the Portuguese Fundação para a Ciência e a Tecnologia (FCT). AA was partially supported by the Strategic Funding UID/Multi/04423/2013 through national funds provided by FCT and the European Regional Development Fund (ERDF) in the framework of the program PT2020, by the European Structural and Investment Funds (ESIF) through the Competitiveness and Internationalization Operational Program – COMPETE 2020 and by National Funds through the FCT under the project PTDC/AAG-GLO/6887/2014 (POCI-01-0124-FEDER-016845), and by the Structured Programs of R&D&I INNOVMAR (NORTE-01-0145-FEDER-000035 – NOVELMAR) and CORAL NORTE (NORTE-01–0145-FEDER-000036), and funded by the Northern Regional Operational Program (NORTE2020) through the ERDF. The funders had no role in the study's design, data collection and analysis, decision to publish, or preparation of the manuscript.

Author details

Deborah Galpert Cañizares[1], Sara del Río García[2], Francisco Herrera[2], Evys Ancede Gallardo[3], Agostinho Antunes[4,5] and Guillermin Agüero-Chapin[4,5]*

*Address all correspondence to: gchapin@ciimar.up.pt

1 Departamento de Ciencias de la Computación, Universidad Central "Marta Abreu" de Las Villas (UCLV), Santa Clara, Cuba

2 Computer Science and Artificial Intelligence Department, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, Granada, Spain

3 Centro de Bioactivos Químicos, Universidad Central "Marta Abreu" de Las Villas (UCLV), Santa Clara, Cuba

4 CIMAR/CIIMAR, Centro Interdisciplinar de Investigação Marinha e Ambiental, Universidade do Porto, Porto, Portugal

5 Departamento de Biologia, Faculdade de Ciências, Universidade do Porto, Porto, Portugal

References

[1] Fitch WM. Distinguishing homologous from analogous proteins. Systematic Zoology. 1970;**19**:99-113

[2] Vashist A, Kulikowski C, Muchnik I. Screening for ortholog clusters using multipartite graph clustering by quasi-concave set function optimization. In: Slezak D et al., editors. RSFDGrC 2005. Vol. 3642. LNAI. Berlin Heidelberg: Springer-Verlag; 2005. pp. 409-419

[3] Hirsh AE, Fraser HB. Protein dispensability and rate of evolution. Nature. 2001;**411**:1046-1049

[4] Östlund G, Schmitt T, Forslund K, Köstler T, Messina DN, Roopra S, et al. InParanoid 7: New algorithms and tools for eukaryotic orthology analysis. Nucleic Acids Research. 2010;**38**:D196-D203

[5] Chen F, Mackey AJ, Stoeckert CJ, Roos DS. OrthoMCL-DB: Querying a comprehensive multi-species collection of ortholog groups. Nucleic Acids Research. 2006;**34**(Database issue):D363-D3D8

[6] Dessimoz C, Cannarozzi G, Gil M, Margadant D, Roth A, Schneider A, Gonnet GH. OMA, A Comprehensive, Automated Project for the Identification of Orthologs from Complete Genome Data: Introduction and First Achievements. In: Comparative Genomics: RECOMB 2005 International Workshop, Dublin, Ireland, September 18-20, 2005 Proceedings. Edited by McLysaght A, Huson DH. Berlin, Heidelberg, Germany; 2005:61-72

[7] Wall DP, Fraser HB, Hirsh AE. Detecting putative orthologs. Bioinformatics. 2003;**19**(13): 1710-1711

[8] Kuzniar A, RCHJv H, Pongor S, JAM L. The quest for orthologs: Finding the corresponding gene across genomes. Trends in Genetics. 2008;**30**:1-13

[9] Towfic F, VanderPIas S, OIiver CA, Couture O, TuggIe CK, GreenIee MHW, et al. Detection of gene orthology from gene co-expression and protein interaction networks. BMC Bioinformatics. 2010;**11**(Suppl 3):S7

[10] Salichos L, Rokas A. Evaluating ortholog prediction algorithms in a yeast model clade. PLoS ONE. 2011;**6**(4):1-11

[11] Koonin EV. Orthologs, paralogs, and evolutionary genomics. Annual Review of Genetics. 2005;**39**:309-338

[12] Kristensen DM, Wolf YI, Mushegian AR, Koonin EV. Computational methods for gene orthology inference. Briefings in Bioinformatics. 2011;**12**(5):379-391

[13] Kamvysselis MK. Computational Comparative Genomics: Genes, Regulation, Evolution. Massachusetts: Massachusetts Institute of Technology; 2003

[14] Zheng XH, Lu F, Wang Z-Y, Zhong F, Hoover J, Mural R. Using shared genomic synteny and shared protein functions to enhance the identification of orthologous gene pairs. Bioinformatics. 2005;**21**(6):703-710

[15] Lechner M, Hernandez-Rosales M, Doerr D, Wieseke N, Thévenin A, Stoye J, et al. Orthology detection combining clustering and synteny for very large datasets. PLoS ONE. 2014;**9**(8):e105015

[16] Chen X, Zheng J, Fu Z, Nan P, Zhong Y, Lonardi S, et al. Assignment of orthologous genes via genome rearrangement. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2005;**2**(4):302-315

[17] Fu Z, Chen X, Vacic V, Nan P, Zhong Y, Jiang T. MSOAR a high-throughput ortholog assignment system based on genome rearrangement. Journal of Computational Biology. 2007;**14**:16

[18] Chen TW, Wu TH, Ng WV. DODO: An efficient orthologous genes assignment tool based on domain architectures. Domain based ortholog detection. BMC Bioinformatics. 2010;**11** (Suppl 7):S6

[19] Sonnhammer ELL, Gabaldón T, Sousa da Silva AW, Martin M, Robinson-Rechavi M, Boeckmann B, Thomas PD, Dessimoz C. Big data and other challenges in the quest for orthologs. Bioinformatics. 2014, **30**(21):1-6

[20] Alexeyenko A, Tamas I, Liu G, Sonnhammer ELL. Automatic clustering of orthologs and inparalogs shared by multiple proteomes. Bioinformatics. 2006;**22**(14):e9-e15

[21] Owen S, Anil R, Dunning T, Friedman E. Mahout in Action, 2011, Manning Publications Co., USA

[22] Krishnan S, Smith V. MLlib. 2013 [enero 2015]. Available from: https://spark.apache.org/docs/

[23] Wood V. *Schizosaccharomyces pombe* comparative genomics; from sequence to systems. In: Comparative genomics 2006 Jan 1 (pp. 233-285). Springer Berlin Heidelberg, Germany

[24] Galpert D, Río Sd, Herrera F, Ancede-Gallardo E, Antunes A, Agüero-Chapin G. An effective big data supervised imbalanced classification approach for ortholog detection in related yeast species. BioMed Research International [serial on the Internet]. 2015;**2015**:748681

[25] Grama A, Gupta A, Karypis G, Kumar V, editors. Introduction to Parallel Computing, Second Edition: Addison Wesley, Boston, USA; 2003

[26] Bonvin N. Linear Scalability of Distributed Applications. Suisse: École polytechnique fédérale de lausanne; 2012

[27] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. OSDI'04: Proceedings of the 6th Symposium on Operating System Design and Implementation. San Francisco, California, USA: USENIX Association; 2004. p. 137-150

[28] Fernández A, Río Sd, López V, Bawakid A, Jesus MJd, Benítez JM, et al. Big Data with cloud computing: An insight on the computing environment, MapReduce, and programming frameworks. WIREs Data Mining and Knowledge Discovery [serial on the Internet]. 2014

[29] Sd R, López V, Benítez JM, Herrera F. On the use of MapReduce for imbalanced Big Data using random forest. Information Sciences. 2014;**285**:112-137

[30] Bishop CM, editor. Pattern Recognition and Machine Learning: Springer Science+Business Media, LLC, NY, USA; 2006.

[31] Kuncheva LI, editor. Combining Pattern Classifiers Methods and Algorithms. Hoboken, New Jersey: John Wiley & Sons, Inc.; 2004

[32] Barandela R, Sánchez JS, García V, Rangel E. Strategies for learning in class imbalance problems. Pattern Recognition. 2003;**36**(3):849-851

[33] Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition. 1997;**30**(7):1145-1159

[34] He H, Garcia EA. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering. 2009;**21**(9):1263-1284

[35] Koch EN, Costanzo M, Bellay J, Deshpande R, Chatfield-Reed K, Chua G, et al. Conserved rules govern genetic interaction degree across species. Genome Biology. 2012;**13**(7)

[36] Byrne KP, Wolfe KH. The yeast gene order browser: Combining curated homology and syntenic context reveals gene fate in polyploid species. Genome Research. 2005;**15**:1456-1461

[37] Krishnan S, Smith V. Linear Support Vector Machines (SVMs). 2013 [enero 2015]. Available from: https://spark.apache.org/docs/latest/mllib-linear-methods.html#linear-support-vector-machines-svms

[38] DeLuca TF, Wu I-H, Pu J, Monaghan T, Peshkin L, Singh S, et al. Roundup: A multi-genome repository of orthologs and evolutionary distance. Bioinformatics. 2006;**22**(16): 2044-2046

[39] Trawinski B, Smetek M, Telec Z, Lasota T. Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms. International Journal of Applied Mathematics and Computer Science. 2012;**22**(4):867-881

[40] Pearson WR. Selecting the right similarity-scoring matrix. Current Protocols in Bioinformatics. 2013;**43**:3.5.1-3.5.9

[41] Triguero I, del Río S, López V, Bacardit J, Benítez JM, Herrera F. ROSEFW-RF: the winner algorithm for the ECBDL'14 big data competition: an extremely imbalanced big data bioinformatics problem. Knowledge-Based Systems. 2015;**87**:69-79

[42] Moreno-Torres JG, Llorà X, Goldberg DE, Bhargava R. Repairing fractures between data using genetic programming-based feature extraction: A case study in cancer diagnosis. Information Sciences. 2013;**222**:805-823

[43] Hagelsieb GM, Latimer K. Choosing BLAST options for better detection of orthologs as reciprocal best hits. Bioinformatics. 2008;**24**(3):319-324