Cluster Computing (2019) 22:487–508
https://doi.org/10.1007/s10586-018-2857-9

Spark-IDPP: high-throughput and scalable prediction of intrinsically disordered protein regions with Spark clusters on the Cloud

Bożena Małysiak-Mrozek[1] · Tomasz Baron[1] · Dariusz Mrozek[1]

Received: 8 November 2017 / Revised: 9 July 2018 / Accepted: 17 October 2018 / Published online: 7 November 2018

Abstract
Intrinsically disorder proteins (IDPs) constitute a significant part of proteins that exist and act in cells of living organisms. IDPs play key roles in central cellular processes and some of them are closely related to various human diseases, like cancer or neurodegenerative disorders. Identification of IDPs and studying their structural characteristics have become an important part of structural bioinformatics and structural genomics. However, growing amount of genomic and protein sequences in public repositories pose a pressure on existing methods for identification of IDPs. Large volumes of protein amino acid sequences need to be analyzed in terms of propensity to form disordered regions, and this task requires novel tools and scalable platforms to cope with this big biological data challenge. In this paper, we show how the identification of disordered regions of 3D protein structures can be efficiently accelerated with the use of Apache Spark cluster established and scaled on the public Cloud. For this purpose, we propose Spark-based meta-predictor (Spark-IDPP), which enables efficient prediction of disordered regions of proteins on a large-scale. Results of our performance tests show that, for large data sets, our method achieves almost linear speedup, when scaling out the computations on the 32-node Spark cluster located in the Azure cloud. This proves that through appropriate partitioning of data and by increasing the degree of parallelism, we can significantly improve efficiency of IDP predictions. Additionally, by using several basic predictors, aggregating their ranks in various consensus modes, and filtering the final outcome with a dedicated fuzzy filter, the Spark-IDPP increases the quality of predictions.

Keywords Bioinformatics · Big Data · Intrinsically disordered proteins · Scalable computations · Apache Spark · Cloud computing

1 Introduction

International efforts focused on understanding living organisms at various levels of molecular organization, including genomic, proteomic, methabolomic, and cell signaling levels, lead to huge proliferation of biological data collected in dedicated, and frequently, public repositories. The amount of data deposited in these repositories increases every year, and cumulated volume has grown to sizes that are difficult to handle with traditional analysis tools. This growth of biological data is stimulated by various international projects, such as 1000 Genomes. The project aims at sequencing genomes of at least one thousand anonymous participants from a number of different ethnic groups in order to establish a detailed catalog of human genetic variations [70]. As a result, it generates terabytes of genetic data. Apart from international initiatives and projects, like the 1000 Genomes, the proliferation of biological data is further accelerated by newly developed technologies for DNA sequencing, like Next Generation Sequencing (NGS) methods. These methods are getting faster and less expensive every year. They produce huge amounts of genetic data that require fast analysis in various phases of molecular profiling, medical diagnostics, and treatment of patients that suffer from serious diseases. However, although very useful, these methods additionally

This work was supported by Microsoft Research within Microsoft Azure for Research Award Grant and by Statutory Research funds of Institute of Informatics, Silesian University of Technology, Gliwice, Poland (Grant No. BK-213/RAu2/2018)

✉ Dariusz Mrozek
dariusz.mrozek@polsl.pl

[1] Institute of Informatics, Akademicka 16, 44-100 Gliwice, Poland

increase the huge gap between the number of known genetic sequences (DNA sequences), protein sequences [38, 61, 79] (which are encoded by genes in DNA, see Fig. 1), and 3D protein structures [36, 52]. For example, as of June 20, 2018 there were 209,775,348 DNA sequences and 639,804,105 WGS sequences in GenBank database [4], 557,713 reviewed and 116,030,110 non-reviewed protein amino acid sequences in UniProtKB/Swiss-Prot and UniProtKB/TrEMBL databases [7, 71], and (only) 141,616 three-dimensional protein structures in Protein Data Bank (PDB) [5]. Since deep insight into 3D protein structures is a key for understanding molecular mechanisms of many civilization diseases and for the production of effective drugs, structural genomics tries to determine and describe the 3D structure of every protein that is encoded by a given sequenced genome. This is done by combining traditional experimental methods, like X-ray crystallography or Nuclear Magnetic Resonance (NMR), with computational modeling approaches that use various prediction methods for structure determination [18, 54, 76, 80].

1.1 Intrinsically disordered proteins

Determination of 3D protein structures became an important part of protein bioinformatics, since the knowledge of 3D protein structures allows to draw conclusions about molecular mechanisms of cellular biochemical reactions or particular diseases, and it supports drug design. However, as was gradually observed from 1990s, not all known proteins have stable (ordered) native 3D structure. Some proteins have shorter or longer segments that indicate instability or flexibility, which are called *intrinsically disordered regions* (IDRs). Such proteins are usually known as *intrinsically disordered proteins* (IDPs), and they may consist of one or several IDRs, or, in extreme case, can be completely unstructured (*intrinsically unstructured proteins*, IUPs). Intrinsically disordered proteins play many important roles in cells of living organisms, and on the basis of various studies carried out by scientists on whole proteomes, it is estimated that the percentage of IDPs in mammals is very large [16]. Determination of 3D structures of intrinsically disordered proteins with traditional methods, like the X-ray crystallography or Nuclear Magnetic Resonance (NMR), is difficult, since, e.g., the lack of electron density in crystal structures (which is marked in PDB files describing protein macromolecules as REMARK465 record). For this reason, IDP predictors have become playing an important role in determination of unstructured regions. IDP prediction became an important part of the computational protein structure determination. It supports studies of protein structural features, functional analysis of proteins, and investigations of the relationships between IDRs and the occurrence of particular diseases. As strictly computational tools, IDP predictors are able to predict protein disorder from pure amino acid sequence. However, taking into account the number of protein amino acid sequences in the UniProtKB/Swiss-Prot database, it is essential to provide efficient and effective tools for IDP prediction. These tools should be able to scale the computational procedure in order to accommodate the growing volume of DNA, and consequently, protein sequences.

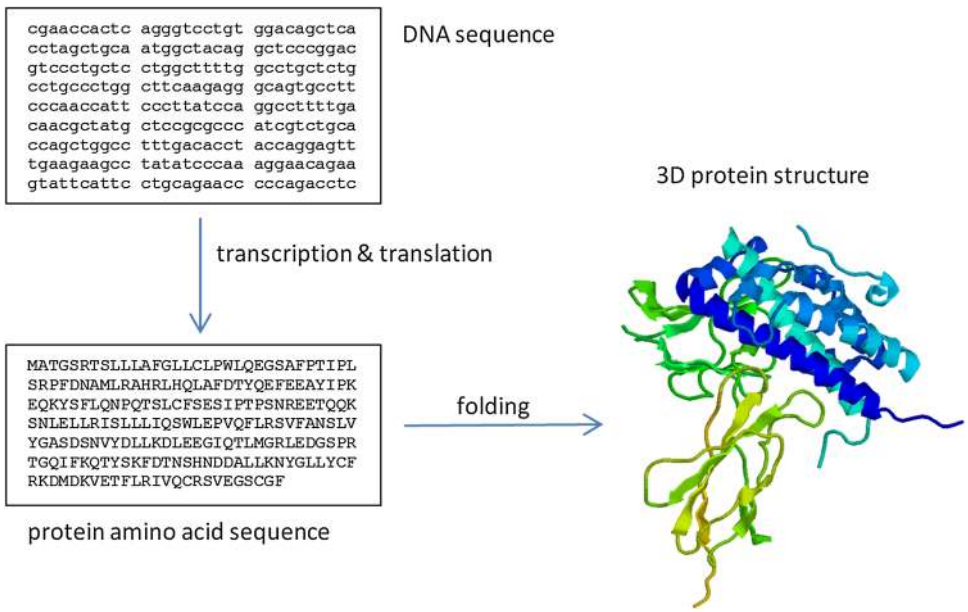

Fig. 1 Relationship between DNA sequence, protein amino acid sequence and 3D protein structure exemplified by a part of *GH1* gene responsible for encoding *Somatotropin* protein in *Homo sapiens* (UniProtKB:P01241). The 3D structure of *Human growth hormone* (PDB:1A22) visualized with the use of RasMol [64]

1.2 Spark computational platform and cloud computing

With the growing *volume* of biological data describing various aspects of living forms, increasing demand for fast data analysis (*velocity*), and a *variety* of data formats, structural bioinformatics has been facing the Big Data challenge. This requires changing the way how we cope with the biological data and redevelopment of existing tools for data analysis and processing, thus directing us to Big Data ecosystems. Apache Hadoop with MapReduce processing model and Apache Spark are both popular big data frameworks. However, since the Spark allows for running most computations in memory, without the necessity to store intermediate results in a file system, it provides better performance for many applications [88]. Therefore, in order to meet the requirements for fast IDP predictions, we decided to use Apache Spark for the development of the efficient IDP meta-predictor described in the paper.

Apache Spark [88] is a platform for large-scale data processing and general-purpose computing. Spark was generally created to run computations on computer clusters allowing to distribute computations over the cluster nodes. Spark divides the computational job into a number of tasks and executes the tasks on Worker nodes of the cluster within processes called *executors*. In addition to performing tasks, executors are responsible for keeping data in memory or disk storage across tasks.

Operational data that must be processed on Spark cluster are delivered as *resilient distributed dataset* (RDD). RDD [87] is a fault-tolerant collection of elements that Spark uses to operate on in parallel. RDDs can be created by parallelizing an already existing collection through a *transformation* or by referencing a data set in an external storage system, such as Hadoop Distributed File System (HDFS). Both methods are used in our solution presented in the paper. Data delivered in the form of RDDs are divided into a number of partitions. Spark then adjusts the number of tasks sent for execution to the number of partitions the RDD collection is cut into. RDDs allow to store processing results in memory in any stage of data processing, e.g., between various transformations. Transformations allow to perform various operations on data and create new data sets (RDDs) from existing ones.

Spark can be deployed in several modes, including private clusters and clusters located on public clouds. The latter solution enables flexible scaling the Spark cluster on-demand in order to accommodate data growth, since the Cloud allows users to access and use configurable computing resources (e.g., servers, storage, applications, networks) as a service [49] without the necessity to build entire infrastructure that supports the resources on premises. The Spark cluster can be hosted on resources provisioned by a cloud provider. This allows to raise the computing power and was applied in our solution presented in the paper. We used Microsoft Azure cloud platform in order to quickly scale the system performing predictions of IDPs on large scale.

1.3 Related works

Accurate and efficient prediction of disordered regions in protein structures on the basis of pure amino acid sequence is one of the challenges in computational biology and structural genomics. Existing methods for IDP prediction rely on the analysis of various features of proteins, e.g., protein amino acid composition or physical-chemical characteristics of particular amino acids. Some of the methods are grounded in statistical observations, other use machine learning algorithms. For example, GlobPlot [40] is a simple prediction method that identifies regions of globularity and disorder within protein sequences on the basis of propensities of particular amino acids to be in globular or non-globular states. GlobPlot calculates a running sum of the propensity for amino acids to be in an ordered or disordered state, and uses the sum to classify regions of the given amino acid sequence to a particular class. IUPred [15] is also footed on the physical foundations and statistical observations of inter-residue interactions in proteins structures. To discriminate between ordered and disordered class, the IUPred uses statistical interaction potentials. On the other hand, DisEMBL [39] uses trained Artificial Neural Networks (ANNs) to predict protein disorders. There are three variants of the method: (1) *Coils* used to predict classic loops and coils as defined by DSSP [29], (2) *Hot loops* for prediction of flexible loops with a high degree of mobility determined by temperature factors (B-factors), (3) *Remark 465* that predicts missing coordinates in X-ray structures, as defined by REMARK465 records in the PDB files describing macromolecular structures of proteins. DISpro [10] uses evolutionary information in the form of protein profiles, predicted secondary structure and relative solvent accessibility, and ensembles of 1D-recursive neural networks. RONN [85] classifies disordered regions on the basis of observed homology between protein sequences. The homology is expressed in calculated alignment scores, while comparing given protein sequences with a series of amino acid sequences of known folding state (ordered, disordered, or a mixture of both). Obtained alignment scores are then used in the prediction process performed by suitably trained regional order neural network. SPINE-D [89] classifies residues into three classes, as structurally ordered, disordered, and semi-ordered. The method also

uses a trained artificial neural network. Several methods, including DISOPRED2 [78], Poodle-s [66], Poodle-l [21], PrDOS [26], and Spritz [73], use SVM-based classifiers to predict disordered regions on the basis of various features extracted from protein sequences. For example, Poodle-s, Poodle-l, and PrDOS perform prediction on the basis of position-specific scoring matrices (PSSMs) generated by PSI-BLAST [1], while Spritz uses amino acid frequencies in disordered regions. PSSMs with respect to physico-chemical properties of amino acids are also used in iPDA server [68] and its underlying DisPSSMP predictor.

Recent works in this area, including Xue et al. [83] and Kozlowski and Bujnicki [33], show that meta-prediction with the use of ensembles composed of subsets of described basic predictors may improve the quality of prediction results. The method presented in our paper also works on the basis of ensemble of basic predictors. However, so far, none of the prediction or meta-prediction methods was designed to perform large-scale identification of disordered regions and to deal with large volumes of protein sequences.

Meanwhile, there are real-life problems, in which massive parallelization of computations on Apache Hadoop or Spark, and the use of scalable environments, like the Cloud, brought significant improvements in performance of data processing and analysis. Big data challenge was observed and solved in various works devoted to intelligent transport and smart cities [11, 19, 42, 43, 74, 75, 84], water monitoring [12, 22, 90], social networks analysis [13, 14, 77], multimedia processing [72, 82], internet of things (IoT) [9], social media monitoring [50], Life sciences [3, 31, 32, 44, 58, 69] and disease data analysis [6, 45, 81], telecommunication [27], and finance [2], to mention just a few. Many hot issues in various sub-fields of bioinformatics were also solved with the use of Big Data ecosystems and Cloud computing, e.g., mapping next-generation sequence data to the human genome and other reference genomes, for use in a variety of biological analyzes including SNP discovery, genotyping and personal genomics [65], sequence analysis and assembly [17, 30, 34, 35, 47, 62], multiple alignments of DNA and RNA sequences [86, 91], codon analysis with local MapReduce aggregations [63], NGS data analysis [8], phylogeny [24, 48], proteomics [37], analysis of protein-ligand binding sites [23], and others. Regarding the analysis of 3D protein structures, it is worth mentioning several works, including Hazelhurst et al. [20] and Małysiak-Mrozek et al. [46] devoted to exploration of various atomic interactions within protein structures, works of Che-Lun Hung and Yaw-Ling Lin [25], and Mrozek et al. [51, 53, 55–57], devoted to comparison and alignment of 3D protein structures, and cloud-based system for 3D protein structure modeling presented in [54]. However, none of the mentioned works was focused on prediction of disordered regions.

1.4 Scope of this work

In this paper, we present an ensemble method for prediction of intrinsically disordered proteins implemented in our IDP meta-predictor (IDPP, Intrinsically Disordered Proteins meta-Predictor, Sect. 2.1). We also show the implementation of the method on the Spark cluster, called *Spark-IDPP* (Sect. 2.4). Spark-IDPP allows for large-scale prediction of intrinsically disordered proteins or intrinsically disordered regions of protein structures on the basis of amino acid sequences. The IDPP classifies disordered regions on the basis of consensus of votes cast by component, basic predictors (Sect. 2.2). In the paper, we examine four consensus modes in terms of the effectiveness of prediction showing that the use of ensemble methods leads to the increased effectiveness (Sect. 3.4). The method is implemented on the Apache Spark, hence Spark-IDPP, in order to provide significantly better performance for large volumes of sequence data that are processed and analyzed in protein bioinformatics. In this paper, we also present results of performance tests on scaling out the Spark cluster on the Microsoft Azure cloud and increasing the degree of parallelism with the intention of improving efficiency of performed predictions (Sect. 3.5).

2 Methods and implementation

In this section, we show architecture of the IDPP meta-predictor, foundations of the prediction method working on the basis of consensus, filtering method used for elimination of outliers, and details of the implementation of the IDPP meta-predictor on the Apache Spark.

2.1 IDPP meta-predictor architecture

IDPP meta-predictor performs prediction of intrinsically disordered proteins or disordered regions for the given protein amino acid sequences. General architecture of the IDPP meta-predictor is presented in Fig. 2. It consists of the following modules:

- Component Predictors Module (CPM),
- Consensus Module (CoM),
- Analysis Module (AM).

The Component Predictors Module (CPM) is responsible for prediction of disordered regions. Prediction can be performed for many protein sequences provided in the FASTA format [41] at the input of the CPM. A sample

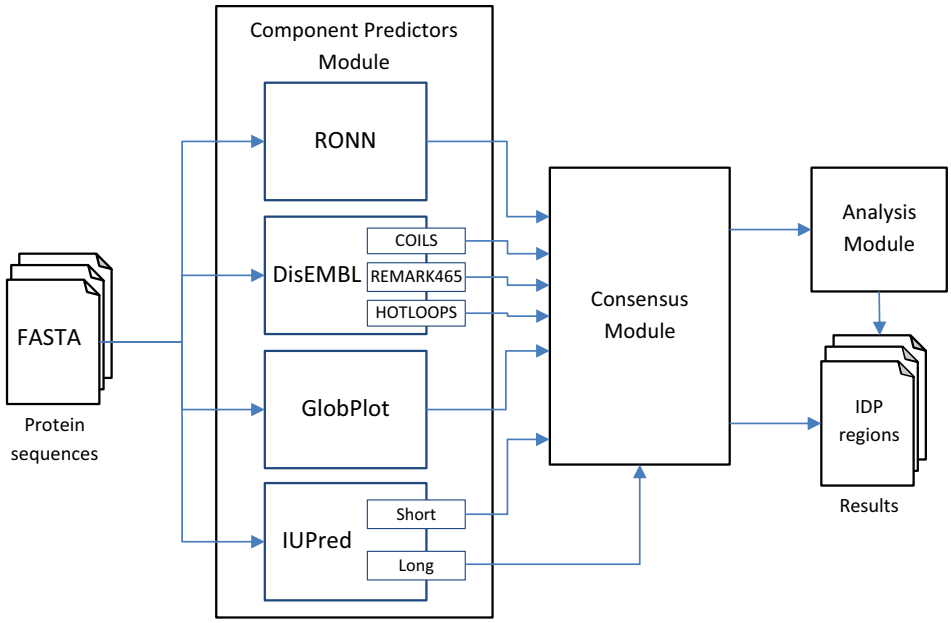

Fig. 2 Architecture of the IDPP meta-predictor

input sequence in the FASTA format is presented in Listing 1. First line of each entry in the FASTA file consists of descriptive information, e.g., identifiers of the protein sequence in various public repositories. Next lines consist of amino acid sequence of the protein.

```
1 >DisProt|DP00004|uniprot|P49913|unigene|Hs.51120|sp|CAMP_HUMAN
2 MKTQRNGHSLGRWSLVLLLLGLVMPLAIIAQVLSYKEAVLRAIDGINQR
3 SSDANLYRLLDLDPRPTMDGDPDTPKPVSFTVKETVCPRTTQQSPEDC
4 DFKKDGLVKRCMGTVTLNQARGSFDISCDKDNKRFALLGDFFRKSKEK
5 IGKEFKRIVQRIKDFLRNLVPRTES
```

Listing 1 A sample input sequence in the FASTA format.

The Component Predictors Module contains several basic predictors for intrinsically disordered regions. Each of the component predictors accepts an amino acid sequence at the input and performs independent predictions of intrinsically disorder regions on the basis of the input sequence. The CPM module of the IDPP meta-predictor implements the following basic prediction algorithms:

- RONN [85];
- DisEMBL in three variants: Coils, REMARK 465, and Hotloops [39];
- IUPred short and IUPred long [15];
- GlobPlot [40].

Each of the component predictors generates its results containing the list of regions which are supposed to be disordered. These results contain the information on the probability that each amino acid belongs to the disordered region or not, binary classification of belonging to the disordered region (1—belongs, 0—does not belong), and lists of disordered regions expressed as ranges of amino acid positions.

The Consensus Module (CoM) aggregates results from component predictors on the basis of consensus. There are four consensus approaches (also called consensus modes) implemented and tested in the Consensus Module: two operating on binary classification from component predictors and two operating on float values of the returned probability. All will be described in details in Sect. 2.2. In the next phase, the CoM performs filtering of the consensus results with the use of fuzzy smoothing filter in order to remove probable outliers. After filtering out single disordered positions, the Consensus Module finally classifies particular amino acids in the protein sequence as belonging to a disorder region or not, on the basis of aggregated probabilities or binary classification results. The final classification is performed for the given cutoff threshold obtained experimentally.

The Analysis Module (AM) allows to assess the quality of classification and to find potential cutoff thresholds for component predictors. The AM allows to analyze results of the prediction process and compare them to a ground truth (actual disordered regions that were discovered

experimentally or collected manually from literature, stored in the DisProt database [59]). Results of the analyzes are provided in the form of True Positive, True Negative, False Positive, and False Negative rates.

Results of the prediction process are returned as text files, saved on hard disk drive, or on the standard output of the IDPP meta-predictor. For example, a result of a prediction of disordered regions for a single amino acid sequence of protein *Cyclin-dependent kinase inhibitor 1B* from *Homo sapiens* (accession number: P46527, entry name: CDN1B_HUMAN in the UniProtKB/Swiss-Prot database) is shown in Listing 2. First line of the result contains the information that allows to identify the protein. The second line contains the information on disordered regions identified in the sequence, expressed as ranges of amino acid positions.

```
1 >DisProt|DP00018|uniprot|P46527|unigene|Hs.238990|sp|CDN1B_HUMAN
2 #22–34 #96–108
```

Listing 2 A result of prediction of disordered regions for a single amino acid sequence.

2.2 Reaching consensus

IDPP meta-predictor consists of several component predictors that classify amino acids as belonging to a disordered region or not. These predictors work on the basis of various algorithms, and they have various prediction effectiveness. However, we assumed that the aggregation of their votes may give improvements in the prediction effectiveness. In our IDPP meta-predictor, we applied the following four consensus modes—two operating on results of binary classification from seven component predictors (RONN, IUPred in two variants, GlobProt, and DisEMBL in three variants) and two operating on float values of the returned probabilities of belonging to a disordered region (as presented in Fig. 3):

- Simple Binary,
- Weighted Binary,
- Simple Float,
- Weighted Float.

The Simple Binary (SB) consensus mode makes use of only binary scores returned by all component predictors (0/1 classification), and votes of all component predictors are equally important while aggregating component decisions (equal weights). In this consensus mode, the IDPP meta-predictor evaluates votes of component predictors and aggregates ranks for each *i*th residue in the protein amino acid sequence according to the following formula:

$$preScore_i^{IDPP-Bin} = \frac{\sum_{j\in J} w_j Score_{i,j}^{Bin}}{\sum_{j\in J} w_j}, \quad (1)$$

where J is the set of basic component predictors (RONN, GlobProt, IUPred short, IUPred long, DisEMBL Hotloops, DisEMBL Remark465, DisEMBL Coils), $Score_{i,j}^{Bin}$ is the binary score returned by the jth component predictor for the ith residue (1—belongs to a disordered region, 0—does not belong), and $w_j = 1.0$ are weights of importance of particular component predictors ($w_j = 0.0$, if we want to eliminate the jth predictor from voting).

Likewise the Simple Binary consensus mode, the Weighted Binary (WB) consensus mode uses only binary scores returned by all component predictors (0/1 classification), but votes of all component predictors have different weights while aggregating component decisions. Weights of importance for particular component predictors (w_j in Eq. 1) are calculated according to the following formula:

$$S_{W,j} = \frac{S}{S_{max}} = \frac{W_{disorder}TP - W_{order}FP + W_{order}TN - W_{disorder}FN}{W_{disorder}(TN + FN) + W_{order}(TN + FP)}, \quad (2)$$

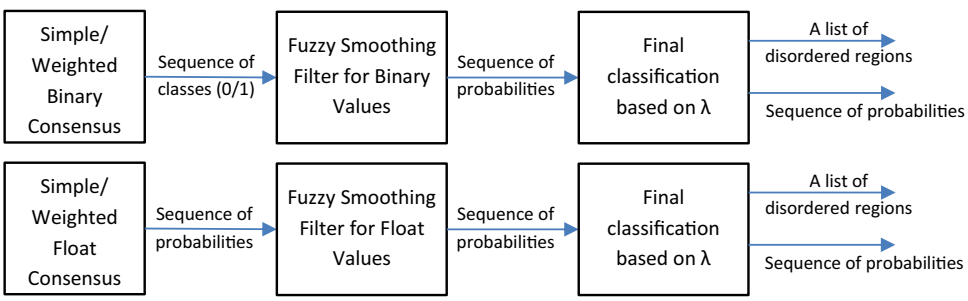

Fig. 3 Overview of the information flow in the Consensus Module in various consensus modes

where: $W_{disorder}$ equals the fraction of disordered residues, W_{order} equals the fraction of ordered residues, TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. The $S_{W,j}$ weighted score rewards a correct disorder prediction higher than a correct order prediction [28]. This is done to avoid over-prediction of an ordered state due the fact that ordered regions are more common in known proteins. Values of the $S_{W,j}$ weighted score for particular component predictors of the IDPP were obtained experimentally (see Table 3 in Sect. 3.4).

In both binary consensus modes, particular amino acids in the protein sequence (i) are then pre-classified, whether they belong to a disordered region or not:

$$preClass_i^{IDPP-Bin} = \begin{cases} 1 & \text{if} \quad preScore_i^{IDPP-Bin} \geq \lambda_0 \\ 0 & \text{if} \quad preScore_i^{IDPP-Bin} < \lambda_0 \end{cases}, \tag{3}$$

where $preScore_i^{IDPP-Bin}$ is the score aggregated in one of the binary consensus modes (Eq. 1), and λ_0 is a qualification threshold determined experimentally to 0.5.

The Simple Float (SF) consensus mode makes use of probabilities returned by all component predictors for each residue of the amino acid chain. Votes of all component predictors are equally important while aggregating component decisions (equal weights, like in the Simple Binary consensus mode). In this consensus mode, the IDPP meta-predictor evaluates votes of component predictors and aggregates probabilities for each ith residue in the protein amino acid sequence according to the following formula:

$$preScore_i^{IDPP-Flo} = \frac{\sum_{j \in J} w_j Prob_{i,j}}{\sum_{j \in J} w_j}, \tag{4}$$

where J is the set of basic component predictors, $Prob_{i,j}$ is the probability returned by the jth component predictor for the ith residue that the residue belongs to a disordered region, and $w_j = 1.0$ are weights of importance of particular component predictors ($w_j = 0.0$, if we want to eliminate the jth predictor from voting).

Similarly, the Weighted Float (WF) consensus mode works on the basis of probabilities that a residue belongs to a disordered region, calculated by component predictors. However, while aggregating component decisions, votes of all component predictors are weighted according to the $S_{W,j}$ score (Eq. 2).

In contrast, to binary consensus modes, results of float consensus modes are not pre-classified, but fuzzy filtering in the next phase is performed directly on aggregated probabilities $preScore_i^{IDPP-Flo}$.

2.3 Filtering outliers

Disordered regions are usually formed by many successive residues in the protein chain, rather than single amino acids. Therefore, single amino acids or short segments predicted as disordered regions separated by short segments of ordered regions should be eliminated from the final result. To this purpose, we implemented fuzzy smoothing filter that discards such small disordered "islands". The filter runs through the sequence of classes (0/1) or probabilities each residue in protein sequence was assigned in the Consensus Module, and replaces each entry by a new probability value. The new probability value for the ith residue is calculated on the basis of classes $preClass_i^{IDPP-Bin}$ (for binary consensus modes):

$$Prob_i^{IDPP} = \frac{\sum_{k=i-2}^{i+2} \mu(l) \cdot preClass_k^{IDPP-Bin}}{\sum_{l=1}^{5} \mu(l)}, \quad l = k - i + 3 \tag{5}$$

or scores $preScore_i^{IDPP-Flo}$ (for float consensus modes):

$$Prob_i^{IDPP} = \frac{\sum_{k=i-2}^{i+2} \mu(l) \cdot preScore_k^{IDPP-Flo}}{\sum_{l=1}^{5} \mu(l)}, \quad l = k - i + 3 \tag{6}$$

for neighboring residues. The pattern of neighboring residues is called the *window*, and slides residue by residue, over the entire sequence of amino acids. In Eqs. 5 and 6 the window size is equal to 5. For the ith residue, the 5-residue window consists of the residues located at absolute positions $i-2, i-1, i, i+1, i+2$ in the protein amino acid chain. The k iterates through neighboring residues (points to absolute positions) in the sliding window with respect to the ith residue being processed, l transforms the absolute

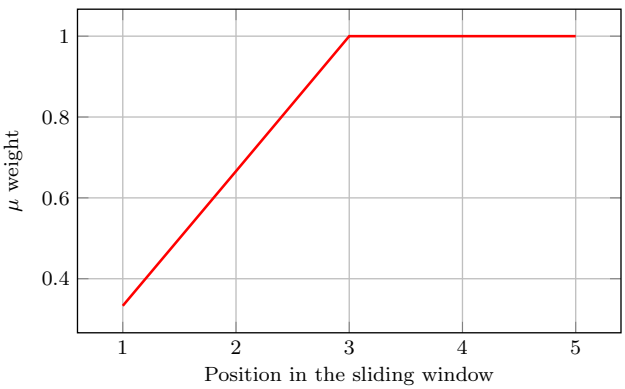

Fig. 4 A fuzzy set assigning weights for disorders identified at particular positions of the sliding window of the used filter

position of the kth residue to the position in the sliding window ($l = 1..5$), and $\mu(l)$ is the weight for the lth residue in the 5-residue sliding window. The $\mu(l)$ weight is calculated as it is presented in Fig. 4. Such a fuzzy set defined by a membership function allows assigning lower weights for the first two residues in the sliding window. In consequence, this weakens the influence of the first two elements (elements preceding the current ith element in a sequence), which is important when switching between classes. If the following elements $i + 1$ and $i + 2$ are of the same class as the ith element, the influence of the preceding two elements $i - 1$ and $i - 2$, especially if one of them is of a different IDP class than the ith element, decreases with the distance from the current element. In other words, this fuzzy set models the uncertainty at the border of regions belonging to two different classes.

With the position-specific weight function as defined in Fig. 4, we can notice that the following condition always holds:

$$\sum_{l=1}^{5} \mu(l) = 4.0. \tag{7}$$

For positions in the sliding window that go beyond the beginning or end of the protein sequence, we assume that the IDPP meta-predictor assigns the class of 1, i.e., disordered, since residues located at each end of proteins sequences are, on average, more frequently disordered than residues located in the middle of the protein chain.

Values of the new probability calculated for each residue in the protein sequence are then compared to the experimentally determined cutoff threshold λ (default value of λ is 0.5). The ith residue is classified as disordered, if the probability is greater or equal to the λ threshold:

$$Class_i^{IDPP} = \begin{cases} 1 & \text{if} \quad Prob_i^{IDPP} \geq \lambda \\ 0 & \text{if} \quad Prob_i^{IDPP} < \lambda \end{cases}. \tag{8}$$

After this step, each residue is finally classified as ordered or disordered, and the Consensus Module returns a list of disordered regions accompanied by the sequence of disorder probabilities, for each protein sequence provided on the input of the IDPP meta-predictor. An example of the filtering applied on a sequence of classes produced in binary classification for a sample protein from the DisProt database is shown in Fig. 5.

2.4 IDPP on the Apache Spark

Spark-IDP meta-predictor (Spark-IDPP) performs predictions of disordered regions on large scale by parallelizing computations on the Apache Spark cluster located on Microsoft Azure cloud. The Spark-IDPP was created and tested on Spark 1.6.1 working on Linux platform within Microsoft Azure HDInsight service HDI 3.4.

2.4.1 Architecture of the Spark-IDPP

Overview of the execution of the IDP prediction on Spark with the use of the Spark-IDPP meta-predictor is presented in Fig. 6. Input data for the Spark-IDPP—many protein sequences in the FASTA format—should be available on the HDFS in the storage space of the Azure HDInsight service. The input data, usually distributed in many FASTA files, are retrieved from the HDFS and loaded into RDD collection located on the Master node of the Spark cluster with the use of `JavaPairRDD<..>wholeTextFiles(String path, int minPartitions)` function. The `wholeTextFiles` function allows to read folders containing many small text files. Each file is read as a single record and is added to the RDD collection as a key-value pair, where a key is a path to the file, and the value is the content of the file. The RDD collection consisting of such key-value pairs is then divided into partitions. The number of partitions is passed as a second argument of the `wholeTextFiles` function. If the argument is not set, the Apache Spark environment sets it automatically by itself, but the number of partitions cannot exceed the number of input files with protein sequences.

Spark creates a task for each data partition in the RDD collection and places it into the FIFO queue. Tasks are then sent to the multi-core Spark worker nodes and executed by *executors*. If the number of partitions is greater than the

```
>DisProt|DP00130|uniprot|P22531|unigene|Hs.568518|sp|SPR2E_HUMAN
MSYQQQQCKQPCQPPPVCPTPKCPEPCPPPKCPEPCPPPKCPQPCPPQQCQQKCPPVTPSPPCQPKCPPKSK

Before filtering:
11111111111111001110000000001000000000100000001011111111111111111111111

After filtering:
11111111111111111111000000000000000000000000000011111111111111111111111

Prediction result:
>DisProt|DP00130|uniprot|P22531|unigene|Hs.568518|sp|SPR2E_HUMAN
#1-20 #48-72
```

Fig. 5 Result of filtering applied on a sequence of classes produced in binary classification for a sample protein from the DisProt database

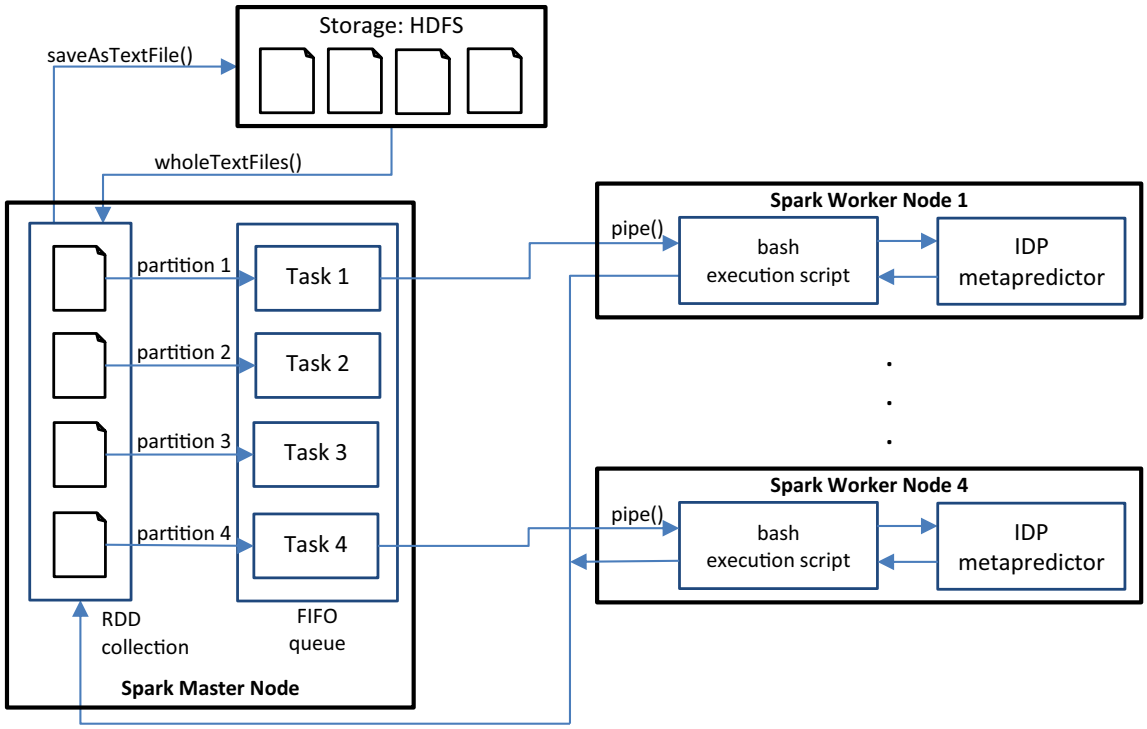

Fig. 6 Spark-IDPP—execution of the IDP prediction on Spark

number of Spark worker nodes, after completing calculations, an idle Worker node takes the next enqueued task from the FIFO queue.

Input data retrieved from the HDFS are passed as data streams to IDPP meta-predictor processes running on Spark worker nodes. This is done by using `staticJavaRDD<String>pipe(command)` transformation. The *pipe* transformation is one of the mechanisms for communication between running processes, and it allows to exchange data between these processes. Each RDD partition is piped through shell commands, i.e., bash execution script. Elements of the RDD partition, i.e., protein amino acid sequences, are printed on standard input of the executed process, and results are written on the standard output of the process and returned as a RDD of character string.

The Bash script executed by the *pipe* transformation saves the data from the standard input to a local text file on a Worker node. It is necessary, since component predictors in the CPM require paths to text files specified as execution arguments. After saving data in a text file, a Worker node executes the IDPP meta-predictor, which returns results on the standard output. These results are saved in the RDD collection on the Master node of the Spark cluster, and then, stored as text files on the HDFS with the use of `saveAsTextFile` action (Fig. 6).

2.4.2 Implementation of the IDPP on Spark

Details of the execution of the IDP prediction on Spark with the Spark-IDPP meta-predictor are shown in Listings 3 and 4. Within the Spark *driver program*, the Spark-IDPP application runs the `main` method presented in Listing 3. The method reads the execution parameters (line 2) and loads the Spark-IDPP configuration (lines 3–4). The configuration is loaded from the configuration file indicated by the file path extracted from one of the execution parameters (line 3). The configuration file consists of many information that control the execution of the IDP prediction on the Spark, including the path to the bash IDPP execution script, the path to the folder with input files, the path to the folder where the output files should be saved, the number of partitions of the Spark to run calculations on. These data set appropriate attributes (respectively, `command`, `inputFolderPath`, `outputFolderPath`, `numberOfPartitions`) of the `SparkTools` class, which is a part of the Spark driver program. Then, the `main` method creates the `JavaSparkContext` class object (line 5) and starts calculations by executing the `run` method of the `SparkTools` class object (line 6), passing the Spark context with the Spark configuration as an argument.

Implementation of the `run` method is shown in Listing 4. At the beginning, after reading the start time of the job (lines 2–3), the method loads the amino acid sequences by reading

```
1 public static void main(String[] args) {
2   Params params = new Params(args);
3   SparkTools sparkTools = new SparkTools(params.getConfigPath());
4   SparkConf sparkConf = sparkTools.getSparkConf();
5   JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);
6   sparkTools.run(sparkContext);
7   sparkContext.close();
8 }
```

Listing 3 Main function of the Spark driver program for the Spark-IDPP meta-predictor.

the files located in a folder indicated by the path specified in the `inputFolderPath` attribute (line 5). The files are loaded with the use of the `wholeTextFiles` function (method of the `JavaSparkContext` class object). The function accepts two input parameters, the first one is the path to the folder with files, the second one is optional, but recommended, and it suggests the number of partitions the input data should be split into. Execution of the function creates a key-value RDD collection (line 5), `JavaPairRDD<String, String>` class object, where the key is the name of the file, and the value contains the content of the file. In the next step, the `pipe` transformation is invoked on the RDD collection (line 6). The argument of the transformation is a bash IDPP execution script (stored in the `command` attribute) that runs the IDPP meta-predictor. This produces the output RDD collection with results. These results are stored in the specified location (indicated by the `outputFolderPath` attribute) by invocation of the `saveAsTextFile` action (executed on the output RDD collection, line 7).

3 Experimental results

Parallel, Spark-based implementation of the IDPP meta-predictor (Spark-IDPP) was extensively tested in order to verify its effectiveness and performance. The main goal of the experiments was to address the following questions:

- What is the quality of predictions provided by the designed meta-predictor for various consensus modes?
- What is the efficiency of massive predictions performed on the Apache Spark with the use of the designed meta-predictor?
- How does the performance depend on the number of input files passed for execution?
- How scalable is the Spark-IDPP?
- What is the efficiency of Spark-IDPP compared to the local, sequential version of the predictor?

3.1 Runtime environment

The Spark cluster used for most of the performed experiments was established on the Microsoft Azure public cloud as the HDInsight service (HDI 3.4) hosted on D13v2-sized virtual machines (VMs) with Linux operating system.

```
1 public void run(JavaSparkContext sparkContext) {
2   SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
      HH-mm-ss");
3   String jobStartDate = dateFormat.format(new Date(this.
      startJobTime));
4
5   JavaPairRDD<String, String> inputFiles = sparkContext.
      wholeTextFiles(this.inputFolderPath, this.numberOfPartitions);
6   JavaRDD<String> output = inputFiles.pipe(this.command);
7   output.saveAsTextFile(this.outputFolderPath + jobStartDate);
8 }
```

Listing 4 Execution of the prediction process in the *run* method.

Virtual machines in the Dv2-series are intended for applications that demand faster CPUs, better temporary storage performance, or have higher memory demands. The whole series is based on the 2.4 GHz Intel Xeon E5-2673 v3 (Haswell) processor, and with the Intel Turbo Boost Technology 2.0, can go up to 3.1 GHz. D13v2-sized virtual machines used by Worker nodes of the Spark cluster had 8 virtual CPUs, 56GB RAM, and 400GB of temporal storage space on SSD drives. Such a prepared Spark cluster was used to run parallel procedures of the Spark-IDPP meta-predictor and carry out assumed performance experiments.

3.2 Data set

During our experiments we used two databases of protein sequences—one for testing effectiveness of the Spark-IDPP meta-predictor and other basic predictors, and another data set for testing efficiency of the Spark-IDPP. While testing effectiveness of the Spark-IDPP meta-predictor, we used the DisProt data set [59, 67]. The data set that we used contained 1539 disordered regions located in 694 proteins. These regions were discovered with the use of experimental methods, and evidences of disorder were manually collected from the literature. The DisProt data set provided a "ground truth" while testing effectiveness of all primary predictors and the Spark-IDPP and allowed us to calculate weighted scores S_W for basic predictors in all consensus modes.

The second database, UniProtKB [71], was used to test efficiency of the Spark-IDPP. This database provides millions of protein amino acid sequences and was used due to its large size. In our experiments we used various subsets of the database.

3.3 A course of experiments

Our experiments involved testing the quality of predictions of disordered regions and efficiency of the proposed meta-predictor on the Spark cluster. The quality of predictions was tested for four consensus modes in order to select the best one. Then, we tested the performance of the meta-predictor working on the Spark cluster for the selected consensus mode. While testing performance, we first experimentally established how to divide data into data chunks and what number and size of data chunks should be used in order to minimize the execution time for the current cluster configuration (the number of nodes). The most efficient division of data into data chunks was used in the following experiments. Afterward, we changed the cluster size for the same amount of input data in order to verify scalability of the solution. And finally, we verified efficiency and speedup achieved by the system built on the 32-node Spark cluster (cluster size was constant) for the growing volume of data.

3.4 Effectiveness of the Spark-IDPP meta-predictor

Effectiveness of the prediction performed with the use of the Spark-IDPP meta-predictor was verified in a series of tests and compared to the effectiveness of component predictors. All component predictors and the Spark-IDPP meta-predictor were tested on the DisProt data set ver. 6.02. This data set contains protein sequences together with the information on the experimentally identified disordered regions. We had to exclude three protein sequences from the data set due to errors generated by particular component predictors: DP00642 and DP00651 (errors generated by the DisEMBL), and DP00195 (not fulfilling the conditions of RONN).

3.4.1 Effectiveness measures

Predictors were examined in terms of sensitivity (recall, TPR), accuracy (ACC), specificity (SPC), precision (PREC), F1-score, and Matthews correlation coefficient (MCC) [60], on the basis of the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) retrieved from the *confusion matrix* obtained for each of the tested predictors. The confusion matrix, also known as a *contingency table* or an *error matrix,* contains information about actual and predicted values (Table 1). TP is the number of disordered protein residues correctly classified, TN is the number of ordered protein residues correctly classified, FP is the number of ordered protein residues incorrectly classified as disordered, FN is the number of disordered protein residues incorrectly classified as ordered.

One of the measures that we used in our evaluation is accuracy (ACC), which is the proportion of properly predicted disordered and ordered residues (both true positives and true negatives) among the total number of cases examined:

$$ACC = \frac{(TP + TN)}{(TP + TN + FP + FN)}. \tag{9}$$

Accuracy measures how well the prediction model correctly identifies particular protein residues as ordered and disordered (the closer to 1.0, the better).

Sensitivity (Recall, or True Positive Rate, TPR) measures the proportion of true positives (TP) that are correctly identified for all the cases that are positive in the diagnostic test:

$$TPR = \frac{TP}{(TP + FN)}. \tag{10}$$

Sensitivity can be treated as the measure that examines the probability of detection of true positive cases (correctly

Table 1 Confusion matrix for performance evaluation of created IDPP meta-predictor

		Predicted	
		Positive	Negative
Real	Positive	TP (actual disordered protein residues that were correctly classified as disordered)	FN (actual disordered protein residues that were incorrectly classified as ordered)
	Negative	FP (actual ordered protein residues that were incorrectly classified as disordered)	TN (actual ordered protein residues that were correctly classified as ordered)

predicted disordered protein residues). With the higher sensitivity (closer to 1.0), fewer real positive cases are misclassified.

Specificity is the proportion of cases that are true negative (correctly classified ordered protein residues) for all of the cases that are assessed as negatives:

$$SPEC = \frac{TN}{(TN + FP)}. \tag{11}$$

With the higher specificity (closer to 1.0) fewer real disordered residues (positive cases) are labelled as ordered, so this ratio can be regarded as the percentage of ordered protein residues (negative cases) correctly predicted as belonging to the ordered region of the protein.

Precision (PREC, or Positive Predictive Value, PPV) is the proportion of positive cases that are correctly identified (TP) for all cases that are classified as positive:

$$PREC = \frac{TP}{(TP + FP)}. \tag{12}$$

High values of the precision (closer to 1.0) indicate better performance of the classification model.

Two additional measures of the quality of the prediction were also used: F-measure and the Matthews correlation coefficient (MCC). F-measure is the weighted harmonic mean that can be used to find the balance between the precision (PREC) and the recall (sensitivity, TPR). It can be calculated according to the following formula:

$$F - measure = 2 \cdot \frac{(PREC \cdot TPR)}{(PREC + TPR)}. \tag{13}$$

The Matthews correlation coefficient (MCC) was introduced by B.W. Matthews in 1975. It takes values from the range $\langle -1; +1 \rangle$, where $+1$ denotes perfect prediction. The MCC can be calculated according to the following formula:

$$MCC = \frac{(TP \cdot TN - FP \cdot FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{14}$$

Additionally, we used the ROC curves (Receiver Operating Characteristic curves) and the Area Under the Curve (AUC) in order to assess the quality of our IDPP meta-predictor and other IDP predictors. ROC curves and the AUC measure are frequently used in prediction of intrinsically disordered proteins to evaluate performance of prediction (classification) models. The ROC curve graphically illustrates the relative trade-off between true positive rate (TPR) indicating benefits and false positive rate (FPR), which indicates the cost, at various settings of the discrimination threshold. AUC can be considered as the measure indicating the accuracy of the predictive model, where 1 is the best possible value and 0.5 is the equivalent to a random prediction.

3.4.2 Evaluation of primary, component predictors

At the beginning, we evaluated primary, component predictors that are known from literature and were also implemented in our IDPP meta-predictor. Results of the evaluation of component predictors are presented in Table 2. As can be observed from Table 2, the RONN predictor achieved the best sensitivity $TPR = 0.695$ with

Table 2 Effectiveness of component predictors (the best achieved result is marked in bold)

Name	TPR	ACC	SPC	PREC	F1-score	MCC
DisEMBL coils	0.507	0.688	0.744	0.379	0.434	0.229
DisEMBL hotloops	0.487	0.683	0.744	0.370	0.420	0.212
DisEMBL REMARK465	0.364	**0.756**	**0.878**	**0.478**	0.413	0.267
IUPred short	0.619	0.727	0.761	0.444	0.517	0.343
IUPred long	0.633	0.727	0.756	0.445	**0.523**	**0.350**
GlobPlot	0.330	0.700	0.814	0.354	0.342	0.148
RONN	**0.695**	0.673	0.667	0.391	0.501	0.311

Table 3 Values of the S_W score and the AUC measure for particular component predictors (ordered by AUC)

Predictor	S_W	AUC
IUPred long	0.390	0.746
IUPred short	0.380	0.740
RONN	0.362	0.721
DisEMBL COILS	0.251	0.639
DisEMBL REMARK465	0.242	0.626
DisEMBL HOTLOOPS	0.231	0.636
GlobPlot	0.145	0.572

the lowest specificity $SPC = 0.667$. On the other hand, out of all component predictors the DisEMBL REMARK465 predictor is characterized by the best specificity $SPC = 0.878$, accuracy $ACC = 0.756$, and the precision $PREC = 0.478$, but also the lowest sensitivity $TPR = 0.364$. The results for IUPred predictors, especially IUPred Long, are quite high for each of the measures (though not the best), and the predictor is characterized by the best $F1score = 0.523$ and the Matthews correlation coefficient $MCC = 0.350$.

On the basis of results of the effectiveness tests, for each component predictor, we plotted the ROC curve and calculated the AUC measure and S_W weighted score (according to Eq. 2). ROC curves for particular component predictors are presented in Fig. 8. Values of the S_W score and the AUC measure for particular component predictors are shown in Table 3.

As can be observed by analyzing the results presented in Table 3 the IUPred, especially IUPred Long, and RONN predictors achieved the highest values of the AUC in our tests. They also had the highest values of the S_W coefficient, and therefore, they contribute with the highest weighted scores to the final decision made in the Consensus Module of our IDPP meta-predictor working in Weighted Binary and Weighted Float modes.

3.4.3 Evaluation of the IDPP meta-predictor

After the calculation of the S_W weighted scores, we examined the effectiveness of the proposed IDPP meta-predictor working in all four consensus modes. All tests were conducted with the use of the same DisProt data set ver. 6.02 as for testing component predictors. On the basis of the confusion matrices obtained for the IDPP meta-predictor working in each of the four consensus modes, we calculated values of the effectiveness measures (Table 4).

Results of the effectiveness tests presented in Table 4 show that IDPP meta-predictor working in the Simple Binary consensus mode (IDPP-SB) is characterized by the worst prediction quality. The value of AUC calculated for the IDPP-SB was 0.572, which reflects that the predictor is close to a random predictor, for which the AUC = 0.5. The IDPP meta-predictor working in the remaining three consensus modes achieved significantly better prediction quality. Results are similar, though the Weighted Float consensus mode led to the highest AUC = 0.752, and the Weighted Binary consensus mode enabled to obtain the best MCC = 0.355. Obtained results are slightly better (AUC, MCC) or close to results achieved by particular component predictors presented in Tables 2 and 3. The Simple Float-based IDPP meta-predictor (IDPP-SF), which uses regular average while striving for the consensus, is slightly worse. It reached the AUC of 0.742 and the MCC of 0.350. The Weighted Binary-based and the Weighted Float-based IDPP meta-predictors (IDPP-WB and IDPP-WF) use the weighted mean while seeking the consensus, where weights are S_W coefficients calculated for particular component predictor (Table 3). The IDPP-WB reached the AUC of 0.743 and the MCC of 0.355, and the IDPP-WF reached the AUC of 0.752 and the MCC of 0.351. This shows that the application of the S_W weighted score is beneficial, especially for IDPP predictors working on the basis of Binary consensus (compare AUC and MCC for SB and WB modes in Table 4). ROC curves for the IDPP meta-predictor working in all four consensus modes are presented in Fig. 7. Comparison of ROC curves for the proposed IDPP meta-predictor and component predictors is shown in Fig. 8. For the clarity of presentation we show the ROC curve for the IDPP meta-predictor working in the Weighted Float consensus mode (IDPP-WF).

3.4.4 IDPP meta-predictor with and without fuzzy filtering

We also checked how the fuzzy filtering influences results of the prediction with the use of the IDPP meta-predictor. In Fig. 9 we show ROC curves for the proposed IDPP

Table 4 Effectiveness of the IDPP predictor working in each of the four consensus modes

Consensus mode	TPR	ACC	SPC	PREC	F1-score	MCC	AUC
Simple binary (SB)	0.166	0.775	0.963	0.579	0.258	0.218	0.572
Weighted binary (WB)	0.663	0.720	0.737	0.438	0.527	0.355	0.743
Simple float (SF)	0.676	0.712	0.723	0.429	0.525	0.350	0.742
Weighted float (WF)	0.708	0.700	0.697	0.419	0.526	0.351	0.752

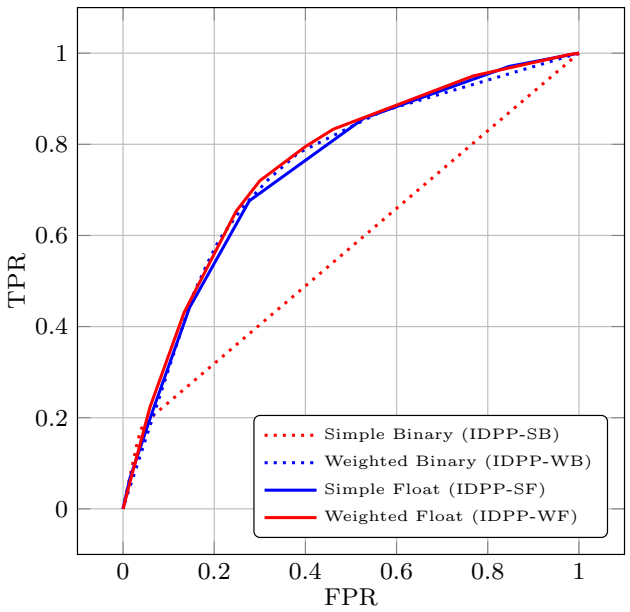

Fig. 7 ROC curves for the proposed IDPP meta-predictor working in various consensus modes: Simple Binary, Weighted Binary, Simple Float, Weighted Float

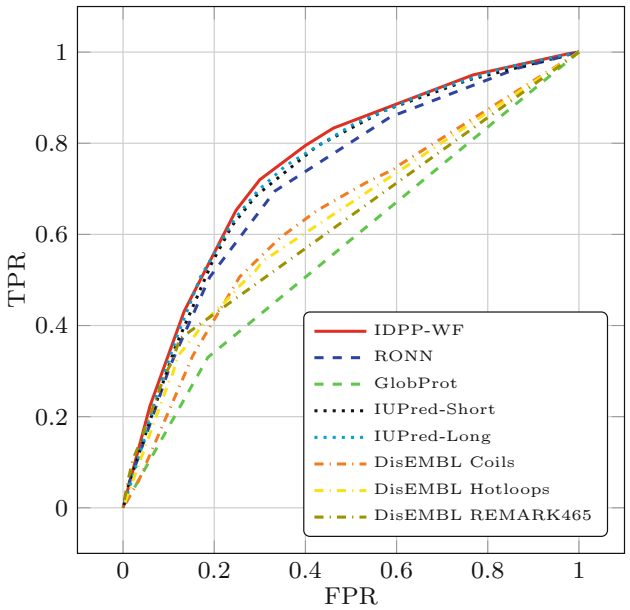

Fig. 8 ROC curves for the proposed IDPP meta-predictor (IDPP-WF) and other component predictors

meta-predictor working with and without fuzzy filtering in all consensus modes. Figure 9 clearly shows that the fuzzy filtering brings improvement in the prediction quality in all consensus modes of the IDPP meta-predictor. This improvement can be measured by the change in the AUC presented in Table 5. Results presented in Table 5 show that relative improvement after using the fuzzy filtering ranges between 2 and 4% in all implemented consensus modes.

3.5 Performance of IDPP-based prediction on the cloud

We conducted a broad series of tests in order to verify performance of the proposed Spark-IDPP method. In those tests we investigated the execution time and speedup achieved for various configurations of the Spark cluster and various data load.

3.5.1 Execution time versus the number and the size of files (data chunks)

In this series of tests we wanted to verify how the performance of the prediction process carried out with the Spark-IDPP meta-predictor depends on the number and the size of input data files (input chunks) stored on the HDFS. These tests allowed to experimentally select the appropriate files-to-nodes ratio for various sizes of input data set used in other performance experiments. In these tests we used a 32 MB part of the UniProtKB/Swiss-Prot database, which was divided into a number of files of various sizes: 16 partial files of size up to 2000 kB, 32 partial files of size up to 1000 kB, 64 partial files of size up to 500 kB, and 320 partial files of size up to 100 kB. Experiments were carried out on the Spark cluster with 16 worker nodes. Results of the experiments are presented in Table 6.

As can be observed from Table 6, for the 32 MB database, the Spark-IDPP achieved the best performance for many (320) files of size up to 100 kB. The execution time was 10,146 s. For the same 32 MB input data set, the configuration with 16 files of size up to 2000 kB and the files-to-nodes ratio equal to 1 was almost twice less efficient (19,703 s) than the configuration with 320 files of size up to 100 kB, for which the number of files is 20 times the number of nodes (files-to-nodes ratio is 20). We can also observe that with the growing size of input chunks and with the decreasing files-to-nodes ratio, the Spark platform automatically decreased the number of data partitions. For higher files-to-nodes ratio (20) the Spark created 16 partitions, and for low files-to-nodes ratio (1) the Spark created only 11 partitions, leaving 5 nodes of the computation cluster idle. For this reason, the configuration with 16 input chunks of size up to 2000 kB and the lowest files-to-nodes ratio turned to be the slowest. This was caused by the fact that not whole capabilities of the Spark cluster were utilized and the workload was unbalanced. Only 11 nodes of the computation cluster were used, and 5 of them had to perform prediction for 2 data chunks, while the other nodes were idle after processing a single data chunk. Results of these tests allowed us to confirm that the best execution

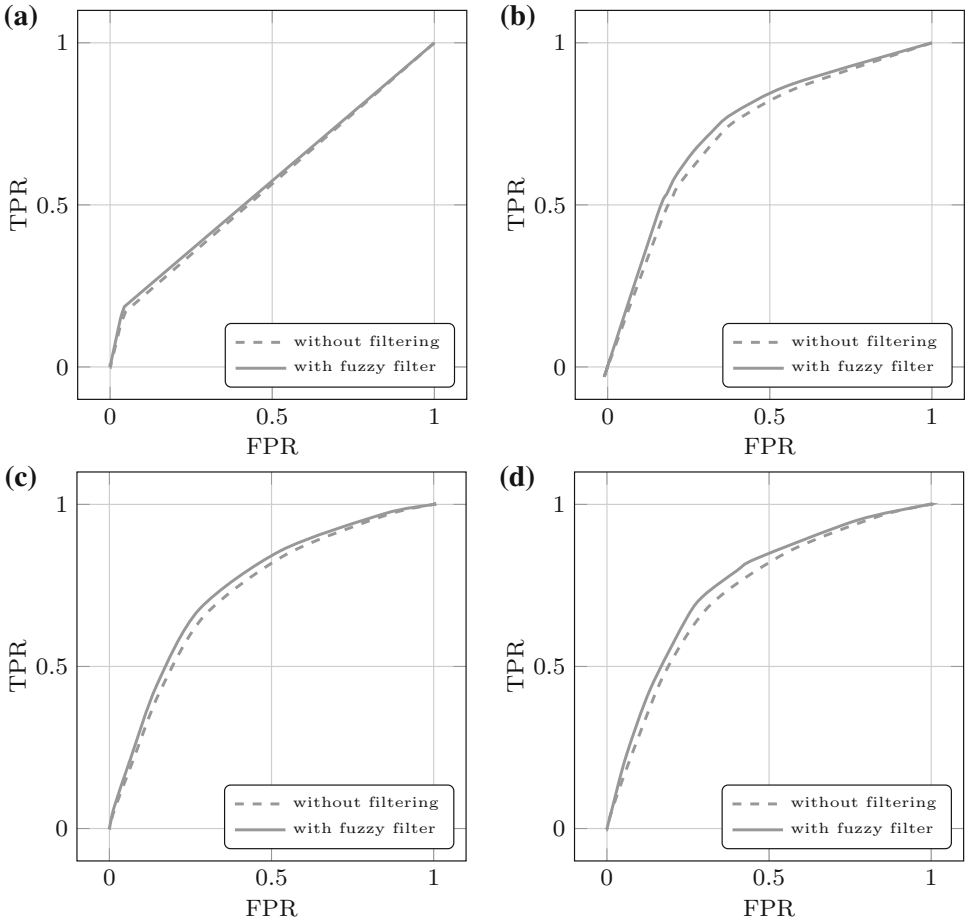

Fig. 9 ROC curves for the proposed IDPP meta-predictor working with and without filtering in various consensus modes: Simple Binary (**a**), Weighted Binary (**b**), Simple Float (**c**), Weighted Float (**d**)

Table 5 Area under the ROC curve (AUC) calculated for the IDPP predictor working with and without the fuzzy filtering in each of the four consensus modes

Consensus mode	AUC without filtering	AUC with filtering	Relative improvement (%)
Simple binary (SB)	0.562	0.572	2
Weighted binary (WB)	0.722	0.743	3
Simple float (SF)	0.721	0.742	3
Weighted float (WF)	0.726	0.752	4

Table 6 Spark-IDPP total execution time for various sizes of input files (input chunks)

File size	#Files	Files-to-nodes ratio	Execution time (s)	#Partitions
100 kB	320	20	**10,146**	16
500 kB	64	4	12,377	15
1000 kB	32	2	14,719	13
2000 kB	16	1	19,703	11

times are achieved when the number of files is much larger than the number of nodes.

3.5.2 Execution time versus cluster size

In the next series of performance tests we examined scalability of the Spark-IDPP by changing the number of working nodes of the Spark cluster. During the experiments, the IDPP meta-predictor was launched on the Spark cluster with 1, 4, 8, 16, and 32 nodes. The size of the whole input data set was constant during the course of experiments and equal to 256.1 MB (the whole Swiss-Prot data set). The data set was divided into smaller chunks in such a way that allowed keeping the files-to-nodes ratio at the

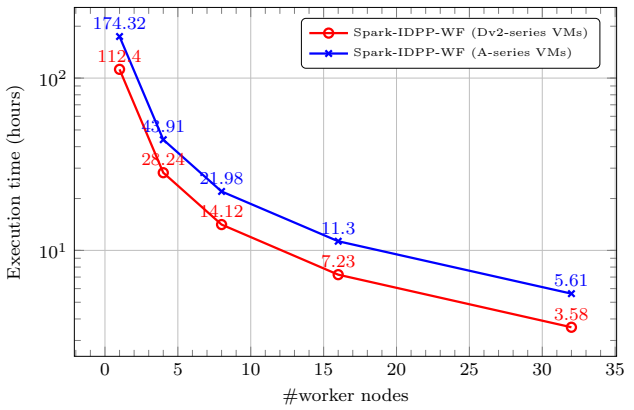

Fig. 10 Dependency between the prediction time (plotted on a logarithmic scale) and the number of cluster nodes for 256.1MB Swiss-Prot data set. Computations performed on the Spark cluster established on A-series and Dv2-series virtual machines

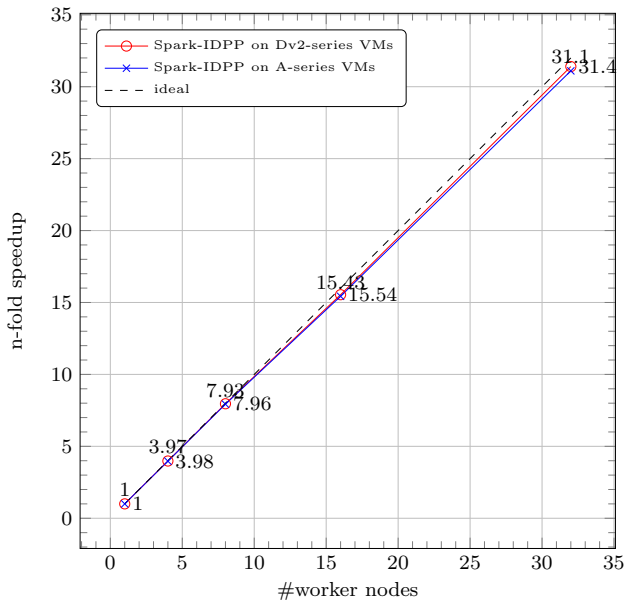

Fig. 11 n-fold speedup for IDP prediction performed on the Spark cluster for various size of the cluster

level of 20, e.g., 640 files up to 410 kB each for the 32-node Spark cluster. The results of these tests are presented in Fig. 10.

The prediction time decreases proportionally with the growing number of Spark worker nodes. On the Spark cluster with only one worker node the prediction took more than 112 h (on Dv2-series VMs). The IDP prediction with the Spark-IDPP executed on the 32-node cluster took less than 4 h on Dv2-series VMs and less than 6 h on A-series VMs. This gave almost ideal n-fold speedup when scaling out the cluster from one to 32 worker nodes on the Azure cloud. The speedup curves are presented in Fig. 11. The n-fold speedup was calculated according to the following equation:

$$S_d = \frac{T_1}{T_d}, \tag{15}$$

where d is the number of nodes of the Spark cluster, T_d is the execution time obtained while performing computations on the d-node cluster, and T_1 is the execution time obtained while performing computations on the 1-node cluster.

3.5.3 Performance for growing volume of data

We also tested the performance of the Spark-IDPP for the growing volume of protein amino acid sequences. We wanted to verify the gain resulting from using the Spark cluster in the IDP prediction with respect to desktop version of the meta-predictor (Desktop-IDPP). The Desktop-IDPP was tested on the workstation PC with CPU Core i7 4700MQ 2.4GHz (4 cores, 8 threads), RAM 16GB, storage HDD 1TB, working under control of the Microsoft Windows 7 64-bit operating system. Spark-IDPP was tested on the 32-node Spark cluster (Dv2-series VMs) located on the Microsoft Azure cloud. Results of the performance tests for both versions of the IDPP predictors are presented in Fig. 12. For predictions performed with the Spark-IDPP data sets were divided into smaller data chunks (files) in such a way that allowed keeping the files-to-nodes ratio at the level of 20. The number of files depended on the whole data set size and the number of protein sequences in the input data set (Table 7).

Results of performance tests presented in Fig. 12 show that the execution time for the Desktop-IDPP predictor grows very quickly with the size of the input data set—for 256.1 MB of input data the prediction took more than 5 days. This shows how time-consuming the prediction process is. Prediction with the Spark-IDPP on 32-node Spark cluster took less than 4 h for the same data set. Significant differences in execution times for large data sets confirm that the use of the Spark cluster is all the more justified, the larger the data set we process. For larger data sets, like 256.1 MB, the reduction of time was significant, and even the necessity of creation of the Spark cluster on the Cloud (which usually takes around 20 min) was just a fraction of time taken by the desktop version of the IDP predictor. In Table 7 we can also observe that the number of data partitions for various sizes of the input data set was constant and equal to the number of nodes of the Spark cluster. This was possible by dividing the input data set into many (640) chunks and keeping the files-to-nodes ratio at relatively high level (20).

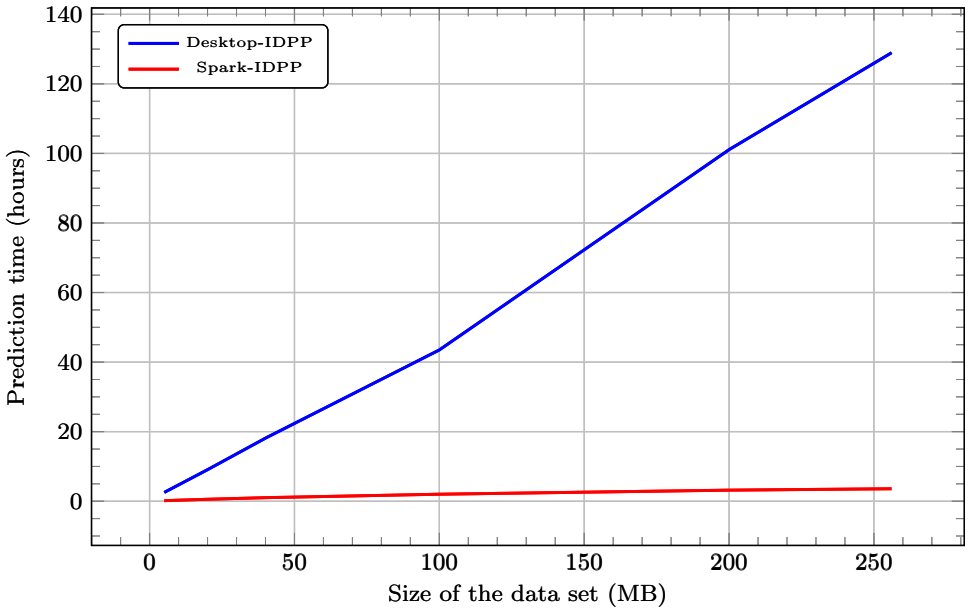

Fig. 12 Comparison of the execution time (prediction time) for the desktop version of the IDPP meta-predictor (Desktop-IDPP) and the Spark-IDPP working on the 32-node Spark cluster for varying size of the input data set

Table 7 Sizes of input data sets used in the prediction process performed on the 32-node Spark cluster, together with sizes of data chunks the data sets were divided into, and the number of partitions created by Spark during execution of the Spark-IDPP

Spark-IDPP on 32-node Spark cluster		
Data size (MB)	Chunk size (kB)	# Partitions
5.06	8	32
20.08	33	32
40.15	65	32
100.0	160	32
200.1	320	32
256.1	410	32

4 Discussion and conclusions

Prediction of disorder regions for protein amino acid sequences became an important branch of 3D protein structure prediction and modeling. The knowledge flowing from correctly resolved protein structures translates very well into drug design, or at least, recognition of molecular mechanisms underlying many civilization diseases. Disordered proteins constitute a wide range of molecules that play important roles in these molecular mechanisms. Since the number of protein amino acid sequences in world repositories grows exponentially, availability of efficient methods that are able to predict IDPs on highly scalable computer clusters is very important. This belief underlies our research.

Spark-IDPP responds to the needs very well by parellelizing computations on the Spark cluster that can be scaled on the Cloud on demand according to current requirements for computing power. Results of our performance tests show that we are able to perform predictions of disordered regions on large-scale and to handle growing amount of protein sequences by scaling the cluster horizontally and vertically. As also shown in this paper, IDP prediction on Spark clusters is the most beneficial, when it is performed for large data sets divided into smaller chunks (files). Best results were obtained when the number of data chunks was much larger than the number of data nodes—we established the *files-to-nodes ratio* at the level of 20. This caused that the computational capabilities of the Spark cluster were fully utilized, as the number of Spark data partitions was equal to the number of nodes. Only then we were able to achieve almost linear, above 31-fold, speedup on 32-node Spark cluster, and to significantly reduce the execution time.

The Spark-IDPP was developed for Spark clusters hosted in local data centers or in the Cloud. However, low entry barrier, a huge storage space, wide compute and flexible scaling capabilities of the Cloud made it an attractive alternative to local compute infrastructures kept on premises. With the wide, horizontal and vertical, scaling capabilities the Cloud enabled us to create the Spark cluster that is able to respond to current needs of IDP prediction and appropriately accommodate the growth of protein data in public repositories. The use of public cloud platforms, like Microsoft Azure or Amazon Web Services, simplifies many tasks related to the creation and maintenance of the Spark cluster. Firstly, within these platforms the Spark is provided as a service, i.e., it can be easily created and configured on-demand, when needed, and removed after performing required computations. This ease in

maintaining Spark clusters clearly distinguishes the solution from local Spark installations and clusters created in IaaS clouds. Although, this comes as a cost of broader configurability that clusters created in local data centers or IaaS clouds can possess. Secondly, once created on the Cloud platform, the Spark cluster can be dynamically scaled out by adding more cluster nodes or scaled down by releasing unnecessary compute resources. Thirdly, cloud platforms usually provide a specialized fleet of virtual machines that have differentiated compute capabilities (various sizes), optimized for different tasks, e.g., for compute-intensive or memory-intensive calculations. As a result, it is easy to scale up by using more powerful virtual machines. We started testing the Spark-IDPP on the Spark cluster created on A-series virtual machines. However, we quickly scaled up the solution to Dv2-series virtual machines, since they provided higher compute capabilities (better CPUs and more memory). This allowed us to further accelerate the IDP prediction for large data sets (see Fig. 10). The necessity of paying for the Spark cluster performing IDP predictions must be mentioned as a disadvantage of using cloud platforms. However, on the other hand, users do not have to cover the costs of maintenance of the whole hardware infrastructure kept on premises.

Results of our experiments on the quality of predictions prove that the proposed method is able to achieve higher prediction effectiveness than primary predictors. With the Spark-IDPP working in the Weighted Float consensus mode (Spark-IDPP-WF) we were able to improve the quality of predictions compared to basic predictors. The quality improvement was not so significant as in MetaDisorder reported in [33] and PONDR-FIT reported in [83], since we were not able to implement all component methods in the Spark-IDPP (e.g., some of them are not available as program packages). Nevertheless, our experiments confirmed that meta-prediction with the use of many component predictors, which operate on various features extracted from amino acid sequences and characteristics of particular amino acids, may increase the prediction quality.

The most important unique feature of the Spark-IDPP is its capability to work with large amounts of protein sequence data and to provide prediction results fast, adequately to the size of the Spark cluster. In such a way, it addresses the *volume* characteristics of the Big Data challenge. To the best of our knowledge, this is the first method that addresses this feature for prediction of IDPs. Among other unique features of the Spark-IDPP, it is worth mentioning its four consensus modes for combining results of basic predictors and fuzzy filtering method. Three out of four consensus modes allow to predict disordered regions with reasonable quality confirmed by the majority of the effectiveness measures that were calculated. The best results, in terms of the AUC and the MCC, were obtained for the Spark-IDPP working in the Weighted Float consensus mode (Spark-IDPP-WF), where we used weighted score proposed in [28] to make decisions on the classification to ordered/disordered classes. The same weighted score was used in the MetaDisorder method [33], but we calculated our own values of weights for used component predictors and for the DisProt data set. The quality of predictions returned by the Spark-IDPP working in the Weighted Binary and the Simple Float consensus modes was only slightly worse in terms of the AUC and the MCC, but even better in terms of accuracy and precision. Only the Simple Binary consensus mode did not bring satisfactory results. Its prediction capabilities turned out to be slightly better than random guessing. In addition, the fuzzy filtering method that we used allowed us to efficiently eliminate short segments outlying from neighboring ones, resulting in smoothing final outcome of the prediction.

Future works will cover further development of the IDPP meta-predictor and the Spark-IDPP. We plan to increase the number of component IDP predictors in the IDPP meta-predictor with the intention of improving the quality of predictions. An interesting option would also be to re-implement the code of the Spark-IDPP in such a way that each prediction performed by a component predictor is executed as a separate Spark transformation. Then, consensus could be achieved in another dedicated transformation on separate RDD collections generated by each of the component predictors. It is difficult to estimate the influence of such an implementation on the performance of the whole meta-prediction, but it would make the Spark-IDPP more extensible toward new prediction methods. We believe that the Spark-IDPP meta-predictor can be a valuable tool for the whole field of protein structure modeling and fold recognition. It complements the collection of existing tools by providing better predictive capabilities with significantly higher performance, which can be adapted to the current needs and amount of input data. This makes it an important alternative for desktop software tools, for which scalability is very limited or sometimes impossible to implement. Spark-IDPP brings a great promise in prediction of IDP showing that advances in computational solutions keep in pace with progress in large-scale data gaining in bioinformatics.

Acknowledgements This work was supported by Microsoft Research within Microsoft Azure for Research Award Grant and by Statutory Research funds of Institute of Informatics, Silesian University of Technology, Gliwice, Poland (Grant No. BK-213/RAu2/2018). Further development of the methods presented in the paper will be carried out by Cloud4Proteins non-profit, scientific group at the Silesian University of Technology, Gliwice, Poland (http://www.zti.aei.polsl.pl/w3/dmrozek/science/cloud4proteins.htm).

References

1. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. **25**(17), 3389–3402 (1997). https://doi.org/10.1093/nar/25.17.3389
2. Bai, C., Dhavale, D., Sarkis, J.: Complex investment decisions using rough set and fuzzy c-means: an example of investment in green supply chains. Eur. J. Oper. Res. **248**(2), 507–521 (2016)
3. Youssef, B.B.: A parallel cellular automata algorithm for the deterministic simulation of 3-D multicellular tissue growth. Clust. Comput. **18**(4), 1561–1579 (2015). https://doi.org/10.1007/s10586-015-0455-7
4. Benson, D.A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Sayers, E.W.: GenBank. Nucleic Acids Res. **45**(D1), D37–D42 (2017). https://doi.org/10.1093/nar/gkw1070
5. Berman, H.: The protein data bank. Nucleic Acids Res. **28**, 235–242 (2000)
6. Bo, Y.: The data clustering based dynamic risk identification of biological immune system: mechanism, method and simulation. Clust. Comput. https://doi.org/10.1007/s10586-018-1960-2
7. Boutet, E., Lieberherr, D., Tognolli, M., Schneider, M., Bansal, P., Bridge, A.J., Poux, S., Bougueleret, L., Xenarios, I.: UniProtKB/Swiss-Prot, the Manually Annotated Section of the UniProt KnowledgeBase: How to Use the Entry View, pp. 23–54. Springer, New York (2016)
8. Ceri, S., Kaitoua, A., Masseroli, M., Pinoli, P., Venco, F.: Data management for heterogeneous genomic datasets. IEEE/ACM Trans. Comput. Biol. Bioinform. **14**(6), 1251–1264 (2017)
9. Chang, H., Mishra, N., Lin, C.: IoT Big-data centred knowledge granule analytic and cluster framework for BI applications: a case base analysis. PLoS ONE **10**, 1–23 (2015)
10. Cheng, J., Sweredoski, M.J., Baldi, P.: Accurate prediction of protein disordered regions by mining protein structure data. Data Min. Knowl. Discov. **11**(3), 213–222 (2005). https://doi.org/10.1007/s10618-005-0001-y
11. Cupek, R., Ziebinski, A., Huczala, L., Erdogan, H.: Agent-based manufacturing execution systems for short-series production scheduling. Comput. Ind. **82**, 245–258 (2016)
12. Czerniak, J.M., Dobrosielski, W.T., Apiecionek, L., Ewald, D.: Representation of a trend in OFN during fuzzy observance of the water level from the crisis control center. In: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 443–447 (2015)
13. Davis, G.B., Carley, K.M.: Clearing the fog: fuzzy, overlapping groups for social networks. Soc. Netw. **30**(3), 201–212 (2008)
14. De Maio, C., Fenza, G., Loia, V., Parente, M.: Time aware knowledge extraction for microblog summarization on Twitter. Inf. Fusion **28**, 60–74 (2016)
15. Dosztányi, Z., Csizmok, V., Tompa, P., Simon, I.: IUPred: web server for the prediction of intrinsically unstructured regions of proteins based on estimated energy content. Bioinformatics **21**(16), 3433–3434 (2005). https://doi.org/10.1093/bioinformatics/bti541
16. Dunker, A.K., Silman, I., Uversky, V.N., Sussman, J.L.: Function and structure of inherently disordered proteins. Curr. Opin. Struct. Biol. **18**(6), 756–764 (2008)
17. Feng, X., Grossman, R., Stein, L.: PeakRanger: a cloud-enabled peak caller for ChIP-seq data. BMC Bioinform. **12**(1), 1–11 (2011). https://doi.org/10.1186/1471-2105-12-139
18. Gu, J., Bourne, P.: Structural Bioinformatics (Methods of Biochemical Analysis), 2nd edn. Wiley-Blackwell, Hoboken (2009)
19. Guo, K., Zhang, R., Kuang, L.: TMR: towards an efficient semantic-based heterogeneous transportation media Big Data retrieval. Neurocomputing **181**, 122–131 (2016)
20. Hazelhurst, S.: PH2: an Hadoop-based framework for mining structural properties from the PDB database. In: Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, pp. 104–112 (2010)
21. Hirose, S., Shimizu, K., Kanai, S., Kuroda, Y., Noguchi, T.: POODLE-L: a two-level SVM prediction system for reliably predicting long disordered regions. Bioinformatics **23**(16), 2046–2053 (2007). https://doi.org/10.1093/bioinformatics/btm302
22. Hu, C., Ren, G., Liu, C., Li, M., Jie, W.: A Spark-based genetic algorithm for sensor placement in large scale drinking water distribution systems. Clust. Comput. **20**(2), 1089–1099 (2017). https://doi.org/10.1007/s10586-017-0838-z
23. Hung, C.L., Hua, G.J.: Cloud Computing for protein-ligand binding site comparison. Biomed. Res. Int. **2013**, 1–7 (2013).
24. Hung, C.L., Lin, C.Y.: Open reading frame phylogenetic analysis on the Cloud. Int. J. Genomics **2013**(614923), 1–9 (2013)
25. Hung, C.L., Lin, Y.L.: Implementation of a parallel protein structure alignment service on Cloud. Int. J. Genomics **439681**, 1–8 (2013)
26. Ishida, T., Kinoshita, K.: PrDOS: prediction of disordered protein regions from amino acid sequence. Nucleic Acids Res. **35**(suppl–2), W460–W464 (2007). https://doi.org/10.1093/nar/gkm363
27. Jensen, K., Nguyen, H.T., Do, T.V., Årnes, A.: A big data analytics approach to combat telecommunication vulnerabilities. Clust. Comput. **20**(3), 2363–2374 (2017). https://doi.org/10.1007/s10586-017-0811-x
28. Jin, Y., Dunbrack, R.: Assessment of disorder predictions in CASP6. Proteins **61**, 167–175 (2005)
29. Kabsch, W., Sander, C.: Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers **22**(12), 2577–2637 (1987)
30. Kelley, D.R., Schatz, M.C., Salzberg, S.L.: Quake: quality-aware detection and correction of sequencing errors. Genome Biol. **11**(11), 1–13 (2010). https://doi.org/10.1186/gb-2010-11-11-r116
31. Kim, S., Choi, J., Kim, Y.: Adaptive application-aware job scheduling optimization strategy in heterogeneous infrastructures. Clust. Comput. **19**(3), 1515–1526 (2016). https://doi.org/10.1007/s10586-016-0588-3
32. Kim, S., Kim, J.S., Hwang, S., Kim, Y.: Towards effective science cloud provisioning for a large-scale high-throughput computing. Clust. Comput. **17**(4), 1157–1169 (2014). https://doi.org/10.1007/s10586-014-0371-2
33. Kozlowski, L.P., Bujnicki, J.M.: MetaDisorder: a meta-server for the prediction of intrinsic disorder in proteins. BMC Bioinform. **13**(1), 111 (2012). https://doi.org/10.1186/1471-2105-13-111
34. Langmead, B., Hansen, K.D., Leek, J.T.: Cloud-scale RNA-sequencing differential expression analysis with Myrna. Genome Biol. **11**(8), 1–11 (2010). https://doi.org/10.1186/gb-2010-11-8-r83
35. Langmead, B., Schatz, M.C., Lin, J., Pop, M., Salzberg, S.L.: Searching for SNPs with Cloud computing. Genome Biol. **10**(11), 1–10 (2009). https://doi.org/10.1186/gb-2009-10-11-r134

36. Lesk, A.: Introduction to Protein Science: Architecture, Function, and Genomics, 2nd edn. Oxford University Press, Oxford (2010)
37. Lewis, S., Csordas, A., Killcoyne, S., Hermjakob, H.: Hydra: a scalable proteomic search engine which utilizes the Hadoop distributed computing framework. BMC Bioinform. **13**, 324 (2012)
38. Liao, V.C.C., Chen, M.S.: DFSP: a depth-first spelling algorithm for sequential pattern mining of biological sequences. Knowl. Inf. Syst. **38**(3), 623–639 (2014). https://doi.org/10.1007/s10115-012-0602-x
39. Linding, R., Jensen, L.J., Diella, F., Bork, P., Gibson, T.J., Russell, R.B.: Protein disorder prediction: implications for structural proteomics. Structure **11**(11), 1453–1459 (2003)
40. Linding, R., Russell, R.B., Neduva, V., Gibson, T.J.: GlobPlot: exploring protein sequences for globularity and disorder. Nucleic Acids Res. **31**(13), 3701–3708 (2003). https://doi.org/10.1093/nar/gkg519
41. Lipman, D., Pearson, W.: Rapid and sensitive protein similarity searches. Science **227**(4693), 1435–1441 (1985)
42. Lu, H., Sun, Z., Qu, W.: Big Data-driven based real-time traffic flow state identification and prediction. Discret. Dyn. Nat. Soc. **2015**, 1–11 (2015)
43. Lu, H., Sun, Z., Qu, W., Wang, L.: Real-time corrected traffic correlation model for traffic flow forecasting. Math. Probl. Eng. **2015**, 1–7 (2015)
44. Mahmud, S., Iqbal, R., Doctor, F.: Cloud enabled data analytics and visualization framework for health-shocks prediction. Future Gener. Comput. Syst. **65**, 169–181 (2016)
45. Małysiak-Mrozek, B., Stabla, M., Mrozek, D.: Soft and declarative fishing of information in Big Data lake. IEEE Trans. Fuzzy Syst. **26**(5), 2732-2747 (2018). https://doi.org/10.1109/TFUZZ.2018.2812157
46. Małysiak-Mrozek, B., Zur, K., Mrozek, D.: In-memory management system for 3D protein macromolecular structures. Curr. Proteomics **15**(3), 175–189 (2018). https://doi.org/10.2174/1570164615666180320151452
47. Matsunaga, A., Tsugawa, M., Fortes, J.: Cloudblast: Combining MapReduce and virtualization on distributed resources for bioinformatics applications. In: Proceedings of the IEEE Fourth International Conference eScience (ESCIENCE '08), pp. 222–229 (2008)
48. Matthews, S.J., Williams, T.L.: MrsRF: an efficient MapReduce algorithm for analyzing large collections of evolutionary trees. BMC Bioinform. **11**(1), 1–9 (2010). https://doi.org/10.1186/1471-2105-11-S1-S15
49. Mell, P., Grance, T.: The NIST definition of cloud computing. Special Publication 800-145. Accessed Oct 10 2017 (2011), http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
50. Meng, L., Tan, A., Wunsch, D.: Adaptive scaling of cluster boundaries for large-scale social media data clustering. IEEE Trans. Neural Netw. Learn. **27**(12), 2656–2669 (2015)
51. Mrozek, D.: High-Performance Computational Solutions in Protein Bioinformatics. Springer, New York (2014)
52. Mrozek, D., Brozek, M., Małysiak-Mrozek, B.: Parallel implementation of 3D protein structure similarity searches using a GPU and the CUDA. J. Mol. Model. **20**, 2067 (2014)
53. Mrozek, D., Daniłowicz, P., Małysiak-Mrozek, B.: HDInsight4PSi: boosting performance of 3D protein structure similarity searching with HDInsight clusters in Microsoft Azure cloud. Inf. Sci. **349–350**, 77–101 (2016)
54. Mrozek, D., Gosk, P., Małysiak-Mrozek, B.: Scaling Ab Initio predictions of 3D protein structures in Microsoft Azure cloud. J. Grid Comput. **13**, 561–585 (2015)
55. Mrozek, D., Kutyła, T., Małysiak-Mrozek, B.: Accelerating 3D protein structure similarity searching on Microsoft Azure Cloud with local replicas of macromolecular data. In: Wyrzykowski, R. (ed.) Parallel Processing and Applied Mathematics—PPAM 2015. Lecture Notes in Computer Science, vol. 9574, pp. 1–12. Springer, Heidelberg (2016)
56. Mrozek, D., Małysiak-Mrozek, B., Kłapciński, A.: Cloud4Psi: cloud computing for 3D protein structure similarity searching. Bioinformatics **30**(19), 2822–2825 (2014)
57. Mrozek, D., Suwała, M., Małysiak-Mrozek, B.: High-throughput and scalable protein function identification with Hadoop and Map-only pattern of the MapReduce processing model. J. Knowl. Inf. Syst. (2018), https://doi.org/10.1007/s10115-018-1245-3
58. Mrozek, D., Kasprowski, P., Małysiak-Mrozek, B., Kozielski, S.: Life sciences data analysis. Inf. Sci. **384**, 86–89 (2017)
59. Piovesan, D., Tabaro, F., Mičetić, I., Necci, M., Quaglia, F., Oldfield, C.J., Aspromonte, M.C., Davey, N.E., Davidović, R., Dosztányi, Z., Elofsson, A., Gasparini, A., Hatos, A., Kajava, A.V., Kalmar, L., Leonardi, E., Lazar, T., Macedo-Ribeiro, S., Macossay-Castillo, M., Meszaros, A., Minervini, G., Murvai, N., Pujols, J., Roche, D.B., Salladini, E., Schad, E., Schramm, A., Szabo, B., Tantos, A., Tonello, F., Tsirigos, K.D., Veljković, N., Ventura, S., Vranken, W., Warholm, P., Uversky, V.N., Dunker, A.K., Longhi, S., Tompa, P., Tosatto, S.C.: DisProt 7.0: a major update of the database of disordered proteins. Nucleic Acids Res. 45(D1), D219–D227 (2017), https://doi.org/10.1093/nar/gkw1056
60. Powers, D.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. Int. J. Mach. Learn. Technol. **2**, 37–63 (2011)
61. Prasad, D.V.V., Jaganathan, S.: Improving the performance of smith–waterman sequence algorithm on gpu using shared memory for biological protein sequences. Clust. Comput. (2018), https://doi.org/10.1007/s10586-018-2421-7
62. Qiu, X., Ekanayake, J., Beason, S., Gunarathne, T., Fox, G., Barga, R., Gannon, D.: Cloud technologies for bioinformatics applications. In: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, pp. 6:1–6:10. MTAGS '09, ACM, New York (2009), https://doi.org/10.1145/1646468.1646474
63. Radenski, A., Ehwerhemuepha, L.: Speeding-up codon analysis on the Cloud with local MapReduce aggregation. Inf. Sci. **263**, 175–185 (2014)
64. Sayle, R.: RasMol, Molecular graphics visualization tool. BiomolecularStructures Group, Glaxo Welcome Research & Development, Stevenage, Hartfordshire (May 2013), http://www.umass.edu/microbio/rasmol/
65. Schatz, M.C.: CloudBurst: highly sensitive read mapping with MapReduce. Bioinformatics **25**(11), 1363–1369 (2009)
66. Shimizu, K., Hirose, S., Noguchi, T.: POODLE-S: web application for predicting protein disorder by using physicochemical features and reduced amino acid set of a position-specific scoring matrix. Bioinformatics **23**(17), 2337–2338 (2007). https://doi.org/10.1093/bioinformatics/btm330
67. Sickmeier, M., Hamilton, J.A., LeGall, T., Vacic, V., Cortese, M.S., Tantos, A., Szabo, B., Tompa, P., Chen, J., Uversky, V.N., Obradovic, Z., Dunker, A.K.: DisProt: the database of disordered proteins. Nucleic Acids Res. **35**(suppl–1), D786–D793 (2007). https://doi.org/10.1093/nar/gkl893
68. Su, C.T., Chen, C.Y., Hsu, C.M.: iPDA: integrated protein disorder analyzer. Nucleic Acids Res. **35**(suppl–2), W465–W472 (2007). https://doi.org/10.1093/nar/gkm353
69. Teijeiro, D., Pardo, X.C., Penas, D.R., González, P., Banga, J.R., Doallo, R.: A cloud-based enhanced differential evolution algorithm for parameter estimation problems in computational systems biology. Clust. Comput. **20**(3), 1937–1950 (2017). https://doi.org/10.1007/s10586-017-0860-1

70. The 1000 Genomes Project Consortium: A global reference for human genetic variation. Nature **526**, 68–74 (2015)
71. The UniProt Consortium: Uniprot: the universal protein knowledgebase. Nucleic Acids Res. **45**(D1), D158–D169 (2017). https://doi.org/10.1093/nar/gkw1099
72. Tripathy, B.K., Mittal, D.: Hadoop based uncertain possibilistic kernelized c-means algorithms for image segmentation and a comparative analysis. Appl. Soft Comput. **46**, 886–923 (2016)
73. Vullo, A., Bortolami, O., Pollastri, G., Tosatto, S.C.E.: Spritz: a server for the prediction of intrinsically disordered regions in protein sequences using kernel machines. Nucleic Acids Res. **34**(suppl–2), W164–W168 (2006). https://doi.org/10.1093/nar/gkl166
74. Wang, C., Li, X., Zhou, X., Wang, A., Nedjah, N.: Soft computing in Big Data intelligent transportation systems. Appl. Soft Comput. **38**, 1099–1108 (2016)
75. Wang, H., Li, J., Hou, Z., Fang, R., Mei, W., Huang, J.: Research on parallelized real-time map matching algorithm for massive GPS data. Clust. Comput. **20**(2), 1123–1134 (2017). https://doi.org/10.1007/s10586-017-0869-5
76. Wang, S.: Improved swarm intelligence algorithm for protein folding prediction. Clust. Comput. (2018), https://doi.org/10.1007/s10586-018-2257-1
77. Wang, Z., Tu, L., Guo, Z., Yang, L.T., Huang, B.: Analysis of user behaviors by mining large network data sets. Future Gener. Comput. Syst. **37**, 429–437 (2014)
78. Ward, J.J., McGuffin, L.J., Bryson, K., Buxton, B.F., Jones, D.T.: The DISOPRED server for the prediction of protein disorder. Bioinformatics **20**(13), 2138–2139 (2004). https://doi.org/10.1093/bioinformatics/bth195
79. Wei, L., Xing, P., Shi, G., Ji, Z.L., Zou, Q.: Fast prediction of protein methylation sites using a sequence-based feature selection technique. IEEE/ACM Trans. Comput. Biol. Bioinform. **1**, 1 (2018). https://doi.org/10.1109/TCBB.2017.2670558
80. Wei, L., Xing, P., Su, R., Shi, G., Ma, Z.S., Zou, Q.: CPPred-RF: A sequence-based predictor for identifying cell-penetrating peptides and their uptake efficiency. J. Proteome Res. **16**(5), 2044–2053 (2017), pMID: 28436664
81. Xing, W., Jie, W., Tsoumakos, D., Ghanem, M.: A network approach for managing and processing big cancer data in clouds. Clust. Comput. **18**(3), 1285–1294 (2015). https://doi.org/10.1007/s10586-015-0456-6
82. Xu, Z., Mei, L., Hu, C., Liu, Y.: The big data analytics and applications of the surveillance system using video structured description technology. Clust. Comput. **19**(3), 1283–1292 (2016). https://doi.org/10.1007/s10586-016-0581-x
83. Xue, B., Dunbrack, R.L., Williams, R.W., Dunker, A.K., Uversky, V.N.: Pondr-fit: a meta-predictor of intrinsically disordered amino acids. Biochimica et Biophysica Acta (BBA)-Proteins Proteomics **1804**(4), 996–1010 (2010)
84. Yang, C.T., Chen, S.T., Yan, Y.Z.: The implementation of a cloud city traffic state assessment system using a novel big data architecture. Clust. Comput. **20**(2), 1101–1121 (2017). https://doi.org/10.1007/s10586-017-0846-z
85. Yang, Z.R., Thomson, R., McNeil, P., Esnouf, R.M.: RONN: the bio-basis function neural network technique applied to the detection of natively disordered regions in proteins. Bioinformatics **21**(16), 3369–3376 (2005). https://doi.org/10.1093/bioinformatics/bti534
86. Yu, L., Moretti, C., Thrasher, A., Emrich, S., Judd, K., Thain, D.: Harnessing parallelism in multicore clusters with the all-pairs, wavefront, and makeflow abstractions. Clust. Comput. **13**(3), 243–256 (2010). https://doi.org/10.1007/s10586-010-0134-7
87. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pp. 15–28. USENIX, San Jose, CA (2012), https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia
88. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I.: Apache spark: a unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016). https://doi.org/10.1145/2934664
89. Zhang, T., Faraggi, E., Li, Z., Zhou, Y.: Intrinsic Disorder and Semi-disorder Prediction by SPINE-D, pp. 159–174. Springer, New York, (2017), https://doi.org/10.1007/978-1-4939-6406-2_12
90. Zhong, Y., Zhang, L., Xing, S., Li, F., Wan, B.: The Big Data processing algorithm for water environment monitoring of the three gorges reservoir area. In: Abstract and Applied Analysis, vol. 2014 (2014)
91. Zou, Q., Hu, Q., Guo, M., Wang, G.: HAlign: fast multiple similar DNA/RNA sequence alignment based on the centre star strategy. Bioinformatics **31**(15), 2475–2481 (2015)

Bożena Małysiak-Mrozek received the M.Sc. and Ph.D. degrees, in computer science, from the Silesian University of Technology in Gliwice, Poland. She is an Assistant Professor in the Institute of Informatics at the Silesian University of Technology in Gliwice, Poland and also a member of the IBM Competence Center. Her scientific interests cover information systems, computational intelligence, bioinformatics, databases, big data, cloud computing, and soft computing methods. She is also a member of the Organizing Committee of the International Conference Beyond Databases, Architectures, and Structures and coeditor of fourteen books devoted to databases and data processing.

Tomasz Baron received the M.Sc. degree in computer science from the Silesian University of Technology in Gliwice, Poland in 2016. He currently works for Comarch S.A. company in Poland as software engineer. His interests cover cloud computing, front-end frameworks, and internet technologies.

Dariusz Mrozek is currently an Associate Professor and Head of Division of Theory of Informatics in Institute of Informatics at the Silesian University of Technology (SUT) in Gliwice, Poland. He received his Ph.D. degree from SUT in 2006. His research interests cover bioinformatics, information systems, parallel and Cloud computing, databases and Big data. He is now focused on the analysis of protein structures, functions and activities, and the use of novel computation techniques to get insights from biological data, including NGS and proteomics data. He is the author of 90+ papers published in conference proceedings and international journals, author of two books on the use of Big Data and high-performance computational solutions in protein bioinformatics published by Springer, co-editor of fourteen other books devoted to databases and data processing, and editor of two special issues in reputable scientific journals. He is a member of the IEEE Engineering in Medicine and Biology Society (EMBS), IEEE Systems, Man, and Cybernetics Society (SMCS), and IEEE Cloud Computing Community. Working in different research projects, he cooperated with qualified institutions, e.g. Imperial College of London (on the Chernobyl Tissue Bank), V P Komisarenko Institute of Endocrinology and Metabolism - Academy of Medical Sciences of the Ukraine, Medical Radiological Research Centre - Russian Academy of Medical Sciences, Helmholtz Zentrum Muenchen Deutsches Forschungszentrum Fuer Gesundheit und Umwelt Gmbh, Microsoft Research in the USA, Institute of Oncology in Gliwice, Poland, Medical University of Silesia, Katowice, Poland.