Published in final edited form as:
SoftwareX. 2020 ; 11: . doi:10.1016/j.softx.2020.100462.

iScore: An MPI supported software for ranking protein–protein docking models based on a random walk graph kernel and support vector machines

Nicolas Renaud[a,*], **Yong Jung**[b], **Vasant Honavar**[b,c], **Cunliang Geng**[a,d], **Alexandre M.J.J. Bonvin**[d], **Li C. Xue**[d,e]

[a]Netherlands eScience Center, Science Park 140, 1098 XG, Amsterdam, The Netherlands

[b]Bioinformatics & Genomics Graduate Program, Pennsylvania State University, University Park, PA 16802, USA

[c]College of Information Science & Technology, Pennsylvania State University, University Park, PA 16802, USA

[d]Bijvoet Centre for Biomolecular Research Faculty of Science - Chemistry, Utrecht University, Padualaan 8, 3584 CH Utrecht, The Netherlands

[e]Center for Molecular and Biomolecular Informatics, Radboudumc, Nijmegen, The Netherlands

Abstract

Computational docking is a promising tool to model three-dimensional (3D) structures of protein–protein complexes, which provides fundamental insights of protein functions in the cellular life. Singling out near-native models from the huge pool of generated docking models (referred to as the scoring problem) remains as a major challenge in computational docking. We recently published iScore, a novel graph kernel based scoring function. iScore ranks docking models based on their interface graph similarities to the training interface graph set. iScore uses a support vector machine approach with random-walk graph kernels to classify and rank protein–protein interfaces. Here, we present the software for iScore. The software provides executable scripts that fully automate the computational workflow. In addition, the creation and analysis of the interface graph can be distributed across different processes using Message Passing interface (MPI) and can be offloaded to GPUs thanks to dedicated CUDA kernels.

Keywords

Protein–protein docking; Scoring; Graph kernel functions; Support vector machines; MPI; Position-specific scoring matrix (PSSM)

[*]Corresponding author. n.renaud@esciencecenter.nl (N. Renaud), Li.Xue@radboudumc.nl (L.C. Xue).

Declaration of competing interest
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Code metadata

Current code version	0.2.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_302
Legal Code License	Apache-2.0
Code versioning system used	git
Software code languages, tools, and services used	python, MPI, CUDA.
Compilation requirements, operating environments & dependencies	numpy, libSVM, pdb2sql, h5x, pytest, biopython, mpi4py, numpy, scipy, h5py, matplotlib
If available Link to developer documentation/manual	https://iscoredoc.readthedocs.io/
Support email for questions	n.renaud@esciencecenter.nl

1. Motivation and significance

Interactions between proteins that lead to the formation of a three-dimensional (3D) complex is a crucial mechanism that underlies major biological activities ranging from immune defense system to enzyme catalysis. The 3D structure of such complexes provides fundamental insights on the protein recognition mechanism and protein functions [1,2]. To complement the labor-intensive experimental characterization of protein complexes computational docking approaches have been developed to predict their 3D structures [3,4]. The prediction of these structures using docking usually consists of two steps: First, the sampling step that consists of systematically (or randomly) rotating and translating individual protein components to generate typically tens of thousands of candidate interaction models; second, the scoring step that evaluates each of the models and selects the ones that are most likely to occur in nature.

The scoring problem has been a highly challenging task for decades. Many methods have been developed and can be largely grouped into five types: (1) Shape complementarity based methods, favoring models that maximize the surface matching with minimal shape penetration [5,6], (2) physical energy-based methods, which sum up all the pairwise interaction energies between interface atom/residue pairs and are widely used in most modern docking software [7-10], (3) statistical potential-based methods, which coverts the interaction frequency of interface atom/residue contact pairs observed in the experimentally solved protein complexes to potentials using the Boltzmann distribution [11,12], (4) machine learning based methods, which typically treat the scoring problem as a binary classification problem, predicting a docked model as near-native or not [13-15], and (5) co-evolution based methods, which score models based on the co-occurrence frequencies of residue pairs in sequence alignments [16]. Different scoring approaches are regularly benchmarked against each other during a community-wide challenge, the Critical Assessment of Prediction of Interactions (CAPRI) [17].

Recently, we introduced a novel graph kernel based machinelearning approach, called iScore [18]. iScore represents the interface of a protein complex as an interface graph, with the nodes being the interface residues and the edges connecting the residues in contact.

By comparing the graph similarity between the query graph and the training graphs, iScore predicts the likelihood how close the query graph is to a near-native model. We have demonstrated in our previous publication [18] that iScore competes with, or even outperforms various state-of-the-art approaches on two independent test sets: the new entries of Docking Benchmark 5.0 set [19] and the CAPRI score set [20]. Using only a small number of features, i.e. 1 evolutionary feature and 3 physics energy terms, iScore performs well compared with IRaPPA [15], the latest machine learning based scoring function, which exploits 91 features. This demonstrates the advantage of representing protein interfaces as graphs as compared to fixed-length feature vectors which discard information about the interaction topology.

We present here the software for iScore. As explained in the following, the software is easy to use thanks to dedicated executable scripts that completely automate the computational workflow. Furthermore, the software leverages distributed and heterogeneous computing technologies to accelerate the generation of the required data and its analysis.

2. Software description

The underlying method is described in details in [18] and only a summary is provided here to highlight the different components of the software. As described in [18], the interface of each protein–protein model is represented as a bipartite graph. Each node is labeled with a 20×1 feature vector from the position-specific scoring matrix (PSSM) of the corresponding residue. PSSMs [21] are widely used in bioinformatics and encode the log-likelihood ratio of the observed frequency of each amino acid type at a specific sequence location against a background frequency. They therefore represent the degrees of conservation for the protein's residues at their specific location in the sequence. The similarity between two graphs is evaluated via a random walk graph kernel (RWGK) approach [22]. The graph-pair similarity matrix is used as input of a support vector machine (SVM) to classify the interface graphs as near-native or non-near native. The decision value of the SVM classification is then combined with energetic terms to score each protein–protein interface (PPI). As for any supervised learning approach, the SVM model is first trained on a well defined dataset before being used to classify new conformations.

The software presented here provides a fully automated end-to-end training and testing platform for the ranking of PPIs following the iScore method. The software is organized as a Python module containing dedicated classes in charge of specific steps in the computational workflow. This workflow is fully automated through executable scripts that orchestrates the entire computation from processing PDB files of the docked models to obtaining the final score of each PPI.

2.1. Software architecture

The general architecture of iScore for training a model and scoring new conformations is represented in Fig. 1. The software only requires PDB files of the docking models contained in the dataset used for training or scoring. All other intermediary files are automatically generated and processed by the software. We provide details in the following different steps of the computational workflow and describe each module.

2.1.1. Generation of the PSSM files—PSSM files of docked conformations are generated by the pssm_gen() class using PDB files for input. The calculation of the PSSM relies on PSI-BLAST [23] using BLAST version 2.7.1+. The default parameters of the BLAST (for example, substitution matrix, gap costs, etc.) were set in agreement with the recommended values provided in the BLAST user guide [24]. Other parameters are provided in [18]. The pssm_gen() class also formats the PSSM files for further processing. The class outputs resulting PSSM files for each chain in the PDB files into a separate folder for further processing.

2.1.2. Generation of interface bipartite graphs—The graph generation is handled by the iscore_graph() function and relies heavily on our pdb2sql tool that allows manipulating PDB files using SQL queries [25]. The contact residues are identified by the interface module of pdb2sql using a default contact distance of 6.0 Å. The PSSM files generated in the previous step are then read and checked against the sequence of the protein. The PSSMs are subsequently mapped onto the interface graph. The resulting graph is then serialized using the pickle library in order to exploit the object hierarchy in the next computational steps. The class also provides options to export multiple graphs in a single HDF5 file for further visual inspection (see Fig. 3).

2.1.3. Random walk graph kernels—The function iscore_kernel() is responsible for the computation of pairwise random-walk graph kernels . For each pair of PPIs contained in the dataset, the corresponding graph files are first “unpickled” and loaded in memory. The different elements necessary to the computation of the RWGK are computed and assembled in the final kernel value (see [18] for details on the calculation). All kernel values are then stored in a dedicated pickle file.

2.1.4. Training the SVM model—The function iscore_svm() can then be used to train an SVM model from the previously computed RWGK. To this end, users must also provide the ground truth, i.e. the binary class 0/1 of each conformation contained in the training dataset. In iScore, we choose the binary labels 1 and 0 to describe near-native and non near-native conformations respectively. The function relies on the libSVM library [26] to train the SVM model. To facilitate the further exploitation of the trained SVM model, the SVM model are efficiently packed into a dedicated archive together with the graphs of all the conformations of the training set. This self contained archive contains all the information required to score new PPIs.

2.1.5. Scoring new PPIs—The workflow for ranking new PPIs is very similar to the one used to train the SVM model. Users only need to provide PDB files of new conformations and compute the corresponding PSSMs and interface graphs. However, the RWGK are now computed between the new conformations and the ones contained in the training set. This can easily be done using the training archive that contains all the relevant information. The resulting kernels are then used as input for the SVM model, and the SVM decision value is used as one component of the final scoring function. The other component of the scoring function is provided from energy terms that are directly computed from the

PDB files of the new conformations. The weight of each term in the final scoring function has been optimized using genetic algorithm as explained in [18].

2.2. Software functionalities

Beyond the individual modules described above, the software provides crucial functions that facilitate and accelerate the process of ranking PPIs.

2.2.1. Automation of the computational workflows—The software provides executable scripts that fully automate the workflows illustrated in Fig. 1. These scripts seamlessly orchestrate all the computational steps at the exception of the calculation of the PSSMs. This calculation can be rather demanding and therefore must be performed as a pre-processing step using the functions provided by the software.

The training of an SVM model from the PDB leads to the creation of a training archive and is fully controlled by the iScore.train.mpi executable script. This script reads the PDB and PSSM files, generates the interface graphs, computes all the pairwise RWGKs, trains the SVM model, and finally assembles the training archive. Similarly, the ranking of new conformations using a trained model can simply be achieved via a single command: iScore.predict.mpi. This script reads the PDB and PSSM files, generates all the interface graphs, computes the RWGK between the new conformations and the conformations included in the training set, and finally scores the new conformations.

To handle the potentially large computational cost associated with the calculation of the interface graphs and their pairwise RWGKs, these executable scripts support the distribution of the computational load across different MPI processes using mpi4py [27]. For the calculation of the graphs, the different conformations are distributed among the different MPI processes, and for the RWGK calculations all the pairwise combinations are distributed among the MPI processes. Simple performance benchmarks are reported in Fig. 2a showing good performance of the MPI distribution. However, note that training the SVM model and scoring new PPIs are done using a single process.

2.2.2. Calculation of the RWKG on GPUs—To accelerate the calculations of the graph kernels, we have developed simple GPGPU kernels using pyCUDA [28]. The utilization of these GPGPU routines can easily be turned ON or OFF through one optional keyword argument of the iscore_kernel() function. Fig. 2b shows the runtime of the CPU and GPU routines computing the RWGKs. As seen on this figure, a sizable improvement of performance can be obtained for large graphs. The software also provides solutions to tune the GPU kernels through the kernel tuner library [29]. This allows to automatically find the optimal configuration of the kernel in terms of blocs size, threads size, etc.

While the GPU routines might be interesting to process very large proteins or for other applications, we have exclusively used the CPU routines in our evaluation of the iScore software tool because our protein interface graphs contain fewer than a hundred nodes.

2.2.3. Visualization—As mentioned in Section 2.1.2, the interface graphs computed by iScore can be stored in a HDF5 file for further analysis. The resulting HDF5 file contains an entry for each graph where all the relevant data are stored. To facilitate the inspection and exploration of these interface graphs, we have developed a simple graphical interface based on the customizable HDF5 browser h5X [30]. This interface is accessible via the executable iScore.h5x. This interface allows to quickly generate all the data for visualization of a given graph connection using PyMol [31]. An example of representation is shown in Fig. 3. This figure shows a single PPI. All the contact residues are highlighted by a stick representation and bright color whereas the rest of the protein structure is represented by thin gray lines. Edges linking the contact residues represent the contact between the two chains and the label of each contact residue is displayed for clarity.

2.3. Code snippets

Beyond the executable scripts mentioned above, iScore can also be used as a Python module and could therefore be integrated in other applications. We illustrate here the use of iScore through a small code snippet.

```python
from iScore.graphrank.graph import GenGraph
from iScore.graphrank.kernel import Kernel

# generate the first graph
pdb = '1ATN.pdb'
pssm = {'A': '1ATN.A.pdb.pssm',
        'B':'1ATN.B.pdb.pssm '}
gen = GenGraph(pdb,pssm)
G1 = gen.get_graph()

# generate the second graph
pdb = '1IRA.pdb'
pssm = {'A':'1IRA.A.pdb.pssm',
        'B':'1IRA.B.pdb.pssm'}
gen = GenGraph(pdb,pssm)
G2 = gen.get_graph()

# compute the kernel
K = Kernel ()
K.compute_kron_mat(G1,G2)
K.compute_px (G1,G2)
K.compute_W0(G1,G2)
ker = K.compute_K(lamb=1.0, walk=4)
```

As we can see on this snippet, iScore provides the solution to generate graphs of given structures using PSSM as a node attribute and to compute the random walk graph kernel between the graphs. The graph generation is done via the GenGraph() class that only takes PDB and PSSM files as input. The random walk graph kernel of the two graphs can then be computed using the Kernel() class and its methods.

3. Illustrative examples

We present here the results on test cases extracted from previous CAPRI competitions. In order to score the conformations contained in the test cases, a representative training set containing 234 distinct PPIs was first assembled. Half of these conformations correspond to real experimental structures of complexes chosen from the Docking Benchmark 4 (DB4) [32]. The second half correspond to non-native docking models generated using the HADDOCK docking software from entries of the DB4. Their i-RMSD values are larger than 10 Å. The resulting dataset is publicly available [33].

The trained model was used to score and rank conformations from previous CAPRI targets, namely targets T32, T41, T47 and T50. Fig. 4 shows the corresponding hit rate plot obtained with iScore and the HADDOCK scoring function. Hit rate plots are commonly used to compare different scoring functions. The hit rate at N represents the fraction of near-native models contained in the best N models predicted by a scoring function. As seen in Fig. 4 and Table 1, iScore performs better than HADDOCK on 2 of these cases (T32 and T41) and shows similar performance on the remaining two. These results are in line with those reported in [18] where iScore performed very well on a large range of test cases.

4. Impact

The software presented in this paper provides ease of use in end-to-end platform for scoring and ranking of PPIs. Thanks to the provided executable scripts, users can easily generate the graphs, compute their pairwise kernels and use them to train a SVM model. The self-contained archive file generated during the training contains all the necessary information to rank new docking conformations. This enables to simplify data handling and facilitates the exchange of trained model between different users. The dedicated scripts briefly described in Section 2.2.1 fully automatize the computational workflows supporting the training and testing of a SVM a model. This workflow not only makes the use of the code easier, therefore facilitating its adoption by the community, but also ensures greater reproducibility of the analysis. The modular architecture of the software facilitates its maintenance and further development.

The distribution of the computational load using MPI significantly reduces the time for training and using SVM models: Training our SVM model used in Section 3 takes under 50 s using 16 cores while scoring the 600 conformations contained in the T32 CAPRI test case takes less than 2 min.

The software presented in this paper has already been used in a recently published paper that describes the underlying methodology and used it on a large range of test cases. In agreement with Fig. 4, the results presented in [18] are competitive compared to widely used scoring functions such as HADDOCK. The software also recently has been used during the CAPRI competition.

While the software has been developed specifically for ranking PPIs, the method is generic and may be generalized for a broad range of applications that involve ranking of graphs.

Such a generalization would require users to specify the nodes and edges features as well as the graph kernels.

5. Conclusions

We have presented a new software package, iScore, that provides an end-to-end solution for ranking protein-protein interfaces. The method is based on a support vector machine classifier using random-walk graph kernels. The software is built as a Python package that can be used either interactively or through the use of dedicated executable scripts that fully automatize the computational workflow. The calculations can be distributed across multiple MPI processes and GPGPU kernels have been developed to accelerate the calculation of graph kernels. The software provides a user friendly solution for ranking PPIs more efficiently and accurately.

Acknowledgments

This work was supported by an Accelerating Scientific Discovery (ASDI) grant from the Netherlands eScience Center (grant no. 027016G04). CG acknowledges financial support from the China Scholarship Council (grant no. 201406220132). LX acknowledges financial support from the Netherlands Organisation for Scientific Research (Veni grant 722.014.005) and from an Accelerating Scientific Discovery (ASDI) grant from the Netherlands eScience Center (grant no. 027016G04). VH acknowledges financial support from the National Science Foundation USA (grant no. ACI 1640834) and the National Institutes of Health (NCATS UL1 TR002014-01), the Center for Big Data Analytics and Discovery Informatics which is cosponsored by the Institute for Cyberscience, USA, the Huck Institutes of the Life Sciences, and the Social Science Research Institute at the Pennsylvania State University, USA, and the Edward Frymoyer Endowed Professorship at Pennsylvania State University, USA and the Sudha Murty Distinguished Visiting Chair in Neurocomputing and Data Science sponsored by the Pratiksha Trust at the Indian Institute of Science.

References

[1]. Aloy P, Russell RB. Structural systems biology: modelling protein interactions. Nat Rev Mol Cell Biol 2006;7:188–97. [PubMed: 16496021]

[2]. Kiel C, Beltrao P, Serrano L. Analyzing protein interaction networks using structural information. Annu Rev Biochem 2008;77(1):415–41, PMID: 18304007. 10.1146/annurev.biochem.77.062706.133317. [PubMed: 18304007]

[3]. Halperin I, Ma B, Wolfson H, Nussinov R. Principles of docking: An overview of search algorithms and a guide to scoring functions. Proteins: Struct Funct Bioinform 2002;47(4):409–43. 10.1002/prot.10115.

[4]. Vangone A, Oliva R, Cavallo L, Bonvin AMJJ. Prediction of biomolecular complexes. In: Rigden JD, editor. From protein structure to function with bioinformatics. Dordrecht: Springer Netherlands; ISBN: 978-94-024-1069-3, 2017, p. 265–92. 10.1007/978-94-024-1069-3_8.

[5]. Macindoe G, Mavridis L, Venkatraman V, Devignes M-D, Ritchie DW. HexServer: an FFT-based protein docking server powered by graphics processors. Nucleic Acids Res 2010;38(Suppl. 2):W445–9. [PubMed: 20444869]

[6]. Schneidman-Duhovny D, Inbar Y, Nussinov R, Wolfson HJ. PatchDock and SymmDock: servers for rigid and symmetric docking. Nucleic Acids Res 2005;33(Suppl. 2):W363–7. [PubMed: 15980490]

[7]. Dominguez C, Boelens R, Bonvin AMJJ. HADDOCK: A protein protein docking approach based on biochemical or biophysical information. J Am Chem Soc 2003;125(7):1731–7, PMID: 12580598. 10.1021/ja026939x. [PubMed: 12580598]

[8]. Pierce BG, Wiehe K, Hwang H, Kim B-H, Vreven T, Weng Z. ZDOCK server: interactive docking prediction of protein–protein complexes and symmetric multimers. Bioinformatics 2014;30(12):1771–3. 10.1093/bioinformatics/btu097. [PubMed: 24532726]

[9]. Lyskov S, Gray JJ. The rosettadock server for local protein–protein docking. Nucleic Acids Res 2008;36:W233–8. [PubMed: 18442991]

[10]. Cheng TM, Blundell TL, Fernandez-Recio J. Pydock: Electrostatics and desolvation for effective scoring of rigid-body protein-protein docking. Proteins: Struct Funct Bioinform 2007;68:503–15.

[11]. Zhou H, Zhou Y. Distance-scaled, finite ideal-gas reference state improves structure derived potentials of mean force for structure selection and stability prediction. Prot Sci 2002;11:2714–6.

[12]. Huang S-Y, Zou X An iterative knowledge-based scoring function for protein–protein recognition. Proteins: Struct Funct Bioinform 2008;72(2):557–79.

[13]. Bourquard T, Bernauer J, Azé J, Poupon A. A collaborative filtering approach for protein-protein docking scoring functions. PLoS One 2011;6(4). e18541. [PubMed: 21526112]

[14]. Khashan R, Zheng W, Tropsha A. Scoring protein interaction decoys using exposed residues (SPIDER): a novel multibody interaction scoring function based on frequent geometric patterns of interfacial residues. Proteins: Struct Funct Bioinform 2012;80(9):2207–17.

[15]. Moal IH, Barradas-Bautista D, Jiménez-García B, Torchala M, van der Velde A, Vreven T, Weng Z, Bates PA, Fernández-Recio J. IRaPPA: information retrieval based integration of biophysical models for protein assembly selection. Bioinformatics 2017;33(12):1806–13. [PubMed: 28200016]

[16]. Andreani J, Faure G, Guerois R. InterEvScore: a novel coarse-grained interface scoring function using a multi-body statistical potential coupled to evolution. Bioinformatics 2013;29(14):1742–9. [PubMed: 23652426]

[17]. Lensink M, Velankar S, Wodak S. Modeling protein-protein and protein-peptide complexes: CAPRI 6-th edition. Proteins: Struct Funct Bioinform 2017;85:359–77. 10.1002/prot.25215.

[18]. Geng C, Jung Y, Renaud N, Honavar V, Bonvin AMJJ, Xue LC. IScore: A novel graph kernel-based function for scoring protein-protein docking models. Bioinformatics 2019. 10.1093/bioinformatics/btz496.

[19]. Vreven T, Moal IH, Vangone A, Pierce BG, Kastritis PL, Torchala M, Chaleil R, Jiménez-García B, Bates PA, Fernandez-Recio J, et al. Updates to the integrated protein–protein interaction benchmarks: docking benchmark version 5 and affinity benchmark version 2. J Mol Biol 2015;427(19):3031–41. [PubMed: 26231283]

[20]. Lensink MF, Wodak SJ. Score_set: a CAPRI benchmark for scoring protein complexes. Proteins: Struct Funct Bioinform 2014;82(11):3163–9.

[21]. Hertz GZ, Stormo GD. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. Bioinformatics (Oxford, England) 1999;15(7):563–77.

[22]. Vishwanathan SVN, Schraudolph NN, Kondor R, BK M. Graph kernels. J Mach Learn Res 2010;11:1201–42.

[23]. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 1997;25(17):3389–402. 10.1093/nar/25.17.3389. [PubMed: 9254694]

[24]. NCBI. BLAST User Guide, https://www.ncbi.nlm.nih.gov/blast/html/sub_matrix.html.

[25]. Renaud N Pdb2sql : fast and versatile PDB parser using SQL queries. 2018, 10.5281/zenodo.3232888, https://github.com/DeepRank/pdb2sql.

[26]. Chang C-C, Lin C-J. LIBSVM: A library for support vector machines. ACM Trans Intell Syst Technol (TIST) 2011;2:27:1–27, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[27]. Lisandro Dn, Rodrigo P, Mario S, Jorge DE. MPI for python: Performance improvements and MPI-2 extensions. J Parallel Distrib Comput 2008;68(5):655–62. 10.1016/j.jpdc.2007.09.005.

[28]. Klöckner A, Pinto N, Lee Y, Catanzaro B, Ivanov P, Fasih A. PyCUDA and PyOpenCL: A Scripting-based approach to GPU run-time code generation. Parallel Comput 2012;38(3):157–74. 10.1016/j.parco.2011.09.001.

[29]. van Werkhoven B Kernel tuner. 2018, 10.5281/zenodo.1489995, URL https://github.com/benvanwerkhoven/kernel_tuner.

[30]. Renaud N H5xplorer : A customizable hdf5 browser with embedded ipython console. 2018, 10.5281/zenodo.3232896, URL https://github.com/DeepRank/h5xplorer.

[31]. Schrödinger, LLC. The PyMOL Molecular Graphics System, Version 1.8. 2015.

[32]. Hwang H, Vreven T, Janin J, Weng Z. Protein–protein docking benchmark version 4.0. Proteins: Struct Funct Bioinform 2010;78(15):3111–4. 10.1002/prot.22830.

[33]. Geng C, Xue LC, Bonvin AMJJ. Docking models for Docking Benchmark 4, 5 and CAPRI score_set. SBGrid Data Bank, V1; 2019, 10.15785/SBGRID/684.

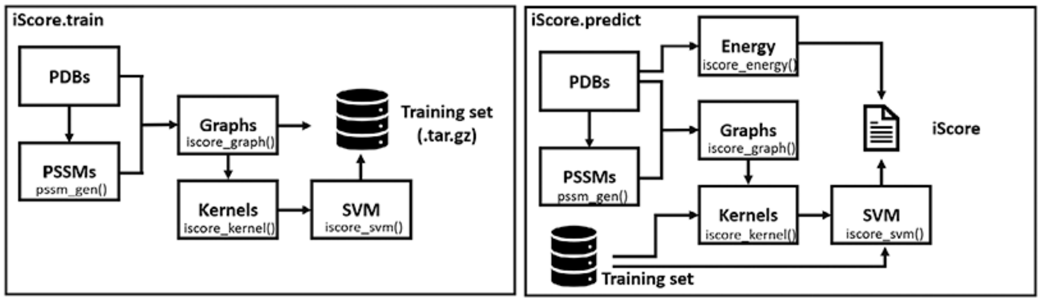

Fig. 1.
Computational workflow of iScore during the training of a SVM model and during the utilization of pre-trained model to rank new PPIs.

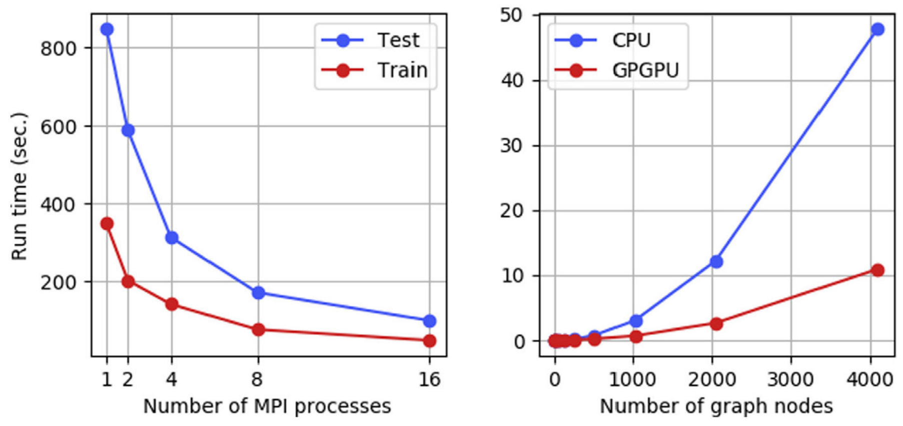

Fig. 2.

(a) Scaling of iScore.train.mpi and iScore.predict.mpi with respect to the number of MPI processes. The training and testing set contained 234 and 599 conformations respectively. (b) Average run time on CPU (Intel Xeon E5-2650 v4 @ 2.20 GHz) and GPU (Nvidia GeForce GTX 1080 Ti) for the calculation of RWGK for two graphs containing n nodes and $3n$ edges.

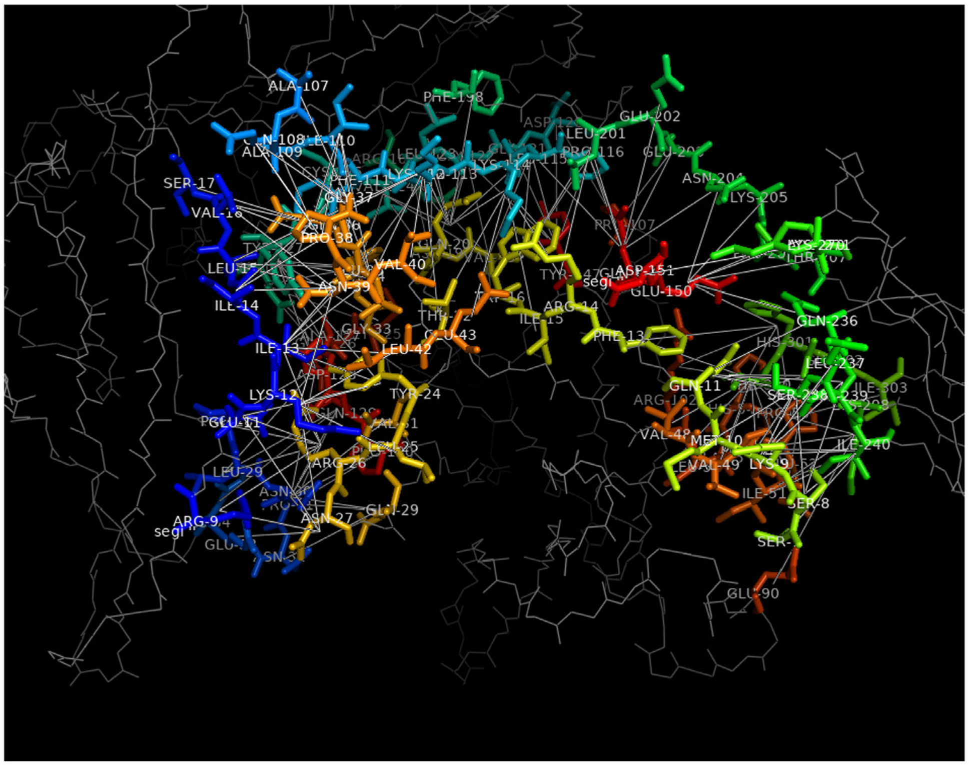

Fig. 3.
Visualization of the connection graph of PDB ID: 1IRA using iScore.h5x with the PyMol molecular viewer. All interface residues are colored differently following a rainbow color palette to facilitate their identification. The residues that are not part of the interface are represented as thin gray lines. The connection between interface residues are shown as white lines.

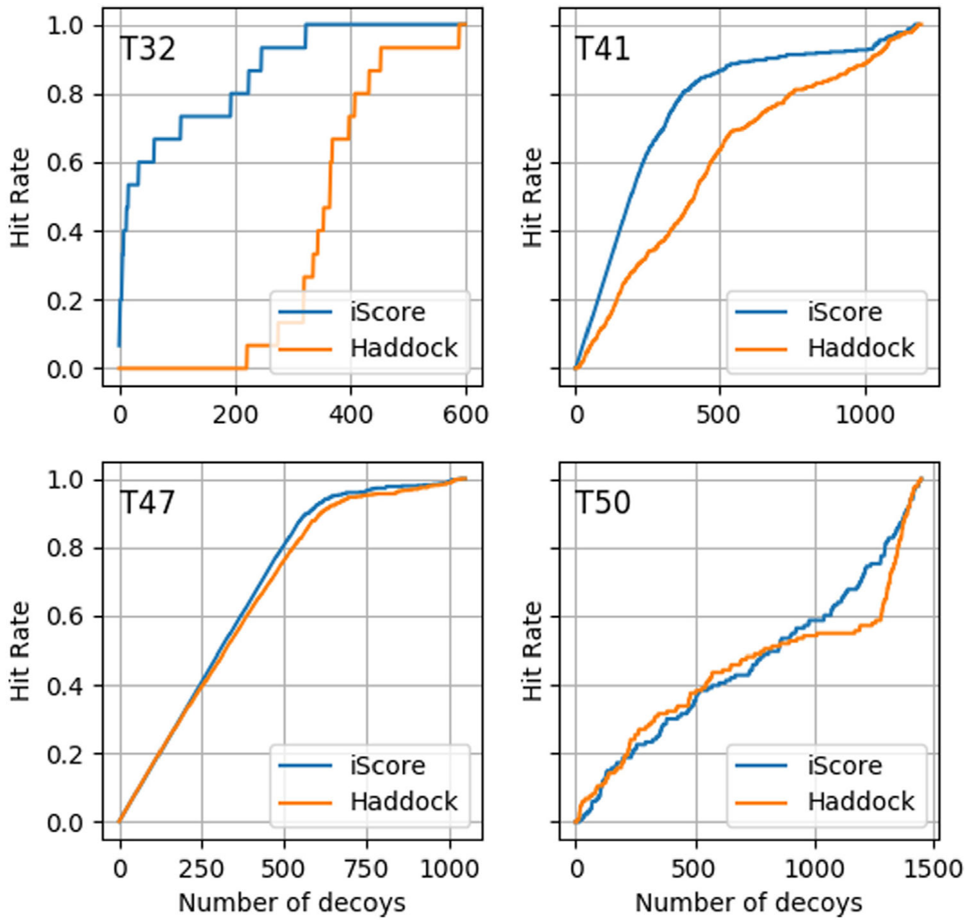

Fig. 4.
Comparison of the hit rates obtained by iScore and HADDOCK for four CAPRI targets.

Table 1

Performance of iScore and HADDOCK scoring functions on four CAPRI test cases. The number in bracket represents the number of near-native conformations for each case. The number in each column represents the number of near-native conformations in the top 10, top 50 and top 100 conformations predicted by the two methods.

	iScore			**HADDOCK**		
	Top 10	**Top 50**	**Top 100**	**Top 10**	**Top 50**	**Top 100**
T32 (15)	6	9	10	0	0	0
T41 (371)	8	48	97	1	24	46
T47 (611)	10	50	99	10	50	100
T50 (133)	0	4	10	1	9	14