Li and Ilie *BMC Bioinformatics* (2017) 18:485
DOI 10.1186/s12859-017-1871-x

BMC Bioinformatics

SOFTWARE Open Access

SPRINT: ultrafast protein-protein interaction prediction of the entire human interactome

Yiwei Li and Lucian Ilie*

Abstract

Background: Proteins perform their functions usually by interacting with other proteins. Predicting which proteins interact is a fundamental problem. Experimental methods are slow, expensive, and have a high rate of error. Many computational methods have been proposed among which sequence-based ones are very promising. However, so far no such method is able to predict effectively the entire human interactome: they require too much time or memory.

Results: We present SPRINT (Scoring PRotein INTeractions), a new sequence-based algorithm and tool for predicting protein-protein interactions. We comprehensively compare SPRINT with state-of-the-art programs on seven most reliable human PPI datasets and show that it is more accurate while running orders of magnitude faster and using very little memory.

Conclusion: SPRINT is the only sequence-based program that can effectively predict the entire human interactome: it requires between 15 and 100 min, depending on the dataset. Our goal is to transform the very challenging problem of predicting the entire human interactome into a routine task.

Availability: The source code of SPRINT is freely available from https://github.com/lucian-ilie/SPRINT/ and the datasets and predicted PPIs from www.csd.uwo.ca/faculty/ilie/SPRINT/.

Keywords: Protein-protein interaction (PPI), PPI prediction, Human interactome

Background

Protein-protein interactions (PPI) play a key role in many cellular processes since proteins usually perform their functions by interacting with other proteins. Genome-wide identification of PPIs is of fundamental importance in understanding the cell regulatory mechanisms [1] and PPI identification is one of the major objectives of systems biology. Various experimental techniques for identifying PPIs have been developed, most notably high throughput procedures such as two-hybrid assay and affinity systems [2]. Such methods are slow and expensive and have a high rate of error. A variety of computational methods have been designed to help predicting PPIs, employing sequence homology, gene co-expression, phylogenetic profiles, etc. [3–5].

Sequence-based approaches [6–17] are faster and cheaper and can be used in addition to other methods, to improve their performance. Several top methods were evaluated by Park [18]. Park and Marcotte [19] made the crucial observation that the datasets previously used for evaluation were biased due to the frequent occurrence of protein pairs common to testing and training data. They have shown that the prediction of the algorithms on the testing protein pairs is improved when the protein sequences are seen in training. To avoid this bias, they have built datasets of three levels of difficulty such that the predictive performance on these datasets generalizes to the population level. The performance of the top methods tested by Park [18] on the unbiased datasets of [19] was significantly lower than previously published, thus raising the bar higher for sequence-based methods.

We introduce a new sequence-based PPI prediction method, SPRINT (Scoring PRotein INTeractions), that is more accurate than the current state-of-the-art methods

*Correspondence: ilie@uwo.ca
Department of Computer Science, The University of Western Ontario, N6A 5B7 London, Ontario, Canada

as well as orders of magnitude faster. The SPRINT algorithm relies on the same basic hypothesis that underlies most sequence-based approaches: a pair of proteins that are pairwise similar with a pair of interacting proteins has a higher chance to interact. However, the way this idea is used is very different. Similar regions are identified using an effective multiple spaced-seed approach and then processed to eliminate elements that occur too often to be involved in interactions. Finally, a score is computed for each protein pair such that high scores indicate increased probability of interactions. Details are given in the "Methods" section.

We compared SPRINT with the top programs considered by Park [18] and Park and Marcotte [19] as well as the new method of Ding et al. [20]. The closest competitors are the machine learning-based programs of Ding et al. [20] and Martin et al. [6], and PIPE2 [7, 21], which does not use machine learning. All comparisons are done using human datasets.

To comprehensively compare the performance, we use multiple datasets, built according to the procedure of Park and Marcotte [19] from six of the most reliable human PPI databases: Biogrid, HPRD, InnateDB (experimentally validated and manually curated PPIs), IntAct, and MINT. SPRINT provides the best predictions overall, especially for the more difficult C2 and C3 types.

Then, we use the entire human interactome to compare the speed. The comparisons of [18] and [19] used fairly small datasets for comparison. In reality, these programs are meant to be used on entire proteomes and interactomes, where all protein sequences and known interactions are involved. SPRINT is several orders of magnitude faster. It takes between 15 and 100 min on a 12-core machine while the closest competitor, Ding's program, requires weeks and Martin's and PIPE2 require years. Moreover, Ding's program is unable to run the larger datasets as its memory requirements are very high.

The source code of SPRINT is freely available.

Results

We compare in this section SPRINT with several state-of-the-art sequence-based programs for PPI prediction on the most important human PPI datasets available. We focus on accurate prediction of the entire human interactome and therefore we have been using only human datasets. We start with a discussion concerning the datasets employed, as the way they are constructed can significantly impact the performance of the predicting programs.

Park and Marcotte's evaluation scheme

Park and Marcotte [19] noticed that all methods have significantly higher performance for the protein pairs in the testing data whose sequences appear also in the training data. Three cases are possible, depending on whether both proteins in the test data appear in training (C1), only one appears (C2), or none (C3). They show that essentially all datasets previously used for cross validation are very close to the C1 type, whereas in the HIPPIE meta-database of human PPIs [22] the C1-type human protein pairs accounts for only 19.2% of these cases, whereas C2-type and C3-type pairs make up 49.2% and 31.6%, respectively. Therefore, testing performed on C1-type data is not expected to generalize well to the full population. The authors proceeded to designing three separate human PPI datasets that follow the C1, C2, and C3-type rules.

Datasets

We first describe the procedure of Park and Marcotte [19] in detail. The protein sequences are from UniProt [23]. The interactions were downloaded from the protein interaction network analysis platform [24] that integrates data from six public PPI databases: IntAct [25], MINT [26], BioGRID [27], DIP [28], HPRD [29] and MIPS MPact [30]. The datasets were processed by [19] as follows. Proteins in each data set were clustered using CD-HIT2 [31] such that they shared sequence identity less than 40%. Proteins with less than 50 amino acids as well as homo-dimeric interactions were removed. Negative PPI data were generated by randomly sampling protein pairs that are not known to interact. See [19] for more details.

The total number of proteins used is 20,117, involving 24,718 PPIs. The training and testing datasets are divided into forty splits (from the file human_random.tar.gz), each consisting of one training file and three testing files, one for each type C1, C2, C3. Therefore, each C1, C2, or C3 curve produced is the average of forty curves. In addition, they tested also 40-fold cross validation on the entire PPI set. In reality, the ratio between interacting and noninteracting protein pairs is believed to be 1:100 or lower. However, this would make it very slow or impossible to run some of the algorithms. Therefore, Park and Marcotte decided to use ratio 1:1.

We have used Park and Marcotte's procedure to design similar testing datasets using six other human PPI databases. Among the most widely known human PPI databases we have chosen six that appear to be the most widely used: Biogrid, HPRD, InnateDB (experimentally validated and manually curated PPIs), IntAct, and MINT. We have used 20,160 human protein sequences downloaded from UniProt. The protein sequences and interactions were downloaded in Oct. 2016. We perform four tests for each program on each dataset: 10 fold cross-validation using all PPIs and C1, C2, and C3 tests, the datasets for which are built as explained above, with the ratio between training and testing pairs of 10:1. The details of all datasets are given in Table 1.

Competing methods

We have compared SPRINT with the four methods considered by [19]. Three of those use machine learning: [6], [8], and [9], whereas the fourth does not: PIPE [7]. Since the first three methods do not have names, we use the first author's name to identify them: Martin [6], Shen [8], and Guo [9]. Note that we have tested the improved PIPE2 [21], the same version that was tested by Park and Marcotte.

Many programs have been proposed for PPI prediction, however, very few are available. We have obtained the source code for two programs: the PPI-PK method of [10] and the program of Ding et al. [20]. The PPI-PK method was too slow on our system to be tested. We managed to run the program of Ding et al. [20] on all datasets. After eliminating the programs of Shen et al. [8] and Guo et al. [9] as placing last on the first datasets, comparison on all subsequent tests were performed against Martin, PIPE2, and Ding.

Note that PIPE2 and SPRINT do not require negative training data as they do not use machine learning algorithms. All the other programs require both positive and negative training sets. Note also that Ding's program uses also additional information concerning electrostatic and hydrophobic properties of amino acids.

Performance comparison

Park and Marcotte datasets

We present first the comparison of all five methods considered on the datasets of Park and Marcotte in Fig. 1. The receiver operating characteristic (ROC) and precision-recall (PR) curves for the four tests, CV, C1, C2, and C3, are presented.

The prediction performance on CV and C1 is very similar. The performance decreases from C1 to C2 and again to C3, both for ROC and PR curves. This is expected due to the way the datasets are constructed. The ROC curves do not distinguish very well between the prediction performance of the five methods. The difference is more clear in the PR curves. The SPRINT curve is almost always on top, especially at the beginning of the curve, where it matters the most for prediction. Ding's and Martin's are very close for CV and C1 datasets, followed by PIPE2. For C2 and C3 tests, the performance of Ding's and Martin's programs deteriorates and PIPE2 advances in second position.

Seven human PPI databases

For a comprehensive comparison, we have compared the top four programs on six datasets, computed as mentioned above from six databases: Biogrid, HPRD Release 9, InnateDB (experimentally validated and manually curated PPIs), IntAct, and MINT. Since the prediction on the CV datasets is similar with C1, we use only C1, C2 and C3 datasets.

For the purpose of predicting new PPIs, the behaviour at high specificity is important. We therefore compare the sensitivity, precision and F_1-score for several high specificity values. The table with all values is given in the Additional file 1. We present here in Table 2 the average values for each dataset type (C1, C2, and C3) over all datasets for each specificity value. At the bottom of the table we give also the average over all three dataset types. The performance of SPRINT with respect to all three measures, sensitivity, precision, and F_1-score is the highest. Only Ding comes close for C1 datasets. the overall average of SPRINT is much higher than Ding's. PIPE2 comes third and Martin last. The performance of PIPE2 decreases much less from C1 to C3 compared with Ding's. It should be noted that a weighted overall average, where the contribution of each dataset type C1,2,3 is proportional with its share of the general population, would place PIPE2 slightly ahead of Ding.

The area under the ROC and PR curves is given in Table 3 for all seven datasets, including the C1-, C2-, and C3-average, as well as the overall average across types. Ding is the winner for the C1 tests and SPRINT

Table 1 The datasets used for comparing PPI prediction methods

Dataset	PPIs			Website
	All	Training	Testing	
Park and Marcotte	24,718	14,186	1250	`www.marcottelab.org/differentialGeneralization`
Biogrid	215,029	100,000	10,000	`https://thebiogrid.org`
HPRD release 9	34,044	10,000	1000	`www.hprd.org`
InnateDB experim. validated	165,655	65,000	6500	`www.innatedb.com`
InnateDB manually curated	9913	3600	360	`www.innatedb.com`
IntAct	111,744	52,500	5250	`www.ebi.ac.uk/intact`
MINT	16,914	7000	700	`mint.bio.uniroma2.it`

The second column contains the total number of PPIs, while the third the fourth columns give the number of PPIs used for training and testing, respectively, in the C1, C2, and C3 tests

Table 2 Performance comparison at high specificity

Dataset	Specificity	Sensitivity				Precision				F1-score			
		Martin	PIPE2	Ding	SPRINT	Martin	PIPE2	Ding	SPRINT	Martin	PIPE2	Ding	SPRINT
C1 average	99.95%	6.07	7.60	11.93	13.35	98.52	98.82	88.05	99.37	11.06	13.55	20.39	22.93
	99.90%	6.53	9.20	14.24	15.91	97.36	98.61	90.65	99.29	11.88	16.45	24.10	27.03
	99.50%	17.27	21.41	29.90	29.50	96.66	97.52	98.20	98.30	28.62	34.73	45.19	45.22
	99.00%	25.48	28.73	38.72	40.14	95.55	96.40	97.28	97.52	39.14	43.69	54.69	56.58
	95.00%	55.35	48.07	65.68	62.02	91.44	90.19	92.72	92.41	68.37	62.09	76.41	73.90
C2 average	99.95%	5.55	10.65	9.22	21.45	96.33	99.42	97.92	99.62	9.78	18.91	14.69	33.16
	99.90%	5.88	11.28	9.78	23.40	93.66	98.96	96.11	99.34	10.40	19.98	15.70	36.08
	99.50%	11.73	19.52	16.59	32.73	93.86	97.11	94.11	98.22	20.17	31.86	26.59	47.77
	99.00%	15.03	24.93	22.55	37.60	91.85	95.64	93.52	97.07	25.26	38.84	34.94	52.97
	95.00%	37.41	40.95	43.83	53.17	86.45	88.43	88.27	90.76	51.17	55.33	57.69	66.18
C3 average	99.95%	1.04	1.46	1.44	6.96	94.80	93.56	91.97	99.01	2.05	2.85	2.78	12.85
	99.90%	1.20	1.78	1.65	8.04	91.31	89.73	85.41	98.50	2.37	3.46	3.18	14.76
	99.50%	4.12	4.74	4.92	19.50	85.62	89.05	85.01	96.63	7.83	8.96	9.03	31.65
	99.00%	7.40	9.89	6.92	24.81	83.64	87.41	82.80	94.99	13.51	17.32	12.48	38.32
	95.00%	24.82	27.35	24.18	39.79	80.99	82.36	81.13	87.38	37.59	40.28	36.73	53.82
Overall AVERAGE	99.95%	4.22	6.57	7.53	13.92	96.55	97.27	92.65	99.33	7.63	11.77	12.62	22.98
	99.90%	4.54	7.42	8.56	15.79	94.11	95.77	90.73	99.04	8.22	13.30	14.33	25.96
	99.50%	11.04	15.23	17.14	27.24	92.05	94.56	92.44	97.71	18.87	25.18	26.94	41.54
	99.00%	15.97	21.19	22.73	34.18	90.35	93.15	91.20	96.52	25.97	33.28	34.04	49.29
	95.00%	39.19	38.79	44.56	51.66	86.30	86.99	87.37	90.18	52.38	52.57	56.94	64.63

Sensitivity, precision, and F_1-score averages for seven datasets are given for each dataset type C1, C2 and C3, as well as overall averages across types. Darker colours represent better results. The best results are in bold

is the winner for the C2 and C3 tests. In the overall average, SPRINT comes on top. Martin is third and PIPE2 last.

All ROC and PR curves are included in the Additional file 2.

Predicting the entire human interactome

The goal of all PPI prediction methods is to predict new interactions from existing reliable ones. That means, in practice we input all known interactions – the entire interactome of an organism – and predict new ones. Of the

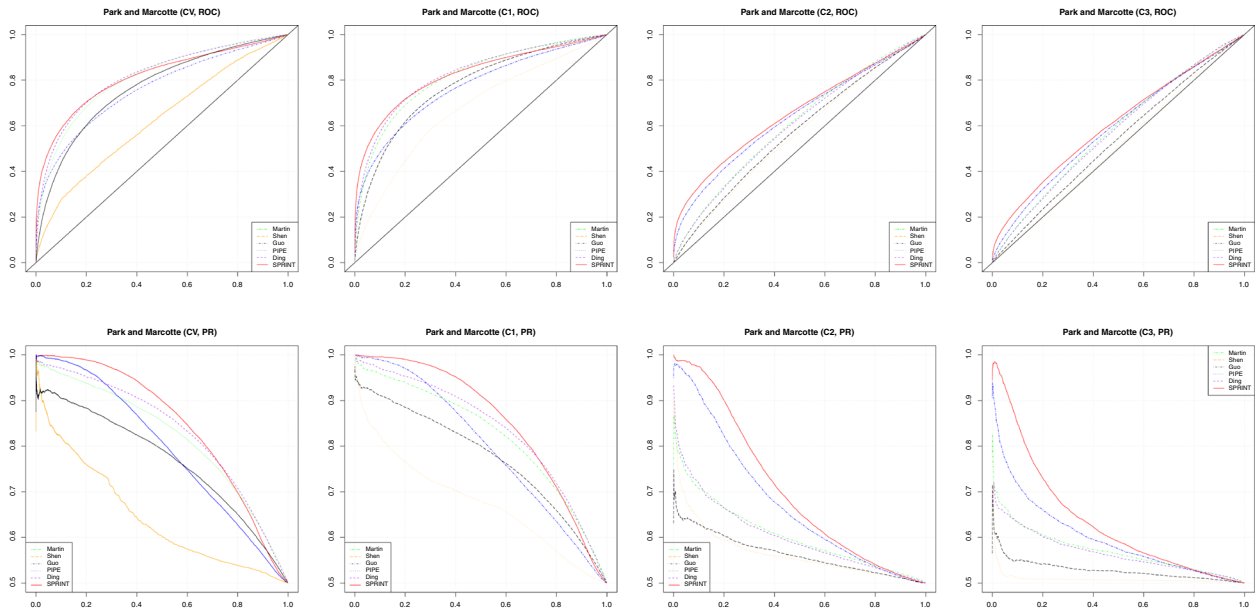

Fig. 1 Performance comparison on Park and Marcotte datasets: ROC and PR curves. The ROC curves (top row) and PR curves (bottom row) for CV, C1, C2, and C3 tests, from left to right

Table 3 Area under curves

Dataset	AUROC				AUPR			
	Martin	PIPE2	Ding	SPRINT	Martin	PIPE2	Ding	SPRINT
	C1							
Biogrid	87.54	79.01	93.06	88.11	87.20	80.52	93.08	89.24
HPRD	86.83	81.53	89.34	86.76	86.93	84.31	90.20	89.32
Innate_Exp	90.18	83.98	93.83	91.34	90.31	85.48	94.14	92.25
Innate_Man	94.11	90.26	94.89	93.09	94.93	92.22	95.73	94.75
IntAct	88.02	80.72	92.18	88.69	87.51	81.68	92.31	89.71
MINT	90.86	83.41	93.54	89.03	91.08	85.93	94.11	91.13
Park & Marcotte	81.49	76.74	82.00	82.35	82.32	79.90	83.00	85.39
	C2							
Biogrid	81.33	76.66	86.57	84.67	80.76	78.25	86.12	86.30
HPRD	83.30	81.55	84.78	86.09	82.85	83.98	84.85	88.37
Innate_Exp	83.96	81.46	87.98	89.31	83.74	82.57	87.91	90.37
Innate_Man	85.87	84.43	84.74	87.64	87.71	87.22	87.10	90.33
IntAct	81.68	77.64	85.63	83.14	80.68	78.69	85.20	85.58
MINT	86.66	81.76	87.17	86.20	86.37	84.08	87.47	88.47
Park & Marcotte	60.67	63.76	60.00	65.52	60.43	67.41	60.00	70.25
	C3							
Biogrid	76.20	71.38	79.16	79.67	74.89	70.25	77.24	80.59
HPRD	79.46	77.14	77.51	83.27	78.51	78.28	75.32	85.08
Innate_Exp	78.10	75.89	80.69	85.70	76.65	74.42	78.55	86.23
Innate_Man	71.75	73.25	65.96	76.57	73.49	74.95	66.81	80.17
IntAct	76.94	73.61	78.81	74.44	74.88	73.11	76.03	78.08
MINT	81.25	78.06	78.94	82.54	80.07	79.28	77.14	84.55
Park & Marcotte	57.86	58.90	57.00	60.60	57.07	59.84	56.00	63.49
	AVERAGES							
C1 average	88.43	82.24	91.26	88.48	88.61	84.29	91.80	90.26
C2 average	80.50	78.18	82.41	83.23	80.36	80.32	82.67	85.67
C3 average	74.51	72.60	74.01	77.54	73.65	72.88	72.44	79.74
Overall AVERAGE	81.15	77.67	82.56	83.08	80.87	79.16	82.30	85.22

AUROC and AUPR curves are given for seven datasets and three types, C1, C2, C3, for each, as well as averages for each type and overall average across types. Darker colours represent better results. The best results are in bold

newly predicted interactions, only those that are the most likely to be true interactions are kept.

For predicting the entire interactome, we need to predict the probability of interaction between any two proteins. For N proteins, that means we need to consider $(N^2 + N)/2$ protein pairs. For our 20,160 proteins, that is about 203 million potential interactions. For example, predicting one pair per second results in over six years of computation time.

We have tested the four programs, Martin's, PIPE2, Ding's, and SPRINT, on the entire human interactome, considering as given PPIs each of the six datasets in Table 1. The tests were performed on a DELL PowerEdge R620 computer with 12 cores Intel Xeon at 2.0 GHz and 256 GB of RAM, running Linux Red Hat, CentOS 6.3.

The time and memory values are shown in Table 4 for all three stages: preprocessing, training, and predicting. For each dataset, training is performed on all PPIs in that dataset and then predictions are made for all 203 million protein pairs.

Note that PIPE2 and SPRINT do not require any training. Also, preprocessing is performed only once for all protein sequences. As long as no protein sequences are added, no preprocessing needs to be done. For SPRINT, we provide all necessary similarities for all reviewed human proteins in UniProt. If new protein sequences are added, the program has an option ("`-add`") that is able to compute only the new similarities, which is very fast.

Therefore, the comparison is between predicting time of PIPE2 and SPRINT and training plus predicting time of Martin and Ding. PIPE2 and Martin are very slow and the predicting times are estimated by running the programs for 100 h and then estimating according to the number of protein pairs left to process. Both take too long to be used on the entire human interactome.

Ding's program is faster than the other two but uses a large amount of memory. It ran out of 256 GB of memory when training on the two largest datasets: Biogrid and InnateDB experimentally validated. It seems able to train

Table 4 Human interactome comparison: running time and peak memory

Dataset	Program	Time (s)			Memory (GB)		
		Preprocess	Train	Predict	Preprocess	Train	Predict
Biogrid	Martin	32,400	> 1,209,600	–	2.5	6.1	–
	PIPE2	312,120	N/A	†1,150,675,200	2.1	N/A	18.9
	Ding	37,708	–	–	3.3	> 256	–
	SPRINT	105,480	N/A	6,120	11.2	N/A	3.0
HPRD Release 9	Martin	32,400	584,640	†107,222,400	2.5	3.2	1.5
	PIPE2	312,120	N/A	†435,628,800	2.1	N/A	18.9
	Ding	37,708	236,551	374,360	3.3	79.5	79.5
	SPRINT	105,480	N/A	1,257	11.2	N/A	3.0
Innate experim. validated	Martin	32,400	> 1,209,600	–	2.5	5.7	–
	PIPE2	312,120	N/A	†872,294,400	2.1	N/A	18.9
	Ding	37,708	–	–	3.3	> 256	–
	SPRINT	105,480	N/A	3,600	11.2	N/A	3.0
Innate manually curated	Martin	32,400	26,280	†30,888,000	2.5	1.9	1.5
	PIPE2	312,120	N/A	†230,342,400	2.1	N/A	18.9
	Ding	37,708	55,532	285,323	3.3	25.4	25.4
	SPRINT	105,480	N/A	930	11.2	N/A	3.0
IntAct	Martin	32,400	> 1,209,600	–	2.5	3.5	–
	PIPE2	312,120	N/A	†616,464,000	2.1	N/A	18.9
	Ding	37,708	> 1,209,600	–	3.3	220	–
	SPRINT	105,480	N/A	2,672	11.2	N/A	3.0
MINT	Martin	32,400	101,160	†52,557,120	2.5	2.3	1.5
	PIPE2	312,120	N/A	†372,902,400	2.1	N/A	18.9
	Ding	37,708	120,720	331,865	3.3	41.1	41.1
	SPRINT	105,480	N/A	952	11.2	N/A	3.0

The predicting time for Martin's and PIPE2 was estimated by running it for 100 h and then estimating the total time according to the number of pairs left to predict. Note that PIPE2 and SPRINT do not require training as they are not using machine learning. For the entries marked with a dash, the program ran out of (256 GB) memory or ran for more than 14 days. Times marked with a dagger† are estimated

on the IntAct dataset but it could not finish training in 14 days, which is the longest we can run a job on our system.

SPRINT is approximately five orders of magnitude faster than PIPE2 and Martin. It is over two orders of magnitude faster than Ding but this is based on the small datasets. The results on IntAct seem to indicate that the difference increases for large datasets.

Another interesting property of SPRINT is that it appears to scale sublinearly with the size of the datasets, that is, the larger the datasets, the faster it runs (per PPI). This means SPRINT will continue to be fast as the datasets will grow, which it is to be expected.

It should be noted that SPRINT runs in parallel whereas the other are serial. Martin's and PIPE2 are much slower, so parallelizing the prediction would not make any difference. Ding's program on the other hand uses a considerable amout of time for training, which cannot be easily parallelized. The very large difference in speed is due to the fact that while Martin, PIPE2, and Ding consider one protein pair at the time, out of the 203 million, SPRINT simply computes all 203 million scores at the same time; see the "Methods" section for details.

In terms of memory, SPRINT requires a very modest amount of memory to predict. We successfully ran SPRINT on all entire human interactome tests in serial mode on an older MacBook (1.4 GHz processor, 4 GB RAM); the running time was between 35 min for Innate manually curated to 11 h for Biogriod.

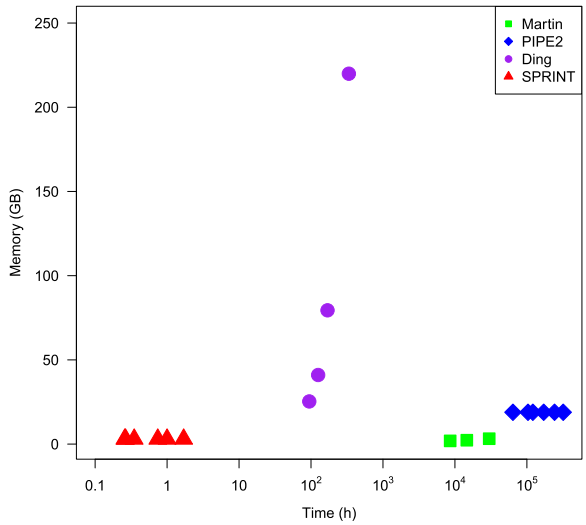

Fig. 2 Time and memory comparison. The time and memory are given for predicting the entire human interactome; the closer to origin, the better

The comparison is more visually clear in Fig. 2 where the time (in hours) and memory are plotted together for the four programs compared and those datasets for which we have either a value or at least an estimate. Note the logarithmic scale for time. The point with the highest memory for Ding's program (for the IntAct dataset) has time value fourteen days, which is the only lower bound we have. The real time may be much larger.

Methods

Basic idea

Proteins similar with interacting proteins are likely to interact as well. That is, if P_1 is known to interact with P_2 and the sequences of P_1 and P_1' are highly similar and the sequences of P_2 and P_2' are highly similar, then P_1' and P_2' are likely to interact as well. In a way or another, this is essentially the idea behind the brute force calculation of PIPE as well as the machine learning algorithms of Martin, Shen, and Guo.

SPRINT uses a complex algorithm to quickly evaluate the contribution of similar subsequences to the likelihood of interaction. The basic idea is illustrated on a toy example in Fig. 3. Assume we have given three protein pairs (P_1, Q_1), (P_2, Q_2), (P_3, Q_3), of which (P_1, Q_1) is a known interaction. Also, assume that we have detected the similar subsequences indicated by blocks of the same colour in the figure. That is, X_1, X_2, and X_3 are similar with each other, Y_1 and Y_3 are similar, etc. In this context, the fact that X_1 and U_1 belong to interacting proteins increases the likelihood that P_2 and Q_2 interact because P_2 contains X_2 that is similar with X_1 and Q_2 contains U_2 that is similar with U_1. Six such subsequence pairs between the interacting proteins P_1 and Q_1 are marked with dashed lines in Fig. 3 and they imply, using the above reasoning, two subsequence pairs in-between P_2 and Q_2 and three in between P_3 and Q_3, also marked with dashed lines. SPRINT is counting the contribution from such dash lines in order to estimate the likelihood of interaction of any protein pair. In our example, SPRINT would count two dash lines for (P_2, Q_2) and three for (P_3, Q_3).

Long similar regions should have a higher weight than short ones. To account for this we assume that all contributing blocks have a fixed length k and that a region of length ℓ contributes $\ell - k + 1$ blocks. As k is fixed, this grows linearly with ℓ. The precise score is given later in this section.

Finding similar subsequences

As described above, the first step of SPRINT is the identification of similar subsequences among the input protein sequences. This is done using spaced seeds. Spaced seeds [32, 33] are an alternative to BLAST's hit-and-extend method, that we briefly recall. Assume a match of size five is used. In this case, an exact match consists of five consecutive matching amino acids between two protein sequences. This is called a *hit*. Any such hit is then *extended* to the left and to the right until the score drops below a given threshold. If the score is sufficiently high, then the two extended subsequences are reported as similar.

Denote the five consecutive matches of a BLAST-like seed by `11111`; this is called a consecutive *seed* of weight five. *Spaced seeds* consists of matches interspersed by don't care positions; here is an example of such a spaced seed: `11****11***1`. A *spaced match* requires only the amino acids in positions corresponding to `1`'s in the seed to match; in the given example, only the amino acids in

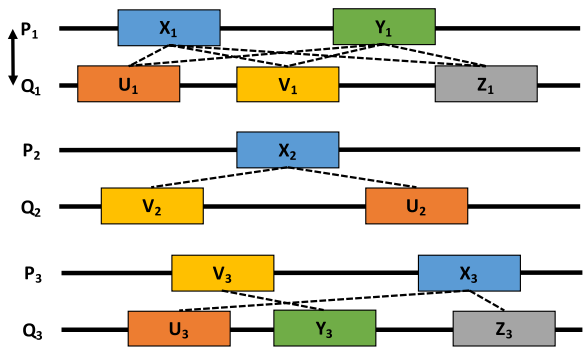

Fig. 3 Interaction inference. The proteins P_1 and Q_1 are known to interact; blocks of the same colour represent occurrences of similar subsequences. Dashed lines indicate potential contributions to interactions: there are six between P_1 and Q_1 and they imply two between P_2 and Q_2 and three between P_3 and Q_3

positions 1, 2, 7, 8, and 12 have to match. Given the spaced seed above, two *exact* spaced matches are underlined in Fig. 4a.

Note that the number of matches (the weight) is the same as for the consecutive seed; five in our case. There is a trade-off between speed and probability of finding similarities. Lower weight has increased sensitivity because it is easier to hit similar regions but lower speed since more random hits are expected and have to be processed. The best value for our problem turned out to be five.

The hit-and-extend approach works in the same way as described above, except that the initial matches are spaced as opposed to consecutive.

Spaced seeds have higher probability of detecting similar subsequences, while the number of hits is the same as for consecutive seeds; the expected number of hits is given by the weight of the seed, which is the same; see [32] for details. Several seeds [33] can detect more similar subsequences as they capture different similarities. The distribution of matches and don't care positions is crucial for the quality of the seeds and we have used SpEED [34, 35] to compute the following seeds used by SPRINT; we have experimentally determined that four seeds of weight five are the best choice: $\text{SEED}_{4,5}$ = {11****11***1, 1**1*1***1*1, 11**1***1**1, 1*1******111}.

In order to further increase the probability of finding similar subsequences, we consider also hits between similar matches, as opposed to exact ones. For example, the two amino acid sequences in Fig. 4b, though similar, do not have any *exact* spaced matches. In order to capture such similarities, we consider also hits consisting of *similar* spaced matches; an example is shown by the underlined subsequences in Fig. 4b.

To make this idea precise, we need a few definitions. *Spaced-mers* are defined analogously with *k*-mers but using a spaced seed. A *k*-mer is a contiguous sequence of *k* amino acids. Given a spaced seed, a spaced-mer consists of *k* amino acids interspersed with spaces, according to the seed. For a spaced seed *s*, we shall call the spaced-mers also *s*-mers. Figure 5 shows an example of all *s*-mers of a sequence, for *s* =11****11***1:

An exact hit therefore consists of two occurrences of the same *s*-mer. An approximate hit, on the other hand, requires two similar *s*-mers. Assume a similarity matrix *M* is given. Given a seed *s* and two *s*-mers *w* and *z*, the score between the two *s*-mers is given by the sum of the scores of the pairs of amino acids in the two *s*-mers, that is, we sum over indexes corresponding to 1's in the seed:

$$S_{s\text{-mer}}(w,z) = \sum_{s[i]=1} M(w_i, z_i)\,. \tag{1}$$

For example, for the *s*-mers w = VL␣␣␣␣KT␣␣␣A and z = HL␣␣␣␣KS␣␣␣A from Fig. 4b, we have $S_{s\text{-mer}}(w,z) = M(\text{V},\text{H}) + M(\text{L},\text{L}) + M(\text{K},\text{K}) + M(\text{T},\text{S}) + M(\text{A},\text{A})$.

Using (1), we define the set of *s*-mers that are *similar* with a given *s*-mer *w*:

$$\text{Sim}(w) = \{z \mid z s\text{-mer}, S_{s\text{-mer}}(w,z) \geq \text{T}_{\text{hit}}\}\,. \tag{2}$$

Note that $\text{Sim}(w)$ depends on the parameter T_{hit} that controls how similar two *s*-mers have to be in order to form a hit. It also depends on the seed *s* and the similarity matrix *M* but we do not include them into the notation, for clarity.

All such hits dues to similar *s*-mers are found and then extended both ways in order to identify similar regions. That means, now we have to evaluate the similarity of all the amino acids involved, so we use the regular *k*-mers. The score between two *k*-mers *A* and *B* is computed as the sum of all scores of corresponding amino acids:

$$S_{k\text{-mer}}(A,B) = \sum_{i=1}^{k} M(A_i, B_i)\,, \tag{3}$$

where A_i is the *i*th amino acid of *A*. Given a hit that consists of two *s*-mers *w* and *z*, we consider the two *k*-mers that contain the occurrences of the two *s*-mers *w* and *z* in the center, denoted *k*-mer(*w*) and *k*-mer(*z*). If $S_{k\text{-mer}}(k\text{-mer}(w), k\text{-mer}(z)) \geq \text{T}_{\text{sim}}$, then the two regions are deemed similar. Note the parameter T_{sim} that controls, together with *k*-mer size *k*, how similar two regions should be in order to be identified as such.

Implementation

Details of the fast implementation are given next. The protein sequences are encoded into bits using five bits per amino acid. (The five bits used for encoding are unrelated with the weight of the spaced seeds employed. It is a coincidence that both numbers are five.) Each protein

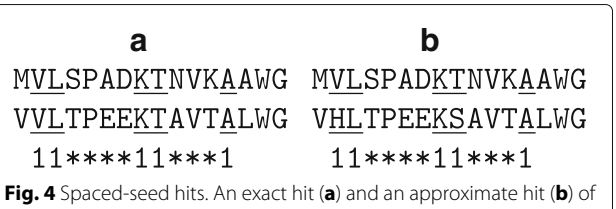

Fig. 4 Spaced-seed hits. An exact hit (**a**) and an approximate hit (**b**) of the same spaced seed

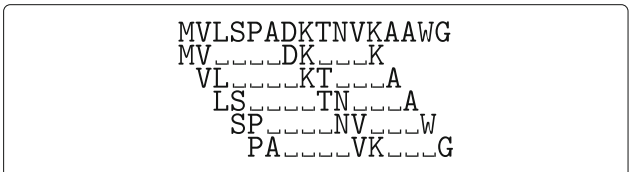

Fig. 5 S-mers. An example of all *s*-mers of a sequence

sequence is encoded as an array of unsigned 64-bit integers; each 64-bit integer stores 12 amino acids within 60 bits and 4 bits are unused. Each spaced seed is encoded using also five bits per position, `11111` for a `1` (match) and `00000` for a $\star$ (don't care). Bitwise operations are then heavily used in order to speed up recording spaced-mers into hash tables.

All spaced-mers in all protein sequences are computed and stored in a hash table, together with their location in the protein sequences. Because of our representation, the computation of each spaced-mer requires only one bitwise AND and one bit SHIFT operation. Once all spaced-mers are stored, for each spaced-mer in the table, all similar spaced-mers are computed and then all hits between the spaced-mer and similar ones are easily collected from the table and extended in search for similarities.

Post-processing similarities

We first process the similar subsequences we computed in the previous phase to remove those appearing too many times as they are believed to be just repeats that occur very often in the protein sequences without any relevance for the interaction process. We explain the algorithm on the toy example below. For the protein sequence `MVLSPADKTNVKAAWG`, assume we have found the similarities marked by lines in Fig. 6a. For example, the top line means that `MVLSP` was found to be similar with another subsequence somewhere else, the bottom line represents the same about the subsequence `KTNVKAAW`, etc.

The counts in the bottom row indicate how many times each position occurs in all similarities found. (In the figure above, this means the number of lines that cover that position). All positions with a high count, above a threshold T_{hc}, will be eliminated from all similarities, which will be modified accordingly. In our example, assuming the threshold is 5, positions 3, 4, 8, 9, and 10 have counts 5 or higher and are eliminated; see Fig. 6b. The new similarities are indicated by the lines above the sequence. For example, `MVLSP` has positions 3 and 4 removed and becomes two similarities, `MV` and `P`. The counterpart of each similarity is modified the same way.

Scoring PPIs

What we have computed so far are similarities, that is, pairs of similar subsequences of the same length. We now show how to compute the scores. First, we extend the definition of the score from k-mers to arbitrary subsequences of equal length. For two subsequences X and Y of length n, the score is given by the sum of the scores of all corresponding k-mer pairs; using (3):

$$S_e(X,Y) = \sum_{i=1}^{n-k+1} S_{k\text{-mer}}(X[\,i\,..\,i+k-1\,], Y[\,i\,..\,i+k-1\,])\,, \tag{4}$$

where $X[\,i\,..\,j] = X_iX_{i+1}\cdots X_j$. It is important to recall that any two similar sequences we find have the same length, therefore the above scoring function can be used.

Finally, we describe how the scores for whole protein sequences are computed. Initially all scores are set to zero. Each pair of proteins (P_1, P_2) that are known to interact has its own contribution to the scores of other pairs. For each computed similarity (X_1, Y_1) between P_1 and another protein Q_1 (X_1 is a subsequence of P_1 and Y_1 is a subsequence of Q_1) and for each similarity (X_2, Y_2) between P_2 and another protein Q_2, the score between Q_1 and Q_2, $S_p(Q_1, Q_2)$, is increased, using (4), by:

$$S_p(Q_1,Q_2) \leftarrow S_p(Q_1,Q_2) + \frac{S_e(X_1,Y_1)(|X_2|-k+1) + S_e(X_2,Y_2)(|X_1|-k+1)}{|Q_1||Q_2|}\,, \tag{5}$$

where $|Q|$ denotes the length of the amino acid sequence Q. That means, the score of each corresponding k-mer pair between X_1 and Y_1 is multiplied by the number of k-mers in X_2, that is, the number of times it is used to support the fact that Q_1 is interacting with Q_2. Similarly, the score of each corresponding k-mer pair between X_2 and Y_2 is multiplied by the number of k-mers in X_1. The score obtained this way is then normalized by dividing it by the product of the lengths of the proteins involved.

Fig. 6 Similarity processing. An example of similarities before (**a**) and after (**b**) post-processing

Predicting interactions

Once the score are computed, by considering all given interactions and similar subsequences and computing their impact on the other scores as above, predicting interactions is simply done according to the scores. All protein pairs are sorted decreasingly by the scores; higher scores represent higher probability to interact. If a threshold is provided, then those pairs with scores above the threshold are reported as interacting.

SPRINT

We put all the above together to summarize the SPRINT algorithm for predicting PPIs. The input consists of the

proteins sequences and PPIs. The default set of seeds is given by $\text{SEED}_{4,5}$ above but any set can be used.

SPRINT(P_s, P_i)
input: protein sequences P_s, protein interactions P_i
global: seed set SEED
output: all protein pairs sorted decreasingly by score
[Hash spaced-mers]
1. **for** each seed s in SEED **do**
2. **for** each protein sequence p in P_s **do**
3. **for** i **from** 0 **to** $|p| - |s|$ **do**
4. $w \leftarrow$ the s-mer at position i in p
5. store w in hash table H_s
6. store i in the list of w [list of positions where w occurs]
[Compute similarities]
7. **for** each seed s in SEED **do**
8. **for** each s-mer w in H_s **do**
9. compute the set $\text{Sim}(w)$ of s-mers similar with w (see (2))
10. **for** each $z \in \text{Sim}(w)$ **do**
11. **for** each position i in the list of w **do**
12. **for** each position j in the list of z **do**
13. **if** $S_{k\text{-mer}}(k\text{-mer}(w), k\text{-mer}(z)) \geq T_{sim}$
14. **then** extend the similarity both ways
15. store the pair of subsequences found
16. Process similarities to remove positions with count higher than T_{hc}
[Compute scores]
17. **for** each pair $(P, Q) \in P_s \times P_s$ **do**
18. $S_p(P, Q) \leftarrow 0$
19. **for** each $(P_1, P_2) \in P_i$ **do**
20. **for** each protein Q_1 and each similarity (X_1, Y_1) in (P_1, Q_1) **do**
21. **for** each protein Q_2 and each similarity (X_2, Y_2) in (P_2, Q_2) **do**
22. increase the score $S_p(Q_1, Q_2)$ as in (5)
[Predict PPIs]
23. sort the pairs in $P_s \times P_s$ decreasingly by score
24. **if** a threshold is provided
25. **then** output those with score above threshold

Note that the behaviour of SPRINT depends on a number of parameters: the similarity matrix M, the k-mer size k, and the thresholds T_{hit}, T_{sim}, and T_{hc}. The default matrix M is PAM120 but SPRINT accepts any similarity matrix. We have tested BLOSUM80 and BLOSUM62 and the results are nearly identical. The default values for the remaining parameters are $k = 20$, $T_{hit} = 15$, $T_{sim} = 35$, and $T_{hc} = 40$. These values have been experimentally determined using only Park and Marcotte's data set. All the other datasets have been used exclusively for testing. The program is quite stable, the results being almost unaffected by small variations of these parameters.

Conclusion

We have presented a new algorithm and software, SPRINT, for predicting PPIs that has higher performance than the current state-of-the-art programs while running orders of magnitude faster and using very little memory. SPRINT is very easy to use and we hope it will make PPI prediction for entire interactomes a routine task. It can be used on its own or in connection with other tools for PPI prediction.

Plenty of room for improvement remains, especially for the C2 and C3 data. Also, we hope to use the algorithm of SPRINT to predict interacting sites. Since they work directly with the sequence of amino acids, sequence-based methods often have an advantage in finding the actual positions where interaction occurs.

Availability and requirements

Project name: SPRINT
Project home page: https://github.com/lucian-ilie/SPRINT
Operating system(s): Platform independent
Programming language(s): C++, OpenMP
License: GPLv3.
Any restrictions to use by non-academics: None.
Data: Park and Marcotte's datasets are available from www.marcottelab.org/differentialGeneralization/. The UniProt protein sequences we used, precomputed similarities for these sequences, the datasets, and the top 1% predicted PPIs for the entire human interactome can be found at www.csd.uwo.ca/faculty/ilie/SPRINT/.

Additional files

Additional file 1: This file contains all sensitivity, precision, and F_1-score values for our tests. The averages were included in Table 2. (XLSX 78 KB)

Additional file 2: This file contains the ROC and PR curves for all tests. (PDF 4782 KB)

Acknowledgements
Evaluation has been performed on our Shadowfax cluster, which is part of the Shared Hierarchical Academic Research Computing Network (SHARCNET: http://www.sharcnet.ca) and Compute/Calcul Canada. We would like to thank Yungki Park for the PIPE2 source code.

Funding
LI has been partially supported by a Discovery Grant and a Research Tools and Instruments Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

Authors' contributions
LI proposed the problem, designed the SPRINT algorithm, computed the spaced seeds, and wrote the manuscript. YL implemented the algorithm, contributed to its design and speed improvement, installed the competing programs, downloaded and processed the datasets and performed all tests. All authors read and approved the final manuscript.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 19 May 2017 Accepted: 17 October 2017
Published online: 15 November 2017

References

1. Bonetta L. Protein-protein interactions: interactome under construction. Nature. 2010;468(7325):851–4.
2. Shoemaker BA, Panchenko AR. Deciphering protein–protein interactions. Part I. experimental techniques and databases. PLoS Comput Biol. 2007;3(3):42.
3. Shoemaker BA, Panchenko AR. Deciphering protein–protein interactions. Part II. Computational methods to predict protein and domain interaction partners. PLoS Comput Biol. 2007;3(4):43.
4. Liu ZP, Chen L. Proteome-wide prediction of protein-protein interactions from high-throughput data. Protein Cell. 2012;3(7):508–20.
5. Zahiri J, Hannon Bozorgmehr J, Masoudi-Nejad A. Computational prediction of protein–protein interaction networks: algorithms and resources. Curr Genom. 2013;14(6):397–414.
6. Martin S, Roe D, Faulon JL. Predicting protein–protein interactions using signature products. Bioinformatics. 2005;21(2):218–26.
7. Pitre S, Dehne F, Chan A, Cheetham J, Duong A, Emili A, Gebbia M, Greenblatt J, Jessulat M, Krogan N, et al. PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs. BMC Bioinformatics. 2006;7(1):1.
8. Shen J, Zhang J, Luo X, Zhu W, Yu K, Chen K, Li Y, Jiang H. Predicting protein–protein interactions based only on sequences information. Proc Natl Acad Sci. 2007;104(11):4337–41.
9. Guo Y, Yu L, Wen Z, Li M. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. Nucleic Acids Res. 2008;36(9):3025–30.
10. Hamp T, Rost B. Evolutionary profiles improve protein–protein interaction prediction from sequence. Bioinformatics. 2015;31(12):1945–50.
11. Chang DT-H, Syu YT, Lin PC. Predicting the protein-protein interactions using primary structures with predicted protein surface. BMC Bioinformatics. 2010;11(1):3.
12. Zhang YN, Pan XY, Huang Y, Shen HB. Adaptive compressive learning for prediction of protein–protein interactions from primary sequence. J Theor Biol. 2011;283(1):44–52.
13. Zahiri J, Yaghoubi O, Mohammad-Noori M, Ebrahimpour R, Masoudi-Nejad A. PPIevo: Protein–protein interaction prediction from PSSM based evolutionary information. Genomics. 2013;102(4):237–42.
14. Zhang SW, Hao LY, Zhang TH. Prediction of protein–protein interaction with pairwise kernel Support Vector Machine. Int J Mol Sci. 2014;15(2):3220–33.
15. Zahiri J, Mohammad-Noori M, Ebrahimpour R, Saadat S, Bozorgmehr JH, Goldberg T, Masoudi-Nejad A. LocFuse: human protein–protein interaction prediction via classifier fusion using protein localization information. Genomics. 2014;104(6):496–503.
16. You ZH, Chan KC, Hu P. Predicting protein-protein interactions from primary protein sequences using a novel multi-scale local feature representation scheme and the random forest. PLoS ONE. 2015;10(5):0125811.
17. You ZH, Li X, Chan KC. An improved sequence-based prediction protocol for protein-protein interactions using amino acids substitution matrix and rotation forest ensemble classifiers. Neurocomputing. 2017;228:277–82.
18. Park Y. Critical assessment of sequence-based protein-protein interaction prediction methods that do not require homologous protein sequences. BMC Bioinformatics. 2009;10(1):1.
19. Park Y, Marcotte EM. Flaws in evaluation schemes for pair-input computational predictions. Nat Methods. 2012;9(12):1134–6.
20. Ding Y, Tang J, Guo F. Predicting protein-protein interactions via multivariate mutual information of protein sequences. BMC Bioinformatics. 2016;17(1):398.
21. Pitre S, North C, Alamgir M, Jessulat M, Chan A, Luo X, Green J, Dumontier M, Dehne F, Golshani A. Global investigation of protein–protein interactions in yeast *Saccharomyces cerevisiae* using re-occurring short polypeptide sequences. Nucleic Acids Res. 2008;36(13):4286–94.
22. Schaefer MH, Fontaine JF, Vinayagam A, Porras P, Wanker EE, Andrade-Navarro MA. HIPPIE: Integrating protein interaction networks with experiment based quality scores. PloS ONE. 2012;7(2):31826.
23. UniProt Consortium and others. Reorganizing the protein space at the Universal Protein Resource (UniProt). Nucleic Acids Research. 2011:gkr981.
24. Wu J, Vallenius T, Ovaska K, Westermarck J, Mäkelä TP, Hautaniemi S. Integrated network analysis platform for protein-protein interactions. Nat Methods. 2009;6(1):75–7.
25. Kerrien S, Alam-Faruque Y, Aranda B, Bancarz I, Bridge A, Derow C, Dimmer E, Feuermann M, Friedrichsen A, Huntley R, et al. IntAct – open source resource for molecular interaction data. Nucleic Acids Res. 2007;35(suppl 1):561–5.
26. Chatr-Aryamontri A, Ceol A, Palazzi LM, Nardelli G, Schneider MV, Castagnoli L, Cesareni G. MINT: the Molecular INTeraction database. Nucleic Acids Res. 2007;35(suppl 1):572–4.
27. Stark C, Breitkreutz BJ, Chatr-Aryamontri A, Boucher L, Oughtred R, Livstone MS, Nixon J, Van Auken K, Wang X, Shi X, et al. The BioGRID interaction database: 2011 update. Nucleic Acids Res. 2011;39(suppl 1):698–704.
28. Salwinski L, Miller CS, Smith AJ, Pettit FK, Bowie JU, Eisenberg D. The database of interacting proteins: 2004 update. Nucleic Acids Res. 2004;32(suppl 1):449–51.
29. Prasad TK, Goel R, Kandasamy K, Keerthikumar S, Kumar S, Mathivanan S, Telikicherla D, Raju R, Shafreen B, Venugopal A, et al. Human protein reference database – 2009 update. Nucleic Acids Res. 2009;37(suppl 1):767–72.
30. Güldener U, Münsterkötter M, Oesterheld M, Pagel P, Ruepp A, Mewes HW, Stümpflen V. MPact: the MIPS protein interaction resource on yeast. Nucleic Acids Res. 2006;34(suppl 1):436–41.
31. Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics. 2006;22(13):1658–9.
32. Ma B, Tromp J, Li M. PatternHunter: faster and more sensitive homology search. Bioinformatics. 2002;18(3):440–5.
33. Li M, Ma B, Kisman D, Tromp J. PatternHunter II: Highly sensitive and fast homology search. J Bioinforma Comput Biol. 2004;2(03):417–39.
34. Ilie L, Ilie S. Multiple spaced seeds for homology search. Bioinformatics. 2007;23(22):2969–77.
35. Ilie L, Ilie S, Bigvand AM. SpEED: fast computation of sensitive spaced seeds. Bioinformatics. 2011;27(17):2433–4.