Bayesian Inference of Protein and Domain Interactions Using The Sum-Product Algorithm

Marcin Sikora*, Faruck Morcos†, Daniel J. Costello, Jr.*, and Jesús A. Izaguirre†
*Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556
Email: msikora@ieee.org, costello.2@nd.edu
†Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556
Email: amorcosg@nd.edu, izaguirr@cse.nd.edu

***Abstract*— In order to fully understand the functions of proteins in living organisms we must study their interactions and construct accurate interaction maps. Each protein can be composed of one or several peptide chains called domains and each protein interaction can be seen as a consequence of an underlying interaction of two domains, one from each protein. Since high-throughput methods of measuring protein interactions, such as yeast two-hybrid assay, have high error rates, the structural similarities between proteins can be exploited to detect some of the experimental errors and predict new, unmeasured interactions. In this paper we solve the problem of Bayesian inference of protein and domain interactions by computing their probabilities conditioned on the measurement results. We formulate the task of calculating these conditional probabilities as a functional marginalization problem, where the multivariate function to be marginalized naturally factors into simpler local functions, and we demonstrate how this equivalent problem can be solved using the sum-product algorithm. We note that this task is structurally similar to decoding low density parity check codes using a message passing algorithm. The robustness and accuracy of our approach is demonstrated by predicting protein and domain interactions on both real and artificial measurement data.**

I. Introduction

Progress in genomic technology has made it possible to map entire genomes of numerous organisms, with new species being examined on a steady basis. As the number of known genes increases, it is the goal of modern systems biology to understand the function of and the interrelation between proteins encoded within them [1]–[4]. Of special interest is establishing protein interaction maps that ultimately allow us to build a complete interactome for a given organism. Reaching this goal is crucial to gain a full understanding of the living processes in a cell.

Interactions, in which proteins transiently or stably bind to each other, can be detected by several experimental methods. Since the typical number of proteins in mammals and plants is between 20 and 40 thousand, hundreds of millions of protein pairs must be examined for a potential interaction. Unfortunately, testing techniques that can be automated and performed in a high throughput setting produce a large number of measurement errors, while the most accurate techniques are too complex, time consuming, and costly to be practical for mapping the entire interaction network. For example, the high-throughput yeast two-hybrid assay [5], [6] is estimated to have a 0.67 false negative and a 0.0007 false positive rate [7]. Clearly, any data processing techniques that can extrapolate from existing measurements to unmeasured protein pairs and deal with noisy results are of great practical interest.

The most promising approaches are based on the observation that similar proteins generally interact with the same partners. A particularly useful tool, which we will call an independent domain model (IDM), models the fact that proteins are generally composed of smaller, independently folding peptide modules called *domains* [7], [8]. Any interaction between two proteins is in fact a consequence of an underlying interaction of two domains, one from each protein. Since most domains appear as part of more than one protein, detecting an interaction between one protein pair can be used as evidence in inferring interactions of other protein pairs sharing the same domain pair. Additionally, knowledge of domain interactions can also be used to gain insight into the physical nature or mechanism of protein interactions.

The practical determination of domain composition of a given protein has been studied thoroughly. Since domains fold independently, it is sufficient to check whether the amino acid sequence representing a given domain is a substring of the protein sequence. Databases of both protein sequences and domain sequences are readily available on the Internet. Processed lists of proteins and their component domains are also available [9], [10].

Several techniques that use protein interaction measurements to determine protein and domain interactions have been presented in the literature, such as set cover techniques [11] and variants of Maximum Likelihood Estimation (MLE) [7], [8]. These methods first search for domain pairs that are likely to interact, compute their likelihood score, and then use this information to obtain the probabilities of protein pair interactions. In [11], the authors interpret protein pairs as elements and domain pairs as sets, with a set containing an element if the two domains are part of the two proteins. The search for interacting domain pairs is then treated as a problem of covering protein pairs that were measured as interacting with domain sets. Proposed optimality criteria include minimum set cover, minimum exact set cover, and maximum specificity set cover (MSSC). The domain interaction score is then computed

[1]This work was supported in part by NSF grants DBI-0450067 and CCF-0622940.

as the ratio of the number of measured elements in the set to the size of the set for domain pairs in the cover set and zero for all other domain pairs. In [7], the domain pairs are assumed to interact randomly, and each domain pair is characterized by its probability of interaction. These probabilities are then estimated using the MLE method, i.e., the values that best explain the observed interactions are found. The authors search for the MLE solution using the Expectation Maximization (EM) algorithm [12].

In this paper we propose to use the IDM and experimental data to infer protein and domain interactions using Bayesian inference and to use the proper conditional probability of interaction as a natural measure of the prediction confidence [13]. We demonstrate how this conditional probability for measured protein pairs, new protein pairs, and domain pairs can be computed using the Sum-Product Algorithm (SPA) [14], an iterative algorithm for computing marginal values of multivariate functions.

The necessary notation is introduced in Section 2. Section 3 states the problem of computing the conditional probability of protein and domain interaction and formulates the SPA. Section 4 demonstrates the prediction capabilities of our technique for different scenarios with real and artificial data, and Section 5 draws some conclusions.

II. Notation and assumptions

In order to precisely state the problem and develop our solution, we must first define the necessary concepts and variables. The main objects in our problem are protein pairs and domain pairs, the interaction of which we measure and predict. We denote the set of such protein pairs as $\mathcal{A}$ and domain pairs as $\mathcal{B}$. For each protein pair $(i,j) \in \mathcal{A}$ we write $\mathcal{B}_{i,j} \subset \mathcal{B}$ to denote the set of domain pairs (x,y) such that domain x is present in protein i and domain y is present in protein j. Analogously, $\mathcal{A}_{x,y} \subset \mathcal{A}$ is a set of protein pairs that contain the domain pair (x,y).

For each protein pair $(i,j) \in \mathcal{A}$ we define an interaction indicator $A_{i,j}$ such that $A_{i,j} = 1$ denotes a hypothesis that proteins i and j interact and $A_{i,j} = 0$ denotes the opposite hypothesis. In an analogous way we define an interaction indicator $B_{x,y}$ for each domain pair (x,y) in $\mathcal{B}$. Also, for each $(i,j) \in \mathcal{A}$, we use $M_{i,j}$ to describe the results of interaction measurements performed on this pair. In general, $M_{i,j}$ can include zero, one, or more measurements.

The indicators $A_{i,j}$, $B_{x,y}$, and $M_{i,j}$ for all protein and domain pairs in $\mathcal{A}$ and $\mathcal{B}$ are grouped into collections $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{M}$, respectively. Furthermore, $\mathbf{B}_{i,j}$ denotes the collection of all domain pairs $B_{x,y}$, such that $(x,y) \in \mathcal{A}_{i,j}$. For each of the above indicators and collections written in capital letters, we use lower case letters as their values. For example, $\mathbf{A} = \mathbf{a}$ refers to the hypothesis that all protein pairs interact according to configuration $\mathbf{a}$ and the probability of such a hypothesis is denoted by $P_{\mathbf{A}}(\mathbf{a})$. Whenever the indicators are clear from the context, we omit the subscript, simply writing $P(\mathbf{a})$.

We also apply a compact notation for marginalizing sums of multivariate functions [14]. For example, for $f(x,y,z,q)$ we write

$$\sum_{\sim\{x\}} f(x,y,z,q) = \sum_{y\in\mathcal{Y}} \sum_{z\in\mathcal{Z}} \sum_{q\in\mathcal{Q}} f(x,y,z,q),$$

where $\sim\{x\}$ indicates that the summation takes place over the complete domains $\mathcal{Y}$, $\mathcal{Z}$, $\mathcal{Q}$ of all arguments of f, except x. Sums with $\sim\{\}$ are taken over all arguments.

III. Bayesian inference and the Sum-Product Algorithm

A. The Sum-Product Algorithm

Bayesian inference of domain and protein interactions involves computing $P_{A_{i,j}|\mathbf{M}}(1|\mathbf{m})$ and $P_{B_{x,y}|\mathbf{M}}(1|\mathbf{m})$, the posterior probabilities of interaction given the available measurements. These probabilities, besides directly measuring our confidence in declaring certain domain or protein pairs as interacting, can be used to compute any optimal Bayesian estimator of interaction. In fact, all such estimates reduce to a simple thresholding operation on the value of $P_{A_{i,j}|\mathbf{M}}(1|\mathbf{m})$ or $P_{B_{x,y}|\mathbf{M}}(1|\mathbf{m})$, with protein or domain pairs declared as interacting if the probability exceeds the threshold and declared noninteracting otherwise.

By applying the Bayes formula we obtain

$$P_{A_{i,j}|\mathbf{M}}(1|\mathbf{m}) = \frac{\sum_{\sim\{a_{i,j}\}} P(\mathbf{a},\mathbf{b},\mathbf{m})\Big|_{a_{i,j}=1}}{\sum_{\sim\{\}} P(\mathbf{a},\mathbf{b},\mathbf{m})}, \quad (1)$$

$$P_{B_{x,y}|\mathbf{M}}(1|\mathbf{m}) = \frac{\sum_{\sim\{b_{x,y}\}} P(\mathbf{a},\mathbf{b},\mathbf{m})\Big|_{b_{x,y}=1}}{\sum_{\sim\{\}} P(\mathbf{a},\mathbf{b},\mathbf{m})}, \quad (2)$$

where the sums do not marginalize $\mathbf{m}$, a collection of known constants. The direct computation of protein and domain interaction probabilities according to formulas (1) and (2) is in most cases prohibitively complex, since the number of summands that must be evaluated is exponential in the number of protein and domain pairs involved. Instead, in this paper we apply the SPA [14], an iterative algorithm for computing marginals of multivariate functions which can be decomposed into products of simpler "local" functions in a smaller number of variables. The particular function to be marginalized in our case is $P(\mathbf{a},\mathbf{b},\mathbf{m})$, which naturally factors into $P(\mathbf{b})P(\mathbf{a}|\mathbf{b})P(\mathbf{m}|\mathbf{a})$. Each of these factors can be further decomposed into

$$P(\mathbf{b}) = \prod_{(x,y)\in\mathcal{B}} P(b_{x,y}), \quad (3)$$

$$P(\mathbf{a}|\mathbf{b}) = \prod_{(i,j)\in\mathcal{A}} P(a_{i,j}|\mathbf{b}_{i,j}), \quad (4)$$

$$P(\mathbf{m}|\mathbf{a}) = \prod_{(i,j)\in\mathcal{A}} P(m_{i,j}|a_{i,j}). \quad (5)$$

Since $a_{i,j}$ is a deterministic function of the interaction variables $\mathbf{b}_{i,j}$, the probability $P(a_{i,j}|\mathbf{b}_{i,j})$ takes the form of a simple indicator function,

$$P(a_{i,j}|\mathbf{b}_{i,j}) = \begin{cases} 1 & \text{if } a_{i,j}=0 \text{ and } \forall_{b_{x,y}\in\mathbf{b}_{i,j}} b_{x,y}=0, \\ 1 & \text{if } a_{i,j}=1 \text{ and } \exists_{b_{x,y}\in\mathbf{b}_{i,j}} b_{x,y}=1, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

The decomposition of $P(\mathbf{a},\mathbf{b},\mathbf{m})$ can be illustrated using the *factor graph* presented in Fig. 1. In a factor graph, a variable node is connected by an edge to a function node if the variable is an argument of the function. Note that $m_{i,j}$ are not included in the graph, since these variables are not marginalized. The SPA computes marginal functions of $P(\mathbf{a},\mathbf{b},\mathbf{m})$ by passing messages between variable and function nodes. A message entering or departing a variable node is itself a function of this variable. The general formula for the messages and the motivation for the message passing operation of the SPA can be found in [14]. When applied to our factor graph, the distinct message types, shown in Fig. 2a, are computed as follows:

$$\alpha_{i,j}(a_{i,j}) = P(m_{i,j}|a_{i,j}), \quad (7)$$

$$\beta_{x,y}(b_{x,y}) = P(b_{x,y}), \quad (8)$$

$$\gamma_{i,j}^{x,y}(b_{x,y}) = \sum_{\sim\{b_{x,y}\}} P(a_{i,j}|\mathbf{b}_{i,j})\alpha_{i,j}(a_{i,j}) \cdot \prod_{\substack{(x',y')\in\mathcal{B}_{i,j} \\ (x',y')\neq(x,y)}} \delta_{i,j}^{x',y'}(b_{x',y'}), \quad (9)$$

$$\delta_{i,j}^{x,y}(b_{x,y}) = \beta_{x,y}(b_{x,y}) \prod_{\substack{(i',j')\in\mathcal{A}_{x,y} \\ (i',j')\neq(i,j)}} \gamma_{i',j'}^{x,y}(b_{x,y}), \quad (10)$$

$$\epsilon_{i,j}(a_{i,j}) = \sum_{\sim\{a_{i,j}\}} P(a_{i,j}|\mathbf{b}_{i,j}) \prod_{(x,y)\in\mathcal{B}_{i,j}} \delta_{i,j}^{x,y}(b_{x,y}). \quad (11)$$

The marginal functions (1) and (2) can then be expressed as a product of messages arriving at the node of the variable not being marginalized, so that

$$\sum_{\sim\{a_{i,j}\}} P(\mathbf{a},\mathbf{b},\mathbf{m}) = \epsilon_{i,j}(a_{i,j})\alpha_{i,j}(a_{i,j}), \quad (12)$$

$$\sum_{\sim\{b_{x,y}\}} P(\mathbf{a},\mathbf{b},\mathbf{m}) = \beta_{x,y}(b_{x,y}) \prod_{(i,j)\in\mathcal{A}_{x,y}} \gamma_{i,j}^{x,y}(b_{x,y}). \quad (13)$$

If the factor graph of $P(\mathbf{a},\mathbf{b},\mathbf{m})$ is free of cycles, it is possible to order the above computations is such way that every message can be computed from values obtained in earlier steps. In such a case, (7)-(13) yield the exact marginal functions. If, however, the factor graph contains cycles, some functions $\gamma_{i,j}^{x,y}$ and $\delta_{i,j}^{x,y}$ must be set to appropriate initial values, after which computations (9) and (10) are performed iteratively. In such a case, the final steps of (12) and (13) are only approximations. Although this iterative processing is not guaranteed to converge to the exact solution, applications of iterative SPA to estimation problems in communication theory have shown its performance is nearly optimal [15].

In the procedure described by (7)-(13), the messages are functions of a single binary variable. This effectively means that the actual message would have to consist of two numbers. For example, computing $\delta_{i,j}^{x,y}(b_{x,y})$ corresponds to calculating both $\delta_{i,j}^{x,y}(0)$ and $\delta_{i,j}^{x,y}(1)$. However, for the purpose of computing (1) and (2), it is sufficient to track only the ratios of these numbers. In particular, the algorithm (7)-(10) can be rewritten as

$$\widetilde{\alpha}_{i,j} = \frac{\alpha_{i,j}(0)}{\alpha_{i,j}(1)} = \frac{P_{M_{i,j}|A_{i,j}}(m_{i,j}|0)}{P_{M_{i,j}|A_{i,j}}(m_{i,j}|1)}, \quad (14)$$

$$\widetilde{\beta}_{x,y} = \frac{\beta_{x,y}(0)}{\beta_{x,y}(1)} = \frac{P_{B_{x,y}}(0)}{P_{B_{x,y}}(1)}, \quad (15)$$

$$\widetilde{\gamma}_{i,j}^{x,y} = \frac{\gamma_{i,j}^{x,y}(0)}{\gamma_{i,j}^{x,y}(1)} = 1 + (\widetilde{\alpha}_{i,j}-1) \prod_{\substack{(x',y') \\ \neq(x,y)}} \frac{\widetilde{\delta}_{i,j}^{x',y'}}{\widetilde{\delta}_{i,j}^{x',y'}+1}, \quad (16)$$

$$\widetilde{\delta}_{i,j}^{x,y} = \frac{\delta_{i,j}^{x,y}(0)}{\delta_{i,j}^{x,y}(1)} = \widetilde{\beta}_{x,y} \prod_{\substack{(i',j') \\ \neq(i,j)}} \widetilde{\gamma}_{i',j'}^{x,y}, \quad (17)$$

where each message is just a single number. Furthermore, (1) and (2) can be expressed as

$$P_{A_{i,j}|\mathbf{M}}(1|\mathbf{m}) = \left(1+\widetilde{\alpha}_{i,j} \prod_{(x,y)} \left((\widetilde{\delta}_{i,j}^{x,y})^{-1}+1\right)\right)^{-1}, \quad (18)$$

$$P_{b_{x,y}|\mathbf{M}}(1|\mathbf{m}) = \left(1+\widetilde{\beta}_{x,y} \prod_{(i,j)} \widetilde{\gamma}_{i,j}^{x,y}\right)^{-1}. \quad (19)$$

Equations (16)-(19) take advantage of the structure of (6) and the calculation (11) is done within (18). The complete algorithm performs the following steps.

1) Initialization: compute (14) and (15), and initialize $\widetilde{\delta}_{i,j}^{x,y}$,
2) Iterative processing: iteratively compute (16) and (17),
3) Final processing: evaluate (18) and (19).

B. Input to the algorithm

The input information to our sum-product algorithm is provided through the parameters $\widetilde{\alpha}_{i,j}$ and $\widetilde{\beta}_{x,y}$. In general, for any protein pair (i,j) of interest, we will have the results of zero, one, or several experiments testing for their interaction. In the case of K experiments, with statistically independent false positive and false negative rates $f_{p,i,j}^{(k)}$ and $f_{n,i,j}^{(k)}$, $k=1,...,K_a$, respectively, the value of the likelihood ratio $\widetilde{\alpha}_{i,j}$ can be expressed as

$$\widetilde{\alpha}_{i,j} = \prod_{k=1}^{K_a} \frac{P_{M_{i,j}^{(k)}|A_{i,j}}(m_{i,j}^{(k)}|0)}{P_{M_{i,j}^{(k)}|A_{i,j}}(m_{i,j}^{(k)}|1)}, \quad (20)$$

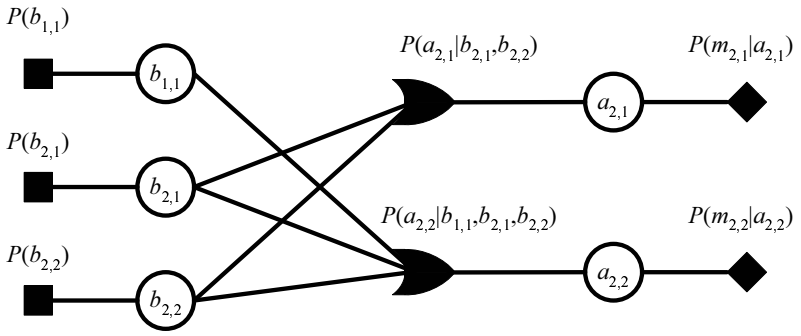

Fig. 1. Factor graph representation of the joint probability distribution function $P_{\mathbf{ABM}}$. The variable nodes $a_{i,j}$ and $b_{x,y}$ are represented by circles, while the factors in (3), (4), and (5) are shown as squares, OR-gate symbols, and diamomds, respectively.

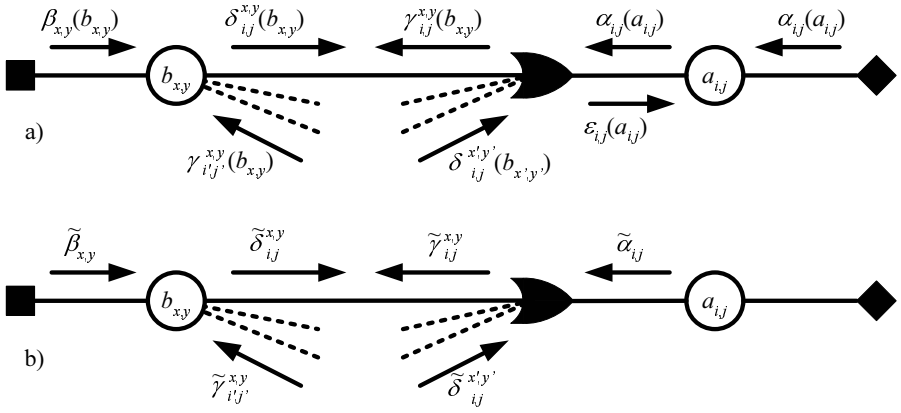

Fig. 2. Messages computed by the SPA a) before and b) after the simplification (14)-(19).

where

$$\begin{aligned} P_{M_{i,j}^{(k)}|A_{i,j}}(1|1) &= 1 - f_{n,i,j}^{(k)}, \\ P_{M_{i,j}^{(k)}|A_{i,j}}(0|1) &= f_{n,i,j}^{(k)}, \\ P_{M_{i,j}^{(k)}|A_{i,j}}(1|0) &= f_{p,i,j}^{(k)}, \\ P_{M_{i,j}^{(k)}|A_{i,j}}(0|0) &= 1 - f_{p,i,j}^{(k)}. \end{aligned}$$

With appropriate statistical models, the parameter $\widetilde{\alpha}_{i,j}$ can also be calculated for measurements with correlated errors and for other evidence that exhibits a statistical dependence on the interaction of proteins (i,j), but not on other protein pairs.

When no direct measurements are available for a protein pair (i,j), the value of $\widetilde{\alpha}_{i,j}$ is set to 1. This corresponds to the case of a protein pair for which we would like to determine the probability of interaction based on measurements of other pairs. In such a case, inspecting (16) reveals that all messages $\widetilde{\gamma}_{i,j}^{x,y}$ for this protein pair equal 1 independently of all incoming messages $\widetilde{\delta}_{i,j}^{x,y}$. As a consequence, these incoming $\widetilde{\delta}_{i,j}^{x,y}$ need not be computed during the iterative evaluation of (16) and (17) and only need to be computed right before evaluating (18). Combined with the fact that the value $\widetilde{\gamma}_{i,j}^{x,y} = 1$ does not influence the product in (17), protein pairs with no direct measurements can be completely omitted in the iterative part of the SPA.

The parameter $\widetilde{\beta}_{x,y}$ represents the a priori probability of interaction between domains (x,y), i.e., the probability of interaction before the measurements $\mathbf{m}$ are observed. If the average probability of randomly picking a domain pair that interacts is denoted by ρ, then $\widetilde{\beta}_{x,y}$ can be set to $(1-\rho)/\rho$. Also, any additional evidence indicative of the interaction between domains (x,y) that was obtained independently of $\mathbf{m}$ can be provided to the algorithm through $\widetilde{\beta}_{x,y}$.

C. Structure of the factor graph and the performance of SPA

As indicated in Section IIIA, the SPA is guaranteed to compute the exact values of conditional interaction probabilities only if the factor graph contains no cycles. If cycles are present, the solution generated by the algorithm can deviate from the exact answer, with shorter cycles causing larger deviations. The shortest possible cycles in the factor graph are cycles of length 4. These 4-cycles arise when two protein pairs are both connected to the same two domain pairs. A significant number of 4-cycles is introduced to the graph if two or more protein pairs are connected to an identical set of domain pairs. Fortunately, this case can be easily eliminated by declaring such protein pairs as equivalent and representing them by only a single variable node in the graph. If each protein pair has been independently measured, all these measurements are combined according to (20) as described in the previous subsection.

Another common source of 4-cycles are domain pairs that are connected to an identical set of protein pairs. In such a case, it is also possible to represent these domain pairs with a single variable node denoting the interaction of at least one of the domains, although some preprocessing and post-processing must be performed by the algorithm. If such domain pairs are denoted as $b_{x,y}^{(k)}$, $k = 1, ..., K_b$, then $\widetilde{\beta}_{x,y}$ for the joint node is obtained from

$$\widetilde{\beta}_{x,y} = \frac{1 - \prod_{k=1}^{K_b} P_{B_{x,y}^{(k)}}(1)}{\prod_{k=1}^{K_b} P_{B_{x,y}^{(k)}}(1)}, \tag{21}$$

and the conditional probability of interaction is computed according to

$$P_{b_{x,y}^{(k)}|\mathbf{M}}(1|\mathbf{m}) = \frac{P_{B_{x,y}^{(k)}}(1)}{1 + \left(\prod_{(i,j)} \widetilde{\gamma}_{i,j}^{x,y} - 1\right) \prod_{k'=1}^{K_b} \left(1 - P_{B_{x,y}^{(k')}}(1)\right)}. \tag{22}$$

Finally, if the factor graph is composed of several disjoint subgraphs, it is possible to perform the SPA for each of them separately. This feature can be important if the total number of protein and domain pairs analyzed by the algorithm is large, since it can considerably reduce the required amount of memory.

IV. Prediction accuracy on simulated data

A. Prediction of domain-domain interactions

In order to verify the performance of the algorithm under controlled conditions, we developed an *in silico* framework where, based on the IDM, we generated artificial domain-domain interactions (DDI) as well as matching protein-protein interactions (PPI). A domain interaction rate was assigned and protein interaction measurements were simulated by perturbing the PPI with a fixed rate of false positives (f_p) and false negatives (f_n). This approach allowed us to calculate a quantitative measure of the estimation performance of several DDI and PPI prediction algorithms. This is particularly important when evaluating the performance of DDI predictors, since there are no standard methods to test DDI in the laboratory. Although the interaction patterns in these simulations are artificial, it is important to mention that proteins and their respective domain configurations were extracted from real biological data, specifically from the Pfam database [9].

The performance of the proposed algorithm was compared with two current methods of DDI estimation mentioned in Section I: MLE and MSSC. The results of this comparison are presented as specificity vs. sensitivity curves in Figure 3.

Specificity is defined as $\mathrm{Sp} = \frac{|P \cap T|}{|P|}$ and sensitivity as $\mathrm{Sn} = \frac{|P \cap T|}{|T|}$, where P is the set of protein pairs predicted to interact and T is the set of actually interacting protein pairs. The plots represent an average over 70000 independently simulated graphs. Table I contains a summary of the parameters used with experimental and simulated interaction data.

Figure 3 shows the performance for a class of factor graphs in Figure 1 whose size is expressed in terms of the number of protein pairs connected to domain pair nodes. If the factor graph contains only a single protein pair, then MLE and the SPA produce the same result. For larger subgraphs, Figure 3 shows how the prediction of DDI can be improved by calculating the marginal probabilities using the SPA. We observe that for the case of subgraphs with 4, 7 and 10 protein pairs, which are commonly encountered in practice, the prediction accuracy of the SPA exceeds that of MLE and MSSC for the entire range of values of the specificity vs. sensitivity curve. Since higher specificity and sensitivity indicates more accurate prediction, a shift to the right represents better performance.

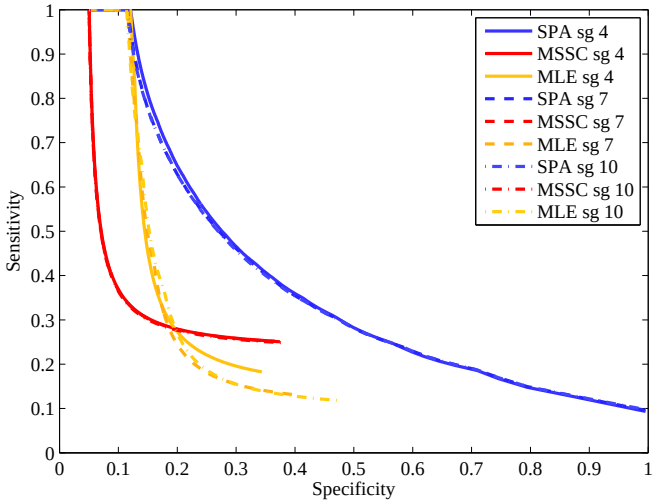

Fig. 3. Performance of prediction algorithms when estimating DDI for subgraphs of size 4, 7, and 10.

TABLE I
PARAMETERS FOR THE PREDICTION ALGORITHMS.

Parameter	DDI Prediction (Figure 3)	PPI prediction (Figure 4)	Error Detection (Figure 5)	Cross Validation (Figure 6)
Training set	4 / 7 / 10	3000	6000	70%
Testing set	2	1000	–	30%
f_p	1e-4	7e-4	7e-4	7e-4
f_n	1e-4	0.65	0.65	0.65
DDI prob.	0.05	0.05	0.05	0.05
Graph size	4 / 7 / 10	random	random	random
Iterations	70000	8500	3500	10000
Total number of proteins: 12,158 Total number of domains: 14,314 Interactions from DIP: 50,590				

B. Prediction of protein-protein interactions

The *in silico* framework that was discussed in the previous section can also be used to assess the quality of the algorithm in predicting interactions of protein pairs that were never directly measured. Again, the same parameters that were discussed above affect the outcome of the PPI prediction. Figure 4 presents specificity vs. sensitivity curves for false positive and false negative measurement rates $f_n = 0.65$ and $f_p = 0.001$, selected to resemble the estimated experimental values in [7]. We see that the SPA specificity/sensitivity curve, shown in blue, demonstrates better protection against noisy

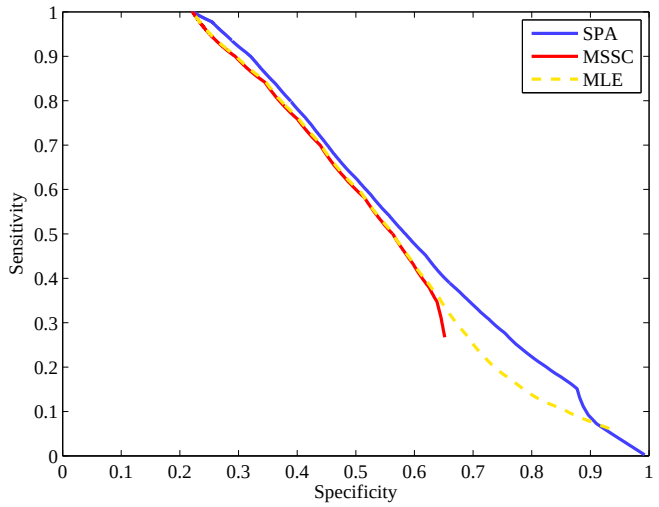

Fig. 4. Performance of predicting algorithms when estimating PPI in the presence of noisy measurements.

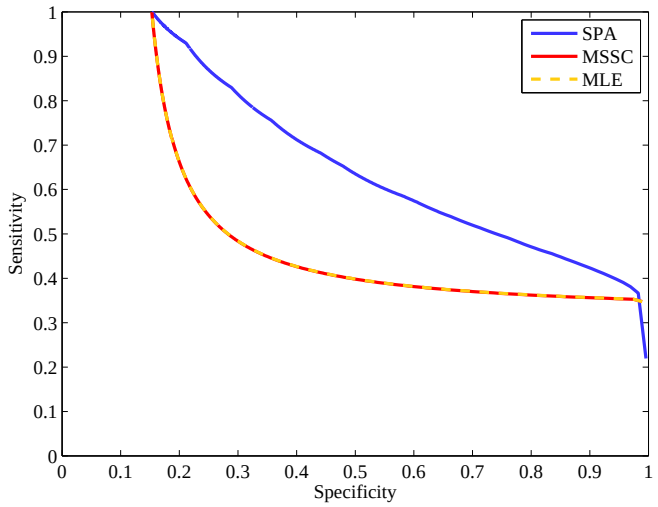

Fig. 5. False positive and false negative detection capabilities of the MLE, MSSC, and SPA algorithms.

measurements caused by the false positive and false negativ rates, compared to the MSSC method and MLE using th EM algorithm. This provides evidence that Bayesian inferenc using the SPA provides a stronger predictor in the presence o noisy PPI measurements.

C. Detection of incorrect measurements

We can also evaluate the capability of the PPI predictio methods to detect which measurements were in fact fals positives or false negatives. This can be done by predictin interactions among protein pairs that were used as training se for the algorithm. The results are compared with the origina PPI interactions before adding noise to the measurements Figure 5 shows the error correction capabilities of the thre algorithms: MSSC, MLE, and SPA. The results show tha Bayesian inference using the SPA is more effective in detecting errors that occur in the experimental measurements, a feature that helps to improve the quality of the input data and improve DDI and PPI prediction.

D. Cross validation on real interaction data

In addition to the simulation environment, we applied the new estimation method to a set of real measurements of interacting protein pairs. The domain structure for the proteins in the interacting data set was retrieved from the Pfam database version 20 [9]. The experimental PPI measurements were obtained from the Database of Interacting Proteins (DIP) [10], which contains results from laboratory experiments (including high throughput methods) to detect protein interactions. In order to test the algorithm, a standard method of cross-validation was applied. Here, 30 percent of the measured interactions were used as testing set and the remaining measurements were used as training set. The performance curves were calculated by averaging over a large number of random partitions of available measurements into test and training sets. Figure 6 shows the sensitivity/specificity curves for the SPA compared

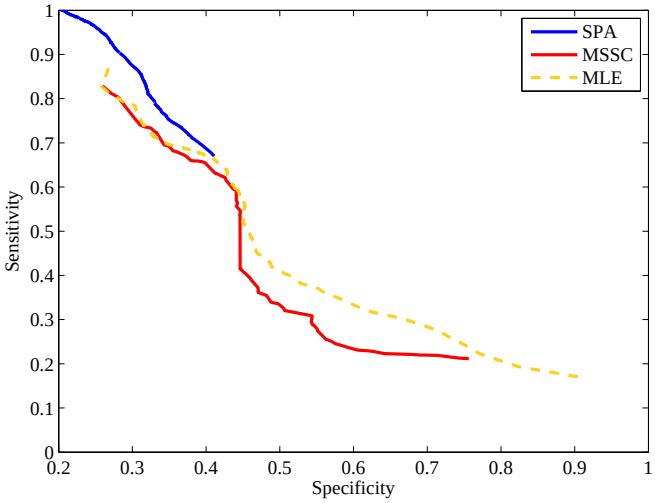

Fig. 6. Performance of prediction algorithms in experimental interaction data from DIP.

to the MLE and MSSC algorithms. SPA's performance in real data is mixed: if one is willing to accept a higher rate of false positives, and hence lower specificity, SPA is superior to both MLE and MSSC. However, with SPA we were not able to reach very high values of specificity. We believe that the reason for this limitation of the SPA algorithm, as compared to MLE and MSSC, is higher sensitivity to assumed values of f_n, f_p, and a priori DDI probability.

V. Conclusions

We have presented a new method to predict domain-domain and protein-protein interactions based on the concept of Bayesian inference and implemented it via the sum-product algorithm. The contributions of the paper are twofold. We provide a new representation of DDI and PPI prediction based on factor graphs, as well as a framework to efficiently and accurately predict DDI and PPI based on a message passing

algorithm. This framework allows us to build a probabilistic DDI network and predict new potential PPI interactions based on that information. In addition, the new method is able to detect false positives and false negatives that are common in the experimental measurements. The present methodology can be used to predict, analyze, and understand domain and protein interaction networks in different organisms. This knowledge has important implications in the understanding of the dynamic behavior of molecular interactions in a cell. Possible extensions of our technique include inference of PPI and DDI jointly with estimation of model parameters, as well as application of short cycle reduction techniques to the factor graph.

REFERENCES

[1] D. Eisenberg, E. M. Marcotte, I. Xenarios, and T. O. Yeates, "Protein function in the post-genomic era," *Nature*, vol. 405, pp. 823–826, 2000.

[2] S. Li, et al., "A map of the interactome network of the metazoan *C. elegans*," *Science*, vol. 303, no. 5657, pp. 540–543, 2004.

[3] L. Giot, et al., "A protein interaction map of *Drosophila melanogaster*," *Science*, vol. 302, no. 5651, pp. 1727–1736, 2003.

[4] J.-F. Rual, et al., "Towards a proteome-scale map of the human protein-protein interaction network," *Nature*, vol. 437, pp. 1173–1178, 2005.

[5] P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and J. M. Rothberg, "A comprehensive analysis of protein-protein interactions in *Saccharomyces cerevisiae*," *Nature*, vol. 403, pp. 623–627, 2000.

[6] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki, "A comprehensive two-hybrid analysis to explore the yeast protein interactome," *PNAS*, vol. 98, no. 8, pp. 4569–4574, 2001.

[7] M. Deng, S. Mehta, F. Sun, and T. Chen, "Inferring domain-domain interactions from protein-protein interactions," *Genome Res*, vol. 12, pp. 1540–1548, Oct 2002.

[8] R. Riley, C. Lee, C. Sabatti, and D. Eisenberg, "Inferring protein domain interactions from databases of interacting proteins," *Genome Biol.*, vol. 6, no. 10, p. R89, 2005.

[9] A. Bateman, L. Coin, R. Durbin, R. D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. L. Sonnhammer, D. J. Studholme, C. Yeats, and S. R. Eddy, "The Pfam protein families database," *Nucleic Acids Res.*, vol. 32, pp. D138–D141, Jan 2004.

[10] I. Xenarios, L. Salwinski, X. J. Duan, P. Higney, S.-M. Kim, and D. Eisenberg, "DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions," *Nucleic Acids Res.*, vol. 30, pp. 303–305, Jan 2002.

[11] C. Huang, F. Morcos, S. P. Kanaan, S. Wuchty, D. Z. Chen, and J. A. Izaguirre, "Predicting protein-protein interactions from protein domains using a set cover approach," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (in print)*, vol. 4, no. 1, pp. 1–10, 2007.

[12] T. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, pp. 47–60, Nov. 1996.

[13] D. J. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[14] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb 2001.

[15] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, pp. 657–670, Feb 2001.