Protein Interaction Inference using Particle Swarm Optimization Algorithm

Mudassar Iqbal, Alex A. Freitas, and Colin G. Johnson

Centre for Biomedical Informatics and Computing Lab., University of Kent, Canterbury, U.K.

{mi26,a.a.freitas,c.g.johnson}@ kent.ac.uk

Abstract. Many processes in the cell involve interaction among the proteins and determination of the networks of such interactions is of immense importance towards the complete understanding of cellular functions. As the experimental techniques for this purpose are expensive and potentially erroneous, there are many computational methods being put forward for prediction of protein-protein interactions. These methods use different genomic features for indirect inference of protein- protein interactions. As the interaction among two proteins is facilitated by domains, there are many methods being put forward for inference of such interactions using the specificity of assignment of domains to proteins. We present here an heuristic optimization method, particle swarm optimization, which predicts protein-protein interaction by using the domain assignments information. Results are compared with another known method which predicts domain interactions by casting the problem of interactions inference as a maximum satisfiability (MAX-SAT) problem.

1 Introduction

Computational inference of protein-protein interactions is an interesting and challenging area of research in modern biology. Computational methods infer potential interactions using one or more genomic features related to the protein pairs as predictor attributes. Many genomic experiments have produced some high quality information regarding genes/proteins which is not directly related to their interaction but could potentially be used for such a purpose.

Many computational methods use a single type of genomic data to predict protein interactions,e.g, using similarity in phylogenetic profiles, gene fusion methods, or the hypothesis involving co-expression or co-localization of interacting partners. Other methods integrate different genomic features using a variety of machine learning methods to infer new protein-protein interactions. In [1–3], one can find a few recent reviews regarding experimental and computational methods for protein-protein interaction prediction.

An important area under focus in many research projects is to infer protein interactions by looking at their domain compositions. Domains are evolutionarily conserved sequence units which are believed to be the responsible for the

interactions among the proteins to which they belong. There are many different methods which infer protein interactions using information on their domain composition. A protein pair is thought to be physically interacting if at least one of their constituent domain pair interacts. Most of the proteins in organisms like *S. Cerevisiae* are assigned one or more domains and information about the domains pairs in high confidence experimentally determined protein interaction data sets can be used to infer domain-domain and hence, protein-protein interaction. As there are no specific domain interaction data available, many methods have been developed for finding potential domain interaction from available experimentally determined high confidence protein-protein interaction datasets and then that information is used to predict back the novel protein-protein interactions as well [4–7]. In other words, these methods infer domain-domain interactions from protein protein interactions and use these inferred domain interactions to predict new protein-protein interactions, given the composition of domains in those proteins.

In a recent work [8, 9], a combinatorial approach is proposed for the inference of protein interactions using domain information. In the framework they use, this inference problem is presented as a satisfiability (more precisely MAX-SAT) problem, as explained in detail in Section 2, which is then solved using linear programming method by relaxing some of constraints of the original MAX-SAT problem.

In this work we propose the use of particle swarm optimization to solve this maximum satisfiability problem, using the problem formulation based on the one originally proposed in [8] and we also implement the technique employed by them to compare the results. Particle swarm optimization (PSO) is a population based heuristic optimization technique [10, 11], inspired by the social behavior of bird flocking or fish schooling [13]. It has been successfully used for optimizing high dimensional complex functions, mostly in continuous application domains. A good recent review about the different developments and applications on PSO can be found in [14].

This paper is organized as following. Section 2 details the formulation of the protein interaction problem into a MAX-SAT problem, as it is done originally in [8]. Section 3 proposes the use of a Particle Swarm Optimization algorithm (PSO) for this problem and discusses the related design issues for the use of PSO. In section 4, data sets about the domain assignments and protein interactions used in the experiments are described, and computational results are reported. Finally, Section 5 concludes the paper.

2 Protein Interaction Inference as MAXSAT Problem

We follow the problem formulation as is done in [8, 9], based on the hypothesis that a protein pair is interacting if and only if at least one pair of their domains (one from each protein) interact and non-interacting otherwise. We denote $P = \{p_1, p_2, ..., p_M\}$ as a set of proteins, $D = \{d_1, d_2, ..., d_N\}$ as a set of domains and

Ω_{ij} as the set of unique domain pairs contained in a protein pair (p_i, p_j). Let us consider two variables defining protein-protein and domain-domain interactions.

$$P_{ij} = \begin{cases} 1 \text{ if proteins } p_i \text{ and } p_j \text{ interact} \\ 0 \qquad\qquad \text{Otherwise} \end{cases}$$

$$D_{nm} = \begin{cases} 1 \text{ if domains } d_n \text{ and } d_m \text{ interact} \\ 0 \qquad\qquad \text{Otherwise} \end{cases}$$

Given the domain-domain interactions, we can predict the protein pairs interacting or non-interacting depending upon their corresponding domain pairs as:

$$P'_{ij} = \bigvee_{d_{nm} \in \Omega_{ij}} D_{nm} \tag{1}$$

Where the true outcome of this logical operation means the corresponding protein pair is interacting (i.e, 1) and false means non-interacting (i.e., 0). Using this relationship one needs to find the best assignment of 1's and 0's to the domain variables which best represents the data, i.e., a SAT (satisfiability) assignment satisfying all interacting and non-interacting protein pairs. As we know there are many false positives and false negatives in experimental data, such an assignment is not possible to find, so we will look for an assignment which satisfies the maximum number of relationships (clauses), which is known as the MAX-SAT problem. These problems are very difficult to solve in general and their exact solutions are not possible in general. This problem is solved in [8] using linear programming by relaxing some of the constraints as described in equations 2 and 3. The following linear program was formulated by relaxing the binary constraints on the variables.

$$\begin{aligned} \text{Minimize} \qquad & \sum_{ij} |P_{ij} - P'_{ij}| \\ \text{Subject To:} \qquad & \sum_{d_{nm} \in \Omega_{ij}} D_{nm} \geq P_{ij} \quad \forall (i,j) \\ & 0 \leq P'_{ij} \leq 1 \quad \forall (i,j) \\ & 0 \leq D_{nm} \leq 1 \quad \forall (n,m) \end{aligned} \tag{2}$$

P_{ij} is 1 or 0 if the two proteins p_i and p_j interact or not respectively, according to experimental data. Equation 2 can also be expressed in the following form.

$$\begin{aligned} \text{Minimize} \qquad & \sum_{P_{ij}=0} P'_{ij} - \sum_{P_{ij}=1} P'_{ij} \\ \text{Subject To:} \qquad & \sum_{d_{nm} \in \Omega_{ij}} D_{nm} \geq P_{ij} \quad \forall (i,j) \\ & 0 \leq P'_{ij} \leq 1 \quad \forall (i,j) \\ & 0 \leq D_{nm} \leq 1 \quad \forall (n,m) \end{aligned} \tag{3}$$

The real values obtained for variables P'_{ij} and D_{nm} after optimization represent the probabilities of they taking the integer value 1, and a threshold can be used to convert them back to binary.

3 Binary Particle Swarm Optimization Algorithm for Inference of Protein Interactions

Particle swarm optimization (PSO) is a population-based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [10–12], inspired by the social behaviour of bird flocking or fish schooling.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions (particles) and searches for optima of the given objective function by iteratively updating the positions of those particles. However, unlike GA, PSO has no genetic operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space as they are attracted by the other particle positions in the neighbourhood representing good quality candidate solutions. An individual's neighbourhood may be defined in several ways,configuring somehow the "social network" of the individual. Several neighbourhood topologies exist (full, ring, star, etc.) depending on whether an individual interacts with all, some, or only one of the rest of the population.

PSO has shown promising results on many applications, especially continuous function optimisation. A good recent review of relevant research in this area can found in [14]. The basic idea of the proposed work here is to extend the application of PSO to a more challenging real world problem, namely the inference of protein interactions, which can be framed as an optimization problem (as discussed in section 2) given the assignment of domains to the proteins, where the goal is to find the network of interactions that best explains the given experimental dataset.

In the Binary version of PSO individual components of a candidate solution (particle) are not real valued, rather 1 or 0, and velocity is interpreted as proportional likelihood, which is used in the logistic function to generate a particle's binary positions, i.e.

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * \phi_1 * (p_{id}^{t} - x_{id}^{t}) + c_2 * \phi_2 * (p_{gd}^{t} - x_{id}^{t}) \tag{4}$$

$$x_{id}^{t+1} = 1 \quad \text{if} \quad \phi_3 < \frac{1}{1 + e^{-k * v_{id}^{t+1}}} \quad \text{else} \quad 0 \tag{5}$$

Where $x_{id} \in \{0, 1\}$ is the value for the d^{th} dimention of particle i and v_{id} is the velocity, which is clamped between to a maximum value, $|V_{max}|$. p_{id} and p_{gd} are the best positions in the d^{th} dimenstion of particle i and its neighbourhood's best particle g respectively. t is the iteration index, and w is the inertia weight, determining how much of the previous velocity of the particle is preserved. This

plays the role of balancing the global and local search ability of PSO [15]. Parameter k in the logistic function is a positive constant which controls the shape of the curve. c_1, c_2 are two positive acceleration constants while ϕ_1, ϕ_2 and ϕ_3 are three uniform random numbers sampled from $U(0, 1)$. For the velocity update equation, in terms of social psychology as a metaphor, the second part of the right hand side of the velocity update equation represents the private thinking by itself; the third part is the social part, which represents the cooperation among the individuals.

3.1 Solution Representation and Objective function

Each particle represents a candidate solution to the inference problem. The position vector of particle m is $X_m = \{d_{ij}\}$ where index ij runs over all unique domains pairs in the data, i.e, a particle consists of a binary string where each bit refers to a distinct unique domain pair in the training data. These are the bits which the particle will try to optimise during the course of evolution by updating its velocity and position according to equations 4 and 5, by interacting with its neighbourhood. The *gbest* version of binary PSO is used for these experiments. Each protein pair expressed in terms of its constituent domain pairs is a clause from the point of view of logic. The objective is to maximize the number of satisfied clauses or equivalently minimize the number of unsatisfied clauses. Let us define a variable P'_{ij} for each protein pair (p_i, p_j) to indicate whether it is predicted interacting or not according to the given assignment of domain pairs by some particle (solution). The objective function can be expressed as a minimization problem.

$$\begin{aligned} \text{Min} \quad & f = \sum_{ij} \mid P_{ij} - P'_{ij} \mid \\ \text{Such that} \quad & P'_{ij} = \bigvee_{d_{nm} \in \Omega_{ij}} D_{nm} \end{aligned} \tag{6}$$

4 Experimental Design and Results

4.1 Protein-Protein Interaction and Domain Assignment Data

We obtained domain assignments from SUPERFAMILY data base [16, 17]. Superfamily database is a library of Hidden Markov Models that represents all proteins of known structure. These models are used to annotate the sequence of over 50 genomes. For *S. Cerevisiae* organism there exists 3346 sequences with at least one domain assignment, which is about 50% of the total sequences. In total 4681 domains are assigned, and there are 685 superfamily domains with at least one assignment.

We obtained the *sS. Cerevisiae* interaction data set from DIP (Data base of Interacting Proteins [18]). We obtained nearly 5000 high confidence positive

interaction in DIP which is a subset of experimentally determined interaction in DIP, called CORE. Negative interactions are hard to find. As used by many researchers in this field (e.g. [20],[21]), we use protein pairs being defined as non-interacting if they are not in same cellular compartment. This gives us many hundreds of thousand of protein pairs which are not co-localized. This is a huge data set compared with the number of positives, so we randomly sample some negatives from this pool of possible negatives, in order to obtain a more balanced class distribution for the classification algorithm. Then we only want to keep those positive or negative pairs which have at least one domain assignment for each protein in the pair in the superfamily database, as there are some proteins which do not have any significant domain assignment. This process reduces our data set of positive interactions to 3070 pairs. We also created two sets of negative examples, one with the same number of negative examples (protein pairs) as the number of positive examples, i.e., 3070, while the other with 4000 negatives. In our experiments, we will call the first dataset containing 3070 positive interactions and 3070 negative interactions as *data1*, while the other data set contraining 3070 positive interactions and 4000 negative interactions will be called as *data2*.

4.2 PSO Parameters

We rely on the standard PSO parameters settings [13]. The two constants c_1 and c_2 are set to 2.0, while parameter k in the logistic function is set to 5.0. Maximum velocity (V_{max}) is set to 4.0 and individual particle's velocities in each dimension are initialized uniformly between $-V_{max}$ and $+V_{max}$. An important issue regarding the initialization of swarm is analyzed in detail.

PSO Initialization: A Data-Driven Approach We have to decide a starting configuration for PSO, e.g., the probability of a particle taking the value 1 (or 0) in each dimension, for all particles in the swarm. Usually the population in PSO is initialized completely randomly, but PSO has dependance on initial conditions (in this case, how many 1's or 0's we put into the system at the start). Hence, one needs to find an objective and consistent way to decide the initial configuration,i.e., initial number of 1's (or zeros for that matter) in the system.

In our case, one solution to this issue is to use the domain assignment information apriori to calculate the initialization probability (of being 1 or 0) for each domain separately, and use that to probabilistically assign 1 or 0 value to all the domain pair variables, that is, for every domain pair ij, we calculate the counts of being in interacting protein pairs and non-interacting protein pairs, denoted C_p^{ij} and C_n^{ij} respectively. We have the probability of being in state 1 given by Eq. 7.

$$F_{ij} = \frac{C_p^{ij}}{(C_p^{ij} + C_n^{ij})} \tag{7}$$

Now for each domain pair ij, we generate a random number r from a uniform probability distribution $U(0,1)$. If this number is less than F_{ij}, we assign 1 to that domain else 0. We use this scheme for all the experiments done using PSO.

4.3 Cross-validation: Predicting Domain-Domain Interactions

In order to solve the linear program formulated in equation 4, as originally done in [8], we used GNU Linear Programming Kit [19](version 4.7). We used the interior point method which is a polynomial time linear programming algorithm within GNU Linear Programming Kit. The P'_{ij} values for protein pair i, j are calculated by summing over all domain pair variables $D_{nm} \in \Omega_{ij}$ and dividing by the number of domain pairs each protein pair contains in order to keep it within the bounds set in the linear program in equation 3. Since the variables D_{nm} are not binary now, we used a threshold of 0.6 to convert them back to binary form, in accordance with the original work.

An important observation about our data sets is that many of the domain pairs occur either in positive protein pairs or negatives protein pairs only. This probably has something to do with our composition of the negative data. So, in the case of PSO, we in fact exclude those domain pairs from the PSO update process, i.e., they are fixed as either zero or one, depending upon which class of protein pairs they occur, but indeed they are included while evaluating the objective function in equation 6. This does not affect the prediction accuracy, but it greatly improves the running time of the algorithm, since the algorithm has fewer unknown variables to optimize.

For both data sets, we do a 10-fold cross validation procedure. For each experiment, we divide the data (for both positive and negative classes separately) randomly in ten equal folds. Each time we use nine out of ten folds as training and the remaining one fold as a test. This process is repeated ten times each time using a different fold as the test set. For *data1* in the Tables 1 and 2, we used 100 particles and PSO was allowed to run for 500 iterations, while in the case of *data2*, the number of iterations was increased to 1000.

For each of the 10 iterations of cross-validation procedure, we infer the domain pair interactions from the training set and use those interactions to predict protein pair interactions in the test set by using the relationship in equation 1, which can also be expressed in the following algebraic form.

$$P'_{ij} = 1 - \prod_{d_{nm} \in \Omega_{ij}} (1 - D_{nm}) \tag{8}$$

Tables 1 and 2 report the average results over all 10 cross-validation folds with corresponding standard deviations, for both datasets corresponding to the particle swarm optimization algorithm as well as the linear programming method (referred as LP in tables) respectively. TPR in the tables is defined as true positives over total number of positives and FPR is defined as false positives over total number of negatives in the data. Sensitivity is the same as TPR while Specificity is defined as $1 - FPR$. The performance of both methods is reported

in terms of accuracy of prediction on test data, their corresponding true and false positive rates as well as the number of domain pairs predicted interacting (column "No. of 1's" in the tables). Protein pairs in test data which do not contain any domain pair from the training data were removed.

Table 1. Results for prediction of protein-protein interactions on test data, *data1*

Method	No. of 1's	Accuracy	TPR	FPR	Sensitivity*Specifity
PSO	1875 ± 11.06	0.826 ± 0.02	0.889 ± 0.015	0.289 ± 0.038	0.63±0.039
LP	1985 ± 9.68	0.81 ± 0.016	0.95 ± 0.01	0.45 ± 0.04	0.52±0.039

Table 2. Results for prediction of protein-protein interactions on test data, *data2*

Method	No. of 1's	Accuracy	TPR	FPR	Sensitivity*Specifity
PSO	1823 ± 13.32	0.81 ± 0.012	0.859 ± 0.017	0.253 ± 0.017	0.64±0.18
LP	1956 ± 13.42	0.78 ± 0.013	0.938 ± 0.014	0.437 ± 0.015	0.53±0.02

We can see from the Tables 1 and 2 that PSO produced better and more balanced results with a much lower rate of false positives. Results with two data sets, i.e., when we increase the proportion of negative examples from *data1* to *data2*, are not much different in the case of PSO, while they are significantly different in the case of the linear programming method. A statistical significance test (more precisely, a two-tailed student's t-test) was performed using the accuracy of both methods, and we obtained P-values for the paired t-test as 0.00073 and 0.0000026 at 95% confidence level corresponding to *data1* and *data2* respectively. The most probable explanation for these differences lies in the definition of the linear program in equation 3, which relaxes the constraint which eventually favours the positive interactions, hence much more false positive predictions. Fig. 1 shows the comparison between the two methods according to the true positive rate over false positive rate (TPR/FPR) for different folds (for data1). A qualitatively similar situation occurs for *data2*, and those results are not shown here for the sake of simplicity.

5 Conclusions

In this work, we have addressed an important bioinformatics problem, namely, the prediction of protein-protein interactions using information on their domain assignments. Particle swarm optimization is a relatively recent but very sucessful method in different optimization problems, but so far it has never been evaluated

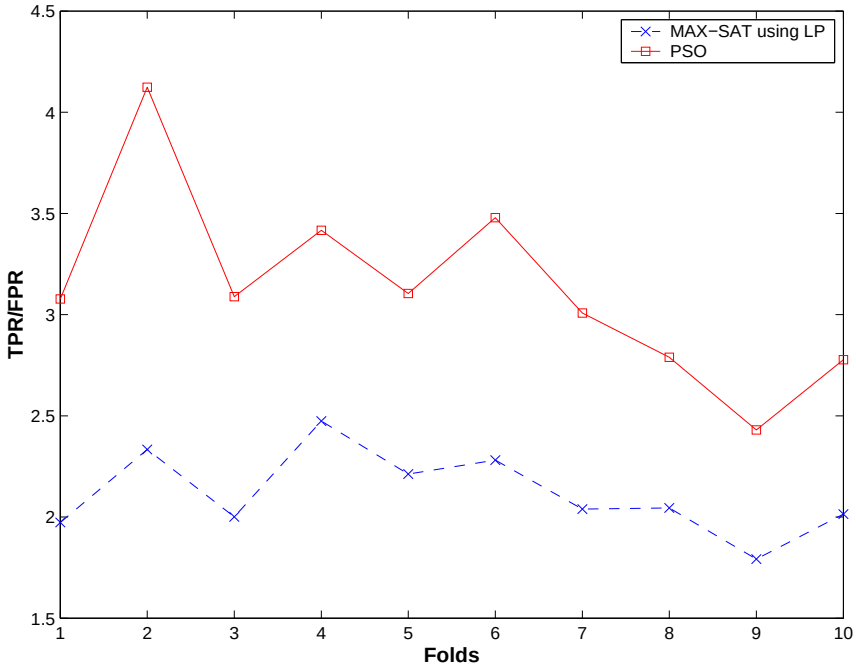

Fig. 1. Ratio of true positive rate and false positive rate for different folds of *data1*

in the type of bionformatics problem addressed here. The problem has been cast as a combinatorial optimization problem, which allowed us to propose a novel use for a binary PSO algorithm. We have compared results with a known method which solve the same problem using linear programming techniques. Comparative results in terms of predictive accuracy on test data (unseen during training) show that PSO is a competitive optimizer in an application domain involving binary variables as well. We show that PSO not just achieves significantly better predictive accuracy overall but also reduces the false positive predictions.

As far as the prediction of protein-protein interaction in general is concerned, domain information might not be enough to determine completely the protein interactions, due to other possible factors. As a future research direction, it will be worth integrating this information with other features like RNA co-expression, etc., and to use data mining techniques for finding some associations between them which can be helpful in further understanding the mechanisms of protein and domain interactions.

Acknowledgements

Mudassar Iqbal acknowledges the support from the Computing Laboratory, University of Kent and the EPSRC grant GR/T11265/01 (eXtended Particle Swarms).

References

1. A. Benjamin *et. al.*Deciphering ProteinProtein Interactions. Part I. Experimental Techniques and Databases. *PLoS Comput Biol* **3(3):e42**, 2007.
2. A. Benjamin *et. al.*Deciphering ProteinProtein Interactions. Part II. Computational Methods to Predict Protein and Domain Interaction Partners. *PLoS Comput Biol* **3(4):e43**, 2007.
3. A. Valencia, F. Pazos, Computational methods for the prediction of protein interactions. *Current Opinion in Structural Biology* **12**,368-373, 2002.
4. R. Riley *et. al.* Inferring Protein Domain Interactions From Databases of Interacting Proteins.*Genome Biology*,**6:R89**,2005.
5. M. Deng *et. al.* Inferring Domain-Domain Interactions From Protin-Protein Interactions.*Genome Res.*,**12**,1540-1548,2002.
6. H. Lee *et. al.* An Integrated Approach to the Prediction of Domain-Domain Interactions. *BMC Bioinformatics.*,**7:269**, 2006.
7. X. Li *et. al.* Improving domain-based protein interaction prediction using biologically-significant negative dataset.*International Journal of Data Mining and Bioinformatics*, **1(2)**,138-149, 2006.
8. Y. Zhang *et al.* Protein Interaction Inference as a MAX-SAT Problem. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, 2005.
9. Y. Zhang *et al.* Towards Inferring Protein Interactions: Challenges and Solutions *EURASIP Journal on Applied Signal Processing*, Article ID 37349, 2006.
10. J. Kennedy and R. Eberhart. Particle swarm optimization, *Proc. IEEE Int. Conf. on Neural Networks*,1942-1948, 1995.
11. R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory, *Proc. 6 th Int. Symposium on Micro Machine and Human Science*, 39-43, 1995.
12. Y. Shi and R. Eberhart. Parameter selection in particle swarm optimization, *Proc. 7th Annual Conf. on Evolutionary Programming*, 591-600, 1998.
13. J. Kennedy and R. Eberhart. Swarm Intelligence, *Morgan Kaufmann Publishers*,2001.
14. R. Poli *et. al.* Particle swarm optimization:An overview , *Swarm Intelligence*, *Aug.* 2007.
15. Y. Shi and R. Eberhart. A modified particle swarm optimizer, *Proc. IEEE Int. Conf. on Evolutionary Computation*, 69-73, 1998.
16. J. Gough *et. al.* SUPERFAMILY:HMMs representing all proteins of known structure. SCOP sequence searches, alignments, and genome assignments. *Nucl. Acids Res.*, **30(1)**, 268-72, 2002.
17. M. Madera *et. al.* The SUPERFAMILY database in 2004: additions and improvements. *Nucleic Acids Res.*, **32(1)**, D235-9, 2004.
18. L. Salwinski *et al.*The Database of Interacting Proteins: 2004 update. *NAR* **32**, Database issue:D449-51,2004.
19. The GNU Linear Programming Kit (version 4.7), *http://www.gnu.org/software/glpk/glpk.html*.
20. R. Jansen *et. al.* A Bayesian Networks Approach for Predicting Protein-Protein Interactions from Genomic Data. *Science*,**302**,449-453,2003.
21. D. R. Rhodes *et. al.*Probabilistic model of the human protein-protein interaction network. *Nature Biotechnology* **23(8)**,951-959, 2005.