IMPRECO: A Tool for Improving the Prediction of Protein Complexes

Mario Cannataro, Pietro Hiram Guzzi, and Pierangelo Veltri

Bioinformatics Laboratory, Experimental Medicine Department,
University Magna Graecia, Catanzaro, Italy
{cannataro,hguzzi,veltri}@unicz.it

Abstract. Proteins interact among them and different interactions form a very huge number of possible combinations representable as protein to protein interaction (PPI) networks that are mapped into graph structures. The interest in analyzing PPI networks is related to the possibility of predicting PPI properties, starting from a set of known proteins interacting among each other. For example, predicting the configuration of a subset of nodes in a graph (representing a PPI network), allows to study the generation of protein complexes. Nevertheless, due to the huge number of possible configurations of protein interactions, automatic based computation tools are required. Available prediction tools are able to analyze and predict possible combinations of proteins in a PPI network which have biological meanings. Once obtained, the protein interactions are analyzed with respect to biological meanings representing quality measures. Nevertheless, such tools strictly depend on input configuration and require biological validation. In this paper we propose a new prediction tool based on integration of different prediction results obtained from available tools. The proposed integration approach has been implemented in an on line available tool, IMPRECO standing for IMproving PREdiction of COmplexes. IMPRECO has been tested on publicly available datasets, with satisfiable results.

1 Introduction

The interactions of proteins within a cell are very huge and frequent. They interact composing a very broad network of interactions, also known as *interactome*. If two or more proteins interact for a long time forming a stable association, their interaction is known as *protein complex*. Interactomics study focuses currently: (i) on the determination of all possible interactions and (ii) on the identification of a meaningful subset of interactions. Due to the high number of proteins within a cell, manual analysis of proteins interactions is unfeasible, so the need to investigate interactions with computational methods arises [1]. We focus on interactomics as the study of Protein-Protein Interaction (PPI) as biochemical reaction among proteins, as well as the study of protein complexes.

The most natural way to model PPIs network is by using graphs [2], where proteins are represented as nodes and interactions as edges linking them. The

M. Bubak et al. (Eds.): ICCS 2008, Part III, LNCS 5103, pp. 148–157, 2008.

simplest representation is an undirected graph in which the nodes are labelled with the protein identifiers, while the edges are simple connections (i.e. no labels or directions).

Once the PPI network has been represented as a graph, the biological investigation consists in studying the structural properties of the graph [3]. For example Fig. 1 reports a graph structure where nodes are proteins and edges represent all possible interactions among proteins (nodes). Subgraphs [4], i.e. a subset of nodes and edges, may represent biological relevant and meaningful proteins interactions. For instance, the circled set of nodes and edges connecting them in Fig. 1 is an example of protein interactions representing complexes.

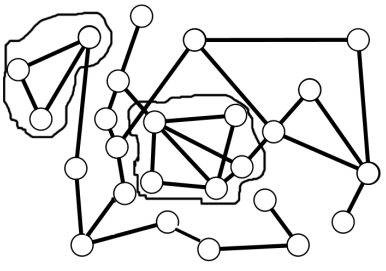

Fig. 1. A graph with dense regions

Consequently, a number of algorithms that predict complexes starting from graphs have been developed [1],[5],[4]. These algorithms, also called complexes prediction tools (or simply predictors), belong to the general class of graph clustering algorithms, where each cluster is defined as a set of nodes of the graph with their connections. Thus, clustering algorithms aim to identify subgraphs. Obviously, the quality of predictors is measured in terms of percentage of complexes biologically meaningful with respect to the meaningless ones. Clustering algorithms take as input a graph representing an interaction network among a set of proteins and an initial configuration (i.e. algorithms parameters such as the number of clusters). While initial configurations mostly depends on clustering algorithms, the initial interaction graph mostly depends on known protein interactions.

Thus, the prediction results are strongly influenced by: (i) the initial configuration of the algorithms and (ii) how valid are the initial protein to protein interactions (i.e., edges in the graph) of the input interaction network (i.e., the graph) [6]. They all apply clustering methodologies based on graph theory. None of them uses biological knowledge while running algorithms to guide the clusters identification or clusters selection.

The idea proposed in this paper (extension of a previous short communication [7]) is to combine different predictor results using an integration algorithm able to gather (partial) results from different predictors, to improve the biological relevance of the protein complexes associated to the output identified clusters. The integration algorithm starts by integrating results (i.e. clusters) obtained by running different available predictors. Three different cases are considered by

evaluating the topological relations among clusters coming from the considered predictors: (i) equality: the same clusters are returned by all (or by a significant number of) predictors; (ii) containment: it is possible to identify a containment relation among (a set of) clusters returned by all (or by a significant number of) predictors; (iii) overlap: it is possible to identify an overlap relation among (a set of) clusters returned by all (or by a significant number of) predictors.

It is possible to tune the minimum number of predictors to consider in the three possible cases as well as the kind of investigated relation. The algorithm considers iteratively all the clusters, examines the conditions trying to find possible corresponding selected clusters.

The proposed algorithm works in three phases: i) it firstly parses results coming from different predictors, then (ii) tries to associate them in one of the three possible considered configurations and finally (iii) , it performs the integration phase among clusters. The latter phase is performed by selecting clusters from the set obtained during the second phase. All phases are integrated into an on line available tool called IMPRECO (for IMproving PREdiction of Complexes). IMPRECO manages different formats of results data obtained from different predictors, integrates them and then presents results with evaluation quality measurements. IMPRECO has been tested considering three existing predictors, (MCODE [4], RNSC [5] and MCL [8]), on publicly available datasets showing considerable improvements. Quality measures are obtained by validating the predicted clusters (representing complexes) with respect to experimentally determined protein complexes.

2 The Clustering Integration Algorithm

Let P be a protein interaction network modelled as a graph G, and let PA be a complexes predictor. PA gets G as input and produces a set of subgraphs $C = \{\xi_1 ... \xi_t\}$, representing clusters, where each cluster ξ_i may be interpreted as a putative protein complex. The proposed integration algorithm receives a set of clustering outputs $CO = \{C_1 ... C_n\}$ obtained from n different predictors, then it tries to verify the topological relations among clusters and builds three set of clusters one for each of the considered topological relations, i.e. equality, containment and overlap. Finally the three sets are used to build an integrated set of clusters IO, with the aim of improving the quality of results (i.e. in terms of biological meanings) merging the clusters of the three obtained sets. We consider the matching subgraphs problem using only nodes, thus, checking for the correspondence of two subgraphs ξ_1 and ξ_m is reduced to the problem of checking their nodes equality. We now show the three algorithms for equality (Exact Matching procedure), containments (Containment procedure) and overlap (Overlap procedure) relations among clusters.

Exact Matching procedure. Let TD be a dimension threshold value. The equality relation procedure considers clusters that have at most TD nodes. Given a cluster ξ of a clustering output that satisfies this property, the procedure searches for identical clusters in the clustering outputs obtained from the others predictors.

The algorithm stores the corresponding clusters in a list called $Matched$. If at least a minimum number of identical clusters TM is found, a representative one is included in a cluster list called $Verify\_Equality$, i.e. the list of clusters that satisfies the equality relation. The procedure ends when iteratively all the clusters have been examined. The following pseudo code explains the so far described procedure.

Procedure. ExactMatching (CO, TD, TM)
// CO contains the set of input clusters
// TD is the threshold of dimension
// TM is the minimum number of required clusters
begin
  $Matched$: List;
  // the list of corresponding clusters
  $Verify\_Equality$: List
  // the list of clusters that verify the equality condition;
    FOR ALL Clusters ξ, $\|\xi\| \leq$ TD,
    // Find the corresponding clusters
    Matched:= FindMatching(ξ, CO);
      IF($\|Matched\|$) $\geq TM$
      $Verify\_Equality$:= $Verify\_Equality$ + ξ;
return $Verify\_Equality$;
end Exact Matching;

Containment procedure. The Containment procedure considers clusters with more than TD nodes. Let ξ a cluster with more than TD nodes. The procedure searches in other cluster result sets that ones including ξ. If at least TM clusters are found, then one of the found clusters is selected, respectively the smallest or the biggest in terms of nodes depending on an input parameter IS. Finally a $Verify\_Containment$ list of clusters is generated. The procedure ends when iteratively all the clusters have been examined. The following pseudo code explains the so far described procedure.

---

procedure. Containment (CO, TD, TM, IS)
// CO contains the set of selected clusters
// TD is the threshold of dimension
// TM is the minimum number of required clusters
// IS determines the selection of the biggest or smallest cluster
begin
  $Matched$: List;
  // the list of corresponding clusters
  Let $Verify\_Containment$: List;
  //list of clusters that verify the Containment condition;
    FOR EACH cluster ξ that has dimension higher than TD
    Matched= FindSub(ξ,CO);
      IF($\|Matched\|$) $\geq TM$

```
        IF IS = smallest
          Verify_Containment:= Verify_Containment+ ξ ;
        ELSE
          ξ̃ = max(ξ ∈ Matched);
          Verify_Containment:= Verify_Containment+ ξ̃ ;
return Verify_Containment
end Containment;
```

For instance, let us consider a scenario in which three clusters are depicted: (i) includes nodes A, B and C, (ii) includes nodes A, B, C and D, and (iii) includes nodes A, B, C, D and F. Let us suppose that the three clusters come from three different algorithms, that the containment procedure starts with (i), and that TM is set to 2. The nodes of cluster (i) are included in clusters (ii) and (iii), so if $Inserting\_Smallest$ is set to $Smallest$, cluster (i) will be included in $Verify\_Containment$, otherwise if it set to $Biggest$ cluster (iii) is included. $Verify\_Containment$ stores all the clusters that verify the relation. The procedure ends when iteratively all the clusters have been examined.

Overlap procedure. The Overlap procedure considers clusters with more than TD nodes. Let ξ a cluster with more than TD nodes. The procedure searches in other cluster result sets that ones that have an overlap with ξ bigger than a threshold PO. If at least TM clusters are found, then one of them is selected, respectively the smallest or the biggest in terms of nodes depending on an input parameter. Finally a $Verify\_Overlap$ list of clusters is generated. The procedure ends when iteratively all the clusters have been examined. The following pseudo code explains the so far described procedure.

```
procedure. Overlap (CO, TD, TM, II, PO)
// CO contains the set of selected clusters
// TD is the threshold of dimension
// TM is the minimum number of required clusters
// II determines the selection of a cluster
// PO determines the threshold of overlap
begin
  Let Matched : List;
  // The list of corresponding clusters
  Let Verify_Overlap: List;
  //list of clusters that verify the Overlap condition;
  FOR EACH cluster ξ that has dimension higher than TD
    Matched= FindOverlap (ξ,CO,PO);
      IF (||Matched||) ≥ TM
        IF II = smallest
          Verify_Overlap:= Verify_Overlap+ ξ ;
        ELSE
          ξ̃ = max(ξ ∈ Matched)
          Verify_Overlap:= Verify_Overlap+ ξ̃ ;
return Verify_Overlap
end Overlap;
```

Let us consider the three clusters ((i) A, B and C; (ii) A, B, D and K; and (iii) A, B, F, E and L) that come from three different algorithms and that the overlap procedure starts with (i), and that TM is 2 and PO is 10%. Cluster (i) has an intersection with both (ii) and (iii), and the overlap is higher than PO. Thus, if $Inserting\_Intersected$ is set to $Smallest$, cluster (i) will be included in IO, otherwise if it set to $Biggest$ cluster (iii) is included.

2.1 Integration Algorithm

The whole procedure of integration receives in input a set of clustering outputs, CO, then it verifies the three relations by calling the procedures described so far that produce three lists of clusters: $Verify\_Equality$, $Verify\_Containment$, and $Verify\_Overlap$. Each cluster that has been inserted in one list verifies one of the previous relations. Finally, it merges these lists. The following pseudo code explains the so far described procedure.

---

procedure. Integration CO, TD, TM, II, II,PO)
// CO contains the set of selected clusters
// TD is the threshold of dimension
// TM is the minimum number of required clusters
// II and IS determine the selection of a cluster
// PO is the threshold of shared nodes
begin
  Let $Verify\_Equality$: List;
  Let $Verify\_Containment$: List;
 Let $Verify\_Overlap$: List;
  FOR ALL clusters with dimensions lower than TD,
    $Verify\_Equality = ExactMatching(CO, TD, TM)$
  FOR ALL clusters with dimensions bigger than TD,
    $Verify\_Containment = Containment(CO, IS, TD, TM)$;
  FOR ALL clusters with dimensions bigger than TD,
    $Verify\_Overlap = Overlap(CO, II, PO, TD, TM)$
 $IO = Integrate(Verify\_Equality, Verify\_Containment, Verify\_Overlap)$;
end Integration;

---

3 Architecture of IMPRECO

The introduced integration algorithm has been fully implemented in a prototype available through a GUI accessible via a web browser and Java Web Start Technology [1]. The IMPRECO architecture comprises the following components, as depicted in Fig. 2.

Data manager module. It collects the outputs of the different predictors and translates them into a a single formalism known to IMPRECO. Although the formalisms used by the existing algorithms are quite similar, but even small

[1] java.sun.com/products/javawebstart/

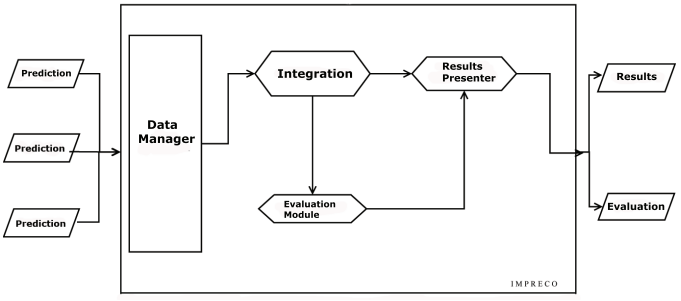

Fig. 2. Architecture of IMPRECO

differences in structure may cause problems in automatic comparison and integration methods. Thus, the data manager must translate such results. Currently, IMPRECO can read the native outputs of MCODE and MCL and can parse the RNSC output format. Users can also specify a clustering in a text file.

Integration Module. It implements the integration strategy. The current version of IMPRECO verifies the three relations in a sequential way. Initially, it builds the set of all clustering outputs starting from data parsed from the data manager. Then it tries to verify the equality relation. Then it finds those that verify the Containment relation, and it finally searches for those that satisfy the Overlap relation.

Evaluation Module. It evaluates the predictions with respect to a reference database, i.e. a catalog of verified complexes . Currently, user can use its own reference databases. Such module compares each predicted complex with those stored in the database and calculates three statistics: sensitivity, positive predictive value and accuracy as defined in [6].

Results presenter. It offers the results through a GUI as represented in Fig. 3. User can also save results in a file for successive processing.

4 Experimental Results

In order to estimate the quality of integration, we used IMPRECO to integrate some clusterings obtained running three existing algorithms over publicly available data. Protein Interaction data of yeast have been downloaded from the website [2]. The website contains data collected from the MIPS database belonging to the catalog of the yeast *Saccharomyces cerevisiae*. Authors collected data of annotated complexes, then they converted them into a graph where each node represents one protein and an edge was created between each pair of polypeptides involved in a common complex. They altered the resulting graph by randomly adding or removing edges. From those graphs we selected a network of 1094 nodes and 14658 edges and we gave it as input to three existing complexes predictors,

[2] http://rsat.scmbb.ulb.ac.be/sylvain/clustering_evaluation

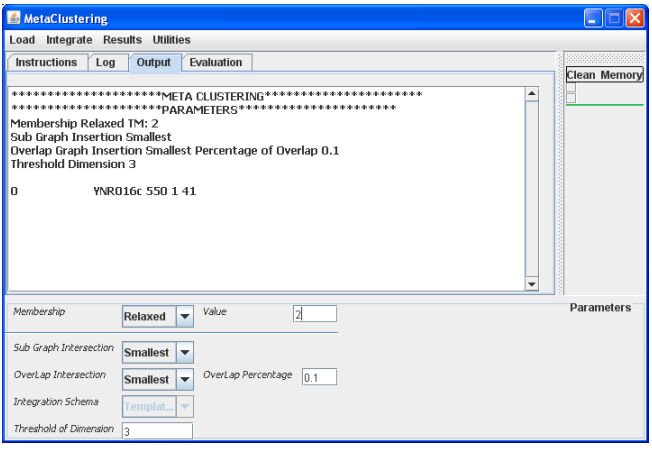

Fig. 3. The GUI of IMPRECO

MCODE, RNSC and MCL, to obtain the proposed clusters. IMPRECO integrated the results inserting in the integrated output those clusters that verified one relation in at least two outputs. For instance, let us consider Fig. 1 that depicts an example of clusters verifying the containment relation that are merged.

Table 1. Example of Considered Clusters

Algorithm	*Cluster*
MCODE-RNSC	YKL193C YDR028C YOR178C YER133W YMR311C
MCL	YKL193C YDR028C YOR178C YER133W YMR311C YER054C YDR028C YOR178C
IMPRECO	YKL193C YDR028C YOR178C YER133W YMR311C YER054C YDR028C YOR178C

We evaluated the performances of both the integrated clustering and the initial clustering. IMPRECO considered each cluster and compared it with the verified complexes. It used as reference the MIPS database catalog [9]. For each complex, it took into account of the matched elements, i.e. the components of a cluster that are presented in the complex. Thus it calculated three statistics as defined in [6]: sensitivity, positive predictive value (PPV) and accuracy(the geometric average of the previous two measures). The first measure represents the ability to detect a complex. The second one estimates the ability to correctly predict a complex. The integrated clustering predicted by IMPRECO resulted in fewer clusters (155) than MCL (165) or RNSC (306) and more than MCODE (73), as shown in Table 2. The integrated set outperforms the other three algorithms in terms of sensitivity (0.89). Conversely, the value of PPV obtained

Table 2. Comparison of IMPRECO vs existing predictions

$QualityMeasures$	$MCODE$	MCL	$RNSC$	$IMPRECO$
Number of Clusters	73	165	306	155
Sensitivity	0.34	0.47	0.46	0.89
PPV	0.48	0.69	0.59	0.59
Accuracy	0.40	0.57	0.52	0.70

(0.59) is lower than the best value for MCL (0.70). However, the final accuracy of the integrated clustering (0.70) outperforms all the others.

In our second experiment, we used a second network of with 1034 nodes and 12235 edges available at [3]. This network is obtained by randomly adding and removing edges from the previous. We ran the MCL, RNSC and MCODE algorithms, obtaining respectively 43, 115 and 267 clusters, as summarized in Table 3.

Table 3. Results of experiment 2

$Parameter--Algorithm$	$MCODE$	MCL	$RNSC$	$A1$	$A2$	$A3$	$A4$
$No.ofClusters$	43	115	267	92	92	92	92
$Sensitivity$	0.141	0.488	0.462	0.556	0.207	0.552	0.271
PPV	0.650	0.649	0.594	0.619	0.616	0.584	0.585
$Accuracy$	0.303	0.572	0.524	0.587	0.357	0.567	0.398

The intent of this second experiment is to assess the variation in integration performance when the parameters IS and II are changed. They determine respectively the insertion of the biggest or smallest matching clusters found during the execution of steps 2 and 3. Consequently, these variations do not influence the number of inserted clusters but only their internal structure. To appreciate the impact, we performed four experiments considering the four possible configurations for IS and II: (i) biggest/biggest, identified as $A1$, (ii) smallest/smallest, identified as $A2$, (iii) smallest/biggest, identified as $A3$, and (iv) biggest/smallest, identified as $A4$. Considering the results shown in Table 3, it is evident that configuration $A1$ gives the best results for sensitivity and PPV (0.556 and 0.619) and for accuracy. In contrast, the configuration ($A2$), in which both parameters are set to *Smallest*, gives the worst results. However, we have noticed that this is not a general rule, but must be verified for each dataset.

5 Conclusion

Starting from a protein interaction network, protein complexes can be predicted by the use of clustering algorithms. The combination of different algorithms is a

[3] http://rsat.scmbb.ulb.ac.be/sylvain/clustering_evaluation

possible way to improve the prediction performances. This paper addressed this problem proposing a possible strategy of integration. This approach is integrated in an on line available tool: IMPRECO [4]. First experimental results show an improvement with respect to existing predictors. We plan to develop a parallel version of IMPRECO that will execute the data translation and the integration phases as well as the evaluation of results in a parallel way by using a grid infrastructure.

References

1. Sharan, R., Ideker, T., Kelley, B., Shamir, R., Karp, R.: Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. J. Comput. Biol. 12(6), 835–846 (2005)
2. Fell, D., Wagner, A.: The small world of metabolism. Nat. Biotechnol. 18(11), 1121–1122 (2000)
3. Lesne, A.: Complex networks: from graph theory to biology. Letters in Mathematical Physics 78(3), 235–262 (2006)
4. Bader, G., Hogue, C.: An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics 4(1), 2 (2003)
5. King, A.D., Przulj, N., Jurisica, I.: Protein complex prediction via cost-based clustering. Bioinformatics 20(17), 3013–3020 (2004)
6. Brohe, S., van Helden, J.: Evaluation of clustering algorithms for protein-protein interaction networks. BMC Bioinformatics 7, 488 (2006)
7. Cannataro, M., Guzzi, P.H., Veltri, P.: A framework for the prediction of protein complexes. In: Abstract in Proceedings of the Bioinformatics Italian Society Conference (BITS 2007) (2007)
8. Enright, A.J., Van Dongen, S., Ouzounis, C.: An efficient algorithm for large-scale detection of protein families. Nucleic Acids Research 30(7), 1575–1584 (2002)
9. Mewes, H.W., Frishman, D., Mayer, K., Mnsterktter, M., Noubibou, O., Pagel, P., Rattei, T., Oesterheld, M.A.R., Stmpflen, V.: Mips: analysis and annotation of proteins from whole genomes in 2005. Nucleic Acids Res. 34(Database issue), D169–D172 (2006)

[4] http://bioingegneria.unicz.it/~guzzi