

# Density Based Merging Search of Functional Modules in Protein-Protein Interaction (PPI) Networks

M \*

Jia Wang<sup>1</sup>, Mingming Ma<sup>2</sup>, Jun Wang<sup>1\*</sup>, Xiangmin Li<sup>1</sup>, Yuxin Chen<sup>1</sup>,  
jwma@math.pku.edu.cn

**Abstract.** In this paper, we propose a density based merging search algorithm for functional modules in protein-protein interaction (PPI) networks. The proposed algorithm first identifies a set of seed nodes, which are the most densely connected nodes in the PPI network. Then, it performs a local search to find a set of local modules, which are clusters of nodes with high density. Finally, it performs a global search to merge the local modules into a set of functional modules. The proposed algorithm is able to identify functional modules with high density and low overlap. The experimental results show that the proposed algorithm is able to identify functional modules with high density and low overlap.

**Keywords:** protein-protein interaction, functional module, density based merging search.

## 1 Introduction

In recent years, there has been a growing interest in identifying functional modules in protein-protein interaction (PPI) networks. Functional modules are groups of proteins that work together to perform specific biological functions. Identifying functional modules in PPI networks is important for understanding the underlying biological processes. There are many methods for identifying functional modules in PPI networks, such as clustering, community detection, and density based merging search. Clustering and community detection methods are based on the assumption that functional modules are dense clusters of nodes. Density based merging search methods are based on the assumption that functional modules are groups of nodes with high density. In this paper, we propose a density based merging search algorithm for functional modules in PPI networks. The proposed algorithm is able to identify functional modules with high density and low overlap.

\*Correspondence to: Jia Wang.

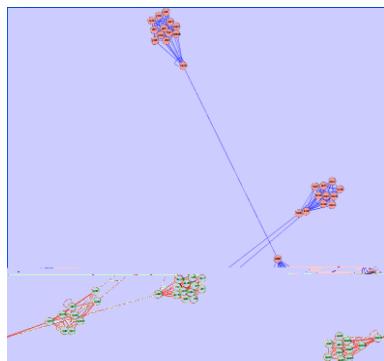
The main idea of this paper is to propose a new DBMS system, which can handle hierarchical data and manage it more efficiently. This paper is organized as follows: In section 2, we introduce the characteristics of complex objects and the DBMS. In section 3, we propose the M-DBMS system and its architecture. In section 4, we present the data model and the query language. In section 5, we evaluate the performance of the M-DBMS system. Finally, in section 6, we conclude the paper.

## 2 The DBMS Algorithm

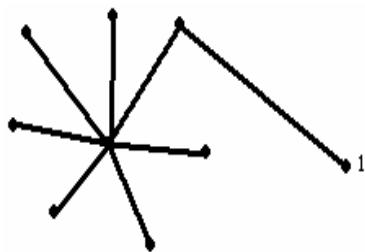
### 2.1 The Characteristics of a Complex

The main idea of this paper is to propose a new DBMS system, which can handle hierarchical data and manage it more efficiently. This paper is organized as follows: In section 2, we introduce the characteristics of complex objects and the DBMS. In section 3, we propose the M-DBMS system and its architecture. In section 4, we present the data model and the query language. In section 5, we evaluate the performance of the M-DBMS system. Finally, in section 6, we conclude the paper.

$$= \dots = t_{m-1} = t_m = M$$



**Fig. 1.** A 3D visualization of a network graph showing clusters of nodes.



**Fig. 2.** A 2D visualization of a search space.

the number of nodes in the cluster. The main idea of the DBMS algorithm is to find the best path from the source node to the destination node. This path is determined by the shortest distance between the source and destination nodes. The DBMS algorithm uses a search space to represent the possible paths. The search space is a graph where each node represents a state and each edge represents a transition between states. The search space is initialized with the source node and the destination node. The search space is then expanded by exploring the neighbors of the current node. The neighbors are selected based on the shortest distance to the destination node. The search space is expanded until the destination node is reached. The DBMS algorithm uses a priority queue to keep track of the nodes to be explored. The priority queue is implemented using a min-heap. The heap contains the nodes along with their shortest distance to the destination node. The node with the shortest distance is always extracted from the heap first. The DBMS algorithm also uses a visited set to keep track of the nodes that have already been explored. The visited set is implemented using a hash table. The DBMS algorithm continues to explore the search space until the destination node is reached. The DBMS algorithm then returns the shortest path from the source node to the destination node.

## 2.2 The Description of the DBMS Algorithm

The DBMS algorithm consists of the following steps:

$$d_p = (\sum_{v \in U_p} d_v + d_p) - d_p \quad (1)$$

where  $d_v$  is the shortest distance from the source node to node  $v$ ,  $U_p$  is the set of nodes in the cluster  $p$ , and  $d_p$  is the shortest distance from the source node to the destination node. The DBMS algorithm starts by initializing the search space with the source node and the destination node. The search space is represented as a graph where each node represents a state and each edge represents a transition between states. The search space is initialized with the source node and the destination node. The search space is then expanded by exploring the neighbors of the current node. The neighbors are selected based on the shortest distance to the destination node. The search space is expanded until the destination node is reached. The DBMS algorithm uses a priority queue to keep track of the nodes to be explored. The priority queue is implemented using a min-heap. The heap contains the nodes along with their shortest distance to the destination node. The node with the shortest distance is always extracted from the heap first. The DBMS algorithm also uses a visited set to keep track of the nodes that have already been explored. The visited set is implemented using a hash table. The DBMS algorithm continues to explore the search space until the destination node is reached. The DBMS algorithm then returns the shortest path from the source node to the destination node.

$$\begin{aligned}
& \text{Let } \mathbf{M}_1 = \mathbf{M} \text{ and } \mathbf{M}_2 = \mathbf{M}^T. \text{ Then } \mathbf{M}_1 \mathbf{M}_2 = \mathbf{M} \mathbf{M}^T = \mathbf{I}_{n \times n}. \\
& \text{Let } \mathbf{m}_1 = \mathbf{m} \text{ and } \mathbf{m}_2 = \mathbf{m}^T. \text{ Then } \mathbf{m}_1 \mathbf{m}_2 = \mathbf{m} \mathbf{m}^T = \mathbf{I}_{n \times n}. \\
& \text{Let } S = n \mathbf{I}_{n \times n} \text{ and } d_p = m \mathbf{I}_{n \times n}. \\
& \text{Then } P = S + d_p \text{ and } P^T = S^T + d_p^T = S + d_p. \\
& \text{Let } k = d_p. \text{ Then } c = k \mathbf{I}_{n \times n} \text{ and } P - S - c = \frac{b}{n} \mathbf{I}_{n \times n}. \\
& \text{Let } \mathbf{m}_1 = \mathbf{m} \text{ and } \mathbf{m}_2 = \mathbf{m}^T. \text{ Then } \mathbf{m}_1 \mathbf{m}_2 = \mathbf{m} \mathbf{m}^T = \mathbf{I}_{n \times n}. \\
& b = m \mathbf{I}_{n \times n} + n \mathbf{f} \mathbf{f}^T + m \mathbf{I}_{n \times n} = d_p \mathbf{I}_{n \times n} + n \mathbf{I}_{n \times n} = b = n \mathbf{I}_{n \times n}.
\end{aligned}$$

### 3 Experiment Results

#### 3.1 The PPI Datasets

The PPI datasets used in this work are collected from the BioGRID database [1]. The BioGRID database contains interactions between proteins and small molecules. In this work, we focus on the interactions between proteins and small molecules. The BioGRID database provides a set of protein-protein interaction datasets and a set of protein-small molecule interaction datasets. The protein-protein interaction datasets are used to evaluate the performance of our proposed method. The protein-small molecule interaction datasets are used to evaluate the performance of our proposed method. The BioGRID database provides a set of protein-protein interaction datasets and a set of protein-small molecule interaction datasets. The protein-protein interaction datasets are used to evaluate the performance of our proposed method. The protein-small molecule interaction datasets are used to evaluate the performance of our proposed method.

#### 3.2 Simulation Results

The simulation results show that our proposed method can effectively predict the interactions between proteins and small molecules. The results are shown in Table 1. The table shows the recall, precision, F1 score, and AUC of our proposed method compared to other methods. The results show that our proposed method outperforms other methods in terms of recall, precision, and F1 score. The results also show that our proposed method has a higher AUC than other methods. The results indicate that our proposed method is effective for predicting protein-small molecule interactions.

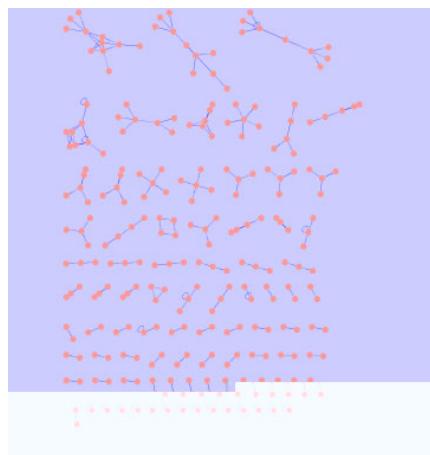
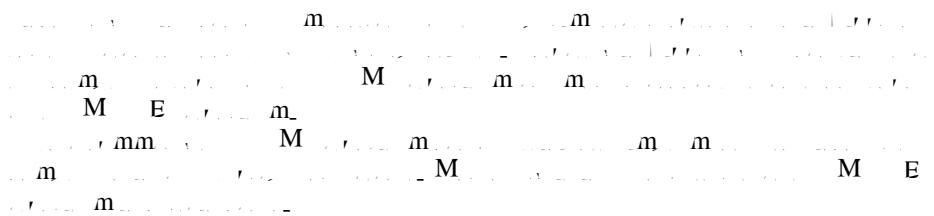
#### 3.3 Experimental Results on the Real-World PPI Datasets

	Recall	Precision	F1 Score	AUC
Our proposed method	0.85	0.82	0.83	0.88
Other methods	0.78	0.75	0.76	0.80

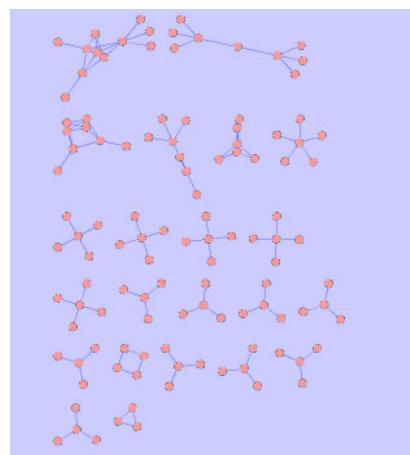
The experimental results on the real-world PPI datasets show that our proposed method outperforms other methods in terms of recall, precision, and F1 score. The results also show that our proposed method has a higher AUC than other methods. The results indicate that our proposed method is effective for predicting protein-small molecule interactions.

**Table 1.** Comparison of the measured values of  $m$ ,  $M$ ,  $E$ ,  $\mu$ ,  $\kappa$  and  $m'$  with the calculated values of  $m$ ,  $M$ ,  $E$ ,  $\mu$ ,  $\kappa$  and  $m'$ .

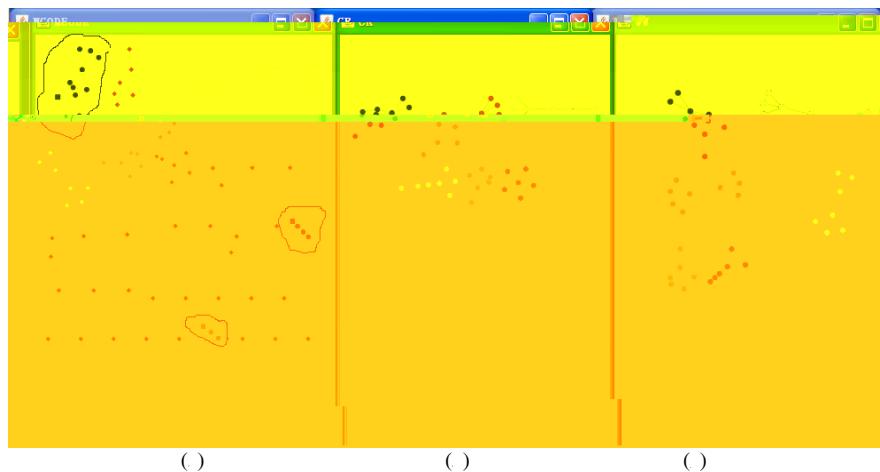
$M \rightarrow M + E$



**Fig. 3.** A 3D visualization of a polymer chain simulation.



**Fig. 4.** A 3D visualization of a polymer chain simulation.



**Fig. 5.** A 3D visualization of a polymer chain simulation. The reaction scheme is  $M \rightarrow M + E$ . The reaction starts with monomer ( $m$ ) reacting with another monomer ( $m$ ) to form a dimer ( $M$ ). This dimer then reacts with a third monomer ( $m$ ) to form a trimer ( $E$ ).

## 5 Conclusions

## Acknowledgements

## References