ISSN: 0067-2904

Topology-Based Modularity and Modularity Density for Detecting Protein Complexes: A Comparative Study

Safa Ahmed Abdulsahib*, Bara'a Ali Attea
.Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq

Received: 16/10/2022 Accepted: 17/4/2023 Published: 30/4/2024

Abstract

Binary relations or interactions among bio-entities, such as proteins, set up the essential part of any living biological system. Protein-protein interactions are usually structured in a graph data structure called "protein-protein interaction networks" (PPINs). Analysis of PPINs into complexes tries to lay out the significant knowledge needed to answer many unresolved questions, including how cells are organized and how proteins work. However, complex detection problems fall under the category of non-deterministic polynomial-time hard (NP-Hard) problems due to their computational complexity. To accommodate such combinatorial explosions, evolutionary algorithms (EAs) are proven effective alternatives to heuristics in solving NP-hard problems. The main aim of this study is to make a close examination of the performance of the EAs where modularity and modularity density are selected as two different objective functions. Topology-based modularity and topology-based modularity density are designed to examine the detection ability of the EAs and to compare their performance. To conduct the experiments, two *yeast Saccharomyces cerevisiae* PPINs are used and evaluated under nine evaluation metrics. The results reveal the potential impact of the topology-based modularity density to outperform the counterpart modularity functions in almost all evaluation metrics.

Keywords: Complex detection, Evolutionary algorithm, Modularity, Modularity density, Protein–protein interaction networks.

النمطية والكثافة النمطية المستندة إلى التخطيط البنيوي للكشف عن مجمعات البروتين: دراسة مقارنة

صفا أحمد عبدالصاحب*، براء علي عطية
قسم الحاسبات ، كلية العلوم ، جامعة بغداد ، بغداد ، العراق

الخلاصة:

العلاقات الثنائية أو التفاعلات بين المكونات الحيوية ، مثل البروتينات ، تنشئ الجزء الأساسي في أي نظام بيولوجي حي. عادة التخطيط البنيوي للتفاعلات البروتينية تمثل في رسم بياني ، يسمى شبكات تفاعل البروتين- بروتين (PPINs). محاولات تحليل شبكات PPINs إلى مركبات بروتينية والكشف عنها تهدف إلى طرح معلومات هامة للإجابة على العديد من الأسئلة التي لم يتم حلها ، وفهم كيفية تنظيم الخلايا وكيفية عمل البروتينات. ومع ذلك ، فإن مشكلة كشف المجمعات تندرج تحت فئة المشاكل الصعبة والتي تندرج تحت تصنيف (NP-hard) بسبب تعقيدها الحسابي. ولمعالجة مثل هكذا حسابات توافقية هائلة ، تم إثبات الخوارزميات التطورية (EAs) كبدائل فعالة للاستدلال في حلها. الهدف الرئيسي من هذه الدراسة هو تقييم

*Email: safa.a@sc.uobaghdad.edu.iq

لأداء وفعالية الخوارزمية التطورية EAs عند اختيار الوحدات النمطية والكثافة النمطية كدالة الهدف. تم تصميم النمطية والكثافة النمطية المستندة الى التخطيط البنيوي لفحص قدرة الاكتشاف لـ EAs ومقارنة أدائها. لإجراء التجارب ، تم استخدام شبكتي الخميرة Saccharomyces cerevisiae PPINs وتقييمها باستخدام تسعة مقاييس للتقييم. تكشف النتائج عن التأثير المحتمل لكثافة النمطية المستندة الى التخطيط البنيوي لتتفوق بالمقارنة على النمطية في جميع مقاييس التقييم تقريبًا.

1. Introduction

Proteins, the workhorses of the cell, execute many cellular functions by interacting with other protein partners through Protein-Protein Interactions (PPIs). Proteins interact with one another to perform a variety of vital biological tasks, for example, DNA transcription and duplication, DNA damage repair, the translation of mRNA, signal transduction, cell cycle, cell metabolism, etc. [1]. These proteins control and mediate many biological activities by regulating and supporting one another, forming protein complexes and functional modules and, thus, large protein-protein interaction networks (PPINs). For example, Figure 1 depicts two yeast *Saccharomyces cerevisiae* PPINs containing, respectively, 4687 interactions over 990 proteins (left) and 6993 interactions over 1443 proteins (right). In recent years, the computational analysis of PPINs has received a lot of attention because they accurately represent the complex interactions that take place between various cell components.

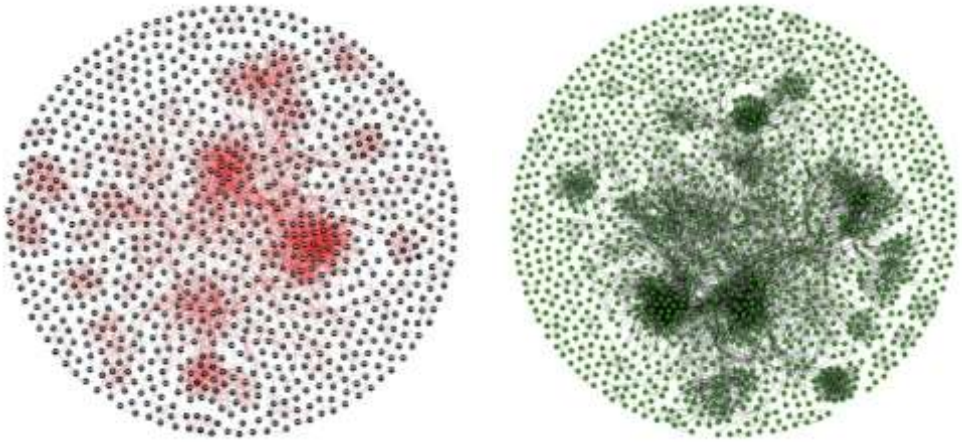


Figure 1: An illustrative example of two yeast *Saccharomyces cerevisiae* PPINs. Yeast-D1 (left) and Yeast-D2 (right).

Biologists have discovered that functionally related proteins tend to cluster together in a network of protein-protein or functional connections. Proteins that interact with each other frequently engage in the same biological processes. PPINs are specialized biological networks in which the cellular components under study are proteins. A PPIN, then, holds up information about the interactome of a given organism. However, not all biological processes are connected. It is worth noting that interacting proteins might be classified as "protein complexes" or "functional modules." Protein complex identification using PPINs can help with understanding the mechanisms that control cell life, describing the evolutionary orthology signal, predicting the biological functions of unknown proteins, and, most importantly, for therapeutic applications. As a result, protein complex prediction is essential, and a race for new high-performance clustering methods to uncover and analyze these networks has started.

A PPIN is an undirected graph in which nodes represent proteins and edges represent pairwise interactions between proteins. An interaction is defined as a “within-complex interaction” or “intra-connection” when both protein partners map into the same protein complex. Otherwise, when an interaction between two proteins maps into two different complexes, it is termed a "between-complex" interaction or "inter-connection." Figure 2 graphically maps one protein example. The proteins and their interactions inside the dashed circle in Figure 2 represent one protein complex.

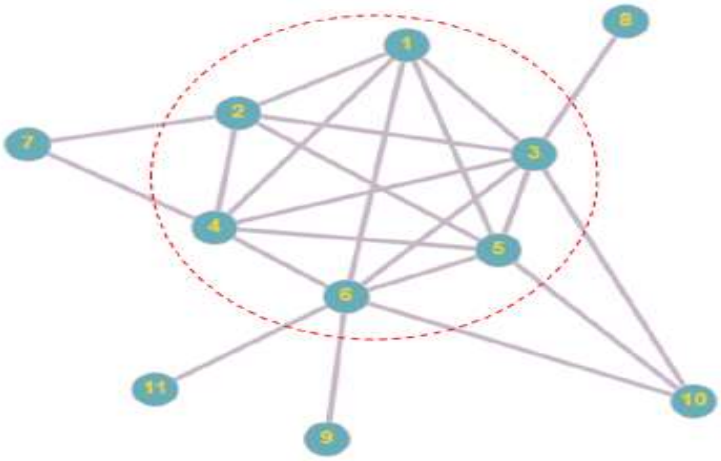


Figure 2: An illustrative example depicting the problem of detecting protein complexes from a PPI network. Here the nodes within the dashed circle represent proteins within one detected complex. Intra-connections are the edges inside the dashed circle. Inter-connections are all the others.

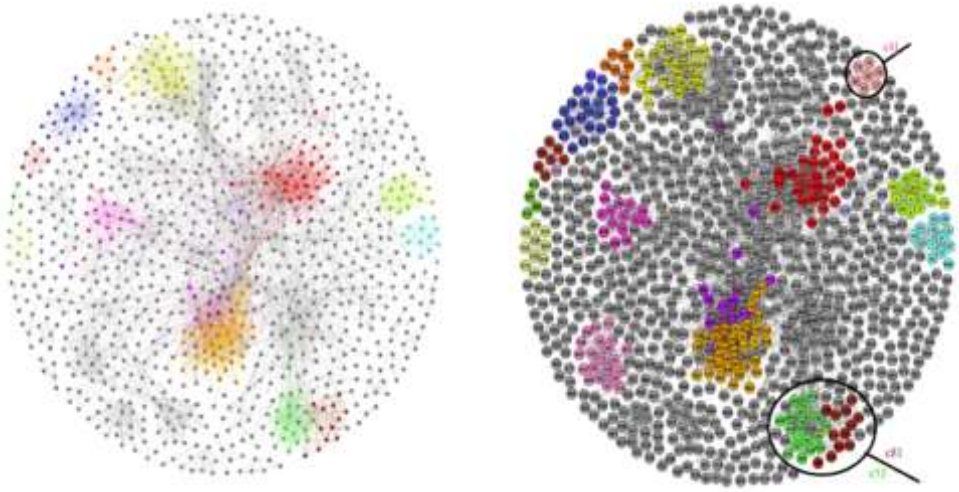


Figure 3: An illustrative example depicting a yeast *Saccharomyces cerevisiae* PPIN from Yeast Protein Database (left) with 990 proteins and 4687 interactions. The PPIN is decomposed into 78 complexes (right) of different sizes.

However, proteins are assembled into many potential complexes with different sizes, each of which performs a distinct function in the cell. In other words, interactions of proteins in PPINs often include interactions between protein assemblies or protein complexes. Proteins that interact with each other are frequently involved in the same biological processes or might be linked to certain biological activities. As an illustrative example, consider Figure 3. In the figure, the yeast *Saccharomyces cerevisiae* PPIN (left) contains 990 proteins with 4687

interactions. These proteins are from the Yeast Protein Database [2]. On the basis of the true 81 known complexes annotated in the Munich Information Center for Protein Sequences (MIPS) database for genomes and protein sequences, the PPIN is decomposed into 78 complexes (right). Three complexes are zoomed out in Figure 4, and further, protein #25 (YGL066W) is zoomed out in Figure 5 with all its intra- and inter-connections.

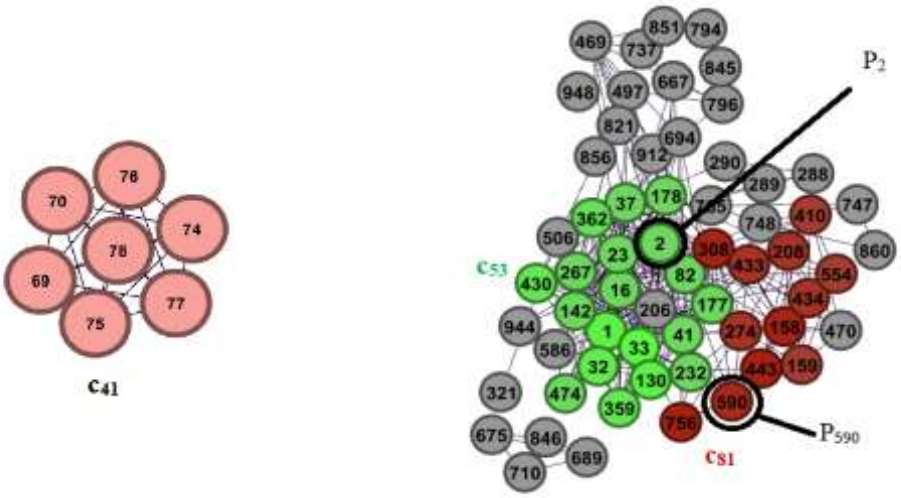


Figure 4: Three complexes are zoomed out from the yeast *Saccharomyces cerevisiae* PPIN.

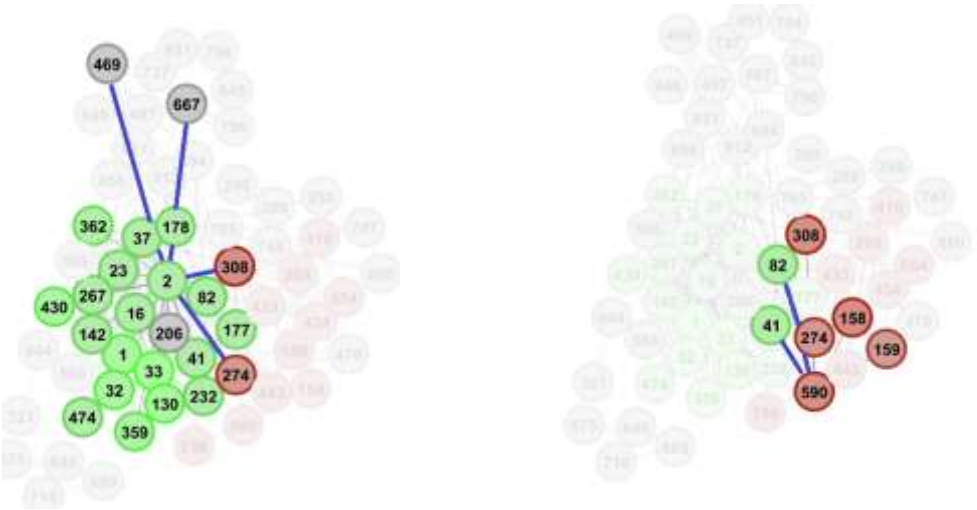


Figure 5: proteins #2 (YHL030W) and #590 (YBL041W) of the yeast *Saccharomyces cerevisiae* PPIN (shown in Figure 3) are zoomed out with all their intra-connections and inter-connections.

Complex networks are useful formalisms for describing the interactions between items in many real-world systems. In the field of complex networks, the challenge of identifying protein complexes can be seen as a community detection (CD) problem [3]. The CD problem is a community discovery problem that asks for a data analysis method to reveal the hidden structure of a large-scale networked dataset into distinct and compact clusters, where the number and size of the subgroups are unknown. However, the CD problem is proved to be a non-deterministic polynomial-time hard (NP-hard) problem. Shortly speaking, the CD

problem is an optimization problem to identify the internal grouping of a huge networked dataset into clusters [4]. Several researchers from different fields, such as physics, statistics, and data mining, have come up with ideas for using artificially intelligent (AI) tools to better understand and discover communities in complex networks.

The main aim of this paper is to develop an evolutionary algorithm (EA) with modularity density and a topological-based mutation operator for detecting protein complexes in PPI networks. The performance of the proposed algorithm is evaluated against one of the state-of-the-art complex detection algorithms, particularly EA with modularity and a topological-based mutation operator. In other words, this study attempts to make a close examination of the performance of the EAs when modularity and modularity density are selected as two different objective functions.

The remainder of this paper is organized as follows: The related strategies for heuristic and evolutionary-based complex detection methods proposed in the literature are presented. This is followed by the main concepts associated with the complex detection problem. This includes the meaning of the interactome and interaction graph, the well-known modularity and modularity density, and the canonical framework for the evolutionary algorithms. In Section 4, we present the main characteristics and formulations of the proposed evolutionary-based complex detection algorithm in two versions. The results and discussions are provided in Section 5. Finally, Section 6 presents the conclusion of the current work and further ramifications of the work.

2. Related works

The two main streams for complex detection algorithms provided in the literature are heuristic-based complex detection algorithms (hCDs) and evolutionary-based complex detection algorithms (ECDs). However, due to problem complexity, the literature proves that ECDs are more powerful than their counterpart hCDs.

Traditionally, local-based search algorithms are characterized by heuristic methods. All these approaches have a common iterative framework operating on a single partial or complete solution. The general framework of the heuristic-based complex detection algorithms is composed of a solution perturbation operator and a quality or objective function. At each iteration, the objective function is used to evaluate how the perturbed solution improves. Different variants of perturbation operators and objective functions are suggested in the literature. Examples of some well-known heuristic-based complex detection algorithms are: Molecular Complex Detection (MCODE) [4], purification of the bait proteins [5], dense-neighborhood extraction using connectivity and confidence features (DECAFF) [6], repeated random walks (RRW) [7], clustering-based on maximal cliques (CMC) [8], and hierarchical link clustering [9, 10]. Although these approaches have been widely adopted, they have limited accuracy and inferior performance as compared with evolutionary-based complex detection algorithms.

The beginning efforts to bring up the evolutionary-based complex detection methods were returned to Pizzuti and Rombo [3, 11], Bandyopadhyay et al. [12], and Ray et al. [13]. Pizzuti and Rombo [3, 11] explored the ability of single-objective genetic algorithms (GAs) to predict protein complexes in PPINs when different topological-based quality functions are employed as fitness functions. These include modularity (Q), community score (CS), conductance (CO), normalized cut (NC), internal density (ID), expansion (EX), and cut ratio (CR). All these

quality functions, but Q, explicitly define, with different formulations, the intra- and inter-complex scores.

The de-facto formulation of the modularity (Q), on the other hand, handles only the intra-complex structure score. The definitions of all these quality functions were based on the topological structure of the PPIN being clustered. However, all other components of the GA proposed in [3, 11], including selection, crossover, and mutation operators, were designed in their more or less canonical form.

Bandyopadhyay et al. and Ray et al. [12, 13] in 2015 and 2016, on the other hand, investigated the performance of the multi-objective genetic algorithm, the so-called "non-dominated sorting genetic algorithm II" (NSGA-II), for solving complex detection problems. Again, they adopted the canonical design for the main components of the NSGA-II. The topological features of the network, however, were reflected in their multi-objective formulations in terms of both intra- and inter-complex structure.

Thereafter, the starting effort to design a topological-based component in the framework of the evolutionary-based complex detection algorithms was suggested by Attea and Abdullah [14] in 2018. They proposed a topological-based mutation operator to work at the protein level and at the complex level. At the protein level, they proposed a migration operator to replace the traditional mutation operator used in [3, 11, 12, 13]. On the other hand, at the complex level, they proposed the so-called protein complex attraction and repulsion operator. They proved that the topological-based design of the mutation operator all right improves the performance of both the single-objective GA of Pizzuti and Rombo in [3, 11] and the NSGA-II of Bandyopadhyay et al. and Ray et al. in [12, 13]. They demonstrated that this improvement is the outcome of the positive collaboration between the topological-based design of the algorithm with the single and multi-objective quality functions, i.e., Q, CS, CO, NC, ID, EX, and CR, adopted in [3, 11], and the multi-objective intra- and inter-complex structure suggested in [12, 13].

Further, Attea and Abdullah [14] supported their findings by defining a multi-objective quality function reflecting a topological intra- and inter-complex structure. Again, they proved their ability to reach a promising collaboration between the suggested multi-objective function and the topological-based mutation operator.

The next effort came in 2018 and 2019 by Abdulateef et al. [15, 16] to support the claim that the topological-based design for the component(s) of the evolutionary-based complex detection algorithms would hopefully improve their detection ability to hit more correct complex structures. They developed a heuristic mutation operator based on the topological properties of the tested PPIN. They also reported improved performance for the single-objective EA and the multi-objective EA to uncover more correct complexes from the tested PPINs.

3. Background

3.1 Interactome and interaction graph

The term "interactome" sets out the set of all molecular interactions in cells, especially in the context of protein-protein interactions. In other words, the interactome is a global description obtained by various methods (mass spectrometry, two-hybrid methods, and genetic studies) to estimate the whole network of protein interactions for a given organism. For example, in 2002, the yeast interactome was estimated to contain up to 80,000 potential

interactions. These interactions are essential for almost all cellular processes, so the full representation of the interaction collection is needed to understand the cell molecular machinery at the system biology level [17].

PPIN is generally represented as an undirected interaction graph, $N(P, E)$, where n nodes $P = \{p_1, p_2, \dots, p_n\}$ represent n proteins and m edges $E = \{e_1, e_2, \dots, e_m\}$ represent pairwise interactions. A graph N is typically represented as a symmetric adjacency matrix, $A = [a_{ij}]^{n \times n}$. If proteins p_i and p_j interact, both entries a_{ij} and a_{ji} of A are assigned with 1, otherwise, both entries are assigned with 0. Figure 6 depicts an illustrative example of eight PPIs from the yeast *Saccharomyces cerevisiae* PPIN (Figure 3), with 990 proteins and 4687 interactions. In Figure 6, a total of 22 interactions out of 4687 interactions are mapped to the adjacency matrix. In other words, 44 entries in the adjacency matrix are set to 1. Thus, for the whole yeast PPIN network, there should be 4687×2 entries set to 1 in the counterpart adjacency matrix.

Adjacency matrix A can be represented in list notation by a collection $L = \{l_1, l_2, \dots, l_n\}$ of n adjacency lists, with one list l_i for each protein $p_i \in P$ aggregating all 1 entries in row i. As a result, $|li| = \sum_{j=1}^{n}(a_{ij})$ and $|L| = \sum_{i=1}^{n}|l_i|$. Mathematically, n is the cardinality of N, $|li|$ is the degree of vertex p_i, and $|L|$ is the volume of N.

Now, assume Ω is the space of all the decomposition solutions divided into various sized complexes. Any clustering algorithm tries to partition the space of A into a set $C = \{c_1, c_2, \dots, c_K\}$ of K complexes. It is widely assumed that a protein $p_i \in c_k$ should have more internal connections $in(p_i)$ than external ones $out(p_i)$. Formally speaking, $in(p_i) = \sum_{p_j \in c_k} a_{ij}$ and $out(p_i) = \sum_{p_j \notin c_k} a_{ij}$ respectively, the number of intra-connections and inter-connections of node p_i which belongs to cluster c_k (i.e., $|l_i| = in(p_i) + out(p_i)$).

	ProteinLabel	YDL097C	YHL030W	...	YER157W	YML071C	YPR105C	YNL051W	...	YGL223C	YGR120C	YNL041C	YGL005C	...	YLL024C	...	YNL082W
ProteinLabel	P	1	2	...	107	108	109	110	...	415	416	417	418	...	604	...	990
YDL097C	1	0	1	...	0	0	0	0	...	0	0	0	0	...	0	...	0
YHL030W	2	1	0	...	0	0	0	0	...	0	0	0	0	...	0	...	0
...	...	...		...	...	...	...	...	...	...	...	...	...	...	...	...	...
YER157W	107	0	0	0	0	1	1	1	...	1	1	1	1	...	0	0	0
YML071C	108	0	0	0	1	0	1	1	...	1	1	1	1	...	0	0	...
YPR105C	109	0	0	0	1	1	0	1	...	1	1	1	1	...	0	0	0
YNL051W	110	0	0	0	1	1	1	0	...	1	1	1	1	...	0	0	0
...	...	...		...	...	...	...	...	...	...	...	...	...	...	...	...	...
YGL223C	415	0	0	0	1	1	1	1	...	0	0	0	0	0	0	0	0
YGR120C	416	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
YNL041C	417	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
YGL005C	418	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
...	...	...		...	...	...	...	...	...	...	...	...	...	...	...	...	...
YLL024C	604	0	0	0	0	0	0	0	...	0	0	0	0	0	0	...	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
YNL082W	990	0	0	0	0	0	0	0	0	0	0	0	0	...	1	...	0

Figure 6: A sample graph depicting a protein complex. On the left, eight nodes, numbered {107, 108, 109, 110, 415, 416, 417, 418}, are interacted via edges. An adjacency, symmetric, matrix, on the right, maps the complex, where "1" indicates the protein pair interaction, and "0" indicates no protein pair interaction. All diagonal elements are assigned with “0”.

3.2 Modularity and modularity density

Many researchers investigated the topological properties of protein complexes in PPINs to address complex detection problems and discovered that many protein complexes are densely

connected among themselves but sparsely connected with the rest of the PPI network. This feature is normally computed using modularity-based methods to compute the modular structure of different complex networks, including PPINs.

In the context of community detection, the choice of a good fitness function is another critical step for obtaining optimal or near-optimal solutions, which was first introduced by Newman and Girvan in [18] using the modularity metric. Modularity, mathematically denoted by Q (Eq. 1) is the first and most well-known quality function used to formally define a community (or a complex) structure. It is a single-objective function reflecting the score of the internal structure of the community. For a complex structure $C = \{c_1, c_2, \dots, c_K\}$ of K modules or complexes, the modularity is defined as:

$$Q = \frac{1}{2m}\sum_{ij}\left(a_{ij} - \frac{d_i d_j}{2m}\right)\delta_{c_i,c_j} \quad (1)$$

Where m is the number of edges in the network, d_i and d_j are the degrees of nodes p_i and p_j, and δ_{c_i,c_j} is the Kronecker delta function, which returns *one* if p_i and p_j are in the same community (i.e. $c_i = c_j$) and *zero* otherwise (i.e. $c_i \neq c_j$). Formally:

$$\delta_{c_i,c_j} = \begin{cases} 1 & c_i = c_j \\ 0 & c_i \neq c_j \end{cases} \quad (2)$$

Assume c_1 and c_2 to be two different complexes, and let m_i be the number of connections for protein $p_i \in c_k$. Further, let $L(c_1, c_2) = \sum_{i\epsilon\, c_1, j\epsilon c_2} a_{ij}$ denotes the set of connections between the proteins of complex c_1 and complex c_2. Then, Eq. (2) can be expressed in a more convenient form as:

$$Q = \sum_{i=1}^{K}\left[\frac{L(c_i, c_i))}{m} - \left(\frac{m_i}{2m}\right)^2\right] \quad (3)$$

For a given complex set structure, the basic idea of modularity is to maximize the number of intra-connections within each complex in the set. This is achieved in Q function by calculating the difference between the actual fraction of interactions within a complex and the expected number of interactions. Note that the expected number of interactions represents the fraction of intra-edges falling in an equivalent network with the same number of complexes but with a random distribution of connections in the complexes. If the fraction of intra-connections is no better than the random distribution, then Q approaches its minimum value, i.e., 0. On the other hand, Q approaches its maximum value (i.e., 1) while meeting strong complex structures.

The main weakness of the formulation for modularity is that it may favor partitions with a few large modules rather than many small modules [19]. Modularity, thus, experiences a resolution limit problem where many small complexes (with respect to the whole network) may remain undetected even when they are well-defined, strong complexes, like cliques. This resolution limit problem often results from the comparison between the number of interconnected complexes and the total number of interconnections in the PPIN. It does not take into account the size of the network (i.e., n) as well as how many nodes there are in a single complex. Additionally, Eq. 3 does not account for the size of complexes. This indicates that partitions derived from modularity maximization could not identify small groups hidden within larger complexes with higher modularity values.

To address this issue and avoid the resolution limit problem of Q, another variant called the modularity density (QD) function is proposed in [20]. It is based on the average degree or the density of sub-graphs (i.e., complexes). It measures the ratio of the difference between the

internal and external degrees, which corresponds to the size of the complex. It is formulated as follows:

$$QD = \sum_{i=1}^{K} \frac{L(c_i,c_i)-L(c_i,c_j)}{|c_i|} \quad (4)$$

3.3 Evolutionary algorithms (the canonical framework):

It has been recognized that EAs can produce acceptable or near-optimal solutions to a broad class of complex real-world optimization problems. Among the metaheuristic algorithms, evolutionary algorithms (EAs) are getting increased attention while supporting powerful performance in solving many real-world optimization problems. The key issues while considering the design of a new competitive EA for solving a particular real-world problem are 1) to formulate the problem with more robust (mostly multi-objective) optimization models, and 2) to develop a cross-fertilization mechanism for combining heuristic operators (that are tailored specifically to fit the problem) with the EA framework [21]. Several studies followed this rule in designing competitive EAs for solving different NP-hard problems [22–26].

The general framework of the EAs is an iterative population-based search model working on a set of individual solutions (a small subset of the whole search space of the problem). Generally, each individual chromosome in the population has a genotype and phenotype representation, and thus, a smooth genotype-phenotype mapping should be formulated. The genotype (or low-level) representation is designed as an EA-aware or algorithm-directed representation to enable the evolutionary operators to work on. It decomposes the decision parameters of the problem into their encoded genes. The phenotype representation, on the other hand, is the final high-level layout for the solution.

The general layout of the algorithm is composed of a set $\Phi = \{\Phi_s, \Phi_\times, \Phi_m\}$ of three main evolutionary operators working iteratively, generation by generation. These operators are selection, crossover, and mutation [21]. Each operator has a distinct role; however, the collaborative endeavor for these evolutionary operators is to evolve a population of randomly initialized solutions toward a promising set of solutions in the problem's search space. To this end, each operator adopts the exploitation-exploration search mechanism in a different mode. Some operators emphasize exploitation at the expense of exploration, while others stress exploration rather than exploitation. However, an appropriate balance between exploitation and exploration eventually has a positive impact on the performance of the EA. The generations of the EA work end when a stopping criterion is set. Various forms of the termination condition could be designed. For example, the search for the EA may end when stagnation occurs in the population and the solutions lack any further or obvious improvement. However, the more usual way to stop the EA course is after reaching a predetermined number of generations.

The main role of the selection operator (,Φ-s.) is to prepare the mating pool of parent solutions. It works as a high-level bias (or preference) operator to prefer some solutions over others according to their quality values. The quality of each solution is determined by the objective or fitness function, whether single- or multi-objective. It operates on different solutions, and copies of the winning solutions will form the mating pool. Crossover and mutation operators, on the other hand, are involved as perturbation operators to modify complete as well as part of solutions, hopefully towards better, ones. The crossover operator (,Φ-×.), with different sorts of design, works as a middle-level operator. It aims to mix different parts (i.e., groups of genes) of two parent solutions to generate a different child

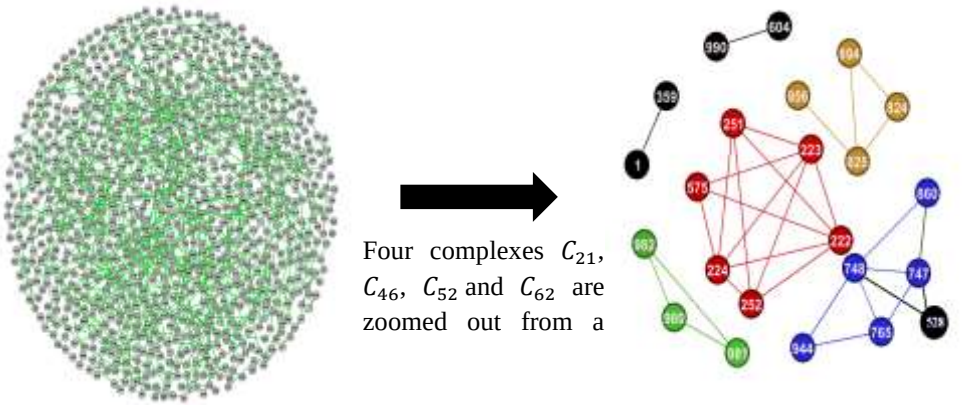
solution that inherits good traits from the crossed parents. The mutation operator (,Φ-m.), on the other hand, works at the low-level operator. It aims to modify different, but single, genes of single solutions to generate new nearby children's solutions. The logical course of the crossover operator and the mutation operator labors under probabilistic control, and they are generally set as opposite. Normally, the crossover operator works with a high probability (,-×.) to cross two selected parents, while mutation, in its general form, works with a low probability (,-m.) to alter a few parts of the solution.

4. The proposed evolutionary-based complex detection algorithm

The computational complexity of the complex detection problem in PPINs is proved to be NP-hard, and thus, it has been modeled as an optimization problem in several efficacious single- and multi-objective evolutionary algorithms [21]. The general framework for an evolutionary-based complex detection $ECD: \mathbf{I} \rightarrow \mathbf{I}$ is an iterated transformation function that starts with an initial population $\mathbf{I} = \{I_1, I_2, \dots, I_N\}$ of N encoded (i.e. genotype) solutions. These solutions are generated randomly from the whole search space of the problem Ω. The encoding scheme is locus-based adjacency representation [27]. An individual $I_{1 \le i \le n} \in \mathbf{I}$ can, thus, be represented as a vector (Eq. 5) of n decision making parameters for the n proteins in the PPIN.

$$I_i = (I_{i,1}, I_{i,2}, \dots, I_{i,n}) \quad (5)$$

Here, each locus j (plural loci) is defined by its index j corresponding to protein p_j and its allele value $I_{i,j}$. In locus-based representation, $I_{i,j}$ refers to a neighbor protein k with which protein j can coexist in the same complex. The global locus-based initialization process designates the direct neighbors of j in A, i.e. $a_{jk} = 1$, to be the possible allele values at locus$j$. The decoding function $\Gamma: I \rightarrow C$, then, maps a genotypic solution I into its corresponding set of complexes $C = \{c_1, c_2, \dots, c_K\}$ of K complexes. Note that for any two solutions, I_i and I_j in the population **I,** K_i and K_j do not necessarily have to be equivalent. In other words, their phenotype solutions $C_i = \{c_1, c_2, \dots, c_{K_i}\}$ and $C_j = \{c_1, c_2, \dots, c_{K_j}\}$ could be dissimilar. Figure 7 depicts an illustrative example of the genotype and phenotype of a random individual for the yeast Saccharomyces cerevisiae PPIN of 990 proteins. The individual is randomly initialized with random neighbor-proteins for the 990 proteins. In the figure, four complexes from the whole phenotype are also zoomed out.



Protien ID	1	...	222	223	224	...	251	252	...	575	...	747	748	...	765	...	824	825	...	860	...	894	944	...	956	...	980	981	982	...	990
allele	359	...	223	251	251	...	252	222	...	224	...	860	944	...	747	...	825	894	...	747	...	825	765	...	825	...	981	980	980	...	604
CmplxID	1	...	21	21	21	...	21	21	...	21	...	46	46	...	46	...	52	52	...	46	...	52	46	...	52	...	62	62	62	...	36

Figure 7: Example of a randomly initialized individual for the yeast *Saccharomyces*

cerevisiae PPIN of 990 proteins. Four complexes; c_{21}, c_{46}, c_{52} and c_{62} are zoomed out in the Top-right. Bottom: the genotype of a random individual. The top vector refers to protein identifiers. The middle vector assigns random neighbor-protein alleles for the corresponding protein identifiers, while the bottom vector represents the phenotype in terms of complex identifiers.

Two quantitative functions; modularity (Q in Eq. 3) and modularity density (QD in Eq. 4), are adopted as objective functions to quantitatively measure the quality of the generated complex partitions. Based on the quality values of the solutions, parent solutions are then selected using binary tournament selection Φ_s: $(I_1, \Theta_1) \times (I_2, \Theta_2) \rightarrow I$, where Θ is the quality value of the solution computed by Q or QD.
For the work of the remaining evolutionary operators, the selection operator prepares a pool of N pairs of parents. Then, for a pair of parent solutions, I_1 and I_2, uniform crossover operator ($\Phi_\times$: $I_1 \times I_2 \times p_\times \rightarrow I$) is adopted to evenly mix their n decision making parameters. Crossover occurs for the parents pair if the probability of crossing p is greater than the probability of crossover, $p_\times$. Here, $p_\times$is set to 0.8.

$$\forall i \in \{1,2,\dots,N\} \wedge \forall j \in \{1,2,\dots,n\}$$
$$I_{i,j} = \begin{cases} I_{1,j} \; if \; rand \leq 0.5 \\ I_{2,j} \; otherwise \end{cases} \quad (6)$$

The mutation operator, on the other hand, is adopted to either change the allele value (i.e., the neighbor of the mutated protein $I_{i,j}$) or the complex-belonging of the mutated protein (i.e., j) itself. The strength, or probability, of the mutation operator (,-m.) is usually set low to let the mutation operator work as a background operator. The first design (Eq. 7) is the so-called "traditional neighbor-based mutation" (,Φ-m.:I×,-m.→I) of Pizzuti and Rombo [3], which works on the genotype representation. This in turn will alter the phenotypic structure of the individual. The traditional neighbor-based mutation Φ_m works as follows: It considers all n loci for all N individual solutions in the population. If a mutation operator is determined on a given locus j for an individual I_i (i.e. when a uniform random number, $rand$, be less than or equal p_m), then the allele value $I_{i,j}$ is exchanged by another value j', such that j' is one of the direct neighbors of protein j. This can be expressed as:

$$\forall i \in \{1,2,\dots,N\} \wedge \forall j \in \{1,2,\dots,n\}$$
$$I_{i,j} = \begin{cases} j' \;\; | a_{jj'} = 1 \; if \; rand \leq p_m \\ I_{i,j} \qquad\qquad otherwise \end{cases} \quad (7)$$

The second design (Eq. 8), on the other hand, is named the "migration operator" (,Φ-top−m.:I×,-m.→I) and it directly operates on the phenotype (or the topological) representation of the PPIN [14]. When the mutation operator is activated on protein j for an individual ,I-i., it will change the complex of this protein to a new complex, say ,c-k., where it could maintain there the maximum function homogeneity (i.e. ,l∈,c-k.-,a-jl.. has its maximum value).

$$\forall i \in \{1,2,\dots,N\} \wedge \forall j \in \{1,2,\dots,n\}$$
$$I_{i,j} = \begin{cases} j' \;\; | j' \in c_k \wedge argmax_{c_k \in \mathcal{C}} (\sum_{l \in c_k} a_{jl}) \; if \; rand \leq p_m \\ I_{i,j} \qquad\qquad\qquad\qquad otherwise \end{cases} \quad (8)$$

This, in the opposite order, will imply a modification to the genotype representation. The general framework for the proposed EA with modularity density and a topology-based mutation operator is sketched out in Algorithm 1 and noted as EA_{Top-QD}.

Algorithm 1: The general framework for the proposed EA_{Top-QD}

Input: N, Φ_s, $\Phi_\times$, Φ_m, $p_\times$, p_m
Output: $C = \{c_1, c_2, \dots, c_K\}$ with maximum QD
begin
$t \leftarrow 0$; // initial generation
$MaxGen \leftarrow 100$; // maximum number of generations
initialize $\mathbf{I}^t \leftarrow \{I_1^t, I_2^t, \dots, I_N^t\}$;
decode: $\Gamma(I_{1\le i\le N}^t)$: $\{C_1^t, C_2^t, \dots, C_N^t\}$; // where $C_i^t = \{c_1, c_2, \dots, c_{K_i}\}$
evaluate: $\mathbf{I}^t$: $\{QD(C_1^t), QD(C_2^t), \dots, QD(C_N^t)\}$;
while $(t \neq MaxGen)$ **do**
  select Φ_s: $\mathbf{I}^{t+1} \leftarrow \{(I_1, QD_1)_{1\le i\le N} \times (I_2, QD_2)_{1\le i\le N}\}$;
  recombine $\Phi_\times$: $\mathbf{I}^{t+1} \leftarrow \{(I_1 \times I_2 \times p_\times)_{1\le i\le N}\}$;
  mutate Φ_m: $\mathbf{I}^{t+1} \leftarrow \{I_{1\le i\le N}^{t+1}, p_m\}$;
  decode: $\Gamma(I_{1\le i\le N}^{t+1})$: $\{C_1^{t+1}, C_2^{t+1}, \dots, C_N^{t+1}\}$; // where $C_i^{t+1} = \{c_1, c_2, \dots, c_{K_i}\}$
  evaluate: $\mathbf{I}^{t+1}$: $\{QD(C_1^{t+1}), QD(C_2^{t+1}), \dots, QD(C_N^{t+!})\}$;
  $t \leftarrow t + 1$;
end while;
return $I_{1\le i\le N}^t$ with **best** $C = \{c_1, c_2, \dots, c_K\}$ with maximum QD**;**
end

5. Results and discussions

5.1 PPIN dataset and reference dataset

To evaluate the performance of the proposed EA with modularity density (QD) against the canonical EA with modularity (Q), two yeast Saccharomyces cerevisiae PPINs (refer to Figure 1) are used in the experiments [28]. The filtered versions of these networks were prepared in [29]. The first PPIN (Yeast-D1) has 4687 interactions over 990 proteins. Only 28 proteins have single interactions, while the remaining proteins have two or more interactions, for an average of 9.4687 interactions per protein. However, the highest number of interactions is 52, which are recognized by the protein "YCR057C" (#170). The second PPIN (Yeast-D2) contains 1443 proteins paired with 6993 interactions. Here, 92 proteins have single interactions, while the remaining proteins have two or more interactions. The maximum number of interactions for a single protein is 59 interactions, which are paired by protein "YHR052W" (#339). The average number of interactions per protein in PPINs is 9.6923.

Two reference sets, as identified by $Cmplx\_D1$ and $Cmplx\ \_D2$, are utilized to validate the quality of the detected complexes over, respectively, Yeast-D1 and Yeast-D2. These complex datasets were created from the Munich Information Center for Protein Sequences (MIPS) genome and protein sequence database. $Cmplx\_D1$ dataset contains 81 golden or true complexes with different sizes ranging from 6 yeast proteins up to 38 yeast proteins. However, out of all 990 proteins in the first PPIN (i.e., yeast-D1), 701 proteins are spread out among only 78 true complexes with other unknown yeast proteins. This leaves the remaining three true complexes free of the known proteins in yeast D1. In short, we found that 701 yeast proteins from Yeast-D1 are distributed over 78 true complexes, with sizes ranging from only one singular protein up to 34 proteins, with an average of 8.9872 proteins per true complex. Thus, the raw representation of $Cmplx\_D1$ and its representation concerning Yeast-D1 can be formally stated in Eqs. 9 and 10, respectively:

$$S^* = \{S_1, S_2, \dots, S_{81}\},\ |S^*| = K^* = 81 \text{ and } \forall_{S_i \in S^*}\ 6 \le |S_i| \le\ 38 \quad (9)$$

$S^* = \{S_1, S_2, \dots, S_{78}\}$, $|S^*| = K^* = 78$ and $\forall_{S_i \in S^*}\ 1 \leq |S_i| \leq\ 34$ (10)

$Cmplx\_D2$ dataset, on the other hand, contains 162 true complexes (Eq. 11), with sizes ranging from 4 yeast proteins up to 266 yeast proteins. However, only 680 yeast proteins in Yeast-D2 are distributed over 150 true complexes (Eq. 13), leaving the remaining 12 true complexes to contain other yeast proteins. Then, the 150 true complexes contain Yeast-D2 proteins with sizes ranging from singular proteins up to 136 proteins, with an average of $\frac{680}{150} = 4.5333$ yeast proteins per true complex. These can be stated mathematically as:

$S^* = \{S_1, S_2, \dots, S_{162}\}$, $|S^*| = K^* = 162$ and $\forall_{S_i \in S^*}\ 4 \leq |S_i| \leq\ 266$ (11)

$S^* = \{S_1, S_2, \dots, S_{150}\}$, $|S^*| = K^* = 150$ and $\forall_{S_i \in S^*}\ 1 \leq |S_i| \leq\ 136$ (12)

Now, for validation, we can say that a predicted complex C_i *matches* (Eq. 13) one of the true complexes from the benchmark set in S^* (say S_j), if the proteins of both complexes overlap or intersect with overlapping score (Eq. 14) that is equal to or greater than a specified threshold σ_{OS} [30].

$$match\left(C_i, S_j\right) = \begin{cases} 1 & if\ \ OS(C_i, S_j) \geq \sigma_{OS} \\ 0 & otherwisex \end{cases} \quad (13)$$

$$OS(C_i, S_j) = \frac{|C_i \cap S_j|^2}{|C_i||S_j|} \quad (14)$$

where $\left|C_i \cap\ S_j\right|$ refers to the number of common proteins to both a predicted complex C_i and a true standard complex S_j. $|C_i|$ and $\left|S_j\right|$ refer to the number of proteins in C_i and S_j, respectively. In the experiments, we set OS to range from 0.1 to 0.8, in an incremental step of 0.05.

5.2 Algorithm settings

In summary, four algorithms are evaluated in the experimental investigations. These are the canonical EA with the modularity function and the neighbor-based mutation operator of Pizzuti and Rombo [3], the canonical EA with the modularity function and the topological-based mutation operator of Attea and Abdullah [14, 20], the proposed EA with the modularity density function and the neighbor-based mutation operator, and the proposed EA with the modularity density function and the topological-based mutation operator. These algorithms are noted in the results as EA_Q [3], EA_{Top-Q} [14], EA_{QD}, and EA_{Top-QD}, respectively. Each algorithm is endorsed for evaluation under a simulation of 30 different runs, and each is initialized with a random population of 100 individual genotype solutions. The evolutionary process of each algorithm is allowed to continue for 100 generations. The average of the 30 different runs (in terms of the best solution obtained) is reported for each algorithm. The best solution for each algorithm is recognized by its objective value (Q or QD). Thus, the best solution for the canonical EA with modularity and the topological-based EA with modularity (i.e. EA_Q and EA_{Top-Q}) is the solution with the largest value of Q. On the other hand, the best solution for the canonical EA with modularity density and the topological-based EA with modularity density (i.e. EA_{QD}, and EA_{Top-QD}) is the solution with the largest value of QD.

5.3 Evaluation metrics

The percentage of the true benchmark complexes that match (with respect to the overlapping score) any of the detected complexes is known as $recall$. On the other hand, $precision$ refers to the fraction of the detected complexes that match any of the true complexes. The F score, then, represents the harmonic mean of both recall and precision.

$$recall = \frac{|S_i|S_i \in S^* \wedge \exists C_j \in C \rightarrow match(S_i, C_j)}{K^*} \quad (15)$$

$$precision = \frac{|C_i|C_i \in C \wedge \exists S_j \in S^* \rightarrow match(C_i, S_j)}{K} \quad (16)$$

$$F = \frac{2 \times recall \times precision}{recall + precision} \quad (17)$$

While $recall$ and $precision$ measure the cumulative quality of the detected complexes for an algorithm prediction at the complex level, $recall_N$ and $precision_N$ can estimate the detection accuracy at the protein level [29]. Finally, F_N score imitates the F score but at the protein level for both $recall_N$ and $precision_N$ measures.

$$recall_N = \frac{\sum_{i=1}^{K_S} |m_i|}{\sum_{i=1}^{K_S} |S_i|} \quad (18)$$

where $|m_i| = max_{|C_i \cap S_j|} \{\forall\, S_j \in\ S^* \wedge match(S_i, C_j) \geq \sigma_{OS}\}$

$$precision_N = \frac{\sum_{i=1}^{K_C} |m_i|}{\sum_{i=1}^{K_C} |C_i|} \quad (19)$$

where $|m_i| = max_{|C_j \cap S_i|} \{\forall\, C_j \in\ C^* \wedge match(C_j, S_i) \geq \sigma_{OS}\}$

$$Fmeasure_N = \frac{2 \times recall_N \times precision_N}{recall_N + precision_N} \quad (20)$$

Three other measures are also used in the evaluation. These are the complex-wise positive predictive value (PPV), the complex-wise sensitivity ($sensitivity$), and the geometric accuracy ($accuracy$) [30]. Both PPV and $sensitivity$ are based on the size of the intersection between the detected complexes and the true benchmark complexes.

$$PPV = \frac{\sum_{j=1}^{K_C} max_{i=1}^{K_S} t_{ij}}{\sum_{j=1}^{K_C} \sum_{i=1}^{K_S} t_{ij}} \quad (21)$$

$$sensitivity = \frac{\sum_{i=1}^{K_S} max_{j=1}^{K_C} t_{ij}}{\sum_{i=1}^{K_S} |S_i|} \quad (22)$$

where t_{ij} acts as the number of proteins shared by both the golden standard complex i and the predicted complex j. The geometric accuracy can be utilized to indicate the trade-off between $sensitvity$ and PPV.

$$accuracy = \sqrt{sensitivity\ * PPV} \quad (23)$$

5.4 Results

The results are reported in Tables 1–10, where the efficacious results of one competitive algorithm as compared with its counterpart algorithm are designated with boldface. Almost all the results presented in Tables 1–10 prove the ability of the EA with modularity density in both versions (i.e., EA_{QD} and EA_{Top-QD}) to outperform (in almost all evaluation metrics) EA_Q of Pizzuti and Rombo [3] and EA_{Top-Q} of Attea and Abdullah [14]. This is mainly due to the modularity density function's ability to handle more complex structures accurately than its counterpart. In other words, QD in EA_{QD} tends to lessen the negative impact of the resolution limit problem of Q in EA_Q. This in turn would promote those solutions with smaller complex sizes than those larger complex sizes preferred by Q in EA_Q.

Further, the results give an indication that positive collaboration can be obtained by QD and the topological-based mutation. Here, while the topologically based mutation attempts to figure out more appropriate complex structures than the canonical neighbor-based mutation, QD more quickly directs its search ability towards such solutions.

Table 1: Performance comparison for **Yeast-D1** in terms of $Recall$, $Precision$, and F for an average of 30 runs of canonical EA with Q model (EA_Q) against the proposed EA with QD model (EA_{QD}).

OS	$Recall$		$Precision$		F	
	EA_Q	EA_{QD}	EA_Q	EA_{QD}	EA_Q	EA_{QD}
0.10	0.8944	**0.9287**	**0.7826**	0.7813	0.8345	**0.8484**
0.15	0.8462	**0.8765**	0.7368	**0.7446**	0.7873	**0.8050**
0.20	0.7962	**0.8361**	0.7244	**0.7374**	0.7583	**0.7814**
0.25	0.7560	**0.8056**	0.7136	**0.7299**	0.7339	**0.7656**
0.30	0.7171	**0.7761**	0.6848	**0.7080**	0.7003	**0.7402**
0.35	0.6885	**0.7406**	0.6588	**0.6818**	0.6729	**0.7098**
0.40	0.6611	**0.7098**	0.6343	**0.6652**	0.6471	**0.6866**
0.45	0.6248	**0.6821**	0.5996	**0.6400**	0.6116	**0.6602**
0.50	0.6081	**0.6624**	0.5881	**0.6247**	0.5976	**0.6429**
0.55	0.5654	**0.6201**	0.5427	**0.5824**	0.5535	**0.6005**
0.60	0.5410	**0.5979**	0.5195	**0.5616**	0.5297	**0.5790**
0.65	0.5162	**0.5774**	0.4958	**0.5425**	0.0550	**0.5592**
0.70	0.4923	**0.5483**	0.4729	**0.5152**	0.4821	**0.5311**
0.75	0.4526	**0.5111**	0.4347	**0.4801**	0.4432	**0.4950**
0.80	0.4218	**0.4782**	0.4052	**0.4494**	0.4131	**0.4632**

Table 2: Performance comparison for **Yeast-D1** in terms of $Recall_N$, $Precision_N$, and F_N for an average of 30 runs of canonical EA with Q model (EA_Q) against the proposed EA with QD model (EA_{QD}).

OS	$Recall_N$		$Precision_N$		F_N	
	EA_Q	EA_{QD}	EA_Q	EA_{QD}	EA_Q	EA_{QD}
0.10	0.8301	**0.8540**	0.6905	**0.7343**	0.7538	**0.7894**
0.15	0.8066	**0.8229**	0.6845	**0.7297**	0.7404	**0.7731**
0.20	0.7647	**0.7942**	0.6793	**0.7269**	0.7192	**0.7587**
0.25	0.7221	**0.7692**	0.6714	**0.7217**	0.6955	**0.7445**
0.30	0.6643	**0.7391**	0.6644	**0.7039**	0.6540	**0.7209**
0.35	0.6351	**0.7097**	0.6209	**0.6844**	0.6278	**0.6967**
0.40	0.7221	**0.6697**	0.6714	**0.6624**	0.6955	**0.6659**
0.45	0.5552	**0.6332**	0.5518	**0.6295**	0.5535	**0.6313**
0.50	0.5273	**0.5998**	0.5290	**0.6013**	0.5282	**0.6005**
0.55	0.4902	**0.5497**	0.4902	**0.5497**	0.4902	**0.5497**
0.60	0.4619	**0.5189**	0.4619	**0.5189**	0.4619	**0.5189**
0.65	0.4408	**0.4893**	0.4408	**0.4893**	0.4408	**0.4893**

0.70	0.4138	**0.4547**	0.4138	**0.4547**	0.4138	**0.4547**
0.75	0.3795	**0.4222**	0.3795	**0.4222**	0.3795	**0.4222**
0.80	0.3586	**0.3992**	0.3586	**0.3992**	0.3586	**0.3992**

Table 3: Performance comparison for **Yeast-D1** in terms of $Recall$, $Precision$, and F for an average of 30 runs of topological-based EA with Q model (EA_{Top-Q}) against the proposed EA with QD model (EA_{Top-QD}).

OS	$Recall$		$Precision$		F	
	EA_{Top-Q}	EA_{Top-QD}	EA_{Top-Q}	EA_{Top-QD}	EA_{Top-Q}	EA_{Top-QD}
0.10	0.8239	**0.9496**	**0.7783**	0.7764	0.8000	**0.8542**
0.15	0.7496	**0.9064**	0.7370	**0.7433**	0.7428	**0.8167**
0.20	0.6906	**0.8650**	0.7348	**0.7425**	0.7116	**0.7989**
0.25	0.6427	**0.8355**	0.7247	**0.7413**	0.6809	**0.7854**
0.30	0.6145	**0.8021**	0.7090	**0.7388**	0.6582	**0.7690**
0.35	0.5966	**0.7812**	0.6979	**0.7313**	0.6431	**0.7553**
0.40	0.5701	**0.7620**	0.6867	**0.7293**	0.6228	**0.7451**
0.45	0.5436	**0.7496**	0.6681	**0.7242**	0.5993	**0.7366**
0.50	0.5329	**0.7231**	0.6576	**0.6999**	0.5885	**0.7112**
0.55	0.5197	**0.7077**	0.6427	**0.65850**	0.5745	**0.6960**
0.60	0.5141	**0.6987**	0.6358	**0.6762**	0.5683	**0.6872**
0.65	0.5030	**0.6795**	0.6220	**0.6576**	0.5560	**0.6682**
0.70	0.4983	**0.6675**	0.6162	**0.6459**	0.5508	**0.6564**
0.75	0.4714	**0.6222**	0.5830	**0.6022**	0.5211	**0.6119**
0.80	0.4449	**0.5893**	0.5502	**0.5704**	0.4918	**0.5796**

Table 4: Performance comparison for **Yeast-D1** in terms of $Recall_N$, $Precision_N$, and F_N for an average of 30 runs of the topological-based EA with Q model (EA_{Top-Q}) against the proposed EA with QD model (EA_{Top-QD}).

OS	$Recall_N$		$Precision_N$		F_N	
	EA_{Top-Q}	EA_{Top-QD}	EA_{Top-Q}	EA_{Top-QD}	EA_{Top-Q}	EA_{Top-QD}
0.10	0.8556	**0.9441**	0.6274	**0.7893**	0.7238	**0.8597**
0.15	0.8058	**0.9042**	0.6237	**0.7854**	0.7028	**0.8406**
0.20	0.7518	**0.8786**	0.6219	**0.7853**	0.6803	**0.8293**
0.25	0.6715	**0.8523**	0.6021	**0.7844**	0.6345	**0.8169**
0.30	0.6067	**0.8174**	0.5630	**0.7825**	0.5839	**0.7995**
0.35	0.5841	**0.8054**	0.5451	**0.7787**	0.5638	**0.7918**
0.40	0.5436	**0.7804**	0.5270	**0.7749**	0.5350	**0.7777**
0.45	0.4931	**0.7629**	0.4880	**0.7620**	0.4905	**0.7624**
0.50	0.4765	**0.6927**	0.4735	**0.6927**	0.4750	**0.6927**
0.55	0.4568	**0.6741**	0.4568	**0.6741**	0.4568	**0.6741**
0.60	0.4507	**0.6595**	0.4507	**0.6595**	0.4507	**0.6595**
0.65	0.4310	**0.6229**	0.4310	**0.6229**	0.4310	**0.6229**
0.70	0.4263	**0.6062**	0.4263	**0.6062**	0.4263	**0.6062**
0.75	0.4009	**0.5403**	0.4009	**0.5403**	0.4009	**0.5403**

0.80	0.3796	**0.5201**	0.3796	**0.5201**	0.3796	**0.5201**

Table 5: Performance comparison for **Yeast-D2** in terms of $Recall$, $Precision$, and F for an average of 30 runs of canonical EA with Q model (EA_Q) against the proposed EA with QD model (EA_{QD}).

OS	$Recall$		$Precision$		F	
	EA_Q	EA_{QD}	EA_Q	EA_{QD}	EA_Q	EA_{QD}
0.10	0.9533	**0.9709**	0.6083	**0.6175**	0.7425	**0.7547**
0.15	0.8884	**0.9113**	0.5716	**0.5822**	0.6953	**0.7103**
0.20	0.8171	**0.8449**	0.5376	**0.5521**	0.6482	**0.6676**
0.25	0.7482	**0.7736**	0.4825	**0.5003**	0.5864	**0.6074**
0.30	0.6709	**0.7009**	0.4578	**0.4760**	0.5439	**0.5667**
0.35	0.6007	**0.6353**	0.4260	**0.4443**	0.4981	**0.5227**
0.40	0.5549	**0.5909**	0.4058	**0.4279**	0.4684	**0.4961**
0.45	0.4913	**0.5236**	0.3722	**0.3969**	0.4233	**0.4513**
0.50	0.4709	**0.4991**	0.3657	**0.3886**	0.4114	**0.4368**
0.55	0.4029	**0.4240**	0.3221	**0.3406**	0.3577	**0.3775**
0.60	0.3669	**0.3853**	0.3018	**0.3196**	0.3309	**0.3492**
0.65	0.3207	**0.3413**	0.2770	**0.2990**	0.2969	**0.3185**
0.70	0.2653	**0.2904**	0.2414	**0.2632**	0.2525	**0.2760**
0.75	0.2296	**0.2584**	0.2106	**0.2369**	0.2194	**0.2470**
0.80	0.1978	**0.2211**	0.1882	**0.2124**	0.1927	**0.2165**

Table 6: Performance comparison for **Yeast-D2** in terms of $Recall_N$, $Precision_N$, and F_N for an average of 30 runs of canonical EA with Q model (EA_Q) against the proposed EA with QD model (EA_{QD}).

OS	$Recall_N$		$Precision_N$		F_N	
	EA_{CanQ}	EA_{CanQD}	EA_{CanQ}	EA_{CanQD}	EA_{CanQ}	EA_{CanQD}
0.10	0.5738	**0.5538**	0.7033	**0.7217**	0.6317	**0.6265**
0.15	0.5532	**0.5395**	0.6931	**0.7113**	0.6150	**0.6134**
0.20	0.5219	**0.5051**	0.6772	**0.6955**	0.5892	**0.5849**
0.25	0.4853	**0.4728**	0.6530	**0.6684**	0.5565	**0.5536**
0.30	0.4344	**0.4386**	0.6248	**0.6415**	0.5123	**0.5207**
0.35	0.3920	**0.4103**	0.5986	**0.6184**	0.4735	**0.4930**
0.40	0.3474	**0.3703**	0.5537	**0.5864**	0.4266	**0.4537**
0.45	0.3078	**0.3314**	0.5175	**0.5542**	0.3858	**0.4146**
0.50	0.2791	**0.3005**	0.4999	**0.5323**	0.3580	**0.3839**
0.55	0.2450	**0.2505**	0.4681	**0.4873**	0.3215	**0.3305**
0.60	0.2189	**0.2247**	0.4372	**0.4571**	0.2914	**0.3009**
0.65	0.1922	**0.1980**	0.3977	**0.4215**	0.2588	**0.2689**
0.70	0.1463	**0.1618**	0.3393	**0.3752**	0.2040	**0.2256**
0.75	0.1253	**0.1445**	0.2955	**0.3401**	0.1756	**0.2025**
0.80	0.1043	**0.1188**	0.2532	**0.2880**	0.1476	**0.1679**

Table 7: Performance comparison for **Yeast-D2** in terms of $Recall$, $Precision$, and F for an average of 30 runs of topological-based EA with Q model (EA_{Top-Q}) against the proposed EA with QD model (EA_{Top-QD}).

OS	$Recall$		$Precision$		F	
	EA_{Top-Q}	EA_{Top-QD}	EA_{Top-Q}	EA_{Top-QD}	EA_{Top-Q}	EA_{Top-QD}
0.10	0.9038	**0.9853**	0.5899	**0.6183**	0.7137	**0.7597**
0.15	0.8098	**0.9318**	0.5730	**0.5906**	0.6709	**0.7229**
0.20	0.7318	**0.8624**	0.5495	**0.5674**	0.6275	**0.6843**
0.25	0.6707	**0.7987**	0.5022	**0.5098**	0.5741	**0.6222**
0.30	0.6093	**0.7327**	0.4970	**0.4997**	0.5472	**0.5940**
0.35	0.5349	**0.6684**	0.4714	**0.4764**	0.5007	**0.5562**
0.40	0.4951	**0.6216**	0.4535	**0.4593**	0.4730	**0.5281**
0.45	0.4284	**0.5520**	0.4186	**0.4289**	0.4230	**0.4825**
0.50	0.4076	**0.5271**	0.4084	**0.4212**	0.4075	**0.4681**
0.55	0.3673	**0.4569**	0.3818	**0.3827**	0.3742	**0.4163**
0.60	0.3373	**0.4222**	0.3625	**0.3646**	0.3492	**0.3911**
0.65	0.3053	**0.3807**	0.3441	**0.3446**	0.3233	**0.3616**
0.70	0.2562	**0.3062**	0.3082	**0.2958**	0.2797	**0.3008**
0.75	0.2420	**0.2909**	0.2931	**0.2840**	0.2650	**0.2873**
0.80	0.2298	**0.2618**	0.2788	**0.2618**	0.2519	**0.2617**

Table 8: Performance comparison for **Yeast-D2** in terms of $Recall_N$, $Precision_N$, and F_N for an average of 30 runs of topological-based EA with Q model (EA_{Top-Q}) against the proposed EA with QD model (EA_{Top-QD}).

OS	$Recall_N$		$Precision_N$		F_N	
	EA_{Top-Q}	EA_{Top-QD}	EA_{Top-Q}	EA_{Top-QD}	EA_{Top-Q}	EA_{Top-QD}
0.10	0.6090	**0.5878**	0.6812	**0.7506**	0.6427	**0.6592**
0.15	0.5753	**0.5764**	0.6785	**0.7448**	0.6223	**0.6498**
0.20	0.5423	**0.5427**	0.6689	**0.7364**	0.5987	**0.6247**
0.25	0.5021	**0.5125**	0.6508	**0.7151**	0.5666	**0.5970**
0.30	0.4587	**0.4855**	0.6224	**0.7051**	0.5276	**0.5750**
0.35	0.3981	**0.4598**	0.5997	**0.6817**	0.4778	**0.5491**
0.40	0.3544	**0.4183**	0.5636	**0.6535**	0.4345	**0.5100**
0.45	0.2974	**0.3710**	0.5157	**0.6255**	0.3767	**0.4657**
0.50	0.2637	**0.3390**	0.4854	**0.6068**	0.3409	**0.4349**
0.55	0.2410	**0.2968**	0.4686	**0.5786**	0.3178	**0.3922**
0.60	0.2161	**0.2679**	0.4458	**0.5568**	0.2905	**0.3616**
0.65	0.1884	**0.2430**	0.4209	**0.5272**	0.2599	**0.3325**
0.70	0.1551	**0.1842**	0.3858	**0.4608**	0.2212	**0.2631**
0.75	0.1408	**0.1730**	0.3490	**0.4347**	0.2005	**0.2474**
0.80	0.1273	**0.1490**	0.3144	**0.3756**	0.1811	**0.2133**

Table 9: Performance comparison for **Yeast-D1** in terms of $Sensitivity$, PPV, and $accuracy$ for the tested EA-based complex detection approaches for PPI Yeast-D1.

$Sensitivity$		PPV		$accuracy$	
EA_Q	EA_{QD}	EA_Q	EA_{QD}	EA_Q	EA_{QD}
0.9082	0.8937	0.6925	**0.7363**	0.7928	**0.8109**
EA_{Top-Q}	EA_{Top-QD}	EA_{TopQ}	EA_{TopQD}	EA_{TopQ}	EA_{TopQD}
0.9782	0.9646	0.6277	**0.7904**	0.7834	**0.8732**

Table 10: Performance comparison for **Yeast-D2** in terms of $Sensitivity$, PPV, and $accuracy$ for the tested EA-based complex detection approaches for PPI Yeast-D2.

$Sensitivity$		PPV		$accuracy$	
EA_Q	EA_{QD}	EA_Q	EA_{QD}	EA_Q	EA_{QD}
0.5933	0.5648	**0.3015**	0.2912	**0.4229**	0.4055
EA_{Top-Q}	EA_{Top-QD}	EA_{TopQ}	EA_{TopQD}	EA_{TopQ}	EA_{TopQD}
0.6567	0.5933	0.2775	**0.3015**	**0.4269**	0.4229

6. Conclusions

The results in the tables reflect that the higher recall values obtained by the proposed algorithm in both versions of the neighbor-based mutation and the topological-based mutation mean that the proposed algorithm predicts a higher number of true complexes of all the true complexes in the benchmark dataset. Further, the results in the tables reveal high values of precision obtained by the proposed algorithm, again in both of its two versions. This implies a more reliable detection, as the detected complex structures are self-possessed with a high fraction of correct neighboring proteins belonging to the true complexes while simultaneously filtering out more false complex structures. Also revealed by the results is the ability of the proposed algorithm in both its versions to handle more compromise between the contradictory intentions of recall and precision. This can be seen from the higher values achieved by their harmonic mean of F score at both complex and protein levels.

In this paper, the problem of complex detection in PPI networks has been redefined as an EA-based optimization problem. The adopted quality function is modularity density, where the main interest is to compare the effectiveness of the proposed EA with modularity density against the canonical EA with modularity function. Moreover, the investigation considers two forms of mutation operator. These are the neighbor-based mutation operator and the topological-based mutation operator. The evaluation is performed using two yeast saccharomyces cerevisiae PPINs. The experimental results demonstrated the ability of the EA modularity density and the topological-based mutation operator to outperform the canonical EA with a neighbor-based mutation operator. Both the proposed EA_{QD} and EA_{Top-QD} achieve higher values for $recall$ and $precision$ at both complex and protein levels than the counterpart EA with modularity. More importantly, the algorithms can handle more compromise between the competing aims of $recall$ and $precision$. This is proved by the improved values obtained by their harmonic mean of F score, again, at both complex and protein levels (i.e. F and F_N).

However, the results of $Sensitivity$, PPV, and $accuracy$ indicated the need for further fruitful design issues. This opens the door for further ramifications to improve the detection ability of the proposed evolutionary-based complex detection algorithms. This would be

possible by playing with other fruitful information in the proteins. The additional information could be injected into the algorithm's design. One of the essential pieces of information is the gene ontology (GO) and its role in identifying the functional, rather than topological, similarities between protein pairs.

References

[1] M. Haque, R. Sarmah, and D. K. Bhattacharyya, "A common neighbor based technique to detect protein complexes in PPI networks," *Journal of Genetic Engineering and Biotechnology,* vol. 16, no. 1, pp. 227-238, 2018, https://doi.org/10.1016/j.jgeb.2017.10.010

[2] M. C. Costanzo *et al.*, "The yeast proteome database (YPD) and Caenorhabditis elegans proteome database (WormPD): comprehensive resources for the organization and comparison of model organism protein information," *Nucleic acids research,* vol. 28, no. 1, pp. 73-76, 2000.

[3] C. Pizzuti and S. E. Rombo, "Algorithms and tools for protein–protein interaction networks clustering, with a special focus on population-based stochastic methods," *Bioinformatics,* vol. 30, no. 10, pp. 1343-1352, 2014.

[4] G. D. Bader and C. W. Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC bioinformatics,* vol. 4, no. 1, pp. 1-27, 2003.

[5] J. Gagneur, R. Krause, T. Bouwmeester, and G. Casari, "Modular decomposition of protein-protein interaction networks," *Genome biology,* vol. 5, no. 8, pp. 1-12, 2004.

[6] X.L. Li, C.-S. Foo, and S.-K. Ng, "Discovering protein complexes in dense reliable neighborhoods of protein interaction networks," In *Computational Systems Bioinformatics*, vol 6, pp. 157-168, 2007.

[7] K. Macropol, T. Can, and A. K. Singh, "RRW: repeated random walks on genome-scale protein networks for local cluster discovery," *BMC bioinformatics,* vol. 10, pp. 1-10, 2009.

[8] G. Liu, L. Wong, and H. N. Chua, "Complex discovery from weighted PPI networks," *Bioinformatics,* vol. 25, no. 15, pp. 1891-1897, 2009.

[9] Y. Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *nature,* vol. 466, no. 7307, pp. 761-764, 2010.

[10] R. W. Solava, R. P. Michaels, and T. Milenković, "Graphlet-based edge clustering reveals pathogen-interacting proteins," *Bioinformatics,* vol. 28, no. 18, pp. i480-i486, 2012.

[11] C. Pizzuti and S. Rombo, "Experimental evaluation of topological-based fitness functions to detect complexes in PPI networks," In *Proceedings of the 14th annual conference on genetic and evolutionary computation*, 2012, pp. 193-200.

[12] S. Bandyopadhyay, S. Ray, A. Mukhopadhyay, and U. Maulik, "A multiobjective approach for identifying protein complexes and studying their association in multiple disorders," *Algorithms for Molecular Biology,* vol. 10, pp. 1-15, 2015.

[13] S. Ray, A. Hossain, and U. Maulik, "Disease associated protein complex detection: a multi-objective evolutionary approach," in *2016 International conference on microelectronics, computing and communications (MicroCom)*, 2016: IEEE, pp. 1-6.

[14] B. A. Attea and Q. Z. Abdullah, "Improving the performance of evolutionary-based complex detection models in protein–protein interaction networks," *Soft Computing,* vol. 22, pp. 3721-3744, 2018.

[15] A. H. Abdulateef, A. A. Bara'a, and A. N. Rashid, "Heuristic modularity for complex identification in protein-protein interaction networks," *Iraqi Journal of Science,* vol. 60, no. 8, pp. 1846-1859, 2019, doi: 10.24996/ijs.2019.60.8.22.

[16] A. H. Abdulateef, A. A. Bara'a, A. N. Rashid, and M. Al-Ani, "A new evolutionary algorithm with locally assisted heuristic for complex detection in protein interaction networks," *Applied Soft Computing,* vol. 73, pp. 1004-1025, 2018.

[17] D. Plewczyński and K. Ginalski, "The interactome: predicting the protein-protein interactions in cells," *Cellular and Molecular Biology Letters,* vol. 14, no. 1, pp. 1-22, 2009.

[18] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences,* vol. 99, no. 12, pp. 7821-7826, 2002.

[19] Z. Li, S. Zhang, R.-S. Wang, X.-S. Zhang, and L. Chen, "Quantitative function for community detection," *Physical review E,* vol. 77, no. 3, p. 036109, 2008.

[20] Q. Z. Abdullah and A. A. Bara'a, "A Heuristic Strategy for Improving the Performance of Evolutionary Based Complex Detection in Protein-Protein Interaction Networks," *Iraqi Journal of Science,* vol. 57, no. 4A, pp. 2513-2528, 2016.

[21] A. A. Bara'a *et al.*, "A review of heuristics and metaheuristics for community detection in complex networks: Current usage, emerging development and future directions," *Swarm and Evolutionary Computation,* vol. 63, p. 100885, 2021.

[22] E. A. Khalil, S. Ozdemir, and A. A. Bara'a, "A new task allocation protocol for extending stability and operational periods in internet of things," *IEEE Internet of Things Journal,* vol. 6, no. 4, pp. 7225-7231, 2019.

[23] M. M. Abdulrahman, A. D. Abood, and B. A. Attea, "An enhanced multi-objective evolutionary algorithm with decomposition for signed community detection problem," in 2020 2nd Annual International Conference on Information and Sciences (AiCIS), 2020: IEEE, pp. 45-50.

[24] M. N. Abbas, A. A. Bara'a, and N. J. Kadhim, "Evolutionary based set covers algorithm with local refinement for power aware wireless sensor networks design," *Iraqi Journal of Science,* vol. 59, no. 4A, pp. 1959-1966, Oct. 2018.

[25] A. A. Bara'a and H. S. Khoder, "A new multi-objective evolutionary framework for community mining in dynamic social networks," *Swarm and Evolutionary Computation,* vol. 31, pp. 90-109, 2016.

[26] N. J. Kadhim and H. H. Saleh, "Improving extractive multi-document text summarization through multi-objective optimization," *Iraqi Journal of Science,* vol. 59, no. 4B, pp. 2135–2149, Nov. 2018.

[27] J. Handl and J. Knowles, "An evolutionary approach to multiobjective clustering," *IEEE transactions on Evolutionary Computation,* vol. 11, no. 1, pp. 56-76, 2007.

[28] A. C. Gavin *et al.*, "Proteome survey reveals modularity of the yeast cell machinery," *Nature,* vol. 440, no. 7084, pp. 631-636, 2006.

[29] N. Zaki, J. Berengueres, and D. Efimov, "Detection of protein complexes using a protein ranking algorithm," *Proteins: Structure, Function, and Bioinformatics,* vol. 80, no. 10, pp. 2459-2468, 2012.

[30] S. Brohee and J. Van Helden, "Evaluation of clustering algorithms for protein-protein interaction networks," *BMC bioinformatics,* vol. 7pp. 1-19, 2006.