MUMI

Chen, Weiqi; Zhu, Zexuan; He, Shan

DOI:
10.1109/TEVC.2019.2952220

License:
Other (please specify with Rights Statement)

Document Version
Peer reviewed version

Citation for published version (Harvard):
Chen, W, Zhu, Z & He, S 2019, 'MUMI: Multitask module identification for biological networks', *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2019.2952220

MUMI: Multitask module identification for biological networks

Weiqi Chen, Zexuan Zhu, and Shan He

***Abstract*—Identifying modules from biological networks is important since modules reveal essential mechanisms and dynamic processes in biological systems. Existing algorithms focus on identifying either active modules or topological modules (communities), which represent dynamic and topological units in the network, respectively. However, high-level biological phenomena, e.g., functions are emergent properties from the interplay between network topology and dynamics. Therefore, to fully explain the mechanisms underlying the high-level biological phenomena, it is important to identify the overlaps between communities and active modules, which indicate the topological units with significant changes of dynamics. However, despite the importance, there are no existing methods to do so. In this paper, we propose MUMI (MUltitask Module Identification) algorithm to detect the overlaps between active modules and communities simultaneously. Experimental results show that our method provides new insights into biological mechanisms by combining information from active modules and communities. By formulating the problem as a multitasking learning problem which searches for these two types of modules simultaneously, the algorithm can exploit their latent complementarities to obtain better search performance in terms of accuracy and convergence. Our MATLAB implementation of MUMI is available at https://github.com/WeiqiChen/Mumi-multitask-module-identification.**

***Index Terms*—Active Module, Community Detection, Multifactorial Evolution.**

I. Introduction

NETWORK biology [1] or recently network medicine [2] have gained more attention due to its ability of gaining insights about the mechanisms of complex biological phenomena such as disease. These approaches first model a complex biological system, e.g., a cell, as a complex network. Then network (graph) analytic algorithms are applied to characterise the topological structure of the network [1], [2]. These structural characteristics are then related to biological functions to understand the underlying biological mechanisms. Among these structural characteristics, modules receive most attention due to the inherited modular structure of biological systems.

Currently, there are two kinds of modules, topological modules and active modules. Topological modules, also known as communities, are topological units in a network. A community is defined as locally dense neighbourhood with more inner interactions than outside interactions. In biological networks, communities are used to approximate the functional units of cellular process and organisation [3], [2]. This is because biological components exert functions by interacting with each other, forming various functional units, and functional units tend to form densely connected area in network, i.e. communities.

Active modules, on the other hand, are dynamic units in a networks that consider network dynamics [4]. They are conceived as the approximation of the dynamic mechanisms of biological systems [4]. An active module is a region (sub-network) in a biological network that show striking changes in molecular activity, e.g., gene expression. Active module is often associated with a given cellular response [4].

However, these two kinds of modules cannot fully depict the underlying mechanisms of a biological system. This is because high-level biological phenomena, e.g., functions, emerge from the interplay between network topology and dynamics [5], [6]. From this perspective, communities (topological units) and active modules (dynamic units), can only partially explain the underlying mechanisms of the emergent phenomena. However, by detecting the overlaps between communities and active modules, we can find out the topological (functional) units with significant change of dynamics. These overlaps might be more informative in terms of revealing the mechanisms underlying the emergent biological phenomena.

From the discussion above, we propose the following overarching hypothesis: by identifying the overlaps

W. Chen and S. He are with the School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom e-mail: (see https://www.cs.bham.ac.uk/~szh/). S. He is also with Department of Computer Science and Engineering, OPAL Laboratory, Southern University of Science and Technology, Shenzhen, China. Z. Zhu is with College of Computer Science and Software Engineering Nanhai Avenue 3688, Shenzhen University, Shenzhen, Shenzhen, CN 518060

between communities and active modules, we can reveal the mechanisms of high-level biological phenomena that cannot be achieved through identifying them separately. We also hypothesise that we can gain better module identification performance in terms of accuracy and convergence by searching communities and active modules simultaneously. The reason is, as mentioned previously, network topology and dynamics complement each other to give rise to the high-level biological phenomena. Therefore, these two modules, i.e., communities (topological units) and active modules (dynamic units) complement each other and might have a similar problem structure. Therefore, by searching them simultaneously, we can exploit their latent complementarities, which lead to better search performance.

To test our hypotheses, we propose a novel multitask module identification algorithm (MUMI) based on multifactorial evolution, which is a evolutionary algorithm that simultaneously solves multiple tasks that may or may not be interdependent [7]. We have also designed a novel unified genetic representation for multiple tasks, problem-specific decoding scheme, and task specific genetic operators. We design experiments based on a set of benchmark networks and two real-world biological networks to validate our two hypotheses.

The rest of this paper is organised as follows. Section II introduces the related work on areas of evolutionary multitasking and module identification. Section III gives a formal definition of problems and a detailed description of MUMI. Experimental studies of applying MUMI and other algorithms on various networks are shown in Section IV, and discussed further in Section V. Section VI concludes this paper. Supplementary materials including supplementary table, MATLAB source code, formatted input data and experimental results are available at https://github.com/WeiqiChen/Mumi-multitask-module-identification .

II. Related Work

A. Evolutionary Multitasking

Evolutionary multitasking investigates into the implicit parallelism of evolutionary optimisation problems. An introductory study [7] on evolutionary multitasking shows that it allows for implicit transfer of genetically encoded information across multiple optimisation tasks. This process, also known as transfer learning, improves the efficiency and convergence speed of evolutionary multitasking on computationally expensive problems.

The idea of accelerating convergence via information transfer between objectives is not newly invented by multitasking. Previous researches on multi co-objective evolution [8] and memetic search [9], [10] have already shown that the knowledge transfer and reuse across objectives is able to improve search performance of evolutionary algorithm on computationally expensive problems. In the context of computational intelligence, memes are referred to as recurring patterns or knowledge embedded in computational representations [11]. In an early study [10] that formulates transfer learning as computational operators, knowledge learned in previous problem-solving process is transferred in the form of memes as building blocks, and helps accelerate future search. A similar study [12] on re-usable knowledge extraction proposes the concept of simultaneous problem learning that emphasises on the interaction between optimiser and problem learning.

Research on evolutionary multitasking is strongly triggered by the need in fast developing cloud computing industry where cross-domain optimisation must be handled with high efficiency. Different to traditional multi-objective optimisation that has one single search space, multitask optimisation is capable of dealing with multiple search spaces, each corresponding to an individual optimisation task [13]. Dependency among tasks are not required for multitasking. The essential point is that it handles cross-domain optimisation through a unified solution representation scheme [7], [13] for objectives across domain. Research [13] has shown that genetic operator applied to the unified genetic space is able to drive knowledge transfer between different optimisation tasks across domain, thus proven that evolutionary multitasking indeed works.

Evolutionary multitasking has a broad range of application that are not restricted to cloud computing or solely cross-domain multitasking. It has been applied to a series of classic combinatorial optimisation problems [14] as well as real world problems like manufacturing process design [15], neural network training [16], bi-level optimisation [17], etc. Nevertheless, it is still a new emerging field that has far not been fully explored. The development of more efficient evolutionary multitasking algorithms and further application to numerous complicated real world problems are promising and attractive future directions in this field.

B. Module Identification

Topological module, also known as community is one of the most studied network features [18], [19]. Identification of community structures have been studied for many decades under different terminologies such as graph partitioning, network division, hierarchical clustering, or block modelling [20]. One of the most successful methods is modularity optimisation [21] proposed

by Newman and Girvan. The modularity optimisation method consists of a scalar measurement called modularity. This modularity can assess the quality of a given division of an undirected network. Several algorithms have been developed to identify communities by maximising the modularity measure, denoted as Q, by dividing a network into communities, e.g., removing edge in remaining network iteratively [21]. Although there have been researches showing that community detection by modularity optimisation may suffer from resolution limit and thus fail to identify communities that are smaller than a scale depending on some network parameters [22], optimisation of modularity Q is still one of the most successful and widely used community detection methods.

Active module [23] reveals dynamic and process-specific i nformation t hat i s c orrelated w ith c ellular or disease states. In a general procedure for active module identification, molecular profiles are incorporated to provide quantified i nformation o f m olecular a ctivities that can be converted into scores for network annotation. After network activity is annotated, algorithms are applied to the network for the identification o f a ctive modules based on a variety of strategies. The extracted modules are tested for statistical significance. M ethod validation and improvement is also performed in this step. As the problem of finding t he m aximal-scoring connected module has proven to be NP-hard [23], heuristic algorithms are broadly used to approximately search for high scoring modules. Commonly used heuristic approaches are simulated annealing [23], greedy search [24], and evolutionary algorithm [25], [26], [27], [28].

III. A Novel multitask module identification algorithm based on Multifactorial Evolution

In this section describe a novel multitask algorithm for identifying active modules and communities simultaneously in a biological network. It is the first formulation of the two widely studied problems into the multifactorial evolution paradigm. We have designed a unified genetic representation for the two different tasks and corresponding task-specific d ecoding m ethod. W e have also designed task-specific m utation a nd l ocal search operators in order to improve the algorithm performance. A solution repairing operator has been developed to improve the module identification results.

A. Multifactorial Evolution

We first give a brief introduction to Multifactorial Evolution proposed in reference [7] to make this paper self-contained.

In the initialisation stage of Multifactorial Evolution, every individual in the population is evaluated with respect to every optimisation task in the multitasking environment. For each task T_j, an individual L_i has a factorial rank r_j^i corresponding to the rank of the individual's objective fitness for this task in the whole population. The lower number the rank is, the better performance individual shows in specified task. For K number of tasks an individual L_i is assigned with a list of K factorial ranks $\{r_1^i, r_2^i, ..., r_K^i\}$. The scalar fitness φ_i of individual L_i is based on its best rank among all the tasks, given by

$$\varphi_i = \frac{1}{min_{j\in\{1,...,K\}}\{r_j^i\}} \tag{1}$$

The scalar fitness φ can be considered as best ever performance one individual is able to achieve across all tasks.

The skill factor τ_i of individual L_i is then given by

$$\tau_i = argmin_j\{r_j^i\}, j \in \{1, 2, ..., K\} \tag{2}$$

meaning the task individual L_i is most effective. A basic structure of multifactorial evolutionary algorithm is described in Algorithm 1. Important steps such as line 2 and 5 are explained as sub-algorithms (Algorithms 2 - 5) in following sections.

Similar to all evolutionary algorithms, multifactorial evolution starts with population initialisation and evaluation, then repeatedly applies crossover and mutation on current population to generate offspring population, mixes parental and offspring population together, and selects the fittest individuals as next generation until a stopping criterion is satisfied. The difference is that in general evolutionary algorithms fitness can be calculated by individual itself while in multifactorial evolutionary algorithm fitness relates to the rank of individual in the whole population. More details on concept definition and multifactorial evolutionary algorithm scheme can be found in reference [7].

B. Definition of Tasks

This multitasking problem contains two tasks: identification of active modules and division of network into structural communities.

1) Task 1: Community detection: To detect communities in a network, we adopt modularity maximisation. For a given division on network with adjacency matrix A_{ij}, the modularity defined by Newman and Girvan [21] and modified to be suitable for edge weighted network is calculated as

$$Q = \frac{1}{2m}\sum(A_{ij} - \frac{k_i k_j}{2m})\delta(c_i, c_j) \tag{3}$$

Algorithm 1: Basic Structure of Multifactorial Evolutionary Algorithm

```
Population initialisation as current-pop ;
Evaluate every individual with respect to every
  optimisation task (Algorithm 2 and 3);
Compute the skill factor τ of every individual ;
while stopping criteria not satisfied do
    Apply Crossover and Mutation on current-pop
      to generate offspring-pop (Algorithm 4 and 5);
    Evaluate offspring individuals for selected
      optimisation tasks ;
    intermediate-pop ← Union(current-pop,
      offspring-pop);
    Update the scalar fitness φ and skill factor τ for
      every individual in intermediate-pop ;
    current-pop ← fittest individuals in
      intermediate-pop
end
```

where

$$m = \frac{1}{2}\sum_{ij} A_{ij} \tag{4}$$

is the number of edges in the network, and the δ function $\delta(u, v)$ is 1 if $u = v$ and 0 otherwise. c_i is the label of community to which node i is assigned in this division. k_i denotes the degree of the i-th node. The intuition of modularity Q is to measure the difference between edge density inside communities given a community division in the network and the same quantity for a network with the same community division but randomly distributed edges.

The community detection problem through modularity maximisation is formulated as following.

Problem 1 (Community Detection Problem): Given a network $G = \{V, E\}$ where V denotes for the set of nodes and E for the set of edges, divide the set of nodes V into m mutually exclusive subsets $\{V_1, V2, ..., V_m\}$, $V_i \in V, V_i \neq \emptyset$ for $i = 1, 2, ..., m$, and $\cup_{i=1}^{m} V_i = V, V_i \cap V_j = \emptyset$ for $i \neq j$, so that the value of modularity Q is maximised.

2) Task 2: Active module identification: For a given protein-protein interaction network with *p*-values indicating the gene differential expression level assigned to each node v, an additive score $S^{FDR}(v)$ can be formulated using a beta-uniform mixture model.

Microarray analysis studies showed that expression data can be effectively estimated by many mixture-model methods that divide genes into two or more groups, one group contains genes that are differentially expressed, and other(s) not differentially expressed. Among those many methods, Pounds and Morris proposed a beta-uniform mixture (BUM) model that very accurately describes the distribution of a large set of *p*-values produced from an microarray experiment [29]. The BUM model considers the distribution of *p*-values as a mixture of a special case of beta distribution ($b = 1$) and a uniform(0, 1) distribution, with a mixture parameter λ. The *p*-values under the null hypothesis are assumed to have a uniform distribution. Under the alternative hypothesis the distribution of *p*-values will have a high density for small *p*-values and can be described by $B(a, 1)$.

A general beta distribution $B(a, b)$ is given by

$$(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} \tag{5}$$

where $\Gamma(.)$ denotes the gamma function. As $\Gamma(1) = 1$, the probability density function of BUM model is then reduced to

$$f(x|a, \lambda) = \lambda + (1-\lambda)ax^{a-1} \tag{6}$$

for $0 < x \leq 1$, $0 < \lambda < 1$ and $0 < a < 1$. Given a set of *p*-values the two parameters of BUM distribution λ and a can be calculated by maximum likelihood estimation.

Following the idea of Dittrich and Klau [30] to decompose signal component from background noise, an additive score to measure the significance of gene's differential expression is calculated as

$$\begin{aligned} S^{FDR}(x) &= \log \frac{B(a,1)(x)}{B(a,1)(\tau)} \\ &= \log \frac{ax^{a-1}}{a\tau^{a-1}} \\ &= (a-1)(\log x - \log \tau) \end{aligned} \tag{7}$$

where τ is a threshold to determine the significance of a *p*-value. In order to control the estimated upper bound of the false discovery rate (FDR) introduced by Benjamini and Hochberg [31], τ could then be selected to ensure that $FDR \leq \alpha$ for some predefined α using the following equation

$$\tau = \left(\frac{\hat{\pi} - \alpha\lambda}{\alpha(1-\lambda)}\right)^{\frac{1}{(a-1)}} \tag{8}$$

where $\hat{\pi} = \lambda + (1-\lambda)a$, meaning the maximum proportion of the set of *p*-values that could arise from the null hypothesis.

After assigning score to each of the genes, the overall score for a given module A is then the summation of all genes' scores in it, given by

$$S_A = \sum_{x \in A} S^{FDR}(x) \tag{9}$$

The active module identification problem is then formulated as following.

Problem 2 (Active Module Identification Problem): Given a network $G = \{V, E\}$ where V denotes for the set of nodes, E denotes for the set of edges, $n = |V|$ is the number of nodes and each node $v_i \in V, i = 1, 2, ..., n$ is assigned with node weight $S^{FDR}(v_i)$, find a connected subgraph $S = \{V_S, E_S\}, V_S \in V, E_S \in E$ so that $\sum_{v_i \in V_S} S^{FDR}(v_i)$ is maximised.

In the previous work [28], we have already shown how to solve problem 2 through a binary vector encoding scheme constrained by algebraic connectivity in a multi-objective evolutionary algorithm. In order to include problem 1 in an environment of evolutionary multitasking, we propose a new unified genetic representation as detailed below.

C. A Unified Genetic Representation for Multiple Tasks and Problem-Specific Decoding Scheme

For a network $G = V, E$ of size $n = |V|$, an individual solution is encoded as an integer vector of length n, each integer representing the label of community to which corresponding node is assigned, i.e. for the i-th individual L_i in population, we have

$$L_i = \{l_i^1, l_i^2, ..., l_i^n\} \tag{10}$$

where $l_i^j \in \{0, 1, ..., n-1\}$ for $j = 1, 2, ..., n$, meaning the available label of communities ranges from 0 to $n-1$.

There have been other genetic representation methods for community detection, such as the locus-based adjacency representation used by Clara Pizutti [32]. The advantage of our representation is that network division can be easily interpreted from reading community labels for each node in chromosome, and that it allows for a second decoding scheme. During the whole evolutionary algorithm, connectivity for each community is not explicitly required in algorithm implementation, however the process of modularity maximisation implicitly drives the network division towards densely connected solutions. A detailed chromosome decoding scheme for community detection task is described in Algorithm 2.

In our previous work of active module identification, we used algebraic connectivity as a constraint to ensure the connectivity of detected active module [28]. In this algorithm algebraic connectivity is no longer used, instead the collection of positions assigned with positive integers is interpreted as a subgraph whose connected component S with highest $\sum_{v \in V_S} S^{FDR}(v)$ is identified as the active module this individual represents. This connected components finding based decoding scheme is inspired by the work of Li et al. [27]. Details of the chromosome decoding scheme for active module identification task is described in Algorithm 3.

Algorithm 2: Chromosome Decoding Scheme For Task 1 (Community detection)

Input: Individual L_i, adjacency matrix A of the whole network
Output: Network Division $\{V_1, V2, ..., V_m\}$, Modularity Q

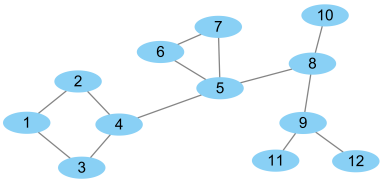
```
// get labels of communities
labels ← unique elements of L_i ;
m ← length(labels) ;
for j ← 1 to m do
    label ← labels(j) ;
    V_j ← ∅ ;
    for k ← 1 to n do
        // get all the nodes in j-th
           community
        if L_i^k == label then
            V_j = V_j ∪ v_k
        end
    end
end
// Now network division represented
   by L_i is decoded
calculate modularity Q for given network division
  {V_1, V2, ..., V_m} ;
return {V_1, V2, ..., V_m}, Q
```

Figure 1 gives a simple example of how to decode given chromosome representation for two tasks. In a network with 12 nodes and 13 edges (Figure 1a), the chromosome is encoded as $[1, 1, 1, 1, 2, 2, 2, 0, 3, 0, 3, 3]$ (Figure 1b). Visualisation of decoding scheme under two tasks can be seen in Figure 1c and 1d.

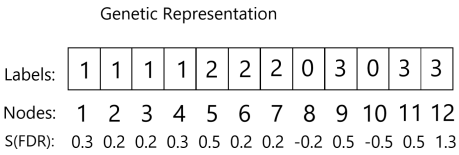
D. Task-Specific Mutation Operator

In order to improve the performance of the algorithm and provide better guidance in searching the solution space, we have mutation and local search operators specially designed for the two different tasks. Upon taking in an individual, the mutation operator first checks its skill factor to decide the task in which this individual is more effective, then it applies different mutation strategy accordingly.

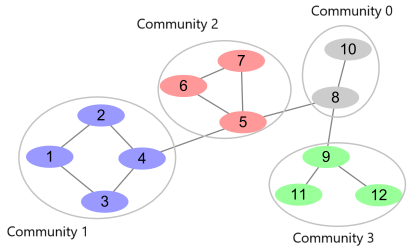
Individual specialised in active module identification goes through a subgraph expanding stage and a node deletion stage. In the first stage, neighbouring nodes with positive weight are added to the subgraph while those with negative weight also have probabilities to be included. In the second stage, negative weighted nodes



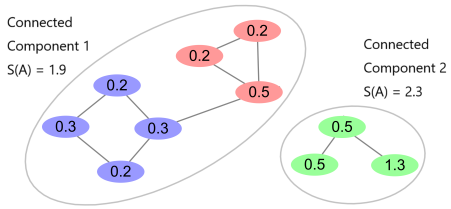
(a) Sample network



(b) Genetic representation



(c) Decoding for network division



(d) Decoding for active module identification

Fig. 1: A simple example of the chromosome encoding and decoding scheme for two tasks. Figure 1a: Visualisation of the sample network with 12 nodes and 13 edges. Figure 1b: The genetic representation of one individual, an integer vector of length 12, each integer representing the community label of corresponding node. The vector $S(FDR)$ gives the active node score. Figure 1c: Network is divided into 4 communities labelled from 0 to 3 according to the individual. Figure 1d: Subgraph formed by all nodes with non-zero labels. Nodes are labelled by active score $S(FDR)$. Module score $S(A)$ is calculated for each connected component in the subgraph. In this example, connected component 2 with a higher active module score $S(A) = 2.3$ is selected as the decoded active module.

Algorithm 3: Chromosome Decoding Scheme for Task 2 (Active module identification)

Input: Individual L_i, adjacency matrix A of the whole network, a list of $S^{FDR}(v)$ assigned to each node

Output: Connected node set of active module V_S, active module score $\sum_{v \in V_S} S^{FDR}(v)$

binary vector $l \leftarrow L_i > 0$;
subgraph $A_l \leftarrow A(l, l)$;
get all the k connected components $\{V_1, V2, ..., V_k\}$ in A_l ;
$S_{max} \leftarrow$ negative infinity ;
$V_S \leftarrow \emptyset$;
for $j \leftarrow 1$ *to* k **do**
    $S_j \leftarrow \sum_{v \in V_j} S^{FDR}(v)$;
    // get the active module score S_j of the j-th connected components V_j
    if $S_j > S_{max}$ **then**
        $S_{max} = S_j$;
        $V_S \leftarrow V_j$;
    end
end
return V_S, S_{max}

in subgraph go through a similar probabilistic deletion process. A detailed description is shown in Algorithm 4.

Individual specialised in network division is applied with a completely different mutation strategy called random community merging. This mutation is an imitation of bottom-up merging strategy in quite a few community detection algorithms. In initialisation stage, every individual is assigned with a random permutation of integers 0 to $n-1$, meaning that every node is the sole member of one of n communities. When such mutation strategy is applied to an individual, two connected communities are randomly selected to be joined together to form a new larger community. As evolution goes on, small communities are gradually merged into large communities, accompanied with a significant increase in modularity Q. In the late stage of evolution the value of Q becomes stable, indicating the algorithm has reached the optima in modularity maximisation task. A detailed description of community merging mutation is shown in Algorithm 5.

E. Uniform Crossover Operator

This algorithm uses uniform crossover to generate two child individuals from two parent individuals. Although uniform crossover has a higher probability to destroy

Algorithm 4: Apply mutation with local search steps to chromosomes specialised in task 1 (skill factor $\tau == 1$)

Input: Individual L_i, adjacency matrix A of the whole network, a list of $S^{FDR}(v)$ assigned to each node
Output: Individual L_i after mutation

// individual L_i is more effective in task 1
node set of subgraph S is given by $V_S \leftarrow \{V_j | L_i^j > 0\}, j = 1, 2, ..., n$;
$V_{neighbours} \leftarrow$ all neighbouring nodes of V_S;
// get labels of communities
labels $\leftarrow$ unique elements of L_i ;
// Stage 1: probabilistic subgraph expanding
for *every node* v_j *in* $V_{neighbours}$ **do**
  if $S^{FDR}(v_j) \geq 0$ **then**
    // if the neighbour v_j is assigned with positive $S^{FDR}(v_j)$, include it
    $L_i^j \leftarrow$ randomly select one label from *labels* (cannot be 0)
  else
    // if the neighbour v_j is assigned with negative $S^{FDR}(v_j)$, include it with probability $exp(S^{FDR}(v_j))$
    if $exp(S^{FDR}(v_j)) > random()$ **then**
      $L_i^j \leftarrow$ randomly select one label from *labels* (cannot be 0)
    end
  end
end
// Stage 2: probabilistic negative weighted node deletion
update node set of subgraph S by $V_S \leftarrow \{V_j | L_i^j > 0\}, j = 1, 2, ..., n$;
for *every node* v_j *in* V_S **do**
  // if the node v_j is assigned with negative $S^{FDR}(v_j)$, delete it with probability $1 - exp(S^{FDR}(v_j))$
  if $exp(S^{FDR}(v_j)) < random()$ **then**
    $L_i^j \leftarrow 0$
  end
end
return L_i

Algorithm 5: Apply mutation with random community merging to chromosomes specialised in task 2 (skill factor $\tau == 2$)

Input: Individual L_i, adjacency matrix A of the whole network
Output: Individual L_i after mutation

// individual L_i is more effective in task 2
randomly select one community label c_1 in L_i ;
node set of community c_1 is given by $V_{c1} \leftarrow \{v_k | L_i^k == c_1\}, k = 1, 2, ..., n$;
$V_{neighbours} \leftarrow$ all neighbouring nodes of V_{c1};
if $V_{neighbours}$ *contains community label different from* c_1 **then**
  randomly select another community label c_2 in $V_{neighbours}$;
  // merge all nodes in community c_1 into community c_2
  $L_i(L_i == c_1) \leftarrow c_2$
end
return L_i

community structures that are already detected than simple one-point or two-points crossover, in practise it is proven to be an effective way to preserve the diversity of population, help explore solution space, and avoid being stuck in local optima.

F. Repair operator for community detection

The multifactorial evolutionary algorithm scheme we used for solving Problems 2 and 1 is already able to provide satisfactory results in terms of objective evaluation. However, because the connectivity of communities is not explicitly required in the design of genetic representation, interpretation or algorithm implementation, the output solutions directly generated from the evolutionary algorithm sometimes still contain communities that are not connected. In a typical solution that fails to ensure connectivity there is one sole node separated from other community members. To solve this issue we designed an extra solution improvement step containing two stages. In the first stage, community with more than one connected components is assigned with new community labels for each of the extra components. This often results in small communities with one sole node or two nodes. Then in the second stage, this small community is merged to its most frequent neighbouring community. Details of this repair operator is shown in Algorithm 6.

Algorithm 6: Repair operator

Input: Individual L_i, adjacency matrix A of the whole network

Output: Individual L_i after improvement

```
// Stage 1: assign disconnected community with different labels for each connected component
```

labels $\leftarrow$ unique elements of L_i ;

```
// new label starts from n+1 to avoid overlap with all original labels
```

newLabel $\leftarrow n+1$;

for $j \leftarrow 1$ *to* $length(labels)$ **do**

  get subgraph A_j of the j-th community $labels(j)$;

  get all the k connected components $\{V_1, V2, ..., V_k\}$ in A_j ;

  if $k > 1$ **then**

    for $ii \leftarrow 2$ *to* k **do**

      $L_i(V_k) \leftarrow newLabel$;

      newLabel ++ ;

    end

  end

end

```
// Stage 2: merge one-node or two-nodes community to neighbouring community
```

labels $\leftarrow$ unique elements of L_i ;

for $j \leftarrow 1$ *to* $length(labels)$ **do**

  if *size of j-th community is no larger than two* **then**

    find community labels of all neighbouring nodes ;

    reassign j-th community with the most frequent neighbouring community label ;

  end

end

return L_i

IV. Experimental Studies

We implemented MUMI using MATLAB V9.2. We evaluated the algorithm on a set of benchmarks networks, and then applied it to two real-world biological networks. All experiments were executed on a laptop with Intel Core i5-6200U CPU@2.30GHz and 8GB RAM.

A. MUMI can identify accurate community structure

To evaluate the performance of our algorithm on the community detection task, we used some social networks as benchmark networks which includes Zachary's karate club network, [33], Dophins network [34], [35], politics books network [36] and football network [37]. We compared the value of modularity Q from this algorithm and other classic modularity optimisation algorithms. To perform active module identification tasks in our MUMI algorithm, we need to calculate the active module scores. To the end, we randomly assigned random values as node weights. Since networks have difference size, we need to adopt different algorithm parameters, which are listed in Table I.

TABLE I: Algorithm parameters

	Karate	Dolphins	Politics Books	Football
population	100	200	20	20
generation	100	150	20	20

Experimental results of the community structure detection task are shown in Table II and Figure 2. Our results have shown that the performance of MUMI on these networks is comparable to classic modularity optimisation algorithms.

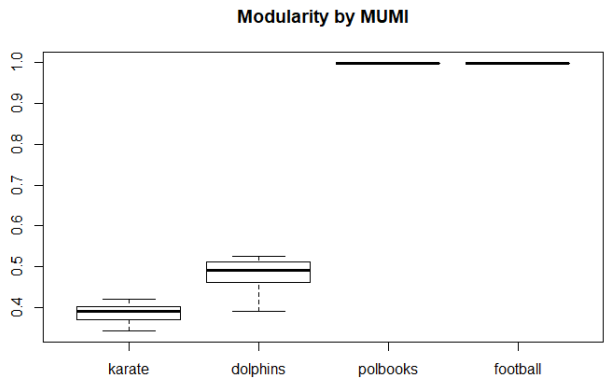


Fig. 2: Boxplot of modularity by MUMI on 4 networks in 50 independent runs. Due to the stochastic nature of evolutionary algorithm, results slightly vary for each run except for the Politic Books network and Football network, both of whom have strong community structure that can be precisely discovered every time.

B. MUMI can identify informative biological modules

We then applied MUMI to two real-world biological networks where the node activity are measured. The aim is to show that MUMI can provide more informative biological interpretation by combining the active modules with communities.

network	size	GN	MNC	Louvain	spectral	MUMI (50 runs)		
						min	max	average
Karate	34	0.4013	0.3807	0.4188	0.3934	0.3431	0.4198	0.3876
Dolphins	62	0.5194	0.4955	0.5158	0.4912	0.3910	0.5265	0.4849
Politics Books	105	0.9977	0.9977	0.9977	0.9977	0.9977	0.9977	0.9977
Football	613	0.9984	0.9984	0.9984	0.9984	0.9984	0.9984	0.9984

TABLE II: Comparison of modularity score obtained by MUMI over 50 independent runs with those from several published algorithms. All the results for published algorithms are from running corresponding functions implemented in igraph package in R. References and the function names of these algorithms are: GN [21], cluster_edge_betweenness; MNC [38], cluster_fast_greedy; Louvain [39], cluster_louvain; spectral [40], cluster_leading_eigen.

1) Yeast galactose utilization pathway (YGUP): We used the small Yease PPI interaction network used in [23]. The network consists of 330 genes (nodes) which represents the galactose utilization pathway in yeast. Each node score is calculated from a gene expression experimental data which describes the significance of each observed change in expression of the 330 genes. We run MUMI with 100 individuals and maximum 1000 generations. We also executed 50 independent runs.

Figure 3 visualises the modules identified by MUMI, which are overlaps between an active module, i.e., the giant connected components whose nodes in diamond shape and with black border and small communities denoted with different colours. The active module is the one with the highest active module score from the 50 runs of MUMI. This figure shows the overlaps between active modules and communities essentially are the fractions or sub-modules of the large active module.

Our GO analysis shows that, of all the 13 sub-modules, 10 have significant functional enrichment (See Table III). Every fraction has only one top level GO term or a small set of closely related terms, which show each fraction has specific biological interpretation. For example, the fraction 63 is specialised in glycolytic process, fraction 126 is targeted in galactose catabolic process via UDP-galactose, and fraction 54 in glutamine family amino acid metabolic process, all of those are highly relevant to galactose metabolic process. Other sub-modules might not be directly related to the process, but serve as an assistance or as essential cellular activities, such as vesicle fusion from fraction 77, response to heat from fraction 348, and regulation of reproductive process from fraction 355.

To investigate whether structure information, i.e., communities alone can provide the same accurate interpretation, we performed GO analysis on each of the 42 communities (Results are listed in supplementary table). Of all the 42 communities, 20 have no significant annotation. We selected three representative communities with significant annotations in Table IV. The results show that the annotations for each community are too general or ambiguous due to many mixed function terms. As a comparison, the active module sub-modules from our algorithm with the same labels have only one annotated function each as shown in Table III. It is clear that communities cannot reflect the biological activity the system is going through accurately. The main reason is that communities fail to incorporate activities, e.g., differential expression information to reveal the essential function changes of the system.

Furthermore, to investigate whether active modules alone can provide the same accurate interpretation, we performed Gene Ontology (GO) analysis [41] on the active modules identified by our methods and jActiveModule, respectively. The results are shown in V and VI. As gene ontology (GO) terms are given in a hierarchical structure, we selected the top level of GO terms to display. From these two tables, both algorithms identified modules relevant to the yeast galactose utilisation activities in the experiments. However, because the whole modules from both algorithms consist large number of genes, the GO annotation terms are too general, which cannot provide specific interpretation for the active modules.

2) Yeast drug reaction network (YDRN): To further test the performance of propose algorithm on real world biological networks, we applied it to another yeast drug reaction network which we have used in our previous work [28]. It was constructed from differential analysis and interactome mapping, containing 1803 nodes and 3356 edges. This network reflects reaction of yeast to diclofenac, a widely used analgesic drug that can cause serious adverse drug reactions. As this is a relatively large scale network, algorithm parameters are set to be 300 individuals and 3000 generations, allowing for enough generation towards convergence.

MUMI has identified an active module with module score 91.80 and a network division with modularity 0.5636. As a comparison, the highest active module score our previous algorithm was 91.01 [28], and the

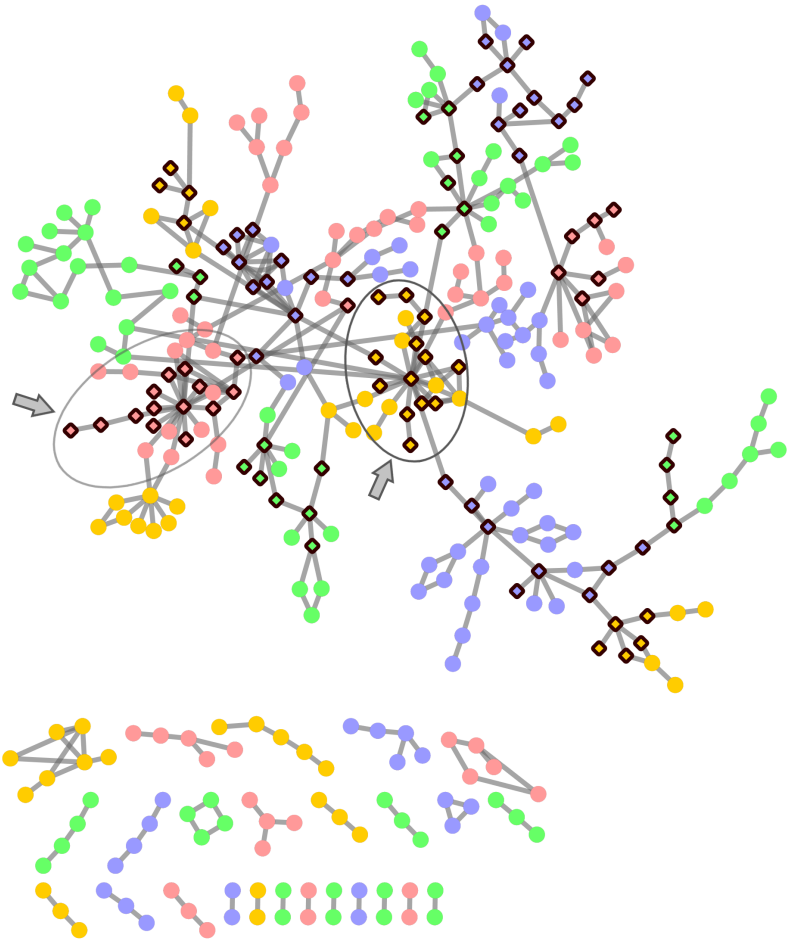


Fig. 3: Visualisation of modules identified by MUMI on the Yeast galactose utilization pathway (YGUP). The overlaps between active modules and communities essentially are the fractions or sub-modules of the large active module. The active module is shown by nodes in diamond shape and with black border, which is further divided into sub-modules according to topological community information. Two circles accompanied with arrows show two representative sub-modules. Gene ontology analysis show that these sub-modules, or active sub-modules, have more precise annotations than the whole active module (See the main text). The active module has a score of 549.9, and the modularity Q of this division is 0.8708. Due to the large number of communities generated in this network, node colouring is based on the rule that nodes in neighbouring communities are filled with different colours while disconnected communities may share the same colour. As a result, this network division with 42 communities is labelled by 4 colours.

modularity given by the Lovain algorithm in R igraph package ranges from 0.5148 to 0.5941.

We performed GO analysis for biological process on the sub-modules identified by MUMI. Supplementary table III shows the 7 sub-modules that have GO annotations. Sub-modules 4 is annotated with ”drug export” and ”plasma membrane acetate transport”, and fraction 5 with ”drug export” and ”positive regulation of cellular response to drug”. In comparison with the GO results on the whole active modules (supplementary table II and III), the active module, with 51 nodes, has more than 30 terms covering a broad range of essential activities in cell. Again we show that the sub-modules identified by MUMI have more precise biological meaning.

C. Multitasking is better than single tasking in terms of accuracy and convergence

To validate our hypothesis that our multitasking formulation can gain better module identification performance in terms of accuracy and convergence, we compare the results from MUMI with the single tasking algorithm with the same genetic operators and parameters. The single tasking algorithm essentially solve each

TABLE III: Gene ontology annotations of the sub-modules in active module divided by community structure in the Yeast galactose utilization pathway (YGUP). It uses the label set generated directly form original results which can be looked up through GitHub repository given in the introduction. Size gives the number of nodes in each fraction. From this table we can see that the functional annotation becomes more specific and clear, i.e., every fraction has only one top level GO term or a small set of closely related terms.

label	size	Typical GO terms	p-value
126	11	galactose catabolic process via UDP-galactose	1.11×10^{-04}
335	8	regulation of protein dephosphorylation	5.67×10^{-04}
		glycogen metabolic process	6.91×10^{-03}
		regulation of mitotic sister chromatid segregation	4.68×10^{-02}
77	11	vesicle fusion	5.93×10^{-04}
54	7	glutamine family amino acid metabolic process	5.95×10^{-04}
348	5	response to heat	2.88×10^{-03}
332	3	romatic amino acid family catabolic process to alcohol via Ehrlich pathway	3.74×10^{-03}
		L-phenylalanine catabolic process	5.38×10^{-03}
		glycolytic fermentation to ethanol	5.38×10^{-03}
		tryptophan catabolic process	1.49×10^{-02}
		branched-chain amino acid catabolic process	1.81×10^{-02}
355	13	regulation of reproductive process	4.21×10^{-03}
271	4	negative regulation of macroautophagy	4.74×10^{-03}
		negative regulation of glycogen biosynthetic process	4.74×10^{-03}
		negative regulation of sequence-specific DNA binding transcription factor activity	7.47×10^{-03}
63	14	glycolytic process	2.20×10^{-02}
201	6	box C/D snoRNP assembly	4.77×10^{-02}

TABLE IV: Gene ontology annotations of 3 representative communities alone in the Yeast galactose utilization pathway (YGUP). It also uses the label set generated directly form original results. Size gives the number of nodes in each fraction. As a contrast, the three sub-modules labelled as 54, 63 and 77 contain only one precisely described ontology term in Table III. GO terms for all communities are shown in supplementary table.

label	size	Typical GO terms	p-value
54	15	urea cycle	3.14×10^{-02}
		heteroduplex formation	4.61×10^{-02}
		telomere maintenance via recombination	1.58×10^{-02}
		glutamine family amino acid metabolic process	6.34×10^{-03}
		alpha-amino acid biosynthetic process	2.99×10^{-03}
		aromatic compound biosynthetic process	1.35×10^{-02}
		heterocycle biosynthetic process	1.26×10^{-02}
		organic cyclic compound biosynthetic process	1.56×10^{-02}
63	28	cellular response to phosphate starvation	1.01×10^{-02}
		egulation of glycolytic process by positive regulation of transcription from RNA polymerase II promoter	1.72×10^{-02}
		gluconeogenesis	1.62×10^{-04}
		glycolytic process	1.95×10^{-05}
		cytoplasmic translation	2.23×10^{-04}
77	14	urea cycle	1.99×10^{-02}
		'de novo' pyrimidine nucleobase biosynthetic process	1.31×10^{-02}
		arginine biosynthetic process	1.39×10^{-02}

individual tasks as a single objective optimisation (SOO) problem. We select two networks for testing, Karate network, where the node activities are randomly initialised regardless the topological structure. The convergence trends are shown in Figure 4. From Figures 4a and 4b, we can see MUMI and SOO performed similarly on the Karate network. This is because node activities in the network are randomly generated independent to communities. Therefore, multitasking formulation does not provide and benefits. However, from Figures 4c and 4d, compared to SOO, MUMI quickly converged to much higher modularity and active module scores on Yeast galactose utilization pathway (YGUP). Such results not only confirmed our hypotheses but also indicated that the two tasks might share similarity and complement each other. Therefore, our multitasking formulation can exploit the latent complementaries of the two tasks, which might be the reason for the superior performance of MUMI.

TABLE V: Gene ontolog annotations of the whole active module identified by MUMI from the Yeast galactose utilization pathway (YGUP). This module has 93 nodes with a active module score (S_A) of 549.9. *p*-value gives the statistical significance of corresponding GO term's enrichment in the gene set.

Typical GO terms	*p*-value
galactose metabolic process	3.64×10^{-05}
ATP metabolic process	3.78×10^{-03}
energy derivation by oxidation of organic compounds	4.41×10^{-03}
ADP metabolic process	5.62×10^{-03}
carbohydrate catabolic process	1.48×10^{-02}
pyruvate metabolic process	2.73×10^{-02}
response to abiotic stimulus	2.78×10^{-02}
small molecule catabolic process	3.29×10^{-02}
regulation of generation of precursor metabolites and energy	4.15×10^{-02}
cellular carbohydrate metabolic process	4.26×10^{-02}

TABLE VI: Active module score and gene ontology annotations of the modules detected by jActiveModule from the Yeast galactose utilization pathway (YGUP). By default 5 active modules are identified by jActiveModule Cytoscape plugin. The highest active module score S_A is 270.79. As a comparison, S_A of active modules detected by MUMI ranges from 529.8 to 549.9.

module	S_A	typical GO terms	*p*-value
1	270.79	galactose catabolic process via UDP-galactose	4.85×10^{-05}
2	169.89	galactose catabolic process via UDP-galactose	1.15×10^{-04}
		cellular carbohydrate metabolic process	3.27×10^{-02}
3	250.39	galactose catabolic process via UDP-galactose	3.42×10^{-04}
		glycolytic fermentation to ethanol	2.72×10^{-03}
		amino acid catabolic process to alcohol via Ehrlich pathway	1.25×10^{-02}
4	58.21	response to heat	2.16×10^{-03}
5	37.05	None	Not available

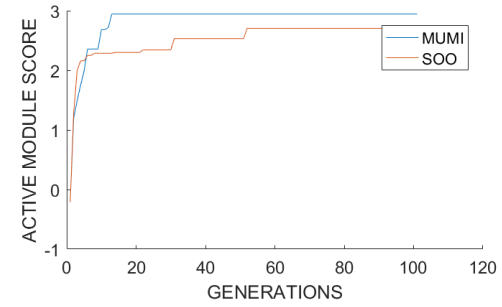
V. Discussion

To confirm our hypothesis we have developed a novel algorithm MUMI and evaluate it on two real-world biological networks. The experimental results on two Yeast molecular networks have confirmed our hypothesis, i.e., the modules identified by MUMI provides new insights into the underlying biological mechanisms that otherwise cannot be discovered through separately identifying each kind of modules. The main reason for the success is MUMI identifies modules by exploiting the complementarities between communities and active modules.

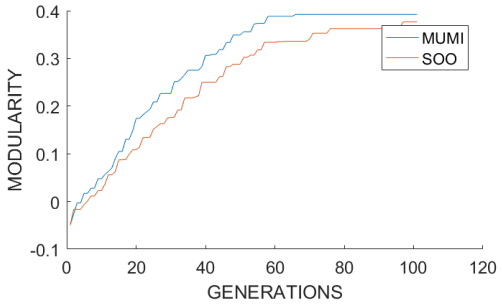
One interesting result is that, compared to the algorithms solving the two tasks individually, our multitasking MUMI algorithm provides better results in terms of accuracy and convergence (see Section IV-C). The main reason is that the two tasks, i.e., identifying communities and active modules, have similar problem structures and hence similar search spaces. Therefore, by unifying the search spaces, we can use genetic operators such as uniform crossover to drive knowledge transfer between the two tasks, which ultimately improve their performance.

However, as the first multitasking module identification algorithm, there is room for improvement. One direction is to improve the efficiency of MUMI to handle large-scale biological networks. Using the parameters and hardware setup specified above, MUMI took a total of 2467 seconds to execute 20 independent runs on the YGUP network with 330 nodes, on average 123 seconds each run. However, it took more than four hours to tackle the YDRN network with 1803 nodes. It is obvious MUMI algorithm does not scale up linearly with the problem size.

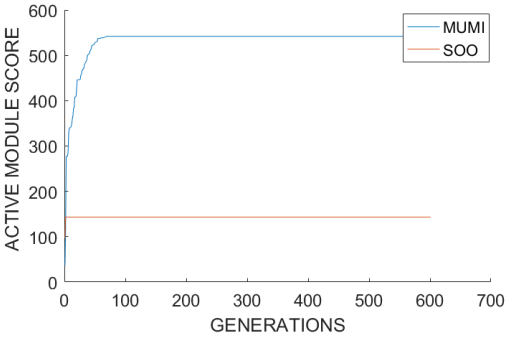
To understand the bottlenecks that reduce MUMI's scalability, we profiled the execution time of 20 runs of the MUMI on the YDRN and YGUP networks. For YGUP with 330 nodes, the biggest bottleneck was the objective function evaluation of the active module identification task, which consumed 47.5% of the total execution time. Further investigation of the objective function showed that, the main reason is the connected components identification algorithm (we used a MATLAB built-in function conncomp()), which consumed 46.0% of the overall execution time. The algorithm is based on Depth First Search with a time complexity of $O(N + E)$, where N and E are the number of nodes and edges, respectively. Therefore, to address this bottleneck, one possible solution is to implement a parallel connected components identification algorithm



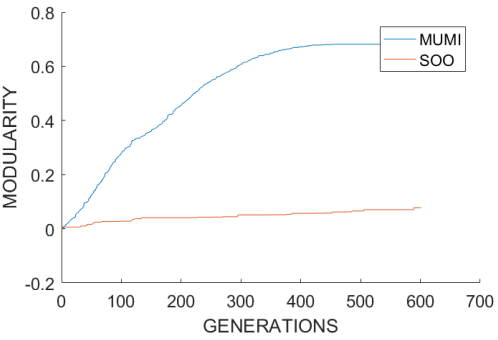
(a) Karate, active module score optimisation



(b) Karate, modularity optimisation



(c) YGUP, active module score optimisation



(d) YGUP, modularity optimisation

Fig. 4: Comparison of convergence trends for two objectives in multitasking optimisation and single objective optimisation. Blue curves correspond to MUMI and red curves represent the results from the single tasking algorithm based on single objective optimisation (SOO).

such as [42] .

For the YDRN network with 1803 nodes, the main bottleneck was the objective function evaluation of the module identification task, which consumed 88.4% of the execution time. The reason is that the modularity Q calculation involves large scale element-wise matrix multiplications, (also known as the Hadamard product), which have the time complexity of $O(N^2)$. Therefore, to tackle this bottleneck, the simplest way is to parallelise the calculation of the Hadamard product. However, we could also follow the idea of the Louvain algorithm [39], i.e., to calculate the modularity change ΔQ that is caused by mutation and crossover of each individual, then update Q using ΔQ and the previously calculated Q. This will be our future work to improve the scalability of our MATLAB implementation.

VI. Conclusion

In this article, we proposed the first multitask module identification (MUMI) algorithmic framework, which provides a new way of using both topological structure and activity information to reveal new insights in biological systems. In addition to this novel framework, we also designed a novel unified genetic encoding and decoding scheme for the multitasks, and task specific genetic operators to improve the performance. Using several benchmark social networks, we have demonstrated the capabilities of MUMI in identifying topological modules. Using two real-world biological networks, we then have showed how MUMI can exploit the latent complementarities of topological and active modules to obtain new insights into the dynamic biological mechanisms.

We expect our MUMI algorithm provides not only a new tool to study biological networks but also new thinkings of studying different aspects of biological networks, which is helpful for understanding the complex and dynamic mechanisms underlying biological systems.

References

[1] A.-L. Barabasi and Z. N. Oltvai, "Network biology: understanding the cell's functional organization," *Nature reviews genetics*, vol. 5, no. 2, pp. 101–113, 2004.

[2] A.-L. Barabási, N. Gulbahce, and J. Loscalzo, "Network medicine: a network-based approach to human disease," *Nature Reviews Genetics*, vol. 12, no. 1, pp. 56–68, 2011.

[3] L. H. Hartwell, J. J. Hopfield, S. Leibler, A. W. Murray *et al.*, "From molecular to modular cell biology," *Nature*, vol. 402, no. 6761, p. C47, 1999.

[4] K. Mitra, A.-R. Carvunis, S. K. Ramesh, and T. Ideker, "Integrative approaches for finding modular structure in biological networks," *Nature Reviews Genetics*, vol. 14, no. 10, pp. 719–732, 2013.

[5] P. Csermely and et. al., "Structure and dynamics of molecular networks: A novel paradigm of drug discovery a comprehensive review," *Pharmacology Therapeutics*, vol. 23, no. 138, pp. 333–408, 2013.

[6] W. A. Lim, C. M. Lee, and C. Tang, "Design principles of regulatory networks: searching for the molecular algorithms of the cell," *Molecular Cell*, vol. 49, no. 2, pp. 202–12, 2013.

[7] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2016.

[8] M. N. Le, Y. S. Ong, S. Menzel, C.-W. Seah, and B. Sendhoff, "Multi co-objective evolutionary optimization: Cross surrogate augmentation for computationally expensive problems," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–8.

[9] L. Feng, Y.-S. Ong, M.-H. Lim, and I. W. Tsang, "Memetic search with interdomain learning: A realization between cvrp and carp," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 644–658, 2015.

[10] L. Feng, Y.-S. Ong, A.-H. Tan, and I. W. Tsang, "Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems," *Memetic Computing*, vol. 7, no. 3, pp. 159–180, 2015.

[11] Y.-S. Ong, M. H. Lim, and X. Chen, "Memetic computation-past, present & future [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 24–31, 2010.

[12] D. Lim, Y.-S. Ong, A. Gupta, C. K. Goh, and P. S. Dutta, "Towards a new praxis in optinformatics targeting knowledge reuse in evolutionary computation: simultaneous problem learning and optimization," *Evolutionary Intelligence*, vol. 9, no. 4, pp. 203–220, 2016.

[13] Y.-S. Ong and A. Gupta, "Evolutionary multitasking: a computer science view of cognitive multitasking," *Cognitive Computation*, vol. 8, no. 2, pp. 125–142, 2016.

[14] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with tsp, qap, lop, and jsp," in *Region 10 Conference (TENCON), 2016 IEEE*. IEEE, 2016, pp. 3157–3164.

[15] A. Gupta, Y.-S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE transactions on cybernetics*, vol. 47, no. 7, pp. 1652–1665, 2017.

[16] R. Chandra, A. Gupta, Y.-S. Ong, and C.-K. Goh, "Evolutionary multi-task learning for modular training of feedforward neural networks," in *International Conference on Neural Information Processing*. Springer, 2016, pp. 37–46.

[17] A. Gupta, J. Mańdziuk, and Y.-S. Ong, "Evolutionary multitasking in bi-level optimization," *Complex & Intelligent Systems*, vol. 1, no. 1-4, pp. 83–95, 2015.

[18] M. E. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.

[19] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010.

[20] M. E. Newman, "Communities, modules and large-scale structure in networks," *Nature physics*, vol. 8, no. 1, p. 25, 2012.

[21] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[22] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[23] T. Ideker, O. Ozier, B. Schwikowski, and A. F. Siegel, "Discovering regulatory and signalling circuits in molecular interaction networks," *Bioinformatics*, vol. 18, no. suppl 1, pp. S233–S240, 2002.

[24] T. Hwang and T. Park, "Identification of differentially expressed subnetworks based on multivariate anova," *BMC bioinformatics*, vol. 10, no. 1, p. 1, 2009.

[25] M. Klammer, K. Godl, A. Tebbe, and C. Schaab, "Identifying differentially regulated subnetworks from phosphoproteomic data," *BMC bioinformatics*, vol. 11, no. 1, p. 1, 2010.

[26] D. Muraro and A. Simmons, "An integrative analysis of gene expression and molecular interaction data to identify dysregulated sub-networks in inflammatory bowel disease," *BMC bioinformatics*, vol. 17, no. 1, p. 1, 2016.

[27] D. Li, Z. Pan, G. Hu, Z. Zhu, and S. He, "Active module identification in intracellular networks using a memetic algorithm with a new binary decoding scheme," *BMC genomics*, vol. 18, no. 2, p. 209, 2017.

[28] W. Chen, J. Liu, and S. He, "Prior knowledge guided active modules identification: an integrated multi-objective approach," *BMC systems biology*, vol. 11, no. 2, p. 8, 2017.

[29] S. Pounds and S. W. Morris, "Estimating the occurrence of false positives and false negatives in microarray studies by approximating and partitioning the empirical distribution of p-values," *Bioinformatics*, vol. 19, no. 10, pp. 1236–1242, 2003.

[30] M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, and T. Müller, "Identifying functional modules in protein–protein interaction networks: an integrated exact approach," *Bioinformatics*, vol. 24, no. 13, pp. i223–i231, 2008.

[31] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the royal statistical society. Series B (Methodological)*, vol. 57, pp. 289–300, 1995.

[32] C. Pizzuti, "Ga-net: A genetic algorithm for community detection in social networks," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2008, pp. 1081–1090.

[33] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.

[34] D. Lusseau, "The emergent properties of a dolphin social network," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 270, no. Suppl 2, pp. S186–S188, 2003.

[35] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.

[36] V. Krebs, "Books about us politics. compiled by valdis krebs." http://www.orgnet.com/, accessed Feb 2018.

[37] 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.

[38] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[39] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[40] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[41] "Gene Ontology Consortium," http://geneontology.org/, accessed Apr 2016.

[42] C. Jain, P. Flick, T. Pan, O. Green, and S. Aluru, "An adaptive parallel algorithm for computing connected components," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 9, pp. 2428–2439, Sep. 2017.