

RESEARCH ARTICLE

Open Access

HIDEN: Hierarchical decomposition of regulatory networks

Günhan Gülsoy*, Nirmalya Bandhyopadhyay and Tamer Kahveci

Abstract

Background: Transcription factors regulate numerous cellular processes by controlling the rate of production of each gene. The regulatory relations are modeled using transcriptional regulatory networks. Recent studies have shown that such networks have an underlying hierarchical organization. We consider the problem of discovering the underlying hierarchy in transcriptional regulatory networks.

Results: We first transform this problem to a mixed integer programming problem. We then use existing tools to solve the resulting problem. For larger networks this strategy does not work due to rapid increase in running time and space usage. We use divide and conquer strategy for such networks. We use our method to analyze the transcriptional regulatory networks of *E. coli*, *H. sapiens* and *S. cerevisiae*.

Conclusions: Our experiments demonstrate that: (i) Our method gives statistically better results than three existing state of the art methods; (ii) Our method is robust against errors in the data and (iii) Our method's performance is not affected by the different topologies in the data.

Background

Genes are the smallest functional units of an organism. They carry out vital functions in cells by interacting with each other and with other molecules. Biological networks model such interactions among genes. Using biological networks, researchers are able to take a holistic approach on the analysis of cellular functions. Such analysis has shown that biological networks have a number of global properties. One of these properties is their hierarchical organization. Hierarchical organization defines a partial ordering of the underlying genes. Recent studies have shown that directed interactions between transcription factors (TFs) in transcriptional regulatory networks (TRNs) impose a hierarchy on TRNs [1-5]. Analysis of the hierarchies of TRNs helps researchers better understand the flow of controlling signals through the transcription machinery [1,3]. TFs are special types of proteins that control the expression of other genes by binding to specific regions of the DNA [6]. Since each protein is coded by genes, we will use the terms *transcription factor* and *gene* to refer to TFs throughout this paper. One way to

model hierarchy in TRNs is to assign levels to the interacting TFs. Figure 1 shows a sample network with its level assignments. In this paper, we consider the problem of finding the hierarchical organization of a TRN. The formal definition of our problem in this paper is as follows:

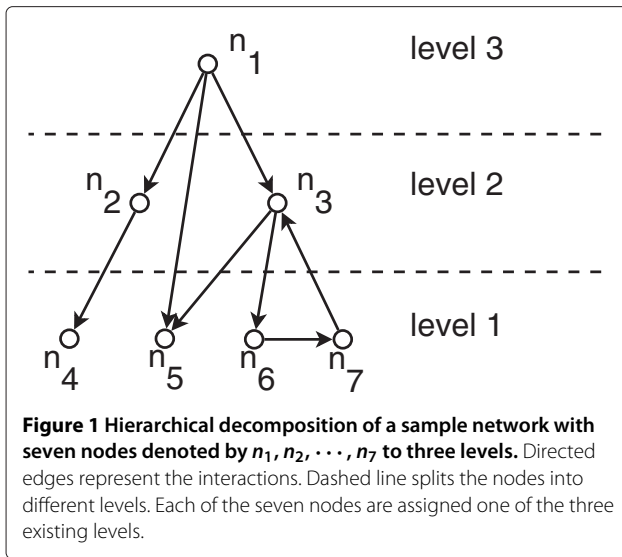
Problem definition

Let us denote a TRN with $\mathcal{G} = (N, E)$. Here, N denotes the set of TFs and E denotes the set of directed interactions (i.e. edges) between the TFs in N . We refer to each TF in N as a *node*. We name the i th node in N as n_i . We represent an edge from the node n_i to n_j with (n_i, n_j) . Also, we denote the maximum possible number of levels in \mathcal{G} with M . We denote the hierarchy level assigned to a node n_i with t_i where t_i is an integer in $\{1, 2, 3, \dots, M\}$. Let $\phi(n_i, n_j) \rightarrow \{0, 1\}$ be a binary function that describes the key topological relationship between n_i and n_j . (We elaborate on the ϕ function below.) We compute a penalty score p_{ij} for each pair of nodes as follows:

$$p_{ij} = \begin{cases} \phi(n_i, n_j), & \text{if } t_i \leq t_j \\ 0, & \text{else} \end{cases}$$

Our aim is to find an assignment of hierarchies to the nodes of N which minimizes $\sum_{(i,j)} p_{ij}$.

*Correspondence: ggulsoy@cise.ufl.edu
Computer and Information Sciences and Engineering, University of Florida,
Gainesville, FL 32611, USA



In this paper, we use two different ϕ functions describing two key topological properties.

1. **Adjacency.** We define $\phi(n_i, n_j) = 1$ if $(n_i, n_j) \in E$ and $\phi(n_i, n_j) = 0$ otherwise.
2. **Reachability.** We define $\phi(n_i, n_j) = 1$ if there exists a path from n_i to n_j in \mathcal{G} by traversing the edges in E . We set $\phi(n_i, n_j) = 0$ otherwise.

Depending on the choice of the two ϕ functions, we name the resulting distance function adjacency distance or reachability distance respectively. In summary, using adjacency distance we aim to assign levels such that every TF is above the others it directly regulates, and below its every direct regulator. On the other hand, using reachability distance, we consider any direct or indirect regulation relation between two TFs when assigning levels.

There has been attempts to devise methods to reveal the underlying hierarchies of TRNs. Yu and Gerstein developed BFS-level method to carry out this task [1]. This method uses breadth first search to assign hierarchies to TFs in a network. Although their method works for most networks, it fails to assign accurate levels for networks that contain cycles. Jothi *et al.* developed vertex sort method [2]. This method incorporates topological sort algorithm for addressing the network hierarchy problem. Vertex sort method does not have any restrictions on network motifs or cycles. However, rather than a certain hierarchy, it assigns a range of possible levels for the TFs. Hartsperger *et al.* devised an algorithm based on breadth first search method to solve the problem [3]. Their solution improves the BFS-level method, and outputs a hierarchy for every network regardless of its topological features. However all these algorithms fail to minimize the number of edges that violate the hierarchy. We name such edges as *conflicting edges*. We will

elaborate on the quality of results calculated by existing methods in Section Comparison with existing hierarchical decomposition methods.

Contributions

In this paper, we develop a novel approach to tackle the problem of discovering underlying network hierarchy. We first consider the topology of the network as a set of constraints. Then, we define two different objective functions using adjacency and reachability penalty functions. We define the minimization of total penalty as the objective of the problem. Using the above explanations, we transform this problem to a mixed integer programming problem (MIPP) [7]. We solve the resulting problem using existing MIPP solvers. We name our method **HI**erarchical **DE**composition of regulatory **N**etworks (**HIDEN**). The main advantage of HIDEN is it introduces a sound mathematical formulation to the network hierarchy problem. Our formulation can work with any objective function that is a linear combination of the edges. One drawback of HIDEN is that it does not scale well to very large networks due to the growing size of the MIPP with increasing number of TFs. In order to address this issue we develop a divide and conquer approach.

The rest of this article is organized as follows: In Section Algorithm, we describe the methods we developed in this paper. In Section Results and discussion, we discuss the results of HIDEN in detail. Finally, in Section Conclusion, we briefly conclude the paper.

Method

In this section, we describe the hierarchical decomposition method we developed. Section HIDEN describes our method. Section Example demonstrates HIDEN on a simple example. Section Divide and Conquer method describes divide and conquer method we employ to scale HIDEN to larger networks.

HIDEN

HIDEN transforms the hierarchical network decomposition problem to a MIPP [7]. Given a TRN, HIDEN first constructs a set of linear constraints and a linear optimization function that collectively describe the penalty of the decomposition. Then it uses existing optimizer software to solve the resulting problem. Next, we will explain how we formulate the MIPP.

Let us denote the given network that will be decomposed with \mathcal{G} . Let us denote the nodes (i.e., TFs) of this network with n_1, n_2, \dots, n_m , where m is the total number of nodes of \mathcal{G} . HIDEN, allows the user to set a limit on the maximum number of allowed levels for hierarchical decomposition. Let us denote this number with M . Also, let us represent the level assigned to node n_i with t_i for all $i \in \{1, 2, \dots, m\}$, (i.e. $\forall i, t_i \in \{1, 2, 3, \dots, M\}$).

We aim to find the level assignment $T = \{t_1, t_2, \dots, t_m\}$ that minimizes the total penalty resulting from this level assignment. Therefore, the objective of our problem is the sum of individual penalty scores for each pair of nodes:

$$\text{minimize } \sum_{1 \leq i, j \leq m} p_{ij}. \quad (1)$$

Next, we set a limit on the number of levels in the hierarchy. We do this by limiting the variables t_i as follows:

$$0 \leq t_i < M. \quad (2)$$

We, then, represent each p_{ij} as a linear constraint. Remember that p_{ij} is a binary function in the following form:

$$p_{ij} = \begin{cases} \phi(n_i, n_j), & \text{if } t_i \leq t_j \\ 0, & \text{else} \end{cases}$$

We can rewrite this function as follows:

$$p_{ij} = \begin{cases} 1, & \text{if } t_i \leq t_j \text{ and } \phi(n_i, n_j) = 1 \\ 0, & \text{else} \end{cases}$$

Let us only consider the cases where $\phi(n_i, n_j) = 1$. We can represent the rest of this function using two linear inequalities. The following set of constraints represent the function p_{ij} :

$$p_{ij} \in \{0, 1\} \quad (3)$$

$$t_j - t_i - M \times p_{ij} \geq -M \quad (4)$$

$$t_j - t_i - M \times p_{ij} \leq -1 \quad (5)$$

In order to prove that these inequalities model the function p_{ij} correctly, we need to inspect all possible scenarios:

1. if $t_i > t_j$ and $p_{ij} = 0$, then $-1 \geq t_j - t_i \geq -(M - 1)$ and $M \times p_{ij} = 0$. Therefore both (4) and (5) holds.
2. if $t_i \leq t_j$ and $p_{ij} = 0$, then $t_j - t_i \geq 0$ and $M \times p_{ij} = 0$. Therefore, (4) holds, however, (5) does not hold.
3. if $t_i > t_j$ and $p_{ij} = 1$, then $-1 \geq t_j - t_i \geq -(M - 1)$ and $M \times p_{ij} = M$. Therefore the expression $t_j - t_i - M \times p_{ij}$ is smaller than or equal to $-M - 1$. This implies that (4) does not hold but (5) holds.
4. if $t_i \leq t_j$ and $p_{ij} = 1$, then $(M - 1) \geq t_j - t_i \geq 0$ and $M \times p_{ij} = M$, therefore both (4) and (5) holds.

Therefore, enforcing the constraints (3), (4) and (5) implies:

$$(p_{ij} = 0 \Leftrightarrow t_i > t_j) \wedge (p_{ij} = 1 \Leftrightarrow t_i \leq t_j) \quad (6)$$

This corresponds to the latter definition of the function p_{ij} except the condition of $\phi(n_i, n_j) = 1$. Since we choose the function ϕ prior to the construction of the MIPP, we know the value of $\phi(n_i, n_j)$ for every pair (n_i, n_j) . Therefore, we can manually ensure this property, by only

considering p_{ij} where $\phi(n_i, n_j) = 1$ and excluding p_{ij} completely from our calculations where $\phi(n_i, n_j) = 0$.

Based on the constraints above, the MIPP we construct to solve the network hierarchy problem is as follows:

$$\text{minimize } \sum_{i, j \text{ s.t. } \phi(n_i, n_j)=1} p_{ij}$$

where

$$\forall n_i$$

$$0 \leq t_i < M$$

t_i is an integer

$$\forall n_i, n_j \text{ such that } \phi(n_i, n_j) = 1$$

$$p_{ij} \in \{0, 1\}$$

$$t_j - t_i - M \times p_{ij} \geq -M$$

$$t_j - t_i - M \times p_{ij} \leq -1$$

Example

In this section, we show the application of HIDDEN on the network in Figure 1. We will use adjacency penalty function in this example. Therefore:

$$\phi(n_i, n_j) = \begin{cases} 1 & \text{if an edge from } n_i \text{ to } n_j \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

Using this ϕ function, the objective of the MIPP is to minimize the following function:

$$\sum_{i, j \text{ s.t. } \phi(n_i, n_j)=1} p_{ij} = p_{12} + p_{13} + p_{15} + p_{24} + p_{35} + p_{36} + p_{67} + p_{73}$$

Now we go over to the constraints. First set of constraints limit t_i :

$$\forall n_i, i \text{ from } 1 \text{ to } 7$$

$$0 \leq t_i < M$$

t_i is an integer

Then, we write the remaining functions as follows:

$$\forall (n_i, n_j) \in \{(n_1, n_2), (n_1, n_3), (n_1, n_5), (n_2, n_4),$$

$$(n_3, n_5), (n_3, n_6), (n_6, n_7), (n_7, n_3)\}$$

$$p_{ij} \in \{0, 1\}$$

$$t_j - t_i - M \times p_{ij} \geq -M$$

$$t_j - t_i - M \times p_{ij} \leq -1$$

In the resulting problem, M is left as a user defined parameter. When we run the above problem with $M = 4$, HIDDEN returns the following result:

$$p_{12} + p_{13} + p_{15} + p_{24} + p_{35} + p_{36} + p_{67} + p_{73} = 1$$

$$(t_1, t_2, t_3, t_4, t_5, t_6, t_7) = (4, 3, 3, 2, 2, 2, 1)$$

Figure 2 shows the result of HIDDEN on the given network. Note that HIDDEN computes the level decomposition successfully despite the existence of a cycle in the network.

Divide and Conquer method

HIDDEN works well for networks that have up to 100 nodes. For larger networks, however, it becomes difficult to solve the resulting MIPP using current hardware. This is mainly because the number of integer variables of the MIPP that describe the problem for the given network increases. This increases the memory consumption and the running time significantly.

In order to solve our problem for networks that have more than 100 nodes we adopt a divide and conquer approach. Given a large TRN, we randomly divide this network into fixed size partitions. We do this by first randomly selecting a node from the given network. This node is the seed of the first partition, and thus it is a member of that partition. We then chose the remaining nodes in that partition iteratively by randomly growing the partition one node at a time. More specifically, at each iteration, we randomly select a node that is not selected so far and that is interacting with at least one of the nodes in the partition. We repeat these iterations until the number of nodes in the partition reaches to a predefined threshold or all the nodes in the TRN are assigned to a partition. Then, we use HIDDEN to decompose the subnetwork defined by the nodes and the edges in this partition into hierarchical levels. Once we determine the levels of all the nodes in the

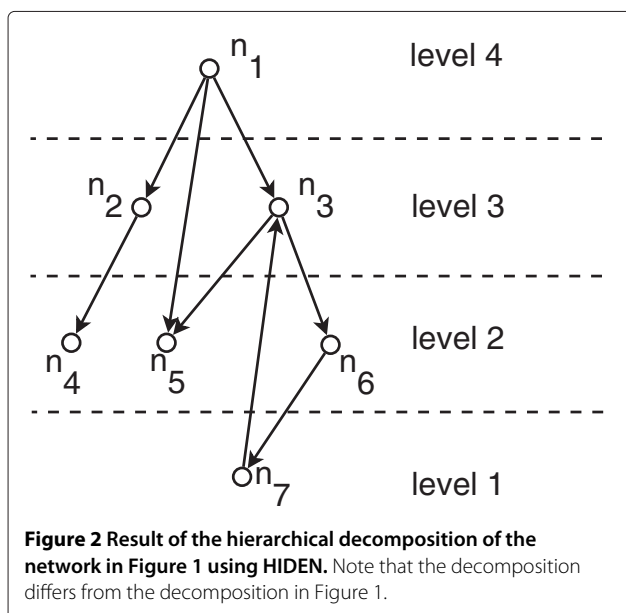
current partition, we store those values as they will remain unchanged in the rest of our solution. Next we randomly pick another node from the given TRN among those that have not been considered yet as the seed of the next partition. We grow the next partition similarly and use HIDDEN to decompose it into hierarchical levels. We repeat these steps until we exhaust all the nodes in the given TRN.

This method greatly reduces the running time of HIDDEN on large networks. Since MIPP is NP-hard, depending on the size and the connectivity of the given TRN, the divide and conquer strategy can be orders of magnitude faster than the unpartitioned HIDDEN. However, due to random selection of the nodes, it is possible for us to not achieve the optimal result. This is possible if the partition of the network we start with does not intersect with one or more of the levels in its underlying hierarchy. It is worth mentioning that this probability is usually very low. We can explain this as follows. Consider an N node network which contains n nodes belonging to a specific level x. If we select k nodes among these N nodes randomly, the probability that none of the k nodes belong to level x is $(N-n \text{ choose } k) / (N \text{ choose } k)$. As k or n increases, this expression quickly converges to zero. In order to reduce this probability further, we repeat the divide and conquer strategy multiple times, each time starting from a randomly selected node. In our implementation, we repeat this process 1000 times for real TRNs. After 1000 iterations, the probability of all the trials starting with an undesired (i.e. does not intersect with all the final levels) partition becomes very small (i.e. if for 1 iteration, the probability is as high as 0.9, after 1000 iterations, the probability becomes $0.9^{1000} \sim 10^{-46}$). Since the running time of partitioned HIDDEN is orders of magnitude less than that of the unpartitioned HIDDEN, 1000 repetitions remains to be practical. It took less than 10 minutes for the largest dataset (*S. cerevisiae*). Our experiments showed that on the average, the results of the divide and conquer method reach its optimum in less than 500 iterations.

Results and discussion

In this section, we evaluate HIDDEN using a number of computational tests. In our tests, we let the underlying MIPP solver to handle the case of multiple optimal results. We only consider the unique result reported by the solver in our discussions. In the rest of this paper, we will use the term *experiment* to refer to *in silico* experiments for simplicity.

DATASET In our experiments, we used TRNs of *E. coli*, *H. sapiens* and *S. cerevisiae*. We used the previously constructed networks, used by existing methods to test our method [1-3,5]. Earlier studies used existing interaction data to construct these three networks [8-17]. For the experiments that require gene function information,



we used the information included in the Gene Ontology Database [18]. We downloaded the list of essential genes for *S. cerevisiae* from the database of essential genes [19].

In the rest of this section, we first compare HIDDEN with other existing hierarchical decomposition methods in Section Comparison with existing hierarchical decomposition methods. In Section Biological evaluation of network hierarchies we evaluate the results our method using a number of biological properties of TFs. Finally in Section Effects of input on HIDDEN, we analyze the behavior of our algorithm with respect to different quantitative properties of the data.

Comparison with existing hierarchical decomposition methods

The objective of hierarchical decomposition is to arrange the TFs of a given network to levels so that the gene that alter the activity of the other appears at a higher level than the other throughout the network as frequently as possible. The two ϕ functions described at the beginning of this paper model this relationship in terms of the adjacency and the reachability of the nodes in the given network. In this experiment we evaluate how well our method, HIDDEN, compares against three state of the art methods, namely *vertex sort* [2], *HiNO* [3] and *BFS-level* [1], in achieving this objective. To perform this comparison, we compute the penalty values obtained by HIDDEN when it is applied on *S. cerevisiae*, *E. coli* and *H. sapiens* networks. We compute the same penalty values for the vertex sort, HiNO and BFS-level methods on the same three datasets for which their hierarchical decompositions are available.

The penalty is a quantitative value that can be used to compare different methods on the same dataset. However, since the size (number of genes and interactions) and the topology of these networks deviate significantly, the resulting penalties will differ significantly across datasets. In order to report a statistically sound value that describes the success of a method independent of the network size and topology, we also compute the Z-scores of the resulting penalty values.

Let us denote the level assignment obtained by a specific method for an m node network with $T = \{t_1, t_2, \dots, t_m\}$. Let γ denote the penalty of T according to a specific ϕ function. In order to compute the Z-score for T , we randomly produce many level assignments using the same level distribution as that of T . For each such assignment, we compute the resulting penalty value using the same ϕ function. Let μ and σ denote the mean and standard deviation of the resulting penalty values of all these random level assignments respectively. We calculate the Z-score as follows,

$$z = \frac{\mu - \gamma}{\sigma} \quad (7)$$

A higher Z-score implies a better level assignment. Typically, a Z-score of four or higher is very significant as they indicate a result which is 4 or more standard deviations more extreme than the mean Table 1 summarizes the penalties and the corresponding Z-score values. For HIDDEN, we reported the results for each of the six values of maximum number of levels ($M = \{3, 4, \dots, 8\}$). For other methods the number of levels is not a configurable parameter. Hence, we reported the level that we obtained after execution of that method. We discuss the results for each organism next.

S. cerevisiae

We compared HIDDEN with all the three competing methods for this dataset. Our method outperformed all the three methods in terms of both adjacency and reachability penalty values as well as the Z-scores regardless of the number of levels. As the number of levels allowed increases, the penalty incurred by HIDDEN monotonically decreases. This, however, is not true for the Z-score as it depends on the distribution of nodes to levels. For instance HIDDEN attains the highest Z-score for adjacency penalty at level eight whereas it attains that using only six levels for the reachability penalty. The biggest drop of penalty takes place when the number of allowed levels increases from three to four. We observe further, yet, smaller improvement in the penalty as the number of allowed levels increases beyond four.

Among the competing methods, the vertex sort method of Jothi *et al.* incurs the lowest penalty. It, however, uses significantly more levels than the HiNO and BFS-level methods. Furthermore, although it uses more levels than HIDDEN as well, it performs worse than HIDDEN in terms of both penalty and Z-score measures. Among the remaining two methods, HiNO and BFS-level, there is no clear winner. BFS-level leads to slightly less penalty at the expense of an additional level. As a result, HiNO produces slightly better Z-scores.

E. coli

For this dataset, we compared HIDDEN with all three existing methods. The penalty values of all the methods for *E. coli* are smaller compared to those of *S. cerevisiae*. This is mainly because *E. coli* network is much smaller. HIDDEN performs the best among all methods for four or more levels according to both penalty and Z-score values. We did not observe any improvement for HIDDEN beyond seven levels. Vertex sort attains statistically better results than HiNO and BFS-level methods.

H. sapiens

We compared HIDDEN with vertex sort and BFS-level methods for this dataset. We omitted HiNO in this experiment because we could not run it on this dataset. The

Table 1 Comparison of HIDDEN with three other methods on different networks

Organism	Method	Num. Level	Adjacency		Reachability	
			Penalty	Z-score	Penalty	Z-score
Yeast	HIDDEN	3	140	10.8	3600	13.9
		4	103	10.8	3027	14.6
		5	88	9.5	2774	14.1
		6	91	10.2	2573	13.4
		7	79	9.8	2469	12.8
		8	79	11.6	2365	13.7
	vertex sort	9	179	7.3	3920	10.2
	BFS-level	4	245	6.0	5734	9.9
	HiNO	3	279	6.8	6205	10.5
E. coli	HIDDEN	3	15	6.3	19	7.1
		4	8	6.5	10	6.8
		5	5	6.4	7	6.7
		6	5	6.2	6	6.6
		7	5	6.2	5	6.6
		8	5	6.2	5	6.6
	vertex sort	6	10	5.7	11	6.5
	BFS-level	4	44	3.7	65	5.1
	HiNO	4	41	4.2	59	5.3
Human	HIDDEN	3	101	7.4	1950	9.4
		4	84	7.4	1608	10.6
		5	75	7.9	1435	10.8
		6	66	7.9	1347	10.2
		7	72	7.5	1287	9.7
		8	72	7.3	1248	9.9
	vertex sort	5	207	1.2	2162	5.8
	BFS-level	3	210	0.72	2163	6.0

^aFor HIDDEN we vary the maximum allowable level from three to eight. We report the adjacency and reachability penalties as well as the Z-scores for these penalties for each experiment. "Num. Level" denotes the maximum number of allowed levels. The results for HiNO on human are omitted, because of problems in execution.

results follow a similar pattern as those of the two other datasets. HIDDEN outperformed vertex sort and BFS-level even when it used fewer levels. The gap between the Z-scores of HIDDEN and the other methods was even more significant than the previous datasets. HIDDEN led to the highest drop of penalty of from three to four levels and continued to improve with increased number of levels.

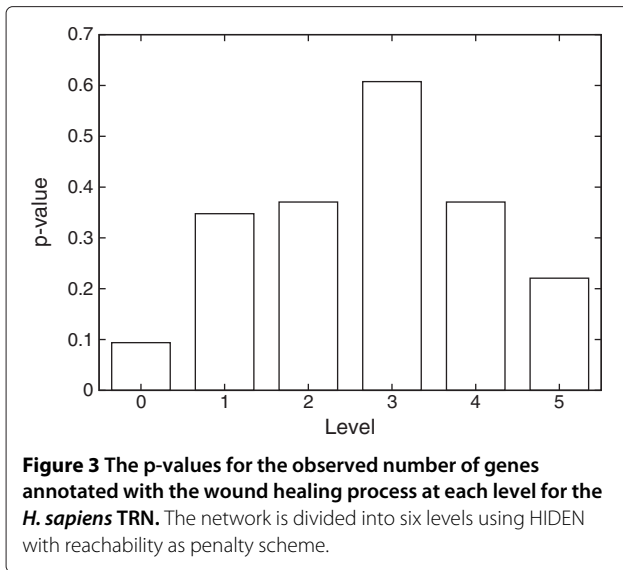
We conclude that, HIDDEN performs significantly better than the competing methods for all the three major datasets.

Biological evaluation of network hierarchies

In this section, we analyze HIDDEN using biological evidence. First, we check functional properties of genes across different levels. Then, we evaluate the locations of essential genes in the hierarchy.

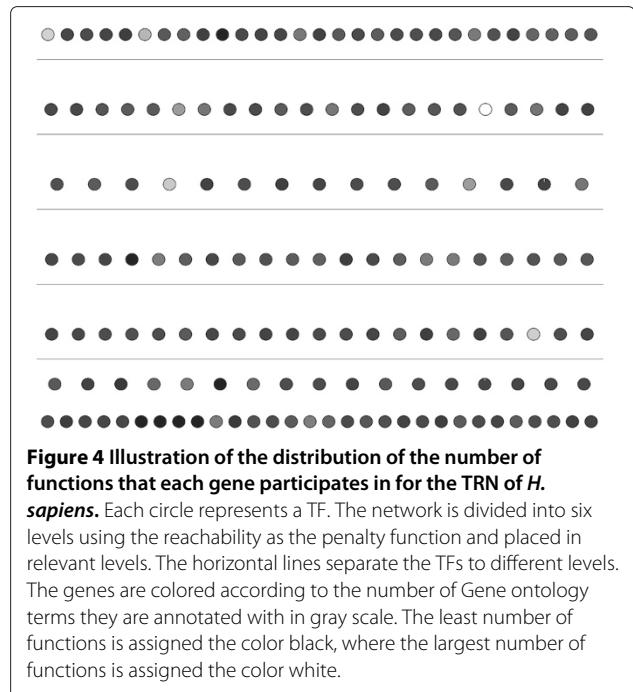
Functions of genes

TRNs regulate the expression of genes that take part in many processes in an organism [13]. Earlier works suggest that the concentration of genes participating in certain functions are closely related to the level in the hierarchy [1]. However, majority of cellular functions in the cell take place through multiple reactions happening in succession. Therefore, we expect a uniform distribution of functions among different levels. In order to confirm this theory, we calculated the functional enrichment score of every single level in the hierarchies we discovered. We first decomposed each of the *H. sapiens*, *E. coli* and *S. cerevisiae* TRNs to each of the three to eight levels. Then, for the resulting 18 combinations (i.e., 3 organisms and 6 levels), we calculated a p-value for each gene ontology term and level pair. We obtain the gene ontology terms from the Gene Ontology database [18]. We calculate these p-values assuming a hypergeometric distribution of the



gene ontology term over all the TFs in the network. We observed that the p-values were similar at all levels of the hierarchy (see Figure 3). This supports our initial theory that majority of the functions the TFs in our network participate are not enriched at any level. One example among many is the *wound healing* function in human network [20]. However, in rare instances, we observed some functions being moderately enriched in some levels. For example, third of the eighth level (the third level when we decompose the network into eight levels) of human TRN is enriched with the *signal transduction* function. However, we could not detect any other levels in any other network enriched with this function.

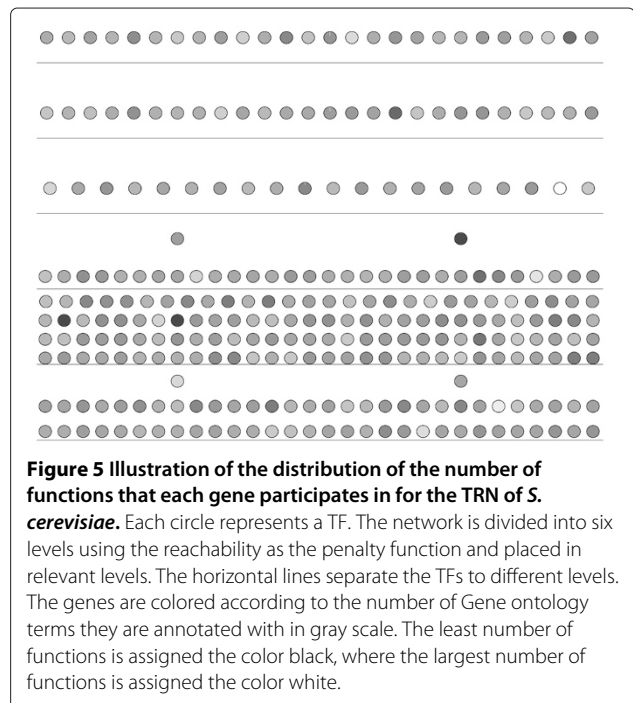
Each gene in an organism takes part in at least one metabolic function. A gene participating in a large number of reactions is a common phenomena in many organisms. In this experiment, we compare the level of each gene with the number of functions they participate in. By doing so, we aim to discover any existing relation between the two. In order to do this, we use the gene ontology database [18]. The participation of a gene in a reaction is represented using gene ontology annotations in the literature. For each gene in our networks, we first count the number of gene ontology terms it is annotated with. We also decomposed each network into six layers using HIDDEN. Then, we visualized the networks using hierarchy information as location and functional information as color of each node. Figures 4, 5 and 6 show our results. Our results suggest that there is no strong correlation between the number of functions of each gene and the level of the gene in the hierarchy. However, in all three organisms, we observed that the genes with the highest number of annotations tend to lie at the middle levels (i.e. 2,3 or 4). This result indicates that regulatory hubs in the

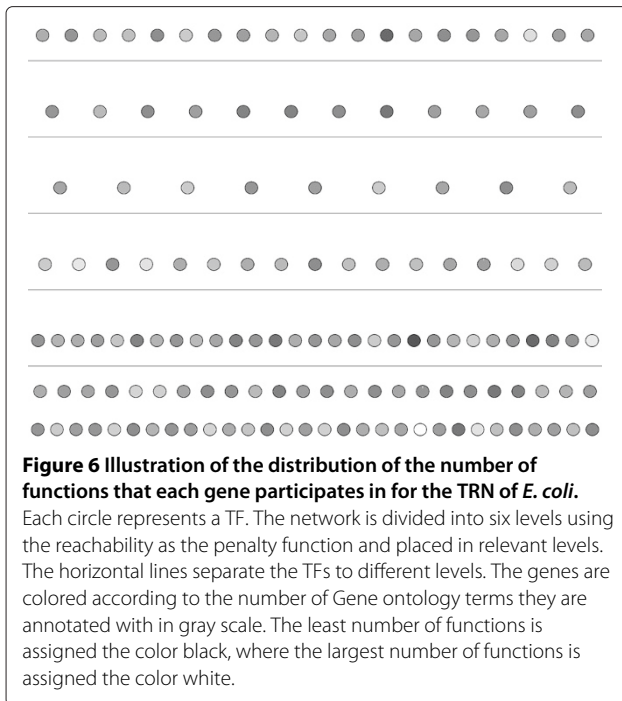


TRNs are not at the top levels. They are rather at the middle levels of the hierarchy.

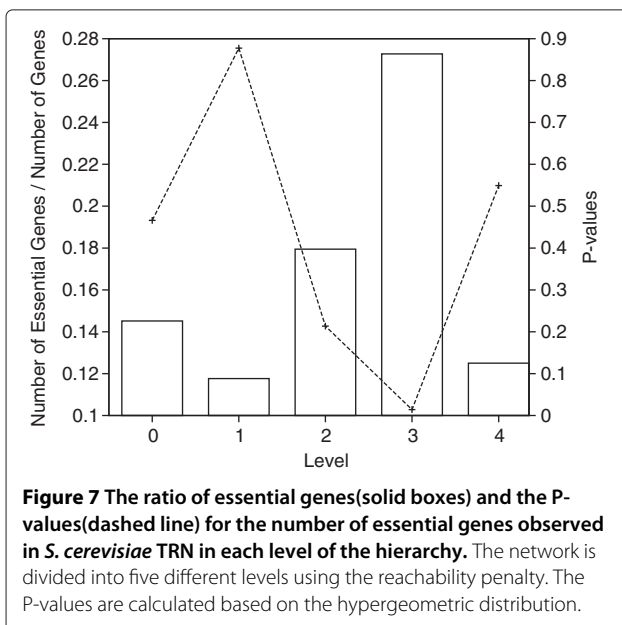
Gene Essentiality

The genes which an organism cannot survive without are called essential genes [19]. Such genes take part in vital functions in the cell. Earlier works proposed that the number of essential genes is strongly correlated to its location in the hierarchy [2]. In this experiment, we aim to find out





if there exists any such relation. In order to do this, we count the number of essential genes at each level of hierarchy in a five level decomposition of *S. cerevisiae* TRN. We then report the ratio of number of essential genes to total number of genes in a level in the hierarchy. We also calculate P-values for the number of genes observed in each level to show how significant the observations are. Figure 7 shows our results for this experiment. We observe that there is a higher density of essential genes at the middle levels of the hierarchy. We also observe that, the P-values we calculated show that the level three (of maximum four)



is highly enriched in essential genes. This result, combined with the results of the previous experiment support our theory that regulatory hubs of an organism are often at the middle levels of the hierarchy, rather than the top level. Indeed strong correlation between hubs and essentiality has been observed in the literature that supports our results [21].

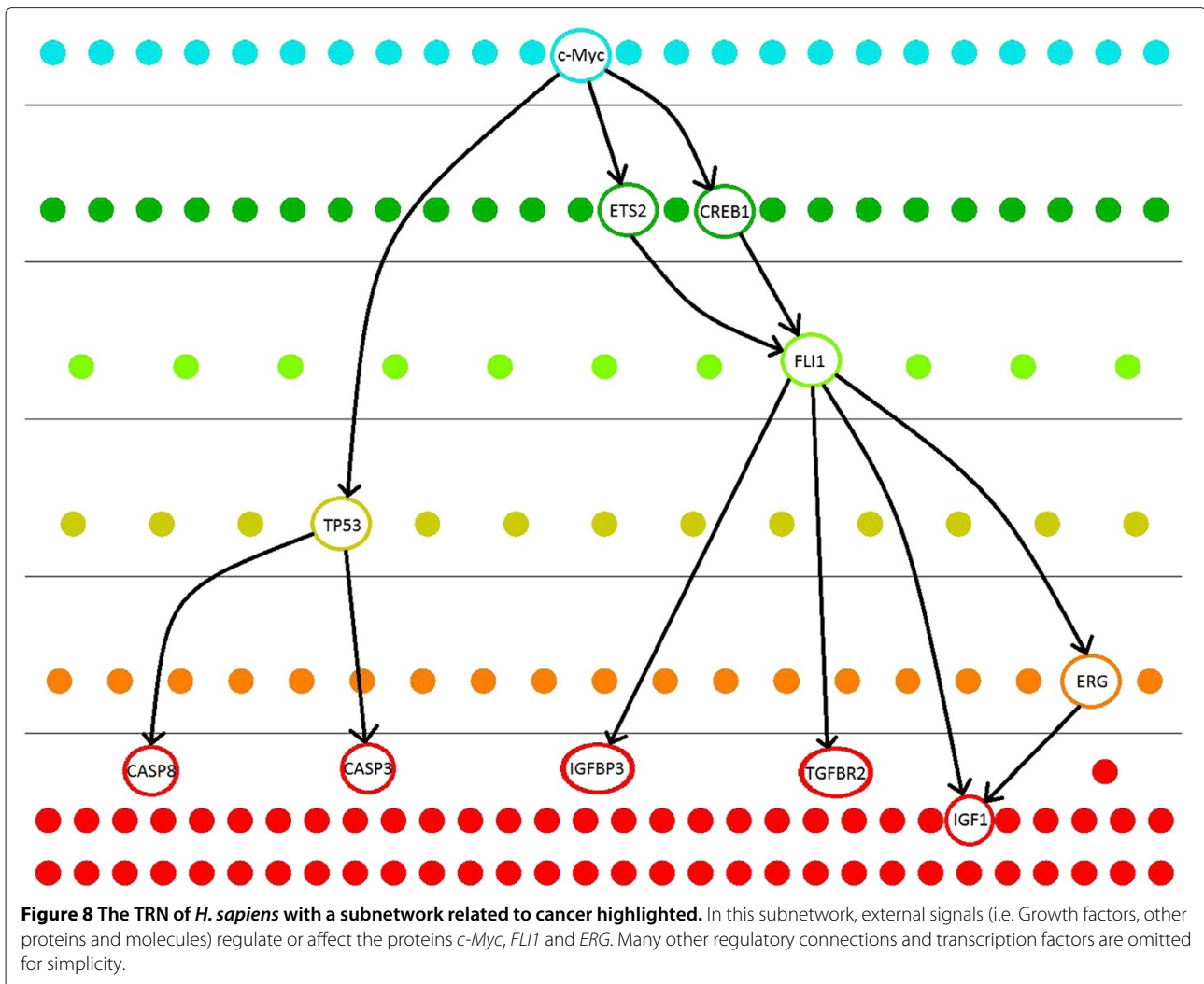
Figure 8 shows a subnetwork of the human TRN. The highlighted TFs are shown to have abnormalities in many types of cancers. *c-Myc* is a TF which has a key role in cell proliferation [22]. Overexpression of *c-Myc* may result in development of different types of cancers. *TP53* is an essential gene which is regulated by *c-Myc*. The expression of this gene prevents formation of tumors by activating DNA repair, inhibiting cell growth and finally inducing apoptosis [23]. *TP53* executes apoptosis by activating caspases (i.e. *CASP8*, *CASP3*) [24]. *FLI1* is another protein regulated indirectly by *c-Myc*. The fusion of proteins *EWSR1/FLI1* and *EWSR1/ERG* due to a mutation creates a master regulator for the development in *Ewing's Sarcoma* [25,26]. *EWSR1/FLI1* causes tumor formation by upregulating genes that are involved in cell proliferation (i.e. *IGF1*) and downregulating genes that control apoptosis and growth inhibition (i.e. *IGFBP3*, *TGFBR2*) [27]. These small scale observations support our previous justifications that regulatory hubs and essential genes are more likely to be situated in the middle layers of the TRNs.

Effects of input on HIDDEN

In this section, we analyze HIDDEN by changing the input of the algorithm. In order to do this, we first change the number of layers we decompose the network into. Then, we assume errors and uncertainties in input networks. Using our results, we explain how reliable our method is under different conditions. Finally, we discuss the quality of our results for different subnetworks.

Navigation of genes across levels in varying hierarchies

The location of a gene in the hierarchy depends highly on the total number of levels. This leads to the following important question: How much can we rely on the relative levels of genes? One key feature of our method is that it allows the user to specify the number of levels in the hierarchical decomposition of the given network. By exploiting this feature, next, we answer this question. Particularly, we show how the change the number of levels affect the locations of the nodes in the hierarchy. In order to do this, we first calculate the levels of every node for *S. cerevisiae*, *E. coli* and *H. sapiens* networks in a six level hierarchy. We use these calculations to place every node in their respective positions. We then decompose the same networks to five levels. We use the result of the second decomposition to assign colors to each node in the network. Figure 9 shows the results of this experiment for



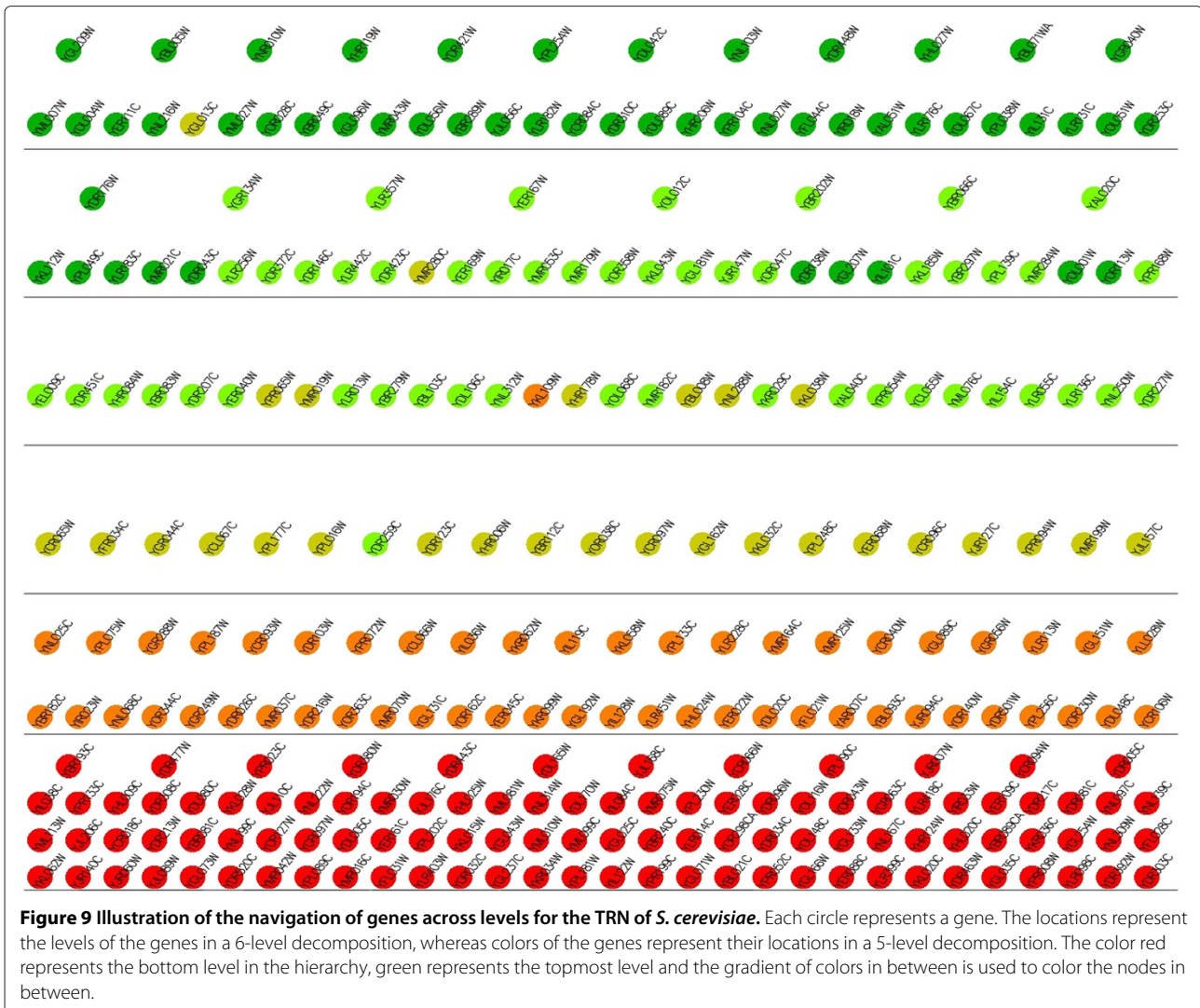
S. cerevisiae. Our results demonstrate that for the majority of the genes, the relative position between different genes is preserved. In different decompositions, discovering genes in the same relative positions with respect to other genes suggest the accuracy of our method for the relevant genes. However, there exists some genes that violate this relationship. For example, in Figure 9, nodes *YGL013C*, *YMR280C* and *YKL109W* are at least two levels away from where they were earlier. Therefore, we conclude that the predicted levels of these genes not as reliable as the others. *This approach can be used for evaluating the reliability of our results.* Figures 10 and 11 present similar results for *E. coli* and *H. sapiens*.

Robustness of HIDDEN

One weakness of all hierarchical decomposition methods arises from the nature of the biological network datasets that they are incomplete and imprecise. As a result, the actual network topology observed can be slightly different than what is given in existing network databases

[28]. Furthermore, studies demonstrate that the network topologies can vary due to many other factors such as external perturbations [29] and varying genetic profiles and disorders [30] even within the same species. This raises the question that how much can we rely on a hierarchical decomposition if the topology of the given network is not accurate?

This section evaluates the *robustness* of HIDDEN with respect to inaccuracies in the given network. In order to do this, we carry out the following steps. Given a network, we first find its hierarchical decomposition, denoted by *T*. We then create many mutant networks from this network for a given mutation percentage using the *degree preserving edge shuffling* model [31]. This is the state of the art network alteration method that preserves the degree distribution of the network. We elaborate on this method later in this section. Thus, each mutant network denotes a potential precise network that is different than the original network. Using the topology of each mutated network, we compute the penalty of the hierarchical decomposition



T we found at the first step. Thus, this penalty shows how bad our results are if our network is inaccurate. We repeat this for many mutant networks and report the average of their penalties.

Briefly, we mutate a given network \mathcal{G} as follows. Let (u, v) and (s, t) denote two randomly selected edges from this network such that (i) u and v are different; s and t are different, and (ii) the edges (u, t) and (s, v) do not exist in \mathcal{G} . We remove edges (u, v) and (s, t) and add edges (u, t) and (s, v) . Let η denote the number of times we repeat this edge shuffling process. Then the mutation percentage of the original network is $\frac{\eta}{|E|} \times 100\%$ rounded to the nearest integer.

We conducted the experiments on *S. cerevisiae*, *E. coli* and *H. sapiens* and on both penalty metrics adjacency and reachability for different number of levels of hierarchy. Figure 12 summarizes the results for *S. cerevisiae*, *E. coli* and *H. sapiens* using the adjacency and reachability penalties when three, six or eight levels are allowed.

The most important observation that follows from our results is that the Z-score remains high even after we mutate the network by 20%. We observe a slight drop as the mutation rate increases, yet the results remain statistically significant. This observation holds for small (3), medium (6) and large (8) number of allowed hierarchical levels. This result has two major implications. First, HIDDEN is extremely robust with respect to network mutations. It was able to identify hierarchical structure using the clues that remain in the topology of the given network after all mutations take place. Thus, even if the original network may be imprecise, the decomposition found by HIDDEN will be a true decomposition with a high probability. Second, the degree preserving edge shuffling does not affect the decomposability of the network. The fact that even the original level assignment T yielded statistically significant penalties on the mutant network proves that it is possible to find another decomposition T' of the mutant network



that is statistically at least as significant as (possibly more significant than) *T*.

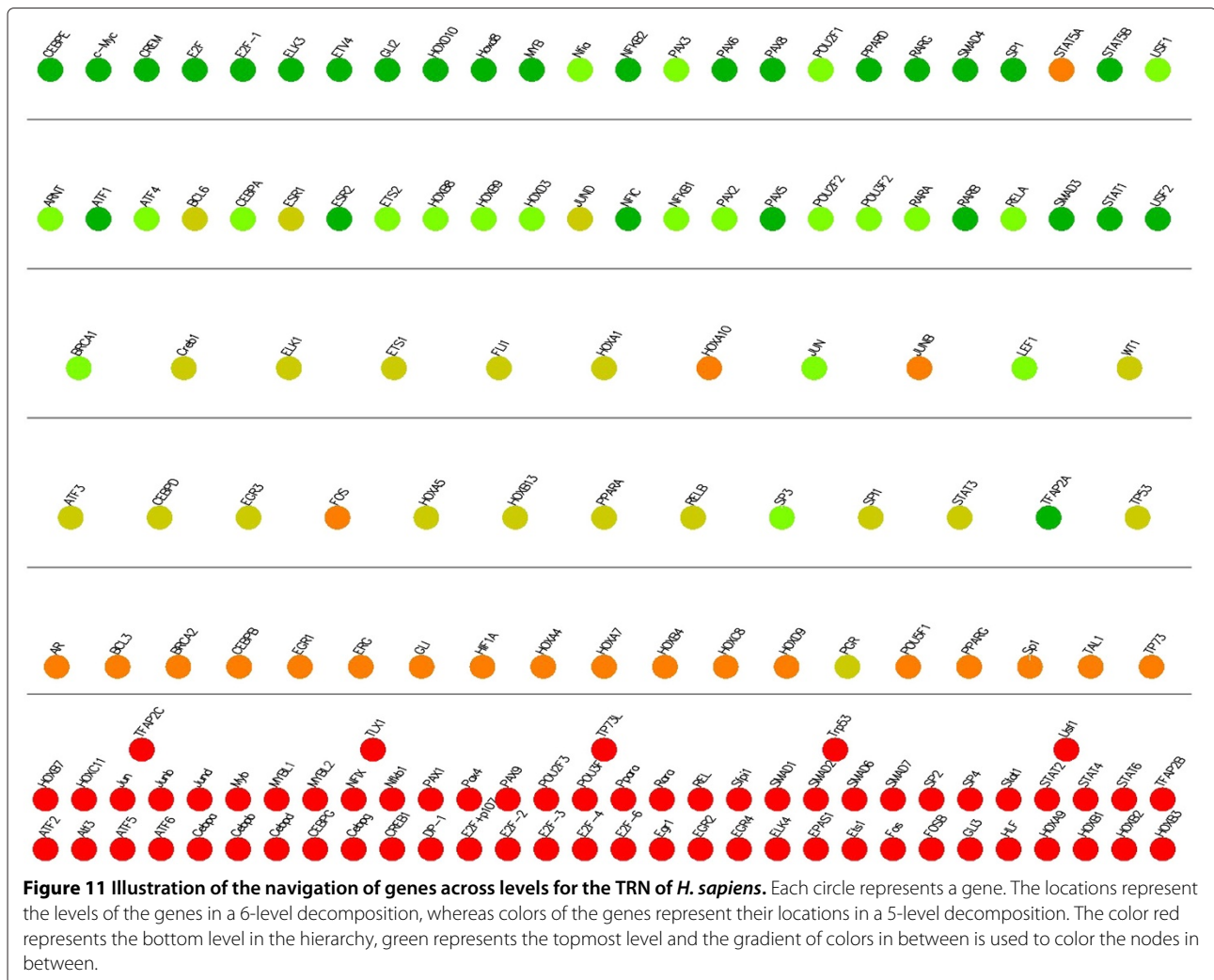
Stability of HIDDEN to network mutations

So far, we have observed that HIDDEN was able to decompose the networks of the given three organisms successfully. This observation along with our last conclusion from the previous section begs the following question: Can HIDDEN decompose the mutant networks or was there a bias in topology of these three networks in favor of HIDDEN? In other words, how stable is HIDDEN with respect to alterations in the network topology?

In order to evaluate the *stability* of HIDDEN with respect to network alterations, we carry out the following steps. Given a network \mathcal{G} , we create many mutant networks \mathcal{G}' from \mathcal{G} for a given mutation percentage using the degree preserving edge shuffling. We then use HIDDEN on each such \mathcal{G}' to find a new hierarchical level assignment T' specifically for that topology. Thus, this penalty shows how much the performance of HIDDEN

is affected from network alterations. We repeat this for many mutant networks and report the average of their results.

Tables 2, 3 summarize the penalties and the corresponding Z-scores for varying mutation percentages as well as varying maximum number of allowed hierarchical levels with according to adjacency and reachability penalties respectively. For all the three organisms, we observe similar patterns in our experiments. The most important observation is that HIDDEN achieves very high Z-scores at all mutation rates. Furthermore, these Z-scores are comparable to those of the original network (i.e., mutation percentage = 0%). The adjacency penalty values are also comparable to those for the original network. This coincides with the observation we made in the robustness test in Section Robustness of HIDDEN that the degree preserving edge shuffling does not alter the decomposability of the given network. As the mutation percentage increases, Z-score and the adjacency penalties do not show a clear trend to increase or decrease. We, thus, reach



to the conclusion that HIDDEN is stable with respect to network alterations.

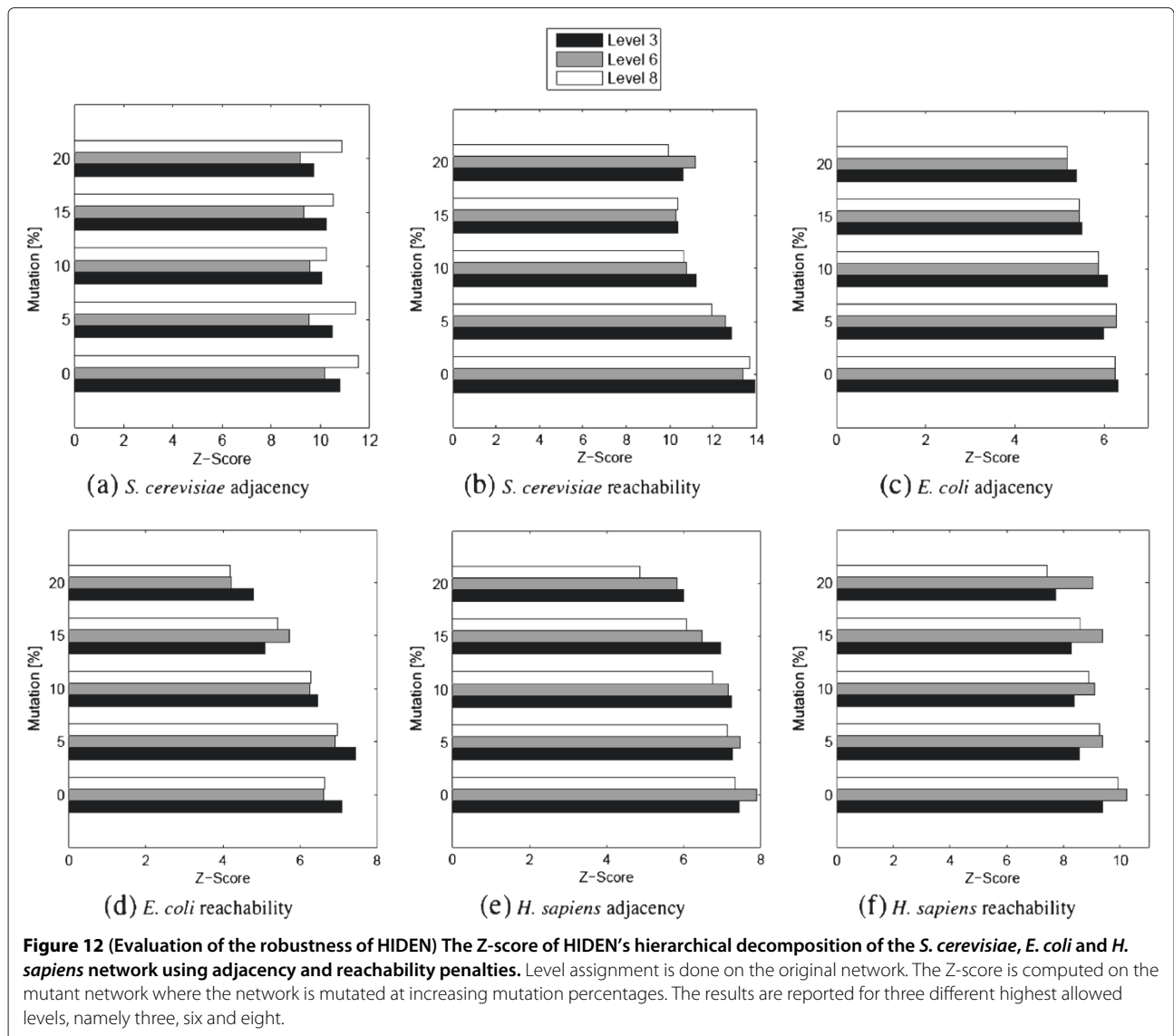
Local versus global hierarchy of subnetworks

The entire biological network of an organism can be considered as a (possibly overlapping) collection of smaller subnetworks where each subnetwork corresponds to a coherent functional group. For instance, cell cycle network describes the interactions that take place during the division and replication of a cell to produce new cells. Similarly, meiosis network describes a special type of cell division only observed in reproductive cells. These smaller subnetworks may follow a hierarchical structure as well within their local topologies. Clearly, we can use HIDDEN on each of these subnetworks to find their hierarchical structure by isolating them from the rest of the network one by one. We call such hierarchical decomposition as *local hierarchy* since it only depends on the topology of the subnetwork. We call the hierarchical decomposition we

obtain for a subnetwork from the entire network's topology as its *global hierarchy*. In this experiment, we evaluate how well the global hierarchy of a subnetwork fits to its local hierarchy.

Let us denote the entire network with \mathcal{G} and a subnetwork of \mathcal{G} with \mathcal{G}' . Let us denote the level assignments for the networks \mathcal{G} and \mathcal{G}' by HIDDEN with T and T' respectively. Let $\hat{T} \subseteq T$ be the global hierarchy of \mathcal{G}' induced from T . We calculate the adjacency penalty and Z-score of \hat{T} and T' using the topology of \mathcal{G}' . Table 4 summarizes the results for *S. cerevisiae* for two major subnetworks, namely *cell cycle* and *meiosis* taken from the KEGG database [32] with different values of maximum number of allowed levels.

The results demonstrate that the local hierarchy is better than the global one. This is not surprising as the global hierarchy is determined based on the entire network. Thus, the levels are determined with the goal of optimizing all the interactions in the network. On the



other hand, local hierarchy is determined only based on the restrictions asserted by the corresponding subnetwork. We observe that the gap between the local and the global hierarchy is small for the cell cycle network. It is, however, significant for the meiosis network. In order to understand the factors that contribute to this gap, we performed a detailed analysis of the topology of the entire *S. cerevisiae* network as well as these two subnetworks. Cell cycle contained 54 genes and 108 interactions. Meiosis was smaller, containing 44 genes and 62 interactions. We define an interaction from a gene that is not in the subnetwork to a gene that is in the subnetwork as an *incoming edge*. If the interaction points the opposite direction, we define it as an *outgoing edge*. We computed the number of incoming and outgoing edges to each subnetwork. The number of incoming edges per node was 1.9 and 3.6 for

cell cycle and meiosis respectively. That for the outgoing edges was 20.6 and 18.8 respectively. This suggests that as the number of incoming edges increase, the chance that the global hierarchy deviates from the local one increases. This is indeed intuitive as the incoming edges influence the hierarchy much more than the outgoing edges. For the local hierarchy, we observe that a small number of levels is sufficient to get a good hierarchical decomposition. For instance, HIDDEN resolved all conflicts for cell cycle in only five levels. It resolved all but one conflict for meiosis in four levels.

These results demonstrate that the local and global hierarchies can deviate significantly depending on the topological relationship between the subnetwork and the rest of the network. Thus, detailed analysis of both decompositions can yield useful information regarding how the

Table 2 Stability Experiment for increasing mutation percentages: The numbers in parenthesis is the average adjacency penalty

Organism	Level	Mutation [%]				
		0	5	10	20	40
Yeast	3	9.10 (118)	11.63 (137)	10.17 (127)	12.12 (127)	10.10 (130)
	4	9.31 (99)	11.20 (117)	11.30 (103)	11.37 (114)	10.91 (104)
	5	9.26 (84)	10.96 (108)	10.97 (92)	11.62 (103)	10.67 (88)
E. coli	3	5.76 (17)	4.98 (16)	5.26 (22)	4.98 (16)	5.34 (15)
	4	5.43 (11)	4.77 (15)	5.67 (14)	4.77 (15)	5.57 (10)
	5	5.46 (9)	4.72 (15)	5.52 (12)	4.72 (15)	5.34 (8)
Human	3	7.44 (101)	9.24 (105)	8.66 (95)	7.79 (106)	9.14 (107)
	4	7.37 (84)	9.09 (90)	8.14 (83)	8.22 (92)	8.98 (92)
	5	7.90 (75)	8.83 (93)	7.70 (81)	8.53 (73)	9.33 (86)

^aThe numbers above them is the corresponding Z-score. Level indicates the maximum number of allowed levels.

functions of a given subnetwork is depends on the other genes. HIDEN is capable of revealing such information.

Conclusion

In this paper, we took a novel approach to the problem of discovering underlying network hierarchy. We first transformed our problem to a MIPP. Then, we solved this problem using existing optimizers. We named this method **H**ierarchical **D**ecomposition of gene regulatory **N**etworks. However, due to the growing size of the MIPP with increasing number of genes, we encountered scalability issues. We proposed a divide and conquer approach to tackle such problems. Later, we experimentally showed that our algorithm outperformed existing solutions in terms of minimizing conflicting edges in hierarchy. We also evaluated our method using biological and statistical tools. Then, we discussed the relation between the hierarchy of a gene in a TRN and its location in cell, essentiality and function, based on our experimental results and biological evidence.

Availability and requirements

The source code for HIDEN can be found in Additional file 1. The code is written in C++. The code requires IBM ILOG CPLEX version 12 or higher to compile and run. Please refer to the documentation of CPLEX for

Table 3 Stability Experiment for increasing mutation percentages: The numbers in parenthesis is the average reachability penalty

Organism	Level	Mutation [%]				
		0	5	10	20	40
Yeast	3	12.35 (3674)	15.21 (3600)	14.62 (3483)	15.20 (3599)	15.31 (3598)
	4	12.33 (3027)	14.74 (3026)	14.47 (2923)	14.73 (3025)	14.66 (3024)
	5	12.27 (2754)	14.18 (2773)	14.29 (2644)	14.18 (2772)	14.18 (2771)
E. coli	3	7.73 (21)	6.58 (26)	6.90 (21)	6.58 (26)	6.37 (27)
	4	7.29 (15)	6.16 (20)	6.21 (15)	6.16 (20)	5.93 (26)
	5	6.95 (14)	6.00 (20)	6.22 (11)	6.00 (20)	5.81 (26)
Human	3	8.25 (1950)	11.17 (1951)	11.17 (1951)	8.66 (1944)	11.17 (1951)
	4	8.88 (1628)	12.02 (1613)	12.02 (1613)	10.45 (1608)	12.02 (16.13)
	5	12.40 (1431)	11.71 (1441)	11.71 (1441)	10.45 (1431)	11.71 (1441)

^aThe numbers above them is the corresponding Z-score. Level indicates the maximum number of allowed levels.

platform specific instructions on how to compile and run applications that use CPLEX libraries. The results of our code using the penalty schemes described in this paper for TRNs of *E. coli*, *H. sapiens* and *S. cerevisiae* can be found in Additional file 2.

Table 4 Comparison of the global hierarchy of subnetworks to their local hierarchy

Subnetwork	Num. Level	Global		Local	
		Penalty	Z-score	Penalty	Z-score
Cell Cycle	3	4	3.2	3	4.2
	4	2	3.3	1	4.0
	5	2	3.2	0	3.7
	6	2	3.1	0	3.7
	7	2	3.0	0	3.7
Meiosis	8	2	2.9	0	3.7
	3	8	0.7	2	3.5
	4	6	1.2	1	3.8
	5	6	1.2	1	3.8
	6	5	1.6	1	3.8
	7	5	1.5	1	3.8
	8	5	1.5	1	3.8

^aThe experiment is conducted on the two subnetworks of *S. cerevisiae*, namely cell cycle and meiosis. "Num. Level" denotes the maximum number of allowed levels.

Additional files

Additional file 1: A c++ implementation of the algorithm developed in this paper.

Additional file 2: The resulting level assignments for the transcriptional regulatory networks of *S. Cerevisiae*, *E. coli* and *H. sapiens* using adjacency and reachability penalties.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

GG and TK designed the method. GG implemented the method. GG and NB gathered experimental results. GG and TK wrote the paper. All authors read and approved the final manuscript.

Acknowledgements

This work was supported partially by NSF under grants CCF-0829867 and IIS-0845439.

Received: 22 March 2012 Accepted: 21 September 2012

Published: 28 September 2012

References

1. Yu H, Gerstein M: **Genomic analysis of the hierarchical structure of regulatory networks.** *Proc Natl Acad Sci U S A* 2006, **103**(40):14724–14731. [http://dx.doi.org/10.1073/pnas.0508637103]
2. Jothi R, Balaji S, Wuster A, Grochow JA, Gsponer J, Przytycka TM, Aravind L, Babu MM: **Genomic analysis reveals a tight link between transcription factor dynamics and regulatory network architecture.** *Mol Syst Biol* 2009, **5**:294. [http://dx.doi.org/10.1038/msb.2009.52]
3. Hartsperger ML, Strache R, Stumpflen V: **HiNO: an approach for inferring hierarchical organization from regulatory networks.** *PLoS One* 2010, **5**(11):e13698. [http://dx.doi.org/10.1371/journal.pone.0013698]
4. Bhardwaj N, Kim PM, Gerstein MB: **Rewiring of transcriptional regulatory networks: hierarchy, rather than connectivity, better reflects the importance of regulators.** *Sci Signal* 2010, **3**(146):ra79. [http://dx.doi.org/10.1126/scisignal.2001014]
5. Bhardwaj N, Yan KK, Gerstein MB: **Analysis of diverse regulatory networks in a hierarchical context shows consistent tendencies for collaboration in the middle levels.** *Proc Natl Acad Sci U S A* 2010, **107**(15):6841–6846. [http://dx.doi.org/10.1073/pnas.0910867107]
6. Latchman SD: **Transcription factors: an overview.** *Int J Biochem and Cell Biol* 1997, **29**(12):1305–1312. [http://www.sciencedirect.com/science/article/pii/S135727259700085X]
7. Garfinkel RS, Nemhauser GL: *Integer programming*. New York: John Wiley & Sons; 1972.
8. Balaji S, Babu MM, Iyer LM, Luscombe NM, Aravind L: **Comprehensive analysis of combinatorial regulation using the transcriptional regulatory network of yeast.** *J Mol Biol* 2006, **360**:213–227. [http://dx.doi.org/10.1016/j.jmb.2006.04.029]
9. Balaji S, Iyer LM, Aravind L, Babu MM: **Uncovering a hidden distributed architecture behind scale-free transcriptional regulatory networks.** *J Mol Biol* 2006, **360**:204–212. [http://dx.doi.org/10.1016/j.jmb.2006.04.026]
10. Gama-Castro S, Salgado H, et al: **RegulonDB version 7.0: transcriptional regulation of *Escherichia coli* K-12 integrated within genetic sensory response units (Sensor Units).** *Nucleic Acids Res* 2011, **39**(Database issue):D98–105. [http://dx.doi.org/10.1093/nar/gkq1110]
11. Harbison CT, Gordon DB, et al: **Transcriptional regulatory code of a eukaryotic genome.** *Nature* 2004, **431**(7004):99–104. [http://dx.doi.org/10.1038/nature02800]
12. Horak CE, Luscombe NM, Qian J, Bertone P, Piccirillo S, Gerstein M, Snyder M: **Complex transcriptional circuitry at the G1/S transition in *Saccharomyces cerevisiae*.** *Genes Dev* 2002, **16**(23):3017–3033. [http://dx.doi.org/10.1101/gad.1039602]
13. Jiang C, Xuan Z, Zhao F, Zhang MQ: **TRED: a transcriptional regulatory element database, new entries and other development.** *Nucleic Acids Res* 2007, **35**(Database issue):D137–D140. [http://dx.doi.org/10.1093/nar/gkl1041]
14. Lee TI, Rinaldi N J, et al: **Transcriptional regulatory networks in *Saccharomyces cerevisiae*.** *Science* 2002, **298**(5594):799–804. [http://dx.doi.org/10.1126/science.1075090]
15. Luscombe NM, Babu MM, Yu H, Snyder M, Teichmann SA, Gerstein M: **Genomic analysis of regulatory network dynamics reveals large topological changes.** *Nature* 2004, **431**(7006):308–312. [http://dx.doi.org/10.1038/nature02782]
16. Svetlov VV, Cooper TG: **Review: compilation and characteristics of dedicated transcription factors in *Saccharomyces cerevisiae*.** *Yeast* 1995, **11**(15):1439–1484. [http://dx.doi.org/10.1002/yea.320111502]
17. Teichmann SA, Babu MM: **Gene regulatory network growth by duplication.** *Nat Genet* 2004, **36**(5):492–496. [http://dx.doi.org/10.1038/ng1340]
18. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G: **Gene ontology: tool for the unification of biology. The gene ontology consortium.** *Nat Genet* 2000, **25**:25–29.
19. Zhang R, Lin Y: **DEG 5.0, a database of essential genes in both prokaryotes and eukaryotes.** *Nucleic Acids Res* 2009, **37**(Database issue):D455–D458. [http://dx.doi.org/10.1093/nar/gkn858]
20. Galko MJ, Krasnow MA: **Cellular and genetic analysis of wound healing in *Drosophila* larvae.** *PLoS Biol* 2004, **2**(8):E239. [http://dx.doi.org/10.1371/journal.pbio.0020239]
21. Jeong H, Mason SP, Barabasi AL, Oltvai ZN: **Lethality and centrality in protein networks.** *Nature* 2001, **411**(6833):41–42. [http://dx.doi.org/10.1038/35075138]
22. Dang CV, Resar LM, Emison E, Kim S, Li Q, Prescott JE, Wonsley D, Zeller K: **Function of the c-Myc Oncogenic Transcription Factor.** *Experimental Cell Research* 1999, **253**:63–77. [http://www.sciencedirect.com/science/article/pii/S0014482799946864]
23. Vousden HK: **p53: death star.** *Cell* 2000, **103**(5):691–694. [http://www.sciencedirect.com/science/article/pii/S0092867400001719]
24. Schuler M, Green DR: **Mechanisms of p53-dependent apoptosis.** *Biochem Soc Trans* 2001, **29**(Pt 6):684–688.
25. Kauer M, Ban J, Kofler R, Walker B, Davis S, Meltzer P, Kovar H: **A molecular function map of Ewing's Sarcoma.** *PLoS One* 2009, **4**(4):e5415. [http://dx.doi.org/10.1371/journal.pone.0005415]
26. Owen LA, Kowalewski AA, Lessnick SL: **EWS/FLI mediates transcriptional repression via NKX2.2 during oncogenic transformation in Ewing's Sarcoma.** *PLoS ONE* 2008, **3**(4):e1965. [http://dx.plos.org/10.1371]
27. Riggi N, Stamenkovic I: **The biology of Ewing sarcoma.** *Cancer Lett* 2007, **254**:1–10. [http://www.sciencedirect.com/science/article/pii/S0304383506006811]
28. Sun N, Carroll RJ, Zhao H: **Bayesian error analysis model for reconstructing transcriptional regulatory networks.** *Proc Natl Acad Sci U S A* 2006, **103**(21):7988–7993. [http://dx.doi.org/10.1073/pnas.0600164103]
29. Bandyopadhyay N, Somaiya M, Kahveci T, Ranka S: **Modeling perturbations using gene networks.** In *Proceedings of Life Sciences Society Computational Systems Bioinformatics (CSB)*; 2010:26–37. [http://www.lifesciencesociety.org/CSB2010/toc/26.2010.html]
30. Bandyopadhyay N, Somaiya M, Ranka S, Kahveci T: **Identifying differentially regulated genes.** In *Computational Advances in Bio and Medical Sciences (ICABS), 2011 IEEE 1st International Conference on*; 2011:19–25. [http://ieeexplore.ieee.org/xpls/abs.all.jsp?arnumber=5729877&tag=1]
31. McSweeney PJ, Ashkenazi M, States D: **Random network plugin.** *Google Summer of Code* 2008:2008. [https://sites.google.com/site/randomnetworkplugin/Home]
32. Ogata H, Goto S, Sato K, Fujibuchi W, Bono H, Kanehisa M: **KEGG: Kyoto encyclopedia of genes and genomes.** *Nucleic Acids Res* 1999, **27**:29–34. doi:10.1186/1471-2105-13-250

doi:10.1186/1471-2105-13-250

Cite this article as: Gülsoy et al.: HIDDEN: Hierarchical decomposition of regulatory networks. *BMC Bioinformatics* 2012 **13**:250.