Cancer driver gene discovery in transcriptional regulatory networks using influence maximization approach

Majid Rahimi\textsuperscript{a}, Babak Teimourpour\textsuperscript{a,*}, Sayed-Amir Marashi\textsuperscript{b}

\textsuperscript{a} Department of Information Technology Engineering, Faculty of Industrial and Systems, Tarbiat Modares University, Tehran, Iran
\textsuperscript{b} Department of Biotechnology, College of Science, University of Tehran, Tehran, Iran

ARTICLE INFO

Keywords:
Cancer driver gene
Cancer genomics
Transcriptional regulatory network
Gene expression data
Influence maximization

ABSTRACT

Cancer driver genes (CDGs) are the genes whose mutations cause tumor growth. Several computational methods have been previously developed for finding CDGs. Most of these methods are sequence-based, that is, they rely on finding key mutations in genomic data to predict CDGs. In the present work, we propose iMaxDriver as a network-based tool for predicting driver genes by application of influence maximization algorithm on human transcriptional regulatory network (TRN). In the first step of this approach, the TRN is pruned and weighted by exploiting tumor-specific gene expression (GE) data. Then, influence maximization approach is used to find the influence of each gene. The top genes with the highest influence rate are selected as the potential driver genes.

We compared the performance of our CDG prediction method with fifteen other computational tools, based on a benchmark of three different cancer types. Our results show that iMaxDriver outperforms most of the state-of-the-art algorithms for CDG prediction. Furthermore, iMaxDriver is able to correctly predict many CDGs that are overlooked by all previously published tools. Due to this relative orthogonality, iMaxDriver can be considered as a complementary approach to the sequence-based CDG prediction methods.

1. Introduction

1.1. Importance of CDG discovery

The quest for key mutated genes which are related to the cancer has been conducted by several recent researches [1–3]. The main idea behind current cancer driver gene (CDG) discovery methods is the assumption that frequent mutations in certain genes causes the cancer. Not all mutations in a cancer genome are related to cancer. Consequently, various computational methods are used to distinguish cancer driver mutations from passenger mutations.


There are some limitation and shortcomings in the currently available methods. These methods have high rate of false positive CDGs resulting low precision and F-measure. Furthermore, all of these methods heavily rely on mutation data, which is noisy, and additionally, may not be always available with desired quality. Moreover, most of CDGs found by each of these methods are in common with the set of CDGs found by other methods. Due to these limitations, we propose a new network-based driver gene prediction method. Our approach relies on the structure of gene regulatory network with no requirement of mutation data. This approach exploits an independent
source of information, and we show that it is able to find many new CDGs, and therefore, can be considered as an orthogonal CDG prediction method. Finally, we show that the performance of our method is better than many of the other existing approaches.

Transcriptional regulatory network (TRN) is the fundamental network for controlling cellular processes. Gene regulation controls the activity of genes at the transcription level. Transcription factors (TF) are key components in the cell that orchestrate the regulations. In other words, a TRN shows how any of the TFs regulates the expression of other TFs and genes. Most diseases, including cancer, are related to some dysfunction of TFs, which indicates the importance of analyzing TFs in biomedical research [18].

In this study, we propose iMaxDriver as a novel network-based method for finding CDGs by applying influence maximization approach on a cancer-specific tailored TRN. We show that iMaxDriver is able to improve the accuracy of CDG prediction, while it correctly finds many CDGs which are not detectable by the current sequence-based approaches.

1.2. Theoretical background

Influence maximization (IM) is a concept widely used in social network analysis to explain diffusion of information in a network [19]. Indeed, the IM problem is finding a minimal subset of nodes (i.e., the seed set) which have the greatest influence on other nodes. This problem is known to be an NP-hard optimization problem [20].

The IM problem is typically modeled by one of the two main approaches, namely, “independent cascade” or “linear threshold”. In these models each of the nodes in the network is labeled as “active” or “inactive”. At the beginning of the algorithm, a set of nodes are assumed to be active, while others are considered as inactive. Each of the active nodes can activate its (inactive) consecutive node based on a certain criterion, which depends on the IM approach. Independent cascade model only exploits probability value of edges to make decision for node activation, while in linear threshold model, in addition, a threshold value for each node is required.

The linear threshold model considers a threshold \( \theta_v \) for each node \( v \) and a weight \( w_{b,v} \) for the incoming edge from node \( b \). The values of \( \theta_v \) and \( w_{b,v} \) should be normalized so that they are in \([0,1]\) interval. Moreover, the sum of weights of incoming edges of any node should not exceed 1:

\[
\sum_{\text{edge } x \to v} w_{x,v} \leq 1 \tag{1}
\]

In each iteration of the algorithm, the active nodes will activate an inactive node if the sum of weights of incoming edges is at least equal to \( \theta_v \):

\[
\sum_{\text{active } b} w_{b,v} \geq \theta_v \tag{2}
\]

In Fig. 1, the linear threshold method is applied to a small toy model, which can be considered as a fictitious TRN. In this model, the regulatory interactions are weighted and an activation threshold value is specified for each node.

We have used a combination of TRN with gene expression (GE) data as input of linear threshold model of IM for finding CDGs. Each of the TFs exists in TRN, has been considered as initial active node to find out coverage (number of genes that it activates) of all nodes. Since only TFs in the network have outcome edges, for simplicity we have considered all of nodes as initial active node. In this model an active node can affect inactive nodes, indeed, the activation notion implies affecting downstream genes and showing the flow of GE change due to transcriptional regulation in the network which occurs in the cascading manner.

2. Methods

In this section we firstly describe the iMaxDriver pipeline, which comprises two different steps, namely, network construction and IM algorithm for finding CDGs. Finally, we describe the assessment of this pipeline, together with fifteen other cancer driver-predicting computational tools, based on three of available dataset benchmarks. The overall procedure of iMaxDriver method is shown in Fig. 2.

2.1. Network construction

2.1.1. Transcription regulatory network data

We considered RegNetwork database [21] (http://www.regnetworkweb.org/) as an example of a weighted TRNs to be used as data source of iMaxDriver for weighted networks (iMaxDriverW). In RegNetwork, the list of gene regulatory interactions from different methods and multiple databases is collected. More specifically, we retrieved the human TRN data of RegNetwork obtained from all databases and all methods. It should be noted that RegNetwork, in addition to the TRNs, also contains microRNA regulatory interactions, which are omitted in the present study. The retrieved dataset contains 150,202 TF-gene and TF-TF regulatory interactions. In addition, we exploited the confidence values for assigning weights to interactions which is available in website of RegNetwork as ‘CONFIDENCE’ column in search page. The edge weighting procedure was done based on the provided confidence of relationships, namely, ‘low’, ‘medium’ and ‘high’. The ‘high’ confidence is used for experimentally validated regulations, ‘medium’ for predicted regulations in more than one method and the rest as ‘low’ confidence. We assumed 0.2, 0.5 and 0.8 respectively for low, medium and high confidence values as edge weights. According to structure of constructed network in influence maximization model, changing these values only could affect results slightly.

As an example of non-weighted TRNs, we considered the TRRUST database [22] (http://www.grnpedia.org/trrust), to be used as data source of iMaxDriver for non-weighted networks (iMaxDriverNP). The TRRUST has been constructed using text mining methods, and then, by manual curation. We retrieved a human TRN from TRRUST v2 which includes with 9,396 TF-gene and TF-TF regulatory interactions.

2.1.2. Gene expression data

We obtained GE data for three specific cancer types from GEO database (Table 1). The GE values were taken from the curated GEO dataset (GDS). In cases where only the.CEL files were available, we extracted the expression values using RMA method implemented in Affy package in R [23]. In each of the selected GEO datasets, GE value is reported for both the tumor and for its adjacent normal tissue. In the preprocessing step, first we removed rows with missing gene ID and then combined gene expression values which their gene ID was synonym by averaging gene expression value of respective columns, afterwards we normalized the GE value \( e \) such that \( e \in [0,1] \) by dividing each value by maximum the value of GE in the relevant tissue type (tumor or normal).

2.1.3. Weighting the network

In the next step, the collected data were preprocessed to be prepared for IM algorithm. The linear threshold approach in IM problem requires a threshold \( \theta_v \) value bound to each gene (node) \( v \) which stored in list \( T \). For each gene \( v \) we assigned a threshold value as following:

\[
\theta_v = |e_{v,n} - e_{v,c}| \tag{3}
\]

where \( e_{v,n} \) is the GE value of \( v \) in the normal tissue and \( e_{v,c} \) is its GE value in the corresponding adjacent tumor tissue. Since all of the expression values have been normalized before, \( \theta_v \in [0,1] \). In the next step, the edge weights should also be normalized such that the constraint in equation (1) is satisfied. In the iMaxDriverW the edge confidence values are provided, and therefore, we exploited them to
Fig. 1. At the beginning of the algorithm, node 1 is considered as an active node (gray) and other nodes are assumed to be inactive (white) (a). In the next step, the potential influence of the only active node on other nodes should be assessed. In (b) and (c), node 1 tries to activate node 3 and node 2 respectively which successfully activates node 2 but cannot activate node 3 because of insufficient weight of the edge connecting node 1 to node 3. There is no inactive node remained in this iteration ($t = 1$) with incoming edge from an active node. Note that newly activated nodes cannot activate others in this iteration. Now, since no other node can be activated furthermore, this iteration will be ended. In the next iteration ($t = 2$) (d), the node 3 cannot be activated, because it only has one incoming edge with insufficient edge weight as assessed in the (b). In (e) node 2 activates node 4 whereas cannot activate node 5 due to insufficient edge weight rather than $\emptyset$ value of node 5 (f), the node 3 cannot be activated as mentioned above (g). In the next iteration ($t = 3$), the node 4 is active and when the incoming edges of node 5 assessed (h), the sum of weights of incoming edges from node 2 and node 4, exceeds $\emptyset$ value of node 5 and it will be activated in this iteration. In the next iteration ($t = 4$) (i), the node 3 cannot be activated. Finally, in (j), no node can be activated furthermore and the algorithm will be terminated.
When the edge weights are unavailable i.e. original gene regulatory network is presented as a non-weighted graph, one may assign random numbers to put weights on the edges, and then, normalize these weights such that equation (1) is satisfied. Equation (4) further is applicable for iMaxDriver\textsuperscript{W}, but since the random weight values are independent, the normalization for iMaxDriver\textsuperscript{N} can be done simpler by division of random edge weight \( t_{v,v} \) by the sum of weights of the incoming edges to node \( v \):

\[
w_{v,v} = \frac{t_{v,v}}{\sum_{edge \rightarrow v} w_{v,v}}
\]

(5)

Indeed, practically both equations generate normalized random values for input of the algorithm, therefore the results will not be changed by using equation (4) or (5). The complete process of the network weighting is shown in Fig. 3.

2.2. The iMaxDriver algorithm

As mentioned above, the IM approach tries to find a minimal subset of initially active nodes with maximum influence. In the classic algorithm, one should test all possible subsets of nodes (i.e., 1-member subsets, 2-member subsets, etc.) as the seed subset. In the present work, we used a modified version of IM which considers only a single active node as the seed. We developed a multi-threaded implementation of the algorithm which could be run on multi-core processors. The implementation of iMaxDriver\textsuperscript{W} and iMaxDriver\textsuperscript{N} algorithms, is available from https://github.com/majid-rh/iMaxDriver.

The network weighting and preprocessing steps are different based on the type of input network. In the case of iMaxDriver\textsuperscript{W} that edge weights are non-random and are based on the provided interaction confidence values, the weights are defined, and therefore, the algorithm is run once (Algorithm 1). Conversely, in the case of iMaxDriver\textsuperscript{N} that network is non-weighted, since the random values can affect final results, we repeated the experiment 100 times and computed the average of all coverage values, to alleviate the effect of random weight assignments (Algorithm 2). The FindCoverage function is the main procedure used by Algorithm 1 and Algorithm 2 for computing coverage (Algorithm 3), representing the modified version of the linear threshold approach.

The input of iMaxDriver algorithm is the graph \( G \) representing TRN and a list \( T \) containing the threshold values. The output of the algorithm is a list of genes sorted by their coverage count in descending manner. In the Algorithm 1, which is provided for weighted graphs, in the first line, we create an empty list of tuples. Each tuple contains the gene name and its coverage count. In line 3, we calculate MaxIncome as the maximum of the sum of the weights of incoming edges (equation (4)). In lines 4–6, the edge weights are normalized (equation (4)). Afterwards, in lines 7–10, FindCoverage function is called for each node \( v \) in the \( G \) and the returned value of coverage count is stored in Result. Finally, in line 11 the Result list is sorted by CoverageCount in descending order.

In Algorithm 2, which is provided for non-weighted graphs, in line 1, we first create an empty list of tuples, each tuple contains gene name and its coverage (number of genes that activated). In line 3, a loop defined to run the coverage discovery for 100 times and for each of the iterations, in lines 4–6 new random values assigned to the edges. Note that in each iteration the edge weights regenerated randomly to overcome random value effects on final results. Next, for normalization of randomly generated edge weights, in line 7, a for loop defined to iterate over all of the nodes and normalize the incoming edges to them. In lines 8–9, the sum of weights of incoming edges of node \( v \) is calculated. Now in lines 10–12, for each incoming edge to node \( v \), the edge weights divided by \( \text{Sum} \) value according to equation (5). Next, in lines 14–16, the core function of coverage finding (FindCoverage) called for each
node v and the results are integrated by averaging coverage values.

The algorithm of $\text{FindCoverage}$ function is provided in Algorithm 3. In this algorithm, the weighted graph $G$ and list $T$ are the inputs and coverage count ($\text{ActiveCount}$) is output. At the beginning, in line 1, initially the gene v considered as the only active gene and is assigned to $\text{ActiveNodes}$. Subsequently, in line 2, the number of currently activated genes by gene v assigned to be zero. The algorithm loops through lines 3–14, until an iteration with no new gene activation. This defined in the algorithm using $\text{while}$ statement in line 3. In line 4, an empty list of the genes that are going to be activated in this iteration ($\text{NewActiveNodes}$), is created. Subsequently, in lines 7–10, for each inactive gene, if difference of sum of incoming edge weights from currently active genes and threshold of the gene, be larger than $\alpha$, the gene will be activated. Finally, in line 13, the newly activated nodes ($\text{NewActiveNodes}$) is added to current active nodes ($\text{ActiveNodes}$).

In line 9, we have used $\alpha$ as a parameter in the algorithm for alleviating the differences exist in threshold values and edge weights causing the algorithm to not work properly. We have set $\alpha = 0.15$ for obtaining the best result. The larger value of $\alpha$ causing less genes to be activated at final step of the algorithm and vice versa. If $\alpha$ be too small, all of the genes will be activated and if $\alpha$ be too large, none of the genes will be activated and results in trivial outcomes.

2.3. Evaluation method

We evaluated iMaxDriver by comparing its results with fifteen popular computational CDG prediction methods. The list of genes introduced as CDGs by the fifteen computational methods are obtained from DriverDB v2 [24] for the evaluation with same input for all methods. The details of computational methods used for evaluation is available in Table 2. These lists are reported in DriverDB webpage (http://driverdb.tms.cmu.edu.tw/driverdbv2/) based on three different cancer datasets, namely breast invasive carcinoma (BRCA) [25], lung squamous cell carcinoma (LUSC) [26] and colon adenocarcinoma (COAD) [27], which are parts of the Cancer Genome Atlas (TCGA) [28]. We also retrieved the lists of genes identified as CDG by each of the fifteen methods for benchmarking.

In the next step, we used iMaxDriver for predicting CDGs, and then, for iMaxDriver and the other fifteen methods, we assessed the accuracy of the predicted CDGs by comparing each list with the Cancer Gene Census (CGC) [29] gene list, as the gold standard (available from https://cancer.sanger.ac.uk/census). Next, the IM approach was independently applied on each of the three GE datasets (see Subsection 3.1) to compute the influence of each TF in the network. The results of iMaxDriver are provided for each cancer type as a list of potential CDGs sorted by their influence (coverage count) in descending order. Subsequently, by discretizing the results based on a threshold value we classified the genes either as CDG or non-CDG. For fine-tuning the threshold value used for binary classification, we exploited pROC [30] package in R.

The $F$-measure is a prevalent measure for evaluating the classifiers and is a good measure considering both of precision and recall.
measures. F-measure is mean of precision and recall in harmonic manner and defined as the following:

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

(6)

where the precision is defined as the following:

$$\text{Precision} = \frac{TP}{TP + FP}$$

(7)

and recall is defined as the following:

$$\text{Recall} = \frac{TP}{TP + FN}$$

(8)

In the above equations, TP stands for the number of true positive, FP stands for the number of false positive and FN stands for number of false negative items. We will use F-measure as classification quality measure for evaluation of the iMaxDriver.

### 3. Results

We weighted the modified TRN using the GE data of three cancer types, including breast invasive carcinoma (BRCA), lung squamous cell carcinoma (LUSC) and colon adenocarcinoma (COAD) independently. Then, the list of predicted CDGs was generated using iMaxDriver (Supplementary Datasets S1 and S2). The iMaxDriver\(W\) could find 103, 143 and 113 CDGs for BRCA, LUSC and COAD, respectively. Subsequently, the iMaxDriver\(W\) can find 88 driver genes in each of BRCA and LUSC tissues and 90 driver genes in COAD. We evaluated our method and the other fifteen methods using cancer gene census (CGC) and functionally validated driver genes provided in Ref. [31] by Kumar et al. and gathered the results for BRCA, LUSC and COAD in Table 3. In each of the tissue types and validation datasets, top three methods with the best prediction results is shown bold. In all of the tissue types, the iMaxDriver\(W\) is one of the top three methods with the best results. Moreover, the iMaxDriver\(N\) in BRCA and LUSC tissue types is one of the top three methods.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Feature(s)</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveDriver [3]</td>
<td>Protein phosphorylation signaling sites</td>
<td>Single nucleotide variants related to phosphorylation</td>
</tr>
<tr>
<td>CoMDP [12]</td>
<td>Co-occurred mutated driver pathways</td>
<td>Mutation data</td>
</tr>
<tr>
<td>Dendrix [9]</td>
<td>Coverage and exclusivity of mutations</td>
<td>Mutation data</td>
</tr>
<tr>
<td>e-Driver [5]</td>
<td>Protein functional region mutation rates</td>
<td>Mutation profiles</td>
</tr>
<tr>
<td>DriverNet [13]</td>
<td>Effect of mutations on mRNA network</td>
<td>Mutation data and transcriptional network</td>
</tr>
<tr>
<td>iPAC [17]</td>
<td>Copy number alterations</td>
<td>GE and mutation data</td>
</tr>
<tr>
<td>MDPFinder [16]</td>
<td>Mutual exclusivity of gene modules</td>
<td>GE and mutation data</td>
</tr>
<tr>
<td>MeMo [18]</td>
<td>Genes correlation</td>
<td>Mutation data and network</td>
</tr>
<tr>
<td>MSEA [11]</td>
<td>Combination of data associated to disease</td>
<td>Pathways and networks</td>
</tr>
<tr>
<td>MutsigCV [15]</td>
<td>Frequency of mutations and spectrum</td>
<td>GE and exome sequence</td>
</tr>
<tr>
<td>NetBox [14]</td>
<td>Functional modules in cellular networks</td>
<td>Mutation data and network</td>
</tr>
<tr>
<td>OncodriveCLUST [8]</td>
<td>Somatic mutation clustering</td>
<td>Mutation data</td>
</tr>
</tbody>
</table>

### Table 2

The details of the computational methods used for comparison with iMaxDriver.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Feature(s)</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveDriver [3]</td>
<td>Protein phosphorylation signaling sites</td>
<td>Single nucleotide variants related to phosphorylation</td>
</tr>
<tr>
<td>CoMDP [12]</td>
<td>Co-occurred mutated driver pathways</td>
<td>Mutation data</td>
</tr>
<tr>
<td>Dendrix [9]</td>
<td>Coverage and exclusivity of mutations</td>
<td>Mutation data</td>
</tr>
<tr>
<td>e-Driver [5]</td>
<td>Protein functional region mutation rates</td>
<td>Mutation profiles</td>
</tr>
<tr>
<td>DriverNet [13]</td>
<td>Effect of mutations on mRNA network</td>
<td>Mutation data and transcriptional network</td>
</tr>
<tr>
<td>iPAC [17]</td>
<td>Copy number alterations</td>
<td>GE and mutation data</td>
</tr>
<tr>
<td>MDPFinder [16]</td>
<td>Mutual exclusivity of gene modules</td>
<td>GE and mutation data</td>
</tr>
<tr>
<td>MeMo [18]</td>
<td>Genes correlation</td>
<td>Mutation data and network</td>
</tr>
<tr>
<td>MSEA [11]</td>
<td>Combination of data associated to disease</td>
<td>Pathways and networks</td>
</tr>
<tr>
<td>MutsigCV [15]</td>
<td>Frequency of mutations and spectrum</td>
<td>GE and exome sequence</td>
</tr>
<tr>
<td>NetBox [14]</td>
<td>Functional modules in cellular networks</td>
<td>Mutation data and network</td>
</tr>
<tr>
<td>OncodriveCLUST [8]</td>
<td>Somatic mutation clustering</td>
<td>Mutation data</td>
</tr>
</tbody>
</table>

### Table 3

The evaluation of the iMaxDriver and the other methods using CGC and Kumar datasets.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>BRCA</th>
<th>LUSC</th>
<th>COAD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of predicted drivers</td>
<td>Fraction of predicted drivers</td>
<td>Number of predicted drivers</td>
</tr>
<tr>
<td>Validation Dataset</td>
<td>Kumar</td>
<td>CGC</td>
<td>Kumar</td>
</tr>
<tr>
<td>iMaxDriver (W)</td>
<td>27</td>
<td>103</td>
<td>0.273</td>
</tr>
<tr>
<td>iMaxDriver (N)</td>
<td>29</td>
<td>88</td>
<td>0.293</td>
</tr>
<tr>
<td>ActiveDriver</td>
<td>8</td>
<td>27</td>
<td>0.081</td>
</tr>
<tr>
<td>CoMDP</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>DawnRank</td>
<td>11</td>
<td>14</td>
<td>0.111</td>
</tr>
<tr>
<td>Dendrix</td>
<td>11</td>
<td>14</td>
<td>0.111</td>
</tr>
<tr>
<td>DriverNet</td>
<td>22</td>
<td>37</td>
<td>0.222</td>
</tr>
<tr>
<td>e-Driver</td>
<td>16</td>
<td>32</td>
<td>0.162</td>
</tr>
<tr>
<td>iPAC</td>
<td>63</td>
<td>250</td>
<td>0.636</td>
</tr>
<tr>
<td>MDPFinder</td>
<td>5</td>
<td>6</td>
<td>0.051</td>
</tr>
<tr>
<td>MEMO</td>
<td>10</td>
<td>14</td>
<td>0.101</td>
</tr>
<tr>
<td>MSEA</td>
<td>27</td>
<td>76</td>
<td>0.273</td>
</tr>
<tr>
<td>MutsigCV</td>
<td>11</td>
<td>28</td>
<td>0.111</td>
</tr>
<tr>
<td>NetBox</td>
<td>2</td>
<td>2</td>
<td>0.020</td>
</tr>
<tr>
<td>OncodriveCLUST</td>
<td>2</td>
<td>4</td>
<td>0.020</td>
</tr>
<tr>
<td>OncodriveFM</td>
<td>6</td>
<td>8</td>
<td>0.061</td>
</tr>
<tr>
<td>Simon</td>
<td>8</td>
<td>16</td>
<td>0.081</td>
</tr>
</tbody>
</table>
Most CDG finding tools predict only a limited number of genes. Although some of these tools predict many CDGs in their output, they are not generally of an acceptable precision. As an example, for BRCA tissue, iPAC and MSEA predict 4821 and 855 genes as CDGs, while their precision is as small as 5.1% and 8.8%, respectively. In contrast, iMaxDriverW predicts 408 genes as CDGs, with a precision value of 33.3%. The binary matrix representation of the genes predicted as CDG by the methods and dendrogram of clustering result are shown for BRCA, LUSC and COAD in Fig. 4.

Furthermore, by comparing the list of classified genes, we can see that more than 38% of the genes classified as driver by iMaxDriverW are reported by none of the other fifteen computational tools. The iMaxDriverW reported 43, 97 and 43 new driver genes in BRCA, LUSC and COAD respectively. Additionally, more than 41% of the genes classified as driver by iMaxDriverW are reported as driver by none of the other fifteen computational tools. Furthermore, the iMaxDriverW reported 41, 68 and 37 new driver genes in BRCA, LUSC and COAD respectively. This means many new driver genes introduced by iMaxDriver. The Venn diagram of the result of gene overlaps among iMaxDriver and union of other fifteen computational tools is shown in Fig. 6. Further, in case of precision and recall iMaxDriver provided acceptable results that is available at Supplementary Figs. S1 and S2.

4. Discussion

The current computational tools available for CDG prediction rely heavily on mutation rates. First group of methods including ActiveDriver [3], CoMDP [12], Dendrix [9], e-Driver [5],
OncodriveCLUST [8], OncodriverFM [7] and Simon [6] exploit mutation profiles as main feature for CDG prediction. The second group of methods including DawnRank [4], DriverNet [13], iPAC [17], MDPFinder [16], MeMo [10] and NetBox [14] rely on mutation data integrated with other omics data such as GE, network and pathways data etc. MSEA [11] exploits disease-related data in addition to functional genomics and gene network data and finally MutsigCV [15] relies on GE and exome sequence for CDG prediction. In contrast, the iMaxDriver exploits GE data together with TRN data without using mutation data. This strategy is shown to result in better F-measure values in comparison with the other state-of-the-art methods. We proposed iMaxDriverW and iMaxDriverN for weighted and non-weighted networks, respectively. The arbitrary values (0.2, 0.5 and 0.8) is used for mapping confidence of relationships for the iMaxDriverW. Changing these values only affects the final results slightly (Supplementary Fig. S3). The random values are used for edge weighting when the network was non-weighted due to lack of information about effect of each transcription factor on the other genes. Furthermore, our results suggest that average of coverage values is a good proxy for the influence of a CDG on the regulation of other genes in iMaxDriverN. The results indicate that iMaxDriverW surpass iMaxDriverN in number of correctly classified CDGs and F-measure since more information has used as input in iMaxDriverW and consequently the results of IM is improved.

The CGC project aims to catalogue genes which causally result in cancer. The results of iMaxDriver method are compared with CGC driver gene list and the precision, recall and F-measure of the iMaxDriver is compared with other fifteen computational tools available for CDG prediction. The results indicate that often the recall (and sometimes the precision) of iMaxDriver are one of the best among the tested methods. However, in case of all cancer tissues, the F-measure of iMaxDriver is superior compared to all of the other methods (Fig. 4). Furthermore, iMaxDriver shows a relative “orthogonality” to other methods, as a considerable portion of its predicted CDGs are not detected any of the other methods. This indicates that iMaxDriver can find complementary driver genes for state-of-the-art computational CDG prediction tools.

**Conflicts of interest**

The authors declare that they have no competing interests.

**Data availability**

Data is presented within the manuscript and the Supplemental Materials.

**Software availability**

The software is available publicly at https://github.com/majid-rh/iMaxDriver.

**Conflicts of interest**

None declared.

### Appendix A. Supplementary data

Supplementary data to this article can be found online at [https://doi.org/10.1016/j.compbiomed.2019.103362](https://doi.org/10.1016/j.compbiomed.2019.103362).

---

**Fig. 6.** The Venn diagram for correctly-predicted CDGs using iMaxDriver and the union of correctly predicted CDGs using all other methods.

![Venn Diagrams](image-url)
Algorithm 1. The iMaxDriver\textsuperscript{W} algorithm.

**Input:** \( G(V, E, W) \): A directed and weighted graph of genes and a list \( T \) of threshold value for each gene ID that constructed according to descriptions provided in Subsection 3.1

**Output:** \( \text{Result} \): A list of genes sorted by their coverage

1. \( \text{Result} \leftarrow \) an empty set of \(<\text{GeneID}, \text{CoverageCount}\rangle\) with initially \( \text{CoverageCount} = 0 \)
2. \( \text{AllNodes} \leftarrow \) all nodes of \( G \)
3. Calculate \( \text{MaxIncome} \) as maximum of sum of weights of incoming edges to the nodes
4. for each edge \( e \) in \( E \):
5. \hspace{1em} \( W(e) \leftarrow W(e) / \text{MaxIncome} \)
6. end for
7. for each node \( v \) in \( V \):
8. \hspace{1em} \( \text{CoverageCount} = \text{FindCoverage}(v, G, T) \)
\hspace{1em} // finding coverage of \( v \) in graph \( G \)
9. \hspace{1em} Add \(<\text{GeneID}(v),\text{CoverageCount}\rangle\) to \( \text{Result} \)
10. end for
11. Sort \( \text{Result} \) by \( \text{CoverageCount} \) descending
Algorithm 2. The iMaxDriver algorithm.

**Input:** $G(V, E)$: A non-weighted directed graph of genes and a list $T$ of threshold value for each gene ID

**Output:** Result: A list of genes sorted by their coverage

1. $Result \leftarrow$ an empty set of $<\text{GeneID}, \text{CoverageCount}>$
   
   with initially $\text{CoverageCount} = 0$

2. $\text{AllNodes} \leftarrow$ all nodes of $G$

3. for $n$ from 1 to 100:

4.   for each edge $e$ in $E$:

5.     $W(e) \leftarrow$ a random value

6.   end for

7.   for each node $v$ in $V$:

8.     $\text{IncomeEdges} \leftarrow$ income edges of $v$

9.     $\text{Sum} \leftarrow$ sum of edge weights of $\text{IncomeEdges}$

10.    for each edge $i$ in $\text{IncomeEdges}$:

11.       $W(i) \leftarrow W(i) / \text{Sum}$

12.   end for

13. end for

14. for each node $v$ in $V$:

15.     $\text{CoverageCount} = \text{FindCoverage}(v, G, T)$

   // finding coverage of $v$ in graph $G$

16.     $Result(v) \leftarrow Result(v) + (\text{CoverageCount} - Result(v)) / n$

17. end if

18. end for

19. end for

20. Sort $Result$ by $\text{CoverageCount}$ descending
Algorithm 3. iMaxDriver core algorithm (FindCoverage).

Input: $G(V, E, W)$ as a directed and weighted graph of genes and list $T$ of threshold values for each gene ID and $v$ as a selected initial node

Output: ActiveCount: Number of genes activated by the selected initial node $v$

1. ActiveNodes ← $v$
2. ActiveCount ← 0
3. While (Count (ActiveNodes) > ActiveCount)
4. NewActiveNodes ← empty set
5. ActiveCount ← Count (ActiveNodes)
6. InactiveNodes ← (AllNodes – ActiveNodes)
7. for each node $n$ in InactiveNodes:
   8. $IncomeWeights \leftarrow \text{sum of weights of all edges from ActiveNodes to } n$
   9. if ($IncomeWeights - T(n) \geq a$)
      10. NewActiveNodes ← NewActiveNodes U $n$
8. end if
9. end for
10. ActiveNodes ← ActiveNodes U NewActiveNodes
11. end while

References