Simon J. Larsen[1] / Jan Baumbach[1]

CytoMCS: A Multiple Maximum Common Subgraph Detection Tool for Cytoscape

[1] Computational Biology group, Department of Mathematics and Computer Science, University of Southern Denmark, DK-5230 Odense M, Denmark, E-mail: sjlarsen@imada.sdu.dk

Abstract:
Comparative analysis of biological networks is a major problem in computational integrative systems biology. By computing the maximum common edge subgraph between a set of networks, one is able to detect conserved substructures between them and quantify their topological similarity. To aid such analyses we have developed CytoMCS, a Cytoscape app for computing inexact solutions to the maximum common edge subgraph problem for two or more graphs. Our algorithm uses an iterative local search heuristic for computing conserved subgraphs, optimizing a squared edge conservation score that is able to detect not only fully conserved edges but also partially conserved edges. It can be applied to any set of directed or undirected, simple graphs loaded as networks into Cytoscape, e.g. protein-protein interaction networks or gene regulatory networks. CytoMCS is available as a Cytoscape app at http://apps.cytoscape.org/apps/cytomcs.

Keywords: networks, network alignment, global network alignment, local search

DOI: 10.1515/jib-2017-0014

Received: March 17, 2017; **Revised:** June 9, 2017; **Accepted:** June 12, 2017

1 Introduction

The analysis and comparison of biological networks (modelled as graphs) is an important problem in computational systems biology. Graphs are used to model a wide range of biological data including, but not limited to, protein-protein interactions, gene regulatory networks, protein structures and drug-target networks. Comparing different graphs (also called graph alignment) can be used to quantify how similar they are, or to determine whether they all contain some common substructure(s) [1], [2]. One way of determining this is to compute the maximum common edge subgraph (MCES) between a given set of input graphs, i.e. the largest graph (wrt. the number of edges) that is a subgraph of each of the compared graphs.

The maximum common edge subgraph problem is closely related to the problem of global network alignment [3]. Global network alignment methods aim to determine a mapping between the vertices of two or more graphs optimizing some biological or topological quality measure, or a combination of both. The development of network alignment methods has mainly been motivated by their application to biological knowledge transfer [4], predicting new interactions [5] and protein structure comparison [6], [7], [8]. The global network alignment problem is equivalent to the MCES problem when the optimized quality measure is the number of fully conserved edges between the aligned graphs. See [4] and [9] for an extensive review of existing global network aligners and commonly used quality measures.

To our knowledge, no maximum common subgraph tool for Cytoscape [10], able to compare large networks, currently exists. One existing app, chemViz2 [11], is able to determine the maximum common substructure for a set of chemical counts compounds, using an exact algorithm, and is thus only applicable to chemical compounds and of limited size (up to ca. 200 nodes). Another Cytoscape app, CytoGEDEVO [12], [13], performs pairwise network alignment of large networks by minimizing the graph edit distance between networks, and is thus not applicable to the MCES problem. To fill this gap we present CytoMCS, a heuristic maximum common edge subgraph detection tool for the Cytoscape network analysis and visualization platform. CytoMCS provides a simple interface allowing researchers to search for a common subgraph conserved between two or more graphs, using a fast iterative local search heuristic. Our algorithm optimizes a squared edge conservation score introduced in [3], but extended to directed networks as well, and is able to not only produce good solutions to the MCES problem, but also report partially conserved edges.

Simon J. Larsen is the corresponding author.

2 Methods

2.1 Problem Formulation

A *simple graph* is an unweighted, undirected graph with no parallel edges and no edge loops. A *simple directed graph* is an unweighted, directed graph with no parallel edges and no edge loops. For the remainder of this paper the word *graph* will refer to a simple graph or a simple directed graph unless otherwise stated. Two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are *isomorphic* if there exists a bijective function $f: V_1 \to V_2$ such that there is an edge (u, v) in E_1 if and only if there is an edge $(f(u), f(v))$ in E_2.

Given a set of graphs $\mathbf{X} = \{G_1(V_1, E_1), G_2(V_1, E_2), \cdots, G_n(V_n, E_n)\}$, the multiple maximum common edge subgraph problem (multi-MCES) is the problem of finding a graph with maximum number of edges that is isomorphic to a subgraph (not necessarily induced) of each graph in **X**. The pairwise MCES problem is a generalization of the subgraph isomorphism problem making it NP-hard [14]. Being a generalization of the pairwise MCES problem, multi-MCES is NP-hard as well.

Without loss of generality, assume that $|V_1| \leq |V_i|$, for $1 \leq i \leq n$. We then define an alignment A as a set of injective functions

$$A = \{f_i : V_1 \to V_i \mid 1 \leq i \leq n\} \tag{1}$$

where $f_1(v) = v$ for all $v \in V_1$. Given an alignment A, one can compute the number of edges conserved between all graphs as

$$f(A) = \sum_{u,v}^{n} \begin{cases} 1 & \text{if } (f_i(u), f_i(v)) \in E_i \;\; \forall i \in [1, n], \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

For directed graphs the sum runs over all ordered pairs (u, v) where $u, v \in V_1$ and $u \neq v$. For undirected graphs the sum runs over all unordered pairs $\{u, v\}$ where $u, v \in V_1$ and $u \neq v$, i.e. the pairs $\{u, v\}$ and $\{v, u\}$ are identical and only counted once. Given these definitions, the multi-MCES problem can then be defined as the problem of finding an alignment A that maximizes $f(A)$.

2.2 Algorithm

CytoMCS uses an iterative local search algorithm to search for an alignment that maximizes the number of edges conserved between all graphs. We extend an earlier version of the algorithm in [3], adding support for directed graphs and reducing the number of hyperparameters and the required virtual memory needed.

An initial alignment is generated by sorting the vertices in each graph on their vertex degree (out-degree for directed networks) and aligning them in this order from largest to smallest degree. In each iteration, the algorithm will perturb the current solution to provide a new starting point for the local search. The local search procedure is then applied to the perturbed solution until no more improving moves are found. The general algorithm is outlined in Algorithm 1:.

Algorithm 1: Overview of CytoMCS algorithm

```
$A^* \leftarrow$ InitialAlignment($\mathbf{X}$)
$A \leftarrow A^*$
repeat
    $A \leftarrow$ Perturbate($A$, $\mathbf{X}$)
    $A \leftarrow$ LocalSearch($A$, $\mathbf{X}$)
    if $f(A) > f(A^*)$ then
        $A^* \leftarrow A$
    end
until not improving
return $A^*$
```

2.3 Perturbation

At the beginning of each iteration, the algorithm perturbs the current solution using a naive perturbation to provide a new starting point for the local search procedure. For each graph being aligned, the perturbation procedure will repeatedly select two random vertices and swap their alignment. This step is repeated $r \cdot |V_i|$ times for each graph G_i, determined by the perturbation parameter r, where $0 < r \leq 1$. Increasing r will increase the algorithm's ability to escape local optima, but a too large value of r will perturb too much of the current solution effectively resulting in a repeated local search instead.

2.4 Local Search

Our algorithm uses a local search procedure in order to improve the current alignment. The local search improves the solution by swapping the alignment of pairs of vertices from the same graph. When aligning more than two graphs, swapping the alignment of two vertices will often not result in an increase in the number of number of conserved edges, even if the resulting alignment provides a better matching of similar vertices. To rectify this we instead use the following fitness function:

$$g(A) = \sum_{u,v} C(u,v,A)^2, \tag{3}$$

where the sum runs over all ordered and unordered vertex pairs for directed and undirected graphs, respectively (similar to Equation 2), and $C(u, v, A)$ is the number of conserved edges between a pair of vertices u, v computed with

$$C(u,v,A) = \sum_{i=1}^{n} \begin{cases} 1 & \text{if } (f_i(u), f_i(v)) \in E_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The sum of edge counts $\sum_{u,v} C(u,v,A)$ is simply the total number of edges among all the aligned graphs and is constant regardless of how vertices are mapped to each other in A. By optimizing the squared edge counts in Equation 3, edges are redistributed to favor fewer well-conserved edges instead of more less conserved ones.

One benefit of this fitness function is that the algorithm will also try to improve the conservation of edges that will not be part of the MCES, i.e. edges that are conserved between most but not all of the aligned graphs. By allowing some number of exceptions k when extracting the conserved subgraph, such that all edges conserved in $\geq n - k$ graphs are included, we can find not only the edges that are conserved between all graphs but also edges that are conserved between most but not all graphs, providing valuable insight into the common structure of the compared graphs. We refer to such edges as *exception edges*. See Figure 1 for an example.

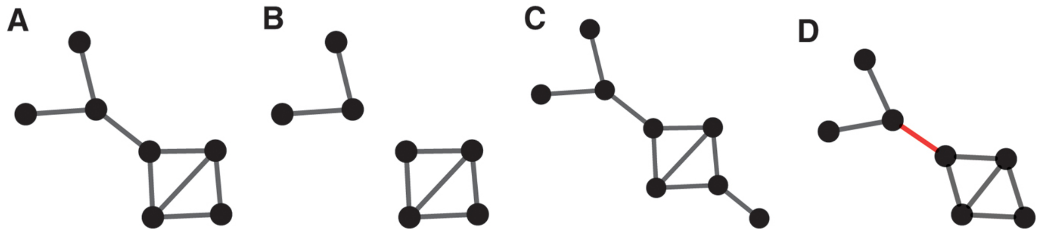


Figure 1: (A)–(C) Three graphs $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$. (D) The computed MCES of $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$ when allowing 1 exception per edge. Fully conserved edges are colored gray while exception edges are colored red.

3 Results

In order to demonstrate our tool we computed common subgraphs for a set of undirected and directed biological networks. We obtained multi-validated physical protein-protein interactions from BioGRID [15] (release 3.4.147) for mouse (`PPI_MM`), thale cress (`PPI_AT`), baker's yeast (`PPI_SC`) and human (`PPI_HS`). An undirected network was created for each species, where each node is a gene and an edge exists between two nodes if the data set contains an interaction between their genes. We also obtained experimentally validated transcription factor-target gene interactions for human (`GRN_HS`) from HTRIdb [16] (obtained April 28 2017) and for *E. coli* K12 (GRN_EC) from RegulonDB [17] (release 9.3). A directed network was created for each species where each node is a gene and an edge exists between two genes g_1, g_2 if g_1 is coding for a transcription factor that regulates g_2. For all networks, parallel edges and edge loops were removed. Furthermore only the largest connected component was used. The network data set is summarized in Table 1.

Table 1: Overview of network files used in benchmarks.

Network	Type	Source	Directed	Nodes	Edges
PPI_MM	Protein-protein interaction	BioGRID	Undirected	1027	1454
PPI_AT	Protein-protein interaction	BioGRID	Undirected	1565	2625
PPI_SC	Protein-protein interaction	BioGRID	Undirected	3536	13591
PPI_HS	Protein-protein interaction	BioGRID	Undirected	7681	31042
GRN_EC	TF-TG regulation	RegulonDB	Directed	1751	4437
GRN_HS	TF-TG regulation	HTRIdb	Directed	18308	51846

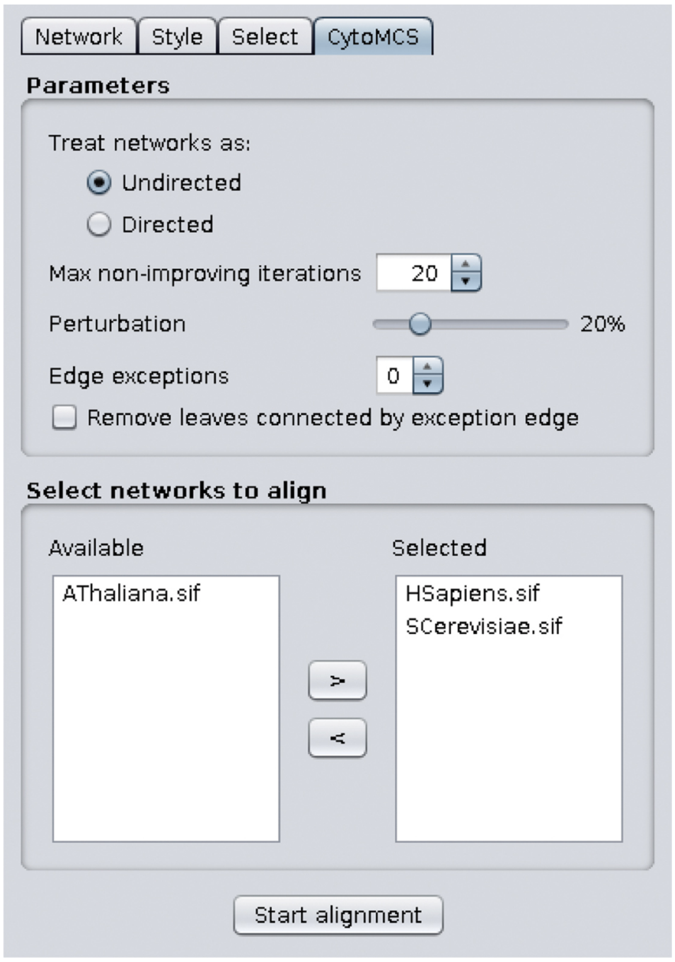
Because our algorithm is randomized (due to the random perturbation step), each benchmark was run five times and the mean solution quality (number of conserved edges) and mean running time (wall time) was reported. The *r*-parameter was kept at the default value of 20% for all benchmarks, and the algorithm was set to terminate after 20 consecutive non-improving iterations. All benchmarks were run on an Intel Xeon E5-2680 v3 2.50 GHz CPU using 8 cores. The results of the benchmarks are shown in Table 2. The reported memory requirement is an upper bound on the heap memory required for each instance, and includes only the memory used by the algorithm, not the memory used by the Cytoscape software. At the time of writing, the authors are not aware of any existing heuristic tools for MCES finding in large networks, besides the algorithm this method is based on [3]. For that reason these results cannot be compared to existing methods, and are simply provided here for reference.

Table 2: Results of benchmark runs. Each test was repeated five times and mean quality and running time is reported. The solution quality is measured as the number of fully conserved edges.

Networks	Mean quality	Mean time	Req. memory
PPI_MM, PPI_AT	880.0	1 m 6 s	15 MB
PPI_HS, PPI_SC	7004.4	33 m 44 s	400 MB
PPI_HS, PPI_SC, PPI_AT	1737.2	48 m 10 s	400 MB
PPI_HS, PPI_SC, PPI_AT, PPI_MM	669.0	1 h 2 m	450 MB
GRN_HS, GRN_EC	3000.8	2 h 6 m	3 GB

4 Cytoscape Integration

CytoMCS is provided as a Cytoscape 3.0 app. Both the app and the underlying algorithm are implemented in Java SE 8. The algorithm makes heavy use of the Java 8 Streams API for implementing simple and efficient parallelization, automatically utilizing multiple cores if available. The app's control panel exposes only one hyperparameter for the algorithm, the *r* parameter (see Section Section 2.3), controlling the amount of perturbation done between iterations. The control panel is shown in Figure 2.



Parameters

Treat networks as:
[✓ Undirected]
Directed

Max non-improving iterations 20

Perturbation 20%

Edge exceptions 0

☐ Remove leaves connected by exception edge

Select networks to align

Available	Selected
AThaliana.sif	HSapiens.sif
	SCerevisiae.sif

Figure 2: The CytoMCS app’s control panel.

The input to the algorithm is provided simply by selecting which of the currently loaded Cytoscape networks to align, with no additional data required. The common subgraph computed by the algorithm is returned as a Cytoscape network. Each edge in the solution is annotated with its number of exceptions (number of graphs the edge is not conserved in) and can be used for visualization using Cytoscape’s visual styles in order to show which edges are more well-conserved than others (Figure 3).

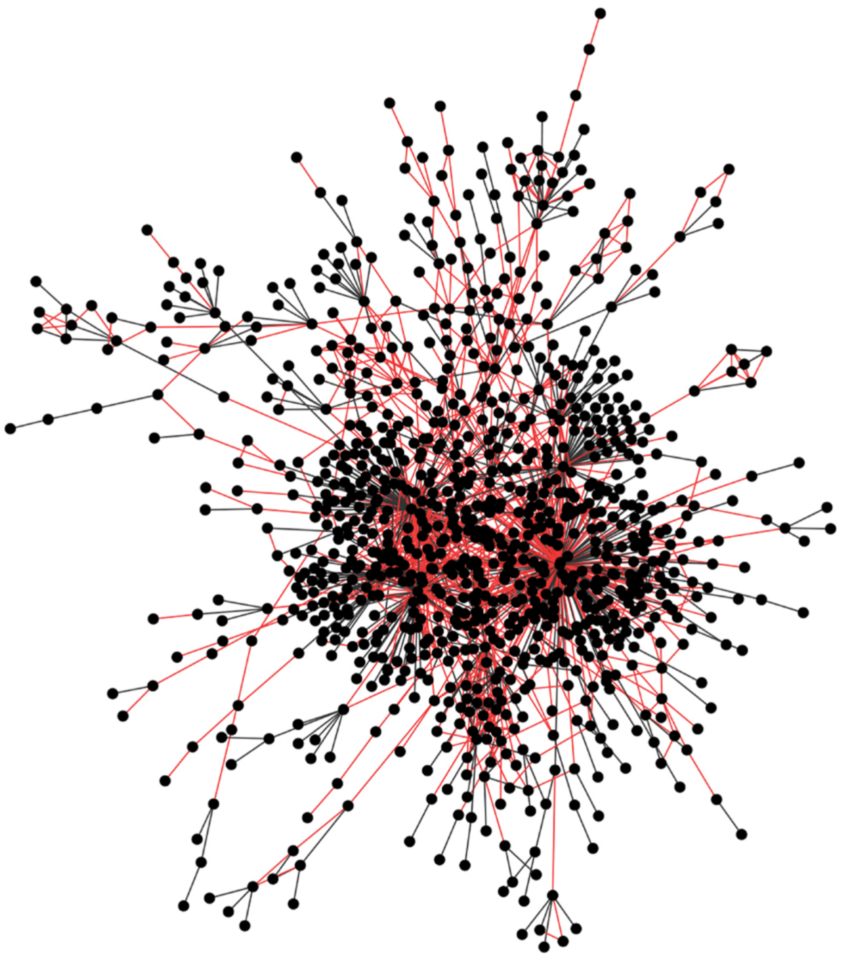
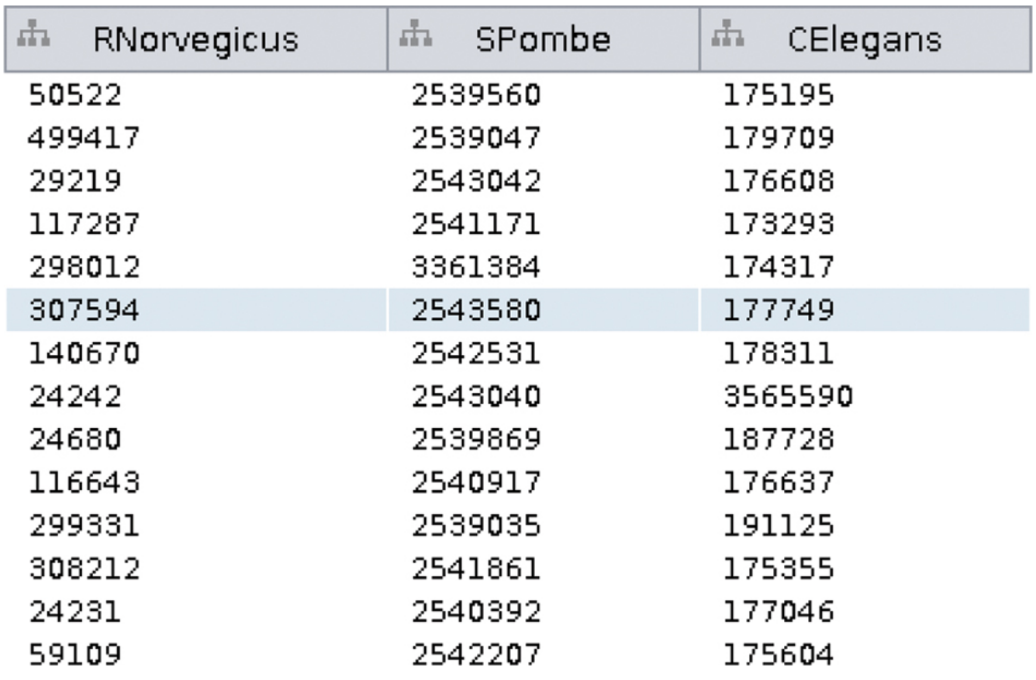


Figure 3: Example of common subgraph between three PPI networks computed by CytoMCS. Fully conserved edges are colored black while edges with one exception are colored red. Leaf nodes connected by exception edges have been filtered.

The node table of the solution is also augmented with a column for each of the compared graphs, indicating which vertices are mapped to each other in the solution (Figure 4). This table can then easily be exported from Cytoscape to a CSV file for use in external analysis. Biological networks, as well as many other types of real-world networks, often have a vertex degree distribution that resembles a power law or Pareto distribution. This results in a large fraction of vertices that are leaves (having degree one). When including exception edges in the results, the algorithm will often produce a large number of exception edges connected to such leaves. These edges are often not interesting for analysis, as they are likely to appear by chance. By enabling the *Remove leaves connected by exception edge* option, all leaf vertices connected to the solution only by an exception edge will be removed.



RNorvegicus	SPombe	CElegans
50522	2539560	175195
499417	2539047	179709
29219	2543042	176608
117287	2541171	173293
298012	3361384	174317
307594	2543580	177749
140670	2542531	178311
24242	2543040	3565590
24680	2539869	187728
116643	2540917	176637
299331	2539035	191125
308212	2541861	175355
24231	2540392	177046
59109	2542207	175604

Figure 4: Table of aligned nodes from alignment of three graphs.

5 Conclusion

In this paper we presented CytoMCS, the first Cytoscape app for heuristically solving the maximum common edge subgraph for multiple directed or undirected graphs while (also for the first time) detecting partially conserved edges. It uses a fast iterative local search heuristic to quickly find large conserved substructures. The underlying algorithm optimizes a squared edge conservation score, that is able to extract not only fully conserved edges, but also partially conserved edges that can give additional insight into the common structure of the aligned graphs. The underlying algorithm is fully parallelized using Java 8 streams to automatically utilize multiple cores where available. The input and output of the app is provided through Cytoscapes standard data types, and edges are annotated with conservation scores for visualization and analysis. CytoMCS is available from the Cytoscape app store at http://apps.cytoscape.org/apps/cytomcs.

Acknowledgements

This work was supported by VILLUM Young Investigator grant no. 13154.

Conflict of interest statement: Authors state no conflict of interest. All authors have read the journal's Publication ethics and publication malpractice statement available at the journal's website and hereby confirm that they comply with all its parts applicable to the present scientific work.

References

[1] Raymond JW, Gardiner EJ, Willett P. RASCAL: calculation of graph similarity using maximum common edge subgraphs. Comput J. 2002;45:631–44.

[2] Kuchaiev O, Stevanović A, Hayes W, Pržulj N. GraphCrunch 2: software tool for network modeling, alignment and clustering. BMC Bioinformatics. 2011;12:24.

[3] Larsen SJ, Alkærsig FG, Ditzel HJ, Jurisica I, Alcaraz N, Baumbach J.. A simulated annealing algorithm for maximum common edge subgraph detection in biological networks. Proceedings of the 2016 Genetic and Evolutionary Computation Conference - GECCO 2016, New York, NY: ACM Press, 2016;341–8.

[4] Faisal FE, Meng L, Crawford J, Milenković T. The post-genomic era of biological network alignment. EURASIP J Bioinform Syst Biol. 2015;2015:3.

[5] Malod-Dognin N, Pržulj N. L-GRAAL: Lagrangian graphlet-based network aligner. Bioinformatics. 2015;31:2182–9.
[6] Godzik A, Skolnick J. Flexible algorithm for direct multiple alignment of protein structures and sequences. Comput Appl Biosci CABIOS. 1994;10:587–96.
[7] Andonov R, Malod-Dognin N, Yanev N. Maximum contact map overlap revisited. J Comput Biol. 2011;18:27–41.
[8] Malod-Dognin N, Pržulj N. GR-Align: fast and flexible alignment of protein 3D structures using graphlet degree similarity. Bioinformatics. 2014;30:1259–65.
[9] Elmsallati A, Clark C, Kalita J. Global alignment of protein-protein interaction networks: a survey. IEEE/ACM Trans Comput Biol Bioinform. 2016;13:689–705.
[10] Smoot ME, Ono K, Ruscheinski J, Wang PL, Ideker T. Cytoscape 2.8: new features for data integration and network visualization. Bioinformatics. 2011;27:431–2.
[11] Morris J, Jiao D. chemViz2: Cheminformatics App for Cytoscape. 2015. Available from: http://www.cgl.ucsf.edu/cytoscape/chemViz2/index.shtml. Accessed June 9, 2017.
[12] Ibragimov R, Malek M, Guo J, Baumbach J.. GEDEVO: an evolutionary graph edit distance algorithm for biological network alignment. Proceedings of the German Conference on Bioinformatics 2013, vol 34. Schloss Dagstuhl-Leibniz–Zentrum fuer Informatik, 2013;68–79.
[13] Malek M, Ibragimov R, Albrecht M, Baumbach J. CytoGEDEVO—global alignment of biological networks with Cytoscape. Bioinformatics. 2015;32:1259–61.
[14] Garey MR, Johnson DS. Computers and intractability Vol. 29. New York, NY: W.H. Freeman, 2002.
[15] Chatr-aryamontri A, Oughtred R, Boucher L, Rust J, Chang C, Kolas NK, et al. The BioGRID interaction database: 2017 update. Nucleic Acids Res. 2016;45:D369–79.
[16] Bovolenta LA, Acencio ML, Lemke N. HTRIdb: an open-access database for experimentally verified human transcriptional regulation interactions. BMC Genomics. 2012;13:405.
[17] Gama-Castro S, Salgado H, Santos-Zavaleta A, Ledezma-Tejeida D, Muñiz-Rascado L, García-Sotelo JS, et al. RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond. Nucleic Acids Res. 2015;44:D133–43.