Noname manuscript No.
(will be inserted by the editor)

Fast Subgraph Matching Strategies based on Pattern-only Heuristics

Antonino Aparo, Vincenzo Bonnici, Giovanni Micale, Alfredo Ferro, Dennis Shasha, Alfredo Pulvirenti, Rosalba Giugno*

Received: date / Accepted: date

Abstract Many scientific applications entail solving the subgraph isomorphism problem, i.e. given an input pattern graph, find all the subgraphs of a (usually much larger) target graph that are structurally equivalent to that input. Because subgraph isomorphism is NP-complete, methods to solve it have to use heuristics. This work evaluates subgraph isomorphism methods to assess their computational behavior on a wide range of synthetic and real graphs. Surprisingly, our experiments show that, among the leading algorithms, certain heuristics based only on pattern graphs are the most efficient.

Keywords Subgraph isomorphism, Networks biology, Search strategy

1 Introduction

In the last decade, technological advances have led to the acquisition of new biological and chemical data about genes, proteins, small molecules and regulatory elements, as well as their interactions within the cell.

International research projects that have accumulated genome-level data include the 1000 Genomes [1], Encyclopedia of DNA Elements [2], and The Cancer Genome Atlas (TGCA) [3].

A. Aparo, V. Bonnici and R. Giugno
Department of Computer Science, University of Verona, Verona, Italy.
E-mail: rosalba.giugno@univr.it

G. Micale, A. Ferro and A. Pulvirenti
Department of Clinical and Experimental Medicine, University of Catania, Catania, Italy.

D. Shasha
Computer Science Department, Courant Institute, New York University, New York, USA

[1] http://www.internationalgenome.org/
[2] http://www.encodeproject.org
[3] https://cancergenome.nih.gov/

Pairwise interactions between molecules are usually encoded by structures called networks, mathematically known as directed graphs. Such networks consist of nodes that represent molecules and edges that represent their interactions. Examples of biological networks include protein-protein interaction networks [33,31], genetic regulatory networks [17,19,34,13], metabolic networks [50,46,47] and disease correlation networks [25,22,26]. Moreover, protein and RNA structures also have spatial representations whose contact information can be encoded as graphs. For example, contact maps are networks that represent protein residues and the cut-off distances among them starting from a three dimensional protein structure [49,24].
Many biomedical data repositories use networks as the underlying data structure [2,11,12,9,51] and several algorithms have been proposed to analyze biological networks [5,53,15,4,21]. For example, network analysis can entail (i) finding network motifs [38,43], i.e. recurrent and statistically over-represented sub-networks on small and large networks [44], (ii) detecting active parts of molecules or regulatory circuits [45,42,32], (iii) finding ligands that bind to a protein [29] and (iv) computing similarity between networks, molecules and proteins [39,36,40,37].
All such forms of network analysis entail solving the subgraph isomorphism problem, which consists of finding all the possible subgraphs of a reference graph (called the target) that are structurally equivalent to another graph (called the pattern)[7].

Since subgraph isomorphism is an NP-complete problem [41] and the size of target networks generated by data-gathering projects is growing faster and faster, we need heuristic strategies in order to achieve scalability for large networks [20,6]. These heuristics typically exploit properties of the pattern and the target networks to constrain the search space and reduce the number of subgraph isomorphism calls. The types and the order of constraints to apply largely determine the performance of a subgraph isomorphism algorithm. Several solutions have been proposed to solve the subgraph isomorphism problem in polynomial time when dealing with graphs that have constraints in size or topology (e.g. graphs of bounded tree-width [1] or graphs of bounded feedback vertex set number [27]). In [28], the authors presented a comparison of general subgraph matching algorithms including VF2[14] and Ullmann[52] on small e medium graphs having from 200 nodes to 5000 nodes. Currently, the most scalable graph searching algorithms applicable in any kind of graphs are VF3 [8,7,10], RI[8,7], and an extension of RI called RI-DS[8]. These methods have been tested on graphs up to 10,000 vertices [10]. In the current work, we have generalized these tests to a wider variety of graphs to discover generally good algorithmic techniques. We used 1,008 targets created according to the Erdös-Rényi, Barabási, Forest Fire models. The experiments also change graph properties such as number of nodes, edge density, and numbers of labels. Altogether, we tested 150,000 different patterns. A set of topologies that are observed in nature in protein-protein interaction (PPI) networks. Vertices of such topologies represent proteins and edges report experimentally validated interactions between proteins. PPI networks of several living species have been

Symbol	Definition	Description
G	(V, E)	a generic graph
V	$\{v^1, v^2, ..., v^{\lvert V\rvert}\}$	the ordered vertex set of a graph
E	$\subseteq V \times V$	the set of edges of a graph
$N(v)$	$\{u \in V : (v, u) \in E\}$	the neighborhood of a vertex v
$N_{out}(v)$	$\{u \in V : \overrightarrow{(v, u)} \in E\}$	the outgoing neighborhood of a vertex v
$N_{in}(v)$	$\{u \in V : \overrightarrow{(u, v)} \in E\}$	the incoming neighborhood of a vertex v
A	$\{a^1, a^2, ..., a^k\}$	a set of labels
G	(V, E, α, β)	a labeled graph
α	$\alpha : V \mapsto A$	a function which maps vertices to their labels
β	$\beta : E \mapsto A$	a function which maps edges to their labels
G_p	$(V_p, E_p, \alpha_p, \beta_p)$	a labeled pattern graph
G_t	$(V_t, E_t, \alpha_t, \beta_t)$	a labeled target graph
v_p	$\{v_p^1, v_p^2, ..., v_p^{\lvert V_p\rvert}\}$	a rearrangement of the ordering of the vertices in V_p
$v_p[i]$	$\{v_p^1, v_p^2, ..., v_p^i\}$	the first i element of v_p
v_t	$\{v_t^1, v_t^2, ..., v_t^{\lvert V_t\rvert}\}$	a rearrangement of the ordering of the vertices in V_t
(v_p^i, v_t^n)	for $v_p^i \in v_p, v_t^n \in v_t$	a mapping of v_p^i to v_t^n, called matching pair

Table 1 List of mathematical symbols used.

evaluated with real and synthetic labeling. We found that algorithms that use only heuristics on patterns perform better than those using heuristics on target graphs, especially on large graphs. Simpler can be better.

2 Methods

2.1 Basic notions and problem definition

In what follows we provide basic definitions of labeled graphs and subgraph isomorphism. A labeled graph is defined as follows. The meaning of mathematical symbols used through the paper are reported in Table 1.

Definition 1 (*Labeled graph)* A labeled graph $G = (V, E, \alpha, \beta)$ is a quadruple, where V is a set of vertices. $\alpha : V \to A$ assigns a single label to each vertex. $E \subset V \times V$ is a set of directed edges between pairs of vertices and $\beta : E \to A$ assigns a single label to each edge.

A graph is *undirected* if the existence of an edge from v to v' implies the existence of an edge from v' to v (which collectively are represented by an undirected arc between them), otherwise the graph is *directed*. If $(u, v) \in E$ we say that v is a neighbor of u. Given a vertex $v \in V$, we denote with $N(v)$ the set of neighbors (i.e. the neighborhood) of node v. The number of neighbors of v, $|N(v)|$, is the degree of v. For directed graphs, we distinguish between the in-neighborhood $N_{in}(v) = \{u \in V : (u, v) \in E\}$ and the out-neighborhood $N_{out}(v) = \{u \in V : (v, u) \in E\}$. $|N_{in}(v)|$ and $|N_{out}(v)$ are the in-degree and the out-degree of v, respectively.

In an undirected graph, the neighborhood of a set of vertices U, denoted $N(U)$, is the set of vertices that are neighbors of at least one vertex in U. Similar definitions hold for $N_{in}(U)$ and $N_{out}(U)$ in the directed case.

The density of a graph is defined as $|E|/|V|^2$, where $|V|$ and $|E|$ are the cardinalities of the two sets. *Dense graphs* are those for which the density approaches the square of the number of vertices, otherwise, graphs are considered *sparse*.

Given a *target* graph and a normally much smaller *pattern* graph, solving the Subgraph Isomorphism problem (SubGI) informally means to find all subgraphs of the target that are topologically equal to the pattern. The Subgraph Isomorphism problem can be defined formally as follows:

Definition 2 (*Subgraph Isomorphism problem)* Given a *target graph* $G_t = (V_t, E_t, \alpha_t, \beta_t)$ and a *pattern graph* $G_p = (V_p, E_p, \alpha_p, \beta_p)$, the Subgraph Isomorphism (SubGI) problem is to find all injective functions $f_i : V_p \to V_i$, with $V_i \subseteq V_t$ and $|V_i| = |V_p|$ mapping each vertex of V_p to a unique vertex of V_i and that satisfy the following conditions:

1. $\forall v \in V_p, f_i(v) \in V_t$
2. $\forall v \in V_p \Rightarrow \alpha_p(v) = \alpha_t(f_i(v))$;
3. $\forall u, v \in V_p : u \neq v \Rightarrow f_i(u) \neq f_i(v)$;
4. $\forall (u, v) \in E_p \Rightarrow (f_i(u), f_i(v)) \in E_t$;
5. $\forall (u, v) \in E_p \Rightarrow \beta_p(u, v) = \beta_t(f_i(u), f_i(v))$.

Functions f_i is an injective mapping because the target graph may have edges that are not present in the pattern graph.

2.2 Heuristics for the SubGI problem

Subgraph isomorphism methods commonly uses State-space Search Tree ([8, 10]) and Constraint Programming techniques([35,52,48]).

State-space Search Tree paradigm. The *State-space Search Tree* (SST) paradigm, for each pattern graph, creates a tree containing all possible mappings of the pattern into the target graph.

Each path in the tree is a possible occurrence of the pattern in the target. Therefore, all paths in the tree have height equal to $|V_p|$, and each node maps a vertex of the pattern to a vertex in the target. Paths where all nodes meet the subgraph isomorphism constraints represent the matchings. Constraints are checked at each node in order to backtrack as soon as possible and to avoid re-traversing failing branches.

The ordering of pattern nodes may be the same in all paths, in which case the search strategy is said to follow a *static variable ordering*. Alternatively the ordering can differ on different paths, in which case the strategy is said to follow a *dynamic variable ordering*. Other aspects that vary in algorithms are whether the search strategy takes into account properties only from the target (*target-dependent*) [23], only from the pattern (*pattern-dependent*) [8], or both [10]. Moreover, methods may prune branches by looking at the values of the variables in the present computation (called State-driven variable ordering [8, 10]), or at the domains of all variables (domain-driven ordering[23]) passing the

subgraph isomorphism constraints. Any possible combination of these choices may significantly influence the performance of the algorithms.

In the *Constraint Programming* (CP) approach, the SubGI problem is modeled as a CP problem with a given set of variables, a set of possible values and a set of constraints which may involve multiple variables. The goal is to find assignments of values to each variable such that the constraints are satisfied. In the case of the SubGI problem, variables correspond to the pattern vertices and values correspond to the target vertices. Label compatibility is modeled by unary constraints, while binary constraints encode edge relationships. Each variable is assigned to a set of possible values, called a domain. The (CP) approach is based on the pre-computation of *compatibility domains* for each pattern vertex, i.e. all vertices in the target that can be matched according to the constraints with the pattern vertex.

This approach can outperform other methods for certain target graphs. However, it suffers from two main disadvantages. (i) The domain reduction between steps of the search process is computationally expensive in terms of running time. (ii) Domains take up a lot of memory.

2.3 Algorithms for the SubGI problem

State-of-the-art algorithms that use an State-Space Search Tree (SST) approach are VF3 [10], RI and RI-DS [7,8]. RI-DS is a modified version of RI taking advantages of pre-computed domains to reduce computational efforts during the matching phase.

VF3 uses a dynamic strategy. The order of the vertices may differ in different breaches of the search tree. The ordering of vertices is determined by taking into account the potential of vertices to reduce the search space based on current target and pattern features. Vertices with high degree in the pattern and target are chosen early. At each state, the algorithm applies a *look-ahead* evaluation on the nodes down the current branch. To reduce running costs, the algorithm computes, in a processing phase, *feasibility sets* for the pattern vertices. Such sets allow the prediction of failing candidate matches.

RI uses a *static* variable sorting that does not depend on the properties on the target graph (it is *target-independent*). The order of the vertices of the pattern is chosen by looking only at the characteristics of the pattern. RI orders vertices based on a scoring function: vertices having high degree and that are highly connected to vertices already present in the partial ordering are assigned higher scores. Higher scoring nodes must satisfy most constraints, so reduce the search space more effectively.

Formally, let $v_p[i] = (v_p^1, v_p^2, ..., v_p^i)$ be a partial ordering of i pattern vertices, with $i \leq |V_p|$. Let $v \in N(v_p[i])$ be a vertex candidate to be included in the order. We define three sets for v:

1. $N_1(v)$: set of neighbors of v that are included in the partial ordering;
2. $N_2(v)$: set of neighbors of v that are not included in the partial ordering but are neighbors of at least one vertex in the partial ordering;

3. $N_3(v)$: set of neighbors of v that are not in the partial ordering and are not even neighbors of vertices in the partial ordering;

The variable ordering strategy of RI consists in comparing the cardinality of these three sets lexicographically. Namely, given two candidate pattern vertices u and v, u is preferred to v iff: (i) $|N_1(u)| > |N_1(v)|$; or, in the case of a tie in (i), $|N_2(u)| > |N_2(v)|$; or in the case of a tie in (i) and (ii), (iii) $|N_3(u)| > |N_3(v)|$. If u and v are tied according to all three criteria, one of the two vertices is chosen arbitrarily.

Matching proceeds by comparing the nodes of the pattern graph with nodes of the target graph during the traversal of the SST tree. If the subgraph isomorphism conditions fail to be satisfied at some point during the traversal, then RI *backtracks* (by cutting the corresponding branch and reducing the search space).

During the search phase of RI, a matching pair (v_p^i, v_t^n) may be tested more than once, if it belongs to several matchings. RI checks the label compatibility each time the pair (v_p^i, v_t^n) is taken into account. Because this check operation is relatively costly, the algorithm may suffer in terms of performance.

By contrast, RI-DS verifies the label compatibility just once and stores the result in a compatibility domain map C. In RI-DS, $v_t^n \in C[i]$ if $\alpha_p(v_p^i) = \alpha_t(v_t^n)$ and $|N(v_p^i)| \leq |N(v_t^n)|$. In addition, before starting the searching process, RI-DS applies an Arch Consistency (AC) reduction procedure. AC reduction ensures that if two pattern vertices are connected, then each target vertex belonging to the compatibility domain of one of the two pattern vertices should be connected with at least one vertex belonging to the compatibility domain of the other pattern vertex. Formally, if two pattern vertices, v_p^i and v_p^j, are connected by an edge then $v_t^n \in C[i] \iff \exists v_t^m \in C[j] : (v_t^n, v_t^m) \in E_t$, and vice versa. AC reduction is applied sequentially to the domains of pattern vertices. The procedure discards target vertices that do not verify this constraint. The AC reduction is performed until no further changes are applied to domains.

3 Results

RI, RI-DS and VF3 have been implemented in C++. We used the source codes of the algorithms kindly provided by the authors. Each graph is stored in a text file in the form required by each algorithm which list the nodes and edges with their corresponding labels. The target graphs were generated in the RI format and converted with a script to obtain the dataset in VF3 format.

We have created a synthetic dataset having 1,008 target graphs to measure the performance of RI, RI-DS and VF3 in different contexts. The dataset holds target graphs of different sizes (from 200 to 20,000 number of target vertices) and density (from sparse to very dense) with few and many labels, according to the stochastic (*Erdös-Rényi*) and scale-free (*Barabási* and *Forest-Fire*) models [16, 3, 30]. Erdös-Rényi graphs were produced by using the python library *networkx*, while Barabási and Forest-Fire graph models were produced by using the *igraph* library.

species	vertices	edges	avg. degree
Bos taurus	8,474	84,468	9.97
Caenorhabditis elegans	6,173	52,368	8.48
Danio rerio	5,720	51,464	9.00
Drosophila melanogaster	8,624	78,932	9.15
Homo sapiens	12,575	173,780	13.82
Mus musculus	9,781	104,322	10.67
Rattus norvegicus	8,763	79,864	9.11
Saccaromyces cerevisiae	6,136	332,458	54.18
Takifugu rubipres	5,872	54,154	9.22
Xenopus tropicalis	6,153	59,538	9.68

Table 2 Number of vertices, number of edges and average degree of the 10 evaluated PPI networks.

Beside the synthetic benchmark, we also investigated the performances of the algorithms on real biological graphs[6]. The networks represent systems of physical interactions (expressed as edges) between proteins (represented by vertices) that have been established for a given living species. Table 2 reports for each species the number of vertices and edges of its protein-protein interactions (PPI). Proteins are often unique to a given species, however they may share functional behaviors that depend on the specific type of analysis researchers want to perform. Those behaviors serve as labels. For our experiments, vertices of PPI have been randomly labeled, by means of a uniform distribution, as it has been done in previously studies [7,8].

To obtain the patterns given all a target graph, a *random walk-based algorithm* traverses portions of that target graph to obtain substructures of different size and density that will constitute the patterns. The extraction algorithm starts from a vertex of the target graph and randomly selects one of its neighbors, proceeding recursively to the required number of vertices. Subsequently, edges not yet selected among the vertices of the obtained graph are added to the graph to reach some desired density.

The machine used for the test was an Intel Core i7-7700 3.60GHz 8-core CPU, 15 GB of RAM and Linux Ubuntu 17.04 64bit OS. The experiments were performed sequentially, setting a 3 minutes timeout for each run.

Code and datasets are available at *https://github.com/GiugnoLab/RI-synthds.*

Comparisons on the Erdös-Rényi dataset.

$G(n,p)$ is a random graph with N vertices where each pair of vertices $v1$ and $v2$ is connected with probability p, independently of any other pair. The expected number of edges is $E = pN(N-1)$. The Erdös-Rényi dataset holds 464 targets graphs having 100, 200, 500, 1,000, 2,000, 5,000, 10,0000, 20,0000 vertices with labels ranging from 0.1% to 30% depending on the number of vertices and with probability p: 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.4. Given these targets, up to 240 patterns are extracted for each target with different number of vertices (4, 8, 16, 24, 32, 64, 128, 256) and with three levels of pattern densities: sparse (0.1), medium dense (0.5), and dense

(≈1). Figure 1 compares the execution times of RI, RI-DS and VF3 on the *Erdös-Rényi* dataset.

In the Figure 1(a) the execution times are clustered according to the number of target vertices. Almost all RI and RI-DS execution times are better than those of VF3 though a few outliers are close. The other three plots display the performance on target graphs having 10,000 vertices. As the number of vertices and density parameter p increases, all algorithms take more time. However, RI and RI-DS show less dependence on size and density of the target graph. For example, for the maximum density value, VF3 triples its execution time while RI and RS-DS are less affected. Figure 1(c) shows the execution time of the three algorithms when the percentage of target labels changes. The running time varies little for RI and RI-DS and has a lower average time compared to VF3.

RI generally outperforms RI-DS in terms of average running time, however, the tail of outliers reported for RI-DS tends to be closer to the average time. A clear evidence of this behavior is shown in Figure 1(b) for grade of target density equal to 0.2. The average running time of RI-DS is higher than the RI performance, but the RI's outliers may require substantially more time to be solved w.r.t. the RI-DS's outliers. Figure 5 (a) reports the number of instances any approach has been registered as the faster solution. Results are grouped by number of target vertices and reported as percentage of the total amount of instances tested for a given group. RI and RI-DS have compatible running times in many tests, thus we grouped such instances with the label RI/RI-DS. The chart shows that on small target graph RI and RI-DS have comparable running times, instead, on increasing the number of target vertices, RI increases its performance. For this graph model, the additional computational effort to pre-compute and filter domains may be on average not advantageous for the RI-DS approach. However, the additional technique helps RI-DS to be more stable than the simpler RI version.

Comparisons on the Barabási dataset.

Biological networks (transcription factor, protein-protein interaction etc) can have a fractal-like topology, where some vertices, called *hubs*, have unusually high degree compared to the other vertices of the network. For example, in a metabolic network, there are organic compounds like ATP or ADP, that provides energy to drive many processes, constituting the hubs.

A model that has an observed stationary *scale-free distribution* using a preferential attachment mechanism like Barabási best reproduces these features. Such a network is constructed as follows. The network begins with m_0 vertices. New nodes are added to the network one at a time with $m(\leq m_0)$ connection to the vertices already in the network. The probability p_i that an edge of the new vertex connects to vertex i of the network depends monotonically on the degree k_i of i $p_i = \frac{k_i}{\sum_j k_j}$; the sum is made over all pre-existing nodes j. After t time steps, the generated network has $N = t + m_0$ vertices and $m_0 + m_t$ edges.

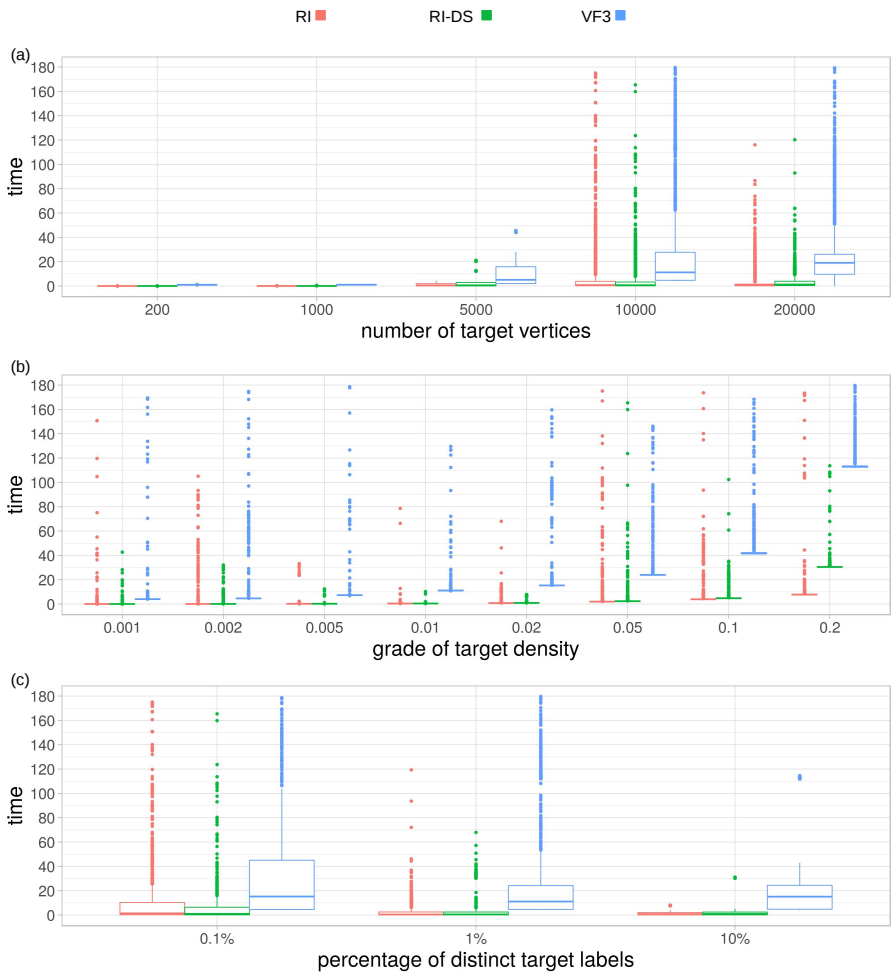

Fig. 1 Comparison of RI (red), RI-DS (green) and VF3 (blue) on the Erdös-Rényi dataset. Execution time is expressed in seconds. (a) Scalability of the algorithms depending on the size of the target graph, the x-axis reports the number of vertices of the graph. (b) Scalability of the algorithms as a function of the density of the target graph having 10k vertices. Here, the x-axis the probability p of the Erdös-Rényi models. (c) Performance of the three algorithms on the number of labels in the target graphs having 10,000 vertices. The horizontal axis is the ratio of number of distinct labels compared to number of nodes expressed as a percentage. Conclusion: RI or RI-DS take less time than VF3.

The Barabási dataset contains 384 target graphs and 28,800 pattern graphs. Target graphs have 200, 500, 1,000, 5,000, 10,000, 20,000 vertices, different number of outgoing edges (1, 2, 3, 6) and different numbers of labels with respect to the number of vertices (0.1%, 1%, 10%). Pattern vertices have different numbers of vertices (4, 8, 24, 32, 64) and different density (0.1, 0.5, ≈1). Times are grouped as in the Figure 1. Performances on this dataset are better

than on Erdös-Rényi. Figure 2 shows running times based on a log scale. RI and RI-DS are again faster than VF3 and less dependent on target feature variability.

For this dataset, the performance of RI-DS varies more than it does when applied to the Erdös-Rényi dataset, but RI-DS's performance outliers are still faster than RI anomalies. For the Erdös-Rényi dataset the three algorithms show an increasing time depending on the target dimension (see Figure 1 (a)). By contrast, performances of RI-DS tends to improve with increasing size and density of target graphs (see Figure 2, (a) and (b)). The improvement allows RI-DS to outperform RI on large and/or dense target graphs. Running time figures report averages that have been computed by filtering out SubGI instances for which at least one of the three approaches reached the 3 minutes timeout. By contrast, figures 5 and 6 are drawn by taking into account all the instances for which at least one approach has finished the execution within the timeout. Figure 5 (b) shows that, over the Barabási dataset, RI-DS is able to finish the largest number of instances within the timeout.

Comparisons on the Forest-Fire dataset. The Forest-Fire model is used to generate graphs such that vertices to be linked by an edge are chosen according to a geometric distribution with mean $p/(1-p)$. The parameter p is also called The forward burning probability. A total of 160 target graph were generated according to the Forest-Fire model by varying number of vertices, forward burning probability and number of distinct labels. Number of vertices ranged from 200, 500, 1,000, 5,000, 10,000 to 20,000. The forward burning probability ranged from 0.1, 0.3, 0.5, 0.7 to 0.9. The number of distinct labels was set to 0.1%, 1% and 10%. 12,000 patterns were extracted by varying the number of vertices (4,8,24,32 and 64). Tests regarding the Forest-Fire dataset show similar trends to those of the Erdös-Rényi model. The main difference between the two benchmarks is given by the improved performance of RI-DS that is on average comparable to, and often better than, the RI running times. This result is also confirmed by Figure 5 (c), which shows that RI and RI-DS have a comparable percentage of instances for which both algorithms are registered as the fastest solution.

Comparisons on the Protein-Protein Interaction (PPI) dataset. The benchmark consists of a total of 10 species-specific PPIs downloaded from the STRING resource [18] and stored as undirected network. PPIs were randomly labeled with 32, 64, 128, 256, 512, 1024 and 2028 distinct number of labels. Additionally, an unlabeled version (with number of labels equal to 1) has been included in the benchmark. Patterns of different densities were extracted from the PPIs for several dimensions (number of edges vary from 4, 8, 16, 32, 64, 128 to 256). Results are shown in Figure 4. In section (a) of the figure, results are grouped by species and exhibit a clear dependence from such a grouping that reflect the differing densities of such structures. In fact, the *Saccharomyces cerevisiae* is the PPI with the highest density and it corresponds to the highest average running time of the RI and RI-DS approaches. The two approaches outperform VF3 independently of how results are grouped, however. The RI approach is generally better than RI-DS. Figure 5 (d) groups results by num-

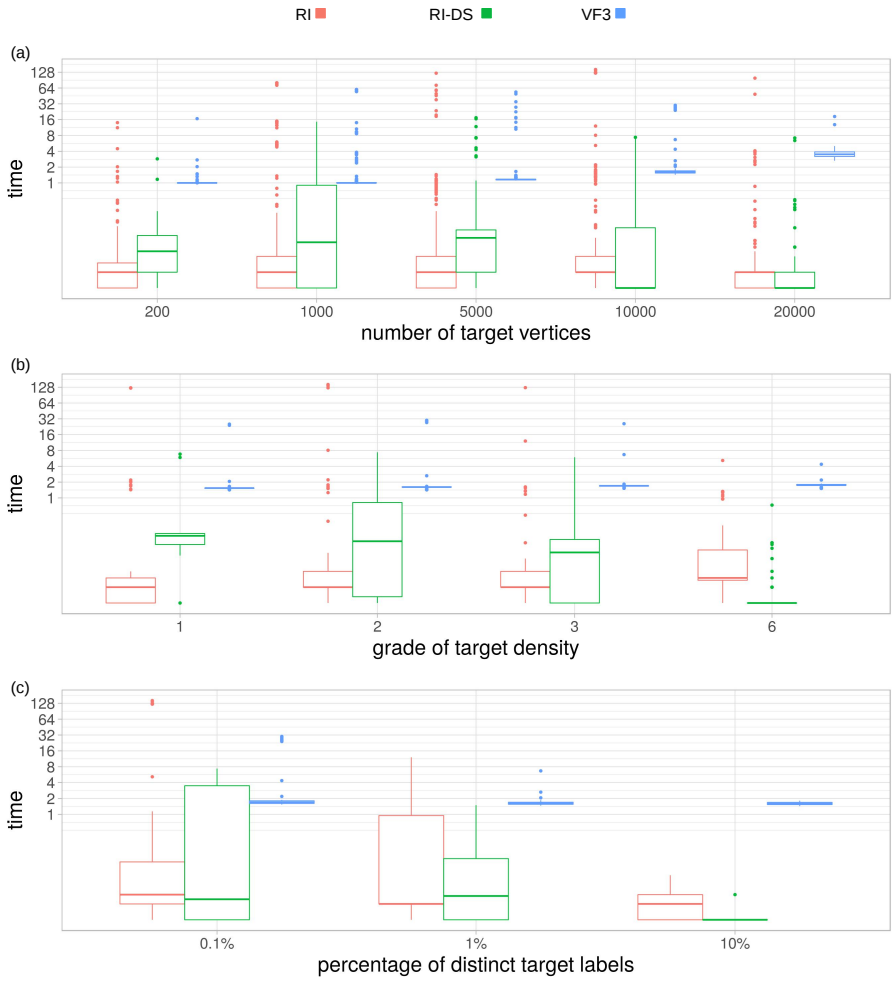

Fig. 2 This graph compares execution time (in seconds). on the scale-free *Barabási-Albetr* dataset. Here colors represent algorithms: RI (red), RI-DS (green) and VF3 (blue). Figure (a) shows how the algorithms scale with the size of the target graph, where the x-axis gives the number of vertices of the target graph, and the y-axis represents the time on a log scale. Figure (b) shows how the algorithms scale with the density of the target graph for targets having 10,000 vertices. The x-axis represents the density using the parameter m of the Barabási-Albert models. Figure (c) shows how the algorithms scale with the number of labels of the target graph for targets having 10,000 vertices. The x-axis represents the number of distinct labels in the target graph as a percentage of the number of vertices. Conclusion: RI and RI-DS take the least time in all scenarios.

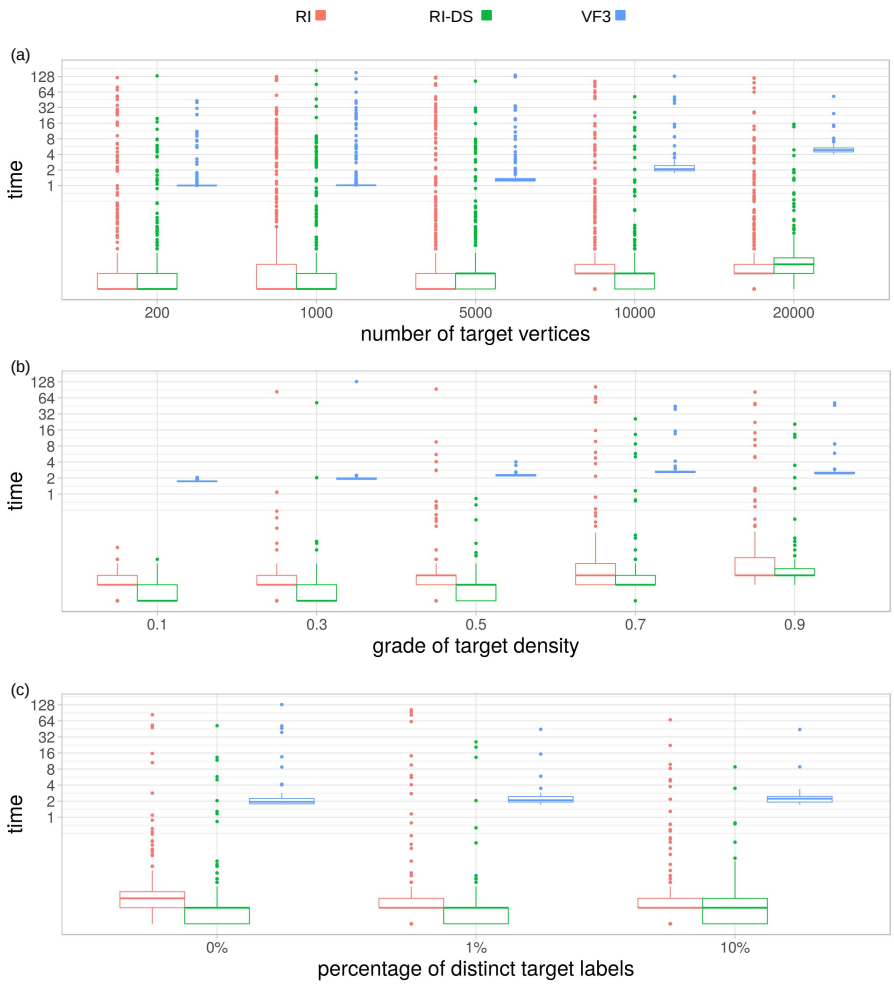

Fig. 3 Comparison of RI (red), RI-DS (green) and VF3 (blue) on the Forest-Fire dataset. Execution time is expressed in seconds. (a) Scalability of the algorithms depending on the size of the target graph, the x-axis reports the number of vertices of the graph. (b) Scalability of the algorithms as a function of the density of the target graph having 10k vertices. Here, the x-axis the forward burning probability p of the Forest-Fire models. (c) Performance of the three algorithms on the number of labels in the target graphs having 10,000 vertices. In the third graph, the number of distinct labels in the target graph is expressed as a percentage of the number of vertices. Conclusion: RI and RI-DS take the least time in all scenarios.

ber of target vertices, and Figure 6 groups results based on the number of distinct target labels and number of pattern vertices. Figure 6 shows that RI enjoys better performance than RI-DS especially for large patterns.

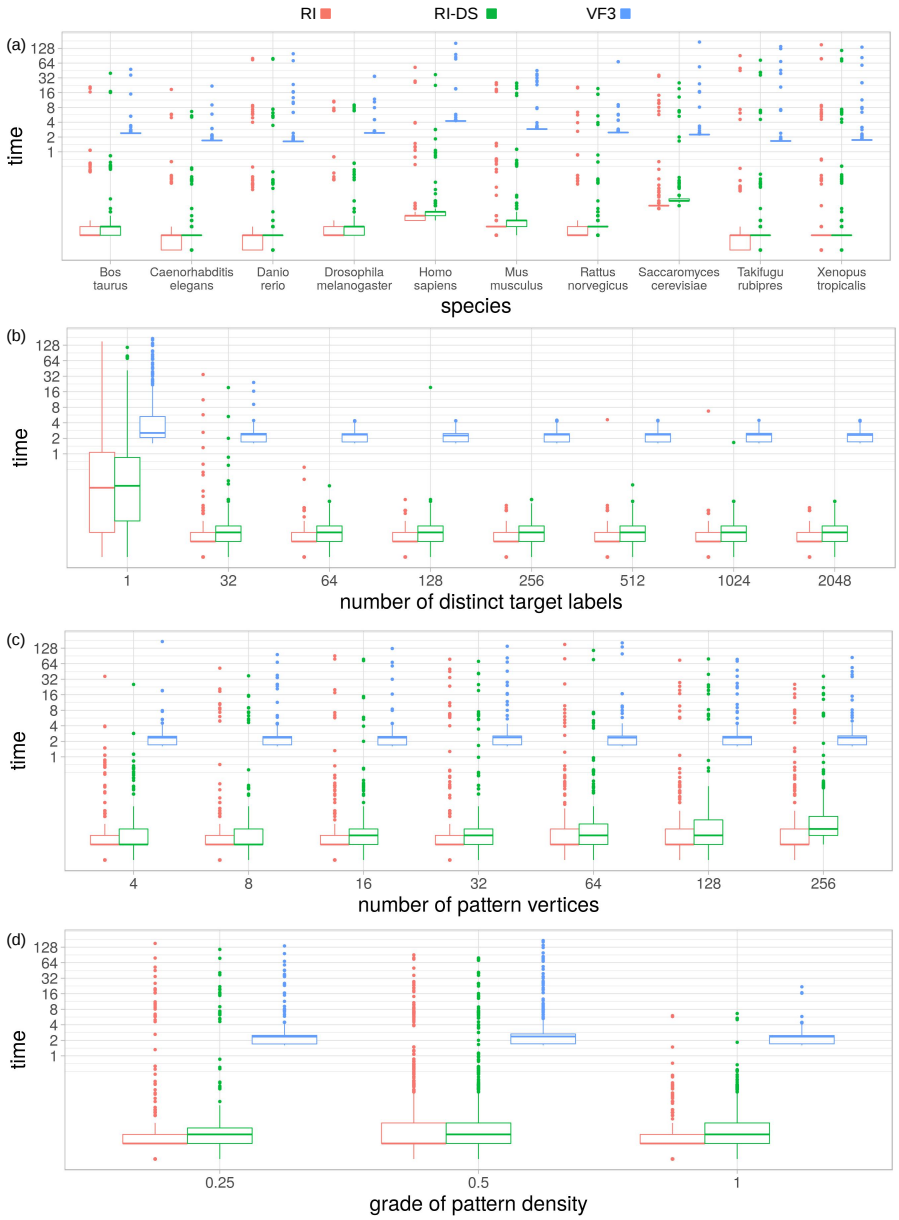

Fig. 4 Comparison of RI (red), RI-DS (green) and VF3 (blue) on the PPI dataset. Execution time is expressed in seconds. (a) Scalability of the algorithms on the various target graphs (for each species, there is one topology but several labelings), the x-axis refers to the reference organism of the graph. Times are expressed in log scale. (b) Performance of compared algorithms on the number of labels in the target graphs. (c) Scalability of the algorithms on the size of the pattern graph, the x-axis refers to the number of vertices of the pattern graph. (d) Scalability of the algorithms on the density of the pattern graph. Conclusion: RI and RI-DS take the least time in all scenarios.

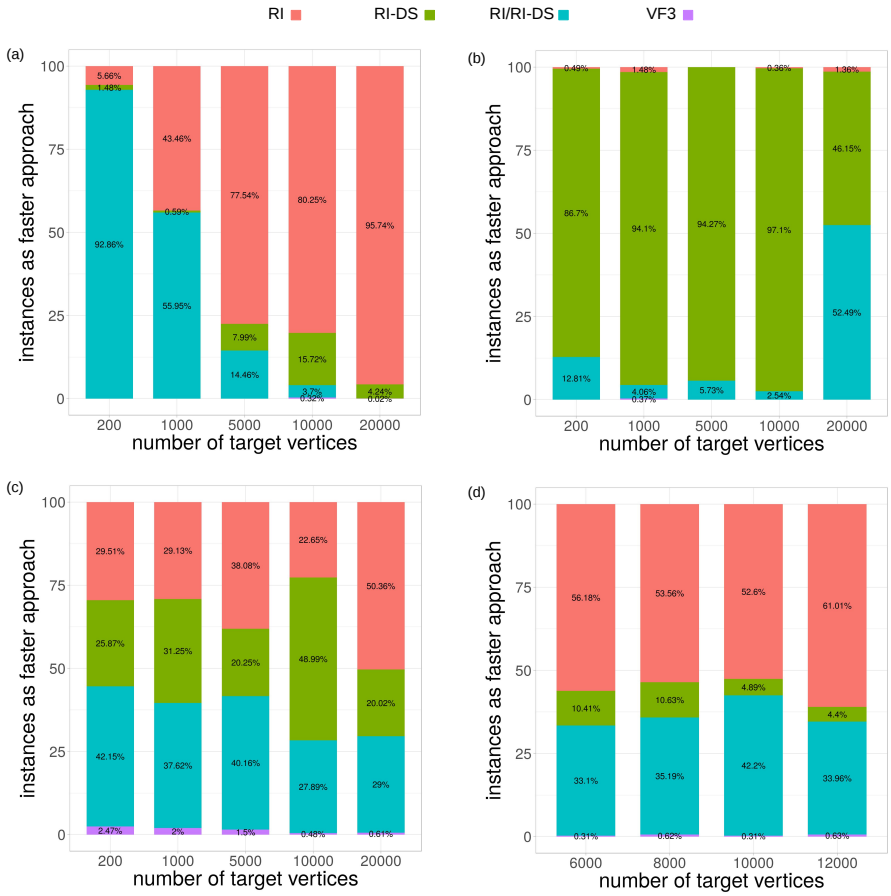

Fig. 5 Number of instances, expressed as percentages w.r.t. the total amount of tests, that one algorithm performed as the fastest approach over all benchmarks: Erdös-Rényi (a), Barabási-Albert (b), Forest-Fire (c) and PPI (d). The cases in which the execution times of RI and RI-DS are comparable are reported as RI/RI-DS. RI in red, RI-DS in green, RI/RI-DS in blue and VF3 in purple. The results are grouped by number of vertices in the target graph. RI or RI/RI-DS tends to be the best when there are many vertices. For few vertices, RI-DS is best.

4 Conclusion

When faced with an NP-complete problem, heuristics are necessary. Subgraph isomorphism has given rise to many sophisticated ones, some that depend on the patterns to be matched and some that depend on the much larger target graph. This work shows that heuristics depending only on the pattern graph work very well, benefiting not only queries on single target graphs but on multiple target graphs.

This is of course a field of ongoing research, therefore we doubt this is the last word.

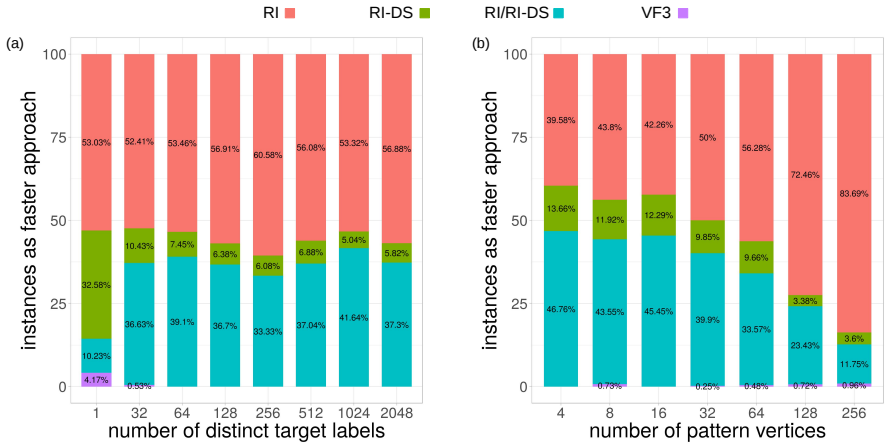

Fig. 6 Number of instances, expressed as percentages w.r.t. the total amount of tests, that one algorithm performed as the fastest approach over the PPI benchmark. The cases in which the execution times of RI and RI-DS are equal are reported as RI/RI-DS. RI in red, RI-DS in green, RI/RI-DS in blue and VF3 in purple. Results are grouped by number of distinct target labels (a) and by number of pattern vertices (b). RI is best when there are many vertices in the pattern graph.

Acknowledgements We would like to thank the authors of algorithm VF3 to have provided the software and for all your support on testing it. R.G., V.B., and A. A. would like to acknowledge the support of the Italian Ministry of Education, Universities and Research (MIUR) "Dipartimenti di Eccellenza 2018-2022", and grants 2016-JPVR16FNCL, 2017-B33C17000440003, GNCS-INDAM. A.P., G.M. and A.F. acknowledge the support of Italian Ministry of Education, Universities and Research (Ministero dell'Istruzione, dell'Università e della Ricerca, MIUR) grant SCN_00451. D.S. would like to acknowledge the support of the U.S. National Science Foundation grants MCB-1158273, IOS-1339362, and MCB-1412232.

References

1. N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995.
2. G. D. Bader, M. P. Cary, and C. Sander. Pathguide: a Pathway Resource List. *Nucleic Acids Research*, 34(suppl1):D504–D506, 2006.
3. AL. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.
4. AL. Barabasi, N. Gulbahce, and J. Loscalzo. Network medicine: a network-based approach to human disease. *Nature reviews. Genetics*, 12(1):56–68, January 2011.
5. AL. Barabasi and ZN. Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews. Genetics*, 5(2):101–113, February 2004.
6. V. Bonnici, F. Busato, G. Micale, N. Bombieri, A. Pulvirenti, and R. Giugno. APPAGATO: An APproximate PArallel and stochastic GrAph querying TOol for biological networks. *Bioinformatics*, 32(14):2159–2166, 2016.
7. V. Bonnici and R. Giugno. On the Variable Ordering in Subgraph Isomorphism Algorithms. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(1):193–203, 2017.

8. V. Bonnici, R. Giugno, A. Pulvirenti, D. Shasha, and A. Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC Bioinformatics*, 14 Suppl 7:S13, 2013.
9. V. Bonnici, F. Russo, N. Bombieri, A. Pulvirenti, and R. Giugno. Comprehensive Reconstruction and Visualization of Non-Coding Regulatory Networks in Human. *Frontiers in Bioengineering and Biotechnology*, 2:69, 2014.
10. V. Carletti, P. Foggia, A. Saggese, and M. Vento. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2017.
11. E. G. Cerami, B. E. Gross, E. Demir, I. Rodchenkov, A. Babur, N. Anwar, N. Schultz, G. D. Bader, and C. Sander. Pathway Commons, a web resource for biological pathway data. *Nucleic Acids Research*, 39(suppl1):D685–D690, 2011.
12. A. Chatr-aryamontri, R. Oughtred, L. Boucher, and J. et al. Rust. The BioGRID interaction database: 2017 update. *Nucleic Acids Research*, 45(D1):d369–d379, 2017. Exported from https://app.dimensions.ai on 2018/08/18.
13. C. Christensen, J. Thakar, and R. Albert. Systems-level insights into cellular regulation: inferring, analysing, and modelling intracellular networks. *IET Syst Biol*, 1(2):61–77, March 2007.
14. L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1367–1372, 2004.
15. P. Csermely, T. Korcsmaros, HJ. Kiss, G. London, and R. Nussinov. Structure and dynamics of molecular networks: A novel paradigm of drug discovery: A comprehensive review. *Pharmacology and Therapeutics*, 138(3):333–408, 2013.
16. P. Erdos and A. Rényi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.
17. P. Faccioli, P. Provero, C. Herrmann, A. M. Stanca, C. Morcia, and V. Terzi. From single genes to co-expression networks: Extracting knowledge from barley functional genomics. *Plant Molecular Biology*, 58(5):739–750, Jul 2005.
18. A. Franceschini, D. Szklarczyk, S. Frankild, M. Kuhn, M. Simonovic, A. Roth, J. Lin, P. Minguez, P. Bork, and C. et al. Von Mering. STRING v9. 1: protein-protein interaction networks, with increased coverage and integration. *Nucleic acids research*, 41(D1):D808–D815, 2012.
19. M. B. Gerstein, A. Kundaje, M. Hariharan, S. G. Landt, KK. Yan, C. Cheng, and Mu et al. Architecture of the human regulatory network derived from ENCODE data. *Nature*, 489(7414):91–100, September 2012.
20. R. Giugno, V. Bonnici, N. Bombieri, A. Pulvirenti, A. Ferro, and D. Shasha. GRAPES: A Software for Parallel Searching on Biological Graphs Targeting Multi-Core Architectures. *PLoS ONE*, 8(10), 2013.
21. A. Giuliani, S. Filippi, and M. Bertolaso. Why network approach can promote a new way of thinking in biology. *Frontiers in genetics*, 5:83, 2014.
22. KI. Goh and IG. Choi. Exploring the human diseasome: the human disease network. *Briefings in Functional Genomics*, 11(6):533–542, 2012.
23. R. M. Haralick and G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3):263 – 313, 1980.
24. J. Hu, X. Shen, Y. Shao, C. Bystroff, and M. J. Zaki. Mining Protein Contact Maps. In *Proceedings of the 2Nd International Conference on Data Mining in Bioinformatics*, BIOKDD'02, pages 3–10, London, UK, UK, 2002. Springer-Verlag.
25. V. Janjic and N. Przulj. Biological function through network topology: a survey of the human diseasome. *Briefings in Functional Genomics*, 11(6):522–532, 2012.
26. W. Kenneth and R. Leslie. Diseasome: an approach to understanding gene-disease interactions. *Annual review of nursing research*, 29:55–72, 2011.
27. S. Kratsch and P. Schweitzer. Isomorphism for graphs of bounded feedback vertex set number. In Haim Kaplan, editor, *Algorithm Theory - SWAT 2010*, pages 81–92, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
28. J. Lee, W. S. Han, R. Kasperovics, and J. H. Lee. An in-depth comparison of subgraph isomorphism algorithms in graph databases. In *Proceedings of the VLDB Endowment*, volume 6, pages 133–144. VLDB Endowment, 2012.
29. N. W. Lemons, B. Hu, and W. S. Hlavacek. Hierarchical graphs for rule-based modeling of biochemical systems. *BMC Bioinformatics*, 12(1):45, Feb 2011.

30. J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005.
31. S. Li, C. M. Armstrong, N. Bertin, H. Ge, and S. et al. Milstein. A Map of the Interactome Network of the Metazoan C. elegans. *Science*, 303(5657):540–543, 2004.
32. S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100(21):11980–11985, 2003.
33. A. R. Mashaghi, A. Ramezanpour, and V. Karimipour. Investigation of a protein complex network. *The European Physical Journal B - Condensed Matter and Complex Systems*, 41(1):113–121, Sep 2004.
34. MN. McCall. Estimation of gene regulatory networks. *J. Postdoctoral Res.*, 1(1):60–69, January 2013.
35. J.J. McGregor. Relational consistency algorithms and their application in finding subgraph and graph isomorphisms. *Information Sciences*, 19(3):229 – 250, 1979.
36. G. Micale, A. Continella, A. Ferro, R. Giugno, and A. Pulvirenti. GASOLINE: a Cytoscape app for multiple local alignment of PPI networks [version 2; referees: 2 approved, 1 approved with reservations] . *F1000Research*, 3(140), 2014.
37. G. Micale, A. Ferro, A. Pulvirenti, and R. Giugno. SPECTRA: An Integrated Knowledge Base for Comparing Tissue and Tumor-Specific PPI Networks in Human. *Frontiers in Bioengineering and Biotechnology*, 3:58, 2015.
38. G. Micale, R. Giugno, A. Ferro, M. Mongiovì, D. Shasha, and A. Pulvirenti. Fast analytical methods for finding significant labeled graph motifs. *Data Mining and Knowledge Discovery*, 32(2):504–531, Mar 2018.
39. G. Micale, A. Pulvirenti, R. Giugno, and A. Ferro. GASOLINE: a Greedy And Stochastic algorithm for Optimal Local multiple alignment of Interaction NEtworks. *PLOS ONE*, 9(6):1–15, 06 2014.
40. G. Micale, A. Pulvirenti, R. Giugno, and A. Ferro. Proteins comparison through probabilistic optimal structure local alignment. *Frontiers in Genetics*, 5:302, 2014.
41. R. G. Michael and S. J. David. Computers and intractability: a guide to the theory of NP-completeness. *WH Free. Co., San Fr*, pages 90–91, 1979.
42. T. Milenkovic and N. Przulj. Uncovering Biological Network Function via Graphlet Degree Signatures. *Cancer Informatics*, 6:CIN.S680, 2008.
43. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, 2002.
44. B. Palsson and K. Zengler. The challenges of integrating multi-omic data sets. *Nature Chemical Biology*, 6:787, 2010.
45. N. Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.
46. E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. Hierarchical Organization of Modularity in Metabolic Networks. *Science*, 297(5586):1551–1555, 2002.
47. H. Redestig, J. Szymanski, M. Y. Hirai, J. Selbig, L. Willmitzer, Z. Nikoloski, and K. Saito. *Data Integration, Metabolic Networks and Systems Biology*, chapter 9, pages 261–316. American Cancer Society, 2018.
48. C. Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artificial Intelligence*, 174(12):850 – 864, 2010.
49. K. Suvarna Vani and K. Praveen Kumar. Feature Extraction of Protein Contact Maps from Protein 3D-Coordinates. In D. K. Mishra, A. T. Azar, and A. Joshi, editors, *Information and Communication Technology*, pages 311–320, Singapore, 2018. Springer Singapore.
50. M. Terzer, N. D. Maynard, M. W. Covert, and J. Stelling. Genome-scale metabolic networks. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 1(3):285–297, 2009.
51. S. Turkarslan, E. J. Wurtmann, WJ. Wu, and N. et al. Jiang. Network portal: a database for storage, analysis and visualization of biological networks. *Nucleic Acids Research*, 42(D1):D184–D190, 2014.
52. J. R. Ullmann. Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism. *J. Exp. Algorithmics*, 15:1–64, 2011.
53. D. Yu, M. Kim, G. Xiao, and T.H. Hwang. Review of Biological Network Data and Its Applications. *Genomics Inform*, 11(4):200–210, 2013.