Algorithms for Regular Tree Grammar Network Search and their application to mining human-viral infection patterns

Ilan Smoly[1] *, Amir Carmel[1] *, Yonat Shemer-Avni[2], Esti Yeger-Lotem[3] **, and Michal Ziv-Ukelson[1]**

[1] Department of Computer Science, Ben-Gurion University of the Negev.
Email: {ilansmoly,karmela,michaluz}@cs.bgu.ac.il
[2] Department of Virology, Ben-Gurion University of the Negev.
Email: yonat@bgu.ac.il
[3] Department of Clinical Biochemistry and Pharmacology, Ben-Gurion University of the Negev.
Email: estiyl@bgu.ac.il

Abstract. Network querying is a powerful approach to mine molecular interaction networks. Most network querying tools support queries in the form of a template sub-network, in case of topology-constrained queries, or a set of colored vertices in case of topology-free queries. A third approach is grammar-based queries, which are more flexible and expressive as they allow the addition of logic rules to the query. Previous grammar-based querying tools defined queries via string grammars and identified paths in graphs. In this paper, we extend the scope of grammar-based queries to regular tree grammar (RTG), and the scope of the identified sub-graphs from paths to trees. We introduce a new problem and propose a novel algorithm to search a given graph for the k highest scoring sub-graphs matching a tree accepted by an RTG. Our algorithm is based on dynamic programming and combines an extension to k-best parsing optimization with color coding. We implement the new algorithm and exemplify its application to mining the human-viral interaction network. Our code is available at *http://www.cs.bgu.ac.il/~smolyi/RTGnet/*

1 Introduction

Molecular interaction networks have become a prominent resource for inferring protein functions, detecting cellular processes, predicting disease pathways, and more [5]. A powerful approach to mine these networks is via network querying: Given a query sub-network, network querying tools identify instances within the network that match the query. Previous network querying tools mostly support queries in the form of a "template" sub-network that is confined to a specific topology, such as paths [33], trees [28,30] or graphs with bounded tree-width [10,30]. Other approaches, denoted topology free, support queries in the

* These authors contributed equally to the paper.
** Coresspoinding authors.

form of a set of colors, and seek in the network vertices with matching colors that can be connected by a spanning tree [7,23].

Here we focus on a third approach to network querying, denoted grammar-based queries. Such queries are given a grammar as input, and seek in the network sub-graphs accepted by the grammar. On one hand, grammar-based queries are more flexible than other approaches because neither the exact topology nor the exact set of colors are specified in advance. On the other hand, they are more expressive as they allow the addition of logic rules to the query. Previous works in this area defined queries via string grammars, including regular expressions [12,22] and context-free grammars [31], and identified paths in graphs. Here, we extend the scope of the grammars to regular tree grammars (RTGs), and the scope of the identified sub-graphs from paths to trees.

Note that even the most basic problem variant, that of finding a path in a network based on a regular grammar query descriptor, is generally intractable [26]. Thus, previous grammar-based search approaches either bounded the size of the sought paths [12] or applied inadmissible heuristics such as seed-and-extend [22]. In our solution, we also bound the size of the sought subtrees.

We introduce a new problem and propose a novel algorithm to search a given graph for the k highest scoring sub-graphs, of size bounded by m, matching a tree accepted by a regular tree grammar (RTG). Our proposed approach is based on a two-stage dynamic programming algorithm combining RTG k-best parsing with network search. One challenge encountered here is that RTGs recognize ordered trees, while the subtrees in the graph to be searched are unordered. We handle this by incorporating redundancy-avoidance logic and show how to extend an efficient k-best parsing algorithm [19] to support it. Another challenge is that in order to support the bottom-up parsing employed in this paper, we need to use a normalized form of the grammar, similarly to the way the CYK algorithm parses context-free string grammars in Chomsky normal form (CNF) [24]. For this purpose, we harness a previously known binarization approach to RTG parsing [25]. To handle the exponentially growing search space, and to avoid cycles, we employ color-coding [2], which was also used by other network querying tools [7,10,30,33], and show how to probabilistically bound the number of color-coding iterations needed to compute the k-best subtrees for a given error threshold.

We implement the proposed algorithm in a software package, denoted RTGnet, and demonstrate its usage in two separate applications which demonstrate the expressive power of RTGs and the topological flexibility of the identified patterns. Due to space constraints, supplementary materials including some of the figures and proofs to all theorems, are deferred to an appendix that is available at *http://www.cs.bgu.ac.il/~smolyi/RTGnet/*

2 Regular Tree Grammars and Curry-Encoding

The patterns we seek are ordered rooted trees, to be recognized by Regular Tree Grammars [9].

Definition 1. *A Regular Tree Grammar (RTG) is a tuple* $A = \{N, \Sigma, P, S\}$, *where* N *is a finite set of non-terminal symbols,* Σ *is an alphabet of terminal symbols,* S *is the initial non-terminal, and* P *is a finite set of productions of type* $X \to a(R)$, *where* R *is a regular expression over* N *and* $a \in \Sigma$.

Each production rule $X \to a(R)$ defines a vertex x labelled a, while R expresses the non-terminal symbols that are used to recursively generate the child subtrees of x. If $R = \epsilon$, then x is a leaf. Let $T(\Sigma)$ denote the set of all ordered, rooted trees with vertex labels in Σ. An ordered rooted vertex-labelled tree $\tau \in T(\Sigma)$ is accepted by A if there is a chain of consecutive derivations of production rules from P that starts in S and generates τ. The language $L(A) \subseteq T(\Sigma)$ is the set of all trees accepted by A. Regular Tree Languages (RTL) is the class of tree languages that are generated by RTGs.

RTGs are classically categorized as either ranked or unranked. Ranked RTGs generate trees for which there is a global bound on the number of children each vertex may have. Unranked RTGs, which are the more general case supported by our framework, have no such bound. For our bottom-up RTG parsing, we harness a recursive deterministic binarization approach, denoted *curryfication* [8] (see Figure S1). This approach encodes an unranked tree τ into a ranked binary tree, denoted $curry(\tau)$, as follows. Given a tree $\tau = a(\tau_1, \ldots, \tau_n) \in T(\Sigma)$, $curry(\tau) = @(curry(\tau'), curry(\tau_n))$ with $\tau' = a(\tau_1, \ldots, \tau_{n-1})$. If $n = 0$ then $curry(\tau) = a$. The *extension operator* $@$ is used to denote the labels of the internal vertices of the binarized trees, while their leaves correspond to the vertices of the original tree. Correspondingly, the given RTG A is also converted to a Curry-encoded RTG (defined below), $curry(A)$, that accepts the curry encodings of the trees in $L(A)$, such that the bijection $curry(L(A)) = L(curry(A))$ is obeyed [8,9].

Definition 2. *A Curry-Encoded Regular Tree Grammar is a tuple* $A = \{N, \Sigma, P, S\}$ *where* N *is a finite set of non-terminal symbols,* Σ *is an alphabet of terminal symbols,* S *is the initial non-terminal, and* P *is a finite set of productions of type* $X \to @(Y, Z)$ *or* $X \to a$, *where* $X, Y, Z \in N$ *and* $a \in \Sigma$ *and* $@ \notin \Sigma$.

In [8,9] it is shown that any unranked RTG A can be translated in linear time to $curry(A)$ via the construction of the corresponding deterministic Stepwise Automaton. Furthermore, it is shown that for any recognizable RTL $L \subseteq T(\Sigma)$ there is a unique minimal deterministic Stepwise Automaton accepting L [25].

Based on this, in the rest of this paper we assume that the input grammar A is in Curry-Encoded form, and denote by g the number of its production-rules. The binarization of the candidate trees will be done during the parsing process. Also, for convenience, we write $X \to Y@Z$ when $X \to @(Y, Z)$.

3 The RTG Network Parse-and-Search Problem

Given an RTG A of size g, a directed vertex-labeled graph $G = (V, E)$, and an integer parameter m. Let $T[G, A, m]$ be a set of labeled trees, such that for any tree $\tau \in T[G, A, m]$: (1) $\tau \in L(A)$, (2) $|\tau| \leq m$, and (3) τ is a subtree

in G. Let $s(\tau) \in \Re$ denote a predefined monotonic scoring scheme on τ, where $\Re$ denotes the real numbers. Let $Max_s^k(T[G, A, m])$ denote the subset of k-best scoring subtrees in $T[G, A, m]$ according to s. Our search is based on the following optimization problem.

Problem 1 (RTG Network Parse-and-Search Problem). Given an RTG $A = \{N, \Sigma, P, S\}$, a monotonic scoring scheme s, a graph $G = (V, E)$, and two integer parameters m and k. The RTG Network Parse-and-Search Problem is to compute $Max_s^k(T[G, A, m])$.

The problem of deciding whether a given ordered rooted tree is accepted by a given RTG can be solved in polynomial time. However, solving Problem 1 entails extending RTG parsing to handle, as input, a *directed graph* rather than an *ordered rooted tree* and to support a local RTG derivation-tree search rather than just tree parsing.

To handle the exponentially growing search space, and to avoid cycles, we apply iterative randomized color-coding to our search space [2]. Color-coding is a probabilistic approach that bounds the error expectancy by executing multiple graph coloring iterations. In each iteration, every vertex in the queried network is assigned a color chosen randomly out of m colors. The colored network is then searched for colorful subgraphs in which each color appears exactly once. Thus, instead of maintaining an enumeration space of size $\binom{n}{m}$, due to all possible selections of subsets of size m from among n vertices, one can maintain a search space of size 2^m enumerating sets of m distinct colors in considerably lower complexity. In each color-coding trial, the probability to obtain a certain tree of the sought k-best trees is $\frac{m!}{m^m} \geq e^{-m}$ (i.e. the probability that a tree of m vertices is colorful). The following theorem, which is proven in the Appendix, bounds the number of required color-coding iterations.

Theorem 1. *For any $\epsilon > 0$, after $e^m \ln(\frac{k}{\epsilon})$ iterations, the output list contains all the k-best subtrees with error probability $\leq \epsilon$.*

We next formalize the work per color-coding iteration, which will be the subject of the next section. Let $C : V \rightarrow \{1, \ldots, m\}$ be a coloring function. We define $T^C[G, A, m]$ to be a subset of $T[G, A, m]$, such that for any tree $\tau \in T^C[G, A, m]$, each vertex of τ is distinctly colored according to C (i.e. τ is colorful). Let $Max_s^k(T^C[G, A, m])$ denote the subset of k-best scoring subtrees in $T^C[G, A, m]$ according to s.

Problem 2 (RTG Colored Network Parse-and-Search Problem). Given an RTG $A = \{N, \Sigma, P, S\}$, a coloring function C, a monotonic scoring scheme s, a vertex-labelled graph $G = (V, E)$, and two integer parameters m and k. The RTG Colored Network Parse-and-Search Problem is to compute $Max_s^k(T^C[G, A, m])$.

An overall illustration of our framework for solving Problem 1, by solving Problem 2 within color-coding iterations, is shown in Figure S2.

Across color-coding trials, we maintain a global sorted list of size k, which stores the trees corresponding to the k-best derivations found so far. After each

color-coding iteration, this list is updated with the k-best derivations yielded by that iteration. This is done in $O(km \log k)$ time and $O(m \cdot k)$ space, i.e. without increasing the overall time and space complexities of the algorithm we propose for solving Problem 2, which will be described and analyzed in the next section.

4 Algorithms for Solving Problem 2

In this section we describe a two-stage algorithm for solving Problem 2. We formulate our dynamic programming algorithm as an instance of optimal derivation in the ordered hypergraph framework [13,19,27,29]. In the first stage of our algorithm (Section 4.1), the curryfied representations of all the candidate subtrees within the graph G are constructed and encoded in an ordered acyclic hypergraph. Then, in the second stage (Section 4.2), the hypergraph is processed again to compute the k-best scoring subtrees in the hypergraph.

4.1 Stage 1: hypergraph construction and its optimal derivation

A directed ordered hypergraph is a pair $H = (V_H, E_H)$, where V_H is the set of hypernodes, and E_H is the set of ordered, directed hyperedges. Each hyperedge e is a pair $\langle head(e), tail(e) \rangle$, where $head(e)$ is a hypernode called the *head* of the hyperedge and $tail(e)$ is an ordered pair of hypernodes, called the *tail* of the hyperedge. Denote by $t_i(e)$ the *i'th* element of the tail of e. Note that the hyperedges in our approach are all binary. Thus, henceforth we write a hyperedge with head h and tail t_1,t_2 as $(h \leftarrow \langle t_1, t_2 \rangle)$.

Given an RTG $A = \{N, \Sigma, P, S\}$, a directed graph $G = (V, E)$, a bound m on the size of the sought trees, and a coloring function C of vertices in the input graph. We define the hypergraph in our framework as follows:

Definition 3. *A* hypernode $x \in V_H$ *is of the form* (W, v, q)*, where* $W \in N$*,* $v \in V$*,* $q \subseteq \{1, \ldots, m\}$ *and* $C(v) \in q$*.*

A hypernode $x = (W, v, q)$ represents a class of subtrees T_x from G, such that for each $\tau \in T_x$, (1) τ is rooted in v, (2) τ is composed of vertices with distinct colors from q, and (3) τ matches (in terms of both topology and vertex labels) a tree obtained by consecutive derivations of production-rules from P, that start with the non-terminal W.

Definition 4. *A* hyperedge $e \in E_H$ *is of the form* $(z \leftarrow \langle x, y \rangle)$*, such that* $x = (X, u, q')$,$y = (Y, v, q'')$,$z = (Z, u, q' \cup q'')$*, and the following conditions hold:*

1. $Z \rightarrow X@Y \in P$
2. $(u, v) \in E$
3. $q' \cap q'' = \emptyset$

Figure 1 exemplifies the construction of hyperedges incoming to a specific hypernode (x_1), for a given colored vertex-labelled graph $G = (V, E)$ (Figure 1.A), and a given RTG $A = \{N, \Sigma, P, S\}$. Two hyperedges (Figure 1.B, black

arrows) are incoming to hypernode x_1, satisfying all three conditions: (1) $(u, v) \in E$, (2) $W \rightarrow Y@Z \in P$, and (3) the tail hypernodes contain disjoint sets of colors. The hyperedge $(x_1 \leftarrow \langle x_2, x_5 \rangle)$ is not constructed, since both x_2 and x_5 contain the color pink, in contradiction to condition 3.

We call the set of hyperedges, with x as their *head*, the *backward star* of x, and denote it by $BS(x) = \{e \in E_H | x = head(e)\}$. Let $L_H \subseteq V_H$ denote the set of leaf nodes of H. Each hyperedge $e \in E_H$ and each leaf hypernode $\ell \in L_H$, is associated with weights $c(e)$, $c(\ell)$. The function $c : E_H \cup L_H \rightarrow \Re$ could be grammar-based, reflecting a derivation-rule specific scoring, or network-based, reflecting the confidence level of the corresponding edges or vertices in the network, or alternatively some combination of the two strategies. For $e \in E_H$, let $s_c(e)$ denote a monotonic scoring scheme according to c. For each hyperedge $e \in E_H$, $s_c(e)$ computes its score based on the score of the optimal derivation of its tails. The score of the optimal derivation of a hypernode x in an acyclic, directed, binary hypergraph H is denoted $D^*(x)$, defined as:

$$D^*(x) = \begin{cases} c(x) & v \text{ is a leaf} \\ \max_{e \in BS(x)} s_c(e) & otherwise \end{cases}$$

In our applications, s_c is an addition function, defined as follows: $s_c(e) = c(e) + D^*(t_1(e)) + D^*(t_2(e))$.

Based on the above definitions, Stage 1 of our algorithm applies an agglomerative bottom-up construction of the hypergraph, while simultaneously computing $D^*(x)$ for all $x \in H$.

The following theorem gives the time and space complexities for Stage 1 of our algorithm. The proof is given in the Appendix, and is based on some principles from [7,30].

Theorem 2. *The time complexity of Stage 1 of the proposed algorithm for solving Problem 2 is $O(mg|E|3^m)$, and the space complexity is $O(|E_H|) = O(g|E|3^m)$.*

4.2 Stage 2: computing k-best scoring trees

The hypernodes V_H of the hypergraph H, constructed in stage 1, encode all trees that are accepted by the grammar A, together with the subtrees used to construct them. To extract the k-best scoring trees from H, we extend the k-best optimization algorithm, *FindKBest*, developed by Huang and Chiang [19].

For exemplifications of our next definitions, we refer the reader again to Figure 1.B. Let $D(x)$ denote the list of the k-best derivations of a hypernode x (gray arrows) and $D^i(x)$ denote the *i'th* best derivation of x. (For example, in the figure, the leftmost hypernode (x_2) shows two derivations, each one consisting of pink and yellow vertices, while the next hypernode (x_3) has one derivation, consisting of a blue vertex). Each derivation in $D(x)$ is a tuple, $(e, (i, j))$ where $e = x \leftarrow \langle t_1, t_2 \rangle$ and (i, j) is a pair of indices specifying two sub-derivations, $D^i(t_1)$ and $D^j(t_2)$, respectively. Each derivation corresponds to a specific tree in

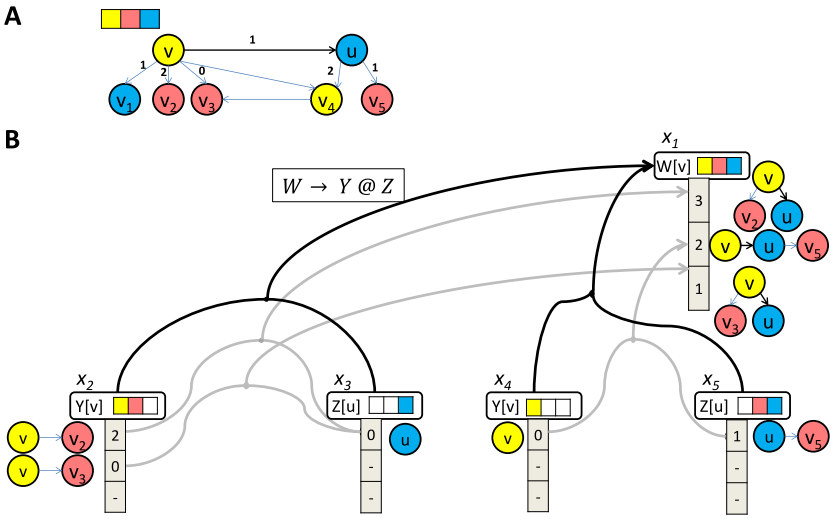

Fig. 1. Example of a step in the construction of the hypergraph H.

T_x. For example, the second derivation of hypernode x_1 in the figure is marked $(x_1 \leftarrow \langle x_2, x_3 \rangle, (2, 1))$.

To compute $D(x)$ for all hypernodes, Algorithm *FindKBest* (see pseudocode in Algorithm 1) is executed for each hypernode $x \in V_H$ in topological order, starting from the leaves and moving up the hypergraph. The algorithm maintains a priority queue $cand[x]$ of potential candidates to be added to $D(x)$, sorted in decreasing score order. Initially, $cand[x]$ is populated with the k top-scoring derivations among all hyperedges in $BS(x)$. This is achieved by calling the procedure *GetCandidates* (line 2 of the pseudocode). If $|D(x)| < k$, then $cand[x]$ is further populated iteratively. In each such iteration, the top derivation $(e, (i, j))$ in $cand[x]$ is removed from the priority queue and added to $D(x)$. Then, its neighboring derivations, $(e, (i, j+1))$ and $(e, (i+1, j))$ are inserted into $cand[x]$. This procedure continues until $|D(x)| = k$ or until all candidate derivations are exhausted.

The correctness of Algorithm *FindKBest* is based on the monotonicity of s_c, by which for any derivation $(e, (i, j))$, the next-best scoring candidate derivation following $(e, (i, j))$ is either $(e, (i, j+1))$ or $(e, (i+1, j))$. This is the essential observation behind the *cube pruning* and *cube growing* approaches [16,19] that is crucial to correctly, yet efficiently, enumerate the k-best derivations.

One obstacle encountered in our application is that the trees in G are unordered while the trees defined by RTGs are ordered. This means that any given subtree τ from G could have a one-to-many mapping to several top-scoring derivations in H. These derivations correspond to ordered trees which are isomorphic under Rooted Unordered Isomorphism. (In Rooted Unordered Isomorphism, trees τ_1 and τ_2 are isomorphic if τ_1 can be obtained from τ_2 by subtree

Algorithm 1: $FindKBest(x, k)$

Input: A hypernode x and an integer k.
Output: Vector $D(x)$ of k-best scoring derivations rooted in x.

```
1  cand[x] ← ∅
2  GetCandidates(x, αrk)
3  D(x) ← ∅
4  B(x) ← ∅
5  while |cand[x]| > 0 and |D(x)| < k do
6      (e, (i, j)) = cand[x].dequeue()
7      if (e, (i, j)) ∉ B(x) then
8          D(x).insert((e, (i, j)))
9          B(x).insert((e, (i, j)))
10     cand[x].insert(e, (i + 1, j))
11     cand[x].insert(e, (i, j + 1))
```

reordering operations). As some or all of these isomorphic derivations could be reported among the k-best derivations, this undermines the correctness of the algorithm as a solution to Problem 2.

To prevent this from happening, we assign to each vertex $v \in G$ a unique integer id, and define a unique canonical encoding for each candidate derivation, representing its corresponding subtree as a parentheses-annotated sequence over lexicographic orderings of its vertex ids. This property ensures that any two isomorphic trees have the same canonical encoding (see Figure S3).

The canonical encoding logic is integrated into Algorithm *FindKBest* (line 7 of the pseudocode), ensuring that, for each hypernode x and for any two isomorphic trees τ_1 and τ_2 that are induced by two distinct derivations and are competing for $D(x)$, only a single, highest-scoring representative is kept. It is implemented as follows. For each hypernode x, we maintain an additional binary-search tree $B(x)$ containing pointers to all elements in $D(x)$, sorted by lexicographic order of their canonical codes. When a new candidate derivation is about to be added to $D(x)$, we first search for its canonical encoding in $B(x)$. If it is found, the new derivation is not added to $D(x)$.

The work per iteration of Algorithm *FindKBest* is computed as follows. Constructing the canonical representation of a subtree can be naively implemented in $O(m)$ time, while operations on $B(x)$ take $O(mlogk)$ time. Moreover, since the candidate subtree is not necessarily added to $D(x)$ (due to the fact that an isomorphic tree could already be in $D(x)$), the number of iterations of the *while* loop in Algorithm *FindKBest* (line 5 of the pseudocode) is no longer limited to k, as was the case in the original algorithm. However, the following observation helps to bound the number of redundant iterations:

Lemma 1. *For hypernode $x = (X, v, q_x)$, the number of distinct incoming derivations that may produce the same tree (up to isomorphism) is bounded by $m \cdot \alpha_X$, where α_X denotes the number of different production rules derived by X.*

This leads to the following theorem which bounds the time and space complexities of Stage 2 of the algorithm.

Theorem 3. *The time complexity of Stage 2 of the proposed algorithm for solving Problem 2 is* $O(|E_H| + |V_H| \cdot km(m \log k + \log \alpha) = O(g \cdot |E| \cdot 3^m + |N| \cdot |V| \cdot 2^m \cdot km(m \log k + \log \alpha))$*, where* $\alpha = \max_{X \in N} \alpha_X$*. The space complexity is* $O(k \cdot |V_H|) = O(k \cdot |N| \cdot |V| \cdot 2^m)$*.*

To obtain the k-best trees which are the final solution to Problem 2, we add to the hypergraph H a pseudo-root hypernode, denoted v^p, such that all hypernodes in H that are marked with the initial symbol S are in the backward star of v^p. After the computation of Stage 2, the k-best list of v^p will contain the k-best trees solving Problem 2.

5 Results

We demonstrate the power of RTG queries in two applications, both of which mine the human protein interaction network for viral infection patterns. The first application exemplifies the expressive power of grammar-based queries. The second application exemplifies the ability to score patterns as part of the grammar.

Our first query aimed to identify temporal responses of human cells to infection by the flu virus Influenza Type A. This is a single-stranded, negative-sense segmented RNA virus that infects hundreds of millions of people and results in numerous hospitalizations and deaths world-wide [6]. The transcriptional response to infection by the virus or its components was measured across time in human primary bronchial epithelial cells [32]. To gain a mechanistic insight into the cellular response to infection, we mapped these data onto a network representing physical interactions between human proteins. We defined a query that searches this network for temporal pathway cascades: Starting with a protein that was up-regulated at the earliest time-point following infection, the cascade proceeds iteratively with one or more proteins that were up-regulated at the same or the subsequent time point, while requiring that the two proteins interact with each other (Figure 2.A). To prioritize cascades that are biologically relevant, we scored instances based on the sum of transcript levels of the proteins in the cascade and the reliability of their interactions. Using RTGnet we identified the top 1,000 up-regulated cascades ($\epsilon = 0.05$) involving at most nine genes, which required 80,248 iterations of the color-coding procedure.

We demonstrate the mechanistic insight that can be gained from this query by discussing the top-scoring up-regulated cascade (Figure 2.A). This top-most cascade delineated important defence steps in the host response to influenza infection, starting from detection of viral particles, inhibition of their production, and ending with cell death. The cascade starts with up-regulation of ISG15, an immune-related ubiquitin-like protein that recruits its interacting partners to fight viral infection [35], and two interferon-induced proteins, IFIT1 and IFIT2, which target viral RNA and the synthesis of viral proteins [1]. It follows with up-regulation of MX1, an interferon-induced GTPase with a known antiviral

activity, and DDX85 (also known as RIG-I), a receptor activating a cascade of antiviral responses, whose conjugation with ISG15 is essential for the initiation of this pathway [3]. Up-regulated at the consecutive time point was TRIM25, forming the DDX58-TRIM25 complex that mediates the DDX58-ISG15 conjugation [14]. Up-regulated at the fourth time point was RBCK1, which regulates the activity of the DDX58-TRIM25 complex [20], and TNF, that together with RBCK1 initiates TNF-mediated gene induction [18]. This is followed by up-regulation of RIPK1, that initiates inflammatory and cell-death cascades upon TNF-receptor signaling [11]. This cascade is especially interesting since several of its proteins interact with the virus: ISG15 and MX1 interact with influenza's NS1 and NP proteins, correspondingly, to inhibit viral replication [17,34], and TRIM25 was found be to be targeted by influenza's NS1 protein to evade recognition of viral RNA by RIG-I [15]. Lastly, the silencing of ISG15, IFIT2, TRIM25 or RBCK1 was found to promote influenza life-cycle and replication [32]. To assess our prioritization scheme we computed the fraction of proteins annotated to the immune system (GO:0002376) [4] across different values of k (Figure 2.B). This fraction increased as we limited k, and was enriched in the top-ranking subtree relative to other subtrees (p=0.0125, Fisher's exact test).

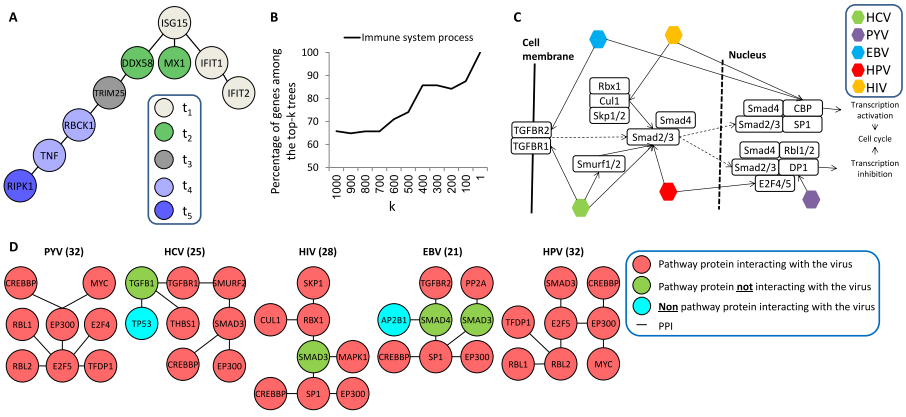

Fig. 2. (A) The maximal-scoring instance of the temporal cascade following influenza infection. Protein colors denote the first time point at which they were up-regulated. (B) The percentage of proteins annotated to 'Immune system process' in the k top-scoring subtrees increases as k is decreased from 1,000 to 1. (C) A schematic view of the TGF-β signaling pathway showing that distinct viruses interact with different proteins in the pathway. (D) Maximal-scoring subtree per virus (and score) in the TGF-β demonstrating the power of an RTG to define various patterns in a single grammar.

Our second query aimed to identify human pathways that a specific virus targets in multiple, related points, as these pathways are likely to be of high importance to its survival and replication. Therefore, we designed a query that accepted subtrees in which several proteins in the pathway interact with each other and with the virus, either directly or via a mediator protein. The accepted subtrees were scored as part of the grammar according to the viral interactions of pathway proteins (see Appendix for details): Proteins with direct interactions scored high (+4), while proteins with indirect interactions scored according to the number of mediator proteins (+1 for each mediator protein). Note that the topology of the subtrees was not predefined. To implement this query we augmented the human protein interaction network described above with data of protein interactions between human and viral proteins for six viruses: Epstein-Barr virus (EBV), hepatitis C virus (HCV), human papillomavirus (HPV), polyoma virus (PYV), human immunodeficiency virus (HIV1), and Influenza A virus subtype H1N1 (H1N1). These viruses were selected because they are evolutionarily remote and their interactions with human proteins were relatively mapped. Associations of human proteins to cellular pathways were extracted from KEGG [21]. As proof-of-concept we focus on the transforming growth factor-β (TGF-β) signaling pathway that regulates cell growth and differentiation, which is crucial for replication of viruses that establish persistent (chronic or latent) infections. Indeed, the only virus without a high-scoring instance was H1N1, a virus that does not establish such persistent infections (Figure 2.C). Top-scoring instances were identified for all other viruses and had distinct topologies, demonstrating the ability of RTGs to define flexible patterns with a single query (Figure 2.D). Results pertaining to other pathways are described in the Appendix for lack of space.

The instances we found could have been identified by other types of network querying approaches, though they would require hard-tailoring of the application to ensure that all specifications are followed. RTGnet on the other hand offers a general framework, in which various such queries can be expressed and scored.

6 Discussion

We introduce a novel framework, RTGnet, that extends the class of grammar-based queries that can be efficiently searched within a network to include all languages defined by RTGs, and produces the list of k-best results. We demonstrated the capabilities of RTGnet in two applications that highlight the generality of the tool and the topological flexibility of the identified instances. The RTG search modelling that we suggest is particularly handful in rich networks with many vertex-labels, enabling the design of more expressive grammars. Another advantage of our framework is the ability of k-best optimization to identify a space of near-optimal solutions, making them suitable for learning stochastic production scores, given a reliable source of training data. Future extensions could include the derivation and usage of more informed scoring schemes, more

complex grammars to express the queries, and the extension of tree-based instances to general sub-networks.

References

1. Abbas, Y.M., Pichlmair, A., Górna, M.W., Superti-Furga, G., Nagar, B.: Structural basis for viral 5 [prime]-ppp-rna recognition by human ifit proteins. Nature (2013)
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. Journal of the ACM (JACM) 42(4), 844–856 (1995)
3. Arimoto, K.I., Konishi, H., Shimotohno, K.: Ubch8 regulates ubiquitin and isg15 conjugation to rig-i. Molecular immunology 45(4), 1078–1084 (2008)
4. Ashburner, M., Ball, C., Blake, J., et al.: Gene ontology: tool for the unification of biology. the gene ontology consortium database resources of the national center for biotechnology information. Nucleic Acids Research 34 (2006)
5. Barabási, A.L., Gulbahce, N., Loscalzo, J.: Network medicine: a network-based approach to human disease. Nature Reviews Genetics 12(1), 56–68 (2011)
6. Belshe, R.B.: Implications of the emergence of a novel h1 influenza virus. New England Journal of Medicine 360(25), 2667–2668 (2009)
7. Bruckner, S., Hüffner, F., Karp, R.M., Shamir, R., Sharan, R.: Topology-free querying of protein interaction networks. Journal of computational biology 17(3), 237–252 (2010)
8. Carme, J., Niehren, J., Tommasi, M.: Querying unranked trees with stepwise tree automata. In: Rewriting techniques and Applications. pp. 105–118. Springer (2004)
9. Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications (2007), release October, 12th 2007
10. Dost, B., Shlomi, T., Gupta, N., Ruppin, E., Bafna, V., Sharan, R.: Qnet: a tool for querying protein interaction networks. Journal of Computational Biology 15(7), 913–925 (2008)
11. Ea, C.K., Deng, L., Xia, Z.P., Pineda, G., Chen, Z.J.: Activation of ikk by tnfα requires site-specific ubiquitination of rip1 and polyubiquitin binding by nemo. Molecular cell 22(2), 245–257 (2006)
12. Fan, W., Li, J., Ma, S., Tang, N., Wu, Y.: Adding regular expressions to graph reachability and pattern queries. In: Data Engineering (ICDE), 2011 IEEE 27th International Conference on. pp. 39–50. IEEE (2011)
13. Finkelstein, A., Roytberg, M.: Computation of biopolymers: a general approach to different problems. BioSystems 30(1), 1–19 (1993)
14. Gack, M.U., Shin, Y.C., Joo, C.H., Urano, T., Liang, C., Sun, L., Takeuchi, O., Akira, S., Chen, Z., Inoue, S., et al.: Trim25 ring-finger e3 ubiquitin ligase is essential for rig-i-mediated antiviral activity. Nature 446(7138), 916–920 (2007)
15. Gack, M.U., Albrecht, R.A., Urano, T., Inn, K.S., Huang, I.C., Carnero, E., Farzan, M., Inoue, S., Jung, J.U., García-Sastre, A.: Influenza a virus ns1 targets the ubiquitin ligase trim25 to evade recognition by the host viral rna sensor rig-i. Cell host & microbe 5(5), 439–449 (2009)
16. Gesmundo, A., Henderson, J.: Faster cube pruning. In: IWSLT. pp. 267–274. Citeseer (2010)
17. Guan, R., Ma, L.C., Leonard, P.G., Amer, B.R., Sridharan, H., Zhao, C., Krug, R.M., Montelione, G.T.: Structural basis for the sequence-specific recognition of human isg15 by the ns1 protein of influenza b virus. Proceedings of the National Academy of Sciences (2011)

18. Haas, T.L., Emmerich, C.H., Gerlach, B., Schmukle, A.C., Cordier, S.M., Rieser, E., Feltham, R., Vince, J., Warnken, U., Wenger, T., et al.: Recruitment of the linear ubiquitin chain assembly complex stabilizes the tnf-r1 signaling complex and is required for tnf-mediated gene induction. Molecular cell 36(5), 831–844 (2009)
19. Huang, L., Chiang, D.: Better k-best parsing. In: Proceedings of the Ninth International Workshop on Parsing Technology. pp. 53–64. Association for Computational Linguistics (2005)
20. Inn, K.S., Gack, M.U., Tokunaga, F., Shi, M., Wong, L.Y., Iwai, K., Jung, J.U.: Linear ubiquitin assembly complex negatively regulates rig-i-and trim25-mediated type i interferon induction. Molecular cell 41(3), 354–365 (2011)
21. Kanehisa, M., Goto, S., Sato, Y., Kawashima, M., Furumichi, M., Tanabe, M.: Data, information, knowledge and principle: back to metabolism in kegg. Nucleic acids research 42(D1), D199–D205 (2014)
22. Koschmieder, A., Leser, U.: Regular path queries on large graphs. In: Scientific and Statistical Database Management. pp. 177–194. Springer (2012)
23. Lacroix, V., Fernandes, C.G., Sagot, M.F.: Motif search in graphs: application to metabolic networks. Computational Biology and Bioinformatics, IEEE/ACM Transactions on 3(4), 360–368 (2006)
24. Lange, M., Leiß, H.: To cnf or not to cnf? an efficient yet presentable version of the cyk algorithm. Informatica Didactica 8, 2008–2010 (2009)
25. Martens, W., Niehren, J.: Minimizing tree automata for unranked trees. In: Database Programming Languages. pp. 232–246. Springer (2005)
26. Mendelzon, A.O., Wood, P.T.: Finding regular simple paths in graph databases. SIAM Journal on Computing 24(6), 1235–1258 (1995)
27. Patro, R., Kingsford, C.: Predicting protein interactions via parsimonious network history inference. Bioinformatics 29(13), i237–i246 (2013)
28. Pinter, R.Y., Rokhlenko, O., Yeger-Lotem, E., Ziv-Ukelson, M.: Alignment of metabolic pathways. Bioinformatics 21(16), 3401–3408 (2005)
29. Ponty, Y., Saule, C.: A combinatorial framework for designing (pseudoknotted) rna algorithms. In: Algorithms in Bioinformatics, pp. 250–269. Springer (2011)
30. Scott, J., Ideker, T., Karp, R.M., Sharan, R.: Efficient algorithms for detecting signaling pathways in protein interaction networks. Journal of Computational Biology 13(2), 133–144 (2006)
31. Sevon, P., Eronen, L.: Subgraph queries by context-free grammars. Journal of Integrative Bioinformatics 5(2), 100 (2008)
32. Shapira, S.D., Gat-Viks, I., Shum, B.O., Dricot, A., de Grace, M.M., Wu, L., Gupta, P.B., Hao, T., Silver, S.J., Root, D.E., et al.: A physical and regulatory map of host-influenza interactions reveals pathways in h1n1 infection. Cell 139(7), 1255–1267 (2009)
33. Shlomi, T., Segal, D., Ruppin, E., Sharan, R.: Qpath: a method for querying pathways in a protein-protein interaction network. BMC bioinformatics 7(1), 199 (2006)
34. Verhelst, J., Parthoens, E., Schepens, B., Fiers, W., Saelens, X.: Interferon-inducible protein mx1 inhibits influenza virus by interfering with functional viral ribonucleoprotein complex assembly. Journal of virology 86(24), 13445–13455 (2012)
35. Zhao, C., Denison, C., Huibregtse, J.M., Gygi, S., Krug, R.M.: Human isg15 conjugation targets both ifn-induced and constitutively expressed proteins functioning in diverse cellular pathways. Proceedings of the National Academy of Sciences of the United States of America 102(29), 10200–10205 (2005)