

Methodology article

Open Access

Algorithms for effective querying of compound graph-based pathway databases

Ugur Dogrusoz*¹, Ahmet Cetintas¹, Emek Demir² and Ozgun Babur¹

Address: ¹Center for Bioinformatics and Computer Engineering Dept., Bilkent University, Ankara, Turkey and ²Computational Biology Center, Memorial Sloan-Kettering Cancer Center, New York, NY, USA

Email: Ugur Dogrusoz* - ugur@cs.bilkent.edu.tr; Ahmet Cetintas - cetintas@gmail.com; Emek Demir - demir@cbio.mskcc.org; Ozgun Babur - ozgun@cs.bilkent.edu.tr

* Corresponding author

Published: 16 November 2009

Received: 2 November 2008

BMC Bioinformatics 2009, **10**:376 doi:10.1186/1471-2105-10-376

Accepted: 16 November 2009

This article is available from: <http://www.biomedcentral.com/1471-2105/10/376>

© 2009 Dogrusoz et al; licensee BioMed Central Ltd.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/2.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Background: Graph-based pathway ontologies and databases are widely used to represent data about cellular processes. This representation makes it possible to programmatically integrate cellular networks and to investigate them using the well-understood concepts of graph theory in order to predict their structural and dynamic properties. An extension of this graph representation, namely hierarchically structured or compound graphs, in which a member of a biological network may recursively contain a sub-network of a somehow logically similar group of biological objects, provides many additional benefits for analysis of biological pathways, including reduction of complexity by decomposition into distinct components or modules. In this regard, it is essential to effectively query such integrated large compound networks to extract the sub-networks of interest with the help of efficient algorithms and software tools.

Results: Towards this goal, we developed a querying framework, along with a number of graph-theoretic algorithms from simple neighborhood queries to shortest paths to feedback loops, that is applicable to all sorts of graph-based pathway databases, from PPIs (protein-protein interactions) to metabolic and signaling pathways. The framework is unique in that it can account for compound or nested structures and ubiquitous entities present in the pathway data. In addition, the queries may be related to each other through "AND" and "OR" operators, and can be recursively organized into a tree, in which the result of one query might be a source and/or target for another, to form more complex queries. The algorithms were implemented within the querying component of a new version of the software tool PATIKAwEB (Pathway Analysis Tool for Integration and Knowledge Acquisition) and have proven useful for answering a number of biologically significant questions for large graph-based pathway databases.

Conclusion: The PATIKA Project Web site is <http://www.patika.org>. PATIKAwEB version 2.1 is available at <http://web.patika.org>.

Background

Especially with the help of novel large-scale analysis methods, a massive amount of data is now being gathered

on cellular processes [1-3]. Unfortunately, most of these data are fragmented and incomplete. One of the biggest challenges of bioinformatics today is to represent and

integrate this type of knowledge effectively to construct a knowledge base that can act as a blueprint for simulations and other analysis methods, enabling us to better understand and predict the behavior of a cell [4].

Even though traditional way of representing cellular pathways with still images often yields very pretty pictures, such drawings are mostly not reusable. In addition, underlying ontology and notation are often far from being uniform or consistent, mostly dependent on implicit conventions rather than explicit, formal rules [4]. Recently, these problems have resulted in a major shift towards the use of more formal ontologies and to the dynamic representation of pathways that support programmatic integration and manipulation of pathways, regardless of the underlying ontology. Among these, graphs, one of the most common discrete mathematical structures [5], have been most popular for "in-silico" modeling of biological pathways, from metabolic pathways to gene regulatory networks to signaling pathways [4,6-8]. Such modeling is crucial for the field of systems biology, which deals with a systems-level understanding of biological networks. Three levels of increasing complexity are listed in [6] for the analysis of cellular networks, where network topology (global structural properties), interaction patterns (local structural connectivity), and network decomposition (hierarchical functional organization) are addressed at each level, respectively. Representing such complex networks as graphs makes it possible to investigate the topology and function of these networks using the well-understood concepts of graph theory and to predict their structural properties or to detect special structures or properties in them. In addition, this representation has made the systematic (i.e., programmatic) integration of these complex networks feasible. A comprehensive survey of such prediction, detection, and reconstruction methods can be found in [6].

Lately, an extension of this graph representation, namely *hierarchically structured graphs* or simply *compound graphs*, has become popular, in which a node of a biological network may recursively contain or include a sub-network of a somehow logically similar group of biological objects [4,9]. This extension provides many benefits for the analysis of biological pathways; most importantly, it reduces the complexity of large networks by decomposing them into distinct components or modules.

There has been much work on querying for the occurrences of sub-structures (from specified subgraphs to special sub-structures), such as graphlets or motifs in graph-based data, including pathways [6]. Most approaches employ a graph-matching algorithm to find one or all (exact or inexact) instances of the specified subgraph [10-

12]. Others take a comparative approach to interpreting molecular networks, contrasting and aligning networks of different species and molecular types under varying conditions [13]. Graph algorithms, such as the shortest paths between a specified pair of objects in a graph database, have been in use for quite a while [14], whereas their use in graph-based pathway databases has only recently become popular [15-17].

Here, we present a framework for querying a compound graph-based pathway database, as well as a number of graph-based queries and algorithms that are needed to implement these queries. We assume a database in which pathways are stored in an integrated manner, as opposed to a list of independent pathways. A query to the database is performed over this integrated, higher-level network of pathways, which aims to find a sub-network of interest (Figure 1). Such a query might require a rich set of graph algorithms. This framework and associated graph algorithms were implemented as part of a new version of the bioinformatics tool *PATIKAWeb* [18]. The range of graph-theoretic queries described as part of this framework is among the most comprehensive to be built so far and, in our best knowledge, is the first querying framework that accounts for *compound* structures (i.e., the grouping or abstractions of biological objects to an arbitrary level of depth) in a graph-based knowledge base.

Definitions

Let $G = (V, E)$ be a graph with a non-empty node set V and an edge set E . An edge, $e = \{x, y\}$ or simply xy , joining



Figure 1
How pathways are integrated. Conceptual illustration of how pathways are integrated in a knowledge base (each pathway is colored distinctly), which is typically on disk, and how a sub-network of interest (parts of three different original pathways) may be extracted and displayed as a result of a query.

nodes x and y is said to be *incident with* both x and y . Node x is called a *neighbor* of y and vice versa. A pathway graph $G = (V, E)$ is a graph, where some of the edges in E are marked as *inhibition* edges (e.g., an interaction that disables or impedes the target reaction node via the source state node).

A path between two nodes n_0 and n_k is a non-empty graph $P = (V', E')$ with $V' = \{n_0, n_1, \dots, n_k\}$ and $E' = \{n_0n_1, n_1n_2, \dots, n_{k-1}n_k\}$, where n_i are all distinct. n_0 and n_k are called the end points of path $P = n_0n_1 \dots n_k$, whose *length*, denoted by $|P|$ is the number of edges on it. A path is said to be *directed* if all its ordered edges are directed in the same direction. A directed path P is called an *incoming* (*outgoing*) path of node n if P ends at *target* (starts at *source*) node n . A directed path is called *positive* (*negative*) if it contains an *even* (*odd*) number of inhibitors (i.e., inhibition edges).

Given node sets A and B , an A - B path is a path with its ends in A and B , respectively, and no node of P other than its ends is from either set A or B . An A -path is a path where one of its end nodes is in A , and no other nodes and interactions are from A .

A path C with identical end nodes is called a cycle. A directed cycle is called *positive feedback* (*negative feedback*) if it contains an *even* (*odd*) number of inhibitors.

The *distance* $d_G(x, y)$, between two nodes x and y in graph G , is the length of a shortest x - y path in G . If $G' = (V', E')$ is a subgraph of $G = (V, E)$, and G' contains all the edges $xy \in E$ with $x, y \in V'$, then G' is a *vertex-induced* or simply *induced* subgraph of G ; we say that V' induces G' in G and write $G' = G[V']$. If node x is the starting node of a directed path that ends up at node y , then node y is said to be in the *downstream* of node x ; similarly, node x is said to be in the *upstream* of node y . A node y in the downstream of a node x is a *potential target* of x ; similarly, x is a *potential regulator* of y . A *compound graph* $CG = (G, I)$ is a 2-tuple of a graph G and a rooted directed *inclusion tree* $I \subseteq V \times V$, which defines nesting relations for V by partitioning it into *base nodes* (i.e., leaves of the inclusion tree) and *compound nodes* (Figure 2). Notice that we assume that edges of G cannot join two vertices, one of which is an ancestor of the other one in the inclusion tree.

Ontology

The query framework and the algorithms described in the paper were all designed and implemented assuming the PATIKA ontology [4], which shows the utmost similarity to standard representations, such as BioPAX [19] and SBN [20]. However, the results should be applicable to other graph-based pathway representations without difficulty.

The PATIKA ontology is based on a two-level qualitative model. At the entity level, interactions and relations can

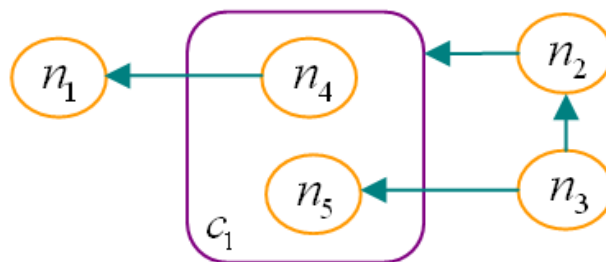


Figure 2

Example compound graph. An example compound graph $(V(G) = \{n_1, n_2, n_3, n_4, n_5, c_1\}, E(G) = \{n_2c_1, n_3n_2, n_3n_5, n_4n_1\}, I = \{c_1n_4, c_1n_5\})$. The order, in which nodes are traversed, depends on the relationship between a compound structure and its members and on the relationship between members of a compound structure.

be addressed in an abstract manner, where the exact state or details of the involved entities is unknown, such as with protein-protein interactions, inferred relations, and literature-derived information. At the state/transition or mechanistic level, each entity is associated with a set of states that interact with each other via transitions. This level can capture more detailed information, such as compartments, molecular complexes, and different types of biological events (e.g., covalent modification, transportation, and association). This two-level representation elegantly covers most biological pathway-related phenomena, and is capable of integrating information present in the literature and in molecular biology databases. Additionally, PATIKA uses the notion of compound graphs to represent abstractions, which are logical groupings that may be used to handle the complex and incomplete nature of the data. Figure 3 shows example pathways drawn at biological entity and state/transition levels.

More formally, a *compound pathway graph* $CP = (G, I)$ is a 2-tuple of a *pathway graph* G and a directed acyclic *inclusion graph*. Notice that this is different than traditional compound graphs as we do not require compound structures in our pathway models to form a tree. I , where

- $V(G)$: union of nodes denoting bioentities, states, transitions, molecular complexes, and abstractions of five distinct types: regular, incomplete state, incomplete transition, homology state, and homology transition;
- $E(G)$: union of interaction edges of various types (such as PPI edges at bioentity level and activator edges between a state and a transition), some of which are directed and/or inhibitory;
- $V(I) = V(G)$;

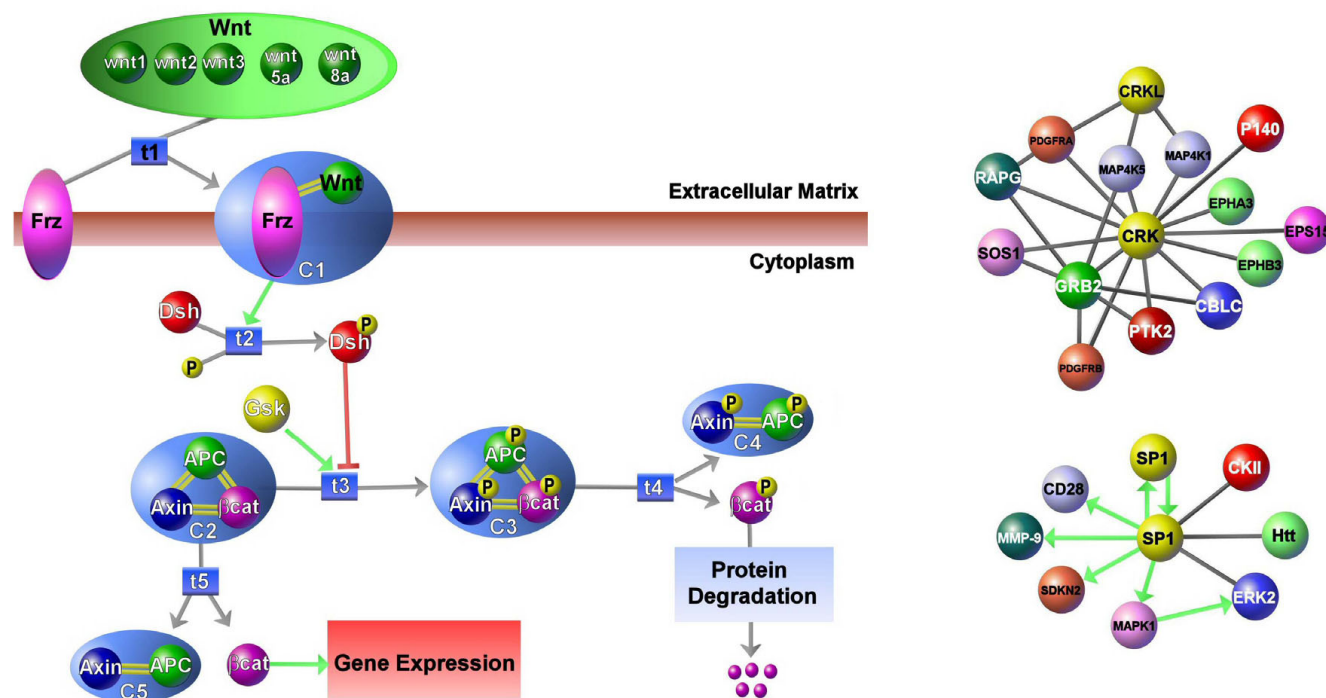


Figure 3
Sample pathways represented by PATIKA ontology. (left) Canonical wnt pathway containing examples of compound structures, such as regular abstractions (e.g., "protein degradation"), homology abstractions (e.g., 5 wnt genes), and molecular complexes (e.g., APC:Axin) [4]. **(right)** Partial human interaction networks containing PPIs and the transcriptional regulation interactions of proteins CRK and SPI, respectively.

- $E(I)$: union of inclusion edges for defining compound structures (molecular complexes and abstractions).

In order for a compound pathway graph $CP = (G, I)$ to comply with the PATIKA ontology, it needs to satisfy certain additional invariants; for instance, regular abstractions cannot have a direct interaction (edge).

Results

PATIKAwEB [18] is a pathway analysis tool with a distributed architecture, where the server is composed of a database component and an application server. The implementation uses the JSP (JavaServer Pages) edition of the Tom Sawyer Visualization technology [21] to handle highly-dynamic and advanced visual content along with Java™, JavaScript™ and DHTML/HTML.

The query component of PATIKAwEB was implemented as a Java applet, embracing all algorithms discussed in this paper. See Additional file 1 for a sample querying scenario comprised of a number of these operations.

Query interface of PATIKAwEB

For utmost flexibility, the queries are allowed to be recursively organized into a tree in PATIKAwEB, in which the

result of one query might be a source and/or target for another. Queries may be related to each other through "AND" and "OR" operators (Figure 4). A query may be executed not only on the database but also on the current local pathway model. When a query initiated from the query dialog finishes, the resulting pathway model is summarized by a Query Result Dialog. The model can be viewed in either or both of bioentity and mechanistic levels. Figure 5 shows a sample query result.

Experiments

We performed a number of experiments to test our algorithms using the implementation within PATIKAwEB. The tests were performed on an ordinary personal computer using a randomly created integrated pathway knowledge base, consisting of about 20,000 pathway nodes and 30,000 edges. The knowledge base was held in memory. However, with the success of current high performance object/relational persistence and query services, the slowdown should not be dramatic when it is on disk. Our experiments revealed that our theoretical analysis is in line with the results of our implementation. Below, we provide the details.

One set of experiments computes the graph of interest $GoI(S, k)$. The time complexity of this algorithm is pro-

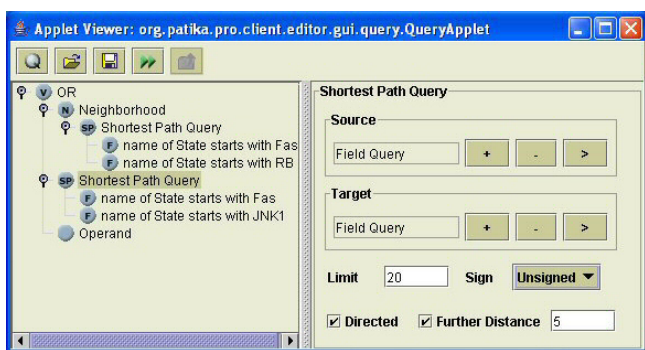


Figure 4
Sample query. Sample query tree to find the **union** of 1-neighborhood of the objects on the shortest path from states whose name starts with "Fas" to states whose name starts with "RB" **with** the shortest path from states whose name starts with "Fas" to states whose name starts with "JNK1" [18].

portional to the number of nodes and edges in the *k*-neighborhood of nodes of interest, as Figure 6 indicates.

Experiments in this set also measure *k* versus the execution time with random source sets of various sizes. As Figure 6 illustrates, independent of the source set size,

execution time increases rapidly (but not exponentially) up to a certain distance (between 12 and 15), after which it remains constant. One might expect an exponential increase in the number of nodes reached (and thus, the execution time) as the distance increases. However, in practice, up to all of the neighbors of a newly visited node might already have been visited, avoiding a combinatorial increase [22]. After a certain number of steps, all nodes are expected to have been visited.

Another set of experiments we performed was to compute the shortest paths $SP(S, T, k, d)$. Remember that the time complexity of this algorithm is $O(l + |NB(S, k)|)$, where *l* is the total length of the paths enumerated. Here, *l* depends on the choice of *S* and *T*, and can be exponential in the size of the graph, in the worst case. However, our experiments show that, on average, that is not the case, and *l* is dominated by the second term. The reason for this is that, roughly speaking, if the shortest length between sets is "too long", the number of paths found is small. Similarly, if the number of paths found is large, the shortest length between sets is short. Obviously, *l* equals the number of paths found multiplied by the shortest path length between sets. Therefore, there is a tradeoff between the number of paths and the shortest length between sets. This hides the exponential behavior expected in the worst

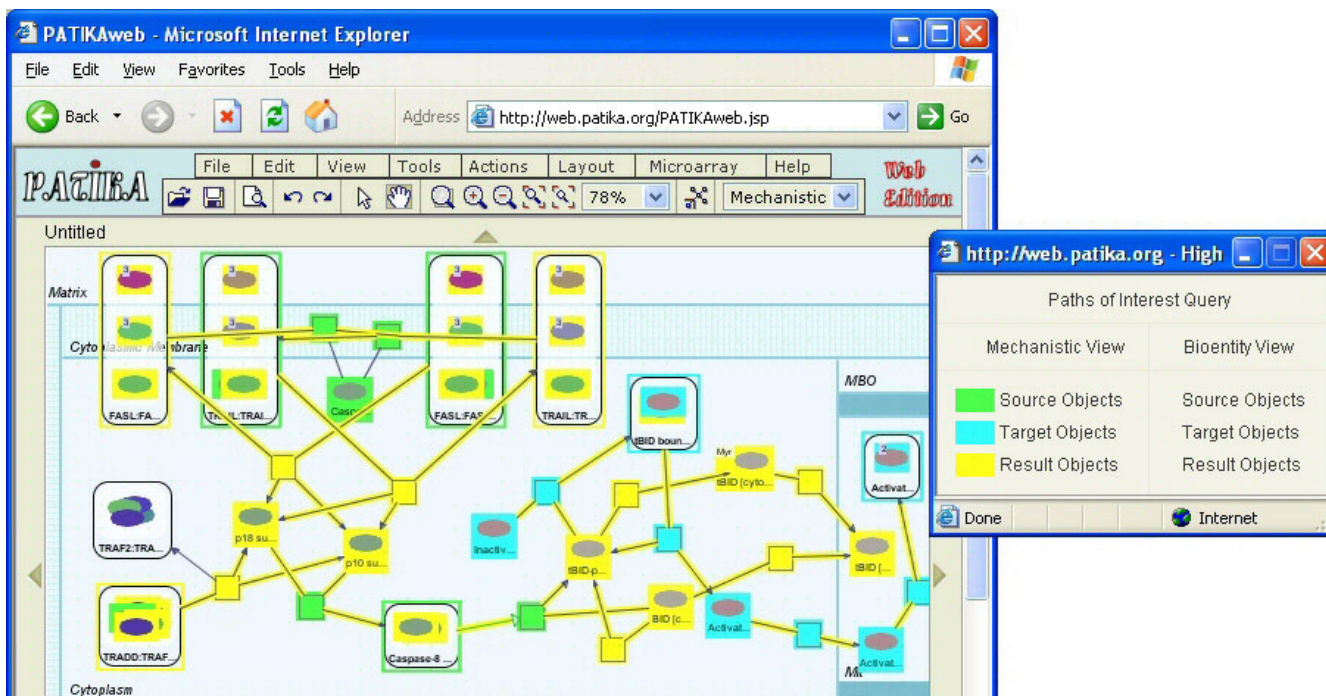


Figure 5
Sample query result. Mechanistic view of the result of the following sample query: paths-of-interest (yellow) with source of all mechanistic nodes whose names contain "caspase-8" (green) and target for those whose names contain "bax" (cyan) with limit 8; Highlight Legend Dialog for this query is shown on the right.

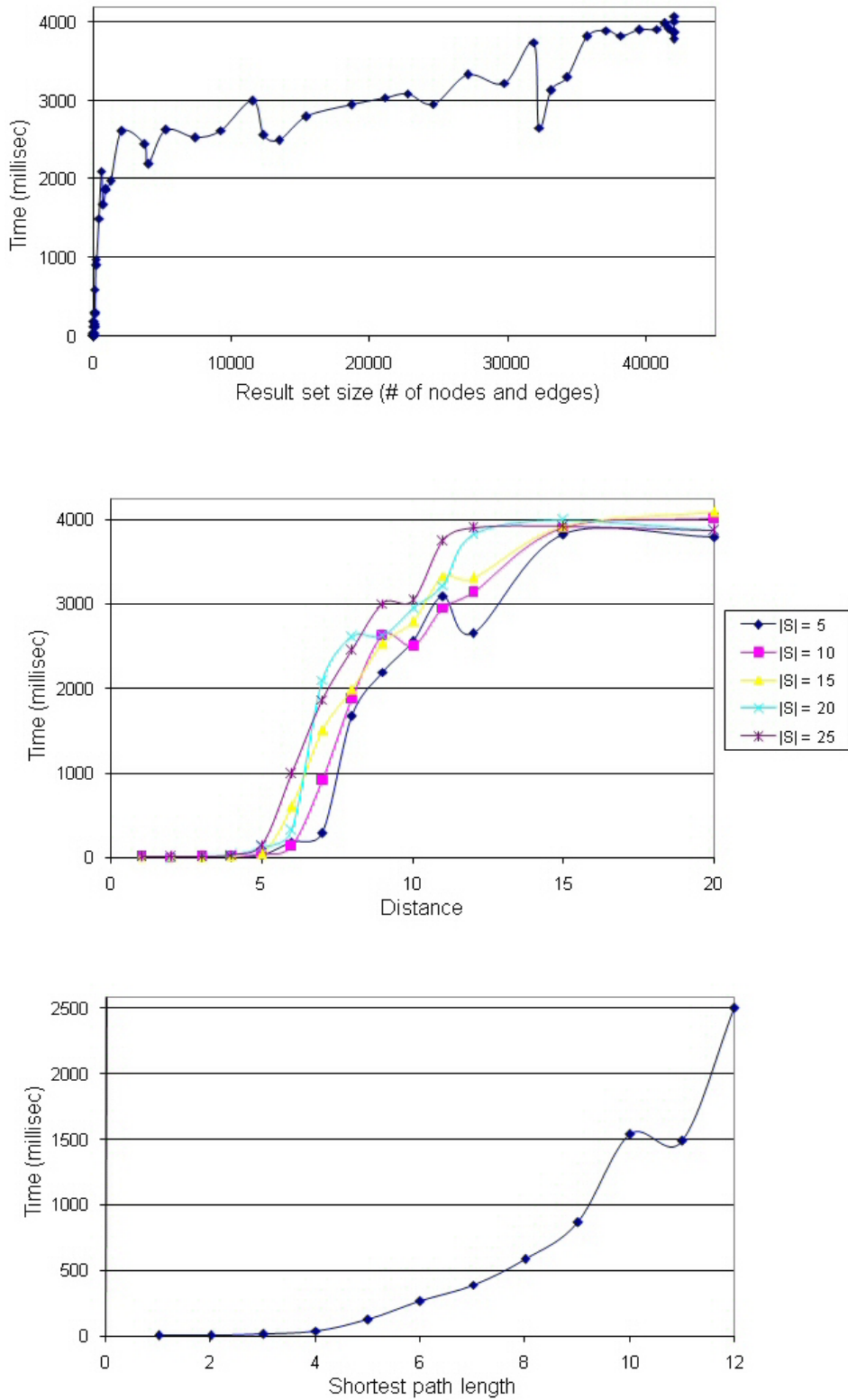


Figure 6
Query parameters vs. execution time. (top) Result set size ($|NB(S, k)|$) vs. execution time for Gol algorithm. **(middle)** Distance (k) vs. execution time for Gol algorithm. **(bottom)** Shortest length vs. execution time of shortest-path algorithm.

case. Figure 6 plots the shortest path length versus the execution time for the shortest path algorithm, where node sets S and T of varying sizes (chosen from set $\{1, 3, 5, 10, 15, 20, 25, 50\}$) were picked randomly. The details of this analysis may be found in Additional file 2.

Discussion

Imagine a researcher who observes a certain gene's expression level oscillating in a manner that can be best described with a feedback loop. Figuring out which molecular interactions caused this behavior might require a review of many research articles and the integration of information across various sources. A review article or a pathway diagram can be immensely helpful, but only if it completely contains the *path* in question. In the case of a single gene, this analysis is time-consuming and painstaking. When using high-throughput data, for which one needs to consider multiple entities in a much broader scope, it becomes virtually impossible.

To face this challenge, hundreds of pathway databases were developed [23], and efforts to form an integrated map of cellular events are underway [24]. However, we also need to be able to automatically construct "pathways" by extracting relevant portions of the underlying network, based on the biological question we are asking. Compared to static pathway diagrams, such "dynamic" queries can provide a more complete picture, and can help manage complexity by removing portions that are not of interest. The algorithms and framework presented here provide such facilities for an elementary set of biological questions.

One can extend this basic set by adding algorithms that perform advanced queries, such as flux balance analysis [25], isomorphic graph matching [6], and differential expression analysis [26]. To complement advances in pathway databases and tools, we expect algorithms such as these to become commonplace in biological research. Towards this goal, we hope to make this framework and basic set of operations available as a public library, ready to accept pathway models in standard formats such as BioPAX.

Methods

We first explain how certain special pathway structures, namely compound structures and ubiquitous entities, are to be handled. Then, we define a number of graph-theoretic problems along with associated algorithms.

Compound structures

All the query operations described later, make use of traversals over a pathway knowledge base to compute the desired sub-graph of interest. The traversal over pathway objects represented by some sort of a compound structure

(e.g., regular abstractions or molecular complexes as defined by the PATIKA ontology) calls for a special mechanism. Take as an example a breadth-first traversal, starting with node n_3 over the compound graph in Figure 2. When the traversal reaches the compound node c_1 , should it also visit its member nodes n_4 and n_5 ? Or if it reaches its member n_5 , should it also visit its sibling n_4 , and continue its downstream towards n_1 ?

The answer depends on the context. If an underlying "equivalence relation" exists between a compound node and its members, a traversal reaching a node in an equivalence set should also reach and visit other nodes in this set. For instance, when a traversal reaches a gene that is a member of a homology abstraction (i.e., it is inside a compound structure along with its homologous genes), it should also be considered to have reached its siblings (i.e., its homologous genes). On the other hand, reaching a member of a regular abstraction should rarely be interpreted as reaching all the other members of that abstraction. Thus, it is best to let the user decide how the traversals over compound structures should be configured for each type of such structures.

This problem may be addressed by using traversal options that define how the traversal continues upon reaching a compound structure or a member of the compound structure. Note, however, that we need two flags per type of compound structure, as they might be set to be different:

- *Link a compound structure and its members*: For instance, should reaching a homology state be interpreted as also reaching its members (i.e., the genes that are homologous) and vice versa?
- *Link members of a compound structure*: For instance, should reaching a member of a molecular complex be interpreted as reaching all members of this complex (thus the traversal should be able to continue from other members as in Figure 7)?

For the ontology that we assume, there are six distinct types of compound structures: five types of abstractions (homology states and transitions, incomplete states and transitions, and regular abstractions) and molecular complexes. Once such a set of options is defined, the modification needed in a query algorithm to support compound structures is rather straightforward. When a compound node or a member of a compound is visited during a traversal, the algorithm forms a set of "equivalent" nodes, and continues the traversal from these equivalent nodes as well as from the visited node. In other words, the algorithm acts as if it's not only this node that is a neighbor of the previously visited node but also its equivalent nodes.

**Figure 7**

Example traversals of complexes and ubiquitous molecules. (left) The traversal reaching complex "c1" from the left transition will continue to the right transition only if the "Link Members of Complex" option is true. **(right)** Whether or not protein states "a" and "b" are in the 4-neighborhood (yellow) of state "c" (green) depends on whether traversal over ubiquitous molecules ("ubique X") is allowed. In this case, it was allowed.

More formally, node x is called *equivalent for traversal* with node y if and only if

- x and y are members of the same compound node, and the user defined flag for linking members of this type of compound structure is true, or
- one is a compound and parent node (possibly an indirect parent through multiple levels of nesting) of the other, and the user defined flag for linking this type of compound structure and its members is true.

Notice that equivalence for traversal is a binary relationship that is *not* transitive.

Thus, node x is called a *compound neighbor* of y if and only if $\{x, y\} \in E$ or y is equivalent for traversal with some node z with $\{x, z\} \in E$. Furthermore, a *compound path* between two nodes n_0 and n_k is a non-empty graph $P = (V', E')$, where

$$V' = \{n_0, n_1, \dots, n_k\}, E' = \{n_0n_1, n_1n_2, \dots, n_{k-1}n_k\} \supseteq E,$$

$n_i n_{i+1} \in E$ or node n_i is equivalent for traversal with n_{i+1} for $i = 0, \dots, k-1$, and

$$n_i n_{i+1} \notin E \Rightarrow n_{i+1} n_{i+2} \in E \text{ for } i = 0, \dots, k-2.$$

Edges outside E do not count towards the length of compound paths.

For instance, a breadth-first traversal over a compound graph can be sketched as follows:

algorithm COMPOUNDBFS(*root*)

```

1 Q := {root}
2 while Q ≠ ∅ do
3   n1 := Q.DEQUEUE()
4   N := n1.COMPOUNDNEIGHBORS()
```

```

5 for n2 ∈ N do
6   if n2.VISITED ≠ true then
7     Q.ENQUEUE(n2)
8 //process n1
9 n1.VISITED := true
```

The only difference from the regular breadth-first traversal, here, is line 4. The neighbors of the currently processed node n_1 are identified not only based on the downstream nodes of n_1 , but also on the nodes equivalent to these downstream nodes with respect to compound structures. This simple difference makes it easier to modify existing graph-theoretic algorithms to add support for compound structures.

Ubiquitous entities

Another type of biological object that requires special attention is *ubiquitous* molecules, which typically participate in many different biological activities, have relatively constant concentration and do not transmit a signal. For instance, a ubiquitous molecule such as ATP might be involved in hundreds if not thousands of reactions at the mechanistic level. Thus, one might prefer not to link two reactions whose only common actors are these kinds of molecules. Therefore, traversal over ubiquitous molecules should also be able to be controlled by customized options. Figure 7 explains this with an example. Modification needed in graph algorithms to handle ubiquitous molecules is rather straightforward.

Algorithms

A careful requirements analysis has yielded the following graph-theoretic problems and associated algorithms that might be useful in querying cellular pathway databases. This list is by no means exhaustive and can easily be

extended. Unless otherwise stated, all algorithms have both directed and undirected versions.

Neighborhood of entities

A simple yet powerful operation for exploring pathways is finding the neighbors of a specified source entity or node within a certain distance (Figure 8). Most pathway visualization tools [18,27,28] provide a way to expand or highlight the neighborhood of selected nodes. Pathway databases [24,29] generate on the fly views based on neighborhoods. Formally, *k*-neighborhood of a node set *S* can be defined as

$$NB(S, k) = S \cup \{x \mid x \text{ is on a compound } S \text{ - path } P \wedge |P| \leq k\} \cup \{e \mid e \text{ is an edge on a compound } S \text{ - path } P \wedge |P| \leq k\}.$$

Upstream or downstream of entities

k upstream (downstream) of an entity *a* is composed of the entities on the incoming (outgoing) compound paths to *a* with length at most *k*. The positive (negative) upstream of

an entity *a* is composed of the entities on the incoming compound path that activates (inhibits) (in the case of a mechanistic pathway, the preceding transition of) entity *a*. For instance, *k* positive upstream of a node *a* can be defined formally as

$$ST\text{-up}^+(a, k) = \{x \mid x \text{ is on a positive incoming compound path } P \text{ of } a \wedge |P| \leq k\}.$$

A node *b* might be in both the positive and negative up or downstream of another node *a*, making the effects of those streams (or associated positive and negative compound paths) ambiguous. Those nodes in the upstream (downstream) of a node *a* that lead to (are reached from) node *a* with *only* positive compound paths form the *unambiguous* positive upstream (downstream) of node *a* (Figure 9).

algorithm STREAM(*v*, *length*, *maxLength*, *sign*, *desired-Sign*, *dir*)

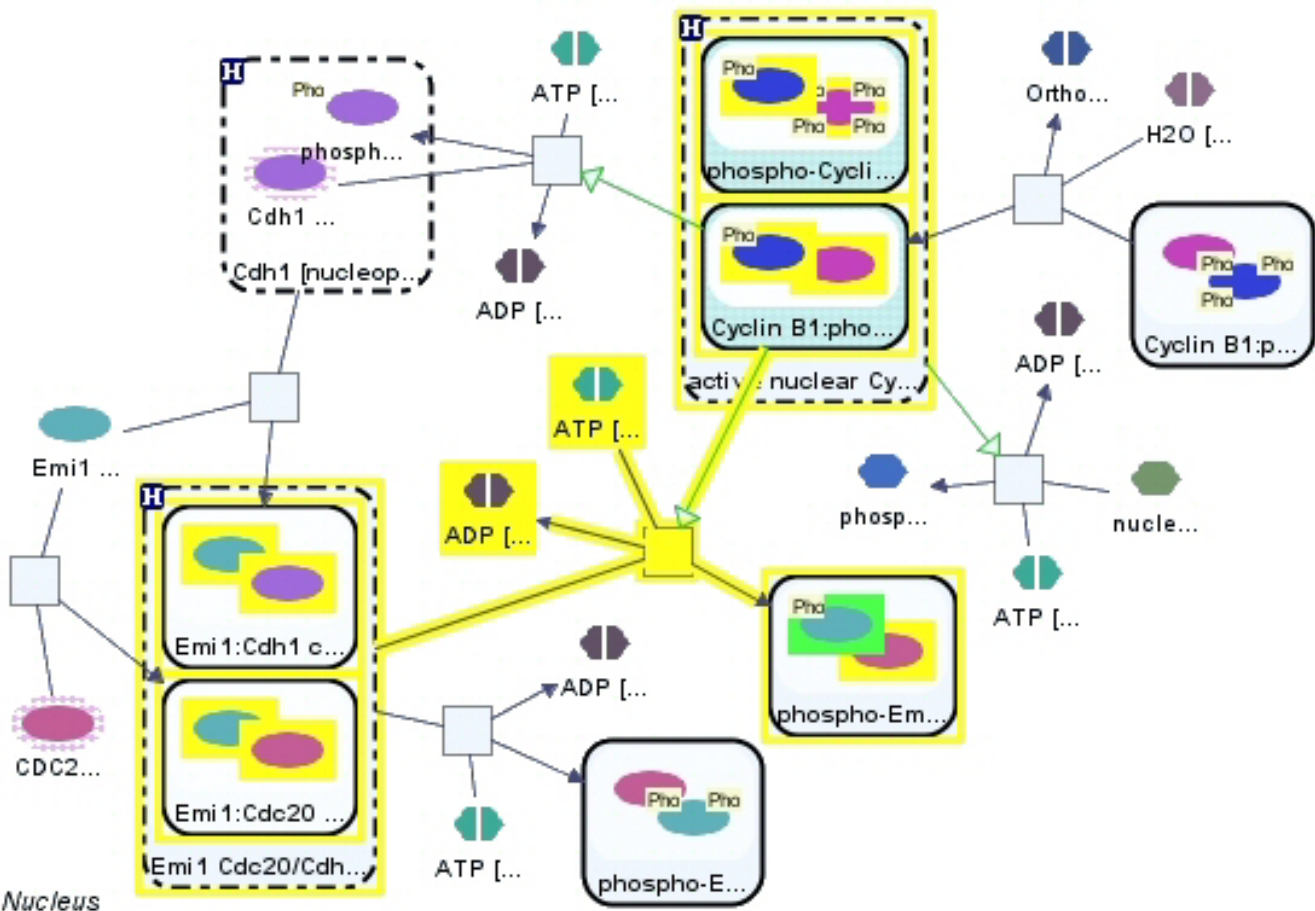


Figure 8 Neighborhood. 2-neighborhood (yellow) of phosphorylated Emi I (Ser I82) (green) in a partial pathway in nucleus. Compound nodes with dashed borders represent homologies, whereas compound nodes with solid borders represent molecular complexes.

```

1  v.AVAILABLE := false
2  if length < maxLength then
3    for w ∈ v.COMPOUNDNEIGHBORS(dir) do
4      if w is an inhibitor then sign := -sign //change sign
of running path
5      if sign = desiredSign then R := R ∪ {w}
6      else A := A ∪ {w}
7      if w.AVAILABLE = true then //prevents infinite loop
8        STREAM(v, length + 1, maxLength, sign, desiredSign,
dir)
9  v.AVAILABLE:= true
10 return R, A

```

The algorithm performs a brute-force traversing of all the nodes in the k -neighborhood of the source node. It is based on a depth-first search. However, after the recursive processing of a node finishes, that node is marked as "unvisited" again, potentially leading to multiple visits of nodes and edges. More specifically, every node and edge is processed as many times as the number of distinct ways they can be reached from the source node. In other words, every possible compound path with a length limit from the source node is examined to determine if it makes a valid stream.

Naturally, the worst-case time complexity of this algorithm is exponential in the size of the k -neighborhood of

the source node. Experiments show, however, that the execution time should be acceptable for most interactive applications for small values of k (for instance, up to 10).

Common targets or regulators

There are already a number of algorithms for inferring highly connected or co-regulated subnetworks of cellular interactions and processes often called modules or pathways [30-32]. When analyzing these modules, we often want to know if there is a process or gene that is upstream of the genes in the module, which can provide a causal explanation for the co-regulation, and ultimately a way to control the module. Similarly, two pathways affecting the same mechanism in the cell is interesting since it suggests that a specific phenotype can have more than one molecular cause. For instance, Engelman et al. [33] discuss that drug resistance in lung cancer is related to an alternative pathway that leads to PI3K activation. Searching for common targets of signaling proteins can help to develop alternative treatment strategies.

Common downstream (upstream) of a source entity set S is the set of potential common target (regulator) entities that are in the downstream (upstream) of *all* entities in S . More formally, the common downstream, $CD(S, k)$, of a source node set S with path length limit k is defined as

$$CD(S, k) = \{x \mid \forall a \in S (\exists P \text{ is a compound path from } a \text{ to } x \wedge |P| \leq k)\}.$$

The common upstream $CU(S, k)$ of a set S can be defined similarly. Figure 10 shows an example of this operation. Below is the pseudocode of this algorithm. The input parameter *dir* specifies whether we are asking for potential targets or regulators, requiring a forward or reverse BFS, respectively. Algorithm CU-COMPOUNDBFS simply increases the *reached* count of nodes in the k -neighborhood of seed node n_1 . The nodes reached during such

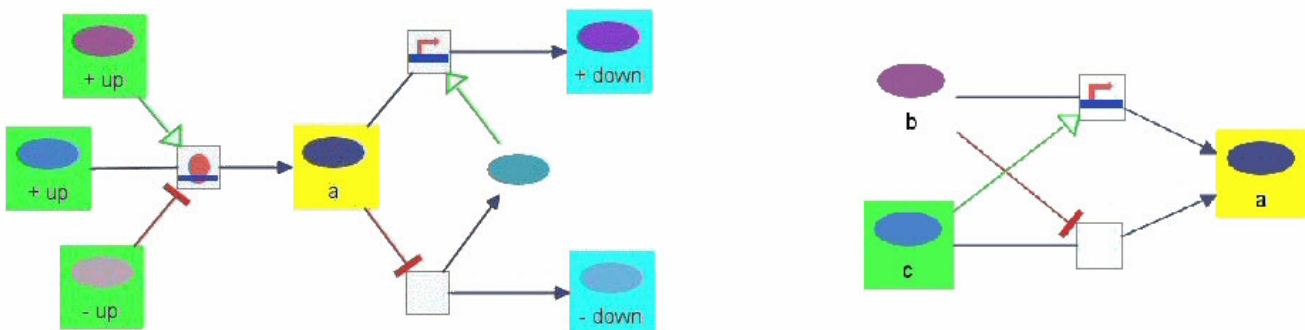


Figure 9

Upstream. (left) Up (green) and down (cyan) stream of protein "a" (yellow) in a partial mechanistic pathway. (right) Unambiguous positive upstream of node "a" (yellow) contains node "c" (green) only, as node "b" is on both positive and negative paths leading to node "a".

searches are combined in a candidate set. Only the nodes in the candidate set that are reached from *all* source nodes are selected to form a result set.

algorithm COMMONUPSTREAM(S, k, dir)

- 1 $C := R := \emptyset$ //candidate and result sets, respectively
- 2 **for** $n_1 \in S$ **do**
- 3 $C := C \cup \text{CU-COMPOUNDBFS}(n_1, k, dir)$
- 4 **for** $n_2 \in C$ **do**
- 5 **if** $n_2.\text{LABEL}(\text{reached}) = |S|$ **then** $R := R \cup \{n_2\}$
- 6 **return** R

This operation takes $O(|S| \cdot |\text{NB}(S, k)|)$ time, as a BFS is performed for each node in S .

In addition, one might require such paths leading to potential targets or originating from potential regulators to be positive or negative. For instance, the common downstream of source nodes S reached by positive compound paths of length up to k only, denoted by $\text{CD}^+(S, k)$, might be of interest. However, we conjecture that the complexity of such an operation is asymptotically higher.

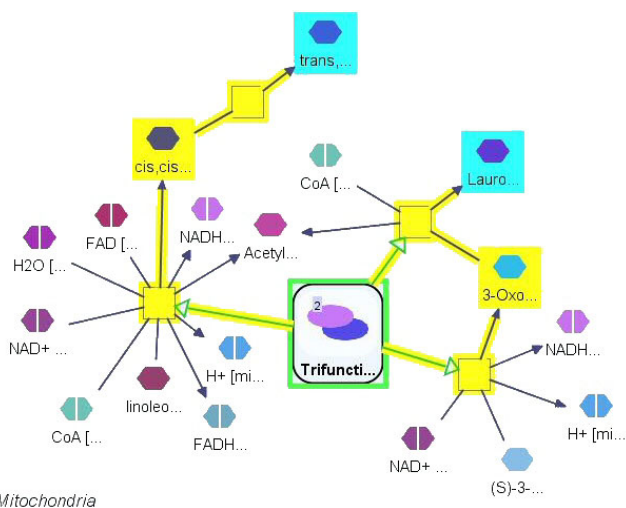


Figure 10

Common regulator. The common upstream, with a path limit of 2, of small molecules containing the word "lauro" in their name (cyan) in this partial mechanistic pathway turns out to be a single node representing a molecular complex (green). The paths from the potential common regulator to the target nodes are highlighted (yellow).

Network of interest

A problem that arises frequently in high-throughput studies is gene/protein selection. For example, new high-throughput sequencing technologies enabled scanning for mutations in a large number of samples. With the current technology, the feasible number of genes that can be sequenced is in the order of tens to hundreds [34]. One way to select new genes for sequencing is to search within the vicinity of the genes that are already implied in that cancer. Particularly, genes that connect one or more of these usual suspects via a signaling path are more likely to be critical for the disease.

Given a graph G and a set of entities of interest S (e.g., genes of interest), network or graph of interest finds in G all compound paths of length at most k between any two entities of a specified entity set. The subgraph of G induced by the nodes of the resulting set gives the graph of interest. More formally,

$\text{GoI}(S, k) = G[B]$, and $B = \{x \mid x \text{ is on a compound } S-S \text{ path } P \wedge |P| \leq k\}$. As the name suggests, this operation is aimed at finding a "maximal" subgraph comprising all the nodes of interest complemented by the "missing links" among these nodes; the parameter k defines how long the paths, which link nodes of interest to form a graph of interest, are allowed to be. Figure 11 explains this operation with an example. Below is the pseudocode for this operation. Here, two separate BFS are to be run in forward and reverse directions, and combined to form a candidate set. The nodes in this candidate set satisfying the maximum path length constraint are put in a result set, which is "purified" by a post-processing phase, during which degree 1 nodes that do not lie on paths between source set nodes (effectively, subgraphs that are trees in the result, coinciding with the source set only at their roots) are pruned iteratively.

The following algorithm can handle directed compound pathways. It could, however, be simplified for undirected pathways by removing a redundant call to GOI-COMPOUNDBFS .

algorithm GRAPHOFINTEREST(S, k)

- 1 $C := \text{GOI-COMPOUNDBFS}(S, k, fwd) \cup \text{GOI-COMPOUNDBFS}(S, k, rev)$
- 2 **for** $q \in C$ **do**
- 3 **if** $q.\text{LABEL}(fwd) + q.\text{LABEL}(rev) \leq k$ **then** $R := R \cup \{q\}$
- 4 $R := \text{PURIFY}(S, R)$

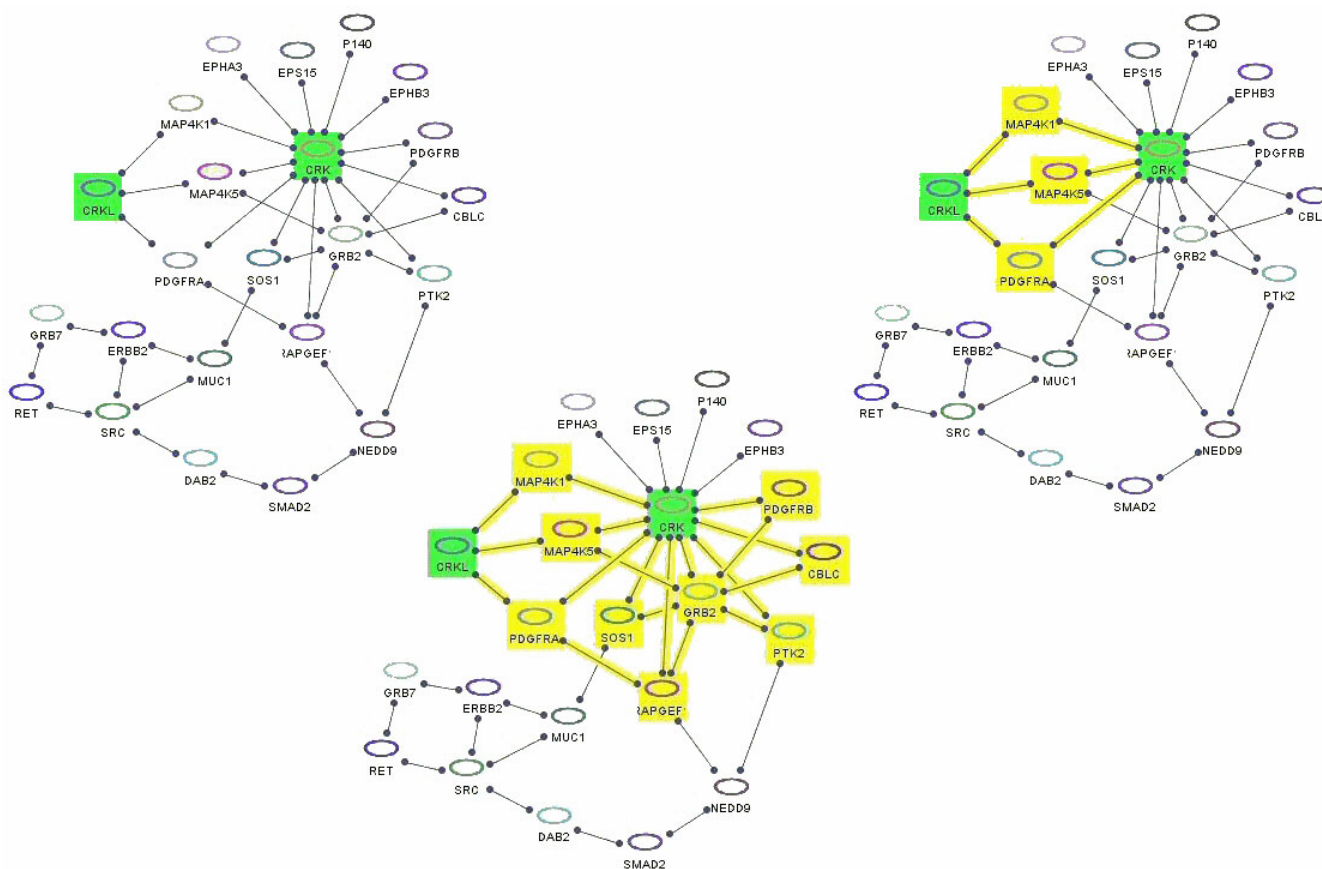


Figure 11
Graph of interest. (left) A PPI network with proteins of interest CRK and CRKL (green); (middle) Graph of interest (yellow) formed by using paths of length up to 3 ($k = 3$) between nodes of interest (green); (right) Graph of interest with $k = 2$ ($k = 1$ returns no results).

```

5  return R
algorithm GOI-COMPOUNDBFS(S, k, dir)
1  Add all the nodes in set S to queue Q
2  Initialize dir labels of all the nodes in S to zero
3  T := ∅
4  while Q ≠ ∅ do
5    n1 := Q.DEQUEUE()
6    for e ∈ n1.INCIDENTEDGES(dir) do
7      if dir = fwd then e.LABEL(dir) := n1.LABEL(dir) + 1
8      else e.LABEL(dir) := n1.LABEL(dir)
9    for n2 ∈ e.COMPOUNDOTHEREND(n1) do
10     T := T ∪ {e, n2}
11     if n2.LABEL(dir) > n1.LABEL(dir) + 1 then
12       n2.LABEL(dir) := n1.LABEL(dir) + 1
13     if n2.LABEL(dir) < k and n2 ∉ S then
14       Q.ENQUEUE(n2)
15  return T

```

The complexity of this operation is clearly $O(|NB(S, k)|)$; that is, linear in the number of nodes and edges in the k -neighborhood of nodes of interest.

The Paths-of-Interest (PoI) query, on the other hand, performs the same operation but in a constrained manner, from a specified set of source molecules to a specified set of target molecules. More formally,

$$PoI(S, T, k) = G[B], \text{ where } B = \{x \mid x \text{ is on an } S - T \text{ path } P \wedge |P| \leq k\}.$$

Shortest paths between entities

Finding shortest paths between a single or all pairs of vertices in a graph is a commonly used graph operation [35]. This query is a more general version of this operation, where we find and list all shortest paths between source and target sets S and T . This operation might be constrained by a parameter denoting the maximum length of such paths. In addition, a parameter for "relaxing" the shortest requirement might be useful. Thus, for instance, the shortest compound paths between two node sets S and T with maximum length k and "further distance" d can be defined formally as

$$SP(S, T, k, d) = \{P \mid P \text{ is a compound } S - T \text{ path} \wedge |P| \leq \min(l + d, k) \wedge l \text{ is the length of a shortest compound path}\}.$$

Figures 12 and 13 illustrate this query with examples. Below, is the pseudocode for finding $SP(S, T, k, d)$, where mod specifies whether edges are to be treated as directed or undirected.

algorithm SHORTESTPATHS(S, T, k, d, mod)

1 $R := SP\text{-COMPOUNDBFS}(S, T, k, d, mod)$

2 **return** BUILDUPANDENUMPATHS(S, T, R)

algorithm BUILDUPANDENUMPATHS(S, T, R)

1 **for** $n \in R$ **do**

2 Construct a new path p for n

3 Add all nodes in set R to queue Q

4 $W := \emptyset$

5 **while** $Q \neq \emptyset$ **do**

6 $n_1 := Q.DEQUEUE()$

7 **for** $n_2 \in n_1.COMPOUNDNEIGHBORS(mod)$ **do**

8 **if** $n_2.LABEL() = n_1.LABEL() - 1$ **then**

9 $W := W \cup \{(n_1, n_2)\}$

10 **if** $n_2 \notin W$ **and** $n_2.LABEL() \neq 0$ **then**

11 $W := W \cup \{n_2\}$

12 $Q.ENQUEUE(n_2)$

13 **if** n_2 is the first neighbor **then** Concatenate n_2 to paths of n_1

14 **else** Clone all the paths of n_1 and add them to paths of n_2

15 Update path list of n_2

16 Update sign of paths of n_2 with edge (n_1, n_2)

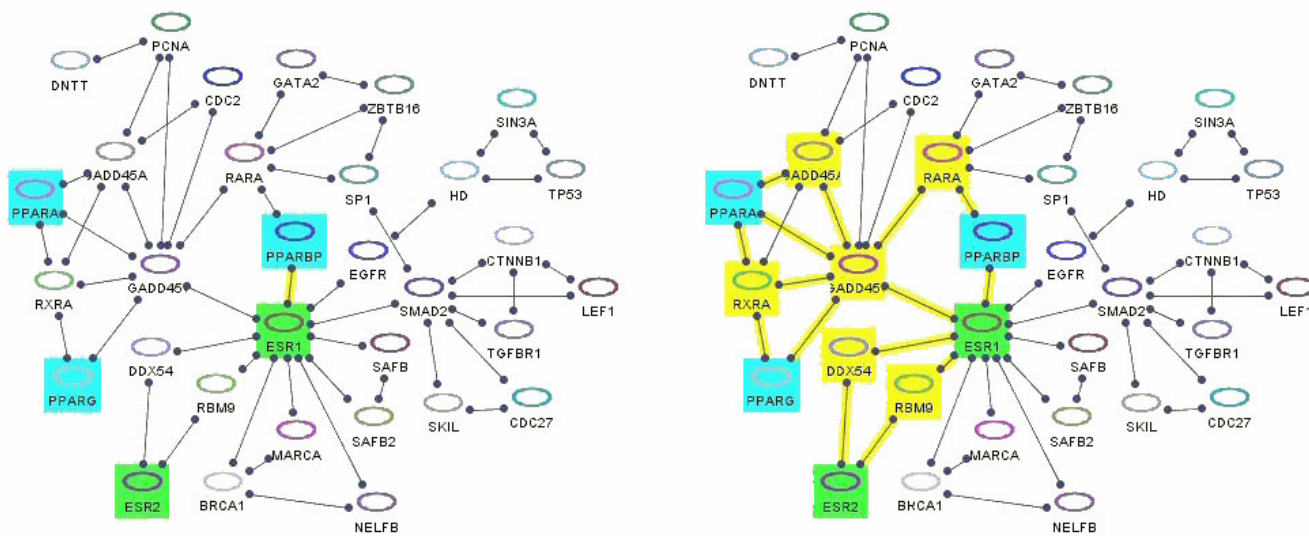


Figure 12
Shortest paths. Shortest paths (yellow) between bioentities whose names start with "PPA" (cyan) and those whose names contain "ESR" (green) with (left) $d = 0$ and (right) $d = 2$. Notice that the length of a shortest path between these two node sets is 1.

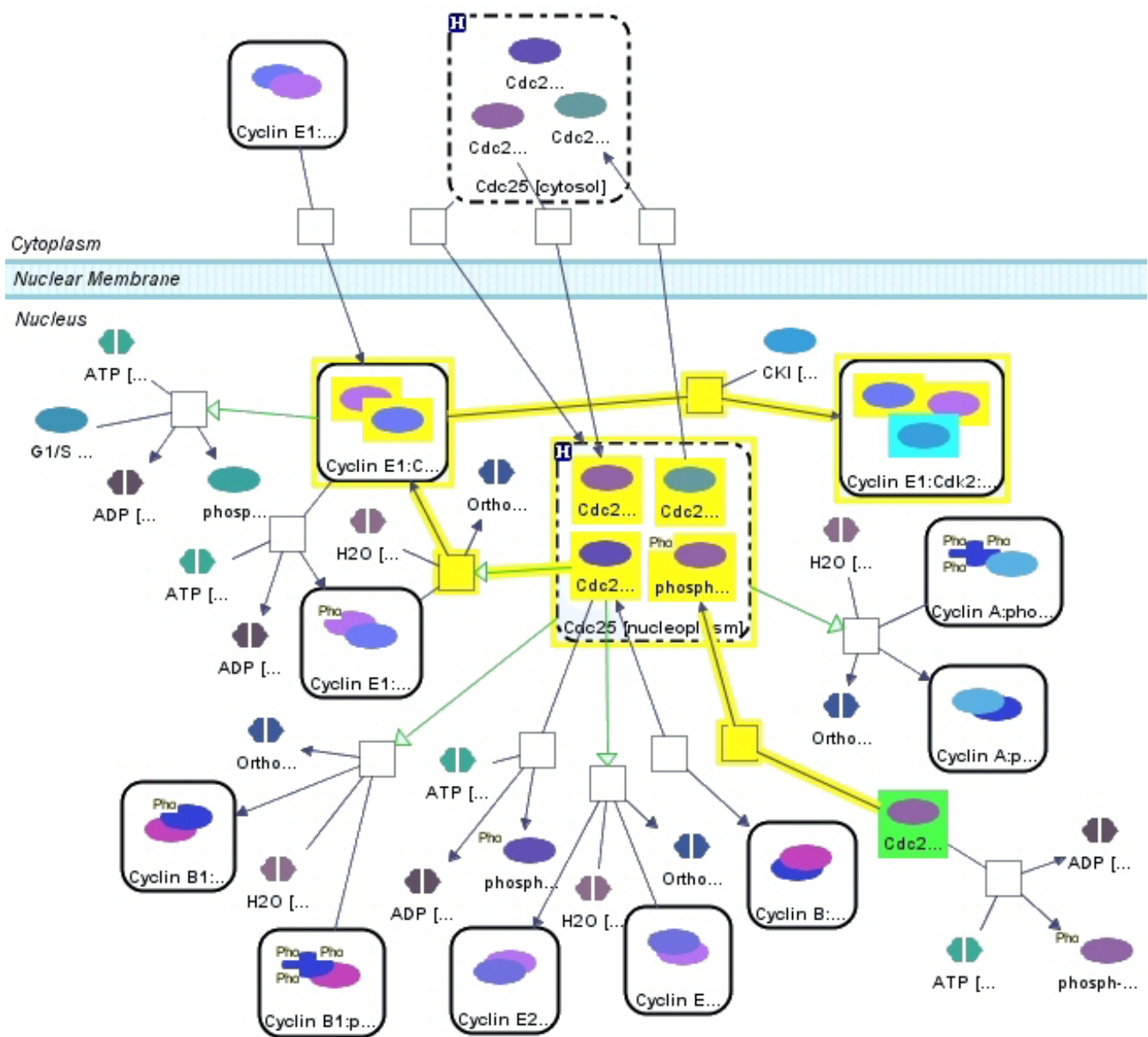


Figure 13
Shortest paths. Shortest path (yellow) between Cdc25C (green) and CKI (cyan) in nucleus. Compound nodes with dashed borders represent homologies, whereas compound nodes with solid borders represent molecular complexes.

17 return W , paths

Here, SP-COMPOUNDBFS runs a BFS, starting with nodes in set S in provided mod , up to maximum depth k or shortest compound path length plus d , whichever is smaller, and returns the reached nodes in T while labeling each node with its distance from source. The overall complexity of the algorithm is $O(l + |NB(S, k)|)$, where l is the total length of the paths enumerated. Here, l can be exponential in the size of the graph, in the worst case. Notice that

the above algorithm *enumerates* all shortest compound paths. If it suffices to find all the nodes and edges on such paths, rather than listing individual paths, BUILDUPANDENUMPATHS may be simplified, resulting in a theoretically faster operation.

In the context of pathways, one might also be interested in only positive, $SP^+(S, T, k, d)$, or negative shortest compound $A-B$ paths in a given pathway graph. Another useful type of operation is to find *first* k shortest compound

paths (not necessarily unique!) between specified node sets. More formally,

$$k\text{-SP}^+(A, B, k) = \{P_1, P_2, \dots, P_k\},$$

Where

$P_i, i = 1, \dots, k$, is a positive compound $A - B$ path and $\sum_{i=1}^k |P_i|$ is minimum over all compound path sets of size k .

Feedback of an entity

Inferring causal relationships between biological entities [36,37] is both critically important and difficult. One problem stems from feedback loops abundant in biological systems. When analyzing the results of these algorithms or methods, one often wants to check if there exists a known feedback loop to flag the inferences that are potentially false.

This operation results in a list of positive or negative compound cycles that contain a specified entity. It is useful for finding feedback signals and metabolic cycles in a network. Positive feedback of a node s with maximum length k is defined as

$$\text{FB}^+(s, k) = \{C \mid C \text{ is a positive compound cycle} \wedge s \text{ is on } C \wedge |C| \leq k\}.$$

Figure 14 illustrates this with an example. Our algorithm is based on generating all cycles starting from a given set of source nodes in a directed graph, as described in [38].

The algorithm starts at source s , and builds a directed path $sn_1n_2n_3 \dots n_k$ in a depth-first manner. A cycle is found if the next vertex n_{k+1} equals s . After generating this cycle, the next edge going out of n_k is explored. If all edges going out from n_k have been explored or the maximum length is exceeded, the algorithm backs up to the previous vertex n_{k-1} , and continues. This process continues until we try to back up past the source node s . At that point, all cycles involving s have been discovered, so s can be removed from the graph, and the process can be repeated until the source set becomes empty.

To prevent traversing cycles originating at a vertex n_i during the search rooted at s , all vertices on the current path are marked as "unavailable" extensions of that path. For this, a flag is maintained, which is set to false as soon as n is appended to the current path. That node will remain unavailable until the algorithm backs up past n to its previous vertex on the graph. If the current path up to n does not lead to a cycle, it will remain unavailable, even if the algorithm backs up past it. This prevents redundant dead-

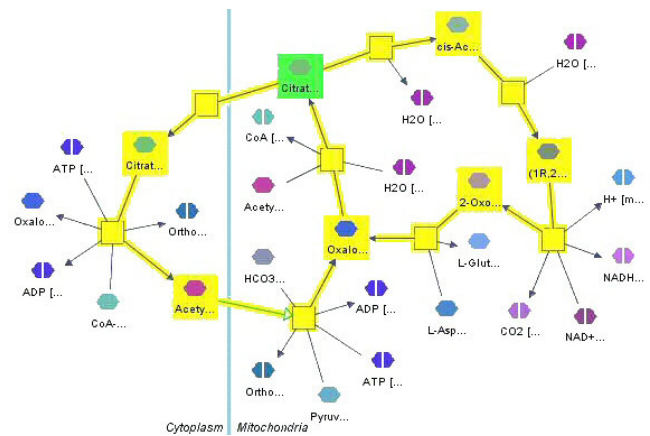


Figure 14

Positive feedback. Positive feedback (yellow) of a specified Citrate state in mitochondria (green) with up to length 10. The result contains two metabolic cycles; one in mitochondria (of length 10) and one through cytoplasm (of length 8).

end searches. Vertex n will, however, be marked available if a cycle could not be found due to cycle length limit, because it is possible for a shorter path to form a cycle by going through n .

Similar to the basis operation given in [38], this algorithm is of $O(|\text{NB}(S, k)| \cdot (c + 1))$ time complexity, where c is the total number of cycles (positive and negative) discovered.

Authors' contributions

All authors participated in the design of the framework and the algorithms. UD directed the research and development. AC implemented most algorithms. ED and OB helped with implementation. All authors read and approved the final manuscript.

Additional material

Additional file 1

A querying scenario. A sample session in which subsequent queries and complexity management operations are performed to form a model that might be of use to a PATIKAWeb user.

Click here for file

[<http://www.biomedcentral.com/content/supplementary/1471-2105-10-376-S1.PDF>]

Additional file 2

Detailed analysis of shortest path length versus execution time. Illustrates the effect of source and target set sizes in execution time for shortest path query.

Click here for file

[<http://www.biomedcentral.com/content/supplementary/1471-2105-10-376-S2.PDF>]

Acknowledgements

The authors thank the rest of the Patikans (especially S. Tekin and H. Kucuk) for their contributions. This work was supported in part by TUBITAK (grant I04E049).

References

- Matthews L, Gopinath G, Gillespie M, Caudy M, Croft D, de Bono B, Garapati P, Hemish J, Hermjakob H, Jassal B, Kanapin A, Lewis S, Mahajan S, May B, Schmidt E, Vastrik I, Wu G, Birney E, Stein L, D'Eustachio P: **Reactome knowledgebase of human biological pathways and processes**. *Nucleic Acids Res* 2009:D619-22.
- Caspi R, Foerster H, Fulcher CA, Kaipa P, Krummenacker M, Latendresse M, Paley S, Rhee SY, Shearer AG, Tissier C, Walk TC, Zhang P, Karp PD: **The MetaCyc Database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases**. *Nucleic Acids Res* 2008:D623-31.
- Okuda S, Yamada T, Hamajima M, Itoh M, Katayama T, Bork P, Goto S, Kanehisa M: **KEGG Atlas mapping for global analysis of metabolic pathways**. *Nucleic Acids Res* 2008:W423-6.
- Demir E, Babur O, Dogrusoz U, Gursoy A, Ayaz A, Gulesir G, Nisanci G, Cetin-Atalay R: **An Ontology for Collaborative Construction and Analysis of Cellular Pathways**. *Bioinformatics* 2004, **20(3)**:349-356.
- Bondy JA, Murty USR: *Graph Theory with Applications* Great Britain: The McMillan Press Ltd; 1976.
- Aittokallio T, Schwikowski B: **Graph-based methods for analysing networks in cell biology**. *Briefings in Bioinformatics* 2006, **7(3)**:243-255.
- Reddy VN, Mavrovouniotis ML, Liebman MN: **Petri Net Representations in Metabolic Pathways**. *1st International Conference on Intelligent Systems for Molecular Biology* 1993:328-336.
- Hofstadt R, Thelen S: **Qualitative modeling of biochemical networks**. In *Silico Biology* 1998, **1**:39-53.
- Fukuda K, Takagi T: **Knowledge representation of signal transduction pathways**. *Bioinformatics* 2001, **17(9)**:829-837.
- Tian Y, McEachin RC, Santos C, States DJ, Patel JM: **SAGA: a subgraph matching tool for biological graphs**. *Bioinformatics* 2007, **23(2)**:232-239.
- Yip KY, Yu H, Kim PM, Schultz M, Gerstein M: **The tYNA platform for comparative interactomics: a web tool for managing, comparing and mining multiple networks**. *Bioinformatics* 2006, **22(23)**:2968-2970.
- Shlomi T, Segal D, Ruppin E, Sharan R: **QPath: a method for querying pathways in a protein-protein interaction network**. *BMC Bioinformatics* 2006, **7**:199.
- Sharan R, Ideker T: **Modeling cellular machinery through biological network comparison**. *Nature Biotechnology* 2006, **24(4)**:427-433.
- Gting R: **GraphDB: Modeling and Querying Graphs in Databases**. *Proc 20th Int Conf on Very Large Databases, Santiago, Chile 1994:297-308* [<http://portal.acm.org/citation.cfm?id=672980>].
- Leser U: **A query language for biological networks**. *Bioinformatics* 2005, **21(2)**:33-39.
- Croes D, Couche F, Wodak SJ, van Helden J: **Metabolic PathFinding: inferring relevant pathways in biochemical networks**. *Nucleic Acids Research* 2005, **33**:W326.
- Baitaluk M, Sedova M, Ray A, Gupta A: **BiologicalNetworks: visualization and analysis tool for systems biology**. *Nucleic Acids Research* 2006:466-471.
- Dogrusoz U, Erson E, Giral E, Demir E, Babur O, Cetintas A, Colak R: **PATIKAWEB: a Web interface for analyzing biological pathways through advanced querying and visualization**. *Bioinformatics* 2006, **22(3)**:374-375.
- BioPAX: **Biological Pathways Exchange**. 2007 [<http://www.biopax.org>].
- SBGN: **Systems Biology Graphical Notation**. 2007 [<http://www.sbgng.org>].
- Dogrusoz U, Feng Q, Madden B, Doorley M, Frick A: **Graph Visualization Toolkits**. *IEEE Computer Graphics and Applications* 2002, **22**:30-37.
- Barabasi AL, Oltvai ZN: **Network biology: understanding the cell's functional organization**. *Nat Rev Genet* 2004, **5(2)**:101-113 [<http://dx.doi.org/10.1038/nrg1272>].
- Bader G, Cary M, Sander C: **Pathguide: a pathway resource list**. *Nucleic Acids Research* 2006:D504-6.
- Pathway Commons: **Pathway Commons**. 2007 [<http://www.pathwaycommons.org>].
- Bonarius HPJ, Schmid G, Tramper J: **Flux analysis of underdetermined metabolic networks: The quest for the missing constraints**. *Trends in Biotechnology* 1997, **15**:308-314.
- Babur O, Demir E, Ayaz A, Dogrusoz U, Sakarya O: **Microarray Data Analysis and Pathway Activity Inference in PATIKA**. 2004.
- Yeung N, Cline MS, Kuchinsky A, Smoot ME, Bader GD: **Exploring biological networks with Cytoscape software**. *Curr Protoc Bioinformatics* 2008, **Chapter 8(Unit 8.13)**.
- Funahashi A, Matsuoka Y, Jouraku A, Morohashi M, Kikuchi N, Kitano H: **CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks**. *Proceedings of the IEEE* 2008, **96(8)**:1254-1265.
- Keshava Prasad TS, Goel R, Kandasamy K, Keerthikumar S, Kumar S, Mathivanan S, Telikicherla D, Raju R, Shafreen B, Venugopal A, Balakrishnan L, Marimuthu A, Banerjee S, Somanathan DS, Sebastian A, Rani S, Ray S, Harrys Kishore CJ, Kanth S, Ahmed M, Kashyap MK, Mohmood R, Ramachandra YL, Krishna V, Rahiman BA, Mohan S, Ranganathan P, Ramabadran S, Chaerkady R, Pandey A: **Human Protein Reference Database-2009 update**. *Nucleic Acids Res* 2009, **37**:D767-772.
- Brohe S, van Helden J: **Evaluation of clustering algorithms for protein-protein interaction networks**. *BMC Bioinformatics* 2006, **7**:488.
- Wong DJ, Nuyten DS, Regev A, Lin M, Adler AS, Segal E, Vijver MJ van de, Chang HY: **Revealing targeted therapy for human cancer by gene module maps**. *Cancer Res* 2008, **68**:369-378.
- Bild AH, Yao G, Chang JT, Wang Q, Potti A, Chasse D, Joshi MB, Harpole D, Lancaster JM, Berchuck A, Olson JA, Marks JR, Dressman HK, West M, Nevins JR: **Oncogenic pathway signatures in human cancers as a guide to targeted therapies**. *Nature* 2006, **439**:353-357.
- Engelman JA, Zejnullahu K, Mitsudomi T, Song Y, Hyland C, Park JO, Lindeman N, Gale CM, Zhao X, Christensen J, Kosaka T, Holmes AJ, Rogers AM, Cappuzzo F, Mok T, Lee C, Johnson BE, Cantley LC, Jnne PA: **MET amplification leads to gefitinib resistance in lung cancer by activating ERBB3 signaling**. *Science* 2007, **316**:1039-1043.
- The Cancer Genome Atlas Research Network: **Comprehensive genomic characterization defines human glioblastoma genes and core pathways**. *Nature* 2008, **455**:1061-1068.
- Cormen TH, Leiserson CE, Rivest RL: *Introduction to Algorithms* Cambridge: McGraw-Hill; 1985.
- Wang K, Alvarez MJ, Bisikirska BC, Linding R, Basso K, Dalla Favera R, Califano A: **Dissecting the interface between signaling and transcriptional regulation in human B cells**. *Pac Symp Biocomput* 2009, **20**:264-275.
- Danos V, Feret J, Fontana W, Harmer R, Krivine J: **Rule-based modeling of cellular signalling**. In *CONCUR'07* Springer Berlin; 2007:17-41.
- Reingold EM, Nievergelt J, Deo N: *Combinatorial Algorithms: Theory and Practice* Englewood Cliffs, NJ: Prentice-Hall; 1977.

Publish with **BioMed Central** and every scientist can read your work free of charge

"BioMed Central will be the most significant development for disseminating the results of biomedical research in our lifetime."

Sir Paul Nurse, Cancer Research UK

Your research papers will be:

- available free of charge to the entire biomedical community
- peer reviewed and published immediately upon acceptance
- cited in PubMed and archived on PubMed Central
- yours — you keep the copyright

Submit your manuscript here:
http://www.biomedcentral.com/info/publishing_adv.asp

