Advanced Querying Interface for Biochemical Network Databases

Brendan Elliott | Stephen Mayes | Ali Cakmak | Gultekin Ozsoyoglu | Z. Meral Ozsoyoglu

Department of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, OH 44106, USA

{elliott, mayes, cakmak, tekin, meral}@case.edu

ABSTRACT

Querying biochemical networks in flexible ways over the web is important to facilitate ongoing biological research. In this paper, we present a querying interface for biological networks, more specifically, metabolic networks. The interface allows for the specification of a large class of containment, path, and neighborhood queries with ease from a web browser. The query specification process is user-friendly, employs hierarchically arranged relationships among biological entities, and uses auto-complete features. The interface is provided as part of PathCase, a system to store, query, visualize and analyze metabolic pathways at different levels of detail.

Categories and Subject Descriptors

H.2.8 **[Database applications]**: Scientific Databases – *query language, visualization, biological networks.*

1. INTRODUCTION

In this paper, we present a simple, generic, extensible, and yet powerful querying interface for biochemical networks. The interface, called the *Advanced Query Interface* (AQI), is designed to allow users to construct their own ad-hoc queries from scratch with ease using only a web browser. We have implemented the AQI for querying metabolic networks and in three different ways, which illustrates its extensibility. Figure 1 shows the main entry to AQI [1] from within the PathCase [2] system, which is in turn a software system for metabolic pathways, providing web-based *process-, enzyme-, and pathway-centric* views, querying, and analysis of metabolic pathways data.

The goal of the AQI is to allow users to create their own custom queries against the PathCase database without needing any knowledge about the structure of the data behind the query or without needing to know any database query language, such as SQL.

With the AQI, users can search for pathways, processes (i.e., reactions), molecular entities, or organisms, via hierarchically arranged relationships among these objects. We call such queries *relationship queries.* Figure 2 illustrates a query that finds processes (reactions) that appear in the *folate synthesis* pathway or the *citrate cycle* pathway as well as the query execution results.

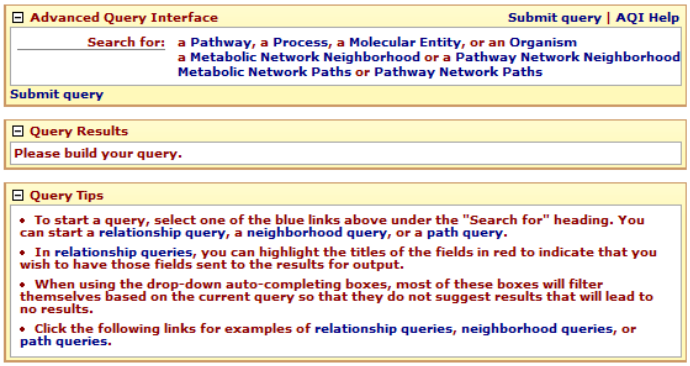

Fig. 1. Starting an AQI Query

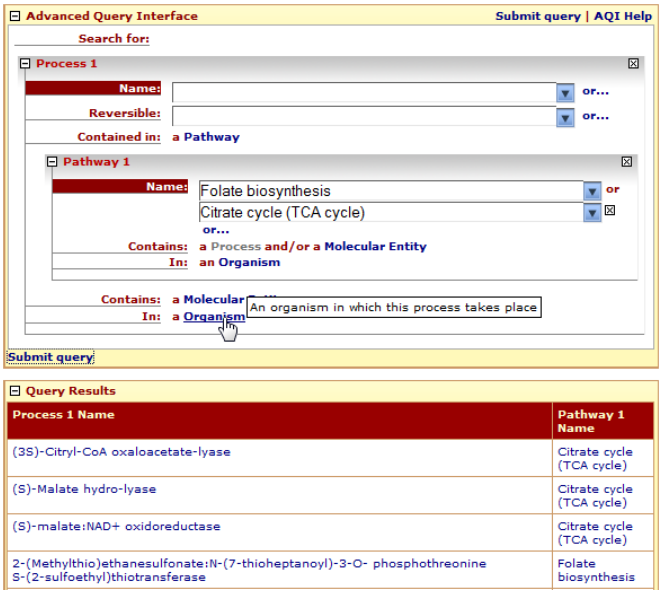

Fig. 2. Query to find processes in either folate biosynthesis pathway or citrate cycle pathway

AQI users can also search for metabolic *network neighborhoods*, classified into metabolic network neighborhoods or pathway links-only neighborhoods. Finally, *path queries* allow users to

SAC'10, March 22-26, 2010, Sierre, Switzerland.

search for paths, classified into metabolic network paths, or pathway links-only paths. The AQI query specification paradigm has five features that contribute to the system's user-friendliness, all illustrated in Figure 2.

1. Through carefully nested nodes and phrases, an AQI query is designed to be "English-like" in order to help users "read" directly what the query specifies. This feature also promotes easy query revisions as a user can read the query specified so far, and then make changes as needed.
2. The query interface is constructed dynamically in an intuitive way by the user according to the query in her/his mind. This feature has the advantage over static interfaces that a wide range of queries can be expressed using a single interface, and the same query can be constructed in different ways.
3. All fill-in (i.e., input) boxes in an AQI query have dynamic auto-completion features (via AJAX programming), eliminating the need for users to remember and type complete and exact names for molecules, reactions, or pathway names--a significant hurdle in using interfaces.
4. Whenever the results of a query are viewable as biochemical networks (i.e., graphs, e.g., neighborhood queries), AQI provides query results in a graph form, in addition to the tabular form.
5. Whenever the AQI provides users a choice to be made via clicking a button, bringing the cursor over the button shows a relevant descriptive text explaining what the choice is about.

A generic interface like the AQI is useful for two reasons.

- Without a generic interface, most systems provide hard-coded interfaces or forms for users to pose pre-determined queries by filling in values and receiving a result. Such an approach lacks flexibility as each new query has to be built by implementing an interface-specific program—a major effort.
- Some users who are not computer savvy prefer not to use an elaborate interface, but request a single and simple form-based interface. However, as users or their preferences change, new interfaces are continuously needed, and each new interface has to be built from scratch by a programmer. In such cases, a generic interface is useful in that, for each new interface, the systems person can simply create and use one specific instance of the generic interface. This is indeed the case for some PathCase "built-in queries" [3], where these queries are in reality specific AQI query instances. Thus, responding to a user's request to add a new built-in query is as simple as saving a specific AQI query instance, and making it available to the users.

PathCase [4] is designed for exploratory querying and visualization, and provides

(a) Visualizations of metabolic pathways at different and multiple abstraction levels of genetic, molecular, biochemical and organismal details,
(b) Querying capabilities for metabolic pathways, including keyword search, ad hoc querying (i.e., the AQI) and predetermined built-in querying (i.e., built-in queries), using both textual and graphical user interfaces.

This paper is organized as follows. In section 2, we discuss related work. Section 3 briefly specifies the PathCase data model and discusses the specifications of three different classes of queries, namely, relationship, neighborhood and path queries, respectively. Section 4 concludes.

2. RELATED WORK

Among pathway data sources with advanced query interfaces, BioCyc provides multiple ways to access and query its underlying metabolic pathway databases [5]. Users can pose a wide-range of queries by invoking Application Programming Interface methods with proper parameters. Alternatively, BioCyc allows users to submit direct SQL queries through BioWarehouse relational database. These access types are highly useful for users who are capable of writing programs, or who have some knowledge of specific query languages and the underlying data model. On the querying interface side, BioCyc's web-based "Structured Advanced Query Page" (SAQP) [6] is a very powerful querying interface involving many entity and relationship types. And, SAQP shares AQI's user-friendliness features 1, 2 and 3 listed in the background. SAQP allows users to dynamically construct queries by adding additional conditions based on their needs. An SAQP query (a) starts by specifying a single entity type, where predicates can only be defined on the attributes of that selected entity, and then (b) reaches out to other entity types iteratively by specifying relationships involving other entities via conditions involving existential and universal quantifications (i.e., "for some object" or "for all objects"). BioCyc has also another query interface, called "Advanced Query Form" (AQF). AQF allows relationship queries between different entities involving single-level binary relationships. In comparison, user-friendliness features 4 and 5 are not available in SAQP or AQF.

Genolink [7] is an advanced graph-based querying system. In its underlying model, each node corresponds to a biological entity, and edges represent semantic relationships between entities. Users can intuitively formulate their queries by constructing a graph that visually represent an advanced query. Genolink also provides powerful and easy-to-use querying facilities, but limits itself to relationship queries, and does not consider path and neighborhood queries. Also, Genolink is not a web-based system, hence users are required to download the application, install it, and then, also download the data to query into their own environment. Being a web-based system, AQI is readily accessible, and it also allows path and neighborhood queries.

Cytoscape [8] is an open-source biological graph visualization project, and allows path and neighborhood queries on the visualized graph. Its advantage is that users can use it to visualize their own data. This is also a disadvantage in that, for any visualization, users first need to transfer data into Cytoscape's environment in a well-defined format, and then Cytoscape runs queries only on this currently visualized data. Cytoscape is also not a web-based application; hence not readily available to the users over a web browser. Last but not the least, Cytoscape does not employ hierarchically organized and relationship-based dynamic query interface model.

KEGG [9] is one of the most commonly used authoritative pathway data sources. However, its querying capabilities involve simple keyword search, and KEGG does not provide an advanced querying mechanism, where AQI running on PathCase with KEGG data fills in this missing capability on such a large curated pathways data.

PATIKAweb, described in [10], is another powerful system that handles biological pathway data similar to PathCase. PATIKAweb also has a querying interface with a hierarchical tree-like interface; however, PATIKAweb queries are more

focused around pre-designed orderings of entities, and it is not clear whether PATIKAweb has support for nesting multiple types of nodes to create joins like in the AQI. While PATIKAweb also allows path queries, including additional queries such as shortest path, it does not provide as many options for specifying the behavior of path and neighborhood queries.

A few other systems, namely Query-by-Example (QBE) [11], Portable Explorer of Structured Objects (PESTO) [12], and the Pathway Query Language (PQL) [13] have attempted to achieve similar goals as the AQI, such as simplifying querying or negating the need for the knowledge of a database schema or new language when querying. QBE simplifies querying, but still requires working directly with a database schema and a query language. PESTO uses boxes within a large canvas to represent individual objects and users can click-and-drag several types of objects on to the canvas to browse them and optionally fill in field data to create a filtered query. PQL provides a highly specialized SQL-like language to execute queries against graph data for paths, neighborhoods, or subgraphs based on the source graph and given parameters.

Finally, there are several other works (e.g., [14 - 19]) that do not directly compare to AQI, but can be considered somewhat relevant due to their focus on paths in metabolic networks. Hence, we briefly include such works in the related work discussion, as well. Croes et al. [14] proposes improvements for pathway inference by eliminating highly connected metabolites from the network, and weighing others metabolites based on their connectivity. Their web interface [15] was not available to study, despite our several attempts at different dates. Beasley and Planes [16] studies a mathematical optimization model to recover already known pathways computationally. This work does not mention any querying interface or tool implementing the associated optimization models. Pathway Hunter Tool [17] performs shortest path analysis on metabolic networks by employing chemical structure information as the basis of similarity between candidate substrates and products. This tool and similar others (e.g., [18]) may be useful not primarily as a generic database query tool, but to infer new (unknown) reactions, thereof paths, between metabolites. Küffner et al. [19] describes a system that unifies pathways from different data sources into a PETRI net. However, this system does not include an advanced query interface, such as AQI.

3. DATA and QUERY MODEL

PathCase database is relational, with pathways, processes, molecular entities, and organisms forming four basic PathCase entities. AQI allows *selected* attributes of these four entity types, as listed in Table 1, to be used by users to specify their query conditions.

PathCase data model represents a metabolic pathway [20] in the form of a graph where nodes represent molecules, and edges represent reactions (or processes) connecting the molecules that take part in the reaction. The term *process* denotes a reaction, which can include a catalyzing protein (which also identifies the reaction and may have an Enzyme Commission (EC) number), co-factors, inhibitors, and activators. Reactions may be one-way or reversible. We refer to molecular objects as *molecular entities* which include basic molecules, proteins, enzymes, genes, and amino acids. A *pathway* can be viewed as a set of interconnected processes, while a process can be viewed as being made up of molecular entities. More generally, an entire pathways database can be viewed as a single large graph of interconnected reactions, in which certain sub-graphs are identified as specific pathways. PathCase metabolic pathways are categorized with respect to the metabolism that they are involved in, such as the amino acid metabolism, lipid metabolism, etc.

Entity Type	Attributes used to specify AQI query conditions
pathway	name
process	name, reversible
molecule	name, name type, molecule type, role, amount
organism	name, group name

Table 1. Basic Entity Attributes Used by AQI queries

The PathCase data model is designed in a flexible manner, in order to hold metabolic network data of multiple data sources: currently, PathCase is made available for (i) Kegg pathways (licensed), (ii) BioCyc (human-only) pathways (for demonstration purposes), and (iii) sample pathways (from literature).

3.1 AQI Relationship Queries

Relationship queries involve hierarchically arranged combinations of the four main entity types, namely, pathway, process, molecular entity, and organism. The AQI is a graphical system where users specify their queries using nodes within a tree-structure. For relationship queries, each node represents an entity of the four basic types listed in Table 1.

Each AQI query is comprised of one root node that determines the type of the query (i.e., relationship, path, or neighborhood query). Each node contains one or more *input fields* (as fill-in boxes) used to specify selection criteria on the selected entity, as well as buttons for adding child nodes.

Child nodes are added *inside* the parent node right after the label that describes the relationship between the parent and the child. The semantics of the query is to find entities where *there exists* an instance of the specified relationship between parent and child.

Example. Another way of reading the query of Figure 2 is: “Find names of all processes that are in (i.e., exist in) pathways “Folate biosynthesis” or “Citrate cycle”.

If a parent node contains more than one child node (as in the case of the query of Figure 6), then the semantics are that *all* such related entities must exist (i.e., an ‘AND’ relationship is satisfied).

The number and the type of child nodes allowed depend on the entity’s relationships with other types of entities. Each input field can either be given a value that is used as a filter in the query, be selected for output to the user, or both. Each field is comprised of one or more input boxes (text boxes, drop-downs, check boxes, etc.) with descriptive text explaining their purpose.

3.1.1 Query Specification

To specify a query, the user clicks to one of the four entity types, namely, pathway, process, molecular entity, or organism, and the corresponding named node appears. Each opened node contains a number of input fields in the form of “attribute condition fill-in boxes”, for attributes listed in Table 1. Note that boxes that are left blank have no affect on the query.

Example. By clicking on the "process" button in the main AQI menu (See Figure 1), a *Process 1* node is opened (See Figure 2), and two attribute condition fill-in boxes, namely, *name* and *reversible*, are made available to the user.

The output attributes of a query are selected (highlighted with a dark-color) and de-selected (highlighted with no color) by clicking on their names. When an entity node is opened, the default output attribute is *Name* (which is highlighted when opened).

Example. In Figure 2, *name* attributes of *Process 1* and *Pathway 1* are highlighted; accordingly, the query execution output produces <*Process 1 Name, Pathway 1 Name*> pairs.

When an entity node is created, in addition to a selected set of its attributes, a number of "relationship" buttons are displayed.

Example. For the *Process 1* node in Figure 2, there are three relationship buttons, namely, "*Contained in a pathway*", "*Contains a molecular entity*", and "*In organism*". And, when the user clicks to "*Contained in a pathway*", *Pathway 1* node is opened to the user, as shown in Figure 2, with two relationship buttons, namely, "*contains a molecular entity*", and "*in organism*".

As stated above, each of the four entities can be nested within one another forming a tree structure, named the AQI *Query tree*, with the tree edges representing different relationships. For example, processes *contain* molecules, and are *contained in* both pathways and organisms, and the tree itself is physically expressed by indenting the entity nodes and explicitly drawing lines between entity nodes, as illustrated in Figure 2.

To simplify the specification of queries, we have made a decision not to allow arbitrarily many and repetitive nestings of entities, and enforce a predefined strict tree structure, specific to a given root entity type. More specifically, no root-to-leaf path in the query tree can have more than one occurrence of the same entity type, limiting the maximum query tree height to four (i.e., the number of major entities in the PathCase data model; see Table 1).

Example. In Figure 2, the button "*Contains a process*" is disabled.

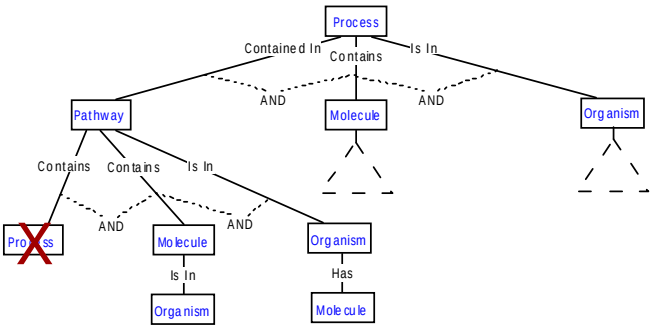

Fig. 4. AQI Query Tree Rooted at Process

Figure 4 specifies the allowed query tree when *Process* is the root of the tree. As an example, note that, if a user specifies a query with *Process* at the root, and *Pathway* as its child, a grandchild node of type *Process* is not provided to the user as an option. Figure 5 shows the full query tree when *Pathway* is the root. The "X" figure underneath a leaf node means that the corresponding node does not have any allowed child node to further expand the query.

3.1.2 *Attribute Condition Specification*

When a user enters a value into an attribute condition fill-in box, a predicate is generated, introducing a restriction (or, filter) to the corresponding attribute. If the user wants to specify multiple OR-connected conditions, s(he) can do so by clicking to the "*or..*" button next to the fill-in box, which adds a new fill-in box.

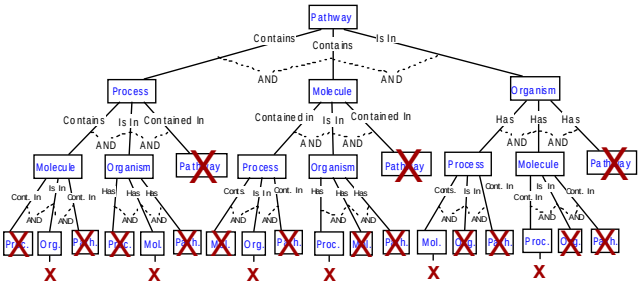

Fig. 5. AQI Query Tree Rooted at Pathway

Example. For the query of Figure 2, the user typed "Folate biosynthesis" to the first *Pathway 1 Name* fill-in box, added a second fill-in box by clicking to the "*or..*" button, and then typed "citrate cycle (TCA cycle)", resulting in the following OR-connected predicate specification:

Name="Folate biosynthesis" OR Name="Citrate cycle (TCA cycle)"

3.1.3 *Query Specification Aids*

There are two query specification aid tools in AQI:

1. Whenever the cursor is moved over a button, AQI displays a descriptive text which explains the functionality of the corresponding button.

Example. In Figure 2, after bringing the cursor over the *In: a organism* button of the *Process 1* node, the text "*An organism in which this process takes place*" is displayed.

2. In a given AQI query, all of the fill-in boxes are designed as auto-completing drop-down boxes. These boxes perform two special functionalities.
   a. The first one is an auto-completion functionality that attempts to complete the user's selection as the first few letters are typed into the box. We give an example.

Example. In Figure 6, after typing "D-Glucose 1" into the *Molecular Entity 1 Name* fill-in box, the system displays four alternatives, and the user chooses "D-Glucose 1-phosphate" as the full-text value to be entered into the box.

   b. The second auto-completion functionality ensures that all the auto-complete alternatives provided to the user are those that are guaranteed to return *at least one* result item for the query at hand. Without this functionality, providing a complete list of about 25,000 Kegg molecules as alternatives is useless.

The second auto-complete functionality is achieved by executing a background PathCase database query on-the-fly using the AQI querying engine as follows. The current query is submitted to the PathCase database in a modified form where

- The only field returned is the suggested fill-in values that are being provided to the user,
- All other conditions on other fields are satisfied, and

- All other conditions specified on that field are temporarily disabled. This feature allows users to quickly construct nonempty queries without needing to first explore the database to learn the names of available pathways, molecules, etc.

Example. In the query of Figure 6, when the user types "D-Glucose 1", a modified AQI query gets executed, which returns the four molecule names shown in Figure 6. However, if the user continues to type and enters 2, resulting in "D-Glucose 12", the AQI returns no suggestions, indicating that, for humans, there is no pathway with a molecule whose name contains the phrase "D-Glucose 12".

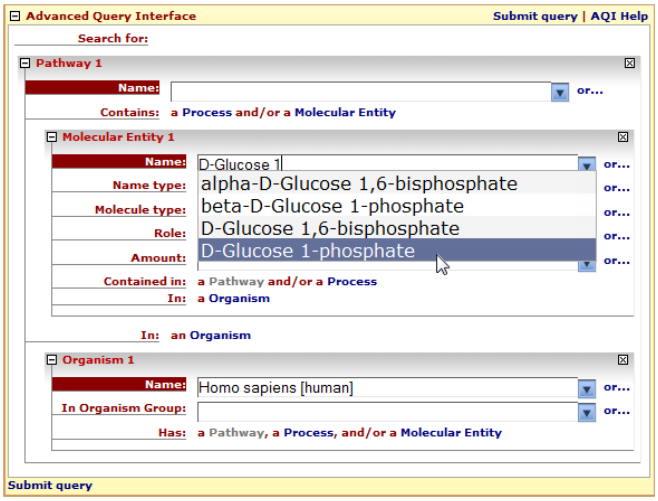

Fig. 6. Formulating a query to find pathways with D-Glucose 1-phosphate in humans using auto-complete

3.1.4 *Query Execution and Implementation*

After adding nodes to the query and entering any necessary data into the fill-in boxes, the query is submitted for execution, and the results are returned in a tabular form.

Example. Query execution results of the query in Figure 6 is shown in Figure 7. Note that, in Figure 6, the user selected as output the triplet <*Pathway 1 Name*, *Molecular Entity 1 Name*, *Organism 1 Name*> which are the column names of the table produced in Figure 7.

Query Results

Pathway 1 Name	Molecule 1 Name	Organism 1 Name
Polyketide sugar unit biosynthesis	D-Glucose 1-phosphate	Homo sapiens [human]
Streptomycin biosynthesis	D-Glucose 1-phosphate	Homo sapiens [human]
Nucleotide sugars metabolism	D-Glucose 1-phosphate	Homo sapiens [human]
Starch and sucrose metabolism	D-Glucose 1-phosphate	Homo sapiens [human]
Galactose metabolism	D-Glucose 1-phosphate	Homo sapiens [human]
Glycolysis / Gluconeogenesis	D-Glucose 1-phosphate	Homo sapiens [human]
Pentose and glucuronate interconversions	D-Glucose 1-phosphate	Homo sapiens [human]
Pyrimidine metabolism	D-Glucose 1-phosphate	Homo sapiens [human]

Fig. 7. Execution Results of the Query in Figure 6.

In summary, the execution of an AQI query proceeds as follows.

At the client-side (JavaScript): the tree-like interface is converted into an XML document and sent to the server (via AJAX).

At the server side: the XML document is parsed and translated into an SQL query which is executed, producing a final output table which is then rendered to the user at the client-side via the *renderer* object functions.

To implement the AQI interface in an extensible manner, we have implemented a central library that contains classes for (i) *the base node*, *field*, *renderer* and *queries* types, (ii) a query parser, and (iii) the querying engine.

The AQI design is extensible in that it allows for virtually any type of query specification and execution by separating the interface from the querying. The interface is managed using two types of objects: the *node definitions* and the *interface renderer*. The *node definition* is where the node specifies which fields are used in the node and which child nodes can appear beneath the node. It also specifies any interface-specific parameters to the *interface renderer*, such as in what order to display the fields and child node links. As nodes are added to the query, the *interface renderer* is responsible for actually drawing the node and displaying it to the user. Once the query has been specified and submitted, the client-side interface is responsible for generating the XML document and passing it on to the querying engine.

The XML document is specified according to a strict schema specification in order to facilitate the communication between the query interface and the library. The node tag is the root element of the document and is also specified as a child of itself if there are child nodes in the query. Each node contains one or more fields based on its definition. Each field can contain multiple values, which are separated by the 'OR' condition as explained above (*attribute-condition specification*).

The AQI's *querier* takes information based on the node definitions, and constructs database SQL queries directly from the query XML document. The engine needs two pieces of knowledge about a particular node: the SQL required for the node and the SQL conditions needed per child node in order to "join" the results of the node with its child nodes. (This simple "joining" scheme becomes much more complex if the capabilities of the AQI are expanded; see [21] for a more powerful, but complex, interface). For the individual node, the engine constructs (i) the list of tables it requires, (ii) any WHERE clauses that universally apply to those tables, and (iii) the SELECT and WHERE clauses for the fields. For the child nodes, each parent node stores the tables and join conditions that model the many-to-many relationship between the entities. Given solely this information, the engine generates the SQL query for each individual node, and joins the tables of parent and child nodes using its "join" method. The basic idea behind the join method is to merge the source table(s) needed for the parent node with the destination tables required for each child node. The SQL query with the parent's information then has (i) all of the child's information added to it, and (ii) each table along the path from the source table to the destination table added to it, along with the proper intermediary join conditions as defined in the parent node's definition. To accommodate for new versions of the AQI for different applications (we are currently planning a version for a metabolomics project), we have chosen this generic and extensible SQL query engine design versus a more specific PathCase-focused implementation.

After the *querier* is finished executing, the result table is given to the *renderer*. For extensibility, the renderer is designed to operate in any manner the query requires, such as rendering controls on a graphical user interface or HTML code for a web page.

The central library is designed with flexibility and extensibility in mind. The interface and the queries all follow a basic structure

and must inherit the base types that are given in the library. The base types themselves are designed to give programmers flexibility when creating different classes of queries so that they could easily change or switch renderers or query engines with very little effort. This allows the programmer to create different renderers for each particular application while only needing a single shared set of node definitions and querying engine.

The above-summarized design allows for the AQI to be easily ported to other projects. If relationship queries are to be processed, then one instance of each type need to be defined along with their input fill-in box values and relationships. If other types of (e.g., neighborhood or path) queries are to be processed, then the software designer implements project-specific queriers and renderers. In fact, neighborhood and path queries are implemented by creating specialized queriers that use the path querying engine (as opposed to the SQL querying engine used for relationship queries).

3.2 NEIGHBORHOOD QUERIES

Neighborhood queries are designed to search for biological entities that are in the vicinity of a given biological entity in a biological network. In PathCase, the metabolic pathways network is represented by two different graph models: (i) the metabolic network graph, and (ii) the pathway links graph.

The *metabolic network graph* represents relationships between molecules and processes. The nodes in this graph represent the individual molecules that are either substrates or products of a process and the *hyperedges* represent processes. Hyperedges are edges that can connect more than two nodes; a process is a hyperedge since each process potentially has multiple molecules as either substrates or products. The edges are directed; pointing from the substrates to the products.

The *pathway links* graph represents the pathways and their relationships at a higher level. The nodes in this graph represent individual pathways, and the edges represent shared molecules between pathways. In order for two pathways A and B to be connected through a shared molecule m, m must be produced by pathway A, and then consumed by pathway B, or vice versa. This production and consumption relationship defines the direction of the edges which point from the pathway that produces the molecule towards the pathway that consumes the molecule.

The AQI allows for the formulation of queries on both graph models. Since the node and edge semantics are different in each graph representation, the query specification involves distinct input fields and constraints depending on the underlying graph representation. Thus, the AQI provides two different neighborhood query specification interfaces.

3.2.1 Metabolic Network Neighborhoods

The neighborhood queries on the metabolic network are accessed by clicking on the "Metabolic Network Neighborhood" button in the main AQI menu (see Figure 1). The first portion of the query is the subgraph definition. The AQI allows for two separate graph restrictions. The first is a flag indicating whether or not the user wishes to include common molecules in the graph, such as H_2O which appears in a large number of reactions and is typically not useful to include in the graph. If excluded, these nodes (and perhaps some reactions that were only linked to these molecules) are removed from the graph.

The second is an option to restrict the graph based on a well-defined subgraph in the overall network, such as a particular pathway or reactions occurring in a particular organism. Also, in this section, a user can choose to execute the query for neighbors located *downstream* or *upstream* of the source biological entity.

Next, the user needs to select the starting (i.e., source) entity, which can be either a molecule or a process. The final section in the interface allows the user to define restrictions on the neighborhood. These restrictions are the length and excluded molecules/reactions restrictions. The length parameter specifies the *border*s of the neighborhood through the minimum and the maximum allowed distance between the source entity and its neighbors. Furthermore, some potential neighbors may be eliminated by enforcing that a path between the source entity and its neighbors does not include a user specified set of molecules/processes. The entities constituting the borders of a neighborhood can be restricted to processes or molecules via selection from the "finishing with" input box. We give an example.

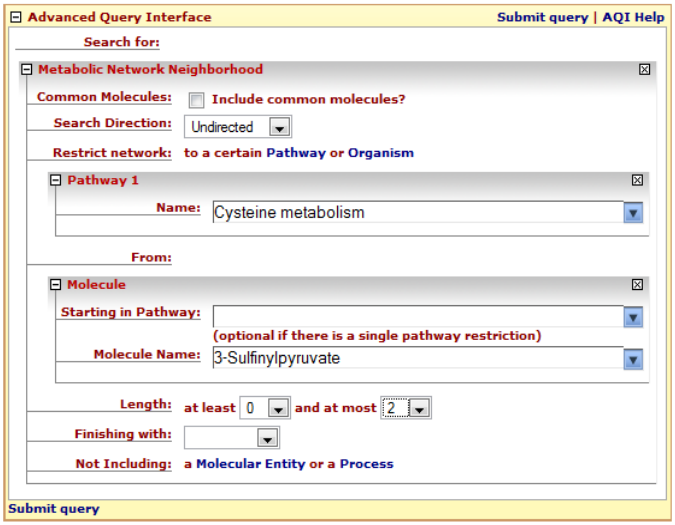

Fig. 8. Metabolic Network Neighborhood Query

Example. Figure 8 shows an AQI query which is set to search for the neighbors of 3-Sulfinylpyruvate (i.e., source molecule) that are located at most 2 steps away (upstream or downstream) from the source molecule, where the metabolic network is restricted to a single pathway, that is, Cysteine metabolism.

Figure 9 depicts the results (in both tabular and graphical formats) for the query in Figure 8. The textual representation of the result set lists the molecules and processes at each *distance* value (i.e., step) up to the maximum distance (i.e., 2, for this query) set by the user. The arrows under molecules column point to processes that are immediately next to the corresponding molecule in the network through substrate/product relationship, and, in the next column, these processes are expanded with their substrates/molecules (i.e., arrows in this column represent edges from a process to its molecules). The graphical output highlights molecules/processes at each distance value in a distinct color, and those processes/molecules that are not in the specified neighborhood are grayed out (i.e. the remainder of the 'cysteine metabolism' pathway).

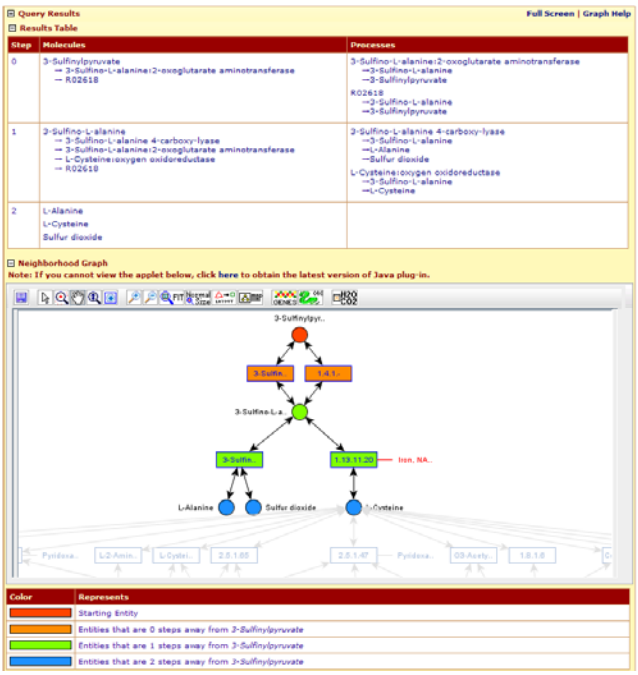

Fig. 9. Execution Results of the Query in Figure 8

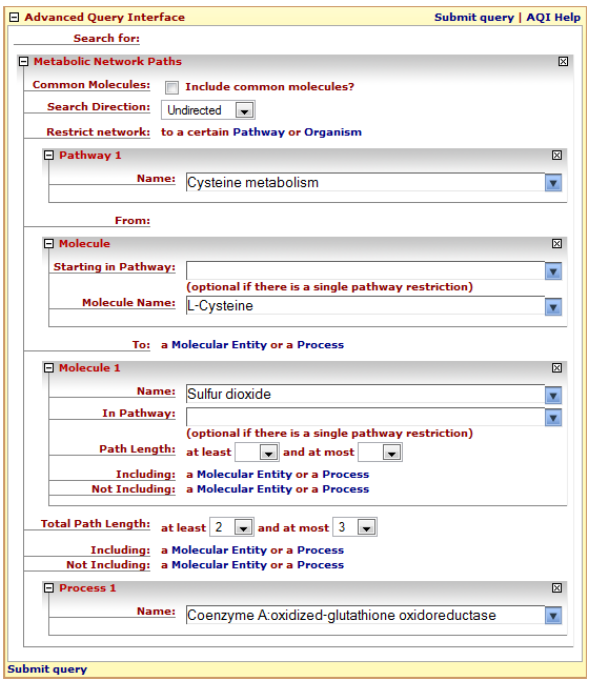

Fig. 10. Metabolic Network Path Query

3.2.2 Pathway Network Neighborhoods

Pathway Network Neighborhood queries have a simpler interface that is similar to the Metabolic Network Neighborhood queries. In this section, we briefly explain the differences. First, the only sub-graph restriction allowed is to limit the graph to the network of a specific organism. Second, the source biological entity can be a molecule or a pathway. Likewise, pathway and/or molecules can be marked for exclusion from the paths between a source entity and its neighbors. For more details, see [23].

3.3 PATH QUERIES

Path queries allow the user to search for all the possible metabolic connection chains between biological entities. Semantically, path queries are a special case of neighborhood queries, with an additional input parameter to specify the "ending" biological entity. As in neighborhood queries, path queries can be executed on either the metabolic network graph or pathway links graph by clicking on the corresponding button in the main AQI menu (see Figure 1).

3.3.1 Metabolic Network Paths

The metabolic network path query interface contains all the input parameter fields from the metabolic network neighborhood query. In addition, path queries require the specification of at least one destination biological entity. The source and the destination entities on the metabolic network can be a process or a molecule.

One unique feature of the AQI path query interface is that it allows the specification of multiple *hops*. A hop is identified by a starting molecule and an ending molecule, and corresponds to a subpath of the overall path from the source entity to the final destination entity. The AQI enforces that the resulting paths contain the user-created hops in the order specified by the user. Hops are created via adding "To" fields on the interface by clicking on the corresponding "To" button(s) more than once.

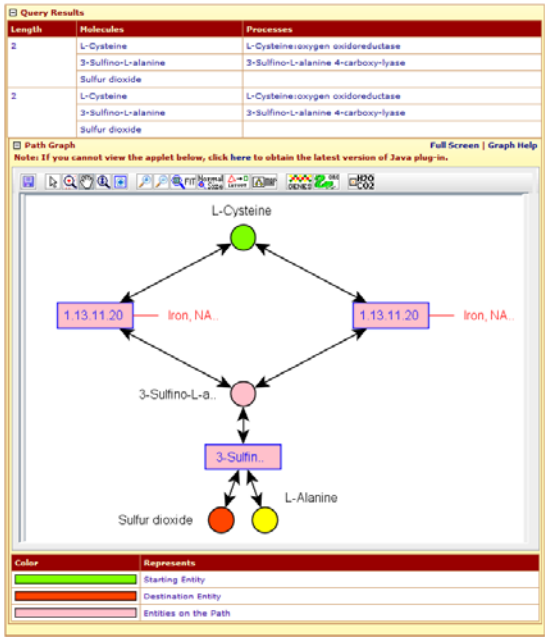

Fig. 11. Execution Results of the Query in Figure 10

Another interesting feature of the AQI path querying scheme is the path restrictions. Each path between the source, hop, or the destination entities can have a restriction; leading to multiple path restrictions if the query contains multiple hops. There are three defined restrictions: a length restriction, an inclusion restriction, and an exclusion restriction. The length restriction specifies a minimum and a maximum length for the path. The inclusion restriction specifies zero or more entities that must be part of the path. And, the exclusion restriction specifies zero or more entities that cannot be part of the path. We give an example.

Example. Figure 10 shows an AQI query specification which is created to search for the paths between L-Cysteine and Sulfur Dioxide, where the query defines two restrictions: (i) the graph is restricted to reactions and molecules of the "cysteine metabolism" pathway, and (ii) path length should be at least 2, and at most 3. Figure 11 presents the result of this query.

The path queries on the pathway links graph are formulated in a similar way. The differences between the path queries on the metabolic network graph and the pathway links graph are the same as those that are discussed for the neighborhood queries.

4. DISCUSSION and CONCLUSIONS

In this paper, we have presented the Advanced Querying Interface (AQI) which enables users to dynamically construct customized query interfaces against biological network databases. The AQI supports three kinds of query types: (i) *relationship queries*, (ii) *neighborhood queries*, and (iii) *path queries* that allow multiple hops.

While the PathCase database currently has many tables (more than 30 tables) involving many other entity types (such as genes, chromosome, gene products, RNA, protein, EC numbers, etc; please see [22] for a simplified subset of the PathCase database schema), as a first step and for simplicity of the interface, we have chosen to specify queries involving only the four entity types *pathway*, *process*, *molecule*, *organism*. The second stage, if the users adapt the AQI, is to expand the design and implementation of the AQI with other entity types in the database. Note that such a type and phrase extension (e.g., the phrase *"encoded by gene mmgD"*, or *"catalyzed by enzyme pyruvate carboxylase")* is quite natural and extensible, and we do not expect that the interface will be overly complicated.

In terms of the metabolic network database size, AQI has scaled very well. PathCase with Kegg data is sizable with 147 pathways for each of 860 organisms, more than 3 million genes, etc. For this database, the AQI query execution is very fast, and scales very well to a browser-based query execution. For performance evaluation of the AQI, please see [23].

5. ACKNOWLEDGMENTS

This research is supported by the National Science Foundation (DBI-0849956, DBI-0743705, CRI-0551603, CCF-0820217).

6. REFERENCES

[1] AQI. 2009. Available at http://nashua.case.edu/PathwaysKegg/Web/LinkForwarder.aspx?rid=AdvancedQueryInterface&rtype=br

[2] PathCase. (2009.a). PathCase System available at http://nashua.case.edu/pathways.

[3] PathCase. (2009.b). PathCase Built-in Queries available at http://nashua.case.edu/PathwaysKegg/Web/LinkForwarder.aspx?rid=SimpleQueries&rtype=br

[4] Elliott, B, Kirac, M, Cakmak, A et al. (2008). PathCase Pathways Database System. Bioinformatics 24(21):2526-2533, November 2008.

[5] Krummenacker, M, Paley, S, Mueller, L, Yan, T, Karp, PD.Querying and Computing with BioCyc Databases, Bioinformatics 21:3454-5 2005.

[6] Structured Advanced Query Page (2009). Available at http://biocyc.org/query.html

[7] Durand, P., Labarre, L., Meil, A. et al. GenoLink: a graph-based querying and browsing system for investigating the function of genes and proteins. BMC Bioinformatics. 2006; 7: 21.

[8] Shannon P, Markiel A, Ozier O et al.. Cytoscape: a software environment for inte-grated models of biomolecular interaction networks. Gen. Res. 2003 Nov; 13(11):2498-504

[9] Kanehisa, M., Goto, S., Hattori, M., Aoki-Kinoshita, K.F. et al. From genomics to chemical genomics: new developments in KEGG. Nucleic Acids Res. 34, D354-357, 2006.

[10] Dogrusoz, U. et al. (2006) PATIKAweb: a Web interface for analyzing biological pathways through advanced querying and visualization, Bioinformatics, 22(3), 374-375.

[11] Zloof, M. (1977). Query by Example. IBM Systems Journal p. 324 – 343 No. 4.

[12] Carey, M. et al. (1996). PESTO: An Integrated Query/Browser for Object Databases. VLDB, p. 203 – 214.

[13] Leser, U. (2005). A Query Language for Biological Networks. Bioinformatics Vol. 21 Suppl. 2 p. ii33 – ii39.

[14] Croes D, Couche F, Wodak SJ, van Helden J. Metabolic PathFinding: inferring relevant pathways in biochemical networks. Nucleic Acids Res. 2005 1;33:W326-30.

[15] MetaPathFinding Tool, (it was offline during the preparation of this manuscript),http://www.scmbb.ulb.ac.be/Users/didier/pathfinding/metabpathfinding.php

[16] Beasley, JE, and Planes, FJ. (2007). Recovering metabolic pathways via optimization. Bioinformatics 23:92-98 2007.

[17] Rahman, SA et al. (2005). Metabolic pathway analysis web service (Pathway Hunter Tool at CUBIC). Bioinformatics 2005, 21: 1189-1193.

[18] Hatzimanikatis, V, Li, C, Ionita, JA, Henry, CS, Jankowski, MD, Broadbelt, LJ. (2005). Exploring the diversity of complex metabolic networks. Bioinformatics, vol. 21, num. 8, 2005, p. 1603-1609.

[19] Küffner R, Zimmer R, Lengauer T. Pathway analysis in metabolic databases via differential metabolic display (DMD). Bioinformatics. 2000 Sep;16(9):825-36.

[20] Michal, G (1999) Biochemical Pathways Spektrum Akademischer Verlag, Heidelberg.

[21] Balkir, H.N. et al (2002), A Graphical Query Language: VISUAL and Its Query Processing, IEEE Transactions on Knowledge and Data Eng., 14(5): 955-978.

[22] PathCase Architecture and Data Model, 2009. Available at http://nashua.case.edu/PathwaysWeb/DataModel.aspx

[23] Mayes, S. (2007) Advanced Interface for Querying Graph Data. Master's thesis, Case Western Reserve University, EECS Dept., Cleveland, Ohio, USA.