Programmatic Access to the SEED Data Via the Network

Authors and Affiliations

R. Overbeek,[3*] R. Aziz,[1, 4] D. Bartels,[1] T. Disz,[1, 2] R. Edwards,[1, 4] S. Gerdes,[3] C. Henry,[1, 2] G. Olsen,[5] R. Olson,[1, 2] A. Osterman,[6] T. Paczian,[2] B. Parrello,[3] G.D. Pusch,[3] A. Rodriguez,[2] R. Stevens,[1, 2] O. Vassieva,[3] V. Vonstein,[3] A. Wilke,[2] and O. Zagnitko[3]

[1]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439
[2]Computation Institute, University of Chicago, Chicago, IL 60637
[3]Fellowship for the Interpretation of Genomes, Burr Ridge, IL, 60527
[4]Department of Computer Science, San Diego State University, San Diego, CA 92182
[5]Department of Microbiology, University of Illinois at Urbana-Champaign, Urbana, IL 61801
[6]The Burnham Institute, San Diego, CA 92037
[*]Corresponding Author

Abstract

The SEED project is a cooperative gene annotation effort initiated in 2003. Researchers from a number of academic and private institutions built the SEED, an integration of genomic data that now contains almost a thousand complete or nearly complete genomes, a constantly updated set of curated annotations embodied in a large and growing collection of encoded subsystems, and a derived set of protein families. All of the SEED code and data are made freely available. Until recently, however, maintaining current copies of the SEED code and data at remote locations has been a pressing issue. This paper describes four network-based servers that address this issue. Specifically, the servers are intended to expose the data in the underlying relational database, support basic annotation services, offer programmatic access to the capabilities of the RAST annotation server, and provide access to a growing collection of metabolic models that support flux balance analysis. Moreover, the four servers offer access to regularly updated data, the ability to annotate prokaryotic genomes, the ability to create metabolic reconstructions and detailed models of metabolism, and access to hundreds of existing metabolic models. Our goal is to support a framework upon which other groups can build independent research efforts. Large integrations of genomic data represent one of the major intellectual resources driving research in biology, and we believe that programmatic access to the SEED data will provide significant utility to a broad collection of potential users.

Author Summary

This paper describes four servers that offer programmatic access to the genomics-related data maintained and distributed via the SEED. Access to the servers, the underlying data, and the code is free to all users. The servers offer convenient programmatic access to a relational database containing approximately 1,000 curated genomes. For several years we have offered network-based interactive access to the SEED data via a number of web sites. This paper announces a facility that supports remote programmatic access to the underlying SEED data and services, allowing users to build independent research efforts upon the work being done to support and extend the SEED. The related RAST annotation server now supports the annotation of 300-400 genomes per month, and we support programmatic data submission and retrieval of results. The servers also offer access to hundreds of metabolic models of prokaryotic genomes and the ability to perform flux balance analysis.

Introduction

In 2003, researchers from several institutions decided to cooperatively construct an integration of genomic data that could be used to support a wide variety of research efforts, including The Project to Annotate 1000 Genomes[1]**.** The intent was to build a common infrastructure that could be shared by the groups. Each group would seek its own funding, contribute to the common infrastructure, and pursue its own goals. The development of the systems architecture and software was done at a number of institutions, with the Fellowship for Interpretation of Genomes (FIG) and Argonne National Laboratory coordinating the effort. This effort was called the SEED project, and the resulting integration is called the SEED.

Motivation

The SEED project focuses on development of technology to support rapid, high-volume, accurate annotation of genomes. Three advances are of central importance:

1. The *subsystems strategy* was adopted as the guiding principle of the effort [2]. This strategy centers on leveraging expert annotations to define a small set of functional roles in all genomes rather than all the functional roles in a small number of genomes.

2. The subsystem effort provided a convenient framework for the curation of a set of protein families that became known as *FIGfams* [3]. The goal was to produce families that contained only *isofunctional homologs*—that is, each family was intended to contain only homologous proteins playing the same functional role. When errors

were detected, corrections were made by updating the underlying subsystems and then regenerating the FIGfams. The rapid evolution of the FIGfam collection has possible made a number of the services described below.

3. Using the subsystems and FIGfams as the underlying technology, the RAST (Rapid Annotations using Subsystems Technology) server was developed and made available in 2007. Thousands of prokaryotic genomes have been annotated with the RAST system, and hundreds more are annotated each month. The RAST system now has several thousand registered users; use of the system is free to anyone, but user registration is required preserve the privacy of each user's genomes.

In addition to these developments relating to the annotation of prokaryotic genomes, several groups have focused on the development of accurate models of metabolism [4, 5]. This work has made it possible to generate hundreds of detailed metabolic models that can be used to support flux-based analysis.

These advances over the past six years have motivated members of the SEED project to provide straightforward, convenient programmatic access to the data developed during the project.

Goals and Architecture of the SEED Servers

In the initial stages of the SEED project, a commitment was made to make the code and data freely available. However, it was not completely clear how best to achieve this goal. Initially, the project used a distributed architecture in which numerous SEED installations were maintained at distinct institutions, and each of these peers could archive and exchange subsystems with one another either directly or via a central server. While support of numerous distinct SEED systems has continued, a major disadvantage of the peer-to-peer approach has been the effort required to continually update and integrate a growing set of data and systems software. One of the installations, the Annotators' SEED, has become the de facto standard used to centralize and synchronize annotations made on remote machines. As errors and conflicts in annotation were detected, they were rapidly corrected on the Annotators' SEED, and the underlying collection of subsystems has continued to be maintained and developed. Thus, the Annotators SEED became the central resource for updating remote copies of the system. A read-only mirror of the Annotators' SEED is maintained by Argonne to provide all users immediate public access to the data (see http://www.theseed.org). A second, writable, public mirror at the University of Chicago is also supported for users to construct their own subsystems and to archive their results, which may then be incorporated into the Annotators' SEED for widespread distribution (http://theseed.uchicago.edu/FIG/index.cgi).

With this centralized data model, code development to support the SEED was done in a distributed framework coordinated through a shared CVS, while curation and annotation used a centralized architecture. However, users of the SEED often desire programmatic access to the latest data, and the lack of a convenient API has hampered effective collaboration, as well as limiting the utility of the data outside the cooperating institutions.

To address this issue, we initially provided a Simple Object Access Protocol (SOAP) server for access to data in the SEED database {ref: Disz et al; http://ws.theseed.org/}. Several problems were encountered with the day-to-day use of that service. In particular, the server abstraction layer consisted of large, monolithic Perl modules that were loaded on each invocation, resulting in a noticeable delay in response to each call made to the server. The encapsulation of the results in SOAP XML also conferred significant parsing and transmission overhead on the data being transferred. Moreover, each operation was atomic and required a single argument that was processed and a single datum returned. In combination, trivial requests such as retrieving all the functions for all of the proteins in a genome took unacceptably long to complete, requiring a separate call for each protein, transferring of many kilobytes of data over the network, and the instantiation of many threads on the server.

The new approach presented here is more extensive and more extensible than our SOAP-based approach. These second-generation servers have significantly lower server-side delay for invocation, as well as significantly reduced network overhead, thus providing more responsive access. Furthermore, the new servers provide a more efficient and flexible computing approach because they are designed to process batches of requests at a time, streaming the responses as they complete. Thus, complex queries can be combined with minimal network and server overhead. The services we provide offer access to the integrated genomic data, subsystems, FIGfams, co-occurrence data, annotation services, RAST annotation submission and job retrieval, and metabolic modeling. All client modules, code examples and documentation are online at http://www.theseed.org/servers/. We are continually expanding these services and improving the underlying documentation.

Results: The Servers and the Services

We describe four servers that we collectively refer to as the SEED servers. These servers currently support approximately 100 methods that can be invoked to extract data and services (see Figure 1).

We maintain server code that resides at the location of the SEED data. Users download a distribution with their choice of runtime environment (currently we support Perl and Java integrations) that they may use to write programs to

access SEED data or perform a number of common bioinformatic tasks using a supplied set of preprogrammed scripts.
The following subsections describe the four servers in more detail.

The Sapling Server

The *Sapling Server* offers access to the underlying integration of genomic data—including genomes, genes, proteins, annotations, subsystems, FIGfams, and co-occurrence data.

On the server side, the Sapling Server accesses a database implemented by an entity-relationship data model (ERDB). The ERDB model is defined by a set of XML metadata describing the entities, relationships, and attributes in a form that can be used to generate queries as well as the documentation and a database diagram (see the web site http://servers.theseed.org/figdisk/FIG/ErdbDocWidget.cgi?database=Sapling). Therefore, the public description of the database remains synchronized with the internal data structures—an important benefit in a database designed for public use.

The Sapling Server is architected such that new features can be added quickly. New data tables may be added by updating the XML metadata, which is processed by a special load program to build the initial database tables. The list of services offered is maintained on the server, so that client software does not need to be updated in order for users to access new features. A web application that converts general database queries to Perl code helps speed implementation of new functions.

A database query is specified by naming the entities and relationships along a path through the ERDB diagram, as shown in Figure 2, along with a list of the data items to be returned and a filter clause that limits the results to the desired data objects (e.g., a particular genome or identifier). The Sapling Server allows direct queries against the database; however, the a set of common data requests are implemented as direct server functions. Sapling Server functions typically accept multiple input values within a single call, allowing a client to minimize the number of requests that must be made to the server. Additional input parameters allow a client to modify the query, for example, to request that the output be in FASTA format or to ask for protein rather than DNA sequences.

A sample **ids_to_sequences** request is shown in Figure 3a. The user specifies four identifiers, and the server returns them as a table (actually a Perl hash) with the associated DNA sequences attached.

The Sapling Server currently supports over 50 functions. These functions are listed on a web page generated automatically from the latest code, ensuring that the documentation remains up to date. A sample showing the web page description of **ids_to_sequences** is shown in Figure 3b.

The Annotation Support Server

The *Annotation Support Server* supports two distinct capabilities relating to the annotation of genomes: *de novo* annotation of either protein or DNA sequences and aggregation of annotations into subsystems. The Annotation Support Server accepts either DNA or protein as input and, depending on the user options, can either use existing gene calls or invoke standard gene callers (e.g., *GLIMMER-3* for protein-encoding genes). The server also houses newly developed high-performance methods to assign function to protein sequences or regions of genomic DNA sequences, based on FIGfams and a unique use of K-mers that act as FIGfam signatures (manuscript in preparation). Below is an example application using these methods that produces a relatively accurate annotation of most microbial genomes within a few minutes. To evaluate the technology, however, users are encouraged to simply submit a known prokaryotic genome to the server for annotation.

Sequences can be submitted to the server in three ways:

1. Programs can directly access the services needed to call genes and assign functions to the proteins encoded within the genome.
2. If the protein-encoding genes have already been identified, the program can assign functions to these sequences. An example program is provided in the download library and is described at http://servers.theseed.org/sapling/server.cgi?pod=svr_assign_using_figfams.pl.
3. A program can take as input fragments of DNA (e.g., from a metagenomic sample) and use the services to detect pieces of protein-encoding genes. Again, an example program is provided in the download library and is described at http://servers.theseed.org/sapling/server.cgi?pod=svr_assign_to_dna_using_figfams.pl).

The server also provides the ability to take as input a set of functional roles (in the controlled vocabulary established by the subsystem collection) and to produce a detailed estimate of which subsystems are represented by those functional roles. That is, one can also use the server to develop a metabolic reconstruction based on the functional roles that have been assigned to the protein-encoding genes.

The RAST Submission/Retrieval Server

The *RAST Submission/Retrieval Server* supports programmatic submission of genomes to the RAST server, retrieval of job status, and retrieval of the final set of annotations. We have run over 1,000 distinct prokaryotic genomes through the RAST server using preprogrammed scripts that are available in the distribution. These scripts and the underlying API enable users to submit genomes to the RAST server, test the status of submitted jobs, and retrieve the output (i.e., annotated genomes).

Three types of input are supported:

1. A FASTA file of contigs that make up the genome to be annotated
2. A file of GenBank formatted entries (with the option to retain the gene calls as given in the uploaded files)
3. ENTREZ ID or genome project ID.

In the last case, the tools we provide will query NCBI for the set of contigs that make up the sequencing project. That set of IDs then becomes the input to the RAST server.

The Metabolic Modeling and Flux Balance Analysis Server

The *Metabolic Modeling and Flux Balance Analysis* (FBA) Server provides programmatic remote access to the SEED biochemistry and genome-scale metabolic model database. The SEED biochemistry database integrates into a single, nonredundant set all the reactions and compounds found in the KEGG database, together with additional curated reactions and compounds [6] and a continuously growing number of published genome-scale metabolic models. Currently this database consists of 15,285 compounds and 12,827 reactions. For compounds, the database also includes database IDs from KEGG and models, names/synonyms, mass, molecular formulas, molecular charge, and estimated Gibbs free energy of formation [7]. For reactions, the database includes database IDs from KEGG and models, names/synonyms, stoichiometry, EC numbers, pathways, and estimated Gibb free energy change of reactions [7]. Compound charge, formula, formation energies and reaction stoichiometry are all calculated for aqueous conditions at neutral pH. The user has two options for accessing this data on the Metabolic Modeling and FBA Server: A precompiled program available for download from http://servers.theseed.org/sapling/server.cgi or the API. The precompiled program accepts a limited number of command-line parameters and returns the compound and reaction data in text format. The API provides a much more flexible interface for accessing the server capabilities and returns server data in an organized data structure. All API functions used to access the Metabolic Modeling and FBA Server capabilities are documented in detail at http://servers.theseed.org/sapling/server.cgi.

The SEED database also contains a large number of genome-scale metabolic models, including 13 published models [8-20] and 154 models generated from the annotated genomes stored in the SEED[21]. The Metabolic Modeling and FBA Server also provides the user with an API to remotely obtain a list of the models in the SEED and to download data on the compounds and reactions in each SEED model. The server returns the following data for each reaction in a specified model: (i) all data from the SEED biochemistry database, (ii) a list of the genes associated with each reaction in the model in a format that captures how the protein products encoded by the genes function to catalyze the reaction (as either independent enzymes or multienzyme complexes), and (iii) a list of compartments in the model where the reaction takes place and the

directionality/reversibility of the reaction in each compartment. For the model compounds, the server returns the data from the SEED biochemistry database. As with the biochemistry data, all the model data in the server is accessible either via the precompiled program or the API.

The Metabolic Modeling and FBA Server also enables users to run various FBA studies on any of the genome-scale metabolic models stored in the SEED database. These studies can be performed while simulating any of 485 distinct media conditions currently encoded in the SEED database (which includes all Biolog media conditions and a variety of complex media formulations). Both the precompiled program and the API enable users to obtain a list of the media conditions currently stored in the SEED and details on the compounds included in each formulation. Once a model and media condition have been selected for simulation, the server provides an interface for running three types of FBA simulation: (i) *simple growth simulation* to predict maximum growth rate of the organism in the selected media, (ii) *flux variability analysis* (FVA) [22] to classify the reactions and compounds in the model according to their behavior during growth in the selected media, and (iii) *single gene knockout analysis* to predict the genes essential for growth in the selected media.

The simple growth simulation returns the maximum predicted growth rate of the model given the input parameters, the predicted flux through the model reactions during maximum growth, and the predicted uptake and production of nutrients from and to the environment during maximum growth.

The FVA simulation returns the predicted class of every reaction and compound in the model during growth given the input parameters. Reactions in the model are classified as *forward essential* or *reverse essential* if they are required for growth to occur, with the *forward* and *reverse* referring to the direction in which the reactions must proceed. Reactions that are not essential for growth but still active are classified as *forward variable*, *reverse variable*, and *variable*, with the *forward* and *reverse* indicating when reactions proceed only in a single direction. Reactions are classified as *blocked* if they cannot carry flux under the conditions specified by the user. Metabolites in the model are classified as *essential nutrients* or *essential products* if their uptake or secretion is required for growth in the input conditions, and they are classified as *transported* if they can be taken up or secreted but are not essential for growth. In addition to classifying the reactions and compounds in the model, the FVA simulation returns the maximum and minimum values for the flux through each reactions and the uptake/secretion of each metabolite.

The single gene knockout analysis rapidly simulates the individual knockout of every gene represented in the model during growth in the input conditions. Based on these simulations, the analysis produces a list of the predicted essential genes and the predicted nonessential genes in the model. Both the precompiled

program and API allow the user to run any of the three simulation types from the command line.

All three simulation types accept the same user input: the name of the model to be run, the name of the media formulation that growth should be simulated in, a list of genes in the model that should be knocked out during the simulation, and a list of the reactions in the model that should be knocked out during the simulation. See http://servers.theseed.org/sapling/server.cgi for detailed documentation on all Metabolic Modeling and FBA Server functions.

Example Applications

To help users begin to use the various services, we provide a set of tutorials and coding examples. In this section we discuss a small set of examples that illustrate the intended use of the system.

Converting Gene and Protein IDs

Dealing with IDs of genes and/or the proteins they encode is often nontrivial. In the SEED project we use IDs that specify protein-encoding genes in a rapidly growing set of genomes, and we support correspondences between these IDs and those used by other annotation efforts. The SEED has two notions of equivalence: (1) two IDs that represent either protein-encoding genes or protein sequences are said to be *sequence equivalent* if the protein sequences are identical and (2) two IDs that represent either exactly the same protein-encoding gene or the precise protein encoded by the gene (that is, "the protein sequence of gene *X* in genome *Y*") are said to be *precisely equivalent*. Unfortunately, in the presence of multiple versions of thousands of genomes, perfect maintenance of the "precisely equivalent" correspondence is virtually impossible.

Our first example script takes a command-line argument containing a single ID and produces a table for all assertions of functions for sequence equivalent IDs. Each ID in the input is associated with the name of the genome containing it, the function for that ID, the source of the functional assignment assertion, and an indication of whether the source of the assertion provided a confidence for their estimate. The code is available at http://servers.theseed.org/sapling/server.cgi?code=server_paper_example1.pl

Generating a Metabolic Reconstruction

Given a set of functional roles, one often wishes to understand which subsystems can be inferred from the set. Our second example script reads as input a set of functional roles and constructs a table of subsystems that can be identified, along with their variation codes. The data displayed in this simple example could form the start of a research project to gather the functional roles not connected to subsystems, to determine whether they were not connected

because a small set of functional roles were not present in the input, and to seek candidates for such "missing functional roles." The ability to easily map functional roles into subsystems will improve as the SEED annotation effort improves its collection of encoded subsystems [23]. The code for this example is shown at http://servers.theseed.org/sapling/server.cgi?code=server_paper_example2.pl.

Creating Custom Interfaces

The SEED provides the ability to graphically display the chromosomal regions around a set of genes (normally from distinct genomes); for example, see http://seed-viewer.theseed.org/seedviewer.cgi?page=Annotation&feature=fig|83333.1.peg.4. The SEED also offers an alternative for creating custom interfaces, moreover, one that does not require the user to know appropriate SEED IDs. This approach exploits the conversion capabilities of the SEED for creating a program to accept arbitrary protein IDs. It also exploits the ability of SEED to map functional roles into subsystems as described in the preceding example. The result is a tool that enables the user to take a SEED ID and a region size and extract the genes that are found within a region centered on the designated gene. The code for this example is shown at

http://servers.theseed.org/sapling/server.cgi?code=server_paper_example3.pl

Accessing Functional Coupling Data

A great deal has been learned from studying genes that tend to occur close to one another in diverse genomes [24, 25] [26-28]. In particular, the co-occurrence of hypothetical and nonhypothetical proteins can be exploited to suggest the function of the former based on the function of the latter.

The example program at
http://servers.theseed.org/sapling/server.cgi?code=server_paper_example4.pl

illustrates the potential for constructing custom tools by going through all of the protein-encoding genes in all of the complete prokaryotic genomes maintained within the SEED looking for "hypothetical proteins" that tend to co-occur with genes encoding functions that can be connected to subsystems. The program constructs a table showing the following:

- Gene
- Function of the gene
- Genome id containing the gene
- Biological name of the genome
- Nonhypothetical gene in a subsystem that appears to have the strongest measure of co-occurrence
- Measure of gene co-occurrence

- Function assigned to the co-occurring gene contained in a subsystem.

This table can therefore be used to suggest functions for hypothetical proteins that could be tested experimentally.

Assigning Functions to Protein Sequences

The SEED can be used to assign functions to a file of protein sequences. The code for this example is at http://servers.theseed.org/sapling/server.cgi?code=server_paper_example6.pl

This program reads a FASTA file of protein sequences and attempts to assign function to those sequences using a K-mer–based algorithm (manuscript in preparation). When a function is proposed, the program will produce a "score" (the number of distinct K-mers that were matched) and an estimate of phylogenetic neighborhood—a representative genome that is "phylogenetically close" to the genome containing the protein, if an estimate can reasonably be given.

Running Flux Balance Analysis on the SEED Model of *E. coli*

In our final example, we demonstrate how to run a variety of FBA algorithms on the SEED model of *E. coli* and how to print all data from the *E. coli* model and the results of the FBA into an output table. For the code, see http://servers.theseed.org/sapling/server.cgi?code=server_paper_example7.pl

The program starts by obtaining a list of all compounds and reactions in the SEED *E. coli* model (Seed83333.1) using the "*get_compound_id_list*" and "*get_reaction_id_list*" functions, respectively. The program then uses these lists to obtain detailed data on all the *E. coli* compounds and reactions (using the "*get_compound_data*" and "*get_reaction_data*" functions, respectively). This data is stored in two tables: one for compounds and one for reactions. Next the "*classify_model_entities*" function is used to run a flux variability analysis (FVA) on the SEED *E. coli* model. In this particular FVA, the reactions and compounds in the *E. coli* model are classified while simulating growth in LB media (called ArgonneLBMedia in the SEED model). At this point, the data returned by the "*classify_model_entities*" function is added onto the compound and reaction tables prepared previously. In the next step, the code uses the "*simulate_model_growth*" function to run a standard FBA on the SEED *E. coli* model, maximizing the model growth rate in simulated glucose minimal media (called Carbon-D-Glucose in the Model SEED). The data returned by this function is also added to the reaction and compound tables. In the final call to the server, the program uses the "*simulate_all_single_gene_knockout*" function to simulate the single knockout of all *E. coli* genes, and the results of this study are

stored in a gene table. The remainder of the program handles the printing of the compound, reaction, and gene tables to the files CompoundTbl.txt, ReactionTbl.txt, and GeneTbl.txt, respectively.

Discussion

The four initial SEED servers provide programmatic access to the data developed by the SEED project. They expose the current data in a form that is conveniently accessed computationally. The installation and maintenance of the client-side software require minimal effort. We have constructed the underlying methods to support relatively large-grained data transfers, allowing the construction of relatively efficient programs.

The four SEED servers provide network-based access to an integration of genomic data containing hundreds of genomes, the ability to locally support rapid annotation of microbial genomes, the ability to submit and retrieve jobs from the RAST server (thereby offering access to our continuing improvements in microbial annotation), and the ability to explore metabolic models for hundreds of organisms.

We believe that the underlying implementation of these new servers is efficient enough to address the needs of most users. We will continue providing occasional stand-alone versions of the SEED to users who need more performance or privacy.

Methods

Distribution of the Server Packages

The SEED servers project is documented and can be downloaded from the servers web site, http://servers.theseed.org.

The Perl distribution contains the following.

Client Packages

1. The Sapling server - SAPserver.pm
2. The MODEL server - MODELserver.pm
3. The Annotation Support Server - ANNOserver.pm
4. The RAST server - RASTserver.pm

Utilities

The package of utilities, called SeedUtils.pm, contains functions that are useful for bioinformatics but that do not require access to the databases.

Programming Using the Servers
The SEED servers provide all necessary network operations in a client package that can be used to access the server functions. One uses these like any other Perl package. For instance, to find all genomes in the SEED, one does the following.

```
#!/usr/bin/perl -w

use strict;
use SAPserver;

my $sapObject = SAPserver->new();
my $genomes = $sapObject->all_genomes();

foreach my $g (sort { $genomes->{$a} cmp $genomes->{$b} }
keys(%$genomes)) {
    print "$g\t$genomes->{$g}\n";
}
```

The function call $sapObject->all_genomes() marshals the correct server-side function call and arguments into a network package, transmits that package to the server, waits for and retrieves the answer, processes any returned error codes, decodes the return package into a Perl data structure, and returns the result. All function calls in all the client packages perform these basic services.

The Java distribution contains the following.

Client Packages
The org.theseed.servers.serverConnections package handles connecting to the server, transmitting and receiving the data, and converting data structures from the server into Java data structures. The classes in org.theseed.servers.servers packages handle connecting to each of the servers and making the appropriate calls.

Programming Using the Servers
We recommend that the code be accessed in eclipse (http://www.eclipse.org/), netbeans (http://www.netbeans.org/), or a similar graphical IDE. These are used like any other class. For instance, to find all genomes in the SEED, one does the following.

```
import java.util.HashMap;
import servers.SAPserver;

public class AllGenomes {
      public static void main(String[] args) {
```

```
            SAPserver sapling = new SAPserver();
            HashMap<String, String> genomes = sapling.allGenomes();
            for (String id : genomes.keySet())
                  System.out.println(id + "\t" + genomes.get(id));
      }
}
```

Availability and Future Directions

The latest documentation and downloads can always be found at http://servers.theseed.org.
G
We are planning packages for use by other programming languages such as Python, and we are planning a SOAP version of these packages. These should all be available in mid- to late 2010.

Acknowledgments

This work was supported in part with Federal funds from the National Institute of Allergy and Infectious Diseases, National Institutes of Health, Department of Health and Human Services, under Contract No. HHSN266200400042C and Contract No. HHSN272200900040C

This work was supported in part by the U.S. Dept. of Energy under Contract DE-AC02-06CH11357.

RE thanks Daniel Cuevas, Josh Hoffman, and Sajia Akhter for help with the Java code.

References

1. Overbeek, R. *The Project to Annotate 1000 Genomes* . Available from: http://www.theseed.org/wiki/index.php/Annotating_1000_genomes#The_Project_to_Annotate_1000_Genomes.
2. Overbeek, R., et al., *The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes.* Nucleic Acids Res, 2005. **33**(17): p. 5691-702.
3. Meyer, F., R. Overbeek, and A. Rodriguez, *FIGfams: yet another set of protein families.* Nucleic Acids Res, 2009. **37**(20): p. 6643-54.
4. Henry, C.S., et al., *iBsu1103: a new genome-scale metabolic model of Bacillus subtilis based on SEED annotations.* Genome Biol, 2009. **10**(6): p. R69.
5. DeJongh, M., et al., *Toward the automated generation of genome-scale metabolic networks in the SEED.* BMC Bioinformatics, 2007. **8**: p. 139.

6. Kanehisa, M., et al., *The KEGG databases at GenomeNet.* Nucleic Acids Research, 2002. **30**(1): p. 42-46.
7. Jankowski, M.D., et al., *Group contribution method for thermodynamic analysis of complex metabolic networks.* Biophysical Journal, 2008. **95**(3): p. 1487-99.
8. Reed, J.L., et al., *An expanded genome-scale model of Escherichia coli K-12 (iJR904 GSM/GPR).* Genome Biology, 2003. **4**(9): p. 1-12.
9. Feist, A.M., et al., *A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1261 ORFs and thermodynamic information.* Mol Syst Biol, 2007. **3**: p. 121.
10. Durot, M., et al., *Iterative reconstruction of a global metabolic model of Acinetobacter baylyi ADP1 using high-throughput growth phenotype and gene essentiality data.* BMC Syst Biol, 2008. **2**: p. 85.
11. Oh, Y.K., et al., *Genome-scale reconstruction of metabolic network in Bacillus subtilis based on high-throughput phenotyping and gene essentiality data.* J Biol Chem, 2007. **282**(39): p. 28791-9.
12. Goelzer, A., et al., *Reconstruction and analysis of the genetic and metabolic regulatory networks of the central metabolism of Bacillus subtilis.* BMC Syst Biol, 2008. **2**: p. 20.
13. Schilling, C.H., et al., *Genome-scale metabolic model of Helicobacter pylori 26695.* Journal of Bacteriology, 2002. **184**(16): p. 4582-4593.
14. Oliveira, A.P., J. Nielsen, and J. Forster, *Modeling Lactococcus lactis using a genome-scale flux model.* BMC Microbiol, 2005. **5**: p. 39.
15. Feist, A.M., et al., *Modeling methanogenesis with a genome-scale metabolic reconstruction of Methanosarcina barkeri.* Mol Syst Biol, 2006. **2**: p. 2006 0004.
16. Jamshidi, N. and B.O. Palsson, *Investigating the metabolic capabilities of Mycobacterium tuberculosis H37Rv using the in silico strain iNJ661 and proposing alternative drug targets.* BMC Syst Biol, 2007. **1**: p. 26.
17. Suthers, P.F., et al., *A genome-scale metabolic reconstruction of Mycoplasma genitalium, iPS189.* PLoS Comput Biol, 2009. **5**(2): p. e1000285.
18. Nogales, J., B.O. Palsson, and I. Thiele, *A genome-scale metabolic reconstruction of Pseudomonas putida KT2440: iJN746 as a cell factory.* BMC Syst Biol, 2008. **2**: p. 79.
19. Duarte, N.C., M.J. Herrgard, and B.O. Palsson, *Reconstruction and validation of Saccharomyces cerevisiae iND750, a fully compartmentalized genome-scale metabolic model.* Genome Research, 2004. **14**(7): p. 1298-1309.
20. Becker, S.A. and B.O. Palsson, *Genome-scale reconstruction of the metabolic network in Staphylococcus aureus N315: an initial draft to the two-dimensional annotation.* BMC Microbiol, 2005. **5**(1): p. 8.
21. Henry, C.S., et al., *Model Seed: a resource for high-throughput generation, optimization, and analysis of genome-scale metabolic models.* . 2010.

22. Mahadevan, R. and C.H. Schilling, *The effects of alternate optimal solutions in constraint-based genome-scale metabolic models.* Metabolic Engineering, 2003. **5**(4): p. 264-276.
23. Venter, J.C., et al., *Environmental genome shotgun sequencing of the Sargasso Sea.* Science, 2004. **304**(5667): p. 66-74.
24. Overbeek, R., et al., *Use of contiguity on the chromosome to predict functional coupling.* In Silico Biol, 1999. **1**(2): p. 93-108.
25. Dandekar, T., et al., *Conservation of gene order: a fingerprint of proteins that physically interact.* Trends Biochem Sci, 1998. **23**(9): p. 324-8.
26. Overbeek, R., et al., *The use of gene clusters to infer functional coupling.* Proc Natl Acad Sci U S A, 1999. **96**(6): p. 2896-901.
27. Wolf, Y.I., et al., *Genome alignment, evolution of prokaryotic genome organization, and prediction of gene function using genomic context.* Genome Res, 2001. **11**(3): p. 356-72.
28. Moreno-Hagelsieb, G., *Inferring functional relationships from conservation of gene order.* Methods Mol Biol, 2008. **453**: p. 181-99.

Figure legends

Figure 1. *The SEED servers architecture*. The client packages (currently available for Perl or Java) handle the HTTP requests and responses and parse the data from the appropriate lightweight data exchange formats to data structures. The four servers access the SEED data.

Figure 2. *Entities and relationships in the SEED*. The entities (boxes) are connected to each other by a series of relationships (diamonds) that describe how the two entities relate. To move from one entity (e.g. “Identifier”) to another (e.g., “DNA Sequence”), the series of connections shown by the shaded arrow is made. This way, any entity can be connected, either directly or indirectly, to any other entity.

Figure 3. *Processing ids_to_sequences*. (a) The ids_to_sequences function call accepts multiple IDs as an argument and uses the Sapling server to process the calls. These are returned as a single table. (b) A detailed description of each call (in this example, the ids_to_sequences) is provided online and is automatically generated from the entity-relationship models shown in Figure 2.

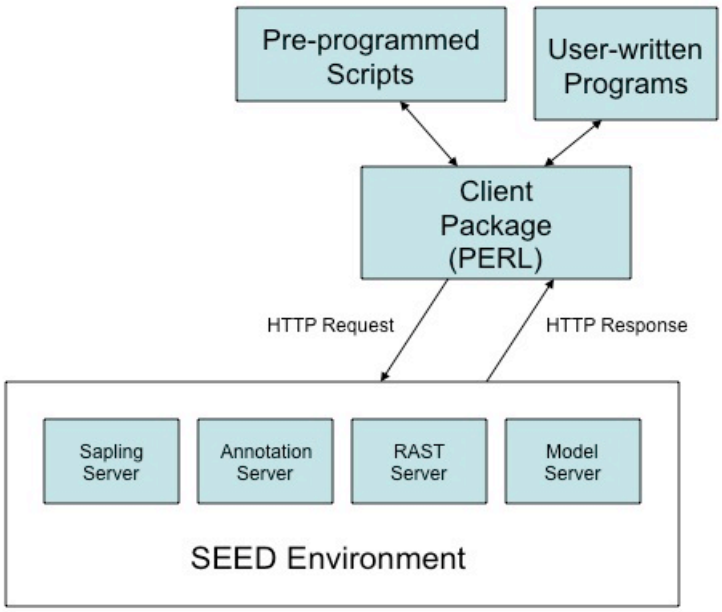

Pre-programmed Scripts
User-written Programs
Client Package (PERL)
HTTP Request
HTTP Response
Sapling Server
Annotation Server
RAST Server
Model Server
SEED Environment

Figure 1.

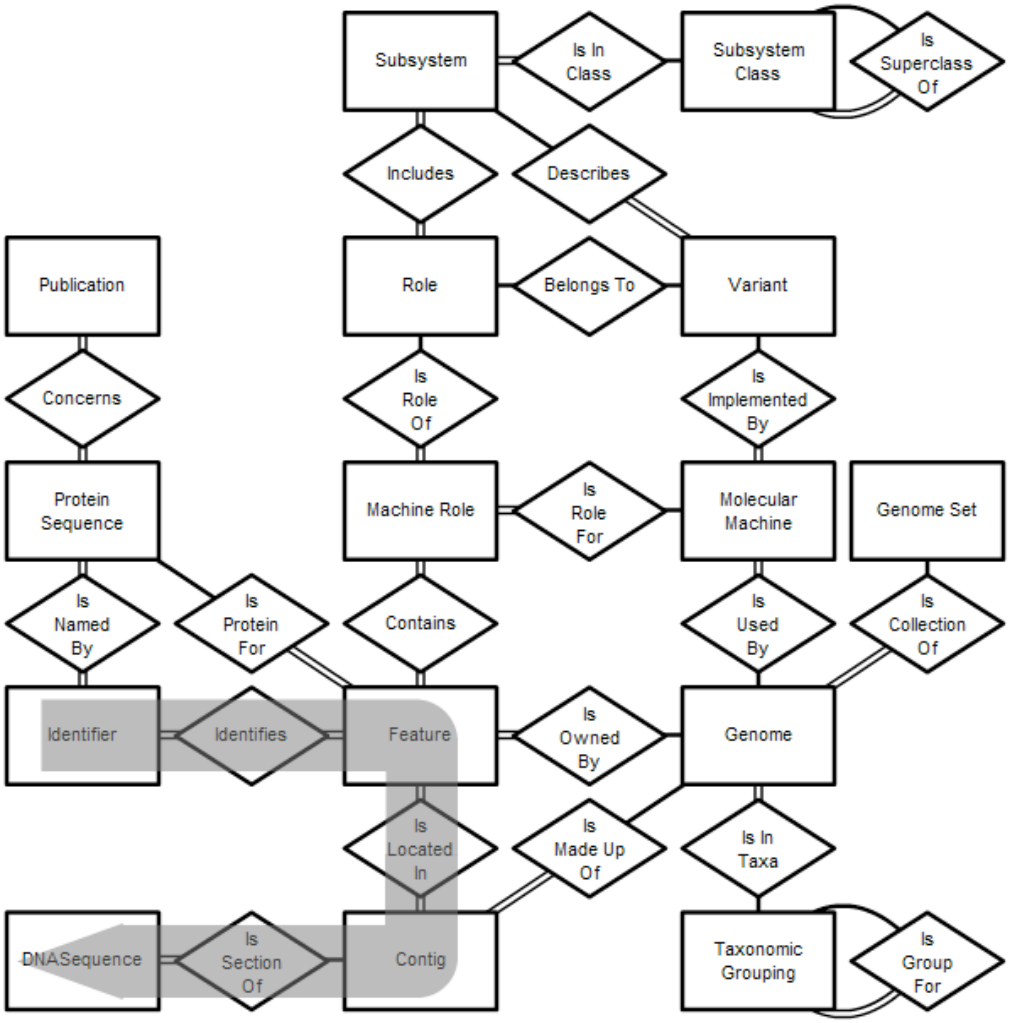
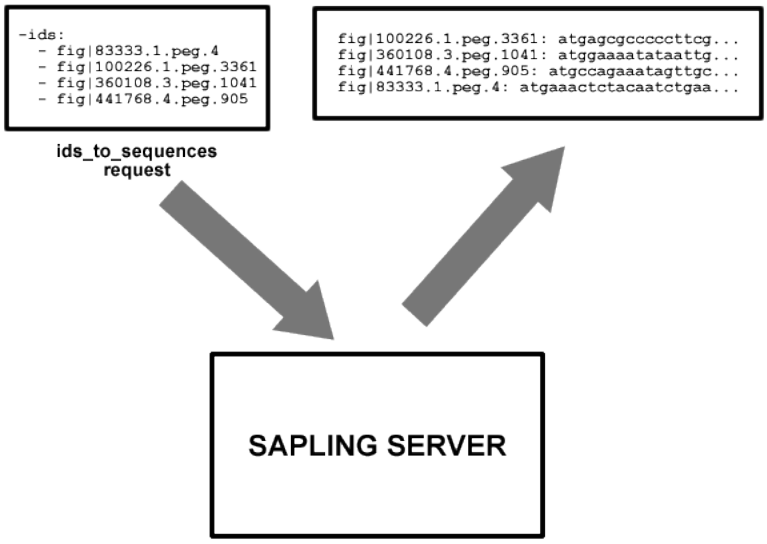

Subsystem
Is In Class
Subsystem Class
Is Superclass Of
Includes
Describes
Publication
Role
Belongs To
Variant
Concerns
Is Role Of
Is Implemented By
Protein Sequence
Machine Role
Is Role For
Molecular Machine
Genome Set
Is Named By
Is Protein For
Contains
Is Used By
Is Collection Of
Identifier
Identifies
Feature
Is Owned By
Genome
Is Located In
Is Made Up Of
Is In Taxa
DNASequence
Is Section Of
Contig
Taxonomic Grouping
Is Group For

Figure 2.

a)


-ids:
- fig|83333.1.peg.4
- fig|100226.1.peg.3361
- fig|360108.3.peg.1041
- fig|441768.4.peg.905
fig|100226.1.peg.3361: atgagcgccccttcg...
fig|360108.3.peg.1041: atggaaaatataattg...
fig|441768.4.peg.905: atgccagaaatagttgc...
fig|83333.1.peg.4: atgaaactctacaatctgaa...
ids_to_sequences request
SAPLING SERVER

b)

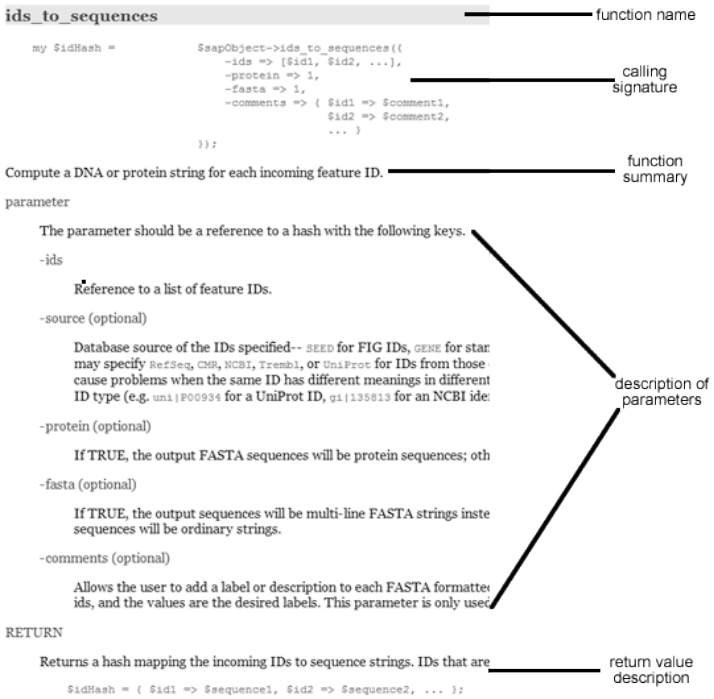

ids_to_sequences
function name
my $idHash = $sapObject->ids_to_sequences({
-ids => [$id1, $id2, ...],
-protein => 1,
-fasta => 1,
-comments => { $id1 => $comment1,
$id2 => $comment2,
... }
});
calling signature
Compute a DNA or protein string for each incoming feature ID.
function summary
parameter
The parameter should be a reference to a hash with the following keys.
-ids
Reference to a list of feature IDs.
-source (optional)
Database source of the IDs specified-- SEED for FIG IDs, GENE for star
may specify RefSeq, CMR, NCBI, Trembl, or UniProt for IDs from those
cause problems when the same ID has different meanings in different
ID type (e.g. uni|P00934 for a UniProt ID, gi|135813 for an NCBI ide
-protein (optional)
If TRUE, the output FASTA sequences will be protein sequences; oth
-fasta (optional)
If TRUE, the output sequences will be multi-line FASTA strings inste
sequences will be ordinary strings.
-comments (optional)
Allows the user to add a label or description to each FASTA formatte
ids, and the values are the desired labels. This parameter is only use
description of parameters
RETURN
Returns a hash mapping the incoming IDs to sequence strings. IDs that are
return value description
$idHash = { $id1 => $sequence1, $id2 => $sequence2, ... };

Figure 3.