Nucleic Acids Research, 2003, Vol. 31, No. 8 **2187–2195**
DOI: 10.1093/nar/gkg312

GenDB—an open source genome annotation system for prokaryote genomes

Folker Meyer*, Alexander Goesmann, Alice C. McHardy, Daniela Bartels, Thomas Bekel, Jörn Clausen[1], Jörn Kalinowski, Burkhard Linke, Oliver Rupp, Robert Giegerich[1] and Alfred Pühler[2]

Center for Genome Research, [1]Technische Fakultät and [2]Department of Biology, Bielefeld University, Bielefeld, Germany

Received October 30, 2002; Revised and Accepted February 4, 2003

ABSTRACT

The flood of sequence data resulting from the large number of current genome projects has increased the need for a flexible, open source genome annotation system, which so far has not existed. To account for the individual needs of different projects, such a system should be modular and easily extensible. We present a genome annotation system for prokaryote genomes, which is well tested and readily adaptable to different tasks. The modular system was developed using an object-oriented approach, and it relies on a relational database backend. Using a well defined application programmers interface (API), the system can be linked easily to other systems. GenDB supports manual as well as automatic annotation strategies. The software currently is in use in more than a dozen microbial genome annotation projects. In addition to its use as a production genome annotation system, it can be employed as a flexible framework for the large-scale evaluation of different annotation strategies. The system is open source.

INTRODUCTION

The process of genome annotation can be defined as assigning meaning to sequence data that would otherwise be almost devoid of information. By identifying regions of interest and defining putative functions for those areas, the genome can be understood and further research initiated. Annotation generally is thought to be of best quality when performed by a human expert. The vast amount of data which has to be evaluated in any whole-genome annotation project, however, has led to the (partial) automation of the procedure. Due to this, software assistance for computation, storage, retrieval and analysis of relevant data has become essential for the success of any genome project.

Comparison of existing tools

A number of genome annotation systems intended for the analysis of prokaryotic and eukaryotic organisms have been designed and presented in the last few years. The first generation was published in 1996 and consisted of the MAGPIE (1), GeneQuiz (2) and Pedant (3) systems. These focused primarily on generating human readable HTML documents based on tables and sometimes in-line graphics. A number of good ideas originated from this first generation of genome annotation systems, which made their way into today's systems. Examples are the intuitive visualizations provided by MAGPIE and the splitting of results by significance levels to enable comparison of different tools (also MAGPIE). Since then, a second generation of mostly commercial genome annotation systems has been published, including ERGO (Integrated Genomics, Inc.), Pedant-Pro (successor to Pedant, Biomax Informatics AG), Phylosopher (Gene Data, Inc.), BioScout (Genequiz, Lion AG), WIT (4) and the open source system Artemis (5). Some systems (MAGPIE, Artemis and Phylosopher) contain extensive visualizations or include multiple genome comparison-based annotation strategies [most notably by ERGO (6)]. With the exception of Artemis, all systems provide an automatic annotation feature. To the best of our knowledge, except ERGO, all systems use a variant of 'best blast hit' as their fixed, built-in annotation strategy. Only MAGPIE, Artemis and the newer versions of Pedant allow the integration of expert knowledge through manual annotation. (In the last few weeks, the Manatee system has been made public by TIGR. The authors have not yet had the opportunity to evaluate this system.)

The substantial commercial interest in the area of genome annotation has led to a situation where, with the noted exception of Artemis, no genome annotation system is in the public domain. Therefore, only the source code of Artemis is available for further analysis by the research community. Even in-depth technical information, such as details about the annotation strategy implemented, is very hard to obtain. This lack of access is a major hurdle when trying to evaluate these complex systems. Together with the omission of well defined application programmers interfaces (APIs), this prevents the extension of existing systems. This situation is counterproductive for science in this field: the best experts in the field have no medium to contribute their experience to the cooperative evolution of better and better annotation systems. The resulting need for a well designed and documented open

*To whom correspondence should be addressed. Tel: +49 521 106 4827; Fax: +49 521 106 5626; Email: fm@genetik.uni-bielefeld.de

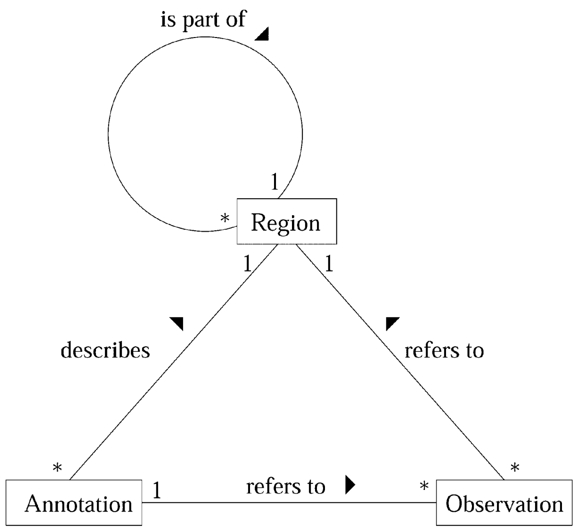

Figure 1. The core data model of GenDB in UML. Only the three central classes are shown; the classes actually represent a hierarchy of specialized objects, e.g. a BLAST observation object and an InterPro observation object.

source genome annotation system led us to develop GenDB. GenDB is a flexible and easily extensible system, which currently is in worldwide use for the annotation of more than a dozen novel microbial genomes.

As with the very successful Linux computer operating system, the open source license of GenDB enables the cooperative development of high quality software for genome annotation. The system is intended to provide a flexible, transparent infrastructure for genome projects, which other groups can adopt and modify to meet their requirements.

The 'System Architecture' and 'Implementation' sections describe in detail the GenDB software system; they are intended to enable bioinformatics scientists to evaluate the system. The next section outlines the bioinformatics methods currently implemented by the system; here the target audience is the biologist looking for a tool to annotate a genome. Finally, the section on applications is intended for a general audience to show the scope of projects in which GenDB currently is being used.

SYSTEM ARCHITECTURE

A surprising lesson learned from the analysis of the existing systems (as far as they are known to the authors) is the lack of consistent internal data representation. However, in our opinion, an internal data representation using a well defined data model is the prerequisite needed to provide an API for any larger software system.

Data model design

We chose a very simple data model, based on only three core types of objects. Regions describe arbitrary (sub-) sequences. A region can be related to a parent region, e.g. a CDS is part of a contig. Observations correspond to information computed by various tools [e.g. BLAST (7) or InterPro (8)] for those regions. Annotations store the interpretation of a (human) annotator. They describe regions based on the evidence stored in the observations. Figure 1 shows the relationships between

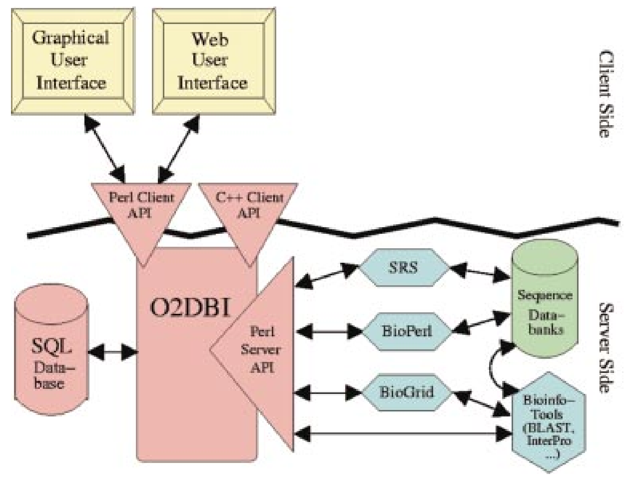

Figure 2. Overview of the GenDB system.

the different core objects. As can be seen, there is a clear distinction between the results from various bioinformatics tools (observations) and their interpretation (annotations), implemented in the data model. While this data model seems very generic, it represents a hierarchy of classes, including the complete EMBL feature set with several extensions. There are additional classes (e.g. tools and annotators) that complement the three core classes.

Since data access is via the objects described above, the classes in GenDB themselves form the API.

This object-oriented approach makes code maintenance easy, and also makes the data and methods in our system accessible to other programs. At the same time, we provide a means to extend the GenDB system.

General overview

Figure 2 illustrates the architecture of the GenDB system: the GenDB objects are mapped onto tables via O2DBI and stored in an SQL database. All access to these data via a Perl client or server API, or via a C++ client interface is again managed by the O2DBI module. On the client side, user interfaces can be implemented that use the functionality of these APIs.

On the server side, sequence databases can be accessed with the SRS (9) system or via the BioPerl (http://www.bioperl.org) interfaces. Computational intensive tools such as BLAST or InterPro can be managed and scheduled via a BioGrid (e.g. Sun GridEngine http://www.sun.com/gridware).

Plug-in architecture

As all data in the system are accessible, almost any task can be performed by a plug-in, defined as a tool that operates on the GenDB data structures. While the core GenDB system provides a mechanism for manual annotation, an automatic annotation plug-in performs automatic assignment of regions (e.g. genes) and/or functional annotation for those regions. Another example of the plug-in architecture is the inclusion of the PathFinder (10) component for the analysis of metabolic data.

Wizards

Repetitive tasks such as updating the position of every downstream gene after a frameshift correction are performed

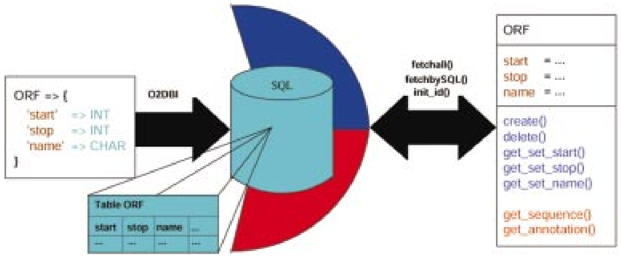

Figure 3. O2DBI maps Perl objects to relational tables, generating both SQL tables and Perl modules.

by wizards. These are software agents, modeling repetitive tasks and/or tasks that require complex and synchronized changes to several data objects. All actions performed using wizards are modeled as annotations. Currently, wizards are implemented for frameshift and sequence data correction, CDS-start correction and reload (update) of contig sequences.

IMPLEMENTATION

We chose Perl (http://www.perl.org) as implementation language using a multitude of existing Perl modules from the BioPerl project. The widespread use of Perl in bioinformatics will enable many researchers to use GenDB as a platform for their implementation of further genome analysis pipelines. Using Perl with GenDB allows the incorporation of additional tools and methods from this area of research. To be able to offer an API to the outside world, the system requires a persistent storage layer. We elected to use an relational storage backend (SQL), which provides a fast, reliable and well tested storage subsystem.

O2DBIv2 (objects to database interface)

The complexity of our system encourages using an object-oriented approach not only in designing (see Fig. 1) but also in implementing the system. While Perl offers various interfaces to DBMS systems, there was no previous tool available for the mapping of Perl objects to relational tables, applicable for our purposes. We therefore used at first the original O2DBI system (O2DBI, J.Clausen, Technical Report, Bielefeld University, 2002) which was then enhanced substantially by B. Linke as O2DBIv2 (B.Linke, in preparation) to map Perl objects automatically to relational tables. Object descriptions in UML (XMI) format are now translated into a library of Perl objects with Perl and C++ client–server bindings. All objects are stored in a relational database [e.g. MySQL (http://www.mysql.com) or PostgreSQL (http://www.postgresql.org)].

Figure 3 shows a simplified version of the role of O2DBI. Classes are described as Perl hashes (denoting objects) which are mapped to relational tables. Perl source code is generated that implements standard methods (create, delete, init, get/set, etc.) for the objects. These automatically generated object methods are stored in Perl modules. Extension of the object functionality is possible in separate Perl modules.

Interfaces

There are several ways of accessing the system, an API, user interfaces and a new client–server interface.

User interfaces. The more widely used frontend is a Gtk-Perl (http://www.gtkperl.org) based graphical user interface (GUI) that offers access to the data in the system by a variety of navigation metaphors (see Figs 4 and 5). Since not all users have access to a platform with Perl/Gtk, a web interface is also provided. The web interface offers somewhat restricted functionality with respect to the GUI. However, due to its HTML standard compliance, the web interface provides access to GenDB for a wide range of platforms.

As stated above, the GenDB classes form the API. Documentation of each class and object property or method is available on our web site. The relative simplicity of our object model, together with the documentation, have led several groups to use GenDB as a platform for their research. The web site has several sample scripts that show the functionality of the GenDB API. Using this interface, programmers are able to extract or manipulate the GenDB data objects. This allows, for example, the user to write simple Perl scripts that compute the molecular weight for every protein in a given genome and to generate a table.

SOAP interface. In addition to the Perl API, we are in the final development stages of a client–server programmers interface. This will not only allow non-Perl platforms to connect to the GenDB system, but will also allow clients to run on remote machines. We use a SOAP (http://www.w3.org/2000/xp/Group/) interface to make our GenDB API available to languages such as C++, Python or Java.

System requirements

Since one aim of the GenDB project is to provide a platform for end users and developers, the system has very modest system requirements. A Unix system with Perl, an SQL database and BioPerl are necessary. If the user wants to compute new observations with GenDB, the required tools will have to be installed on the system or have to be available via some kind of queuing system. For a complete local installation, the sequence databases used by the tools and some sequence retrieval mechanism are required. We currently use SRS and BioPerl for this purpose. Of the systems available today, only SRS provides user-friendly views on the sequence databases.

The system can be installed on a single (e.g. Linux) server or can be spread out over multiple machines, creating a client–server installation. Locally, several test and development installations exist on single CPU Linux platforms, while our production environment includes a client–server environment with a server for the frontend, a dedicated database server and a BioGrid to perform the computation of observations.

License

To provide a resource to the academic community, we distribute the complete system (including source code) to non-commercial users under an open source license. Special commercial licenses are available on request.

Documentation and availability

The complete system including the source code, documentation, a guided tour and installation instructions is available from our web site: http://gendb.Genetik.Uni-Bielefeld.DE.

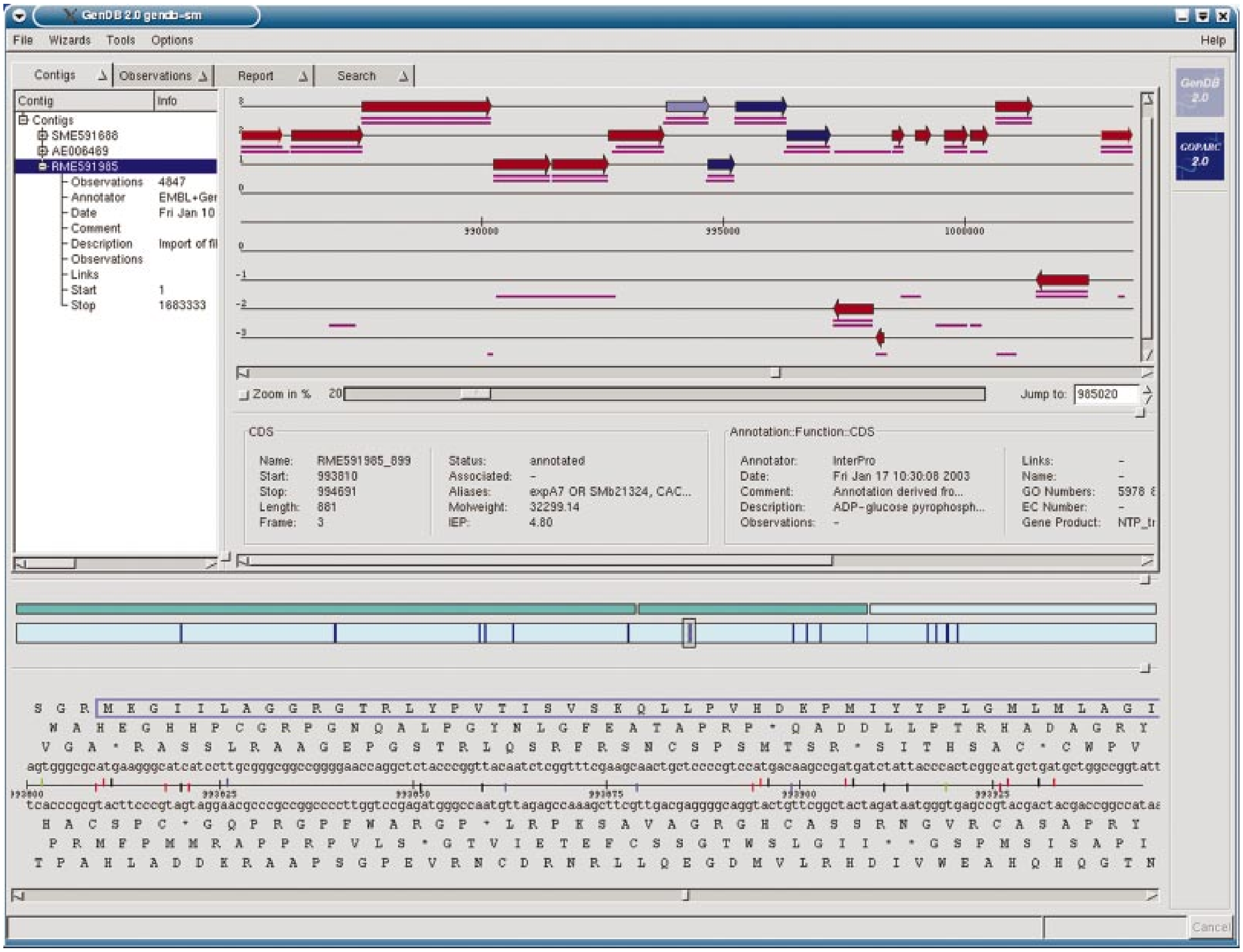

Figure 4. Main window of the GenDB system for navigation via the contig. A list in the left column allows the selection of a contig (here the pSymB megaplasmid of *S.meliloti*). The graphical overview in the top right window (RegionCanvas) displays the subregions of a selected contig (e.g. CDS, signal peptides, etc.) and computed observations of several gene predictors (here Glimmer and Critica). The RegionCanvas and the sequence browser at the bottom are synchronized. The window below the RegionCanvas contains information about the selected region or different plots (GC content, GC skew, etc.). The small contig overview in the middle can be used to display the positions of selected genes (here several genes of the metabolism of nucleotide sugars).

The documentation includes the details on the system architecture, the API and data model. An XML file describing the complete data model in great detail and hyperlinks to both versions of O2DBI can also be found on the web site.

BIOINFORMATICS METHODS

Data import and export

An important step for any genome analysis project is the availability of good import and export facilities in the genome annotation system. Currently, the GenDB system allows data import from GenBank, EMBL and fasta format files. Supported export formats are GenBank, EMBL, fasta format files and GFF (genome feature format; see http://www.sanger.ac.uk/Software/formats/GFF). A user-configurable linear or circular whole-genome view (see Fig. 5), which can be exported as a PNG file, complements the export formats. For each gene annotated with GenDB, a printable gene report can be generated.

Integration of tools

As described in the System Architecture section, GenDB allows the incorporation of arbitrary programs for different kinds of bioinformatics analysis. According to the system design, these programs are integrated as tools, which create observations for a specific kind of region. The inclusion of such tools in GenDB is very easy, with the most time-consuming step typically being the implementation of a parser for the result files. For the prediction of regions, such as coding sequences (CDS) or tRNA-encoding genes, GLIMMER (11), CRITICA (12) and tRNAscan-SE (13) have been integrated into the system. Homology searches at the DNA or amino acid level against arbitrary sequence databases can be done using the BLAST program suite. In addition to using HMMer (14) for motif searches, we also search the BLOCKS (15) and InterPro databases to classify sequence data based on a combination of different kinds of motif search tools. A number of additional tools have been integrated for the characterization of certain features of coding sequences, such as TMHMM

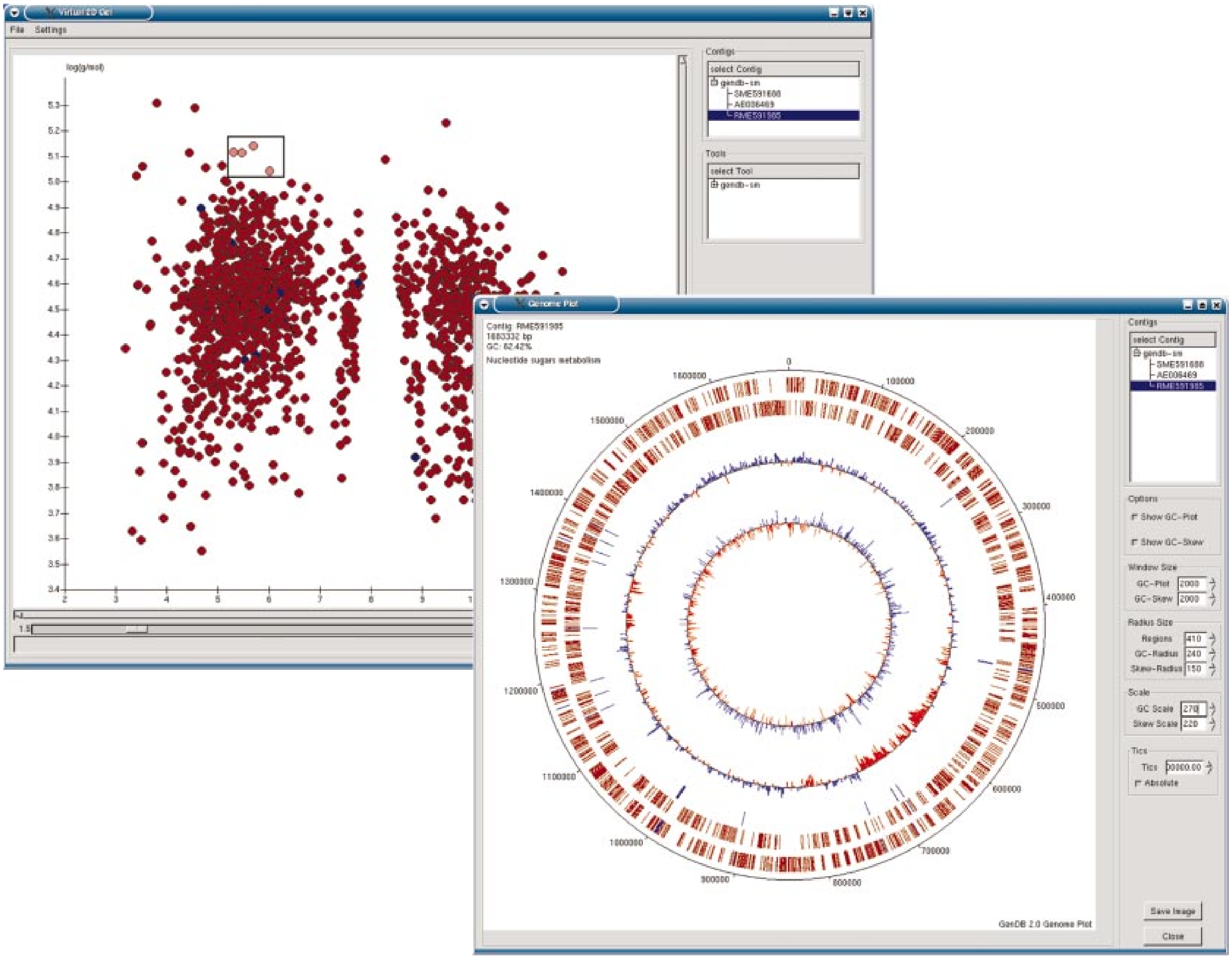

Figure 5. Visualization of a virtual 2D gel and navigation via a whole-genome representation. Highlighted spots and regions again show some genes of the metabolism of nucleotide sugars.

(16) for the prediction of α-helical transmembrane regions, SignalP (17) for signal peptide prediction, or CoBias (A.C.McHardy *et al.*, in preparation) for analyzing trends in codon usage.

Whereas some tools only return a numeric score and/or an E-value as a result, other tools such as BLAST or HMMer additionally provide more detailed information, such as an alignment. Although the complete tool results are available to the annotator, only a minimum data subset is stored in the form of observations. Based on this subset, the complete tool result record can be recomputed on demand. Storing only a minimal subset of data reduces the storage demands by two orders of magnitude when compared with the traditional 'store everything' approach. Our performance measurements have shown this also to be more time efficient than data retrieval from a disk subsystem for any realistic genome project. The computation of tool results is done via a plug-in that connects to a BioGrid using the Sun GridEngine software. The graphical user interface for the display of tool results is depicted in Figure 6. Upon selection of a certain region, all available tool results for this region are visualized in a completely customizable list. More information about the underlying database record is available by a cross-link to the corresponding sequence databases with the SRS system.

Data navigation metaphors

The design of the GenDB system allows the projection of data from any component or plug-in onto all views (see also Fig. 7). This allows the user to navigate the genome with a wide variety of synchronized views.

Annotation

As already mentioned, the GenDB data model features a strict separation of tool results (observations) and their interpretation (annotation). This confers a large amount of flexibility and enables researchers to define their application-specific annotation strategies freely. The GenDB system supports both manual annotation and the application of automated annotation strategies. For manual annotation, the user interface provides a 'one click' infrastructure; for automatic annotation, the API can be used.

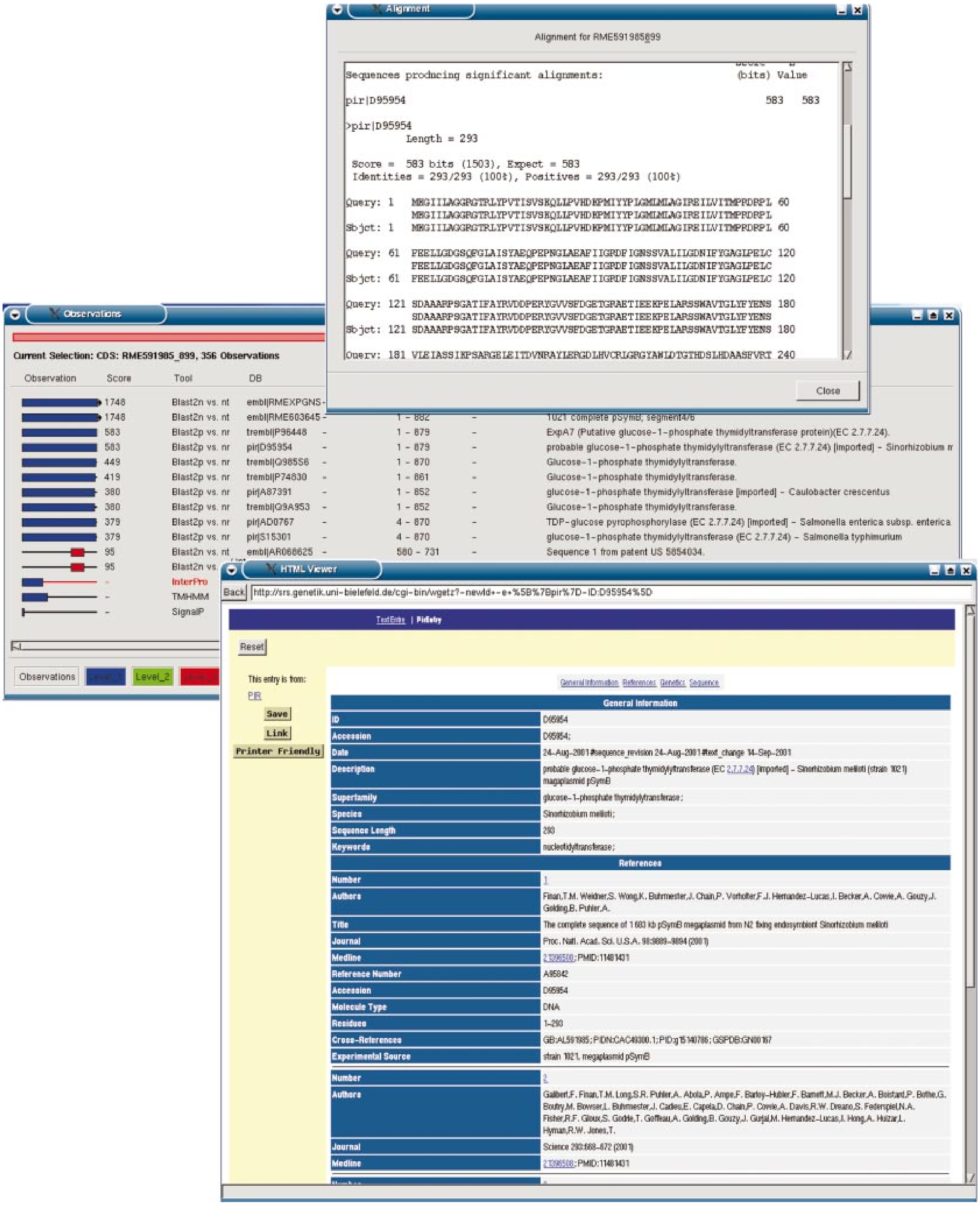

Figure 6. The observations, a single BLAST report and the underlying database record (via SRS) for a CDS region as shown by GenDB. The user can create a manual annotation by clicking on an observation.

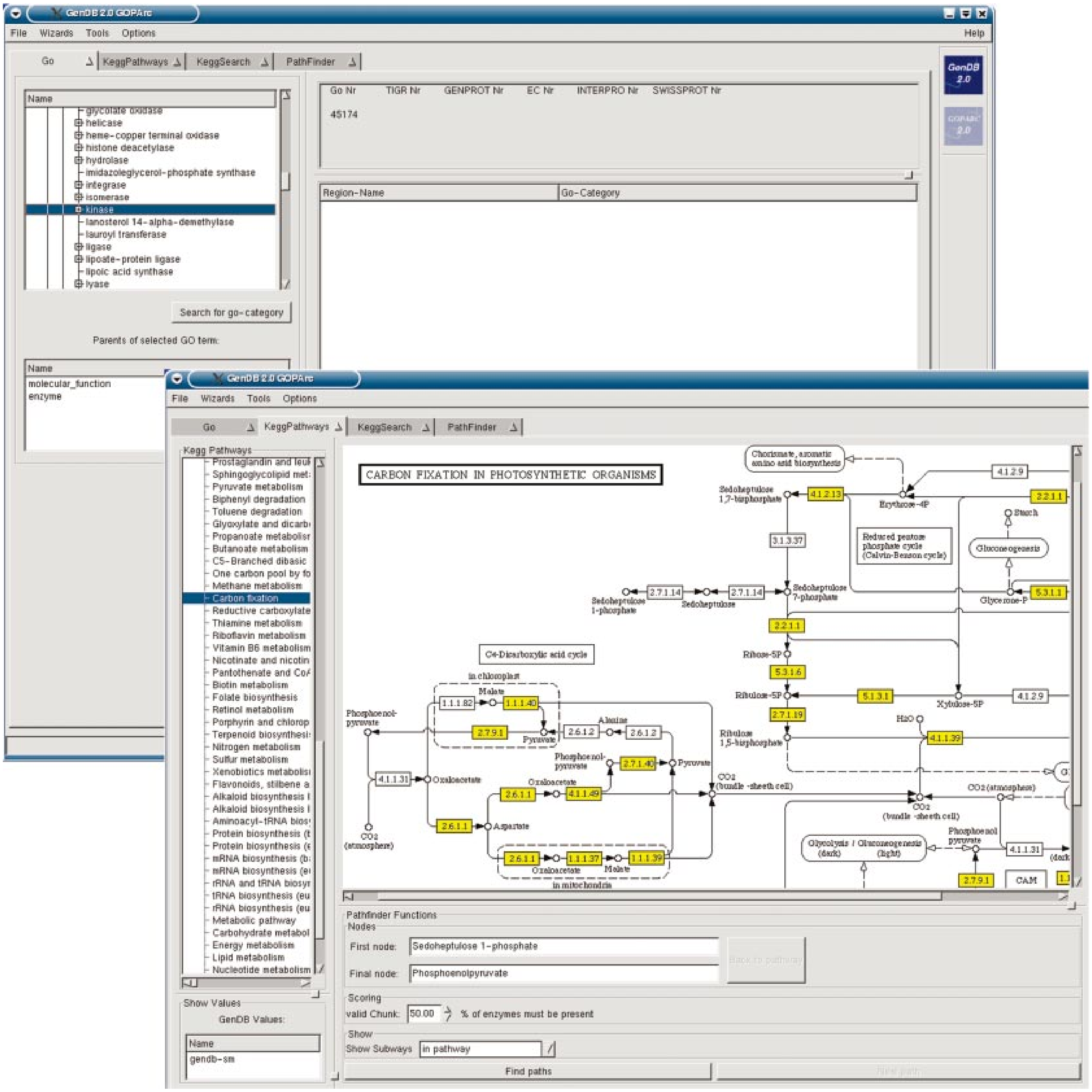

Figure 7. Data navigation via KEGG pathways (here all annotated enzymes of the metabolism of nucleotide sugars for *S.meliloti*) or gene ontologies (here identified regions for a selected GO number).

The core GenDB system offers simple automatic annotation functions which allow the application of user-defined 'best tool result' strategies. In addition to this, the GenDB-Annotate plug-in provides more complex annotation strategies based on the integration of an expert system. Here, the user can define a set of rules to be used for automatic annotation of regions, or assignment of function to those regions. Owing to the consistent, internal data representation of GenDB, all GenDB objects can be accessed directly by an expert system. While implementing a new annotation strategy currently entails writing programming code, we are in the process of establishing a graphical editor (with XML export) for editing of annotation rules and a processor for computing annotations based on these rules.

For annotation projects, the linear contig with its list of genes often is only a starting point. The knowledge about metabolic pathways and the enzymes contained in them is connected to the data in GenDB via the GOPArc (Gene Ontology and Pathway Architecture) module. GOPArc integrates our previously described PathFinder system. It is a tool for the integration of metabolic pathway and ontology knowledge into GenDB. Using O2DBI, we created an object

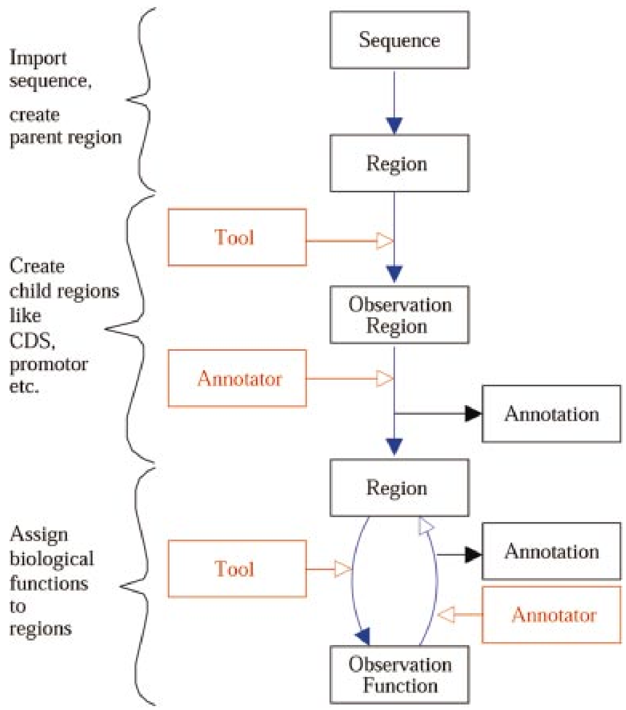

Figure 8. A sample genome analysis pipeline implemented with GenDB.

model representation of the complete KEGG database. Knowledge from other databases [e.g. MetaCyc (http://www.metacyc.org), BRENDA (http://www.brenda.uni-koeln.de)] can be incorporated. In addition to that, the system also provides access to the complete Gene Ontologies (GO) (http://www.geneontology.org) and navigation metaphors that allow browsing genomic data via the GO categories.

Annotation pipeline

Figure 8 shows an example of a genome annotation pipeline that has been implemented with GenDB. Upon import of the raw sequence data, a parent region object describing the genome sequence is created. Following this step, user-defined tools for the prediction of different kinds of regions, such as coding sequences (CDS) or tRNA-encoding genes, can be run. The output of these tools is stored as observations which refer to the parent region object. Based on these observations, an annotator, human or machine, performs 'region annotation'. This means confirming or disregarding the results of gene prediction tools by creating region objects such as CDSs or tRNAs. The annotations form a complete protocol of all 'region annotation' events. Following the creation of different kinds of regions, additional tools such as BLAST, HMMer or CoBias can be run, creating information related to their potential function. Finally, a 'function annotation' step can be performed by an annotator in which a putative function is assigned to these regions by an interpretation of the observations.

APPLICATIONS

The GenDB system can be used for the annotation of novel genomes, as a model organism database (MOD) for the curation of already annotated genomes, or as a platform for software development.

Using GenDB for genome annotation

The GenDB system has already been installed at a number of European and worldwide institutions, including the German Max Planck network. GenDB currently is being used for the annotation of a number of microbial genomes. The genomes of *Sinorhizobium meliloti* (18) and *Corynebacterium glutamicum* ATCC 13032 (J.Kalinowski *et al.*, in preparation) in addition to a large number of bacterial artificial chromosomes, cosmids and plasmids [e.g. Tauch *et al.* (19,20)] have already been analyzed with GenDB at Bielefeld University. Six novel genomes currently are being analyzed by other European groups with their own installations of GenDB. An additional five genomes (*Sorangium cellulosum*, *Xhantomonas campestris* pv. *vesicatoria*, *Alcanivorax borkumensis*, *Azoarcus* sp. and *Clavibacter michiganensis*) are analyzed by a network of German groups, which use the GenDB platform established at Bielefeld University.

GenDB as model organism database

For curation of already annotated genomes, these can be imported from EMBL or GenBank format files into the system. Any annotation information contained in these is stored in the form of GenDB objects. The data corresponding to these objects again are available via the GenDB API and user interfaces. Once there is a standard data model for prokaryote genomes (such as GMOD for the eukaryote world, see http://www.gmod.org), GenDB will be updated to support that data model.

GenDB as a platform for software development

Due to its versatility, the system is also well suited for use as a platform for novel software development, for which it has already been employed for 2 years at Bielefeld University. Recently, a number of German groups have started to implement their algorithms in the framework of GenDB, e.g. groups in the Max Planck Institutes in Tübingen and Bremen have implemented individual gene prediction strategies for their microbial genome projects using the GenDB framework.

DISCUSSION

We present a new open source platform, for both biologists and bioinformatics researchers, that implements the state of the art for genome annotation systems and enhances it in several areas. The system has been in use for 2 years at Bielefeld University and for more than a year at various other institutions. The key features of the system are its flexibility and extensibility. With respect to the genome annotation process, the system provides a flexible framework for implementing various user-defined annotation strategies, instead of relying on a single built-in annotation approach. Our past experiences have also shown the system to be well suited as an extensible platform for the integration of different kinds of functionality. It currently is used for the implementation of a system which links microarray data to gene annotation. We have implemented a wide range of metaphors for data navigation, which allow fast and easy access to different kinds of information during the genome annotation process. We hope that the positive features of the system which we provide to the research community will help to initiate research in new directions and will also be used for generating novel knowledge.

The well designed and documented API has also enabled other researchers to build their own tools based on GenDB. This proves that the main benefit of the open source approach, the cooperative development of high quality software, is already emerging. The ongoing work on GenDB is in the direction of more sophisticated automatic annotation methods. Another direction is the integration of GenDB with other programs and data sources to build a platform for systems biology.

Since only 60–70% of the genes typically found in a bacterial genome can be characterized functionally using a purely sequence-based approach, there is a clear need for adding more information to the analysis process. The GenDB system is an ideal platform to link transcriptome and proteome evidence to the genome, facilitating further analysis of previously uncharacterized genes.

ACKNOWLEDGEMENTS

The authors would like to thank all GenDB users for their time, patience and feedback that helped greatly in optimizing numerous details of the system. A.C.M. was supported by the DFG-Graduiertenkolleg 635 Bioinformatik. The work of F.M. and A.G. is financed by the BMBF grant 031U213D.

REFERENCES

1. Gaasterland,T. and Sensen,C.W. (1996) MAGPIE: automated genome interpretation. *Trends Genet.*, **12**, 76–78.
2. Andrade,M.A., Brown,N.P., Leroy,C., Hoersch,S., de Daruvar,A., Reich,C., Franchini,A., Tamames,J., Valencia,A., Ouzounis,C. and Sander,C. (1999) Automated genome sequence analysis and annotation. *Bioinformatics*, **15**, 391–412.
3. Frishman,D., Albermann,K., Hani,J., Heumann,K., Metanomski,A., Zollner,A. and Mewes,H.W. (2001) Functional and structural genomics using PEDANT. *Bioinformatics*, **17**, 44–57.
4. Overbeek,R., Larsen,N., Pusch,G.D., D'Souza,M., Selkov,E., Kyrpides,N., Fonstein,M., Maltsev,N. and Selkov,E. (2002) WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction. *Nucleic Acids Res.*, **28**, 123–125.
5. Rutherford,K.M., Parkhill,J., Crook,J., Horsnell,T., Rice,P., Rajandream,M.-A. and Barrell,B. (2000) Artemis: sequence visualisation and annotation. *Bioinformatics*, **16**, 944–945.
6. Overbeck,R., Fontstein,M., D'Souza,M., Pusch,G.D. and Maltsev,N. (1999) The use of gene clusters to infer functional coupling. *Proc. Natl Acad. Sci. USA*, **96**, 2896–2901.
7. Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
8. Apweiler,R., Attwood,T.K., Bairoch,A., Bateman,A., Birney,E., Biswas,M., Bucher,P., Cerutti,L., Corpet,F., Croning,M.D.R., Durbin,R., Falquet,L., Fleischmann,W., Gouzy,J., Hermjakob,H., Hulo,N., Jonassen,I., Kahn,D., Kanapin,A., Karavidopoulou,Y., Lopez,R., Marx,B., Mulder,T.M., Oinn,N.J., Pagni,M., Servant,F., Sigrist,C.J.A. and Zdobnov,E.M. (2001) The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Res.*, **29**, 37–40.
9. Etzold,T. and Argos,P. (1993) SRS: an indexing and retrieval tool for flat file data libraries. *CABIOS*, **9**, 49–57.
10. Goesmann,A., Haubrock,M., Meyer,F., Kalinowski,J. and Giegerich,R. (2002) Pathfinder: reconstruction and dynamic visualization of metabolic pathways. *Bioinformatics*, **18**, 124–129.
11. Delcher,A.L., Harmon,D., Kasif,S., White,O. and Salzberg,S.L. (1999) Improved microbial gene identification with GLIMMER. *Nucleic Acids Res.*, **27**, 4636–4641.
12. Badger,H. and Olsen,G.J. (1999) CRITICA: coding region identification tool invoking comparative analysis. *Mol. Biol. Evol.*, **16**, 512–524.
13. Lowe,T.M. and Eddy,S.R. (1997) tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Res.*, **25**, 955–964.
14. Eddy,S.R. (1998) Profile hidden Markov models. *Bioinformatics*, **14**, 755–763.
15. Henikoff,S. and Henikoff,J.G. (1991) Automated assembly of protein blocks for database searching. *Nucleic Acids Res.*, **19**, 6565–6572.
16. Sonnhammer,E.L.L., von Heijne,G. and Krogh,A. (1998) A hidden Markov model for predicting transmembrane helices in protein sequences. In Glasgow,J., Littlejohn,T., Major,R., Lathrop,F., Sankoff,D. and Sensen,C. (eds), *Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Menlo Park, CA, pp. 175–182.
17. Nielsen,H., Engelbrecht,J., Brunak,S. and von Heijne,G. (1997) Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Eng.*, **10**, 1–6.
18. Galibert,F., Finan,T.M., Long,S.R., Pühler,A., Abola,P., Ampe,F., Barloy-Hubler,F., Barnett,M.J., Becker,A., Boistard,P., Bothe,G., Boutry,M., Bowser,L., Buhrmester,J., Cadieu,E., Capela,D., Chain,P., Cowie,A., Davis,R.W., Dreano,S., Federspiel,N.A., Fisher,R.F., Gloux,S., Godrie,T., Goffeau,A., Golding,B., Gouzy,J., Gurjal,M., Hernandez-Lucas,I., Hong,A., Huizar,L., Hyman,R.W., Jones,T., Kahn,D., Kahn,M.L., Kalman,S., Keating,D.H., Kiss,E., Komp,C., Lelaure,V., Masuy,D., Palm,C., Peck,M.C., Pohl,T.M., Portetelle,D., Purnelle,B., Ramsperger,U., Surzycki,R., Thebault,P., Vandenbol,M., Vorhölter,F.J., Weidner,S., Wells,D.H., Wong,K., Yeh,K.C. and Batut,J. (2001) The composite genome of the legume symbiont *Sinorhizobium meliloti*. *Science*, **29**, 668–672.
19. Tauch,A., Schneiker,S., Selbitschka,W., Pühler,A., van Overbeek,L.S., Smalla,K., Thomas,C.M., Bailey,M.J., Forney,L.J., Weightman,A., Ceglowski,P., Pembroke,T., Tietze,E., Schroder,G., Lanka,E. and van Elsas,J.D. (2002) The complete nucleotide sequence and environmental distribution of the cryptic, conjugative, broad-host-range plasmid pIPO2 isolated from bacteria of the wheat rhizosphere. *Microbiology*, **148**, 1637–1653.
20. Tauch,A., Schlüter,A., Bischoff,N., Goesmann,A., Meyer,F. and Pühler,A. (2002) The 79,370bp conjugative plasmid pb4 consists of an incp-beta backbone loaded with a chromate resistance transposon, the stra–strb streptomycin resistance gene pair, the oxacillinase gene bla(nps-1), and a tripartite antibiotic efflux system of the resistance–nodulation–division family. *Mol. Gen. Genomics*, **268**, 570–584.