

BRINet: A BioResource Integration Network

Jason E. Sew Hoy¹

Alan F. McCulloch²

John R. McDonald¹

¹Department of Computer Science
University of Otago

PO Box 56, Dunedin, New Zealand 9001

²Invermay Agricultural Centre
AgResearch New Zealand

Puddle Alley, Private Bag 50034, Mosgiel, New Zealand

Email: jsewhoy@cs.otago.ac.nz

Abstract

Addressing queries interactively to a chain of two or more biological resources is an everyday task for many scientists, as they use *in-silico* methods to characterize a biological feature of interest. In this paper, we present BRINet: a software platform which provides a general, extensible framework allowing biologists to retrieve information from a complex network of resources, by specifying only the type of the input, and the type of the desired output. The core data structure is the semantic routing table, which is implemented as a sparse adjacency matrix of directly connected semantic nodes. Each entry of this matrix contains a reference to an operation driver that can execute the hop between two specific nodes, and this approach of creating purpose built drivers allows the integration of any number of heterogeneous resources. This is combined with a novel semantic routing algorithm, which calculates every possible retrieval strategy between any two types, facilitating route discovery. The result is a compact package that automates the complex, and often monotonous task of navigating through a wide range of disparate resources in search of desired information.

Keywords: BRINet, RINet, Resource Integration Network, Semantic Routing, Route Discovery

1 Introduction

In-silico query strategies often involve assembling a discrete number of database queries or web services into a chain, with the results of one such operation becoming the input for the next. A chain could be visualised as a series of hops comprising a path, or a route through a logical network of many disparate resources. Designing such a route is a daunting task in the face of a bewildering and ever-increasing universe of resources, and often requires considerable time investment in order to achieve a sensible and desirable result.

Another consideration is the repetitive nature of *in-silico* analysis. Often a given route needs to be applied to many inputs, and manual execution is then very tedious. Furthermore, this becomes quite impractical for the vast amount of input data that may be thrown up by biological experiments utilizing high throughput technologies.

While it is common to see one-off software utilities scripted to automate the execution of a particular

route, this project aims to provide a general, extensible framework that allows efficient, high-throughput route execution and automatic route discovery, using the idea of semantic routing.

We believe that an emphasis on automatic route discovery is becoming more important, as the sheer number and variety of resources available allows alternative (possibly more efficient or more informative) approaches to the *in-silico* investigation of a particular biological feature to be easily overlooked. The **BioResource Integration Network** (BRINet) software platform proposed in this project uses routing matrices to calculate all possible retrieval strategies between all integrated resources. It allows a user to view all possible routes between a specific pair of source and destination types, and it automates the retrieval of the desired information by invoking in sequence the operation drivers contained within the chosen route.

The rest of this paper is organized as follows. In Section 2, we define the notion of semantic routing, and describe how it has been incorporated into the BRINet platform. Section 3 outlines the theory behind our novel routing algorithm, and we report upon our implementation of BRINet in Section 4. Section 5 reviews some related work and, finally, we conclude in Section 6.

2 A Definition of Semantic Routing

In a physical computer network, routing is a term used to describe the task of establishing a communication path between two uniquely named hosts via interconnected intermediate hosts. This takes place within the network layer of the seven layer Open System Interconnection (OSI) reference model for network communications, using particular routing protocols and algorithms that rely upon services provided by the lower level data-link and physical layers (Zimmerman 1980).

Our definition of semantic routing builds on this theme. Semantic routing involves establishing a transition between two uniquely named types via a series of atomic transitions between interconnected intermediate types. Similarly, the semantic routing layer relies upon the services of the lower layer, which contains the operation drivers used to carry out the atomic transitions. The approach of using operation drivers is a key concept, as it allows us to define a common interface for the integration of disparate protocols and resources.

Some examples of the types that might be incorporated as semantic nodes into BRINet are: DNA sequence, RefSeq BLAST result, Swiss-Prot BLAST result, Top BLAST hit, Gene locus name, PubMed ID, PubMed reference, and PubMed neighbour reference.

The atomic operations which drive the transitions between semantic nodes could include: BLASTN, BLASTX, BLAST Result parsers, database queries, web service queries, local scripts, personal flat file or spreadsheet lookups, and URL retrievals.

Using the above semantic nodes, a user would be able to establish routes from a given input DNA sequence, to a BLAST top hit, or a set of related PubMed references, or even a gene name, via the appropriate intermediate nodes.

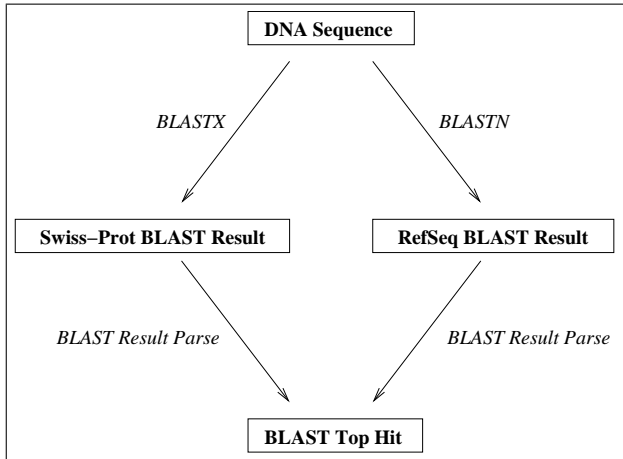


Figure 1: Two possible routes from a DNA sequence to a BLAST top hit.

There are further important similarities between routing in a computer network, and routing in a semantic network:

- There may be more than one route between two nodes. Figure 1 depicts the two possible routes from a DNA sequence to a BLAST top hit that would be discovered by an example semantic routing network.
- The node to node connections are unreliable. In a semantic network, there are a number of possible ways in which a node to node connection may fail. The most obvious reason is the simple physical unavailability of a resource or service for any number of reasons (e.g. physical network disruption). However, failure may also occur during a call to a resource or service, due to the provider having changed some aspect of the resource, such that its associated driver no longer achieves a desired outcome (e.g. modification of a web-services interface, changed password on a database).
- There is a higher level transport layer to retry alternate routes if the primary route fails. For example, a non-coding sequence BLASTed against SwissProt will produce no results, and thus the system will need to try an alternate route to reach the destination. This alternate route could involve BLASTing against a different database—such as RefSeq, which contains non-coding sequences.
- Cost or efficiency metrics can be used to rank alternative routes.
- Additional layers exist above the routing layer, as well as below it. In a computer network, the session, presentation, and application layers form the basis of the actual software interface that the user interacts with. In a semantic network, execution of a route between two types may generate a very large amount of data that need to

be organised and summarised into some sort of report, and this could be done at a higher level. For example, extracting related PubMed references from a set of DNA sequences may generate a large number of references, which can then be analysed for commonly occurring references.

3 Resource Integration and Routing

In this section, we describe the theory behind our network of integrated resources, and the novel routing algorithm which facilitates the discovery of all possible routes. The semantic network consists of named types, and the operations that effect the transition between pairs of types.

3.1 Directed Graph Representation

The relationship between named types and operations can be modelled by a directed graph. Each of the nodes in the graph represents a distinct semantic type, while the directed arcs between nodes represent the operations converting one specific type to another. Since the operations are directed arcs, the transition between nodes occurs in one direction only, and thus we can specify source and destination types for any particular operation. Figure 2 depicts an example of a directed graph containing five types and seven operations.

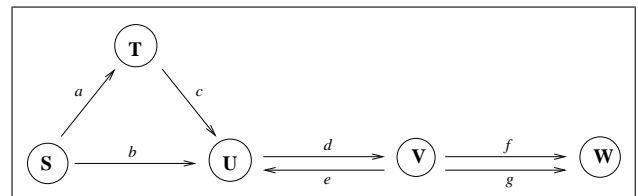


Figure 2: A directed graph showing five semantic types S, T, U, V, and W connected by the seven operations a, b, c, d, e, f, and g.

To further illustrate this representation, if type **S** represents a DNA sequence, and type **T** represents a BLAST result, then arc **a** represents one way of obtaining a BLAST result given a DNA sequence (e.g. the NCBI BLAST web service (Altschul, Gish, Miller, Myers & Lipman 1990)).

The existence of multiple arcs with the same orientation between two nodes, indicates that there are multiple methods of converting information in the specific source type to the specific destination type. For example, either of the operations **f** or **g** could be executed to move from type **V** to type **W**. If we extend the previous example to this situation, then operation **f** could be the BLAST web service, while operation **g** could be the retrieval of a stored BLAST result from a database of previous BLAST executions. If a bidirectional relationship between biological types exists, then this relationship is represented by two separate arcs.

As an operation is a single directed arc, and a route is a sequence of one or more operations, it should be clear that two routes exist if we want to get from type **S** to type **U**. One route would be simply to execute operation **b**, while the alternative route would be to execute operation **a**, followed by operation **c**. The choice of route is important in determining the exact nature of the results, but the final information will have the same semantic meaning regardless of which route is chosen.

3.2 Matrix Representation to form a Routing Table

A directed graph can be represented as an adjacency matrix, and an adjacency matrix allows the derivation of a routing table, using an adapted form of matrix algebra. The adjacency matrix shown in Figure 3 corresponds to the graph in Figure 2. As the name implies, an adjacency matrix shows all operations affecting a transition between two adjacent nodes. Thus we can see that this matrix has non-zero, non-identity entries wherever a path of length 1 exists between two adjacent nodes. These entries contain a set of one or more atomic operations, each of which could be used to execute the required transition.

		Destination				
		S	T	U	V	W
Source	S	1	a	b	0	0
	T	0	1	c	0	0
	U	0	0	1	d	0
	V	0	0	e	1	$\{f, g\}$
	W	0	0	0	0	1

Figure 3: Adjacency matrix for the graph in Figure 2. This shows all paths of length 1 between nodes.

Self-multiplication of the adjacency matrix then yields a product matrix with path-length 2, as shown in Figure 4. This new matrix allows us to determine which nodes are reachable from any given node in two or fewer hops.

		Destination				
		S	T	U	V	W
Source	S	1	a	$\{b, ac\}$	bd	0
	T	0	1	c	cd	0
	U	0	0	1	d	$\{df, dg\}$
	V	0	0	e	1	$\{f, g\}$
	W	0	0	0	0	1

Figure 4: Matrix with path-length 2, showing all routes containing 2 or fewer hops.

The scalar multiplication and addition rules for combining the sets of routes stored in each matrix element, are defined as follows:

- The scalar multiplication of two routes results in route concatenation, while the scalar multiplication of two route sets results in the cross product of those sets. For example:

- $ab * cd = abcd$
- $d * \{f, g\} = \{df, dg\}$
- $\{d, e\} * \{f, g\} = \{df, dg, ef, eg\}$

- The scalar addition of routes results in the union. For example:

- $b + ac + b = \{b, ac\}$

More generally, raising the adjacency matrix to the power N will show which nodes are reachable from a given node in N hops or fewer, and each matrix element will contain the set of alternative routes that could be used.

Eventually this repeated matrix self-multiplication converges on a stable product known as the routing table, provided there are no circular routes. By looking up the routing table for a particular source and destination node pair, we can determine whether a connecting route or set of routes exists, and also the sequence of operations that make up each route.

Matrices representing cyclic graphs present a problem, in that without appropriate measures the matrix power series will never converge. However, various measures can be taken to ensure that self-multiplication of cyclic matrices does indeed terminate. One such measure is to ensure that the matrix diagonal only ever contains identity ‘1’ values. Although matrix multiplication of cyclic matrices will attempt to define routes on the main diagonal, it clearly does not make sense to execute a route beginning and ending with the same semantic type. Another measure involves canceling out pairs of adjacent inverse operations whenever they arise during route concatenation. It is never productive to re-visit any nodes in growing a route of operations.

For each element of the matrix, a cost metric is accumulated as route concatenation occurs, and this allows alternate routes to be ranked by efficiency. This metric may be updated dynamically, as routes are executed and their actual costs are measured by some heuristic.

4 BRINet Implementation

The BRINet platform has been implemented in the Java programming language. This section describes the main component parts of the platform, which are the routing engine, the operation drivers, and the user interface. It also contains an outline of what happens during a BRINet execution, and a description of BRINet’s plug-in interface.

4.1 The Routing Engine

The routing engine forms the core of the BRINet platform, as it generates routing tables, which contain all possible routes between integrated resources. As presented in Section 3, this routing table is calculated by repeated self-multiplication of the adjacency matrix until the product becomes stable. Matrix multiplication is a resource intensive process, involving many scalar addition and scalar multiplication steps, especially if the matrix contains many nodes. In addition, many of the entries within the adjacency matrix will be zero entries containing no routes. Thus we have based our implementation of the routing engine upon sparse matrices, which store only non-zero elements thus improving the memory usage of the program. Sparse matrices also improve efficiency in calculating the routing table, since in matrix multiplication we then only have to look at those elements which are non-zero.

4.2 The Operation Drivers

As described previously, a route between two nodes is simply a sequence of operations that must be executed in order. Since there is no restriction on the types of operations and resources that can be integrated, each operation must have an associated driver object. These driver objects allow the seamless integration of operations into the network, and they all exhibit the same common behaviour. When a driver object is invoked, it reads input from the central data repository, and it determines the particular parameters that are to be used in this execution. It then applies its specific operation to the input data using the execution parameters, before writing the output of the operation back to the central data repository. It is important that an audit trail of all results is constructed, so that the entire procedure could be later verified if necessary. The structure of the central data repository is described in more detail in Section 4.4.

4.3 The User Interface

The BRINet user interface is based on the Java Swing toolkit, which provides a fully-featured user interface component library. It is structured as a tabbed pane, with separate tabs for each of the source, destination and output panels. In addition, the user interface is responsible for gathering any additional required information from the user through pop-up dialog boxes.

4.3.1 Source Tab

The Source tab allows the user to specify the input to a particular execution. If manual input is required, then the user can copy and paste one item of input into the corresponding window. Alternatively, the user can select one or more files as input, using the pop-up file browser. A drop-down box allows the user to declare the input type, and this becomes the source type for the execution.

4.3.2 Destination Tab

The Destination tab allows the user to specify the destination type for a particular execution. Once an input has been entered and a destination has been chosen, a routing diagram is drawn to show information about all possible routes between the specified source and destination types. It outlines all the intermediate types in the routes, and the specific operations that are required to get from one type to the next. The use of distinguishing colours highlights the similarities and differences between alternate routes, where multiple routes exist. If there are no routes between the specified types, a message is displayed to the user to convey this.

4.3.3 Output Tab

The Output tab shown in Figure 5 allows the user to view the results of all executions. The user can browse through all intermediate results in a logical order, by expanding and collapsing nodes in the tree data structure. Each execution of BRINet produces a timestamped directory containing all log files relevant to that execution, and each log file can be independently viewed as required. The “Show HTML Output” button retrieves the web page corresponding to a particular logfile of results, where applicable. This allows users to retrieve the exact page of output that would have been generated in a manual analysis. The “Apply Plug-in” button allows the user to select an external plug-in to run, and this function is described in Section 4.5.

4.4 Executing BRINet

If the user has selected a source/destination pair where multiple routes of execution are possible, the program prompts the user to select one specific route. A pop-up box presents the alternate routes to the user, along with their associated cost metrics. The user can then opt to execute the cheapest route, or a route of their choice.

The input, the output, and all intermediate results of a BRINet execution are stored within the BRINet central data repository, which is called a Route Data Tree. Each node of the Route Data Tree stores data of one type, and possibly other nodes containing data of the next type in the chain. This nested structure facilitates the storage of data in layers as it is generated, and it clearly communicates the flow of information from one node to the next.

The Route Data Tree also allows fine control over the structure of the execution results. As an execution proceeds, there may be steps where multiple results are generated within a node. These results may either be stored together in a single node, or separately, where each result is stored in its own node. The advantage of grouping results together in a single node is that the results are kept together and the next operation is applied to only one node. If the user chooses to keep results distinct, then the tree branches out at this node as possibly many output nodes are generated. The next operation in the chain must then be applied separately to the data in each of the output nodes. However, the advantage of keeping results distinct is that each individual end result has a lineage and its specific parents can be traced up the tree. Before an execution starts, a pop-up box allows the user to specify whether any grouping is required. If the user does not specify any grouping, then all results are kept distinct by default. If users wish to group results, then they must specify the stage at which grouping should be applied. This has the effect of keeping results distinct up to a certain point, and grouping results thereafter.

4.5 Plug-in Support

As stated in Section 2, execution of a route between two types may generate a very large amount of data that need to be organised and summarised into some sort of report. Another task that does not fit into the semantic routing layer is preprocessing of data to generate the appropriate input to a BRINet route. The inclusion of a plug-in interface allows users to accomplish these tasks both after and before a BRINet execution as required. The support of plug-ins is a significant addition, as it allows developers to write specific scripts and programs to handle the intricate preprocessing and reporting of data in a user-specific manner. Furthermore, these external scripts and programs can be written in any programming language, and this allows us to take advantage of their individual strengths—an example being the powerful text processing capability of Perl.

The plug-ins to BRINet reside in a designated plug-in directory, and each integrated plug-in must have an associated properties file. This properties file contains lines of metadata which describe how the plugin is executed, and what it does. This allows new plug-ins to be included dynamically as modules, and simply dropped into the plug-in directory without the need for any recompilation of the BRINet platform. Each property in the property file occurs on a single line and takes the form of a key:value pair. Some simple properties include:

- TITLE - the title of the plug-in.

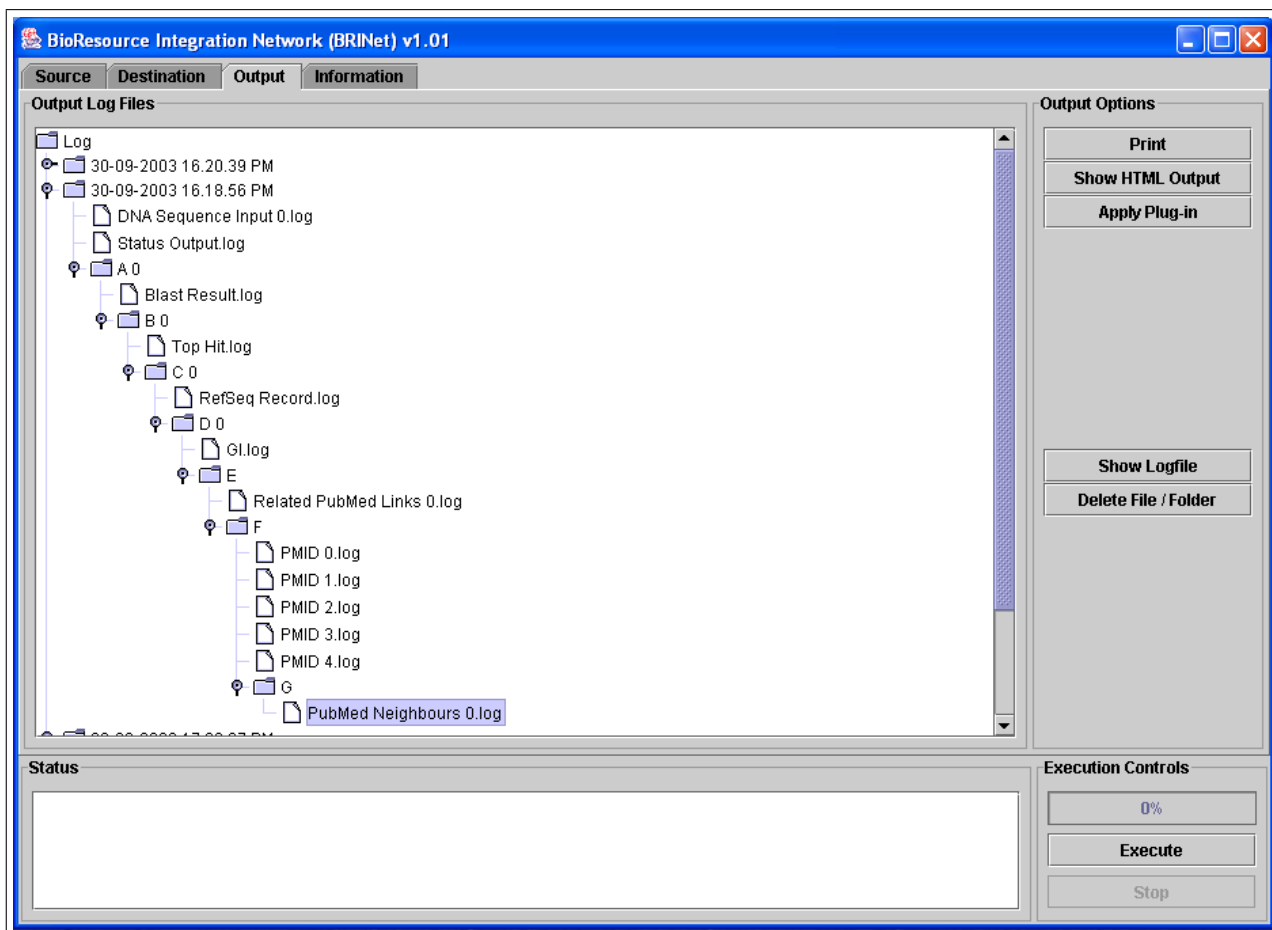


Figure 5: A screenshot of the BRINet output tab

- DESCRIPTION - a description of what the plug-in actually does and how it can be used.
- CMD - the command which should be called by BRINet in order to execute the plug-in.
- OUTPUT_FORMAT - the type of output produced by the plug-in, e.g. text or HTML. If the output of the plug-in is simply text, then BRINet can execute the plug-in and then display its results within a pop-up dialog box. However, if the plug-in produces HTML output, BRINet displays it automatically using the default web browser.

5 Related Work

There are a number of very significant projects and systems which target the area of biological data and services integration. In this section, we draw similarities and differences between BRINet and two other projects: BioMoby and myGrid. We then highlight the novel aspects of a semantic integration network.

BioMoby (Wilkinson & Links 2002) and myGrid (Stevens, Robinson & Goble 2003) are similar community projects, which endeavour to define and develop an infrastructure that will allow integration of resources offered by participating service providers. Initial design and setup cost of the infrastructure is high but, once achieved, the cost of expanding the system is simply that borne by the service provider in registering and integrating a new service. The simple semantic routing approach described in this paper does not require providers to develop to a new infrastructure. We envisage an intensively curated routing table, and a set of supporting drivers that interface

with a disparate range of resources, which may have volatile access mechanisms and external interfaces. Thus where BioMoby attempts to standardize web services on integration, BRINet aims to integrate a much wider range of resources, without requiring any assistance from service providers. The advantages of this approach are:

- the ability to cover a much wider range of resources,
- no requirement for service providers to conform to a standard framework,
- low initial setup cost,
- explicit support for fine-grained access control to proprietary nodes.

BRINet accepts that very regular maintenance of the routing tables and underlying resource drivers will be required, and this high ongoing cost is the main disadvantage of our approach.

In terms of forming experiments, myGrid regards *in-silico* experiments as distributed queries and workflows. Data and parameters are taken as input to an analysis or database service, and output from these is taken - perhaps after interaction with the user - as input to further tools or database queries. This description of a workflow is exactly the same as the concept of a route, or a sequence of operations, as proposed in this integration network. A scientist requires a great deal of background knowledge in order to build workflows efficiently and effectively from the available data sources and analytical tools. myGrid attempts to relieve some of this burden by describing: (i) the services that process objects, and (ii)

the objects themselves (Baker, Goble, Bechhofer, Paton, Stevens & Brass 1999). By annotating services with semantic ontologies, workflows can be built by matching the characteristics of services, such as inputs, outputs, tasks performed, and resources used. Thus when a user wants to compose services together, the semantic types of the services and objects can be checked for compatibility. This means that myGrid provides users with a safe method of building up their route of operations on-the-fly. However, users still have to spend a considerable amount of time planning the route that they want to execute, and at each step there is still a potentially overwhelming number of alternative choices, since there is no concept of focusing attention towards a particular destination type.

An execution of BRINet requires much less input from the user in this regard. Although it does require the user to specify a particular destination type for the execution, in a well-planned experiment this should be a definite known. The concept of route discovery is a core aspect of BRINet and we believe that the automatic calculation of all routes between the source and destination types is a novel contribution.

6 Conclusion and Future Development

BRINet is a software platform that integrates a wide variety of disparate biological resources and operations. While the concept of a workflow or a route through a vast network of resources is shared with projects such as BioMoby and myGrid, we present the use of semantic routing as a new approach to integrating resources. Automatic route discovery removes the burden associated with planning a route and, furthermore, it carries the huge benefit of finding every possible route between a given source and destination type. BRINet users are thus insulated from the complexity of linking integrated resources together, allowing them to focus their time on the important task of interpreting results.

The main task for immediate future development is to consolidate BRINet as a practical and useful platform, and this principally involves the integration of more types and operations. User testing has also identified the following areas for future development:

- Implementation of personal routing tables, allowing for proprietary nodes that may only be visible to a limited number of users.
- Gathering operation-specific execution parameters from the user, to enable support for such features as minimum significance limits for results.
- Defining more informative dynamic cost metrics that can be used to rank alternate routes.
- Enabling full support for automatic fail-over to alternate routes.

Acknowledgements

The authors would like to acknowledge and thank John McEwan (AgResearch Invermay), Nathan Rountree (Department of Computer Science, University of Otago), Anar Khan (Department of Biochemistry, University of Otago), and AgResearch Molecular Biology Unit staff (Department of Biochemistry, University of Otago) for their help in this project.

This project has been supported by the AgResearch (New Zealand) Invermay Summer Research Bursary Scheme, and a joint AgResearch (New Zealand) and New Zealand Foundation for Research,

Science and Technology Enterprise Scholarship, at the University of Otago.

References

- Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. (1990), Basic local alignment search tool, *in* 'Journal of Molecular Biology', Vol. 215, 403–410.
- Baker, P.G., Goble, C.A., Bechhofer, S., Paton, N.W., Stevens, R. & Brass, A. (1999), An ontology for bioinformatics applications, *in* 'Bioinformatics', Vol. 15, 510-520.
- Stevens, R.D., Robinson, A.J. & Goble, C.A. (2003), myGrid: personalized bioinformatics on the information grid. *in* 'Bioinformatics', Vol. 19, i302-i304.
- Wilkinson, M.D. & Links, M (2002), BioMoby: An open source biological web services proposal, *in* 'Briefings in Bioinformatics', Vol. 3(4), 331-341.
- Zimmerman, H. (1980), OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection, *in* 'IEEE Transactions on Communications', COM-28 (4), 425-432.