BIOINFORMATICS

Vol. 00 no. 00 2013
Pages 1–3

libRoadRunner: A Universal, High Performance SBML Compliant Simulator

A. Somogyi [1,*], M. T. Karlsson [2], M. Swat [1] and H. M Sauro [3*]

[1,3]Biocomplexity Institute, Indiana University, Simon Hall MSB1, 212 S. Hawthorne Drive, Bloomington, IN 47405-7003
[2]Dune Scientific, 10522 Lake City Way NE, #302 Seattle WA
[3]Department of Bioengineering, University of Washington, Seattle, WA, 98195

Version 0.75

Associate Editor: XXXXX

ABSTRACT

Summary: Here we describe libRoadRunner, a cross-platform, open-source, high performance SBML compliant simulator written in C++. libRoadRunner was created primarily to address deficiencies of existing simulators: speed of run, ease of use and ability to be used as a component in third-party applications. Additionally we focused on providing an extensible architecture with a wide variety of extension modules (such as...) that should address the needs of the biomodeling community. By JIT compiling SBML models at runtime we achieved significant performance improvements over existing tools. LibRoadRunner can run very complex and elaborate biochemical models with near real time performance. To achieve full SBML compliance we implemented robust support for event handling and provided convenience functions such as conservation analysis. We have also developed Python and C APIs which greatly facilitate reuse of libRoadRunner as a plug-able component.

Availability and Implementation: The core of libRoadRunner is written in C++ but to support linking with legacy C-code we developed additional thin layer that supports a pure C API. To enable reuse of libRoadRunner in scripting languages we provide extensive bindings for Python (and can with minimal effort also develop binding for any SWIG language, and we can compile as Managed C++ to support any .net language). libRoadRunner supports SBML via libSBML and uses CVODE for differential equation implementation and event handling. We also use NLEQ2 for solving nonlinear equations and LAPACK for stoichiometric analysis. We tested our library on Windows, Mac OS X and Linux and we provide binary packages for these operating systems. The library is open source and licensed under the Apache License, Version 2.0.

Contact: hsauro@u.washington.edu

Supplementary information: Online documentation, build instructions and source code are available at http://www.libroadrunner.org

1 INTRODUCTION

There is a long history (Bag *et al*., 2001) of researchers developing biochemical network simulators. Most are monolithic, making it very difficult to reuse the code for other applications. Recently there has been more interest in the development of reusable libraries. Most notable of these are COPASI, libSBMLSim, RoadRunner C#, SOSLib and the Systems Biology Simulation Core Library (SBSCL). Each has strengths and weaknesses. For example COPASI has excellent support for optimization but its API can be challenging to use (see https://code.google.com/p/copasi-simple-api/). LibSBMLSim passes all the tests in the SBML test suite but its API and functionality are somewhat limited and modifying its code would be challenging for those unfamiliar with FORTRAN77-style code. RoadRunner C# is written in C# which is excellent for .NET developers but less so for others. SOSLib has a reasonable API, however it does not provide conservation analysis and is no longer supported. Finally SBSCL is written in Java which makes it an excellent choice for Java developers but less so for others and it also doesn't support conservation analysis. None of these libraries support in-memory JIT compilation; all have performance issues and stability concerns (see benchmarking); many have problems tracking events except perhaps RoadRunner C#; all have a non-extensible architectures.

Here we describe libRoadRunner which is based on the redesign of a previously developed C# roadRunner [ref]. libRoadRunner is written in C++ with additional Python and C APIs. The APIs are designed with the modeler in mind, offer a wide range of functionality and provide high performance access to model and simulation data, making libRoadRunner ideal for realtime interactive environments. libRoadRunner also provides a "plugin" API making it possible to easily write extensions (See Plugins).

One of the key and novel features of libRoadRunner is use of the LLVM (formerly Low Level Virtual Machine) which is designed for real-time optimization and dynamic compilation of application software. This has allowed us to develop the fastest biochemical simulator currently available. This speed is particularly important for large simulation applications, e.g., multi-cell models where each cell runs several biochemical network submodels. Figure 1 shows the architectural overview of libRoadRunner.

2 METHODS

2.1 Features

List of features here

*to whom correspondence should be addressed

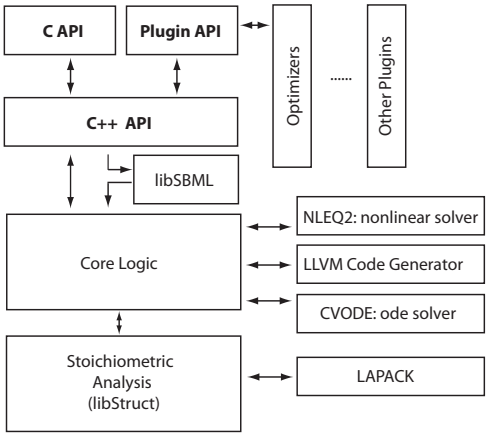

Fig. 1. Architectural Overview of libRoadRunner.

2.2 LLVM Backend

The use of LLVM allows us to directly JIT compile SBML models in memory into native machine code at runtime. LLVM can generate runtime code for a large number of architectures including ARM (for example, Android and iOS tablets and embedded devices such as Raspberry Pi) as well as GPUs. Because the configuration of the libRoadRunner LLVM-backend takes place in memory, we avoid the performance cost and hassle of converting SBML to C, writing C code files, installing and calling external C compilers to generate shared libraries and loading these libraries as RoadRunner C# does.

2.3 Extensibility

Extensibility was an importance consideration in the design of the roadRunner library as noted in the previous section. libRoadRunner is based on a component based design where all the internal components such as integrator, steady state solver, SBML compiler, etc., only communicate with each other through pure virtual interfaces. This approach allows us to develop new internal components (such as a planned stochastic GPU based integrator) without altering any existing components.

LibRoadRunner currently has two SBML compiler components: the previously mentioned LLVM compiler and another which converts SBML to C and calls an external compiler to generate a shared library at run time. The C generating SBML compiler might be useful for those who wish to compile librRoadRunner, but are using older (pre GCC 4.2 or pre MSVC 2010) C++ compilers that cannot compile LLVM.

LibRoadRunner can be extended externally through the use of the APIs (Python, C++ or C). The easiest way to add new external extensions is by writing Python scripts: this is simple but may have the disadvantage of potentially poorer performance. The alternative is to write native code (C,C++, Fortran) plugins that can be loaded on demand by the RoadRunner plugin system. (A plugin is a shared library that has a set of C functions.) A plugin API is available as a pure C API so that plugins can be written in almost any language. Any external code, including plugins, has full access to the roadRunner API which includes the ability to create new roadRunner instances. A primary advantage of plugins is that additional functionality can be developed without disturbing any of core roadRunner code. In addition, the plugin exposes a set of methods to allow generic access to any exported plugin functionality. This means that scripting languages such as Python can easily access a plugin without having to implement a new API for every plugin. Plugins also support a degree of runtime type information that allows graphical user interface applications to create user interfaces at runtime when the plugin is loaded. The roadRunner distribution comes with two plugins that illustrate how plugins can be built. These include a plugin that implements the Levenberg-Marquardt algorithm (ref), which in turn allows the plugin to fit models to experimental data; a smaller plugin can then be used to generate synthetic experimental data for testing the Levenberg-Marquardt plugin.

2.4 Python API

LibRoadrunner provides a comprehensive and *pythonic* Python extension module which is designed to have the same feel as the existing SciPy package. This is the recommended way to use libRoadRunner for most external applications. The Python API allows access to any SBML element along with a variety of calculated values using the Python dictionary protocol. It allows direct access to all libRoadRunner owned data arrays by treating them as numpy arrays. This approach allows high performance access to any roadrunner data such as simulation results without copying any data – a numpy multiarray is simply wrapped around existing data blocks. It also permits instantiations of multiple RoadRunner simulators and thus offers a very natural way of using libRoadRunner as the Reaction-Kinetics engine in multi cell simulators such as CompuCell3D (www.compucell3d.org). The Python API exposes all of the functionality of the public C++ API of the actual simulator object that the end user needs to construct sophisticated SBML-based models.

We also provide a Python binding to the libRoadRunner C API. This is useful for prototyping legacy C code which will interact with the libRoadRunner C API for testing the C API.

2.5 Performance

We compared three other simulator libraries with libRoadRunner (ref): libSBMLSim (ref), COPASI (ref) and Systems Biology Simulation Core Library (ref). We tested the ability to pass the SBML test suite, the ability to simulate very large models and the ability to simulate systems with large numbers of events. Table 1 lists the times recorded from the four different libraries based on three different types of model. The first model is a moderate sized model that tests the general performance in integration and in evaluations of the differential equations. The second model tests the ability of the libraries to scale to very large models and the third library tests the ability of the library to deal with a large number of discrete events. The timings also include the amount of time required to load the SBML model and compile it to executable code.

2.6 Implementation Details

The dedicated web site, libroadrunner.org includes all the necessary information concerning libRoadRunner as well as documentation for the Python, C++ and C APIs. Full build instructions are provided for Linux, Windows and Mac OS X. All source code is stored on

Table 1. All times are in milliseconds. Test models are available at libroadrunner.org

Application	Simulation	Large Models	Event Handling
libRoadRunner	23	45	67
COPASI	200	400	fail
libSBMLSim	450	fail	fail
SBSCL	500	800	fail

Github; documentation is generated via Doxygen and Sphinx for Python. The build system is based around CMake, Linux binaries are generated using GCC, Mac binaries via CLang Bofelli *et al*., 2000 and Windows binaries via Visual Studio 2010. Binaries are provided for Windows, Mac OS X, including binaries for Python 2.7 on Windows and Mac OS X.

2.7 Using libRoadRunner in multi-cell simulations

As a proof of concept we have successfully deployed libRoadRunner as Reaction-Kinetic solver engine in CompuCell3D. CompuCell3D is a simulation environment used to build multi-cell based models of tissues. We have used the libRoadRunner Python API to associate each CompuCell3D with a collection of libRoadRunner-based simulators. Since CompuCell3D uses Python scripting to describe its models integrating libRoadRunner was particularly easy. The clean design of the libRoadRunner Python API was helpful in designing intuitive part of the CompuCell3D Python API that deals with biochemical simulator handling. Previous versions of CompuCell3D relied on biochemical simulators that were either interpreters or relied on external C compilers, and unlike libRoadRunner they were difficult to deploy and maintain.

We ran Delta-Notch patterning simulation **Swat03** on the array of 36 adjacent cells with different reaction-kinetics models solvers and determined that libRoadRunner solvers led to 2x shorter overall simulation runtimes as compared to the previous implementations present in older version of CompuCell3D. One should keep in mind that in multi-cell simulations time spent on solving reaction-kinetics models makes up only a fraction of the total runtime. Thus 2x speedup of the overall simulation is quite significant and is representative of the high-performance focus of the new libRoadRunner.

3 DISCUSSION

In this application note we describe a new high performance SBML compliant simulator. The reason for developing libRoadRunner was to satisfy a number of specific applications that require high performance, good SBML compatibility and broad functionality. These applications include interactive desktop simulation, multicellular simulations often requiring 10,000 or more simultaneous simulations, and multi-session needs for web based interactive simulation. For the future we are planning a number of additional features including stochastic GPU integrators, parallel distributed model JIT compilation and simulation, spatial models as well as new plugins such as alternative optimizers, and a bifurcation analysis. Expand this: Currently used by: CompuCell3D, JDesigner, Tellurium, sysb.io

ACKNOWLEDGEMENT

We wish to acknowledge Frank Bergmann who together with Herbert Sauro wrote the original roadRunner library in C# and wrote libStruct, the structural analysis library along with Ravi Rao. We wish to also thank Michal Galdzicki who assisted in putting together the libroadrunner.org website. Finally we wish to thank Stanley Gu for testing libRoadRunner in his web based environment.

Funding: Work supported by the National Institute of General Medical Sciences of the National Institutes of Health under award numbers R01-GM081070. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

REFERENCES

Bofelli,F., Name2, Name3 (2003) Article title, *Journal Name*, **199**, 133-154.

Bag,M., Name2, Name3 (2001) Article title, *Journal Name*, **99**, 33-54.

Swat,M.H., Thimas, G.L., Belmonte J.M., Shirinifard A, Hmeljak D, Glazier J.A (2013) Multi-Scale Modeling of Tissues Using CompuCell3D, *Methods Cell Biol.*, **110**, 325-366.

Yoo,M.S. *et al*. (2003) Oxidative stress regulated genes in nigral dopaminergic neurnol cell: correlation with the known pathology in Parkinson's disease. *Brain Res. Mol. Brain Res.*, **110**(Suppl. 1), 76–84.

Lehmann,E.L. (1986) Chapter title. *Book Title*. Vol. 1, 2nd edn. Springer-Verlag, New York.

Crenshaw, B.,III, and Jones, W.B.,Jr (2003) The future of clinical cancer management: one tumor, one chip. *Bioinformatics*, doi:10.1093/bioinformatics/btn000.

Auhtor,A.B. *et al*. (2000) Chapter title. In Smith, A.C. (ed.), *Book Title*, 2nd edn. Publisher, Location, Vol. 1, pp. ???–???.

Bardet, G. (1920) Sur un syndrome d'obesite infantile avec polydactylie et retinite pigmentaire (contribution a l'etude des formes cliniques de l'obesite hypophysaire). PhD Thesis, name of institution, Paris, France.