EPIS: a grid platform to ease and optimize multi-agent simulators runnning

E. Blanchart and C. Cambier and C. Canape and B. Gaudou and T.-N. Ho and T.-V. Ho and C. Lang and F. Michel and N. Marilleau and L. Philippe

Abstract This paper presents the work done during the first year of the EPIS project. This project deals with the process of conducting multiple and parallel multi agents-based simulations (MABS) on a cluster or a grid in order to generate sufficient data for scientific use (*e.g.* in the case of a sensibility analysis of a simulation). We provide a new, general and user-friendly approach to marry MABS and High-Performance Computing (HPC). We, thus, propose a workflow and an associated HPC infrastructure. These two permit to easily deploy a lot of simulations on a cluster without any prior parallelizing work. The method wants to be as generic as possible: no particular MABS targeted, no overhead and HPC compliance work has to be done only once. Moreover the user is guided by a web interface that handles the workflow.

Key words: Multi-agent simulation, distributed simulation, parallelization, High-Performance Computing

Eric Blanchart
UMR 210 Eco&Sols (INRA, IRD, SupAgro), IRD, Montpellier, France

Christophe Cambier
UMI 209 UMMISCO, IRD, UCAD, Dakar, Sénégal

Clive Canape
Institut de Recherche pour le Développement (IRD), Montpellier, France

Benoit Gaudou
UMR 5505 IRIT, CNRS, Université de Toulouse, Toulouse, France

Ho The Nhan and Ho Tuong Vinh,
UMI 209 UMMISCO, IRD, Institut de la Francophonie pour l'Informatique (IFI), Hanoi, Vietnam

Christophe Lang and Laurent Philippe
LIFC, Univesité de Franche-Comté, 16 route de Gray, 25030 Besançon cedex, France

Fabien Michel
LIRMM, CNRS, Université Montpellier II, 161 rue Ada 34392 Montpellier Cedex 5, France

Nicolas Marilleau
UMI 209 UMMISCO, Institut de Recherche pour le Développement (IRD), Bondy, France
Contact author: e-mail: `nicolas.marilleau@ird.fr`

1 Introduction

In [16], Shannon identifies twelve working steps which should be present in any simulation process: from (1) problem definition to (12) results reporting and model documentation. As Shannon explains, efficiently achieving these steps mostly relies on using good scientific and engineering practices inside the simulation team, so that he argues on *the art and science of simulation*. Still, considering Shannon's proposal, there are at least two steps which success implicitly relies on being able to perform numerous simulation runs: (10) ***Experimentation***. *Executing the simulation to generate the desired data and to perform sensitivity analysis*; (11) ***Analysis and Interpretation***. *Drawing inferences from the data generated by the simulation runs.*

In this respect, whatever the quality of a simulation team, having (1) enough computing resources and (2) the ability of using them to produce a sufficient number of runs is a critical issue considering the success of a simulation study. Indeed, in most cases, numerous runs should be done to study the different aspects of a simulation model (robustness, sensitivity, impact of initial conditions, output statistical analysis, verification and validation, etc.).

This is especially true about the modeling and simulation of Multi-Agent Systems (MAS). Because they describe the trajectory of each agent, MAS models embed dynamics that could lead to gigantic solution spaces, even when only few parameters are used to describe the system. Parunak discusses this issue in [12] and proposes a very interesting heuristical approach that explores several model trajectories with only one run. Nonetheless, due to their intrinsic complexity, Multi-Agent Based Simulations (MABS) rely on models which design process involves a large exploration of the parameter domains, especially with respect to calibration, tweaking, testing, verification and validation. This requires a lot of computing resources.

Despite the interest of exploring MABS models through multiple simulation runs, this work is seldom done. Meanwhile, High Performance Computing (HPC) is today a hot topic and many computing resources are actually available through grids or clusters of computers. So, there is obviously a major issue considering the use of HPC by MABS. Until now, MABS using HPC mainly focus on speeding or scaling up MABS relying on implementations designed so that they are already compliant with HPC (*e.g.* [1]). Still, almost all the MABS platforms are not HPC compliant. In such cases, one has to first work on how to deploy his regular MABS on a HPC architecture. This requires HPC programming skills and could thus be a hard task if planned on the fly. So, practitioners often do not even consider this opportunity and translating regular MABS into HPC compliant ones is still a major issue.

Addressing this issue, efforts have been done in the scope of the RePast platform [11]. [10] proposes a middleware that allows the distribution of RePast sequential models. [4] also uses a middleware approach together with an Aspect Oriented Programming (AOP), again in order to minimize *code intrusions* in the original model.

Such approaches are of interest but have some drawbacks: (1) even if they try to be generalizable, their solutions are strongly related to the RePast architecture, (2) they add an overhead to the simulation process as they rely on catching particular

events during the execution of the models, and more importantly (3) even if light-weighted, the HPC translation work has to be done for each new model.

In this paper, we address these drawbacks by using a more general approach. Linking MABS and HPC, the idea is to work at the platform level rather than the model level. So, our proposal relies on two main features: A workflow and a HPC infrastructure supporting it. The workflow is intended to guide MABS platform developers so that they easily make their platform compliant with the proposed HPC infrastructure. The main idea is that the infrastructure will only be a means to easily deploy a lot of simulation runs over a cluster of nodes, without any prior parallelizing work. So, (1) our approach does not target a particular MABS platform, (2) there is no overhead as it is only about deploying multiple model instances and (3) the HPC compliance work has to be done only once as it works at the platform level.

The outlines of this paper are as follows. The two first sections present the context of our work. Then an overview of the framework is proposed. The two last sections are focused on the two major parts of the framework. We illustrate our framework with the example of the SWORM simulator [2].

2 SWORM simulation

The aim of the Sworm (Simulated WORM) project is to identify soil functioning by studying soil biota (microorganisms, fauna and roots) evolution. In this context, a work (presented in [2]) aims at reproducing the earthworms influence on the soil structure and the nutrient availability by simulation. For this purpose a MAS model has been developed. To model this system, a model based on a fractal and Multi-agent system has been created: the fractal theory was chosen to model such a real complex environment; the MAS allows to simulate situated agents (*e.g.* earthworms) in a virtual world (*e.g.* soil) represented by a fractal in the MAS environment.

The Sworm model has been implemented in a simulator to get results. Despite of many and many code optimizations, a sworm simulation spends about one week to give interesting results. This long duration result is due to the nature of the studied complex system: an heterogeneous multiscale system in which evolve various entities (earthworms, microbes, bacterias) in a 3 dimensional environment. Due to the time between the start and the end of a simulation, it is almost impossible to realize a complete analysis of the model without using a cluster or a grid. Note that this problem is not specific to Sworm, it occurs in many models and simulators.

3 Overview of grid computing framework dedicated to simulation domains

Due to the needs for using powerful calculator to play experiments, several researches, started few years ago, try to simplify the access to Grid. These works

provides algorithms [8], architectures [9, 18] or frameworks [13, 5] allowing to distribute experiments and/or simulators on several processors.

Two main approaches can be used to reduce experiment computation time to get results by taking advantage of the power of a grid or a cluster. The first one intends to *parallelize experiment plans* on a grid. The second one intends to *distribute a unique simulation* on a grid. We develop below the first approach which is up to now the only one used in the EPIS project. For a description of the second one, readers can refer to [15, 6].

An experiment plan is composed of a set of simulations qualified by a set of valuated parameters. Parallelizing a plan aims at deploying and running its simulations on several nodes (processors). For example, if a Sworm experiment plan defines thousand simulations to be run on a grid (composed of 5 nodes), two hundred simulations could be affected to each node. Each simulation is independent of each other (*i.e.* simulations does not exchange any data). Note that, this approach is rather simple to apply. A real gain can be observed if many simulations are needed: this approach does not reduce the compute duration of one simulation, it permits only to take advantage of grid architecture to run simultaneously several simulations.

Several works try to provide frameworks that allow to parallelize one simulation. In this context OpenMole [14] decomposes a simulator into few independent modules. The schedule of these modules is organized according to a workflow defined by a script. This script written in Groovy language is interpreted by the OpenMole framework which starts and manages the simulation. This work focuses on simulator optimization and forget user aspect. On contrary, projects such as SimExplorer [3] (an extension of OpenMole) or GPGCloud [7] take a particular attention to the user interface. These works try to develop a user-friendly graphical user interface permitting to define experiment plans and to execute them on a Grid. Nevertheless, these tools suffer of either a lack of genericity or a lack of simplicity. For example, the use of SimExplorer needs to setup the software and to have skills in Groovy. Moreover GPGCloud does not allowed to run simulators coming from another platform.

The aim of our work is to provide a platform associating genericity with simplicity. It is a user friendly portal (dedicated to non-computer scientists) allowing grid computing, especially parallel simulation running without the complexity of the grid using. The grid is also hidden behind a web access.

4 Epis overview

4.1 Using workflow

Let consider a modeler faced to a complex problem, such as the one tackled by the Sworm model. He will thus implement his own simulator or reuse the existing Sworm, depending on his needs, on his own computer. To have significant results, he wants to play many and many experiments of the model. Due to the complexity

of the studied system, the simulator needs a huge computer power to give intended results (*e.g.* 1 week is needed to play the simple Sworm experiment).

The modeler thus wants to get benefits from the computational power of the cluster to explore the influence of the simulation parameters. He connects to the EPIS portal through its preferred web browser. Then, he selects a simulator (*e.g.* Sworm) in a list and creates a new experiment.

Note that users can plug their own simulators in the portal: (i) if the new simulator is based on a famous agent based framework such as NetLogo [19] or GAMA[17], a simple upload of simulator files is needed because these frameworks are supported by EPIS; in contrarily (ii) if the new simulator is ad-hoc, the modeler has only to develop a driver allowing the simulator control respecting an interface defined by the EPIS Framework.

After creating the experiment, the modeler has to specify his experiment plan by defining the ranges of parameters, constraints, outputs and so on. In order to make the configuration easier for any researcher, we have developed a user-friendly web interface dedicated to drive the modeler through the definition of an experiment plan. Figure 1 summarizes the data exchanges. The interface allows the modeler to determine the variation domain of each parameter of the simulator and which simulation outputs he wants to observe. An illustration of that is, for the Sworm simulator, to identify a fixed population of agents representing earthworms and the range of soil parameter values that determines different kinds of soil. This kind of experiment plans can exhibit the impact of soil structure on earthworm dynamics and behavior.

Once the user has clicked on the “send” button, the interface produces, from the modeler’s experiment plan, an XML file describing all the simulations that must be launched. Then the web server calls a script (in an EJB component on the application server) to transform this XML file into an SGE file (file that can be executed on the cluster). These two files are then sent via an SSH tunnel to the cluster and the jobs are launched. Once the jobs are performed, the output XML files are stored on the application server and the web server allows the user to download the results.

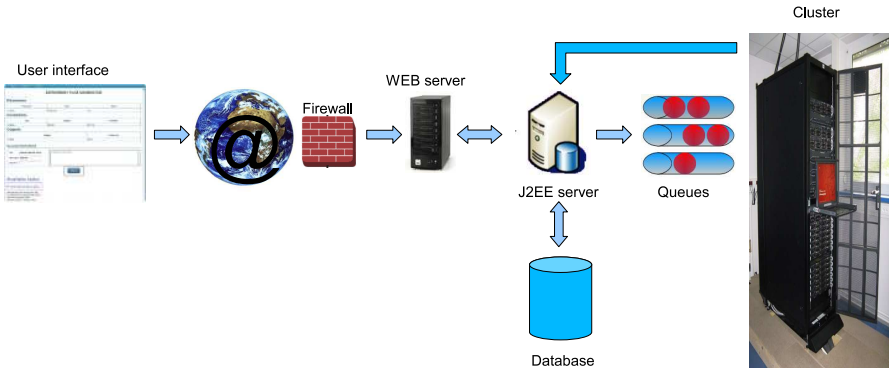

Fig. 1 Workflow of the EPIS project

4.2 Framework architecture

As shown on Figure 1, the proposed framework is based on standard components: web server, application server, database, and queue manager for the cluster.

As usual, the web server (a Tomcat server[1]) is in charge of the presentation part of the framework. The presentation part covers all the web pages needed to download the model, to give the set of studied parameters, to start the simulation and to return the results to the user. The download pages are generic to all simulators. They provide an interactive way to download the model or the simulator. The parameter pages are specific to a simulator. They are dynamically generated from the data collected on the model or simulator definition.

The application server (a Jonas server[2]) is the core of the framework. It is in charge of uploading the data from the web server, of recording them in the database, of starting the simulations on the cluster, of gathering the results from the simulator runs and of presenting them to the user. Aside from the web server, the application also provides an access for heavy clients, *i.e.* with a web service interface, presented in Section 5.

The database contains all the data needed for the simulation runs: the model (or the simulator if the platform is not supported by the portal), the description files, the parameter files and the resulting data. Conceptually a model is stored with its name, description, identifying number, a number identifying its simulation platform, the set of its parameters and the set of its outputs (practically these two sets are stored in their own tables).

The cluster part is in charge of submitting the runs to the queue manager. Most of the exchanges between the application server and the cluster part are done remotely, from the application server, by using remote commands as *scp* or *ssh*. As several runs will be issued from one parameter set, a global management of the job submissions must be performed. The cluster is indeed a resource shared between several users and it is not allowed to start a too big number of jobs at a time. There a risk that the job instances occupy all the nodes, leaving no processing power for the other users. So the job submission is limited as presented in Section 6.

5 A web cluster access

The portal provides three main web services: it allows the modeler to (i) install his simulator, (ii) design an experiment plan for a chosen simulator and (iii) get the experiment results/outputs (details on Figure 2).

Several interfaces are supported by the web server to implement the simulator runs. The first interface is generic. It proposes functions to upload simulators (on platforms supported by the portal). Users can choose the simulator file to upload

[1] `http://tomcat.apache.org/`

[2] `http://wiki.jonas.ow2.org`

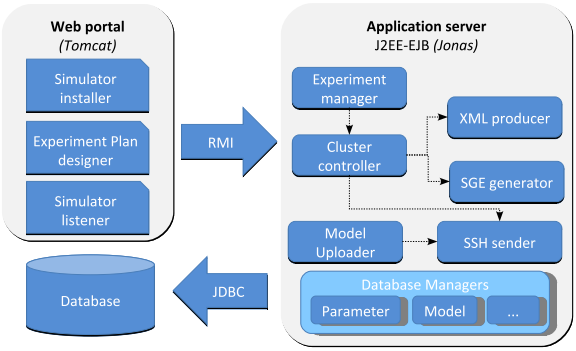

Fig. 2 Details of the web service provided by the portal

(if the simulator needs several files, he has to zip them, the Application Server will unzip it on the server). He then describes the parameters (name, type...) and the outputs of the simulator. After validation, the model, its parameters and its outputs will be stored in the database (via relevant `Database managers` session beans components). On the cluster, the supported platforms are installed as standard libraries *i.e.* in the software directory. Once the user has uploaded its model or simulator file on the application server (via the `Model Uploader` component), the later copies it in the user's directory on the cluster (via the `SSH Sender` component). To provide the user with a clean execution environment, before starting the runs, we use modules to set all the configuration variables needed. All the parameters of the runs are set in the SGE script file (see Section 6).

The main web service provided by the portal is the possibility to design an experiment plan. As presented above, once a simulator has been chosen, the modeler can define the variation domains for the simulator parameters and the outputs that he wants to observe. After validation, the plan experiment (with outputs, constraints on parameters...) are stored in the database with `Database managers`. Then via the `Cluster controller`, an XML file will be produced describing all the possible simulations (with the `XML producer`). This file will be used to generate the SGE file (using the `SGE generator`). These two files are then sent to the cluster via an SSH tunnel (with the `SSH Sender`).

The third service provided by the portal, that is the management and analysis of the outputs, is for the moment limited to the download of the files containing output data of the simulation. To this purpose, the web server queries the database via the `Database Manger` related to the experiment plan outputs.

6 Cluster execution

Computing centers provides academic users with computing resources that they are not able to afford on their own. To use these resources they must submit their jobs

(or runs) to a resource manager. The submission request must specify the resources needed for execution (number of cores and memory), the expected running time and the queue that will be used. Then, depending on the priority of the job and the resources availability, the resource manager will start the job on one of the cluster nodes. In our case the cluster used has 76 nodes for a total of 700 cores and peak performance of 7 Tflops. The resources manager installed on the cluster is SGE (Sun Grid Engine).

An experiment plan usually corresponds to several runs of the same simulator with different parameters. As the number of runs in a plan may be huge it is not possible to simply submit the whole set of runs in a cluster queue. The reason is that our jobs are not parallelized, they just use one core, and they may cause starvation for parallel jobs using a large number of cores. It is however possible to limit the maximum number of job submitted at the same time as presented on table 3. The resource manager first starts 20 runs out of 100 at the same time. Then it starts a new job as soon as one of the running jobs finishes.

```
#!/bin/bash -l
#$ -q normal2h
#$ -t1-100 -tc 20
#$ -o \$JOB_NAME.\$JOB_ID.out
#$ -e \$JOB_NAME.\$JOB_ID.err
./epis input\$SGE_TASK_ID.xml output\$SGE_TASK_ID.xml
```

Fig. 3 Exemple of SGE script to run a experience plan

From the environment variables and the directory paths which are specific to this run, the application server generates a SGE script and simulation parameter file. Parameter files are, in fact, XML files differentiated by `$JOB_ID`. These file determine (see figure 4): (i) the used simulator (*e.g. driver="Simulator.Sworm"*); (ii) the final step (*e.g. finalstep="1000"*); (iv) parameters value (*e.g. nbOfAnomala - representing the number worm at the begining*); (v) outputs and their framerate (*e.g. 2DDisplay-15 - containing soil snapshot at the depth 15th of a soil volume*)

```
<?xml version="1.0" encoding="UTF-8"?>
<Simulation id="2" driver="Simulator.Sworm" finalstep="1000">
 <Parameters>
  <Parameter name="nbOfAnomala" type="INT" value="200" />
  <Parameter name="assimRateAnomala" type="FLOAT" value="9.0f" />
  </Parameters>
 <Outputs>
   <Output id="2" name="SoilOrganicMatter" framerate="1"/>
   <Output id="3" name="2DDisplay-15" framerate="10"/>
 </Outputs>
</Simulation>
```

Fig. 4 Extract of xml input file needed for sworm simulation

Then, the application server sends the SGE script and XML input file to the front cluster node (the only one available outside the cluster) and submits the job. Because of the cluster local NFS filesystem, SGE and XML files are available for each computing node. When enough computing nodes are free, the SGE script is run. It implies that the epis module (`./epis`) is started and associated with a different XML input file on each computing node.

Each *epis* process also reads their attributed input file. It loads the selected simulator, setups the simulation according to chosen parameters and plays the simulation step by step. Between each step, output values are stored into a XML output file. When the simulation is achieved, the process is destroyed and a new epis process is started, and so on.

During the simulation and at the end, users are able to download results files. Thus, they can check the state of the simulation and stop it if the generated results are enough or if there is a problem.

7 Conclusion

The EPIS framework is an interesting solution giving a simple access to high performance computing. It allow scientists to get earlier their results or to play bigger experiments.

Without prior knowledge in computer sciences, users are able to determine experiment plans, and to play it on a grid or a cluster. They are guided by a web interface that handles a simple workflow (from the simulator upload to the parallel running of simulations).

The presented platform is based on technologies coming from Web and distributed systems. The framework is based on a J2EE application server and a SGE queue manager for the cluster. The J2EE application server aims at managing web portal and controlling the cluster and jobs (simulation) pushed in cluster queue.

We propose a modular and extensible architecture permitting, up to now, to: (i) select a simulator; (ii) create an experiment plans; (iii) start a huge number of simulations on a cluster, (iv) upload and set up new simulators. Tomorrow, this architecture will be improved by adding new modules dedicated to, for example, experiments result analysis. In addition, users are able to deploy and try themselves simulators they have developed. So, more simulators will be plugged in the portal and offered to scientific communities.

But the major contribution of the EPIS project (maybe done during its second year) will be in the domain of agent-based distributed simulation. Up to now, we focus on the experiment plan parallelizing. Another interesting way intends to distribute a single simulation over several nodes of a cluster. A such simulation needs to propose specific simulator architecture and algorithms ensuring the clock synchronizing, the MAS environment coherency, and so on. Some results have been proposed in for example [6] and [15]. But, this way must be investigated further.

References

1. Aaby, B.G., Perumalla, K.S., Seal, S.K.: Efficient simulation of agent-based models on multi-gpu and multi-core clusters. In: Simutools '10: Proceedings of the 3nd International Conference on Simulation Tools and Techniques/ OMNeT++ 2010 Workshop (2010)
2. Blanchart, E., Marilleau, N., Chotte, J., Drogoul, A., Perrier, E., Cambier, C.: SWORM: an agent-based model to simulate the effect of earthworms on soil structure. European Journal of Soil Science **60**(1), 13–21 (2009)
3. Chuffart, F., Dumoulin, N., Faure T. Deffuant, G.: Simexplorer: Programming experimental designs on models and managing quality of modelling process. International Journal of Agricultural and Environmental Information Systems (IJAEIS) **1**, 55–68 (2010)
4. Cicirelli, F., Furfaro, A., Giordano, A., Nigro, L.: Distributed simulation of repast models over hla/actors. In: S.J. Turner, D. Roberts, W. Cai, A. El-Saddik (eds.) 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, Singapore, 25-28 October 2009, pp. 184–191. IEEE Computer Society (2009)
5. Gutknecht, O., Ferber, J., Michel, F.: Integrating tools and infrastructures for generic multi-agent systems. In: fifth international conference on Autonomous agents, AA 2001, pp. 441–448. ACM Press (2001)
6. Hassoumi, I., Marilleau, N., Lang, C.: Mise en place et évaluation d'un algorithme de répartition de charge pour les plateformes de simulations distribuées basées sur les systèmes multi-agents. In: Journée Francophone des Systèmes Multi-Agents : Défis Sociétaux, pp. 85–94. Madhia, Tunisie (2010)
7. Kato, Y., Yamaki, H., Asai, Y.: Gpgcloud: Model sharing and execution environment service for simulation of international politics and economics. In: J.J. Yang, M. Yokoo, T. Ito, Z. Jin, P. Scerri (eds.) Principles of Practice in Multi-Agent Systems, *Lecture Notes in Computer Science*, vol. 5925, pp. 616–623. Springer (2009)
8. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978). DOI http://doi.acm.org/10.1145/359545.359563
9. Lees, M., Logan, B., Theodoropoulos, G.: Distributed simulation of agent-based systems with hla. ACM Trans. Model. Comput. Simul. **17**(3), 11 (2007)
10. Minson, R., Theodoropoulos, G.K.: Distributing repast agent-based simulations with hla. Concurrency and Computation: Practice and Experience **20**(10), 1225–1256 (2008)
11. North, M., Tatara, E., Collier, N., Ozik, J.: Visual agent-based model development with Repast Simphony. In: Agent 2007 Conference on Complex Interaction and Social Emergence, pp. 173–192. Argonne National Laboratory, Argonne, IL, USA (2007)
12. Parunak, H.V.D.: Pheromones, probabilities, and multiple futures. In: T. Bosse, C. Jonker, A. Geller (eds.) MABS 2010, 11th Int. Workshop on Multi-Agent-Based Simulation (2010)
13. Quesnel, G., Duboz, R., Ramat, E., Traoré, M.K.: Vle: a multimodeling and simulation environment. In: SCSC: Proceedings of the 2007 summer computer simulation conference, pp. 367–374. Society for Computer Simulation International, San Diego, CA, USA (2007)
14. Reuillon, R., Chuffart, F., Leclaire, M., Faure, T., Dumoulin, N., Hill, D.: Declarative task delegation in openmole. In: High Performance Computing and Simulation (HPCS), pp. 55–62. Caen, France (2010)
15. Sébastien, N.: Distribution et parallelisation de simulations orientées agents. Ph.D. thesis, University of La Réunion (2009)
16. Shannon, R.E.: Introduction to the art and science of simulation. In: WSC '98: Proceedings of the 30th conference on Winter simulation, pp. 7–14. IEEE Computer Society Press, Los Alamitos, CA, USA (1998)
17. Taillandier, P., Drogoul, A., Vo, D., Amouroux, E.: GAMA: a simulation platform that integrates geographical information data, agent-based modeling and multi-scale control. In: PRIMA 2010. India (2010). To appear
18. Vangheluwe, H.: Devs as a common denominator for multi-formalism hybrid systems modelling. In: IEEE (ed.) Computer-Aided Control System Design, CACSD 2000, pp. 129–134. Anchorage, AK , USA (2000)
19. Wilensky, U.: NetLogo (1999). URL `http://ccl.northwestern.edu/netlogo/`