SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2007; **37**:1289–1318
Published online 1 February 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe.806

Evolution@home: observations on participant choice, work unit variation and low-effort global computing

Laurence Loewe*,†

Institute of Evolutionary Biology, School of Biological Sciences, University of Edinburgh, Edinburgh EH9 3JT, U.K.

SUMMARY

Many simulation efforts in ecology and evolutionary biology employ individual-based models that are well suited for including many biological details. These models often pose serious computational challenges if all biologically interesting parameter combinations are to be explored. The challenges are even greater for biologists who often lack supercomputing facilities and the manpower for implementing complex global computing systems such as SETI@home. Under such limiting conditions, evolution@home started as a one-man effort to distribute simulations of Muller's ratchet to Internet-connected computers of participants from the general public. This paper addresses experiences in low-effort global computing made with evolution@home over more than four years. It shows how allowing participants to choose the class of computational complexity they want to contribute to can help to deal with the bewildering variety of computational complexities that easily result from individual-based models. Results suggest that, as a first rough approximation, participants' complexity choices are distributed surprisingly even over all reasonable classes of CPU-time and RAM requirements. More often than not, participants tend to finish the simulations they start, if they are committed enough to submit any results at all. Potential uses of intermediate simulation results are discussed and the error of magnitude is introduced to help to deal with imprecise CPU-time predictions. Experiences with the choices of over 300 users who have contributed more than 100 000 simulations with a total of over 80 years CPU time are reviewed. Copyright © 2007 John Wiley & Sons, Ltd.

Received 15 January 2004; Revised 14 July 2006; Accepted 24 July 2006

KEY WORDS: Internet distributed computing; individual-based simulations of evolution; Muller's ratchet; computational complexity; user choices; Grid; error of magnitude

*Correspondence to: Laurence Loewe, Institute of Evolutionary Biology, School of Biological Sciences, University of Edinburgh, Ashworth Laboratories, King's Buildings, West Mains Road, Edinburgh EH9 3JT, U.K.
†E-mail: Laurence.Loewe@evolutionary-research.net

Contract/grant sponsor: Leverhulme Trust

1. INTRODUCTION

Individual-based models are interesting and useful tools to study questions in evolutionary biology and ecology [1,2]. An individual-based model of an evolutionary process explicitly models all important details of all individuals in the population which is being simulated and tracks all important changes in each generation over a long period of time. Many individual-based models tend to include interesting parameter combinations with a very high computational complexity, as many biologically realistic scenarios require following large populations of complex individuals over long periods of time. It is therefore axiomatic that applying more computational power to a particular problem will result in better science, especially when stochastic simulations of many parameter combinations are needed [3,4]. The resulting computational complexity can be very hard to predict, as models frequently focus on the details of the underlying evolutionary process that are often poorly understood. Evolution@home, the first global computing system for evolutionary biology, is being developed to help to alleviate these problems.

Global computing systems making use of volunteered computing power offer one approach to obtaining computing power without investing in hardware and electricity: at the cost of investing in the creation of a system for making use of volunteer computers spread around the globe. Global computing was pioneered by GIMPS and *distributed.net* in 1995 and it was made much more widely known by SETI@home in 1999 (see [5–10]). This paper describes a very simple system developed to perform stochastic simulations of a particular problem in evolutionary theory: Muller's ratchet.

Muller's ratchet is a population genetic process that leads to the stochastic accumulation of deleterious mutations. If deleterious mutation rates in a population remain high enough for long enough, then Muller's ratchet can degrade the fitness of all of the individuals in a population of organisms up to the point where extinction becomes inevitable. Current models of Muller's ratchet assume no recombination and could therefore be important for asexual genetic systems such as some haploid organisms and the non-recombining mitochondrial genomes that are essential for many otherwise sexual organisms. More details about Muller's ratchet and biological aspects of some of the simulation results presented in this study are discussed elsewhere [11]. Individual-based simulations are currently the most accurate way of predicting the rate of mutation accumulation from Muller's ratchet for many interesting parameter combinations [11]. If simple extensions are made to the basic model (e.g. include some distributions of mutational effects on fitness), then such simulations often become the only means of prediction. Their practical disadvantage is the large amount of computing time required for predictions. Simulator005 of evolution@home was started in 2001 to allow more numerous and better predictions of the behaviour of Muller's ratchet.

1.1. Computing needs of Simulator005 of evolution@home

It is hard to assess exactly how much computing power is needed for Simulator005, because it is certainly not possible to observe all potentially interesting input–parameter combinations with the highest desirable precision. The following treatment reduces input–parameter space to seven parameters that shall not be discussed in detail here, except to say that the corresponding parameter values of potential biological interest include approximately five, eight, three, five, eight, three and seven orders of magnitude, respectively. The last parameter plays a special role as it determines population size and therefore has a strong influence on RAM and CPU-time complexity.

DOI: 10.1002/spe

Demanding 10 values per order of magnitude in this setting would require computation of $50 \times 80 \times 30 \times 50 \times 80 \times 30 > 10^{10}$ parameter combinations for each of the 70 different population sizes of potential interest. As each parameter combination may require between 10 and 100 (or sometimes even 1000) stochastic repeats and the larger population sizes require weeks of CPU-time or more, it is easy to see that exhaustive pre-computation of all potentially interesting results would require many millions of years of CPU-time. Fortunately, most specific biological questions can be answered by much smaller samples from parameter space, if the possibility exists to 'zoom' into particularly interesting regions of parameter space. Therefore, all actual computing tasks for simulators of evolution@home are organized into 'projects' that are identified by their unique ProjectID number. By restricting either the number of parameters varied in a project or the number of values per order of magnitude or the range of values per parameter, each project can be broken down to a more manageable size. This is without loss of generality, because future questions can always be investigated by starting new projects with the appropriate parameter ranges. Depending on the existence of special knowledge, these projects can also filter out parameter combinations that already have well-known answers to further reduce computing complexity.

For example, the Project1 of Simulator005 (S005P1) that generated all of the data used in this paper limits itself effectively to changes in only two parameters (10×16 parameter combinations) which are then computed for 12 different population sizes (10 to 10^7 individuals). While it excludes certain trivial parameter combinations using analytical theory, it also allows for different terminating conditions to keep CPU-times for the large populations in check. The details are not really of interest here, except to say that the public part of Project1 contains 1166 different simulations that would take about 26.7 years on a PentiumIII at 500 MHz. CPU-time requirements for each simulation in this collection have been limited to larger than 15 minutes and less than one month each. Simulations with predicted CPU-times of less than 15 minutes have not been published, as these are easily computed in-house and the effort of distributing the tasks exceeds their cost of computation (see Section 4). Extending the upper CPU-time limit to 18 months adds more than 300 years of CPU-time in over 714 simulations. As all of these numbers represent the computation of only one stochastic repeat, at least 10 times the effort will be required to ensure minimal statistical decency.

It may be worth mentioning that the current system is not built to deal with the largest simulations of Muller's ratchet that biologists might eventually want to schedule for computation, as these would require more than 100 years of CPU-time for a single simulation! To attack these problems, a completely different approach is needed: each of these simulations needs to be partitioned into smaller tasks that are then distributed to many CPUs. This approach is not being developed for evolution@home at the present time, because (i) most participants from the general public do not have corresponding clusters, (ii) it is very demanding to develop the corresponding code, (iii) it might be possible that thorough analyses of smaller population sizes over many orders of magnitude might give important clues as to what would happen at these very large population sizes and (iv) it does not make sense to aim for these most complex cases before the simpler cases have been solved (i.e. everything that can be already computed with evolution@home using single CPU machines).

1.2. Related work: why not use existing frameworks?

Faced with the excessive need for computing power and the lack of access to suitable supercomputers, one might employ one of the various global computing frameworks that are available.

DOI: 10.1002/spe

Global computing carries the potential of accessing otherwise unused CPU-cycles of computers from voluntary participants from the general public. Evolution@home, in common with SETI@home and all similar projects, has been built around the hope that enough people will donate enough CPU-time to solve the respective computing needs. This allows scientists to shift the limit of complexity of answerable problems by several orders of magnitude. Therefore, additional computing resources at low to no cost significantly expand biologists' abilities to analyse large models.

So why not use one of the existing frameworks for global computing such as BOINC, XtremWeb, COSM, FIDA or others, as reviewed in [5–10]? Unfortunately, the choice of a good framework for global computing is not trivial. Benefits include full automation, sophisticated transmission and security features, at the cost of generating dependencies that can turn out to be very expensive if the framework is no longer supported or if it turns out later that some non-obvious but important feature is missing. Furthermore, it should be noted that support for extremely heterogeneous tasks with very poorly predictable complexities is currently not available in most frameworks. In the case of evolution@home, the key reasons for not using any of the existing frameworks were the lack of support for handling stochastic simulation results and missing support for work units (i.e. simulations) with computational complexities that span many orders of magnitude. A short survey of other global computing projects [7] shows that the computational complexity of individual work units ranges from fractions of seconds (Photons in Xpulsar@home) or a few minutes (Porivo's peerReview) up to several weeks (some primes in GIMPS) or even more than a year (complex models from *climateprediction.net*). Most projects partition their work into units of several hours up to a few days such as SETI@home or Folding@home. However, no current project has such a diverse work unit complexity as evolution@home.

All such projects encounter the problem of scheduling the computation of their large numbers of work units. Thus, they share the application-centric view of application-level scheduling, an important topic in computational Grid research [12,13]. In contrast to classical resource or job scheduling on supercomputers, application-level scheduling optimizes timely execution from the perspective of the application, for example by avoiding delays due to batch queues or slow connectivity. Optimization criteria might be turnaround time, result quality or others. This requires a sophisticated software system that has been specifically adapted to the application and monitors performance relevant features such as CPU and network load. These observations are then used to compute the best schedule under the given circumstances. An application-level scheduling system might allow users to work (more or less) interactively with applications that are too complex for any single workstation [13,14]. Recent efforts to support parameter-sweep applications extend application-level scheduling to extremely complex problems that are similar to most global computing projects [13,15].

Global computing projects can be divided into problems with equally interesting work units (e.g. cryptography) and problems with unequally interesting work units (e.g. evolutionary models). While minimizing overall completion time is probably the most important goal of potential scheduling strategies for the former, the latter add a new aspect to scheduling, as they require fast processing of the most important work units first and overall completion time is less important. The interesting work units play a key role in steering the whole project by helping to determine interesting regions of parameter space for more intense investigation. The bulk of the computing power is then needed to increase the statistical power of conclusions that have been suggested much earlier by the initial high-priority simulations. Overall completion of the project is often arbitrarily defined, as additional statistical power is always welcome, but at some point conclusions seem to be established well enough

DOI: 10.1002/spe

to justify an official end. The task of finding these important work units is highly specific to a problem domain. Thus, such projects can easily be limited by the need for a sophisticated data-management and priority-setting system.

For situations where a sophisticated global computing framework is not available, the semi-automated participants-free-choice system presented in the following has one distinctive feature: simplicity. It does not require a complicated server suite, a specially adapted scheduler, a Network Weather Service, a list of known computers or standing Internet connections, and neither does it depend on an external and potentially very complicated global computing framework with its own set of problems. It can be operated with a simple static Web server and an e-mail account on the operator's side and occasional dial-in connections on the participant's side. Administrative overheads for scheduling are minimal, as are requirements for participants. If turnaround time is less critical than both, a low implementation complexity and easy access to more computing power, then this minimalist approach using the participants-free-choice paradigm can be recommended. In other cases more sophisticated application-level scheduling [13] or a specially adapted global computing suite [6,7] will be necessary. A more in-depth discussion of the advantages and disadvantages of various global computing and Grid software frameworks such as BOINC and Condor can be found elsewhere [5,6].

1.3. Goals of this paper

This paper has three goals that are reflected in the next three sections. Section 2 describes the simplest global-computing system that can possibly be conceived, making use of the participants-free-choice paradigm (described in the following). Section 3 describes the experiences that have been made with this system while running evolution@home over the course of more than four years. This section shall help future researchers to estimate how much effort is required to receive a particular amount of computing time. Section 4 describes the results of applying the participants-free-choice paradigm, a novel concept in global computing whereby users can choose the complexity of the work units they devote their CPU cycles to. This section discusses the interrelation between the ability to have people voluntarily participate in a global computing project, the variability of the length of the work units, the quality of the prediction of their computing time and the chances that these simulations will run to completion.

2. GLOBAL COMPUTING WITH MINIMAL EFFORT

2.1. Overview

2.1.1. Semi-automated global computing

Global computing projects can be operated at two levels. In fully automated projects, the worker-software contacts the corresponding project servers on the Internet automatically for getting tasks and submitting results; it may even update itself to new versions. Participation in such systems is easier, but implementing them is more difficult. The long list of desirable features for smooth global computing has been reviewed elsewhere [5,6,8,16,17]. If time for implementation is extremely limited, it may be worth considering a simple semi-automated global computing project instead. In semi-automated projects, occasional manual interaction is needed to download a group of tasks and to submit results.

DOI: 10.1002/spe

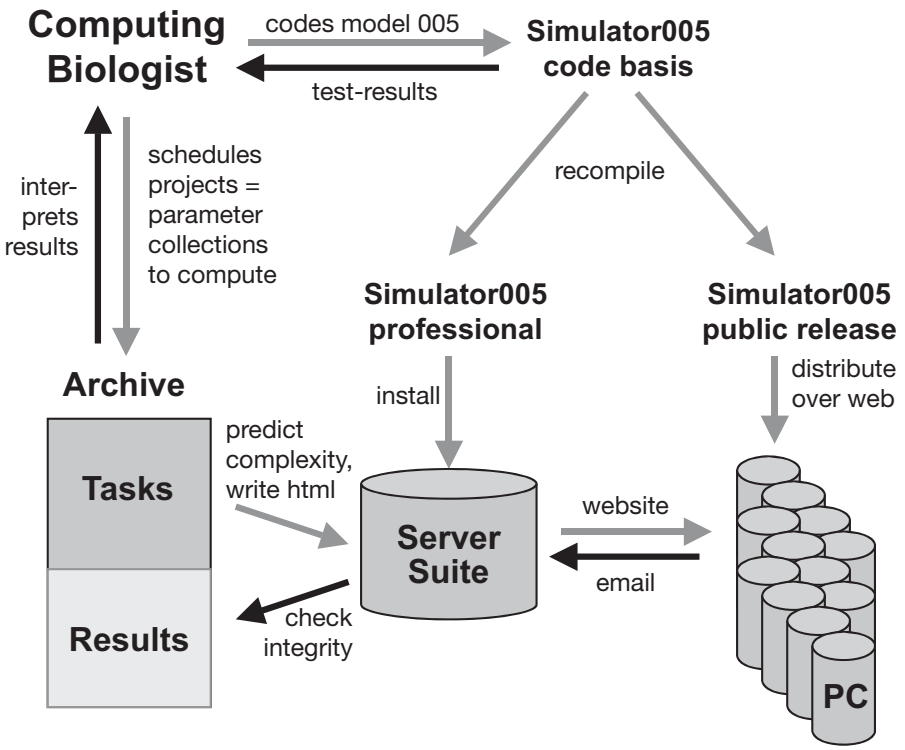

Figure 1. General workflow for research with evolution@home. Simulator005 stands for any specific simulator that encodes a specific individual-based model.

This reduces the complexity of implementation at the expense of ease of use. The core design proposed here delegates all data-transfer functionality to the user and his favourite programs for surfing the Web (download new tasks) and e-mailing (submit results). An overview over the general workflow is presented in Figure 1. The system includes a substantial amount of functionality that depends on the actual computing tasks and that is independent from the methods used for data transfer.

2.1.2. Preparations for going global

To prepare a simulation code for global computing, more than the usual amount of time should be spent on developing the code to avoid as many complicated upgrades as possible (allow for enough flexibility through input parameters, report all potentially interesting output parameters, etc.). The worker software has to (i) compute checksums for results to allow verification of correct transmission, (ii) include its version in all results to allow the filtering of results if released code contained bugs that touch the science of the model, (iii) be as easy to use as possible and (iv) have additional features that are important to the general public (e.g. enter high scores information to make

DOI: 10.1002/spe

contributions identifiable). Actual operation of such a project requires the maintenance of a good Web site that includes regularly updated high scores to acknowledge CPU-time contributions.

2.1.3. Limitations

The semi-automated approach allows little control over the number of times that a particular simulation is repeated, as many people may download identical tasks. This setup is only feasible for stochastic simulations that need to be repeated many times and where every simulation automatically gets an individual random seed. If little control is needed over the number of repeats of tasks, then one can avoid the complexities of a dynamic Web server linked to a complex database.

2.2. Particular technology choices

2.2.1. Data transfer

While any simple static Web space can be used for the distribution of binaries and tasks, more thought is required for submissions, which should always go to a dedicated e-mail address. The e-mail client that receives submissions should (i) automatically move every e-mail sent to the simulator submission address into a particular mailbox, (ii) automatically move every attachment belonging to these e-mails to a special folder on the harddisk for simple further treatment and (iii) save the content of many e-mails into a single file without changing anything in their content including line-endings. Eudora (light, v6.2 and earlier) was used for these purposes. There were no recommendations about e-mail clients on the participant's side. While e-mailing was no problem in most cases, it turned out to be unreliable for a very low percentage of results that were submitted as text in the body of an e-mail. In these cases a number of strange interactions between e-mail programs and servers led to the random break-up of a very long and important line of text in the special file format used by Simulator005 up to release 6. Sending results as zipped attachments removed such interference.

The use of a unique start and stop character for each single-run result proved to be extremely helpful in finding results in a stream of raw e-mail data that included SPAM. The same technique was used to find tasks in a downloaded run-file in HTML format.

A special script (Applescript) was used to rename all received e-mail attachments to a running number. This allowed a simple loop over possible filenames to import results into the professional version of the simulator that was used to check results integrity and compile the big single-run-results table described below.

2.2.2. Security

Semi-automated global computing has a number of advantages from a security perspective because of its simplicity. If people trust that the code being distributed is not malicious and the executable itself has no security vulnerabilities, such as buffer overflows from reading manipulated input files, then no further vulnerabilities are introduced because the system relies on the security of an already existing Web browser and e-mail client for all potentially dangerous Internet connections. Transparency is also increased by the fact that input and output files of the worker software are both simple text files. Security issues on the server side are reduced to the security of (i) the e-mail server, (ii) Eudora,

DOI: 10.1002/spe

(iii) the program for unzipping some attachments and (iv) the results-reading code of the professional version of the simulator. It is much easier to deal with these issues than with the plethora of potential security issues that come with many fully automated global computing solutions.

2.2.3. Data management

All simulation results were stored in a special file format and an unmodified log of all submitted results was kept. The most important output parameters of those results that passed integrity checks were collected in a special table-like file that was used for all further analyses. The latter was a tab-delimited file, where each line corresponded to a single stochastic simulation run belonging to a particular input parameter combination, so that a simple lookup in the table quickly led to the corresponding output parameters. Analyses of this large single-run results table were performed with the open-source statistical programming language R [18]. R proved very effective for producing scientific plots and compiling computing high scores.

2.2.4. Programming language

The worker was implemented in C++ as a standalone executable without external dependencies to maximize computing speed and ease of installation, while keeping external dependencies to a minimum. Java was not chosen to avoid the extra complexity of having to install a particular virtual machine on a wide range of participants' machines to guarantee proper functioning of the worker.

3. CASE STUDY: EVOLUTION@HOME

Evolution@home is the first global computing system for evolutionary biology and also the first to distribute individual-based models. It started its first semi-automated global computing project in April 2001 (see [19,20]). A global computing solution was preferred over use of the local supercomputer centre, because the former promised dynamic growth of CPU-power, while the latter appeared rather limited in its flexibility to accommodate long-running computations with ill-defined complexities that are managed by an early-career researcher who might change to other institutions in the middle of the project.

Evolution@home started as a minimalistic semi-automated global computing project without high scores, as described above. After setting up everything, an e-mail to a few global computing project review sites (e.g. [7]) was enough to attract participants that contributed more than five years of computing time in four months on the promise of eventual publication of high scores. An update of the Web site continued to sustain this rate for another four months. However, then nine months without any Web site update followed for technical reasons. In that time participation seemed to decline with a half-life of around three months. As can be seen in Figure 2, this trend could be reversed only by more frequent Web-site updates, the introduction of a discussion forum and the production of computing high scores in the second half of 2002. Frequent high-score updates are an often requested feature, as most participants want to see their contributions acknowledged as quickly as possible for various reasons, including competitiveness and the assurance that data did not get lost on the way.

DOI: 10.1002/spe

Right from the start, participation was distributed over a wide range of countries and continents. By now results have been received from over 30 countries and from all six densely populated continents.

3.1. Specialities of evolution@home

3.1.1. Benchmarking

When the simulator is started for the first time, it asks for a maximal RAM limit on this computer to avoid the use of virtual memory, an effective knockout even for fast pre-emptive multi-tasking operating systems, because Simulator005 would trigger endless paging. Before the first simulation is started, a benchmark population is generated that (i) will test whether the promised amount of memory is really available and (ii) measures an initial performance under current real-world conditions. This performance is used for initial computing-time predictions. However, once a real simulation is run, the actual performance is monitored and used for predictions requested by the user. After each simulation, computing time and complexity are recorded in the preferences file to allow computation of a long-term average performance that is used for future performance predictions. Thus the 'benchmark' adapts as close to the real application as possible.

The performance of Simulator005 on a CPU is measured in average complex 'operations on individuals' per second, where each operation includes the simulation of all biological processes of one individual in one generation. This measure is convenient for the model behind Simulator005, since all individuals take up the same amount of memory and behave almost identically, leading to comparable numbers of CPU-instructions per individual, at least in S005P1. The basic idea is that the number of individuals in the population mainly determines the CPU-time needed to advance the simulated time by one generation. A performance of 10^6 operations on individuals per second could therefore either stand for the simulation of a population of 1000 individuals over 1000 generations or for the simulation of one generation of a population of a million individuals per second. Correspondingly, the overall computational complexity of a simulation is measured in GigaIndividuals (GI), where 1 GI represents the simulation of 10^9 individuals over one generation each. Future simulators might have to abandon this concept, as the definition of 'operations on individuals' would have to be changed to 'effective operations on individuals' if individuals differ widely in their computational complexities. As benchmarking is a murky business anyway, it may be equally justified to use double floating-point operations to measure the performance and the total amount of work done. This is planned for the future.

3.1.2. Intermediate results

Intermediate results of Simulator005 are formally indistinguishable from full results. Both include the same list of output parameters. The difference is the point in simulated time at which those parameters were observed. A full result is defined to be an observation of the simulated population in the last moment before the simulation officially stops. An intermediate result is any observation before that. Simulator005 writes intermediate results to disk at regular intervals to avoid full loss of information about the simulation in the case of interrupted computations.

DOI: 10.1002/spe

Why are intermediate results helpful? It is one of the main objectives of Simulator005 to observe the rate of mutation accumulation in a population with the properties defined by a given set of input parameters. This leads to an elegant opportunity to make use of certain intermediate results that might be generated by prematurely interrupted simulations. The most natural termination criterion for many of these simulations is extinction, but that would allow some of the populations to run forever or for longer than necessary. These simulations are stopped after an arbitrary number of generations or after accumulation of an arbitrary number of mutations, whichever of these input–parameter-specified limits is reached first. The arbitrariness in these terminating conditions suggests that not all simulations that terminate earlier have to be useless; sometimes a less accurate estimate of the rate of mutation accumulation can be derived from observing the accumulation of a few mutations and there is no reason why corresponding results should be omitted. However, one should also note that extremely short intermediate results could indeed be close to useless for some questions. Therefore, under ideal circumstances all simulations are completed if their computing times are predicted correctly and the global computing framework will use the incomplete simulations to improve computing-time predictions.

3.1.3. Work unit sizes

While other global computing projects have relatively uniform work unit sizes, the work units of evolution@home can span more than seven orders of magnitude (see Section 1.1 and Figure 8). Considerable effort was needed to deal with this circumstance and this would not have been reduced if an existing global-computing framework was used: one will always have to find reasonable predictors of computational complexity and one will always have to sort tasks according to several dozens of complexity classes. Feedback from participants welcomed the opportunity to choose the computational complexity class they want to contribute to, although they would prefer to see this feature in a fully automated context.

3.1.4. Random seeds

To start many different stochastic simulations from the same run-file found on a static Web site requires some mechanism for generating new random seeds. In this work, date and time (in seconds) from the participants machine are used to generate a random seed that is incremented every second after a particular absolute date. This mechanism works quite well, because the probability of starting two new simulations with identical parameter combinations in the same second is quite low if compared between different computers. Only multi-processor machines where each processor runs a simulator have to make sure that they use different run-files. Corresponding explanations were given on the Web site, but in the long term it is clearly desirable that a fully automated system will generate unique random numbers, especially as multi-processor machines are becoming increasingly common. Also, the present system uses these random seeds to compute the start time of a simulation as reported in Figures 2 and 3 (and also in Figures 11 and 12 in Section 4).

3.2. Description of add-ons and performance history

For evolution@home the initial bout of CPU-power could not solve the computing needs at hand. Therefore, tools had to be developed to monitor performance and various measures had to be taken

DOI: 10.1002/spe

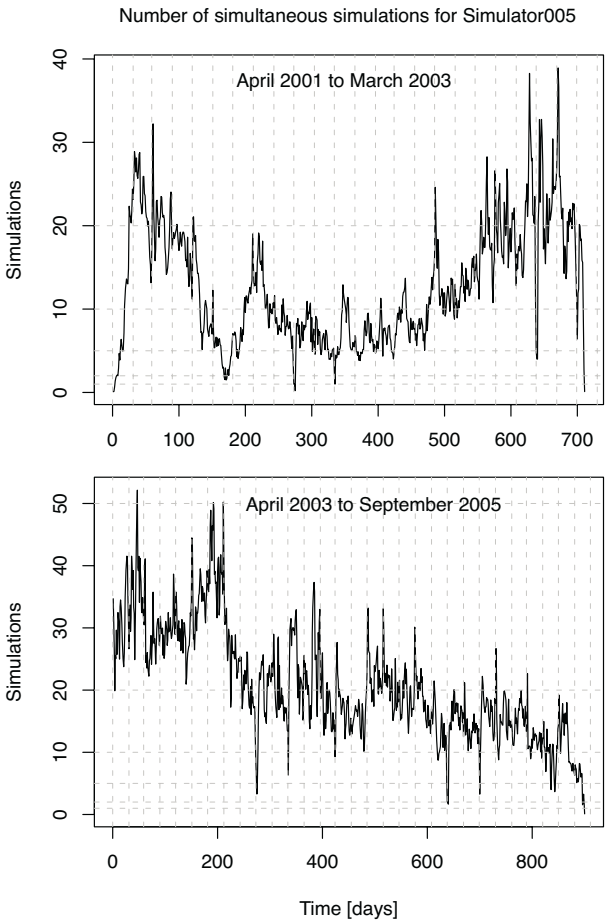

Figure 2. Activity of the evolution@home network over time. Day 1 corresponds to the 1st April. The first mail to some global computing review sites led to appreciable activity, which declined during a long period of no Web-site activity (September 2001–July 2002). With the publication of monthly computing high scores and occasional Web-site updates, activity started to gain momentum again. The number of simultaneously running simulations was computed from the start and stop times as determined by the system clock of the participants computer. Web-site activity during 2004 and 2005 was mainly limited to high-score updates every four to eight weeks and even rarer news updates due to severe time constraints on the part of the project manager. This shows in a slow decline of concurrently running simulations. Most other plots are divided into the same upper and lower part to convey an impression of how the various statistics changed over the years. The upper plot includes all data from the first two years and contains 47 843 simulations that used up a total of 24.9 years of computing time (32.7 years predicted) and simulated about 1.5×10^{14} standard individuals. Of these, 11 080 simulations were private (0.38 years, prediction 0.0472 years) and 2613 ended prematurely (5.57 years in these intermediate results, prediction 12.8 years). The lower plot includes all data after the first two years and contains 64 785 simulations that used up a total of 56.9 years of computing time (201 years predicted) and simulated about 5.5×10^{14} standard individuals. Of these, 997 simulations were private (0.02 years, prediction 6×10^{-5} years) and 4363 ended prematurely (13.4 years in these intermediate results, prediction 72.5 years).

DOI: 10.1002/spe

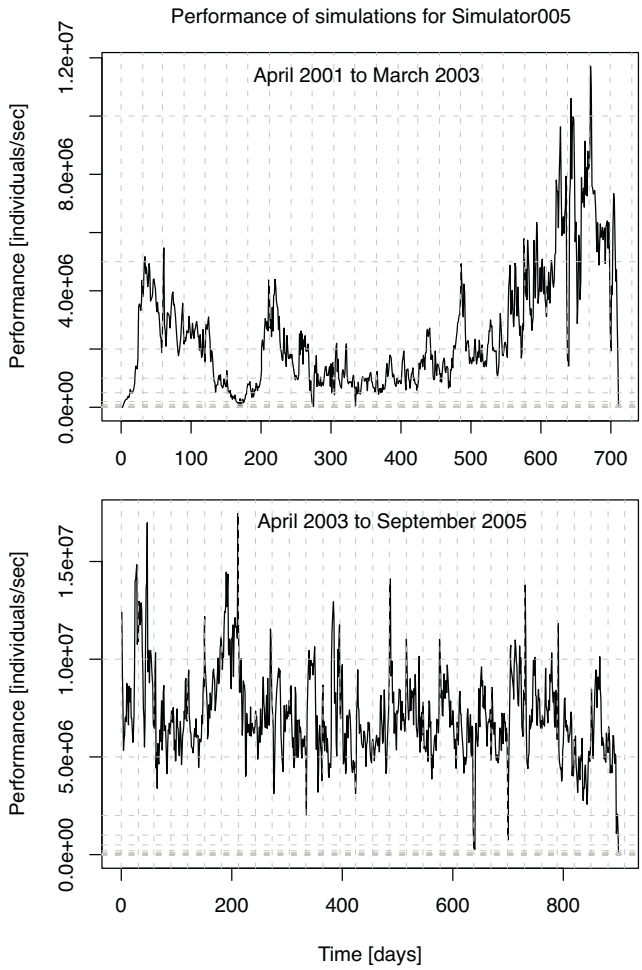

Figure 3. Performance of the evolution@home network over time. This plot contains the same simulations as Figure 2, but now reports the aggregated performance from all simulations that were running on a particular day. Note that the absolute decline in simultaneous simulations that can be seen in the second part of Figure 2 does not translate into an absolute decline in CPU power. This can be regarded as a side effect of Moore's law and hardware turnover on the side of participants. Performance is measured as 'operations on individuals per second'.

to keep participants interested. The latter involved occasional Web site updates and running of a Web-based forum for user-interactions (this was offered by a third party and after the first forum closed down it was not difficult to find someone else to host a follow-up forum, just by asking for volunteers on the evolution@home Web site).

DOI: 10.1002/spe

An important remark concerning the importance of high scores is in order. There were many discussions over this with the community of participants, who demanded something like automated daily updates of high scores. Unfortunately this was not possible due to technical reasons (production of high scores was semi-automated like the rest of the project and, up to now, time-constraints neither allowed updating every other day nor upgrading of the whole system to full automation). The publication intervals of every four to six weeks are enough to prove that the project is still running, but are not enough to attract groups that specialize in global computing as a hobby. Some of these groups have repeatedly offered their computing power of several thousand members, but demanded daily updates of high scores, as the meaning of a 'high score race' is lost if updates take too long.

Thus, a further increase in performance can be expected from full automation that will also allow more frequent high scores. Unfortunately, the development of a fully automated system is not straightforward, as this system touches many issues (e.g. schedule new tasks, set priorities, re-order priorities, distribute tasks, collect results, validate results, manage annotations to results, generate high scores, store results for long-term use by scientists and export parts of results for generating scientific plots). Thus, it appears that projects such as evolution@home can easily be limited by the need for a sophisticated data-management and priority-setting system that is completely independent of global computing issues. Most data-management issues follow from special properties of individual-based simulations. It is therefore not clear how much time can be saved by using an existing global computing framework; it will clearly help transport files over the Internet, but it might also be an obstacle to implementing a biologically meaningful workflow. The importance of these data-handling issues for overall success cannot be underestimated. Future work is needed on how to best manage complex biological global computing results in the long term.

3.3. Usability strongly influences participation

It is trivial that many more people will use simpler worker software and that only few are dedicated enough to put up with complex worker software. A good example of the importance of this principle could be observed during the operation of evolution@home. About 20% of the CPU-time seemed to come from Macintosh users. This was a surprisingly high fraction, if compared to Apple's market-share of 5% or less. A reason for this might have been the different degrees of user-friendliness that came with the two versions. While both were command-line based, the Mac version had a very simple way to stop the current simulation (just press any key and you will get a dialogue), where as the Windows version was very awkward to stop (move a file called 'break.txt' into the simulators folder and you will get a dialogue). This was because the author did not take the time to implement a way of shutting down the computation that most Windows users would find reasonable. Participation by Windows users was disproportionately low as a result. Apparently, every minor complication of the user-interface will directly translate into reduced CPU-power.

A rough comparison of the number of downloaded executables (Web site log) to probably participating persons suggests that at a point in time only about 1% of all downloads resulted in any submitted results; this assumes that the anonymously submitted results that represent about 50% of CPU-time came from approximately as many participants as those 50% where participants entered high-score relevant information. Obviously, there is a huge margin of error associated with that estimate, but even if the order of magnitude is correct, then this strongly suggests that the usability of a non-GUI, semi-automated command-line worker can be a major obstacle on the way

DOI: 10.1002/spe

to more CPU-power. This result seems to be very different from the observations made by the *climateprediction.net* system [16], at least on the surface. They followed new participants after two special, media-promoted 'launch dates' and found that about two-thirds of all newly registered participants went on to compute for at least one day, where as only 4% and 28% of newly registered machines were still active after six months, respectively [16]. The difference between the two long-term values might be caused by the differences in the underlying global computing framework (4%, self-developed; 28%, BOINC). It would be not unexpected, if their self-developed solution attracted more global-computing newcomers on the basis that it appeared to be easier to install, whereas the corresponding BOINC client might have been installed by many veterans that have used BOINC before. The list of differences between evolution@home and *climateprediction.net* that bias this ratio in favour of the latter is long: a spectacular GUI with the visualization of current results, full automation, daily high scores, checkpointing and a large community that exert considerable persuasive power to continue participation. However, there is also a less obvious factor that will lead to a much higher rate of 'loss' of participants for evolution@home in the calculation presented above. While registration is compulsory for *climateprediction.net*, new would-be participants of evolution@home do not have to fill out a Web form to become part of the project. They simply download the simulator and start with the work, and can remain anonymous; submission of results makes them 'official' participants. In this way many undecided visitors are likely to download the relatively small simulator (about 1 MB), where as the determination to register and download 10 MB of climate modelling data can be expected to result in a much higher probability of completing at least the first day.

4. PARTICIPANT CHOICE AND WORK UNIT VARIATION

The global computing system described above, evolution@home, is the first of its kind that allows participants to choose the complexity of work units. Initially, it was not clear how participants would respond to this flexibility. Feedback indicated that participants like the option to choose the complexity class they contribute to, but would prefer it in a fully automated system. Several issues arise when participants can choose the complexity of their contributions.

- How accurate are the predictions?
- How can computer systems be benchmarked?
- How faithful are participants regarding their commitment?
- How biased are the choices of all participants as a group?

The remainder of this section discusses these issues in the light of the patterns of choice observed in S005P1 of evolution@home (3 April 2001–28 September 2005 over 10^5 results, more than 70 years CPU time, over 300 participants; for more precise statistics, see Figure 2 or [19]). A previous version of this section analysed 13 000 public and 9000 non-public results as of December 2001 (see [20]).

As argued above in more detail, the work units of evolution@home span more than seven orders of magnitude owing to problem domain inherent constraints. To continue extremely long runs on other computers would lead to prohibitive communication and other server-side costs, as the longest runs usually generate huge amounts of temporary data. Therefore, participants should choose their commitment in terms of physical RAM and computing time to increase the probability that a given

DOI: 10.1002/spe

run would be completed. To this end, RAM and computing-time complexity were predicted for each parameter combination and participants chose their commitment by downloading one of about 50 run-files, where each file contained up to 60 different simulation tasks to allow easy scheduling of many simulations. As each command that starts a simulation also contains its predicted complexity, users could (and did) compose their own special run-files. Population sizes in the current project S005P1 led initially to RAM requirements from 1 kB to 900 MB with predicted computing times between 0.01 s and 180 years assuming 200 000 operations on individuals per second (on something like a 500 MHz PentiumIII). While some evolutionary questions would still ask for more, simulations with more than 30 days predicted computing time were excluded from publication until mid-2002. At the other end of the spectrum were simulations that were predicted to need less than 15 min; these were run on a non-public computer. While one can debate on their exact location, the lower and upper boundaries T_{low} and T_{high} indisputably exist for tasks, whose global computation is feasible.

4.1. Boundaries for distributable tasks

4.1.1. Lower limits

The lower boundary T_{low} can be derived when communication costs are considered. If the transfer of data takes longer than local computation, then distribution is not feasible:

$$T_{\text{low}} \approx \frac{\text{Transfer size}}{\text{Transfer speed}} \tag{1}$$

where 'Transfer size' is the sum of all bytes of all communication needed to distribute the task and collect the result over a network with 'Transfer speed' as the real-world bandwidth in bytes s^{-1}. Thus, it makes no sense to distribute work units of less than 25 s expected computing time over a 28K modem line if they generate 100 kB of traffic. While this figure shrinks to 1 ms with Gigabit Ethernet, it rises considerably for semi-automated projects where occasional manual interaction is needed.

4.1.2. Upper limits

An extreme upper limit for the upper boundary T_{high} can be derived from a variety of sources that may include Moore's law, the reliability of hardware components and the need for users to install new operating systems. In the absence of a good checkpointing solution, T_{high} is much more limited by the frequent need for rebooting due to various causes, including the installation of security patches by the operating system. When discussing the upper limits of computing time it is important to consider the absolute computing time on the actual machine and not just the size of the task on some standard machine, as the performance differences between CPUs participating in evolution@home (see Figure 5 below) can easily render such general predictions useless for the individual user. The main point here is really that a global computing framework for simulating evolution needs to be able to handle an upper limit for tasks, whatever it may be.

4.1.3. Parallelization

Single simulations that are too complex for current CPUs need to be parallelized for running on many processors. This consideration has certainly gained significant weight in the very recent past, because

DOI: 10.1002/spe

the increase of the average performance of CPUs seems to have come almost to a halt [21]. However, parallelization rarely comes easy and it is particularly challenging for simulations of evolution, if it is to be done automatically by the simulation software framework to simplify the coding of simulations of evolution. Currently, evolution@home deals only with the large number of much less-complex simulations that still sum up to enough CPU power to make good use of global computing in its present form. In other words, it makes much more sense at present to explore simulation models that can be computed in a reasonable time, rather than to spend much time on writing highly parallel code for models whose biological significance is not yet clear.

4.1.4. The ideal work unit size

Ideally, work units have a high computing to network traffic ratio. Therefore, it might make sense to lump together enough simpler tasks to transfer a day's work at once (especially for dial-up connections). On the other extreme, months of computation without network traffic are very detrimental to the motivation of many participants that want to see their latest achievements in the evening, when they come home from their day-job. SETI@home seemed to have struck just the right balance in this respect; this issue is so important that projects such as *climateprediction.net* provide a mechanism for updating high scores on the basis of daily progress of long ongoing simulations.

4.2. Problems in predicting computing time

To predict computing time one has to consider the local performance and biological knowledge about the system under investigation. In many simulations that tackle long-term evolutionary questions, models have a general structure that waits for certain events, for example the accumulation of 500 mutations or extinction in S005P1. In most cases, some related parameters indicate the arrival of these events. To predict the computational complexity of scheduled simulations, one might want to start the actual simulation to measure the initial rate of change in various indicators and from them extrapolate to the total CPU-time needed. Unfortunately, this is very error-prone and complex enough to require a separate global computing system. Thus, very simple calculations have to be used to arrive at reasonable predictions in a reasonable time. However, systems where such simple formulae give the correct answer are usually not at the cutting edge of research. Thus, one of the challenges in developing a simulator is the need for such a formula with reasonable prediction quality. This is no small task, as not even a crude order of magnitude can be predicted for some problems in evolutionary biology, including Muller's ratchet, the topic of Simulator005. Thus, participants and administrators have to live with this inaccuracy in predicted computing times. After all, no simulations are necessary, if the answer was known already.

4.3. Definition of the error of magnitude

To estimate prediction quality of Simulator005, predicted computational complexities were compared to actually observed complexities for completed runs. As can be seen in Figure 4, considerable prediction errors exist in both directions. Unfortunately, one cannot just compute the relative error, as it generates seriously misleading figures as a result of its inherent asymmetry ($[10^{-7}$ *actual* $-$ 1 *estimated*$]/1$ *estimated* leads to approximately -1 as relative error, although the observation is seven orders of magnitude off and would lead to a relative error of 10^7, if 10^7 had been actually observed).

DOI: 10.1002/spe

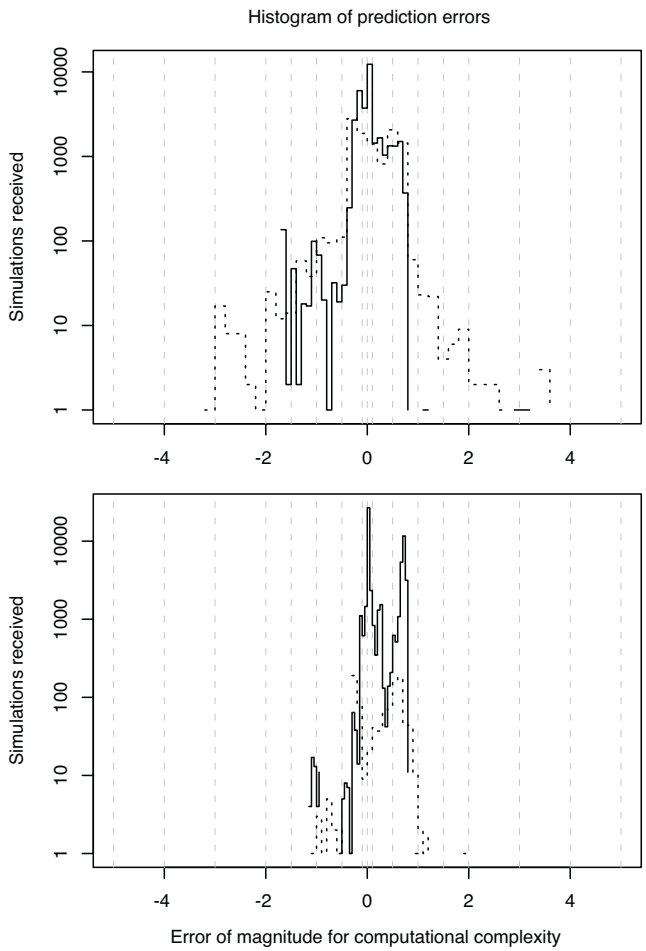

Figure 4. Histogram of prediction errors of magnitude. The solid line denotes completed public results, the dotted line completed non-public simulations. See the main text for the definition of the error of magnitude. Although most predictions are accurate within half an order of magnitude, outliers can be extreme. Results in this plot were grouped in the same way as in Figure 2. The upper plot contains 34 150 completed public results (observed 19.0 CPU-years) and 11 049 completed private results (observed 0.221 CPU-years). The lower plot contains 59 425 completed public results (observed 43.5 CPU-years) and 996 completed private results (observed 0.0202 CPU-years).

DOI: 10.1002/spe

This suggests the introduction of a new type of error, the error of magnitude E_{m}. The fact that all relevant values have the same sign allows the error of magnitude of predicted values to be defined relative to actually observed values as

$$E_{\mathrm{m}}\{x\} = \log_{10}\left(\frac{x_{\mathrm{estimated}}}{x_{\mathrm{actual}}}\right) = \log_{10}\left(1 + \frac{x_{\mathrm{estimated}} - x_{\mathrm{actual}}}{x_{\mathrm{actual}}}\right) \tag{2}$$

where $x_{\mathrm{estimated}}$ denotes the guess or prediction of x_{actual}, the true value of interest, which is determined by observation in this case. In our case x stands for computational complexity. The right-hand side of (2) is more prone to rounding errors, but emphasizes the close relation of E_{m} to the relative error. The error of magnitude has some useful, intuitive features.

- It is symmetric around 0, where 0 indicates no error.
- Negative values indicate that estimates are below the actual values.
- Positive values indicate that estimates are above the actual values.
- An E_{m} of 1 or 3 has the intuitive meaning of missing the mark by a factor of 10 or 1000, respectively. Increasing E_{m} by 1 indicates a further tenfold deviation from the target.
- An E_{m} of 0.1 or 0.01 has the intuitive meaning of missing the mark by less than 26% or 2.4%, respectively. Lowering E_{m} values below 1 by an order of magnitude also lowers relative errors approximately by an order of magnitude.

To the best of the author's knowledge, (2) has not yet been proposed for the use of quantifying errors in the order of magnitude. The enormous uncertainty that surrounds many biological parameter values suggests that the error of magnitude might be useful for quantifying the properties of biological systems that cannot have different signs by definition, but otherwise fluctuate by many orders of magnitude.

4.4. How accurate are predictions?

Figure 4 shows the errors of magnitude of the predicted values relative to the observed values. Most predictions deviate by $E_{\mathrm{m}} = 0.5$ or less. This accuracy within a factor of three is remarkable, given the fact that we are dealing with more than seven orders of magnitude here and mutation accumulation due to Muller's ratchet, the biological process under investigation, is notoriously hard to predict [22]. However, initially observed extremes ranged up to two orders of magnitude below and above the predicted value. Improvements in the prediction algorithm that were introduced with Release 6 of Simulator005 in mid-2002 can be seen in the results (see the lower panel of Figure 4).

In light of such prediction problems, the proper handling of intermediate results becomes crucial, especially as this problem is not rare in evolutionary biology. Given the strict rules regarding computing time limits in most supercomputer facilities, global computing permits astonishing flexibility here. To improve the usability of the system, it is proposed that users are not only asked for the highest RAM and computing-time limits when configuring a simulator, but also for an upper limit for the corresponding prediction errors of magnitude. Thus, runs with poor complexity predictions are deliberately forced into intermediate results that can then be used by the framework to improve predictions for the corresponding parameter combination.

DOI: 10.1002/spe

4.5. Benchmarking: how variable is the performance of participants' systems?

Benchmarking is necessary to give the user a prediction of computing time on their system, and the importance of hitting at least the correct order of magnitude should not be underestimated. Unfortunately, the comparison of computing systems is very complex and many of the benchmarks devised concentrate only on a small part of the performance spectrum [23,24]. Thus, some have reached the conclusion that the only valid benchmark is the final application. In global computing the situation is worse, as only idle CPU cycles are used and simultaneous execution of any other software by the owner of the machine will decrease performance. Simulators of evolution@home have even more problems. As their tasks span nearly six orders of magnitude in RAM complexity, cache effects play an important role, because some of the very small populations completely fit into the cache of some modern CPUs.

The details of the benchmarking method used in evolution@home are explained above. Equipped with this background, the performance spectrum of all single-run results of evolution@home can be understood (Figure 5). Performance observations from computers that contributed results span about two orders of magnitude, if cache-sized simulations are considered. For simulations that are too large for current caches, performance differences are a little less extreme. Using Moore's law and the questionable assumptions that the slower runs were not the result of busy CPUs, this figure could be used to infer that the participants of global computing projects use systems that range from brand new to more than a decade in age. Interestingly, such reasoning is confirmed by CPU information that has been manually entered by participants: if these entries are as credible as they seem to be, then the full range engulfs everything from a 40 MHz 386 AMD processor to a 3.4 GHz Pentium 4. Considering the computers that are probably owned by the general public supports this: some will put their ancient PC in the basement to good use, while others will use their latest top-of-the-range gaming machine for advancing science. Most probably, global computing will always have to deal with a broader performance spectrum than other applications, as some people might use it to give their old PCs a new destiny. Projects with variable work unit complexities are particularly well suited from this perspective. Altogether, these results underscore the importance of taking local performance into account when predicting total run time, especially for complex simulations.

4.6. Intermediate results: do participants honor their commitment?

Nearly 7% of results and a quarter of the computing time received from public participants came from interrupted runs. As the current release of the simulator cannot save an intermediate state to disk and continue from there, completed simulations mean that the corresponding computer was running all of the time and the simulator application had not been closed. To avoid losing everything in the case of a crash, the list of observed parameters was written to file every hour (for exploratory analysis as explained above). Given this fact, it is remarkable that more than 90% of all simulations could run uninterrupted, although many of them had long computing times.

The generation of intermediate results might have two causes. First, prediction quality might have been so bad that the user stopped the apparently never-ending run, with every moral right on his side. Second, prediction quality might have been fine, but the user changed his commitment deliberately or involuntary (e.g. system crash). Figure 6 shows that in the first two years of operation 14% of all

DOI: 10.1002/spe

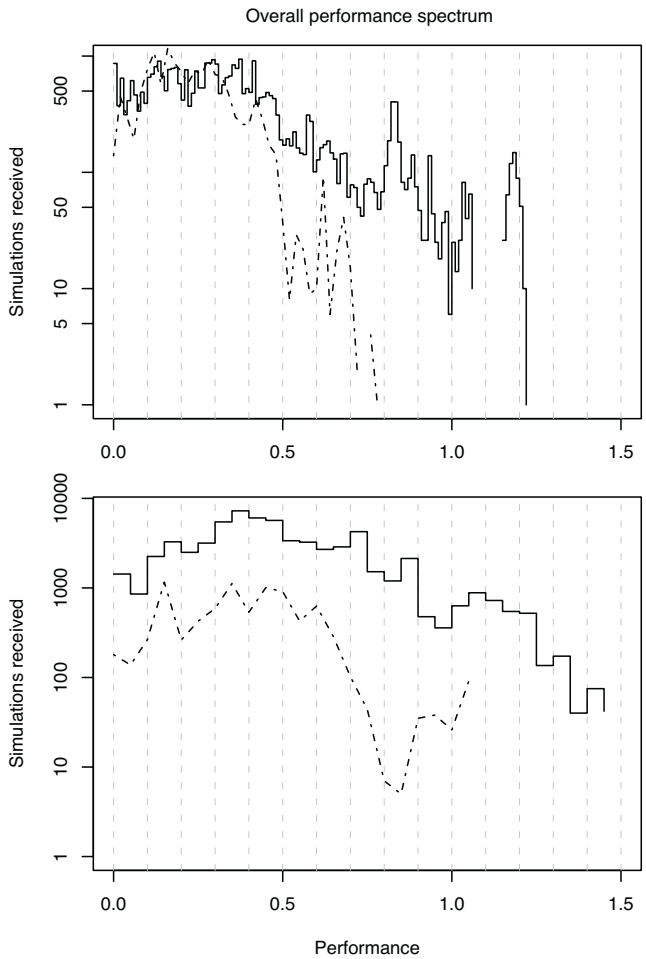

Figure 5. Performance of computers working for evolution@home as measured in million operations on individuals per second. The solid line represents all public simulations (complete and intermediate results) and includes cache effects on performance. The dash-dotted line represents public simulations that required more than 8 MB RAM, which should exceed any modern cache. Results in this plot were grouped in the same way as in Figure 2, where summary statistics of the corresponding time intervals can be found for the solid lines. The dash-dotted lines in the upper and lower plot are based on 13 678 and 8282 simulations that required 14.2 and 27.2 actual CPU-years for computation, respectively.

DOI: 10.1002/spe

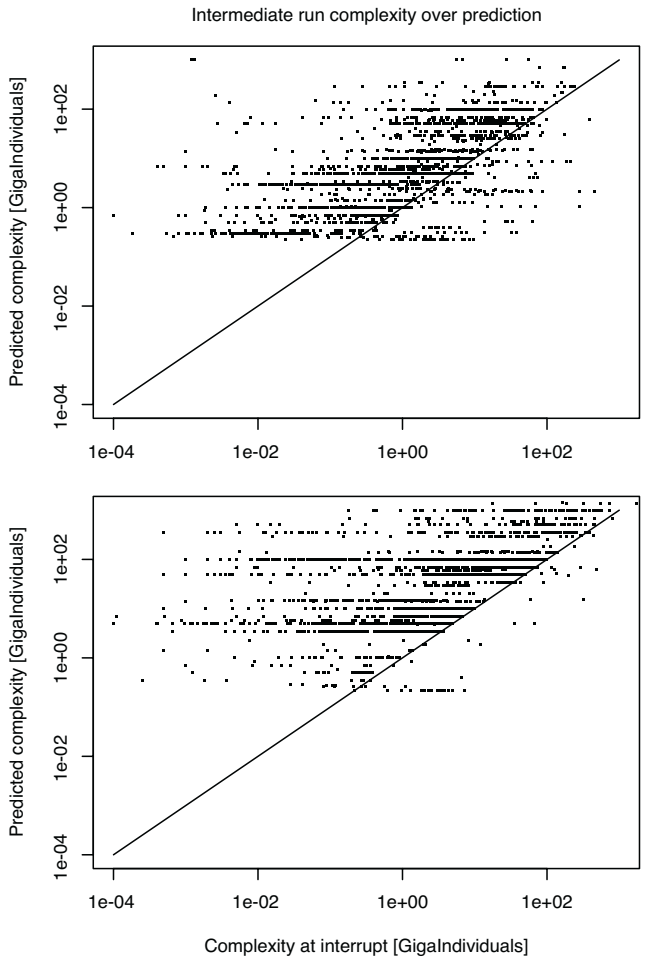

Figure 6. Hints at causes for interrupts may be deduced by comparing predicted and actual computing complexities. Of the 2613 interrupted public simulations in the upper plot, 14% might have been interrupted because actual computing time exceeded predicted computing time. Of the 4363 interrupted public simulations in the lower plot, 2.18% might have been interrupted because actual computing time exceeded predicted computing time. Results in this plot were grouped in the same way as in Figure 2, where summary statistics of the corresponding time intervals can be found.

DOI: 10.1002/spe

intermediate results might have been caused by bad predictions, where as the majority comes from real interruptions (please note that many completed runs also had bad predictions, see Figure 4). This fraction fell to about 2% in the following years, which may either be the result of improvements in the prediction algorithm or an increase of tolerance of long-term participants. Comparing the overall fraction of intermediate results in the earlier years with the later years suggests a very limited or no influence of prediction quality on the frequency with which simulations are interrupted (data from Figure 2).

Figure 7 shows no trend in the memory complexity of intermediate results, except that simulations with extraordinarily large RAM complexities such as 450 and 900 MB are more rare. The dash-dotted curves in Figures 8 and 9 indicate a similar pattern for computing time complexities, as only higher complexities show a higher fraction of interrupted simulations (Figure 8) and a higher proportion of overall computing time in intermediate results (Figure 9).

The main value of intermediate results is to (i) provide rough estimates for situations where no other results are available, (ii) provide additional insights into how some parameter values evolve during a simulation and (iii) allow improved estimates of computing times for some parameter combinations. The latter might permit significant improvements of computing time predictions for evolutionary models and their erratic behaviour on the long term, if this is supported by a corresponding global computing framework.

4.7. Participant choice shows surprisingly small bias

It was initially unclear whether the choices of participants would be so strongly biased that only a small part of the complexity spectrum would be sampled (e.g. everybody clicks on the simplest, first link of the page). This is not the case, if one considers the complexity of runs. Figure 8 shows a histogram of the number of simulations over the log of predicted computing times. Dotted curves represent non-public simulations of very small complexity for comparison. Non-histogram-like curves represent actual computing times for the same set of simulations. The choice of participants was more or less uniform for all simulations up to about 16 hours. Longer computing times were submitted less often with increasing complexity. This is easy to understand, as participants picking complex runs will not submit as many single-run results as participants picking small runs. Therefore, commitments appear to be quite uniform when total computing times invested in the corresponding categories are considered (Figure 9). Thus, a simulation scheduling system needs considerable fine-tuning capabilities for complex runs, while shorter tasks are much easier to handle. This experience is similar to what has been reported by *climateprediction.net* [16].

If the distribution of simulations per RAM complexity class is considered, the same largely uniform picture appears, except for very large simulations. There are considerable fewer simulations in the 450 and 900 MB class; a clear reflection of current PC memory sizes (Figure 10). To find out how this more or less uniform pattern is distributed in time, predicted computing times and RAM complexities were plotted over the date when simulations were started (Figures 11 and 12). This confirms overall rough uniformity, but there is enough noise so that one would only want to use this simplest of all possible participants-free-choice systems if applications are inherently robust against related fluctuations. It is also interesting to note that changes in the run-files distributed over the Web can be tracked in these plots with a delay of about three weeks; an indicator of current scheduling flexibility.

DOI: 10.1002/spe

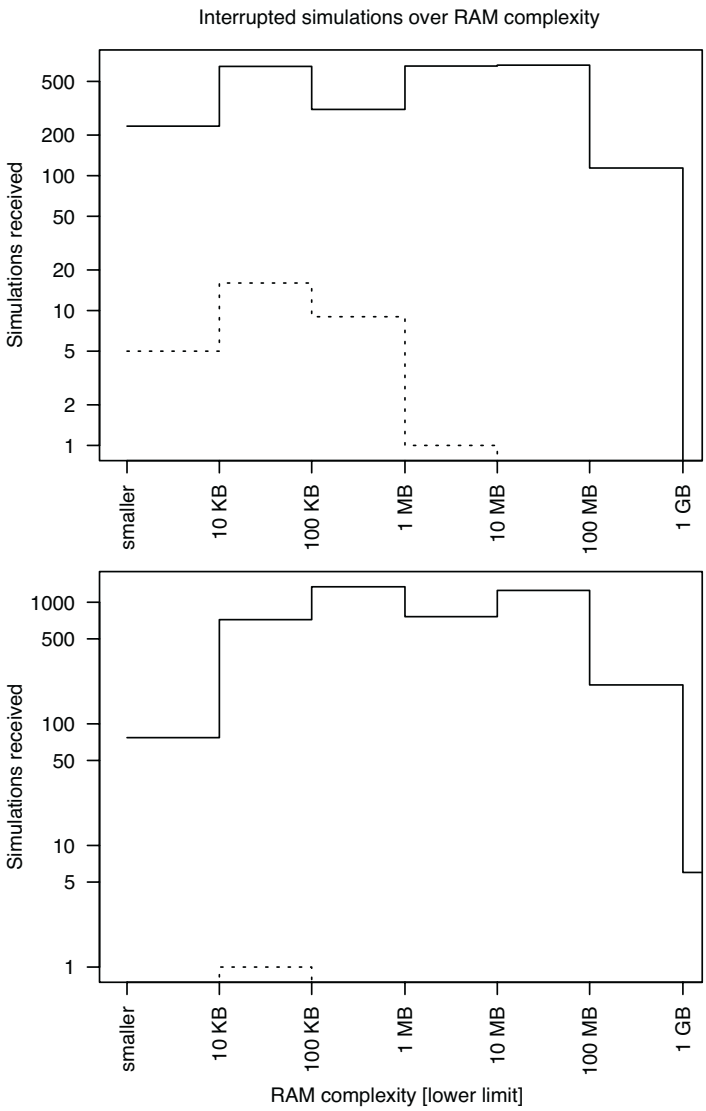

Figure 7. No significant bias could be detected in the RAM complexity of interrupted simulations. Public runs are represented by solid lines, non-public runs by dotted lines. Results in this plot were grouped in the same way as in Figure 2, where summary statistics of the corresponding time intervals can be found.

DOI: 10.1002/spe

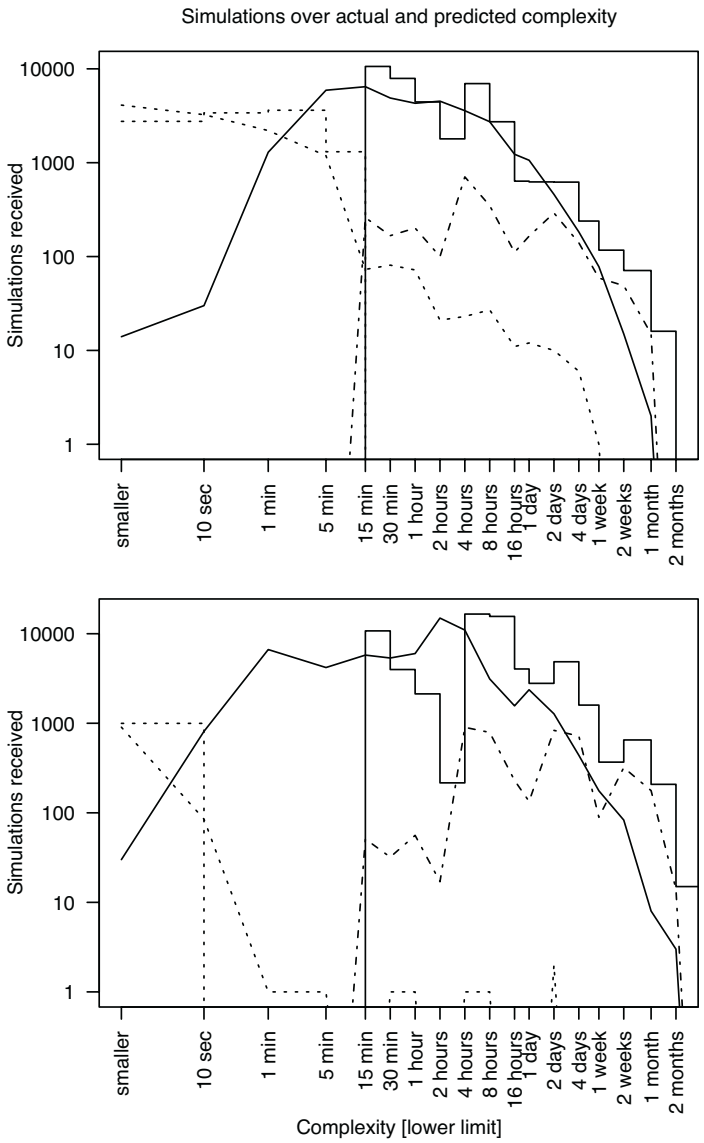

Figure 8. Simulations received per complexity class. Public runs are represented by solid lines, non-public runs by dotted lines; both include complete and intermediate results. The dash-dotted line indicates the predicted complexity of interrupted simulations that gave intermediate results. Histogram-like lines indicate participant choices. Other straight lines indicate actual complexities. Results in this plot were grouped in the same way as in Figure 2, where summary statistics of the corresponding time intervals can be found.

DOI: 10.1002/spe

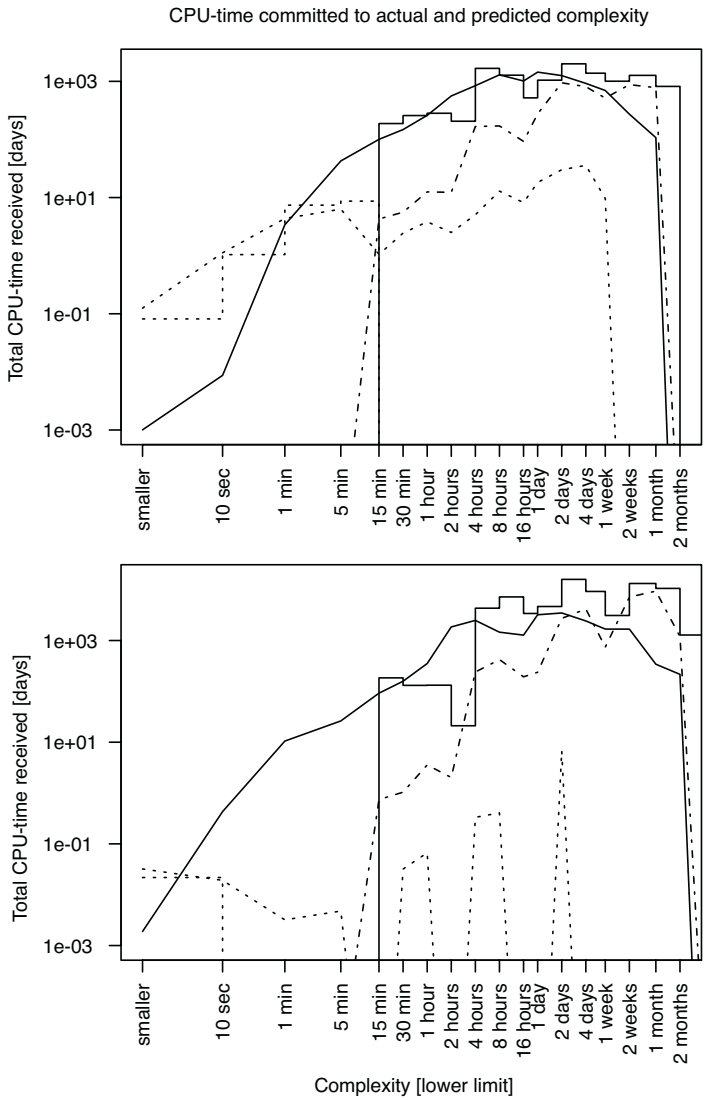

Figure 9. CPU-time committed to actual and predicted complexity. This plot shows that users invest approximately the same amount of computing time into each time-category, although they submit significantly fewer results in the more complex categories. Public runs are represented by solid lines, non-public runs by dotted lines; both include complete and intermediate results. The dash-dotted line indicates predicted complexity of interrupted simulations that gave intermediate results. Histogram-like lines indicate participant choices. Other straight lines indicate actual complexities. Results in this plot were grouped in the same way as in Figure 2, where summary statistics of the corresponding time intervals can be found.

DOI: 10.1002/spe

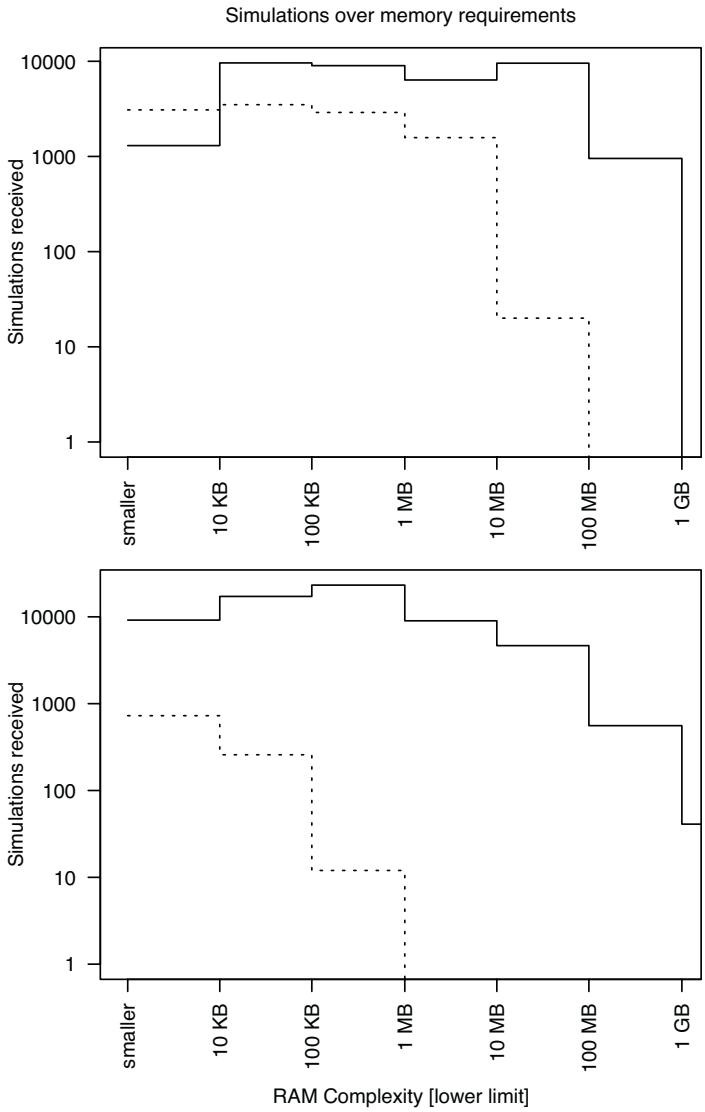

Figure 10. No significant bias in RAM complexity choices could be detected for RAM sizes that could be regarded as reasonable in private computers. Public runs are represented by solid lines, non-public runs by dotted lines. Results in this plot were grouped in the same way as in Figure 2, where summary statistics of the corresponding time intervals can be found.

DOI: 10.1002/spe

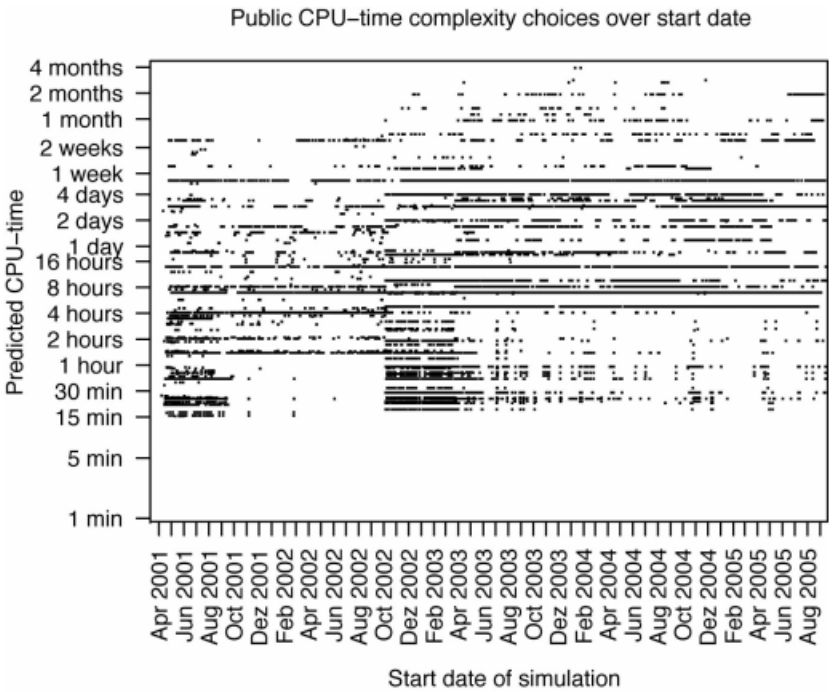

Figure 11. Essentially uniform distribution of CPU-time choices over time. The pronounced gaps below 1 and 4 hours are the result of the removal of tasks from the Web site, while no tasks had been removed from the upper complexity limit. This graph includes intermediate results.

5. LESSONS AND CONCLUSIONS

One of the goals for this paper was to describe a global computing system that is as simple as possible. The results presented here show what performance characteristics one can expect from such a system and what needs to be done to implement it. It is very difficult to come up with an actual number of months that are needed to go global with a computing problem, as this will heavily depend on the already existing expertise in computing and in describing the problem to the public. None of the steps described here are particularly difficult, however the large number of details add up to a time commitment that can no longer be regarded as negligible.

Significantly more work is needed to implement a large full-automated global computing system. While many things can be simplified if one can use an already existing framework that fits the problem very well [6,16], it may require considerable efforts to expand an existing framework to make it support the various peculiarities of many individual-based simulations. These include (i) large numbers of stochastic repeats, (ii) intermediate results for analysing trends before full results exist for tasks with unknown complexity and (iii) extremely heterogeneous work unit complexities.

DOI: 10.1002/spe

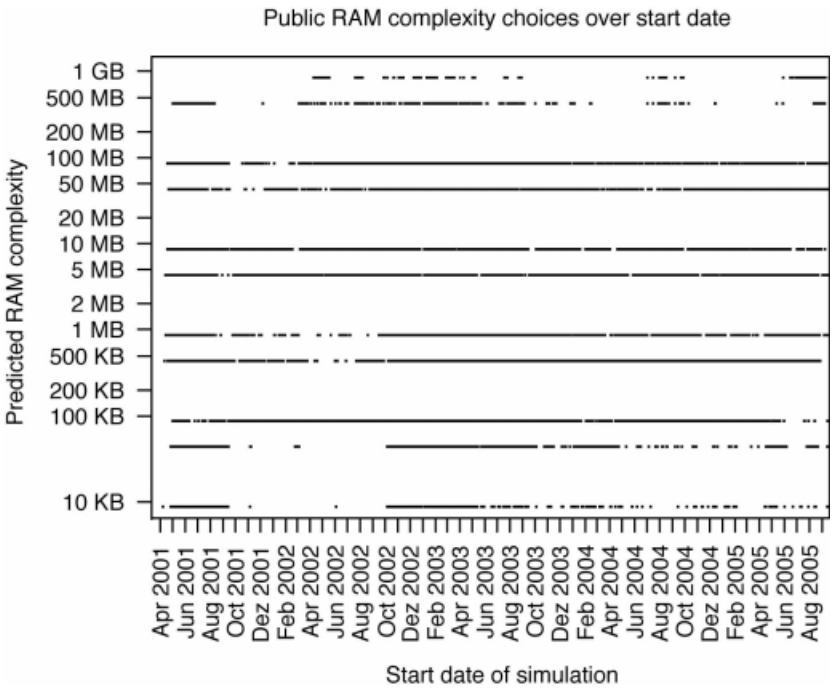

Figure 12. Patterns of RAM complexity choices. The gaps below 100 kB are the result of a lack of published tasks, while large tasks had never been removed from the Web. 900 MB results were submitted for the first time after high-scores had made evident that such results were needed but missing.

The extra work is caused by the fundamental differences in data management between deterministic computations which need only one repeat per parameter combination that is guaranteed to be correct (original BOINC design) and stochastic simulations that require many repeats to estimate variability (evolution@home). In addition, massively parallel runs of complex individual-based models can pose considerable challenges in handling huge volumes of data for scientific analysis, because access to almost all results is needed at the same time for effectively analysing the data.

Everybody will have to make up their own mind about whether global computing is worth the effort; it certainly helps to have a look at the bewildering diversity of small and large, active and past global computing projects [7] and to carefully consider the type of computational problem that shall be solved [5].

This work explored a new approach to handle the work units of extremely heterogeneous computing complexities by allowing participants to choose their preferred complexities. It was not initially clear whether participants would devote computing time to all complexity classes or rather stick with the simplest tasks, for example. The bandwidth of received results was encouraging and suggests that stochastic parameter-sweep applications might use a similar system, if they need to repeat their runs

DOI: 10.1002/spe

many times and if there is no pressing deadline for getting particular results. The positive experiences with the participants-free-choice paradigm also suggest that this might be worth implementing in dedicated, fully automated global computing frameworks. BOINC seems to take steps in that direction.

A weakness of the current minimalistic semi-automated global computing system is its dependency on the manual analysis of results. This leads to considerable slowness in updating progress and run-files on the Web. Ideally, the Web site would contain only the hottest run-files of the hour, making sure that every simulation with a particular complexity is computed once, before any simulation is computed twice. Such a scheduling mechanism is very complex as it requires constant access to an overview over all received results, but because many individual-based simulations share the same issues it appears to be worth implementing the corresponding features in global computing frameworks. This is certainly one of the next steps for evolution@home.

Another important feature should have been incorporated from the beginning. When users configure their simulator by providing upper complexity limits, they should have been given the opportunity to specify their tolerance limit for the prediction error of magnitude. This would have helped with the enormous prediction inaccuracies and will be included in future versions.

The next big challenge for evolution@home is clearly the transition to full automation. This requires the following: (i) the design of long-term results databases that include a clever scheduling mechanism for finding the next most important simulations depending on already received results; (ii) the selection of one of the many solutions to transmit data automatically over the Internet; (iii) a solution for many security issues that come with full automation; (iv) the transition of Simulator005 to a GUI-based code that can be compiled for many more platforms and gives participants a nicer computing experience; (v) the implementation of daily high-score updates; (vi) a way to allow participants to update their personal high-score profiles; and (vii) the integration of all this with a pleasant Web experience that allows sharing the burden of generating content and translations of the Web site by use of a reasonable Web content management system. The author intends to continue working on a global computing system that is secure, easier to use and well suited to solve the particular problems of stochastic individual-based simulations of evolution that were outlined in this paper. After publication of the first results [11], more genetic systems will be analysed, where Muller's ratchet has been suspected to play a role. These results will be published on the Web site as soon as their peer-review process is completed, to hopefully draw enough attention and computing power to evolution@home to allow addressing the next set of questions with the next simulator.

ACKNOWLEDGEMENTS

Special thanks go to the more than 300 participants of evolution@home for donating more than 80 years of CPU-time (top contributors were Seth A. Keel, *rechenkraft.net* and David Corbett). I wish to thank Eberhard Bertsch for helpful discussions, five anonymous referees and Frank Cappello for their comments on an earlier version of the manuscript. This work was supported by the FML Institute of Microbiology (Technische Universität München) and by a Leverhulme Trust grant to Brian Charlesworth.

REFERENCES

1. Grimm V, Railsback SF. *Individual-based Modeling and Ecology*. Princeton University Press: Princeton, NJ, 2005.
2. Huston M, Deangelis D, Post W. New computer models unify ecological theory. *BioScience* 1988; **38**:682–691.
3. Law AM, Kelton WD. *Simulation Modeling and Analysis* (3rd edn). McGraw-Hill: Boston, MA, 2000.

DOI: 10.1002/spe

4. Bart J. Acceptance criteria for using individual-based models to make management decisions. *Ecological Applications* 1995; **5**:411–420.
5. Loewe L. Global computing for bioinformatics. *Briefings in Bioinformatics* 2002; **3**:377–388.
6. Søttrup CU, Pedersen JG. Developing distributed computing solutions combining Grid computing and public computing. *Thesis*, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark, 2005; 1–102. Available at: http://www.fatbat.dk/thesis/boincThesis.pdf.
7. Pearson K. Internet-based distributed computing projects, 2006. http://distributedcomputing.info/ [18 December 2006].
8. Godfrey B. A primer on distributed computing, 2002. http://www.bacchae.co.uk/docs/dist.html [18 December 2006].
9. Buyya R. The Grid Computing Info Centre (GRID Infoware), 2006. http://www.gridcomputing.com/ [28 December 2006].
10. Buyya R. Economic-based distributed resource management and scheduling for Grid computing. *Thesis*, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia, 2002. Available at: http://www.buyya.com/thesis/.
11. Loewe L. Quantifying the genomic decay paradox due to Muller's ratchet in human mitochondrial DNA. *Genetical Research* 2006; **87**:133–159.
12. Berman F, Wolski R, Figueira S, Schopf J, Shao G. Application-level scheduling on distributed heterogeneous networks. *Proceedings of Supercomputing '96*, Pittsburgh, PA, 1996. ACM/IEEE Computer Society Press, 1996.
13. Berman F *et al*. Adaptive computing on the Grid using AppLeS. *IEEE Transactions on Parallel and Distributed Systems* 2003; **14**:369–382.
14. Smallen S, Cirne W, Frey J, Berman F, Wolski R, Su M-H, Kesselman C, Young S, Ellisman M. Combining workstations and supercomputers to support Grid applications: The parallel tomography experience. *Proceedings of the 9th Heterogenous Computing Workshop*, Cancun, Mexico, 2000. IEEE Computer Society Press: Los Alamitos, CA, 2000.
15. Casanova H, Legrand A, Zagorodnov D, Berman F. Heuristics for scheduling parameter sweep applications in Grid environments. *Proceedings of the 9th Heterogeneous Computing Workshop (HCW'2000)*, Cancun, Mexico. IEEE Computer Society Press: Los Alamitos, CA, 2000; 349–363. Available at: http://apples.ucsd.edu/pubs/hcw00_pst.ps.
16. Christensen C, Aina T, Stainforth D. The challenge of volunteer computing with lengthy climate model simulations. *Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing*, 5–8 December 2005, Melbourne, Australia, 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005. Available at: http://www.climateprediction.net/science/pubs/cpdn-computing.pdf.
17. Anderson DP. The software infrastructure of SETI@home II, 2002. http://conferences.oreillynet.com/presentations/et2002/anderson_david.ppt [18 December 2006].
18. R-Project. R: A language for data analysis and graphics, 2006. http://www.r-project.org/ [18 December 2006].
19. Loewe L. The evolution@home Web site, 2006. http://evolutionary-research.net [18 December 2006].
20. Loewe L. evolution@home: Experiences with work units that span more than 7 orders of magnitude in computational complexity. *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002)*, Berlin, Germany, 2002, Bal HE, Löhr K-P, Reinefeld A (eds.). IEEE Computer Society Press: Los Alamitos, CA, 2002; 425–431. Available at: http://archive.evolution.ws/pdf/loe/loewe-l/Loewe2002-EaHworkunits.pdf.
21. Sutter H. A fundamental turn toward concurrency in software. *Dr. Dobb's Journal* 2005; **30**:16–22. Available at: http://www.gotw.ca/publications/concurrency-ddj.htm.
22. Stephan W, Kim Y. Recent applications of diffusion theory to population genetics. *Modern Developments in Theoretical Population Genetics*, Slatkin M, Veuille M (eds.). Oxford University Press: Oxford, 2002; 72–93.
23. Gustafson JL, Snell QO. HINT: A new way to measure computer performance. *Proceedings of the 28th Hawaii International Conference on System Sciences*, Hawaii, 1995. IEEE Computer Society Press: Los Alamitos, CA, 1995. Available at: http://hint.byu.edu.
24. Gustafson JL, Snell Q, Todi R. HINT Web site, 2006. http://hint.byu.edu/ [18 December 2006].

DOI: 10.1002/spe